ADOBE® DIRECTOR® 11.5
スクリプト辞書

Adobe

© 2009 Adobe Systems Incorporated. All rights reserved.

Adobe® Director® 11.5 Scripting Dictionary

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/us/

Adobe, the Adobe logo, Director, Flash, and Shockwave, are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Microsoft and Windows are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

Bitstream is a trademark or a registered trademark of Bitstream Inc.

This product contains either BSAFE and/or TIPEM software by RSA Security, Inc.

This product includes software developed by the Apache Software Foundation (http://www.apache.org).

Adobe Flash 9 video compression and decompression is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. http://www.on2.com.

Portions of this product contain code that is licensed from Gilles Vollant.

Portions of this product contain code that is licensed from Nellymoser, Inc. (www.nellymoser.com)

Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

Copyright © 1995-2002 Opera Software ASA and its supplier. All rights reserved.

MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson. mp3 Surround audio coding technology licensed from Fraunhofer IIS, Agere Systems and Thomson. mp3PRO audio coding technologies licensed from Coding Technologies, Fraunhofer IIS and Thomson Multimedia.

PhysX is a trademark or registered trademark of NVIDIA Corporation in the United States and/or other countries.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are “Commercial Items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

第 1 章：はじめに

対象ユーザ 1
Director スクリプト記述の新機能 1
このドキュメントで新しく変更された点 2
Director でのスクリプト情報の検索 3

第 2 章：Director スクリプトに関する重要事項

スクリプトの種類 4
スクリプト用語集 5
スクリプトシンタックス 7
データの種類 10
リテラル値 12
変数 14
演算子 18
条件の構成 21
イベント、メッセージ、およびハンドラ 25
リニアリストおよびプロパティリスト 30
JavaScript シンタックス配列 37

第 3 章：Director でのスクリプトの記述

Lingo と JavaScript シンタックスの間での選択 40
ドットシンタックス形式でのスクリプト記述 41
Director オブジェクトについて 42
オブジェクトモデル図 43
トップレベルの関数とプロパティ 44
Director でのオブジェクト指向プログラミングについて 45
Lingo シンタックスでのオブジェクト指向プログラミング 46
JavaScript シンタックスでのオブジェクト指向プログラミング 54
スクリプトウィンドウでのスクリプトの記述 62

第 4 章：Director でのスクリプトのデバッグ

スクリプトのデバッグについて 75
スクリプト記述時の適切な習慣 75
基本的なデバッグ方法 76
スクリプトウィンドウでのデバッグ 78
メッセージウィンドウでのデバッグ 79
オブジェクトインスペクタでのデバッグ 83
デバッガウィンドウでのデバッグ 86

プロジェクタおよび Shockwave ムービーのデバッグ 90
高度なデバッグ方法 91

第 5 章：Director のコアオブジェクト

キャストライブラリ 92
グローバル 93
キー 94
メンバ 95
ミクサ 96
マウス 97
ムービー 98
プレーヤ 100
サウンド 102
サウンドチャネル 104
スプライト 105
スプライトチャネル 107
システム 108
ウィンドウ 109

第 6 章：メディアタイプ

アニメーション GIF 112
ビットマップ 113
ボタン 114
ByteArray 114
カラーパレット 114
カーソル 115
DVD 116
フィールド 117
フィルムループ 118
Flash コンポーネント 119
Flash ムービー 120
フォント 121
リンクされたムービー 122
MP4Media/FLV 122
QuickTime 123
RealMedia 125
Shockwave 3D 126
Shockwave オーディオ 126
サウンド 127
テキスト 128
ベクターシェイプ 129
Windows Media 130

第 7 章：スクリプトオブジェクト

オーディオフィルタ 132
ByteArray 144
Fileio 148
MUI Xtra 149
NetLingo 150
SpeechXtra 151
XML Parser 151
XML Xtra 152

第 8 章：3D オブジェクト

3D オブジェクトについて 154
カメラ 154
グループ 156
ライト 157
メンバ 157
モデル 159
モデルリソース 160
モーション 160
レンダラサービス 161
シェーダ 161
スプライト 162
テクスチャ 163

第 9 章：定数

" (ストリング) 164
BACKSPACE 164
EMPTY 165
ENTER 165
FALSE 166
PI 167
QUOTE 167
RETURN (定数) 168
SPACE 168
TAB 169
TRUE 170
VOID 170

第 10 章：イベントとメッセージ

on activateApplication 171
on activateWindow 172
on beginSprite 172
on closeWindow 173

on cuePassed 174
on deactivateApplication 175
on deactivateWindow 176
on DVDeventNotification 176
on endSprite 179
on enterFrame 180
on EvalScript 181
on exitFrame 183
on getBehaviorDescription 184
on getBehaviorTooltip 185
on getPropertyDescriptionList 185
on hyperlinkClicked 186
on idle 187
on isOKToAttach 188
on keyDown 189
on keyUp 191
on mouseDown (イベントハンドラ) 192
on mouseEnter 193
on mouseLeave 194
on mouseUp (イベントハンドラ) 195
on mouseUpOutside 196
on mouseWithin 197
on moveWindow 198
on openWindow 199
on prepareFrame 199
on prepareMovie 200
on resizeWindow 201
on rightMouseDown (イベントハンドラ) 202
on rightMouseUp (イベントハンドラ) 202
on runPropertyDialog 203
on savedLocal 204
on sendXML 205
on startMovie 206
on stepFrame 207
on stopMovie 208
on streamStatus 209
on timeOut 210
trayIconMouseDoubleClick 211
trayIconMouseDown 211
trayIconRightMouseDown 212
on zoomWindow 213

第 11 章：キーワード

\（継続シンボル） 214
case 214
char...of 215
end 216
end case 217
exit 217
exit repeat 218
field 218
global 219
if 220
INF 221
item...of 221
line...of 222
loop （キーワード） 223
me 223
menu 224
NAN 225
next 226
next repeat 226
on 227
otherwise 227
property 228
put...after 229
put...before 230
put...into 230
repeat while 231
repeat with 232
repeat with...down to 233
repeat with...in list 234
return （キーワード） 234
set...to, set...= 235
sprite...intersects 236
sprite...within 237
version 237
word...of 238

第 12 章：メソッド

_system.gc() 239
abort 239
abs() 240
activateAtLoc() 241

activateButton() 241
add 242
add (3D texture) 243
addAt 243
addBackdrop 244
addCamera 245
addChild 245
addModifier 246
addOverlay 247
addProp 248
addToWorld 249
addVertex() 249
alert() 250
Alert() 251
append 252
applyFilter() 253
appMinimize() 253
atan() 254
beep() 255
beginRecording() 256
bitAnd() 257
bitNot() 258
bitOr() 259
bitXor() 260
breakLoop() 260
breakLoop （サウンドオブジェクト） 261
browserName() 261
ByteArray 262
ByteArray(str) 262
build() 263
cacheDocVerify() 264
cacheSize() 265
call 266
callAncestor 268
callFrame() 269
camera() 270
cameraCount() 270
cancelIdleLoad() 271
castLib() 272
channel() （トップレベル） 272
channel() （サウンド） 273
chapterCount() 273

charPosToLoc() 274
chars() 275
charToNum() 275
clearAsObjects() 276
clearCache 277
clearError() 278
clearFrame() 279
clearGlobals() 280
clone 281
cloneDeep 281
cloneModelFromCastmember 282
cloneMotionFromCastmember 283
close() 283
closeFile() 284
closeXlib 284
color() 285
compress() 286
constrainH() 286
constrainV() 287
copyPixels() 288
copyToClipBoard() 290
cos() 291
count() 291
createFile() 292
createMask() 293
createMatte() 293
createSoundObject 294
crop() （画像） 295
crop() （ビットマップ） 296
cross 296
crossProduct() 297
cursor() 297
date() （フォーマット） 301
date() （システム） 303
delay() 303
delete() 304
delete() (FileIO) 305
deleteFile() 305
deleteAt 306
deleteCamera 306
deleteFrame() 307
deleteGroup 308

deleteLight 308
deleteModel 309
deleteModelResource 309
deleteMotion 310
deleteOne 310
deleteProp 311
deleteShader 312
deleteSoundObject 312
deleteTexture 313
deleteVertex() 314
displayOpen() 314
displaySave() 315
do 315
doneParsing() 316
dot() 316
dotProduct() 317
downloadNetThing 317
draw() 318
duplicate()（画像） 319
duplicate()（リスト関数） 320
duplicate()（メンバ） 321
duplicateFrame() 321
enableHotSpot() 322
enableSoundTrack(trackNum) 323
endRecording() 323
erase() 324
error() 325
externalEvent() 326
extrude3D 327
externalParamName() 328
externalParamValue() 329
extractAlpha() 331
fadeIn() 331
fadeOut() 332
fadeTo() 333
fileName() 333
fileOpen() 334
fileSave() 335
fill() 336
filter() 337
findLabel() 338
findEmpty() 338

findPos 339
findPosNear 339
finishIdleLoad() 340
flashToStage() 341
float() 341
floatP() 342
flushInputEvents() 343
forget() （ウィンドウ） 344
forget() (Timeout) 344
framesToHMS() 345
frameReady() (Movie) 346
frameStep() 347
freeBlock() 347
freeBytes() 348
generateNormals() 348
getaProp 349
getAt() 350
getCharSet 351
getError() (Flash, SWA) 352
getError() (XML) 354
getErrorString() 354
getFinderInfo() 355
getFlashProperty() 356
getFrameLabel() 357
getHardwareInfo() 357
getHotSpotRect() 358
getInstalledCharSets 359
GetItemPropList 359
getLast() 360
getLatestNetID 360
getLength() 361
getNetByteArray 362
getNetText() 362
getNormalized 364
getNthFileNameInFolder() 364
getOne() 365
getOSDirectory() 366
getPixel() 367
getPixels() 368
getPlayList() 369
getPos() 370
getPosition() 371

getPref() 372
getPref()（プレーヤ） 372
getProp() 373
getPropAt() 374
getPropRef()（JavaScript のみ） 374
getRendererServices() 375
getSoundObject() 376
getSoundObjectList 377
getStreamStatus() 377
getSystemCharSet 378
getURL() 378
getVal() 379
getVariable() 380
GetWidgetList() 381
GetWindowPropList 381
getWorldTransform() 382
go() 383
goLoop() 385
goNext() 385
goPrevious() 386
goToFrame() 387
gotoNetMovie 387
gotoNetPage 388
group() 389
halt() 389
handler() 390
handlers() 390
hilite （コマンド） 391
hitTest() 392
HMStoFrames() 393
hold() 394
importByteArrayInto() 394
identity() 395
idleLoadDone() 396
ignoreWhiteSpace() 396
ilk() 397
ilk (3D) 399
image() 400
importFileInto() 401
Initialize 403
insertBackdrop 403
insertFrame() 404

insertOverlay 405
inside() 406
installMenu 406
integer() 407
integerP() 408
Interface() 408
interpolate() 409
interpolateTo() 409
intersect() 410
inverse() 411
invert() 411
isBusy() 412
isCharSetInstalled 413
isInWorld() 413
isPastCuePoint() 414
ItemUpdate() 415
keyPressed() 416
label() 417
last() 418
lastClick() 419
lastEvent() 419
length() 420
light() 420
lineHeight() 421
linePosToLocV() 421
linkAs() 422
list() 422
listP() 423
loadFile() 424
loadPolicyFile() 424
locToCharPos() 427
locVToLinePos() 427
log() 428
makeList() 428
makeScriptedSprite() 429
makeSubList() 430
map() 430
map (3D) 431
mapMemberToStage() 432
mapStageToMember() 432
marker() 433
matrixAddition() 434

matrixMultiply() 434
matrixMultiplyScalar() 435
matrixTranspose() 435
max() 436
maximize() 437
mci 437
member() 438
mergeDisplayTemplate() 439
mergeProps() 440
mesh （プロパティ） 440
meshDeform （モディファイア） 441
min 442
minimize() 443
model (3D) 443
modelResource 444
modelsUnderLoc 445
modelsUnderRay 446
modelUnderLoc 448
motion() 448
move() 449
moveTo 450
moveToBack() 450
moveToFront() 451
moveVertex() 452
moveVertexHandle() 452
multiply() 453
mute （ミクサ） 453
mute （サウンドオブジェクト） 454
neighbor 454
netAbort 455
netByteArrayResult 456
netDone() 456
netError() 457
netLastModDate() 459
netMIME() 460
netStatus 461
netTextResult() 462
new() 463
newCamera 466
newColorRatio 466
newCurve() 467
newGroup 468

newLight 468
newMatrix() 469
newMember() 469
newMesh 470
newModel 471
newModelResource 472
newMotion() 473
newObject() 473
newShader 474
newTexture 475
normalize 476
nothing 477
nudge() 478
numColumns() 479
numRows() 480
numToChar() 480
objectP() 481
offset() (string function) 482
offset() （矩形関数） 483
open() （プレーヤ） 484
open() （ウィンドウ） 485
openFile() 486
openXlib 486
param() 487
paramCount() 488
parseByteArray 488
parseString() 489
parseString (XML Xtra) 490
parseURL() 490
parseURL (XML Xtra) 492
pass 493
pasteClipBoardInto() 494
pause() (DVD) 494
pause() （ミクサ） 495
pause (MP4Media/FLV) 496
pause() (3D) 496
pause() (RealMedia, SWA, Windows Media) 497
pause() （サウンドチャネル） 497
pause （サウンドオブジェクト） 498
perlinNoise() 498
perpendicularTo 500
pictureP() 500

play() (3D) 501
play() (DVD) 502
play() （ミクサ） 504
play() (MP4Media/FLV) 504
play() (RealMedia, SWA, Windows Media) 505
play() （サウンドチャネル） 506
play （サウンドオブジェクト） 507
playFile() 507
playNext() （サウンドチャネル） 508
playNext() (3D) 509
playerParentalLevel() 509
point() 510
pointAt 511
pointInHyperlink() 512
pointToChar() 513
pointToItem() 513
pointToLine() 514
pointToParagraph() 515
pointToWord() 516
postNetByteArray 517
postNetText 518
power() 519
preLoad() （メンバ） 520
preLoad() (Movie) 521
preLoadBuffer() 522
preLoadMember() 522
preLoadMovie() 523
preloadNetThing() 524
preMultiply 525
preRotate 525
preScale() 527
preTranslate() 528
print() 529
printAsBitmap() 529
printFrom() 530
propList() 531
proxyServer 532
ptToHotSpotID() 532
puppetPalette() 533
puppetSprite() 534
puppetTempo() 535
puppetTransition() 536

put() 538
qtRegisterAccessKey() 539
qtUnRegisterAccessKey() 539
queue() 539
queue() (3D) 541
QuickTimeVersion() 542
quit() 542
ramNeeded() 543
random() 544
randomVector() 545
randomVector 546
rawNew() 546
readBoolean 547
readByteArray 547
readByteArray (FileIO Xtra) 548
readChar() 548
readFile() 549
readFloat32 550
readFloat64 550
readInt8 551
readInt16 551
readInt32 551
readLine() 552
readRawString 552
readString 553
readToken() 553
readWord() 554
realPlayerNativeAudio() 555
realPlayerPromptToInstall() 556
realPlayerVersion() 557
recordFont 558
rect() 559
registerByteArrayCallback 560
registerCuePointCallback 563
registerEndOfSpoolCallback() 564
registerForEvent() 565
registerScript() 566
removeBackdrop 568
removeFromWorld 568
removeLast() 569
removeModifier 569
removeOverlay 570

removeScriptedSprite() 570
replaceMember 571
reset （ミクサ） 572
resetWorld 572
resolveA 573
resolveB 573
restart() 574
restore() 574
result 575
resume() 576
returnToTitle() 576
revertToWorldDefaults 577
rewind() (MP4Media/FLV) 577
rewind() （サウンドチャネル） 578
rewind() (Windows Media) 578
rewind() （アニメーション GIF、Flash） 579
rollOver() 579
rootMenu() 581
rotate 581
run 582
runMode 583
save （ミクサ） 584
Save （サウンドオブジェクト） 584
save castLib 585
saveMovie() 586
scale （コマンド） 586
script() 587
scrollByLine() 588
scrollByPage() 589
seek() 590
seek(mSec) (MP4Media/FLV) 590
seek（サウンドオブジェクト） 591
selectAtLoc() 591
selectButton() 592
selectButtonRelative() 593
selection() （関数） 593
sendAllSprites() 594
sendEvent 595
sendSprite() 595
setAlpha() 596
setaProp 597
setAt 598

setCallback() 599
setCharSet 600
setCollisionCallback() 601
setFilterMask() 601
setFinderInfo() 602
setFlashProperty() 602
setNewLineConversion() 603
setPixel() 604
setPixels() 605
setPlayList() 606
setPosition() 607
setPref() 608
setPref()（プレーヤ） 609
setProp 610
setScriptList() 611
settingsPanel() 611
setTrackEnabled() 612
setVal() 613
setVariable() 614
showLocals() 614
showProps() 615
showGlobals() 616
shutDown() 616
sin() 617
sort 617
sound() 618
sprite() 618
spriteSpaceToWorldSpace 619
sqrt() 620
stageBottom 620
stageLeft 621
stageRight 621
stageToFlash() 622
stageTop 623
startSave（ミクサ） 624
startSave（サウンドオブジェクト） 624
status() 625
stop() (DVD) 625
stop() (Flash) 626
stop()（ミクサ） 627
stop() (MP4Media/FLV) 627
stop() (RealMedia, SWA, Windows Media) 628

stop() （サウンドチャネル） 628
stop （サウンドオブジェクト） 629
stop 629
stopEvent() 630
stopSave （ミクサ） 631
stopSave （サウンドオブジェクト） 631
stream() 632
string() 633
stringP() 634
subPictureType() 634
substituteFont 635
swing() 636
symbol() 637
symbolP() 637
tan() 638
tellStreamStatus() 638
tellTarget() 639
time() （システム） 640
timeout() 641
titleMenu() 642
toHexString 642
top (3D) 643
topCap 643
topRadius 644
trace() 644
transform （コマンド） 645
translate 646
uncompress() 647
union() 647
unLoad() （メンバ） 648
unLoad() （ムービー） 649
unLoadMember() 649
unLoadMovie() 650
unmute （ミクサ） 651
unmute （サウンドオブジェクト） 652
unregisterAllEvents 652
unregisterByteArrayCallback 653
unregisterCuePointCallback 653
unregisterEndOfSpoolCallback() 654
update 654
updateFrame() 655
updateStage() 656

URLEncode 657
value() 658
vector() 659
version() 660
voiceCount() 660
voiceGet() 661
voiceGetAll() 661
voiceGetPitch() 662
voiceGetRate() 663
voiceGetVolume() 664
voiceInitialize() 664
voicePause() 665
voiceResume() 666
voiceSet() 666
voiceSetPitch() 667
voiceSetRate() 667
voiceSetVolume() 668
voiceSpeak() 668
voiceState() 669
voiceStop() 670
voiceWordPos() 670
voidP() 671
window() 672
WindowOperation 672
windowPresent() 673
worldSpaceToSpriteSpace 674
writeBoolean 674
writeByteArray 675
writeByteArray (FileIO Xtra) 675
writeChar() 676
writeFloat32 676
writeFloat64 677
writeInt8 677
writeInt16 678
writeInt32 678
writeRawString 679
writeReturn() 679
writeString() 680
writeString (byte array) 681
xtra() 681
zoomBox 682

第 13 章：演算子

(シンボル) 683
. （ドット演算子）（どっとえんざんし） 684
- (マイナス記号) 685
-- (comment) 686
&、+（連結演算子） 686
&&、+（連結演算子） 687
() (括弧かっこ) 688
*（乗算） 689
+ (加算) 690
+（加算）（3D） 690
-（マイナス）（3D） 690
*（乗算演算子（3D）） 691
/ (除算) 691
/（除算）（3D） 692
< (より小さい) 692
<= (以下) 693
<> (等しくない) 693
= (等号) 693
> (より大きい) 694
>= (以上) 694
[ ] (ブラケットアクセス) 694
[ ] (list) 695
@ (パス名演算子) 697
and 698
contains 699
mod 699
not 701
or 702
ランダムアクセス演算子らんだむあくせすえんざんし 703
starts 703
String 704

第 14 章：プロパティ

_global 705
_key 705
_mouse 706
_movie 707
_player 707
_sound 708
_system 709
aboutInfo 709

actionsEnabled 710
active3dRenderer 710
activeCastLib 711
activeWindow 712
actorList 713
alertHook 713
alignment 715
allowCustomCaching 715
allowGraphicMenu 716
allowSaveLocal 716
allowTransportControl 716
allowVolumeControl 717
allowZooming 717
alphaThreshold 718
ambient 718
ambientColor 719
ancestor 719
angle (3D) 720
angle (DVD) 721
angleCount 721
animationEnabled 722
antiAlias 722
antiAliasingEnabled 723
antiAliasingSupported 723
antiAliasThreshold 724
antiAliasType 724
appearanceOptions 725
applicationName 727
applicationPath 727
aspectRatio 728
attenuation 728
attributeName 729
attributeValue 729
audio (DVD) 730
audio (MP4Media/FLV) 730
audio (RealMedia) 731
audio (Windows Media) 732
audioChannelCount 732
audioExtension 733
audioFormat 733
audioSampleRate 734
audioStream 734

audioStreamCount 735
auto 735
autoblend 736
autoCameraPosition 736
autoMask 737
autoTab 737
axisAngle 738
back 739
backColor 739
backdrop 740
backgroundColor 741
beepOn 742
bevelDepth 742
bevelType 743
bgColor（ウィンドウ） 744
bgColor（スプライト、3D メンバ） 744
bias 745
bitDepth（ミクサ） 745
bitDepth（サウンドオブジェクト） 746
bitmapSizes 746
bitRate 747
bitsPerSample 747
blend (3D) 748
blend（スプライト） 749
blendConstant 749
blendConstantList 750
blendFactor 751
blendFunction 751
blendFunctionList 752
blendLevel 753
blendRange 753
blendSource 754
blendSourceList 754
blendTime 755
bone 756
bonesPlayer（モディファイア） 756
border 758
bottom 758
bottom (3D) 758
bottomCap 759
bottomRadius 759
bottomSpacing 760

boundary 760
boundingSphere 761
boxDropShadow 762
boxType 762
brightness 762
broadcastProps 763
bufferSize 764
bufferSize（ミクサ） 764
buttonCount 765
buttonsEnabled 765
buttonStyle 766
buttonType 767
byteArray 767
BytesRemaining 767
bytesStreamed 768
bytesStreamed (3D) 769
camera 769
cameraPosition 770
cameraRotation 770
castLib 771
castLibNum 771
castMemberList 772
center 773
centerRegPoint 773
centerStage 774
changeArea 775
channel 775
channelCount（ミクサ） 776
channelCount（サウンドチャネル） 777
channelCount（サウンドオブジェクト） 777
chapter 778
chapterCount 778
characterSet 779
charSpacing 779
checkMark 780
child (3D) 781
child (XML) 781
chunkSize 782
clearAtRender 782
clearValue 783
clickLoc 783
clickMode 784

clickOn 785
closed 786
closedCaptions 786
collision (modifier) 787
collisionData 788
collisionNormal 789
color() 790
color (fog) 792
color (light) 792
colorBufferDepth 792
colorDepth 793
colorList 794
colorRange 795
colors 795
colorSteps 796
commandDown 797
comments 798
compressed 798
connectionStatus (MP4Media/FLV) 799
connectionStatus（サウンドオブジェクト） 799
constraint 800
controlDown 801
controller 802
copyrightInfo（ムービー） 803
copyrightInfo（SWA） 803
count 804
count (3D) 804
count (castLib) 805
cpuHogTicks 806
creaseAngle 806
creases 807
creationDate 807
crop 808
cuePointNames 809
cuePointTimes 809
currentLoopState 810
currentSpriteNum 810
currentTime (3D) 811
currentTime (DVD) 812
currentTime (QuickTime, AVI) 813
currentTime (MP4Media/FLV) 813
currentTime (RealMedia) 814

currentTime（サウンドオブジェクト） 815
currentTime （スプライト） 815
cursor 816
cursorSize 819
curve 820
debug 821
debugPlaybackEnabled 821
decayMode 822
defaultRect 822
defaultRectMode 823
density 824
depth (3D) 825
depth （ビットマップ） 826
depthBufferDepth 826
deskTopRectList 827
diffuse 828
diffuseColor 828
diffuseLightMap 829
digitalVideoTimeScale 830
digitalVideoType 830
direction 831
directionalColor 831
directionalPreset 832
directToStage 833
directToStage (MP4Media/FLV) 834
disableImagingTransformation 834
displayFace 835
displayMode 835
displayRealLogo 836
displayTemplate 837
distribution 838
dither 839
dockingEnabled 839
domain 840
doubleClick 840
drag 841
drawRect 842
dropShadow 843
duration (3D) 843
duration (DVD) 844
duration （メンバ） 844
duration (MP4Media/FLV) 845

duration (RealMedia, SWA) 845
editable 846
editShortCutsEnabled 847
elapsedTime 847
elapsedTime（ミクサ） 848
elapsedTime （サウンドオブジェクト） 849
emissive 849
emitter 850
emulateMultibuttonMouse 850
enabled 851
enabled (collision) 852
enabled（フィルタ） 852
enabled (fog) 853
enabled (sds) 853
enableFlashLingo 854
endAngle 854
endColor 855
endFrame 856
endian 856
endTime （サウンドチャネル） 857
endTime （サウンドオブジェクト） 857
environmentPropList 858
error 859
eventPassMode 860
exitLock 861
externalParamCount 862
face 863
face[ ] 864
far (fog) 864
fieldOfView 865
fieldOfView (3D) 865
fileFreeSize 866
fileName （キャスト） 866
fileName （メンバ） 867
fileName (MP4Media/FLV) 868
fileName （ウィンドウ） 868
fileSize 869
fileVersion 870
fillColor 870
fillCycles 871
fillDirection 872
filled 872

fillMode 873
fillOffset 874
fillScale 874
filterlist 875
filterList（ミクサ） 875
filterList（サウンドオブジェクト） 876
firstIndent 876
fixedLineSpace 877
fixedRate 877
fixStageSize 878
flashRect 879
flat 880
flipH 880
flipV 881
floatPrecision 881
fog 882
folder 883
font 884
fontSize 885
fontStyle 885
foreColor 887
frame 887
frameCount 888
frameLabel 888
framePalette 889
frameRate 890
frameRate (DVD) 891
frameRate (MP4Media/FLV) 892
frameScript 892
frameSound1 893
frameSound2 893
frameTempo 894
frameTransition 894
front 895
frontWindow 895
fullScreen 896
getBoneID 896
globals 897
glossMap 897
gravity 898
gradientType 899
group 899

height 900
height (3D) 901
height (MP4Media/FLV) 901
heightVertices 902
highlightPercentage 902
highlightStrength 903
hilite 903
hinting 904
hither 905
hotSpot 905
hotSpotEnterCallback 906
hotSpotExitCallback 906
HTML 907
hyperlink 908
hyperlinkRange 908
hyperlinks 909
hyperlinkState 909
idleHandlerPeriod 910
idleLoadMode 911
idleLoadPeriod 912
idleLoadTag 912
idleReadChunkSize 913
image （画像） 913
image (RealMedia) 914
image (MP4Media/FLV) 915
image （ウィンドウ） 915
imageCompression 916
imageEnabled 917
imageQuality 918
immovable 919
ink 919
inker （モディファイア） 920
inlineImeEnabled 921
interval 922
invertMask 922
isPlayable (MP4Media/FLV) 923
isSaving （ミクサ） 923
isSaving （サウンドオブジェクト） 924
isVRMovie 924
itemDelimiter 925
kerning 926
kerningThreshold 927

key 927
keyboardFocusSprite 928
keyCode 929
keyDownScript 930
keyframePlayer（モディファイア） 931
keyUpScript 932
labelList 933
lastChannel 933
lastClick 934
lastError 934
lastEvent 935
lastFrame 935
lastKey 936
lastRoll 936
left 937
left (3D) 938
leftIndent 938
length (byte array) 939
length (3D) 939
lengthVertices 939
level 940
lifetime 940
light 941
lineColor 941
lineCount 942
lineDirection 942
lineHeight 943
lineOffset 943
lineSize 944
linked 944
loaded 945
loc (backdrop and overlay) 946
locH 946
lockTranslation 947
locV 947
locZ 948
lod（モディファイア） 949
loop (3D) 950
loop (emitter) 950
loop（メンバ） 951
loop (MP4Media/FLV) 951
loop (Flash) 952

loop (Windows Media) 952
loopBounds 953
loopCount 954
loopCount （サウンドオブジェクト） 955
loopEndTime （サウンドチャネル） 955
loopEndTime （サウンドオブジェクト） 956
loopsRemaining 956
loopsRemaining （サウンドオブジェクト） 957
loopStartTime 957
loopStartTime （サウンドオブジェクト） 958
magnitude 959
margin 959
markerList 960
mask 960
maxInteger 961
maxSpeed 962
media 962
mediaReady 963
mediaStatus (DVD) 963
mediaStatus (MP4Media/FLV) 964
mediaStatus (RealMedia, Windows Media) 965
mediaXtraList 966
member 966
member （キャスト） 967
member （ムービー） 967
member （サウンドチャネル） 968
member （サウンドオブジェクト） 968
member （スプライト） 969
memorySize 970
meshDeform （モディファイア） 971
milliseconds 972
minSpeed 972
missingFonts 973
mixer 973
mode (emitter) 974
mode (collision) 974
model 975
modelA 976
modelB 976
modelResource 977
modified 978
modifiedBy 978

modifiedDate 979
modifier 979
modifier[] 980
modifiers 980
mostRecentCuePoint 981
mostRecentCuePoint （サウンドオブジェクト） 982
motion 982
motionQuality 983
mouseChar 983
mouseDown 984
mouseDownScript 985
mouseH 986
mouseItem 987
mouseLevel 987
mouseLine 988
mouseLoc 989
mouseMember 990
mouseOverButton 990
mouseUp 991
mouseUpScript 992
mouseV 993
mouseWord 994
moveableSprite 995
movie 995
multiSound 996
name 996
name (3D) 997
name （メニュープロパティ） 997
name （メニュー項目プロパティ） 998
name （ミクサ） 998
name （サウンドオブジェクト） 999
name （スプライト） 999
name （スプライトチャネル） 1000
name (timeout) 1001
name (XML) 1001
near (fog) 1002
nearFiltering 1002
netPresent 1003
netThrottleTicks 1004
node 1004
nodeEnterCallback 1005
nodeExitCallback 1005

nodeType 1006
normalList 1006
normals 1007
number（キャスト） 1008
number（文字） 1008
number（項目） 1009
number（行） 1009
number（メンバ） 1010
number（メニュー） 1011
number（メニュー項目） 1011
number（スプライトチャネル） 1012
number（システム） 1012
number（単語） 1013
number of members 1013
number of xtras 1014
numBuffersToPreload 1014
numChannels 1015
numParticles 1015
numSegments 1016
obeyScoreRotation 1016
optionDown 1017
organizationName 1018
originalFont 1018
originH 1019
originMode 1020
originPoint 1021
originV 1022
orthoHeight 1022
overlay 1023
pageHeight 1024
palette 1024
paletteMapping 1025
paletteRef 1025
pan 1026
pan（QTVR プロパティ） 1027
panMatrix、toChannels、および useMatrix（ミクサ） 1027
panMatrix、toChannels、および useMatrix（サウンドオブジェクト） 1028
paragraph 1030
parent 1030
password 1031
path（ムービー） 1032
path (3D) 1033

pathName （Flash メンバ） 1033
pathStrength 1034
pattern 1034
pausedAtStart （Flash、デジタルビデオ） 1035
pausedAtStart (MP4Media/FLV) 1035
pausedAtStart (RealMedia, Windows Media) 1036
percentBuffered 1037
percentPlayed 1038
percentStreamed (3D) 1039
percentStreamed （メンバ） 1039
percentStreamed (MP4Media/FLV) 1040
percentStreamed（サウンドオブジェクト） 1041
period 1041
persistent 1042
picture （メンバ） 1042
picture （ウィンドウ） 1043
platform 1043
playBackMode 1044
playing 1045
playing (3D) 1045
playlist 1046
playRate (3D) 1047
playRate (DVD) 1047
playRate（QuickTime、AVI、MP4、FLV） 1047
playRate（サウンドオブジェクト） 1048
playRate (Windows Media) 1049
pointAtOrientation 1049
pointOfContact 1050
position (byte array) 1050
position （トランスフォーム） 1051
positionReset 1051
posterFrame 1052
preferred3dRenderer 1052
preLoad (3D) 1054
preLoad （メンバ） 1054
preLoadEventAbort 1055
preLoadMode 1055
preLoadRAM 1056
preLoadTime 1057
primitives 1057
productName 1058
productVersion 1058

projection 1059
purgePriority 1059
quad 1060
quality 1061
quality (3D) 1062
radius 1063
randomSeed 1063
recordFont 1064
rect (camera) 1064
rect （画像） 1065
rect （メンバ） 1066
rect （スプライト） 1067
rect （ウィンドウ） 1067
ref 1068
reflectionMap 1068
reflectivity 1069
region 1070
regPoint 1070
regPoint (3D) 1071
regPointVertex 1071
renderer 1072
rendererDeviceList 1073
renderFormat 1073
renderStyle 1074
resizable 1075
resolution (3D) 1075
resolution (DVD) 1076
resolve 1076
resource 1077
right 1077
right (3D) 1078
rightIndent 1078
rightMouseDown 1078
rightMouseUp 1079
romanLingo 1080
rootLock 1080
rootNode 1081
rotation 1081
rotation (backdrop and overlay) 1083
rotation (engraver shader) 1083
rotation （トランスフォーム） 1083
rotationReset 1084

RTF 1085
safePlayer 1085
sampleCount （サウンドチャネル） 1086
sampleCount （サウンドオブジェクト） 1087
sampleRate (ミクサ) 1087
sampleRate （サウンドチャネル） 1088
sampleRate （サウンドオブジェクト） 1089
sampleSize 1089
savew3d 1090
saveWorld 1090
scale (3D) 1091
scale (backdrop and overlay) 1091
scale （メンバ） 1092
scale （トランスフォーム） 1093
scaleMode 1093
score 1094
scoreColor 1095
scoreSelection 1095
script 1096
scripted 1097
scriptingXtraList 1098
scriptInstanceList 1098
scriptList 1099
scriptNum 1099
scriptsEnabled 1100
scriptSyntax 1100
scriptText 1101
scriptType 1101
scrollTop 1102
sds （モディファイア） 1103
searchCurrentFolder 1104
searchPathList 1104
selectedButton 1105
selectedText 1106
selection 1106
selection （テキストキャストメンバプロパティ） 1107
selEnd 1107
selStart 1108
serialNumber 1109
shader 1109
shaderList 1110
shadowPercentage 1111

shadowStrength 1112
shapeType 1112
shiftDown 1113
shininess 1113
silhouettes 1114
size 1114
sizeRange 1115
sizeState 1115
skew 1116
smoothness 1117
sound （メンバ） 1118
sound （プレーヤ） 1118
soundChannel (SWA) 1119
soundChannel (RealMedia) 1119
soundDevice 1120
soundDeviceList 1121
soundEnabled 1122
soundKeepDevice 1122
soundLevel 1123
soundMixMedia 1124
soundObjectList 1124
source 1125
sourceFileName 1125
sourceRect 1126
specular (light) 1126
specular (shader) 1127
specularColor 1127
specularLightMap 1128
spotAngle 1129
spotDecay 1129
sprite （ムービー） 1130
sprite （スプライトチャネル） 1130
spriteNum 1131
stage 1132
startAngle 1132
startFrame 1133
startTime (サウンドチャネル) 1133
startTime （サウンドオブジェクト） 1134
startTimeList 1135
state (3D) 1136
state (Flash, SWA) 1136
state (RealMedia) 1138

static 1139
staticQuality 1140
status 1141
status（ミクサ） 1141
status（サウンドオブジェクト） 1142
stillDown 1143
stopTime 1143
stopTimeList 1144
streamMode 1145
streamName 1146
streamSize 1146
streamSize (3D) 1147
strokeColor 1147
strokeWidth 1148
style 1148
subdivision 1149
subPicture 1150
subPictureCount 1150
suspendUpdates 1150
switchColorDepth 1151
systemTrayIcon 1152
systemTrayTooltip 1152
tabCount 1153
tabs 1153
target 1154
targetFrameRate 1154
tension 1155
text 1155
texture 1156
textureCoordinateList 1157
textureCoordinates 1157
textureLayer 1158
textureList 1158
textureMember 1159
textureMode 1160
textureModeList 1160
textureRenderFormat 1161
textureRepeat 1162
textureRepeatList 1163
textureTransform 1164
textureTransformList 1165
textureType 1167

thumbNail 1167
tilt 1168
time （タイムアウトオブジェクト） 1168
timeoutHandler 1169
timeoutList 1170
timeScale 1170
title (DVD) 1171
title （ウィンドウ） 1171
titlebarOptions 1172
titleCount 1173
toolXtraList 1173
toon （モディファイア） 1174
top 1175
topSpacing 1175
traceLoad 1176
traceLogFile 1176
traceScript 1177
trackCount （メンバ） 1178
trackCount （スプライト） 1178
trackEnabled 1179
trackInfo 1179
trackNextKeyTime 1180
trackNextSampleTime 1180
trackPreviousKeyTime 1181
trackPreviousSampleTime 1181
trackStartTime （メンバ） 1182
trackStartTime （スプライト） 1182
trackStopTime （メンバ） 1183
trackStopTime （スプライト） 1183
trackText 1184
trackType （メンバ） 1184
trackType （スプライト） 1185
trails 1186
transform （プロパティ） 1186
transitionType 1188
transitionXtraList 1188
translation 1188
transparent 1189
triggerCallback 1190
trimWhiteSpace 1191
tunnelDepth 1192
tweened 1192

tweenMode 1193
type (light) 1193
type （メンバ） 1194
type （モデルリソース） 1195
type (motion) 1196
type (shader) 1197
type (sprite) 1197
type (texture) 1198
type （ウィンドウ） 1198
updateLock 1199
updateMovieEnabled 1200
URL 1200
useAlpha 1201
useDiffuseWithTexture 1201
useFastQuads 1202
useHypertextStyles 1203
useLineOffset 1203
userData 1204
userName 1205
userName (RealMedia) 1205
useTargetFrameRate 1206
vertex 1207
vertexList 1207
vertexList (mesh generator) 1208
vertexList (mesh deform) 1209
vertices 1209
video (QuickTime, AVI) 1210
video (MP4Media/FLV) 1210
video (RealMedia, Windows Media) 1211
videoFormat 1211
videoForWindowsPresent 1212
viewH 1212
viewPoint 1213
viewScale 1214
viewV 1215
visible 1216
visible （スプライト） 1217
visibility 1217
volume (DVD) 1218
volume （メンバ） 1218
volume （ミクサ） 1219
volume (MP4Media/FLV) 1219

volume （サウンドチャネル） 1220
volume （サウンドオブジェクト） 1220
volume （スプライト） 1221
volume (Windows Media) 1221
warpMode 1222
width 1223
width (3D) 1223
width (MP4Media/FLV) 1224
widthVertices 1224
wind 1225
window 1225
windowBehind 1226
windowInFront 1226
windowList 1227
wordWrap 1227
worldPosition 1228
worldTransform 1228
wrapTransform 1229
wrapTransformList 1229
x （ベクトル） 1230
xAxis 1230
xtra 1231
xtraList （ムービー） 1232
xtraList （プレーヤ） 1232
y （ベクトル） 1233
yAxis 1233
yon 1234
z （ベクトル） 1234
zAxis 1235

第 15 章：物理エンジン

物理ワールドのプロパティ 1236
物理ワールドのメソッド 1242
剛体の管理メソッド 1246
剛体のプロパティ 1253
剛体のメソッド 1265
制限のメソッド 1267
制限のプロパティ 1273
スプリングのプロパティ 1277
衝突および衝突コールバックのメソッド 1281
地形のメソッド 1287
地形のプロパティ 1289

6DOF ジョイントのメソッド 1292
6DOF のプロパティ 1293
レイキャスティングのメソッド 1310
エラーコード 1311

索引 1313

第 1 章：はじめに

このリファレンスは、Adobe® Director® 11 のスクリプト記述に関する概念的な情報と手引き情報を記載しています。また、スクリプトの記述で使用するスクリプトアプリケーションプログラミングインターフェース（API）のリファレンスについて説明し、その実例を示しています。

スクリプト API は、スクリプトを通じて Director の機能にアクセスするための手段で、これにより、インタラクティビティをムービーに追加できます。この API を使用すると、Director 付属のあらかじめ用意されたビヘイビアーが提供する機能と同じインタラクティブ機能だけでなく、これよりもさらに強力で、かつ多様な機能を作成できます。

あらかじめ用意されたビヘイビアーを使用すると、基本的なインタラクティブ機能をムービーに追加して、再生ヘッドをフレーム番号やマーカに移動したり、スプライトをクリックしてズームインしたりできます。また、このビヘイビアーにより、スプライトアニメーション、メディアのロード、フレームナビゲーションなどの非インタラクティブ機能も有効になります。スクリプト API を使用すると、これらの機能を拡張したりカスタマイズしたりできます。

対象ユーザ

このリファレンスは、次の作業を行う必要があるユーザを対象としています。

- あらかじめ用意されたビヘイビアーの既存の機能を、スクリプトを使用して拡張するユーザ
- あらかじめ用意されたビヘイビアーではなくスクリプトを使用して、ムービーに機能を追加するユーザ
- あらかじめ用意されたビヘイビアーが提供する機能よりも強力で、かつ多様なカスタム機能をムービーに追加するユーザ

このリファレンスは、スクリプトを使用してインタラクティビティをムービーに追加するのに必要な情報を提供することを目的としています。これには、基本的な情報から詳細情報まで、すべての情報が含まれています。したがって、これまでにスクリプト記述の経験がなくても、Director で効果的なスクリプトを記述することができます。

Director、Lingo、または JavaScript™ シンタックスの経験のレベルにかかわらず、スクリプトの記述を開始する前に、「4 ページの「Director スクリプトに関する重要事項」」と「40 ページの「Director でのスクリプトの記述」」に目を通すようにしてください。他の製品と同様、Director にはスクリプト記述に関する固有の規則やデータの種類などがあります。効果的なスクリプトを記述するには、こうした Director 固有の特性に慣れておく必要があります。

Director スクリプト記述の新機能

以前のバージョンの Director でスクリプトを記述したことがある場合は、この最新リリースのスクリプト記述に関する新機能と重要な変更に注意してください。

強化されたオーディオ機能 5.1 チャネルサラウンドサウンド、リアルタイムミキシング、オーディオ効果および DSP フィルタを使用する場合と同じぐらい、ゲームサウンドをリアルにします。詳細については、「サウンド」、「ミクサ」、および「オーディオフィルタ」を参照してください。

ビデオ拡張機能 フルスクリーン、高解像度ビデオ、ゲームを提供する H.264 ビデオ統合により、マルチメディア機能を強化します。詳細については、「122 ページの「MP4Media/FLV」」を参照してください。

新しいデータ型 ByteArray データ型を使用して、バイナリデータのアクセスと操作を行います。詳細については、「144 ページの「ByteArray」」を参照してください。

強化された物理エンジン 動的な凹型の剛体をサポートします。詳細については、「1265 ページの「剛体のメソッド」」を参照してください。

クロスドメインポリシー 詳細については、「loadPolicyFile()」を参照してください。

テキストエンジンの機能強化 •Macintosh OS でサポートされる OTF（Open Type Font）の埋め込み。

- クラスベースのカーニングのサポート。
- メンバレベルでのヒンティングのサポート。「hinting」を参照してください。

プラットフォームとブラウザサポートの機能強化 •Macintosh OSX 10.5（オーサリングおよびランタイム）のサポート。

- Firefox 3.0 のサポート。

  詳細については、Read Me ファイルを参照してください。

Director での Unicode サポートの制限

Adobe Director は、Unicode でのスクリプト記述をサポートしています。

- 右から左に記述する言語はサポートされていません。
- readchar()、getLength()、getPosition() などのファイル I/O Xtra 関数は、1 バイト文字入力でのみ動作します。2 バイトまたは 3 バイトの Unicode 文字を読み取るには、readFile() メソッドを使用して、ファイル全体をストリングオブジェクトに読み込みます。その後で、char...of メソッドを使用して Unicode 文字を読み取ります。
- Director Physics のコンポーネントは Unicode をサポートしていません。
- HTTP パスの Unicode 名はサポートされていません。
- kMoaMmDictType_MessageTable レジストリキーを使用して、スクリプティング Xtra に Unicode ストリングの名前を付けることはできません。また、Unicode で名前が付けられた Lingo 関数を、スクリプティング Xtra を使用して公開できません。
- スクリプティング Xtra は Lingo 関数を公開します。Xtra を通じて公開されたこれらの関数名は、Unicode ではサポートされていません。
- 3D モデル名は Unicode ではサポートされていません。

  スクリプトウィンドウには、スクリプトエディタ以外にエクスプローラパネルがあります。デフォルトでは、エクスプローラパネルはスクリプトエディタの左側に表示されます。エクスプローラパネルは、辞書ビューまたはスクリプトブラウザビューで表示できます。

このドキュメントで新しく変更された点

以前のバージョンの Director でスクリプトを記述する方法について既に習得している場合は、このリリースのスクリプトドキュメントにおける変更点を把握する必要があります。『**Director** スクリプトリファレンス』は、以前のバージョンの Director 付属の『**Lingo** 辞書』に取って代わるものです。このリファレンスの構成は『**Lingo** 辞書』とは異なっています。

『**Lingo** 辞書』では、スクリプトモデルに関する情報が機能別に記載されていました。例えば、スクリプトでのスプライトの動作方法について調べるには、スプライト見出しの下にあるセクションのいずれか、例えば、「スプライトのドラッグ」、「スプライトのサイズ」などで、その情報を探す必要がありました。また、すべてのスクリプト API が 1 つのアルファベット順のリストになっていました。関数、プロパティ、イベントなど、すべてが一緒になってアルファベット順に並べられていたのです。

『**Director** スクリプトリファレンス』では、スクリプトモデルに関する情報がオブジェクト別に整理されており、スクリプトで使用する実際のスクリプトオブジェクトの構成が詳細に反映されます。例えば、スクリプトでのスプライトの動作方法を調べるには、Director のコアオブジェクトの章でスプライトのセクションを探します。

スクリプト API は、引き続きアルファベット順に並べられていますが、API の種類ごとに分類されています。例えば、メソッド見出しの下にはすべてのメソッドが、プロパティ見出しの下にすべてのプロパティがアルファベット順にリストされています。

Director でのスクリプト情報の検索

新しく構成された『**Director Scripting Reference**』のトピックを次に示します。

Director スクリプトに関する重要事項　スクリプトの基本的な概念、および Director でスクリプトを記述する際に使用するコンポーネントに関する情報を記載しています。

Director でのスクリプトの記述　Director のスクリプト環境、および高度なスクリプト概念と技法について説明します。

Director でのスクリプトのデバッグ　スクリプトが思ったとおりに動作しない場合にそのスクリプトの問題点を見つける方法について説明します。

Director のコアオブジェクト　Director の再生エンジン、ムービーウィンドウ、スプライト、音など、Director のコアな機能にアクセスする際に使用する API、およびオブジェクトのリストを記載しています。

メディアタイプ　RealMedia、DVD、アニメーション GIF など、キャストメンバとしてムービーに追加された様々な Director メディアタイプの機能にアクセスする際に使用する API、およびメディアタイプのリストを記載しています。

スクリプトオブジェクト　Xtra 拡張機能としても知られているスクリプトオブジェクトのリスト、および Director のコアな機能を拡張する際に使用する API を記載しています。Xtra 拡張機能は、フィルタの読み込みやインターネットへの接続などの機能を提供します。

3D オブジェクト　3D 機能をムービーに追加する際に使用するオブジェクトのリストを記載しています。

定数　Director で使用できる定数のリストを記載しています。

イベントとメッセージ　Director で使用できるイベントのリストを記載しています。

キーワード　Director で使用できるキーワードのリストを記載しています。

メソッド　Director で使用できるメソッドのリストを記載しています。

演算子　Director で使用できる演算子のリストを記載しています。

プロパティ　Director で使用できるプロパティのリストを記載しています。

第 2 章：Director スクリプトに関する重要事項

Director® でスクリプトを記述するのが初めての場合は、まず最初に、Director でスクリプトを記述する方法を理解するのに欠かせない基本的なスクリプトの概念について習得することをお勧めします。この必要不可欠な基本的概念には、重要用語の定義、シンタックス規則、使用可能なデータ型、および Director でのスクリプト記述で使用する変数、配列、演算子などの基本エレメントに関する情報があります。

スクリプトの種類

Director ムービーには、ビヘイビアー、ムービースクリプト、親スクリプト、およびキャストメンバスクリプトの 4 種類のスクリプトを含めることができます。ビヘイビア、ムービースクリプト、および親スクリプトはすべて、独立したキャストメンバとしてキャストウィンドウに表示されます。キャストメンバスクリプトは、キャストウィンドウでそのキャストメンバに関連付けられており、独立したメンバとしては表示されません。

- **ビヘイビア**は、スコア内のスプライトまたはフレームに割り当てられるスクリプトであり、スプライトビヘイビアまたはフレームビヘイビアと呼ばれます。キャストウィンドウでは、各ビヘイビアーのサムネイルの右下隅に、ビヘイビアーアイコンが表示されます。

  『**Director** スクリプトリファレンス』で ビヘイビアー という用語を使用する場合は、スプライトまたはフレームに割り当てるスクリプトすべてを指すものとします。これは、Director のライブラリパレットにあるビヘイビアーとは異なります。Director に組み込まれているこれらのビヘイビアーについて詳しくは、Director ヘルプパネルの「Director ユーザガイド」のトピックを参照してください。

  キャストに追加されたすべてのビヘイビアは、ビヘイビアインスペクタのビヘイビアポップアップメニューに表示されます。その他のスクリプトは、そのポップアップメニューに表示されません。

  同じビヘイビアーをスコア内の複数の場所に割り当てることもできます。ビヘイビアーを編集すると、スコア内のすべての場所で使用されているビヘイビアーに変更結果が反映されます。

- **ムービースクリプト**には、グローバルまたはムービーレベルで使用できるハンドラが含まれています。ムービースクリプトのイベントハンドラは、ムービーの再生時にムービー内の別のスクリプトから呼び出せます。

  ムービースクリプトをキャストウィンドウに表示すると、サムネイルの右下隅にムービースクリプトアイコンが表示されます。

  ムービースクリプトは、再生中のフレームや操作中のスプライトに関係なく、ムービー全体に適用されます。ムービーがウィンドウ内で再生される場合も、リンクされたムービーとして再生される場合も、ムービースクリプトはそのムービーのみに適用されます。

- **親スクリプト**は、子オブジェクトの作成に使用する Lingo を含む特別なスクリプトです。親スクリプトを使用すると、同じような動作および反応をするスクリプトオブジェクトでありながら、互いに独立して動作するオブジェクトを作成することができます。親スクリプトをキャストウィンドウに表示すると、サムネイルの右下隅に親スクリプトアイコンが表示されます。

  親スクリプトと子オブジェクトの使用について詳しくは、「46 ページの「Lingo シンタックスでのオブジェクト指向プログラミング」」を参照してください。

  JavaScript シンタックスは、親スクリプトまたは子オブジェクトではなく、通常の JavaScript シンタックススタイルのオブジェクト指向プログラミング技術を使用しています。JavaScript シンタックスでのオブジェクト指向プログラミングについて詳しくは、54 ページの「JavaScript シンタックスでのオブジェクト指向プログラミング」を参照してください。

- 非表示のステージ（Flash スプライト）上に配置された **Flash® メンバのコンポーネント**にアクセスするには、メンバオブジェクトを使用するのが唯一の方法です。不可視のプロパティを持つ Flash スプライトの Sprite オブジェクトはエラーになります。
- キャストメンバ**スクリプト**は、スコアに関係なく、キャストメンバに直接割り当てられます。キャストメンバをスプライトに割り当てると、いつでもキャストメンバスクリプトが利用できます。

  ビヘイビアー、ムービースクリプト、および親スクリプトとは異なり、キャストメンバスクリプトはキャストウィンドウに表示されません。しかし、環境設定 キャストダイアログボックスで「表示」オプションの「スクリプトアイコン」ボックスをオンにすると、スクリプトが割り当てられているキャストメンバの場合は、キャストウィンドウのサムネイルの左下隅に小さなスクリプトアイコンが表示されます。

スクリプト用語集

Lingo および JavaScript シンタックスで使用する用語は、それぞれの言語に固有のものがあれば、両方の言語で共有しているものもあります。

以下に、重要なスクリプト用語をリストアップします。これらの用語は『**Director** スクリプトリファレンス』全体で使用されているため、先へ進む前に理解しておくと便利です。

- **定数**は、値が変化しないエレメントです。例えば、Lingo では、TAB、EMPTY、RETURN などの定数の値は常に同じで、変更できません。JavaScript シンタックスについては、Math.PI、Number.MAX_VALUE などの定数の値が常に同じで、変更できません。キーワード const を使用して、JavaScript シンタックスで独自のカスタム定数を作成することもできます。

  定数について詳しくは、13 ページの「定数」を参照してください。

- **イベント**は、ムービーの再生中に発生するアクションです。ムービーが停止したとき、スプライトが開始したとき、再生ヘッドがフレームに入ったとき、ユーザがキーボード入力を行ったときなどに、イベントが発生します。Director のイベントはすべて事前に定義されており、常に同じ意味を持っています。

  イベントについて詳しくは、25 ページの「イベント」を参照してください。

- **式**は、値を生成するステートメントの一部です。例えば、2 + 2 という記述は式です。
- **関数**は、トップレベル関数または特定の種類の JavaScript シンタックスコードを指します。

  トップレベル関数は、ムービーの再生中にムービーに何らかの動作を行うよう指示するか、値を返しますが、どの特定のオブジェクトからも呼び出されません。例えば、シンタックス list() を使用して、トップレベル list() 関数を呼び出します。関数と同様、メソッドも、ムービーの再生中にムービーに何らかの動作を行うよう指示するか、値を返しますが、メソッドは常にオブジェクトから呼び出されます。

  関数は JavaScript シンタックスで使用され、イベントハンドラ、カスタムオブジェクト、カスタムメソッドなどを表します。この場合の JavaScript 関数の使用については、このリファレンスの該当するトピックで後ほど説明します。

- **ハンドラ**（イベントハンドラ）は、特定のイベントおよびそれに続くメッセージに応答して実行される、スクリプト内のステートメントのセットです。イベントが発生すると、対応するメッセージが生成されスクリプトに送信されます。そして、そのメッセージに応答して、対応するハンドラが実行されます。ハンドラの名前は、そのハンドラが応答するイベントおよびメッセージと常に同じです。

  注意：JavaScript シンタックスでは、実際は関数によってイベントが処理されますが、このリファレンスでハンドラという用語が使用されている場合は、通常、イベントを処理する JavaScript シンタックス関数と Lingo ハンドラの両方を指します。

ハンドラについて詳しくは、27 ページの「ハンドラ」を参照してください。

- **キーワード**は、特別な意味を持った予約語のことです。例えば、Lingo では、キーワード end は、そこでハンドラが終了することを示します。JavaScript シンタックスでは、キーワード var は、それに続く用語が変数であることを示します。
- **リスト**（Lingo）または**配列**（JavaScript シンタックス）は、順序付けられた値のセットであり、変数のセットに割り当てる一連の名前や値など、データ配列の追跡や更新に使用します。簡単な例としては、[1, 4, 2] のような数値のリストが挙げられます。

  Lingo および JavaScript シンタックスでのリストの使用について詳しくは、30 ページの「リニアリストおよびプロパティリスト」を参照してください。

  JavaScript シンタックス配列の使用について詳しくは、37 ページの「JavaScript シンタックス配列」を参照してください。
- **メッセージ**は、ムービーで特定のイベントが発生したときに Director がスクリプトに送る通知です。例えば、再生ヘッドが特定のフレームに入ると、enterFrame イベントが発生し、Director が enterFrame メッセージを送ります。スクリプトに enterFrame ハンドラが含まれていると、そのハンドラが enterFrame メッセージを受け取るので、ハンドラ内のステートメントが実行されます。どのスクリプトにもメッセージに対応するハンドラがない場合、そのメッセージはスクリプト内で無視されます。

  メッセージについて詳しくは、26 ページの「メッセージ」を参照してください。
- **メソッド**は、ムービーの再生中にムービーに何らかの動作を行うよう指示するか値を返し、オブジェクトから呼び出されます。例えば、シンタックス _movie.insertFrame() を使用して、Movie オブジェクトから insertFrame() メソッドを呼び出します。機能的にはトップレベル関数と似ていますが、メソッドは常にオブジェクトから呼び出され、トップレベル関数はそうではありません。
- **演算子**は、1 つまたは複数の値から新しい値を計算する用語です。例えば、加算（+）演算子は 2 つまたはそれ以上の値を足し算して、1 つの新しい値を生成します。

  演算子について詳しくは、18 ページの「演算子」を参照してください。
- **パラメータ**は、スクリプトに値を渡すためのプレースホルダーです。パラメータは、メソッドおよびイベントハンドラにのみ適用されます。プロパティには適用されません。また、あるメソッドでは必須のパラメータが、他のパラメータでは省略できる場合があります。

  例えば、Movie オブジェクトの go() メソッドは、再生ヘッドを特定のフレームに送信し、そのフレームを含むムービーの名前をオプションで指定します。このタスクを実行するには、少なくも 1 つのパラメータが go() メソッドに必須です。そして、その他にパラメータをもう 1 つ指定します。最初の必須パラメータは、再生ヘッドの送信先フレームを指定し、2 番目のオプションのパラメータはフレームを含むムービーを指定します。最初のパラメータは必須なので、go() メソッドが呼び出されたときに、このパラメータが存在していないとスクリプトエラーが発生します。2 番目のパラメータはオプションなので、このパラメータが存在しなくてもメソッドはこのタスクを実行します。
- **プロパティ** は、オブジェクトを定義する属性です。例えば、ムービーのスプライトには、幅、高さ、バックカラーなどを指定する属性があります。この 3 つの特定の属性の値にアクセスするには、Sprite オブジェクトの width プロパティ、height プロパティ、および backColor プロパティを使用します。

  変数へのプロパティの割り当てについて詳しくは、15 ページの「変数の値の格納と更新」を参照してください。
- **ステートメント** は、Director が実行できる有効な命令です。すべてのスクリプトがステートメントのセットで構成されています。次の Lingo は単一の完全なステートメントです。

  ```
  _movie.go("Author")
  ```

  スクリプトステートメントの記述について詳しくは、41 ページの「ドットシンタックス形式でのスクリプト記述」を参照してください。.

- **変数**は、値を格納して更新するのに使用するエレメントです。変数の最初の文字は、アルファベットの文字、アンダースコア（_）、またはドル記号（$）にする必要があります。変数名でその後に続くに文字には、数値（0 ～ 9）を使用することもできます。変数への値の割り当てや、多数のプロパティの値の変更には、等号演算子（=）を使用します。

  変数の使用について詳しくは、14 ページの「変数」を参照してください。

スクリプトシンタックス

Lingo および JavaScript シンタックスに適用される一般的なシンタックス規則は次のとおりです。

- コメントマーカは、Lingo および JavaScript シンタックス間で異なります。

  Lingo コメントはすべて、その直前に二重ハイフン (--) を挿入します。複数行にわたるコメントについては、各行の先頭に二重ハイフンを付ける必要があります。

  ```
  -- This is a single-line Lingo comment

  -- This is a
  -- multiple-line Lingo comment
  ```

  JavaScript シンタックスのコメントが 1 行の場合、そのコメントの直前に二重スラッシュ（//）を挿入します。複数行にわたるコメントについては、コメントの先頭に /* を、末尾には */ を付けます。

  ```
  // This is a single-line JavaScript syntax comment

  /* This is a
  multiple-line JavaScript syntax comment */
  ```

  コメントは、独立した行にも、ステートメントの後にも記述できます。コメントマーカに続く同一行のテキストはすべて無視されます。

  コメントは、特定のスクリプトやハンドラに関するメモ、または目的がはっきりしないステートメントに関するメモなど、必要であれば自由に記述できます。コメントを付けておくと、作成者がいない場合でも、作成者以外のだれかが処理手順を容易に理解できます。

  多数のコメントを追加しても、ムービーファイルを圧縮した DCR または DXR ファイルとして保存すれば、容量は大きくなりません。コメントは、圧縮の過程でファイルから削除されます。

  また、コメントマーカはテストまたはデバッグの目的で機能させたくないコードのセクションを無視する場合にも使用できます。コードを削除するのではなくコメントマーカを追加することにより、セクションを一時的にコメントにできます。オンまたはオフにしたいコードを選択し、スクリプトウィンドウの「コメントを設定」または「コメントを解除」ボタンを使用すると、コメントマーカの追加または削除をすばやく行うことができます。

- すべてのメソッド名および関数名の後ろに括弧を付ける必要があります。例えば、Sound オブジェクトの beep() メソッドを呼び出す場合は、beep の後ろに括弧を挿入します。括弧を挿入しないと、スクリプトエラーが発生します。

  ```
  // JavaScript syntax
  _sound.beep(); // this statement will work properly
  _sound.beep; // this statement will result in a script error
  ```

  メソッド、関数、またはハンドラを、他のメソッド、関数、またはハンドラ内から呼び出すときは、呼び出しステートメントに括弧を挿入する必要があります。次の例では、spriteClicked ハンドラへの呼び出しが modifySprite() メソッドに含まれています。spriteClicked ハンドラへの呼び出しには括弧を含める必要があります。括弧がないと、スクリプトエラーが発生します。

```
// JavaScript syntax
function modifySprite() {
    spriteClicked(); // this call to the handler will work properly
    spriteClicked; // this call to the handler results in a script error
}
function spriteClicked() {
    // handler code here
}
```

また、括弧は、数値演算の優先順位を変更したり、ステートメントを読みやすくしたりするためにも使用します。例えば、次に示す最初の数式の計算結果は 13 になります。一方、2 番目の数式の計算結果は 5 になります。

```
5 * 3 - 2 -- yields 13
5 * (3 - 2) -- yields 5
```

- イベントハンドラは、Lingo および JavaScript シンタックス間で異なります。Lingo では、ハンドラはシンタックス on **handlerName** を使用します。JavaScript シンタックスでは、ハンドラは関数として実装され、シンタックス function **handlerName**() を使用します。例えば、次のステートメントは、マウスボタンをクリックすると警告音を再生するというハンドラを構成します。

```
-- Lingo syntax
on mouseDown
    _sound.beep()
end
// JavaScript syntax
function mouseDown() {
    _sound.beep();
}
```

- イベントハンドラパラメータは、Lingo および JavaScript シンタックス間で異なります。Lingo と JavaScript シンタックスの両方が、括弧内でハンドラに渡されるエンクロージングパラメータをサポートしています。複数のパラメータが渡される場合は、各パラメータの間はカンマで区切ります。Lingo では、括弧で囲まれていないパラメータを渡すこともできます。例えば、次の addThem ハンドラは、a と b の 2 つのパラメータを受け取ります。

```
-- Lingo syntax
on addThem a, b -- without parentheses
    c = a + b
end

on addThem(a, b) -- with parentheses
    c = a + b
end

// JavaScript syntax
function addThem(a, b) {
    c = a + b;
}
```

- const キーワードは JavaScript シンタックスで使用します。このキーワードを使用すると、値が変化しない定数を指定できます。Lingo には、事前定義済みの独自の定数セット（TAB、EMPTY など）があります。したがって、キーワード const は、Lingo には適用されません。

  例えば、次のステートメントは、intAuthors という名前の定数を指定し、値を 12 に設定します。この値は常に 12 になり、スクリプトを通じて変更することはできません。

```
// JavaScript syntax
const intAuthors = 12;
```

- JavaScript シンタックスの var キーワードは用語の前に配置して、その用語が変数であることを指定します。次のステートメントは、startValue という名前の変数を作成します。

```
// JavaScript syntax
var startValue = 0;
```

注意：JavaScript シンタックスでの var の使用はオプションですが、var を使用して、常にローカル JavaScript シンタックスを変数、または関数内の変数として宣言することをお勧めします。変数の使用について詳しくは、14 ページの「変数」を参照してください。

- Lingo の行継続文字（\）は、サンプルコードの長文が 2 行以上に区切られていることを示します。このように区切られた Lingo の行は、別々の行を示しているわけではありません。例えば、次のコードは引き続き動作します。

```
-- Lingo syntax
tTexture = member("3D").model("box") \
    .shader.texture
```

JavaScript シンタックスには、行継続文字は含まれません。複数行にわたる JavaScript シンタックスコードを区切るには、行の末尾に改行を追加してから、次の行でそのコードを続けて入力します。

- セミコロンを使用すると、JavaScript シンタックスコードのステートメントの終わりを指定できます。セミコロンは Lingo には適用されません。

セミコロンの使用はオプションです。使用する場合は、完全なステートメントの末尾に配置されます。例えば、次のステートメントは両方とも、startValue という名前の変数を作成します。

```
// JavaScript syntax
var startValue = 0
var startValue = 0;
```

セミコロンは、必ずしも JavaScript シンタックスコードの終わりを指定しているわけではありません。このセミコロンによって、複数のステートメントを 1 つの行に配置することもできます。ただし、読みやすさを考慮し、別々のステートメントは別個の行に配置することをお勧めします。例えば、次の 3 つのステートメントは 1 行のコードになっており、正しく機能しますが、このコードは読みにくくなっています。

```
// JavaScript syntax
_movie.go("Author"); var startValue = 0; _sound.beep();
```

- 式やステートメント内の文字スペースは、Lingo でも JavaScript シンタックスでも無視されます。引用符で囲まれたストリングでは、スペースは文字として扱われます。ストリングにスペースが必要な場合は、明示的に挿入する必要があります。例えば、次の最初のステートメントでは、リスト項目間のスペースは無視されます。2 番目のステートメントには、スペースが含まれます。

```
-- Lingo syntax
myList1 = ["1",      "2",      "3"] -- yields ["1", "2", "3"]
myList2 = ["  1  ", "  2  ", "  3  "] -- yields ["  1  ", "  2  ", "  3  "]
```

- 大文字と小文字の区別は、Lingo および JavaScript シンタックス間で異なります。

Lingo では大文字と小文字を区別しないため、大文字と小文字を自由に組み合わせて使用できます。例えば、次の 4 つのステートメントはいずれも同じ意味になります。

```
-- Lingo syntax
member("Cat").hilite = true
member("cat").hiLite = True
MEMBER("CAT").HILITE = TRUE
Member("Cat").Hilite = true
```

Lingo では大文字と小文字が区別されませんが、大文字と小文字の規則を自分で決め、スクリプトを通して常にその規則に従うことをお勧めします。これにより、ハンドラ、変数、キャストメンバなどの名前を識別しやすくなります。

JavaScript シンタックスでは、オブジェクト、またはオブジェクトを参照するトップレベルのプロパティまたはメソッドを参照するとき、あるいはユーザ定義の変数を参照するとき、大文字と小文字を区別します。例えば、トップレベルの sprite() メソッドは特定の Sprite オブジェクトへの参照を返し、Director ではすべて小文字で実装されます。次の示す最初のステートメントは、ムービー内の最初のスプライトの名前を参照しますが、一方、2 番目と 3 番目のステートメントはスクリプトエラーになります。

```
// JavaScript syntax
sprite(1).name // This statement functions normally
Sprite(1).name // This statement results in a script error
SPRITE(1).name // This statement results in a script error
```

リテラルストリングは、Lingo および JavaScript シンタックスの両方で常に大文字と小文字を区別します。

ストリングの使用について詳しくは、12 ページの「ストリング」を参照してください。

データの種類

データ型は、似たような定義済み特性の値を持つデータのセットです。Director のすべての変数およびプロパティ値が特定のデータ型です。また、メソッドで返される値も特定のデータ型です。

例えば、次の 2 つのステートメントについて考えてみます。最初のステートメントでは、変数 intX に自然数の値 14 に割り当てられています。この値は整数です。したがって、変数 intX のデータ型は整数型になります。2 番目のステートメントでは、変数 stringX に文字シーケンスが割り当てられています。この値はストリングです。したがって、変数 stringX のデータ型はストリング型になります。

```
-- Lingo syntax
intX = 14
stringX = "News Headlines"

// JavaScript syntax
var intX = 14;
var stringX = "News Headlines";
```

メソッドまたは関数によって返される値も、固有のデータ型です。例えば、Player オブジェクトの windowPresent() メソッドは、ウィンドウが存在するかどうかを指定する値を返します。戻り値は TRUE（1）または FALSE（0）です。

データ型の中には、Lingo と JavaScript 間で共有されるものがあります。また、言語固有のデータ型もあります。Director がサポートするデータ型のセットは固定されており、変更できません。つまり、新しいデータ型を追加したり、既存のデータ型を削除したりすることはできません。Director がサポートするデータ型を次に示します。

データ型	解説
# (シンボル)	状態やフラグを表すために単独で使用することができます。#list、#word など。
配列型	（JavaScript シンタックスのみ）文字どおりのデータ型ではありませんが、Array オブジェクトを使用して値のリニアリストを操作できます。Array オブジェクトの機能は、Lingo のリストデータ型の機能と似ています。
ブール値	TRUE（1）または FALSE（0）の値。Lingo では、TRUE または FALSE の値はすべてシンプルな整数定数で、TRUE は 1、FALSE は 0 です。JavaScript シンタックスでは、true または false の値は、デフォルトでは実際のブール型の値 true または false です。ただし、必要があれば、Director でシンプルな整数定数に自動的に変換されます。 Lingo では、TRUE および FALSE は小文字でも大文字でも構いません。JavaScript シンタックスでは、true および false は常に小文字でなければなりません。
色	オブジェクトの色を表します。
定数型	データの一部です。このデータの値は変化しません。
日付型	文字どおりのデータ型ではありませんが、JavaScript シンタックスでは、Date オブジェクトを使用して日付を操作できます。Lingo では、date() メソッドを使用して Date オブジェクトを作成し、日付を操作します。
浮動小数	（Lingo のみ）浮動小数点数です。2.345、45.43 など。
機能	（JavaScript シンタックスのみ）文字どおりのデータ型ではありませんが、Function オブジェクトを使用して、実行するコードのストリングを指定できます。
整数	（Lingo のみ）整数です。5、298 など。

データ型	解説
List	値で構成されたリニアリスト、またはプロパティの名前と値で構成されたプロパティリストです。
数学型	（JavaScript シンタックスのみ）文字どおりのデータ型ではありませんが、Math オブジェクトを使用して演算関数を実行できます。
null	（JavaScript シンタックスのみ）数値のコンテキストでは 0 として、ブール値のコンテキストでは false として動作する値を持つ変数を意味します。
数値	（JavaScript シンタックスのみ）文字どおりのデータ型ではありませんが、Number オブジェクトを使用して、最大値、非数（NaN）、無限値などの数値定数を表すことができます。
オブジェクト	文字どおりのデータ型ではありませんが、Object オブジェクトを使用して、データ、およびそのデータ上で機能するメソッドを含むコンテナを作成し、カスタム名を付けることができます。
ポイント	水平座標および垂直座標の両方を持つ座標空間のポイントです。
矩形型	座標空間内の長方形です。
RegExp 型	（JavaScript のみ）ストリング内の文字の組み合わせを一致させるのに使用する正規表現パターンです。
ストリング	キーボード上の記号または文字の値の連続シーケンスです。「Director」、「$5.00」など。
未定義型	（JavaScript シンタックスのみ）値を持たない変数を意味します。
Vector	3D 空間のポイントです。
VOID	（Lingo のみ）空の値を意味します。

注意：JavaScript シンタックス固有のデータ型およびオブジェクトの多くに独自のメソッドおよびプロパティのセットが含まれており、このメソッドやプロパティを使用すると、これらの型をさらに詳しく操作できます。『Director スクリプトリファレンス』では、こうしたメソッドやプロパティを参照している場合がありますが、詳しいリファレンス情報については触れていません。データ型やオブジェクト、およびおそのメソッドやプロパティについて詳しくは、市販の関連リソースを参照してください。

Cast オブジェクトの name プロパティなど、Director の組み込みプロパティには、そのプロパティと同じデータ型の値のみを割り当てることができます。例えば、Cast オブジェクトの name プロパティの固有のデータ型はストリング型です。したがって、値は、News Headlines などのストリングでなければなりません。このプロパティに、別のデータ型の値、例えば整数 20 を割り当てようとすると、スクリプトエラーが発生します。

独自のカスタムプロパティを作成する場合、そのプロパティの値は、初期値のデータ型に関係なく任意のデータ型にできます。

Lingo と JavaScript シンタックスの両方が動的に型付けされています。つまり、変数のデータ型を、宣言時に指定する必要はありません。データ型は、スクリプトの実行中に必要に応じて変換されます。

例えば、次の JavaScript シンタックスは、最初は変数 myMovie を整数に設定しています。この変数は、後からスクリプト内でストリングに設定されます。スクリプトが実行されると、myMovie のデータ型は自動的に変換されます。

```
-- Lingo syntax
myMovie = 15 -- myMovie is initially set to an integer
...
myMovie = "Animations" -- myMovie is later set to a string

// JavaScript syntax
var myMovie = 15; // myMovie is initially set to an integer
...
myMovie = "Animations"; // myMovie is later set to a string
```

リテラル値

リテラル値は、変数やスクリプトエレメントとしてではなく、文字どおりの内容として使用されるステートメントまたは式の一部です。スクリプトで使用されるリテラル値には、ストリング、整数、小数、キャストメンバの名前と番号、フレームとムービーの名前と番号、シンボル、および定数があります。

リテラル値には、タイプによって異なる表記規則があります。

ストリング

ストリングは、変数ではなく通常の文字として扱われる単語や文字のかたまりです。ストリングは二重引用符で囲む必要があります。例えば、ストリングを使用して、ムービーを見るユーザにメッセージを表示したり、キャストメンバに名前を付けることができます。次のステートメントでは、Hello と Greeting はどちらもストリングです。Hello はテキストキャストメンバに挿入されるリテラルテキストであり、Greeting はそのキャストメンバの名前です。

```
-- Lingo syntax
member("Greeting").text = "Hello"
```

同様に、ストリングをテストする場合も、次の例のように各ストリングを二重引用符で囲みます。

```
-- Lingo syntax
if "Hello Mr. Jones" contains "Hello" then soundHandler
```

Lingo および JavaScript シンタックスの両方で、ストリングの先頭や末尾にあるスペースもストリングのリテラル部分として扱います。次の式では、**to** という語の後にスペースが挿入されています。

```
// JavaScript syntax
trace("My thoughts amount to ");
```

Lingo では、キャストメンバや変数などを参照するときは大文字と小文字を区別しませんが、リテラルストリングでは大文字と小文字を区別します。例えば、次の 2 つのステートメントでは Hello と HELLO がリテラルストリングのため、指定したキャストメンバには異なるテキストが入ります。

```
-- Lingo syntax
member("Greeting").text = "Hello"
member("Greeting").text = "HELLO"
```

Lingo では、string() 関数は数値をストリングに変換します。JavaScript シンタックスでは、toString() メソッドが数値をストリングに変換できます。

注意：JavaScript シンタックスで、null または undefined 値に対して toString() メソッドを使用しようとすると、スクリプトエラーになります。これとは対照的に、Lingo の場合、string() 関数は、VOID を含めすべての値で動作します。

数値

Lingo では、整数と小数の 2 つの種類の数値があります。

整数は、小数部分や小数点を持たない、-2,147,483,648 ～ +2,147,483,647 の範囲内の自然数です。整数は、カンマを使わずに入力します。負の数には、先頭にマイナス記号（-）を付けます。

小数は浮動小数（浮動小数点数）とも呼ばれ、小数点を含むすべての数値を意味します。Lingo では、floatPrecision プロパティは、小数の表示に使用する小数点以下の桁数を制御します。Director では計算には必ず 15 桁までの完全数を使用します。15 桁を超える数は、計算では四捨五入されます。

JavaScript シンタックスは、整数と浮動小数を区別せず、単に数値を使用します。例えば、次のステートメントは、数値 1 が Lingo では整数、JavaScript シンタックスでは数値であることを、小数 1.05 が Lingo では浮動小数点数、JavaScript シンタックスでは数値であることを示しています。

```
-- Lingo syntax
put(ilk(1)) -- #integer
put(ilk(1.05)) -- #float

// JavaScript syntax
trace(typeof(1)) // number
trace(typeof(1.05)) // number
```

Lingo では、小数を整数に変換するには、integer() 関数を使用します。整数で数値演算を行って整数を小数に変換することもできます。例えば、整数と小数を乗算します。JavaScript シンタックスでは、ストリングまたは小数を自然数に変換するには、parseInt() 関数を使用します。Lingo の integer() 関数とは異なり、parseInt() では切り捨てが行われます。例えば、次のステートメントでは、Lingo の場合は小数 3.9 が四捨五入され整数 4 に、JavaScript シンタックスの場合は切り捨てられ数値 3 に変換されています。

```
-- Lingo syntax
theNumber = integer(3.9) -- results in a value of 4

// JavaScript syntax
var theNumber = parseInt(3.9); // results in a value of 3
```

Lingo では、value() 関数はストリングを数値に変換します。

また、-1.1234e-100 や 123.4e+9 のように、小数と指数表記を併用することもできます。

Lingo では、整数またはストリングを小数に変換するには、float() 関数を使用します。JavaScript シンタックスでは、ストリングを小数に変換するには、parseFloat() 関数を使用します。例えば、次のステートメントは、変数 theNumber に、Lingo の場合は 3.0000、JavaScript シンタックスの場合は 3 という値を格納します。

```
-- Lingo syntax
theNumber = float(3) -- results in a value of 3.0000

// JavaScript syntax
var theNumber = parseFloat(3) // results in a value of 3
```

定数

定数は、名前を持ち、内容が変化しない値です。

Lingo では、定義済みの TRUE、FALSE、VOID、および EMPTY は、その値が常に同じなので定数です。定義済みの BACKSPACE、ENTER、QUOTE、RETURN、SPACE、および TAB は、キーボード文字を参照する定数です。例えば、最後に押されたキーがスペースバーかどうかをテストするには、次のステートメントを使用します。

```
-- Lingo syntax
if _key.keyPressed() = SPACE then beep()
```

JavaScript シンタックスでは、JavaScript シンタックス固有のデータ型のいくつかを使用して、定義済み定数にアクセスできます。例えば、Number オブジェクトには Number.MAX_VALUE や Number.NaN などの定数が、Math オブジェクトには Math.PI や Math.E などの定数が含まれています。

注意：このリファレンスでは、JavaScript シンタックスの定義済み定数の詳細については説明していません。これらの定数について詳しくは、市販の関連リソースを参照してください。

JavaScript シンタックスでは、const キーワードを使用して、独自のカスタム定数を定義することもできます。例えば、次のステートメントは、items という名前の定数を作成し、その定数に値 20 を割り当てます。一度作成した定数の値は変更できません。

```
// JavaScript syntax
const items = 20;
```

定数について詳しくは、164 ページの「定数」を参照してください。

シンボル

Lingo では、シンボルはポンド記号 (#) で始まるストリングなどの値です。

シンボルはユーザが定義する定数です。シンボルを使用した比較は通常、高速で実行されるため、コードがより効率的になります。

注意：Lingo では、大文字のシンボルは小文字にマップされます。シンボル関数を使用して Sonia というストリングをシンボルに変換すると、sonia と出力されます。

例えば、次の最初のステートメントは、2 番目のステートメントよりも高速に実行されます。

```
-- Lingo syntax
userLevel = #novice
userLevel = "novice"
```

シンボルには、スペースや句読点は使用できません。

Lingo および JavaScript シンタックスの両方で、ストリングをシンボルに変換するには、symbol() 関数を使用します。

```
-- Lingo syntax
x = symbol("novice") -- results in #novice

// JavaScript syntax
var x = symbol("novice"); // results in #novice
```

シンボルをストリングに戻すには、Lingo の場合は string() 関数を、JavaScript シンタックスの場合は toString() メソッドを使用します。

```
-- Lingo syntax
x = string(#novice) -- results in "novice"

// JavaScript syntax
var x = symbol("novice").toString(); // results in "novice"
```

JavaScript シンタックスでは、同じシンボルを参照しているかどうかを確認するために、同じ名前のシンボルを比較することはできません。同じ名前のシンボルを比較するには、toString() メソッドを使用して、そのシンボルをストリングに変換してから、比較を実行します。

変数

Director では、値の格納と更新に変数を使用します。変数は、その名前が示すように、ムービーの再生時に変更または更新できる値を格納します。ムービーの再生時に変数の値を変更することによって、Director は URL の保存、オンラインチャットセッションへの参加回数の記録、またはネットワーク処理が完了したかどうかの記録、およびその他の様々なオプションを実行できます。

最初に変数を宣言するときには、変数に既知の値を割り当てることをお勧めします。これを変数の初期化といいます。変数を初期化することにより、ムービーの再生時に変数の値を追跡して比較するのがより容易になります。

変数にはグローバル変数とローカル変数があります。ローカル変数は、定義されているハンドラの実行中にのみ存在します。グローバル変数は、ムービーが他のムービーに分岐しても、Director が実行されている限り存在し続けます。変数は個々のハンドラ、特定のスクリプト、またはムービー全体の中でグローバルになることができ、そのスコープは、グローバル変数がどのように初期化されているかによって異なります。

ムービー全体で変数を使用できるようにしたい場合は、その変数を、Lingo の場合は on prepareMovie ハンドラで、JavaScript シンタックスの場合は function prepareMovie() ハンドラで宣言することをお勧めします。これで確実に、ムービーの冒頭から変数を使用できます。

グローバル変数およびローカル変数の使用について詳しくは、16 ページの「グローバル変数の使用」および 17 ページの「ローカル変数の使用」を参照してください。

変数の値の格納と更新

変数には、整数、ストリング、TRUE または FALSE の値、シンボル、リスト、計算結果など、Director で使用されるすべてのデータ型のデータを格納できます。等号演算子（=）を使用すると、プロパティや変数の値を格納できます。

このリファレンスの「データ型」セクションで説明したように、変数は Lingo でも JavaScript シンタックスでも動的に型付けされています。つまり、これらの変数はいつでも、異なるデータ型を格納することができます。（変数の型を変更できる点で、Lingo は他の言語と大きく異なります。Java™ や C++ などの言語では、変数の型を変更することはできません。）

例えば、x = 1 というステートメントは変数 x を作成します。この変数には整数が割り当てられているので、整数変数になります。次に x = "one" というステートメントを実行すると、変数 x にはストリングが含まれることになるので、ストリング変数になります。

ストリングを数値に変換するには、Lingo の場合は value() 関数を、JavaScript シンタックスの場合は parseInt() メソッドを使用します。また、数値をストリングに変換するには、Lingo の場合は string() 関数を、JavaScript シンタックスの場合は toString() メソッドを使用します。

プロパティの中には値を設定したり（値は割り当てられます）返したり（値は取得されます）できるものがあれば、返すことのみが可能なプロパティ値もあります。値を設定したり返したりできる値を持つプロパティは読み取り / 書き込み、返すことのみが可能な値を持つプロパティは読み取り専用と呼ばれます。

多くの場合、このようなプロパティは Director で制御できない条件を記述したものです。例えば、Adobe® Shockwave® コンテンツが含まれるムービー内のチャネル数を示す numChannels キャストメンバプロパティに、値を設定することはできません。ただし、キャストメンバの numChannels プロパティを参照することにより、チャネルの数を返すことができます。

変数への値の割り当て

❖ 等号演算子（=）を使用します。

例えば、次のステートメントは、変数 placesToGo に URL を割り当てます。

```
// JavaScript syntax
var placesToGo = "http://www.adobe.com";
```

変数には、数値演算の結果を格納することもできます。例えば、次のステートメントは、変数 mySum に加算の結果を割り当てます。

```
-- Lingo syntax
mySum = 5 + 5 -- this sets mySum equal to 10
```

もう 1 つの例として、次のステートメントは、スプライトの member プロパティを取得して、スプライト 2 に割り当てられたキャストメンバを返し、それを変数 textMember に配置します。

```
-- Lingo syntax
textMember = sprite(2).member
```

変数には、その用途を示す変数名を付けることをお勧めします。これによりスクリプトが読みやすくなります。例えば、変数 mySum は、数値の合計を格納する変数を示します。

プロパティまたは変数の値のテスト

❖ メッセージウィンドウで put() 関数または trace() 関数を使用するか、ウォッチャーウィンドウで値を調べます（put() と trace() は同じ機能を提供し、Lingo および JavaScript シンタックスの両方で使用できます）。

例えば、次のステートメントは、変数 myNumber に割り当てられている値をメッセージウィンドウに表示します。

```
-- Lingo syntax
myNumber = 20 * 7
put(myNumber) -- displays 140 in the Message window

// JavaScript syntax
var myNumber = 20 * 7;
trace(myNumber) // displays 140 in the Message window
```

グローバル変数の使用

グローバル変数は、複数のハンドラ、スクリプト、またはムービーで共用できる変数です。Director が実行されている限り、または clearGlobals() メソッドを呼び出すまで、グローバル変数は存在し、その値は保持されます。

Adobe Shockwave Player では、グローバル変数は goToNetMovie() メソッドで表示されるムービーの中では存続しますが、goToNetPage() メソッドで表示されるムービーの中では存続しません。

変数をグローバル変数として宣言するすべてのハンドラは、その変数の現在値を使用できます。ハンドラにより変数の値が変更された場合、変数をグローバルとして処理する他のすべてのハンドラで新しい値を使用できます。

すべてのグローバル変数の名前を小文字の **g** で始めることをお勧めします。これは、コードを調べるときに、グローバル変数を識別するのに役立ちます。

Director では、現在のすべてのグローバル変数とその現在の値を表示したり、すべてのグローバル変数の値を消去したりできます。

現在のすべてのグローバル変数とその現在値の表示

❖ メッセージウィンドウで、Global オブジェクトの showGlobals() メソッドを使用します。

メッセージウィンドウについて詳しくは、79 ページの「メッセージウィンドウでのデバッグ」を参照してください。

現在のすべてのグローバル変数の消去

❖ メッセージウィンドウで Global オブジェクトの clearGlobals() メソッドを使用して、すべてのグローバル変数の値を、Lingo の場合は VOID に、JavaScript シンタックスの場合は undefined に設定します。

ムービーの再生中にグローバル変数の値を監視するには、オブジェクトインスペクタを使用します。オブジェクトインスペクタについて詳しくは、83 ページの「オブジェクトインスペクタでのデバッグ」を参照してください。

Lingo のグローバル変数

Lingo では、変数はデフォルトでローカルと見なされ、キーワードを変数名の前に付ける必要がありません。グローバル変数を宣言するには、変数の前にキーワード global を付けます。

スクリプトの先頭、およびハンドラの前でグローバル変数を宣言すると、その変数は、その特定のスクリプト内のすべてのハンドラで使用できます。ハンドラ内でグローバル変数を宣言すると、変数は、そのハンドラでのみ使用可能になります。ただし、2 つの別個のハンドラで同じ名前のグローバル変数を宣言した場合は、一方のハンドラで変数の値を更新すると、その更新内容がもう一方のハンドラの変数に反映されます。

次の例は、2 つのグローバル変数、gScript と gHandler を示しています。gScript は、スクリプト内のすべてのハンドラで使用できます。一方、gHandler は、その定義ハンドラ、およびハンドラの先頭行でその変数を宣言している他のハンドラ内で使用できます。

```
-- Lingo syntax
global gScript -- gScript is available to all handlers

on mouseDown
    global gHandler
    gScript = 25
    gHandler = 30
end

on mouseUp
    global gHandler
    trace(gHandler) -- displays 30
end
```

Lingo では、global を使用してグローバル変数を定義すると、変数の初期値が自動的に VOID に設定されます。

JavaScript シンタックスのグローバル変数

JavaScript シンタックスでは、変数はデフォルトでグローバルと見なされます。グローバル変数のスコープは、その変数がどこでどのように宣言されたかによって決まります。

- JavaScript シンタックス関数の中で、変数名の前にキーワード var を付けずに変数を宣言すると、その変数はそのスクリプト内のすべての関数で使用できます。
- JavaScript シンタックス関数の外で変数を宣言する場合、キーワード var を付けても付けなくても、その変数はそのスクリプト内のすべての関数で使用できます。
- JavaScript シンタックス関数の中または外で、シンタックス _global.**varName** を使用して変数を宣言すると、その変数はムービー内のすべてのスクリプトで使用できます。

次の例では、1 つのスクリプトでシンタックス _global.gMovie を使用して、変数 gMovie をグローバルとして宣言しています。この変数は、ムービー内のすべてのスクリプトで使用できます。

```
// JavaScript syntax
_global.gMovie = 1; // Declare gMovie in one script

// Create a function in a separate script that operates on gMovie
function mouseDown() {
    _global.gMovie++;
    return(_global.gMovie);
}
```

次の例では、グローバル変数 gScript を 1 つのスクリプトで宣言しています。この変数は、そのスクリプト内の関数でのみ使用できます。

```
// JavaScript syntax
var gScript = 1; // Declare gScript in a script

// gScript is available only to functions in the script that defines it
function mouseDown() {
    gScript++;
    return(gScript);
}
```

JavaScript シンタックスでは、任意のハンドラの前で変数を定義すると、変数の初期値が自動的に undefined に設定されます。

ローカル変数の使用

ローカル変数は、定義されているハンドラの実行中にのみ存在します。ただし、ローカル変数の作成後、その変数を定義したハンドラが実行中であれば、他の式でローカル変数を使用したり、格納されている値を変更したりできます。

1 つのハンドラ内で一時的に変数を使用する場合は、変数をローカル変数とすることをお勧めします。それにより、別のハンドラにある同じ変数名の値を誤って変更することがなくなります。

ローカル変数の作成

- Lingo では、等号演算子（=）を使用して、変数に値を割り当てます。
- JavaScript シンタックスでは、関数内で変数名の前にキーワード var を付けてから、等号演算子を使用して変数に値を割り当てます。

  注意：JavaScript シンタックスの変数はデフォルトでグローバルなので、キーワード var を使用せずに関数内でローカル変数を宣言しようとすると、スクリプトによって予期せぬビヘイビアーが生じることがあります。したがって、var の使用はオプションですが、ローカル JavaScript シンタックス変数を宣言するときは、予期せぬビヘイビアーを回避するためにも var を使用することを強くお勧めします。

ハンドラ内の現在のローカル変数をすべて表示

❖ Lingo でのみ、showLocals() 関数を使用します。

Lingo では、メッセージウィンドウまたはハンドラ内でこのメソッドを使用すると、デバッグに役立ちます。コマンドの実行結果はメッセージウィンドウに表示されます。showLocals() メソッドは、JavaScript シンタックスには適用されません。

ムービーの再生中にローカル変数の値を監視するには、オブジェクトインスペクタを使用します。オブジェクトインスペクタについて詳しくは、83 ページの「オブジェクトインスペクタでのデバッグ」を参照してください。

演算子

演算子は、式の値の組み合わせ、比較、または変更の方法を Lingo および JavaScript シンタックスのスクリプトに指示するエレメントです。Director の演算子の多くは Lingo および JavaScript シンタックス間で共有されていますが、それぞれの言語に固有の演算子もあります。

演算子の種類の一部を次に示します。

- **算術演算子**（+、-、/、* など）。
- **比較演算子**（<、>、>= など）。これらの演算子は 2 つの引数を比較します。
- **論理演算子**（not、and、or）。これらの演算子は単純な条件を組み合わせて複合条件を生成します。
- **ストリング演算子**（&、&&、+ など）。これらの演算子はストリングを結合します。

注意：JavaScript シンタックスの演算子の種類は Lingo よりも多く、このリファレンスでは触れていない演算子もあります。JavaScript 1.5 のその他の演算子について詳しくは、市販の関連リソースを参照してください。

演算の対象項目はオペランドと呼ばれます。Lingo には 2 項演算子しかありません。JavaScript シンタックスには、2 項演算子と単項演算子の両方があります。**2** 項演算子 には、2 つのオペランドが必要です。1 つは演算子の前、もう 1 つは演算子の後ろに付けます。単項演算子には 1 つのオペランドが必要です。オペランドの位置は演算子の前でも後ろでも構いません。

次の例では、最初のステートメントが 2 項演算子を示しており、変数 x と y がオペランドで、プラス記号（+）が演算子です。2 番目のステートメントは単項演算子を示しており、変数 i がオペランドで、++ が演算子です。

```
// JavaScript syntax
x + y; // binary operator
i++; // unary operator
```

演算子のリファレンス情報については、683 ページの「演算子」を参照してください。

演算子の優先順位の理解

同じステートメントで複数の演算子を使用している場合は、一部の演算子が他に優先して実行されます。どの演算子を最初に実行するかは厳密に指定されています。これを演算子の優先順位といいます。例えば、乗算は常に加算より先に実行されます。ただし、括弧で囲まれたアイテムは乗算より先に実行されます。次の例では、ステートメントに括弧がないため、乗算が先に実行されます。

```
-- Lingo syntax
total = 2 + 4 * 3 -- results in a value of 14
```

加算の部分を括弧で囲むと、加算が先に実行されます。

```
-- Lingo syntax
total = (2 + 4) * 3 -- results in a value of 18
```

演算子の種類とその優先順位については、次の項で説明します。実行は、優先順位の高い演算子が先になります。例えば、優先順位が 5 の演算子は、優先順位が 4 の演算子より先に実行されます。同じ優先順位の演算子は左から右に実行されます。

算術演算子

算術演算子は、加算、減算、乗算、除算などの算術演算を実行します。括弧やマイナス記号も算術演算子です。

演算子	効果	優先順位
( )	演算をグループ化して、優先順位を制御します。	5
-	数値の前に配置した場合は、数値の符号を反転します。	5
*	乗算を実行します。	4
mod	（Lingo のみ）モジュロ演算（剰余演算）を実行します。	4
/	除算を実行します。	4
%	（JavaScript シンタックスのみ）2 つのオペランドで除算したときの整数の余りを返します。	4
++	（JavaScript シンタックスのみ）オペランドに 1 を加えます。接頭演算子（++x）として使用した場合は、オペランドに 1 を加えた後にその値を返します。接尾演算子（x++）として使用した場合は、オペランドに 1 を加える前にその値を返します。	4
--	（JavaScript シンタックスのみ）オペランドから 1 を引きます。戻り値は増分演算子のものと似ています。	4
+	2 つの数値の間に配置した場合は、加算を実行します。	3
-	2 つの数値の間に配置した場合は、減算を実行します。	3

注意：Lingo では、演算に整数のみを使用すると、その結果も整数になります。同じ計算に整数と浮動小数点数を使用すると、その結果は浮動小数点数になります。JavaScript シンタックスでは、基本的にはすべての計算結果が浮動小数点数になります。

整数を別の整数で割ったときに結果が自然数にならない場合は、結果が四捨五入されます。例えば、4/3 の結果は 1 になります。JavaScript シンタックスでは、実際の浮動小数点値である 1.333 が返されます。

結果を四捨五入しないで値を計算させるには、式の 1 つまたは複数の値を float() で表記します。例えば、4/float(3) の結果は 1.333 になります。

比較演算子

比較演算子は、2 つの値を比較して結果の真偽を判断します。

演算子	意味	優先順位
==	（JavaScript シンタックスのみ）2 つのオペランドが等しくなっています。オペランドのデータ型が異なる場合、JavaScript シンタックスは、比較を行うためにオペランドを適切なデータ型に変換しようとします。	1
===	（JavaScript シンタックスのみ）2 つのオペランドが等しく、データ型も同じです。	1
!=	（JavaScript シンタックスのみ）2 つのオペランドが等しくありません。オペランドのデータ型が異なる場合、JavaScript シンタックスは、比較を行うためにオペランドを適切なデータ型に変換しようとします。	1
!==	（JavaScript シンタックスのみ）2 つのオペランドが等しくありません。また、データ型も異なります。	1
<>	（(Lingo のみ）2 つのオペランドが等しくありません。	1
<	左のオペランドが右のオペランドよりも小さくなっています。	1
<=	左のオペランドが右のオペランド以下です。	1
>	左のオペランドが右のオペランドよりも大きくなっています。	1
>=	左のオペランドが右のオペランド以上です。	1
=	（Lingo のみ）2 つのオペランドが等しくなっています。	1

代入演算子

代入演算子は、右のオペランドの値に基づいて左のオペランドに値を割り当てます。次の代入演算子のショートカットはすべて、JavaScript シンタックスにのみ適用されます。ただし、基本的な代入演算子である等号（=）は除きます。

演算子	意味	優先順位
=	等号	1
x += y	（JavaScript シンタックスのみ）x = x + y	1
x -= y	（JavaScript シンタックスのみ）x = x - y	1
x *= y	（JavaScript シンタックスのみ）x = x * y	1
x /= y	（JavaScript シンタックスのみ）x = x / y	1
x %= y	（JavaScript シンタックスのみ）x = x % y	1

論理演算子

論理演算子は、2 つの論理式の真偽をテストします。

Lingo および JavaScript シンタックスで論理演算子とストリング演算子を使用する場合は注意が必要です。例えば、JavaScript シンタックスでは、&& は、2 つの式が真であるかどうかを判断する論理演算子ですが、Lingo では、&& はストリング演算子で、2 つのストリングを連結し、2 つの式の間に半角スペースを挿入します。

演算子	効果	優先順位
and	（Lingo のみ）両方の式が共に真であるかどうかを判断します。	4
&&	（JavaScript シンタックスのみ）両方の式が共に真であるかどうかを判断します。	4
or	（Lingo のみ）式の両方または一方が真であるかどうかを判断します。	4

演算子	効果	優先順位
||	（JavaScript シンタックスのみ）式の両方または一方が真であるかどうかを判断します。	4
not	（Lingo のみ）式を否定します。	5
!	（JavaScript シンタックスのみ）式を否定します。	5

not 演算子（Lingo）または ! 演算子（JavaScript シンタックス）は、TRUE または FALSE の値を反転させるのに便利です。例えば、次のステートメントは、サウンドが現在オフの場合はオンにし、オンの場合はオフにします。

```
-- Lingo syntax
_sound.soundEnabled = not (_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

ストリング演算子

ストリング演算子は、ストリングを結合および定義します。

Lingo および JavaScript シンタックスで論理演算子とストリング演算子を使用する場合は注意が必要です。例えば、JavaScript シンタックスでは、&& は、2 つの式が真であるかどうかを判断する論理演算子ですが、Lingo では、&& はストリング演算子で、2 つのストリングを連結し、2 つの式の間に半角スペースを挿入します。

演算子	効果	優先順位
&	（Lingo のみ）2 つのストリングを連結します。	2
+	（JavaScript シンタックスのみ）2 つのストリングの値を連結し、2 つのオペランドを結合したものを 3 番目のストリングとして返します。	2
+=	（JavaScript シンタックスのみ）1 つのストリング変数と 1 つのストリング値を連結し、戻り値をストリング変数に割り当てます。	2
&&	（Lingo のみ）2 つのストリングを連結し、その間に半角スペースを挿入します。	2
"	ストリングの始まりと終わりを示します。	1

条件の構成

デフォルトでは、Director は常に、最初のスクリプトステートメントから最後のステートメントまで順番どおりに実行するか、またはどこかにジャンプを指示するステートメントがある場合には、そのとおりに実行します。

ステートメントが実行される順序は、ステートメントを配置した順序に影響します。例えば、ある計算値を求めるステートメントを記述する場合は、値を計算するステートメントを最初に記述します。

次の例では、最初のステートメントで 2 つの数の足し算を行い、2 番目のステートメントで、合計値を Answer という名前のフィールドキャストメンバに割り当てます。このフィールドキャストメンバはステージ上に配置されています。変数 x が定義されていないため、2 番目のステートメントを最初のステートメントの前に配置することはできません。

```
-- Lingo syntax
x = 2 + 2
member("Answer").text = string(x)

// JavaScript syntax
var x = 2 + 2;
member("Answer").text = x.toString();
```

Lingo および JavaScript シンタックスには両方とも、デフォルトの実行順序またはスクリプトステートメントを変更し、特定の条件に従ってアクションを実行するための規則が用意されています。例えば、スクリプトで次の操作を行うことができます。

- 論理条件が真の場合にステートメントセットを実行するか、論理条件が偽の場合に代替ステートメントを実行します。
- 式を評価して、式の値を特定の条件に合わせようとします。
- 特定の条件を満たすまで、ステートメントセットを繰り返し実行します。

論理条件のテスト

特定の条件が真または偽の場合にステートメントを実行するには、Lingo の場合は if...then...else 構造、JavaScript シンタックスの場合は if...else 構造を使用します。例えば、インターネットからのテキストのダウンロードが完了したかを確認し、完了していればテキストの書式設定を試みるといった if...then...else 構造または if...then 構造を作成することができます。これらの構造は、次のパターンを使用して論理条件をテストします。

- Lingo および JavaScript シンタックスともに、条件の真偽をチェックするステートメントは if. で始まります。
- Lingo の場合、条件が存在すると、then の後に続くステートメントが実行されます。JavaScript シンタックスでは、中括弧（{}）が Lingo の then の代わりをしており、この中括弧で、if ステートメント、else ステートメント、else if ステートメントをそれぞれ囲む必要があります。
- Lingo および JavaScript シンタックスの両方で、条件が存在しない場合は、else または else if を使用して スクリプトがハンドラ内の次のステートメントにスキップします。
- Lingo では、end if が if テストの終わりを指定します。JavaScript シンタックスでは、if テストは自動的に終了します。したがって、テストを明示的に終了するものはありません。

スクリプトのパフォーマンスを最適化するには、最も可能性の高い条件から先にテストします。

次のステートメントは、複数の条件をテストします。else if という項目は、前の条件が偽の場合に実行するもう 1 つのテストを指定します。

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
else if _mouse.mouseMember = member(2) then
    _movie.go("Nairobi")
else
    _player.alert("You're lost.")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
else if (_mouse.mouseMember = member(2)) {
    _movie.go("Nairobi");
}
else {
    _player.alert("You're lost.");
}
```

Lingo で if...then 構造を記述するときは、then に続くステートメントを、その then と同じ行に配置できます。また、then の後ろに改行を挿入し、新しい行にステートメントを配置することも可能です。改行を挿入する場合は、if...then 構造の最後に end if ステートメントも記述する必要があります。

JavaScript シンタックスで、if 構造を記述するときは、if に続くステートメントを、その if と同じ行に配置できます。また、if の後ろに改行を挿入し、新しい行にステートメントを配置することも可能です。

例えば、次のステートメントはいずれも同じ意味になります。

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then _movie.go("Cairo")

if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) { _movie.go("Cairo"); }

if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
```

if...then...else 構造および if...else 構造の使用に関するリファレンス情報については、if を参照してください。

式の評価と条件一致

case（Lingo）または switch...case（JavaScript シンタックス）は、複数の分岐構造を設定する場合に if...then...else 構造または if...then 構造を使用する代わりとなる簡潔な構造です。case 構造および switch...case 構造は、多くの場合、多数の if...then...else 構造または if...then 構造よりも効率的で、読みやすくなっています。

Lingo では、テストする条件は、case 構造の最初の行にある case の後に記述します。テスト条件と一致する式が見つかるまで、各行の比較が順番に実行されます。一致する式が見つかると、その式の直後にある Lingo が実行されます。

JavaScript シンタックスでは、テストする条件は、構造の最初の行にある switch の後に記述します。テストの各比較は、テストを含む各行の case の後ろにあります。各 case 比較を終了するには、オプションの break を使用します。break を挿入すると、switch 構造が終了し、その構造に続くステートメントが実行されます。break が省略されている場合は、次の case 比較が実行されます。

case または switch...case 構造では、テスト条件として比較を利用できます。

例えば、次の case 構造および switch...case 構造では、最後に押されたキーをテストし、それに応じた対応をします。

- A キーが押された場合は、ムービーはりんごというマーカに移動します。
- B または C キーが押された場合は、ムービーは指定されているトランジションを実行し、オレンジというマーカに移動します。
- 上記以外の文字キーが押された場合は、警告音が鳴ります。

```
-- Lingo syntax
case (_key.key) of
    "a" : _movie.go("Apple")
    "b", "c":
         _movie.puppetTransition(99)
         _movie.go("Oranges")
    otherwise: _sound.beep()
end case

// JavaScript syntax
switch (_key.key) {
    case "a" :
        _movie.go("Apple");
        break;
    case "b":
    case "c":
         _movie.puppetTransition(99);
        _movie.go("Oranges");
        break;
        default: _sound.beep()
}
```

注意：JavaScript シンタックスでは、case ステートメントごとに 1 つの比較しか実行できません。

case 構造の使用に関するリファレンス情報については、case を参照してください。

アクションの繰り返し

Lingo および JavaScript シンタックスでは、指定した回数だけ、または指定した条件が存在する限り、アクションを繰り返すことができます。

Lingo では、指定した回数だけアクションを繰り返すには、repeat with 構造を使用します。repeat with の後に、繰り返す回数を範囲で指定します。

JavaScript シンタックスでは、指定した回数だけアクションを繰り返すには、for 構造を使用します。for 構造は 3 つのパラメータを持ちます。最初のパラメータは、通常、カウンタ変数を初期化し、2 番目のパラメータは、ループを通るたびに評価する条件を指定し、3 番目のパラメータは、通常、カウンタ変数を更新または増分する際に使用します。

repeat with 構造と for 構造は、複数のオブジェクトに対して同じ操作を実行する場合に便利です。例えば、次のループは、スプライト 2 ～ 10 のインクを「Background Transparent」に設定します。

```
-- Lingo syntax
repeat with n = 2 to 10
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=2; n<=10; n++) {
    sprite(n).ink = 36;
}
```

次の例も同じようなアクションを実行しますが、スプライトの番号を大きいものから小さくしていきます。

```
-- Lingo syntax
repeat with n = 10 down to 2
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=10; n>=2; n--) {
    sprite(n).ink = 36;
}
```

Lingo では、指定した条件が存在する限り一連の命令を繰り返すには、repeat while 構造を使用します。

JavaScript シンタックス では、指定した条件が存在する限り一連の命令を繰り返すには、while 構造を使用します。

例えば、次のステートメントは、マウスボタンが押されている間は警告音を鳴らし続けます。

```
-- Lingo syntax
repeat while _mouse.mouseDown
    _sound.beep()
end repeat

// JavaScript syntax
while (_mouse.mouseDown) {
    _sound.beep();
}
```

Lingo および JavaScript シンタックスは両方とも、条件が真でなくなるまで、またはループ内のステートメントのいずれかがスクリプトをループの外に送信するまで、ループ内のステートメントをループし続けます。上記の例では、マウスボタンが離されると mouseDown という条件が真でなくなり、ループが終了します。

Lingo では、ループを終了するには、exit repeat ステートメントを使用します。

JavaScript シンタックスでは、ループを終了するには、break を使用します。また、条件が真でなくなると、ループは自動的に終了します。

例えば、次のステートメントは、マウスポインタがスプライト 1 の上に置かれている場合を除き、マウスボタンが押されている間は警告音を鳴らし続けます。ポインタをスプライト 1 に重ねると、ループは終了し、警告音は鳴り止みます。rollover() メソッドは、ポインタが指定のスプライト上に置かれているかどうかを示します。

```
-- Lingo syntax
repeat while _mouse.stillDown
    _sound.beep()
    if _movie.rollOver(1) then exit repeat
end repeat

// JavaScript syntax
while (_mouse.stillDown) {
    _sound.beep();
    if (_movie.rollOver(1)) {
        break;
    }
}
```

repeat while 構造および while 構造のリファレンス情報については、repeat while を参照してください。

イベント、メッセージ、およびハンドラ

強力かつ便利なスクリプトを作成するには、イベント、メッセージ、およびハンドラの概念と機能について理解することが重要です。イベントおよびメッセージが送受信される順序を知っておけば、特定のスクリプトまたはスクリプトの一部をどのタイミングで実行するかを決める際に便利です。また、発生するべきアクションが発生しない場合に、スクリプトをデバッグするときにも役立ちます。

ムービーの再生時に発生するアクションを次に示します。

- システムのアクションまたはユーザ定義のアクションのいずれかに応答して、イベントが発生します。
- これらのイベントに対応するメッセージがムービー内のスクリプトに送信されます。
- スクリプト内のハンドラには、特定のメッセージを受け取ったときに実行する命令が記述されています。

イベントの名前は、そのイベントによって生成されるメッセージの名前に対応します。また、イベントを処理するハンドラは、イベントとメッセージの両方に対応します。例えば、mouseDown という名前のイベントが発生すると、mouseDown という名前のメッセージが生成されます。生成されたメッセージはスクリプトに送信され、mouseDown という名前のハンドラによって処理されます。

イベント

ムービーの再生時に発生するイベントには 2 つのカテゴリがあります。

- **システムイベント**は、ユーザがムービーを操作しなくても発生し、事前に定義されているだけでなく、名前も付けられています。このイベントは、例えば、再生ヘッドがフレームに入ったとき、スプライトがクリックしたときなどに発生します。
- **ユーザ定義イベント** は、定義されたアクションに応答して発生します。例えば、スプライトのバックカラーが赤から青に変更されたとき、サウンドが 5 回再生されたときなどに発生するイベントを作成できます。

  ムービーの再生中、prepareFrame、beginSprite など、多数のシステムイベントが、事前定義された順番で自動的に発生します。また、mouseDown、mouseUp などのマウスイベントのように、ムービーの再生中に必ずしも自動的に発生するわけではないシステムイベントもあります。こうしたイベントは、ユーザがトリガしたときに発生します。

例えば、初めてムービーを開始するとき、最初に必ず発生するのが prepareMovie イベントです。そして、次に prepareFrame などが発生します。しかし、mouseDown イベントや mouseUp イベントについては、ユーザがムービーをクリックしてイベントをトリガしない限り発生しません。

次に、ムービーの再生中に必ず発生するシステムイベントを、その発生順序に従って示します。

ムービーの再生中、prepareFrame、beginSprite など、多数のシステムイベントが、事前定義された順番で自動的に発生します。また、mouseDown、mouseUp などのマウスイベントのように、ムービーの再生中に必ずしも自動的に発生するわけではないシステムイベントもあります。こうしたイベントは、ユーザがトリガしたときに発生します。

例えば、初めてムービーを開始するとき、最初に必ず発生するのが prepareMovie イベントです。そして、次に prepareFrame などが発生します。しかし、mouseDown イベントや mouseUp イベントについては、ユーザがムービーをクリックしてイベントをトリガしない限り発生しません。

次に、ムービーの再生中に必ず発生するシステムイベントを、その発生順序に従って示します。

初めてムービーの再生を開始すると、次の順序でイベントが発生する

1. prepareMovie
2. prepareFrame prepareFrame イベントの直後に、Director はサウンドを再生し、スプライトを描画し、トランジションまたはパレット効果を実行します。このイベントは、enterFrame イベントの前に発生します。フレームの描画の前に実行するスクリプトは、prepareFrame ハンドラに記述するとよいでしょう。
3. beginSprite このイベントは、再生ヘッドがスプライトスパンに入るときに発生します。
4. startMovie このイベントは、再生される最初のフレームで発生します。

ムービーがフレームを検出すると、次の順序でイベントが発生する

1. beginSprite このイベントは、そのフレームで新しいスプライトが開始される場合にのみ発生します。
2. stepFrame
3. enterFrame enterFrame 後、exitFrame の前に、テンポ設定、アイドルイベント、キーボードイベント、およびマウスイベントに必要な遅延時間が処理されます。
4. exitFrame
5. endSprite このイベントは、再生ヘッドがフレームのスプライトを出る場合にのみ発生します。

ムービーの再生が終了すると、次の順序でイベントが発生する

1. endSprite このイベントは、スプライトがその時点でムービー内にある場合にのみ発生します。
2. stopMovie

メッセージ

適切なスクリプトステートメントのセットを適切なタイミングで実行するには、Director ムービー内で起こっていることを認識し、特定のイベントに対してどのステートメントを実行するかを決定する必要があります。

スプライトのクリック、キー入力、ムービーの開始、フレームへの再生ヘッドの出入り、またはスクリプトが結果を返すなど、ムービー内で特定のイベントが発生すると、Director はメッセージを送信します。

メッセージをオブジェクトに送信する一般的な順序は次のとおりです。

1. メッセージはまず、イベントに関連するスプライトに割り当てられたビヘイビアーに送られます。スプライトに複数のビヘイビアーが割り当てられている場合は、スプライトに割り当てられた順番でビヘイビアーがメッセージに応答します。
2. メッセージは次に、スプライトに割り当てられたキャストメンバのスクリプトに送られます。
3. メッセージはその後、現在のフレームに割り当てられたビヘイビアーに送られます。

4 最後に、メッセージはムービースクリプトに送られます。

独自のメッセージ名を定義することはできますが、ムービー内で発生する一般的なイベントのほとんどには、内蔵メッセージ名が付いています。

Director 内蔵メッセージのリファレンス情報については、171 ページの「イベントとメッセージ」を参照してください。

カスタムメッセージの定義

内蔵メッセージ名を使用することに加えて、独自のメッセージとそれに対応するハンドラ名を定義することもできます。カスタムメッセージは、別のスクリプトやハンドラ、またはステートメント自体のハンドラを呼び出すことができます。呼び出されたハンドラの実行が終了すると、呼び出し元のハンドラが再開されます。

カスタムメッセージおよびハンドラの名前は、次の条件を満たす必要があります。

- 最初の文字はアルファベットにします。
- 使用できる文字は英数字のみで、特殊文字や句読点は使えません。
- 1 語のみで構成し、複数語で構成する場合はアンダースコア（_）で連結します。スペースは使用できません。
- 定義済みの Lingo または JavaScript エレメントとは違う名前にします。

定義済みの Lingo または JavaScript キーワードと同じ名前を、メッセージおよびハンドラに付けると混乱を招きます。こうした名前をメッセージまたはハンドラ名として使用して、意図的に Lingo または JavaScript エレメントの機能を置換または拡張することも可能ですが、これはよほど特別な状況を除いてお勧めできません。

機能の似ているハンドラが複数ある場合は、最初の数文字を同じにしておくと、編集／検索／ハンドラコマンドで表示するリストなどで連続して表示されるようになるので便利です。

ハンドラ

ハンドラは、特定のイベントおよびそれに続くメッセージに応答して実行される、スクリプト内のステートメントのセットです。Director には、内蔵イベントおよびメッセージが組み込まれていますが、処理するイベントとメッセージのペアごとに独自のカスタムハンドラを作成する必要があります。

ハンドラを配置する場所の決定

ハンドラはすべての種類のスクリプトに配置することができ、スクリプトには複数のハンドラを含めることができます。しかし、メンテナンスを簡単にするために、関連するハンドラを 1 つの場所にまとめることをお勧めします。

多くの一般的な状況では、次のガイドラインが役に立ちます。

- 特定のスプライトにハンドラを関連付け、特定のスプライト上のアクションに応答してハンドラが実行されるよう設定するには、スプライトに割り当てられたビヘイビアーにハンドラを配置します。
- ムービーが特定のフレームにあるときだけ利用できるハンドラを設定するには、フレームに割り当てられたビヘイビアーにハンドラを配置します。

  例えば、再生ヘッドがフレーム内にあるときに、クリックする場所に関係なくマウスクリックに応答するハンドラを設定するには、スプライトビヘイビアーではなくフレームビヘイビアーに、mouseDown または mouseUp ハンドラを配置します。
- ムービーのどこからでもイベントについてのメッセージに応答して実行されるハンドラを設定するには、ムービースクリプトにハンドラを配置します。
- スプライトに関係なく、キャストメンバに影響するイベントに応答して実行されるハンドラを設定するには、キャストメンバスクリプトにハンドラを配置します。

ハンドラがメッセージを受け取るタイミングの決定

メッセージがスクリプトに送信されたら、ハンドラが一定の順序でチェックされます。

1. まず、メッセージ送信元のオブジェクトにハンドラが存在するかどうかが確認されます。ハンドラが見つかった場合、メッセージは途中で遮られ、ハンドラ内のスクリプトが実行されます。
2. ハンドラが見つからなかった場合は、キャストメンバが昇順にチェックされ、メッセージのハンドラを含む関連ムービースクリプトがないかどうかが確認されます。ハンドラが見つかった場合、メッセージは途中で遮られ、ハンドラ内のスクリプトが実行されます。
3. ハンドラが見つからなかった場合、次は、フレームスクリプトにメッセージのハンドラが含まれていないかどうかが確認されます。ハンドラが見つかった場合、メッセージは途中で遮られ、ハンドラ内のスクリプトが実行されます。
4. ハンドラが見つからなかった場合は、スプライトが昇順にチェックされ、メッセージのハンドラを含むスプライトに関連付けられているスクリプトがないかどうかが確認されます。ハンドラが見つかった場合、メッセージは途中で遮られ、ハンドラ内のスクリプトが実行されます。

あるハンドラがメッセージを受け取った後、そのメッセージが他の場所へ自動的に送られることはありません。ただし、Lingo では、pass() メソッドを使用すると、このデフォルト設定を無効にして、メッセージを他のオブジェクトに渡すことができます。

メッセージを可能なすべての場所に渡しても対応するハンドラが見つからない場合は、そのメッセージは無視されます。

メッセージが送られるオブジェクトの正確な順序は、メッセージによって異なります。特定のメッセージが送られるオブジェクトの順序については、171 ページの「イベントとメッセージ」の各メッセージの項目を参照してください。

パラメータによるハンドラへの値の受け渡し

値のパラメータを使用することで、ムービー内でハンドラを呼び出す場所やタイミングに関係なく、必要な値を正確に、特定のタイミングでハンドラに渡すことができます。状況によって、パラメータが省略可能な場合と必要な場合があります。

- Lingo では、パラメータをハンドラ名の後ろに置きます。
- JavaScript シンタックスでは、パラメータを括弧で囲んでからハンドラ名の後ろに置きます。

複数のパラメータを区切るには、カンマを使用します。

ハンドラを呼び出すときは、そのハンドラが使用するパラメータの値を指定する必要があります。数字、値が割り当てられた変数、ストリングなど、様々なタイプの値を使用できます。呼び出し側のステートメントの値は、ハンドラのパラメータと同じ順序で設定し、括弧で囲む必要があります。

次の例では、変数割り当て mySum が addThem メソッドを呼び出しています。このメソッドには 2 つの値、2 と 4 が渡されています。addThem ハンドラは、パラメータのプレイスホルダーである a と b を、渡された 2 つの値に置き換えて、その結果をローカル変数 c に格納し、キーワード return を使用して元のメソッドに返します。そして、その結果は、mySum に再度割り当てられます。

2 はパラメータリストの最初にあるため、ハンドラ内の a に代入されます。同様に、4 はパラメータリストの 2 番目にあるため、ハンドラ内の b に代入されます。

```
-- Lingo syntax
mySum = addThem(2, 4) -- calling statement

on addThem a, b -- handler
    c = a + b
    return c -- returns the result to the calling statement
end

// JavaScript syntax
var mySum = addThem(2, 4); // calling statement

function addThem(a, b) { // handler
    c = a + b;
    return c; // returns the result to the calling statement
}
```

Lingo では、オブジェクトからカスタムメソッドを呼び出すとき、メモリ内の Script オブジェクトへの参照が、常に最初の暗黙パラメータとしてそのメソッドのハンドラに渡されます。つまり、ハンドラで Script オブジェクト参照を説明する必要があります。

例えば、1 つの整数をパラメータとして取る jump() という名前のカスタムスプライトメソッドを記述し、そのメソッドをビヘイビアーに置いたとします。jump() を Sprite オブジェクト参照から呼び出すとき、ハンドラには 1 つの整数だけでなく、Script オブジェクト参照を表すパラメータも含める必要があります。この場合、暗黙パラメータは、キーワード me によって表され、すべての項目が動作するようになります。

```
-- Lingo syntax
myHeight = sprite(2).jump(5)

on jump(me,a)
    return a + 15 -- this handler works correctly, and returns 20
end

on jump(a)
    return a + 15 -- this handler does not work correctly, and returns 0
end
```

値に式を使用することもできます。例えば、次のステートメントは、a に 3+6 を、b に 8>2（TRUE を示す 1）を代入して、10 を返します。

```
-- Lingo syntax
mySum = addThem(3+6, 8>2)
```

Lingo では、各ハンドラは on という語で始まり、その後にハンドラが応答するメッセージ、つまりハンドラ名が続きます。ハンドラの最終行は end という語になります。end の後にハンドラ名を繰り返すこともできますが、これは省略してもかまいません。

JavaScript シンタックス では、各ハンドラは function という語で始まり、その後にハンドラが応答するメッセージ、つまりハンドラ名が続きます。JavaScript シンタックス関数と同様、ハンドラを構成するステートメントは角括弧で囲まれています。

ハンドラから結果を返す

ハンドラで条件や実行したアクションの結果をレポートしたい場合がよくあります。

❖ キーワード return を使用すると、ハンドラに条件やアクションの結果をレポートさせることができます。例えば、次の findColor ハンドラはスプライト 1 の現在のカラーを返します。

```
-- Lingo syntax
on findColor
    return sprite(1).foreColor
end

// JavaScript syntax
function findColor() {
    return(sprite(1).foreColor);
}
```

キーワード return だけを使用して、値を返さずに現在のハンドラを終了することもできます。例えば、次の jump ハンドラは、aVal パラメータ が 5 の場合は何も返していません。それ以外の場合は値を返します。

```
-- Lingo syntax
on jump(aVal)
    if aVal = 5 then return

    aVal = aVal + 10
    return aVal
end
// JavaScript syntax
function jump(aVal) {
    if(aVal == 5) {
        return;
    }
    else {
        aVal = aVal + 10;
        return(aVal);
    }
}
```

結果を返すハンドラを定義するときに、別のハンドラから呼び出す場合には、ハンドラの直後に括弧を記述する必要があります。例えば、put(findColor()) というステートメントは、on findColor ハンドラを呼び出し、その結果をメッセージウィンドウに表示します。

リニアリストおよびプロパティリスト

スクリプトで、変数のセットに割り当てられている名前や値などのデータのリストを、追跡および更新するとよいでしょう。Lingo と JavaScript シンタックスは両方とも、リニアリストとプロパティリストにアクセスできます。リニアリストでは、リスト内の各エレメントが 1 つの値です。プロパティリストでは、リスト内の各エレメントに 2 つの値が含まれています。1 つはプロパティ名で、もう 1 つはそのプロパティに関連付けられている値です。

Lingo と JavaScript シンタックスの両方がリニアリストとプロパティリストにアクセスできるので、コード内の値が Lingo スクリプトおよび JavaScript シンタックススクリプトで共有されている場合は、リニアリストとプロパティリストを使用することをお勧めします。

コード内の値が JavaScript シンタックススクリプトでのみ使用されている場合は、JavaScript Array オブジェクトを使用してデータリストを操作することをお勧めします。配列の使用について詳しくは、37 ページの「JavaScript シンタックス配列」を参照してください。

リニアリストの作成

リニアリストを作成するには、次のいずれかを実行します。

- Lingo の場合は、トップレベル list() 関数またはリスト演算子（[ ]）のいずれかを使用します。リスト内のアイテムはカンマで区切ります。

- JavaScript シンタックスの場合は、トップレベル list() 関数を使用します。リスト内のアイテムはカンマで区切ります。

リニアリストのインデックスは常に 1 から始まります。

トップレベル list() 関数を使用するとき、リストのエレメントは関数のパラメータとして指定します。この関数は、角括弧を入力できないキーボードを使用する場合に役立ちます。

次のステートメントはすべて、3 つの名前を持つリニアリストを作成し、そのリストを変数に割り当てます。

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- using the Lingo list operator
workerList = list("Bruno", "Heather", "Carlos") -- using list()

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // using list()
```

空のリニアリストを作成することもできます。次のステートメントは、空のリニアリストを作成します。

```
-- Lingo syntax
workerList = [] -- using the Lingo list operator
workerList = list() -- using list() with no parameters

// JavaScript syntax
var workerList = list(); // using list() with no parameters
```

プロパティリストの作成

プロパティリストを作成するには、次のいずれかを実行します。

- Lingo の場合は、トップレベル propList() 関数またはリスト演算子（[:]）のいずれかを使用します。リスト演算子を使用してプロパティリストを作成するときは、コロンまたはカンマのいずれかを使用して名前と値のエレメントの指定し、カンマを使用してリスト内のエレメントを区切ります。
- JavaScript シンタックスの場合は、トップレベル propList() 関数を使用し、カンマを挿入をして、名前と値のエレメントを指定してリスト内のアイテムを区切ります。

トップレベル propList() 関数を使用するとき、プロパティリストのエレメントは関数のパラメータとして指定します。この関数は、角括弧を入力できないキーボードを使用する場合に役立ちます。

プロパティは、指定されたプロパティリスト内で重複して使用することができます。

次のステートメントはすべて、4 つのプロパティ名（left、top、right、bottom）と、それに対応する値を持つプロパティリストを作成します。

```
-- Lingo syntax
sprite1Loc = [#left:100, #top:150, #right:300, #bottom:350]
sprite1Loc = ["left",400, "top",550, "right",500, "bottom",750]
sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750)

// JavaScript syntax
var sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750);
```

空のプロパティリストを作成することもできます。次のステートメントは、空のプロパティリストを作成します。

```
-- Lingo syntax
sprite1Loc = [:] -- using the Lingo property list operator
sprite1Loc = propList() -- using propList() with no parameters

// JavaScript syntax
var sprite1Loc = propList(); // using propList() with no parameters
```

リスト内のアイテムの設定と取得

リスト内の各アイテムを設定および取得できます。リニアリストとプロパティリストではシンタックスが異なります。

リニアリストの値の設定

次のいずれかの操作を行います。

- 等号演算子（=）を使用します。
- setAt() メソッドを使用します。

次のステートメントは、Heather という値が含まれるリニアリスト workerList を定義し、そのリスト内の 2 番目の値として Carlos を追加しています。

```
-- Lingo syntax
workerList = ["Heather"] -- define a linear list
workerList[2] = "Carlos" -- set the second value using the equal operator
workerList.setAt(2, "Carlos") -- set the second value using setAt()

// JavaScript syntax
var workerList = list("Heather"); // define a linear list
workerList[2] = "Carlos"; // set the second value using the equal operator
workerList.setAt(2, "Carlos"); // set the second value using setAt()
```

リニアリストの値の取得

1. リスト変数を使用し、その後にリスト内の値の位置を示す番号を指定します。番号は角括弧で囲んでください。
2. getAt() メソッドを使用します。

次のステートメントは、リニアリスト workerList を作成し、そのリストの 2 番目の値を変数 name2 に割り当てています。

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
name2 = workerList[2] -- use bracketed access to retrieve "Heather"
name2 = workerList.getAt(2) -- use getAt() to retrieve "Heather"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos");
var name2 = workerList[2] // use bracketed access to retrieve "Heather"
var name2 = workerList.getAt(2) // use getAt() to retrieve "Heather"
```

プロパティリストの値の設定

次のいずれかの操作を行います。

- 等号演算子（=）を使用します。
- setaProp() メソッドを使用します（Lingo のみ）。
- ドットシンタックスを使用します。

  次の Lingo ステートメントは等号演算子を使用して、sushi を、プロパティ Bruno に関連付けられている新しい値として設定します。

  ```
  -- Lingo syntax
  foodList = [:] -- define an empty property list
  foodList[#Bruno] = "sushi" -- associate sushi with Bruno
  ```

  次の Lingo ステートメントは setaprop() を使用して、sushi を、プロパティ Bruno に関連付けられている新しい値として設定します。

```
-- Lingo syntax
foodList = [:] -- define an empty property list
foodList.setaProp(#Bruno, "sushi") -- use setaProp()

// JavaScript syntax
foodList = propList() -- define an empty property list
foodList.setaProp("Bruno", "sushi") -- use setaProp()
```

次のステートメントはドットシンタックスを使用して、Bruno に関連付けられている値を sushi から teriyaki に変更します。

```
-- Lingo syntax
foodList = [#Bruno:"sushi"] -- define a property list
trace(foodList) -- displays [#Bruno: "sushi"]
foodList.Bruno = "teriyaki" -- use dot syntax to set the value of Bruno
trace(foodList) -- displays [#Bruno: "teriyaki"]

// JavaScript syntax
var foodList = propList("Bruno", "sushi"); // define a property list
trace(foodList); // displays ["Bruno": "sushi"]
foodList.Bruno = "teriyaki" // use dot syntax to set the value of Bruno
trace(foodList) -- displays [#Bruno: "teriyaki"]
```

プロパティリストの値の取得

次のいずれかの操作を行います。

- リスト変数を使用し、その後に値に関連付けられたプロパティの名前を指定します。プロパティは角括弧で囲んでください。
- getaProp() メソッドまたは getPropAt() メソッドを使用します。
- ドットシンタックスを使用します。

次のステートメントは、ブラケットアクセスを使用して、プロパティ breakfast およびプロパティ lunch に関連付けられている値を取得します。

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList[#breakfast]) -- displays "Waffles"
trace(foodList[#lunch]) -- displays "Tofu Burger"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList["breakfast"]); // displays Waffles
trace(foodList["lunch"]); // displays Tofu Burger
```

次のステートメントは、getaProp() を使用して、プロパティ breakfast に関連付けられている値を取得し、getPropAt() を使用してリスト内の 2 番目のインデックスにあるプロパティを取得します。

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.getaProp(#breakfast)) -- displays "Waffles"
trace(foodList.getPropAt(2)) -- displays #lunch

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.getaProp("breakfast")) // displays Waffles
trace(foodList.getPropAt(2)) // displays lunch
```

次のステートメントはドットシンタックスを使用して、プロパティリストのプロパティに関連付けられている値にアクセスします。

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.breakfast) -- displays "Waffles"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.lunch); // displays Tofu Burger
```

リスト内のアイテムの確認

リストの特性やリスト内にあるアイテムの数を調べるには、次のメソッドを使用します。

- リストの内容を表示するには、put() 関数または trace() 関数を使用して、リストが含まれている変数をパラメータとして渡します。
- リスト内のアイテム数を調べるには、count() メソッド（Lingo のみ）または count プロパティを使用します。
- リストの種類を調べるには、ilk() メソッドを使用します。
- リスト内の最大値を調べるには、max() メソッドを使用します。
- リスト内の最小値を調べるには、min() 関数を使用します。
- 特定のプロパティの位置を調べるには、findPos、findPosNear、または getOne コマンドを使用します。

次のステートメントは、count() および count を使用してリスト内のアイテム数を表示します。

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.count()) -- displays 3
trace(workerList.count) -- displays 3

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.count); // displays 3
```

次のステートメントは、ilk() を使用してリストの種類を確認します。

```
-- Lingo syntax
x = ["1", "2", "3"]
trace(x.ilk()) // returns #list

// JavaScript syntax
var x = list("1", "2", "3");
trace(x.ilk()) // returns #list
```

次のステートメントは、max() および min() を使用して、リスト内の最大値および最小値を確認します。

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.max()) -- displays "Heather"
trace(workerList.min()) -- displays "Bruno"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.max()); // displays Heather
trace(workerList.min()); // displays Bruno
```

次のステートメントは、findPos を使用して、プロパティリスト内の特定のプロパティのインデックス位置を取得します。

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.findPos(#lunch)) -- displays 2

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.findPos("breakfast")); // displays 1
```

リスト内のアイテムの追加と削除

リストのアイテムを追加または削除するには、次のメソッドを使用します。

- リストの最後にアイテムを追加するには、append() メソッドを使用します。
- 並べ替えたリストの適切な位置にアイテムを追加するには、add() メソッドまたは addProp() メソッドを使用します。
- リニアリスト内の特定の位置にアイテムを追加するには、addAt() メソッドを使用します。
- プロパティリスト内の特定の位置にアイテムを追加するには、addProp() メソッドを使用します。
- リストからアイテムを削除するには、deleteAt() メソッド、deleteOne() メソッド、または deleteProp() メソッドを使用します。
- リストのアイテムを置き換えるには、setAt() メソッドまたは setaProp() メソッドを使用します。

次のステートメントは、append() を使用してリストの最後にアイテムを追加します。

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
workerList.append("David")
trace(workerList) -- displays ["Bruno", "Heather", "Carlos", "David"]

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
workerList.append("David");
trace(workerList); // displays ["Bruno", "Heather", "Carlos", "David"]
```

次のステートメントは、addProp() を使用して、プロパティおよび関連付けられている値をプロパティリストに追加します。

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
foodList.addProp(#dinner, "Spaghetti") -- adds [#dinner: "Spaghetti"]

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
foodList.addProp("dinner", "Spaghetti"); // adds ["dinner": "Spaghetti"]
```

リストを明示的に消去する必要はありません。リストを参照する変数がなくなると、そのリストは自動的に消去されます。リスト以外のオブジェクトは、そのオブジェクトを参照する変数を、Lingo の場合は VOID に、JavaScript シンタックスの場合は null に設定して、明示的に削除する必要があります。

リストのコピー

リストをある変数に割り当て、その変数を別の変数に割り当てても、リストの独立したコピーは作成されません。例えば、次の最初のステートメントは、2 つの大陸名を含むリストを作成し、そのリストを変数 landList に割り当てています。2 番目のステートメントは、同じリストを新しい変数 continentList に割り当てます。3 番目のステートメントでは、Australia を

landList に追加することで、Australia がリスト continentList に自動的に追加されています。これは、いずれの変数もメモリ内の同じ List オブジェクトを参照するためです。JavaScript シンタックスの配列を使用すると、これと同じビヘイビアーが発生します。

```
-- Lingo syntax
landList = ["Asia", "Africa"]
continentList = landList
landList.add("Australia") -- this also adds "Australia" to continentList
```

他のリストから独立したリストのコピーの作成

❖ duplicate() メソッドを使用します。

例えば、次のステートメントは、リストを作成してから、そのリストから独立したコピーを作成します。

```
-- Lingo syntax
oldList = ["a", "b", "c"]
newList = oldList.duplicate() -- makes an independent copy of oldList

// JavaScript syntax
var oldList = list("a", "b", "c");
var newList = oldList.duplicate(); // makes an independent copy of oldList
```

newList を作成した後で oldList または newList を編集しても、もう一方には変更は反映されません。

リストの並べ替え

リストは英数字順にソートされています。その際、数字はストリングより優先されます。ストリングは長さに関係なく、最初の文字によって順番に並べ替えられます。並べ替えられたリストは、並べ替えられていないリストより少しだけ速く処理されます。

リニアリストはリスト内の値に従ってソートされています。プロパティリストはリストまたは配列内のプロパティ名に従ってソートされています。

リニアリストまたはプロパティリストの値がソートされた後、値が追加または削除されても、そのソートは維持されたままです。

リストの並べ替え

❖ sort() メソッドを使用します。

例えば、次のステートメントは、ソートされていないアルファベットのリストをアルファベット順に並べ替えます。

```
-- Lingo syntax
oldList = ["d", "a", "c", "b"]
oldList.sort() -- results in ["a", "b", "c", "d"]

// JavaScript syntax
var oldList = list("d", "a", "c", "b");
oldList.sort(); // results in ["a", "b", "c", "d"]
```

多次元リストの作成

多次元リストを作成することもできます。多次元リストでは、複数のリストの値を同時に操作できます。

次の例では、最初の 2 つのステートメントで、list1 と list2 の 2 つのリニアリストを作成しています。3 番目のステートメントでは、多次元リストを作成し、そのリストを mdList に割り当てています。多次元リストの値にアクセスするために、4 番目と 5 番目のステートメントは角括弧を使用して、リスト内の値にアクセスします。最初の角括弧は指定のリストに、2 番目の角括弧はリスト内の指定のインデックスにある値にアクセスできるようにします。

```
-- Lingo syntax
list1 = list(5,10)
list2 = list(15,20)
mdList = list(list1, list2)
trace(mdList[1][2]) -- displays 10
trace(mdList[2][1]) -- displays 15

// JavaScript syntax
var list1 = list(5,10);
var list2 = list(15,20);
var mdList = list(list1, list2);
trace(mdList[1][2]); // displays 10
trace(mdList[2][1]); // displays 15
```

JavaScript シンタックス配列

JavaScript シンタックス配列は、配列の各エレメントが 1 つの値になっている点で Lingo のリニアリストと似ています。JavaScript シンタックス配列と Lingo のリニアリストで違うのは、配列のインデックスが 0 で始まるという点です。

JavaScript シンタックス配列は、Array オブジェクトを使用して作成します。配列を作成するには、角括弧（[ ]）または Array コンストラクタのいずれかを使用します。次の 2 つのステートメントは、2 つの値を持つ配列を作成しています。

```
// JavaScript syntax
var myArray = [10, 15]; // using square brackets
var myArray = new Array(10, 15); // using the Array constructor
```

空の配列を作成することもできます。次の 2 つのステートメントは、空の配列を作成します。

```
// JavaScript syntax
var myArray = [];
var myArray = new Array();
```

注意：『Director スクリプトリファレンス』には、JavaScript シンタックスの Array オブジェクトのリファレンスがすべて含まれているわけではありません。Array オブジェクトの使用について詳しくは、市販の関連リソースを参照してください。

配列内のアイテムの確認

配列の特性や配列内にあるアイテムの数を調べるには、次のメソッドを使用します。

- リストの内容を表示するには、put() 関数または trace() 関数を使用して、リストが含まれている変数をパラメータとして渡します。
- 配列内のアイテム数を調べるには、Array オブジェクトの length プロパティを使用します。
- 配列の種類を調べるには、constructor プロパティを使用します。

次の例では、length プロパティを使用して配列内のアイテム数を確認し、constructor プロパティを使用してオブジェクトの種類を返しています。

```
// JavaScript syntax
var x = ["1", "2", "3"];
trace(x.length) // displays 3
trace(x.constructor == Array) // displays true
```

配列内のアイテムの追加と削除

配列のアイテムを追加または削除するには、次のメソッドを使用します。

- 配列の最後にアイテムを追加するには、Array オブジェクトの push() メソッドを使用します。

- ソートされた配列内の適切な位置にアイテムを追加するには、Array オブジェクトの splice() メソッドを使用します。
- 配列内の特定の位置にアイテムを追加するには、Array オブジェクトの splice() メソッドを使用します。
- 配列からアイテムを削除するには、Array オブジェクトの splice() メソッドを使用します。
- 配列内のアイテムを置き換えるには、Array オブジェクトの splice() メソッドを使用します。

次の例では、Array オブジェクトの splice() メソッドを使用して、配列内でアイテムを追加、削除、および置換しています。

```
// JavaScript syntax
var myArray = new Array("1", "2");
trace(myArray); displays 1,2

myArray.push("5"); // adds the value "5" to the end of myArray
trace(myArray); // displays 1,2,5

myArray.splice(3, 0, "4"); // adds the value "4" after the value "5"
trace(myArray); // displays 1,2,5,4

myArray.sort(); // sort myArray
trace(myArray); // displays 1,2,4,5

myArray.splice(2, 0, "3");
trace(myArray); // displays 1,2,3,4,5

myArray.splice(3, 2); // delete two values at index positions 3 and 4
trace(myArray); // displays 1,2,3

myArray.splice(2, 1, "7"); // replaces one value at index position 2 with "7"
trace(myArray); displays 1,2,7
```

配列のコピー

配列をある変数に割り当て、その変数を別の変数に割り当てても、配列の独立したコピーは作成されません。

例えば、次の最初のステートメントは、2 つの大陸名を含む配列を作成し、その配列を変数 landList に割り当てています。2 番目のステートメントは、同じリストを新しい変数 continentList に割り当てます。3 番目のステートメントでは、Australia を landList に追加することで、Australia が配列 continentList に自動的に追加されています。これは、いずれの変数もメモリ内の同じ Array オブジェクトを参照するためです。

```
// JavaScript syntax
var landArray = new Array("Asia", "Africa");
var continentArray = landArray;
landArray.push("Australia"); // this also adds "Australia" to continentList
```

他の配列から独立した配列のコピーの作成

❖ Array オブジェクトの slice() メソッドを使用します。

例えば、次のステートメントは、配列を作成してから、slice() を使用してその配列から独立したコピーを作成します。

```
// JavaScript syntax
var oldArray = ["a", "b", "c"];
var newArray = oldArray.slice(); // makes an independent copy of oldArray
```

newArray を作成した後で oldArray または newArray を編集しても、もう一方には変更は反映されません。

配列の並べ替え

配列は英数字順にソートされています。その際、数字はストリングより優先されます。ストリングは長さに関係なく、最初の文字によって順番に並べ替えられます。

配列の並べ替え

❖ Array オブジェクトの sort() メソッドを使用します。

次のステートメントは、ソートされていないアルファベットの配列を並べ替えます。

```
// JavaScript syntax
var oldArray = ["d", "a", "c", "b"];
oldArray.sort(); // results in a, b, c, d
```

次のステートメントは、ソートされていないアルファベットと数字の配列を並べ替えます。

```
// JavaScript syntax
var oldArray = [6, "f", 3, "b"];
oldArray.sort(); // results in 3, 6, b, f
```

配列を並べ替えると、配列がソートされて新しい配列が表示されます。

多次元配列の作成

多次元配列を作成することもできます。多次元配列では、複数の配列の値を同時に操作できます。

次の例では、最初の 2 つのステートメントで、array1 と array2 の 2 つの配列を作成しています。3 番目のステートメントでは、多次元配列を作成し、その配列を mdArray に割り当てています。多次元配列の値にアクセスするために、4 番目と 5 番目のステートメントは角括弧を使用して、配列内の値にアクセスします。最初の角括弧は指定の配列に、2 番目の角括弧は配列内の指定のインデックスにある値にアクセスできるようにします。

```
// JavaScript syntax
var array1 = new Array(5,10);
var array2 = [15,20];
var mdArray = new Array(array1, array2);
trace(mdArray[0][1]); // displays 10
trace(mdArray[1][0]); // displays 15
```

第 3 章：Director でのスクリプトの記述

Director® 11 のスクリプトでは、ムービーで使用でき、スクリプトを記述しなければできないようなあらゆる種類の機能がサポートされています。スクリプトを記述していると、Director ムービーで複雑なインタラクティビティをサポートするために、どんどん高度なスクリプトが必要になってくる場合があります。ここでは、中級および上級のスクリプトの概念とテクニックについて、Director のオブジェクト指向スクリプティングに関する情報も含めて説明します。

Director でスクリプトを記述するのが初めての方は、ここの説明に加えて 4 ページの「Director スクリプトに関する重要事項」もお読みください。

Lingo と JavaScript シンタックスの間での選択

Lingo と JavaScript シンタックスのどちらでも、使用できるオブジェクト、イベントおよびスクリプト API は同じです。このため、スクリプトの記述にどちらの言語を使用するかは大きな問題ではありません。自分がよく知っていて使いやすい方の言語を選択してください。

Director のスクリプト言語で個々のオブジェクトとイベントモデルがどのように扱われるかを理解するには、以下の点について考慮する必要があります。

- 一般に、使用しているスクリプト言語は、つまり Lingo や JavaScripts などのシンタックスは、特定のオブジェクトとイベントへのアクセスを提供するためにそのオブジェクトとイベントのモデルに従います。
- JavaScript は、ブラウザのオブジェクトとイベントへのアクセスを提供するために、Web ブラウザのオブジェクトとイベントのモデルに従った ECMAScript 標準の実装です。
- ActionScript は、Adobe® Flash® のオブジェクトとイベントへのアクセスを提供するために、Flash のオブジェクトとイベントのモデルに従った ECMAScript 標準の実装です。
- JavaScript シンタックスの Director での実装は、Director のオブジェクトとイベントへのアクセスを提供するために、Director のオブジェクトとイベントのモデルに従った ECMAScript の実装です。
- Lingo は、Director のオブジェクトとイベントへのアクセスを提供するために、Director のオブジェクトとイベントのモデルに従ったカスタムシンタックスです。

Lingo と JavaScript のシンタックスは、Director のオブジェクトとイベントの同じモデルにアクセスするために使用できる 2 つの言語です。どちらかの言語で記述されたスクリプトは、他方の言語で記述されたスクリプトと同じ機能を持ちます。

そのため、どちらかの言語でスクリプト API にアクセスする方法がわかれば、基本的には他方の言語でそれにアクセスする方法もわかったことになります。例えば、JavaScript シンタックスのコードは、シンボル、リニアリスト、プロパティリストなどの Lingo のデータタイプにアクセスでき、Lingo の親スクリプトとビヘイビアーを作成して呼び出したり、Xtra 拡張機能を作成して呼び出したり、Lingo スクリプトのチャンク式を使用したりすることもできます。また、JavaScript シンタックスと Lingo スクリプトの両方を単一のムービー内で使用することもできます。ただし、単一のスクリプトキャストメンバには、どちらか片方のシンタックスしか入れることができません。

Lingo と JavaScript のシンタックスには、主に次の 2 つの大きな違いがあります。

- それぞれが、その言語に固有のいくつかの用語とシンタックス規則を持ちます。例えば、イベントハンドラのシンタックスは、Lingo と JavaScript とでは次のように異なります。

```
-- Lingo syntax
on mouseDown
    ...
end

// JavaScript syntax
function mouseDown() {
    ...
}
```

各言語で使用される用語とシンタックス規則について詳しくは、5 ページの「スクリプト用語集」および 7 ページの「スクリプトシンタックス」を参照してください。

- スクリプト API の中には、アクセス方法が言語ごとに若干異なるものもあります。例えば、テキストキャストメンバの最初の段落の 2 つ目の単語にアクセスするために使用するシンタックスは、次のように異なります。

```
-- Lingo syntax
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

ドットシンタックス形式でのスクリプト記述

Lingo と JavaScript シンタックスのいずれでスクリプトを記述しても、スクリプトの記述にドットシンタックス形式を使用することになります。ドットシンタックスは、オブジェクトに関連付けられているプロパティまたはメソッドにアクセスするために使用します。ドットシンタックスのステートメントは、オブジェクトへの参照で始め、その後にピリオド（ドット）を続け、さらに指定するプロパティ、メソッドまたはテキストチャンクの名前を続けます。ステートメント内の各ドットは、基本的に、オブジェクト階層内で上位にある汎用的なレベルから、オブジェクト階層内で下位にある具体的なレベルへの移動を表します。

例えば、次のステートメントは、まず「News Stories」というキャストライブラリへの参照を作成し、次にドットシンタックスを使用して、そのキャストライブラリ内のキャストメンバの数にアクセスしています。

テキストの一部を特定するには、テキスト内のより具体的なアイテムを参照する項目をドットの後に記述します。例えば、下の 1 つ目のステートメントは、「News Items」というテキストキャストメンバの最初の段落を指しています。下の 2 つ目のステートメントは、最初の段落の 2 つ目の単語を指しています。

```
-- Lingo syntax
member("News Items").paragraph[1]
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1);
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

データまたは特定のキャストメンバの種類にアクセスするカスケードプロパティを扱うような一部のオブジェクトでは、上の 2 つのステートメントで示したような、通常の JavaScript シンタックスを使用したプロパティへのアクセスはサポートされていません。このため、JavaScript シンタックスでカスケードプロパティにアクセスするには getPropRef() メソッドおよび getProp() メソッドを使用してください。

JavaScript シンタックスのこの例外については、次に挙げるようにいくつか注意すべき点があります。

- JavaScript シンタックスを使用して 3D オブジェクト、テキストキャストメンバ、フィールドキャストメンバおよび XML Parser の Xtra 拡張機能にアクセスするには、この方法を使用してください。
- JavaScript シンタックスを使用して前述のオブジェクトのいずれかへの参照またはそのプロパティを保存するには、getPropRef() メソッドを使用してください。

- JavaScript シンタックスを使用して前述のオブジェクトのいずれかのプロパティの値またはそのプロパティを取得するには、getProp() メソッドを使用してください。
- JavaScript シンタックスで 3D オブジェクトとそのプロパティにアクセスするには、完全修飾名を使用してください。例えば、Lingo では shaderList[1] プロパティのショートカットとして shader プロパティを使用できますが、JavaScript シンタックスでは必ず shaderList[1] プロパティを使用してください。

Director オブジェクトについて

オブジェクトは、基本的な用語としては、データに名前を付けて論理的にグループ分けしたもので、そのデータに作用するメソッドも持つことができます。Director のこのリリースでは、スクリプト API がグループ分けしてオブジェクトに入れられ、オブジェクトを通じてアクセスされるようになりました。各オブジェクトが、特定の一連の名前付きデータと何らかの機能へのアクセスを提供します。例えば、スプライトオブジェクトはスプライトのデータと機能へのアクセスを提供し、ムービーオブジェクトはムービーのデータと機能へのアクセスを提供します。

Director で使用されるオブジェクトには、コアオブジェクト、メディアタイプ、スクリプトオブジェクト、3D オブジェクトの 4 つのカテゴリがあります。追加する機能の種類、および機能が追加されるムービーの部分に応じて、これらのカテゴリの 1 つ以上のオブジェクトを使用します。

コアオブジェクト

このカテゴリのオブジェクトは、Director で使用できる中心機能へのアクセスを提供します。Director プレーヤエンジン、ムービーウィンドウ、スプライト、サウンドなどへのアクセスを提供するオブジェクトです。これらのオブジェクトは基本レイヤーを表しており、これらを通じてすべての API および他のオブジェクトカテゴリのオブジェクトにアクセスできます。

すべてのコアオブジェクトへの直接アクセスを可能にするトップレベルのメソッドとプロパティのグループも存在します。これらを使用すれば、オブジェクト階層を辿って特定のコアオブジェクトにアクセスする必要がありません。

使用できるコアオブジェクトとその API のリファレンスについては、92 ページの「Director のコアオブジェクト」を参照してください。

メディアタイプ

このカテゴリのオブジェクトは、キャストメンバとしてムービーに追加される RealMedia、DVD、アニメーション GIF などの様々なメディアタイプの機能へのアクセスを提供します。

メディアタイプは実際にはオブジェクトというよりも、特定のメディアタイプに属するキャストメンバに近い概念です。メディアタイプをムービーにキャストメンバとして追加すると、コアメンバオブジェクトの機能が継承されるだけでなく、特定のメディアタイプ専用の追加機能を提供することで、メンバオブジェクトが拡張されます。例えば、RealMedia キャストメンバはメンバオブジェクトのメソッドとプロパティにアクセスできることに加えて、RealMedia 固有のメソッドとプロパティも利用できます。その他のメディアタイプもすべて、このような動作を示します。

使用できるメディアタイプとその API のリファレンスについては、112 ページの「メディアタイプ」を参照してください。

スクリプトオブジェクト

このカテゴリのオブジェクトは、**Xtra** 拡張機能としても知られており、Director と一緒にインストールされて Director の基本機能を拡張する XML Parser、Fileio、SpeechXtra などのソフトウェアコンポーネントの機能へのアクセスを提供します。既存の Xtra 拡張機能は、フィルタの読み込みやインターネットへの接続などの機能を提供します。C プログラミング言語でプログラミングできれば、独自のカスタム Xtra 拡張機能を作成できます。

使用できるスクリプトオブジェクトとその API のリファレンスについては、132 ページの「スクリプトオブジェクト」を参照してください。

3D オブジェクト

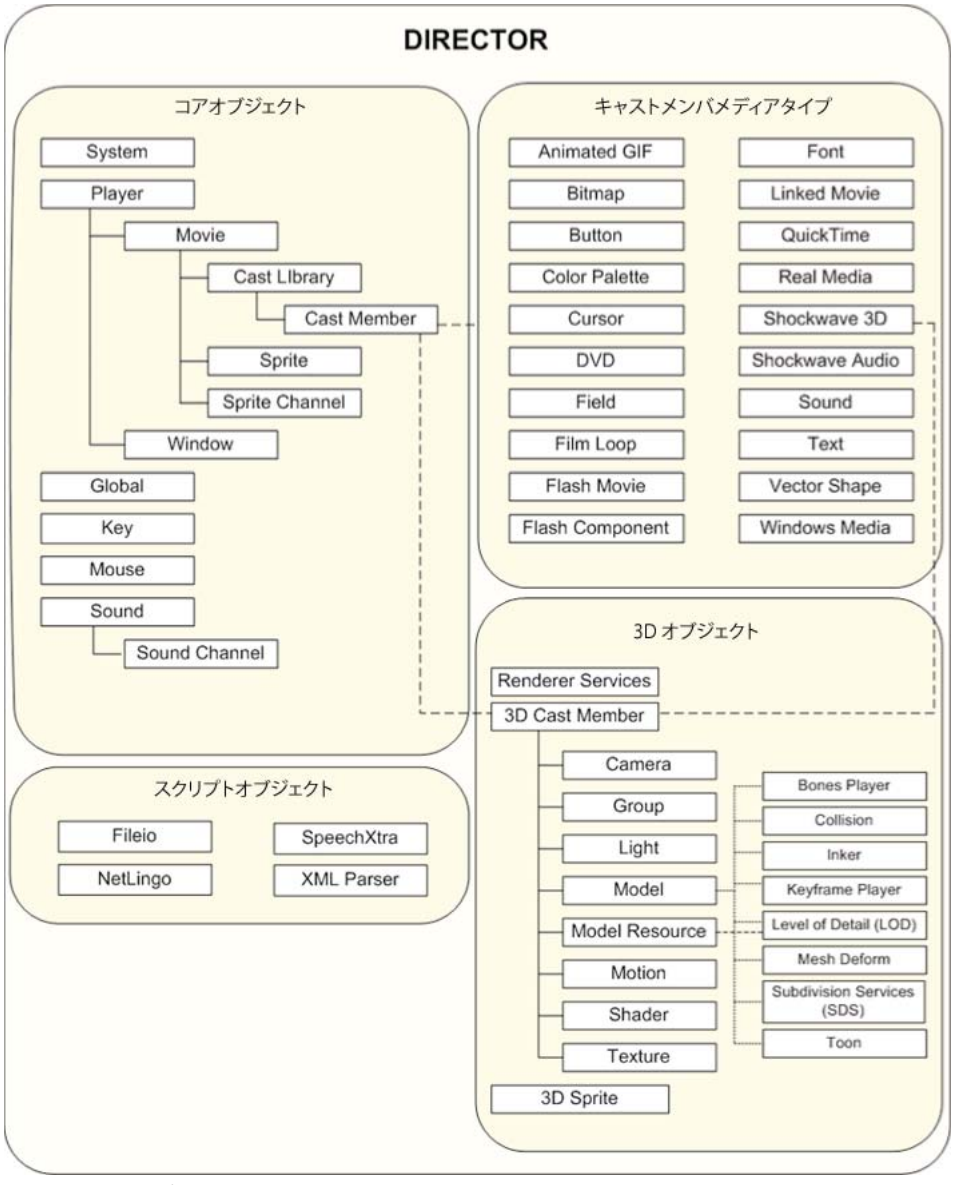
このカテゴリのオブジェクトは、3D ムービーの作成に使用されるキャストメンバおよびテキストの機能へのアクセスを提供します。

3D ムービーについて詳しくは、Director ヘルプパネルの「Director の使い方」トピックを参照してください。

使用できる 3D オブジェクトとその API のリファレンスについては、154 ページの「3D オブジェクト」を参照してください。

オブジェクトモデル図

次の図は、Director の上位レベルのオブジェクトグループとその階層の間の基本的な関係を図示したものです。オブジェクトの作成、プロパティとメソッド、およびその他の API について詳しくは、関連 API のリファレンストピックを参照してください。



オブジェクトグループとその階層

トップレベルの関数とプロパティ

Director のコアオブジェクトとその機能への直接アクセスを提供するトップレベルの関数とプロパティは多数あります。コアオブジェクト、新しい画像、リストなどへの参照を作成すると、これらの関数やプロパティの多くを幅広く活用することになります。例えば、トップレベルの _movie プロパティはコアムービーオブジェクトを直接参照し、トップレベルの list() 関数はリニアリストを作成します。

次の表で、トップレベルの関数とプロパティの一覧を示します。

トップレベルの関数	
castLib()	randomVector()
channel() （トップレベル）	rect()
color()	script()
date() （フォーマット）, date() （システム）	showLocals()
image()	sound()
isBusy()	sprite()
list()	symbol()
member()	timeout()
point()	trace()
propList()	vector()
put()	window()
random() らんだむ ()	xtra()

トップレベルのプロパティ	
_global	_player
_key	_sound
_mouse	_system
_movie	

Director でのオブジェクト指向プログラミングについて

Lingo または JavaScript シンタックスを使用すると、スクリプトにオブジェクト指向プログラミングの原則を適用することができます。一般に、オブジェクト指向の原則を適用すると、記述するコード量が少なくなり、作業を実行するために使用するロジックがシンプルになり、さらにコードの再利用性やモジュール性が向上して、結果的にプログラミングが簡単になります。

使用するスクリプト言語に応じて、次のいずれかのパラダイムを使用してこれらの原則を適用できます。

- Lingo では、親スクリプト、アンセスタ（祖先）スクリプトおよび子オブジェクトを使用して、オブジェクト指向プログラミングをシミュレートします。
- JavaScript シンタックスでは、標準の JavaScript スタイルのオブジェクト指向プログラミングを使用してクラスとサブクラスを作成します。

どちらのパラダイムでもスクリプトにオブジェクト指向プログラミングの利点を適用することができるため、どちらのスクリプト言語を使用するかは実際には問題ではありません。ただ単に異なる方法でこの原則を適用するだけです。

各スクリプト言語で異なるパラダイムを使用してオブジェクト指向の原則を適用することになるため、片方の言語用に説明した手法は、他方の言語では動作しません。このため、以下から自分が使用するスクリプト言語に当てはまる内容だけお読みください。

- Lingo でオブジェクト指向プログラミングをシミュレートする場合について詳しくは、「46 ページの「Lingo シンタックスでのオブジェクト指向プログラミング」」を参照してください。

- JavaScript シンタックスでのオブジェクト指向プログラミングについて詳しくは、54 ページの「JavaScript シンタックスでのオブジェクト指向プログラミング」を参照してください。

Lingo シンタックスでのオブジェクト指向プログラミング

Lingo では、親オブジェクトがオブジェクト指向プログラミングの利点を実現します。親スクリプトを使用すると、同じような動作および反応をするスクリプトオブジェクトでありながら、互いに独立して動作するオブジェクトを作成することができます。

Lingo を使用すると、1 つの親スクリプトから複数のコピー（インスタンス）を作成することができます。親スクリプトの各インスタンスを子オブジェクトと呼びます。子オブジェクトは、必要に応じてムービーの再生時に作成することができます。同じ親スクリプトから作成できる子オブジェクトの数に制限はありません。子オブジェクトは、コンピュータのメモリ容量の許す限り、いくつでも作成できます。

Director では、1 つのビヘイビアーから異なるスプライトに複数のインスタンスを作成できるのと同様に、同じ 1 つの親スクリプトから複数の子オブジェクトを作成することができます。親スクリプトをテンプレートと考え、親テンプレートを実装したものが子オブジェクトと考えるとよいでしょう。

Lingo の親スクリプトと子オブジェクトについての説明では、親スクリプトを記述して、子オブジェクトを作成し、使用する基本的な方法を説明し、スクリプト例も示します。ここではオブジェクト指向プログラミングの基本概念については説明しませんが、親スクリプトと子オブジェクトを使いこなすには、オブジェクト指向プログラミングの原則を理解する必要があります。オブジェクト指向プログラミングの基本については、市販の関連資料を参照してください。

他のオブジェクト指向言語との類似性

Java や C++ などのオブジェクト指向プログラミング言語をご存知の方は、親スクリプトの基本的な概念を既に理解されている場合がありますが、Lingo とは違う名前で理解しているでしょう。

Director で親スクリプトと子オブジェクトの説明に使用する用語は、次の一般的なオブジェクト指向プログラミングの用語に対応しています。

Director での**親スクリプト**は、オブジェクト指向プログラミングでのクラスに対応します。

Director での**子オブジェクト**は、オブジェクト指向プログラミングでのインスタンスに対応します。

Director での**プロパティ変数**は、オブジェクト指向プログラミングでのインスタンス変数またはメンバ変数に対応します。

Director での**ハンドラ**は、オブジェクト指向プログラミングでのメソッドに対応します。

Director での**アンセスタスクリプト**は、オブジェクト指向プログラミングでのスーパークラスまたは基本クラスに対応します。

親スクリプトと子オブジェクトの基本

Lingo では、親スクリプトは、子オブジェクトを定義するハンドラとプロパティのセットで、子オブジェクトそのものではありません。子オブジェクトは、親スクリプトから独立して存在する自立型のインスタンスです。同じ親スクリプトから作成されたすべての子オブジェクトは同一のハンドラとプロパティを共有するため、同じ親から作成された子オブジェクトはイベントやメッセージに対して同じような応答をします。

通常、親スクリプトは、ムービーロジックの編成を簡単にする子オブジェクトを作成するために使用されます。このような子オブジェクトは、ムービーで、同じロジックを異なるパラメータで何度も同時に実行する必要がある場合に特に便利です。また、アニメーションを制御する方法として、子オブジェクトをスプライトオブジェクトの scriptInstanceList またはムービーオブジェクトの actorList に追加することもできます。

同じ親スクリプトから作成されたすべての子オブジェクトは同一のハンドラを持っているため、イベントに対して同じように応答します。ただし、子オブジェクトは、親スクリプトで定義されたプロパティに対してそれぞれ独自の値を保持するため、同じ親スクリプトのインスタンスであっても、子オブジェクトごとに異なる動作をすることができます。

例えば、編集可能なテキストフィールドになる子オブジェクトを定義する親スクリプトを作成したとすると、その各子オブジェクトは、その他のテキストフィールドの設定に関係なく、それぞれ独自のプロパティ設定、テキストおよびカラーを持ちます。どれか 1 つの子オブジェクトのプロパティの値を変更すれば、同じ親スクリプトから作成された他の子オブジェクトには一切影響を与えることなく、ムービーの再生に合わせてその子オブジェクトの特性を変更することができます。

同様に、ある子オブジェクトのプロパティを、同じ親スクリプトから作成された他の子オブジェクトのプロパティ設定に関係なく、TRUE または FALSE に設定できます。

親スクリプトは、プロパティ変数とハンドラが入っているスクリプトキャストメンバの名前を参照します。親スクリプトから作成された子オブジェクトは、基本的に、スクリプトキャストメンバの新しいインスタンスとなります。

子オブジェクトとビヘイビアーの違い

子オブジェクトとビヘイビアーは複数のインスタンスを持つことができるという点で似ていますが、重要な違いもいくつかあります。子オブジェクトとビヘイビアーの最も大きな違いは、ビヘイビアーはスプライトに割り当てられるため、スコア内の位置に関連付けられていることです。再生ヘッドがフレーム間で移動し、ビヘイビアが割り当てられているスプライトに到達すると、スコアに格納されたイニシャライザからビヘイビアオブジェクトが自動的に作成されます。一方、親スクリプトから子オブジェクトを作成するには、ハンドラで明示的に作成する必要があります。

ビヘイビアーと子オブジェクトは、スプライトに割り当てられる方法も異なります。ビヘイビアーは割り当てられるスプライトと自動的に関連付けられますが、子オブジェクトはスプライトと明示的に関連付ける必要があります。子オブジェクトはスプライト参照を必要とせず、メモリ内にのみ存在します。

アンセスタの基本

親スクリプトではアンセスタを宣言できます。アンセスタとは追加のスクリプトであり、そのハンドラおよびプロパティを子オブジェクトから呼び出して使用できます。

アンセスタスクリプトを使用すると、複数の親スクリプトに繰り返し使用できるハンドラとプロパティのセットを作成できます。

親スクリプトは、ancestor プロパティにスクリプトを割り当てることによって、別の親スクリプトをアンセスタにします。例えば、次のステートメントは What_Everyone_Does というスクリプトを、このステートメントがある親スクリプトのアンセスタにします。

```
-- Lingo syntax
ancestor = new(script "What_Everyone_Does")
```

子オブジェクトにハンドラとプロパティが定義されていない場合は、子オブジェクトのアンセスタでハンドラとプロパティが検索されます。検索は子オブジェクトの親スクリプトから開始されます。ハンドラを呼び出した場合、またはプロパティをテストした場合に、親スクリプトに定義されていなければ、アンセスタスクリプトで定義が検索されます。定義がアンセスタスクリプトにあれば、その定義が使用されます。

子オブジェクトは同時に 1 つのアンセスタしか持てませんが、そのアンセスタスクリプトはそれ自身のアンセスタを持つことができ、その先のアンセスタについても同様です。そのため、ハンドラを子オブジェクトに適用できる一連の親スクリプトを作成できます。

親スクリプトの記述

親スクリプトには、子オブジェクトを作成し、そのアクションやプロパティを定義するのに必要なコードが含まれています。まず、子オブジェクトの動作を決定してください。その上で、次のように親スクリプトを記述します。

- 必要に応じて、適切なプロパティ変数を宣言します。これらの変数は、それぞれの子オブジェクトが他の子オブジェクトとは独立して値を持つことができるプロパティを表します。
- on new ハンドラで、子オブジェクトのプロパティと変数の初期値を設定します。
- 子オブジェクトのアクションを制御する追加のハンドラを記述します。

プロパティ変数の宣言

同じ親スクリプトから作成されるそれぞれの子オブジェクトには、最初は同じプロパティ変数の値が含まれています。プロパティ変数の値は、関連付けられている子オブジェクトだけに属します。それぞれのプロパティ変数とその値は、子オブジェクトが存在する限り維持されます。通常、プロパティ変数の初期値は on new ハンドラで設定します。設定しない場合は、初期値は VOID になります。

プロパティ変数を宣言する

❖ 親スクリプトの冒頭にキーワード property を記述します。

子オブジェクトの外部からプロパティ変数を設定およびテストする

❖ **objectRef.propertyName** のシンタックスを使用することにより、スクリプトで他の任意のプロパティに行うのと同じ方法でプロパティ変数を設定し、テストします。

例えば、次のステートメントは、オブジェクト car1 の speed プロパティを設定します。

```
car1.speed = 55
```

new ハンドラの作成

通常、それぞれの親スクリプトには on new ハンドラが使用されています。このハンドラは、他のスクリプトが new(script 親スクリプト名) コマンドを発行したときに新しい子オブジェクトを作成します。このコマンドは、指定した親スクリプトに自身の子オブジェクトの作成を指示するものです。親スクリプトの on new ハンドラは、必要に応じて子オブジェクトの初期プロパティ値を設定することもできます。

on new ハンドラは、on new, という句で始め、その後に me 変数と、新しい子オブジェクトに渡すパラメータを指定します。

次に示す on new ハンドラは、親スクリプトから新しい子オブジェクトを作成し、その子オブジェクトの spriteNum プロパティを aSpriteNum パラメータに渡された値で初期化します。return me ステートメントは、最初に on new ハンドラを呼び出したハンドラに子オブジェクトを返します。

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end
```

on new ハンドラの呼び出しについて詳しくは、49 ページの「子オブジェクトの作成」を参照してください。

その他のハンドラの追加

子オブジェクトの動作を決めるには、目的の動作を実行するハンドラを親スクリプトに記述します。例えば、スプライトのカラーを変更するハンドラを追加したりできます。

次の親スクリプトは、spriteNum プロパティの値を定義し、2 番目のハンドラでスプライトの foreColor プロパティを変更しています。

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on changeColor me
    spriteNum.foreColor = random(255)
end
```

現行オブジェクトの参照

通常、1 つの親スクリプトから多数の子オブジェクトが作成され、それぞれの子オブジェクトには複数のハンドラがあります。特殊パラメータ変数 me は、子オブジェクトのハンドラに、他の子オブジェクトのプロパティに対してではなく、その子オブジェクト自体のプロパティに対して動作するように指示します。これにより、子オブジェクト内のハンドラがプロパティを参照する際には、そのハンドラが属している子オブジェクトのプロパティ値を使用するようになります。

親スクリプトでの各ハンドラ定義では、常に me を最初のパラメータ変数として記述してください。同一の親スクリプト内の他のハンドラを呼び出す必要がある場合は、常に me を親スクリプトの最初のパラメータとして定義し、これを渡すことが重要です。その理由は、親スクリプト内の子オブジェクトごとにハンドラが生成されるためです。

アンセスタスクリプトに定義されているプロパティを参照する場合は、参照元として me パラメータを指定する必要があります。これは、プロパティが、アンセスタスクリプトで定義されていても、子オブジェクトのプロパティであるためです。例えば、次のステートメントは、me 変数を使ってオブジェクトを参照し、オブジェクトのアンセスタで定義されているプロパティにアクセスしています。

```
-- Lingo syntax
x = me.y -- access ancestor property y
```

me 変数は子オブジェクトの各ハンドラ内にあるため、すべてのハンドラが同じ子オブジェクトを制御することを示します。

子オブジェクトの作成

子オブジェクトはその全体がメモリ内に存在します。ムービーと共に保存されることはありません。ディスク上に存在するのは親スクリプトとアンセスタスクリプトだけです。

新しい子オブジェクトを作成するには、new() メソッドを使用し、後で識別して操作できるように、子オブジェクトに変数名またはリスト内の位置を割り当てます。

子オブジェクトを作成して変数に割り当てるには、次のシンタックスを使用します。

```
-- Lingo syntax
variableName = new(script "scriptName", parameter1, parameter2, ...)
```

scriptName パラメータは親スクリプトの名前、**parameter1**, **parameter2**, **...** は子オブジェクトの on new ハンドラに渡すパラメータです。new() メソッドは、**scriptName** をアンセスタとする子オブジェクトを作成します。次に、指定されたパラメータで、子オブジェクト内の on new ハンドラを呼び出します。

new() ステートメントは、ムービー内のどこからでも発行できます。子オブジェクトの初期設定をカスタマイズするには、new() ステートメントで渡すパラメータの値を変更します。

子オブジェクトは、プロパティと変数の現在の値、および親スクリプトへの参照を記録するのに十分なメモリ容量のみを必要とします。そのため、多くの場合は、子オブジェクトを必要な数だけ作成し、保持することができます。

同じ親スクリプトからさらに子オブジェクトを作成するには、複数の new() ステートメントを発行します。

rawNew() メソッドを使用すれば、プロパティ変数をすぐに初期化せずに子オブジェクトを作成することができます。rawNew() メソッドは、親スクリプトの on new ハンドラを呼び出さずに子オブジェクトを作成します。多数の子オブジェクトが必要な場合は、rawNew() を使用すれば、前もってオブジェクトを作成し、各オブジェクトが必要になるまでプロパティ値の割り当てを保留にすることができます。

次のステートメントは、プロパティ変数を初期化せずに親スクリプト Car から子オブジェクトを作成し、変数 car1 に割り当てます。

```
-- Lingo syntax
car1 = script("Car").rawNew()
```

このようにして作成した子オブジェクトのプロパティを初期化するには、その子オブジェクトの on new ハンドラを呼び出します。

```
car1.new
```

子オブジェクトのプロパティの確認

個々の子オブジェクトの特定のプロパティ変数の値を確認するには、オブジェクト名 . プロパティ名という単純なシンタックスを使用します。例えば、次のステートメントは、変数 car1 に格納されている子オブジェクトの carSpeed プロパティの値を変数 x に代入します。

```
-- Lingo syntax
x = car1.carSpeed
```

オブジェクトのプロパティをオブジェクトの外部から照会できると、レーシングゲームにおけるすべての Car オブジェクトの平均スピードなど、オブジェクトのグループに関する情報を取得する場合に便利です。また、1 つのオブジェクトのプロパティを使用して、それに依存している他のオブジェクトの動作を決めることもできます。

割り当てたプロパティを確認できるだけでなく、子オブジェクトが特定のハンドラを含むかどうかを確認したり、オブジェクトの元になった親スクリプトを特定したりすることもできます。これは、似ているが微妙に異なる親スクリプトから作成されたオブジェクトが混在しているような場合に役立ちます。

例えば、複数の親スクリプトのいずれか 1 つを使用して子オブジェクトを作成するとします。このような場合、オブジェクトの親スクリプトの名前を返す script() 関数を使用すると、子オブジェクトの元になった親スクリプトを特定できます。

次のステートメントは、オブジェクト car1 が Car という名前の親スクリプトから作成されたかどうかを確認します。

```
-- Lingo syntax
if car1.script = script("Car") then
    _sound.beep()
end if
```

また、handlers() メソッドを使用して子オブジェクトのハンドラリストを取得することや、handler() メソッドを使用して特定のハンドラが子オブジェクト内に存在するかどうかを確認することもできます。

次のステートメントは、子オブジェクト car1 のハンドラリストを変数 myHandlerList に代入します。

```
-- Lingo syntax
myHandlerList = car1.handlers()
```

代入後のリストは次のようになります。

```
[#start, #accelerate, #stop]
```

次のステートメントは、handler() メソッドを使用して、ハンドラ on accelerate が子オブジェクト car1 内に存在するかどうかを確認します。

```
-- Lingo syntax
if car1.handler(#accelerate) then
    put("The child object car1 contains the handler named on accelerate.")
end if
```

子オブジェクトの削除

ムービーから子オブジェクトを削除するには、子オブジェクトを参照するすべての変数を別の値に設定します。子オブジェクトが actorList などのリストに割り当てられている場合は、そのリストからも子オブジェクトを削除する必要があります。

子オブジェクトおよびそれを参照する変数を削除する

❖ 各変数を VOID に設定します。

子オブジェクトに対する参照がなくなると、子オブジェクトは自動的に削除されます。次の例では、特定の子オブジェクトに対する参照が ball1 にしか含まれておらず、それを VOID に設定して、メモリからオブジェクトを削除しています。

```
-- Lingo syntax
ball1 = VOID
```

actorList からオブジェクトを削除する

❖ リストから項目を削除するには delete() メソッドを使用します。

scriptInstanceList の使用

scriptInstanceList プロパティを使用すると、スプライトに新しいビヘイビアーを動的に追加することができます。通常、scriptInstanceList は、スコア内で定義されたビヘイビアーイニシャライザから作成されたビヘイビアーインスタンスのリストです。このリストに親スクリプトから作成された子オブジェクトを追加すると、子オブジェクトは他のビヘイビアーに送られたメッセージを受け取ります。

例えば、次のステートメントは、スプライト 10 の scriptInstanceList プロパティに子オブジェクトを追加します。

```
-- Lingo syntax
add(sprite(10).scriptInstanceList, new(script "rotation", 10))
```

上のステートメントが参照する親スクリプトには、次のようなものがあります。

```
-- Lingo syntax parent script "rotation"
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on prepareFrame me
    sprite(spriteNum).rotation = sprite(spriteNum).rotation + 1
end
```

子オブジェクトを scriptInstanceList に追加する場合は、子オブジェクトの spriteNum プロパティを初期化する必要があります。通常、この処理は on new ハンドラに渡されるパラメータを使用して実行します。

注意：beginSprite メッセージは、動的に追加した子オブジェクトには送信されません。

scriptInstanceList のリファレンス情報については、scriptInstanceList を参照してください。

actorList の使用

再生ヘッドがフレームに入るか、updateStage() メソッドでステージが更新されるたびに、それ自体のメッセージを受け取る特別な子オブジェクト (またはその他のオブジェクト) のリストを設定することができます。

この特別なリストが actorList であり、リストに明示的に追加されたオブジェクトを含んでいます。

受け取るメッセージは stepFrame メッセージです。このメッセージは、再生ヘッドがフレームに入ったとき、および updateStage() コマンドが使用されたときにのみ送信されます。

actorList 内のオブジェクトは、各フレームで enterFrame メッセージではなく、stepFrame メッセージを受け取ります。オブジェクトに利用可能な on stepFrame ハンドラがある場合は、再生ヘッドが新しいフレームに入るか、updateStage() メソッドで ステージが更新されるたびに、このハンドラ内のスクリプトが実行されます。

actorList および stepFrame は、スプライトとして使用する子オブジェクトのアニメーション化や、再生ヘッドがフレームに入った回数を記録するカウンタの更新に使用することもできます。

on enterFrame ハンドラでも同じ結果が得られますが、actorList プロパティと stepFrame ハンドラは、パフォーマンスを高めるために最適化されています。actorList 内のオブジェクトは、enterFrame メッセージや updateStage() メソッドの後に送られるカスタムメッセージよりも stepFrame メッセージに対して効率的に応答します。

actorList にオブジェクトを追加する

❖ actorList プロパティを次のように使用します。**childObject** は、追加する子オブジェクトへの参照です。

```
-- Lingo syntax
_movie.actorList.add(childObject)
```

親スクリプトまたはアンセスタスクリプト内にあるオブジェクトの stepFrame ハンドラは、再生ヘッドが進むたびに自動的に実行されます。オブジェクトが最初のパラメータ me として on stepFrame ハンドラに渡されます。

別のムービーに分岐した場合でも actorList の内容は消去されません。これが、新しいムービー内で予期せぬ動作を引き起こすことがあります。現在のムービーの子オブジェクトを新しいムービーでは使用したくない場合は、新しいムービーの on prepareMovie ハンドラに actorList の内容を消去するステートメントを挿入してください。

actorList から子オブジェクトを消去する

❖ actorList を [ ] (空のリスト) に設定します。

actorList のリファレンス情報については、actorList を参照してください。

timeout オブジェクトの作成

timeout オブジェクトは、タイマのように動作し、タイマが切れるとメッセージを送信するスクリプトオブジェクトです。このオブジェクトは、一定の間隔で、または一定の時間が経過した後で特定のアクションを起こす必要がある場合に便利です。

timeout オブジェクトは、子オブジェクトまたはムービースクリプト内のハンドラを呼び出すメッセージを送信できます。timeout オブジェクトを作成するには、new() キーワードを使用します。オブジェクトの名前、呼び出すハンドラ、およびハンドラを呼び出す頻度を指定する必要があります。timeout オブジェクトを作成すると、現在アクティブな timeout オブジェクトのリストが格納された timeOutList というリストが Director により維持されます。

以下で説明するシンタックスは、Adobe Director 11 でオーサリングされたすべての新しいムービー、および Adobe Director 11 で再生され、その scriptExecutionStyle プロパティの値が 10 に設定されている古いムービーに必要なものです。Director MX またはそれ以前の Director でオーサリングされたムービーは、scriptExecutionStyle プロパティの値が 9 に設定されているため、Director MX 以前で使用されていたシンタックスを使用できます。

timeout オブジェクトを作成する

```
-- Lingo syntax when scriptExecutionStyle is set to 9
variableName = timeout().new(timeoutName, timeoutPeriod, #timeoutHandler, {, targetObject})

-- Lingo syntax when scriptExecutionStyle is set to 10
variableName = timeout().new(timeoutName, timeoutPeriod, timeoutHandler, targetObject)
variableName = new timeout(timeoutName, timeoutPeriod, timeoutHandler, targetObject)

// JavaScript syntax
variableName = new timeout(timeoutName, timeoutPeriod, timeoutFunction, targetObject)
```

このステートメントでは、次のエレメントを使用します。

- **variableName** は、timeout オブジェクトを格納する変数です。
- timeout は、作成する Lingo オブジェクトのタイプを示します。
- **timeoutName** は、timeout オブジェクトに付ける名前です。この名前は、timeOutList で使用されます。これはオブジェクトの #name プロパティです。
- new は、新しいオブジェクトを作成します。
- **intMilliseconds** は、指定したハンドラを timeout オブジェクトが呼び出す頻度を示します。これはオブジェクトの #period プロパティです。例えば、2000 という値にすると、指定したハンドラが 2 秒ごとに呼び出されます。
- #**handlerName** は、オブジェクトが呼び出すハンドラの名前です。これはオブジェクトの #timeOutHandler プロパティです。名前の前に # 記号を付けてシンボルとして表します。例えば、on accelerate というハンドラなら、#accelerate と指定します。
- **targetObject** は、ハンドラを呼び出す子オブジェクトを示します。これはオブジェクトの #target プロパティです。これにより、複数の子オブジェクトに同じハンドラが含まれている場合でも特定することができます。このパラメータを省略すると、指定したハンドラがムービースクリプト内で検索されます。

次のステートメントは、子オブジェクト car1 の on accelerate ハンドラを 2 秒ごとに呼び出す timer1 という timeout オブジェクトを作成します。

```
-- Lingo syntax
myTimer = timeOut("timer1").new(2000, #accelerate, car1)
```

特定の timeout オブジェクトから次の timeout メッセージがいつ送られてくるかを調べるには、その #time プロパティをチェックします。返される値は、次の timeout メッセージが送られる時間をミリ秒単位で表したものです。例えば、以下のステートメントは、timeout オブジェクトの timer1 から次の timeout メッセージが送られてくる時間を調べて、それをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(timeout("timer1").time)
```

timeOutList の使用

timeout オブジェクトの作成を始める際、timeOutList を使用すると、特定のタイミングでアクティブになっている timeout オブジェクトの数をチェックできます。

次のステートメントは、count プロパティを使用して、変数 x を timeOutList のオブジェクトの数に設定しています。

```
-- Lingo syntax
x = _movie.timeoutList.count
```

リスト内の番号で、個々の timeout オブジェクトを参照することもできます。

次のステートメントは、forget() メソッドを使用して、timeOutList 内の 2 番目の timeout オブジェクトを削除しています。

```
-- Lingo syntax
timeout(2).forget()
```

timeout オブジェクトを使用したシステムイベントの中継

特定の子オブジェクトをターゲットとする timeout オブジェクトを作成すると、その子オブジェクトがシステムイベントを受け取るようにすることができます。timeout オブジェクトは、これらのイベントをターゲットの子オブジェクトに中継します。子オブジェクトが受け取ることのできるシステムイベントには、prepareMovie、startMovie、stopMovie、prepareFrame、exitFrame などがあります。これらのイベントのハンドラを子オブジェクトに記述することにより、どのような目的に対しても、子オブジェクトがシステムイベントに応答するようにできます。子オブジェクトが受け取るシステムイベントは、それらに応答するように設計されたムービースクリプトやフレームスクリプトなどの他のスクリプトにも受け取られます。

次の親スクリプトには、システムイベント exitFrame のハンドラとカスタムハンドラ slowDown が含まれています。

```
-- Lingo syntax
property velocity

on new me
    velocity = random(55)
end

on exitFrame
    velocity = velocity + 5
end

on slowDown mph
    velocity = velocity - mph
end
```

カスタムプロパティと timeout オブジェクトの関連付け

カスタムプロパティと timeout オブジェクトを関連付けたい場合、スクリプトインスタンスオブジェクトへの参照以外のものをターゲットとして使用して timeout オブジェクトを作成すると便利です。この手法を使用すると、ターゲットデータが timeout オブジェクトに関連付けられたデータとなり、timeout ハンドラ内で使用できるようになります。

次の例で、この手法の使い方を示します。

```
-- Lingo syntax
-- initialize a timeout object and pass it a data property list (tData)
-- instead of a reference to a script instance object
tData = [#beta: 0]
tTO = timeout("betaData").new(50,#targetHandler,tData)

-- within a movie script, create the targetHandler handler
on targetHandler (aData)
    -- increment and display the beta property
    tData.beta = tData.beta + 1
    put(tData.beta)
end targetHandler
```

上の例では、beta プロパティは 1 ずつ増えていきます。つまり、同じムービースクリプトハンドラを呼び出す複数の timeout オブジェクトを初期化して、各 timeout オブジェクトにそれに関連付けられた独自のデータリストを持たせることができるのです。

基本的に、以下のことを心に留めておいてください。

- スクリプトインスタンスへの参照をターゲットとして使用する場合は、その特定のスクリプトインスタンスでのターゲットハンドラが呼び出されます。この方法では、カスタムプロパティは使用できません。
- オブジェクトインスタンス以外の何か（プロパティリストなど）へのリファレンスをターゲットとして使用する場合は、ムービースクリプト内のターゲットハンドラが呼び出されます。この方法では、カスタムプロパティを使用できます。

JavaScript シンタックスでのオブジェクト指向プログラミング

JavaScript シンタックスでのオブジェクト指向プログラミングは、Java や C++ などのその他のオブジェクト指向言語でのものとは若干異なります。オブジェクト指向言語にはクラスベースのものもありますが、JavaScript シンタックスはプロトタイプベースなのです。

次の 2 つの箇条書き項目では、クラスベースの言語と JavaScript シンタックスのようなプロトタイプベースの言語を、高いレベルで比較しています。

- クラスベースの言語では、クラス定義を作成し、クラス定義の中でそのクラスから作成されたすべてのインスタンスの特性を決定する初期プロパティとメソッドを定義します。クラス定義には、そのクラスのインスタンスの作成に使用される、コンストラクタメソッドと呼ばれる特別なメソッドも含まれます。特定のコンストラクタメソッドに関連させて new 演算子を使用することによりインスタンスを作成すると、そのインスタンスは、その親クラスのすべてのプロパティを継承します。そのインスタンスは、呼び出されたコンストラクタに応じて、そのインスタンス固有のその他の処理を実行することもできます。

  クラス定義では、新しいプロパティを定義するだけでなく、親クラスのすべてのプロパティを継承するサブクラスを作成することによって継承を実行することもでき、さらに必要に応じて継承したプロパティに変更を加えることもできます。サブクラスを作成する際に基にする親クラスは、スーパークラスとも呼ばれます。

- JavaScript シンタックスなどのプロトタイプベースの言語では、クラス、インスタンス、サブクラスなどの間の区別がありません。これらはすべて、オブジェクトと呼ばれます。JavaScript シンタックスでは、クラス定義を使用するのではなく、プロトタイプオブジェクトをテンプレートとして使用して新しいオブジェクトを作成します。新しいオブジェクトの作成には、JavaScript シンタックスでも、クラスベースの言語と同じようにコンストラクタ関数を関連させた new 演算子を使用します。

  JavaScript シンタックスでは、スーパークラスやサブクラスを使用するのではなく、プロトタイプオブジェクトをコンストラクタ関数に関連付けて継承を実行します。このプロセスは、用語が異なるだけで、スーパークラスとサブクラスの使用に非常によく似ています。

  JavaScript シンタックスでは、クラスベースの言語と違って、実行時にオブジェクトにプロパティを追加したり、オブジェクトからプロパティを削除したりできます。プロパティの追加と削除は、一連のオブジェクトに対して行うこともできます。例えば、実行時にプロトタイプオブジェクトにプロパティを追加すると、そのオブジェクトをプロトタイプとするインスタンスオブジェクトもすべてそのプロパティを持つようになります。

オブジェクト指向の用語おぶじぇくとしこうのようご

JavaScript シンタックスでは、すべてのタイプがオブジェクトと呼ばれるため、スーパークラス、サブクラス、クラス、インスタンスといったクラスベースの用語は、技術的には文字どおりの意味を持ちません。しかし、これらの用語はすべて、基本的には JavaScript シンタックスでのオブジェクトにも対応付けられており、JavaScript シンタックスでのオブジェクトの様々なタイプを表すために一般的な意味で使用すると便利です。このため、JavaScript シンタックスでのオブジェクト指向プログラミングに関する説明では、これらのクラスベースの用語が次のような意味でオブジェクトに置き換えて使用されます。

スーパークラス それを基にサブクラス（オブジェクト）が作成されるクラス。親クラス。

サブクラス スーパークラス（オブジェクト）から作成されたクラス。子クラス。

クラスくらす スーパークラスやサブクラスの総称。親クラスまたは子クラス。

インスタンスまたはオブジェクトインスタンス スーパークラスから作成された単一のオブジェクト。

カスタムクラス

オブジェクト指向プログラミングの大きな利点の 1 つは、独自のカスタムクラスを作成して、スクリプトにカスタム機能を追加できる点です。Object、String、Math といった JavaScript シンタックスで提供されている定義済みクラスは、用途によっては便利ですが、目的の作業を達成するために必要な機能を備えていない場合もあります。例えば、ムービー内の一部のオブジェクトで、車、ボート、飛行機などの輸送手段の種類を表し、各カテゴリに独自の特性と機能を示させたかったとします。この場合に必要な機能は、定義済みの JavaScript シンタックスクラスでも定義済みの Director オブジェクトでも直接には提供されていません。このため、輸送手段の種類ごとに新しいクラスを作成し、それぞれの種類に固有の特性を定義する必要があります。

JavaScript シンタックスのカスタムクラスを作成する場合でも、定義済みの Director オブジェクトのどの機能にもアクセスできることを覚えておいてください。つまり、定義済みの Director オブジェクトが目的の機能そのものを提供していなかったとしても、カスタムクラスでそれらを使用して、その値や定義済みの機能にアクセスすることができるのです。

コンストラクタ関数

JavaScript シンタックスでは、コンストラクタ関数は、新しいオブジェクトインスタンスの作成の基にするテンプレートを含んだクラスを表します。コンストラクタ関数は、新しいオブジェクトのプロパティを作成して初期化します（つまりプロパティのデフォルト状態を設定します）。

コンストラクタ関数の形式は、基本的には JavaScript シンタックスの通常のメソッド関数と同じです。コンストラクタ関数とメソッド関数の違いは、コンストラクタ関数では特別な this キーワードを使用して初期化の対象となっている新しいオブジェクトへの参照を表すという点です。メソッド関数は、通常は、与えられた一連のオブジェクトデータに対して何らかの操作を実行するだけです。

次の例で、新しい Rectangle オブジェクトの高さと幅を初期化するために使用する Rectangle コンストラクタ関数を作成する 1 つの方法を示します。

```
function Rectangle(w, h) {
    this.width = w;
    this.height = h;
}
```

コンストラクタ関数は、関数リテラルシンタックスを使用して作成することもできます。関数リテラルシンタックスは、コンストラクタを記述する単なる代替手段に過ぎず、上に示した例で使用されているシンタックスと同じ機能を提供します。次の例では、関数リテラルシンタックスを使用して、上で示したのとよく似た Rectangle コンストラクタを作成する方法を示します。

```
Rectangle = function(w, h) {
    this.width = w;
    this.height = h;
}
```

注意：ムービーに適用するコンストラクタ関数を定義する際には、コンストラクタ関数をムービースクリプト内に記述して、グローバルに使用できるようにしてください。

コンストラクタ関数にはその機能に対応した名前を付け、Rectangle、Circle などと名前の先頭を大文字にするのが、スクリプトを記述する際のよい慣習とされています。

コンストラクタ関数は、一般に新しいオブジェクトの初期化のみに使用されますが、必要であればオブジェクトを返すこともできます。初期化したオブジェクトを返すと、返されたオブジェクトが new 式の値になります。

オブジェクトインスタンス

新しいオブジェクトインスタンスを作成する最も一般的な方法は、new 演算子を使用し、その後ろにコンストラクタ関数の名前と続けるという方法です。次の例は、新しいオブジェクトインスタンスを作成します。

```
var objRandom = new Object(); // assigns a reference to an Object object
var objString = new String(); // assigns a reference to a String object
```

コンストラクタ関数では、必要であれば、オブジェクトインスタンスの状態を初期化するために、新しいオブジェクトインスタンスがコンストラクタ関数に渡すパラメータを定義することができます。新しいオブジェクトインスタンスの初期化に使用するパラメータをコンストラクタ関数で定義する場合、プロパティの値は次のように初期化されます。

- 初期化の中でコンストラクタ関数に値を渡した場合、値を受け取ったプロパティはその値に設定されます。
- 初期化の中でコンストラクタ関数に値を渡さなかった場合、値を受け取らなかったプロパティは undefined に設定されます。

新しいオブジェクトインスタンスを作成する際、関連付けられたコンストラクタ関数の本文の中では、新しいオブジェクトインスタンスを指すのにキーワード this が使用されます。このため、新しいオブジェクトインスタンスは、this.**propertyName** シンタックスを使用して定義されたすべてのプロパティで初期化されます。

次の例では、Circle コンストラクタ関数でキーワード this を使用して、新しいオブジェクトインスタンスに関連付けられる 3 つのプロパティの名前を指定しています。コンストラクタに続くステートメントは、コンストラクタに値を渡すことによって新しいオブジェクトインスタンスを初期化しています。これらの値は、キーワード this で指定されたプロパティの初期値として使用されます。

```
// Circle constructor function
function Circle(x, y, r) {
    this.xCoord = x;
    this.yCoord = y;
    this.radius = r;
}

// xCoord = 10, yCoord = 15, radius = 5
var objCircle = new Circle(10, 15, 5);
```

これで objCircle は初期化され、そのプロパティにアクセスできるようになりました。こうやって作成した objCircle インスタンスを使用すれば、何らかの変数をそのプロパティの値に設定することができます。

```
var theXCoord = objCircle.xCoord; // assigns the value 10 to theXCoord
var theYCoord = objCircle.yCoord; // assigns the value 15 to theYCoord
var theRadius = objCircle.radius; // assigns the value 5 to theRadius
```

注意：ドットシンタックスを使用してオブジェクトのプロパティやメソッドにアクセスする方法については、41 ページの「ドットシンタックス形式でのスクリプト記述」を参照してください。

新しいオブジェクトにはその機能に対応する名前を与え、objRectangle、objCircle のように小文字を使用して名前を付けるのが、スクリプトを記述する際のよい慣習とされています。

オブジェクトリテラルシンタックスを使用してオブジェクトインスタンスを作成することもできます。この方法では、new 演算子とコンストラクタ関数が必要ありません。一般に、この方法を使用するのは、コンストラクタ関数で定義されていないオブジェクトを作成しようとしていて、インスタンスが 1 つしか必要ない場合だけです。次の例では、x = 1、y = 2、radius = 2 のオブジェクトインスタンスを作成しています。

```
var objSmallCircle = { x:1, y:2, radius:2 };
```

オブジェクトの継承

オブジェクト指向プログラミングでオーサリングすることの主な利点は、独自のカスタムクラスを作成できることだけでなく、サブクラスを作成して、基になったスーパークラスのプロパティとメソッドを継承することができる点です。継承により、既に組み込みのプロパティと機能を備えているオブジェクトを簡単に作成することが可能になります。

JavaScript シンタックスでは、その他のすべてのサブクラスの基になる基本クラスとして機能する 1 つのスーパークラスがあります。Object スーパークラスです。Object スーパークラスには、いくつかの基本的なプロパティとメソッドが含まれています。Object をテンプレートとして作成されたサブクラスは、必ずこれらの基本プロパティと基本メソッドを継承し、おそらくはそれに加えて独自のプロパティとメソッドが定義されることになります。これらのクラスのサブクラスは、Object からも、その他のスーパークラスからも継承を行います。自分で作成するその他のオブジェクトはすべて、この継承の連鎖を続けます。

例えば、Object には、constructor プロパティと toString() メソッドが含まれています。SubObj1 という名前の新しいクラスを作成する場合、それは Object のサブクラスであり、そのため Object の constructor プロパティと toString() メソッドを継承します。その後、SubObj1 をスーパークラスとして SubObj2 という別のクラスを作成すると、SubObj2 は Object の constructor プロパティと toString() メソッドを継承し、さらに SubObj1 で定義されているカスタムプロパティとカスタムメソッドがあれば、それらも継承します。

カスタムコンストラクタ関数が Object スーパークラスから継承する 2 つの重要なプロパティとして、prototype と constructor があります。prototype プロパティは、クラスのプロトタイプオブジェクトを表します。これにより、オブジェクトインスタンスに変数（プロパティ）とメソッドを追加することが可能になります。これが、JavaScript シンタックスで通常実装されている継承の手段です。constructor プロパティは、コンストラクタ関数自体を表します。以降では、この 2 つのプロパティの使用方法について説明します。

プロトタイプオブジェクト

既に説明したように、サブクラスを作成すると、それはその基になったスーパークラスのプロパティとメソッドを自動的に継承します。JavaScript シンタックスでは、継承は、通常、プロトタイプオブジェクトを使用して実装されます。サブクラスは、実際には、そのスーパークラスのプロトタイプオブジェクトからプロパティとメソッドを継承しており、スーパークラス自体から継承しているわけではありません。これは非常に重要なポイントで、これによりはっきりとした利点が得られます。必ずしもすべてのプロパティとメソッドが、クラスからそのクラスのオブジェクトインスタンスにそのままコピーされる必要はありません。そのため、新しいオブジェクトインスタンスに必要なメモリの量が大幅に減少します。

JavaScript シンタックスでのクラスは、定義済みの Object クラスも含めて、すべてプロトタイプオブジェクトを 1 つだけ含んでいます。1 つのクラスから作成されたすべてのオブジェクトインスタンスが、そのクラスのプロトタイプオブジェクト内のプロパティとメソッドにアクセスすることができます。このため、通常は、クラスのプロトタイプオブジェクトが、そのクラスのプロパティとメソッドを実際に格納している唯一のオブジェクトです。オブジェクトインスタンスには、そのインスタンスの初期化に必要なプロパティしか含まれていません。

コードでは各オブジェクトインスタンスからこれらのプロパティとメソッドに直接アクセスできるため、各オブジェクトインスタンスが実際にこれらのプロパティとメソッドを持っているように見えますが、実際には、インスタンスはプロトタイプオブジェクトを使用してこれらにアクセスしているのです。クラスのプロトタイプオブジェクトは、クラスを作成したときに自動的に作成されます。プロトタイプオブジェクトにアクセスするには、クラスの prototype プロパティを使用します。

クラスのプロトタイプオブジェクトに格納されているプロパティはすべてのオブジェクトインスタンスによって共有されるため、これらは、すべてのオブジェクトインスタンスによって値が共有されるプロパティとメソッドを定義するのに理想的です。オブジェクトインスタンス間でプロパティとメソッドを共有することにより、定義済みのデフォルトビヘイビアーを示すインスタンスを簡単に作成できるうえ、デフォルトビヘイビアーから逸脱したインスタンスをカスタマイズすることもできます。

プロトタイプオブジェクトは、基本的に、オブジェクトインスタンスごとに値が変わるプロパティやメソッドの定義には適していません。オブジェクトインスタンスごとに値が変わるような場合には、一般に、それらのプロパティとメソッドをクラス自体の中で定義します。

カスタムプロパティまたはメソッドの範囲を指定するために、インスタンス変数、インスタンスメソッド、クラス変数、またはクラスメソッドとしてこれを定義します。

インスタンス変数

インスタンス変数とは、コンストラクタ関数内で定義され、そのコンストラクタの各オブジェクトインスタンスにコピーされる変数（プロパティ）です。すべてのオブジェクトインスタンスが、インスタンス変数の自分専用のコピーを持ちます。つまり、Circle クラスの 5 つのオブジェクトインスタンスが存在する場合は、そのクラスで定義されている各インスタンス変数の 5 つのコピーが存在することになります。インスタンス変数は、各オブジェクトインスタンスがそれ専用のコピーを持っているため、各オブジェクトインスタンスで独自の値を割り当てることができます。これを行っても、そのインスタンス変数の他のコピーの値が変わることはありません。インスタンス変数には、それが格納されているオブジェクトインスタンスから直接アクセスすることになります。

次の例では、make、model、color、speed の 4 つのインスタンス変数をコンストラクタ関数内で定義しています。この 4 つのインスタンス変数は、Car コンストラクタのどのオブジェクトインスタンスからも直接使用できます。

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = 0;
}
```

次のオブジェクトインスタンス objCar には、4 つのインスタンス変数すべてが含まれています。インスタンス変数 speed の値は Car コンストラクタに渡されていませんが、それでも Car コンストラクタ内で speed 変数が定義されているため、objCar は初期値が 0 になった speed プロパティを持ちます。

```
// objCar.make="Subaru", objCar.model="Forester",
// objCar.color="silver", objCar.speed = 0
var objCar = new Car("Subaru", "Forester", "silver");
```

インスタンスメソッド

インスタンスメソッドとは、オブジェクトインスタンスを通じてアクセスされるメソッドです。オブジェクトインスタンスが、インスタンスメソッドの自分専用のコピーを持つわけではありません。そうではなく、インスタンスメソッドはまず関数として定義され、次にコンストラクタ関数のプロトタイプオブジェクトのプロパティが関数の値に設定されます。インスタンスメソッドでは、定義するコンストラクタ関数の本文内で、処理の対象となっているオブジェクトインスタンスを指すためにキーワード this が使用されます。オブジェクトインスタンスはインスタンスメソッドのコピーを持ちませんが、それでもインスタンスメソッドには関連付けられているオブジェクトインスタンスから直接アクセスすることになります。

次の例では、Car_increaseSpeed() という関数を定義しています。定義した後、関数名を Car クラスのプロトタイプオブジェクトの increaseSpeed プロパティに割り当てています。

```
// increase the speed of a Car
function Car_increaseSpeed(x) {
    this.speed += x;
    return this.speed;
}
Car.prototype.increaseSpeed = Car_increaseSpeed;
```

これで、Car のオブジェクトインスタンスは、次のようなシンタックスを使用して increaseSpeed() メソッドにアクセスし、その値を変数に代入することができるようになります。

```
var objCar = new Car("Subaru", "Forester", "silver");
var newSpeed = objCar.increaseSpeed(30);
```

インスタンスメソッドは、関数リテラルシンタックスを使用して作成することもできます。関数リテラルシンタックスを使用すると、関数を定義したり、プロパティ名を関数名に割り当てたりする必要がなくなります。

次の例では、上の例で定義した increaseSpeed() 関数と同じ機能を持つ increaseSpeed() メソッドを、関数リテラルシンタックスを使用して定義しています。

```
// increase the speed of a Car
Car.prototype.increaseSpeed = function(x) {
    this.speed += x;
    return this.speed;
}
```

クラス変数

クラス変数は、オブジェクトインスタンスではなくクラスに関連付けられている変数（プロパティ）で、スタティック変数とも呼ばれます。そのクラスから作成されたオブジェクトインスタンスの数に関係なく、クラス変数のコピーは常に 1 つしかありません。クラス変数では、プロトタイプオブジェクトを使用して継承が実装されることはありません。クラス変数には、オブジェクトインスタンスを通じてではなく、クラスを通じて直接アクセスします。コンストラクタ関数内でクラスを定義しなければ、クラス変数を定義することはできません。

次の例では、2 つのクラス変数、MAX_SPEED と MIN_SPEED を定義しています。

```
function Car() { // define a Car class
    ...
}

Car.MAX_SPEED = 165;
Car.MIN_SPEED = 45;
```

MAX_SPEED クラス変数と MIN_SPEED クラス変数には、Car クラスから直接アクセスします。

```
var carMaxSpeed = Car.MAX_SPEED; // carMaxSpeed = 165
var carMinSpeed = Car.MIN_SPEED; // carMinSpeed = 45
```

クラスメソッド

クラスメソッドは、オブジェクトインスタンスではなくクラスに関連付けられているメソッドで、スタティックメソッドとも呼ばれます。そのクラスから作成されたオブジェクトインスタンスの数に関係なく、クラスメソッドのコピーは常に 1 つしかありません。クラスメソッドでは、プロトタイプオブジェクトを使用して継承が実装されることはありません。クラスメソッドには、オブジェクトインスタンスを通じてではなく、クラスを通じて直接アクセスします。コンストラクタ関数内でクラスを定義しなければ、クラスメソッドを定義することはできません。

次の例では、新しい car インスタンスのデフォルトの速度を変更する setInitialSpeed() という関数を定義しています。その後、関数名を Car クラスの setInitialSpeed プロパティに割り当てています。

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = Car.defaultSpeed;
}
Car.defaultSpeed = 10; // initial speed for new Car instances
// increase the speed of a Car
function Car_setInitialSpeed(x) {
    Car.defaultSpeed = x;
}
Car.setInitialSpeed = Car_setInitialSpeed;
```

setInitialSpeed() クラスメソッドには、Car クラスから直接アクセスします。

```
var newSpeed = Car.setInitialSpeed(30);
```

クラスメソッドは、関数リテラルシンタックスを使用して作成することもできます。次の例では、上の例で定義した setInitialSpeed() 関数と同じ機能を持つ setInitialSpeed() メソッドを、関数リテラルシンタックスを使用して定義しています。

```
// increase the speed of a Car
Car.setInitialSpeed = function(x) {
    Car.defaultSpeed = x;
}
```

クラスを定義するための推奨手順

ここでは、クラスを定義する場合に従うとよい推奨手順を示します。

1. すべてのオブジェクトインスタンスの初期化にテンプレートとして使用されるコンストラクタ関数を定義します。必要であれば、オブジェクトインスタンスを指すのにキーワード this を使用することによって、コンストラクタ関数の中で任意のインスタンス変数を定義することもできます。
2. クラスのプロトタイプオブジェクトに格納されるインスタンスメソッドを定義します。必要であれば、追加のインスタンス変数も定義できます。これらのインスタンスメソッドと変数は、すべてのオブジェクトインスタンスから使用可能で、クラスのプロトタイプオブジェクトを通じてアクセスすることができます。
3. クラス自体の中に保存されるクラスメソッド、クラス変数、および定数を定義します。これらのクラスメソッドおよびクラス変数には、クラス自体を通じてのみアクセスできます。

JavaScript シンタックスでは、コード内でオブジェクトインスタンスのプロパティにアクセスすると、オブジェクトインスタンス自体からプロパティが検索されます。インスタンス内にプロパティが含まれていない場合は、そのインスタンスの作成に使用されたスーパークラスのプロトタイプオブジェクトが検索されます。オブジェクトインスタンスの作成に使用されたクラスのプロトタイプオブジェクトよりもオブジェクトインスタンスの方が先に検索されるため、オブジェクトインスタンスは、基本的に、そのスーパークラスのプロトタイプのプロパティを隠すことになります。つまり、オブジェクトインスタンスとそのスーパークラスで、実際に同じ名前のプロパティを別の値で定義することも可能です。

変数の削除

クラス変数およびインスタンス変数は、delete 演算子を使用することにより削除できます。次の例で、このプロセスを示します。

```
function Car() { // define a Car constructor function
    ...
}
Car.color = "blue"; // define a color property for the Car class
Car.prototype.engine = "V8"; // define an engine property for the prototype

var objCar = new Car();

trace(Car.color); // displays "blue"
trace(objCar.engine); // displays "V8"

delete Car.color;
delete Car.prototype.engine;

trace(Car.color); // displays undefined
trace(objCar.engine); // displays undefined
```

プロトタイプオブジェクトの constructor プロパティへのアクセス

JavaScript シンタックスでは、コンストラクタ関数を作成することによりクラスを定義すると、そのクラスのプロトタイプオブジェクトが作成されます。作成されたプロトタイプオブジェクトには、初期状態では、コンストラクタ関数自体を指す constructor プロパティが含まれています。プロトタイプオブジェクトの constructor プロパティを使用すると、そのオブジェクトのタイプを調べることができます。

次の例では、constructor プロパティには、オブジェクトインスタンスの作成に使用されたコンストラクタへの参照が格納されています。constructor プロパティの値は、実際にはコンストラクタ自体への参照であり、コンストラクタの名前を格納された文字列なわけではありません。

```
function Car() { // define a Car class
    // initialization code here
}
var myCar = new Car(); // myCar.constructor == function Car() {}
```

プロパティを動的に作成する

プロトタイプオブジェクトを使用して継承を実装することのもう 1 つの利点は、プロトタイプオブジェクトに追加されたプロパティとメソッドが自動的にオブジェクトインスタンスから使用可能になることです。これは、プロパティやメソッドを追加するより前に作成していたオブジェクトインスタンスにもこのようになります。

次の例では、Car のオブジェクトインスタンスを作成した後に、Car クラスのプロトタイプオブジェクトに color プロパティを追加しています。

```
function Car(make, model) { // define a Car class
    this.make = make;
    this.model = model;
}
var myCar = new Car("Subaru", "Forester"); // create an object instance

trace(myCar.color); // returns undefined

// add the color property to the Car class after myCar was initialized
Car.prototype.color = "blue";

trace(myCar.color); // returns "blue"
```

インスタンスを作成した後に、オブジェクトインスタンスにプロパティを追加することもできます。特定のオブジェクトインスタンスにプロパティを追加すると、そのプロパティは、そのオブジェクトインスタンスからのみ使用可能になります。前の例で作成した myCar オブジェクトインスタンスを使用して、次のステートメントでは、既に作成されている myCar に color プロパティを追加しています。

```
trace(myCar.color); // returns undefined

myCar.color = "blue"; // add the color property to the myCar instance

trace(myCar.color); // returns "blue"

var secondCar = new Car("Honda", "Accord"); // create a second object instance

trace(secondCar.color); // returns undefined
```

スクリプトウィンドウでのスクリプトの記述

ムービーのスクリプトを記述する場合、スクリプトの数と種類が非常に多くなることがあります。使用するメソッドとプロパティ、スクリプトを効果的に構造化する方法、スクリプトを配置する場所などの決定には、ムービーが複雑になればなるほど特に、十分な計画とテストが必要になります。

スクリプトの記述を始める前に、目標を明確にし、何を達成したいのかを理解します。これは通常時間のかかる作業ですが、作品のストーリーボードを作成するのと同じくらい重要です。

ムービーの全体象が決まったら、スクリプトの記述およびテストを開始してみましょう。これには時間がかかります。スクリプトを思ったとおり機能させるには、記述、テスト、およびデバッグのサイクルを何度も繰り返さなければならないことがよくあります。

最良の方法は、簡単なスクリプトの記述から始めることと、頻繁にテストを行うことです。スクリプトの 1 つの部分が動作するようになったら、次の部分の記述を開始します。この方法により、効率的にバグを特定でき、スクリプトが複雑になってもスクリプトの安定性を確保できます。

スクリプトウィンドウには、スクリプトの作成と編集を支援する様々な機能が用意されています。

Director のスクリプトウィンドウでは、スクリプトベースの高度なインタラクティビティをムービーに追加することができます。スクリプトウィンドウでは、Lingo と JavaScript シンタックスのどちらを使用してもコーディングできます。Lingo は、Director の従来からあるスクリプト言語です。JavaScript シンタックスは、JavaScript での作業を好むマルチメディア開発者をサポートするために、最近導入されました。

ステージでのスプライトの移動やサウンドの再生といった Director のグラフィカルインターフェイスで行えるのと同じ作業の多くが、スクリプトウィンドウでスクリプトを記述することによって実行できます。しかし、スクリプトを記述することの実用性のほとんどは、それがムービーにもたらす柔軟性にあります。ムービーにスクリプトを持たせると、一連のフレームをスコアの指示どおりに再生するのではなく、特定の条件やイベントに応答してフレームの再生を制御することができるのです。

注意：Director には、独自のスクリプトを作成できるスクリプトウィンドウだけでなく、スプライトやフレームにドラッグするだけで使用できるパッケージ化された命令セット（ビヘイビアーと呼ばれる）も含まれています。ビヘイビアを使用すると、スクリプトを記述しなくてもスクリプトベースのインタラクティビティを追加できます。ビヘイビアについて詳しくは、Director ヘルプパネルのビヘイビアのトピックを参照してください。

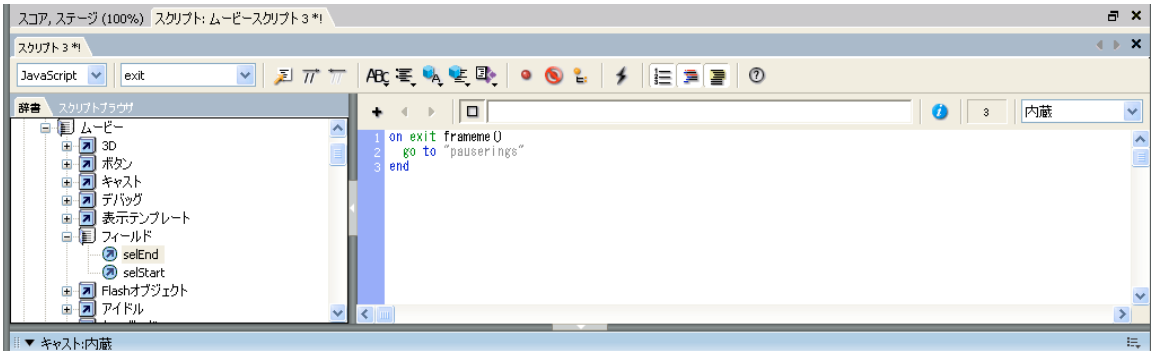
スクリプトウィンドウには、エクスプローラパネルとスクリプトエディタがあります。デフォルトでは、エクスプローラパネルはスクリプトエディタの左側に表示されます。エクスプローラパネルは、辞書ビューまたはスクリプトブラウザビューで表示できます。エクスプローラパネルのデフォルト位置を設定するには、スクリプトウィンドウの環境設定ダイアログボックス（編集／環境設定／スクリプト）を使用します。

辞書ビュー

辞書ビューには、組み込みの Lingo スクリプト / JavaScript 関数のリストがツリー構造で表示されます。関数はカテゴリを基に分類され、インデックスではアルファベット順で表示されます。

辞書ビューを使用して、次の作業を行います。

- Lingo スクリプトおよび JavaScript の組み込み関数を参照します。
- 内蔵関数を使用してスクリプトを作成できます。



辞書ビューを使用して、関数を参照し、スクリプトを作成する

1. ウィンドウ／スクリプトを選択します。スクリプトウィンドウが表示されます。
2. 「辞書」タブをクリックします。
3. ポップアップメニューから「Lingo」または「JavaScript」を選択します。対応する内蔵関数が下のパネルに表示されます。

   Lingo：Lingo スクリプト、3D Lingo Scripts および現在のムービーに使用されているスクリプト Xtras の関数が表示されます。関数は、カテゴリ（グローバル、ムービー、プレーヤなど）で整理されています。

   JavaScript：JavaScripts、3D JavaScripts および現在のムービーに使用されているスクリプト Xtras の関数が表示されます。関数は、カテゴリ（グローバル、ムービー、プレーヤなど）で整理されています。

4. 各カテゴリの横にあるプラス記号（+）をクリックしてカテゴリを展開すると、そのカテゴリに関連のある関数が表示されます。関数をアルファベット順で表示するには、インデックスカテゴリを展開します。
5. スクリプトを作成するためにスクリプトエディタに関数を追加するには、追加する関数をダブルクリックします。
6. スクリプトを保存します。
7. 全スクリプトをリコンパイルアイコンをクリックします。

スクリプトブラウザビュー

スクリプトブラウザビューには、ムービーで使用されているスクリプトと関連ハンドラが表示されます。このビューで、新しいスクリプトおよびハンドラを作成できます。

スクリプトブラウザビューを使用して、次の作業を行います。

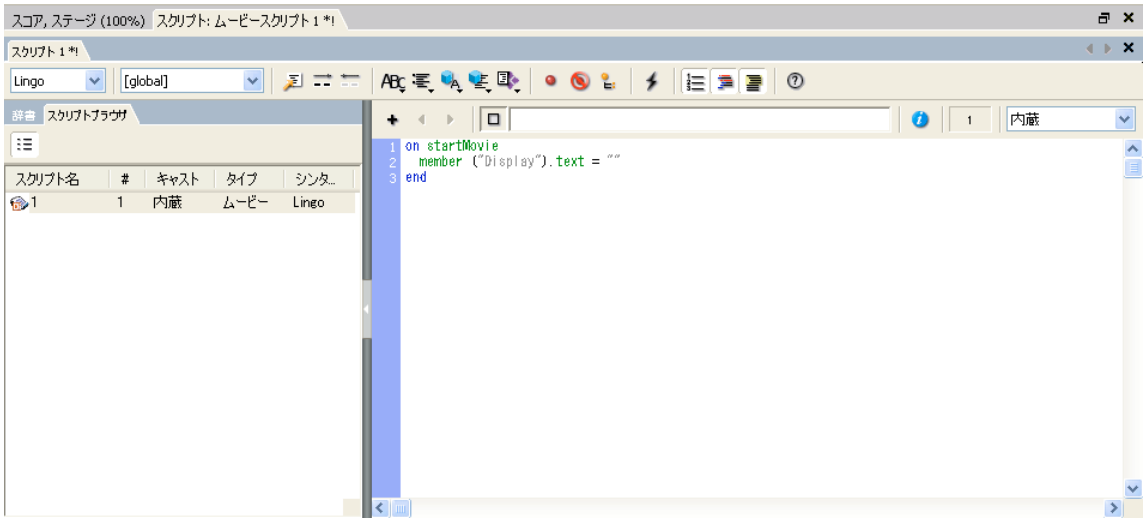
- 現在のムービー内のスクリプトとハンドラをツリーまたはリストとして参照できます。
- リストビューでは、スクリプトをスクリプト名、キャスト名、キャスト番号またはスクリプトタイプでソートできます。
- スクリプトエディタでのハンドラの場所を見つけることができます。
- 各スクリプトタイプまたはスクリプトキャストの下でスクリプトを作成できます。

スクリプトブラウザビューを使用して、スクリプトを参照し、作成する

1. ウィンドウ／スクリプトを選択して、スクリプトウィンドウを開きます。
2. 「スクリプトブラウザ」タブをクリックします。スクリプトが、下のパネルにツリー構造で表示されます。スクリプトをリストとして表示するには、「スクリプトブラウザビュー」ボタンをクリックします。

   ツリービュー：スクリプトは、その作成に使用されたスクリプトタイプ（ビヘイビアスクリプト、ムービースクリプト、親スクリプト、キャストライブラリなど）を基に分類されます。キャストメンバスクリプトもここにリストされます。スクリプトエディタでスクリプトを開くには、スクリプトをダブルクリックします。スクリプトに属するハンドラが、関連スクリプトノードの下にツリーとして表示されます。
3. スクリプト内でコンパイルされているハンドラのリストを表示するには、スクリプトの <name> ノードを展開します。コンパイルされていないハンドラは表示されません。
4. スクリプトエディタでハンドラの場所を見つけるには、ハンドラをダブルクリックします。ハンドラが、スクリプトエディタ内でハイライト表示されます。スクリプトバーでハンドラへ移動アイコンをクリックしても同じことが行えます。詳しくは、70 ページの「スクリプト内のハンドラとテキストの検索」を参照してください。
5. スクリプトにコメントを追加したり、スクリプトからコメントを削除するには、スクリプトバーのコメントを設定アイコンまたはコメントを解除アイコンをクリックします。
6. ブレークポイントのオン / オフを切り替えるには、スクリプトバーのブレークポイントの切り替えアイコンをクリックします。F9 キーを押すか、そのコード部分の横の青いバーをクリックすることでも、同じことが行えます。
7. **リストビュー：**スクリプトが、スクリプト名、キャスト名、キャスト番号およびタイプに基づいてリストに列で表示されます。列でリストをソートするには、ソートに使用する項目の列見出しをクリックします。
8. ツリービューで新しいスクリプトを作成するには、スクリプトタイプを右クリックし、ポップアップメニューから「<script type> を新規追加」を選択します。スクリプトエディタにスクリプトの名前を入力すると、そのスクリプト名がリストに表示されます。
9. スクリプトを保存します。
10. 全スクリプトをリコンパイルアイコンをクリックします。

キャストメンバスクリプトは、ツリービューではキャストメンバスクリプトフォルダ内にリストされ、リストビューではスクリプト名として表示されます。これらのスクリプトは特定のキャストメンバに割り当てられており、実際にはキャストメンバではないため、割り当て先のキャストメンバを削除した場合にしかスクリプトブラウザから削除されません。



スクリプトを作成したり開いたりするための上記以外の方法について詳しくは、71 ページの「一般的な作業の実行」を参照してください。

注意：スクリプトタブを閉じるには、タブの × ボタンをクリックするか、タブ領域を右クリックしてから「<script type:script name> を閉じる」オプションを選択します。

複数のスクリプトの開き方と閉じ方

スクリプトウィンドウでは、複数のスクリプトを別々のタブで開くことができます。スクリプトウィンドウに納まるタブの数は限られているため、一部のタブが非表示になる場合もあります。非表示になっているタブに移動するには、スクリプト参照タブアイコンの「>」と「<」を使用します。

複数のスクリプトウィンドウを開く

❖ 次のいずれかの操作を行います。

- スクリプトウィンドウから、ウィンドウ／新規スクリプトウィンドウを選択します。
- Alt+W+N キーを押します。

Alt+W+N ショートカットキーを使用すると、任意のアクティブウィンドウの追加インスタンスを開くことができます。例えば、ベクトルウィンドウを開いていて、Alt+W+N ショートカットキーを使用すると、ベクトルウィンドウがもう 1 つ開かれます。

スクリプトタブを閉じる

1 スクリプトウィンドウで、閉じるタブをクリックします。

2 次のいずれかの操作を行います。

- タブをクリックしてから（アクティブタブになっていなかった場合）、× ボタンをクリックします。
- タブまたはタブの横の領域を右クリックし、「<script type:script name> を閉じる」を選択します。

スクリプトウィンドウの環境設定

スクリプトウィンドウでは、テキストのフォントを変更したり、様々なコードコンポーネントごとに異なるカラーを定義することができます。スクリプトウィンドウに表示するテキストのデフォルトのフォントや、様々なコードエレメントのカラーを変更するには、スクリプトウィンドウの環境設定を使用します。[オートカラー] をオフにしないかぎり、コードエレメントはタイプ別に自動的に色分けされます。

エクスプローラパネルを他のウィンドウで表示するには、スクリプトエディタとエクスプローラパネルの間にある分割バーの矢印アイコンをクリックします。

スクリプトウィンドウを環境設定する

1 編集／環境設定／スクリプトを選択します。

2 デフォルトのフォントを選択するには、[フォント] ボタンをクリックし、[フォント] ダイアログボックスから設定を選択します。

3 スクリプトウィンドウに表示するテキストのデフォルトのカラーを選ぶには、[カラー] メニューからカラーを選択します。

4 スクリプトウィンドウのバックカラーを選ぶには、[背景] カラーメニューからカラーを選択します。

5 新しいスクリプトウィンドウで特定のコードエレメントを自動的に色分けするには、「オートカラー」を選択します。このオプションは、デフォルトでオンに設定されています。「オートカラー」をオフにすると、すべてのテキストがデフォルトのカラーで表示されます。

6 新しいスクリプトウィンドウでスクリプトを適切なインデントで自動的にフォーマットするには、「自動フォーマット」を選択します。このオプションは、デフォルトでオンに設定されています。

   注意：オートカラーと自動フォーマットは、JavaScript シンタックスのコードには適用されません。このため、JavaScript シンタックスを使用してスクリプトをオーサリングしている場合は、スクリプトウィンドウの「オートカラー」ボタンと「自動フォーマット」ボタンは無効になり、function、var、this などの用語もデフォルトのテキストカラーで表示されます。

7 新しいスクリプトウィンドウでスクリプトの行番号を表示するには、「行番号」を選択します。このオプションは、デフォルトでオンに設定されています。

8 [オートカラー] がオンの場合は、対応するカラーメニューから次のコードエレメントのカラーを選択します。

   - キーワード
   - コメント
   - リテラル
   - カスタム (独自のコードで定義する項目)

9 行番号の列のバックカラーを変更するには、行番号カラーメニューをクリックし、新しいカラーを選択します。

10 デバッガウィンドウでコールスタック、変数およびウォッチャーペインの位置を変更するには、デバッガペインメニューから「上」、「下」、「左」または「右」を選択します。

11 デフォルトのスクリプトタイプポップアップメニューから「Lingo」または「JavaScript」を選択します。エクスプローラパネルを開いたときには、ここで選択したオプションがデフォルトとして使用されます。

   注意：これは、Director を一旦閉じてから開き直した後にも保持されるアプリケーションレベルの変更です。

12 エクスプローラパネルをスクリプトウィンドウ内のスクリプトエディタに隣接する別の場所へ移動するには、エクスプローラパネルポップアップメニューから「左」、「上」、「右」または「下」を選択します。デフォルトでは、エクスプローラパネルはスクリプトエディタの左側に表示されます。

注意：スクリプトをデバッグしているときには、スクリプトエディタの別のインスタンスがデバッガパネルと一緒に開かれます。このドキュメントウィンドウは、開いている他のウィンドウと入れ替えることにより、アプリケーション内で並べ替えることができます。

スクリプトの削除

スクリプトの削除は、エクスプローラパネルを使用して行えます。

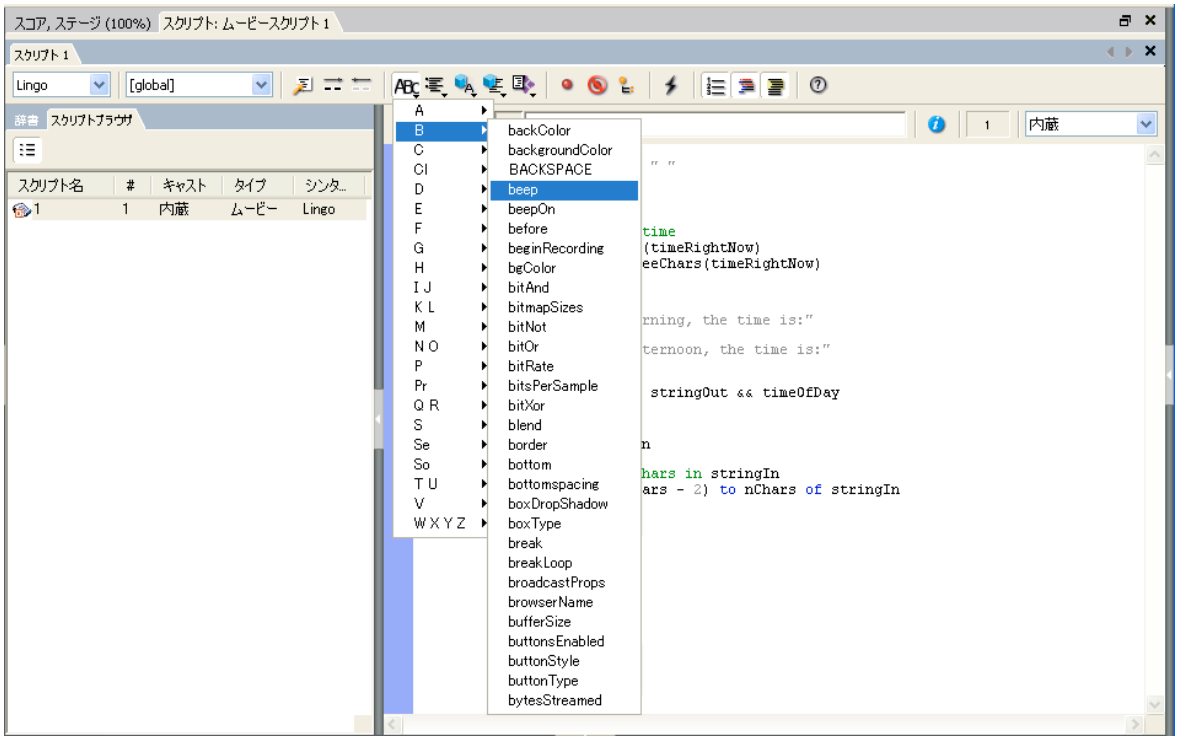
❖ 次のいずれかの操作を行います。

- スクリプトブラウザで、スクリプトエントリを右クリックして「削除」を選択します。
- スクリプトエクスプローラのリストビューで、スクリプトを選択して「削除」を押します。

注意：開いているスクリプトを削除すると、そのスクリプトに対応するタブもスクリプトウィンドウから削除されます。開いていて削除したスクリプトがムービー内の唯一のスクリプトだった場合は、スクリプトウィンドウが無効になり、スクリプトが表示されていたタブには「Script」というテキストが表示されます。テキストエディタを有効にするには、スクリプトエディタパネルのプラス記号（+）をクリックします。

一般的なスクリプト項目の挿入

スクリプトウィンドウには、一般的なスクリプト項目のポップアップメニューが用意されています。このメニューを使用することで、スクリプトにステートメントを挿入できます。メッセージウィンドウにも同じメニューが表示されます。



一般的なスクリプト項目の挿入

スクリプトウィンドウとメッセージウィンドウのどちらでも、特定のスクリプトにどちらのスクリプトシンタックスを使用するかを選択することができます。

スクリプトシンタックスを選択するには：

❖ スクリプトシンタックスポップアップメニューから「Lingo」または「JavaScript」を選択します。

スクリプトシンタックスを選択した後は、選択したシンタックスでコードを入力してください。選択していないシンタックスで記述したスクリプトをコンパイルしようとすると、スクリプトエラーが発生します。

スクリプトウィンドウにスクリプトを入力するときには、「コメントを設定」ボタンと「コメントを解除」ボタンを使用して、コードの単一行でも複数の行でもコメント化したり、コメント解除したりできます。「コメントを設定」ボタンと「コメントを解除」ボタンを使用すると、選択したスクリプトシンタックスに応じて、そのシンタックスでの正しいコメントマーカが表示されます。Lingo では、二重ハイフン（--）、JavaScript シンタックスでは二重スラッシュ（//）が使用されます。

コードをコメント化する

❖ コメント化するコード行をハイライト表示にし、「コメントを設定」をクリックします。

注意：JavaScript シンタックスコードで「コメントを設定」ボタンを使用して複数の行をコメント化すると、各行の先頭に二重スラッシュが挿入されます。コメント化するコードの先頭行の前に /* を挿入し、最終行の後に */ を挿入することでも複数行をコメント化できますが、これは手動で行わなければなりません。

コードをコメント解除する

❖ コメントでなくするコード行をハイライト表示にし、「コメントを解除」をクリックします。

スクリプトウィンドウとメッセージウィンドウの両方に、以下のメニューがあります。

- [Lingo (アルファベット順)] メニューには、3D Lingo 以外のエレメントがアルファベット順に表示されます。
- [Lingo (カテゴリ別)] メニューには、エレメントが使用頻度に応じてカテゴリ別に表示されます。このメニューにも 3D Lingo は含まれていません。
- [3D Lingo (アルファベット順)] メニューには、すべての 3D Lingo エレメントがアルファベット順に表示されます。
- [3D Lingo (カテゴリ別)] メニューには、すべての 3D Lingo エレメントが用途に応じてカテゴリ別に表示されます。
- Xtra のスクリプティングポップアップメニューには、Adobe 製の Xtra 拡張機能かサードパーティ製の拡張機能かに関係なく、見つかったすべてのスクリプト作成 Xtra 拡張機能のメソッドとプロパティが表示されます。

注意：Xtra のスクリプティングポップアップメニューにリストされるスクリプト作成 Xtra 拡張機能は、Interface() メソッドをサポートしているものだけで、実際にその名前がポップアップメニューに表示されます。3D や DVD などの一部のキャストメンバメディアタイプも Interface() メソッドをサポートしていますが、これらは Director ではスクリプト作成 Xtra 拡張機能として実装されていないため、Xtra のスクリプティングポップアップメニューには表示されません。

ポップアップメニューからエレメントを選択すると、スクリプトウィンドウの挿入ポイントにそのエレメントが挿入されます。

エレメントに追加パラメータが必要な場合は、必要な追加情報の位置にプレイーホルダー名が表示されます。複数のパラメータが必要な場合には、最初のものがハイライト表示されるため、入力するだけで置き換えられます。他のパラメータは、自分で選択して変更する必要があります。

一部のキャストメンバタイプとスクリプト作成 Xtra 拡張機能には、ポップアップメニューに表示されないスクリプト項目が含まれています。これらのメンバタイプと Xtra 拡張機能には、独自のマニュアルが用意されていることが多く、一部の情報は Director 内で入手することができます。

利用可能な Xtra 拡張機能のリストを表示する

❖ メッセージウィンドウで put(_player.xtraList) または trace(_player.xtraList) を発行します。

利用可能なスクリプト作成 Xtra 拡張機能のリストを表示する

❖ メッセージウィンドウで、put(_player.scriptingXtraList) または trace(_player.scriptingXtraList) を発行します。

Xtra 拡張機能のメソッドとプロパティのリストを表示する

❖ エクストラスクリプティングポップアップメニューで、Xtra 拡張機能をポイントし、サブメニューの「インターフェイスを表示」をクリックします。その Xtra 拡張機能のメソッドとプロパティが、メッセージウィンドウに表示されます。

テキストの入力と編集

スクリプトウィンドウでのテキストの入力および編集は、他のフィールドでのテキストの入力および編集に似ています。

スクリプトウィンドウで実行する一般的な編集作業は次のとおりです。

- 単語を選択するには、その単語をダブルクリックします。
- スクリプト全体を選択するには、編集メニューから「すべてを選択」を選択します。
- 改行するには、Enter キー (Windows) または Return キー (Macintosh) を入力します。
- Lingo では、継続記号を挿入して長いコード行を折り返すには、ソフト改行を挿入する位置で Alt+Enter キー（Windows®）または Option+Return キー（Mac®）を押します。ステートメントが次の行に続いていることを示す継続記号 (\) が表示されます。

  JavaScript シンタックスでは、長いコード行を折り返すには、Enter キー（Windows）または Return キー（Mac）を押して通常の改行を挿入します。JavaScript シンタックスのスクリプトでは、Lingo の継続記号が入っているとスクリプトエラーが発生します。
- 現在のスクリプトでのハンドラの場所を見つけるには、スクリプトウィンドウのハンドラへ移動ポップアップメニューからハンドラの名前を選択します。
- 変更を加えたスクリプトをコンパイルするには、スクリプトウィンドウの「全スクリプトをリコンパイル」ボタンをクリックするか、またはスクリプトウィンドウを閉じます。スクリプトを編集すると、スクリプトウィンドウのタイトルバーにアスタリスクが表示されて、スクリプトをリコンパイルする必要があることが示されます。
- ムービー内のすべてのスクリプトをコンパイルするには、制御メニューの「全スクリプトをリコンパイル」を選択します。
- スクリプトを適切なインデントで再フォーマットするには、スクリプトウィンドウで Tab キーを押します。

  シンタックスが正しければ、ステートメントが自動的にインデントされます。行が適切にインデントされない場合は、その行のシンタックスに問題があります。
- スクリプトウィンドウをもう 1 つ開くには、スクリプトウィンドウで Alt キー（Windows）または Option キー（Mac）を押しながら「新規キャストメンバ」ボタンをクリックします。これは、長いスクリプトの 2 つの部分を同時に編集する場合に便利です。
- 行番号の表示 / 非表示を切り替えるには、「行番号」ボタンをクリックします。
- オートカラーを切り替えるには、[オートカラー] ボタンをクリックします。オートカラーは、Lingo エレメント (プロパティ、コマンドなど) をタイプごとに異なるカラーで表示します。
- 自動フォーマットのオン / オフを切り替えるには、[自動フォーマット] ボタンをクリックします。自動フォーマットは、改行を追加するか、または Tab キーを押すたびに、スクリプトに適切なインデントを追加します。

  注意：オートカラーと自動フォーマットは、JavaScript シンタックスのコードには適用されません。このため、JavaScript シンタックスを使用してスクリプトをオーサリングしている場合は、スクリプトウィンドウの「オートカラー」ボタンと「自動フォーマット」ボタンは無効になり、function、var、this などの用語もデフォルトのテキストカラーで表示されます。

スクリプト内のハンドラとテキストの検索

ハンドラの検索、およびテキストやハンドラの検索と編集には、[編集] メニューの [検索] コマンドが便利です。

スクリプト内のハンドラを検索する

1 編集／検索／ハンドラを選択します。

[検索 ハンドラ] ダイアログボックスが表示されます。

[検索 ハンドラ] ダイアログボックスの左端の列には、ムービーで使用している各ハンドラの名前が表示されます。真ん中の列には、そのハンドラのスクリプトに関連付けられているキャストメンバの数が、キャストメンバの名前と一緒に表示されます。右端の列には、そのキャストメンバが入っているキャストライブラリがリストされます。

2 検索するハンドラを選択します。

3 [検索] ボタンをクリックします。

スクリプトウィンドウにハンドラが表示されます。

スクリプトウィンドウの上部にあるタイトルバーには、スクリプトの種類が表示されます。

スクリプト内のテキストを検索する

1 スクリプトウィンドウをアクティブにします。

2 編集／検索／テキストを選択します。

検索 テキストダイアログボックスが表示されます。

3 検索するテキストを [検索] フィールドに入力し、[検索] ボタンをクリックします。

デフォルトでは、検索では大文字と小文字は区別されません。例えば、ThisHandler、thisHandler、および THISHANDLER はすべて同じものとして検索されます。「大文字と小文字を区別する」チェックボックスをオンにすると、検索で大文字と小文字が区別されるようになります。

検索するキャストメンバを指定する

❖「検索範囲 : スクリプト」の下にある適切なオプションを選択します。

最後まで検索した後に最初から検索を続ける

❖ [くり返し検索] オプションを選択します。

単語の一部ではなく、完全に一致する単語だけの検索

❖ [単語が一致したとき] オプションを選択します。

「検索文字」フィールドで指定されているテキストに一致する次のテキストを検索する

❖ 編集／次を検索を選択します。

選択範囲内のテキストから一致するテキストを検索する

1 テキストを選択します。

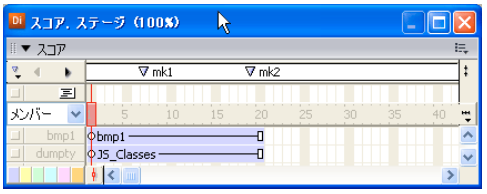
2 編集／検索／選択範囲を選択します。

一般的な作業の実行

スクリプトの作成や割り当て、およびスクリプトを開くための一般的な作業手順は次のとおりです。

フレームビヘイビアー（フレームに割り当てられるスクリプト）を作成する

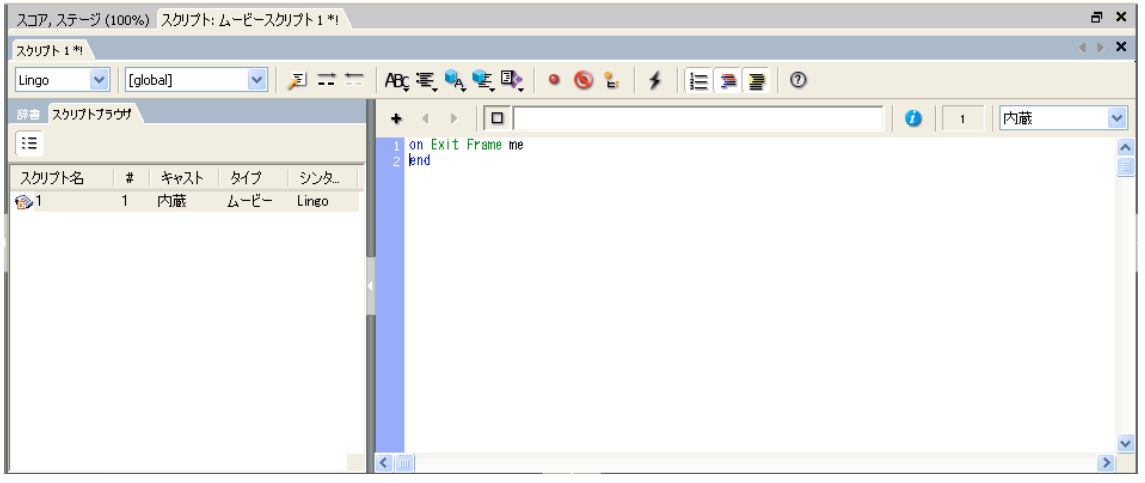
❖ ビヘイビアーを割り当てるフレームのビヘイビアーチャネルをダブルクリックします。



ビヘイビアチャネル

新しいビヘイビアーを作成すると、現在のキャストウィンドウで利用できる最初の場所のキャストライブラリ番号がそのビヘイビアーに割り当てられます。

新しいフレームビヘイビアーを作成すると、既に Lingo の on exitFrame ハンドラが入った状態のスクリプトウィンドウが開きます。最初の行には on exitFrame というコードが入っており、その後ろに点滅する挿入ポイントの置かれた行が続き、その後に end というコードの入った行が続きます。これにより、一般的なビヘイビアーをフレームに短時間で簡単に割り当てることができます。このハンドラを JavaScript シンタックスで使用できるようにするには、on exitFrame を function exitFrame() { に置き換え、end を } に置き換えます。



最も共通するフレームビヘイビアーの 1 つは、再生ヘッドを 1 つのフレーム内でループさせるものです。このビヘイビアーは、ユーザによるボタンのクリックや、デジタルビデオやサウンドの再生終了を待っている間、ムービーを 1 つのフレームで再生し続ける場合に便利です。

再生ヘッドを 1 つのフレーム内に維持する

❖ フレームビヘイビアーで、on exitFrame ステートメント（Lingo）または function exitFrame() ステートメント（JavaScript シンタックス）の次の行に以下のステートメントを入力します。

```
-- Lingo syntax
_movie.go(_movie.frame)

// JavaScript syntax
_movie.go(_movie.frame);
```

ムービーオブジェクトの frame プロパティは、現在再生ヘッドが入っているフレームを指しています。このステートメントは、再生ヘッドを「現在選択しているフレームに戻す」ように指示します。

スプライトビヘイビアー（スプライトに割り当てられるスクリプト）を作成する

❖ スコアまたはステージで、ビヘイビアーを割り当てるスプライトを選択します。次に、ウィンドウ／ビヘイビアインスペクタを選択し、ビヘイビアポップアップメニューから「新規ビヘイビア」を選択します。

新しいスプライトビヘイビアーを作成すると、既に Lingo の on mouseUp ハンドラが入った状態のスクリプトウィンドウが開きます。最初の行には on mouseUp というコードが入っており、その後ろに点滅する挿入ポイントの置かれた行が続き、その後に end というコードの入った行が続きます。これにより、スプライトに共通のビヘイビアーを即座に割り当てることが容易になります。このハンドラを JavaScript シンタックスで使用できるようにするには、on mouseUp を function mouseUp() { に置き換え、end を } に置き換えます。

ビヘイビアーを編集するために開く

1 キャストウィンドウでビヘイビアーをダブルクリックします。

   ビヘイビアインスペクタが表示されます。

2 ビヘイビアインスペクタで [スクリプトウィンドウ] アイコンをクリックします。

   スクリプトウィンドウにビヘイビアーが表示されます。

または、スクリプトウィンドウを開いて、目的のビヘイビアーが表示されるまでスクリプトを順番に移動するという方法もあります。

スコア内からビヘイビアーを削除する

❖ 場所を選択し、プロパティインスペクタの [ビヘイビア] タブに表示されたリストからスクリプトを削除します。

既存のビヘイビアーをスプライトまたはフレームに割り当てる

❖ 次のいずれかの操作を行います。

- ビヘイビアーをキャストからスコア内のスプライトまたはフレームにドラッグします。複数のスプライトの場合は、ステージ上のスプライトにドラッグします。
- スコアで、ビヘイビアーを割り当てるスプライトまたはフレームを選択します。次に、ウィンドウ／ビヘイビアインスペクタを選択し、ビヘイビアポップアップメニューから既存のビヘイビアを選択します。

ムービースクリプト（スプライトに割り当てられるスクリプト）を作成する

次のいずれかの操作を行います。

- スクリプトエクスプローラをツリービューで使用してムービースクリプトを作成するには：
  1 ムービースクリプトを追加するキャストの下のムービーノードを選択します。
  2 右クリックして、「新しいムービースクリプトを追加」を選択します。
- スクリプトウィンドウの現在のスクリプトがムービースクリプトである場合は、スクリプトウィンドウの [新規スクリプト] ボタンをクリックします。[新規スクリプト] ボタンをクリックすると、常に現在のスクリプトと同じ種類のスクリプトが作成されます。
- スクリプトウィンドウの現在のスクリプトがムービースクリプトではない場合は、「新規スクリプト」ボタンをクリックし、プロパティインスペクタの「スクリプト」タブにあるタイプポップアップメニューでスクリプトのタイプを変更します。

- キャスト、スコア、またはステージでスプライトもスクリプトも選択していなかった場合は、新しいスクリプトウィンドウが表示されます。このウィンドウは、デフォルトで新しいムービースクリプトを作成します。

ムービースクリプトまたは親スクリプトを編集するために開く

❖ スクリプトエクスプローラを使用してスクリプトを開いた後に、キャストウィンドウでスクリプトをダブルクリックします。

スクリプトのタイプを変更する

1 キャストウィンドウでスクリプトを選択するか、スクリプトウィンドウでスクリプトを開きます。

2 プロパティインスペクタの「スクリプト」タブをクリックし、スクリプトタイプポップアップメニューからスクリプトのタイプを選択します。

スクリプトウィンドウで前後にあるスクリプトを表示する

❖ スクリプトウィンドウの上部にある [前のキャストメンバ] および [次のキャストメンバ] の矢印を使用して、前後のスクリプトを表示します。

スクリプトを複製する

❖ キャストウィンドウでスクリプトを選択して、[編集] メニューから [複製] を選択します。

特定のキャストメンバから作成するすべてのスプライトに自動的に割り当てられるスクリプトを作成するには、そのキャストメンバ自体にスクリプトを割り当てます。

キャストメンバに割り当てられているスクリプトを作成する、または既存のスクリプトを開く

❖ 次のいずれかの操作を行います。

- キャストウィンドウでキャストメンバを右クリックするか（Windows）、Control キーを押したままクリックし（Mac）、コンテキストメニューから「キャストメンバスクリプト」を選択します。
- キャストウィンドウでキャストメンバを選択し、[キャストメンバスクリプト] ボタンをクリックします。

リンクされたスクリプトの使用

内部キャストメンバとして格納されたスクリプトに加えて、スクリプトを外部テキストファイルに保存し、Director ムービーにリンクすることもできますこれらのリンクされたスクリプトは、Director ムービーに読み込むことのできるリンクされた画像やデジタルビデオファイルに似ています。

リンクされたスクリプトを利用する利点には次のものがあります。

- 1 人が Director ファイルで作業している間に、もう 1 人がスクリプトで作業をすることができます。
- スクリプトを他の人と簡単に交換できます。
- Microsoft® Visual SourceSafe® や Perforce Software 社の Perforce® といったソースコード管理アプリケーションで、Director ファイルからスクリプトを独立して管理することができます。このようなアプリケーションは、同じ Director プロジェクトで作業している複数のプログラマが、お互いのコードに上書きすることを防ぎます。

リンクされたスクリプトは、オーサリング時にのみ使用されます。実行時には、Director プロジェクタおよび Adobe® Shockwave® Player では、ムービー内に格納されたスクリプトデータの特別な内部コピーが使用されます。そのため、リンクされたスクリプトはムービーと共に配信する必要がなく、エンドユーザがコピーすることもできません。

スクリプトをリンクされたテキストファイルとして読み込む

1 ファイル／読み込みを選択します。

2 読み込むファイルの種類としてスクリプトを選択します。

3 読み込むスクリプトファイルを選択します。

拡張子が .txt、.ls または .js のファイルを読み込むことができます。.ls という拡張子は、Director のリンクされたスクリプトの拡張子です。

読み込むファイルのリストを作成するには、[追加] および [すべてを追加] ボタンを使用します。これは、複数の場所からスクリプトを読み込む場合に特に便利です。

4 [メディア] ポップアップメニューから [外部ファイルにリンク] を選択します。

5 「読み込み」をクリックします。

通常、リンクされたスクリプトは Director のスクリプトウィンドウで編集できます。変更内容は、Director ムービーを保存するたびに外部ファイルに書き込まれます。リンクされたスクリプトを UNIX® サーバから読み込んだ場合、UNIX の行末コードは保持されます。テキストファイルがロック状態のスクリプトを読み込んだ場合は、Director でそのスクリプトを編集することはできません。

スクリプトウィンドウでは、カスタムテキストカラーをリンクされたスクリプトに適用することはできません。ただし、スクリプトのオートカラーはリンクスクリプトにも適用できます。

内部スクリプトキャストメンバを、リンクされた外部スクリプトキャストメンバに変更する

1 内部キャストメンバを選択し、プロパティインスペクタの「スクリプト」タブをクリックします。

2 [リンク] をクリックします。

3 [名前を付けて保存] ダイアログボックスにスクリプトファイルの名前を入力します。

4 「保存」をクリックします。

リンクされたスクリプトを編集した後で再ロードする

❖ メンバオブジェクトの unload() メソッドを使用します。

リンクされたスクリプトを Director 以外で編集した場合は、メッセージウィンドウで unload() メソッドを使用して再ロードすることができます。次のステートメントは、スクリプトメンバ myScript をアンロードした後に再ロードします。

```
-- Lingo syntax
member("myScript").unload()

// JavaScript syntax
member("myScript").unload();
```

第 4 章：Director でのスクリプトのデバッグ

スクリプトのデバッグについて

スクリプトは、常に最初から思ったとおりに動作するとは限りません。スクリプトには、単語の綴りの間違いやスクリプトの一部の欠落といったシンタックスエラーがよく見られます。また、スクリプトが動作しても、思ったとおりの結果が得られない場合もあります。スクリプトを記述する際にはほぼ必ず間違いやバグが生じるため、マルチメディアタイトルを作成する場合にはデバッグに十分な時間をかける必要があります。

スクリプトの記述に関するスキルが上達してくると、1 つの分野をマスターしても、他の分野の習得を開始するにつれて、様々な種類の問題に遭遇することでしょう。しかし、ここで説明する基本的なトラブルシューティングの方法は、慣れていないユーザにも上級ユーザにも役に立ちます。

スクリプトのバグを修正する最良の方法は状況によって異なります。問題を解決するのに、いつも決まった方法があるわけではありません。次のように、様々なツールや方法を使用する必要があります。

- ムービー内のスクリプトが相互に作用する仕組みの概要と理解
- 一般的なデバッグ方法の習熟と実践

次のツールは、スクリプトの問題を識別するのに役立ちます。

- **メッセージウィンドウ**には、トレースをオンにすると、再生されるフレームのレコードとムービーで実行されるハンドラが表示されます。
- **デバッガウィンドウ**には、グローバル変数の値、現在実行中のスクリプトのプロパティ、現時点までに実行された一連のハンドラ、および選択した変数や式の値が表示されます。
- **スクリプトウィンドウ**では、コメントの入力、スクリプトへの停止点の挿入、オブジェクトインスペクタに値が表示される変数の選択などを行うことができます。
- **オブジェクトインスペクタ**を使用すると、選択したオブジェクトおよびプロパティの値の表示や設定を行うことができます。

スクリプト記述時の適切な習慣

スクリプトを記述する際に、適切な習慣を身に付けていると、スクリプトの記述に関する多くの問題を当初から回避することができます。

- スクリプトを小さなステートメントセットに分割して記述し、それぞれを作成するたびにテストするようにします。これにより、潜在的な問題が隔離され、識別が容易になります。
- スクリプトのステートメントの意図された実行内容や、スクリプトの値の目的などを説明するコメントを挿入します。これにより、スクリプトを後で見直す場合や、作成者以外のだれかが作業する場合に、スクリプトを理解しやすくなります。例えば、次のステートメントに記述されたコメントは、if...then 構造と繰り返しループの目的を明確にしています。

```
-- Lingo syntax
-- Loop until the "s" key is pressed
repeat while not(_key.keyPressed("s"))
    _sound.beep()
end repeat

// JavaScript syntax
// Loop until the "s" key is pressed
while(!_key.keyPressed("s")) {
    _sound.beep();
}
```

- スクリプトのシンタックスが正しいことを確認します。スクリプトウィンドウのポップアップメニューを使用して、事前にフォーマットされたスクリプトエレメントを挿入します。このリファレンスの API に関するトピックを頼りに、ステートメントが正しく設定されていることをチェックします。
- 変数の目的を示す変数名を使用します。例えば、数値を含む変数には、ABC ではなく newNumber のような名前を付けます。

基本的なデバッグ方法

デバッグには、標準の段階的な手順ではなく計画と分析が必要です。この項では、Lingo や JavaScript のシンタックスだけでなく、あらゆるコードのデバッグに使用する、基本的なデバッグ方法について説明します。

ムービーを大幅に変更するときには、その前に常にバックアップコピーを作成します。ムービーの各段階を追跡するには、fileName_01.dir、fileName_02.dir、fileName_03.dir のように、ムービーに付ける名前を 1 ずつ大きくしていくと便利です。

問題の識別

明らかなことですが、デバッグにおいて最初に行うのは問題の識別です。ボタンが間違った動作をするのかを確認します。ムービーが間違ったフレームに移動するかを確認します。編集可能なはずのフィールドが編集できないかを確認します。

また、特定のスクリプトに期待する動作を明らかにし、そのスクリプトの実際の動作と比較します。これにより、目標を明確に定義し、その目標のどの部分が達成できていないかを確認することができます。

スクリプトまたはスクリプトの一部を他のムービーやサンプルからコピーした場合は、そのスクリプトが特定の条件に合わせて設計されたものかどうかをチェックします。スプライトチャネルが既に記述されていることが必要な場合があります。キャストメンバ名が、特定のスタイル表記方法に従う必要がある可能性もあります。

問題の場所の特定

問題の場所の特定を始めるには：

- チェーンをさかのぼって、予想とは異なる結果が生じ始めた場所を特定します。
- メッセージウィンドウを使用して、ムービーが通過するフレームとスクリプトが実行するハンドラをトレースします。
- スクリプトが実行すべきことを確認し、ステートメント内の何が問題に関係しているかを考えます。例えば、テキストキャストメンバが編集可能になるべきときにならない場合は、ムービーのどこでスクリプトがそのキャストメンバの editable プロパティを設定しているか（または設定していないか）を調べます。
- また、ステージ上でスプライトが思ったとおりに変化しない場合は、どこかで updateStage() メソッドが必要ではないかを検討します。

- 問題が発生するのは特定のコンピュータだけかどうかを確認します。問題が発生するのは、ディスプレイが千数百万色に設定されている場合だけなのかを確認します。コンピュータの何かが、アプリケーションを妨害している可能性があります。

スクリプトを一時停止してデバッガウィンドウを呼び出すブレイクポイントを行中に挿入すると、スクリプトの特定の行に集中することができます。これにより、スクリプトが続行する前にそのポイントで状況を分析する機会が与えられます。スクリプトにブレイクポイントを挿入する方法の詳細については、86 ページの「デバッガウィンドウでのデバッグ」を参照してください。

単純な問題の解決

バグを見つけるときは、まず単純な問題から調べることをお勧めします。

最初のデバッグテストは、スクリプトをコンパイルするときに実行されます。スクリプトをコンパイルするには、次のいずれかの操作を行います。

- スクリプトウィンドウで「全スクリプトをリコンパイル」をクリックします。
- 制御／全スクリプトをリコンパイルをクリックします。
- Shift+F8 キーを押します。
- スクリプトウィンドウを閉じます。

通常は、スクリプトをコンパイルするには、上記の方法のうち最初の 3 つを使用すると便利です。4 つ目の方法では、スクリプトをコンパイルするたびにスクリプトウィンドウを閉じる必要があります。

スクリプトをコンパイルすると、スクリプトに間違ったシンタックスが含まれている場合はエラーメッセージが表示されます。そのメッセージには通常、問題が最初に検出された行も表示されます。問題が最初に発見された行内の箇所には、疑問符が表示されます。

例えば、上記のエラーメッセージの 1 行目は、問題のスクリプトエラーがシンタックスエラーであることを示し、どのようなシンタックスエラーかを示しています。エラーメッセージの 2 行目には、シンタックスエラーが含まれている実際のコード行が表示されています。

シンタックスエラーの確認

スクリプトで最も一般的なバグは、おそらくシンタックスエラーです。スクリプトが機能しない場合は、まず次のことを確認してください。

- 項目の綴りが正しく、スペースが正しい位置にあり、必要な句読点が使用されていること。Director は、正しくないシンタックスを解釈できません。
- ステートメント内にあるキャストメンバ、マーカ、およびストリングの名前が引用符で囲まれていること。
- 必要なパラメータがすべて揃っていること。具体的なパラメータは、個々のエレメントによって決まります。エレメントに必要なその他のパラメータを確認するには、このリファレンスの API に関するトピックエントリを参照してください。

その他の単純なバグの確認

スクリプトのコンパイル時にエラーメッセージが表示されなかった場合も、バグが含まれている可能性があります。スクリプトが意図したとおりに動作しない場合は、次の点をチェックしてください。

- パラメータの値が正しいこと。例えば、beep() メソッドで発生させたい警告音の回数に誤った値を使用すると、間違った回数の警告音が鳴ります。
- 変数やテキストキャストメンバのコンテンツなどの変化する値に、必要な値が設定されていること。スクリプトウィンドウでオブジェクトの名前を選択して「オブジェクトの検査」をクリックするか、メッセージウィンドウで put() 関数や trace() 関数を使用すると、オブジェクトインスペクタに値を表示できます。

- スクリプトエレメントが思ったとおりの動作をすること。このリファレンスで API に関するトピックエントリを参照すると、エレメントの動作を調べることができます。
- 大文字と小文字が正しく使い分けられていること（スクリプトが JavaScript シンタックスで記述されている場合）。JavaScript シンタックスでは大文字と小文字が区別されます。つまり、すべてのメソッド、関数、プロパティ、および変数を参照するには、大文字と小文字を正しく使い分ける必要があります。

  大文字と小文字を間違って使用してメソッドや関数を呼び出そうとすると、スクリプトエラーが発生します。

  大文字と小文字を間違って使用して変数やプロパティにアクセスしようとすると、スクリプトエラーは発生しませんが、スクリプトが意図したとおりに動作しないことがあります。例えば、次の mouseUp ハンドラには、itemLabel プロパティにアクセスしようとするステートメントが含まれていますが、大文字と小文字を間違って使用しています。このスクリプトでは、スクリプトエラーが発生しない代わりに、大文字と小文字の間違った新しい変数が動的に作成されます。新しく作成された変数の値は undefined です。

```
// JavaScript syntax
function beginSprite() {
    this.itemLabel = "Blue prints";
}

function mouseUp() {
    trace(this.itemlabel) // creates the itemlabel property
}
```

スクリプトウィンドウでのデバッグ

スクリプトウィンドウには、スクリプトのデバッグに役立ついくつかの機能が含まれています。

スクリプトウィンドウを開く

❖ ウィンドウ／スクリプトを選択します。

現在のコード行をコメントにする

❖ [コメントを設定] ボタンをクリックします。

現在のコード行からのコメントの削除

❖ [コメントを解除] ボタンをクリックします。

現在のコード行にあるブレイクポイントのオンとオフの切り替え

❖ [ブレイクポイントの切り替え] ボタンをクリックします。

すべてのブレイクポイントをオフにする

❖ [全ブレイクポイントを無視] ボタンをクリックします。

選択した式または変数のオブジェクトインスペクタへの追加

❖ [オブジェクトの検査] ボタンをクリックします。

メッセージウィンドウでのデバッグ

メッセージウィンドウには、スクリプトコマンドをテストする方法や、ムービーの再生中にスクリプトで行われていることを監視する方法が用意されています。

メッセージウィンドウを開くには：

❖ ウィンドウ／メッセージを選択します。

メッセージウィンドウの管理

メッセージウィンドウには、入力ペインと出力ペインがあります。入力ペインは編集できます。出力ペインは読み取り専用です。出力ペインにテキストを表示するには、put() 関数または trace() 関数を呼び出すのが唯一の方法です。

入力ペインと出力ペインのサイズを調整するには、2 つのペインを分離する水平方向の仕切りをドラッグします。

出力ペインのサイズを変更するには：

❖ 水平方向の仕切りを新しい位置にドラッグします。

出力ペインを完全に非表示にするには：

❖ 水平方向の仕切りの中央にあるボタンをクリックします。

出力ペインを非表示にすると、実行されるスクリプトからの出力は入力ペインに表示されます。

非表示の出力ペインを表示するには：

❖ セパレータバーの中央のボタンをもう一度クリックします。

メッセージウィンドウの内容を消去するには：

❖ [クリア] ボタンをクリックします。

出力ペインが表示されている場合は、その内容が削除されます。

出力ペインが表示されていない場合は、入力ペインの内容が削除されます。

出力ペインの内容の一部を削除するには：

1 削除するテキストを選択します。

2 BackSpace キーまたは Delete キーを押します。

入力ペインまたは出力ペインのテキストをコピーするには：

1 テキストを選択します。

2 編集／コピーを選択します。

メッセージウィンドウでのスクリプトのテスト

Lingo および JavaScript シンタックスのステートメントをテストして動作を確認するには、メッセージウィンドウにコマンドを入力して、その結果を観察します。メッセージウィンドウにコマンドを入力すると、ムービーが再生されているかどうかに関係なく、そのコマンドがすぐに実行されます。

テストするステートメントを入力する前に、どちらのスクリプトシンタックスをテストするか（Lingo または JavaScript シンタックス）を選択する必要があります。

スクリプトシンタックスを選択するには：

❖ スクリプトシンタックスポップアップメニューから「Lingo」または「JavaScript」を選択します。

1 行のステートメントをテストするには：

1 メッセージウィンドウに、直接ステートメントを入力します。

2 Enter キー（Windows®）または Return キー（Macintosh®）を押します。ステートメントが実行されます。

ステートメントが有効であれば、メッセージウィンドウの下部にある出力ペインにステートメントの結果が表示されます。スクリプトが無効な場合は、警告メッセージが表示されます。

例えば、メッセージウィンドウに次のステートメントを入力します。

```
-- Lingo syntax
put(50+50)

// JavaScript syntax
trace(50+50);
```

Enter キー（Windows）または Return キー（Macintosh）を押すと、次のような結果が出力ペインに表示されます。

```
-- Lingo syntax
-- 100

// JavaScript syntax
// 100
```

また、メッセージウィンドウに次のステートメントを入力します。

```
-- Lingo syntax
_movie.stage.bgColor = 255

// JavaScript syntax
_movie.stage.bgColor = 255;
```

Enter キー（Windows®）または Return キー（Macintosh®）を押すと、ステージが黒になります。

複数行のコードをすべて一度にテストするには、ステートメントをメッセージウィンドウにコピー & ペーストするか、または各コード行の後で Shift+Return を押します。

コピー & ペーストで複数行のコードを実行するには：

1 コード行をクリップボードにコピーします。

2 メッセージウィンドウに空白行を入力します。

3 メッセージウィンドウの入力ペインにコードをペーストします。

4 最終行の末尾に挿入ポイントを置きます。

5 Ctrl+Enter （Windows）または Control+Return （Macintosh）を押します。挿入ポイントより上にある最初の空白行が検出され、その空白行より後にある各行が連続して実行されます。

複数行のコードを手動で入力するには：

1 メッセージウィンドウに空白行を入力します。

2 最初のコード行を入力します。

3 その行の末尾で Shift+Return を押します。

4 コードの最終行を入力するまで、手順 2 および 3 を繰り返します。

5 Ctrl+Enter （Windows）または Control+Return （Macintosh）を押します。挿入ポイントより上にある最初の空白行が検出され、その空白行より後にある各行が連続して実行されます。

ムービーを実行せずにハンドラをテストするには、ムービースクリプトウィンドウまたはビヘイビアスクリプトウィンドウにハンドラを記述し、メッセージウィンドウからそのハンドラを呼び出します。

ハンドラをテストするには：

1 前述の 2 つの手順で説明したとおりに、複数行のハンドラをメッセージウィンドウにコピー & ペーストするか、手動で入力します。

2 最終行の末尾に挿入ポイントを置きます。

3 Enter キー（Windows）または Return キー（Macintosh）を押します。ハンドラが実行されます。

ハンドラの put() ステートメントまたは trace() ステートメントからの出力はメッセージウィンドウに表示されます。

スクリプトウィンドウと同様に、メッセージウィンドウにもスクリプトコマンドのポップアップメニューがあります。ポップアップメニューの 1 つからコマンドを選択すると、そのコマンドが、入力する必要のある最初のパラメータが選択された状態でメッセージウィンドウに表示されます。スクリプト項目のカタログ全体に簡単にアクセスできるように、複数のメニューが用意されています。

ポップアップメニューには、次のものがあります。

- Lingo（アルファベット順）メニューには、3D Lingo を除くすべてのコマンドがアルファベット順に表示されます。
- Lingo（カテゴリ別）メニューには、3D Lingo を除くすべてのコマンドがカテゴリ別に表示されます。
- [3D Lingo (アルファベット順)] メニューには、すべての 3D Lingo がアルファベット順に表示されます。
- [3D Lingo (カテゴリ別)] メニューには、すべての 3D Lingo がカテゴリ別に表示されます。
- Xtra のスクリプティングメニューには、見つかったすべての Xtra のスクリプティング拡張のメソッドおよびプロパティが表示されます。Xtra の拡張が Adobe® 製かサードパーティ製かは問いません。

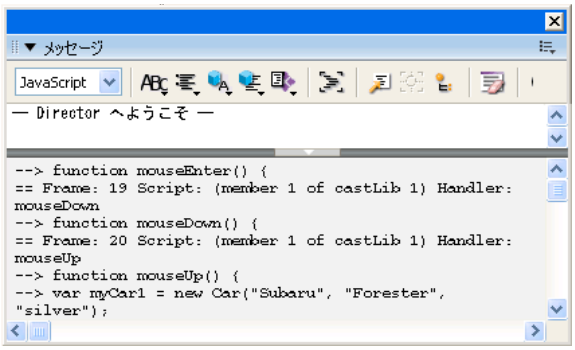
注意：Xtra のスクリプティングポップアップメニューにリストされるスクリプト作成 Xtra 拡張機能は、Interface() メソッドをサポートしているものだけで、実際にその名前がポップアップメニューに表示されます。3D や DVD などの一部のキャストメンバメディアタイプも Interface() メソッドをサポートしていますが、これらは Director ではスクリプト作成 Xtra 拡張機能として実装されていないため、Xtra のスクリプティングポップアップメニューには表示されません。

メッセージウィンドウでのスクリプトの監視

ムービーの再生中に実行されるステートメントのレコードが表示されるように、メッセージウィンドウの出力ペインを設定することができます。これは、コードの流れを追跡したり、特定のステートメントの結果を確認するのに便利です。それには次の 2 つの方法があります。

- メッセージウィンドウで「トレース」をクリックします。

- ムービーオブジェクトの traceScript プロパティを TRUE に設定します。



二重等号（==）に続くエントリは、ムービーが入ったフレーム、実行中のスクリプト、関数の結果や値を設定した結果など、ムービー内で発生したことを示します。

例えば、次の行はいくつかのことを示しています。

```
== Frame: 39 Script: 1 Handler: mouseUp
```

- ムービーがフレーム 39 に入ったこと。
- ムービーがスクリプト 1 (フレーム 39 に割り当てられた最初のスクリプト) を実行したこと。
- ムービーがフレームに入った後で、スクリプト 1 の mouseUp ハンドラを実行したこと。

矢印より後の二重ハイフンと右向きの山かっこ（-->）で構成されるエントリは、実行されたコード行を示します。例えば、

```
--> _sound.fadeOut(1, 5*60)
--> if leftSide < 10 then
--> if leftSide < 200 then
--> _movie.go("Game Start")
```

上記は、これらの Lingo ステートメントが実行されたことを示します。" ゲーム開始 " というマーカに再生ヘッドが移動しなかった理由を特定しようとしていると仮定します。--> _movie.go("Game Start") という行がメッセージウィンドウに表示されなかった場合は、前のステートメントの条件が予想したものではなかった可能性があります。

[トレース] ボタンをオンにすると、メッセージウィンドウの出力ペインには大量のテキストが表示されます。出力ペインの内容を削除するには、[クリア] ボタンをクリックします。出力ペインが表示されていない場合は、入力ペインの内容が削除されます。

変数やその他のオブジェクトの値を追跡するには、メッセージウィンドウでオブジェクトの名前を選択し、[オブジェクトの検査] ボタンをクリックします。そのオブジェクトがオブジェクトインスペクタに追加され、ムービーの再生中にその値が表示および更新されます。オブジェクトインスペクタについて詳しくは、83 ページの「オブジェクトインスペクタでのデバッグ」を参照してください。

デバッグモードのとき、メッセージウィンドウで変数を選択し、[ウォッチ] ボタンをクリックすると、変数の変化を追跡することができます。その変数がデバッガウィンドウのウォッチャーペインに追加され、デバッガウィンドウでの操作中にその値が表示および更新されます。ウォッチャーペインの詳細については、86 ページの「デバッガウィンドウでのデバッグ」を参照してください。

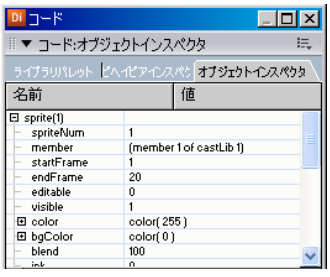
オブジェクトインスペクタでのデバッグ

オブジェクトインスペクタを使用すると、プロパティインスペクタに表示されない様々な種類のオブジェクトのプロパティを表示および設定できます。これらのスクリプトオブジェクトには、グローバル変数、リスト、親スクリプトから作成された子オブジェクト、すべての 3D キャストメンバプロパティ、スプライトプロパティ、スクリプト式などがあります。さらに、オブジェクトインスペクタには、スクリプトによる変更やスプライトのスコアプロパティに対する変更など、ムービーの再生中に生じるオブジェクトプロパティの変更が表示されます。このような実行時の変更は、ムービーの再生時にプロパティインスペクタには表示されません。

オブジェクトインスペクタで JavaScript 変数の値を表示するには、前に **var** を付けずに宣言する必要があります。

オブジェクトインスペクタを開くには：

❖ ウィンドウ／オブジェクトインスペクタを選択します。



オブジェクトインスペクタ

オブジェクトの構造について

オブジェクトインスペクタは、複雑なオブジェクトの構造を理解するのに非常に役立ちます。例えば、3D キャストメンバにはプロパティのレイヤーが複数あります。オブジェクトインスペクタには、これらのプロパティのネスト構造が視覚的に表現されるため、プロパティおよびそれらの相互関係が理解しやすくなります。Director におけるオブジェクトのプロパティ構造を理解することは、スクリプトを記述する際に重要です。

ムービーの再生中にプロパティの値の変化を観察できる機能は、ムービーで行われていることを理解するのに役立ちます。特に、記述したスクリプトに基づく値の変化を観察できるため、スクリプトのテストやデバッグの際に便利です。

Director のデバッガウィンドウにもこの情報は表示されますが、デバッガウィンドウが利用できるのはデバッグモードの場合のみです。デバッグの詳細については、91 ページの「高度なデバッグ方法」を参照してください。

表示できるオブジェクト

オブジェクトインスペクタに入力できるオブジェクトには、次のようなものがあります。

- スプライト：sprite(3) など
- キャストメンバ：member("3d") など
- グローバル変数：gMyList など
- 子オブジェクト：gMyChild など
- Adobe® Flash® オブジェクト：gMyFlashObject など（Director での Flash オブジェクトの使用の詳細については、Director ヘルプパネルの「Director の使い方」のトピックを参照してください。
- スクリプト式：sprite(7).blend など

オブジェクトの表示

オブジェクトインスペクタでオブジェクトを表示する方法には 3 つあります。オブジェクトインスペクタにアイテムを直接ドラッグするか、アイテムの名前を手動で入力するか、メッセージウィンドウおよびスクリプトウィンドウの「オブジェクトの検査」ボタンを使用します。

オブジェクトインスペクタにアイテムをドラッグするには：

❖ 次のいずれかの操作を行います。

- スコアウィンドウでスプライトを選択し、オブジェクトインスペクタにドラッグします。
- キャストウィンドウでキャストメンバを選択し、オブジェクトインスペクタにドラッグします。
- スクリプトウィンドウ、メッセージウィンドウ、またはテキストウィンドウでオブジェクトの名前を選択し、オブジェクトインスペクタにドラッグします。

オブジェクトインスペクタに手動でオブジェクトを入力するには：

1. オブジェクトインスペクタの [オブジェクト] 列で最初の空白セルをダブルクリックします。
2. そのセルにオブジェクトの名前を入力します。スクリプトでオブジェクトを参照するのと同じ名前を使用します。
3. Enter キー（Windows）または Return キー（Macintosh）を押します。オブジェクトにサブプロパティがある場合は、左側にプラス記号（+）が表示されます。
4. プラス記号をクリックします。その下に、オブジェクトのプロパティが表示されます。サブプロパティのあるプロパティには、左側にプラス記号が表示されます。それぞれのプラス記号をクリックすると、さらにサブプロパティが表示されます。

「オブジェクトの検査」ボタンを使用してオブジェクトを表示するには：

1. スクリプトウィンドウで、オブジェクトを参照するステートメントの一部をハイライトします。
2. スクリプトウィンドウで「オブジェクトの検査」をクリックします。オブジェクトにサブプロパティがある場合は、左側にプラス記号（+）が表示されます。
3. プラス記号をクリックします。その下に、オブジェクトのプロパティが表示されます。サブプロパティのあるプロパティには、左側にプラス記号が表示されます。それぞれのプラス記号をクリックすると、さらにサブプロパティが表示されます。

注意：オブジェクトインスペクタで多数のオブジェクトや大きな個々のオブジェクトを検査すると、オーサリング時（特に、「自動ポーリング」が有効な場合）に、パフォーマンス上の大きな問題が発生することがあります。例えば、10,000 個のエントリが含まれるリストを検査すると、表示の更新中に Director が停止することがあります。

オブジェクトのナビゲーション

オブジェクトインスペクタでは、キーボードの矢印キーでも移動できます。

アイテムのリスト内を上下に移動するには：

❖ 上下の矢印キーを使用します。

アイテムのサブプロパティの表示

❖ アイテムを選択して、右矢印キーを押します。

アイテムのサブプロパティの非表示

❖ アイテムを選択して、左矢印キーを押します。

自動ポーリングの使用

milliseconds や colorDepth, などのシステムプロパティが更新されるのは、オブジェクトインスペクタで「自動ポーリング」オプションがオンになっている場合だけです。「自動ポーリング」を使用するとプロセッサの負荷が増すため、オブジェクトインスペクタに多数のシステムプロパティを追加すると、ムービーのパフォーマンスが低下する可能性があります。

自動ポーリングをオンにするには：

1 オブジェクトインスペクタで、マウスの右ボタンをクリック（Windows）するか、または Control キーを押しながらクリック（Macintosh）します。オブジェクトインスペクタのコンテキストメニューが表示されます。

2 コンテキストメニューから [自動ポーリング] を選択します。[自動ポーリング] をオンにすると、コンテキストメニューの [自動ポーリング] 項目の隣にチェックマークが表示されます。

自動ポーリングをオフにするには：

❖ もう一度、コンテキストメニューから「自動ポーリング」を選択します。

オブジェクトまたはプロパティの値の変更

オブジェクトインスペクタでオブジェクトやプロパティの値を設定するには、オブジェクト名またはプロパティ名の右にあるボックスに新しい値を入力します。

オブジェクトまたはプロパティの値を設定するには：

1 アイテム名の右にある値をダブルクリックします。

2 アイテムに新しい値を入力します。

3 Enter キー（Windows）または Return キー（Macintosh）を押します。新しい値が設定され、直ちにムービーに反映されます。

アイテムの値として、スクリプト式を入力することもできます。例えば、sprite(3).locH の値を sprite(8).locH + 20 という式に設定することができます。

オブジェクトの削除

オブジェクトインスペクタからアイテムを削除することもできます。

オブジェクトインスペクタからアイテムを 1 つだけ削除するには：

❖ アイテムを選択して、Backspace キー（Windows）または Delete キー（Macintosh）を押します。

オブジェクトインスペクタの内容全体を消去するには：

❖ オブジェクトインスペクタでマウスの右ボタンをクリック（Windows）するか、または Control キーを押しながらクリック（Macintosh）し、コンテキストメニューから「すべてクリア」を選択します。

作業しているムービーとは別のムービーを開いても、オブジェクトインスペクタに入力したオブジェクトはそのまま残っています。これを利用すれば、同じムービーの異なるバージョンを簡単に比較できます。Director を終了すると、オブジェクトインスペクタのアイテムは失われます。

デバッガウィンドウでのデバッグ

デバッガウィンドウはスクリプトウィンドウの特別なモードです。このウィンドウには、スクリプト内の問題の原因を見つけるためのツールがいくつか用意されています。デバッガウィンドウを使用すると、コードの問題を引き起こしている部分をすばやく見つけることができます。デバッガウィンドウでは、スクリプトを 1 行ずつ実行すること、ネストされたハンドラをスキップすること、スクリプトのテキストを編集すること、変数や他のオブジェクトの値の変化を表示することなどができます。デバッガウィンドウのツールの使用法を習得すると、より効率的なプログラミングを行うのに役立ちます。

デバッガウィンドウでは、スクリプトのエラーを検出して訂正することができます。このウィンドウには、次のことを実行できるツールが用意されています。

- 現在のコード行を含むスクリプトの一部を表示します。
- 現在のハンドラに到達する前に呼び出された一連のハンドラを追跡します。
- 現在のハンドラの選択した部分を実行します。
- 現在のハンドラから呼び出した、ネストされたハンドラの選択した部分を実行できます。
- 調べているコードに関連するローカル変数、グローバル変数、またはプロパティの値を表示します。

デバッグモードへの切り替え

デバッガウィンドウにアクセスするには、スクリプトでブレイクを発生させる必要があります。ブレイクは、Director がスクリプトエラーを検出した場合、またはスクリプトのブレイクポイントに遭遇した場合に発生します。

スクリプトエラーが発生すると、[スクリプトエラー] ダイアログボックスが表示されます。このダイアログボックスには、発生したエラーのタイプに関する情報が表示され、スクリプトをデバッグするか、スクリプトウィンドウでスクリプトを編集するか、またはキャンセルするかが求められます。

デバッグモードに切り替えるには：

❖ 次のいずれかの操作を行います。

- [スクリプトエラー] ダイアログボックスで [デバッグ] をクリックします。
- スクリプトにブレイクポイントを配置します。

Director が実行中にブレイクポイントを検出すると、スクリプトの実行が停止され、スクリプトウィンドウがデバッグモードに切り替わります。ムービーはまだ再生されていますが、デバッガウィンドウを使用して Director に処理方法を指示するまで、スクリプトの実行は停止します。複数のスクリプトウィンドウを開いている場合は、ブレイクポイントのあるスクリプトが表示されているウィンドウが検索され、デバッグモードに切り替えられます。

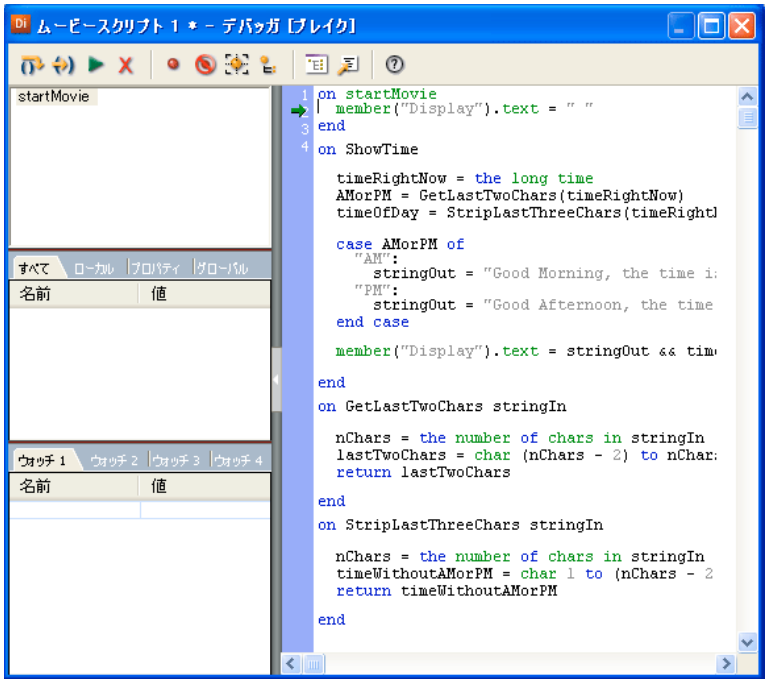
デバッガウィンドウを手前に表示するには、Ctrl+F11 を押します。

注意：ムービーの再生を開始した後にデバッガウィンドウに戻るには、最大化タブモードを終了する必要があります。この操作に Control+TAB キーを使用することはできません。

デバッガウィンドウを開くブレイクポイントを追加するには：

1. スクリプトウィンドウで、ブレイクポイントを追加するスクリプトを開きます。
2. ブレイクポイントを表示したいコード行の隣で、スクリプトウィンドウの左マージンをクリックします。または、コード行に挿入ポイントを置き、「ブレイクポイントの切り替え」をクリックします。この行の先頭でコードの実行が停止され、スクリプトウィンドウがデバッグモードに切り替わります。

Director がスクリプトエラーやブレイクポイントを検出したときにスクリプトウィンドウが表示されていると、スクリプトウィンドウがデバッガウィンドウに置き換えられます。



デバッガウィンドウ

デバッグを停止するには：

❖ 次のいずれかの操作を行います。

- デバッガウィンドウで [ラン] ボタンをクリックします。これにより、通常のスクリプトの実行が再開されます。
- デバッガウィンドウで [デバッグを停止] ボタンをクリックします。これにより、デバッグセッションとムービーの両方が停止します。

  デバッガウィンドウの代わりに、スクリプトウィンドウが再び表示されます。

デバッガウィンドウが開くと、現在のコード行が表示され、次に実行する選択肢がいくつか提示されます。

現在のコード行を確認するには：

❖ スクリプトペインでコードの行の隣にある緑の矢印を探します。

緑の矢印は現在の行を示しています。スクリプトペインで別のコード行をクリックしても選択することはできません。

デバッガウィンドウでのコールスタックの表示

コールスタックペインには、現在のコード行の前に実行された、ネストされたハンドラの配列が表示されます。この配列をコールスタックといいます。コールスタックは、デバッグ中にコードの構造を追跡するために使用します。特定のハンドラに関連付けられた変数を表示するには、コールスタックペインでハンドラ名をクリックします。変数は変数ペインに表示されます。

デバッガウィンドウでの変数の表示

デバッガウィンドウの変数ペインには、現在のハンドラに関連付けられた変数が表示されます。現在のハンドラは、スクリプトペインに表示されたハンドラであり、コールスタックペインに表示された最後のハンドラです。コールスタックの前のハンドラに関連付けられた変数を表示することもできます。スクリプトをステップ実行していくと、変数値に対する変更が赤で表示されます。スクリプトのステップ実行の詳細については、89 ページの「デバッガウィンドウでのスクリプトのステップ実行」を参照してください。

コールスタックのハンドラに関連付けられた変数を表示するには：

❖ コールスタックペインでハンドラの名前をクリックします。変数は変数ペインに表示されます。

変数ペインには、変数を表示するための 4 つのタブがあります。

[**すべて**] **タブ**では、現在のハンドラに関連付けられたグローバル変数とローカル変数がすべて表示されます。

[**ローカル**] **タブ**では、選択したハンドラに関連付けられたローカル変数のみが表示されます。

[**プロパティ**] **タブ**では、現在のスクリプトで宣言されているプロパティが表示されます。

[**グローバル**] **タブ**では、選択したハンドラに関連付けられたグローバル変数のみが表示されます。

変数ペインで変数を表示するには：

- 変数を名前で並べ替えるには、変数名の上に表示されている " 名前 " という語をクリックします。
- 変数を逆アルファベット順に並べ替えるには、" 名前 " という語をもう一度クリックします。

変数ペインでは、現在のハンドラのローカル変数およびグローバル変数の値を変更できます。現在のハンドラにないローカル変数の値は変更できません。

変数ペインで変数の値を変更するには：

1 [値] 列で変数の値をダブルクリックします。

2 変数の新しい値を入力します。

3 Enter キー（Windows）または Return キー（Macintosh）を押します。

デバッガウィンドウでのオブジェクトの表示

デバッガウィンドウのウォッチャーペインでは、現在のハンドラに関連付けられた変数やその他のデータオブジェクトだけでなく、他のハンドラに関連付けられたオブジェクトも表示できます。ウォッチャーペインにオブジェクトを追加すると、そのオブジェクトの値がスクリプトによって変化しても追跡できます。コード行の実行によってオブジェクトの値が変化すると、ウォッチャーペインに表示されるオブジェクト名のカラーが赤に変わります。

ウォッチャーペインには、そこに追加されたオブジェクトのみが表示されます。ウォッチャーペインにある 4 つのタブを活用して、オブジェクトをグループ別に整理することができます。

スクリプトペインに名前の表示されているオブジェクトをウォッチャーペインに追加するには：

1 スクリプトペインでオブジェクトの名前を選択します。

2 [ウォッチ] ボタンをクリックします。

スクリプトペインに名前が表示されていないオブジェクトをウォッチャーペインに追加するには：

1 ウォッチャーペインのオブジェクト列で最初の空白セルをダブルクリックします。

2 そのセルにオブジェクトの名前を入力し、Enter キー（Windows）または Return キー（Macintosh）を押します。

オブジェクトにプロパティがある場合は、オブジェクト名の隣にプラス記号（+）が表示されます。

オブジェクトのプロパティの表示

❖ オブジェクト名の隣にあるプラス記号をクリックします。

ウォッチャーペインを使用することで、オブジェクトをいくつかの方法で整理できます。

ウォッチャーペインでオブジェクトを整理するには：

❖ 次のいずれかの操作を行います。

- ウォッチャーペインでオブジェクトをソートするには、左側の列の上にある [名前] という列見出しをクリックします。列のオブジェクト名が、アルファベット順に表示されます。
- オブジェクトを逆アルファベット順にソートするには、[名前] という列見出しをもう一度クリックします。
- オブジェクトをグループ別に整理するには、ウォッチャーペインのタブを使用します。オブジェクトを特定のタブに追加するには、オブジェクトを追加する前に使用するタブをクリックして選択します。
- ウォッチャーペインのタブの内容を消去するには、タブを選択し、ウォッチャーペインでマウスの右ボタンをクリック（Windows）するか、Control キーを押したままクリック（Macintosh）して、「すべてクリア」を選択します。

デバッガウィンドウでのスクリプトのステップ実行

デバッガウィンドウにはスクリプトを少しずつ実行するツールが用意されているため、各コード行がムービーに与える影響をウォッチできます。コードを一度に 1 行ずつ実行することができ、またネストされたハンドラを 1 行ずつ実行するか一度に実行するかを選択することもできます。

緑の矢印で示されている現在のコード行だけを実行するには：

❖ [ステップ] ボタンをクリックします。

多くのハンドラには、他のハンドラの呼び出しステートメントが含まれています。このようなネストされたハンドラに注目するか、またはそれらを無視して現在のハンドラのコードに焦点を絞ることができます。

ネストされたハンドラが思ったとおりに実行されているという確信があり、現在のハンドラのコードに集中したい場合は、デバッガウィンドウでネストされたハンドラをステップオーバーして、直ちに現在のハンドラの次の行に進むこともできます。デバッガは、ネストされたハンドラをステップオーバーするときにそのハンドラを実行しますが、ハンドラのコードを表示しないか、またはネストされたハンドラ内で停止します。

ネストされたハンドラをステップオーバーするには：

❖ デバッガウィンドウで [ステップ] ボタンをクリックします。

「ステップ」ボタンは現在のコード行を実行し、その行が呼び出すネストされたハンドラを実行し、ハンドラの次の行で停止します。

ネストされたハンドラが思ったとおりに実行されるかが確かではなく、その動作を調べる必要がある場合は、デバッガウィンドウでネストされたハンドラを 1 行ずつ実行することもできます。

ネストされたハンドラを 1 行ずつ実行するには：

❖ デバッガウィンドウで [イントゥー] ボタンをクリックします。

「イントゥー」ボタンをクリックすると、まず現在のコード行が実行され、次に通常のフローに従い、その行から呼び出されるネストされたハンドラが実行されます。ネストされたハンドラの実行が終わると、デバッガウィンドウは上位のハンドラ内にある次のコード行で停止します。

デバッグが完了したら、いつでもデバッガウィンドウを終了できます。

通常のコードの実行を再開し、デバッガウィンドウを終了するには：

❖「ラン」ボタンをクリックします。

デバッガを終了し、ムービーの再生を停止するには：

❖「デバッグを停止」ボタンをクリックします。

デバッグモードでのスクリプトの編集

デバッグモードの場合でも、デバッガウィンドウで直接スクリプトを編集することができます。そのため、バグを見つけたらすぐに修正して、デバッグ作業を続行することができます。

デバッガウィンドウでスクリプトを編集するには：

1 スクリプトペインをクリックして、入力を始める位置に挿入ポイントを置きます。

2 スクリプトの変更内容を入力します。

特定のハンドラにジャンプするには、ハンドラの名前を選択し、「ハンドラへ移動」ボタンをクリックします。

3 スクリプトのデバッグと編集を完了したら、[デバッグを停止] ボタンをクリックします。スクリプトウィンドウがスクリプトモードに戻ります。

4 [全スクリプトをリコンパイル] ボタンをクリックします。

プロジェクタおよび Shockwave ムービーのデバッグ

ここでは、Adobe® Shockwave® コンテンツが含まれている Director プロジェクタおよびムービーの実行時におけるデバッグについて説明します。Shockwave コンテンツが含まれているプロジェクタおよびムービーをデバッグするには、メッセージウィンドウを使用するか、またはスクリプトエラーダイアログボックスを完全に有効にします。

メッセージウィンドウを使用してデバッグするには：

❖ Player オブジェクトの debugPlaybackEnabled プロパティを TRUE に設定します。

このプロパティを TRUE に設定すると、Shockwave コンテンツが含まれているプロジェクタまたはムービーを再生するとメッセージウィンドウ（Windows）またはメッセージテキストファイル（Macintosh）が開き、put() 関数または trace() 関数の呼び出し結果がこれらのフォーマットに出力されます。

ムービーの途中で debugPlaybackEnabled プロパティを FALSE に設定すると、メッセージウィンドウまたはテキストファイルが閉じ、セッション中に debugPlaybackEnabled を再び TRUE に設定しても、その再生セッション中は再び開くことができなくなります。

スクリプトエラーダイアログを完全に有効にしてデバッグするには：

❖ Shockwave コンテンツが含まれているプロジェクタまたはムービーの .ini ファイルで、DisplayFullLingoErrorText プロパティを 1 に設定します。

こうすると、スクリプトエラーダイアログボックスに、通常のエラーテキストの代わりにより詳細なエラーまたはテキストが表示されます。例えば、通常のエラーメッセージは次のように表示されます。

```
Script error: Continue?
```

DisplayFullLingoErrorText プロパティを 1 に設定すると、次のようなエラーメッセージが生成されます。

```
Script error: list expected
```

Shockwave コンテンツが含まれているプロジェクタまたはムービーの適切な .ini ファイルの作成または変更の詳細については、Director のインストールフォルダのルートにある Director.ini テンプレートファイルを参照してください。

高度なデバッグ方法

問題の識別が容易でない場合は、次の方法を試みます。

- どの部分に問題があるかを特定します。例えば、ボタンをクリックすると間違った結果になる場合は、そのボタンに割り当てられているスクリプトを調べます。

  スプライトが間違ったことを実行する場合は、そのスプライトのプロパティの値をチェックしてください。必要なときに必要な値に設定されているかを調べます。

- スクリプトのフローを把握します。ムービーのあるセクションが思ったとおりに動作しない場合は、まず頭の中でムービーのイベントの順番を追跡してみます。メッセージ階層で他のスクリプトを調べて、正しいハンドラが実行されているかどうかを確認します。
- メッセージウィンドウのトレースを追跡します。これにより、ムービーが通過するフレームやムービーの再生中に呼び出されるハンドラが表示されます。
- デバッガウィンドウの「ステップ」および「イントゥー」機能を使用して、結果が予想したものと同じかどうかを確認します。
- 変数と式を調べます。ムービー再生中の値の変化を分析します。値が間違った時点で変化するか、またはまったく変化しないかを確認します。同じ変数を複数のハンドラで使用している場合は、変数を使用する各ハンドラに、その変数がグローバルであると記述されていることを確認します。

  変数や式の値を追跡するには、デバッガウィンドウのウォッチャーペインまたはオブジェクトインスペクタにその値を表示します。

- 変更は一度に 1 つずつ行います。ハンドラに変更を加えることをためらう必要はありません。変更によって問題が解決されるか、または問題の指摘に役立つ結果が得られるかどうかを確かめます。

  ただし、新たな問題を発生させないように注意してください。変更は 1 つずつ行い、問題が解決されない場合は変更を元に戻します。問題を解決する前に変更を加え過ぎると、元の問題が何であったかが特定できなくなったり、新しい問題が生じる可能性さえあります。

- セクションを再度作成します。問題が見つからない場合は、そのセクションを最初から再度作成してみます。例えば、ポインタをスプライト上に移動しても、スプライトが思ったとおりに動作しない場合は、rollOver() メソッドを使用して、そのスプライトとハンドラのみを含む単純なムービーを作成します。

  スクリプトを単にコピー & ペーストしないでください。これによって問題がコピーされる場合があります。セクションを再度作成することで、ロジックを最も基本的なレベルで再構成し、思ったとおりに Director が機能しているかどうかを検証できます。再度作成したセクションが適切に機能しない場合は、そのセクションのロジックに間違いがある可能性が大きくなります。

  再度作成したセクションが適切に機能する場合は、そのセクションと元のムービーを比較して、両者の違いを確認します。再度作成したセクションを元の部分にコピーして、問題が修正されるかどうかを確認することもできます。

第 5 章：Director のコアオブジェクト

Director® のコアオブジェクトを使用すると、Director、プロジェクタ、Adobe® Shockwave® Player で利用できるコア機能にアクセスできます。コアオブジェクトには、Director プレーヤエンジン、ムービーウィンドウ、スプライト、サウンドなどが含まれます。コアオブジェクトは、これを通じてほぼすべての API およびその他のオブジェクトカテゴリにアクセスできるベースレイヤーを表します。ただし、Director のコア機能を拡張するスクリプトオブジェクトにはアクセスできません。

コアオブジェクト同士の関係と、コアオブジェクトと Director 内の他のオブジェクトとの関係を表す図は、43 ページの「オブジェクトモデル図」を参照してください。

キャストライブラリ

ムービー内の単一のキャストライブラリを表します。

ムービーは、1 つ以上のキャストライブラリで構成されます。キャストライブラリは、1 つ以上のキャストメンバで構成されます。キャストメンバは、ムービー内のサウンド、テキスト、グラフィックス、その他のムービーなどのメディアです。

キャストライブラリへの参照を作成するには、最上位の castLib() 関数を使用するか、ムービーオブジェクトの castLib プロパティを使用します。例えば、ムービーに scripts というキャストライブラリが含まれる場合、このキャストライブラリへの参照を作成するには、次のようにします。

- 最上位の castLib() メソッドを使用します。

```
-- Lingo syntax
libScript = castLib("scripts")

// JavaScript syntax
var libScript = castLib("scripts");
```

- ムービーオブジェクトの castLib プロパティを使用します。

```
-- Lingo syntax
libScript = _movie.castLib["scripts"]

// JavaScript syntax
var libScript = _movie.castLib["scripts"];
```

キャストライブラリオブジェクトのメソッドの概要

メソッド
findEmpty()

キャストライブラリオブジェクトのプロパティの概要

プロパティ
fileName （キャスト）
member （キャスト）
name

プロパティ
number （キャスト）
preLoadMode
selection

関連事項

castLib, castLib(), メンバ , ムービー , プレーヤ , スプライト , ウィンドウ

グローバル

グローバル変数を格納したりグローバル変数にアクセスできる場所を表します。グローバル変数は、Lingo および JavaScript シンタックスで利用できます。

グローバルオブジェクトにアクセスするには、最上位の _global プロパティを使用します。_global を変数に割り当てるか、_global プロパティを使用すると、グローバルオブジェクトのメソッドや任意の定義済みグローバル変数に直接アクセスできます。

- _global を変数に割り当てます。

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

- _global プロパティを直接使用します。

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

- グローバル変数にアクセスします。

```
-- Lingo syntax
global gSuccess

on mouseDown
    gSuccess = "Congratulations!"
    put(gSuccess) -- displays "Congratulations!"
end

// JavaScript syntax
_global.gSuccess = "Congratulations!";

function mouseDown() {
    trace(_global.gSuccess); // displays "Congratulations!"
}
```

グローバルオブジェクトのメソッドの概要

メソッド
clearGlobals()
showGlobals()

関連事項
_global

キー

ユーザのキーボード操作を監視するために使用されます。

キーオブジェクトにアクセスするには、最上位の _key プロパティを使用します。_key を変数に割り当てるか、_key プロパティを使用すると、キーオブジェクトのメソッドやプロパティに直接アクセスできます。

- _key を変数に割り当てます。

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

- _key プロパティを直接使用します。

```
-- Lingo syntax
isCtrlDown = _key.controlDown

// JavaScript syntax
var isCtrlDown = _key.controlDown;
```

キーオブジェクトのメソッドの概要

メソッド
keyPressed()

キーオブジェクトのプロパティの概要

プロパティ
commandDown
controlDown
key
keyCode
optionDown
shiftDown

関連事項
_key

メンバ

キャストライブラリ内のキャストメンバを表します。キャストメンバとは、ムービー内のメディアやスクリプトアセットです。メディアキャストメンバには、テキスト、ビットマップ、シェイプなどがあります。スクリプトキャストメンバには、ビヘイビアーやムービースクリプトなどがあります。

キャストメンバは、番号または名前のいずれかで参照できます。

- キャストメンバを番号で参照する場合、そのキャストメンバの特定のキャストライブラリが検索され、メンバのデータが取得されます。この方式は、キャストメンバを名前で参照するよりも高速です。ただし、スクリプト内のキャストメンバ番号への参照は自動的には更新されないので、キャストライブラリ内でキャストメンバが移動した場合、番号での参照は機能しなくなります。
- キャストメンバを名前で参照する場合、ムービーのすべてのキャストライブラリが最初から最後まで検索され、その名前のメンバが見つかればデータが取得されます。この方式は、キャストメンバを番号で参照するよりも低速です。特に、キャストライブラリとキャストメンバが多数含まれる大容量のムービーを参照する場合はその傾向が顕著になります。ただし、キャストメンバを名前で参照すると、キャストメンバがキャストライブラリ内で移動しても、参照はそのまま機能します。

キャストメンバへの参照を作成するには、最上位の member() 関数を使用するか、キャスト、ムービー、またはスプライトオブジェクトの member プロパティを使用します。

次の例では、キャストメンバへの参照の作成方法を示します。

- 最上位の member() 関数を使用します。

```
-- Lingo syntax
objTree = member("bmpTree")

// JavaScript syntax
var objTree = member("bmpTree");
```

- スプライトオブジェクトの member プロパティを使用します。

```
-- Lingo syntax
objTree = sprite(1).member;

// JavaScript syntax
var objTree = sprite(1).member;
```

メンバオブジェクトのメソッドの概要

メソッド
copyToClipBoard()
duplicate() （メンバ）
erase()
importFileInto()
move()

メソッド
pasteClipBoardInto()
preLoad() （メンバ）
unLoad() （メンバ）

メンバオブジェクトのプロパティの概要

プロパティ	
castLibNum	modifiedDate
comments	name
creationDate	number （メンバ）
fileName （メンバ）	purgePriority
height	rect （メンバ）
hilite	regPoint
linked	scriptText
loaded	size
media	thumbNail
mediaReady	type （メンバ）
member	width
modified	
modifiedBy	

関連事項

メディアタイプ , member(), member （キャスト）, member （ムービー）, member （スプライト）, ムービー , プレーヤ , スクリプトオブジェクト , スプライト , ウィンドウ

ミクサ

ミクサは、含まれるサウンドオブジェクトをミックスして結果の出力を再生するコンテナです。ミクサは、複数のオーディオソースを単一のオーディオソースにマージするので、サウンドカードに転送されるデータの量が減り、リソースの節約になります。

ミクサオブジェクトのメソッドの概要

メソッド	
createSoundObject	deleteSoundObject
getSoundObject()	getSoundObjectList
mute （ミクサ）	pause() （ミクサ）

メソッド	
play() （ミクサ）	reset （ミクサ）
save （ミクサ）	startSave （ミクサ）
stop() （ミクサ）	stopSave （ミクサ）
unmute （ミクサ）	

ミクサオブジェクトのプロパティの概要

プロパティ	
bufferSize （ミクサ）	bitDepth（ミクサ）
channel	channelCount（ミクサ）
elapsedTime（ミクサ）	filterList （ミクサ）
isSaving （ミクサ）	name （ミクサ）
numBuffersToPreload	panMatrix（ミクサ）
sampleRate (ミクサ)	status （ミクサ）
soundObjectList	toChannels
useMatrix（ミクサ）	volume （ミクサ）

マウス

マウスの移動やクリックなど、ユーザのマウス操作にアクセスできます。

マウスオブジェクトにアクセスするには、最上位の _mouse プロパティを使用します。_mouse を変数に割り当てるか、_mouse プロパティを使用すると、マウスオブジェクトのプロパティに直接アクセスできます。

- _mouse を変数に割り当てます。

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

- _mouse プロパティを直接使用します。

```
-- Lingo syntax
isDblClick = _mouse.doubleClick

// JavaScript syntax
var isDblClick = _mouse.doubleClick;
```

マウスオブジェクトのプロパティの概要

プロパティ	
clickLoc	mouseLoc
clickOn	mouseMember
doubleClick	mouseUp
mouseChar	mouseV
mouseDown	mouseWord
mouseH	rightMouseDown
mouseItem	rightMouseUp
mouseLine	stillDown

関連事項

_mouse

ムービー

Director プレーヤで再生中のムービーを表します。

Director プレーヤは、1 つ以上のムービーを含むことができます。ムービーは、1 つ以上のキャストライブラリで構成されます。キャストライブラリは、1 つ以上のキャストメンバで構成されます。キャストメンバとは、ムービー内のメディアやスクリプトアセットです。メディアキャストメンバには、テキスト、ビットマップ、シェイプなどがあります。スクリプトキャストメンバには、ビヘイビアーやムービースクリプトなどがあります。スプライトはキャストメンバから作成され、ムービーのステージ上で使用されます。

現在アクティブなムービーを参照するには、最上位の _movie プロパティを使用します。ウィンドウオブジェクトの movie プロパティを使用すると、プレーヤの任意のムービーを参照できます。

- 現在アクティブなムービーを参照します。

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

- ウィンドウオブジェクトの movie プロパティを使用して、特定のウィンドウのムービーにアクセスします。

```
-- Lingo syntax
objMovie = _player.window[2].movie

// JavaScript syntax
var objMovie = _player.window[2].movie;
```

ムービーへの参照を使用すると、ムービー自体のメソッドやプロパティにアクセスできるだけでなく、Lingo や JavaScript シンタックスのハンドラを呼び出して、ムービーのキャストメンバやスプライトのメソッドやプロパティにアクセスすることもできます。これが以前のバージョンの Director との相違点です。以前のバージョンでムービーを操作するには tell コマンドを使用する必要がありました。ムービーオブジェクトを使用すると、ムービーを簡単に操作できます。

ムービーオブジェクトのメソッドの概要

メソッド	
beginRecording()	marker()
call	mergeDisplayTemplate()
callAncestor	newMember()
cancelIdleLoad()	preLoad() (Movie)
clearFrame()	preLoadMember()
constrainH()	preLoadMovie()
constrainV()	printFrom()
delay()	puppetPalette()
deleteFrame()	puppetSprite()
duplicateFrame()	puppetTempo()
endRecording()	puppetTransition()
finishIdleLoad()	ramNeeded()
frameReady() (Movie)	rollOver()
getNthFileNameInFolder()	saveMovie()
go()	sendAllSprites()
goLoop()	sendSprite()
goNext()	stopEvent()
goPrevious()	unLoad() （ムービー）
halt()	unLoadMember()
idleLoadDone()	unLoadMovie()
insertFrame()	updateFrame()
label()	updateStage()
loadPolicyFile()	

ムービーオブジェクトのプロパティの概要

プロパティ	
aboutInfo	idleReadChunkSize
active3dRenderer	imageCompression
activeCastLib	imageQuality
actorList	keyboardFocusSprite
allowTransportControl	lastChannel
allowVolumeControl	lastFrame

プロパティ	
allowZooming	markerList
beepOn	member （ムービー）
buttonStyle	name
castLib	paletteMapping
centerStage	path （ムービー）
channel	preferred3dRenderer
copyrightInfo （ムービー）	preLoadEventAbort
displayTemplate	selEnd
editShortCutsEnabled	selection
enableFlashLingo	selStartselStart
exitLock	score
fileFreeSize	scoreSelection
fileSize	script
fileVersion	sprite （ムービー）
fixStageSize	stage
frame	timeoutList
frameLabel	traceLoad
framePalette	traceLogFile
frameScript	traceScript
frameSound1	updateLock
frameSound2	updateMovieEnabled
frameTempo	useFastQuads
frameTransition	window
idleHandlerPeriod	xtraList （ムービー）
idleLoadMode	xtraList （ムービー）
idleLoadPeriod	xtraList （ムービー）
idleLoadTag	xtraList （ムービー）

関連事項

_movie, キャストライブラリ , メンバ , movie, プレーヤ , スプライト , ウィンドウ

プレーヤ

オーサリング環境、ウィンドウ内ムービー（MIAW）、プロジェクタ、Shockwave Player の管理や実行に使用されるコア再生エンジンを表します。

プレーヤオブジェクトを使用すると、利用可能な Xtra 拡張に加えて、管理対象のムービーやウィンドウのすべてにアクセスできます。

プレーヤオブジェクトへの参照を作成するには、最上位の _player プロパティを使用します。

- _player を変数に割り当てます。

```
-- Lingo syntax
objPlayer = _player

// JavaScript syntax
var objPlayer = _player;
```

- _player プロパティを直接使用します。

```
-- Lingo syntax
_player.alert("The movie has ended.")

// JavaScript syntax
_player.alert("The movie has ended.");
```

プレーヤオブジェクトのメソッドの概要

メソッド	
alert()	getPref()（プレーヤ）
appMinimize()	installMenu
cursor()	open()（プレーヤ）
externalParamName()	quit()
externalParamValue()	setPref()（プレーヤ）
flushInputEvents()	windowPresent()

プレーヤオブジェクトのプロパティの概要

プロパティ	
activeWindow	netThrottleTicks
alertHook	organizationName
applicationName	productName
applicationPath	productVersion
currentSpriteNum	runMode
debugPlaybackEnabled	safePlayer
digitalVideoTimeScale	scriptingXtraList
disableImagingTransformation	searchCurrentFolder
emulateMultibuttonMouse	searchPathList
externalParamCount	serialNumber
frontWindow	sound（プレーヤ）
inlineImeEnabled	switchColorDepth

プロパティ	
itemDelimiter	toolXtraList
lastClick	transitionXtraList
lastEvent	userName
activeWindow	window
lastKey	windowList
lastRoll	xtra
mediaXtraList	xtraList （プレーヤ）
netPresent	

関連事項

_player, キャストライブラリ , メンバ , ムービー , スプライト , ウィンドウ

サウンド

Director のサウンドオブジェクトは、使用可能な 16 すべてのサウンドチャネルでオーディオの再生を制御します。

古いワークフロー

サウンドオブジェクトは、サウンドチャネルオブジェクトで構成されます。サウンドチャネルオブジェクトは、個々のサウンドチャネルを表します。

サウンドオブジェクトへの参照を作成するには、最上位の _sound プロパティを使用します。

- _sound を変数に割り当てます。

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

- _sound プロパティを使用して、サウンドオブジェクトの soundDevice プロパティにアクセスします。

```
-- Lingo syntax
objDevice = _sound.soundDevice

// JavaScript syntax
var objDevice = _sound.soundDevice;
```

サウンドオブジェクトのメソッドの概要（古いワークフロー）

メソッド
beep()
channel() （サウンド）

サウンドオブジェクトのプロパティの概要（古いワークフロー）

プロパティ
soundDevice
soundDeviceList
soundEnabled
soundKeepDevice
soundLevel
soundMixMedia

関連事項

_sound, サウンドチャネル

新しいワークフロー

サウンドオブジェクトがミクサに追加されています。ミクサは、サウンドを個別または同時に再生できるサウンドオブジェクトの集まりを表します。サウンドオブジェクトに関連付けられているミクサを使用することにより、サウンドオブジェクトへの参照を作成することができます。

❖ サウンドオブジェクトを変数に関連付けるには：

```
--- Lingo syntax
objSound = mixerobjectref.getSoundObject(soundobjectname)

// JavaScript syntax
var objSound = mixerobjectref.getSoundObject(soundobjectname);
```

サウンドオブジェクトのメソッドの概要（新しいワークフロー）

メソッド	
breakLoop （サウンドオブジェクト）	Save （サウンドオブジェクト）
moveTo	seek（サウンドオブジェクト）
mute （サウンドオブジェクト）	startSave （サウンドオブジェクト）
pause （サウンドオブジェクト）	stop （サウンドオブジェクト）
play （サウンドオブジェクト）	stopSave （サウンドオブジェクト）
registerByteArrayCallback	unmute （サウンドオブジェクト）
registerCuePointCallback	unregisterByteArrayCallback
registerEndOfSpoolCallback()	unregisterCuePointCallback
replaceMember	unregisterEndOfSpoolCallback()

サウンドオブジェクトのプロパティの概要（新しいワークフロー）

プロパティ	
bitDepth（サウンドオブジェクト）	channelCount （サウンドオブジェクト）
connectionStatus（サウンドオブジェクト）	currentTime（サウンドオブジェクト）
elapsedTime （サウンドオブジェクト）	endTime （サウンドオブジェクト）
filterList （サウンドオブジェクト）	isSaving （サウンドオブジェクト）
loopCount （サウンドオブジェクト）	loopEndTime （サウンドオブジェクト）
loopsRemaining （サウンドオブジェクト）	loopStartTime （サウンドオブジェクト）
member （サウンドオブジェクト）	mixer
mostRecentCuePoint （サウンドオブジェクト）	name （サウンドオブジェクト）
percentStreamed（サウンドオブジェクト）	パンマトリクス
playRate（サウンドオブジェクト）	sampleCount （サウンドオブジェクト）
sampleRate （サウンドオブジェクト）	startTime （サウンドオブジェクト）
status （サウンドオブジェクト）	toChannels
ユースマトリクス	volume （サウンドオブジェクト）

関連事項
ミクサ

サウンドチャネル

サウンドオブジェクト内にある個々のサウンドチャネルを表します。

利用できるサウンドチャネルは 8 つです。8 つのサウンドチャネルのいずれかにアクセスしたり変更したりするには、サウンドチャネルオブジェクトをスクリプトで使用します。

注意：Director ユーザインターフェイスのスコアで変更できるのは、最初の 2 つのサウンドチャネルのみです。

サウンドチャネルオブジェクトへの参照を作成するには、最上位の sound() メソッドを使用するか、プレーヤオブジェクトの sound プロパティを使用するか、サウンドオブジェクトの channel() メソッドを使用します。例えば、サウンドチャネル 2 を参照するには、次のようにします。

- 最上位の sound() メソッドを使用します。

```
-- Lingo syntax
objSoundChannel = sound(2)

// JavaScript syntax
var objSoundChannel = sound(2);
```

- プレーヤオブジェクトの sound プロパティを使用します。

```
-- Lingo syntax
objSoundChannel = _player.sound[2]

// JavaScript syntax
var objSoundChannel = _player.sound[2];
```

- サウンドオブジェクトの channel() メソッドを使用します。

```
-- Lingo syntax
objSoundChannel = _sound.channel(2)

// JavaScript syntax
var objSoundChannel = _sound.channel(2);
```

サウンドチャネルオブジェクトのメソッドの概要

メソッド	
breakLoop()	play() （サウンドチャネル）
fadeIn()	playFile()
fadeOut()	playNext() （サウンドチャネル）
fadeTo()	queue()
getPlayList()	rewind() （サウンドチャネル）
isBusy()	setPlayList()
pause() （サウンドチャネル）	stop() （サウンドチャネル）

サウンドチャネルオブジェクトのプロパティの概要

プロパティ	
channelCount （サウンドチャネル）	member （サウンドチャネル）
elapsedTime	pan
endTime （サウンドチャネル）	sampleCount （サウンドチャネル）
loopCount	sampleRate （サウンドチャネル）
loopEndTime （サウンドチャネル）	startTime (サウンドチャネル)
loopsRemaining	status
loopStartTime	volume （サウンドチャネル）

関連事項

channel() （サウンド）, sound （プレーヤ）, sound(), サウンド

スプライト

スコアのスプライトチャネル内に登場するキャストメンバを表します。

スプライトオブジェクトは、スプライトスパンに対応します。スプライトスパンとは、特定のスプライトチャネル内のフレームの範囲です。スプライトチャネルオブジェクトは、1 つのスプライトチャネル全体を表します。含まれるスプライトの数は関係しません。

注意：非表示のステージ（Flash スプライト）上に配置された Flash® メンバコンポーネントにアクセスするには、メンバオブジェクトを使用するのが唯一の方法です。非表示のプロパティを持つ Flash スプライトにスプライトオブジェクトを使用してアクセスすると、エラーになります。

スプライトは、番号または名前のいずれかで参照できます。

- スプライトを番号で参照する場合、スコアの現在のフレームに存在するすべてのスプライトが小さい番号のチャネルから検索され、その番号のスプライトが見つかればデータが取得されます。この方式は、スプライトを名前で参照するよりも高速です。ただし、スクリプト内のスプライト番号への参照は自動的には更新されないので、ステージ上でスプライトが移動した場合、番号での参照は機能しなくなります。
- スプライトを名前で参照する場合、スコアの現在のフレームに存在するすべてのスプライトが小さい番号のチャネルから検索され、その名前のスプライトが見つかればデータが取得されます。この方式は、スプライトを番号で参照するよりも低速です。特に、キャストライブラリ、キャストメンバ、スプライトが多数含まれる大容量のムービーを参照する場合はその傾向が顕著になります。ただし、スプライトを名前で参照すると、スプライトがステージ上で移動しても、参照はそのまま機能します。

スプライトオブジェクトへの参照を作成するには、最上位の sprite() 関数を使用するか、ムービーオブジェクトの sprite プロパティを使用するか、スプライトチャネルオブジェクトの sprite プロパティを使用します。

- 最上位の sprite() 関数を使用します。

```
-- Lingo syntax
objSprite = sprite(1)

// JavaScript syntax
var objSprite = sprite(1);
```

- ムービーオブジェクトの sprite プロパティを使用します。

```
-- Lingo syntax
objSprite = _movie.sprite["willowTree"]

// JavaScript syntax
var objSprite = _movie.sprite["willowTree"];
```

- スプライトチャネルオブジェクトの sprite プロパティを使用します。

```
-- Lingo syntax
objSprite = channel(3).sprite

// JavaScript syntax
var objSprite = channel(3).sprite;
```

スプライトオブジェクトへの参照を使用すると、そのスプライトの作成元であるキャストメンバにアクセスできます。スプライトの作成元であるキャストメンバへの変更内容は、スプライトにも反映されます。次の例で、スプライト 5 の作成元であるテキストキャストメンバのテキストを変更する方法を示します。キャストメンバへの変更内容は、スプライト 5 にも反映されます。

```
-- Lingo syntax
labelText = sprite(5)
labelText.member.text = "Weeping Willow"

// JavaScript syntax
var labelText = sprite(5);
labelText.member.text = "Weeping Willow";
```

スプライトオブジェクトのプロパティの概要

プロパティ	
backColor	locV
blend （スプライト）	locZ
bottom	member （スプライト）
constraint	name （スプライト）
cursor	quad
editable	rect （スプライト）
endFrame	right
filterlist	rotation
flipH	skew
flipV	spriteNum
foreColor	startFrame
height	top
ink	width
left	
locH	

関連事項

キャストライブラリ , メンバ , ムービー , プレーヤ , sprite （ムービー） , sprite （スプライトチャネル） , sprite(), スプライトチャネル , ウィンドウ

スプライトチャネル

スコア内の個々のスプライトチャネルを表します。

スプライトオブジェクトは、スプライトスパンに対応します。スプライトスパンとは、特定のスプライトチャネル内のフレームの範囲です。スプライトチャネルオブジェクトは、1 つのスプライトチャネル全体を表します。含まれるスプライトの数は関係しません。

デフォルトではスプライトチャネルはスコアによって制御されます。スコア生成セッション中にスプライトチャネルの制御をスクリプトに切り替えるには、スプライトチャネルオブジェクトを使用します。

スプライトチャネルは、番号または名前のいずれかで参照できます。

- スプライトチャネルを番号で参照する場合、チャネルに直接アクセスすることになります。この方式は、スプライトチャネルを名前で参照するよりも高速です。ただし、スクリプト内のスプライトチャネル番号への参照は自動的には更新されないので、スコア内でスプライトチャネルが移動した場合、番号での参照は機能しなくなります。
- スプライトチャネルを名前で参照する場合、すべてのチャネルが小さい番号のチャネルから検索され、その名前のスプライトチャネルが見つかればデータが取得されます。この方式は、スプライトチャネルを番号で参照するよりも低速です。特に、キャストライブラリ、キャストメンバ、スプライトが多数含まれる大容量のムービーを参照する場合はその傾向が顕著になります。ただし、スプライトチャネルを名前で参照すると、スプライトチャネルがスコア内で移動しても、参照はそのまま機能します。

スプライトチャネルオブジェクトへの参照を作成するには、最上位の channel() メソッドを使用して、チャネルを番号か名前で参照します。

```
-- Lingo syntax
objSpriteChannel = channel(2) -- numbered reference
objSpriteChannel = channel("background") -- named reference

// JavaScript syntax
var objSpriteChannel = channel(2); // numbered reference
var objSpriteChannel = channel("background"); // named reference
```

スプライトチャネルオブジェクトへの参照を使用すると、特定のスプライトチャネルで現在使用中のスプライトにアクセスできます。次の例では、スプライトチャネル 2 で現在使用中のスプライトのバックカラーにアクセスする方法を示します。

```
-- Lingo syntax
labelSprite = channel(2).sprite.backColor

// JavaScript syntax
var labelSprite = channel(2).sprite.backColor;
```

スプライトチャネルオブジェクトのメソッドの概要

メソッド
makeScriptedSprite()
removeScriptedSprite()

スプライトチャネルオブジェクトのプロパティの概要

プロパティ
name （スプライトチャネル）
number （スプライトチャネル）
scripted
sprite （スプライトチャネル）

関連事項

キャストライブラリ , channel() （トップレベル） , メンバ , ムービー , プレーヤ , スプライト , ウィンドウ

システム

システムレベルのメソッドなど、システムや環境の情報にアクセスできます。

システムオブジェクトへの参照を作成するには、最上位の _system プロパティを使用します。

- _system を変数に割り当てます。

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

- _system プロパティを直接使用します。

```
-- Lingo syntax
sysDate = _system.date()

// JavaScript syntax
var sysDate = _system.date();
```

システムオブジェクトのメソッドの概要

メソッド
date() （システム）
getInstalledCharSets
getSystemCharSet
isCharSetInstalled
restart()
shutDown()
time() （システム）

システムオブジェクトのプロパティの概要

プロパティ
colorDepth
deskTopRectList
environmentPropList
milliseconds

関連事項

_system

ウィンドウ

ムービーを再生しているウィンドウを表します。ステージウィンドウや、その他の使用中のウィンドウ内ムービー（MIAW）が含まれます。

ウィンドウオブジェクトへの参照を作成するには、最上位の window() 関数を使用するか、プレーヤオブジェクトの window プロパティを使用するか、プレーヤオブジェクトの windowList プロパティを使用します。

- 最上位の window() メソッドを使用します。

```
-- Lingo syntax
objWindow = window("Sun")

// JavaScript syntax
var objWindow = window("Sun");
```

- プレーヤオブジェクトの window プロパティを使用します。

```
-- Lingo syntax
objWindow = _player.window["Sun"]

// JavaScript syntax
var objWindow = _player.window["Sun"];
```

- プレーヤオブジェクトの windowList プロパティを使用します。

```
-- Lingo syntax
objWindow = _player.windowList[1]

// JavaScript syntax
var objWindow = _player.windowList[1];
```

注意：最上位の window() 関数またはプレーヤオブジェクトの window プロパティを使用してウィンドウへの参照を作成する場合、そのウィンドウへの参照が作成されるのは、その名前が存在する場合のみです。その名前のウィンドウが存在しない場合、参照には VOID（Lingo）または null（JavaScript シンタックス）が含まれます。

ムービーオブジェクトの scriptExecutionStyle プロパティはデフォルトで 10 に設定され、windowType は廃止される予定でデフォルトでは appearanceOptions および titlebarOptions プロパティリストが使用されます。scriptExecutionStyle が 9 に設定されている場合、windowType は正常に機能します。

ウィンドウオブジェクトのメソッドの概要

メソッド	
close()	moveToBack()
forget() （ウィンドウ）	moveToFront()
maximize()	open() （ウィンドウ）
mergeProps()	restore()
minimize()	

ウィンドウオブジェクトのプロパティの概要

プロパティ	
appearanceOptions	resizable
bgColor （ウィンドウ）	sizeState
dockingEnabled	sourceRect
drawRect	title （ウィンドウ）
fileName （ウィンドウ）	titlebarOptions
image （ウィンドウ）	type （ウィンドウ）
movie	visible

プロパティ	
name	windowBehind
picture （ウィンドウ）	windowInFront
rect （ウィンドウ）	

関連事項

キャストライブラリ , メンバ , ムービー , プレーヤ , スプライト , window(), window, windowList

第 6 章：メディアタイプ

Director® のメディアタイプを使用すると、RealMedia®、MP4Media、FLV、DVD、アニメーション GIF など、キャストメンバとしてムービーに追加される様々なメディアタイプの機能にアクセスできます。

メディアタイプは実際にはオブジェクトというよりも、特定のメディアタイプに属するキャストメンバに近い概念です。メディアタイプをムービーにキャストメンバとして追加すると、コアメンバオブジェクトの機能が継承されるだけでなく、特定のメディアタイプ専用の追加機能を提供することで、メンバオブジェクトが拡張されます。例えば、RealMedia キャストメンバはメンバオブジェクトのメソッドとプロパティにアクセスできることに加えて、RealMedia 固有のメソッドとプロパティも利用できます。その他のメディアタイプもすべて、このような動作を示します。

キャストメンバメディアタイプ同士の関係と、キャストメンバメディアタイプと Director 内の他のオブジェクトとの関係を表す図は、43 ページの「オブジェクトモデル図」を参照してください。

アニメーション GIF

アニメーション GIF キャストメンバを表します。

アニメーション GIF キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#animgif)

// JavaScript syntax
_movie.newMember("animgif");
```

次のメソッドやプロパティの中には、アニメーション GIF キャストメンバから作成されるスプライトにのみ適用できるものがあります。

アニメーション GIF メディアタイプのメソッドの概要

メソッド
resume()
rewind() （アニメーション GIF、Flash）

アニメーション GIF メディアタイプのプロパティの概要

プロパティ
directToStage
frameRate
linked
path （ムービー）
playBackMode

関連事項

メンバ

ビットマップ

ビットマップキャストメンバを表します。

ビットマップ画像オブジェクトを使用すると、ビットマップキャストメンバ全体の内容に対してメンバのバックカラーやフォアカラーを変更するなど簡単な操作を実行したり、トリミング、描画、コピーなど画像のピクセルを細かく操作したりすることができます。

ビットマップキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#bitmap)

// JavaScript syntax
_movie.newMember("bitmap");
```

次のメソッドやプロパティの中には、ビットマップキャストメンバから作成されるスプライトにのみ適用できるものがあります。

ビットマップメディアタイプのメソッドの概要

メソッド
crop() （画像）
pictureP()

ビットマップメディアタイプのプロパティの概要

プロパティ	
alphaThreshold	imageCompression
backColor	imageQuality
blend （スプライト）	palette
depth （ビットマップ）	picture （メンバ）
dither	rect （画像）
foreColor	trimWhiteSpace
image （画像）	useAlpha

関連事項

メンバ

ボタン

ボタンまたはチェックボックスキャストメンバを表します。

ボタンキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#button)

// JavaScript syntax
_movie.newMember("button");
```

ボタンメディアタイプのプロパティの概要

プロパティ
hilite

関連事項

メンバ

ByteArray

ByteArray キャストメンバ

#byteArray キャストメンバは、Director、Shockwave、および Projector で使用できます。このスクリプト専用キャストメンバには、制限付きのユーザインターフェイスがあります。

UI による ByteArray メンバの作成 挿入／メディアエレメント／バイト配列を選択して、ByteArray キャストメンバを挿入します。

Lingo による ByteArray メンバの作成 new メソッドを使用して、ByteArray キャストメンバを作成します。

使用法

```
m = new(#byteArray)
```

詳細については、「144 ページの「ByteArray」」を参照してください。

カラーパレット

ビットマップキャストメンバに関連付けられたカラーパレットを表します。

カラーパレットキャストメンバから直接アクセスできるメソッドやプロパティはありません。次のメソッドやプロパティは、単にカラーパレットに関連付けられています。

カラーパレットキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#palette)

// JavaScript syntax
_movie.newMember("palette");
```

ビットマップキャストメンバをカラーパレットキャストメンバに関連付けるには、ビットマップキャストメンバの palette プロパティを設定します。次の例では、ビットマップキャストメンバ bmpMember の palette プロパティをカラーパレットキャストメンバ colorPaletteMember に設定します。palette プロパティの値は、カラーパレットキャストメンバの番号を表します。

```
-- Lingo syntax
member("bmpMember").palette = member("colorPaletteMember")

// JavaScript syntax
member("bmpMember").palette = member("colorPaletteMember");
```

ビットマップキャストメンバをカラーパレットキャストメンバと関連付けた後、カラーパレットキャストメンバを削除するには、まずビットマップキャストメンバとの関連付けを削除する必要があります。

カラーパレットメディアタイプのメソッドの概要

メソッド
color()

カラーパレットメディアタイプのプロパティの概要

プロパティ
depth （ビットマップ）
palette
paletteMapping

関連事項

ビットマップ , メンバ , palette

カーソル

カーソルキャストメンバを表します。

カーソルキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#cursor)

// JavaScript syntax
_movie.newMember("cursor");
```

カーソルメディアタイプのプロパティの概要

プロパティ
castMemberList
cursorSize
hotSpot
interval

関連事項

メンバ

DVD

DVD キャストメンバを表します。

DVD キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#dvd)

// JavaScript syntax
_movie.newMember("dvd");
```

次のメソッドやプロパティの中には、DVD キャストメンバから作成されるスプライトにのみ適用できるものがあります。

DVD メディアタイプのイベントの概要

次の DVD イベントは、常に DVDeventNotification イベントハンドラによって処理されます。これらのイベントのいずれかが発生すると、DVDeventNotification イベントハンドラはイベントをパラメータとして受け取ります。これらのイベントの中には、2 番目または 3 番目のパラメータとして DVDeventNotification に渡される追加情報を含むものがあります。DVDeventNotification イベントハンドラによる次のイベントの使用方法については、on DVDeventNotification を参照してください。

イベント	
on DVDeventNotification	noFirstPlayChain
audioStreamChange	parentalLevelChange
buttonChange	playbackStopped
chapterAutoStop	playPeriodAutoStop
chapterStart	rateChange
diskEjected	stillOff
diskInserted	stillOn
domainChange	titleChange
エラーえらー	UOPchange
karaokeMode	warning

DVD メディアタイプのメソッドの概要

メソッド	
activateAtLoc()	rootMenu()
activateButton()	selectAtLoc()
frameStep()	selectButton()
chapterCount()	selectButtonRelative()

メソッド	
pause() (DVD)	stop() (DVD)
play() (DVD)	subPictureType()
returnToTitle()	titleMenu()

DVD メディアタイプのプロパティの概要

プロパティ	
angle (DVD)	duration (DVD)
angleCount	folder
aspectRatio	frameRate (DVD)
audio (DVD)	fullScreen
audioChannelCount	mediaStatus (DVD)
audioExtension	playRate (DVD)
audioFormat	resolution (DVD)
audioSampleRate	selectedButton
audioStream	startTimeList
audioStreamCount	stopTimeList
buttonCount	subPicture
chapter	subPictureCount
chapterCount	title (DVD)
closedCaptions	titleCount
currentTime (DVD)	videoFormat
domain	volume (DVD)

関連事項

メンバ

フィールド

フィールドキャストメンバを表します。

フィールドキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#field)

// JavaScript syntax
_movie.newMember("field");
```

フィールドメディアタイプのメソッドの概要

メソッド	
charPosToLoc()	pointToItem()
lineHeight()	pointToLine()
linePosToLocV()	pointToParagraph()
locToCharPos()	pointToWord()
locVToLinePos()	scrollByLine()
pointToChar()	scrollByPage()

フィールドメディアタイプのプロパティの概要

プロパティ	
alignment	fontStyle
autoTab	lineCount
border	margin
boxDropShadow	pageHeight
boxType	scrollTop
dropShadow	selEnd
editable	selStart
font	text
fontSize	wordWrap

関連事項

メンバ

フィルムループ

フィルムループキャストメンバを表します。

フィルムループキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#filmloop)

// JavaScript syntax
_movie.newMember("filmloop");
```

フィルムループメディアタイプのプロパティの概要

プロパティ
media
regPoint

関連事項

メンバ

Flash コンポーネント

Flash コンテンツを含むキャストメンバまたはスプライトに埋め込まれる Adobe® Flash® コンポーネントを表します。

Flash コンポーネントには、Flash コンテンツを含むキャストメンバやスプライトの既存機能を拡張する機能が備わっています。これらの拡張機能は、あらかじめパッケージ化されています。Flash コンポーネントの作成とサポートは、Director 開発コミュニティによって全面的に行われます。

Director では次の Flash コンポーネントがサポートされています。

Flash コンポーネント	解説
ボタン	サイズが変更できる矩形のユーザインターフェイスボタンです。
CheckBox	フォームまたは Web アプリケーションの基本的なパーツです。相互に排他的ではない true または false の値のセットを収集する必要がある場合に使用します。
DateChooser	ユーザが日付を選択できるカレンダーです。
ラベル	1 行のテキストです。
List	スクロール可能な単一選択または複数選択のリストボックスです。
NumericStepper	順序付けられた数値のセットをユーザが操作できます。
RadioButton	フォームまたは Web アプリケーションの基本的なパーツです。ユーザに選択肢のグループから 1 つを選択させる場所で使用します。
ScrollPane	スクロール可能領域にムービークリップ、JPEG ファイル、および SWF ファイルを表示します。
TextArea	複数行のテキストフィールドです。
TextInput	ネイティブの ActionScript TextField オブジェクトを折り返す単一行コンポーネントです。
Tree	ユーザが階層データを表示できます。

Flash コンポーネントを使用すると、Flash コンポーネント固有の機能に加えて、通常の Flash キャストメンバやスプライトがアクセスできるのと同じ API にアクセスできます。これらの Flash コンポーネントの使用方法について詳しくは、Director ヘルプパネルの「Director の使い方」を参照してください。

Flash コンポーネントキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#flashcomponent)

// JavaScript syntax
_movie.newMember("flashcomponent");
```

関連事項

Flash ムービー , メンバ

Flash ムービー

Flash コンテンツを含むキャストメンバまたはスプライトを表します。

Flash ムービーキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#flash)

// JavaScript syntax
_movie.newMember("flash");
```

Flash ムービーキャストメンバまたはスプライトに Flash コンポーネントを含めることもできます。Flash コンポーネントには、Flash ムービーキャストメンバやスプライトの既存機能を拡張する機能が備わっています。これらの拡張機能は、あらかじめパッケージ化されています。Director でサポートされる Flash コンポーネントの詳細については、「119 ページの「Flash コンポーネント」」を参照してください。

次のメソッドやプロパティの中には、Flash ムービーキャストメンバから作成されるスプライトにのみ適用できるものがあります。

Flash ムービーメディアタイプのメソッドの概要

メソッド	
callFrame()	printAsBitmap()
clearAsObjects()	rewind() （アニメーション GIF、Flash）
clearError()	setCallback()
findLabel()	setFlashProperty()
flashToStage()	settingsPanel()
getFlashProperty()	setVariable()
getVariable()	showProps()
goToFrame()	stageToFlash()
hitTest()	stop() (Flash)
hold()	stream()
newObject()	tellTarget()
print()	

Flash ムービーメディアタイプのプロパティの概要

プロパティ	
actionsEnabled	originPoint
broadcastProps	originV
bufferSize	playBackMode
buttonsEnabled	playing
bytesStreamed	posterFrame
centerRegPoint	quality
clickMode	rotation
defaultRect	scale （メンバ）
defaultRectMode	scaleMode
eventPassMode	sound （メンバ）
fixedRate	static
flashRect	streamMode
frameCount	streamSize
imageEnabled	viewH
linked	viewPoint
mouseOverButton	viewScale
originH	viewV
originMode	

関連事項

Flash コンポーネント , メンバ

フォント

フォントキャストメンバを表します。

フォントキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#font)

// JavaScript syntax
_movie.newMember("font");
```

フォントメディアタイプのプロパティの概要

プロパティ
bitmapSizes
characterSet
fontStyle
originalFont
recordFont

関連事項

メンバ

リンクされたムービー

リンクされたムービーキャストメンバを表します。

リンクされたムービーキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#movie)

// JavaScript syntax
_movie.newMember("movie");
```

リンクされたムービーメディアタイプのプロパティの概要

プロパティ
scriptsEnabled

関連事項

メンバ

MP4Media/FLV

Director では、H.264 でエンコードされた MP4、F4V、および FLV のファイル形式がサポートされています。

MP4Media/FLV アセットのプロパティの概要

プロパティ	
audio (MP4Media/FLV)	connectionStatus (MP4Media/FLV)
currentTime (MP4Media/FLV)	directToStage (MP4Media/FLV)
duration (MP4Media/FLV)	frameRate (MP4Media/FLV)

プロパティ	
fileName (MP4Media/FLV)	height (MP4Media/FLV)
image (MP4Media/FLV)	isPlayable (MP4Media/FLV)
loop (MP4Media/FLV)	percentStreamed (MP4Media/FLV)
trackInfo	video (MP4Media/FLV)
volume (MP4Media/FLV)	width (MP4Media/FLV)

MP4Media/FLV スプライトのプロパティの概要

プロパティ	
audio (MP4Media/FLV)	connectionStatus (MP4Media/FLV)
currentTime (MP4Media/FLV)	directToStage (MP4Media/FLV)
image (MP4Media/FLV)	isPlayable (MP4Media/FLV)
loop (MP4Media/FLV)	mediaStatus (MP4Media/FLV)
mixer	pausedAtStart (MP4Media/FLV)
percentStreamed (MP4Media/FLV)	video (MP4Media/FLV)
volume (MP4Media/FLV)	

MP4Media/FLV アセットのメソッドの概要

メソッド	
enableSoundTrack(trackNum)	pause (MP4Media/FLV)
play() (MP4Media/FLV)	seek(mSec) (MP4Media/FLV)
stop() (MP4Media/FLV)	

MP4Media/FLV スプライトのメソッドの概要

メソッド	
pause (MP4Media/FLV)	play() (MP4Media/FLV)
rewind() (MP4Media/FLV)	seek(mSec) (MP4Media/FLV)
stop() (MP4Media/FLV)	

QuickTime

QuickTime® キャストメンバを表します。

QuickTime キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#quicktimemedia)

// JavaScript syntax
_movie.newMember("quicktimemedia");
```

次のメソッドやプロパティの中には、QuickTime キャストメンバから作成されるスプライトにのみ適用できるものがあります。

QuickTime メディアタイプのメソッドの概要

メソッド	
enableHotSpot()	qtRegisterAccessKey()
getHotSpotRect()	qtUnRegisterAccessKey()
nudge()	setTrackEnabled()
ptToHotSpotID()	swing()
QuickTimeVersion()	

QuickTime メディアタイプのプロパティの概要

プロパティ	
audio (RealMedia)	scale （メンバ）
currentTime (QuickTime, AVI)	staticQuality
fieldOfView	tilt
hotSpotEnterCallback	trackCount （メンバ）
hotSpotExitCallback	trackCount （スプライト）
invertMask	trackEnabled
isVRMovie	trackNextKeyTime
loopBounds	trackNextSampleTime
mask	trackPreviousKeyTime
motionQuality	trackPreviousSampleTime
mouseLevel	trackStartTime （メンバ）
node	trackStartTime （スプライト）
nodeEnterCallback	trackStopTime （メンバ）
nodeExitCallback	trackStopTime （スプライト）
nodeType	trackText
pan （QTVR プロパティ）	trackType （メンバ）
percentStreamed （メンバ）	trackType （スプライト）

プロパティ	
playRate（QuickTime、AVI、MP4、FLV）	translation
preLoad （メンバ）	triggerCallback
rotation	warpMode

関連事項

メンバ

RealMedia

RealMedia キャストメンバを表します。

RealMedia キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#realmedia)

// JavaScript syntax
_movie.newMember("realmedia");
```

次のメソッドやプロパティの中には、RealMedia キャストメンバから作成されるスプライトにのみ適用できるものがあります。

RealMedia メディアタイプのメソッドの概要

メソッド
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
realPlayerNativeAudio()
realPlayerPromptToInstall()
realPlayerVersion()
seek()
stop() (RealMedia, SWA, Windows Media)

RealMedia メディアタイプのプロパティの概要

プロパティ	
audio (RealMedia)	password
currentTime (RealMedia)	pausedAtStart (RealMedia, Windows Media)
displayRealLogo	percentBuffered
duration (RealMedia, SWA)	soundChannel (RealMedia)

プロパティ	
image (RealMedia)	state (RealMedia)
lastError	userName (RealMedia)
mediaStatus (RealMedia, Windows Media)	video (RealMedia, Windows Media)

関連事項

メンバ

Shockwave 3D

Adobe® Shockwave® 3D キャストメンバを表します。

Shockwave 3D（以下、3D）キャストメンバは他のキャストメンバとは異なり、3D キャストメンバに完全な 3D ワールドが含まれています。3D ワールドには、3D キャストメンバに固有のオブジェクトセットが含まれ、それによって 3D 機能をムービーに追加できます。

3D キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#shockwave3d)

// JavaScript syntax
_movie.newMember("shockwave3d");
```

3D キャストメンバで利用できるオブジェクトと API について詳しくは、「154 ページの「3D オブジェクト」」を参照してください。

関連事項

メンバ

Shockwave オーディオ

Shockwave オーディオキャストメンバを表します。

Shockwave オーディオキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#swa)

// JavaScript syntax
_movie.newMember("swa");
```

Shockwave オーディオメディアタイプのイベントの概要

イベント
on cuePassed

Shockwave オーディオメディアタイプのメソッドの概要

メソッド
getError() (Flash, SWA)
getErrorString()
isPastCuePoint()
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
preLoadBuffer()
stop() (RealMedia, SWA, Windows Media)

Shockwave オーディオメディアタイプのプロパティの概要

プロパティ	
bitRate	percentStreamed （メンバ）
bitsPerSample	preLoadTime
channelCount （サウンドチャネル）	sampleRate （サウンドチャネル）
copyrightInfo（SWA）	sampleSize
cuePointNames	soundChannel (SWA)
cuePointTimes	state (Flash, SWA)
duration (RealMedia, SWA)	streamName
loop （メンバ）	URL
mostRecentCuePoint	volume （メンバ）
numChannels	

関連事項

メンバ

サウンド

サウンドサンプルの格納および参照に使用されるキャストメンバを表します。

サウンドサンプルは、サウンドオブジェクトとサウンドチャネルオブジェクトによって制御されます。サウンドキャストメンバは独自の API を持たないので、そのビヘイビアーを制御するにはサウンドオブジェクトやサウンドチャネルオブジェクトの API を使用します。

サウンドキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#sound)

// JavaScript syntax
_movie.newMember("sound");
```

サウンドサンプルの制御に利用できるオブジェクトと API について詳しくは、「102 ページの「サウンド」」と「104 ページの「サウンドチャネル」」を参照してください。

関連事項

メンバ

テキスト

テキストキャストメンバを表します。

テキストキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#text)

// JavaScript syntax
_movie.newMember("text");
```

テキストメディアタイプのイベントの概要

イベント
on hyperlinkClicked

テキストメディアタイプのメソッドの概要

メソッド
count()
pointInHyperlink()
pointToChar()
pointToItem()
pointToLine()
pointToParagraph()
pointToWord()

テキストメディアタイプのプロパティの概要

プロパティ	
antiAlias	hyperlink
antiAliasThreshold	hyperlinkRange
bottomSpacing	hyperlinks
charSpacing	hyperlinkState
firstIndent	kerning

プロパティ	
fixedLineSpace	kerningThreshold
font	RTF
fontStyle	selectedText
HTML	useHypertextStyles

関連事項

メンバ

ベクターシェイプ

ベクトルシェイプキャストメンバを表します。

ベクトルシェイプキャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#vectorshape)

// JavaScript syntax
_movie.newMember("vectorshape");
```

次のメソッドやプロパティの中には、ベクトルシェイプキャストメンバから作成されるスプライトにのみ適用できるものがあります。

ベクトルシェイプメディアタイプのメソッドの概要

メソッド
addVertex()
deleteVertex()
moveVertex()
moveVertexHandle()
newCurve()
showProps()

ベクトルシェイプメディアタイプのプロパティの概要

プロパティ	
antiAlias	imageEnabled
backgroundColor	originH
broadcastProps	originMode
centerRegPoint	originPoint

プロパティ	
closed	originV
curve	regPointVertex
defaultRect	scale （メンバ）
defaultRectMode	scaleMode
endColor	strokeColor
fillColor	strokeWidth
fillCycles	vertex
fillDirection	vertexList
fillMode	viewH
fillOffset	viewPoint
fillScale	viewScale
flashRect	viewV
gradientType	

関連事項

メンバ

Windows Media

Windows Media® キャストメンバを表します。

Windows Media キャストメンバをムービーに追加するには、ムービーオブジェクトの newMember() メソッドを使用します。

```
-- Lingo syntax
_movie.newMember(#windowsmedia)

// JavaScript syntax
_movie.newMember("windowsmedia");
```

次のメソッドやプロパティの中には、Windows Media キャストメンバから作成されるスプライトにのみ適用できるものがあります。

Windows Media メディアタイプのメソッドの概要

メソッド
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
rewind() (Windows Media)
stop() (RealMedia, SWA, Windows Media)

Windows Media メディアタイプのプロパティの概要

プロパティ	
audio (Windows Media)	pausedAtStart (RealMedia, Windows Media)
directToStage	playRate (Windows Media)
duration （メンバ）	video (RealMedia, Windows Media)
height	volume (Windows Media)
loop (Windows Media)	width
mediaStatus (RealMedia, Windows Media)	

関連事項

メンバ

第 7 章：スクリプトオブジェクト

Xtra 拡張機能とも呼ばれる Director® のスクリプトオブジェクトは、Director にインストールされているソフトウェアコンポーネントの機能へのアクセスを提供し、また Director の中核機能を拡張します。既存の Xtra 拡張機能は、フィルタの読み込みやインターネットへの接続などの機能を提供します。C プログラミング言語でプログラミングできれば、独自のカスタム Xtra 拡張機能を作成できます。

スクリプトオブジェクトがどのように相互に関連し、また Director の他のオブジェクトとどのように関連するかを示す図については、43 ページの「オブジェクトモデル図」を参照してください。

オーディオフィルタ

オーディオフィルタは、MP3、WAV、MP4 などの形式のオーディオファイルに効果を適用するプラグインオーディオエディタです。オーディオフィルタは PCM サンプルに作用します。

これらのフィルタの構文は次のとおりです。

```
<audioFilter Object Reference> audioFilter (<symbol>, <paramList>)
```

フィルタは、パラメータが適切に指定されている場合はフィルタオブジェクトを返します。一方、パラメータが正しく指定されていない場合、または範囲外の場合は、エラーを返します。

オーディオフィルタのプロパティの概要

プロパティ名
enabled（フィルタ）

EchoFilter

エコーは、リスナーに到達した波形が若干の遅延の後で反射されたものです。エコーは、音波の単一の反射または複数の反射のどちらでもかまいません。

Director のエコーフィルタは、減少しながら反復する一連のエコーを音波に追加します。

プロパティの概要

プロパティ	解説	範囲	デフォルト
echoLevel:Number	未処理の信号に対する処理後の信号の比率。	0 ～ 1	0.8
feedback:Number	入力にフィードバックされる出力の割合。	0 ～ 1	0.5
delayTime:Number	フィードバック間の遅延。	1 ～ 2000	1000

例文

次の例では、デフォルトのパラメータを持つエコーフィルタをサウンドオブジェクトに適用します。

```
-- Lingo syntax
on mouseUp me
mixref = new(#mixer)
soundobj = mixref.createsoundobject("so",member(2))
myfilter=audiofilter(#echofilter,[#echoLevel:0.8,#feedback:0.5,#delaytime:1000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer = _movie.newMember(symbol("mixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("Echofilter"),propList(symbol("echoLevel"),0.8,symbol("feedback"),0.5,symbol(
"delaytime"),1000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FlangeFilter

フランジは、元の信号とそのエコーの間の遅延が非常に短くて時間と共に変化するエコー効果です。この効果はスイープサウンドと呼ばれることもあります。

プロパティの概要

プロパティ	解説	範囲	デフォルト
mix	オリジナル（ドライ）とフランジ適用（ウェット）信号のミックスを調整します。フランジ適用中に発生する特有のキャンセルと増強を実現するには、両方の信号がある程度必要です。Original を 1 にすると、フランジ適用は行われません。Delayed を 0 にすると、結果として揺れる音になります。	0 ～ 1	0.8
feedback	フランジャーにフィードバックされるフランジ適用された信号のパーセンテージを決定します。フィードバックを使用しない場合、エフェクトではオリジナルの信号のみを使用します。フィードバックを追加すると、エフェクトでは再生の現在位置の前の地点から、影響を受けた信号のパーセンテージを使用します。	0 ～ 1	0.5
delayTime	入力信号のコピーで使用される最小の遅延。	1 ～ 30	10
width	delayTime パラメータでの遅延に加えて信号に追加する最大の遅延。	1 ～ 30	10
rate	元のオーディオに適用される低周波数オシレータの周波数。	0.01 - 60	0.25
waveform	使用する波形（正弦波、三角波、対数波）。	正弦、三角、対数	正弦

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#FlangeFilter, [#mix:0.8, #feedback:0.5, #delayTime:10, #width:10, #rate:0.25,
#waveform:#sine])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FlangeFilter"),propList(symbol("Mix"),0.8,symbol("feedback"),0.5,symbol("del
aytime"),10,symbol("width"),10,symbol("frequency"),0.25,symbol("rate"),0.25,symbol("waveform"),symbol("si
ne")));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

DistortionFilter

ひずみの効果を使用するとブローンカースピーカー、オーバードライブアンプなどをシミュレーションできます。Director では、振幅ひずみをサポートしています。

distortionValues, を使用したユーザによって提供される値でシグナル強度を変更するか、指定した領域内でランダムな値を追加することにより、振幅ひずみを実現します。

プロパティの概要

プロパティ	解説	範囲	デフォルト
パーセンテージ	言及している量が絶対値であるか、パーセンテージであるか。	0-1	1
量	ひずみ量より小さいランダムな値は、オーディオシグナルデータのひずみに追加されます。パーセンテージの場合、値は 0 から 100 の範囲である必要があります。そうでない場合、値は 0 から 32000 の間で変化します。	0-100/0-32000	25
useRandom	ランダムな値または distortionValues マトリクスの値。useRandom が True に設定されている場合、0 と前述の値の間の値が使用されます。False に設定されている場合、distortionValues マトリクスの値が使用されます。	0-1	1
distortionValues	列が from decibel and to decibel の 2 次元配列、および様々なデシベルレベルのひずみ値を指定するひずみ係数。distortionValues[x][0] と distortionValues[x][1] の領域内のシグナル強度には、distortionValues[x][2] の値をかけます。	distortion[x][2] の値は 0 から 10 の範囲です。	
numRange	distortionValues （マトリクスの行数）の値の数。		

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audiofilter(#Distortionfilter) -- Creates the distortion filter
-- using default parameters.
     soundobj.filterlist.append(myfilter)
     rows=3
     columns=3
     freq=100
     amp=5
     myMatrix = newmatrix(rows,columns) -- Creates a 3X3 matrix.

     repeat with i = 1 to rows  -- Initializes the matrix.
       repeat with j = 1 to columns
         if j=3 then
           myMatrix.setval(i,j,amp)
           amp=amp+5
         else
           myMatrix.setval(i,j,freq)
           freq=freq+100
         end if
       end repeat
     end repeat

     myfilter.userandom=0
     myfilter.distortionvalues = myMatrix -- Assigns myMatrix to the Distortion matrix.
     myfilter.numrange=rows
     mixref.play()

end

// JavaScript syntax
function mouseUp()
{
    mixref =_movie.newMember(symbol("mixer"));
    soundobj=mixref.createSoundobject("so",member(2));
    myfilter=audioFilter(symbol("DistortionFilter"));// Creates the distortion filter
// using default parameters.
    soundobj.filterlist.append(myfilter);
    rows=3;
    columns=3;
    freq=100;
    amp=5;

     myMatrix = newMatrix(rows,columns ); //Creates a 3X3 matrix.

     for(i = 1;i<=rows;i++)
     {
```

```
        for(j = 1;j<=columns;j++)
        {
           if (j==3)
           {
            myMatrix.setval(i,j,amp);
            amp=amp+5;
            }
           else
           {
            myMatrix.setval(i,j,freq);
            freq=freq+100;
           }
        }
     }
    myfilter.userandom=0;
    myfilter.distortionvalues = myMatrix; // Assigns myMatrix to the Distortion matrix.
    myfilter.numrange=rows;
    mixref.play();

}
```

AmplifierFilter

アンプフィルタは、入力オーディオ信号を減衰または増幅します。アンプフィルタが増幅の間に歪められると、オーディオは歪んだままになります。

プロパティの概要

プロパティ	解説	範囲	デフォルト
Factor	入力を増幅する係数。	0 ～ 10	2

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#AmplifierFilter, [#factor:2])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("AmplifierFilter"),propList(symbol("factor"),2));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

EnvelopeFilter

エンベロープは、周期的に振動する波形（sine wave, sawTooth または rectangularwave）をメインのオーディオ信号に重ね合わせます。

使用法

```
audioFilter(#EnvelopeFilter, [#frequency:1, #waveType:#Sine])
```

プロパティの概要

プロパティ	解説	範囲	デフォルト
Rate	メインのオーディオ信号に重ね合わせるオシレータ波のレート。	0.01 ～ 100	1
ByteArray	バイト配列を作成することにより、カスタムエンベロープが得られます。バイト配列内の各バイトはエンベロープボリュームを表します。音量は 0（静か）から 255（フルボリューム）の範囲内とすることができます。このバイト配列を使用するには、波形を #input に設定します。バイト配列のコンテンツは 1 つの波形として扱われ、持続時間は周波数パラメータにより制御されます。バイト配列のサイズが 2 バイトで、周波数が 1 Hz の場合、バイト配列の各バイトはオーディオの 0.5 秒に適用されます。	0 ～ 255	バイト配列のデフォルト値。
Waveform	オシレータ波の種類。	正弦、のこぎり、矩形、三角、入力	正弦

例文

```
--Lingo syntax
on mouseUp me
mixref = new(#mixer )
soundobj = mixref.createSoundObject("so", member(2))
myfilter = audioFilter(#envelopeFilter, [ #rate:1, #waveform:#Sine])
soundobj.filterlist.append(myfilter)
mixref.play()
end

//JavaScript  syntax
function mouseUp()
{
mixer = _movie.newMe mber(symbol("m ixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("envelopeFilter"),propList(symbol("rate"),1,symbol("waveform"),symbol("Sine")
));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

バイト配列の使用例を次に示します。

```
mix = new(#mixer)
so = mix.createSoundObject("1", member(1)) so.filterlist.append(audiofilter(#envelopefilter,
[#bytearray:b, #waveform:#input])) mix.play()
```

バイト配列は次のようなムービースクリプトで指定されます。

```
on startmovie me
b = bytearray()
repeat with i = 1 to 51000
b[i] = i/200
end repeat
```

FadeOutFilter

サウンドの強度を現在のレベルからゼロまで低下させます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
duration	オーディオ信号強度がゼロまで低下するのに要する時間。duration はミリ秒の単位で指定します。	1 ～ 1000000	1000
startDelay	フィルタが適用された後の初期遅延。	0 ～ 8000000	0

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeOutFilter, [#duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeOutFilter"),propList(symbol("duration"),1000, symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FadeInFilter

サウンドのオーディオ強度を現在のレベルから最大値まで上昇させます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
duration	オーディオ信号強度が最大まで上昇するのに要する時間。duration はミリ秒の単位で指定します。	1 ～ 1000000	1000
startDelay	フィルタが適用された後の初期遅延。サイレンスは startDelay 時間の前に追加されます。	0 ～ 8000000	0

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeInFilter, [#duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeInFilter"),propList(symbol("duration"),1000, symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FadeToFilter

オーディオ信号のサウンド強度を、指定したレベルまで増減します。

プロパティの概要

プロパティ	解説	範囲	デフォルト
duration	オーディオ信号強度が指定したレベルまで上昇 / 低下するのに要する時間。duration はミリ秒の単位で指定します。	1 ～ 1000000	1000
toValue	サウンド強度を上昇または低下させて最終的に達するレベル。	0 ～ 255	0
startDelay	フィルタが適用された後の初期遅延。	0 ～ 8000000	0

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeToFilter, [#toValue:0, #duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeToFilter"),propList(symbol("toValue"),0,symbol("duration"),1000,
symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

LowPassFilter

低パスフィルタを使用すると、指定したカットオフ周波数より低い周波数を通過させることができます。他のすべての周波数は遮断されます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
lowCutOffFreq	カットオフ周波数。この周波数より低い周波数だけが通過できます。	20 ～ 96000	1000

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#LowPassFilter, [#lowCutOffFreq:1000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("LowPassFilter"),propList(symbol("lowCutOffFreq"),1000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

HighPassFilter

高パスフィルタを使用すると、指定したカットオフ周波数より高い周波数を通過させることができます。他のすべての周波数は遮断されます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
highCutOffFreq	カットオフ周波数。この周波数より高い周波数だけが通過できます。	20 ～ 96000	4000

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#HighPassFilter, [#highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("HighPassFilter"),propList(symbol("highCutOffFreq"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

BandPassFilter

バンドパスフィルタを使用すると、指定した 2 つのカットオフ周波数の間の周波数を通過させることができます。他のすべての周波数は遮断されます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
lowCutOffFreq	下限と上限のカットオフ周波数の間の周波数は通過できます。他のすべての周波数は遮断されます。	20 ～ 96000	1000
highCutOffFreq	下限と上限のカットオフ周波数の間の周波数は通過できます。他のすべての周波数は遮断されます。	20 ～ 96000	4000

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#BandPassFilter, [#lowCutOffFreq: 1000, #highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("BandPassFilter"),propList(symbol("lowCutOffFreq"),1000,symbol("highCutOffFre
q"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

BandStopFilter

バンド停止フィルタは、指定した 2 つのカットオフ周波数の間の周波数を遮断します。他のすべての周波数は通過できます。

プロパティの概要

プロパティ	解説	範囲	デフォルト
lowCutOffFreq	下限と上限のカットオフ周波数の間の周波数は遮断されます。他のすべての周波数は通過できます。	20 ～ 96000	1000
highCutOffFreq	下限と上限のカットオフ周波数の間の周波数は遮断されます。他のすべての周波数は通過できます。	20 ～ 96000	4000

例文

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#BandStopFilter, [#lowCutOffFreq: 1000, #highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("BandStopFilter"),propList(symbol("lowCutOffFreq"),1000,symbol("highCutOffFre
q"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ReverbFilter

反響フィルタは、乱反射つまり反響するサウンドの効果を作成します。

プロパティの概要

プロパティ	解説	範囲	デフォルト
wetMix	出力に占める処理された信号の割合。	0 ～ 1	0
dryMix	出力に占める元の信号の割合。	0 ～ 1	0
reverbWidth	反響の幅。	0.1 ～ 100	1
lowCutOffFreq	このしきい値より低い周波数は減衰されます。	20 ～ 96000	500
highCutOffFreq	このしきい値より高い周波数は減衰されます。	20 ～ 96000	5000
decayRate	初期反射後にオーディオ信号が減少する速さを示します。	0.01 - 0.99	0.2

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#ReverbFilter, [#wetMix:0.3, #dryMix:0, #reverbWidth:1, #lowCutOffFreq:500,
#highCutOffFreq:5000, #decayRate:0.2])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixref =_movie.newMember(symbol("mixer"));
soundobj =mixref.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("ReverbFilter"),propList(symbol("wetMix"),0.3,symbol("dryMix"),0,symbol("reve
rbWidth"),1,symbol("lowCutOffFreq"),500,symbol("highCutOffFreq"),5000), symbol("decayRate"),0.2);
soundobj.filterlist.append(myfilter);
mixref.play();
}
```

PitchShiftFilter

オーディオ信号の周波数を指定した係数だけシフトします。

プロパティの概要

プロパティ	解説	範囲	デフォルト
Shift	オーディオ信号の周波数をシフトさせる係数。	0.1 ～ 5	2

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#PitchShiftFilter, [#shift:2])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter =audioFilter(symbol("PitchShiftFilter"),propList(symbol("shift"),2));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ParamEqFilter

解説

パラメトリックイコライザは、指定した周波数の信号を増幅または減衰します。

プロパティの概要

プロパティ	解説	範囲	デフォルト
centerFrequency	中心周波数	20 ～ 96000	8000
bandWidth	帯域幅。	20 ～ 96000	200
Gain	db 単位のゲイン。	-20 ～ 20	0

例文

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audiofilter(#ParamEqFilter,[#centerFrequency:8000, #bandWidth:200, #gain:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

//JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("ParamEqFilter"),propList(symbol("centerFrequency"),8000,symbol("bandWidth"),
200,symbol("gain"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ByteArray

Lingo オブジェクト ByteArray は、バイナリデータの読み取り、書き込み、より高度な処理をサポートします。ByteArray は、JavaScript および Xtras でアクセスできるスクリプトオブジェクトです。

ByteArray キャストメンバ

#byteArray キャストメンバは、Director、Shockwave、および Projector で使用できます。このスクリプト専用キャストメンバには、制限付きのユーザインターフェイスがあります。

UI による ByteArray メンバの作成 挿入／メディアエレメント／バイト配列を選択して、ByteArray キャストメンバを挿入します。

Lingo による ByteArray メンバの作成 new メソッドを使用して、ByteArray キャストメンバを作成します。

使用法

```
m = new(#byteArray)
```

ByteArray メンバのプロパティ

ByteArray キャストメンバには、1 つのプロパティ、byteArray があります。

ByteArray オブジェクトの演算子の概要

演算子	戻り値の型
ランダムアクセス演算子らんだむあくせすえんざんし	整数
String	ストリング

ByteArray オブジェクトのプロパティの概要

プロパティ名	タイプ
BytesRemaining	整数（読み取り専用）
endian	Symbol. 使用できる値は #bigEndian と #littleEndian です。デフォルト値は #littleEndian. です。
length (byte array)	整数（読み取り専用）
position (byte array)	整数

ByteArray オブジェクトのメソッドの概要

メソッド名	
ByteArray	ByteArray(str)
readBoolean	writeBoolean
readInt8 readInt16 readInt32	writeInt8 writeInt16 writeInt32
readFloat32 readFloat64	writeFloat32 writeFloat64
readByteArray	writeByteArray
readString	writeString (byte array)
readRawString	writeRawString
compress()	uncompress()
toHexString	

Lingo 画像オブジェクトとの統合

次のメソッドを使用すると、バイト配列として画像ピクセルバッファにアクセスし、バイト配列からピクセルバッファにデータを入れることができます。

メソッド名
getPixels()
setPixels()

バイト配列をキャストメンバとして読み込む

この新しいメソッド importByteArrayInto() は importFileInto() と同様に、バイト配列コンテンツをメディア出力として読み込みやすいように追加されています。

サウンドオブジェクトの入力としてのバイト配列

バイト配列をサウンドオブジェクトに渡すことができます。バイト配列をファイルから生成する必要がある場合、コールバックメソッドが True を返すたびに、バイト配列は FileIO を使用してファイルのコンテンツを読み取り、バイト配列のコンテンツをサウンドオブジェクトに渡します。

注意：Director のオーディオでは、この操作を行うために、8 ビットの信号なし、および 16 ビットと 24 ビットの信号付きオーディオファイルのみをサポートしています。

使用法

```
SoundObject Mixer.createSoundObject(SoundObjname,callbackFunction,[castMemRef],
[sampleRate,channelcount,bitDepth])
```

パラメータ

パラメータ	解説	デフォルト値	必須 / オプション
SoundObjname	サウンドオブジェクトと関連付ける名前。複数のサウンドオブジェクトに同じ名前を付けることはできません。		必須
callbackFunction	Lingo または JavaScript 関数を createSoundObject に渡すと、サウンドオブジェクトは再生用のデータを受信する準備が整ったときに callbackFunction を呼び出します。		必須
castMemRef	コールバックメソッドが親スクリプトのメンバである場合、キャストメンバ参照を渡します。コールバックメソッドが movieScript 内にある場合は、パラメータを指定しないか、void として指定してください。		オプション
sampleRate	オーディオのサンプリング周波数。	48000 kHz	オプション
channelCount	オーディオのチャネル数。	2	オプション
bitDepth	オーディオのビット深度。	16	オプション

callbackFunction の詳細

Lingo または JavaScript 関数を createSoundObject に渡すと、サウンドオブジェクトは再生用のデータを受信する準備が整ったときに callbackFunction を呼び出します。

または、コールバック関数を使用してキャストメンバの参照を渡すこともできます。

callbackFunction に入力を渡したときの処理

- サウンドオブジェクトが再生されるとすぐに、登録済みのコールバック関数が呼び出されます。
- コールバック関数に渡されるパラメータは長さがゼロのバイト配列です。このバイト配列にオーディオデータが入ります。バイト配列には、無制限の長さのデータをコピーできます。
- データが再生されると、次のコールバック関数が呼び出されます。
- コールバック関数の戻り値は True または False です。True は、サウンドオブジェクトがコールバック関数を次回連続して呼び出すことを示し、False は渡されたすべてのオーディオデータを再生した後に停止することを示します。
- 必要に応じて、コールバック関数内で手動によりサウンドオブジェクトを停止できます。サウンドオブジェクトを手動で停止すると、コールバックメソッドも停止します。

callbackFunction のシンタックスは、以下のとおりです。

```
On callbackFunction outByteArray
--outByteArray is filled with bytearray samples.
end
```

注意：コールバックハンドラの内部では、いずれかのスクリプトで実行時エラー（プロパティが見つからないなど）が発生した場合、エラーメッセージは表示されず、ハンドラの実行はエラーが発生した時点で中止されます。後続のすべてのステートメントは実行されません。

関数へのバイト配列入力には、bytearray サンプルの要求されたバイトが入ります。

注意： bytearray サンプルは、createSoundObject. で指定された形式と同じ形式である必要があります。

createSoundObject で形式を指定しなかった場合、次の仕様のデフォルトの形式が使われます。

ビット深度	16
サンプリング周波数	48000 Hz
チャネル数	2

デフォルトの形式を無効にするには、プロパティリストで次のシンボルを設定します。

```
#bitDepth --Bit depth of the audio.
#sampleRate --Sampling frequency of the audio.
#channelCount --Number of channels in the audio.
```

例文

```
    -- BUTTON BEHAVIOR --
    global gSoundIOInstance --  instance  of FileIO for  reading sound file
    global Sound             -- soundObject
    global gMixer             -- mixer object
    on mouseUp(me)
    ---------------------------------------------------------------------
    -- ACTION: Opens a sound file in the same folder as the movie
    -- and creates a soundObject to play it back. The sound file
    -- will be read in when requested by a callback from the
    -- sound object. The callback handler HAS to be in a Movie Script, which
    -- means that globals MUST be used to play back the sound file.
    -- See the Movie Script for callback details.
    ---------------------------------------------------------------------
-- Open a sample file to read the data
    gSoundIOInstance = new xtra("fileIO")
    gSoundIOInstance.openFile(_movie.path & "RawSound.snd",0)
    gMixer = new(#mixer)
-- Create the sound object with the SampleRate, ChannelCount &
-- BitDepth. Specify the callback method.
    vName = "Imported sound mixer"
    vCallback  = #audioInput
    vByteArray = VOID -- Callback method is available in the Movie Script, can specify
    -- scriptReference also
    vSettings  = [#samplerate:48000, #channelcount:2, #bitdepth:16]
    gSound = gMixer.createSoundObject(\
    vName, \
    vCallback, \
    VOID, \
    vSettings)
-- Play the sound
    gMixer.play()
    end mouseUp
```

コールバックメソッド（ムービースクリプト）

```
-- MOVIE SCRIPT --
global gSoundIOInstance -- instance of FileIO for reading sound file
global gSound           -- soundObject
on audioInput(aByteArray)
-----------------------------------------
-- SOURCE: Called back from gSound whenever gSound has finished
-- playing its current contents.
-- INPUT: <aByteArray> will be an empty byteArray which has to be
-- filled and the modified byteArray to send back.
-- ACTION: Transfers the (remaining) contents of the file to
-- aByteArray
-- OUTPUT: Returns TRUE if there was any data to pas to aByteArray
-- FALSE if not.
-------------------------------------------------------------------
 -- Checking the if we've reached the end of the file
vFileLength = gSoundIOInstance.getLength()
vPosition   = gSoundIOInstance.getPosition()
if vPosition = vFileLength then
-- We've reach the end of the file.  Close it and tell the
-- soundObject to stop calling back.
gSoundIOInstance.closefile()
return FALSE -- stop calling the callback
end if
-- Read entire contents of the file into a byteArray
vByteArray   = gSoundIOInstance.readByteArray(vFileLength)
vArrayLength = vByteArray.length
put vArrayLength, vFileLength
-- Copy from the temporary byteArray to the one passed in as a
-- parameter
if vArrayLength <> 0 then
vResult = aByteArray.writeByteArray(vByteArray, 1, vArrayLength)
end if
-- Ensure this callback is made again when the sound has played out
return TRUE
end audioInput

on stopmovie
gSoundIOInstance.closefile()
end stopmovie
```

サウンドオブジェクトからの出力としてのバイト配列

registerByteArrayCallback および unregisterByteArrayCallback を使用して、バイト配列としてサウンドオブジェクトのコンテンツを取得します。コールバック関数が実行されると、バイト配列には、サウンドオブジェクトから受け取ったオーディオサンプルデータが入ります。サウンドオブジェクトを再生する前に、バイト配列データを変更できます。

Fileio

ファイルの入力および出力操作を実行できます。

new 演算子を使用することで、Fileio オブジェクトへの参照を作成することができます。

```
-- Lingo syntax
objFileio = new xtra("fileio")

// JavaScript syntax
var objFileio = new xtra("fileio");
```

FileIO Xtra は、任意のエンコーディングのファイルを読み書きできます。また、バイト配列を使用してバイナリデータをファイルに書き込んだり、ファイルから読み取ったりすることもできます。

さらに、FileIO は、バイト配列からのデータの直接読み取り、および外部ファイルへのバイト配列の書き込みもサポートするようになっています。

Fileio オブジェクトのメソッドの概要

メソッド	
closeFile()	createFile()
delete() (FileIO)	deleteFile()
displayOpen()	displaySave()
error()	fileName()
getCharSet	getFinderInfo()
getLength()	getOSDirectory()
getPosition()	openFile()
readByteArray (FileIO Xtra)	readChar()
readFile()	readLine()
readToken()	readWord()
setFilterMask()	setFinderInfo()
setNewLineConversion()	setPosition()
status()	setCharSet
writeByteArray (FileIO Xtra)	writeChar()
writeReturn()	writeString()
version()	

MUI Xtra

MUI Xtra は、指定どおりに設定された完全に機能するダイアログボックスを提供します。提供されるダイアログボックスは、ダイアログボックスをシミュレートする MIAW のメモリまたはディスクのフットプリントを必要としません。

new 演算子を使用することで、MUI Xtra オブジェクトへの参照を作成することができます。

```
-- Lingo syntax
objMui = new xtra("Mui")

// JavaScript syntax
var objMui = new xtra("Mui");
```

XML Parser オブジェクトのメソッドの概要

メソッド
Alert()
fileOpen()
fileSave()
GetItemPropList
getURL()
GetWidgetList()
GetWindowPropList
Initialize
ItemUpdate()
run
stop
WindowOperation

NetLingo

ネットワークからのメディアの取得またはストリーミング、ネットワークの利用可能性の確認、ネットワーク操作の進行の確認などのネットワーク操作を実行できます。

new 演算子を使用することで、NetLingo オブジェクトへの参照を作成することができます。

```
-- Lingo syntax
objNetLingo = new xtra("netlingo")

// JavaScript syntax
var objNetLingo = new xtra("netlingo");
```

Director には、インターネット経由でバイト配列を送受信するための Lingo （および JavaScript）のメソッドが用意されています。

NetLingo オブジェクトのメソッドの概要

メソッド	
browserName()	cacheDocVerify()
cacheSize()	clearCache
downloadNetThing	externalEvent()
getLatestNetID	getNetByteArray
getNetText()	getStreamStatus()
gotoNetPage	gotoNetMovie
netAbort	netByteArrayResult

メソッド	
netDone()	netError()
netLastModDate()	netMIME()
netStatus	netTextResult()
postNetByteArray	postNetText
preloadNetThing()	proxyServer
tellStreamStatus()	URLEncode

SpeechXtra

ムービーに音声読み上げ機能を追加できます。

new 演算子を使用することで、SpeechXtra オブジェクトへの参照を作成することができます。

```
-- Lingo syntax
objSpeech = new xtra("speechxtra")

// JavaScript syntax
var objSpeech = new xtra("speechxtra");
```

SpeechXtra オブジェクトのメソッドの概要

メソッド	
voiceCount()	voiceSet()
voiceGet()	voiceSetPitch()
voiceGetAll()	voiceSetRate()
voiceGetPitch()	voiceSetVolume()
voiceGetRate()	voiceSpeak()
voiceGetVolume()	voiceState()
voiceInitialize()	voiceStop()
voicePause()	voiceWordPos()
voiceResume()	

XML Parser

XML 構文解析を実行できます。

new 演算子を使用することで、XML Parser オブジェクトへの参照を作成することができます。

```
-- Lingo syntax
objXml = new xtra("xmlparser")

// JavaScript syntax
var objXml = new xtra("xmlparser");
```

XML Parser オブジェクトのメソッドの概要

メソッド
count()
doneParsing()
getError() (XML)
ignoreWhiteSpace()
makeList()
makeSubList()
parseString()
parseURL()

XML Parser オブジェクトのプロパティの概要

プロパティ
attributeName
attributeValue
child (XML)
name (XML)

XML Xtra

XML ドキュメントは任意のエンコーディングを使用して作成でき、エンコーディング情報は XML 宣言タグを使用して埋め込まれます。

```
<?xml version="1.0" encoding="utf-8"?>
```

さらに、XML ドキュメントは内部にバイナリデータの BLOB を含む可能性があり、その場合は UTF-8 を使用して XML ドキュメント全体を表すことはできません。

Director XML パーサーは、ドキュメントの XML 宣言タグを使用して、ドキュメントのテキストエンコーディングを識別します。

XML Xtra のメソッドの概要

メソッド名
parseByteArray
parseString (XML Xtra)
parseURL (XML Xtra)

第 8 章：3D オブジェクト

3D オブジェクトについて

3D オブジェクトを使用すると、ムービーに 3D 機能を追加できます。これらのオブジェクトは、Director、プロジェクタ、および Adobe® Shockwave® Player 内で、Lingo と JavaScript シンタックス両方に開示されます。

Shockwave 3D（または単に 3D）キャストメンバを通じて、以下に示す 3D オブジェクトにアクセスすることができます。3D キャストメンバから 3D スプライトを作成することもできます。3D キャストメンバと 3D スプライトの両方には、3D キャストメンバおよびスプライトに固有の機能が含まれています。またこれらは非 3D キャストメンバおよびスプライトの機能にもアクセスできます。非 3D キャストメンバおよびスプライトの API は、それぞれ中核的な Member オブジェクトと Sprite オブジェクトにより指定されます。

3D キャストメンバには完全な 3D ワールドが含まれているという点において、3D キャストメンバは非 3D キャストメンバとは異なります。3D ワールドには、3D 機能へのアクセスを提供するオブジェクトが含まれます。3D ワールドのすべてのオブジェクトは、ノードと呼ばれる基本オブジェクトに基づいています。3D ワールドのノードの最も単純な形式はグループオブジェクトです。グループオブジェクトは、その本質上、最も基本的なノードです。3D ワールドのその他すべてのオブジェクトはグループオブジェクトに基づいています。これは、その他のオブジェクトはオブジェクトに固有の機能を含むことに加え、グループオブジェクトの機能を継承していることを意味します。

3D オブジェクトがどのように相互に関連し、また Director の他のオブジェクトとどのように関連するかを示す図については、43 ページの「オブジェクトモデル図」を参照してください。

Director® には、3D オブジェクトへのアクセスを可能にする次の 2 つの Xtra 拡張機能が付属しています。

- 3D Asset Xtra（Windows® では 3DAuth.x32、Mac® では 3D Auth Xtra）は、Director の内部で 3D メディアウィンドウをサポートします。
- 3D Media Xtra（Windows では Shockwave 3D Asset.x32、Mac では 3D Asset Xtra）は、3D メディアそれ自体をサポートします。

  オーサリング時または実行時に 3D オブジェクトにアクセスするには、ムービーに 3D Asset Xtra が含まれている必要があります。

カメラ

カメラオブジェクトを表します。

カメラは、3D スプライトがどのように 3D ワールドを表示するかを制御します。3D スプライトは、特定のカメラから見たワールドのビューです。

3D Member オブジェクトの camera プロパティを使用することで、カメラへの参照を作成することができます。camera プロパティは、カメラのリストの指定されたインデックス位置でカメラを取得します。Lingo では、3D Member オブジェクトから camera プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ「family room」の第 2 のカメラへの参照を作成して、それを変数 myCamera に割り当てます。

```
-- Lingo syntax
myCamera = member("family room").camera[2]

// JavaScript syntax
var myCamera = member("family room").getPropRef("camera", 2);
```

カメラオブジェクトのメソッドの概要

メソッド
addBackdrop
addOverlay
insertBackdrop
insertOverlay
removeBackdrop
removeOverlay

カメラオブジェクトのプロパティの概要

プロパティ	
backdrop	fog.far (fog)
backdrop[].blend (3D)	fog.near (fog)
backdrop[].loc (backdrop and overlay)	hither
backdrop[].regPoint (3D)	orthoHeight
backdrop[].rotation (backdrop and overlay)	overlay
backdrop[].scale (3D)	overlay[].blend (3D)
backdrop[].source	overlay[].loc (backdrop and overlay)
backdrop.count (3D)	overlay[].regPoint (3D)
child (3D)	overlay[].rotation (backdrop and overlay)
colorBuffer.clearAtRender	overlay[].scale (3D)
colorBuffer.clearValue	overlay[].source
fieldOfView (3D)	overlay.count (3D)
fog.color()	projection
fog.decayMode	rootNode
fog.enabled (fog)	yon

関連事項

グループ , ライト , モデル , モデルリソース , モーション , シェーダ , テクスチャ

グループ

リソースまたはシェーダを持たないモデルを表します。

グループは最も基本的なノードで、トランスフォームにより表現されるスペース内のポイントにすぎません。モデル、ライト、カメラまたはその他のグループをグループ化するために、このノードに子や孫を割り当てることができます。

最も基本的なグループはワールドと呼ばれます。本質的には 3D キャストメンバと同じ意味です。

3D Member オブジェクトの group プロパティを使用することで、グループへの参照を作成することができます。group プロパティは、グループのリストの指定されたインデックス位置でグループを取得します。Lingo では、3D Member オブジェクトから group プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ space の最初のグループへの参照を作成して、それを変数 myGroup に割り当てます。

```
-- Lingo syntax
myGroup = member("space").group[1]

// JavaScript syntax
var myGroup = member("space").getPropRef("group", 1);
```

グループオブジェクトのメソッドの概要

メソッド	
addChild	pointAt
addToWorld	registerScript()
clone	removeFromWorld
cloneDeep	rotate
getWorldTransform()	scale （コマンド）
isInWorld()	translate

グループオブジェクトのプロパティの概要

プロパティ
name (3D)
parent
pointAtOrientation
transform （プロパティ）
userData
worldPosition

関連事項

カメラ , ライト , モデル , モデルリソース , モーション , シェーダ , テクスチャ

ライト

3D ワールドでのライトを表します。

ライトは、3D ワールドの照明に使用します。ライトがないと、ワールド内のオブジェクトは見えません。

3D Member オブジェクトの light プロパティを使用することで、ライトへの参照を作成することができます。light プロパティは、ライトのリストの指定されたインデックス位置でライトを取得します。Lingo では、3D Member オブジェクトから light プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ「film room」の第 3 のライトへの参照を作成して、それを変数 myLight に割り当てます。

```
-- Lingo syntax
myLight = member("film room").light[3]

// JavaScript syntax
var myLight = member("film room").getPropRef("light", 3);
```

ライトオブジェクトのプロパティの概要

プロパティ
attenuation
color (light)
specular (light)
spotAngle
spotDecay
type (light)

関連事項

カメラ , グループ , モデル , モデルリソース , モーション , シェーダ , テクスチャ

メンバ

Shockwave 3D キャストメンバを表します。

Shockwave 3D（または単に 3D）キャストメンバは、それ自身が 1 つの完全な 3D ワールドです。3D ワールドには、ムービーに 3D 機能を追加するために使用するオブジェクトのセットが含まれています。

最上位の member() 関数を使用するか、ムービーまたはスプライトオブジェクトの member プロパティを使用することで、3D キャストメンバへの参照を作成することができます。これらは、非 3D キャストメンバへの参照を作成するために使用できるテクニックと同じです。

- 最上位の member() 関数を使用します。

  ```
  -- Lingo syntax
  3dMember = member("magic")

  // JavaScript syntax
  var 3dMember = member("magic");
  ```

- スプライトオブジェクトの member プロパティを使用します。

```
-- Lingo syntax
3dMember = sprite(1).member;

// JavaScript syntax
var 3dMember = sprite(1).member;
```

メンバオブジェクトのメソッドの概要

メソッド	
camera()	model (3D)
cloneModelFromCastmember	modelResource
cloneMotionFromCastmember	motion()
deleteCamera	newCamera
deleteGroup	newGroup
deleteLight	newLight
deleteModel	newMesh
deleteModelResource	newModel
deleteMotion	newModelResource
deleteShader	newShader
deleteTexture	newTexture
extrude3D	resetWorld
group()	revertToWorldDefaults
light()	
loadFile()	

メンバオブジェクトのプロパティの概要

プロパティ	
ambientColor	loop (3D)
animationEnabled	model
bevelDepth	modelResource
bevelType	motion
bytesStreamed (3D)	percentStreamed (3D)
camera	preLoad (3D)
cameraPosition	reflectivity
cameraRotation	shader
diffuseColor	smoothness
directionalColor	specularColor

プロパティ	
directionalPreset	state (3D)
directToStage	streamSize (3D)
displayFace	texture
displayMode	textureMember
group	textureType
light	tunnelDepth

関連事項

カメラ , グループ , ライト , モデル , モデルリソース , モーション , シェーダ , スプライト , テクスチャ

モデル

3D ワールド内でユーザが参照する、表示されるオブジェクトを表します。

モデルは、モデルリソースを使用して、3D ワールド内の特定の位置と向きに表示されるオブジェクトです。モデルリソースとは、3D モデルの描画に使用できる 3D ジオメトリのエレメントのことです。また、モデルは、使用するテクスチャやシェーダなど、モデルリソースの外観も定義します。モデルとモデルリソースの間の関係について詳しくは、Director ヘルプパネルの「Director ユーザガイド」トピックを参照してください。

3D Member オブジェクトの model プロパティを使用することで、モデルへの参照を作成することができます。model プロパティは、モデルのリストの指定されたインデックス位置でモデルを取得します。Lingo では、3D Member オブジェクトから model プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ Transportation の第 2 のモデルへの参照を作成して、それを変数 myModel に割り当てます。

```
-- Lingo syntax
myModel = member("Transportation").model[2]

// JavaScript syntax
var myModel = member("Transportation").getPropRef("model", 2);
```

モデルには、モデルのレンダリング方法とそのアニメーションの動作方法を制御するモディファイアが含まれています。モディファイアをモデルに割り当てるには、addModifier() メソッドを使用します。モデルに割り当てたモディファイアのプロパティは、スクリプトを使用して操作できます。

モデルには次のモディファイアが使用できます。

モディファイア	解説
ボーンプレーヤ	時間の経過と共にモデルのジオメトリを変化させます。
衝突	モデルで衝突の通知を受けたり、衝突に応答したりできるようになります。
インカー	既存のモデルにシルエット、折り目、および境界のエッジを追加します。
キーフレームプレーヤ	時間の経過と共にモデルの transform プロパティを変化させます。
レベルオブディテール（LOD）	モデルのレンダリングに使用するポリゴンの数を、カメラからのモデルの距離に基づいてモデルごとに制御します。 LOD モディファイアは、モデルリソースにも使用できます。

モディファイア	解説
メッシュ変形	実行時に既存のモデルリソースのジオメトリを変更します。
サブディビジョンサーフェス（SDS）	現在カメラが見ているモデルの領域のレンダリングに、詳細なジオメトリを追加します。
トゥーン処理	モデルのレンダリングをカートゥーン描画スタイルに変更します。

モディファイアに使用できるメソッド、プロパティ、およびイベントについて詳しくは、Director ヘルプパネルの「Director ユーザガイド」トピックを参照してください。

モデルリソース

3D モデルの描画に使用される 3D ジオメトリのエレメントを表します。

モデルは、モデルリソースを使用して、3D ワールド内の特定の位置と向きに表示されるオブジェクトです。また、モデルは、使用するテクスチャやシェーダなど、モデルリソースの外観も定義します。

モデルとモデルリソースの間の関係、およびモデルとモデルリソースの使用法について詳しくは、Director ヘルプパネルの「Director ユーザガイド」トピックを参照してください。

3D Member オブジェクトの modelResource プロパティを使用することで、モデルリソースへの参照を作成することができます。modelResource プロパティは、モデルリソースのリストの指定されたインデックス位置でモデルリソースを取得します。Lingo では、3D Member オブジェクトから modelResource プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ wheels の第 2 のモデルリソースへの参照を作成して、それを変数 myModelResource に割り当てます。

```
-- Lingo syntax
myModelResource = member("wheels").modelResource[2]

// JavaScript syntax
var myModelResource = member("wheels").getPropRef("modelResource", 2);
```

モーション

モデルやモデルコンポーネントの動きを決定する、定義済みのアニメーションシーケンスを表します。

モーションは設定により、単独で再生することも、他のモーションと共に再生することもできます。例えば、走るモーションとジャンプするモーションを組み合わせて、人が水たまりを飛び越える様子をシミュレートできます。

3D Member オブジェクトの motion プロパティを使用することで、モーションへの参照を作成することができます。motion プロパティは、モーションのリストの指定されたインデックス位置でモーションを取得します。Lingo では、3D Member オブジェクトから motion プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ athlete の第 4 のモーションへの参照を作成して、それを変数 myMotion に割り当てます。

```
-- Lingo syntax
myMotion = member("athlete").motion[4]

// JavaScript syntax
var myMotion = member("athlete").getPropRef("motion", 4);
```

レンダラサービス

すべての 3D キャストメンバおよびスプライトに使用する共通のレンダリングプロパティに影響する値を持つプロパティリストを含むグローバルオブジェクトを表します。

最上位の getRendererServices() 関数を使用することで、グローバルレンダラサービスオブジェクトにアクセスすることができます。

次の例では、グローバルレンダラサービスオブジェクトの renderer プロパティにアクセスし、変数 myRenderer に値を割り当てます。

```
-- Lingo syntax
myRenderer = getRendererServices().renderer

// JavaScript syntax
var myRenderer = getRendererServices().renderer;
```

レンダラサービスオブジェクトのメソッドの概要

メソッド
getHardwareInfo()

レンダラサービスオブジェクトのプロパティの概要

プロパティ
modifiers
primitives
renderer
rendererDeviceList
textureRenderFormat

関連事項

メンバ , スプライト

シェーダ

モデルの表面の色を表します。

画像は、テクスチャを各シェーダに適用することで、モデルのサーフェスに描画できます。

3D Member オブジェクトの shader プロパティを使用することで、シェーダへの参照を作成することができます。shader プロパティは、シェーダのリストの指定されたインデックス位置でシェーダを取得します。Lingo では、3D Member オブジェクトから shader プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ triangle の第 2 のシェーダへの参照を作成して、それを変数 myShader に割り当てます。

```
-- Lingo syntax
myShader = member("triangle").shader[2]

// JavaScript syntax
var myShader = member("triangle").getPropRef("shader", 2);
```

スプライト

Shockwave 3D キャストメンバから作成された 3D スプライトを表します。

最上位の sprite() 関数、ムービーオブジェクトの sprite プロパティ、またはスプライトチャネルオブジェクトの sprite プロパティを使用することで、3D スプライトへの参照を作成することができます。これらは、非 3D スプライトへの参照を作成するために使用できるテクニックと同じです。

- 最上位の sprite() 関数を使用します。

```
-- Lingo syntax
3dSprite = sprite(1)

// JavaScript syntax
var 3dSprite = sprite(1);
```

- ムービーオブジェクトの sprite プロパティを使用します。

```
-- Lingo syntax
3dSprite = _movie.sprite["willowTree"]

// JavaScript syntax
var 3dSprite = _movie.sprite["willowTree"];
```

- スプライトチャネルオブジェクトの sprite プロパティを使用します。

```
-- Lingo syntax
3dSprite = channel(3).sprite

// JavaScript syntax
var 3dSprite = channel(3).sprite;
```

スプライトオブジェクトのメソッドの概要

メソッド
addCamera
cameraCount()
deleteCamera

スプライトオブジェクトのプロパティの概要

プロパティ
antiAliasingEnabled
backColor
camera
directToStage

関連事項

カメラ , メンバ

テクスチャ

シェーダに適用されたテクスチャを表します。

3D Member オブジェクトの texture プロパティを使用することで、テクスチャへの参照を作成することができます。texture プロパティは、テクスチャのリストの指定されたインデックス位置でテクスチャを取得します。Lingo では、3D Member オブジェクトから texture プロパティを直接使用して参照を作成します。JavaScript シンタックスでは、参照を作成するには getPropRef() メソッドを使用する必要があります。

次の例では、3D キャストメンバ triangle の最初のテクスチャへの参照を作成して、それを変数 myTexture に割り当てます。

```
-- Lingo syntax
myTexture = member("triangle").texture[1]

// JavaScript syntax
var myTexture = member("triangle").getPropRef("texture", 1);
```

第 9 章：定数

この項には、Director® で使用できるすべての定数のアルファベット順リストが記載されています。

以下に示す定数の大部分は Lingo にのみ適用されます。JavaScript シンタックスには、この項に示されている Lingo の定数に類似する定数がいくつか含まれています。そのため該当する場合は、Lingo の定数の機能と、JavaScript シンタックスにある最も近い相当物を対応させることができるように、JavaScript シンタックスの使用法と例が記載されています。JavaScript シンタックスの定数について詳しくは、この主題に関する多数のサードパーティのリソースのいずれかを参照してください。

"（ストリング）

使用法

```
--Lingo syntax
"

// JavaScript syntax
"
```

解説

ストリングの前後の二重引用符は、そのストリングがリテラル値であり、変数や数値、スクリプトエレメントではないことを表します。キャストメンバ、キャスト、ウィンドウ、外部ファイルなどのリテラル名は、常にこの二重引用符で囲みます。

例文

次のステートメントは、二重引用符を使って「サンフランシスコ」というストリングがリテラルストリングであることを表しています。この例では、キャストメンバの名前になっています。

```
--Lingo syntax
put member("San Francisco").loaded

// JavaScript syntax
put(member("San Francisco").loaded);
```

関連事項

QUOTE

BACKSPACE

使用法

```
-- Lingo syntax
BACKSPACE

// JavaScript syntax
51 // value of _key.keyCode
```

解説

Backspace キーを表す定数です。このキーのラベルは Backspace（Windows®）および Delete（Mac®）です。

例文

次の on keyDown ハンドラは、BackSpace キーが押されたかどうかを調べ、押された場合は clearEntry ハンドラを呼び出します。

```
--Lingo syntax
on keyDown
    if (_key.key = BACKSPACE) then clearEntry
    _movie.stopEvent()
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 51) {
        clearEntry();
        _movie.stopEvent();
    }
}
```

EMPTY

使用法

```
--Lingo syntax
EMPTY

// JavaScript syntax
""
```

解説

空のストリング (文字を全く含まないストリング、"") を表す文字定数です。

例文

次のステートメントは、フィールドを「EMPTY」に設定することによって「お知らせ」フィールドキャストメンバ内の文字をすべて削除します。

```
--Lingo syntax
member("Notice").text = EMPTY

// JavaScript syntax
member("Notice").text = "";
```

ENTER

使用法

```
--Lingo syntax
ENTER

// JavaScript syntax
3 // value of _key.keyCode
```

解説

改行を意味する Enter（Windows の場合）または Return（Macintosh の場合）を表す文字定数です。

PC 互換機のキーボードの場合、ここで指す ENTER は、テンキーにある Enter です。

アプレットとして再生されるムービーでは、RETURN を指定することで、Return（Windows の場合）および Enter（Macintosh の場合）の両方を表せます。

例文

次のステートメントは、Enter が押されたかどうかを調べ、押されていれば再生ヘッドをフレーム addSum まで移動します。

```
-- Lingo syntax
on keyDown
    if (_key.key = ENTER) then _movie.go("addSum")
end

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 3) {
        _movie.go("addSum");
    }
}
```

関連事項

RETURN (定数)

FALSE

使用法

```
-- Lingo syntax
FALSE

// JavaScript syntax
false
```

解説

"2 > 3" のように論理的に FALSE（偽）になる式に適用される論理定数です。数値を使って処理する場合、FALSE（偽）は 0 になります。逆に 0 は FALSE（偽）と見なされます。

例文

次のステートメントは、「soundEnabled」プロパティを FALSE に設定することによって無効な状態にします。

```
-- Lingo syntax
_sound.soundEnabled = FALSE

// JavaScript syntax
_sound.soundEnabled = false;
```

関連事項

if, not, TRUE

PI

使用法

```
-- Lingo syntax
PI

// JavaScript syntax
Math.PI
```

解説

円周率 (π) の値を浮動小数で返す定数です。円周率は、円周を直径で割った値です。この値は、floatPrecision プロパティで設定した精度の浮動小数に四捨五入されます。

例文

次のステートメントは、円の面積を計算するために PI 定数を使っています。

```
-- Lingo syntax
vRadius = 3
vArea = PI*power(vRadius, 2)
trace(vArea) -- results in 28.2743

// JavaScript syntax
var vRadius = 3;
vArea = Math.PI*Math.pow(vRadius, 2);
trace(vArea); // results in 28.274333882308138
```

QUOTE

使用法

```
--Lingo syntax
QUOTE

// JavaScript syntax
\"
```

解説

二重引用符文字を表す文字定数です。ストリング内で二重引用符をストリングの一部として用いるときに使用します (単に二重引用符を入力しても Lingo はストリングを区切る制御コードとして解釈してしまいます)。

例文

次のステートメントは、二重引用符を文字としてストリング内に挿入します。

```
-- Lingo syntax
put("Can you spell" && QUOTE & "Adobe" & QUOTE & "?")

// JavaScript syntax
put("Can you spell \"Adobe\"?");
```

結果は、次のように Adobe® が二重引用符で囲まれたストリングになります。

```
Can you spell "Adobe"?
```

RETURN (定数)

使用法

```
-- Lingo syntax
RETURN

// JavaScript syntax
36 // value of _key.keyCode
\n // when used in a string
```

解説

キャリッジリターン (改行) キーを表す文字定数です。

例文

次のステートメントは、ユーザがキャリッジリターンキーを押したときに、一時停止していたムービーを続行させます。

```
-- Lingo syntax
if (_key.key = RETURN) then _movie.go(_movie.frame + 1)

// JavaScript syntax
if (_key.keyCode == 36) {
    _movie.go(_movie.frame + 1);
}
```

次のステートメントは、RETURN 文字定数を使って、警告メッセージ内の 2 つの行間にキャリッジリターンキーを挿入します。

```
-- Lingo syntax
_player.alert("Last line in the file." & RETURN & "Click OK to exit.")

// JavaScript syntax
_player.alert("Last line in the file." + "\n" + " Click OK to exit");
```

Windows では、各行の終わりにラインフィード文字を追加することが一般的です。次のステートメントは、ラインフィードを追加する 2 文字のストリング「CRLF」を作成します。

```
CRLF = RETURN & numToChar(10)
```

SPACE

使用法

```
-- Lingo syntax
SPACE

// JavaScript syntax
49 // value of _key.keyCode
```

解説

空白文字を表す読み取り専用の定数です。

例文

次のステートメントは「Age Of Aquarius」をメッセージウィンドウに表示します。

```
-- Lingo syntax
put("Age"&SPACE&"Of"&SPACE&"Aquarius")
```

TAB

使用法

```
-- Lingo syntax
TAB

// JavaScript syntax
48 // value of _key.keyCode
```

解説

Tab キーを表す定数です。

例文

次のステートメントは、タイプされたキーが Tab かどうかをチェックし、Tab の場合は doNextField ハンドラを呼び出します。

```
-- Lingo syntax
if (_key.key = TAB) then doNextField

// JavaScript syntax
if (_key.keyCode == 48) {
    doNextField();
}
```

次のステートメントは、ユーザが Tab キーまたは Shift+Tab キーを押したときに、再生ヘッドを前または後ろに移動させます。

```
-- Lingo syntax
if (_key.key = TAB) then
    if (_key.shiftDown) then
        _movie.go(_movie.frame - 1)
    else
        _movie.go(_movie.frame + 1)
    end if
end if

// JavaScript syntax
if (_key.keyCode == 48) {
    if (_key.shiftDown) {
        _movie.go(_movie.frame - 1);
    } else {
        _movie.go(_movie.frame + 1);
    }
}
```

関連事項

BACKSPACE, EMPTY, RETURN (定数)

TRUE

使用法

```
-- Lingo syntax
TRUE

// JavaScript syntax
true
```

解説

"2 < 3" のように、論理的に真となる式の値を表す定数です。TRUE は伝統的に数値の 1 で表されますが、比較では 0 以外のすべての整数が TRUE として評価されます。

例文

次のステートメントは、soundEnabled プロパティを TRUE に設定することによってこれをオンにします。

```
-- Lingo syntax
_sound.soundEnabled = TRUE

// JavaScript syntax
_sound.soundEnabled = true;
```

関連事項

FALSE, if

VOID

使用法

```
-- Lingo syntax
VOID

// JavaScript syntax
null
```

解説

値がないこと (VOID) を意味する定数です。

例文

次のステートメントは、変数 currentVariable の値が VOID かどうかをチェックします。

```
-- Lingo syntax
if currentVariable = VOID then
    put("This variable has no value")
end if

// JavaScript syntax
if (currentVariable == undefined) {
    put("This variable has no value");
}
```

関連事項

voidP()

第 10 章：イベントとメッセージ

この項には、Director® で使用できるすべてのイベントとメッセージのアルファベット順リストが記載されています。

on activateApplication

使用法

```
-- Lingo syntax
on activateApplication
    statement(s)
end

// JavaScript syntax
function activateApplication() {
    statement(s);
}
```

解説

プロジェクタを前面に表示すると実行されるビルトインハンドラです。このハンドラは、ウィンドウ内でプロジェクタを再生しながら、他のアプリケーションと切り替えるような場合に役に立ちます。プロジェクタが前面にもどると、このハンドラが起動します。MIAW をプロジェクタで実行する場合にもこのハンドラを利用できます。

■訳文不要■

Windows® 環境では、プロジェクタを最小化しても、他のアプリケーションが前面に表示されていない場合は、このハンドラは起動しません。

例文

次のハンドラは、プロジェクタが前面にもどるたびにサウンドを再生します。

```
-- Lingo syntax
on activateApplication
    sound(1).queue(member("openSound"))
    sound(1).play()
end

// JavaScript syntax
function activateApplication() {
    sound(1).queue(member("openSound"));
    sound(1).play();
}
```

関連事項

on deactivateApplication, activeCastLib, on deactivateWindow

on activateWindow

使用法

```
-- Lingo syntax
on activateWindow
        statement(s)
end

// JavaScript syntax
function activateWindow()
    statement(s);
}
```

解説

ユーザが非アクティブウィンドウをクリックし、そのウィンドウが前景になったときに実行するステートメントを含むシステムメッセージとイベントハンドラです。

ムービーがアクティブになるたびに実行されるようにしたいスクリプトには on activateWindow ハンドラを使用します。

メインムービー (メインステージ) をクリックしても on activateWindow ハンドラは生成されません。

例文

次のハンドラは、ムービーを再生するウィンドウがアクティブになったときにサウンド「万歳」を再生します。

```
-- Lingo syntax
on activateWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function activateWindow() {
    sound(2).play(member("Hurray"));
}
```

関連事項

activeWindow, close(), on deactivateWindow, frontWindow, on moveWindow, open() （ウィンドウ）

on beginSprite

使用法

```
-- Lingo syntax
on beginSprite
    statement(s)
end

// JavaScript syntax
function beginSprite() {
    statement(s);
}
```

解説

まだ検出されていないスプライトを含むフレームに再生ヘッドが入ったときに実行するステートメントを定義する、システムメッセージとイベントハンドラです。endSprite と同様、再生ヘッドがフレームをループしていても、このイベントは 1 回だけ生成されます。これは、まだ再生ヘッドが検出していないスプライトを見つけたときにのみ生成されるためです。このイベントは prepareFrame イベントの前に生成されます。

beginSprite メッセージが送られると、Director はスプライトに割り当てられているすべてのビヘイビアースクリプトのインスタンスを作成します。

このイベントがビヘイビアーで使われていると、オブジェクト参照 me が渡されます。このメッセージはビヘイビアーとフレームスクリプトに送られます。

ムービー内で再生される最初のフレームでスプライトが開始する場合は、beginSprite メッセージは prepareMovie メッセージの後、prepareFrame および startMovie メッセージの前に発生します。

注意：rect などいくつかのスプライトプロパティは、beginSprite ハンドラからはアクセスできません。これは、スプライトが描画される時点までに必要なプロパティの値が計算されないためです。

go、play、updateStage コマンドは、on beginSprite ハンドラ内では無効です。

例文

次のハンドラは、スプライトが開始するときにサウンドキャストメンバ " スティービー・ワンダー " を再生します。

```
-- Lingo syntax
on beginSprite me
    sound(1).play(member("Stevie Wonder"))
end

// JavaScript syntax
function beginSprite() {
    sound(1).play(member("Stevie Wonder"));
}
```

関連事項

on endSprite, on prepareFrame, scriptInstanceList

on closeWindow

使用法

```
-- Lingo syntax
on closeWindow
    statement(s)
end

// JavaScript syntax
function closeWindow() {
    statement(s);
}
```

解説

ウィンドウ内ムービーとして再生されているムービーのウィンドウをユーザがクローズボックスをクリックして閉じたときに実行されるシステムメッセージとイベントハンドラです。

on closeWindow ハンドラは、ムービーのウィンドウが閉じるたびに実行する Lingo を入れるのに適しています。

例文

次のハンドラは、再生していたムービーのウィンドウが閉じたとき、現行ウィンドウに forget コマンドを実行します。

```
-- Lingo syntax
on closeWindow
    -- perform general housekeeping here
    window(1).forget()
end

// JavaScript syntax
function closeWindow() {
    // perform general housekeeping here
    window(1).forget();
}
```

on cuePassed

使用法

```
-- Lingo syntax
on cuePassed({me,} channelID, cuePointNumber,cuePointName)
    statement(s)
end

// JavaScript syntax
function cuePassed(channelID, cuePointNumber,cuePointName) {
    statement(s);
}
```

解説

メディア内でサウンドまたはスプライトがキューポイントを通過するたびに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

- me オプションの me パラメータは呼び出されるスクリプトの scriptInstanceRef 値です。ビヘイビアーでメッセージを使用するときは、このパラメータを使用する必要があります。このパラメータが省略されると、他のパラメータが正しく処理されません。
- channelID キューポイントが発生したファイルのサウンドまたはスプライトチャネルの番号を示します。
- cuePointNumber キャストメンバのキューポイントのリスト内のイベントを引き起こすキューポイントの順序番号です。
- cuePointName 検出されたキューポイントの名前です。

  メッセージは、スプライト、キャストメンバ、フレーム、ムービースクリプトの順に送られます。QuickTime® ムービーまたは SWA キャストメンバのようなサウンドのソースである場合、スプライトがイベントを受信することができます。isPastCuePoint プロパティを使って、サウンドを生成しないスプライト上のビヘイビアーのキューを確認します。

例文

ムービーまたはフレームスクリプトに含まれる次のハンドラは、サウンドチャネル 1 のキューポイントをすべてメッセージウィンドウに報告します。

```
-- Lingo syntax
on cuePassed channel, number, name
    if (channel = #Sound1) then
        put("CuePoint" && number && "named" && name && "occurred in sound 1")
    end if
end

// JavaScript syntax
function cuePassed(channel, number, name) {
    if (channel == symbol("Sound1")) {
        put("CuePoint " + number + " named " + name + "occurred in sound 1");
    }
}
```

関連事項

scriptInstanceList, cuePointNames, cuePointTimes, isPastCuePoint()

on deactivateApplication

使用法

```
-- Lingo syntax
on deactivateApplication
    statement(s)
end

// JavaScript syntax
function deactivateApplication() {
    statement(s);
}
```

解説

プロジェクタが他のアプリケーションのバックグラウンドにまわると実行されるビルトインハンドラです。このハンドラは、ウィンドウ内でプロジェクタを再生しながら、他のアプリケーションと切り替えるような場合に役に立ちます。MIAW をプロジェクタで実行する場合にもこのハンドラを利用できます。

■訳文不要■

Windows 環境では、プロジェクタを最小化しても、他のアプリケーションが前面に表示されていない場合は、このハンドラは起動しません。

例文

次のハンドラは、プロジェクタがバックグラウンドになるたびにサウンドを再生します。

```
-- Lingo syntax
on deactivateApplication
    sound(1).queue(member("closeSound"))
    sound(1).play()
end

// JavaScript syntax
function deactivateApplication() {
    sound(1).queue(member("closeSound"));
    sound(1).play();
}
```

関連事項

add (3D texture), activeCastLib, on deactivateWindow

on deactivateWindow

使用法

```
-- Lingo syntax
on deactivateWindow
    statement(s)
end

// JavaScript syntax
function deactivateWindow() {
    statement(s);
}
```

解説

ムービーが再生されているウィンドウがアクティブでなくなったときに実行されるステートメントを含むシステムメッセージおよびイベントハンドラです。on deactivate イベントハンドラは、ウィンドウがアクティブでなくなったときに必ず実行される Lingo を入れる場所として適しています。

例文

このハンドラは、ムービーが再生されているウィンドウがアクティブでなくなったときに、「いびき音」サウンドを鳴らします。

```
-- Lingo syntax
on deactivateWindow
    sound(2).play(member("Snore"))
end

// JavaScript syntax
function deactivateWindow() {
    sound(2).play(member("Snore"));
}
```

on DVDeventNotification

使用法

```
-- Lingo syntax
on DVDeventNotification objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}
    statement(s)
end DVDeventNotification

// JavaScript syntax
function DVDeventNotification (objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}) {
    statement(s);
}
```

解説

作成者が指定する DVD イベントハンドラ。DVD の再生中に発生するイベントに応答して実行されるステートメントが含まれています。

このハンドラを使用するとすべての DVD イベントを追跡できます。上記のスクリプト例では、DVDeventNotification ハンドラに渡される最初のパラメータ **objectRef** は、DVDeventNotification オブジェクトそれ自体への参照です。実際に発生するイベントは、常に第 2 のパラメータ **event** として渡されます。イベントの中には、第 3 のパラメータ **eventArg1** として渡される、イベントに関する追加情報が含まれるものもあります。場合によっては、第 4 および第 5 のパラメータである **eventArg2** および **eventArg3** に追加のイベント情報が含まれる場合もあります。

次の表に、DVD の再生中に発生する可能性があるイベントを示します。

イベント	解説
angleChange	使用可能なアングルの数が変化した場合、または現在のユーザアングル番号が変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 使用可能なアングルの数を示す整数。使用可能なアングルの数が 1 である場合、現在のビデオはマルチアングルではありません。 • **eventArg3** - 現在のユーザアングル番号を示す整数。
audioStreamChange	メインタイトルの現在のユーザオーディオストリーム番号が変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 新しいユーザオーディオストリーム番号を示す整数。ストリーム 0xFFFFFFFF は、ストリームが選択されていないことを示します。
buttonChange	使用可能なボタンの数が変化した場合、または現在選択されているボタン番号が変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 使用可能なボタンの数を示す整数。 • **eventArg3** - 現在選択されているボタン番号を示す整数。選択されているボタン番号が 0 であれば、ボタンが選択されていないことを意味します。
chapterAutoStop	自動停止の結果として再生が停止した場合に発生します。
chapterStart	title ドメインの新しいプログラムの再生が始まった場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 新しいチャプタ番号を示す整数。
diskEjected	DVD が取り出された場合に発生します。
diskInserted	DVD が挿入された場合に発生します。
domainChange	DVD プレーヤのドメインが変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg1**。新しいドメインを示す値。新しいドメインは次のいずれかの値になります。 • firstPlay。DVD ナビゲータが DVD のデフォルトの初期化を実行しています。 • videoManagerMenu。DVD ナビゲータがディスク全体のメニューを再生しています。 • videoTitleSetMenu。DVD ナビゲータが現在のタイトルセットのメニューを再生しています。 • title。DVD ナビゲータが現在のタイトルを再生しています。 • stop。DVD ナビゲータが stop ドメインにあります。

イベント	解説
エラー	DVD エラー状態が発生した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2**。エラー状態を示す値。エラー状態は次のいずれかの値になります。 • copyProtectFail。DVD コピー保護用のキー交換に失敗しました。再生が停止します。 • invalidDVD1_0Disc。DVD-Video ディスクは、仕様バージョン 1.**x** に対して正しくオーサリングされていません。再生が停止します。 • invalidDiscRegion。ディスクが当該システムリージョンで再生されるようにはオーサリングされていないため、DVD-Video ディスクを再生できません。 • lowParentalLevel。プレーヤの親レベルが、DVD コンテンツで使用できる最低の親レベルよりも低くなっています。再生が停止します。 • macrovisionFail。マクロビジョンディストリビューションに失敗しました。再生が停止します。 • incompatibleSystemAndDecoderRegions。システムリージョンがデコーダリージョンと一致しないため、ディスクを再生できません。 • incompatibleDiscAndDecoderRegions。ディスクがデコーダのリージョンで再生されるようにはオーサリングされていないため、ディスクを再生できません。 • unexpected。想定外のエラーが発生しました。通常はコンテンツが正しくオーサリングされていません。再生が停止します。
karaokeMode	オーディオモードが karaoke に設定されている場合に発生します。
noFirstPlayChain	DVD ディスクに FP_PGC（First Play Program Chain）がなく、DVD ナビゲータが自動的には PGC をロードせず、再生を開始しない場合に発生します。
parentalLevelChange	オーサリングされたコンテンツの親レベルが変化しようとするときに発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2**。プレーヤで設定されている新しい親レベルを示す整数。
playbackStopped	再生が停止した場合に発生します。DVD ナビゲータは PGC の再生を完了していますが、それ以降を再生するためのその他の分岐命令が見つかりません。
playPeriodAutoStop	自動停止の結果として再生が停止した場合に発生します。
rateChange	再生レートが変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2**。新しい再生レートを示す整数。0 よりも小さい（<）値は、逆再生モードを示します。0 よりも大きい（>）値は、通常方向の再生モードを示します。この値は、実際の再生レートに 10,000 を掛けた値です。
stillOff	任意のスチル（PGC、Cell、または VOBU）の終了時点で発生します。
stillOn	任意のスチル（PGC、Cell、または VOBU）の開始時点で発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - ボタンが使用可能であるかどうかを示すブール値。ゼロ（0）はボタンが使用可能であることを示します。1 はボタンが使用できないことを示します。 • **eventArg3** - スチルが継続する秒数を示す整数またはアドレス。0xFFFFFFFF は無限のスチルを示します。

イベント	解説
titleChange	現在のタイトル番号が変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 新しいタイトル番号を示す整数またはアドレス。
UOPchange	使用可能な再生または検索メカニズムのいずれかが変化した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - DVD ディスクがどの再生または検索メカニズムを明示的に無効にしたかを示す整数またはアドレス。
warning	DVD 警告状態が発生した場合に発生します。 このイベントが発生した場合、次の追加情報が DVDeventNotification に渡されます。 • **eventArg2** - 警告状態を示す整数またはアドレス。警告状態は次のいずれかの値になります。 • invalidDVD1_0Disc。DVD-Video ディスクが正しくオーサリングされていません。再生は継続可能ですが、想定外の動作が生じる場合があります。 • formatNotSupported。デコーダは現在の形式をサポートしていません。ストリームの再生は機能しない場合があります。 • illegalNavCommand。内部の DVD ナビゲーションコマンドプロセッサが無効なコマンドを処理しようとしました。 • open。 • seek。 • read。

関連事項

DVD

on endSprite

使用法

```
-- Lingo syntax
on endSprite
    statement(s)
end

// JavaScript syntax
function endSprite() {
    statement(s);
}
```

解説

再生ヘッドがスプライトからスプライトなしのフレームに移動すると実行される Lingo を含む、システムメッセージおよびイベントハンドラです。これは exitFrame の後に生成されます。

on endSprite ハンドラはビヘイビアースクリプトに入れます。

スプライトに割り当てられたすべてのビヘイビアースクリプトのインスタンスは、endSprite イベントが発生した直後に破棄されます。

ビヘイビアー内で使われる場合、イベントハンドラには、そのビヘイビアーまたはフレームスクリプトの参照である "me" が渡されます。再生ヘッドがフレームの最後まで移動すると、この endSprite メッセージが exitFrame メッセージの後に送られます。

go()、play()、updateStage() メソッドは、on endSprite ハンドラ内では無効です。

例文

次のハンドラは、再生ヘッドがスプライトを抜け出したときに実行されます。

```
-- Lingo syntax
on endSprite me
    -- clean up
    gNumberOfSharks = gNumberOfSharks - 1
    sound(5).stop()
end

// JavaScript syntax
function endSprite() {
    // clean up
    gNumberOfSharks--;
    sound(5).stop();
}
```

関連事項

on beginSprite, on exitFrame

on enterFrame

使用法

```
-- Lingo syntax
on enterFrame
    statement(s)
end

// JavaScript syntax
function enterFrame() {
    statement(s);
}
```

解説

再生ヘッドがフレームに入るたびに実行されるステートメントを含むシステムメッセージおよびイベントハンドラです。

on enterFrame ハンドラは以下のように、用途に応じて、ビヘイビアー、フレーム、ムービースクリプトのいずれかに入れます。

- ハンドラを個々のスプライトに割り当てる場合は、スプライトに割り当てられているビヘイビアー内にハンドラを入れます。
- ハンドラを個々のフレームに割り当てる場合は、フレームスクリプト内にハンドラを入れます。
- すべてのフレームにハンドラを割り当てる場合、明示的にムービーに指示するのでなければ、ムービースクリプトに on enterFrame ハンドラを入れます。このハンドラは、再生ヘッドが個々のフレームに入るたびに実行されます。ただし、フレームスクリプトが独自のハンドラを持っている場合は実行されません。フレームスクリプトが独自のハンドラを持つ場合は、ムービースクリプトの on enterFrame ハンドラではなく、フレームスクリプトの on enterFrame ハンドラが有効になります。

フレームイベントは、stepFrame、prepareFrame, enterFrame、exitFrame の順で発生します。

ビヘイビアー内で使用される場合、イベントにはそのオブジェクトの参照である "me" が渡されます。

例文

次のハンドラは、再生ヘッドがフレームに入るたびにスプライト 1 ～ 5 のパペット条件をオフにします。

```
-- Lingo syntax
on enterFrame
    repeat with i = 1 to 5
        _movie.puppetSprite(i, FALSE)
    end repeat
end

// JavaScript syntax
function enterFrame() {
    for (i=1;i<=5;i++) {
        _movie.puppetSprite(i, false);
    }
}
```

on EvalScript

使用法

```
-- Lingo syntax
on EvalScript aParam
    statement(s)
end

// JavaScript syntax
function EvalScript(aParam) {
    statement(s);
}
```

解説

Adobe® Shockwave® コンテンツが付属するムービー内のシステムメッセージおよびイベントハンドラです。ハンドラがブラウザから EvalScript メッセージを受け取ると実行されるステートメントが含まれます。パラメータはブラウザから渡されるストリングです。

- EvalScript メッセージには、Director が Lingo ステートメントとして解釈できるストリングを入れることができます。Lingo はネストしたストリングを処理できないため、ストリングのパラメータをとるようなハンドラを呼び出す場合は、パラメータをシンボルとして渡します。
- on EvalScript ハンドラがブラウザの JavaScript または VBScript の EvalScript() スクリプトメソッドによって呼び出されます。

on EvalScript には、セキュリティを考慮して、ユーザによる制御を許可したいビヘイビアーのみが含まれるようにします。すべてのビヘイビアーにアクセスできるようなハンドラを作成しないでください。

注意：EvalScript ハンドラの末尾が「return」で終わるようにすると、返される値はブラウザの JavaScript で使用できます。

例文

次の例では、パラメータとして渡されるフレームに従って、再生ヘッドが特定のフレームにジャンプします。

```
-- Lingo syntax
on EvalScript aParam
    _movie.go(aParam)
end

// JavaScript syntax
function EvalScript(aParam) {
    _movie.go(aParam);
}
```

次のハンドラは、パラメータとして犬、猫、木のいずれかを含む EvalScript メッセージを受け取った場合、_movie.go(aParam) ステートメントを実行します。

```
-- Lingo syntax
on EvalScript aParam
    case aParam of
        "dog", "cat", "tree": _movie.go(aParam)
    end case
end

// JavaScript syntax
function EvalScript(aParam) {
    switch(aParam) {
        case "dog", "cat", "tree": _movie.go(aParam);
    }
}
```

これを呼び出すための JavaScript ステートメントは、例えば EvalScript ("dog") のようになります。

次のハンドラは、数字またはシンボルをパラメータとして受け取れます。

```
-- Lingo syntax
on EvalScript aParam
    if word 1 of aParam = "myHandler" then
        _movie.go(aParam)
    end if
end

// JavaScript syntax
function EvalScript(aParam) {
    if (aParam.indexOf("myHandler",0)) {
        _movie.go(aParam);
    }
}
```

次のハンドラは、通常、パラメータとしてストリングを要求します。パラメータをシンボルとして受け取り、string 関数を使ってハンドラ内でストリングに変換します。

```
-- Lingo syntax
on myHandler aParam
    _movie.go(string(aParam))
end

// JavaScript syntax
function myHandler(aParam) {
    _movie.go(aParam.toString());
}
```

関連事項

externalEvent(), return （キーワード）

on exitFrame

使用法

```
-- Lingo syntax
on exitFrame
    statement(s)
end

// JavaScript syntax
function exitFrame() {
    statement(s);
}
```

解説

再生ヘッドが on exitFrame ハンドラが割り当てられているフレームから抜け出すたびに実行されるステートメントを含む、システムメッセージおよびイベントハンドラです。on exitFrame ハンドラは、フレームを抜け出した時点で不適切になる状況をリセットする Lingo を置く場所として適しています。

on exitFrame ハンドラは以下のように、用途に応じてビヘイビアー、フレームスクリプト、ムービースクリプトのいずれかに入れます。

- ハンドラを個々のスプライトに割り当てる場合は、スプライトに割り当てられているビヘイビアー内にハンドラを入れます。
- ハンドラを個々のフレームに割り当てる場合は、フレームスクリプト内にハンドラを入れます。
- すべてのフレームに有効なハンドラを割り当てる場合は、ムービースクリプト内に入れます。フレームスクリプト内に独自の on exitFrame ハンドラを持っていない限り、再生ヘッドがフレームを抜け出るたびにムービースクリプト内の on exitFrame ハンドラが実行されます。フレームスクリプトが独自の on exitFrame ハンドラを含んでいる場合、フレームスクリプト内の on exitFrame ハンドラが実行され、ムービースクリプト内のハンドラは無効になります。

ビヘイビアー内で使用される場合、イベントにはそのスプライトスクリプトまたはフレームスクリプトの参照である "me" が渡されます。フレームイベントが生成されるのは、prepareFrame、enterFrame、exitFrame の順です。

例文

次のハンドラは、再生ヘッドがフレームを抜け出すときにすべてのパペット設定をオフにします。

```
-- Lingo syntax
on exitFrame me
    repeat with i = 48 down to 1
        sprite(i).scripted = FALSE
    end repeat
end

// JavaScript syntax
function exitFrame() {
    for (i=48; i>=1; i--);
        sprite(i).scripted = false;
    }
}
```

次のハンドラは、再生ヘッドがフレームを抜け出す時点でグローバル変数 vTotal の値が 1000 を超えている場合に、再生ヘッドを特定のマーカに移動させます。

```
// JavaScript syntax
function exitFrame() {
    if (_global.vTotal > 1000) {
        _movie.go("Finished");
    }
}
```

関連事項

on enterFrame

on getBehaviorDescription

使用法

```
-- Lingo syntax
on getBehaviorDescription
    statement(s)
end

// JavaScript syntax
function getBehaviorDescription() {
    statement(s);
}
```

解説

ビヘイビアを選択したときに、ビヘイビアインスペクタの説明ペインに表示されるストリングを返す Lingo を含むシステムメッセージとイベントハンドラです。

詳細ストリングはオプションです。

ビヘイビアインスペクタが開くときに、Director はスプライトに割り当てられたビヘイビアに getBehaviorDescription メッセージを送信します。ビヘイビアーに on getBehaviorDescription ハンドラを入れます。

ハンドラには、説明を複数行で記述するために、埋め込み改行文字を入れることができます。

例文

次のステートメントは、説明ペインに「テキストフィールド用縦スクロールバー」と表示します。

```
-- Lingo syntax
on getBehaviorDescription
    return "Vertical Multiline textField Scrollbar"
end

// JavaScript syntax
function getBehaviorDescription() {
    return "Vertical Multiline textField Scrollbar";
}
```

関連事項

on getPropertyDescriptionList, on getBehaviorTooltip, on runPropertyDialog

on getBehaviorTooltip

使用法

```
-- Lingo syntax
on getBehaviorTooltip
    statement(s)
end

// JavaScript syntax
function getBehaviorTooltip() {
    statement(s);
}
```

解説

ライブラリパレットのスクリプトのツールチップに表示されるストリングを返す Lingo を含むシステムメッセージとイベントハンドラです。

カーソルがライブラリパレット上にあるときに、Director は getBehaviorTooltip メッセージをスクリプトに送信します。ビヘイビアーに on getBehaviorTooltip ハンドラを入れます。

ハンドラの使用はオプションです。ハンドラがない場合は、ツールチップにはキャストメンバ名が表示されます。

ハンドラには、説明を複数行で記述するために、埋め込み改行文字を入れることができます。

例文

次のステートメントは、説明パネルに「ジグソーパズルのピース」を表示します。

```
-- Lingo syntax
on getBehaviorTooltip
    return "Jigsaw puzzle piece"
end

// JavaScript syntax
function getBehaviorTooltip() {
    return "Jigsaw puzzle piece";
}
```

関連事項

on getPropertyDescriptionList, on getBehaviorDescription, on runPropertyDialog

on getPropertyDescriptionList

使用法

```
-- Lingo syntax
on getPropertyDescriptionList
    statement(s)
end

// JavaScript syntax
function getPropertyDescriptionList() {
    statement(s);
}
```

解説

ビヘイビアーの [パラメータ] ダイアログボックスに表示されるパラメータの定義やラベルのリストを生成する Lingo を含むシステムメッセージやイベントハンドラです。

on getPropertyDescriptionList ハンドラは、ビヘイビアースクリプト内に入れます。on getPropertyDescriptionList ハンドラが含まれないビヘイビアーは、パラメータダイアログボックスに表示されないため、Director のインターフェースからは編集できません。

ビヘイビアインスペクタが開くアクションが発生すると、つまりユーザがビヘイビアーをスコアにドラッグするか、ユーザがビヘイビアインスペクタでビヘイビアーをダブルクリックすると、on getPropertyDescriptionList メッセージが送信されます。

#default、#format、#comment は、各パラメータに対して設定する必要があります。これらの設定の値としては、次のようなものが使われます。

#default	パラメータの初期設定です。
#format	#integer #float #string #symbol #member #bitmap #filmloop #field #palette #picture #sound #button #shape #movie #digitalvideo #script #richtext #ole #transition #xtra #frame #marker #ink #boolean
#comment	パラメータダイアログボックスの編集可能なフィールドの左側に表示されるパラメータのラベルです。
#range	プロパティに割り当てることができる値の範囲です。範囲はリニアリストとしていくつかの値で指定されるか、またはプロパティリスト [#min: minValue, #max: maxValue] の形式の最小値と最大値として指定されます。

例文

次のハンドラは、[パラメータ] ダイアログボックスに表示されるビヘイビアーのパラメータを定義します。addProp で始まる各ステートメントは description というリストにパラメータを追加します。リストに追加された各エレメントは、プロパティおよびそのプロパティの #default、#format、#comment の値を定義します。

```
on getPropertyDescriptionList
    description = [:]
    description.addProp(#dynamic, [#default:1, #format:#boolean, #comment:"Dynamic"])
    description.addProp(#fieldNum, [#default:1, #format:#integer,#comment:"Scroll which sprite:"])
    description.addProp(#extentSprite, [#default:1,#format:#integer,#comment: "Extend Sprite:"])
    description.addProp(#proportional, [#default:1,#format:#boolean, #comment: "Proportional:"])
    return description
end
```

関連事項

addProp, on getBehaviorDescription, on runPropertyDialog

on hyperlinkClicked

使用法

```
-- Lingo syntax
on hyperlinkClicked me, data, range
    statement(s)
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    statement(s);
}
```

解説

ハイパーリンクが実際にクリックされたときの実行内容を定義するシステムメッセージとイベントハンドラです。

このイベントハンドラのパラメータは、次のとおりです。

- me ビヘイビアー内で、そのス ライトインスタンスを識別するために使用します。
- data ハイパーリンクのデータ自体です。テキストキャストメンバの編集時にテキストインスペクタに入力されるストリングです。
- range テキストでハイパーテキ トとして割り当てられている部分の範囲を示します。該当するテキストは、キャストメンバ .char[range[1]..range[2]] というシンタックスで得ることができます。

このハンドラは、ビヘイビアースクリプトとしてスプライトに添付してください。キャストメンバスクリプト内に記述することは避けてください。

例文

次のビヘイビアーは、クリックされたハイパーリンクを調べるためのリンクを示し、必要に応じて該当する URL にジャンプし、メッセージウィンドウにリンクのテキストを表示します。

```
property spriteNum
on hyperlinkClicked(me, data, range)
    if data starts "http://" then
        gotoNetPage(data)
    end if
    currentMember = sprite(spriteNum).member
    anchorString = currentMember.char[range[1]..range[2]]
    put("The hyperlink on"&&anchorString&&"was just clicked.")
end
// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st = ht) {
        gotoNetPage(data);
    }
    var currentMember = sprite(this.spriteNum).member;
    var r1 = currentMember.getPropRef("char", range[1]).hyperlinkRange;
    var a = r1[1] - 1;
    var b = r1[2];
    var st = new String(currentMember.text);
    var anchorString = st.slice(a, b);
    put("The hyperlink on " + anchorString + " was just clicked.");
}
```

on idle

使用法

```
-- Lingo syntax
on idle
    statement(s)
end

// JavaScript syntax
function idle() {
    statement(s);
}
```

解説

ムービー内に処理するイベントが他に存在しないときに実行されるステートメントを含むシステムメッセージとイベントハンドラです。グローバル変数値や現在のムービーの状況を示す表示を更新するステートメントのように、なるべく頻繁に実行したい Lingo ステートメントを入れる場所として適しています。

on idle ハンドラ内のステートメントは頻繁に実行されるので、**on idle** ハンドラには、処理に時間を要する Lingo は入れないようにします。

on idle ハンドラは、ムービースクリプトではなくフレームスクリプトに入れておくと、状況に応じて on idle ハンドラを無効にできます。

idle イベントの処理中に、内部または外部キャストからキャストメンバを読み込むことはできます。ただし、リンクされたキャストメンバを idle イベント中に読み込むことはできません。

idle メッセージは、フレームスクリプトおよびムービースクリプトにのみ送られます。

例文

次のハンドラは、他に処理するイベントが存在しないときにムービー内に表示される " 時間 " を更新します。

```
-- Lingo syntax
on idle
    member("Time").text = _system.time()
end idle

// JavaScript syntax
function idle() {
    member("Time").text = _system.time();
}
```

関連事項

idleHandlerPeriod

on isOKToAttach

使用法

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    statement(s)
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    statement(s)
}
```

解説

組み込みハンドラ。ビヘイビアーに追加することにより、ビヘイビアーが添付されているスプライトのタイプを調べたり、不適切なスプライトタイプにビヘイビアーが添付されるのを防げます。

ビヘイビアーをスプライトに添付すると、このハンドラが実行され、スプライトのタイプとスプライト番号を渡します。パラメータ me にはスプライトに添付するビヘイビアーへの参照が含まれています。

このハンドラの実行後に、on getPropertyDescriptionList ハンドラが実行されます。

オーサリング時には、2 種類のスプライトを調べることができます。#graphic には、シェイプ、ビットマップ、デジタルビデオ、テキストなどのすべてのグラフィックキャストメンバが含まれています。#script は、ビヘイビアーがスクリプトチャネルに添付されたことを知らせます。この場合、[スプライト番号] は 1 になります。

上記のスプライトタイプごとに、ハンドラは TRUE または FALSE を返す必要があります。値が TRUE のときは、そのビヘイビアーをスプライトに添付できます。値が FALSE のときは、ビヘイビアーはスプライトには添付されません。

ビヘイビアーに on isOKToAttach ハンドラが記述されていない場合は、そのビヘイビアーを任意のスプライトまたはフレームに添付できます。

このハンドラは、ビヘイビアが最初にスプライトまたはスクリプトチャネルに添付されるときに呼び出されます。また、ビヘイビアインスペクタを使用して新しいビヘイビアを添付するときも、このハンドラが呼び出されます。

例文

次のステートメントは、ビヘイビアーが添付されているスプライトのタイプを調べ、シェイプ以外のグラフィックスプライトであれば TRUE を返し、スクリプトチャネルであれば FALSE を返します。

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    case aSpriteType of
        #graphic: -- any graphic sprite type
            return sprite(aSpriteNum).member.type <> #shape
            -- works for everything but shape cast members
        #script: --the frame script channel
            return FALSE -- doesn't work as a frame script
    end case
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    switch (aSpriteType) {
        case symbol("graphic"): // any graphic sprite type
            return sprite(aSpriteNum).member.type != symbol("shape");
            // works for everything but shape cast members
        case symbol("script"): // the frame script channel
            return false; // doesn't work as a frame script
    }
}
```

on keyDown

使用法

```
-- Lingo syntax
on keyDown
    statement(s)
end

// JavaScript syntax
function keyDown() {
    statement(s);
}
```

解説

キーが押されたときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

キーが押されると、Director は、初期イベントハンドラ、編集可能フィールドスプライトスクリプト、フィールドキャストメンバのスクリプト、フレームスクリプト、ムービースクリプトの順番で、on keyDown ハンドラを検索します。スプライトとキャストメンバでは、on keyDown ハンドラは編集可能なテキストメンバとフィールドメンバに対してのみ動作します。ビットマップなど異なる種類のキャストメンバでは、keyDown イベントは効果がありません。ムービー全体を通して、キーが押されたときの反応を同じにするには、keyDownScript プロパティを設定します。

検索は、最初に on keyDown ハンドラを含むスクリプトを見つけた時点で止まります。ただし、次の場所に keyDown メッセージを渡すことを明示する pass コマンドが、そのハンドラ内に含まれている場合を除きます。

on keyDown イベントハンドラは、ユーザがキーを押したときに実行するキーボードショートカットやインターフェイス機能などを設定するのに適しています。

ムービーがアプレットとして再生されている場合、on keyDown ハンドラはハンドラが空の場合でもキーの押下を捕らえます。編集可能フィールドにキー入力した文字がフィールドに表示されるようにするには、フィールドに添付する on keyDown ハンドラに pass コマンドを入れておく必要があります。

on keyDown ハンドラを置く場所に応じて、その動作が次のように異なります。

- ハンドラを特定の編集可能フィールドスプライトに適用するには、そのスプライトのスクリプト内にハンドラを置きます。
- ハンドラを一般の編集可能フィールドキャストメンバに適用するには、そのキャストメンバのスクリプト内にハンドラを置きます。
- ハンドラをフレーム全体に適用するには、フレームスクリプト内にハンドラを置きます。
- ハンドラをムービー全体に適用するときは、ムービースクリプト内にハンドラを置きます。

on keyDown ハンドラを無効にする場合は、Lingo の検索順でそのハンドラよりも前に、別の on keyDown ハンドラを設定します。例えば、キャストメンバに割り当てた on keyDown を無効にするには、スプライトスクリプト内に別の on keyDown ハンドラを設定します。

例文

次のハンドラは、Return キーが押されたかどうかをチェックし、押された場合は再生ヘッドを別のマーカ "AddSum" に移動します。

```
-- Lingo syntax
on keyDown
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

関連事項

charToNum(), keyDownScript, keyUpScript, key, keyCode, keyPressed()

on keyUp

使用法

```
-- Lingo syntax
on keyUp
    statement(s)
end

// JavaScript syntax
function keyUp() {
    statement(s);
}
```

解説

キーが離されたときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。on keyUp ハンドラは、on keyDown ハンドラと似ていますが、フィールドまたはテキストスプライトが画面上で編集可能である場合、文字が表示された後にこのイベントが起きるという点で異なります。

キーが離されると、Lingo は、初期イベントハンドラ、編集可能フィールドスプライトスクリプト、フィールドキャストメンバのスクリプト、フレームスクリプト、ムービースクリプトの順番で、on keyUp ハンドラを検索します。スプライトとキャストメンバでは、on keyUp ハンドラは編集可能なメンバまたはスプライトに対してのみ動作します。ビットマップなど異なる種類のキャストメンバでは、keyUp イベントは効果がありません。ムービー全体を通して、キーが離されたときの反応を同じにするには、keyUpScript プロパティを設定します。

検索は、最初に on keyUp ハンドラを含むスクリプトを見つけた時点で止まります。ただし、次の場所に keyUp メッセージを渡すことを明示する pass コマンドが、そのハンドラ内に含まれている場合を除きます。

on keyUp イベントハンドラは、ユーザがキーを離したときに実行するキーボードショートカットやインターフェイス機能などを設定するのに適しています。

ムービーがアプレットとして再生されている場合、on keyUp ハンドラはハンドラが空の場合でもキーの押し下げを捕らえます。編集可能フィールドにキー入力した文字がフィールドに表示されるようにするには、フィールドに添付する on keyUp ハンドラに pass コマンドを入れておく必要があります。

on keyUp ハンドラを置く場所に応じて、その動作が異なります。

- ハンドラを特定の編集可能フィールドスプライトに適用するには、ビヘイビアー内にハンドラを置きます。
- ハンドラを一般の編集可能フィールドキャストメンバに適用するには、キャストメンバスクリプト内にハンドラを置きます。
- ハンドラをフレーム全体に適用するときは、フレームスクリプト内にハンドラを置きます。
- ハンドラをムービー全体に適用するときは、ムービースクリプト内にハンドラを置きます。

on keyUp ハンドラを無効にする場合は、Lingo の検索順でそのハンドラよりも前に、別の on keyUp ハンドラを設定します。例えば、キャストメンバに割り当てた on keyUp ハンドラを無効にするには、スプライトスクリプト内に別の on keyUp ハンドラを設定します。

例文

次のハンドラは、Return キーが離されたかどうかをチェックし、離された場合は再生ヘッドを別の "AddSum" マーカに移動します。

```
-- Lingo syntax
on keyUp
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyUp

// JavaScript syntax
function keyUp() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

関連事項

on keyDown, keyDownScript, keyUpScript

on mouseDown (イベントハンドラ)

使用法

```
-- Lingo syntax
on mouseDown
    statement(s)
end

// JavaScript syntax
function mouseDown() {
    statement(s);
}
```

解説

マウスボタンが押されているときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

マウスボタンが押されると、Lingo は、初期イベントハンドラ、スプライトスクリプト、キャストメンバスクリプト、フレームスクリプト、ムービースクリプトの順番で、onmouseDown ハンドラを検索します。on mouseDown ハンドラを含む最初の場所を見つけたときに、Lingo は検索を中止します。ただし、次の場所に mouseDown メッセージを渡すことを明示する pass コマンドが、そのハンドラ内に含まれている場合を除きます。

マウスボタンを押したときに、ムービー全体を通して同じ処理を実行するには、mouseDownScript を設定するか、ムービースクリプトに on mouseDown ハンドラを入れます。

on mouseDown イベントハンドラは、マウスボタンが押されたときに、画像の点滅、サウンド効果、スプライトの移動などを実行する Lingo を置く場所として適しています。

on mouseDown ハンドラを置く場所によって、適用範囲が異なります。

- ハンドラを特定のスプライトに適用するには、スプライトのスクリプト内にハンドラを置きます。
- ハンドラを一般のキャストメンバに適用するときは、そのキャストメンバのスクリプト内にハンドラを置きます。
- ハンドラをフレーム全体に適用するときは、フレームスクリプト内にハンドラを置きます。
- ハンドラをムービー全体に適用するときは、ムービースクリプト内にハンドラを置きます。

on mouseDown ハンドラを無効にする場合は、Lingo のハンドラの検索順でそのハンドラよりも前に別の on mouseDown ハンドラを置きます。例えば、キャストメンバに割り当てた on mouseDown ハンドラを無効にするときには、スプライトスクリプト内に別の on mouseDown ハンドラを置きます。

ハンドラをビヘイビアーで使用する場合は、スプライトスクリプトまたはフレームスクリプトの参照 "me" がこのイベントに渡されます。

例文

次のハンドラは、ユーザがステージ内でクリックしたかどうかを調べ、クリックした場合は、再生ヘッドを "AddSum" マーカに移動します。

```
-- Lingo syntax
on mouseDown
    if (_mouse.clickOn = 0) then _movie.go("AddSum")
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 0) {
        _movie.go("AddSum");
    }
}
```

次のハンドラは、スプライトスクリプトに割り当てられており、スプライトがクリックされると「クリック音」サウンドを再生します。

```
-- Lingo syntax
on mouseDown
    sound(1).play(member("Crickets"))
end

// JavaScript syntax
function mouseDown() {
    sound(1).play(member("Crickets"));
}
```

関連事項

clickOn, mouseDownScript, mouseUpScript

on mouseEnter

使用法

```
-- Lingo syntax
on mouseEnter
    statement(s)
end

// JavaScript syntax
function mouseEnter() {
    statement(s);
}
```

解説

マウスポインタが初めてスプライトのアクティブ領域に入るときに実行するステートメントを含むシステムメッセージとイベントハンドラです。マウスボタンを押していなくても実行されます。

スプライトがビットマップキャストメンバであり、Matte （光沢のない）インクを適用されている場合、アクティブな領域は表示される画像の部分となります。それ以外の場合は、アクティブな領域は、スプライトの境界矩形です。

ハンドラをビヘイビアーで使用する場合は、スプライトスクリプトまたはフレームスクリプトの参照 "me" がこのイベントに渡されます。

例文

次の単純なボタンビヘイビアーは、マウスポインタがボタンの上へ、またボタンの外へと移動したときに、ボタンのビットマップを切り替えます。

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

関連事項

on mouseLeave, on mouseWithin

on mouseLeave

使用法

```
-- Lingo syntax
on mouseLeave
    statement(s)
end

// JavaScript syntax
function mouseLeave() {
    statement(s);
}
```

解説

マウスポインタがスプライトのアクティブ領域から出るときに実行するステートメントを含むシステムメッセージとイベントハンドラです。マウスボタンを押していなくても実行されます。

スプライトがビットマップキャストメンバであり、Matte（光沢のない）インクを適用されている場合、アクティブな領域は表示される画像の部分となります。それ以外の場合は、アクティブな領域は、スプライトの境界矩形です。

ハンドラをビヘイビアーで使用する場合は、スプライトスクリプトまたはフレームスクリプトの参照 "me" がこのイベントに渡されます。

例文

次のステートメントは、マウスポインタがボタンの上へ、またボタンの外へと移動したときに、ボタンのビットマップを切り替える簡単なボタンビヘイビアーです。

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

関連事項

on mouseEnter, on mouseWithin

on mouseUp (イベントハンドラ)

使用法

```
-- Lingo syntax
on mouseUp
    statement(s)
end

// JavaScript syntax
function mouseUp() {
    statement(s);
}
```

解説

マウスボタンが離されたときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

マウスボタンが離されるとと、Lingo は、初期イベントハンドラ、スプライトスクリプト、キャストメンバスクリプト、フレームスクリプト、ムービースクリプトの順番で、on mouseUp ハンドラを検索します。検索は、最初に on mouseUp ハンドラを含むスクリプトを見つけた時点で止まります。ただし、次の場所に mouseUp メッセージを渡すことを明示する pass コマンドが、そのハンドラ内に含まれている場合を除きます。

マウスボタンを離したときに、ムービー全体にわたり同じ処理を実行させるには、the mouseUpScript を設定します。

on mouseUp イベントハンドラは、ボタンなどをクリック後に視覚的に変化させる Lingo を置く場所として適しています。ユーザがスプライトをクリックしてからマウスボタンを離した時点で、そのスプライトに割り当てているキャストメンバを入れ替えると、スプライトがクリックされたことを視覚的に表すことができます。

以下に示すように、on mouseUp ハンドラを置く場所によって、適用範囲が異なります。

- ハンドラを特定のスプライトに適用するには、スプライトのスクリプト内にハンドラを置きます。
- ハンドラを一般のキャストメンバに適用するときは、そのキャストメンバのスクリプト内にハンドラを置きます。
- ハンドラをフレーム全体に適用するときは、フレームスクリプト内にハンドラを置きます。
- ハンドラをムービー全体に適用するときは、ムービースクリプト内にハンドラを置きます。

on mouseUp ハンドラを無効にするには、Lingo のハンドラの検索順でそのハンドラよりも前に別の on mouseUp ハンドラを置きます。例えば、キャストメンバに割り当てた on mouseUp ハンドラを無効にするときは、スプライトスクリプト内に別の on mouseUp ハンドラを置きます。

ハンドラをビヘイビアーで使用する場合は、スプライトスクリプトまたはフレームスクリプトの参照 "me" がこのイベントに渡されます。

例文

次のハンドラは、ユーザがスプライトをクリックしてマウスボタンを離した時点で、スプライト 10 に割り当てられているキャストメンバを入れ替えます。

```
-- Lingo syntax
on mouseUp
    sprite(10).member = member("Dimmed")
end

// JavaScript syntax
function mouseUp() {
    sprite(10).member = member("Dimmed");
}
```

関連事項

on mouseDown (イベントハンドラ)

on mouseUpOutside

使用法

```
-- Lingo syntax
on mouseUpOutside me
    statement(s)
end

// JavaScript syntax
function mouseUpOutside() {
    statement(s);
}
```

解説

ユーザがマウスボタンをスプライト領域内で押し、スプライト領域外に出てから離したときに送信されるシステムメッセージとイベントハンドラです。

例文

次のステートメントは、ユーザがマウスボタンをスプライト領域内で押し、スプライトの境界矩形の外側で離したときにサウンドを再生します。

```
-- Lingo syntax
on mouseUpOutside me
    sound(1).play(member("Professor Long Hair"))
end

// JavaScript syntax
function mouseUpOutside() {
    sound(1).play(member("Professor Long Hair"));
}
```

関連事項

on mouseEnter, on mouseLeave, on mouseWithin

on mouseWithin

使用法

```
-- Lingo syntax
on mouseWithin
    statement(s)
end

// JavaScript syntax
function mouseWithin() {
    statement(s);
}
```

解説

マウスポインタがスプライトのアクティブ領域の内部にあるときに実行するステートメントを含むシステムメッセージとイベントハンドラです。マウスボタンを押していなくても実行されます。

スプライトがビットマップキャストメンバであり、matte インクが適用されている場合、アクティブな領域は画像の表示される部分になります。それ以外の場合は、スプライトの境界矩形がアクティブな領域になります。

ハンドラをビヘイビアーで使用する場合は、スプライトスクリプトまたはフレームスクリプトの参照 "me" がこのイベントに渡されます。

例文

次のステートメントは、マウスポインタがスプライトの上にあるときに、マウスの水平位置を表示します。

```
-- Lingo syntax
on mouseWithin
    member("Display").text = string(_mouse.mouseH)
end

// JavaScript syntax
function mouseWithin() {
    member("Display").text = _mouse.mouseH.toString();
}
```

関連事項

on mouseEnter, on mouseLeave

on moveWindow

使用法

```
-- Lingo syntax
on moveWindow
    statement(s)
end

// JavaScript syntax
function moveWindow() {
    statement(s);
}
```

解説

ステージ上でムービーを別の場所にドラッグするなどして、ウィンドウが移動したときに実行されるステートメントを含むシステムメッセージとイベントハンドラです。このハンドラは、ムービーのウィンドウが移動されるたびに実行する Lingo を置く場所として適しています。

例文

次のハンドラは、ムービーが再生されているウィンドウが移動したときに、メッセージウィンドウにメッセージを表示します。

```
-- Lingo syntax
on moveWindow
    put("Just moved window containing" && _movie.name)
end

// JavaScript syntax
function moveWindow() {
    put("Just moved window containing " + _movie.name);
}
```

関連事項

activeWindow, name (3D), windowList

on openWindow

使用法

```
-- Lingo syntax
on openWindow
    statement(s)
end

// JavaScript syntax
function openWindow() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラです。Director がウィンドウ内のムービーとしてムービーを開くときに実行するステートメントを含みます。on openWindow は、ウィンドウ内でムービーが開くたびに実行する Lingo を置く場所として適しています。

例文

次のハンドラは、ムービーを再生するウィンドウが開いたときにサウンドメンバ「万歳」を再生します。

```
-- Lingo syntax
on openWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function openWindow() {
    sound(2).play(member("Hurray"));
}
```

on prepareFrame

使用法

```
-- Lingo syntax
on prepareFrame
    statement(s)
end

// JavaScript syntax
function prepareFrame {
    statement(s);
}
```

解説

現在のフレームが描かれる前に実行するステートメントを含む、システムメッセージとイベントハンドラです。

beginSprite および endSprite イベントと異なり、prepareFrame イベントは、再生ヘッドがフレームに入るたびに生成されます。

on prepareFrame ハンドラは、スプライトが描かれる前にスプライトプロパティを変更する場所として有効です。

ビヘイビアーで使う場合、on prepareFrame ハンドラにはスクリプトインスタンスの参照 me が渡されます。

go、play、および updateStage コマンドは、on prepareFrame ハンドラで使用することはできません。

例文

次のハンドラは、ビヘイビアーが割り当てられているスプライトの locH プロパティを設定します。

```
-- Lingo syntax
on prepareFrame me
    sprite(me.spriteNum).locH= _mouse.mouseH
end

// JavaScript syntax
function prepareFrame() {
    sprite(spriteNum).locH= _mouse.mouseH;
}
```

関連事項

on enterFrame

on prepareMovie

使用法

```
-- Lingo syntax
on prepareMovie
    statement(s)
end

// JavaScript syntax
function prepareMovie() {
    statement(s);
}
```

解説

ムービーがキャストメンバをプリロードした後で、しかもムービーが次のことを開始する前に実行されるステートメントを含むシステムメッセージとイベントハンドラです。

- 再生される最初のフレーム内のスプライトにビヘイビアーのインスタンスを作成します。
- 再生される最初のフレームを準備します。これには、フレームを描く、いずれかのサウンドを再生する、トランジションやパレット効果を行うなどが含まれます。

最初のフレーム内のスプライトビヘイビアーに使われる新規のグローバル変数は、onprepareMovie ハンドラで初期設定される必要があります。以前のムービーで既に設定済みのグローバル変数は、設定しなおす必要はありません。

on prepareMovie ハンドラは、グローバル変数の作成、変数の初期化、ムービーの残りをメモリにロードする間のサウンドの再生、または色深度などのコンピュータの状態のチェックおよび調整を行う Lingo を置く場所として適しています。

go、play、updateStage コマンドは、on prepareMovie ハンドラ内では無効です。

例文

次のハンドラは、ムービーが開始したときにグローバル変数を作成します。

```
-- Lingo syntax
on prepareMovie
    global currentScore
    currentScore = 0
end

// JavaScript syntax
function prepareMovie() {
    _global.currentScore = 0;
}
```

関連事項

on enterFrame, on startMovie

on resizeWindow

使用法

```
-- Lingo syntax
on resizeWindow
    statement(s)
end

// JavaScript syntax
function resizeWindow() {
    statement(s);
}
```

解説

ムービーがウィンドウ内ムービー（MIAW）として再生されているときに、ユーザがウィンドウのサイズ変更ボックスまたはエッジの一部をドラッグしてウィンドウのサイズを変更したときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

on resizeWindow イベントハンドラは、スプライトの配置やビデオのクロップなど、ウィンドウのサイズと関連する Lingo を入れておくのに適しています。

例文

次のハンドラは、ムービーを再生中のウィンドウのサイズが変更されたときに、スプライト 3 を変数 centerPlace に格納されている座標まで移動します。

```
-- Lingo syntax
on resizeWindow centerPlace
    sprite(3).loc = centerPlace
end

// JavaScript syntax
function resizeWindow(centerPlace) {
    sprite(3).loc = centerPlace;
}
```

関連事項

drawRect, sourceRect

on rightMouseDown (イベントハンドラ)

使用法

```
-- Lingo syntax
on rightMousedown
    statement(s)
end

// JavaScript syntax
function rightMouseDown() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。Windows でマウスの右ボタンが押されたときに実行するステートメントを指定します。Mac では、emulateMultiButtonMouse プロパティが TRUE になっている場合、マウスボタンと Ctrl キーが同時に押されたときにステートメントが実行されます。FALSE に設定されているときは、このイベントハンドラは Mac では何の意味もありません。

例文

次のステートメントは、Windows でユーザがマウスの右ボタンをクリックしたときに、ウィンドウ " ヘルプ " を開きます。

```
-- Lingo syntax
on rightMousedown
    window("Help").open()
end

// JavaScript syntax
function rightMouseDown() {
    window("Help").open();
}
```

on rightMouseUp (イベントハンドラ)

使用法

```
-- Lingo syntax
on rightMouseUp
    statement(s)
end

// JavaScript syntax
function rightMouseUp() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。Windows でマウスの右ボタンが離されたときに実行するステートメントを指定します。Mac では、emulateMultiButtonMouse プロパティが TRUE の場合は、Ctrl キーを押しながらマウスボタンを離すとステートメントが実行されます。プロパティが FALSE に設定されていると、このイベントハンドラは Mac では何の意味もありません。

例文

次のハンドラは、Windows でユーザがマウスの右ボタンを離したときにヘルプウィンドウを開きます。

```
-- Lingo syntax
on rightMouseUp
    window("Help").open()
end

// JavaScript syntax
function rightMouseUp() {
    window("Help").open();
}
```

on runPropertyDialog

使用法

```
-- Lingo syntax
on runPropertyDialog me, currentInitializerList
    statement(s)
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    statement(s);
}
```

解説

パラメータダイアログボックスでビヘイビアーのパラメータに特定の値を定義する Lingo を含む、システムメッセージおよびイベントハンドラです。ビヘイビアーをスプライトに添付したとき、またはスプライトのビヘイビアーの初期プロパティ値を変更したときは、必ず、runPropertyDialog メッセージが送信されます。

ビヘイビアーの初期プロパティの現在の設定値は、プロパティリストとしてハンドラに渡されます。on runPropertyDialog ハンドラがビヘイビアー内で定義されていない場合は、on getPropertyDescriptionList ハンドラから返されたプロパティリストに基づいて、ビヘイビアーのカスタマイズ用のダイアログボックスが起動します。

例文

次のハンドラは、ビヘイビアーのパラメータダイアログボックスで設定されたビヘイビアーの値を上書きします。上書きされた値は、[currentInitializerList] に含まれます。通常、[パラメータ] ダイアログボックスでは、ユーザが質量定数および重力定数を設定することができますが、このハンドラが、代わりにそのパラメータ定数値を割り当てます。

```
-- Lingo syntax
property mass
property gravitationalConstant

on runPropertyDialog me, currentInitializerList
    --force mass to 10
    currentInitializerList.setaProp(#mass, 10)
    -- force gravitationalConstant to 9.8
    currentInitializerList.setaProp(#gravitationalConstant, 9.8)
    return currentInitializerList
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    //force mass to 10
    currentInitializerList.setaProp("mass", 10)
    //force gravitationalConstant to 9.8
    currentInitializerList.setaProp("gravitationalConstant", 9.8)
    return(currentInitializerList)
}
```

関連事項

on getBehaviorDescription, on getPropertyDescriptionList

on savedLocal

使用法

```
-- Lingo syntax
on savedLocal
    statement(s)
end

// JavaScript syntax
function savedLocal() {
    statement(s);
}
```

解説

Shockwave Player の将来のバージョンでの機能拡張のために用意されているシステムメッセージおよびイベントハンドラです。何らかの理由で保存操作が実行されなかった場合は、このメッセージは送信されません。

関連事項

allowSaveLocal

on sendXML

使用法

```
-- Lingo syntax
on sendXML "sendxmlstring", "window", "postdata"
    statement(s)
end

// JavaScript syntax
function sendXML(sendxmlstring, window, postdata) {
    statement(s);
}
```

解説

イベントハンドラ。機能的には、getURL スクリプティングメソッドと似ています。getURL メソッドも、Adobe® Flash® Asset Xtra 拡張機能で利用できます。Flash スプライトまたは Flash XML オブジェクトで **XMLobject**.send ActionScript メソッドが実行されるときに、on sendXML ハンドラが呼び出されます。

ActionScript では、**XMLobject**.send メソッドは、XML オブジェクト内の XML データに加え、2 つのパラメータを渡します。この 2 つのパラメータを次に示します。

- **url** は、XML データの送り先の URL です。通常、これは、XML データの処理を待っているサーバスクリプトの URL になります。
- **window** は、サーバから返されたデータを表示するブラウザウィンドウです。

ActionScript の **XMLobject**.send メソッドは、Director 内で Flash スプライトまたは Lingo 内で作成されたグローバル Flash XML オブジェクトのどちらからも呼び出すことができます。これが発生すると、Lingo の on sendXML ハンドラが呼び出され、同じパラメータがハンドラに渡されます。

次の Lingo は、パラメータが on sendXML ハンドラによってどのように受け取られるかを示します。

```
on sendXML me, theURL, targetWindow, XMLdata
```

これらのパラメータは、**XMLobject**.send のパラメータと次のように対応します。

- theURL は、XML データの送り先の URL です。
- targetWindow は、サーバから返されたデータを表示するブラウザウィンドウです。
- XMLdata は、Flash XML オブジェクト内の XML データです。

  Director ムービー内に on sendXML ハンドラを作成することにより、Flash スプライトまたはグローバル Flash オブジェクトで生成される **XMLobject**.send イベントをムービー内で処理できます。

  Flash スプライトは、外部 XML データをロードしたり、内部 XML データの構文を解析することもできます。Flash Asset Xtra 拡張機能は、ブラウザ内の Flash 5 または Flash MX コンテンツと同様の方法でこれらの機能を処理します。

例文

次の Lingo コマンドは、Flash スプライトから **XMLobject**.send メソッドの情報を取得します。そして、ブラウザ経由でその XML データを URL に送信します。

```
-- Lingo syntax
on sendXML me, theURL, targetWindow, xmlData
    gotoNetPage(theURL, targetWindow)
    postNetText(theURL, xmlData)
end

// JavaScript syntax
function sendXML(theURL, targetWindow, xmlData) {
    gotoNetPage(theURL, targetWindow);
    postNetText(theURL, xmlData);
}
```

on startMovie

使用法

```
-- Lingo syntax
on startMovie
    statement(s)
end

// JavaScript syntax
function startMovie() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。再生ヘッドがムービーの最初のフレームに入る直前に実行されるステートメントを含みます。startMovie イベントは、prepareFrame イベントの後、enterFrame イベントの前に起こります。

on startMovie ハンドラに、ムービーの最初のフレームにあるスプライトを初期化する Lingo を置いておくと便利です。

例文

次のハンドラは、ムービーが開始するとスプライトチャネル 10 ～ 50 が見えなくなるようにします。

```
-- Lingo syntax
on startMovie
    repeat with counter = 10 to 50
        sprite(counter).visible = 0
    end repeat
end startMovie

// JavaScript syntax
function startMovie() {
    for(counter=10;counter<=50;counter++) {
        sprite(counter).visible = 0;
    }
}
```

関連事項

on prepareMovie

on stepFrame

使用法

```
-- Lingo syntax
on stepFrame
    statement(s)
end

// JavaScript syntax
function stepFrame() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。actorList のスクリプトインスタンスで動作します。これらのオブジェクトだけが、on stepFrame メッセージを受け取るからです。このイベントハンドラは、再生ヘッドがフレームに入ったとき、またはステージが更新されたときに実行されます。

on stepFrame ハンドラは、特定のオブジェクトのセットに対してしばしば実行する Lingo を置いておくと便利です。on stepFrame ハンドラ内の Lingo を実行したいときは、これらのオブジェクトを actorList に割り当てます。Lingo を停止したいときは、オブジェクトを actorList から削除します。オブジェクトが actorList にある間は、再生ヘッドがフレームに入るか、updateStage コマンドが発行されるたびに、オブジェクトの on stepFrame ハンドラが実行されます。

stepFrame メッセージは、prepareFrame メッセージの前に送信されます。

オブジェクトを actorList に割り当てると、これらのオブジェクトは stepFrame メッセージに反応します。オブジェクトには、この actorList の組み込み機能を使うための on stepFrame ハンドラが必要です。

go、play、updateStage コマンドは on stepFrame ハンドラ内では使えません。

例文

子オブジェクトが actorList に割り当てられている場合は、再生ヘッドがフレームに入るたびに、親スクリプトの on stepFrame ハンドラが、mySprite プロパティに格納されているスプライトの位置をランダムに更新します。

```
-- Lingo syntax
property mySprite

on new me, theSprite
    mySprite = theSprite
    return me
end

on stepFrame me
    sprite(mySprite).loc = point(random(640),random(480))
end

// JavaScript syntax
// define a constructor class that contains the mySprite property
function Frame(theSprite) {
    this.mySprite = theSprite;
}

function stepFrame() {
    var myFrame = new Frame(sprite(spriteName).spriteNum);
    sprite(myFrame.mySprite).loc = point(random(640),random(480));
end
```

on stopMovie

使用法

```
-- Lingo syntax
on stopMovie
    statement(s)
end

// JavaScript syntax
function stopMovie() {
    statement(s);
}
```

解説

ムービーの再生を停止したときに実行するステートメントを含むシステムメッセージおよびイベントハンドラです。

on stopMovie ハンドラは、ムービーが終了したときにクリーニングタスクを実行する Lingo を置く場所として適しています。クリーニングタスクには、リソースファイルを閉じる、グローバル変数の消去、フィールドの削除、およびオブジェクトの破棄などがあります。

MIAW 用ムービーに設定された on stopMovie ハンドラは、MIAW ムービーが最後まで再生されるか、他のムービーにジャンプするときのみ呼び出されます。このハンドラは、ウィンドウが close window コマンドによって閉じているか、forget window コマンドによって削除された場合は呼び出されません。

例文

次のハンドラは、ムービーが停止したときにグローバル変数をクリアします。

```
-- Lingo syntax
global gCurrentScore

on stopMovie
    gCurrentScore = 0
end

// JavaScript syntax
_global.gCurrentScore;

function stopMovie() {
    _global.gCurrentScore = 0;
}
```

関連事項

on prepareMovie

on streamStatus

使用法

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal, error
    statement(s)
end

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal, error) {
    statement(s);
}
```

解説

インターネットからダウンロードされたオブジェクトの数を特定するために、定期的に呼び出されるシステムメッセージおよびイベントハンドラです。tellStreamStatus (TRUE) が呼び出され、ハンドラがムービースクリプトに追加されている場合のみ、このハンドラが呼び出されます。

on streamStatus イベントハンドラには、以下のパラメータがあります。

URL	取得するデータのインターネットアドレスを表します。
state	ダウンロードするストリームの状態を表します。Connecting、Started、InProgress、Complete、Error のいずれかの値になります。
ストリーム済バイト数	これまでにネットワークから取得したバイト数を表します。
全バイト数	ストリーム内の合計バイト数を表します（確認できる場合）。HTTP サーバがコンテンツの長さの情報を MIME ヘッダに含めない場合は、値が 0 になることがあります。
エラーえらー	ダウンロードが終了していない場合は空のストリング ("") となります。ダウンロードが正常に終了した場合は "**OK**" を、また、ダウンロードが失敗した場合はエラーコードを含みます。

これらのパラメータは、ダウンロードの進行状況に従い、Director によって自動的に指定されます。このハンドラは Director によって自動的に呼び出され、次の呼び出しがいつになるかを制御する方法はありません。特定の操作についての情報が必要な場合は、getStreamStatus() を呼び出します。

Lingo コマンドを使ってネットワークストリームを開始することができます。これには、URL のメディアをリンクするか、URL の外部キャストメンバを使います。streamStatus ハンドラが、すべてのネットワークストリームに関する情報と共に呼び出されます。

streamStatus ハンドラはムービースクリプトに配置します。

例文

次のハンドラは、ストリームされたオブジェクトの状態を判断し、そのオブジェクトの URL を表示します。

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal
    if state = "Complete" then
        put(URL && "download finished")
    end if
end streamStatus

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal) {
    if (state == "Complete") {
        put(URL + " download finished");
    }
}
```

関連事項

getStreamStatus(), tellStreamStatus()

on timeOut

使用法

```
-- Lingo syntax
on timeOut
    statement(s)
end

// JavaScript syntax
function timeOut() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。で指定されている時間内に、マウスやキーボード操作がなかったときに実行されるステートメントを含みます。timeoutLength ムービースクリプトには、常に on timeOut ハンドラを配置します。

ムービー全体を通して同形式のタイムアウト処理を実行したい場合は、the timeoutScript を使用して、タイムアウトのビヘイビアーを一元的に制御します。

例文

次のハンドラは、timeoutLength プロパティで設定されている時間が経過しても、ユーザが何の操作もしなかったときに、「呼び掛けループ」ムービーを再生します。この方法は、ユーザがコンピュータの前から離れているときの対策として使えます。

```
-- Lingo syntax
on timeOut
    _movie.play("Attract Loop")
end timeOut

// JavaScript syntax
function timeOut() {
    _movie.play("Attract Loop");
}
```

trayIconMouseDoubleClick

使用法

```
-- Lingo syntax
on trayIconMouseDoubleClick
    statement(s)
end

// JavaScript syntax
function trayIconMouseDoubleClick() {
    statement(s);
}
```

解説

ムービーとウィンドウのイベントハンドラ（Microsoft Windows 専用）。ユーザがシステムトレイアイコンをダブルクリックしたときに実行されるステートメントを含みます。

trayIconMouseDoubleClick イベントがハンドラに送信されるのは、systemTrayIcon プロパティが TRUE に設定されている場合のみです。

例文

次のハンドラは、ユーザがシステムトレイアイコンをダブルクリックしたときにムービーを一時停止します。

```
-- Lingo syntax
on trayIconMouseDoubleClick
    _movie.delay(500)
end

// JavaScript syntax
function trayIconMouseDoubleClick() {
    _movie.delay(500);
}
```

関連事項

ムービー , systemTrayIcon, trayIconMouseDown, trayIconRightMouseDown, ウィンドウ

trayIconMouseDown

使用法

```
-- Lingo syntax
on trayIconMouseDown
    statement(s)
end

// JavaScript syntax
function trayIconMouseDown() {
    statement(s);
}
```

解説

ムービーとウィンドウのイベントハンドラ（Microsoft Windows 専用）。ユーザがシステムトレイアイコンをシングルクリックしたときに実行されるステートメントを含みます。

trayIconMouseDown イベントがハンドラに送信されるのは、systemTrayIcon プロパティが TRUE に設定されている場合のみです。

例文

次のハンドラは、マウスがシステムトレイアイコン上にある場合にユーザがマウスをクリックしたときにムービーを一時停止します。

```
-- Lingo syntax
on trayIconMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconMouseDown() {
    _movie.delay(500);
}
```

関連事項

ムービー , systemTrayIcon, trayIconMouseDoubleClick, trayIconRightMouseDown, ウィンドウ

trayIconRightMouseDown

使用法

```
-- Lingo syntax
on trayIconRightMouseDown
    statement(s)
end

// JavaScript syntax
function trayIconRightMouseDown() {
    statement(s);
}
```

解説

ムービーとウィンドウのイベントハンドラ（Microsoft Windows 専用）。ユーザがシステムトレイアイコンを右クリックしたときに実行されるステートメントを含みます。

trayIconRightMouseDown イベントがハンドラに送信されるのは、systemTrayIcon プロパティが TRUE に設定されている場合のみです。

例文

次のハンドラは、ユーザがシステムトレイアイコンを右クリックしたときにムービーを一時停止します。

```
-- Lingo syntax
on trayIconRightMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconRightMouseDown() {
    _movie.delay(500);
}
```

関連事項

ムービー , systemTrayIcon, trayIconMouseDoubleClick, trayIconMouseDown, ウィンドウ

on zoomWindow

使用法

```
-- Lingo syntax
on zoomWindow
    statement(s)
end

// JavaScript syntax
function zoomWindow() {
    statement(s);
}
```

解説

システムメッセージおよびイベントハンドラ。ウィンドウ内ムービーとして再生されているムービー（MIAW）のサイズが変更されるたびに実行されるステートメントを含みます。ウィンドウのサイズは、ユーザが最小化または最大化ボタン（Windows）、またはズームボタン（Mac）をクリックすると変更されます。サイズ変更後のウィンドウの大きさは、オペレーティングシステムによって決まります。

on zoomWindow イベントハンドラは、ウィンドウの大きさを変更したときにスプライトを再配置する Lingo を置く場所として適しています。

例文

次のハンドラは、ムービーを再生中のウィンドウのサイズが変更されたときに、スプライト 3 を変数 centerPlace に格納されている座標まで移動します。

```
-- Lingo syntax
on zoomWindow
    centerPlace = point(10, 10)
    sprite(3).loc = centerPlace
end

// JavaScript syntax
function zoomWindow() {
    var centerPlace = point(10, 10);
    sprite(3).loc = centerPlace;
}
```

関連事項

drawRect, sourceRect, on resizeWindow

第 11 章：キーワード

この項では、Adobe® Director® で使用できるすべてのキーワードをアルファベット順に紹介します。

これらのキーワードは Lingo でのみ有効です。JavaScript シンタックスにも下記の Lingo のキーワードと同等の機能を持つキーワードおよび構文がありますが、それらについては、ここでは紹介しません。JavaScript シンタックスのキーワードおよび構文の詳細については、「4 ページの「Director スクリプトに関する重要事項」」を参照してください。

\（継続シンボル）

使用法

```
-- Lingo syntax
first part of a statement on this line \
second part of the statement \
third part of the statement
```

解説

行の末尾に使用すると、ステートメントがその次の行に続くことを表す継続シンボルです。その行は 1 行の連続したステートメントとして解釈されます。

例文

次のステートメントは、\ を使用してステートメントを 2 行にワードラップしています。

```
-- Lingo syntax
if sprite("mySprite").member = member("myMember") then \
_player.alert("The sprite was created from myMember")
```

case

使用法

```
-- Lingo syntax
case expression of
    expression1: Statement
    expression2: Statement(s)
    expression3, expression4: Statement
    {otherwise: Statement(s)}
end case
```

解説

if...then ステートメントを繰り返し記述するよりも、分岐する論理構造を効率よく定義するキーワードです。

Lingo は、まず [ケース式] 内の値をその次の行の式と比較します。比較は、[ケース式] と一致する式を見つけるまで行ごとに順番に行われます。

一致する式が見つかると、その式の後のコロンに続くステートメントが実行されます。ステートメントが 1 つだけの場合には式とステートメントを同じ行に記述し、複数行の場合はその式の次の行からインデントして記述します。

同じステートメントを実行する条件となる式が複数ある場合には、式をコンマで区切って記述します。上記の [式 **3**] と [式 **4**] のシンタックス行の例を参照してください。

Lingo によって最初に一致する式が見つかった時点で、式を見つけるためのテストは中止されます。

ケース構造の最後にオプションの otherwise ステートメントを記述すると、一致する式が存在しないときに otherwise に続くステートメントが実行されます。

例文

次のハンドラは、ユーザが最後に押したキーを調べ、その結果に対応するステートメントを実行します。

- A キーが押された場合は、ムービーはりんごというマーカに移動します。
- B または C キーが押された場合は、ムービーは指定されているトランジションを実行し、オレンジというマーカに移動します。
- その他のキーが押された場合には、警告音が鳴ります。

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

次の case ステートメントは、カーソルがロールオーバーしているスプライト 1、2、3 の番号に対応する Lingo を実行します。

```
case _movie.rollOver() of
    1: sound(1).play(member("Horn"))
    2: sound(1).play(member("Drum"))
    3: sound(1).play(member("Bongos"))
end case
```

char...of

使用法

```
-- Lingo syntax
textMemberExpression.char[whichCharacter]
char whichCharacter of fieldOrStringVariable
textMemberExpression.char[firstCharacter..lastCharacter]
char firstCharacter to lastCharacter of fieldOrStringVariable
```

解説

チャンク式内の文字または範囲内の文字を指定して参照するキーワードです。チャンク式は、ストリングが格納されているテキストソース（フィールドキャストメンバや変数など）内の任意の文字、単語、項目、行です。

- [位置] を指定した式は、単独の文字を参照します。
- 先頭位置と末尾位置を指定した式は、範囲内の文字を参照します。

チャンク内の文字または文字の範囲を指定するには整数を使用します。文字には英字、数字、句読点、スペースなどの他、Tab や Return などの制御文字も含まれます。

char...of キーワードで参照されるストリングは調べられますが、変更はできません。ストリングを変更するには put...into コマンドを使います。

例文

次のステートメントはストリング "$9.00" の最初の文字を表示します。

```
//Lingo
put(("$9.00").char[1..1])
-- "$"

// Javascript
trace(("$9.00").substring(0,1))
-- "$"
```

次のステートメントは、ストリング "$9.00" 全体を表示します。

```
//Lingo
put(("$9.00").char[1..5])
-- "$9.00"

// Javascript
trace(("$9.00").substring(0))
-- "$9.00"
```

次のステートメントは、テキストキャストメンバ " クイズ " の 3 行目の 2 番目にある単語の最初の 5 文字を変更します。

```
//Lingo
member("quiz").line[3].word[2].char[1..5] = "?????"

// Javascript
var s = member(1).getPropRef("line",3).getProp("word",2)
s=s.replace(s.substring(0),"????")
member(1).getPropRef("line",3).setProp("word",2,s)
```

関連事項

mouseMember, mouseItem, mouseLine, mouseWord

end

使用法

```
-- Lingo syntax
end
```

解説

ハンドラや複数行のコントロール構造の終了を示すキーワードです。

例文

次の mouseDown ハンドラは、end mouseDown ステートメントで終了します。

```
on mouseDown
     _player.alert("The mouse was pressed")
end mouseDown
```

end case

使用法

```
-- Lingo syntax
end case
```

解説

case ステートメントを終了するキーワードです。

例文

次のハンドラは、end case キーワードを使用して case ステートメントを終了します。

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

関連事項

case

exit

使用法

```
-- Lingo syntax
exit
```

解説

ハンドラの処理を終了して呼び出された場所に戻るためのキーワードです。ハンドラが別のハンドラから呼び出された場合は、呼び出し元に Lingo の処理が戻されます。

例文

次のステートメントは、最初にモニタが白黒に設定されているかどうかを調べ、設定されている場合は終了します。

```
on setColors
    if _system.colorDepth = 1 then exit
    sprite(1).foreColor = 35
end
```

関連事項

abort, halt(), quit(), pass, return （キーワード）

exit repeat

使用法

```
-- Lingo syntax
exit repeat
```

解説

繰り返しループから抜け出し、end repeat の後のステートメントに処理を進めるためのキーワードです。ただし、現行ハンドラやメソッドからは抜け出しません。

exit repeat は、2 つの値が一致した場合や変数が特定の値になった場合など、指定の条件が満たされたときに、繰り返しループを抜け出すために使用するキーワードです。

例文

次のハンドラは、変数 testString で示されたストリングの最初の母音の位置を探して、見つかった時点で exit repeat コマンドにより繰り返しループを抜け出し、return i ステートメントに移ります。

```
on findVowel testString
    repeat with i = 1 to testString.char[testString.char.count]
        if "aeiou" contains testString.char[i] then exit repeat
    end repeat
    return i
end
```

関連事項

repeat while, repeat with

field

使用法

```
field(whichField)
```

解説

[フィールド] で指定したフィールドキャストメンバを参照するキーワードです。

- [フィールド] がストリングの場合には、キャストメンバ名による参照が行われます。
- [フィールド] が整数の場合には、キャストメンバ番号による参照が行われます。

  フィールドでは、ストリングを割り当てたり、読み出すことができます。また、チャンク式をフィールドに適用することもできます。

  field エレメントは、前のバージョンの Director でも使用されていました。前のバージョンと対応させるため、現在でも継続して使用することができます。新しいムービーでは、member を使用してフィールドキャストメンバを参照します。

例文

次のステートメントは、フィールド " エントリ " 内の 5 ～ 10 番目の文字を変数 myKeyword に設定します。

```
myKeyword = field("entry").char[5..10]
```

このステートメントは、ユーザが「机」という言葉を入力したかどうかを調べ、入力した場合にはマーカ " 机の価格 " に移動します。

```
if member("bid") contains "desk" then _movie.go("deskBid")
```

関連事項

char...of, item...of, line...of, word...of

global

使用法

```
global variable1 {, variable2} {, variable3}...
```

解説

他のハンドラやムービーが共有できるグローバル変数として変数を定義するキーワードです。

グローバル変数の中身を調べたり変更したりするすべてのハンドラは、変数をグローバルとして識別する global キーワードを使用する必要があります。global キーワードを使用しない場合は、他のハンドラでグローバルとして定義されていても、そのハンドラ内では変数をローカル変数として処理します。

注意：ムービー全体でグローバル変数を確実に使用できるようにするために、prepareMovie ハンドラでグローバル変数を定義し、初期化します。これにより、ムービーを離れてまた他のムービーから戻った場合、グローバル変数が設定されていないことを最初に確認しない限り、初期値にリセットされます。

グローバル変数は任意のハンドラやスクリプトで定義できます。その値は、変数をグローバルとして定義した他のハンドラやスクリプトでも使用できます。スクリプトにより変数の値が変更された場合、そのグローバル変数を利用する他のすべてのハンドラで新しい値を使用できます。

グローバル変数は最初に定義されているかどうかにかかわらず、任意のスクリプトやムービーで使用できます。他のフレーム、ムービー、ウィンドウに移動したときに自動的にクリアされることはありません。

メッセージウィンドウで操作する任意の変数は、明示的に定義しなくても、自動的にグローバルになります。

インターネットで再生される Shockwave® コンテンツによるムービーは、同じ HTML ページで再生されていたとしても、他のムービーのグローバル変数を利用できません。ムービーがグローバル変数を共有できるのは、埋め込まれたムービーが他のムービーに移動するために、goToNetMovie または go movie でそれ自体を置き換えた場合に限られます。

例文

次の例では、グローバル変数 startingPoint にまだ値が格納されていない場合、初期値 1 を設定します。これにより、保存されたデータを失うことなくムービー間を移動できます。

```
//Lingo
global gStartingPoint

on prepareMovie
    if voidP(gStartingPoint) then gStartingPoint = 1
end

// Javascript
function prepareMovie(){
    if (_global.gStartingPoint==null) then _global.gStartingPoint = 1
}
```

関連事項

showGlobals(), property, gotoNetMovie

if

使用法

```
if logicalExpression then statement
if logicalExpression then statement
else statement
end if
if logicalExpression then
        statement(s)
end if
if logicalExpression then
        statement(s)
else
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else if logicalExpression2 then
        statement(s)
else if logicalExpression3 then
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else logicalExpression2
end if
```

解説

[論理式] で指定した論理式を評価する if...then 構文のキーワードです。

- 条件が TRUE の場合は、then に続くステートメントが実行されます。
- 条件が FALSE の場合は、else に続くステートメントを実行します。else の後にステートメントがない場合は、if...then 構文から抜け出します。
- 複数条件の場合、条件のすべての部分が評価されます。最初の条件が満たされるかどうかにかかわらず、最初の条件で実行は停止しません。必要のない評価を行わないようにして処理速度を改善するには、if...then ステートメントをすべて最初の 1 行に書くのではなく、独立した行に分けて入れ子にします。

条件にプロパティが指定されている場合、Lingo は自動的にそのプロパティが TRUE かどうかを調べるので、プロパティの後に「= TRUE」というフレーズを明記する必要はありません。

ステートメントの else 部分はオプションです。複数の [**then** ステートメント] または [**else** ステートメント] を使うには、end if で終了する必要があります。

else は常に直前にある if ステートメントと対になっています。場合によって、else キーワードを適切な if キーワードと対応させるために、else nothing ステートメントが必要になることがあります。

注意：スクリプトで if が正しく対になっているかどうか、スクリプトウィンドウで Tab キーを押して確かめることができます。Director はスクリプトを検査し、その内容をインデントして示します。正しく対になっていないものがある場合、表示が変化するのでわかります。

例文

次のステートメントは、Return キーが押されたかどうかを調べ、押されている場合は次に進みます。

```
if the key = RETURN then go the frame + 1
```

次のハンドラは、Command+Q キーが押されたかどうかを調べ、押されている場合はその後に続くステートメントを実行します。

```
on keyDown
    if (_key.commandDown) and (_key.key = "q") then
        cleanUp
        quit
    end if
end keyDown
```

次の 2 つの記述で、処理性能を比較します。最初の書き方では、AND の左側と右側の両方の条件を評価するため、必ず MeasureTimeSinceIStarted 関数で時間を調べる必要があり、多少処理時間を要します。2 つ目の記述では、最初の条件が TRUE の場合にのみ 2 つ目の条件を評価します。

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) and MeasureTimeSinceIStarted() then
    _player.alert("You found the hidden treasure!")
end if
```

処理時間を短縮するには、次のように記述します。

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) then
    if MeasureTimeSinceIStarted() then
        _player.alert("You found the hidden treasure!")
    end if
end if
```

関連事項

case

INF

使用法

```
-- Lingo syntax
INF
```

解説

戻り値。指定した Lingo 式を無限数で評価することを指定したことを表します。

関連事項

NAN

item...of

使用法

```
-- Lingo syntax
textMemberExpression.item[whichItem]
item whichItem of fieldOrStringVariable
textMemberExpression.item[firstItem..lastItem]
item firstItem to lastItem of fieldOrStringVariable
```

解説

チャンク式内の項目、または項目範囲を特定するキーワードです。この場合の項目とは、itemDelimiter プロパティで定義される区切り文字で区切られた一連の文字を指します。

[項目]、[先頭項目]、[末尾項目] は、チャンク内での項目の位置を示す整数値または整数値を導く式で指定する必要があります。

チャンク式には、ストリングのソースとなる文字、単語、項目、行などを指定できます。ストリングのソースには、フィールド、テキストキャストメンバやストリングを含む変数なども含まれます。

末尾項目として指定された値が実際のチャンク式内の項目数よりも大きい場合、式内の最後の項目が末尾項目として扱われます。

例文

次のステートメントは、色の名前で構成されたチャンク式内の 3 番目の項目を調べ、結果をメッセージウィンドウに表示します。

```
put("red,yellow,blue green,orange".item[3])
-- "blue green"
```

結果は "blue green" になります。blue と green の間にはカンマがないためチャンク式内では 1 つの項目として認識されます。

次のステートメントは、チャンク式内の 3 番目～ 5 番目までの項目を調べようとしますが、チャンク式内には 4 つしか項目がないため、5 番目の代わりに 4 番目が末尾項目として扱われたコマンドの実行結果はメッセージウィンドウに表示されます。

```
put("red,yellow,blue green,orange".item[3..5])
-- "blue green, orange"
put item 5 of "red, yellow, blue green, orange"
-- ""
```

次のステートメントは、フィールドキャストメンバ "All Bids" の 2 行目の 4 番目の項目として " 机 " を挿入します。

```
member("All Bids").line[2].item[4] = "Desk"
```

関連事項

char...of, itemDelimiter, number of members, word...of

line...of

使用法

```
-- Lingo syntax
textMemberExpression.line[whichLine]
line whichLine of fieldOrStringVariable
textMemberExpression.line[firstLine..lastLine]
line firstLine to lastLine of fieldOrStringVariable
```

解説

チャンク式内の行または行の範囲を指定するキーワードです。行チャンクとは、改行と改行の間にある一連の文字のことを指します。テキストの自動回り込みによる改行は、改行として処理されません。

[行]、[開始行]、および [終了行] には、チャンク内の 1 つの行を示す整数を指定する必要があります。

チャンク式には、文字のソースとなる文字、単語、項目、行などを指定できます。文字のソースとしては、フィールドキャストメンバやストリングを含む変数なども含まれます。

例文

次のステートメントは、変数 Action の最初の 4 行を " 予定 " フィールドキャストメンバに割り当てます。

```
member("To Do").text = Action.line[1..4]
```

次のステートメントは、変数 Notes に割り当てられているストリングの 3 行目の 2 番目にある単語の後に単語 and を挿入します。

```
put "and" after Notes.line[3].word[2]
```

関連事項

char...of, item...of, word...of, number of members

loop （キーワード）

使用法

```
-- Lingo syntax
_movie.goLoop()
```

解説

マーカを参照するキーワードです。

例文

次のハンドラは、現行フレームと 1 つ前のマーカ間でムービーをループさせます。

```
on exitFrame
    _movie.goLoop()
end exitFrame
```

me

使用法

```
-- Lingo syntax
me
```

解説

現行オブジェクト（親スクリプトまたはビヘイビアーのインスタンス）を参照するため、親スクリプトまたはビヘイビアーで使用される特別な変数です。基本的には、オブジェクトのメモリアドレスを格納する変数です。

これは Lingo に定義されている用語ではありません。me は、慣例的に使用されている用語です。

完成したムービーの me の実例については、Director アプリケーションフォルダの Learning/Lingo_Examples/ フォルダにある parent scripts.dir を参照してください。

例文

次のステートメントは、オブジェクト myBird1 にスクリプト " 鳥 " を設定します。me キーワードは、パラメータスクリプト「鳥」を受け取り、そのパラメータを返すために使用されます。

```
myBird1 = new script("Bird")
```

次のハンドラは、スクリプト " 鳥 " の on new ハンドラです。

```
on new me
    return me
end
```

次の 2 つのハンドラは、親スクリプトです。最初のハンドラは、変数名に me を使って子オブジェクトを参照します。2 番目のハンドラは変数 myAddress を使って子オブジェクトを参照します。それ以外は、親スクリプトは同じです。

最初のハンドラ :

```
property myData

on new me, theData
    myData = theData
    return me
end

on stepFrame me
    ProcessData me
end
```

2 番目のハンドラ :

```
property myData

on new myAddress, theData
    myData = theData
    return myAddress
end

on stepFrame myAddress
    ProcessData myAddress
end
```

関連事項

new(), ancestor

menu

使用法

```
-- Lingo syntax
menu: menuName
itemName | script
itemName | script
...
[more menus]
```

解説

installMenu コマンドと合わせてカスタムメニューの実際の内容を指定するためのキーワードです。メニューの定義はフィールドキャストメンバに格納します。メニューの定義は、そのキャストメンバ名またはキャストメンバ番号で参照します。

menu キーワードのすぐ後に、コロン、スペース、メニュー名を指定します。次の行から、そのメニューに含まれるメニュー項目を定義します。"|" 記号の後にスクリプトを記述することにより、ユーザがメニュー項目を選択したときに実行するスクリプトを設定します。新しいメニューは、次に menu キーワードが使われている部分で定義されます。

注意：メニューは、Shockwave Player では使用できません。

Macintosh では、カスタムメニューを定義するために次の特殊文字を使用することができます。これらの特殊文字は大文字と小文字を区別するため、太字にする場合は、"b" ではなく "**B**" を指定します。

これらの特殊文字は、項目名と "|" 記号の間に挿入します。複数の特殊文字を使用して、メニュー項目を定義することもできます。例えば、<B<U では、項目の文字スタイルがアンダーラインつきの太字になります。

Windows® ではこのフォーマットはサポートされていないため、クロスプラットフォームムービーにはこの特殊文字フォーマットは使用しないでください。

Symbol	例文	解説
@	menu: @	* Macintosh® では、アップルメニューを定義するときに、アップル ® マークを作成し、Macintosh のメニューバーを有効にします。
!Ã	!ÃEasy Select	* Macintosh では、チェックマークのついたメニューを調べます（Option + v）。
<B	Bold<B	* Macintosh では、メニュー項目のスタイルを太字にします。
<I	Italic<I	* Macintosh では、メニュー項目の文字スタイルをイタリック（斜体）にします。
<U	Underline<U	* Macintosh では、メニュー項目の文字スタイルをアンダーライン付き文字にします。
<O	Outline<O	* Macintosh では、メニュー項目の文字スタイルを袋文字にします。
<S	Shadow<S	* Macintosh では、メニュー項目の文字スタイルを影文字にします。
|	Open/O | go to frame "Open"	スクリプトをメニュー項目に関連付けます。
/	Quit/Q	コマンドキーショートカットを定義します。
(	Save(	メニュー項目を選択できないようにします。
(-	(-	メニューの中に選択できない区切り行を作成します。

* Macintosh だけで機能するフォーマットタグを示します。

例文

次の例は、"CustomMenu2" というフィールドキャストメンバのテキストです。これを使用して、カスタムの [ファイル] メニューの内容を指定します。このメニューをインストールするには、ムービーの実行中に "installMenu member("CustomMenu2")" を使用します。メニュー項目の "Convert" はカスタムハンドラ convertThis を実行します。

```
menu: File
Open/O | _movie.go("Open")
Close/W | _movie.go("Close")
Convert/C | convertThis
    (-
Quit/Q | _movie.go("Quit")
```

関連事項

installMenu, name, number （メニュー項目）, checkMark, enabled, script

NAN

使用法

```
-- Lingo syntax
NAN
```

解説

特定の Lingo 式が実数ではない（Not A Number）ことを示す戻り値です。

次のステートメントは、-1 の平方根（実数ではない値となる）をメッセージウィンドウに表示しようとします。

```
-- Lingo syntax
put((-1).sqrt) -- NAN
```

関連事項

INF

next

使用法

```
-- Lingo syntax
next
```

解説

ムービー内の次のマーカを参照するためのキーワードです。このキーワードは the marker (+ 1) というフレーズと同じ役割を果たします。

例文

次のステートメントは、再生ヘッドをムービー内の次のマーカ位置まで移動します。

```
go next
```

次のハンドラは、右矢印キーが押されたたときにムービーをスコア内の次のマーカに移動し、左矢印キーが押されたときにムービーを 1 つ前のマーカに移動します。

```
on keyUp
    if (_key.keyCode = 124) then _movie.goNext()
    if (_key.keyCode = 123) then _movie.goPrevious()
end keyUp
```

関連事項

loop （キーワード）, goPrevious()

next repeat

使用法

```
-- Lingo syntax
next repeat
```

解説

スクリプトの繰り返しループ内で、次のステップに移行するためのキーワードです。ループ処理を抜け出すための exit repeat キーワードとは異なります。

例文

次の繰り返しループは、奇数のみをメッセージウィンドウに表示します。

```
repeat with i = 1 to 10
    if (i mod 2) = 0 then next repeat
    put(i)
end repeat
```

on

使用法

```
-- Lingo syntax
on handlerName {argument1}, {arg2}, {arg3} ...
    statement(s)
end handlerName
```

解説

ハンドラの先頭を示すためのキーワードです。ハンドラは、パラメータを入力値として受け取り、関数の結果として値を返します。

ハンドラは、ビヘイビアー、ムービースクリプト、キャストメンバスクリプトで定義できます。キャストメンバスクリプト内のハンドラは、同じスクリプト内の別のハンドラだけから呼び出せますが、ムービースクリプト内のハンドラはスクリプトの外からも呼び出せます。

ハンドラスクリプトを共有キャストに置くと、複数のムービーで同じハンドラを使うことができます。

otherwise

使用法

```
-- Lingo syntax
otherwise statement(s)
```

解説

これ以前に示された case ステートメントのいずれの条件にも一致しないときに Lingo が実行する命令の前に記述するキーワードです。

このキーワードは範囲外の値や無効な種類を入力したことを通知するために使用でき、オーサリング時のデバッグ作業にも役立ちます。

例文

次のハンドラは、ユーザが最後に押したキーを調べ、その結果に対応するステートメントを実行します。

- ユーザが "A"、"B"、"C" のいずれかのキーを押した場合、ムービーは of キーワードの後の対応するアクションを実行します。
- ユーザがその他のキーを押した場合、ムービーは otherwise キーワードの後のステートメントを実行します。この場合は、単純なアラートを表示します。

```
//Lingo
on keyDown
     case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _player.alert("That is not a valid key.")
    end case
end keyDown

// Javascript
function keyDown()
{
    switch(_key.key)
    {
        case "a":
        _movie.go("Apple");
        break;
        case "b":
        case "c":
        _movie.puppetTransition(99);
        _movie.go("Oranges");
        break;
        default:
        _player.alert("That is not a valid key.");
    }
}
```

property

使用法

```
-- Lingo syntax
property {property1}{, property2} {,property3} {...}
```

解説

[プロパティ **1**]、[プロパティ **2**] などで指定したプロパティがプロパティ変数であることを示すキーワードです。

プロパティ変数は、親スクリプトまたはビヘイビアースクリプトの冒頭で宣言します。親スクリプトまたはビヘイビアースクリプトの外からプロパティ変数にアクセスするときは、the 演算子を使用します。

注意：spriteNum プロパティはすべてのビヘイビアーに対して有効で、定義するだけでアクセスできます。

親スクリプトやビヘイビアースクリプト内のプロパティは me キーワードを使うことなく参照できます。これに対し、親スクリプトのアンセスタのプロパティを参照するには、me.property という形式を使用します。

ビヘイビアーに関しては、1 つのビヘイビアースクリプトで定義したプロパティは、同一のスプライトに割り当てられたその他のビヘイビアーからもアクセスできます。

子オブジェクトのプロパティは、ほかのプロパティを操作するのと同じようなシンタックスを通して、オブジェクトの親スクリプトの外部から直接操作することができます。例えば、次のステートメントでは、子オブジェクト myBouncingObject の motionStyle プロパティを #frenetic に設定します。

```
set the motionStyle of myBouncingObject to #frenetic
```

count 関数を使うと、子オブジェクトの親スクリプト内のプロパティの数を確認することができます。これらのプロパティの名前は、getPropAt を使って得ることができます。オブジェクトにプロパティを追加するには、setaProp() を使います。

完成したムービーでの property の実例については、Director アプリケーションフォルダの Learning/Lingo_Examples/ フォルダにある parent scripts.dir を参照してください。

例文

次のステートメントは、1 つの親スクリプトから作成された子オブジェクトが、それぞれ独自の位置と速度の設定を含むようにプロパティ変数を定義します。

```
property location, velocity
```

次の親スクリプトハンドラは、pMySpriteNum をプロパティとして定義します。

```
-- script Elder
property pMyChannel
on new me, whichSprite
    me.pMyChannel = whichSprite
    return me
end
```

元のビヘイビアースクリプトでは、ancestor プロパティを用意し、spriteNum プロパティをすべてのビヘイビアーに渡します。

```
property spriteNum

property ancestor

on beginSprite me
    ancestor = new script("Elder", spriteNum)
end
```

関連事項

end, ancestor, spriteNum

put...after

使用法

```
-- Lingo syntax
put expression after chunkExpression
```

解説

Lingo の [式] を評価し、値をストリングに変換してから結果をコンテナ内の指定したチャンクの後に挿入するためのコマンドです。[チャンク式] で存在しないチャンクが指定されている場合は、ストリング値がコンテナ内の適切な位置に挿入されます。

[チャンク式] には、文字、単語、項目、行など、文字を含むコンテナであれば何でも指定することができます。コンテナには、フィールドキャストメンバ、テキストキャストメンバ、ストリングを含む変数、特定の文字、単語、項目、行、コンテナ内の範囲などが含まれます。

例文

次のステートメントは、フィールドキャストメンバ " アニマルリスト " の内容の後に、" 狐 犬 猫 " という文字列を追加します。

```
put("fox dog cat") after member("Animal List")
```

次のステートメントでも同じ操作をすることができます。

```
put "fox dog cat" after member("Animal List").line[1]
```

関連事項

char...of, item...of, line...of, paragraph, word...of, put...before, put...into

put...before

使用法

```
-- Lingo syntax
put expression before chunkExpression
```

解説

Lingo の [式] を評価し、値をストリングに変換してから結果をコンテナ内の指定したチャンクの前に挿入するためのコマンドです。[チャンク式] で存在しないチャンクが指定されている場合は、ストリング値がコンテナ内の適切な位置に挿入されます。

[チャンク式] には、文字、単語、項目、行など、文字を含むコンテナであれば何でも指定することができます。コンテナには、フィールドキャストメンバ、テキストキャストメンバ、ストリングを含む変数、特定の文字、単語、項目、行、コンテナ内の範囲などが含まれます。

例文

次のステートメントは、変数 animalList に " 狐 犬 猫 " というストリングを設定し、" 鹿 " という単語をリストの 2 番目の単語の前に挿入します。

```
put "fox dog cat" into animalList
put "elk " before word 2 of animalList
```

結果は、" 狐 鹿 犬 猫 " というストリングになります。

次のシンタックスでも同じ作業を行うことができます。

```
put "fox dog cat" into animalList
put "elk " before animalList.word[2]
```

関連事項

char...of, item...of, line...of, paragraph, word...of, put...after, put...into

put...into

使用法

```
-- Lingo syntax
put expression into chunkExpression
```

解説

Lingo [式] を評価して結果をストリングに変換し、そのストリングをコンテナ内で指定したチャンクと置き換えるためのコマンドです [チャンク式] で存在しないチャンクが指定されている場合は、ストリング値がコンテナ内の適切な位置に挿入されます。

[チャンク式] には、文字、単語、項目、行など、文字を含むコンテナであれば何でも指定することができます。コンテナには、フィールドキャストメンバ、テキストキャストメンバ、ストリングを含む変数、特定の文字、単語、項目、行、コンテナ内の範囲などが含まれます。

ムービーがアプレットとして再生しているとき、put...into コマンドは、テキストのチャンクを置き換えるのではなく、コンテナ内のすべてのテキストを置き換えます。

値を変数に割り当てるには、set コマンド、または、= 演算子を使います。

例文

次のステートメントは、フィールドキャストメンバ " 批評 " の 2 行目を " アグネス・グーチの批評 " に変更します。

```
put "Reviewed by Agnes Gooch" into line 2 of member("Review Comments")
```

次のシンタックスを使ったテキストキャストメンバでも同じ操作を行えます。

```
put "Reviewed by Agnes Gooch" into member("Review Comments").line[2]

// Javascript
member("Review Comments").setProp("line",2,"Reviewed by Agnes Gooch")
repeat while:
```

関連事項

char...of, item...of, line...of, paragraph, word...of, put...before, put...after, set...to, set...=

repeat while

使用法

```
-- Lingo syntax
repeat while testCondition
    statement(s)
end repeat
```

解説

[テスト条件] で指定した条件が TRUE である限り、ステートメントを繰り返し実行するキーワード構文です。例えば、ファイルをすべて読み込むまでストリングの読み込みを続ける、リストの終わりまで項目をチェックさせる、または、ユーザがマウスボタンをクリックするかリリースするまで特定の動作を繰り返し実行させるなどの Lingo にこの構文を使います。

繰り返しループ内にいる間は、Lingo は他のイベントを無視します。繰り返しループ内で現在のキーを確認するには、keyPressed プロパティを使います。

一度に実行できるのは 1 つのハンドラだけです。Lingo が繰り返しループ内に長時間留まると、その他のイベントが評価されるために待機することになります。したがって、繰り返しループは、短くて時間のかからない操作、またはユーザが他の操作をしないときに使うことをお勧めします。

数秒以上の処理が必要な場合は、カウンタを使用してループ内の関数を評価するか、進行状況をチェックします。

終了条件に達しない、または繰り返しループから抜け出す条件が定義されていない場合は、Windows では Ctrl+Alt+ ピリオド（"."）、Macintosh では Command+ ピリオド（"."）を押すことにより、Director を停止できます。

例文

次のハンドラは、タイマを作動させ、0 にリセットしてから 60 ミリ秒までカウントさせます。

```
on countTime
    _system.milliseconds
    repeat while _system.milliseconds < 60
    -- waiting for time
    end repeat
end countTime

// Javascript
function countTime()
{
    _system.milliseconds
    while (_system.milliseconds < 60)
    {
        -- waiting for time
    }
}
```

関連事項

exit, exit repeat, repeat with, keyPressed()

repeat with

使用法

```
-- Lingo syntax
repeat with counter = start to finish
    statement(s)
end repeat
```

解説

[ステートメント] で指定した Lingo を、[カウンタ] で指定した回数のみ実行するキーワード構文です。[カウンタ] の値は、[開始値] と [終了値] との差になります。カウンタの値は、Lingo がループを繰り返すごとに 1 ずつ増えていきます。

repeat with 構文は、一連のスプライトに同じ効果を適用したり、一連の数値に対してべき乗を求めるなど、処理や計算を繰り返したいときに使うと便利です。

繰り返しループ内にいる間は、Lingo は他のイベントを無視します。繰り返しループ内で現在のキーを確認するには、keyPressed プロパティを使います。

一度に実行できるのは 1 つのハンドラだけです。Lingo が繰り返しループ内に長時間留まると、その他のイベントが評価されるために待機することになります。したがって、繰り返しループは、短くて時間のかからない操作、またはユーザが他の操作をしないときに使うことをお勧めします。

数秒以上の処理が必要な場合は、カウンタを使用してループ内の関数を評価するか、進行状況をチェックします。

終了条件に達しない、または繰り返しループから抜け出す条件が定義されていない場合は、Windows では Ctrl+Alt+ ピリオド（"."）、Macintosh では Command+ ピリオド（"."）を押すことにより、Director を停止できます。

例文

次のハンドラは、スプライト 1 ～ 30 をパペットスプライトにします。

```
//Lingo
on puppetize
    repeat with channel = 1 to 30
    _movie.puppetSprite(channel, TRUE)
    end repeat
end puppetize

// Javascript
function puppetize()
{
    for(var channel=1;channel<30;channel++)
    {
    _movie.puppetSprite(channel, true);
    }
}
```

関連事項

exit, exit repeat, repeat while, repeat with...down to, repeat with...in list

repeat with...down to

使用法

```
-- Lingo syntax
repeat with variable = startValue down to endValue
```

解説

[開始値] から [終了値] までを 1 つずつカウントダウンしていくキーワードです。

一度に実行できるのは 1 つのハンドラだけです。Lingo が繰り返しループ内に長時間留まると、その他のイベントが評価されるために待機することになります。したがって、繰り返しループは、短くて時間のかからない操作、またはユーザが他の操作をしないときに使うことをお勧めします。

繰り返しループ内にいる間は、Lingo は他のイベントを無視します。繰り返しループ内で現在のキーを確認するには、keyPressed プロパティを使います。

数秒以上の処理が必要な場合は、カウンタを使用してループ内の関数を評価するか、進行状況をチェックします。

終了条件に達しない、または繰り返しループから抜け出す条件が定義されていない場合は、Windows では Ctrl+Alt+ ピリオド（"."）、Macintosh では Command+ ピリオド（"."）を押すことにより、Director を停止できます。

例文

次のハンドラは、20 ～ 15 までカウントダウンする繰り返しループを含んでいます。

```
on countDown
    repeat with i = 20 down to 15
        sprite(6).member = 10 + i
        _movie.updateStage()
    end repeat
end
```

repeat with...in list

使用法

```
-- Lingo syntax
repeat with variable in someList
```

解説

指定したリストからの一連の値を変数に割り当てるためのキーワードです。

繰り返しループ内にいる間は、Lingo は keypress 以外の他のイベントを無視します。繰り返しループ内で現在のキーを確認するには、keyPressed プロパティを使います。

一度に実行できるのは 1 つのハンドラだけです。Lingo が繰り返しループ内に長時間留まると、その他のイベントが評価されるために待機することになります。したがって、繰り返しループは、短くて時間のかからない操作、またはユーザが他の操作をしないときに使うことをお勧めします。

数秒以上の処理が必要な場合は、カウンタを使用してループ内の関数を評価するか、進行状況をチェックします。

終了条件に達しない、または繰り返しループから抜け出す条件が定義されていない場合は、Windows では Ctrl+Alt+ ピリオド（"."）、Macintosh では Command+ ピリオド（"."）を押すことにより、Director を停止できます。

例文

次のステートメントは、4 個の値をメッセージウィンドウに表示します。

```
//Lingo
repeat with i in [1, 2, 3, 4]
    put(i)
end repeat

// Javascript
var nl = new Array()
nl =[1,2,3,4]
var i
for(i in nl)
{
    trace(nl[i])
}
```

return （キーワード）

使用法

```
-- Lingo syntax
return expression
```

解説

キーワード。[式] の値を返し、ハンドラを終了します。[式] パラメータは、Lingo 値である必要があります。

ユーザ定義関数として機能するハンドラで値を返すものを呼び出す場合は、パラメータのリストを括弧で囲む必要があります。result 関数で紹介した diceRoll 関数ハンドラのように、パラメータを受け取らない場合でも括弧は必要です。

return キーワードは exit コマンドと同じであると考えてください。ただし、return がハンドラの呼び出しステートメントに値を返す場合を除きます。ハンドラ内の return コマンドは、直ちにハンドラから抜け出しますが、同時にコマンドを呼び出した Lingo に値を返すことができます。

オブジェクト指向のスクリプトにおける return の使い方は、最初は理解するのが難しいこともあります。自分で関数を作成し、ハンドラから抜け出すために return を使うことから始めるとよいでしょう。慣れてくると、on new ハンドラ内の return me 行が、作成されたオブジェクトの参照を呼び出しステートメントに返し、変数にオブジェクトを割り当てられることが、より明確に理解できます。

return キーワードは、改行である RETURN キャラクタ定数とは異なります。Director は文脈によって用法を区別します。

返された値を得るには、呼び出しステートメント内のハンドラ名の後に括弧を使って、指定したハンドラが関数であることを示します。

完成したムービーでの return (keyword) の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "parent scripts.dir" を参照してください。

例文

次のハンドラは、5 ～ 100 の間のランダムな 5 の倍数を返します。

```
//Lingo
on getRandomScore
    theScore = 5 * random(20)
    return theScore
end getRandomScore

// Javascript
function getRandomScore()
{
    theScore = 5 * random(20);
    return theScore;
}
```

このハンドラは、次のようなステートメントを使って呼び出します。

```
thisScore = getRandomScore()
```

この例では、変数 thisScore に getRandomScore() 関数から返された値が割り当てられています。これはオブジェクト参照を返す親スクリプトと同じ値の受け渡し方法です。呼び出しコード内の変数名がそのオブジェクトへの参照となっています。

関連事項

result, RETURN (定数)

set...to, set...=

使用法

```
-- Lingo syntax
lingoProperty = expression
variable = expression
```

解説

式で指定した式を評価し、結果を [**Lingo** プロパティ] で指定したプロパティまたは [変数] で指定した変数に割り当てるためのコマンドです。

例文

次のステートメントはメンバ 3 を " 夕焼け " という名前に設定します。

```
member(3).name = "Sunset"
```

次のステートメントは、soundEnabled プロパティに、現在の状態とは逆の状態を設定します。soundEnabled が TRUE の場合 (つまり、サウンドがオンになっている) は、これをオフにし、soundEnabled が FALSE (つまりサウンドがオフになっている) は、これをオンにします。

```
_sound.soundEnabled = not(_sound.soundEnabled)
```

次のステートメントは変数 vowels をストリング "aeiou" に設定します。

```
vowels = "aeiou"
```

関連事項

property

sprite...intersects

使用法

```
-- Lingo syntax
sprite(sprite1).intersects(sprite2)
sprite sprite1 intersects sprite2
```

解説

2 つのスプライトの位置を比較するための演算子です。[スプライト **1**] の境界矩形が [スプライト **2**] の境界矩形に接しているときは TRUE、いないときは FALSE になります。

両方のスプライトに matte インクが設定されている場合は、境界矩形ではなく、実際の輪郭が使用されます。スプライトの輪郭とは、白以外のピクセルで実際にその形を表している境界のことを意味します。

この演算子の評価優先度は 5 です。

注意：[スプライト 1] が単純な式ではない、つまり計算処理を含む式の場合は、常にドット演算子を使います。

例文

次のステートメントは、2 つのスプライト（i と j）が交差しているかどうかをチェックし、交差している場合は「お知らせ」フィールドキャストメンバの内容を「正しく配置されました。」に変更します。:

```
// Lingo Syntax
if sprite i intersects j then put("You placed it correctly.") into member("Notice")

// Javascript
if (sprite(i).intersects(sprite(j)))
{
    member("Notice").text="You placed it correctly";
}
```

関連事項

sprite...within, quad

sprite...within

使用法

```
-- Lingo syntax
sprite(sprite1).within(sprite2)
sprite sprite1 within sprite2
```

解説

2 つのスプライトの位置を比較するための演算子です。[スプライト **1**] の矩形が、[スプライト **2**] の矩形内に完全に収まっている場合は TRUE になり、そうでない場合は FALSE になります。

両方のスプライトに matte インクが設定されている場合は、境界矩形ではなく、実際の輪郭が使用されます。スプライトの輪郭とは、白以外のピクセルで実際にその形を表している境界のことを意味します。

この演算子の評価優先度は 5 です。

注意：[スプライト 1] が単純な式ではない、つまり計算処理を含む式の場合は、常にドット演算子を使います。

例文

次のステートメントは、スプライト 3 がスプライト 2 内に完全に収まっているかどうかをチェックし、収まっている場合は doInside ハンドラを呼び出します。

```
//Lingo
if sprite(3).within(2) then doInside

// Javascript
if (sprite(3).within(2))
{
    doInside();
}
```

関連事項

sprite...intersects, quad

version

使用法

```
-- Lingo syntax
_player.productVersion
```

解説

キーワード。Director のバージョンストリングを含むシステム変数です。同じストリングが、Macintosh の Finder の情報ウィンドウに表示されます。

例文

次のステートメントは、Director のバージョンをメッセージウィンドウに表示します。

```
put(_player.productVersion)
```

word...of

使用法

```
-- Lingo syntax
member(whichCastMember).word[whichWord]
textMemberExpression.word[whichWord]
chunkExpression.word[whichWord]
word whichWord of fieldOrStringVariable
fieldOrStringVariable. word[whichWord]
textMemberExpression.word[firstWord..lastWord]
member(whichCastMember).word[firstWord..lastWord]
word firstWord to lastWord of chunkExpression
chunkExpression.word[whichWord..lastWord]
```

解説

チャンク式内の単語、または単語の範囲を特定するためのチャンク式です。単語チャンクとは、半角スペースで区切られた文字シーケンスのことを意味します (Tab や改行などの非表示文字は、スペースとして処理されます)。

[単語]、[先頭の単語]、[末尾の単語] の各式は、チャンク内の単語を示す整数値として評価される必要があります。

チャンク式には、文字のソースとなる文字、単語、項目、行などを指定できます。文字のソースには、フィールドおよびテキストキャストメンバや、ストリングを含む変数があります。

完成したムービーの word...of の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "text.dir" ムービーを参照してください。

例文

次のステートメントは、変数 animalList を " 狐 犬 猫 " というストリングに設定し、そのリストの 2 番目に " 鹿 " という単語を挿入します。

```
animalList = "fox dog cat"
put "elk" before animalList.word[2]
```

結果は、" 狐 鹿 犬 猫 " というストリングになります。

次のステートメントは、上記と同じリスト内の 5 番目の単語をメッセージウィンドウに表示します。

```
put "fox elk dog cat".word[5]
```

このストリングには 5 番目の単語が存在しないので、空白のストリングを意味する 2 つの二重引用符（""）がメッセージウィンドウに表示されます。

関連事項

char...of, line...of, item...of, count(), number （単語）

第 12 章：メソッド

このセクションでは、Director® で使用できるすべてのメソッドをアルファベット順に説明します。

_system.gc()

使用法

```
_system.gc();
```

解説

JavaScript ガベージコレクションメソッドを呼び出します。_system.gc() は、JavaScript のみでサポートされています。このメソッドを使用すると、ガベージコレクションが発生する時点を制御できます。また、Lingo コードのみを記述し、グローバル変数を利用する場合にも、このメソッドを使用することができます。メモリを管理する 2 つの異なる方法があります。JavaScript ランタイムエンジンはガベージコレクションの対象ですが、Lingo ランタイムエンジンは対象ではありません。JavaScript により参照されるメモリの未使用ブロックは、JavaScript ガベージコレクションが発生するまで、実際にはシステムには返されません。_system.gc() ルーチンを実行することにより、JavaScript ガベージコレクションを実装することができます。JavaScript エンジンは実行時にガベージコレクションルーチンを定期的に実行します。ただし、Lingo コードのみが実行中である場合、_system.gc() 呼び出しは行われません。純粋な Lingo コード中であっても、変数がグローバルとして宣言された場合は、変数を参照する必要がある将来の JavaScript コード用に、JavaScript 参照オブジェクトが作成されます。Lingo が変数をクリアした場合、ガベージコレクションルーチンにより、JavaScript 参照オブジェクトおよび変数データを削除することができます。

JavaScript が実行中ではない場合は、_system.gc() は呼び出されません。また、システムに解放されるのを待機しているオブジェクトに対して、メモリ全体を使い果たす可能性があります。ムービースクリプトで簡単な JavaScript ルーチンを記述し、それを Lingo コードから呼び出します。

```
function clearJSMemory()
{
  _system.gc()
}
```

abort

使用法

```
--Lingo syntax
abort

// JavaScript syntax
abort();
```

解説

現在のハンドラおよびそのハンドラを呼び出したすべてのハンドラに残っているステートメントを実行せずに、抜け出すためのコマンドです。このコマンドは、現在のハンドラを呼び出したハンドラに戻る exit キーワードとは異なります。

abort コマンドでは Director は終了しません。

パラメータ
なし

例文
次のステートメントでは、未使用メモリが 50K 未満になると、現在のハンドラおよびそのハンドラを呼び出したハンドラから抜け出します。

```
-- Lingo syntax
if the freeBytes < 50*1024 then abort

// JavaScript syntax
if (_player.freeBytes < 50*1024) {
    abort()
}
```

関連事項
exit, halt(), quit()

abs()

使用法

```
--Lingo syntax
abs (numericExpression)

// JavaScript syntax
Math.abs (numericExpression)
```

解説
abs() 関数には様々な用途がありますが、例えば、マウスやスプライトの動きを追うときに、座標値の差 (正または負数) から距離 (常に正数) を換算するために使うことができます。また、abs() 関数は sqrt() や log() などの演算関数の処理にも使うことができます。

JavaScript シンタックスでは、Math オブジェクトの abs() 関数を使用します。

パラメータ
numericExpression 必須。絶対値を計算する整数または浮動小数点数。**numericExpression** が整数の場合は、その絶対値も整数になります。**numericExpression** が浮動小数点数の場合は、その絶対値も浮動小数点数になります。

例文
次のステートメントは、現在のマウスの位置と変数 startV の差の絶対値が 30 より大きいかどうかを調べます（距離として負の数値は使用しないため）。大きい場合にはスプライト 6 のフォアカラーを変更します。

```
-- Lingo syntax
if (the mouseV - startV).abs > 30 then sprite(6).forecolor = 95

// JavaScript syntax
if ((_mouse.mouseV - Math.abs(_mouse.startV)) > 30) {
    sprite(6).foreColor = 95;
}
```

activateAtLoc()

使用法

```
-- Lingo syntax
dvdObjRef.activateAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.activateAtLoc(point(x, y));
```

解説

DVD メソッド。指定されたステージ位置の下にある埋め込み DVD メニュー項目の hilite をアクティブにします。

このメソッドは、成功すると 0 を返します。

パラメータ

point**(x, y)** 必須。埋め込み DVD メニュー項目の位置を指定するステージ座標で示された点。

例文

次のステートメントは、指定されたステージ位置にあるメニュー項目の hilite をアクティブにします。

```
-- Lingo syntax
member("movie1").activateAtLoc(point(100, 200))

// JavaScript syntax
member("movie1").activateAtLoc(point(100, 200));
```

関連事項

DVD

activateButton()

使用法

```
-- Lingo syntax
dvdObjRef.activateButton()

// JavaScript syntax
dvdObjRef.activateButton();
```

解説

DVD メソッド。現在選択されているメニューボタンをアクティブにします。

このメソッドは、成功すると 0 を返します。

注意：このメソッドは、Mac®-Intel® ではサポートされていません。

パラメータ

なし

例文

次のステートメントは、指定されたキャストメンバのメニューボタンをアクティブにします。

```
-- Lingo syntax
sprite(1).member.activateButton()

// JavaScript syntax
sprite(1).member.activateButton();
```

関連事項

DVD

add

使用法

```
-- Lingo syntax
linearList.add(value)

// JavaScript syntax
array.push(value);
```

解説

リストコマンド。リニアリストについてのみ、値をリニアリストに追加します。ソート済みリストの場合は適切な場所に挿入されます。未ソートのリストの場合、値はリストの最後に追加され、

このコマンドはリニアリストのみに適用され、プロパティリストに使用するとエラーが発生します。

注意：add コマンドを、加算に使う + 演算子やストリングの連結に使う & 演算子と混同しないようにしてください。

パラメータ

value 必須。リニアリストに追加する値。

例文

次のステートメントは、値 2 をリスト bids に追加します。

```
-- Lingo syntax
bids = [3, 4, 1]
bids.add(2)

// JavaScript syntax
bids = new Array(3,4,1);
bids.push(2);
```

次のステートメントは、値 2 をソート済みのリニアリスト [3, 4, 1] に追加します。結果は [3, 4, 1, 2] になります。

```
-- Lingo syntax
-- sort the list using Lingo
bids.sort()
bids.add(2)

// JavaScript syntax
// sort the list using JavaScript
bids.sort();
bids.push(2);
```

bids は値でソートされ、結果は [1, 2, 3, 4] になります。

関連事項

sort, addAt, append

add (3D texture)

使用法

```
--Lingo syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()

// JavaScript syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()
```

解説

3D meshdeform モディファイアコマンド。モデルのメッシュに空のテクスチャレイヤーを追加します。

以下のコードを使用して、レイヤー間のテクスチャ座標をコピーできます。

```
modelReference.meshdeform.texturelayer[a].texturecoordinatelist =
modelReference.meshdeform.texturelayer[b].texturecoordinatelist
```

パラメータ

なし

例文

次のステートメントは、" 耳 " というモデルの最初のメッシュに新しいテクスチャレイヤーを作成します。

```
--Lingo syntax
member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.add()

// JavaScript syntax
member("Scene").getprop("model","Ear").meshdeform.mesh[1].textureLayer.add();
```

関連事項

meshDeform （モディファイア）, textureLayer, textureCoordinateList

addAt

使用法

```
list.AddAt(position, value)
```

解説

リストコマンド。リニアリストについてのみ、値をリスト内の指定された位置に追加します。

このコマンドはリニアリストのみに適用され、プロパティリストに使用するとエラーが発生します。

パラメータ

position 必須。**value** で指定されている値を追加するリスト内の位置を指定する整数。

value 必須。リストに追加する値。

例文

次のステートメントは、値 8 をリスト bids の 4 番目の位置に挿入します。元のリストは [3, 2, 4, 5, 6, 7] です。

```
--Lingo
bids = [3, 2, 4, 5, 6, 7]
bids.addAt(4,8)

// Javascript
bids = list(3, 2, 4, 5, 6, 7)
bids.addAt(4,8)
```

挿入の結果、bids の値は [3, 2, 4, 8, 5, 6 ,7] になります。

addBackdrop

使用法

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite, rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation);
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite, rotation);
```

解説

3D カメラコマンド。カメラの背景リストの最後に背景を追加します。

パラメータ

texture 必須。背景に適用するテクスチャ。

locWithinSprite 必須。3D スプライト内の背景が表示される 2D の位置。この位置は、スプライトの左上隅から測定されます。

rotation 必須。テクスチャを回転する角度の値を指定する整数。

例文

次のステートメントの 1 行目は、" ラフ " というテクスチャを、" 杉 " というキャストメンバから作成します。作成したテクスチャは、変数 t1 に格納されます。2 行目は、スプライト "5" 内の (220, 220) の位置に、テクスチャを背景として適用します。テクスチャの回転は 0° です。最後の行は、同じテクスチャを、" シーン " というキャストメンバのカメラ 1 の背景として適用します。位置は (20, 20) で、回転は 45° です。

```
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0)
member("Scene").camera[1].addBackdrop(t1, point(20, 20), 45)

// Javascript
var t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0) ;
member("Scene").getPropRef("camera",1).addBackdrop(t1, point(20, 20), 45) ;
```

関連事項

removeBackdrop

addCamera

使用法

```
-- Lingo syntax
sprite(whichSprite).addCamera(whichCamera, index)
-- JavaScript syntax
sprite(whichSprite).addCamera(whichCamera, index);
```

解説

3D コマンド。スプライトのカメラリストにカメラを追加します。各カメラのビューは、インデックスの数が低いカメラのビューの上に表示されます。各カメラの rect プロパティを設定して、スプライト内に複数のビューを表示できます。

パラメータ

whichCamera 必須。スプライトのカメラリストに追加するカメラへの参照。

index 必須。**whichCamera** を追加するカメラリスト内のインデックスを指定する整数。インデックスが cameraCount() の値より大きい場合、カメラはリストの末尾に追加されます。

例文

次のステートメントは、"FlightCam" というカメラを、スプライト "12" のカメラリストの 5 番目のインデックス位置に挿入します。

```
--Lingo syntax
sprite(12).addCamera(member("scene").camera("FlightCam"), 5)

// JavaScript syntax
sprite(12).addCamera(member("scene").getPropRef("camera", i), 5);
// where i is the number index for the camera "FlightCam".
```

関連事項

cameraCount(), deleteCamera

addChild

使用法

```
-- Lingo syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node(whichChildNode)
{,#preserveWorld})

// JavaScript syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node(whichChildNode)
{,symbol(preserveWorld)})
```

解説

3D コマンド。ノードを別のノードの子のリストに追加し、以前の親の子のリストから削除します。

このメソッドと同等の処理は、子ノードの parent プロパティに親ノードを設定することです。

パラメータ

addMemberRef 必須。追加するノードを含むキャストメンバへの参照。

addNodeRef 必須。追加するノードへの参照。このノードは、モデル、グループ、カメラ、またはライトのいずれかです。

symPreserveParentOrWorld オプション。スプライトのカメラリストに追加するカメラへの参照。有効な値は、#preserveWorld または #preserveParent です。#preserveParent を指定して子を追加すると、親基準の子のトランスフォームは変わりません。子は、新しい親のスペースのトランスフォームにジャンプします。子のワールドトランスフォームは再計算されます。#preserveWorld を指定して子を追加すると、子のワールドトランスフォームは変わらず、子は新しい親のスペースのトランスフォームにもジャンプしません。親基準のトランスフォームは再計算されます。

例文

次のステートメントは、" タイヤ " というモデルを " 車 " というモデルの子リストに追加します。

```
-- Lingo syntax
member("3D").model("Car").addChild(member("3D").model("Tire"))

// JavaScript syntax
member("3D").getProp("model" , i ).addChild(member("3D").getProp("model" , j));
// where i is the number index for model "Car" and j is the number index for model "Tire".
```

次のステートメントは、Bird というモデルを MyCamera というカメラの子リストに追加します。その際、#preserveWorld 引数を使用して、Bird のワールド位置を保持します。

```
-- Lingo syntax
member("3D").camera("MyCamera").addChild(member("3D").model
("Bird"), #preserveWorld)

// JavaScript syntax
member("3D").getPropRef("camera",j).addChild(member("3D").getProp("model",i),symbol("preserveWorld"))
// where i the number index of the model "Bird" and j is the number index of the camera "MyCamera"
```

関連事項

parent, addToWorld, removeFromWorld

addModifier

使用法

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addModifier(#modifierType)

// JavaScript syntax
member(whichCastmember).model(whichModel).addModifier(symbol(modifierType));
```

解説

3D モデルコマンド。指定したモディファイアをモデルに追加します。このコマンドにデフォルト値はありません。

パラメータ

symbolModType 必須。追加するモディファイアを指定するシンボル。指定できるモディファイアは次のとおりです。

- #bonesPlayer
- #collision
- #inker
- #keyframePlayer
- #lod (レベルオブディテール)
- #meshDeform

- #sds
- #toon

各モディファイアの詳細については、それぞれのモディファイアのエントリを参照してください。

例文

次のステートメントは、" ボックス " というモデルにトゥーンモディファイアを追加します。

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , a ).addModifier(symbol("toon"));
// where a is the number index for the "Box" model.
```

関連事項

bonesPlayer （モディファイア）, collision (modifier), inker （モディファイア）, keyframePlayer （モディファイア）, lod （モディファイア）, meshDeform （モディファイア）, sds （モディファイア）, toon （モディファイア）, getRendererServices(), removeModifier, modifier, modifier[], modifiers

addOverlay

使用法

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)
```

解説

3D カメラコマンド。カメラのオーバーレイリストの最後にオーバーレイを追加します。

パラメータ

texture 必須。オーバーレイに適用するテクスチャ。

locWithinSprite 必須。3D スプライト内のオーバーレイが表示される 2D の位置。この位置は、スプライトの左上隅から測定されます。

rotation 必須。テクスチャを回転する角度の値を指定する整数。

例文

次のステートメントの 1 行目は、" ラフ " というテクスチャを、"Cedar" というキャストメンバから作成します。作成したテクスチャは、変数 t1 に格納されます。2 行目は、スプライト "5" の (220, 220) の位置に、テクスチャをオーバーレイとして適用します。テクスチャの回転は 0° です。ステートメントの最後の行は、同じテクスチャを、「シーン」というキャストメンバのカメラ「20」のオーバーレイとして適用します。位置は (20, 20) です。テクスチャの回転は 45° です。

```
-- Lingo syntax
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addOverlay(t1, point(220, 220), 0)
member("Scene").camera[1].addOverlay(t1, point(20, 20), 45)

// JavaScript syntax
t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"));
sprite(5).camera.addOverlay(t1, point(220, 220), 0);
member("Scene").getPropRef("camera",1).addOverlay(t1, point(20, 20), 45);
```

関連事項

removeOverlay, overlay

addProp

使用法

```
list.addProp(property, value)
addProp list, property, value
```

解説

プロパティリストコマンド。プロパティリストについてのみ、指定されたプロパティとその値をプロパティリストに追加します。

未ソートのリストの場合、値はリストの最後に追加され、ソート済みリストの場合は適切な場所に挿入されます。

リスト内にそのプロパティが既に存在する場合は、Lingo および JavaScript のどちらのシンタックスでも、プロパティの複製が作成されます。プロパティの重複を防ぐには、setaProp() コマンドを使用して、新しい項目のプロパティを変更します。

このコマンドはプロパティリストのみに適用され、リニアリストに使用するとエラーが発生します。

パラメータ

property 必須。リストに追加するプロパティ。

value 必須。リストに追加するプロパティの値。

例文

次のステートメントは、kayne という名前のプロパティと、そのプロパティに割り当てられている値 3 を、プロパティリスト bids に追加します。このプロパティリストの内容は [#gee: 4, #ohasi:1] になります。このリストはソート済みのため、新しい項目は、アルファベット順の適切な位置に挿入されます。

```
--Lingo
bids.addProp(#kayne, 3)

// Javascript
bids.addProp("kayne",3)
```

結果のリストは [#gee: 4, #kayne: 3, #ohasi: 1] になります。

次のステートメントは、項目 kayne: 7 をリスト bids に追加します。このリストの内容は [#gee: 4, #kayne: 3,#ohasi: 1] です。リストには既に kayne プロパティが存在するので、プロパティの複製が作成されます。

```
--Lingo
bids.addProp(#kayne, 7)

// Javascript
bids.addProp("kayne",7)
```

結果のリストは [#gee: 4, #kayne: 3, #kayne: 7, #ohasi: 1] になります。

addToWorld

使用法

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
member(whichCastmember).light(whichLight).addToWorld()

// JavaScript syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
memberwhichCastmember).light(whichLight).addToWorld()
```

解説

3D コマンド。キャストメンバの 3D ワールドに、モデル、グループ、カメラ、またはライトを、" ワールド " というグループの子として挿入します。

モデル、グループ、カメラ、またはライトを作成または複製すると、自動的にワールドに追加されます。モデル、グループ、カメラ、またはライトを、削除せずに 3D ワールドから取り除くには、removeFromWorld コマンドを使用します。モデル、グループ、カメラ、またはライトが、ワールドに対して追加または削除されたかどうかを確認するには、isInWorld() コマンドを使用します。

パラメータ

なし

例文

次のステートメントは、"gbCyl" というモデルを、"Scene" というキャストメンバの 3D ワールドに追加します。

```
-- Lingo syntax
member("Scene").model("gbCyl").addToWorld()

// JavaScript syntax
member("Scene").getProp("model","gbCyl").addToWorld();
```

関連事項

isInWorld(), removeFromWorld

addVertex()

使用法

```
-- Lingo syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV, \ vertControlLocV ], [
horizControlLocH, vertControlLocV ]})

// JavaScript syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV, vertControlLocV ], [
horizControlLocH, vertControlLocV ]});
```

解説

指定したベクトルシェイプキャストメンバに新しい頂点を追加するベクトルシェイプコマンドです。

水平位置、垂直位置は、頂点シェイプキャストメンバの原点を基準とした相対位置です。

末尾に 2 つのパラメータを追加すると、頂点のコントロールハンドルの場所を指定できます。コントロールハンドルの場所は頂点の位置を基準とした相対位置であり、指定を省略すると (0, 0) という水平および垂直オフセットを指定したと見なされます。

パラメータ

indexToAddAt 必須。メンバを追加するインデックスを指定する整数。

pointToAddVertex 必須。メンバを追加する位置を指定するポイント。

horizControlLocH オプション。水平コントロールハンドルの水平部分の位置を指定する整数。

horizControlLocV オプション。水平コントロールハンドルの垂直部分の位置を指定する整数。

vertControlLocH オプション。垂直コントロールハンドルの水平部分の位置を指定する整数。

vertControlLocV オプション。垂直コントロールハンドルの垂直部分の位置を指定する整数。

例文

次のステートメントは、ベクトルシェイプ "Archie" に頂点が 2 つ定義されているときに、その間に水平位置 25、垂直位置 15 の点を追加します。

```
-- Lingo syntax
member("Archie").addVertex(2, point(25, 15))

// JavaScript syntax
member("Archie").addVertex(2, point(25, 15));
```

関連事項

vertexList, moveVertex(), deleteVertex(), originMode

alert()

使用法

```
-- Lingo syntax
_player.alert(displayString)

// JavaScript syntax
_player.alert(displayString);
```

解説

プレーヤメソッド。システムの警告音を鳴らし、指定された文字列を含むアラートダイアログボックスを表示します。

アラートメッセージはストリングです。アラート内に変数値を含めるには、alert() に渡す前に、変数をストリングに変換します。

パラメータ

displayString 必須。アラートダイアログボックスに表示するテキストを表すストリング。ストリングは 255 文字以下です。

例文

次のステートメントは、CD-ROM ドライブが接続されていないことを通知するためのアラートメッセージを表示します。

```
-- Lingo syntax
_player.alert("There is no CD-ROM drive connected.")

// JavaScript syntax
_player.alert("There is no CD-ROM drive connected.");
```

次のステートメントはファイルが見つからなかったことを通知するアラートメッセージを表示します。

```
-- Lingo syntax
_player.alert("The file" && QUOTE & filename & QUOTE && "was not found.")

// JavaScript syntax
_player.alert("The file \"" + filename + "\" was not found.");
```

関連事項

プレーヤ

Alert()

使用法

```
Alert( MUIObject,alertPropertiesList)
```

解説

このコマンドは、MUI Xtra のインスタンスから作成されたアラートダイアログボックスを表示します。alert コマンドによって生成される簡単なアラート以外に、この機能があります。

MUI Xtra はモーダルアラートを提供します。アラートは移動可能または移動不可にすることができます。アラートを作成するには、MUI Xtra オブジェクトを作成し、アラートのプロパティを含むリストを 2 番目のパラメータとして指定した alert コマンドを発行します。

指定する必要があるプロパティと指定できる値は以下のとおりです。

プロパティ	使用できる値	指定する内容
#buttons	#Ok#OkCancel#AbortRetryIgnore#YesNoCancel#YesNo#RetryCancel	アラートに表示されるボタンのセット。ボタンは各シンボルで指定された順番に表示されます。
#default	デフォルトとなるボタンの順番を示す値。例えば、アラートのボタンが「OK」と「キャンセル」である場合、2 は「キャンセル」を指定します。デフォルトなしの場合は 0 を指定します。	デフォルトのボタン。
#icon	#stop #note #caution #question #error	アラートに表示されるアイコンのタイプ。アイコンなしの場合は 0 を指定します。
#message	ストリング	アラートに表示されるメッセージ。
#movable	TRUE FALSE	アラートが移動可能かどうか。
#title	ストリング	アラートのタイトル

アラートの各プロパティを明示的に指定する必要があります。MUI Xtra ではデフォルトのアラートプロパティのリストは提供されません。Lingo ではユーザがクリックしたボタンの値が返されます。

アラートはほぼ画面と同じ大きさで表示できるので、必要に応じて長い説明を表示できます。

次のステートメントは、アラートダイアログボックスを作成して表示します。

- 最初のステートメントは、ダイアログボックスとして使用されるオブジェクトである MUI Xtra のインスタンスを作成します。
- 2 番目のステートメントは、アラートのプロパティのリストを設定します。
- 最後のステートメントは Alert コマンドを使用してアラートを表示し、ユーザがどのボタンをクリックしたかをレポートします。

例文

```
-- Lingo syntax
set alertObj = new(xtra "MUI")
set alertInitList = [ #buttons : #YesNo, ¬
#title : "Alert Test", #message : "This shows Yes and No buttons",\
#movable : TRUE]
if objectP ( alertObj ) then
        set result = Alert( alertObj, alertInitList )
        case result of
        1 : -- the user clicked yes
        2 : -- the user clicked no
        otherwise : -- something is seriously wrong
    end case
end if
```

append

使用法

```
list.append(value)
append list, value
```

解説

指定した [値] をリニアリストの最後に追加するコマンドです。このコマンドは、ソート済みの [リスト] に、リスト内の順序に従って値を追加する add コマンドとは異なります。

このコマンドはリニアリストのみに適用され、プロパティリストに使用するとスクリプトエラーが発生します。

プロパティ

value 必須。リニアリストの最後に追加する値。

例文

次のステートメントは 2 という値を bids という名前のソート済みリスト [1, 3, 4] の最後に追加します。値はソート順にかかわらずリストの最後に追加されます。

```
--Lingo
set bids = [1, 3, 4]
bids.append(2)

// Javascript
bids = list(1, 3, 4)
bids.append(2)
```

結果のリストは [1, 3, 4, 2] になります。

関連事項

add (3D texture), sort

applyFilter()

使用法

```
-- Lingo syntax
imageObjRef.applyFilter(filterObjRef {, rectObj})

// JavaScript syntax
imageObjRef.applyFilter(filterObjRef {, rectObj});
```

解説

画像メソッド。指定したフィルタを画像オブジェクトに適用します。画像の一部にフィルタを適用するには、rectObj パラメータ（オプション）を使用します。

applyFilter() は、画像オブジェクトを直接変更します。ただし、このメソッドは画像のサイズを変更しません。rectObj により指定される境界の外のフィルタ効果は無視されます。

重要：applyFilter() を使用して適用されたフィルタ効果を元に戻すことはできません。

パラメータ

パラメータ	解説	必須 / オプション
filterObjRef	指定した画像オブジェクトに適用するフィルタ。filter() メソッドを使用して filterObjRef を作成します。	必須
rectObj	フィルタが適用される対象である画像内の矩形領域。矩形の座標は、画像の原点が基準です。対象の矩形が指定されていない場合、フィルタは画像全体に適用されます。	オプション

例文

次の例では、ビットマップキャストメンバである Earth の一部に blurFilter を適用します。

```
-- Lingo
member("Earth").image.applyFilter(filter(#blurfilter), rect(50, 50, 100, 100))

// Javascript
member("Earth").image.applyFilter(filter(#blurfilter), rect(50, 50, 100, 100));
```

関連事項

rect （画像）, image （画像）

appMinimize()

使用法

```
-- Lingo syntax
_player.appMinimize()

// JavaScript syntax
_player.appMinimize();
```

解説

プレーヤメソッド。Microsoft Windows では、プロジェクタを Windows タスクバーに最小化します。Mac では、プロジェクタを非表示にします。

Mac では、非表示のプロジェクタを Dock から再び開きます。

このメソッドは、タイトルバーなしで再生できるプロジェクタや MIAW で使用するのに適しています。

パラメータ

なし

例文

```
--Lingo syntax
on mouseUp me
    _player.appMinimize()
end


// JavaScript syntax
function mouseUp() {
     _player.appMinimize();
}
```

関連事項

プレーヤ

atan()

使用法

```
-- Lingo syntax
(number).atan
atan (number)

// JavaScript syntax
Math.atan(number);
```

解説

数学関数 (Lingo のみ)。指定されている数値のアークタンジェント (逆正接) を計算します。結果は PI/2 と +PI/2 の間のラジアンになります。

JavaScript シンタックスでは、Math オブジェクトの atan() 関数を使用します。

パラメータ

なし

例文

次のステートメントは 1 のアークタンジェントを返します。

```
(1).atan
```

結果は小数点以下が 4 桁の 0.7854 (約 PI/4) になります。

三角関数はほとんどがラジアンで計算するので、必要に応じて度からラジアンに単位を変換します。

次のハンドラは度からラジアンに単位を変換します。

```
-- Lingo syntax
on DegreesToRads degreeValue
    return degreeValue * PI/180
end

// JavaScript syntax
function DegreesToRads(degreeValue) {
    return degreeValue * PI/180
}
```

次のステートメントは 30° をラジアンに変換してメッセージウィンドウに表示します。

```
put DegreesToRads(30)
-- 0.5236
```

関連事項

cos(), PI, sin()

beep()

使用法

```
-- Lingo syntax
_sound.beep({intBeepCount})

// JavaScript syntax
_sound.beep({intBeepCount});
```

解説

サウンドメソッド。**intBeepCount** で指定された回数だけ警告音を鳴らします。**intBeepCount** が指定されていない場合には警告音が 1 回だけ鳴ります。

- Windows の警告音は、サウンドプロパティダイアログボックスで指定したサウンドです。
- Macintosh では、コントロールパネルの「サウンド」で設定した警告音が鳴ります。サウンドコントロールパネルで音量が 0 に設定されているときには、音が鳴る代わりにメニューバーが点滅します。

パラメータ

intBeepCount オプション。警告音を鳴らす回数を指定する整数。

例文

```
-- Lingo syntax
on mouseUp me
    _sound.beep(1)
end mouseUp
// JavaScript syntax
function mouseUp() {
    _sound.beep(1);
}
```

関連事項

サウンド

beginRecording()

使用法

```
-- Lingo syntax
_movie.beginRecording()

// JavaScript syntax
_movie.beginRecording();
```

解説

ムービーメソッド。スコアの生成セッションを開始します。

beginRecording() を呼び出すと、再生ヘッドが自動的に 1 フレームだけ進み、そのフレームで録画を開始します。このように動作させずに、beginRecording() を呼び出したフレームで録画を開始するには、_movie.go(_movie.frame - 1) のようなステートメントを、beginRecording() と endRecording() の間に置きます。

1 つのムービー内で同時に複数の更新セッションをアクティブにすることはできません。

beginRecording() のすべての呼び出しには、スコア生成セッションを終了する endRecording() の呼び出しが対応している必要があります。

パラメータ

なし

例文

次のハンドラは、beginRecording キーワードを使ってキャストメンバ "Ball" をスプライトチャネル 20 に割り当て、一連のフレーム上で上下左右にアニメートする Score 生成セッションを開始します。フレーム数はパラメータ numberOfFrames で定義されています。

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

関連事項

endRecording(), ムービー , updateFrame()

bitAnd()

使用法

```
bitAnd(integer1, integer2)
```

解説

関数（Lingo のみ）。指定された 2 つの整数を 32 ビットのバイナリ値に変換し、2 つの 2 進数の対応する桁の数値が両方とも 1 の場合はその桁には 1 を返し、それ以外の場合はそれぞれの桁に 0 を返します。この結果としてできた 2 進数は、Lingo によって 10 を基数とする整数として表示されます。

整数	2 進数 (簡略化して示します)
6	00110

整数	2 進数 (簡略化して示します)
7	00111
結果	
6	00110

JavaScript シンタックスでは、ビットごとの演算子 "&" を使用します。

パラメータ

integer1 必須。1 番目の整数。

integer2 必須。2 番目の整数。

例文

次のステートメントは、2 進数の 6 と 7 とを比較した後、その結果を整数で返します。

```
--Lingo
put bitAnd(6, 7)
-- 6



// Javascript
trace ( 6 & 7)
// 6
```

関連事項

bitNot(), bitOr(), bitXor()

bitNot()

使用法

```
(integer).bitNot
bitNot(integer)
```

解説

関数（Lingo のみ）。指定した整数を 32 ビットの 2 進数に変換し、1 を 0 と入れ替え、0 を 1 と入れ替えることにより、2 進数の数を反転する関数です。反転後の 2 進数が基数 10 の整数として表示されます。

整数	2 進数
1	00000000000000000000000000000001
結果	
-2	11111111111111111111111111111110

JavaScript シンタックスでは、ビットごとの演算子 "~" を使用します。

パラメータ

なし

例文

次のステートメントは、整数 1 の 2 進数表記を反転し、反転後の数字を返します。

```
--Lingo
put (1).bitNot
-- -2

// Javascript
trace(~1)
// -2
```

関連事項

bitAnd(), bitOr(), bitXor()

bitOr()

使用法

```
bitOr(integer1, integer2)
```

解説

関数（Lingo のみ）。指定された 2 つの整数を 32 ビットのバイナリ値に変換し、2 つの 2 進数の対応する桁の数値のどちらか一方でも 1 の場合はその桁には 1 を返し、それ以外の場合はそれぞれの桁に 0 を返します。この結果としてできた 2 進数は、Lingo によって 10 を基数とする整数として表示されます。

整数	**2 進数 (簡略化して示します)**
5	0101
6	0110
結果	
7	0111

JavaScript シンタックスでは、ビットごとの演算子 "|" を使用します。

パラメータ

integer1 必須。1 番目の整数。

integer2 必須。2 番目の整数。

例文

次のステートメントでは、5 と 6 という 2 つの整数を 32 ビット 2 進数で比較し、その比較結果を整数で返します。

```
-- Lingo
put bitOr(5, 6)
-- 7

// Javascript
trace(5|6)
// 7
```

関連事項

bitNot(), bitAnd(), bitXor()

bitXor()

使用法

```
bitXor(integer1, integer2)
```

解説

関数。指定の 2 つの整数を 32 ビットの 2 進数に変換し、2 進数を返します。変換された 2 つの 2 進数の対応する桁同士が違う数であればその桁は 1 に、同じ数であればその桁は 0 にして返します。この結果としてできた 2 進数は、Lingo によって 10 を基数とする整数として表示されます。

整数	2 進数 (簡略化して示します)
5	0101
6	0110
結果	
3	0011

JavaScript シンタックスでは、ビットごとの演算子 "^" を使用します。

パラメータ

integer1 必須。1 番目の整数。

integer2 必須。2 番目の整数。

例文

次のステートメントでは、5 と 6 という 2 つの整数を 32 ビット 2 進数で比較し、その比較結果を整数で返します。

```
-- Lingo
put bitXor(5, 6)
-- 3

// Javascript
trace(5^6)
// 3
```

関連事項

bitNot(), bitOr(), bitAnd()

breakLoop()

使用法

```
-- Lingo syntax
soundChannelObjRef.breakLoop()

// JavaScript syntax
soundChannelObjRef.breakLoop();
```

解説

サウンドチャネルメソッド。現在 **soundChannelObjRef** でループ再生中のサウンドのループを停止し、endTime まで再生します。

現時点でループがない場合、このメソッドは無効になります。

パラメータ
なし

例文
次のハンドラを実行すると、サウンドチャネル 2 でループ中のバックグランドミュージックがループを停止し、終わりまで再生されます。

```
-- Lingo syntax
on continueBackgroundMusic
    sound(2).breakLoop()
end

// JavaScript syntax
function continueBackgroundMusic() {
    sound(2).breakLoop();
}
```

関連事項
endTime（サウンドチャネル）, サウンドチャネル

breakLoop（サウンドオブジェクト）

使用法

```
soundObject.breakLoop()
```

解説
サウンドオブジェクトメソッド。サウンドオブジェクトのループを停止して、endTime まで再生します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.breakloop() -- Stops the looping of the currently looping sound object.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.breakLoop(); // Stops the looping of the currently looping sound object.
}
```

関連事項
endTime（サウンドオブジェクト）

browserName()

使用法

```
browserName pathName
browserName()
browserName(#enabled, trueOrFalse)
```

解説

ブラウザのパスまたは場所を指定するシステムプロパティ、コマンド、関数です。FileIO Xtra を使うと、ユーザがブラウザを検索するときに使うダイアログボックスを表示することができます。[開く] ダイアログボックスを表示するには、FileIO Xtra の displayOpen() を使います。

browserName() を使うと、現在指定されているブラウザの名前が返されます。browserName (fullPathToApplication) のように、FileIO Xtra で見つけたパス名をパラメータに指定して、このプロパティ値を設定できます。browserName(#enabled, **True** または **False**) を実行すると、goToNetPage コマンドが実行されたときに指定ブラウザが自動的に起動するようになります。

このコマンドはプロジェクタや Director で再生する場合にのみ有効です。ブラウザで再生している場合は使えません。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、Firefox® ブラウザの場所を示します。

```
browserName "My Disk:My Folder:FireFox"
```

メッセージウィンドウにブラウザ名を表示します。

```
put browserName()
```

ByteArray

使用法

```
ByteArray([intSize],[intInitialValue])
```

解説

バイト配列メソッド。サイズ intSize のバイト配列を作成します。このメソッドは intInitialValue で初期化されます。

パラメータ

パラメータ	解説	デフォルト値
intSize	バイト配列のサイズ。	0
intInitialValue	バイト配列はこの値で初期化されます。つまり、すべてのバイトにこの値が格納されます。	0

例文

```
--Lingo syntax
bArray=byteArray(10,1)

//JavaScript syntax
bArray=byteArray(10,1);
```

ByteArray(str)

使用法

```
ByteArray(str)
```

解説
バイト配列メソッド。指定したストリングからバイト配列を作成します。

パラメータ

パラメータ	解説	デフォルト値
str	バイト配列の内容。	Void

例文
```
--Lingo syntax
bArray=byteArray("Director")

//JavaScript syntax
bArray=byteArray("Director");
```

build()

使用法
```
-- Lingo syntax
member(whichCastmember).modelResource(whichModelResource).build()

// JavaScript syntax
member(whichCastmember).modelResource(whichModelResource).build();
```

解説
3D メッシュコマンド。メッシュを構築します。このコマンドは、#mesh というタイプのモデルリソースでのみ使用されます。

メッシュを最初に構築するとき、メッシュの任意の face プロパティを変更した後、および generateNormals() コマンドを使用した後は、build() コマンドを使う必要があります。

パラメータ
なし

例文
次の例は、まず #mesh というタイプの単純なモデルリソースを作成し、そのプロパティを指定します。次に、モデルリソースを使用して新しいモデルを作成します。プロセスの概略は次のようになります。

1 行目は、" プレーン " というメッシュを作成します。1 つの面と、3 つの頂点、および最大で 3 つのカラーがあります。法線の数とテクスチャ座標の数は設定されていません。法線は generateNormals コマンドで作成されます。

2 行目は、" プレーン " の頂点として使用されるベクトルを定義します。

3 行目は、" プレーン " の最初の面の頂点にベクトルを割り当てます。

4 行目は、設定可能な 3 つの色を newMesh コマンドで定義します。

5 行目は、" プレーン " の最初の面にカラーを割り当てます。カラーリストの 3 番目のカラーが、" プレーン " の最初の頂点に適用され、2 番目のカラーが 2 番目の頂点に割り当てられます。1 番目のカラーは 3 番目の頂点に割り当てられます。カラーは、" プレーン " の最初の面全体にグラデーションで広がります。

6 行目は、generateNormals() コマンドで " プレーン " の法線を作成します。

7 行目は、build() コマンドを呼び出してメッシュを構築します。

```
-- Lingo syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)

// JavaScript syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0);
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)];
nm.face[1].vertices = [1,2,3];
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)];
nm.face[1].colors = [3,2,1];
nm.generateNormals(#smooth);
nm.build();
nm = member("Shapes").newModel("TriModel", nm);
```

関連事項

generateNormals(), newMesh, face[ ]

cacheDocVerify()

使用法

```
-- Lingo syntax
cacheDocVerify #setting
cacheDocVerify()

// JavaScript syntax
cacheDocVerify symbol(setting);
cacheDocVerify();
```

解説

インターネット上のページコンテンツが、プロジェクタのキャッシュデータによってどれだけ頻繁にリフレッシュされるのかを設定する関数です。

cacheDocVerify() を使うと、キャッシュの現行設定を返します。

cacheDocVerify 関数は、ムービーが Director またはプロジェクタで再生されているときにのみ有効です。この関数は Adobe® Shockwave® ムービーには使用できません。Shockwave ムービーの場合はブラウザのネットワーク設定に依存するためです。

```
-- Lingo syntax
on resetCache
    current = cacheDocVerify()
    if current = #once then
        alert "Turning cache verification on"
        cacheDocVerify #always
    end if
end

// JavaScript syntax
function resetCache() {
    current = cacheDocVerify();
    if (current == symbol("once")) {
        alert("Turning cache verification on");
        cacheDocVerify(symbol("always"))
    }
}
```

パラメータ

cacheSetting オプション。インターネット上のページのコンテンツをリフレッシュする頻度を指定するシンボル。指定できる値は、#once (デフォルト) または #always です。#once を指定すると、インターネットから取り込んでキャッシュに保存したムービーファイルを次回からそのまま使い、インターネット上で更新チェックしません。#always を指定すると、URL を指定してムービーを呼び出すたびに、更新チェックするようになります。

関連事項

cacheSize(), clearCache

cacheSize()

使用法

```
-- Lingo syntax
cacheSize Size
cacheSize()

// JavaScript syntax
cacheSize(Size);
cacheSize();
```

解説

Director のキャッシュサイズを設定する関数とコマンドです。

cacheSize 関数は、ムービーが Director またはプロジェクタで再生されているときにのみ有効です。この関数は Shockwave ムービーには使用できません。Shockwave ムービーの場合はブラウザのネットワーク設定に依存するためです。

パラメータ

newCacheSize オプション。キロバイト単位でキャッシュサイズを指定する整数。

例文

次のハンドラは、ブラウザのキャッシュ設定が 1 MB 以下になっているかどうかを調べます。キャッシュ設定が 1 MB 以下になっている場合、ハンドラはアラートを表示し、キャッシュサイズを 1 MB に設定します。

```
-- Lingo syntax
on checkCache if
    cacheSize()<1000 then
        alert "increasing cache to 1MB"
        cacheSize 1000
    end if
end

// JavaScript syntax
function checkCache() {
    if (cacheSize() < 1000) {
        alert("increasing cache to 1MB");
        cacheSize(1000);
    }
}
```

関連事項

cacheDocVerify(), clearCache

call

使用法

```
call #handlerName, script, {args...}
call (#handlerName, scriptInstance, {args...})
```

解説

指定されたスクリプトまたはスクリプトリストのハンドラを呼び出すメッセージを送信するコマンドです。

call コマンドは変数をハンドラの名前として使用できます。call を使って渡されたメッセージは、スプライトに割り当てられた他のスクリプト、キャストメンバスクリプト、フレームスクリプト、ムービースクリプトには渡されません。

パラメータ

symHandlerName 必須。アクティブ化するハンドラを指定するシンボル。

scriptInstance 必須。ハンドラを含むスクリプトまたはスクリプトリストへの参照。**scriptInstance** が単一のスクリプトインスタンスの場合、ハンドラがスクリプトのアンセスタスクリプトで定義されていないと、エラーのアラートが発生します。**scriptInstance** がスクリプトインスタンスのリストの場合は、リストの各項目にメッセージが送られます。この場合は、ハンドラがアンセスタスクリプトで定義されていなくても、アラートは発生しません。

args オプション。ハンドラに渡す任意のオプションパラメータ。

例文

次のハンドラは、スプライト 1 に割り当てられている最初のビヘイビアースクリプトにメッセージ bumpCounter を渡します。

```
-- Lingo syntax
on mouseDown me
    -- get the reference to the first behavior of sprite 1
    set xref = getAt (the scriptInstanceList of sprite 1,1)
    -- run the bumpCounter handler in the referenced script,
    -- with a parameter
    call (#bumpCounter, xref, 2)
end

// JavaScript syntax
function mouseDown() {
    // get the reference to the first behavior of sprite 1
    xref = getAt(sprite(1).script(1));
    // run the bumpCounter handler in the referenced script
    call(symbol("bumpcounter"), xref, 2);
}
```

次の例では、call ステートメントがビヘイビアーや親スクリプトまたはそのアンセスタからどのようにハンドラを呼び出すのかを紹介します。

- 親スクリプト

```
-- Lingo syntax
-- script Man
property ancestor

on new me
    set ancestor = new(script "Animal", 2)
    return me
end
on run me, newTool
    put "Man running with "&the legCount of me&" legs"
end
```

- アンセスタスクリプト

```
-- script Animal
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- 次のステートメントは親スクリプトとアンセスタスクリプトを使用します。

  このステートメントは、親スクリプトのインスタンスを作成します。

```
set m = new(script "man")
```

  次のステートメントは人を歩かせます。

```
call #walk, m
-- "Animal walking with 2 legs"
```

  次のステートメントは人間を走らせます。

```
set msg = #run
call msg, m
-- "Man running with 2 legs and rock"
```

次のステートメントは、親スクリプトの 2 つ目のインスタンスを作成します。

```
set m2 = new(script "man")
```

次のステートメントは、親スクリプトの 2 つのインスタンス m と m2 にメッセージを送ります。

```
call msg, [m, m2]
-- "Man running with 2 legs "
-- "Man running with 2 legs "
```

callAncestor

使用法

```
callAncestor handlerName, script, {args...}
```

解説

子オブジェクトのアンセスタスクリプトにメッセージを送るコマンドです。

アンセスタがさらにそのアンセスタを含むこともできます。

callAncestor では、変数をハンドラの名前として使用できます。また、もとのスクリプトのハンドラをバイパスしてアンセスタスクリプトのハンドラに直接アクセスすることができます。

パラメータ

symHandlerName 必須。アクティブ化するハンドラを指定するシンボル。

scriptInstance 必須。ハンドラを含むスクリプトまたはスクリプトリストへの参照。**scriptInstance** が単一のスクリプトインスタンスの場合、ハンドラがスクリプトのアンセスタスクリプトで定義されていないと、エラーのアラートが発生します。**scriptInstance** がスクリプトインスタンスのリストの場合は、リストの各項目にメッセージが送られます。この場合は、ハンドラがアンセスタスクリプトで定義されていなくても、アラートは発生しません。

args オプション。ハンドラに渡す任意のオプションパラメータ。

例文

次の例では、callAncestor ステートメントを使ってアンセスタスクリプトのハンドラを直接呼び出す方法を紹介します。

- 親スクリプト

```
-- script "man"
property ancestor

on new me, newTool
    set ancestor = new(script "Animal", 2)
    return me
end
on run me
    put "Man running with "&the legCount of me&"legs"
end
```

- アンセスタスクリプト

```
-- script "animal"
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- 次のステートメントは親スクリプトとアンセスタスクリプトを使用します。

  このステートメントは、親スクリプトのインスタンスを作成します。

  ```
  set m = new(script "man")
  ```

  次のステートメントは人を歩かせます。

  ```
  call #walk, m
  -- "Animal walking with 2 legs"
  ```

  次のステートメントは人間を走らせます。

  ```
  set msg = #run
  callAncestor msg, m
  -- "Animal running with 2 legs"
  ```

  次のステートメントは、親スクリプトの 2 つ目のインスタンスを作成します。

  ```
  set m2 = new(script "man")
  ```

  次のステートメントは両方の人間のアンセスタスクリプトにメッセージを送ります。

  ```
  callAncestor #run,[m,m2]
  -- "Animal running with 2 legs"
  -- "Animal running with 2 legs"
  ```

関連事項

ancestor, new()

callFrame()

使用法

```
-- Lingo syntax
spriteObjRef.callFrame(flashFrameNameOrNum)

// JavaScript syntax
spriteObjRef.callFrame(flashFrameNameOrNum);
```

解説

Flash® ムービースプライトのフレーム内にある一連のアクションを呼び出すコマンドです。

このコマンドは、Flash® ActionScript エンジンにメッセージを送り、Flash ムービー内でアクションの実行を開始します。

パラメータ

flashFrameNameOrNum 必須。呼び出すフレームの名前または番号を指定するストリングまたは値。

例文

次の Lingo は、スプライト "1" の Flash ムービーのフレーム 10 に設定されたアクションを実行します。

```
-- Lingo syntax
sprite(1).callFrame(10)

// JavaScript syntax
sprite(1).callFrame(10);
```

camera()

使用法

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

解説

3D エレメント。3D ワールドを表示するベクトル位置にあるオブジェクトです。

各スプライトにはカメラのリストがあります。リストの各カメラのビューは、インデックス の数が低いカメラのビューの上に表示されます。各カメラの rect (camera) プロパティを設定して、スプライト内に複数のビューを表示できます。

カメラは、キャストメンバのカメラパレットに保存されます。3D キャストメンバでカメラを作成および削除するには、newCamera および deleteCamera コマンドを使用します。

スプライトの camera プロパティは、スプライトのカメラリストにある最初のカメラです。sprite(スプライト).camera で参照されるカメラは、sprite(スプライト).camera(1) と同じです。3D スプライトのカメラリストを作成するには、addCamera および deleteCamera コマンドを使用します。

例文

次のステートメントは、スプライト "1" のカメラを、"Picnic" というキャストメンバの "TreeCam" というカメラに設定します。

```
sprite(1).camera = member("Picnic").camera("TreeCam")
```

次のステートメントは、スプライト "1" のカメラを、" ピクニック " というキャストメンバの 2 つ目のカメラに設定します。

```
sprite(1).camera = member("Picnic").camera[2]
```

関連事項

bevelDepth, overlay, modelUnderLoc, spriteSpaceToWorldSpace, fog, clearAtRender

cameraCount()

使用法

```
-- Lingo syntax
sprite(whichSprite).cameraCount()

// JavaScript syntax
sprite(whichSprite).cameraCount();
```

解説
3D コマンド。スプライトのカメラリストにある項目数を返します。

パラメータ
なし

例文
次のステートメントは、スプライト "5" に 3 つのカメラがあることを示します。

```
-- Lingo syntax
put sprite(5).cameraCount()
-- 3

// JavaScript syntax
put(sprite(5).cameraCount());
// 3
```

関連事項
addCamera, deleteCamera

cancelIdleLoad()

使用法

```
-- Lingo syntax
_movie.cancelIdleLoad(intLoadTag)

// JavaScript syntax
_movie.cancelIdleLoad(intLoadTag);
```

解説
ロードタグで指定されたすべてのキャストメンバのロードをキャンセルするムービーメソッドです。

パラメータ
intLoadTag 必須。コンピュータがアイドル状態のときにロードのためのキューに登録されているキャストメンバのグループを指定する整数。

例文
次のステートメントは、ロードタグ "20" が付けられたすべてのキャストメンバのロードをキャンセルします。

```
-- Lingo syntax
_movie.cancelIdleLoad(20)

// JavaScript syntax
_movie.cancelIdleLoad(20);
```

関連事項
idleLoadTag, ムービー

castLib()

使用法

```
-- Lingo syntax
castLib(castNameOrNum)

// JavaScript syntax
castLib(castNameOrNum);
```

解説

指定されたキャストライブラリへの参照を返すトップレベル関数です。

デフォルトのキャストライブラリの番号は 1 です。キャスト番号が 1 以外のキャストライブラリのキャストメンバを指定するには、別のキャストライブラリを指定するように castLib() を設定します。

パラメータ

castNameOrNum 必須。キャストライブラリ名を指定するストリング。または、キャストライブラリ番号を指定する整数。

例文

次のステートメントでは、変数 parts を 2 番目のキャストライブラリに設定しています。

```
-- Lingo syntax
parts = castLib(2)

// JavaScript syntax
var parts = castLib(2);
```

関連事項

キャストライブラリ , castLibNum

channel() （トップレベル）

使用法

```
-- Lingo syntax
channel(soundChannelNameOrNum)

// JavaScript syntax
channel(soundChannelNameOrNum);
```

解説

Sound Channel オブジェクトへの参照を返すトップレベル関数です。

パラメータ

soundChannelNameOrNum 必須。サウンドチャネルの名前を指定するストリング。または、サウンドチャネルのインデックス位置を指定する整数。

Lingo または JavaScript で、_movie.channel[1] を使用して、チャネル 1 にアクセスできます。

例文

次のステートメントは、変数 newChannel をサウンドチャネル 9 に設定します。

```
-- Lingo syntax
newChannel = channel(9)

// JavaScript syntax
var newChannel = channel(9);
```

関連事項

サウンドチャネル

channel() （サウンド）

使用法

```
-- Lingo syntax
_sound.channel(intChannelNum)

// JavaScript syntax
_sound.channel(intChannelNum);
```

解説

指定されたサウンドチャネルへの参照を返すサウンドメソッドです。

このメソッドの機能は、トップレベルの sound() メソッドと同じです。

パラメータ

intChannelNum 必須。参照するサウンドチャネルを指定する整数。

例文

次のステートメントは、変数 myChannel をサウンドチャネル 2 に設定します。

```
-- Lingo syntax
myChannel = _sound.channel(2)

// JavaScript syntax
var myChannel = _sound.channel(2);
```

関連事項

サウンド , sound(), サウンドチャネル

chapterCount()

使用法

```
-- Lingo syntax
dvdObjRef.chapterCount({intTitle})

// JavaScript syntax
dvdObjRef.chapterCount({intTitle});
```

解説
タイトルで使用できるチャプタの数を示す DVD メソッドです。

パラメータ
intTitle オプション。カウントするチャプタを含むタイトルを指定する整数。省略すると、chapterCount() は現在のタイトルで使用できるチャプタの数を返します。

例文
次のステートメントは、現行タイトルのチャプタ数を返します。

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

関連事項
chapterCount, DVD

charPosToLoc()

使用法

```
--Lingo syntax
memberObjRef.charPosToLoc(nthCharacter)

// JavaScript syntax
memberObjRef.charPosToLoc(nthCharacter);
```

解説
フィールドキャストメンバ全体 (ステージに表示されていない部分も含む) の中で指定された文字に最も近いポイントを返すフィールド関数です。個々の文字の位置を確認するときに使用できます。

charPosToLoc の値はフィールドキャストメンバの左上隅からのピクセル数で表されます。[**n** 番目の文字] パラメータは、1 が最初の文字、2 が 2 番目の文字というように、「n」が n 番目の文字を示します。

パラメータ
nthCharacter 必須。検査する文字。

例文
次のステートメントは、フィールドキャストメンバ " 見出し " の 50 番目の文字の位置をチェックし、結果を変数 location に割り当てます。

```
-- Lingo syntax
location = member("Headline").charPosToLoc(50)

// JavaScript syntax
var location = member("Headline").charPosToLoc(50);
```

chars()

使用法

```
chars(stringExpression, firstCharacter, lastCharacter)
```

解説

式の中にある文字のサブストリングを識別する関数です (Lingo のみ)。

式 [先頭位置] および [末尾位置] は、ストリング内の位置を指定する必要があります。

[先頭位置] と [末尾位置] の値が同じ場合には、ストリング内で指定の位置にある 1 文字が返されます。ストリングの範囲外の値が [末尾位置] に指定されている場合は、ストリングの末尾までがサブストリングとして返されます。[末尾位置] が [先頭位置] よりも前にある場合には、EMPTY が返されます。

完成したムービーの chars() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Text ムービーを参照してください。

JavaScript シンタックスでは、String オブジェクトの substr() 関数を使用します。

パラメータ

stringExpression 必須。サブストリングを返す式を指定するストリング。

firstCharacter 必須。サブストリングが開始する位置を指定する整数。

lastCharacter 必須。サブストリングが終了する位置を指定する整数。

例文

次のステートメントは、**Adobe** という単語の 2 番目の文字を識別します。

```
put chars("Adobe", 2, 2)
-- "d"
```

次のステートメントは、**Adobe** という単語の 2 ～ 5 番目の文字を識別します。

```
put chars("Adobe", 2, 5)
-- "dobe"
```

次のステートメントは、Adobe という単語の 6 ～ 20 番目の文字を識別しようとしますが、単語には 10 文字しかないため、6 番目～ 10 番目までの文字のみが返されます。

```
put chars ("Adobe", 2, 20)
-- "dobe"
```

関連事項

char...of, length(), offset() (string function), number （文字）

charToNum()

使用法

```
(stringExpression).charToNum
charToNum(stringExpression)
```

解説

式の最初の文字に対応する ASCII コードを返す関数です (Lingo のみ)。

charToNum() 関数は、Ctrl または Control キーと文字キーなど、複数のキーの組み合わせで入力した文字の ASCII 値を調べるときに特に有効です。

Director は、等号 (=) 演算子を使って比較する場合は大文字と小文字の区別をしません。例えば、put ("M" = "m") というステートメントでは 1 か TRUE が返ってきます。

大文字と小文字を区別して比較する場合は、charToNum() を使って ASCII コードを取得しておき、ASCII コードで文字を照合するようにします。

JavaScript シンタックスでは、String オブジェクトの charCodeAt() 関数を使用します。

パラメータ

stringExpression 必須。検査する式を指定するストリング。

例文

次のステートメントは、アルファベットの "A" の ASCII コードを返します。

```
put ("A").charToNum
-- 65
```

次のハンドラは、入力された文字が大文字の A であるかどうかを判定し、大文字の場合または小文字の場合のマーカに分岐します。

```
-- Lingo syntax
on CheckKeyHit theKey
    if (theKey).charToNum = 65 then
        go "Correct Answer"
    else
        go "Wrong Answer"
    end if
end

// JavaScript syntax
function CheckKeyHit(theKey) {
    if (theKey.charToNum() == 65)
        go("Correct Answer");
    } else {
        go("Wrong Answer");
    }
}
```

関連事項

numToChar()

clearAsObjects()

使用法

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

解説

ActionScript オブジェクトで使用されていたグローバル Flash Player をリセットし、すべての ActionScript オブジェクトをメモリから消去するコマンドです。このコマンドは、Lingo 内で保管されているこれらのオブジェクトへの参照はリセットおよび消去しません。Lingo での参照は残りますが、存在しないオブジェクトへの参照となります。VOID を参照するよう、それぞれの参照を個別に設定する必要があります。

clearAsObjects() コマンドは、次のステートメントで作成された配列のように、グローバルオブジェクトだけに影響します。

```
-- Lingo syntax
myGlobalArray = newObject(#array)

// JavaScript syntax
myGlobalArray = new Array();
```

clearAsObjects() コマンドは、次のような、スプライト内で作成されたオブジェクトには効果がありません。

```
myArray = sprite(2).newObject(#array)
```

パラメータ

なし

例文

次のステートメントは、グローバルとして作成されたすべての ActionScript オブジェクトをメモリから消去します。

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

関連事項

newObject(), setCallback()

clearCache

使用法

```
clearCache
```

解説

このコマンドは、Director のネットワークキャッシュをクリアします。

clearCache コマンドは、ブラウザのキャッシュから独立している Director のキャッシュのみをクリアします。

ファイルが使用中の場合は、キャッシュをクリアできません。

パラメータ

なし

例文

次のハンドラは、ムービーが始まるときにキャッシュをクリアします。

```
-- Lingo syntax
on startMovie
    clearCache
end

// JavaScript syntax
function startMovie() {
    clearCache();
}
```

関連事項

cacheDocVerify(), cacheSize()

clearError()

使用法

```
-- Lingo syntax
memberObjRef.clearError()

// JavaScript syntax
memberObjRef.clearError();
```

解説

ストリーミング Flash キャストメンバのエラー状態を 0 にリセットする Flash コマンドです。

キャストメンバをメモリにストリーミングしている最中にエラーが発生すると、Director はキャストメンバの state プロパティを -1 にして、エラーが発生したことを示します。エラーが発生したときは、getError 関数を使用して、どのようなエラーが発生したのかを調べ、clearError コマンドを使用して、キャストメンバのエラー状態を 0 にリセットします。メンバのエラー状態を 0 にすると、Director ムービーでそのキャストメンバが必要な場合は、Director がそれを開こうとします。キャストメンバの pathName、linked、preload プロパティを設定しても、エラー状態は自動的にクリアされます。

パラメータ

なし

例文

次のハンドラは、"Dali" という Flash キャストメンバをメモリにストリーミングする最中に、メモリ不足エラーが発生していないか調べます。メモリ不足エラーが発生していた場合は、スクリプトは unloadCast コマンドを使用して、メモリの解放を試みます。次に、再生ヘッドを "Artists" という Director ムービーの、Flash ムービースプライトが最初に表示されるマーカに移動し、Flash ムービーを再び再生できるようにします。メモリ不足以外のエラーが発生した場合、スクリプトは、要求された Flash ムービーを再生できないことを表示する「お詫び」というマーカに移動します。

```
-- Lingo syntax
on CheckFlashStatus
    if (member("Dali").getError() = #memory) then
        member("Dali").clearError()
        member("Dali").unload()
        unloadCast
    else
        _movie.go("Sorry")
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
    var ge = member("Dali").getError();
    if (ge = "memory") {
        member("Dali").clearError();
        unloadCast;
        _movie.go("Artists");
    } else {
        _movie.go("Sorry");
    }
}
```

関連事項

state (Flash, SWA), getError() (Flash, SWA)

clearFrame()

使用法

```
-- Lingo syntax
_movie.clearFrame()

// JavaScript syntax
_movie.clearFrame();
```

解説

スコア録画中にフレームの全スプライトチャネルをクリアするムービーメソッドです。

パラメータ

なし

例文

次のハンドラでは、スコア生成時に各フレームの内容を消去してから編集しています。

```
-- Lingo syntax
on newScore
    _movie.beginRecording()
    repeat with counter = 1 to 50
        _movie.clearFrame()
        _movie.frameScript = 25
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end

// JavaScript syntax
function newScore() {
    _movie.beginRecording();
    for (var i = 1; i <= 50; i++) {
        _movie.clearFrame();
        _movie.frameScript = 25;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

関連事項

beginRecording(), endRecording(), ムービー , updateFrame()

clearGlobals()

使用法

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

解説

すべてのグローバル変数を VOID (Lingo) または null (JavaScript シンタックス) に設定するグローバルメソッドです。

グローバル変数を初期化するときや、別のグローバル変数のセットが必要なムービーを開くときに役立ちます。

パラメータ

なし

例文

次のハンドラでは、すべてのグローバル変数を VOID（Lingo）または null（JavaScript）に設定します。

```
-- Lingo syntax
on mouseDown
    _global.clearGlobals()
end

// JavaScript syntax
function mouseDown() {
    _global.clearGlobals();
}
```

関連事項

グローバル

clone

使用法

```
member(whichCastmember).model(whichModel).clone(cloneName)
member(whichCastmember).group(whichGroup).clone(cloneName)
member(whichCastmember).light(whichLight).clone(cloneName)
member(whichCastmember).camera(whichCamera).clone(cloneName)
```

解説

3D コマンド。モデル、グループ、ライト、またはカメラと、その子すべてのコピーを作成します。clone は、複製元のモデル、グループ、ライト、またはカメラの親を共有します。

モデルの複製は同じモデルリソースを使用し、オリジナルのモデルと同じ shaderList が割り当てられます。

複製名を指定しない場合、または "" を指定した場合、複製は count メソッドでカウントされません。ただし、シーンには表示されます。

パラメータ

cloneName 必須。新しい複製の名前を指定します。

例文

次のステートメントは、"Teapot" というモデルから "Teapot2" という複製を作成し、その複製への参照を返します。

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").clone("Teapot2")

// Javascript
teapotCopy = member("3D World").getProp("model","Teapot").clone("Teapot2")
```

関連事項

cloneDeep, cloneModelFromCastmember, cloneMotionFromCastmember, loadFile()

cloneDeep

使用法

```
member(whichCastmember).model(whichModel).cloneDeep(cloneName)
member(whichCastmember).group(whichGroup).cloneDeep(cloneName)
member(whichCastmember).light(whichLight).cloneDeep(cloneName)
member(whichCastmember).camera(whichCamera).cloneDeep(cloneName)
```

解説

3D コマンド。モデル、グループ、ライト、カメラに加えて、以下のすべてのコピーを作成します。

- オリジナルのモデルまたはグループで使用されているモデルリソース、シェーダ、およびテクスチャ
- モデル、グループ、ライト、またはカメラの子
- 子で使用されているモデルリソース、シェーダ、およびテクスチャ

このメソッドは、clone コマンドよりメモリを多く消費し、終了するまでの時間も長くなります。

パラメータ

cloneName 必須。新しい複製の名前を指定します。

例文

次のステートメントは、"Teapot" というモデルとその子のコピーを作成します。また、Teapot とその子で使用されているモデルリソース、シェーダ、テクスチャのコピーも作成します。変数 teapotCopy は、複製されたモデルへの参照です。

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").cloneDeep("Teapot2")

// Javascript
teapotCopy =member("3DWorld").getProp("model","Teapot").cloneDeep("Teapot2")
```

関連事項

clone, cloneModelFromCastmember, cloneMotionFromCastmember, loadFile()

cloneModelFromCastmember

使用法

```
member(whichCastmember).cloneModelFromCastmember(newModelName, sourceModelName, sourceCastmember)
```

解説

キャストメンバからモデルをコピーして、名前を変更し、それをキャストメンバに 3D ワールドの子として挿入する 3D コマンドです。

このコマンドは、モデルとその子で使用されているモデルリソース、シェーダ、およびテクスチャと同様に、ソースモデル名の子もコピーします。

このコマンドを正しく実行するには、ソースキャストメンバがロードされている必要があります。

パラメータ

newModelName 必須。新しく複製するモデルの名前を指定します。

sourceModelName 必須。複製するモデルを指定します。

sourceCastMember 必須。複製するモデルを含むキャストメンバを指定します。

例文

次のステートメントは、"Scene" というキャストメンバの "Pluto" というモデルのコピーを作成し、"Scene2" というキャストメンバに "Planet" の名前で挿入します。"Pluto" とその子で使用されているモデルリソース、シェーダ、およびテクスチャと同様に、"Pluto" の子も読み込まれます。

```
--Lingo
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"))

// Javascript
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"));
```

関連事項

cloneMotionFromCastmember, clone, cloneDeep, loadFile()

cloneMotionFromCastmember

使用法

```
member(whichCastmember).cloneMotionFromCastmember(newMotionName, sourceMotionName, sourceCastmember)
```

解説

キャストメンバからモーションをコピーして、名前を変更し、それをキャストメンバに挿入する 3D コマンドです。

このコマンドを正しく実行するには、ソースキャストメンバがロードされている必要があります。

パラメータ

newMotionName 必須。新しく複製するモーションの名前を指定します。

sourceMotionName 必須。複製するモーションを指定します。

sourceCastMember 必須。複製するモーションを含むキャストメンバを指定します。

例文

次のステートメントは、"ParkScene" というキャストメンバから " 歩く " というモーションをコピーし、これに "FunnyWalk" という名前を付けます。作成したコピーは、"gbMember" というキャストメンバに挿入します。

```
--Lingo
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"))

// Javascript
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"));
```

関連事項

map (3D), cloneModelFromCastmember, clone, cloneDeep, loadFile()

close()

使用法

```
-- Lingo syntax
windowObjRef.close()

// JavaScript syntax
windowObjRef.close();
```

解説

ウィンドウを閉じるウィンドウメソッドです。

既に閉じているウィンドウを閉じようとしてもエラーは発生しません。

ウィンドウを閉じても、そこで再生されていたムービーが自動的に停止することはなく、メモリから消えることもありません。このメソッドは、単に再生されているウィンドウを閉じるだけです。open() (Window) メソッドを使用すると、ウィンドウをすばやく再び開くことができます。あらかじめ準備しておいて、すばやくウィンドウを開いて再生するために使うと便利です。

完全にウィンドウを廃棄し、メモリからも消去する場合は、forget() メソッドを使用します。forget() メソッドを使用すると、そこで再生していたムービーも廃棄されます。スクリプトで廃棄したウィンドウにアクセスしようとするとエラーが発生します。

パラメータ
なし

例文
次のステートメントは、現行ムービーのフォルダ内の MIAW Source サブフォルダ内にあるパネルというウィンドウを閉じます。

```
-- Lingo syntax
window(_movie.path & "MIAW Sources\Panel").close()

// JavaScript syntax
window(_movie.path + "MIAW Sources\\Panel").close();
```

次のステートメントは、windowList の 5 番目のウィンドウを閉じます。

```
-- Lingo syntax
window(5).close()

// JavaScript syntax
window(5).close();
```

関連事項
forget() （ウィンドウ）, open() （ウィンドウ）, ウィンドウ

closeFile()

使用法

```
-- Lingo syntax
fileioObjRef.closeFile()

// JavaScript syntax
fileioObjRef.closeFile();
```

解説
ファイルを閉じる Fileio メソッドです。

パラメータ
なし

関連事項
Fileio

closeXlib

使用法

```
closeXlib whichFile
```

解説
Xlibrary ファイルを閉じるコマンドです。

Xtra は Xlibrary ファイル内に保存されています。Xlibrary ファイルは Xtra を含むリソースファイルです。Xlibrary には HyperCard XCMDs と XFCNs などを保存しておくこともできます。

closeXlib コマンドは URL には使用できません。

Windows では、オプションとして Xtra に DLL 拡張子を使うこともできます。

既に使用し終わったファイルは、随時閉じるようにします。

注意：このコマンドは、Shockwave Player ではサポートされていません。

パラメータ

whichFile オプション。閉じる Xlibrary ファイルを指定します。**whichFile** が現在のムービーと異なるフォルダにある場合は、**whichFile** でパス名を指定する必要があります。**whichFile** を指定しないと、開いているすべての Xlibrary が閉じられます。

例文

次のステートメントは、現在開いているすべての Xlibrary ファイルを閉じます。

```
closeXlib
```

次のステートメントは、ムービーと同じフォルダにある Video Disc Xlibrary という Xlibrary ファイルを閉じます。

```
closeXlib "Video Disc Xlibrary"
```

次のステートメントは、ムービーと同じフォルダ内の "New Xtras" フォルダにある "Xlibrary Transporter Xtra" という Xlibrary ファイルを閉じます。ディスクは変数 currentDrive によって識別されます。

```
closeXlib "@:New Xtras:Transporter Xtras"
```

関連事項

Interface(), openXlib

color()

使用法

```
-- Lingo syntax
color(intPaletteIndex)
color(intRed, intGreen, intBlue)

// JavaScript syntax
color(intPaletteIndex);
color(intRed, intGreen, intBlue);
```

解説

トップレベル関数およびデータ型。RGB または 8 ビットのパレットインデックス値を使用して、Color データオブジェクトを返します。

返された Color オブジェクトは、キャストメンバ、スプライト、および該当するステージに適用できます。

パラメータ

intPaletteIndex 8 ビットのパレット値を使用する場合に必須。使用する 8 ビットのパレット値を指定する整数。有効な値の範囲は 0 ～ 255 です。他の値はすべて切り捨てられます。

intRed RGB 値を使用する場合に必須。現在のパレットにおける赤い色の成分を指定する整数。有効な値の範囲は 0 ～ 255 です。他の値はすべて切り捨てられます。

intGreen RGB 値を使用する場合に必須。現在のパレットにおける緑の色の成分を指定する整数。有効な値の範囲は 0 ～ 255 です。他の値はすべて切り捨てられます。

intBlue RGB 値を使用する場合に必須。現在のパレットにおける青い色の成分を指定する整数。有効な値の範囲は 0 ～ 255 です。他の値はすべて切り捨てられます。

例文

次のステートメントは、メッセージウィンドウのスプライト 6 の色を表示し、スプライト 6 の色を新しい値に設定します。

```
-- Lingo syntax
put(sprite(6).color) -- paletteIndex(255)
sprite(6).color = color(137)
put(sprite(6).color) -- paletteIndex(137)

// JavaScript syntax
put(sprite(6).color) // paletteIndex(255);
sprite(6).color = color(137);
put(sprite(6).color) // paletteIndex(137);
```

compress()

使用法

```
byteArrayObject.compress()
```

解説

バイト配列メソッド。標準の Zlib 圧縮を使用して、バイト配列の内容を圧縮します。圧縮後に、位置は 1 に設定されます。

例文

```
--Lingo syntax
bArray.compress()

//JavaScript syntax
bArray.compress();
```

constrainH()

使用法

```
-- Lingo syntax
_movie.constrainH(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainH(intSpriteNum, intPosn);
```

解説

スプライトの左側と右側の水平座標に依存する整数値を返すムービーメソッドです。

返される整数は、次の 3 つの値のいずれかです。

- **intPosn** パラメータがスプライトの左座標と右座標の値の間である場合は、**intPosn** と等しい整数が返されます。

- **intPosn** パラメータがスプライトの左座標の値より小さい場合は、スプライトの左座標の値が返されます。
- **intPosn** パラメータがスプライトの右座標の値より大きい場合は、スプライトの右座標の値が返されます。

このメソッドはスプライトのプロパティを変更しません。

constrainH() メソッドと constrainV() メソッドはどちらも、1 つの軸のみを制限します。

パラメータ

intSpriteNum 必須。水平座標を **intPosn** と比較して評価するスプライトを指定する整数。

intPosn 必須。**intSpriteNum** で示されるスプライトの左側と右側の水平座標と比較して評価する整数。

例文

次のステートメントは、左右の水平座標値が 40 と 60 になっているスプライト 1 の constrainH を調べます。

```
-- Lingo syntax
put(constrainH(1, 20)) -- 40
put(constrainH(1, 55)) -- 55
put(constrainH(1, 100)) -- 60

// JavaScript syntax
put(constrainH(1, 20)); // 40
put(constrainH(1, 55)); // 55
put(constrainH(1, 100)); // 60
```

次のステートメントは、マウスポインタが目盛りの端より先に移動したとき、移動可能なスライダ (スプライト 1) が動く範囲を目盛り (スプライト 2) の端までに制限します。

```
-- Lingo syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH);
```

関連事項

constrainV(), ムービー

constrainV()

使用法

```
-- Lingo syntax
_movie.constrainV(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainV(intSpriteNum, intPosn);
```

解説

スプライトの上側と下側の垂直座標に依存する整数値を返すムービーメソッドです。

返される整数は、次の 3 つの値のいずれかです。

- **intPosn** パラメータがスプライトの上座標と下座標の値の間である場合は、**intPosn** と等しい整数が返されます。
- **intPosn** パラメータがスプライトの上座標の値より小さい場合は、スプライトの上座標の値が返されます。
- **intPosn** パラメータがスプライトの下座標の値より大きい場合は、スプライトの下座標の値が返されます。

このメソッドはスプライトのプロパティを変更しません。

constrainV() メソッドと constrainH() メソッドはどちらも、1 つの軸のみを制限します。

パラメータ

intSpriteNum 必須。垂直座標を **intPosn** と比較して評価するスプライトを示す整数。

intPosn 必須。**intSpriteNum** で示されるスプライトの上側と下側の垂直座標と比較して評価する整数。

例文

次のステートメントは、上下の垂直座標値が 40 と 60 になっているスプライト 1 の constrainV をチェックします。

```
-- Lingo syntax
put(constrainV(1, 20)) -- 40
put(constrainV(1, 55)) -- 55
put(constrainV(1, 100)) -- 60

// JavaScript syntax
put(constrainV(1, 20)); // 40
put(constrainV(1, 55)); // 55
put(constrainV(1, 100)); // 60
```

次のステートメントは、マウスポインタが目盛りの端より先に移動したとき、移動可能なスライダ (スプライト 1) が動く範囲を目盛り (スプライト 2) の端までに制限します。

```
-- Lingo syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH);
```

関連事項

constrainH(), ムービー

copyPixels()

使用法

```
-- Lingo syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList})

// JavaScript syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList});
```

解説

画像メソッド。既存の画像オブジェクト内の矩形領域の内容を、新しい画像オブジェクトにコピーします。

キャストメンバのある領域から別のキャストメンバの領域にピクセルをコピーするとき、オリジナルにコピーする前に複製した画像オブジェクトにピクセルをコピーします。同じ画像内である領域から別の領域に直接コピーするのは、不適切です。

copyPixels() でマットインクをシミュレートするには、createMatte() でマットオブジェクトを作成し、copyPixels() の #maskImage パラメータとしてそのオブジェクトを渡します。

完成したムービーの quad の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダにある Quad ムービーを参照してください。

copyPixel 中に、コピー元画像とコピー先画像のいずれかで useAlpha プロパティが false の場合、コピー先はアルファ情報を持ちません。コピー先にアルファコンテンツを含めるには、コピー元とコピー先の両方で useAlpha プロパティが true である必要があります。

パラメータ

sourceImgObj 必須。ピクセルのコピー元のソース画像オブジェクトへの参照。

destRectOrQuad スクリーン座標の矩形または浮動小数点数 quad にピクセルをコピーする場合は必須。ピクセルがコピーされる先の矩形または quad。

sourceRect 必須。ピクセルのコピー元のソース矩形。

paramList オプション。コピーされるピクセルを **destRect** または **destQuad** に配置する前にピクセルを操作するために使用できるパラメータリスト。プロパティリストには、次のパラメータのすべてまたは一部が含まれています。

プロパティ	使用方法と効果
#color	カラー効果に適用するフォアカラーです。デフォルトのカラーは、黒です。
#bgColor	カラー効果またはバックカラー透明度に適用するバックカラーです。デフォルトのカラーは、白です。
#ink	コピーしたピクセルに適用するインクの種類です。インクシンボルまたは該当するインク数値です。デフォルトのインクは、#copy です。
#blendLevel	コピーしたピクセルに適用する blend (透明度) の度合いです。値の範囲は 0 ～ 255 です。デフォルト値は 255 (不透明) です。255 より小さい値を使用すると、#ink の設定が #blend になります。または、元の値が #backgroundTransparent であった場合は、#blendTransparent になります。#blendLevel を #blend で置き換えることもできます。その場合は、0 ～ 100 の範囲の値を使用します。
#dither	8 ビットおよび 16 ビットの画像の **destRect** 内に配置するときに、コピーされるピクセルをディザリングするかどうかを決定する、TRUE または FALSE の値。デフォルト値は FALSE で、これはコピーしたピクセルを **imageObjRef** のカラーパレットに直接マップします。
#useFastQuads	**destQuad** にピクセルをコピーするときに、Director で使用できる正確さよりも速さを優先する方法を使用して quad 計算を行うかどうかを指定する、TRUE または FALSE の値。単純な回転操作および傾斜操作に quad を使用する場合は、TRUE に設定します。パースの変形に使用する場合など、任意の quad に対しては FALSE に設定します。デフォルト値は FALSE です。
#maskImage	コピーするピクセルのマスクとして使用するマスクオブジェクトまたはマットオブジェクトを指定します。これらのオブジェクトは、creatMask() または createMatte() で作成します。これにより、マスクおよびマットスプライトインクの効果を複製することができます。ソース画像にアルファチャネルがあり、その useAlpha プロパティが TRUE の場合は、アルファチャネルが使用され、指定したマスクまたはマットは無視されます。デフォルトでは、マスクが指定されていません。
#maskOffset	#maskImage で指定したマスクに適用する x オフセットおよび y オフセットの大きさを示すポイント。オフセットの基準は、コピー元画像の左上隅です。既定のオフセットは、(0,0) です。

例文

次のステートメントは、メンバ " ハッピー " の画像全体をメンバ " 花 " にコピーします。メンバ同士のサイズが違う場合、メンバ " ハッピー " の画像がメンバ " 花 " のサイズに調整されます。

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect)

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect);
```

次のステートメントは、メンバ " ハッピー " の画像の一部をコピーし、メンバ " 花 " の一部にコピーします。" ハッピー " からコピーされる画像の部分は、矩形 (0, 0, 200, 90) の内部です。メンバ " 花 " の画像内の矩形 (20,20,100,40) にペーストされます。ハッピーのコピーした部分は、ペーストされる矩形に合うようにサイズ調整されます。

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image, rect(20,20,100,40),rect(0,0,200,90))

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image, rect(20,20,100,40),rect(0,0,200,90))
```

次のステートメントは、メンバ " ハッピー " の画像全体をメンバ " 花 " の矩形にコピーします。コピーしたメンバ " ハッピー " の画像がペーストされた矩形は、メンバ " ハッピー " の矩形と同じサイズで、コピーした画像は、サイズ調整されていません。コピーした画像の blend 値は、50 の半透明で、メンバ " 花 " の一部の上にペーストされたことがわかります。

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310), member("Happy").image.rect,
[#blendLevel: 50])

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310), member("Happy").image.rect,
\propList(symbol("blendLevel"),50))
```

関連事項

color(), image()

copyToClipBoard()

使用法

```
-- Lingo syntax
memberObjRef.copyToClipBoard()

// JavaScript syntax
memberObjRef.copyToClipBoard();
```

解説

指定されたキャストメンバをクリップボードにコピーするメンバメソッドです。

キャストウィンドウがアクティブになっていなくても、このメソッドを呼び出すことができます。

このメソッドは、ムービーとアプリケーションの間でキャストメンバをコピーするときに便利です。

パラメータ

なし

例文

次のステートメントは、キャストメンバ " 椅子 " をクリップボードにコピーします。

```
-- Lingo syntax
member("chair").copyToClipBoard()

// JavaScript syntax
member("chair").copyToClipBoard();
```

次のステートメントは、キャストメンバ 5 をクリップボードにコピーします。

```
--- Lingo syntax
member(5).copyToClipBoard()

// JavaScript syntax
member(5).copyToClipBoard();
```

関連事項

メンバ , pasteClipBoardInto()

cos()

使用法

```
(angle).cos
cos (angle)
```

解説

指定した角度のコサイン（余弦）を計算する関数です（Lingo のみ）。角度はラジアンで表記する必要があります。

JavaScript シンタックスでは、Math オブジェクトの cos() 関数を使用します。

パラメータ

angle 必須。検査する角度を指定する整数。

例文

次のステートメントは、PI/2 のコサインを計算し、結果をメッセージウィンドウに表示します。

```
put (PI/2).cos
```

関連事項

atan(), PI, sin()

count()

使用法

```
-- Lingo syntax
list.count
object.count

// JavaScript syntax
list.count;
object.count;
```

解説

リニアリストまたはプロパティリストの項目数を返す関数です。親スクリプトのプロパティ数を数える場合、アンセスタスクリプトのプロパティや図、線または単語などテキスト表記のチャンクは数には入れません。

count コマンドはリニアリスト、プロパティリスト、親スクリプトで作成されたオブジェクトおよび the globals プロパティのいずれでも動作します。

完成したムービーの count() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Text ムービーを参照してください。

次のシンタックスと共に関数として count を使用すると、

```
x = count(object)
x = object.count()
```

ベースオブジェクトのプロパティの数が正しく返されます。次のシンタックスと共にプロパティとして count を使用すると、

```
x = object.count
```

アンセスタのプロパティの数が返されます。

パラメータ
なし

例文
次のステートメントは、リスト内の項目数 "3" を表示します。

```
--Lingo syntax
put([10,20,30].count) -- 3

// JavaScript syntax
put(list(10,20,30).count); // 3
```

Java スクリプトは、count(object) をサポートしません。

関連事項
globals

createFile()

使用法

```
-- Lingo syntax
fileioObjRef.createFile(stringFileName)

// JavaScript syntax
fileioObjRef.createFile(stringFileName);
```

解説
指定されたファイルを作成する Fileio メソッドです。

パラメータ
stringFileName 必須。作成するファイルのパスと名前を指定するストリング。

例文
次のサンプルは、c:\ に xtra.txt という名前のファイルを作成します。

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile("c:\xtra.txt")

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile("c:\xtra.txt");
```

関連事項
Fileio

createMask()

使用法

```
imageObject.createMask()
```

解説

この関数は、copyPixels() 関数と一緒に使用されるマスクオブジェクトを作成し、返します。

マスクオブジェクトは、画像オブジェクトではありません。マスクスプライトインクの効果を複製する際に copyPixels() 関数とのみ有効です。時間を節約するには、複数回マスクとして同じ画像を使う場合、マスクオブジェクトを作成し、再利用できるように変数として保存します。

例文

次のステートメントは、メンバの茶色の正方形の画像内の矩形にハッピーメンバの全体画像をコピーします。メンバ "gradient2" は、コピーした画像と共にマスクとして使用されます。マスクは、メンバ " ハッピー " の画像がペーストされている矩形の 10 ピクセル上と左にオフセットされます。

```
member("brown square").image.copyPixels(member("Happy").image, rect(20, 20, 150, 108),
member("Happy").rect,[#maskImage:member("gradient2").image.createMask(), maskOffset:point(-10, -10)])
```

関連事項

copyPixels(), createMatte(), ink

createMatte()

```
imageObject.createMatte({alphaThreshold})
```

解説

この関数は、マットスプライトインクの効果を複製する際に copyPixels() 関数と一緒に使用できるマットオブジェクトを作成し、返します。マットオブジェクトは、特定の画像オブジェクトのアルファレイヤーから作成されます。オプションパラメータアルファしきい値は、マットからアルファチャネル値がしきい値を下回るすべてのピクセルを除外します。アルファチャネルを持つ 32 ビット画像とのみ使用します。アルファしきい値 は、0 ～ 255 の間の値です。

マットオブジェクトは、画像オブジェクトではありません。マスクスプライトインクの効果を複製する際に copyPixels() 関数とのみ使用します。時間を短縮するために、1 度以上マットとして同じ画像を使おうとする場合、マットを作成し、再利用できるように変数として保存します。

例文

次のステートメントは、画像オブジェクト testImage のアルファレイヤーから新しいマットオブジェクトを作成し、50% (255 の半分) を下回るアルファ値を持つピクセルを無視します。

```
newMatte = testImage.createMatte(128)
```

関連事項

copyPixels(), createMask()

createSoundObject

使用法

```
SoundObject Mixer.createSoundObject (SoundObjname, filepath, [startTime, endTime, loopCount, loopStartTime,
loopEndTime, preLoadTime])
SoundObject Mixer.createSoundObject(SoundObjname, castMem, [startTime, endTime, loopCount, loopStartTime,
loopEndTime, preLoadTime])
SoundObject Mixer.createSoundObject(SoundObjname,callbackFunction,[castMemRef],
[sampleRate,channelcount,bitDepth]) -- For more information, see 146 ページの「サウンドオブジェクトの入力としてのバイト配列」])
```

戻り値

SoundObject

解説

オーディオメソッド。サウンドオブジェクトを作成します。

パラメータ

パラメータ	解説	デフォルト値
SoundObjName	サウンドオブジェクトと関連付ける名前。複数のサウンドオブジェクトに同じ名前を付けることはできません。	
filepath	Director でサウンドオブジェクトを使用して再生するファイルの名前と位置。	
castMem	そのサウンドオブジェクトを使用して Director で再生するサウンドキャストメンバの名前。	
proplist	サウンドオブジェクトのプロパティの指定に使用するプロパティリスト。	
startTime	現在のサウンドオブジェクトの開始時間をミリ秒単位で指定します。	サウンドの開始
endTime	現在のサウンドオブジェクトの終了時間をミリ秒単位で示します。	サウンドの終了
loopCount	#loopStartTime と #loopEndTime によって定義されるサウンドオブジェクトの部分を再生する回数。	1
loopStartTime	サウンドオブジェクトのループする部分の開始位置（サウンドオブジェクト内でのミリ秒位置）。	
loopEndTime	サウンドオブジェクトのループする部分の終了位置（サウンドオブジェクト内でのミリ秒位置）。	
preLoadTime	再生を開始する前に Director がバッファに格納するサウンドの長さ（ミリ秒単位）。	

例文

次の例文は、member(3) から指定の propertylist を使用することにより、soundObject1 というサウンドオブジェクトを作成します。新しいサウンドオブジェクトは mixer1 に追加されます。作成されたサウンドオブジェクトに対して、参照 soundObjRef が返されます。

```
-- Lingo syntax
on mouseUp me
mixer.createSoundObject("SoundObj1", member(3), [ #startTime: 20000, #endTime: 80000, #loopCount: 5,
#loopStartTime: 50000 , #loopEndTime :75000, #preLoadTime:6000]
end

//JavaScript syntax
function mouseup()
{
soundObjRef=mixer1.createSoundObject("soundobj1",member(3),propList(symbol("startTime")
,10000,symbol("endTime") ,40000, symbol("loopcount") ,5 ,symbol("loopstarttime") ,20000,
symbol("loopendtime") ,30000, symbol("preLoadTime") ,6000));}
```

関連事項

ミクサ

crop() （画像）

使用法

```
-- Lingo syntax
imageObjRef.crop(rectToCropTo)

// JavaScript syntax
imageObjRef.crop(rectToCropTo);
```

解説

画像メソッド。指定された矩形で切り取られたソース画像オブジェクトのコピーを含む新しい画像オブジェクトを返します。

crop() を呼び出しても、ソース画像オブジェクトは変化しません。

新しい画像オブジェクトは、どのキャストメンバにも属さず、ステージとは何の関連付けもありません。新しい画像をキャストメンバに割り当てるには、キャストメンバの image プロパティを設定します。

パラメータ

rectToCropTo 必須。新しい画像を切り取る矩形。

例文

次のステートメントは、" インタビュー " デジタルビデオキャストメンバをスプライトに合わせて切り取るように Lingo に指示します。

```
-- Lingo
Dot syntax:
member("Interview").crop = TRUE
Verbose syntax:
set the crop of member "Interview" to TRUE

// Javascript
member("Interview").crop=true
```

関連事項

image （画像）, image(), rect （画像）

crop() （ビットマップ）

使用法

```
-- Lingo syntax
memberObjRef.crop()

// JavaScript syntax
memberObjRef.crop();
```

解説

ビットマップのキャストメンバを特定のサイズに切り取るビットマップコマンドです。

既存のビットマップメンバをトリムしたり、ステージの画像と共に、スナップショットをとり、表示サイズに切り取るために crop を使用できます。

ビットマップが元の位置に応じて移動しないように、登録ポイントは、同じ場所に保存されます。

パラメータ

rectToCropTo 必須。キャストメンバを切り取る矩形を指定します。

例文

次のステートメントは、既存のビットマップメンバ " ステージ画像 " にステージのスナップショットを保存し、その画像サイズをスプライト 10 に等しい矩形に切り取ります。

```
-- Lingo syntax
stageImage = (_movie.stage).image
spriteImage = stageImage.crop(sprite(10).rect)
member("sprite snapshot").image = spriteImage

// JavaScript syntax
var stageImage = (_movie.stage).image;
var spriteImage = stageImage.crop(sprite(10).rect);
member("sprite snapshot").image = spriteImage;
```

関連事項

picture （メンバ）

cross

使用法

```
vector1.cross(vector2)
```

解説

3D ベクトルメソッド。ベクトル **1** およびベクトル **2** の両方に垂直なベクトルを返します。

例文

次の例の pos1 は x 軸上のベクトルであり、pos2 は y 軸上のベクトルです。pos1.cross(pos2) で返される値は vector(0.0000, 0.0000, 1.00000e4) です。このベクトルは、pos1 および pos2 の両方に垂直です。

```
ppos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.cross(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.cross(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

関連事項

crossProduct(), perpendicularTo

crossProduct()

使用法

```
vector1.crossProduct(vector2)
```

解説

3D ベクトルメソッド。ベクトル **1** およびベクトル **2** の両方に垂直なベクトルを返します。

例文

次の例では、pos1 は x 軸上のベクトルで、pos2 は y 軸上のベクトルです。pos1.crossProduct(pos2) で返される値は vector(0.0000, 0.0000, 1.00000e4) です。このベクトルは、pos1 および pos2 の両方に垂直です。

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.crossProduct(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.crossProduct(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

関連事項

perpendicularTo, cross

cursor()

使用法

```
-- Lingo syntax
_player.cursor(intCursorNum)
_player.cursor(cursorMemNum, maskMemNum)
_player.cursor(cursorMemRef)

// JavaScript syntax
_player.cursor(intCursorNum);
_player.cursor(cursorMemNum, maskMemNum);
_player.cursor(cursorMemRef);
```

解説

カーソルに使用するキャストメンバまたは内蔵カーソルを変更するプレーヤメソッドです。カーソルを 0 に設定して解除するまで有効です。

- カーソルおよびそのオプションのマスクとして使用するキャストメンバの番号を指定するには、_player.cursor(**cursorMemNum**, **maskMemNum**) というシンタックスを使用します。この場合、キャストメンバのセンターポイントがカーソルのホットスポットとして使われます。

  指定するキャストメンバは、1 ビットのキャストメンバにする必要があります。キャストメンバが 16 × 16 ピクセル以上の場合、Director は画像の左上隅から 16 × 16 の正方形に収まるように切り取ります。この場合もやはり、キャストメンバのセンターポイントがカーソルのホットスポットとして使われます。

- カーソル Xtra を通じて使用可能なカスタムカーソルの場合は、_player.cursor(**cursorMemRef**) というシンタックスを使用します。

  注意：カーソル Xtra では異なるキャストライブラリタイプのカーソルを使用できますが、テキストキャストメンバをカーソルとして使用することはできません。

- デフォルトのシステムカーソルを指定するには、_player.cursor(**intCursorNum**) というシンタックスを使用します。**intCursorNum** には、次のいずれかの整数値を指定します。

値	解説
-1、0	矢印
1	I 字
2	十字
3	クロスバー
4	時計（Mac）または砂時計（Windows）
5	東西南北（NSEW）
6	南北（NS）
200	空白（カーソルは非表示）
254	ヘルプ
256	えんぴつ
257	消しゴム
258	選択
259	バケツ
260	ハンド
261	矩形ツール
262	角丸矩形ツール
263	円ツール
264	ラインツール
265	リッチテキストツール
266	テキストフィールドツール
267	ボタンツール

値	解説
268	チェックボックスツール
269	ラジオボタンツール
270	配置ツール
271	センターポイントツール
272	投げ縄
280	指
281	スポイト
282	マウスボタン押下待ち 1
283	マウスボタン押下待ち 2
284	垂直サイズ
285	水平サイズ
286	対角線サイズ
290	閉じたハンド
291	非ドロップハンド
292	コピー（閉じたハンド）
293	逆矢印
294	回転
295	傾斜
296	水平二重矢印
297	垂直二重矢印
298	南西北西の二重矢印
299	北西南東の二重矢印
300	にじみ / ソフトブラシ
301	エアブラシ
302	ズームイン
303	ズームアウト
304	ズームキャンセル
305	シェイプを開始
306	ポイントを追加
307	シェイプを閉じる
308	ズームカメラ
309	カメラを移動
310	カメラを回転
457	カスタム

ファイルの読み込みなどのシステムイベントを実行している間は、システムの腕時計カーソルが表示され、制御がアプリケーションに戻された時点で矢印 (ポインタ) カーソルに変わります。これは、システムカーソルが cursor コマンド設定より優先されるために起こります。複数のムービーでカスタムカーソルを使うプレゼンテーションなどで cursor() をすべての新しいムービーの最初から使うためには、カスタムカーソルリソースの番号をムービー間のメモリにグローバル変数として格納します。

cursor コマンドは Xtra などの外部エージェントに割り込まれることもあります。Director 内でカーソル用の値を設定しても、Xtra などの外部エージェントがカーソルを制御する場合、Director はカーソルが変わっていることに気づかないためカーソルを元の値に再設定しても効果がありません。このような状況になった場合は、一旦カーソルに他の値 (0 など) を仮設定として割り当ててから、元の値に設定し直してください。

パラメータ

intCursorNum カーソルの指定に整数を使用する場合は必須。カーソルとして使用する組み込みカーソルを指定する整数です。

cursorMemNum カーソルの指定にキャストメンバ番号とそのオプションのマスクを使用する場合は必須。カーソルとして使用するキャストメンバ番号を指定する整数です。

maskMemNum カーソルの指定にキャストメンバ番号とそのオプションのマスクを使用する場合は必須。**cursorMemNum** のマスク番号を指定する整数です。

cursorMemRef カーソルの指定にキャストメンバ参照を使用する場合は必須。カーソルとして使用するキャストメンバへの参照です。

例文

次のステートメントは、変数 status の値が 1 のときに、Windows ではカーソルを砂時計カーソルに、Mac では腕時計カーソルにします。

```
-- Lingo syntax syntax
if (status = 1) then
    _player.cursor(4)
end if

// JavaScript syntax
if (status == 1) {
    _player.cursor(4);
}
```

次のハンドラは、変数に割り当てられているキャストメンバが 1 ビットであるかどうかをチェックし、1 ビットである場合、それをカーソルとして使用します。

```
-- Lingo syntax syntax
on myCursor(someMember)
    if (member(someMember).depth = 1) then
        _player.cursor(someMember)
    else
        _sound.beep()
    end if
end

// JavaScript syntax
function myCursor(someMember) {
    if (member(someMember).depth == 1) {
        _player.cursor(someMember);
    }
    else {
        _sound.beep();
    }
}
```

関連事項

プレーヤ

date() （フォーマット）

使用法

```
-- Lingo syntax syntax
date({stringFormat})
date({intFormat})
date({intYearFormat, intMonthFormat, intDayFormat})

// JavaScript syntax
Date({"month dd, yyyy hh:mm:ss"});
Date({"month dd, yyyy"});
Date({yy,mm,dd,hh,mm,ss});
Date({yy,mm,dd});
Date({milliseconds});
```

解説

トップレベル関数およびデータ型。フォーマットされた標準の日付オブジェクトのインスタンスを作成します。数学的な演算で他の日付オブジェクトのインスタンスと一緒に使用されたり、プラットフォーム間およびインターナショナルフォーマットで日付を処理するときにも使用されます。

Lingo の日付オブジェクトと JavaScript シンタックスの日付オブジェクトは異なります。したがって、JavaScript シンタックスを使用して Lingo の日付オブジェクトを作成することはできず、Lingo シンタックスを使用して JavaScript シンタックスの日付オブジェクトを作成することはできません。

新しい JavaScript シンタックス日付オブジェクトは、new Date() シンタックスを使用して作成します。JavaScript シンタックスでは大文字と小文字の区別が重要です。例えば、new date() を使用するとランタイムエラーになります。

Lingo を使用して日付を作成するには、年を表すのに 4 桁、月と日にはそれぞれ 2 桁を使用します。次の式はすべて、2004 年 10 月 21 日の日付オブジェクトを返します。

日付フォーマット	使用法
ストリング	date(“20041021”)
整数	date(20041021)
カンマ区切り :	date(2004, 10, 21)

返される日付オブジェクトの個々のプロパティは次のとおりです。

プロパティ	解説
#year	年を表す整数
#month	その年の月を表す整数
#day	その月の日を表す整数

日付の加算および減算は、日数の加算および減算として計算されます。

パラメータ

stringFormat Lingo 日付オブジェクトを作成するときはオプション。新しい日付オブジェクトを指定するストリングです。

intFormat Lingo 日付オブジェクトを作成するときはオプション。新しい日付オブジェクトを指定する整数です。

intYearFormat Lingo 日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの 4 桁の年を指定する整数です。

intMonthFormat Lingo 日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの 2 桁の月を指定する整数です。

intDayFormat Lingo 日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの 2 桁の日を指定する整数です。

month JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの月を指定するストリングです。有効な値の範囲は 0 (1 月) ～ 11 (12 月) です。

dd JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの日を指定する 2 桁の整数です。有効な値の範囲は 0 (日曜日) ～ 6 (土曜日) です。

yyyy JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの年を指定する 4 桁の整数です。

hh JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの時を指定する 2 桁の整数です。有効な値の範囲は 0 (午前 0 時) ～ 23 (午後 11 時) です。

mm JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの分を指定する 2 桁の整数です。有効な値の範囲は 0 ～ 59 です。

ss JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの秒を指定する 2 桁の整数です。有効な値の範囲は 0 ～ 59 です。

yy JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトの年を指定する 2 桁の整数です。有効な値の範囲は 0 ～ 99 です。

milliseconds JavaScript シンタックスの日付オブジェクトを作成するときはオプション。新しい日付オブジェクトのミリ秒を指定する整数です。有効な値の範囲は 0 ～ 999 です。

例文

次のステートメントは、2 つの日付の間の日数を計算します。

```
-- Lingo syntax syntax
myBirthday = date(19650712)
yourBirthday = date(19450529)
put("There are" && abs(yourBirthday - myBirthday) && "days between our birthdays.")

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
var yourBirthday = new Date(1945, 05, 29);
put("There are " + Math.abs(((yourBirthday - myBirthday)/1000/60/60/24)) + " days between our birthdays.");
```

次のステートメントは、日付の個々のプロパティにアクセスします。

```
-- Lingo syntax syntax
myBirthday = date(19650712)
put("I was born in month number" && myBirthday.month)

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
put("I was born in month number " + myBirthday.getMonth());
```

date() （システム）

使用法

```
-- Lingo syntax
_system.date({yyyymmdd})

// JavaScript syntax
_system.date({yyyymmdd});
```

解説

システムクロックで現在の日付を返すシステムメソッドです。

Director が使用する日付のフォーマットは、コンピュータが日付をどのようなフォーマットで表示するかによって決まります。

- Windows では、[地球のオプション] コントロールパネルで日付表示フォーマットをカスタマイズすることができます。Windows では、現在使用されている日付のショートフォーマットが System.ini ファイルに保存されます。この値を使用して、日付のショートフォーマットの各部が何を示すのかを決めます。
- Macintosh では、日付 & 時刻コントロールパネルで表示フォーマットをカスタマイズすることができます。

パラメータ

yyyymmdd オプション。返される日付の 4 桁の年 (**yyyy**)、2 桁の月 (**mm**)、および 2 桁の日 (**dd**) を指定する値です。

例文

次のステートメントは、日付の最初の 4 文字が "1/1/" になっているかどうかを調べて、現在の日付が 1 月 1 日かどうかを判断します。1 月 1 日だった場合は、" あけましておめでとうございます !" を表示します。

```
-- Lingo syntax
if (_system.date().char[1..4] = "1/1/") then
    _player.alert("Happy New Year!")
end if

// JavaScript syntax
if (_system.date().toString().substr(0, 4) == "1/1/") {
    _player.alert("Happy New Year!");
}
```

関連事項

システム

delay()

使用法

```
-- Lingo syntax
_movie.delay(intTicks)

// JavaScript syntax
_movie.delay(intTicks);
```

解説

再生ヘッドを指定された時間だけ停止するムービーメソッドです。

停止中に可能なマウスまたはキーボード操作は、ムービー自体を停止する操作、つまり、Ctrl+Alt+ ピリオド (Windows の場合) または Command+ ピリオド (Mac の場合) だけです。delay() は、個々のフレームの時間を長くするので、一連のフレームにおける再生レートを制御するのに便利です。

delay() メソッドは、再生ヘッドが移動しているときにのみ適用できます。ただし、delay() が有効であっても、ハンドラは動作します。再生ヘッドが止まるだけで、スクリプトの実行は停止されません。delay() を使用するスクリプトは、enterFrame ハンドラまたは exitFrame ハンドラに置きます。

再生ヘッドが動いていないときにハンドラ内で擬似的に休止状態にするには、System オブジェクトの milliseconds プロパティを使用し、指定した時間だけ待ってからフレームを終了します。

パラメータ

intTicks 必須。再生ヘッドを一時停止するチックの数を指定する整数。1 チックは 1/60 秒です。

例文

次のハンドラは、キーが押されると 2 秒だけムービーを停止します。

```
-- Lingo syntax
on keyDown
    _movie.delay(2*60)
end

// JavaScript syntax
function keyDown() {
    _movie.delay(2*60)
}
```

次のハンドラは、Return キーを押している間、ムービーをランダムなチック数だけ遅らせます。このハンドラは、フレームスクリプト内に入れることもできます。

```
-- Lingo syntax
on keyDown
    if (_key.key = "x") then
        _movie.delay(random(180))
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "x") {
        _movie.delay(random(180));
    }
}
```

関連事項

endFrame, milliseconds, ムービー

delete()

使用法

```
delete chunkExpression
```

解説

文字コンテナ内の任意の文字、単語、項目、行をチャンク式として指定できます。

パラメータ
なし

例文
```
delete member(1).line[1]
```
このステートメントは、テキストメンバの 1 番目の行を削除します。
```
delete member(1).word[1]
```
このステートメントは、テキストメンバの 1 番目の語を削除します。
```
delete member(1).char[1]
```
このステートメントは、テキストメンバの 1 番目の文字を削除します。

delete() (FileIO)

使用法
```
-- Lingo syntax
fileioObjRef.delete()

// JavaScript syntax
fileioObjRef.deleteFile();
```

解説
ファイルを削除する Fileio メソッドです。JavaScript では、Director 11 で deleteFile() のサポートが追加されました。

パラメータ
なし

関連事項
Fileio

deleteFile()

使用法
```
-- Lingo syntax
fileioObjRef.deleteFile()

// JavaScript syntax
fileioObjRef.deleteFile();
```

解説
ファイルを削除する Fileio メソッドです。このメソッドは Director 11 で追加されました。

パラメータ
なし

関連事項
Fileio

deleteAt

使用法

```
list.deleteAt(number)
deleteAt list, number
```

解説
リニアリストまたはプロパティリストから項目を削除するリストコマンドです。

deleteAt コマンドは、削除する項目がリストにあるかどうかを確認します。もし、削除する項目がない場合、Director はアラートを出します。

パラメータ
number 必須。削除する項目のリスト内での位置を指定します。

例文
次のステートメントは、[" ジー ", " ケイン ", " オーハシ "] を含む designers リストの 2 番目の項目を削除します。

```
--Lingo
designers = ["gee", "kayne", "ohashi"]
designers.deleteAt(2)

// Javascript
Designers = list("gee","kayne","ohashi");
Designers.deleteAt(2);
```

結果は、[" ジー ", " オーハシ "] というリストになります。

次のハンドラは、theIndex の位置の項目が theList リストに含まれているかどうかを調べてから削除を実行します。

```
on myDeleteAt theList, theIndex
    if theList.count < theIndex then
        beep
    else
        theList.deleteAt(theIndex)
    end if
end
```

関連事項
addAt

deleteCamera

使用法

```
member(whichCastmember).deleteCamera(cameraName)
member(whichCastmember).deleteCamera(index)
sprite(whichSprite).deleteCamera(cameraOrIndex)
```

解説

3D コマンド。キャストメンバでは、このコマンドはキャストメンバおよび 3D ワールドからカメラを削除します。カメラの子は、3D ワールドからは取り除かれますが、実際には削除されません。

キャストメンバのデフォルトのカメラを削除することはできません。

スプライトでは、このコマンドはスプライトのカメラリストからカメラを削除します。カメラは、キャストメンバからは削除されません。

パラメータ

cameraNameOrNum 必須。削除するカメラの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次のステートメントは、「部屋」というキャストメンバから 2 つのカメラを削除します。1 つは「カメラ 06」というカメラで、もう 1 つは 1 番目のカメラです。

```
member("Room").deleteCamera("Camera06")
member("Room").deleteCamera(1)
```

次のステートメントは、スプライト 5 のカメラリストから 2 つのカメラを削除します。1 つはリストの 2 番目のカメラで、もう 1 つは「カメラ 06」です。

```
sprite(5).deleteCamera(2)
sprite(5).deleteCamera(member("Room").camera("Camera06"))
```

関連事項

newCamera, addCamera, cameraCount()

deleteFrame()

使用法

```
-- Lingo syntax
_movie.deleteFrame()

// JavaScript syntax
_movie.deleteFrame();
```

解説

スコア生成セッションの間にのみ、現在のフレームを削除し、次のフレームを新しい現在のフレームにするムービーメソッド。

パラメータ

なし

例文

次のハンドラは、現在のフレームのチャネル 10 内のスプライトが 640 × 480 ピクセルのステージの右端から外に出たかどうかを確認し、出ている場合はフレームを削除します。

```
-- Lingo syntax
on testSprite
    _movie.beginRecording()
    if (sprite(10).locH > 640) then
        _movie.deleteFrame()
    end if
    _movie.endRecording()
end

// JavaScript syntax
function testSprite() {
    _movie.beginRecording();
    if (sprite(10).locH > 640) {
        _movie.deleteFrame();
    }
    _movie.endRecording();
}
```

関連事項

beginRecording(), endRecording(), ムービー , updateFrame()

deleteGroup

使用法

```
member(whichCastmember).deleteGroup(whichGroup)
member(whichCastmember).deleteGroup(index)
```

解説

3D コマンド。キャストメンバおよび 3D ワールドからグループを削除します。グループの子は、3D ワールドからは取り除かれますが、実際には削除されません。

" ワールド " というグループを削除することはできません。これはデフォルトのグループです。

パラメータ

groupNameOrNum 必須。削除するグループの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次の例の 1 行目は、キャストメンバ " シーン " からグループ " ダミー 16" を削除します。2 行目は、" シーン " の 3 番目のグループを削除します。

```
member("Scene").deleteGroup("Dummy16")
member("Scene").deleteGroup(3)
```

関連事項

newGroup, child (3D), parent

deleteLight

使用法

```
member(whichCastmember).deleteLight(whichLight)
member(whichCastmember).deleteLight(index)
```

解説
3D コマンド。キャストメンバおよび 3D ワールドからライトを削除します。ライトの子は、3D ワールドからは取り除かれますが、実際には削除されません。

パラメータ
lightNameOrNum 必須。削除するライトの名前またはインデックス位置を指定するストリングまたは整数です。

例文
次の例は、" 部屋 " というキャストメンバからライトを削除します。

```
member("Room").deleteLight("ambientRoomLight")
member("Room").deleteLight(6)
```

関連事項
newLight

deleteModel

使用法

```
member(whichCastmember).deleteModel(whichModel)
member(whichCastmember).deleteModel(index)
```

解説
3D コマンド。キャストメンバおよび 3D ワールドからモデルを削除します。モデルの子は、3D ワールドからは取り除かれますが、実際には削除されません。

パラメータ
modelNameOrNum 必須。削除するモデルの名前またはインデックス位置を指定するストリングまたは整数です。

例文
次の例の 1 行目は、"gbWorld" というキャストメンバから "Player3" というモデルを削除します。2 行目は、"gbWorld" の 9 番目のモデルを削除します。

```
member("gbWorld").deleteModel("Player3")
member("gbWorld").deleteModel(9)
```

関連事項
newModel

deleteModelResource

使用法

```
member(whichCastmember).deleteModelResource(whichModelResource)
member(whichCastmember).deleteModelResource(index)
```

解説
3D コマンド。キャストメンバおよび 3D ワールドからモデルリソースを削除します。

削除されたモデルリソースを使用しているモデルは、ジオメトリを失うため表示されなくなります。ただし、ワールドからは削除されていません。

パラメータ

resourceNameOrNum 必須。削除するモデルリソースの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次の例は、"StreetScene" というキャストメンバから 2 つのモデルリソースを削除します。

```
member("StreetScene").deleteModelResource("HouseB")
member("StreetScene").deleteModelResource(3)
```

関連事項

newModelResource, newMesh

deleteMotion

使用法

```
member(whichCastmember).deleteMotion(whichMotion)
member(whichCastmember).deleteMotion(index)
```

解説

3D コマンド。キャストメンバからモーションを削除します。

パラメータ

motionNameOrNum 必須。削除するモーションの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次の例の 1 行目は、"PicnicScene" というキャストメンバから "BackFlip" というモーションを削除します。2 行目は、"PicnicScene" の 5 番目のモーションを削除します。

```
member("PicnicScene").deleteMotion("BackFlip")
member("PicnicScene").deleteMotion(5)
```

関連事項

newMotion(), removeLast()

deleteOne

使用法

```
list.deleteOne(value)
deleteOne list, value
```

解説

リニアまたはプロパティリストから [値] を削除するコマンドです。プロパティリストの場合、deleteOne は、削除された値に関連するプロパティも一緒に削除されます。指定された値がリスト内に複数存在する場合に、deleteOne を指定すると、[リスト] の先頭に近い値が削除されることになります。

プロパティを削除しようとしても何の効果もありません。

filterlist の add メソッドまたは append メソッドを使用してフィルタを追加すると、複製が作成されてリストに追加されます。deleteOne、getPos、findPos、getOne などのメソッドは、複製値ではなく、リスト内の正確な値を使用します。

このような場合は、deleteOne コマンドを次のように使用できます。

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.deleteOne(f)
```

スクリプトの 3 行目は、フィルタ値の参照をリストに追加します。

パラメータ

value 必須。リストから削除する値。

例文

次のスクリプトでは、最初のステートメントで、[" 火曜日 ", " 水曜日 ", " 金曜日 "] というリストを作成し、次のステートメントで、作成したリストから " 水曜日 " を削除します。

```
--Lingo
days = ["Tuesday", "Wednesday", "Friday"]
days.deleteOne("Wednesday")
put days

// Javascript
days = list("Tuesday", "Wednesday", "Friday");
days.deleteOne("Wednesday");
trace(days);
```

put days ステートメントにより、結果がメッセージウィンドウに表示されます。

```
-- ["Tuesday", "Friday"].
```

deleteProp

使用法

```
list.deleteProp(item)
deleteProp list, item
```

解説

指定リストから指定項目を削除するコマンドです。

- リニアリストの場合、[項目] には、削除したい項目の位置の番号を指定します。リニアリストで使用する deleteProp コマンドは、deleteAt コマンドと同じ結果になります。番号がリストの項目数より大きい場合は、スクリプトエラーが発生します。
- プロパティリストの場合、[項目] には、削除したいプロパティ名を指定します。プロパティを削除すると、そのプロパティに対応している値も削除されます。プロパティリストに重複したプロパティがある場合には、最初のプロパティのみが削除されます。

パラメータ

item 必須。リストから削除する項目。

例文

次のステートメントは、spriteAttributes という名前のリスト [#height:100, #width: 200, #color: 34, #ink: 15] から color プロパティを削除します。

```
spriteAttributes.deleteProp(#color)
```

結果のリストは [#height:100, #width: 200, #ink: 15] になります。

関連事項

deleteAt

deleteShader

使用法

```
member(whichCastmember).deleteShader(whichShader)
member(whichCastmember).deleteShader(index)
```

解説

3D コマンド。キャストメンバからテクスチャを削除します。

パラメータ

shaderNameOrNum 必須。削除するシェーダの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次の例の 1 行目は、"StreetScene" というキャストメンバからシェーダ "Road" を削除します。2 行目は、"StreetScene" の 3 番目のシェーダを削除します。

```
-- Lingo
member("StreetScene").deleteShader("Road")
member("StreetScene").deleteShader(3)

// Javascript
member("StreetScene").deleteShader("Road");
member("StreetScene").deleteShader(3);
```

関連事項

shader, shaderList, newShader

deleteSoundObject

使用法

```
mixer.deleteSoundObject(soundObjRef)
mixer.deleteSoundObject(soundObjName)
```

解説

オーディオメソッド。指定のサウンドオブジェクトの参照または名前を使用してサウンドオブジェクトを削除します。

パラメータ

パラメータ	解説	必須 / オプション
SoundObjRef	サウンドオブジェクト参照。	必須
SoundObjName	指定のサウンドオブジェクトの名前。	必須

例文

```
-- Lingo syntax
on mouseUp me
     mixer1.deleteSoundObject(soundObjRef) --Deletes the sound object with the reference soundobjRef.
--OR
     mixer1.deleteSoundObject("SoundObj1") --Deletes the sound object with the name "soundobj1".

end

// JavaScript syntax
function mouseup()
{
mixer1.deleteSoundObject(soundObjRef) //Deletes the sound object with the reference soundobjRef.
//OR
mixer1.deleteSoundObject("SoundObj1"); //Deletes the sound object with the reference "soundobj1".

}
```

関連事項

ミクサ

deleteTexture

使用法

```
member(whichCastmember).deleteTexture(whichTexture)
member(whichCastmember).deleteTexture(index)
```

解説

キャストメンバからテクスチャを削除する 3D コマンドです。

パラメータ

textureNameOrNum 必須。削除するテクスチャの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次の例の 1 行目は、"PicnicScene" というキャストメンバから "Sky" というテクスチャを削除します。2 行目は、"PicnicScene" の 5 番目のテクスチャを削除します。

```
-- Lingo
member("PicnicScene).deleteTexture("Sky")
member("PicnicScene").deleteTexture(5)

// Javascript
member("PicnicScene).deleteTexture("Sky");
member("PicnicScene").deleteTexture(5)
```

関連事項
newTexture, texture

deleteVertex()

使用法

```
-- Lingo syntax
memberObjRef.deleteVertex(indexToRemove)

// JavaScript syntax
memberObjRef.deleteVertex(indexToRemove);
```

解説
インデックスの指定位置にあるベクトルシェイプキャストメンバである既存の頂点を削除するベクトルシェイプコマンドです。

パラメータ
indexToRemove 必須。削除する頂点のインデックス位置を指定する整数。

例文
次の行は、ベクトルシェイプ "Archie" にある 2 番目の頂点を削除します。

```
-- Lingo syntax
member("Archie").deleteVertex(2)

// JavaScript syntax
member("Archie").deleteVertex(2);
```

関連事項
addVertex(), moveVertex(), originMode, vertexList

displayOpen()

使用法

```
-- Lingo syntax
fileioObjRef.displayOpen()

// JavaScript syntax
fileioObjRef.displayOpen();
```

解説
開くダイアログボックスを表示する Fileio メソッドです。

このメソッドは、選択されたファイルの完全なパスと名前をスクリプトに返します。

パラメータ
なし

関連事項
Fileio

displaySave()

使用法

```
-- Lingo syntax
fileioObjRef.displaySave(stringTitle, stringFileName)

// JavaScript syntax
fileioObjRef.displaySave(stringTitle, stringFileName);
```

解説
保存ダイアログボックスを表示する Fileio メソッドです。

このメソッドは、保存されたファイルの完全なパスと名前をスクリプトに返します。

パラメータ
stringTitle 必須。保存ダイアログボックスに表示するタイトルを指定するストリング。

stringFileName 必須。保存するファイルの完全なパスと名前を指定するストリング。

関連事項
Fileio

do

使用法

```
do stringExpression
```

解説
ストリングを評価し、その結果をスクリプトのステートメントとして実行するコマンドです。ユーザが入力した式を評価するときや、ストリング変数、フィールド、リスト、ファイル内に格納されているコマンドを実行するときに使います。

do コマンド内で初期化されていないローカル変数を使うとコンパイラエラーが発生します。ローカル変数は事前に初期化しておく必要があります。

注意：このコマンドではグローバル変数を定義することはできないので、事前に定義する必要があります。

do コマンドは複数の行からなるストリングにも 1 行のストリングにも使うことができます。

パラメータ
stringExpression 必須。評価するストリング。

例文
次のコマンドは引用符で囲まれている部分のステートメントを実行します。2 行目は、commandList リストの 3 番目のコマンドを実行します。

```
do "beep 2"
do commandList[3]
```

doneParsing()

使用法

```
parserObject.doneParsing()
```

解説

関数。parseURL() を使用してパーサーがドキュメントの構文解析を完了したときに、1 (TRUE) を返します。構文解析が完了するまで、戻り値は 0 (FALSE) です。

パラメータ

なし

関連事項

parseURL()

dot()

使用法

```
vector1.dot(vector2)
```

解説

3D ベクトルメソッド。2 つのベクトルの x、y、および z コンポーネントの積の和を返します。ベクトルが両方とも正規化されている場合、dot は 2 つのベクトル間の角度のコサインになります。

2 つのベクトルのドットを手動で計算するには、vector1 の x コンポーネントと vector2 の x コンポーネントの積、vector1 の y コンポーネントと vector2 の y コンポーネントの積、および vector1 の z コンポーネントと vector2 の z コンポーネントの積をそれぞれ計算し、最後に 3 つの積の和を計算します。

このメソッドは dotProduct() 関数と同じです。

パラメータ

vector2 必須。合計を返す 2 番目のベクトル。

例文

次の例で、ベクトル pos5 と pos6 の角度は 45 度です。getNormalized 関数は、正規化された pos5 および pos6 の値を返し、それぞれ変数 norm1 および norm2 に格納します。norm1 および norm2 の dot は 0.7071 です。これは 45 度のコサインです。

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dot(norm2)
-- 0.7071
```

関連事項
dotProduct(), getNormalized, normalize

dotProduct()

使用法
```
vector1.dotProduct(vector2)
```

解説
3D ベクトルメソッド。2 つのベクトルの x、y、および z コンポーネントの積の和を返します。ベクトルが両方とも正規化されている場合、dotproduct は 2 つのベクトル間の角度のコサインになります。

2 つのベクトルのドットを手動で計算するには、vector1 の x コンポーネントと vector2 の x コンポーネントの積、vector1 の y コンポーネントと vector2 の y コンポーネントの積、および vector1 の z コンポーネントと vector2 の z コンポーネントの積をそれぞれ計算し、最後に 3 つの積の和を計算します。

このメソッドは dot() 関数と同じです。

パラメータ
vector2 必須。合計を返す 2 番目のベクトル。

例文
次の例では、ベクトル pos5 と pos6 の角度は 45 度です。getNormalized 関数は、正規化された pos5 および pos6 の値を返し、それぞれ変数 norm1 および norm2 に格納します。norm1 および norm2 の dotProduct は 0.7071 です。これは 45 度のコサインです。

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dotProduct(norm2)
-- 0.7071
```

関連事項
dot(), getNormalized, normalize

downloadNetThing

使用法
```
downloadNetThing URL, localFile
```

解説
このコマンドは、ムービーの再生中にインターネットからローカルディスクにファイルをダウンロードします。ダウンロードの終了は、netDone を使って確認します。

オーサリング環境やプロジェクタで再生している Director ムービーは、downLoadNetThing コマンドをサポートしますが、このコマンドは Shockwave Player では使用できません。これは、ユーザのローカルディスクにインターネットから意図せずにファイルがコピーされるのを防ぐためです。

一度に複数のネットワーク操作をアクティブにすることができますが、4 つ以上の操作を同時に行うと操作処理速度が著しく低下します。

ムービーのキャッシュサイズ設定や、[Check Documents] オプションは、downloadNetThing コマンドの動作には影響しません。

パラメータ

URL 必須。ダウンロード可能なオブジェクトの URL です。例えば、FTP や HTTP サーバ、HTML ページ、外部キャストメンバ、Director ムービー、グラフィックなどがあります。

localFile 必須。ローカルディスクのファイルのパス名とファイル名です。

例文

次のステートメントは、URL から外部キャストメンバ "Thousands.cst" をダウンロードして Director と同じフォルダに入れ、そのファイルを外部キャストメンバ " キャスト集 " として設定します。

```
downLoadNetThing("http://www.cbDeMille.com/Thousands.cst", the\applicationPath&"Thousands.cst")
castLib("Cast of Thousands").fileName = the applicationPath&"Thousands.cst"
```

関連事項

importFileInto(), netDone(), preloadNetThing()

draw()

使用法

```
-- Lingo syntax
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList)
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.draw(rect, colorObjOrParamList)

// JavaScript syntax
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList);
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.draw(rect, colorObjOrParamList);
```

解説

画像メソッド。指定された画像オブジェクトの矩形領域内に、指定された色で、線または塗りつぶされていないシェイプを描画します。

このメソッドは、エラーがなければ値 1 を返します。

オプションのパラメータリストを省略すると、draw() は、指定された 1 番目と 2 番目の点の間、または指定された矩形の左上隅と右下隅の間に、1 ピクセルの線を描画します。

8 ビット以下の画像で、インデックス化されたカラー値を含むカラーオブジェクトを使用すると、最高のパフォーマンスが得られます。16 ビット画像および 32 ビット画像では、RGB カラー値を使います。

領域を塗りつぶすには、fill() メソッドを使用します。

パラメータ

x1x 座標と y 座標を使用して線を描画する場合は必須。線の始点の x 座標を指定する整数。

y1x 座標と y 座標を使用して線を描画する場合は必須。線の始点の y 座標を指定する整数。

x2x 座標と y 座標を使用して線を描画する場合は必須。線の終点の x 座標を指定する整数。

y2x 座標と y 座標を使用して線を描画する場合は必須。線の終点の y 座標を指定する整数。

colorObjOrParamList 必須。線またはシェイプの境界の色を指定するカラーオブジェクトまたはパラメータリスト。単純なカラーオブジェクトの代わりにパラメータリストを使用して、次のプロパティを指定できます。

プロパティ	解説
#shapeType	#oval、#rect、#roundRect、または のシンボル値。#line デフォルトは #line です。
#lineSize	シェイプを描く際のライン幅です。
#color	シェイプの境界の色を決定するカラーオブジェクトです。

point**(x, y),** point**(x, y)** 点を使用して線を描画する場合は必須。線の始点と終点を指定する 2 つの点。

rect シェイプを描画する場合は必須。シェイプを描画する矩形領域を指定する矩形。

例文

次のステートメントは、ステージの画像内の点 (20, 20) から点 (20, 60) に、1 ピクセルで深紅の線を描画します。

```
-- Lingo
objImage = _movie.stage.image
objImage.draw(point(20, 20), point(20, 60), rgb(255, 0 ,0))

// Javascript
var objImage = _movie.stage.image ;
objImage.draw(point(20, 20), point(20, 60), color(255, 0 ,0)) ;
```

関連事項

color(), copyPixels(), fill(), image(), setPixel()

duplicate() （画像）

使用法

```
-- Lingo syntax
imageObjRef.duplicate()

// JavaScript syntax
imageObjRef.duplicate();
```

解説

画像メソッド。指定された画像のコピーを作成して返します。

新しい画像はオリジナルのものとは完全に独立で、依存関係はなく、キャストメンバにはまったくリンクされていません。画像に多くの変更を加えたい場合には、キャストメンバと無関係のコピーを作成する方が良いでしょう。

パラメータ

なし

例文

次のステートメントは、キャストメンバ "Lunar Surface" の画像から新しい画像オブジェクトを作成し、変数 workingImage に代入します。

```
workingImage = member("Lunar Surface").image.duplicate()
```

関連事項

image()

duplicate() （リスト関数）

使用法

```
(oldList).duplicate()
duplicate(oldList)
```

解説

ネストされたリスト (項目自体がリストであるようなリスト) であっても、通常のリストであっても、リストの複製を作成し返す関数です。この関数はリストの現在のデータを保存するときに使用します。

注意：この関数を使用して filterlist() を複製することはできません。

リストを変数から変数に受け渡すとき、変数にはリストの実体ではなく、リストへのポインタが渡されます。このため、コピーされたリストのデータを変更すると、元のリストのデータも変更されます。

完成したムービーの duplicate() (list function) の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Vector Shapes ムービーを参照してください。

パラメータ

oldList 必須。複製するリストを指定します。

例文

次のステートメントは、CustomersToday というリストのコピーを作成し、変数 CustomerRecord に割り当てます。

```
-- Lingo
CustomersToday = [1, 3, 5]
CustomerRecord = CustomersToday.duplicate()

// Javascript

CustomersToday = list(1, 3, 5);
CustomerRecord = CustomersToday.duplicate();
```

関連事項

image()

duplicate() （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.duplicate({intPosn})

// JavaScript syntax
memberObjRef.duplicate({intPosn});
```

解説

指定されたキャストメンバのコピーを作成するメンバコマンドです。

このメソッドは、ランタイムよりもオーサリング環境での使用に適しています。キャストメンバの複製はメモリ上で処理されるため、メモリの問題が発生することがあります。

キャストメンバの変更をファイルに永続的に保存するには、このメソッドを使用します。

パラメータ

intPosn オプション。複製したキャストメンバのキャストウィンドウ内での位置を指定する整数。省略すると、複製されたキャストメンバはキャストウィンドウ内で最初に空いている位置に配置されます。

例文

次のステートメントは、キャストメンバ " 机 " のコピーを作成し、キャストウィンドウ内で最初に空いている位置に挿入します。

```
-- Lingo syntax
member("Desk").duplicate()

// JavaScript syntax
member("Desk").duplicate();
```

次のステートメントは、キャストメンバ " 机 " のコピーを作成し、キャストウィンドウの 125 番に挿入します。

```
-- Lingo syntax
member("Desk").duplicate(125)

// JavaScript syntax
member("Desk").duplicate(125);
```

関連事項

メンバ

duplicateFrame()

使用法

```
-- Lingo syntax
_movie.duplicateFrame()

// JavaScript syntax
_movie.duplicateFrame();
```

解説

現在のフレームとその内容を複製するムービーメソッドです。現在のフレームの後に複製したフレームを挿入し、複製したフレームを現在のフレームにします。このメソッドは、スコア生成時のみに使用できます。

このメソッドは、insertFrame() メソッドと同じ機能を実行します。

パラメータ

なし

例文

次のハンドラで使用すると、duplicateFrame コマンドは、スプライトチャネル 20 に割り当てられた外部キャスト "Toys" のキャストメンバ "Ball" を含む一連のフレームを作成します。フレームの数はパラメータ numberOfFrames によって決まります。

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    sprite(20).member = member("Ball", "Toys")
    repeat with i = 0 to numberOfFrames
        _movie.duplicateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    sprite(20).member = member("Ball", "Toys");
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.duplicateFrame();
    }
    _movie.endRecording();
}
```

関連事項

insertFrame(), ムービー

enableHotSpot()

使用法

```
-- Lingo syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse)

// JavaScript syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse);
```

解説

QTVR（QuickTime® VR）スプライト内のホットスポットを有効にするか（TRUE）無効にするか（FALSE）を指定する QTVR コマンドです。

パラメータ

hotSpotID 必須。検査する QTVR スプライト内のホットスポットを指定します。

trueOrFalse 必須。QTVR スプライトを有効にするかどうかを指定する TRUE または FALSE の値。

enableSoundTrack(trackNum)

使用法

```
trackInfo = member(1).trackInfo
member(1).enableSoundTrack(trackNumber)
```

解説

メンバメソッド。複数のサウンドトラックがあるビデオで再生するサウンドトラックのトラック番号を指定します。デフォルトでは、ビデオはファイル内の最初のトラックを再生します。一度に 1 つのトラックのみを有効にすることができます。

注意：FLV メンバには 1 つのサウンドトラックしかないため、このプロパティは FLV メンバには適用されません。

パラメータ

パラメータ	解説	必須 / オプション
trackNum	ビデオで再生するサウンドトラックを指定します。	必須

例文

```
-- Lingo syntax
trackInfo = member(1).trackInfo
member(1).enableSoundTrack(2)

// JavaScript syntax
trackInfo = member(1).trackInfo ;
    member(1).enableSoundTrack(2) ;
```

endRecording()

使用法

```
-- Lingo syntax
_movie.endRecording()

// JavaScript syntax
_movie.endRecording();
```

解説

スコア更新セッションを終了するムービーメソッドです。

endRecording() を呼び出した後は、スクリプトを使用して、スコアチャネルの制御を再開できます。

パラメータ

なし

例文

次のハンドラは、endRecording キーワードを使用してスコア生成セッションを終了します。

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

関連事項

beginRecording(), ムービー , updateFrame()

erase()

使用法

```
-- Lingo syntax
memberObjRef.erase()

// JavaScript syntax
memberObjRef.erase();
```

解説

指定されたキャストメンバを削除するメンバメソッドです。そのメンバが使用していたキャストウィンドウ内のスロットは空になります。

このメソッドは、オーサリング環境で使用するのが最も効果的です。プロジェクタでこのメソッドを使用すると、メモリの問題が発生する可能性があります。

パラメータ
なし

例文
次のステートメントは、" ハードウェア " キャスト内のキャストメンバ " ギア " を削除します。

```
-- Lingo syntax
member("Gear", "Hardware").erase()

// JavaScript syntax
member("Gear", "Hardware").erase();
```

次のハンドラは、start から finish までの番号のキャストメンバを削除します。

```
-- Lingo syntax
on deleteMember start, finish
    repeat with i = start to finish
        member(i).erase()
    end repeat
end deleteMember

// JavaScript syntax
function deleteMember(start, finish) {
    for (var i=start; i<=finish; i++) {
        member(i).erase();
    }
}
```

関連事項
メンバ, new()

error()

使用法

```
-- Lingo syntax
fileioObjRef.error(intError)

// JavaScript syntax
fileioObjRef.error(intError);
```

解説
指定されたエラーメッセージを返す Fileio メソッドです。

パラメータ
intError 必須。エラーを指定する整数です。有効な値は、0 ("OK") または 1 (" メモリ割り当て失敗 ") です。他のすべての値は " 不明エラー " になります。

例文
次の例は、c:\check.txt という名前のファイルを作成します。ファイルの作成中にエラーが発生すると、それを error に渡します。

```
-- Lingo syntax
objFileio = new xtra("fileio")
isok = objFileio.createFile("c:\check.txt")
if ( isok = 0 ) then
    alert( " file creation successful")
else
    alert( " file creation failed " & objFileio.error(isok))
end if

// JavaScript syntax
var objFileio = new xtra("fileio");
isok = objFileio.createFile("c:\check.txt");
if ( isok == 0 )
{
    _player.alert( " file creation successful");
}
else
{
    _player.alert( " file creation failed " + objFileio.error(isok));
}
```

関連事項

Fileio

externalEvent()

使用法

```
externalEvent "string"
```

解説

ブラウザがスクリプト言語として認識できる [ストリング] を送るためのコマンドです。このコマンドを利用してムービー再生やブラウザと、それが埋め込まれている HTML ページとをコミュニケートさせることができます。

このコマンドはブラウザ内のムービーのみに有効です。

注意：externalEvent コマンドは値を返さないため、ブラウザがイベントを認識したかどうかを確認できる直接的な方法はありません。ムービーにメッセージを返すには、ブラウザ内で on EvalScript を使用します。

パラメータ

string 必須。ブラウザに送信するストリングです。このストリングは、そのブラウザが対応しているスクリプト言語で記述されている必要があります。

例文

次のステートメントは、LiveConnect スクリプト環境で externalEvent を使用します。

LiveConnect は externalEvent で渡されたストリングを関数の呼び出しとして認識します。JavaScript の作者は、この関数を HTML ヘッダ内で定義し名前を付ける必要があります。ムービーでは関数の名前とパラメータが externalEvent のストリングとして定義されます。パラメータは、個別のストリングとしてブラウザに認識される必要があるため、それぞれのパラメータを単一引用符で囲みます。

HTML 内のステートメント

```
function MyFunction(parm1, parm2) {
    //script here
}
```

ムービー内のスクリプトに記述されたステートメント

```
externalEvent ("MyFunction('parm1','parm2')")
```

次のステートメントは、Windows® の Internet Explorer® で使われる ActiveX スクリプト環境で externalEvent を使用する例です。ActiveX は externalEvent をイベントとして処理し、このイベントとストリングのパラメータを、ボタンオブジェクト内の onClick イベントと同様に処理します。

HTML 内のステートメント

```
Sub
NameOfShockwaveInstance_externalEvent(aParam)
    'script here
End Sub
```

関数を定義する代わりに、イベントのスクリプトを定義することもできます。

```
<SCRIPT FOR="NameOfShockwaveInstance"
EVENT="externalEvent(aParam)"
LANGUAGE="VBScript">
    'script here
</SCRIPT>
```

ムービー内では、関数名とすべてのパラメータをストリングとして externalEvent に渡します。

```
externalEvent ("MyFunction ('parm1','parm2')")
```

関連事項

on EvalScript

extrude3D

使用法

```
member(whichTextCastmember).extrude3D(member(which3dCastmember))
```

解説

テキストキャストメンバのテキストから 3D キャストメンバの新しい #extruder モデルリソースを作成する 3D コマンドです。

これは、テキストキャストメンバの 3D displayMode プロパティを使用するのとは異なる操作です。

extrude3D を使用してのモデルの作成

1 3D キャストメンバに新しい #extruder モデルリソースを作成します。

```
textResource = member("textMember").extrude3D(member("3DMember"))
```

2 手順 1 で作成したモデルリソースを使用して新しいモデルを作成します。

```
member("3DMember").newModel("myText", textResource)
```

パラメータ

which3dCastmember 必須。新しい #extruder モデルリソースを作成するキャストメンバ。

例文

次の例で " ロゴ " はテキストキャストメンバ、" シーン " は 3D キャストメンバです。1 行目は、" シーン " にモデルリソースを作成します。これは、" ロゴ " のテキストの 3D バージョンです。2 行目は、このモデルリソースを使用して "3dLogo" というモデルを作成します。

```
-- Lingo
myTextModelResource =member("Logo").extrude3d(member("Scene"))
member("Scene").newModel("3dLogo", myTextModelResource)

// Javascript
myTextModelResource =member("Logo").extrude3d(member("Scene"));
member("Scene").newModel("3dLogo", myTextModelResource);
```

関連事項

bevelDepth, bevelType, displayFace, smoothness, tunnelDepth, displayMode

externalParamName()

使用法

```
-- Lingo syntax
_player.externalParamName(paramNameOrNum)

// JavaScript syntax
_player.externalParamName(paramNameOrNum);
```

解説

HTML の <EMBED> タグまたは <OBJECT> タグから、外部パラメータのリストで指定されているパラメータ名を返すプレーヤメソッドです。

名前でパラメータを指定すると、このメソッドは **paramNameOrNum** と一致するパラメータ名を返します。大文字と小文字は区別されません。一致するパラメータ名が見つからない場合は、VOID (Lingo) または null (JavaScript シンタックス) を返します。

番号でパラメータを指定すると、このメソッドはパラメータリストで **paramNameOrNum** の位置にあるパラメータ名を返します。一致するパラメータ位置が見つからない場合は、VOID または null を返します。

このメソッドは、ブラウザ内で実行している Shockwave コンテンツのムービーのみに有効です。Director またはプロジェクタで再生するムービーには適用できません。

次のリストでは、使用できる定義済みの外部パラメータについて説明します。

パラメータ	定義
swAudio	ムービーで再生する Shockwave Audio ファイルの場所を指定するストリングです。値は完全修飾された URL です。
swBackColor	ムービーの Stage カラープロパティを変更するためのカラー値です。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swBanner	ムービーでバナーとして使用するテキストを指定するストリングです。
swColor	特定のオブジェクトの色を変更するために使用するカラー値です。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swForeColor	新しいフォアカラーの値です。フィールドキャストメンバに書き込まれるテキストは、現在アクティブなフォアカラーでレンダリングされます。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swFrame	ムービーの特定のフレームに割り当てられた名前を示すストリング値です。
swList	スクリプトで解析できる、項目のカンマ区切りリストです。リストの値に使用できるのは、キー / 値ペア、ブール項目、整数、ストリングです。
swName	ムービーで表示または使用する、ユーザ名などの名前です。

パラメータ	定義
swPassword	ムービーで使用するパスワードです。通常は、swName プロパティと組み合わせて使用します。
swPreloadTime	サウンドの再生が始まる前にオーディオファイルサウンドをプリロードする必要のある秒数を指定する整数値です。Shockwave Audio と共に使用すると、再生が開始する前に既にダウンロードされているオーディオの量が増えるので、再生のパフォーマンスが向上します。
swSound	Director ムービーで再生するサウンドの名前、またはそもそもサウンドを再生するかどうかを指定するストリング値です。
swText	ムービーで使用するテキストを指定するストリング値です。
swURL	Shockwave コンテンツまたは Shockwave Audio ファイルを含む別のムービーの場所を指定するストリングの URL です。
swVolume	ムービーからのサウンド出力のボリュームレベルを制御する整数値 (0 ～ 10 を推奨) です。0 はオフ (無音)、10 は最大ボリュームです。
sw1 ～ sw9	作成者がパラメータを定義するための 9 個の追加プロパティです。

パラメータ

paramNameOrNum 必須。返すパラメータの名前を指定するストリング。または、返すパラメータ名のインデックス位置を指定する整数。

例文

次のステートメントは、指定した外部パラメータ "swURLString" の値を変数 myVariable に割り当てます。

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

関連事項

externalParamValue(), ムービー

externalParamValue()

使用法

```
-- Lingo syntax
_player.externalParamValue(paramNameOrNum)

// JavaScript syntax
_player.externalParamValue(paramNameOrNum);
```

解説

HTML の <EMBED> タグまたは <OBJECT> タグから、外部パラメータのリストで指定されているパラメータの値を返します。

名前でパラメータ値を指定すると、このメソッドは名前が **paramNameOrNum** と一致する最初のパラメータの値を返します。大文字と小文字は区別されません。一致するパラメータ値が見つからない場合は、VOID (Lingo) または null (JavaScript シンタックス) を返します。

インデックスでパラメータ値を指定すると、このメソッドはパラメータリストで **paramNameOrNum** の位置にあるパラメータの値を返します。一致するパラメータ位置が見つからない場合は、VOID または null を返します。

このメソッドは、ブラウザ内で実行している Shockwave コンテンツのムービーのみに有効です。Director またはプロジェクタで再生するムービーには適用できません。

次のリストでは、使用できる定義済みの外部パラメータについて説明します。

パラメータ	定義
swAudio	ムービーで再生する Shockwave Audio ファイルの場所を指定するストリングです。値は完全修飾された URL です。
swBackColor	ムービーの Stage カラープロパティを変更するためのカラー値です。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swBanner	ムービーでバナーとして使用するテキストを指定するストリングです。
swColor	特定のオブジェクトの色を変更するために使用するカラー値です。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swForeColor	新しいフォアカラーの値です。フィールドキャストメンバに書き込まれるテキストは、現在アクティブなフォアカラーでレンダリングされます。値は、0 ～ 255 の範囲の任意の整数です。8 ビットカラーのムービーには 0 ～ 255 を使用し、4 ビットカラーのムービーには 0 ～ 15 を使用します。
swFrame	ムービーの特定のフレームに割り当てられた名前を示すストリング値です。
swList	スクリプトで解析できる、項目のカンマ区切りリストです。リストの値に使用できるのは、キー / 値ペア、ブール項目、整数、ストリングです。
swName	ムービーで表示または使用する、ユーザ名などの名前です。
swPassword	ムービーで使用するパスワードです。通常は、swName プロパティと組み合わせて使用します。
swPreloadTime	サウンドの再生が始まる前にオーディオファイルサウンドをプリロードする必要のある秒数を指定する整数値です。Shockwave Audio と共に使用すると、再生が開始する前に既にダウンロードされているオーディオの量が増えるので、再生のパフォーマンスが向上します。
swSound	Director ムービーで再生するサウンドの名前、またはそもそもサウンドを再生するかどうかを指定するストリング値です。
swText	ムービーで使用するテキストを指定するストリング値です。
swURL	別の Shockwave ムービーまたは Shockwave Audio ファイルの場所を指定するストリングの URL です。
swVolume	ムービーからのサウンド出力のボリュームレベルを制御する整数値 (0 ～ 10 を推奨) です。0 はオフ (無音)、10 は最大ボリュームです。
sw1 ～ sw9	作成者がパラメータを定義するための 9 個の追加プロパティです。

パラメータ

paramNameOrNum 必須。返すパラメータ値の名前を指定するストリング。または、返すパラメータ値のインデックス位置を指定する整数。

例文

次のステートメントは、指定した外部パラメータ "swURLString" の値を変数 myVariable に割り当てます。

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

関連事項

externalParamName(), ムービー

extractAlpha()

使用法

```
imageObject.extractAlpha()
```

解説

指定された 32 ビット画像からアルファチャネルをコピーして、新しい画像オブジェクトとして返す関数です。結果はアルファチャネルを表す 8 ビットグレースケールの画像です。

32 ビット画像をアルファチャネルでダウンサンプリングするときに便利な関数です。

例文

次のステートメントは、メンバ 1 のアルファチャネルを変数 mainAlpha に設定します。

```
-- Lingo
mainAlpha = member(1).image.extractAlpha()

// Javascript
mainAlpha = member(1).image.extractAlpha();
```

関連事項

setAlpha()

fadeIn()

使用法

```
-- Lingo syntax
soundChannelObjRef.fadeIn({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeIn({intMilliseconds});
```

解説

サウンドチャネルの volume を直ちにゼロに設定した後、一定のミリ秒数の後で元の音量に戻す、サウンドチャネルメソッドです。

フェードしている間、パンの設定は現行のまま保持されます。

パラメータ

intMilliseconds オプション。音量を下げてから元の値に戻すまでのミリ秒数を指定する整数。値が指定されない場合のデフォルト値は 1000 ミリ秒 (1 秒) です。

例文

次の Lingo は、キャストメンバ introMusic2 の開始から 3 秒かけてサウンドチャネル 3 をフェードインします。

```
-- Lingo syntax
sound(3).play(member("introMusic2"))
sound(3).fadeIn(3000)

// JavaScript syntax
sound(3).play(member("introMusic2"));
sound(3).fadeIn(3000);
```

関連事項

fadeOut(), fadeTo(), pan, サウンドチャネル , volume (Windows Media)

fadeOut()

使用法

```
-- Lingo syntax
soundChannelObjRef.fadeOut({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeOut({intMilliseconds});
```

解説

サウンドチャネルの volume を徐々に小さくし、指定されたミリ秒数の後に 0 にする、サウンドチャネルメソッドです。

フェードしている間、パンの設定は現行のまま保持されます。

パラメータ

intMilliseconds オプション。音量を 0 まで下げる期間のミリ秒数を指定する整数。値が指定されない場合のデフォルト値は 1000 ミリ秒 (1 秒) です。

例文

次のステートメントは、サウンドチャネル 3 を 5 秒かけてフェードアウトします。

```
-- Lingo syntax
sound(3).fadeOut(5000)

// JavaScript syntax
sound(3).fadeOut(5000);
```

関連事項

fadeIn(), fadeTo(), pan, サウンドチャネル , volume (Windows Media)

fadeTo()

使用法

```
-- Lingo syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds});
```

解説

サウンドチャネルの volume を徐々に変更し、指定されたミリ秒数の後に指定された値にする、サウンドチャネルメソッドです。

フェードしている間、パンの設定は現行のまま保持されます。

完成したムービーでの fadeTo() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Sound Control ムービーを参照してください。

パラメータ

intVolume 必須。変更後の音量を指定する整数。指定できる値の範囲は 0 ～ 255 です。

intMilliseconds オプション。音量を **intVolume** まで変化させる期間のミリ秒数を指定する整数。値が指定されない場合のデフォルト値は 1000 ミリ秒（1 秒）です。

例文

次のステートメントは、サウンドチャネル 4 の音量を 2 秒後に 150 になるように変更します。フェード開始時点でのサウンドチャネル 4 の元の音量によって、フェードアップまたはフェードダウンとなります。

```
-- Lingo syntax
sound(4).fadeTo(150, 2000)

// JavaScript syntax
sound(4).fadeTo(150, 2000);
```

関連事項

fadeIn(), fadeOut(), pan, サウンドチャネル , volume (Windows Media)

fileName()

使用法

```
-- Lingo syntax
fileioObjRef.fileName()

// JavaScript syntax
fileioObjRef.fileName();
```

解説

開いているファイルの完全なパスと名前を返す Fileio メソッドです。

先に openFile() を呼び出してファイルを開いた後、fileName() を使用してファイル名を取得する必要があります。

パラメータ
なし

例文
次のステートメントは、ファイルを作成し、そのファイルの場所を出力します。

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile(_player.ApplicationPath)
put objFileio.fileName()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile(_player.ApplicationPath);
trace(objFileio.fileName());
```

関連事項
Fileio , openFile()

fileOpen()

使用法

```
FileOpen(MUIObject, string)
```

解説
この関数は、MUI Xtra のインスタンスによって提供される、ファイルを開く標準ダイアログボックスを表示します。

2 番目のパラメータは、ダイアログボックスが開いたときに編集可能なフィールドに表示されるストリングを指定します。ユーザは編集可能フィールドにファイル名を入力して、開くファイルを指定できます。ユーザがボタンをクリックすると、テキストが返されます。

- ユーザが「キャンセル」をクリックした場合、返されるテキストは渡された値と同じです。
- ユーザが「OK」をクリックした場合、返されるテキストはプラットフォーム固有のパスです。

パラメータ
なし

例文
次のステートメントは、ファイルを開く標準ダイアログボックスを作成して表示します。

- 最初のステートメントは、ダイアログボックスとして使用されるオブジェクトである MUI Xtra のインスタンスを作成します。
- 2 番目のステートメントは、変数 fileString にストリングを割り当てます。割り当てたストリングは後から FileOpen コマンドの 2 番目のパラメータとして使用されます。
- 3 番目のステートメントは、FileOpen コマンドを使用してファイルを開くダイアログボックスを生成します。
- 最後のステートメントは、FileOpen コマンドで送信された元のストリングが、ユーザがボタンをクリックしたときに返されたストリングと同じかどうかをチェックします。値が異なる場合、ユーザが開くファイルを選択したことを示します。

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")
set fileString = "Open this file"
set result = fileOpen(aMuiObj, fileString)
-- Check to see if the dialog was canceled
if (result <> fileString) then
    -- The dialog wasn't canceled, do something with the new path data.
else
    put "ERROR - fileOpen requires a valid aMuiObj"
end if
```

関連事項

Fileio , openFile()

fileSave()

使用法

```
FileSave( MUIObject, string, message )
```

解説

この関数は、現行ファイルを保存する標準のファイル保存ダイアログボックスを表示します。ダイアログボックスは MUI Xtra のインスタンスから作成されます。

- string パラメータは、ダイアログボックスのファイル名フィールドに表示されるストリングを指定します。ユーザはこのフィールドを使用してファイルに付ける新しいファイル名を入力できます。ユーザがボタンをクリックすると、Lingo ではフィールドの内容を含むストリングの値が返されます。ユーザが「キャンセル」をクリックした場合、返されるストリングは元のストリングと同じです。
- message パラメータは、ダイアログボックスの編集可能フィールドの上に表示されるストリングです。

パラメータ

なし

例文

次のステートメントは、ファイル保存ダイアログボックスを作成して表示します。

- 最初のステートメントは、ダイアログボックスとして使用されるオブジェクトである MUI Xtra のインスタンスを作成します。
- 2 番目のステートメントは、変数 fileString にストリングを割り当てます。割り当てたストリングは後から FileSave コマンドの 2 番目のパラメータとして使用されます。
- 3 番目のステートメントは、FileSave コマンドを使用してファイル保存ダイアログボックスを生成します。
- 最後のステートメントは、ユーザがボタンをクリックした後の結果が、ダイアログボックスが開かれたときに送信されたストリングと同じかどうかをチェックします。異なる場合は、ユーザが「キャンセル」以外の何かをクリックしたことを示します。

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")
set fileString = "Save this file"
set result = fileSave(aMuiObj, fileString, "with this prompt")
-- Check to see if the dialog was canceled
if (result <> fileString) then
    -- The dialog wasn't canceled, do something with the new path data.
end if
```

fill()

使用法

```
-- Lingo syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList)
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.fill(rect, colorObjOrParamList)

// JavaScript syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList);
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.fill(rect, colorObjOrParamList);
```

解説

画像メソッド。指定された画像オブジェクトの中の指定された色で、矩形領域を塗りつぶします。

エラーがなければ 1 を返し、エラーがあれば 0 を返します。

8 ビット以下の画像で、インデックス化されたカラー値を含むカラーオブジェクトを使用すると、最高のパフォーマンスが得られます。16 ビットおよび 32 ビットの画像では、RGB カラー値を使用します。

パラメータ

left 座標で指定した領域を塗りつぶす場合は必須。塗りつぶす領域の左端を指定する整数。

top 座標で指定した領域を塗りつぶす場合は必須。塗りつぶす領域の上端を指定する整数。

right 座標で指定した領域を塗りつぶす場合は必須。塗りつぶす領域の右端を指定する整数。

bottom 座標で指定した領域を塗りつぶす場合は必須。塗りつぶす領域の下端を指定する整数。

colorObjOrParamList 必須。領域の塗りつぶしに使用する色を指定するカラーオブジェクトまたはパラメータリスト。単純なカラーオブジェクトの代わりにパラメータリストを使用して、次のプロパティを指定できます。

プロパティ	解説
#shapeType	#oval、#rect、#roundRect、または のシンボル値。#line デフォルトは #line です。
#lineSize	シェイプを描く際のライン幅です。
#color	領域を塗りつぶす色を指定するカラーオブジェクトです。
#bgColor	領域の境界の色を決定するカラーオブジェクトです。

point**(x, y),** point**(x, y)** 点を使用して領域を塗りつぶす場合は必須。塗りつぶす領域の左上と右下の隅を指定する 2 つの点。基準は、指定された画像オブジェクトの左上隅です。

rect 矩形を使用して領域を塗りつぶす場合は必須。塗りつぶす矩形領域を指定する矩形。

例文

次のステートメントは、左上隅と右下隅の 2 つの点で指定した領域を塗りつぶします。座標の基準は、指定した画像オブジェクトの左上隅です。

```
-- Lingo
objImage = _movie.stage.image
objImage.fill(point(20, 20), point(30, 60), rgb(255, 0 ,0))

// Javascript
var objImage = _movie.stage.image ;
objImage.fill(point(20, 20), point(30, 60), color(255, 0 ,0)) ;
```

関連事項

color(), draw(), image()

filter()

使用法

```
filter(filter symbol)
```

解説

ビットマップフィルタメソッド。ビットマップフィルタを作成してスプライトに適用するために使用されます。filter() 関数は次のシンボルを取ります。

#blurfilter ぼかしフィルタを作成します

#glowfilter グローフィルタを作成します

#bevelfilter ベベルフィルタを作成します

#dropshadowfilter 影フィルタを作成します

#adjustcolorfilter カラー調整フィルタを作成します

#gradientglowfilter グラデーショングローフィルタを作成します

#gradientbevelfilter グラデーションベベルフィルタを作成します

#convolutionmatrixfilter 畳み込みマトリックスフィルタフィルタを作成します

#displacementmapfilter 変位マップフィルタを作成します

例文

最初のステートメントは、myFilter という変数に Blur フィルタを設定します。次の行は、ぼかしフィルタを sprite(1) に設定します。

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)
// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

関連事項

『Director ユーザガイド』の「ビットマップフィルタ」。

findLabel()

使用法

```
-- Lingo syntax
spriteObjRef.findLabel(whichLabelName)

// JavaScript syntax
spriteObjRef.findLabel(whichLabelName);
```

解説

指定のラベル名に該当する Flash ムービー内のフレーム番号を返す関数です。

ラベルが存在しない場合や、Flash ムービーのその部分がまだストリームしていない場合は 0 が返されます。

パラメータ

whichLabelName 必須。検索するフレームラベルを指定します。

例文

次の例は、ラベル名 [GetMe] に関連付けられたフレーム番号 (member(1) の Flash ムービー内) を返します。

```
-- Lingo syntax
sprite(1).findLabel("GetMe")

// JavaScript syntax
sprite(1).findLabel("GetMe");
```

findEmpty()

使用法

```
-- Lingo syntax
castObjRef.findEmpty({memberObjRef})

// JavaScript syntax
castObjRef.findEmpty({memberObjRef});
```

解説

次の空のキャストメンバ位置または指定されたキャストメンバの後の位置を表示するキャストライブラリメソッドです。

このメソッドは、現在のキャストライブラリでのみ使用できます。

パラメータ

memberObjRef オプション。キャストメンバの参照。このキャストメンバより後にある最初の空のキャストメンバ位置が表示されます。省略すると、次の空のキャストメンバ位置が表示されます。

例文

次のステートメントは、キャストメンバ "100" 以降で最初に見つかった空白キャストメンバを表示します。

```
-- Lingo syntax
trace(castLib(1).findEmpty(member(100)))

// JavaScript syntax
trace(castLib(1).findEmpty(member(100)));
```

関連事項

キャストライブラリ , メンバ

findPos

使用法

```
list.findPos(property)
findPos(list, property)
```

解説

プロパティリスト内でのプロパティの位置を識別するリストコマンドです。

findPos コマンドをリニアリストに使用すると、[プロパティ] の値が数字の場合には不正の数値が返され、[プロパティ] の値がストリングの場合にはスクリプトエラーが発生します。

findPos コマンドは findPosNear コマンドと同じ処理を実行しますが、指定したプロパティがリスト内に存在しない場合、findPos コマンドは VOID を返します。

filterlist の add メソッドまたは append メソッドを使用してフィルタを追加すると、複製が作成されてリストに追加されます。deleteOne、getPos、findPos、getOne などのメソッドは、複製値ではなく、リスト内の正確な値を使用します。

このような場合は、findPos メソッドを次のように使用できます。

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.findPos(f)
```

スクリプトの 3 行目は、フィルタ値の参照をリストに追加します。

パラメータ

property 必須。位置を識別するプロパティ。

例文

次のステートメントは、[#a:10, #b:12, #c:15, #d:22] という内容の Answers リスト内でのプロパティ c の位置を調べます。

```
-- Lingo
Answers.findPos(#c)

// Javascript
Answers.findPos("c");
```

c は 3 番目にリストされているので結果は 3 になります。

関連事項

findPosNear, sort

findPosNear

使用法

```
sortedList.findPosNear(valueOrProperty)
findPosNear(sortedList, valueOrProperty)
```

解説

ソート済みリストについてのみ、指定されたリスト内での項目の位置を識別するリストコマンドです。

findPosNear コマンドは、ソート済みリストのみに使用できます。ソート済みリニアリストの場合は、[値またはプロパティ] に値を指定し、ソート済みプロパティリストの場合は、プロパティを指定します。

findPosNear コマンドは findPos コマンドと同じ処理を実行しますが、指定したプロパティがリスト内に存在しない場合、findPosNear コマンドは、ソート順で最も近いプロパティの位置を返します。名前順にソートされているプロパティから探している名前に最も近いものを見つけるときなどに便利なコマンドです。

パラメータ

valueOrProperty 必須。位置を識別する値またはプロパティ。

例文

次のステートメントは、[#Nile:2, #Pharaoh:4, #Raja:0] という内容のソート済みリスト Answers 内でのプロパティの位置を返します。

```
Answers.findPosNear(#Ni)
```

リスト内で Ni に最も近い最初のプロパティは Nile なので、結果は 1 になります。

関連事項

findPos

finishIdleLoad()

使用法

```
-- Lingo syntax
_movie.finishIdleLoad(intLoadTag)

// JavaScript syntax
_movie.finishIdleLoad(intLoadTag);
```

解説

指定されたロードタグを含む全キャストメンバへのロードを強制的に完了するムービーコマンドです。

パラメータ

intLoadTag 必須。ロードするキャストメンバのロードタグを指定する整数。

例文

次のステートメントは、ロードタグ 20 番目までのすべてのキャストメンバの読み込みを完了します。

```
-- Lingo syntax
_movie.finishIdleLoad(20)

// JavaScript syntax
_movie.finishIdleLoad(20);
```

関連事項

idleHandlerPeriod, idleLoadDone(), idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, ムービー

flashToStage()

使用法

```
-- Lingo syntax
spriteObjRef.flashToStage(pointInFlashMovie)

// JavaScript syntax
spriteObjRef.flashToStage(pointInFlashMovie);
```

解説

Flash ムービースプライト内の特定の座標と一致する Director ステージ上の座標を返す関数です。この関数は、Flash チャネルとムービー座標を受け取り、Director ステージの座標を point(300,300) のように Director のポイント値として返します。

Flash ムービーの座標は、Flash ムービーのピクセル数で示されます。このピクセル数は、ムービーが Flash 内で作成されたときの元のサイズによって決まります。Flash ムービーの座標を計算するには、Flash ムービーの (0,0) ポイントが常に左上隅に位置している必要があります。キャストメンバの originPoint プロパティは回転やスケーリングにのみ使用され、ムービーの座標計算には使用されません。

flashToStage とそれに対応する stageToFlash 関数は、直接 Director のステージの座標に重なる Flash ムービーの座標を特定するときに使用できます。Flash および Director のどちらでも、point(0,0) は Flash ステージまたは Director ステージの左上の隅に位置します。Flash スプライトを伸張、拡大、縮小、または回転した場合は、Director ステージ上でこれらの座標値が一致しないことがあります。

パラメータ

pointInFlashMovie 必須。座標を返す Flash ムービースプライト内の点。

例文

次のハンドラはポイント値とスプライトの参照をパラメータとして受け取り、指定のスプライトの左上隅の座標をチャネル 10 の Flash ムービースプライト内の指定ポイントに設定します。

```
-- Lingo syntax
on snapSprite(whichFlashPoint, whichSprite)
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint)
    _movie.updatestage()
end

// JavaScript syntax
function snapSprite(whichFlashPoint, whichSprite) {
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint);
    _movie.updateStage();
}
```

関連事項

stageToFlash()

float()

使用法

```
(expression).float
float (expression)
```

解説

式を浮動小数点数に変換する関数です (Lingo のみ)。小数点以下の桁数は floatPrecision プロパティを使って設定します。これは、表示用の桁数で計算には影響しません。

JavaScript シンタックスでは、parseFloat() 関数を使用します。

パラメータ

expression 必須。浮動小数点数に変換する式。

例文

次のステートメントは、整数の 1 を浮動小数の 1 に変換します。

```
put (1).float
-- 1.0
```

算術演算は、float を使用して実行できます。いずれかの項が float 値の場合、演算全体が float で実行されます。

```
put 2 + 2
-- 4
put (2).float + 2
-- 4.0
the floatPrecision = 4
put 22/7
-- 3
put (22).float / 7
-- 3.1429"
```

関連事項

floatPrecision, ilk()

floatP()

使用法

```
(expression).floatP
floatP(expression)
```

解説

式が浮動小数点数か (1 または TRUE) そうではないか (0 または FALSE) を示す関数です (Lingo のみ)。

floatP の "**P**" は "**predicate (** 属性 **)**" を意味しています。

パラメータ

expression 必須。検査する式。

例文

次のステートメントは "3.0" が浮動小数かどうかを調べます。結果が TRUE であることを表す数字の "1" がメッセージウィンドウに表示されます。

```
put (3.0).floatP
-- 1
```

次のステートメントは "3" が浮動小数かどうかを調べます。結果が FALSE であることを表す数字の "0" がメッセージウィンドウに表示されます。

```
put (3).floatP
-- 0
```

関連事項

float(), ilk(), integerP(), objectP(), stringP(), symbolP()

flushInputEvents()

使用法

```
-- Lingo syntax
_player.flushInputEvents()

// JavaScript syntax
_player.flushInputEvents();
```

解説

マウスまたはキーボードの待ちイベントを Director のメッセージキューから削除するプレーヤメソッドです。

一般に、このメソッドは、タイトなループを作成し、マウスクリックまたはキーボードの押し下げを見逃さないようにするときに役に立ちます。

このメソッドは、ランタイム時にのみ有効であり、オーサリング時には無効です。

パラメータ

なし

例文

次のような繰り返しループを実行しているときは、マウスやキーボードのイベントは無効になります。

```
-- Lingo syntax
repeat with i = 1 to 10000
    _player.flushInputEvents()
    sprite(1).loc = sprite(1).loc + point(1, 1)
end repeat

// JavaScript syntax
for (var i = 1; i <= 10000; i++) {
    _player.flushInputEvents();
    sprite(1).loc = sprite(1).loc + point(1, 1);
}
```

関連事項

on keyDown, on keyUp, on mouseDown (イベントハンドラ), on mouseUp (イベントハンドラ), プレーヤ

forget() （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.forget()

// JavaScript syntax
windowObjRef.forget();
```

解説

ウィンドウが使用されなくなり、ウィンドウを参照している変数がなくなった場合に、ウィンドウを閉じて再生を停止するようスクリプトに指示するウィンドウメソッドです。

ウィンドウに対して forget() を呼び出すと、windowList からもそのウィンドウの参照が削除されます。

forget() メソッドを呼び出すと、ウィンドウとウィンドウ内のムービー (MIAW) は、stopMovie、closeWindow、deactivateWindow などのハンドラを呼び出すことなく消えます。

ウィンドウ内のムービーに複数のグローバル参照がある場合、ウィンドウは forget() メソッドに反応しません。

パラメータ

なし

例文

次のステートメントは、ムービーの [制御パネル] ウィンドウが不要になったときにそのウィンドウを削除します。

```
-- Lingo syntax
window("Control Panel").forget()

// JavaScript syntax
window("Control Panel").forget();
```

関連事項

close(), open() （ウィンドウ）, ウィンドウ , windowList

forget() (Timeout)

使用法

```
timeout("timeoutName").forget()
forget(timeout("timeoutName"))
```

解説

timeoutList からタイムアウトオブジェクトを削除し、それ以上タイムアウトイベントが送信されないようにするタイムアウトオブジェクト関数です。

パラメータ

なし

例文

次のステートメントは、timeout オブジェクト AlarmClock を timeoutList から削除します。

```
-- Lingo
timeout("AlarmClock").forget()

// Javascript
timeout("AlarmClock").forget();
```

関連事項

timeout(), timeoutHandler, timeoutList, new()

framesToHMS()

使用法

```
framesToHMS(frames, tempo, dropFrame, fractionalSeconds)
```

解説

指定した数のフレームの再生時間を時、分、秒に変換するための関数です。ムービーの実際の再生時間を予測するときや、ビデオ再生機器を制御するときに使用できます。

結果のストリングは "sHH:MM:SS.FFD" という形式になります。

s	時間が 0 未満のときは "-" に、0 以上の場合は半角スペースになります。
HH	時
MM	分
SS	秒
FF	**fractionalSeconds** が TRUE の場合は秒の端数を示し、**fractionalSeconds** が FALSE ・の場合は・ frames ・を示します。
D	**dropFrame** が TRUE の場合は "d" が使用され、**dropFrame** が FALSE の場合はスペースが使用されます。

パラメータ

frames 必須。フレームの数を指定する整数式。

tempo 必須。テンポをフレーム / 秒で指定する整数式。

dropFrame 必須。パラメータは、通常は FALSE です。このパラメータが有効になるのは FPS が 30 フレーム / 秒に設定されているときのみです。ドロップフレームとは、正確には 30 フレーム / 秒ではないカラー NTSC フレームレート用の補正メソッドです。通常、このパラメータは FALSE に設定されます。

fractionalSeconds 必須。残りのフレームがどのように処理されるかを定義します。TRUE の場合、残りのフレームは 1/100 秒単位の近似値に変換され、FALSE の場合、残りのフレームはフレーム数を表す整数で返されます。

例文

次のステートメントは、フレーム数が 2710、フレームレートが 30 フレーム / 秒のムービーを変換します。dropFrame と fractionalSeconds はいずれもオフにしてあります。

```
put framesToHMS(2710, 30, FALSE, FALSE)
-- " 00:01:30.10 "
```

関連事項

HMStoFrames()

frameReady() (Movie)

使用法

```
-- Lingo syntax
_movie.frameReady({intFrameNum})
_movie.frameReady(frameNumA, frameNumB)

// JavaScript syntax
_movie.frameReady({intFrameNum});
_movie.frameReady(frameNumA, frameNumB);
```

解説

Director ムービー、プロジェクタ、および Shockwave コンテンツを含むムービーについて、フレームまたはフレーム範囲のキャストメンバがダウンロードされているかどうかを判定するムービーメソッド。

このメソッドは、指定されたキャストメンバがダウンロードされている場合は TRUE を返し、ダウンロードされていない場合は FALSE を返します。

Director ムービーで使用する frameReady() メソッドの詳細については、Director のヘルプの「Streaming Shockwave」ムービーを参照してください。

パラメータ

intFrameNum 単一のフレームのキャストメンバがダウンロードされているかどうかを検査する場合はオプション。検査する個別のフレームを指定する整数。省略すると、frameReady() は、スコアのいずれかのフレームで使用されているキャストメンバがダウンロードされたかどうかを調べます。

frameNumA フレーム範囲内のキャストメンバがダウンロードされているかどうかを検査する場合は必須。範囲の最初のフレームを指定する整数。

frameNumB フレーム範囲内のキャストメンバがダウンロードされているかどうかを検査する場合は必須。範囲の最後のフレームを指定する整数。

例文

次のステートメントは、フレーム 20 のキャストメンバがダウンロードされており、表示できる状態になっているかどうかを調べます。

```
-- Lingo syntax
on exitFrame
    if (_movie.frameReady(20)) then
        _movie.go(20)
    else
        _movie.go(1)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (_movie.frameReady(20)) {
        _movie.go(20);
    }
    else {
        _movie.go(1);
    }
}
```

次のフレームスクリプトは、チャネル 5 の Flash ムービースプライトのフレーム 25 をレンダリングできるかどうかを調べます。レンダリングできない場合、Director ムービーの現行フレーム内で再生ヘッドのループを継続します。フレーム 25 をレンダリングできる場合、スクリプトは Flash ムービーの再生を開始し、再生ヘッドを Director ムービーの次のフレームに移動します。

関連事項

mediaReady, ムービー

frameStep()

使用法

```
-- Lingo syntax
dvdObjRef.frameStep(intFrames)

// JavaScript syntax
dvdObjRef.frameStep(intFrames);
```

解説

再生が一時停止されるときに、指定されたフレーム数だけ現在の位置を前方に移動する DVD メソッド。

Windows または Mac システムの DVD 再生用ソフトウェアでは、後方への移動はサポートされていません。

パラメータ

intFrames 必須。前方に移動するフレーム数を指定する整数。

例文

次のステートメントは、100 フレームだけ前方にジャンプします。

```
-- Lingo syntax
member("drama").frameStep(100)

// JavaScript syntax
member("drama").frameStep(100);
```

関連事項

DVD

freeBlock()

使用法

```
the freeBlock
```

解説

メモリ内の連続した最大空きブロックのサイズをバイト数で返す関数です。1 KB は 1024 バイトであり、1 MB は 1024 KB です。キャストメンバを読み込むには、そのキャストメンバのサイズ以上の空きブロックが必要です。

パラメータ

なし

例文

次のステートメントは、連続した最大空きブロックが 10KB よりも小さいかどうかどうかを調べ、小さい場合はアラートを表示します。

```
-- Lingo syntax
if (the freeBlock < (10 * 1024)) then alert "Not enough memory!"

// JavaScript syntax
if (freeBlock < (10 * 1024)) {
    alert("Not enough memory!")
}
```

関連事項

freeBytes(), memorySize, ramNeeded(), size

freeBytes()

使用法

```
the freeBytes
```

解説

連続しているかどうかにかかわらず、空きメモリの合計バイト数を返す関数です。1 KB は 1024 バイトであり、1 MB は 1024 KB です。

この関数は freeBlock と異なり、連続した空きメモリではなく未使用メモリすべての合計を返します。

Mac では、Director の「環境設定」またはプロジェクタのオプションダイアログボックスで「システムのメモリを一時的に使用」がオンになっている場合、freeBytes 関数はアプリケーションが使用できるすべての空きメモリを返します。これは、アプリケーションのアプリケーション情報ダイアログボックスに表示される割り当て量、または Mac についてダイアログボックスの「未使用の最大ブロック」の値と同じです。

パラメータ

なし

例文

次のステートメントは、200K 以上のメモリが使用可能かどうかを調べ、使用可能な場合はカラーバージョンのムービーを再生します。

```
if (the freeBytes > (200 * 1024)) then play movie "colorMovie"
```

関連事項

freeBlock(), memorySize, objectP(), ramNeeded(), size

generateNormals()

使用法

```
member(whichCastmember).modelResource(whichModelResource).generateNormals(style)
```

解説

3D #mesh モデルリソースコマンド。メッシュの各頂点の normal ベクトルを計算します。

スタイルパラメータが #flat に設定されている場合、各頂点の法線は、その頂点が属しているフェイスの法線となります。フェイスの 3 つの頂点の法線はすべて同じになります。例えば、face[1] の頂点の法線がすべて normal[1] で、face[2] の頂点の法線がすべて normal[2] の場合、2 つのフェイスが vertex[8] を共有しているなら、vertex[8] の法線は face[1] の normal[1] であり、face[2] の normal[2] です。#flat パラメータを使用すると、メッシュのフェイスは非常にはっきりと描画されます。

スタイルパラメータが #smooth に設定されている場合、各頂点の法線は、頂点の属しているフェイスの数にかかわらず 1 つだけとなります。フェイスの 3 つの頂点の法線は、すべて異なる場合もあります。各頂点の法線は、頂点を共有するフェイスすべての法線の平均となります。#smooth パラメータを使用すると、メッシュのフェイスはより丸みを帯びた外観となります。ただし、メッシュのシルエットにあるフェイスの外側のエッジはシャープなままです。

頂点の法線は、頂点の「前方向」を示す方向ベクトルです。頂点の法線がカメラの方を向いている場合、その法線で制御されるメッシュ領域に表示されるカラーは、シェーダによって決定されます。頂点の法線がカメラの方を向いていない場合、その法線で制御されるメッシュの領域は表示されません。

generateNormals() コマンドを使用した後は、build() コマンドを使用してメッシュを再構築する必要があります。

パラメータ

style 必須。頂点のスタイルを指定するシンボル。

例文

次のステートメントは、"FloorMesh" というモデルリソースの頂点の法線を計算します。スタイルパラメータは #smooth に設定され、mesh の各頂点の法線は 1 つだけになります。

```
-- Lingo
member("Room").modelResource("FloorMesh").generateNormals(#smooth)

// Javascript
member("Room").getProp("modelResource", "index (of the modelresource)").generateNormals("smooth");
```

関連事項

build(), face[ ], normalList, normals, flat

getaProp

使用法

```
propertyList.propertyName
getaProp(list, item)
list[listPosition]
propertyList [ #propertyName ]
propertyList [ "propertyName" ]
```

解説

リニアリストとプロパティリストについて、[リスト] に指定したリスト内にある [項目]、[リストの位置]、[プロパティ名] に割り当てられた値を返すリストコマンドです。

- リニアリストの場合、[リストの位置] に示すようにリスト内の項目の位置を示す番号を [項目] に指定します。結果はその位置の値になります。
- プロパティリストの場合、リスト内の [プロパティ名] で示されるプロパティを [項目] に指定します。結果はそのプロパティに割り当てられた値になります。

指定した値がリスト内に存在しない場合、getaProp コマンドは VOID を返します。

リニアリストの場合、getaProp は getAt コマンドと同じ結果を返します。

パラメータ

itemNameOrNum 必須。リニアリストの場合は、返すリスト内の値のインデックス位置を指定する整数。プロパティリストの場合は、値を返すプロパティを指定するシンボル (Lingo) またはストリング (JavaScript syntax)。

例文

次のステートメントは、[#john:10, #joe:12, #cheryl:15, #barbara:22] という内容のプロパティリスト ages からプロパティ #joe の値を返します。

```
put getaProp(ages, #joe)
```

#joe のプロパティ値は 12 であるため、結果は 12 となります。

同じリストにブラケットアクセスを使用して同じ結果を出すこともできます。

```
put ages[#joe]
```

この結果も 12 になります。

リストの特定の位置にある値を調べる場合も、ブラケットアクセスを使用できます。リストの 3 番目のプロパティに割り当てられた値を得るには、次のシンタックスを使用します。

```
put ages[3]
-- 15
```

注意：プロパティが存在しないときに VOID を返す getAProp コマンドと異なり、ブラケットアクセスを使用した場合は、プロパティが存在しないときにスクリプトエラーが発生します。

関連事項

getAt(), getOne(), getProp(), setaProp, setAt

getAt()

使用法

```
getAt(list, position)
list [position]
```

解説

指定されたリストの指定された位置にある項目を識別するリストコマンドです。指定した位置がリストに含まれているエレメント数より大きい場合、スクリプトエラーが発生します。

getAt コマンドはリニアリストとプロパティリストの両方に適用できます。リニアリストでは getaProp コマンドと同じ結果を返します。

このコマンドは deskTopRectList などのほかのリストから項目を取り出すときに使用できます。

パラメータ

list 必須。項目が存在するリストを指定します。

position 必須。リスト内での項目のインデックス位置を指定します。

例文

次のステートメントは [10, 12, 15, 22] という内容のリスト answers にある 3 番目の項目の値をメッセージウィンドウに表示します。

```
put getAt(answers, 3)
-- 15
```

ブラケットアクセスを使用しても同じ値が返されます。

```
put answers[3]
-- 15
```

次の例では、氏名、部門、社員番号情報を指定する 2 つのエントリを含むリストから、最初のエントリを抽出します。次に新規に抽出したリストの 2 番目のエレメントを返し、リストの最初の人が所属する部門を識別します。リストのフォーマットは [["Dennis", " 顧問 ", 510]、["Sherry", " 配送 ", 973]] で、リスト名は employeeInfoList です。

```
firstPerson = getAt(employeeInfoList, 1)
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = getAt(firstPerson, 2)
put firstPersonDept
-- "consulting"
```

中間ステップで変数に値を割り当てずに、getAt コマンドをネストすることもできます。次のフォーマットは読んだり記述したりするのは困難ですが、簡素な表記になります。

```
firstPersonDept = getAt(getAt(employeeInfoList, 1), 2)
put firstPersonDept
    -- "consulting"
```

ブラケットアクセスも使用できます。

```
firstPerson = employeeInfoList[1]
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = firstPerson[2]
put firstPersonDept
-- "consulting"
```

getAt と同様に、ブラケットアクセスでもネストできます。

```
firstPersonDept = employeeInfoList[1][2]
```

関連事項

getaProp, setaProp, setAt

getCharSet

使用法

```
FileIO.getCharSet()
```

解説

FileIO Xtra メソッド。アクティブな FileIO 文字セットを返します。デフォルトの文字セットは UTF-8 です。

例文

```
--Lingo syntax
FileIO.getCharSet()

//JavaScript syntax
FileIO.getCharSet();
```

関連事項

readChar(), readFile(), readLine(), readToken(), readWord(), writeChar(), writeString()

getError() (Flash, SWA)

使用法

```
-- Lingo syntax
memberObjRef.getError()

// JavaScript syntax
memberObjRef.getError();
```

解説

Shockwave Audio (SWA) または Flash キャストメンバについて使用し、キャストメンバがメモリに読み込まれ、値を返したときにエラーが発生しかたどうかを示す関数です。

Shockwave Audio キャストメンバの場合に getError() が返す整数の値と getErrorString() が返す値の対比表を、次に示します。

getError() **値**	getErrorString() **メッセージ**
0	OK
1	メモリ
2	network
3	playback device
99	other

Flash ムービーキャストメンバの場合、getError は次の値を返します。

- FALSE - エラーは発生していません。
- #memory - キャストメンバを読み込むためのメモリが不足しています。
- #fileNotFound - キャストメンバのアセットを含むファイルが見つかりません。
- #network - ネットワークエラーが発生したためキャストメンバを読み込むことができません。
- #fileFormat - ファイルは見つかりましたが、種類が異なっているか、またはファイルの読み込み中にエラーが発生しました。
- #other - 他のエラーが発生しました。

キャストメンバがメモリに読み込まれているときにエラーが起きると、Director はキャストメンバの状態プロパティを -1 に設定します。getError 関数を使用してエラーの種類を調べます。

パラメータ

なし

例文

次のハンドラは、getError を使用して、Shockwave Audio キャストメンバ "Norma Desmond Speaks" に関してエラーが発生したかどうかを調べ、フィールドにエラーを示す適切なストリングを表示します。

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond \ Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

次のハンドラは、"Dali" という Flash キャストメンバでメモリへの読み込み中にエラーが発生したかどうかを調べます。エラーが発生し、それがメモリエラーの場合、このスクリプトは unloadCast コマンドを使用してメモリを開放しようとします。次に、Flash ムービースプライトが最初に表示される "Artists" という Director ムービーのマーカに再生ヘッドを移動します。これによって、Director は Flash ムービーの読み込みと再生を再び試みます。メモリ不足以外のエラーが発生した場合、スクリプトは、要求された Flash ムービーを再生できないことを表示する「お詫び」というマーカに移動します。

```
-- Lingo syntax
on CheckFlashStatus
errorCheck = member("Dali").getError()
    if errorCheck <> 0 then
        if errorCheck = #memory then
            member("Dali").clearError()
            unloadCast()
            _movie.go("Artists")
        else
            _movie.go("Sorry")
        end if
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
var errorCheck = member("Dali").getError();
    if (errorCheck != 0) {
        if (errorCheck == "memory") {
            member("Dali").clearError();
            unloadCast();
            _movie.go("Artists");
        } else {
            _movie.go("Sorry");
        }
    }
}
```

関連事項

clearError(), getErrorString(), state (Flash, SWA)

getError() (XML)

使用法

```
parserObject.getError()
```

解説

関数 ; 所定のエラー番号に関連するエラーの説明 (エラーが発生した XML の行と列の番号も含む) を返します。エラーがない場合、この関数は <VOID> を返します。

パラメータ

なし

例文

次のステートメントは、XML データを含むストリングを構文解析した後にエラーを確認します。

```
-- Lingo
errCode = parserObject.parseString(member("XMLtext").text)
errorString = parserObject.getError()
if voidP(errorString) then
    -- Go ahead and use the XML in some way
else
    alert "Sorry, there was an error " & errorString
    -- Exit from the handler
    exit
end if

// Javascript
errCode = parserObject.parseString(member("XMLtext").text);
errorString = parserObject.getError();
if (errorString != null)
{
    // Go ahead and use the XML in some way
}else{
    _player.alert("Sorry, there was an error " + errorString);
    // Exit from the handler
}
```

getErrorString()

使用法

```
-- Lingo syntax
memberObjRef.getErrorString()

// JavaScript syntax
memberObjRef.getErrorString();
```

解説

Shockwave Audio (SWA) キャストメンバについて使用し、getError() 関数が返すエラーの値に対応するエラーメッセージストリングを返す関数です。

getError() と getErrorString() が返す値の対比表を次に示します。

getError() 値	getErrorString() メッセージ
0	OK
1	メモリ
2	network
3	playback device
99	other

パラメータ
なし

例文
次のハンドラは、getError() を使用して、Shockwave Audio キャストメンバ "Norma Desmond Speaks" にエラーが発生したかどうかを調べます。エラーが発生している場合は、getErrorString を使ってエラーメッセージを取得し、メッセージをフィールドキャストメンバに割り当てます。

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
    var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

関連事項
getError() (Flash, SWA)

getFinderInfo()

使用法

```
-- Lingo syntax
fileioObjRef.getFinderInfo()

// JavaScript syntax
fileioObjRef.getFinderInfo();
```

解説
開いているファイルの Finder 情報を返す Fileio メソッドです (Mac のみ)。

先に openFile() を呼び出してファイルを開いた後、getFinderInfo() を使用してファイルの Finder 情報を取得する必要があります。

パラメータ
なし

例文
```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.getFinderInfo()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.getFinderInfo() ;
```

関連事項
Fileio, openFile()

getFlashProperty()

使用法
```
-- Lingo syntax
spriteObjRef.getFlashProperty(targetName, symProp)

// JavaScript syntax
spriteObjRef.getFlashProperty(targetName, symProp);
```

解説
指定した Flash スプライトの Flash アクションスクリプト関数 getProperty() を呼び出します。この Flash アクションスクリプト関数を使用して、ムービークリップのプロパティの値、または Flash ムービー内のレベルを調べます。これは、Director のスプライトプロパティを調べるのとよく似ています。

Flash スプライトのグローバルプロパティを調べるには、空ストリング ("") を [ターゲット名] として渡します。調べることができるグローバル Flash プロパティは、#focusRect と #spriteSoundBufferTime です。

これらのプロパティの詳細については、Flash のマニュアルを参照してください。

パラメータ
targetName 必須。指定された Flash スプライト内でプロパティを取得するムービークリップまたはレベルの名前を指定するストリング。

symProp 必須。取得するプロパティの名前を指定するシンボル。有効な値は以下のとおりです。#posX、#posY、#scaleX、#scaleY、#visible、#rotate、#alpha、#name、#width、#height、#target、#url、#dropTarget、#totalFrames、#currentFrame、#cursor、#lastframeLoaded。

例文
次のステートメントは、sprite 3 の Flash メンバのムービークリップ "Star" の #rotate プロパティの値を調べます。

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate)
sprite(3).getFlashProperty("Star")

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"));
sprite(3).getFlashProperty("Star");
```

getFrameLabel()

使用法

```
sprite(whichFlashSprite).getFrameLabel(whichFlashFrameNumber)
getFrameLabel(sprite whichFlashSprite, whichFlashFrameNumber)
```

解説

要求したフレーム番号に該当する Flash ムービーのフレームラベルを返します。フレームラベルが存在しない場合、または Flash ムービーのその部分がまだ組み込まれていない場合は、空ストリングを返します。

パラメータ

whichFlashFrameNumber 必須。フレームラベルと関連付けられているフレーム番号を指定します。

例文

次のハンドラは、スプライト 1 で再生されている Flash ムービーのフレーム 15 に設定されているラベルが " ライオン " であるかを調べます。ラベルが " ライオン " の場合は、Director のムービーがマーカ " ライオン " に移動します。そうでない場合は、現行フレーム内にとどまり、Flash ムービーの再生を続行します。

```
-- Lingo syntax
on exitFrame
    if sprite(1).getFrameLabel(15) = "Lions" then
        go "Lions"
    else
        go the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    if (sprite(1).getFrameLabel(15) == "Lions") {
        _movie.go("Lions");
    } else {
        _movie.go(_movie.frame);
    }
}
```

getHardwareInfo()

使用法

```
getRendererServices().getHardwareInfo()
```

解説

3D rendererServices メソッド。ユーザのビデオカードに関する情報のプロパティリストを返します。このリストには、次のプロパティが含まれます。

#present は、コンピュータにハードウェアビデオアクセラレーター機能があるかどうかを示すブール値です。

#vendor は、ビデオカードの製造元の名前を示します。

#model は、ビデオカードのモデル名を示します。

#version は、ビデオドライバのバージョンを示します。

#maxTextureSize は、ピクセル単位でのテクスチャの最大幅と高さのリニアリストです。このサイズを超えるテクスチャは、サイズ内に収まるようにサンプリングされます。テクスチャがサンプリングされないようにするには、様々なサイズのテクスチャを作成し、#maxTextureSize の値を超えないものを実行時に選択します。

#supportedTextureRenderFormats は、ビデオカードでサポートされているテクスチャピクセルフォーマットのリニアリストです。詳細については、textureRenderFormat を参照してください。

#textureUnits は、カードで利用できるテクスチャユニット数を示します。

#depthBufferRange は、depthBufferDepth プロパティに設定できるビット深度解像度のリニアリストです。

#colorBufferRange は、colorBufferDepth プロパティに設定できるビット深度解像度のリニアリストです。

例文

次のステートメントは、ユーザのハードウェアに関する情報の詳細なプロパティリストを表示します。

```
-- Lingo
put getRendererServices().getHardwareInfo()
-- [#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS (Dell)", #version:
"4.12.01.0532", #maxTextureSize: [2048, 2048], #supportedTextureRenderFormats: [#rgba8888, #rgba8880,
#rgba5650, #rgba5551, #rgba5550, #rgba4444], #textureUnits: 2, #depthBufferRange: [16, 24],
#colorBufferRange: [16, 32]]

// Javascript
trace(getRendererServices().getHardwareInfo())
//<[#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS (Dell)", #version:
"4.12.01.0532", #maxTextureSize: [2048, 2048], #supportedTextureRenderFormats: [#rgba8888, #rgba8880,
#rgba5650, #rgba5551, #rgba5550, #rgba4444], #textureUnits: 2, #depthBufferRange: [16, 24],
#colorBufferRange: [16, 32]]>
```

関連事項

getRendererServices()

getHotSpotRect()

使用法

```
-- Lingo syntax
spriteObjRef.getHotSpotRect(hotSpotID)

// JavaScript syntax
spriteObjRef.getHotSpotRect(hotSpotID);
```

解説

ホットスポットのおおよその境界矩形を返す QuickTime VR 関数です。ホットスポットが存在しない場合、またはステージで見えない状態になっている場合、この関数は rect(0, 0, 0, 0) を返します。ホットスポットが部分的に見えている状態の場合、この関数は見えている部分の境界矩形を返します。

パラメータ

hotSpotID 必須。境界の矩形を返すホットスポットを指定します。

getInstalledCharSets

使用法

```
_system.getInstalledCharSets()
```

解説

文字セットメソッド。コンピュータにインストールされている文字セットのリストを返します。

例文

```
--Lingo syntax
put _system.getInstalledCharSets()

//JavaScript syntax
put(_system.getInstalledCharSets());
```

GetItemPropList

使用法

```
GetItemPropList(MUIObject)
```

解説

この関数は、汎用目的ダイアログボックスのコンポーネントに使用する MUI Xtra の定義済みプロパティのリストを返します。汎用目的ダイアログボックスに新しいコンポーネントを定義するときに役立ちます。包括的なプロパティと値のリストを取得してから個々のプロパティを必要に応じて編集するには、GetItemPropList を使用します。

プロパティと値のリストは以下のとおりです。

プロパティ	デフォルト値
#value	0
#type	#checkBox
#attributes	[],
#title	"title"
#tip	"tip"（これは以降のバージョンの MUI Xtra で使用する可能性があるため予約されています。）
#locH	20
#locV	24

プロパティ	デフォルト値
#width	200
#height	210
#enabled	1

例文

次のステートメントはダイアログボックスウィンドウの開始位置を定義します。

- 最初のステートメントは、ダイアログボックスとして使用されるオブジェクトである MUI Xtra のインスタンスを作成します。
- 2 番目のステートメントは、ダイアログのコンポーネントのデフォルト設定のリストを変数 tempItemProps に割り当てます。
- 3 番目のステートメントは、コンポーネントのタイプを #windowBegin に変更することによってコンポーネントをダイアログボックスの開始位置にします。

```
--Lingo syntax
set aMuiObj = new (Xtra "MUI")
set tempItemProps = GetItemPropList(aMuiObj)
set the type of tempItemProps = #windowBegin
```

getLast()

使用法

```
list.getLast()
getLast(list)
```

解説

[リスト] で指定したリニアリストまたはプロパティリスト内にある最後の項目の値を返すリスト関数です。

パラメータ

なし

例文

次のステートメントは、"Answers" というリストの最後の項目 "22" を返します。このリストの内容は、[10, 12, 15, 22] です。

```
put Answers.getLast()
```

次のステートメントは、"Bids" というリストの最後の項目 "850" を返します。このリストの内容は [#Gee:750, #Kayne:600, #Ohashi:850] です。

```
put Bids.getLast()
```

getLatestNetID

使用法

```
getLatestNetID
```

解説

直前に開始されたネットワーク操作で使用された識別子を返します。

getLatestNetID で返される識別子を netDone, netError, および netAbort の各関数のパラメータとして使用することで、直前のネットワーク操作を特定することができます。

注意：この関数は前のバージョンとの互換性のために使用可能となっています。getLatestNetID はなるべく使用せず、代わりにいずれかの net 関数から返されるネットワーク ID を使用してください。やむを得ず getLatestNetID を使用する場合は、いずれかの netLingo コマンドを発行した直後に使用してください。

パラメータ

なし

例文

次のスクリプトは、getNetText 演算のネットワーク ID をフィールドキャストメンバ "Result" に割り当てます。これにより、後でこの演算の結果を利用することができます。

```
on startOperation
    global gNetID
    getNetText("url")
    set gNetID = getLatestNetID()
end
on checkOperation
    global gNetID
    if netDone(gNetID) then
        put netTextResult into member "Result"
    end if
end
```

関連事項

netAbort, netDone(), netError()

getLength()

使用法

```
-- Lingo syntax
fileioObjRef.getLength()

// JavaScript syntax
fileioObjRef.getLength();
```

解説

開いているファイルの長さを返す Fileio メソッドです。

先に openFile() を呼び出してファイルを開いた後、getLength() を使用してファイルの長さを取得する必要があります。

パラメータ

なし

例文

次の例では、c:\check.txt という名前のファイルを開き、ファイルの長さを取得します。

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile("c:\check.txt",2)
put objFileio.getLength()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\check.txt",2);
trace(objFileio.getLength())
```

関連事項

Fileio, openFile()

getNetByteArray

使用法

```
getNetByteArray(URL, [propertyList], [serverOSString])
```

解説

Net Lingo メソッド。Web ページをバイト配列としてダウンロードします。このメソッドは getNetText と似ています。

パラメータ

パラメータ	解説	必須 / オプション
URL	必要なバイト配列が存在するファイルの URL。	必須
propertyList	CGI クエリに使用するプロパティリストを指定します。	オプション
serverOSString	propertyList で返される文字のエンコードを指定します。	オプション

例文

```
--Lingo syntax
id = getNetByteArray("http://www.bytearray.com/byte.cgi")

//JavaScript syntax
id = getNetByteArray("http://www.bytearray.com/byte.cgi");
```

getNetText()

使用法

```
getNetText(URL {, serverOSString} {, characterSet})
getNetText(URL, propertyList {, serverOSString} {, characterSet})
```

解説

HTTP または FTP サーバにあるテキストファイルの読み取りを開始したり、CGI クエリを開始するための関数です。

最初のシンタックスはテキストの検索を開始します。HTTP の CGI をこのようにしてサブミットする場合は、正しく URL 形式でコード化することが必要です。2 番目のシンタックスではプロパティのリストが組み込まれており、CGI のクエリをサブミットして、正しく URL 形式でコード化できるようにします。

オプションのパラメータ [プロパティリスト] を使用すると、CGI クエリのプロパティリストを取得できます。このプロパティリストは URL 形式でコード化され、URL は (urlstring & "?" & encodedproplist) が送信されます。

オプションのパラメータ [サーバ **OS** ストリング指定] を使用すると、返却文字を [プロパティリスト] にコード化することができます。この値はデフォルトでは UNIX ですが、Win や Mac を設定することで、[プロパティリスト] パラメータに含まれるすべての改行を、サーバで使用する形式の改行文字に変換できます。ほとんどのアプリケーションでは、この設定は必要ありません。フォームでの応答では通常、改行を使用しないためです。

オプションパラメータ [キャラクタセット指定] は、Director を shiftJIS（日本語）システムで実行する場合にのみ使用します。文字セットの設定には、JIS、EUC、ASCII、AUTO があります。検索されたデータは、shift-JIS から目的の文字セットに変換します。AUTO 設定を使用することにより、検索した文字セットがどの文字セットの中にあるかを調べ、ローカルマシンに変換することができます。デフォルトの設定は ASCII です。

getNetText 操作が完了したかどうかを確認するには、netDone を使用します。getNetText 操作が正常に完了したかどうかを確認するには、netError を使用します。getNetText で取得したテキストを返すには netTextResult を使用します。

ファイル指定には相対 URL を使用することもできます。

完成したムービーの getNetText() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Forms andPost ムービーを参照してください。

パラメータ

URL 必須。取得するテキストが存在するファイルの URL。

propertyList オプション。CGI クエリに使用するプロパティリストを指定します。

serverOSString オプション。**propertyList** で返される文字のエンコードを指定します。

characterSet オプション。文字ストリングを指定します。

例文

次のスクリプトは、マウスボタンがクリックされると URL "http://BigServer.com/sample.txt" からテキストを読み取り、マウスポインタが置かれていたフィールドキャストメンバを更新します。

```
property spriteNum

property theNetID

on mouseUp me
    theNetID = getNetText ("http://BigServer.com/sample.txt")
end

on exitFrame me
    if netDone(theNetID) then
        sprite(spriteNum).member.text = netTextResult(theNetID)
    end if
end
```

次の例文は、CGI クエリの検索結果を読み取ります。

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi?name=Bill")
```

前記の例文と同じですが、プロパティリストを使用して CGI クエリをサブミットし、ユーザに代わって URL をコード化します。

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi", [#name:"Bill"])
```

関連事項

netDone(), netError(), netTextResult()

getNormalized

使用法

```
getNormalized(vector)
vector.getNormalized()
```

解説

3D ベクトルメソッド。ベクトルをコピーし、コピーの x、y、および z コンポーネントの値を、元のベクトルの長さで割ります。計算結果のベクトルの長さは、1 ワールド単位になります。

このメソッドはコピーを返し、元のベクトルは変更しません。元のベクトルを正規化するには、normalize コマンドを使用します。

例文

次のステートメントは、ベクトル "MyVec" の正規化した値を変数 Norm に格納します。"Norm" の値は vector(-0.1199, 0.9928, 0.0000) となり、"Norm" の大きさは 1 になります。

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
Norm = MyVec.getNormalized()
put Norm
-- vector( -0.1199, 0.9928, 0.0000 )
put Norm.magnitude
-- 1.0000

// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
Norm = MyVec.getNormalized();
trace(Norm);
// vector( -0.1199, 0.9928, 0.0000 )
trace(Norm.magnitude);
// 1.0000
```

関連事項

normalize

getNthFileNameInFolder()

使用法

```
getNthFileNameInFolder(folderPath, fileNumber)
```

解説

パスで指定したフォルダ内の、指定した番号に対応するファイルまたはフォルダの名前を返すムービーメソッドです。Director ムービーファイルを getNthFileNameInFolder 関数で見つけるには、そのファイルがシステムのフォルダ構造において可視である必要があります。ただし、Mac ではその他のファイルを可視または不可視の状況にかかわらず見つけることができます。関数が EMPTY を返した場合は、指定したファイル番号がフォルダ内の項目数より大きいことを表しています。

getNthFileNameInFolder 関数は、URL には使用できません。

その他のフォルダ名を指定するには、@ pathname 演算子を使うか、またはムービーが再生されるプラットフォームの形式で完全なパスを指定します。例えば次のようになります。

- Windows では、"C:/Director/Movies" のようにディレクトリパスで指定します。

- Mac の場合は、" ハードディスク :Director: ムービー :" のような形式で記述します。デスクトップ上のファイルを見つけるには " ハードディスク :Desktop Folder" を指定します。
- この関数は Shockwave Player では使用できません。

パラメータ

folderPath 必須。ファイルが存在するフォルダへのパスを指定します。

fileNumber 必須。フォルダ内でのファイルのインデックス位置を指定します。

例文

次のハンドラは、現行パスにあるフォルダ内のファイル名リストを最大 100 個まで返します。このように結果を返すハンドラでは、put currentFolder() というように括弧を使って呼び出します。

```
-- Lingo
on currentFolder
fileList = []
repeat with i = 1 to 100
        n=getNthFileNameInFolder(the moviePath, i)
        if n = EMPTY then exit repeat
        fileList.append(n)
end repeat
return fileList
end currentFolder

// Javascript
function currentFolder()
{
    fileList = list();
    var i=1;
    while(i<100)
    {
        n = _movie.getNthFileNameInFolder(_movie.path,i);
                if (n==="")
        {
            i=101;
        }
        else
        {
            fileList.append(n);
        }
        i++;
    }
    return fileList;
}
```

関連事項

@ (パス名演算子), ムービー

getOne()

使用法

```
list.getOne(value)
getOne(list, value)
```

解説

リスト内で値と関連付けられている位置 (リニアリストの場合) またはプロパティ (プロパティリストの場合) を識別するリスト関数です。

リスト内に値が重複して存在する場合は、最初の値だけが取得されます。指定した値がリスト内に存在しない場合、getOne コマンドは 0 を返します。

リニアリストでは、getOne コマンドと getPos コマンドの結果は同じです。

filterlist の add メソッドまたは append メソッドを使用してフィルタを追加すると、複製が作成されてリストに追加されます。deleteOne、getPos、findPos、getOne などのメソッドは、複製値ではなく、リスト内の正確な値を使用します。

このような場合は、getOne メソッドを次のように使用できます。

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getOne(f)
```

スクリプトの 3 行目は、フィルタ値の参照をリストに追加します。

パラメータ

value 必須。位置またはプロパティと関連付けられている値を指定します。

例文

次のステートメントは、[10, 12, 15, 22] という内容の Answers リストで値が 12 の項目の位置を返します。

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

"12" はリストの 2 番目の値であるため、結果は "2" になります。

[#a:10, #b:12, #c:15, #d:22] という内容のプロパティリスト Answers があった場合、次のステートメントはその中で "12" の値を含むプロパティ名を返します。

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

12 という値を含むプロパティは #b であるため、結果は "#b" となります。

関連事項

getPos()

getOSDirectory()

使用法

```
-- Lingo syntax
getOSDirectory()

// JavaScript syntax
getOSDirectory();
```

解説
System フォルダ (Mac) または Windows ディレクトリ (Windows) への完全なパスを返す関数です。

パラメータ
なし

例文
次のステートメントは、マシンの OS ディレクトリを取得します。この場合は c:\windows です。

```
-- Lingo
Put getOSDirectory()

// Javascript
trace(getOSDirectory());
```

関連事項
Fileio

getPixel()

使用法

```
-- Lingo syntax
imageObjRef.getPixel(x, y {, #integer})
imageObjRef.getPixel(point(x, y) {, #integer})

// JavaScript syntax
imageObjRef.getPixel(x, y {, #integer});
imageObjRef.getPixel(point(x, y) {, #integer});
```

解説
画像メソッド。指定された画像の指定された点におけるピクセルのインデックスカラーまたは RGB カラーを返します。

返される画像の行および列のインデックスは、0 から始まります。したがって、画像の左上隅のピクセルの位置は、(1,1) ではなく (0,0) で指定します。画像の高さが **h** ピクセルで幅が **w** ピクセルの場合、右下隅のピクセルにアクセスするには、(**w**,1)、(**h**,1) と指定します。

指定されたピクセルが指定された画像の外側にある場合、このメソッドは 0 を返します。

多数のピクセルを別の 1 ピクセルと同じカラーに設定する場合は、raw number として設定するほうが高速です (オプションの #integer パラメータを使用して)。また、画像が 32 ビットの場合、raw number のカラー値にはカラー情報とアルファレイヤー情報の両方が含まれているので便利です。アルファチャネル情報は、raw number を 2^8+8+8 で除算することにより抽出できます。

パラメータ
xx 座標と y 座標を使用してピクセルを指定する場合は必須。ピクセルの x 座標を指定する整数。

yx 座標と y 座標を使用してピクセルを指定する場合は必須。ピクセルの y 座標を指定する整数。

#integer オプション。返されるカラー値の raw number を指定するシンボル。

point**(x, y)** 点を使用してピクセルを指定する場合は必須。ピクセルの位置を指定する点。

例文

次のステートメントは、メンバ画像ステージのポイント (20, 20) のピクセルのカラーを取得します。

```
-- Lingo
objImage = _movie.stage.image
objImage.getPixel(20, 20)
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.getPixel(20, 20) ;
put (objImage) ;
```

関連事項

color(), image(), power(), setPixel()

getPixels()

使用法

```
Bytearray image.getPixels(#symbolImageFormat)
Possible values form symImageFormat are:
#bgra8888 => 32 bit BGRA
#argb8888=> 32 bit ARGB
#rgba8888=> 32 bit RGBA
#rgb888=> 24bit RGB
#bgr888=> 24bit BGR
```

解説

バイト配列メソッド。画像データで初期化されるバイト配列を作成します。

注意：32 ビットの Lingo 画像オブジェクトのみがサポートされています。

例文

次のコードスニペットは、画像を赤色で塗りつぶし、アルファ値を 50 に設定します。

```
--Lingo syntax
on mouseUp me
i32=image(128,128,32) -- Creating a 32 bit image
i32.useAlpha=1 -- Setting the Alpha to 1
ba=i32.getPixels(#RGBA8888) -- Getting the pixels of the image using 32 bit RGBA format
ba2=byteArray(128*128*4,0) -- Creating the bytearray
i=0
repeat with r=1 to 128
repeat with c=1 to 128
ba2[i+1]=255 -- Value for RED
ba2[i+2]=0 -- Value for GREEN
ba2[i+3]=0 -- Value for BLUE
ba2[i+4]=50 -- Value for Alpha
i=i+4
end repeat
end repeat
i32.setPixels(ba2,#RGBA8888) -- Setting the pixels of the image using 32 bit RGBA format
member("test").image=i32
end
//JavaScript Syntax
function mouseUp(me)
{
```

```
var i=0;
i32=image(128,128,32);
i32.useAlpha=1;
ba=i32.getPixels(symbol("RGBA8888"));
ba2=byteArray(128*128*4,0);
i=0;
for(r=1;r<=128;r++)
{
for(c=1;c<=128;c++)
{
ba2[i+1]=255;
ba2[i+2]=0;
ba2[i+3]=0;
ba2[i+4]=50;
i=i+4;
}
}
i32.setPixels(ba2,symbol("RGBA8888"));
member("test").image=i32;
}
```

関連事項

setPixels()

getPlayList()

使用法

```
-- Lingo syntax
soundChannelObjRef.getPlayList()

// JavaScript syntax
soundChannelObjRef.getPlayList();
```

解説

サウンドチャネルのキューに登録されたサウンドのリストのコピーを返すサウンドチャネルメソッドです。

返されるリストは、現在再生中のサウンドは含まれず、直接編集することはできません。setPlayList() を使用してください。

playlist は、各プロパティリストを 1 次元のリストにしたものです。各プロパティリストは、1 つの待機サウンドキャストメンバに対応しています。1 つの待機サウンドで次のようなプロパティを指定できます。

プロパティ	解説
#member	キューに登録するサウンドキャストメンバ。このプロパティは指定する必要があります。これ以外はすべてオプションです。
#startTime	再生を開始するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの先頭です。「startTime」を参照してください。
#endTime	再生を終了するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの最後です。「endTime」を参照してください。
#loopCount	#loopStartTime と #loopEndTime で定義されたループを再生する回数。デフォルトは 1 です。「loopCount」を参照してください。

プロパティ	解説
#loopStartTime	ループを開始するサウンド内の時間 (単位はミリ秒)。「loopStartTime」を参照してください。
#loopEndTime	ループを終了するサウンド内の時間 (単位はミリ秒)。「loopEndTime」を参照してください。
#preloadTime	再生前にバッファに溜めておくサウンドの量 (単位はミリ秒)。「preloadTime」を参照してください。

パラメータ

なし

例文

このハンドラは、2 つのサウンドをサウンドチャネル 2 に待機させ、再生を開始し、playList をメッセージウィンドウに表示します。playList には 2 番目の待機サウンドしか含まれませんが、それは既に 1 番目のサウンドが再生されているからです。

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000, #loopCount:5,
#loopStartTime:8000, #loopEndTime:8900])
    put(sound(2).getPlayList())
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(member("Chimes"));
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000, "endTime",10000,
"loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    put(sound(2).getPlayList());
sound(2).play();
}
```

関連事項

endTime （サウンドチャネル）, loopCount, loopEndTime （サウンドチャネル）, loopStartTime, メンバ, member, preLoadTime, queue(), setPlayList(), サウンドチャネル , startTime (サウンドチャネル)

getPos()

使用法

```
list.getPos(value)
getPos(list, value)
```

解説

リスト内での値の位置を識別するリスト関数です。指定した値がリスト内に存在しない場合、getPos コマンドは 0 を返します。

リスト内に値が重複して存在する場合は、最初の値だけが取得されます。リニアリストでは、このコマンドと getOne コマンドの結果は同じです。

filterlist の add メソッドまたは append メソッドを使用してフィルタを追加すると、複製が作成されてリストに追加されます。deleteOne、getPos、findPos、getOne などのメソッドは、複製値ではなく、リスト内の正確な値を使用します。

このような場合は、getPos メソッドを次のように使用できます。

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getPos(f)
```

スクリプトの 3 行目は、リストに追加されたフィルタ値の参照を取得します。

パラメータ

value 必須。位置と関連付けられている値を指定します。

例文

次のステートメントは、[#a:10, #b:12, #c:15, #d:22] という内容の Answers リストで値が 12 の項目の位置を返します。

```
-- Lingo
put Answers.getPos(12)

// Javascript
trace(Answers.getPos(12));
```

"12" はリストの 2 番目の値であるため、結果は "2" になります。

関連事項

getOne()

getPosition()

使用法

```
-- Lingo syntax
fileioObjRef.getPosition()

// JavaScript syntax
fileioObjRef.getPosition();
```

解説

ファイルの位置を返す Fileio メソッドです。

パラメータ

なし

例文

次のステートメントは、ファイル c:\xtra.txt を開き、現在の位置を取得します。

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
put objFileio.getPosition()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
trace(objFileio.getPosition());
```

関連事項

Fileio

getPref()

使用法

```
getPref(prefFileName)
```

解説

指定したファイルの内容を読み取る関数です。

[ファイル名] には、既に setPref 関数で作成されたファイルの名前を指定します。ファイルがない場合、getPref 関数は VOID を返します。

[ファイル名] に指定できるのは、フルパス名ではなく、有効なファイル名のみです。パスは Director によって付加されます。ファイルへのパスは Director が処理します。有効な拡張子は .txt および .htm のみで、その他の拡張子は拒否されます。

このコマンドは読み込み専用ファイルやロックされたメディアへのアクセスには使用できません。

注意：ブラウザでは setPref で記述されたデータはプライベートではありません。Shockwave コンテンツを含む任意のムービーからこの情報を読み取り、サーバにアップロードできます。部外秘情報は setPref を使って保存しないでください。

完成したムービーの getPref() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Read and WriteText ムービーを参照してください。

パラメータ

prefFileName 必須。コンテンツを取得するファイルを指定します。

例文

次のハンドラは、テストファイルの内容を読み取り、そのテキストを合計スコアフィールドに割り当てます。

```
on mouseUp
    theText = getPref("Test")
    member("Total Score").text = theText
end
```

関連事項

setPref()

getPref()（プレーヤ）

使用法

```
-- Lingo syntax
_player.getPref(stringPrefName)

// JavaScript syntax
_player.getPref(stringPrefName);
```

解説

指定されたファイルの内容を取得するプレーヤメソッドです。

このメソッドを使用するときは、**stringPrefName** を、setPref() メソッドで作成されたファイルの名前に置き換えます。ファイルが存在しない場合、getPref() は VOID（Lingo）または null（JavaScript シンタックス）を返します。

stringPrefName に指定できるのは、フルパス名ではなく、有効なファイル名のみです。パスは Director によって付加されます。ファイルへのパスは Director が処理します。**stringPrefName1** に対して有効な拡張子は .txt および .htm のみで、その他の拡張子は拒否されます。

このメソッドは読み込み専用ファイルやロックされたメディアへのアクセスには使用できません。

注意：ブラウザでは setPref() で記述されたデータはプライベートではありません。Shockwave コンテンツを含むムービーで、この情報を読み取り、サーバにアップロードできます。機密情報は、setPref() を使用して保存しないでください。

完成したムービーの getPref() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の Read and WriteText ムービーを参照してください。

パラメータ

stringPrefName 必須。コンテンツを取得するファイルを指定するストリング。

例文

次のハンドラは、テストファイルの内容を読み取り、そのテキストを合計スコアフィールドに割り当てます。

```
-- Lingo syntax
on mouseUp
    theText = _player.getPref("Test")
    member("Total Score").text = theText
end

// JavaScript syntax
function mouseUp() {
    var theText = _player.getPref("Test");
    member("Total Score").text = theText;
}
```

関連事項

プレーヤ , setPref()

getProp()

使用法

```
getProp(list, property)
list.property
```

解説

プロパティリスト内のプロパティと関連付けられている値を識別するプロパティリスト関数です。

getProp コマンドは、getaProp コマンドと同じ処理を実行します。指定したプロパティがリスト内にない場合や、リニアリストを指定した場合は、getProp コマンドはエラーメッセージを表示します。

パラメータ

list 必須。**property** を取得するプロパティリストを指定します。

property 必須。識別された値が関連付けられているプロパティを指定します。

例文

次のステートメントは、プロパティリスト "Answers" で、プロパティ #c に割り当てられている値を返します。"Answers" リストの内容は [#a:10, #b:12, #c:15, #d:22] です。

```
-- Lingo
getProp(Answers, #c)

/ Javascript
Answers.getProp("c");
```

#c の値は 15 であるため、結果として 15 を返します。

関連事項

getOne()

getPropAt()

使用法

```
list.getPropAt(index)
getPropAt(list, index)
```

解説

プロパティリストについてのみ、プロパティリスト内の指定された位置と関連付けられているプロパティ名を識別するプロパティリスト関数です。指定した項目がリスト内に存在しない場合、またはリニアリストで getPropAt() を使った場合、スクリプトエラーが発生します。

パラメータ

index 必須。プロパティリスト内でのプロパティのインデックス位置を指定します。

例文

[#a:10, #b:12, #c:15, #d:22] という内容のプロパティリスト Answers で 2 番目のプロパティを識別します。

```
-- Lingo
put Answers.getPropAt(2)
-- #b

// Javascript
trace(Answers.getPropAt(2))
// b
```

getPropRef()（JavaScript のみ）

使用法

```
getPropRef(list, property)
```

解説

プロパティリスト関数。プロパティリスト内のプロパティの参照を返します。JavaScript シンタックス内でカスケードプロパティにアクセスするために使用します。また、getPropRef() メソッドを使用して、前述のオブジェクトまたはそのプロパティへの参照を格納することもできます。

パラメータ

プロパティ	解説
list	必須。property を取得するプロパティリストを指定します。
property	必須。識別された値が関連付けられているプロパティを指定します。

例文

```
put(member(1).getPropRef("paragraph",1))
//Retrieves the first paragraph from the text member.
```

getRendererServices()

使用法

```
getRendererServices()
getRendererServices().whichGetRendererServicesProperty
```

解説

3D コマンド。rendererServices オブジェクトを返します。このオブジェクトには、3D スプライトおよびキャストメンバのすべてに影響を与える、ハードウェア情報とプロパティが含まれます。

rendererServices オブジェクトには、次のプロパティがあります。

- renderer は、すべての 3D スプライトのレンダリングに使用するソフトウェアラスタライザを示します。
- rendererDeviceList は、ユーザのシステムで利用可能なソフトウェアラスタライザのリストを返します。返される値は、#openGL、#directX5_2、#directX7_0、#directx9 および #software です。renderer の値は、これらの値のいずれかです。このプロパティは、値を調べられますが、設定はできません。
- textureRenderFormat は、レンダラが使用するピクセルフォーマットを示します。返される値は、#rgba8888、#rgba8880、#rgba5650、#rgba5550、#rgba5551、または #rgba4444 です。各シンボルの 4 桁の数字は、赤、緑、青、およびアルファの各コンポーネントに使用されるビット数を示します。
- depthBufferDepth は、ハードウェアの出力バッファのビット深度を示します。
- colorBufferDepth は、カラーバッファのビット深度を示します。このプロパティは、値を調べられますが、設定はできません。
- modifiers は、3D キャストメンバのモデルで使用可能なモディファイアのリニアリストです。返される値は、#collision、#bonesPlayer、#keyframePlayer、#toon、#lod、#meshDeform、#sds、#inker、またはサードパーティの Xtra ベースのモディファイアです。このプロパティは、値を調べられますが、設定はできません。
- primitives は、新しいモデルリソースの作成に利用できるプリミティブの種類を表すリニアリストです。返される値は、#sphere、#box、#cylinder、#plane、#particle、およびサードパーティの Xtra ベースのプリミティブの種類です。このプロパティは、値を調べられますが、設定はできません。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

パラメータ

なし

例文

次のステートメントは、rendererServices オブジェクトを取得し、レンダラの情報を取得します。

```
-- Lingo
Ro = getRendererServices()
Put Ro.renderer
-- #directX7_0

// Javascript
Var ro = getRendererServices();
trace(ro.renderer);
// #directX7_0
```

関連事項

renderer, preferred3dRenderer, active3dRenderer, rendererDeviceList

getSoundObject()

使用法

```
soundObjRef = mixer.getSoundObject(SoundObjName)
```

解説

オーディオメソッド。指定した名前のミクサと関連付けられたサウンドオブジェクトを返します。Director が一致するサウンドオブジェクトを検出しない場合、void を返します。

戻り値

SoundObjRef

パラメータ

パラメータ	解説	必須 / オプション
SoundObjRef	サウンドオブジェクト参照。	必須
SoundObjName	取得するサウンドオブジェクトの名前。	必須

例文

```
-- Lingo syntax

 on mouseUp me

    mixer1.getSoundObject("SoundObj1") -- Returns the reference of the sound object named "soundobj1".
 end

// JavaScript syntax
 function mouseup()
 {
 mixer1.getSoundObject("SoundObj1"); //Returns the reference of the sound object named "SoundObj1".

 }
```

関連事項

ミクサ

getSoundObjectList

使用法

```
soundObjList = mixer.getSoundObjectList()
```

解説

オーディオメソッド。削除されていないミクサ内のすべてのサウンドオブジェクトのリストを返します。

戻り値

サウンドオブジェクトのリスト

例文

```
-- Lingo syntax
on mouseUp me

    mixer1.getSoundObjectList() -- Returns the list of the sound objects in mixer1.
end

// JavaScript syntax

function mouseup()
{
mixer1.getSoundObjectList();  // Returns the list of sound objects in mixer1.


}
```

関連事項

ミクサ

getStreamStatus()

使用法

```
getStreamStatus(netID)
getStreamStatus(URLString)
```

解説

グローバルに利用可能な tellStreamStatus 関数に使用されるフォーマットと一致するプロパティリストを返す関数で、スプライトやオブジェクトへのコールバックで使用できます。リストの内容は次のとおりです。

#URL	ネットワーク操作を開始するために使用される URL の位置を含むストリングです。
#state	"Connecting"、"Started"、"InProgress"、"Complete"、"Error"、または "NoInformation" を含むストリングです。最後のストリングは、ネット ID が古すぎてステータス情報が欠落した場合や、URLString で指定した URL がキャッシュで見つからない場合に使用されます。
#bytesSoFar	その時点までにネットワークから読み取ったバイト数です。
#bytesTotal	確認できる場合は、ストリームのバイト総数です。HTTP サーバがコンテンツの長さの情報を MIME ヘッダに含めない場合は、値が 0 になることがあります。
#error	ダウンロードの結果を示すストリングです。値は、ダウンロードが完了していない場合は ""（EMPTY）に、正常に完了した場合は OK に、ダウンロードがエラーで終了した場合はエラーコードとなります。

例えば、getNetText() でネットワーク操作を開始して getStreamStatus() で進行を追跡することができます。

パラメータ

netID 必須。操作するテキストのストリームを表すネットワーク操作です。

例文

次のステートメントは、getNetText() で始まるダウンロードの現行ステータスをメッセージウィンドウに表示し、その結果、変数 netID にネットワーク上の ID が設定された状態で実行したところです。

```
-- Lingo
put getStreamStatus(netID)
-- [#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000, #error: EMPTY]

// Javascript
trace(getStreamStatus(netID))
// <[#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000, #error: EMPTY]>
```

関連事項

on streamStatus, tellStreamStatus()

getSystemCharSet

使用法

```
_system.getSystemCharSet()
```

解説

文字セットメソッド。コンピュータのネイティブ文字セットを返します。

例文

```
--Lingo syntax
put _system.getSystemCharSet()

//JavaScript syntax
put(_system.getSystemCharSet());
```

getURL()

使用法

```
GetUrl(MUIObject, message, MovableOrNot)
```

解説

この関数は、URL を入力するためのダイアログボックスを表示し、ユーザが入力した URL を返します。

- **message** は、URL を入力するためのフィールドに表示するメッセージを指定します。ダイアログボックスが最初に開いたとき、このストリングが定義済みの値として送信されます。ユーザがボタンをクリックすると、Lingo ではユーザが入力したストリングが返されます。ユーザが「キャンセル」をクリックした場合、返されるストリングは元の値と同じです。

- Macintosh では、**MovableOrNot** はダイアログボックスが移動可能かどうかを指定します。TRUE を指定するとダイアログボックスは移動可能になります。FALSE を指定するとダイアログボックスは移動できません。Windows では、GetURL ダイアログボックスは常に移動可能です。

例文

次のステートメントは、URL を入力するためのダイアログボックスを表示します。

- 最初のステートメントは、ダイアログボックスとして使用されるオブジェクトである MUI Xtra のインスタンスを作成します。
- 2 番目のステートメントは、GetUrl 関数を使用して URL を入力するための移動可能なダイアログボックスを表示し、ダイアログボックスを変数 result に割り当てます。URL 入力ダイアログボックスのフィールドに「ここに URL を入力してください」というメッセージを表示します。
- 最後のステートメントは、ユーザがボタンをクリックした後の結果が、ダイアログボックスが開かれたときに送信されたストリングと同じかどうかをチェックします。異なる場合は、ユーザが URL を入力して「OK」をクリックしたことを示します。

```
-- Lingo
set MUIObj = new (xtra "Mui")
set result = GetUrl(MUIObj, "Enter a URL", TRUE )
if objectP ( MUIObj) then
    set result = GetUrl( MUIObj, "Enter a URL", TRUE )
    if ( result <> "Enter a URL" ) then
        goToNetPage result
    end if
end if
```

getVal()

使用法

```
<float> Matrix.getVal(whichRow, whichColumn)
```

解説

マトリックスメソッドです。指定したマトリックスの指定した要素の値を取得します。

パラメータ

行 必須。値を読み取る要素の行番号です。

列 必須。値を読み取る要素の列番号です。

例文

次の関数では、マトリックスの getVal() メソッドを使用して、マトリックスをリニアリストに変換しています。

```
--Lingo
on matrixToList(mat)
     rows = mat.numRows
     cols = mat.numColumns
     matrixList = []
     repeat with i = 1 to rows
       repeat with j = 1 to cols
         matrixList.append(mat.getVal(i,j))
       end repeat
     end repeat
     return matrixList
end
//Java Script
function matrixToList(mat)
{
     rows = mat.numRows;
     cols = mat.numColumns;
     matrixList = list();
     for( i = 1; i <= rows; i++)
     {
       for( j = 1; j <= cols; j++)
         matrixList.append(mat.getVal(i,j));
}
     return matrixList;
}
```

関連事項

setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

getVariable()

使用法

```
-- Lingo syntax
spriteObjRef.getVariable(variableName {, returnValueOrReference})

// JavaScript syntax
spriteObjRef.getVariable(variableName {, returnValueOrReference});
```

解説

指定された [Flash スプライト] の、ある [変数] の現在の値を返す関数です。Flash 変数は Flash バージョン 4 で導入されました。

この関数には、2 とおりの使い方があります。

オプションの [戻り値または参照] パラメータを TRUE に設定すると (デフォルト)、この関数は現在の値をストリングで返します。FALSE の場合は、Flash 変数の現在のリテラル値を返します。

Flash 変数の値がオブジェクト参照の場合は、戻り値がオブジェクト参照できるように、[戻り値または参照] パラメータは FALSE に設定する必要があります。この値をストリングとして返すと、ストリングは有効なオブジェクト参照にはなりません。

パラメータ

variableName 必須。値を取得する変数の名前を指定します。

returnValueOrReference オプション。値をストリングとして返すか (TRUE)、またはオブジェクト参照として返すか (FALSE) を指定します。

例文

次のステートメントは、変数 "tValue" に、sprite 3 の Flash ムービーの Flash 変数 "gOtherVar" のストリング値を設定します。

```
-- Lingo syntax
tValue = sprite(3).getVariable("gOtherVar", TRUE)
put(tValue) -- "5"
// JavaScript syntax
var tValue = sprite(3).getVariable("gOtherVar", true);
trace(tValue); // 5
```

次のステートメントは、スプライト 3 の Flash ムービーの Flash 変数 gVar と同じオブジェクトを参照するよう tObject 変数を設定します。

```
-- Lingo syntax
tObject = sprite(3).getVariable("gVar",FALSE)
// JavaScript syntax
var tObject = sprite(3).getVariable("gVar",0);
```

次のステートメントは、sprite 3 の Flash キャストメンバの変数 currentURL の値を返し、メッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(3).getVariable("currentURL"))

// JavaScript syntax
trace(sprite(3).getVariable("currentURL"));
```

関連事項

setVariable()

GetWidgetList()

使用法

```
GetWidgetList(MUIObject)
```

解説

この関数は、MUI Xtra のインスタンスでサポートされる汎用目的ダイアログボックスのコンポーネントのタイプに応じたシンボルのリニアリストを返します。

例文

次のステートメントは、MUI Xtra のインスタンスである MUIObject がサポートするウィジェットのリストを表示します。

```
-- Lingo
put GetWidgetList(MUIObject)
-- [#dividerV, #dividerH, #bitmap, #checkBox, #radioButton, #PopupList, #editText, #WindowBegin,
#WindowEnd, #GroupHBegin, #GroupHEnd, #GroupVBegin, #GroupVEnd, #label, #IntegerSliderH, #FloatSliderH,
#defaultPushButton, #cancelPushButton, #pushButton, #toggleButton]
```

GetWindowPropList

使用法

```
GetWindowPropList(MUIObject)
```

解説

この関数は、汎用目的ダイアログボックスのウィンドウに使用する MUI Xtra の定義済みプロパティのリストを返します。

新しい汎用目的ダイアログボックスを定義するときは、GetWindowPropList 関数を使用してダイアログボックスのプロパティと値の包括的なリストを取得し、次に個々のプロパティを必要に応じて編集します。この方法を使用すると、より便利なだけでなく、プロパティが追加される可能性がある今後のバージョンの MUI Xtra との互換性が保証されます。

GetWindowPropList から返されるウィンドウのプロパティと定義済みの値は以下のとおりです。

プロパティ	定義済み値
#type	#normal
#name	"window"
#callback	"nothing"
#mode	#data
#xPosition	100
#yPosition	120
#width	200
#height	210
#modal	1
#toolTips	0
#closeBox	1
#canZoom	0

例文

次のステートメントは、新しい汎用目的ダイアログボックスを定義します。最初のステートメントは、定義済みプロパティのリストを変数 thePropList に割り当てます。後続のステートメントは、定義済みプロパティの設定を変更してダイアログボックスをカスタマイズします。

```
-- Lingo
set thePropList = GetWindowPropList(muiObject)
set the name of thePropList = "Picture Window"
set the callback of thePropList = "theWindowCallback"
set the mode of thePropList = #data
set the modal of thePropList = TRUE
set the closeBox of thePropList = FALSE
```

getWorldTransform()

使用法

```
member(whichCastmember).node(whichNode).getWorldTransform()
member(whichCastmember).node(whichNode).getWorldTransform().position
member(whichCastmember).node(whichNode).getWorldTransform().rotation
member(whichCastmember).node(whichNode).getWorldTransform().scale
```

解説

3D コマンド。node で表されるモデル、グループ、カメラ、またはライトの、ワールド基準のトランスフォームを返します。

ノードの transform プロパティは、ノードの親のトランスフォームを基準に計算され、親を基準とする値になります。getWorldTransform() コマンドでは、ノードのトランスフォームは 3D ワールドの原点を基準に計算され、ワールドを基準とする値となります。

member(キャストメンバ).node(ノード).getWorldTransform().position を使用して、ノードのワールドを基準とするトランスフォームの位置プロパティを取得します。また、getWorldTransform().position の代わりに worldPosition も使用できます。

member(キャストメンバ).node(ノード).getWorldTransform().rotation を使用して、ノードのワールドを基準とするトランスフォームの回転プロパティを取得します。

member(キャストメンバ).node(ノード).getWorldTransform().scale を使用して、ノードのワールドを基準とするトランスフォームのスケールプロパティを取得します。

これらのプロパティはテストできますが、設定はできません。

例文

次のステートメントは、" ボックス " というモデルのワールド基準のトランスフォームを、位置および回転プロパティと共に示します。

```
put member("3d world").model("Box").getworldTransform()
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, - 94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("Box"). getworldTransform().position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("Box"). getworldTransform().rotation
--vector(0.0000, 0.0000, 0.0000)

// Javascript
trace(member("3d world").getProp("model","Box").getworldTransform()
// <transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().position
// <vector(-94.1448, 119.0128, 0.0000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().rotation
//<vector(0.0000, 0.0000, 0.0000)>
```

関連事項

worldPosition, transform （プロパティ）

go()

使用法

```
-- Lingo syntax
_movie.go(frameNameOrNum {, movieName})

// JavaScript syntax
_movie.go(frameNameOrNum {, movieName});
```

解説

指定したムービー内の指定したフレームに再生ヘッドをジャンプさせるムービーメソッドです。

このメソッドを使用すると、再生ヘッドは直前のマーカへ戻ってループ再生するので、スクリプトをアクティブにしたまま再生ヘッドを同じフレーム内に留めておくのに便利です。

フレーム番号はムービーを編集すると変わることがあるため、フレーム番号ではなく、**frameNameOrNum** のマーカラベルを使用するとよいでしょう。マーカラベルを使う方が、スクリプトの内容を理解しやすくなります。

movieName パラメータを使用して go() を呼び出すと、ムービーのフレーム 1 がロードされます。go() がハンドラ内から呼び出された場合、そのハンドラの実行は継続されます。

再生するムービーを指定するときに、ムービーが別のフォルダに入っている場合はそのパスも指定します。ただし、ロードに失敗する可能性があるので、ファイルの拡張子に .dir、.dxr、.dcr を付けないようにします。

URL にあるムービーを再生する場合、まず downloadNetThing() メソッドでムービーファイルをローカルディスクにダウンロードし、次に go() メソッドと **movieName** パラメータを使用してローカルディスク上のムービーとして呼び出す方が効率的です。

goLoop() メソッドは、ムービー内の直前のマーカに playhead を移動します。これは、Lingo または JavaScript のシンタックスをアクティブにしたまま、再生ヘッドを同じフレーム内に留めておくのに便利です。

beepOn プロパティと constraint プロパティ、keyDownScript、mouseDownScript、および mouseUpScript、cursor スプライトプロパティと immediate スプライトプロパティ、cursor() コマンドと puppetSprite() コマンド、およびカスタムメニューは、ムービーを読み込むときにリセットされます。ただし、timeoutScript はムービーのロード時にはリセットされません。

パラメータ

frameNameOrNum 必須。再生ヘッドのジャンプ先のフレームのマーカラベルを指定するストリング、または再生ヘッドのジャンプ先のフレームの番号を指定する整数です。

movieName オプション。**frameNameOrNum** によって指定されるフレームを含むムービーを指定するストリングです。この値では、ムービーファイルを指定する必要があります。ムービーが別のフォルダに含まれている場合は、**movieName** でパスも指定する必要があります。

例文

次のステートメントは、再生ヘッドを " 開始 " というマーカの位置まで移動します。

```
-- Lingo syntax
_movie.go("start")

// JavaScript syntax
_movie.go("start");
```

次のステートメントは、再生ヘッドを " 物語 " というムービーの " 追憶 " マーカの位置まで移動します。

```
-- Lingo syntax
_movie.go("Memory", "Noh Tale to Tell")

// JavaScript syntax
_movie.go("Memory", "Noh Tale to Tell");
```

次のハンドラは、現行フレームでムービーをループします。ムービーの再生中に 1 つのフレームにムービーを留め、ムービーがイベントに応答できるようにするときに使用できます。

```
-- Lingo syntax
on exitFrame
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    _movie.go(_movie.frame);
}
```

関連事項

downloadNetThing, goLoop(), ムービー

goLoop()

使用法

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

解説

再生ヘッドを現在の位置から一番近い前のマーカまで送るムービーメソッドです。現行フレームにマーカがない場合はその前のマーカに送り、現行フレームにマーカがある場合は現行フレームに送ります。

現在位置より左側にマーカがない場合、再生ヘッドの移動先は次のようになります。

- 現行フレームにマーカがない場合、現行位置より右側の一番近いマーカ
- 現行フレームにマーカがある場合、現行フレーム
- ムービーにマーカがない場合、フレーム 1

パラメータ

なし

例文

次のステートメントは、ムービーを現行フレームと直前のマーカとの間でループさせます。

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

関連事項

go(), goNext(), goPrevious(), ムービー

goNext()

使用法

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

解説

再生ヘッドを現在位置から一番近い次のマーカまで送るムービーメソッドです。

再生ヘッドの現行位置より右側にマーカが設定されていない場合は、左に向かって一番近いマーカにジャンプします。左側にもマーカがない場合、フレーム 1 に再生ヘッドを移動します。

パラメータ

なし

例文

次のステートメントは、再生ヘッドをムービー内の次のマーカ位置まで移動します。

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

関連事項

go(), goLoop(), goPrevious(), ムービー

goPrevious()

使用法

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

解説

再生ヘッドを現在位置から一番近い前のマーカまで送るムービーメソッドです。

現行フレームにマーカがない場合、現行フレームから 2 つ戻ったマーカまで送り、現行フレームに 1 つのマーカがある場合、現行フレームから 1 つ戻ったマーカまで送ります。

現在位置より左側にマーカがない場合、再生ヘッドの移動先は次のいずれかになります。

- 現行フレームにマーカがない場合、現行位置より右側の一番近いマーカ
- 現行フレームにマーカがある場合、現行フレーム
- ムービーにマーカがない場合、フレーム 1

パラメータ

なし

例文

次のステートメントは、再生ヘッドをムービー内の前のマーカ位置まで移動します。

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

関連事項

go(), goLoop(), goNext(), ムービー

goToFrame()

使用法

```
-- Lingo syntax
spriteObjRef.goToFrame(frameNameOrNum)

// JavaScript syntax
spriteObjRef.goToFrame(frameNameOrNum);
```

解説

[フレーム番号] で指定されたフレームから始まる Flash ムービースプライトを再生するコマンドです。フレーム番号を示す整数とラベル名を示すストリングのいずれでも識別できます。goToFrame コマンドは、Flash ムービースプライトの frame プロパティと同じ効果があります。

例文

次のハンドラは、チャネル 5 の Flash ムービーの異なるフレームに分岐します。どのフレームに分岐するかは whereTo パラメータで指定します。

```
-- Lingo syntax
on Navigate(whereTo)
    sprite(5).goToFrame(whereTo)
end

// JavaScript syntax
function Navigate(whereTo) {
    sprite(5).goToFrame(whereTo);
}
```

gotoNetMovie

使用法

```
gotoNetMovie URL
gotoNetMovie (URL)
```

解説

HTTP または FTP サーバから Shockwave コンテンツを含む新しいムービーを読み取って再生するためのコマンドです。新しいムービーが再生できるようになるまでは、現行のムービーが続けて再生されます。

パラメータに指定できるのは URL のみです。URL にはムービーのファイル名またはマーカを指定します。ムービーがインターネットサーバにある場合は、相対 URL を使用できます。ファイル名は拡張子も含めて指定する必要があります。

ローカルディスクやネットワークで動作を確認する場合、メディアは、dswmedia ディレクトリまたはその下位ディレクトリに格納する必要があります。

gotoNetMovie を実行し、これが完了する前にもう 1 つの gotoNetMovie コマンドを発行すると、最初のコマンドが 2 番目のコマンドによってキャンセルされます。

パラメータ

URL 必須。再生する Shockwave コンテンツの URL を指定します。

例文

次のステートメントの URL は、Director のファイル名を指定します。

```
gotoNetMovie "http://www.yourserver.com/movies/movie1.dcr"
```

次のステートメントの URL は、ファイル名内のマーカを示します。

```
gotoNetMovie "http://www.yourserver.com/movies/buttons.dcr#Contents"
```

次のステートメントでは、gotoNetMovie が関数として処理され、操作のネットワーク ID を myNetID に返します。

```
myNetID = gotoNetMovie ("http://www.yourserver.com/movies/buttons.dcr#Contents")
```

gotoNetPage

使用法

```
gotoNetPage "URL", {"targetName"}
```

解説

Shockwave コンテンツを含むムービーやその他の MIME タイプのファイルをブラウザで開くコマンドです。

パラメータに指定できるのは URL のみです。ムービーが HTTP または FTP サーバにあるときは、相対 URL を使用できます。

オーサリング環境では、通常使用するブラウザを指定しておくと、gotoNetPage コマンドはそのブラウザを起動します。プロジェクタでは、[ネットワーク環境設定] ダイアログボックスまたは browserName コマンドで設定したブラウザを起動します。どちらの方法でもブラウザが設定されていない場合、goToNetPage コマンドはコンピュータ上にあるブラウザを検索します。

パラメータ

URL 必須。再生する Shockwave コンテンツを含むムービーまたは MIME タイプのファイルの URL を指定します。

targetName オプション。ページがロードされるフレームまたはウィンドウを識別する HTML パラメータです。

- [ターゲット名] がブラウザ内のウィンドウまたはフレームの場合、gotoNetPage は指定されたウィンドウまたはフレームの表示内容を入れ替えます。
- [ターゲット名] が現在開いているウィンドウまたはフレームの名前と一致しない場合、goToNetPage は新しいウィンドウを開きます。ストリング "_new" を使用すると、常に新しいウィンドウが開かれます。
- [ターゲット名] が省略されている場合、gotoNetPage は位置にかかわらず現行 HTML ページを入れ替えます。

例文

次のスクリプトは、"frwin" という名前のウィンドウまたはフレームに "Newpage.html" ファイルを読み込みます。現行ウィンドウに "frwin" というウィンドウまたはフレームが存在する場合は、そのウィンドウまたはフレームが使用されます。frwin ウィンドウが存在しない場合は、frwin という名前の新しいウィンドウが作成されます。

```
on keyDown
    gotoNetPage "Newpage.html", "frwin"
end
```

次のハンドラは、ブラウザが現在開いているウィンドウにかかわらず、常に新しいウィンドウを開きます。

```
on mouseUp
    gotoNetPage "Todays_News.html", "_new"
end
```

関連事項

browserName(), netDone()

group()

使用法

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

解説

3D エレメント。名前、トランスフォーム、親、および子のプロパティのみを持つ 3D ワールドのノードです。

各 3D キャストメンバには、"World" というデフォルトのグループがあります。このグループは削除できません。3D ワールドに存在するモデル、ライト、カメラ、およびグループすべての親階層は、最後が group("world") で終わります。

例文

次のステートメントは、キャストメンバ "newAlien" の 4 番目のグループが "Direct01" であることを示します。

```
-- Lingo
put member("newAlien").group[4]

// Javascript
put member("newAlien").getPropRef("group",4) ;
-- group("Direct01")
```

関連事項

newGroup, deleteGroup, child (3D), parent

halt()

使用法

```
-- Lingo syntax
_movie.halt()

// JavaScript syntax
_movie.halt();
```

解説

現行ハンドラと、それを呼び出したハンドラから抜け出すためのムービーメソッドです。オーサリングを実行している場合はムービーを停止し、プロジェクタから再生している場合はプロジェクタを終了します。

パラメータ

なし

例文

次のステートメントは、空きメモリの量が 50KB 未満かどうかを調べ、50KB 未満の場合は、それを呼び出したすべてのハンドラから抜け出してムービーを停止します。

```
-- Lingo syntax
if (_system.freeBytes < (50*1024)) then
    _movie.halt()
end if

// JavaScript syntax
if (_system.freeBytes < (50*1024)) {
    _movie.halt();
}
```

関連事項

ムービー

handler()

使用法

```
scriptObject.handler(#handlerSymbol)
```

解説

この関数は、指定された [スクリプトオブジェクト] に指定されたハンドラが含まれていると TRUE を返し、含まれていないと FALSE を返します。スクリプトオブジェクトは親スクリプト、子オブジェクト、ビヘイビアーであることが必要です。

パラメータ

symHandler 必須。ハンドラの名前を指定します。

例文

このコードは、呼び出したハンドラが存在する場合にのみ、そのハンドラを呼び出します。

```
-- Lingo
if spiderObject.handler(#pounce) = TRUE then
    spiderObject.pounce()
end if

// Javascript
if (spiderObject.handler(symbol("pounce")) == true)
{
    spiderObject.pounce();
}
```

関連事項

handlers(), new(), rawNew(), script()

handlers()

使用法

```
scriptObject.handlers()
```

解説

この関数は、指定された [スクリプトオブジェクト] に含まれているハンドラのリニアリストを返します。このリニアリストには、ハンドラの名前がシンボルで記述されます。この関数はムービーのデバッグに便利です。

スクリプトキャストメンバのハンドラを直接取得することはできません。スクリプトキャストメンバの script プロパティを通して取得することが必要です。

パラメータ
なし

例文
次のステートメントは、子オブジェクト RedCar のハンドラのリストをメッセージウィンドウに表示します。

```
put RedCar.handlers()
-- [#accelerate, #turn, #stop]
```

次のステートメントは、親スクリプトメンバ "CarParentScript" のハンドラのリストをメッセージウィンドウに表示します。

```
put member("CarParentScript").script.handlers()
-- [#accelerate, #turn, #stop]
```

関連事項
handler(), script()

hilite （コマンド）

使用法

```
fieldChunkExpression.hilite()
hilite fieldChunkExpression
```

解説
フィールドスプライト内の指定されたチャンク式 (文字のまとまり) をハイライトする (選択する) ためのコマンドです。選択する内容は、Lingo で定義可能な文字、単語、行などです。Mac の場合は、「カラー」コントロールパネルで設定されている色がハイライトの色として使用されます。

パラメータ
なし

例文
次のステートメントは、"Thought for the Day" という内容の " コメント " フィールドキャストメンバ内の 4 番目の単語 (Day) をハイライトします。

```
-- Lingo
member("Comments").word[4].hilite()

// Javascript
member("Comments").getPropRef("word",4).hilite() ;
```

関連事項
char...of, item...of, line...of, word...of, delete(), mouseChar, mouseLine, mouseWord, field, selection() （関数）, selEnd, selStart

hitTest()

使用法

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

解説

Director ステージ上の指定した位置にある Flash ムービーの部分を示す関数です。Director ステージの位置は、Director の点の値として表されます。例えば、point(100,50) です。hitTest 関数は次の値を返します。

- #background - 指定した位置は、Flash ムービースプライトの背景です。
- #normal - 指定した位置には、塗りつぶされたオブジェクトがあります。
- #button - 指定した位置は、アクティブなボタン上です。
- #editText - 指定したステージ位置は、Flash 編集可能テキストフィールド内にあります。

パラメータ

point 必須。テストするポイントを指定します。

例文

次のフレームスクリプトは、チャネル 5 の Flash ムービースプライトのボタン上にマウスがあるかどうかを調べ、マウスがある場合はテキストフィールドにその説明を表示します。

```
-- Lingo syntax
on exitFrame
    if sprite(5).hitTest(_mouse.mouseLoc) = #button then
        member("Message Line").text = "Click here to play the movie."
        _movie.updatestage()
    else
        member("Message Line").text = ""
    end if
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    var hT = sprite(5).hitTest(_mouse.mouseLoc);
    if (hT.toString() == "#button")
    {
        member("Message Line").text = "Click here to play the movie.";
        _movie.updatestage();
    } else {
        member("Message Line").text = "";
    }
    _movie.go(_movie.frame)
}
```

HMStoFrames()

使用法

```
HMStoFrames(hms, tempo, dropFrame, fractionalSeconds)
```

解説

時、分、秒で測定されているムービーの長さをフレーム数に変換するための関数です。また、[テンポ] パラメータを 1 (1 フレーム = 1 秒) に設定してこの関数を使うと、時、分、秒を表す数字を秒数に変換することもできます。

パラメータ

hms 必須。"sHH:MM:SS.FFD" という形式で時間を指定するストリング式です。

s	時間が 0 未満のときは "-" に、0 以上の場合は半角スペースになります。
HH	時
MM	分
SS	秒
FF	fractionalSeconds が TRUE の場合は秒の端数になり、fractionalSeconds が FALSE のときはフレーム数になります。
D	[ドロップフレーム] が TRUE のときは d が使用され、FALSE のときは半角スペースが使用されます。

tempo 必須。テンポをフレーム / 秒で指定します。

dropFrame 必須。フレームがドロップフレームであるか（TRUE）、ドロップフレームでないか（FALSE）を示す論理式です。[時分秒] が "**d**" で終わる場合は、[ドロップフレーム] の値にかかわらず時間がドロップフレームとして処理されます。

fractionalSeconds 必須。秒の端数の処理方法を定義する論理式です。TRUE に設定すると、秒の後の数字は 1/100 秒単位で四捨五入され、FALSE に設定すると、秒の後の数字が残りのフレーム数を表します。

例文

次のステートメントは、テンポが 30 フレーム / 秒に設定されているときに、1 分 30.1 秒のムービーのフレーム数を表示します。dropFrame と fractionalSeconds パラメータはいずれもオフにしてあります。

```
put HMStoFrames(" 00:01:30.10 ", 30, FALSE, FALSE)
-- 2710
```

次のステートメントは、600 秒を分に変換します。

```
put framesToHMS(600, 1,0,0)
-- " 00:10:00.00 "
```

次のステートメントは 1 時間 30 分を秒に変換します。

```
put HMStoFrames("1:30:00", 1,0,0)
-- 5400
```

関連事項

framesToHMS()

hold()

使用法

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

解説

オーディオを再生し続けながら、現在のフレームで再生中の Flash ムービースプライトを停止する Flash コマンドです。

パラメータ

なし

例文

次のフレームスクリプトは、チャネル 5 ～ 10 で再生されている Flash ムービースプライトを現在のフレームで停止しますが、オーディオは再生し続けます。

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).hold()
    end repeat
end

// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).hold();
        i++;
    }
}
```

関連事項

playRate（QuickTime、AVI、MP4、FLV）

importByteArrayInto()

使用法

```
memberRef.importByteArrayInto(ByteArrayRef,strFileType)
```

解説

バイト配列の内容からメンバを読み込むバイト配列メソッドです。

パラメータ

なし

例文

```
--Lingo Syntax
--Read JPEG file from disk and import it as bitmap
file = new xtra("fileio")
file.openFile(_movie.path & "import.jpg", 1)
ba = file.readByteArray(file.getLength())
m=new(#bytearray)
m.bytearray=ba
member(5).importByteArrayInto(m.bytearray,"jpg")
//Javascript Syntax
file = new xtra("fileio");
file.openFile(_movie.path + "import.jpg", 1);
ba = file.readByteArray(file.getLength());
m=_movie.newMember(symbol("bytearray"));
m.bytearray=ba;
member(5).importByteArrayInto(m.bytearray,"jpg");
```

identity()

使用法

```
member(whichCastmember).model(whichModel).transform.identity()
member(whichCastmember).group(whichGroup).transform.identity()
member(whichCastmember).camera(whichCamera).transform.identity()
sprite(whichSprite).camera{(index)}.transform.identity()
member(whichCastmember).light(whichLight).transform.identity()
transformReference.identity()
```

解説

3D コマンド。トランスフォームをアイデンティティトランスフォームに設定します。つまり、transform(1.0000,0.0000,0.0000,0.0000, 0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,1.0000,0.0000, 0.0000,0.0000,0.0000,1.0000) となります。

アイデンティティトランスフォームの position プロパティは vector(0, 0, 0) です。

アイデンティティトランスフォームの rotation プロパティは vector(0, 0, 0) です。

アイデンティティトランスフォームの scale プロパティは vector(1, 1, 1) です。

アイデンティティトランスフォームは親を基準とします。

パラメータ

なし

例文

次のステートメントは、" ボックス " というモデルのトランスフォームをアイデンティティトランスフォームに設定します。

```
-- Lingo
member("3d world").model("Box").transform.identity()

// Javascript
member("3d world").getProp("model",1).transform.identity()
```

このスクリプトでは、" ボックス " が使用可能な最初のモデルであることを前提とします。

関連事項

transform （プロパティ）, getWorldTransform()

idleLoadDone()

使用法

```
-- Lingo syntax
_movie.idleLoadDone(intLoadTag)

// JavaScript syntax
_movie.idleLoadDone(intLoadTag);
```

解説

指定したロードタグを持つすべてのキャストメンバの読み込みが完了した（TRUE）か、完了していない（FALSE）かを返すムービーメソッドです。

パラメータ

intLoadTag 必須。テストするキャストメンバのロードタグを指定する整数です。

例文

次のステートメントは、ロードタグが "20" のキャストメンバがすべて読み込まれているかどうかを調べ、読み込まれている場合はムービー " キオスク " を再生します。

```
-- Lingo syntax
if (_movie.idleLoadDone(20)) then
    _movie.play(1, "Kiosk")
end if

// JavaScript syntax
if (_movie.idleLoadDone(20)) {
    _movie.play(1, "Kiosk");
}
```

関連事項

idleHandlerPeriod, idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, ムービー

ignoreWhiteSpace()

使用法

```
XMLparserObject.ignoreWhiteSpace(trueOrFalse)
```

解説

XML コマンド。Lingo リストを作成するときに、パーサーが余白を無視するか残すかを指定します。ignoreWhiteSpace() がデフォルトの TRUE に設定されると、パーサーは余白を無視します。FALSE に設定されると、パーサーは余白を残し、実際のデータとして処理します。

<sample> </sample> のようにエレメントに個別の開始タグと終了タグがある場合、エレメント内の文字データが余白だけから構成されていれば、そのデータは無視されます。余白以外のスペースがある場合、または ignoreWhiteSpace() が FALSE に設定された場合は、余白を含め実際のテキストを持つ CDATA ノードが存在します。

パラメータ

trueOrFalse 必須。パーサーが余白を無視するか（TRUE）、無視しないか（FALSE）を指定する値です。

例文

次の Lingo ステートメントは、ignoreWhiteSpace() をデフォルトの TRUE に設定されたままにし、所定の XML をリスト内へ構文解析します。エレメント <sample> のリスト内には子がありません。

```
XMLtext = "<sample> </sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:]]]]
```

次の Lingo ステートメントは ignoreWhiteSpace() を FALSE に設定し、次に所定の XML をリスト内へ構文解析します。エレメント <sample> には現在、1 つのスペース文字を含む 1 つの子があります。

```
XMLtext = "<sample> </sample>"
parserObj.ignorewhitespace(FALSE)
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA": " "]]]
```

これらの Lingo ステートメントは、ignoreWhiteSpace() をデフォルトの TRUE に設定されたままにし、所定の XML を構文解析します。<sample> タグの子ノードは 1 つしかなく、また <sub> タグの子ノードも 1 つしかありません。

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "sub": ["!ATTRIBUTES": [:],
"!CHARDATA": " phrase 1 "]]]]
```

これらの Lingo ステートメントは ignoreWhiteSpace() を FALSE に設定し、次に所定の XML を構文解析します。<sample> タグの子ノードは現在 2 つあり、最初の子ノードは単一スペース文字です。

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
gparser.ignoreWhiteSpace(FALSE)
gparser.parseString(XMLtext)
theList = gparser.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA": " ", "sub":
["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

ilk()

使用法

```
ilk(object)
ilk(object, type)
```

解説

オブジェクトのタイプを示す関数です。

オブジェクトの各種類ごとに ilk() 関数が返す値を下表に示します。

オブジェクトのタイプ	ilk (オブジェクト) の戻り値	次の [タイプ] の場合にのみ ilk (オブジェクト , タイプ) の戻り値は 1 になる	例文
リニアリスト	#list	#list or #linearlist	ilk ([1,2,3])
プロパティリスト	#proplist	#list or #proplist	ilk ([#his: 1234, #hers: 7890])
整数	#integer	#integer or #number	ilk (333)
浮動小数点数	#float	#float or #number	ilk (123.456)
ストリング	#string	#string	ilk ("asdf")
rect	#rect	#rect or #list	ilk (sprite(1).rect)
ポイント	#point	#point or #list	ilk (sprite(1).loc)
color	#color	#color	ilk (sprite(1).color)
日付	#date	#date	ilk (the systemdate)
シンボル	#symbol	#symbol	ilk (#hello)
void	#void	#void	ilk (void)
picture	#picture	#picture	ilk (member (2).picture)
親スクリプトインスタンス	#instance	#object	ilk (new (script "blahblah"))
Xtra インスタンス	#instance	#object	ilk (new (xtra "fileio"))
member	#member	#object または #member	ilk (member 1)
Xtra	#xtra	#object または #xtra	ilk (xtra "fileio")
スクリプト	#script	#object または #script	ilk (script "blahblah")
キャスト	#castlib	#object または #castlib	ilk (castlib 1)
スプライト	#sprite	#object または #sprite	ilk (sprite 1)
サウンド	#instance または #sound (Sound Control Xtra が存在しないとき)	#instance または #sound	ilk (sound "yaddayadda")
ウィンドウ	#window	#object または #window	ilk (the stage)
メディア	#media	#object または #media	ilk (member (2).media)
timeout	#timeout	#object または #timeout	ilk (timeOut("intervalTimer"))
image	#image	#object または #image	ilk ((the stage).image)

パラメータ

object 必須。テストするオブジェクトを指定します。

type オプション。**object** の比較対照となるタイプを指定します。オブジェクトが指定したタイプの場合、ilk() 関数は TRUE を返します。オブジェクトが指定したタイプでない場合は、ilk() 関数は FALSE を返します。

例文

次の ilk ステートメントは、"Bids" という名前のオブジェクトのタイプを識別します。

```
Bids = [:]
put ilk( Bids )
-- #proplist
```

次の ilk ステートメントは、変数 Total がリストかどうかを調べ、結果をメッセージウィンドウに表示します。

```
Total = 2+2
put ilk( Total, #list )
-- 0
```

変数 Total はリストでないため、メッセージウィンドウには FALSE を表す数値 0 が表示されます。

次の例は、変数 myVariable が日付であるかどうかを調べ、日付である場合、メッセージウィンドウに表示します。

```
myVariable = the systemDate
if ilk(myVariable, #date) then put myVariable
-- date( 1999, 2, 19 )
```

ilk (3D)

使用法

```
ilk(object)
ilk(object,type)
object.ilk
object.ilk(type)
```

解説

Lingo 関数。オブジェクトのタイプを示します。

3D オブジェクトの各種類ごとに ilk() 関数が返す値を下表に示します。この辞書で説明していない非 3D オブジェクトの戻り値のリストについては、メインの Lingo 辞書を参照してください。

オブジェクトのタイプ	ilk (オブジェクト) の戻り値	ilk(オブジェクト , タイプ) が TRUE を返すときの種類
レンダラサービス	#renderer	#renderer
モデルリソース	#modelresource, #plane, #box, #sphere, #cylinder, #particle, #mesh	ilk(object) の戻り値と同じです。ただし #modelresource は、読み込んだ W3D ファイルで生成されたリソースの ilk です。
モデル	#model	#model
モーション	#motion	#motion または #list
shader	#shader	#shader または #list
テクスチャ	#texture	#texture または #list
グループ	#group	#group
camera	#camera	#camera
衝突データ	#collisiondata	#collisiondata
ベクトル	#vector	#vector
transform	#transform	#transform

パラメータ

object 必須。テストするオブジェクトを指定します。

type オプション。**object** の比較対照となるタイプを指定します。オブジェクトが指定したタイプの場合、ilk() 関数は TRUE を返します。オブジェクトが指定したタイプでない場合、ilk() 関数は FALSE を返します。

例文

次のステートメントは、"MyObject" がモーションオブジェクトであることを示しています。

```
put MyObject.ilk
-- #motion
```

次のステートメントは、"MyObject" がモーションオブジェクトであるかどうか調べます。戻り値が 1 ならば、オブジェクトがモーションオブジェクトであることを示しています。

```
put MyObject.ilk(#motion)
-- 1
```

関連事項

tweenMode

image()

使用法

```
-- Lingo syntax
image(intWidth, intHeight, intBitDepth, <optional:paletteRef>)

// JavaScript syntax
image(intWidth, intHeight, intBitDepth, <optional:paletteRef>);
```

解説

指定されたサイズで新しい画像を作成して返すトップレベル関数です。

トップレベルの image() 関数を使用して新しい画像を作成した場合、その新しい画像は画像データの自立型セットであり、他のすべての画像から独立して存在します。したがって、他の画像に変更を加えても、新しい画像に影響は及びません。

変数をキャストメンバやステージの画像などのソース画像と等価に設定することによって画像を参照する場合、その変数にはソース画像への参照データが含まれます。したがって、ソースオブジェクトまたは変数内の画像に加えた変更は、他方の画像に反映されます。

このビヘイビアーを避け、ソース画像から独立した画像のコピーを作成するには、duplicate() メソッドを使用します。duplicate() メソッドは、ソース画像のコピーを返します。このコピーは、ソース画像のすべての値を継承しますが、ソース画像には結び付けられません。したがって、ソース画像またはソース画像の新しいコピーに加えられた変更は、他方の画像に影響を与えません。

キャストメンバを参照することによって画像オブジェクトを作成する場合、新しいオブジェクトにはそのメンバの画像への参照データが含まれます。画像に加えた変更は、キャストメンバと、そのメンバから作成されたスプライトに反映されます。

新しい画像オブジェクトを作成した場合、バックカラーはデフォルトで白（color(255,255,255)）になり、アルファチャネルは完全に不透明（color(0,0,0)）になります。

100% の透明度のアルファチャネルは白（color(255,255,255)）、100% の不透明度のアルファチャネルは黒（color(0,0,0)）です。

完成したムービーの image() の実例は、Director アプリケーションフォルダの Learning\Lingo フォルダ内の imaging.dir を参照してください。

パラメータ

intWidth 必須。新しい画像の幅を指定する整数です。

intHeight 必須。新しい画像の高さを指定する整数です。

intBitDepth 必須。新しい画像のビット深度を指定する整数です。1、2、4、8、16、32 が値として有効です。

例文

次の例は、幅 200 ピクセル、高さ 200 ピクセルの 8 ビット画像を作成します。

```
-- Lingo syntax
objImage = image(200, 200, 8)

// JavaScript syntax
var objImage = image(200, 200, 8);
```

次の例は、ステージの画像を参照することによって画像を作成します。

```
-- Lingo syntax
objImage = _movie.stage.image

// JavaScript syntax
var objImage = _movie.stage.image;
```

関連事項

duplicate() （画像）, fill(), image （画像）

importFileInto()

使用法

```
-- Lingo syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList)

// JavaScript syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList);
```

解説

指定したキャストメンバの内容を指定したファイルに置き換えるためのメンバメソッドです。

importFileInto() メソッドは、以下の状況で有益です。

- ムービーを終了または開発するとき、外部のリンクメディアを埋め込み、プロジェクト中に編集できるようにします。
- ムービー作成中で Lingo または JavaScript のシンタックスからスコアを生成するとき、新しいキャストメンバにコンテンツを割り当てます。
- インターネットからファイルをダウンロードするとき、特定の URL のファイルをダウンロードし、リンクされたメディアのファイル名を設定します。

  注意：URL からファイルを読み込むときは、preloadNetThing() を使用してファイルをローカルディスクにダウンロードし、ローカルディスクからファイルを読み込む方が効率的です。preloadNetThing() を使用すると、ダウンロードの潜在的な問題を最小限に抑えることもできます。
- RTF や HTML のドキュメントを書式やリンク情報と共にテキストキャストメンバに取り込みます。

importFileInto() をプロジェクタで使用すると急速にメモリが消費されるため、読み込んだデータにはできるだけ同じメンバを使用します。

Director やプロジェクタでは、importFileInto() は自動的にファイルをダウンロードします。Shockwave Player では、preloadNetThing() を呼び出してダウンロード終了まで待ち、それから importFileInto() でファイルを読み込みます。

パラメータ

fileOrUrlString 必須。キャストメンバの内容を置き換えるファイルを指定するストリングです。

propertyList は省略可能であり、以下のプロパティをサポートします。

#dither：0; ディザしない（デフォルト）

#dither：ゼロ以外 ; ディザする

#trimWhiteSpace：0; 画像の外側のエッジの白いスペースをトリムしない

#trimWhiteSpace：ゼロ以外 ; 白いスペースをトリムする（デフォルト）

#linked：0; 内部メンバとして読み込む（デフォルト）

#linked：ゼロ以外 ; リンクメンバとして読み込む

#remapImageToStage：0; 画像の独自の深度を使用する

#remapImageToStage：ゼロ以外 ; 画像をステージ深度にリマップする（デフォルト）

例文

次のハンドラは、GIF ファイルを含む URL を変数 "tempURL" に割り当ててから、importFileInto コマンドを使って、URL にあるそのファイルを新規ビットマップキャストメンバとして読み込みます。

```
-- Lingo syntax
on exitFrame
    tempURL = "http://www.dukeOfUrl.com/crown.gif"
    _movie.newMember(#bitmap).importFileInto(tempURL)
end

// JavaScript syntax
function exitFrame() {
    var tempURL = "http://www.dukeOfUrl.com/crown.gif";
    _movie.newMember("bitmap").importFileInto(tempURL);
}
```

次のステートメントは、" 追憶 " サウンドキャストメンバの内容を " 風 .wav" サウンドファイルに置き換えます。

```
-- Lingo syntax
member("Memory").importFileInto("Wind.wav")

// JavaScript syntax
member("Memory").importFileInto("Wind.wav");
```

次のステートメントは、Director の入っているフォルダに URL から外部ファイルをダウンロードし、そのファイルをサウンドキャストメンバ " ノーマ・デズモンドが語る " として読み込みます。

```
-- Lingo syntax
downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF", _player.applicationPath & "Talkies.AIF")
member("Norma Desmond Speaks").importFileInto(_player.applicationPath & "Talkies.AIF")

// JavaScript syntax
downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF",_player.applicationPath + "Talkies.AIF");
member("Norma Desmond Speaks").importFileInto(_player.applicationPath +"Talkies.AIF");
```

関連事項

downloadNetThing, fileName （ウィンドウ）, メンバ, preloadNetThing()

Initialize

使用法

```
Initialize (MUIObject, initialPropertyList)
```

解説

このコマンドは、MUI Xtra のインスタンスから汎用目的のダイアログボックスを設定します。initialPropertyList はダイアログボックスの属性の定義を取得する場所を指定します。

- #windowPropList プロパティに関連付けられたプロパティリストは、Director が全般ダイアログボックスの属性の定義として使用するリストです。
- #windowItemList プロパティに関連付けられたリニアリストは、Director が個々のコンポーネントの定義として使用するリストです。リスト内の各項目は 1 つのコンポーネントを定義するプロパティリストです。

例文

次のステートメントは、MUI Xtra のインスタンスである MUIObject から作成された汎用目的ダイアログボックスを初期化します。aWindowPropList リストには全般ダイアログボックス用の定義が含まれています。aWindowItemList リストには、ダイアログボックスの個々のコンポーネント用の定義が含まれています。

```
-- Lingo
Initialize(MUIObj, [#windowPropList:aWindowPropList, \
#windowItemList:aWindowItemList])
```

insertBackdrop

使用法

```
sprite(whichSprite).camera{(index)}.insertBackdrop(index, texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).insertBackdrop(index, texture, locWithinSprite, rotation)
```

解説

3D カメラコマンド。カメラの背景リストの指定した位置に、背景を追加します。

パラメータ

index 必須。カメラの背景リスト内の、背景を追加するインデックス位置を指定します。

texture 必須。追加する背景のテクスチャを指定します。

locWithinSprite 必須。3D スプライト内の背景が表示される 2D の位置。この位置は、スプライトの左上隅から測定されます。

rotation オプション。追加する背景の回転を指定します。

例文

次の例の 1 行目は、" 杉 " というテクスチャを作成します。2 行目は、スプライト "5" のカメラの背景リストの最初の位置にテクスチャを挿入します。背景の位置は、スプライトの左上隅から測定して (300, 120) の位置になります。回転は 45° です。

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertBackdrop(1, t1, point(300, 120), 45)

// Javascript
Var t1= member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertBackDrop(1,t1,point(300,120),45);
```

関連事項

removeBackdrop, bevelDepth, overlay, backdrop

insertFrame()

使用法

```
-- Lingo syntax
_movie.insertFrame()

// JavaScript syntax
_movie.insertFrame();
```

解説

現行フレームとその内容を複製するムービーメソッドです。

複製されたフレームは現行フレームの後に挿入され、現行フレームになります。

このメソッドは、スコア録画セッション中においてのみ使用することができます。このコマンドと duplicateFrame() メソッドの結果は同じになります。

パラメータ

なし

例文

次のハンドラは、トランジションチャネルに割り当てられたトランジションキャストメンバ "Fog" を含む 1 つのフレームを生成します。生成したフレームの後には、一連の空白フレームが続いて生成されます。後続のフレーム数は numberOfFrames パラメータで指定します。

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    _movie.frameTransition = member("Fog").number
    _movie.go(_movie.frame + 1)
    repeat with i = 0 to numberOfFrames
        _movie.insertFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    _movie.frameTransition = member("Fog").number;
    _movie.go(_movie.frame + 1);
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.insertFrame();
    }
    _movie.endRecording();
}
```

関連事項

duplicateFrame(), ムービー

insertOverlay

使用法

```
sprite(whichSprite).camera{(index)}.insertOverlay(index, texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).insertOverlay(index, texture, locWithinSprite, rotation)
```

解説

3D カメラコマンド。カメラのオーバーレイリストの指定した位置に、オーバーレイを追加します。

パラメータ

index 必須。カメラのオーバーレイリスト内の、オーバーレイを追加するインデックス位置を指定します。

texture 必須。追加するオーバーレイのテクスチャを指定します。

locWithinSprite 必須。3D スプライト内のオーバーレイが表示される 2D の位置。この位置は、スプライトの左上隅から測定されます。

rotation オプション。追加するオーバーレイの回転を指定します。

例文

次の例の 1 行目は、" 杉 " というテクスチャを作成します。2 行目は、スプライト "5" のカメラのオーバーレイリストの最初の位置にテクスチャを挿入します。オーバーレイの位置は、スプライトの左上隅から測定して (300, 120) の位置になります。回転は 45° です。

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertOverlay(1, t1, point(300, 120), 45)

// Javascript
Var t1=member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertOverlay(1,t1,point(300,120),45);
```

関連事項

removeOverlay, overlay, bevelDepth

inside()

使用法

```
point.inside(rectangle)
inside(point, rectangle)
```

解説

指定したポイントが指定した矩形の内側にあるか（TRUE）、矩形の外側にあるか（FALSE）を示す関数です。

パラメータ

rectangle 必須。テストするポイントを含む矩形を指定します。

例文

次のステートメントは、Center ポイントが Zone 矩形の内側にあるかどうかを調べ、結果をメッセージウィンドウに表示します。

```
-- Lingo
put Center.inside(Zone)

// Javascript
trace(Center. inside(Zone));
```

関連事項

map(), mouseH, mouseV, point()

installMenu

使用法

```
installMenu whichCastMember
```

解説

プレーヤメソッド。**whichCastMember** で指定したフィールドキャストメンバ内で定義されているメニューをインストールします。これらのカスタムメニューは、ムービーの再生時に表示されます。カスタムメニューを削除するには、パラメータを指定せずに installMenu コマンドを使うか、またはパラメータを 0 に設定して installMenu コマンドを使います。このコマンドは階層メニューでは使用できません。

フィールドキャストメンバ内でのメニュー項目の定義方法については、menu キーワードを参照してください。

システムリソースに影響を及ぼすため、メニューを何度も変更しないようにしてください。

メニューが画面より長い場合、Windows ではメニューの一部のみが表示されます。Mac ではメニューをスクロールすることができます。

注意：メニューは、Shockwave Player では使用できません。

パラメータ

fieldMemberObjRef オプション。メニューがインストールされるフィールドキャストメンバを指定します。

例文

次のステートメントは、フィールドキャストメンバ 37 に定義されているメニューをインストールします。

```
installMenu 37
```

次のステートメントは、フィールドキャストメンバ " メニューバー " に定義されているメニューをインストールします。

```
installMenu member "Menubar"
```

次のステートメントは、installMenu コマンドでインストールしたメニューをオフにします。

```
installMenu 0
```

関連事項

menu

integer()

使用法

```
(numericExpression).integer
integer(numericExpression)
```

解説

式の値を整数に四捨五入する関数です（Lingo のみ）。

string() 関数を使って、整数を強制的にストリングにすることもできます。

JavaScript シンタックスでは、parseInt() 関数を使用します。

パラメータ

numericExpression 必須。整数に四捨五入する数値です。

例文

次のステートメントは、3.75 という値を四捨五入して整数にします。

```
put integer(3.75)
-- 4
```

次のステートメントは、括弧内の値を四捨五入して、locH スプライトプロパティで使えるように整数値にします。

```
sprite(1).locH = integer(0.333 * stageWidth)
```

関連事項

float(), string()

integerP()

使用法

```
expression.integerP
(numericExpression).integerP
integerP(expression)
```

解説

指定した式が整数として評価できる（1 または TRUE）かできない（0 または FALSE）かを調べる関数です（Lingo のみ）。integerP の **P** は、**predicate**(属性) を意味しています。

パラメータ

expression 必須。検査する式。

例文

次のステートメントは、3 が整数と評価されるかどうかを調べます。3 は整数であるため、メッセージウィンドウには TRUE を表す値 1 が表示されます。

```
put(3).integerP
-- 1
```

次のステートメントは、"3" が整数と評価されるかどうかを調べます。二重引用符で囲まれている 3 はストリングになり、整数としては認識されないため、メッセージウィンドウには FALSE を表す値 0 が表示されます。

```
put("3").integerP
-- 0
```

次のステートメントは、フィールドキャストメンバ " エントリ " 内のストリングの値が整数かどうかを調べ、整数でない場合は警告を表示します。

```
if field("Entry").value.integerP = FALSE then alert "Please enter an integer."
```

関連事項

floatP(), integer(), ilk(), objectP(), stringP(), symbolP()

Interface()

使用法

```
xtra("XtraName").Interface()
Interface(xtra "XtraName")
```

解説

Xtra を説明し、そのメソッドをリストするストリングを、Return で区切ったストリングとして返す関数です。この関数は、以前のバージョンの mMessageList 関数に代わるものです。

パラメータ

なし

例文

次のステートメントは、QuickTime Asset Xtra で使われている関数の出力をメッセージウィンドウに表示します。

```
-- Lingo
put Xtra("QuickTimeSupport").Interface()

// Javascript
trace(xtra("QuickTimeSupport").Interface());
```

interpolate()

使用法

```
transform1.interpolate(transform2,percentage)
```

解説

3D transform メソッド。トランスフォーム **1** の位置および回転とトランスフォーム **2** の位置および回転を、指定したパーセンテージで補間して作成した、トランスフォーム **1** のコピーを返します。オリジナルのトランスフォーム **1** は変化しません。トランスフォーム **1** を補間するには、interpolateTo() を使用します。

手動で補間するには、2 つの数の差とパーセンテージを乗算します。例えば、4 と 8 を 50% で補間すると 6 になります。

例文

次の例で、"tBox" は "Box" というモデルのトランスフォームです。また、"tSphere" は "Sphere" というモデルのトランスフォームです。例の 3 行目は、"Box" のトランスフォームと "Sphere" のトランスフォームを中間で補間し、そのコピーを tNew に返します。

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tNew = tBox.interpolate(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model",a).transform;
// where a is the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model", i).transform;
// where i is the number index for the model "Sphere"
var tNew = tBox.interpolate(tSphere, 50);
```

関連事項

interpolateTo()

interpolateTo()

使用法

```
transform1.interpolateTo(transform2, percentage)
```

解説

3D transform メソッド。トランスフォーム **1** の位置および回転と新しいトランスフォームの位置および回転を、指定したパーセンテージで補間することにより、トランスフォーム **1** を変更します。オリジナルのトランスフォーム **1** が変更されます。トランスフォーム **1** のコピーを補間するには、interpolate() 関数を使用します。

手動で補間するには、2 つの数の差とパーセンテージを乗算します。例えば、4 と 8 を 50% で補間すると 6 になります。

パラメータ

transform2 必須。指定したトランスフォームを補間するトランスフォームを指定します。

percentage 必須。トランスフォーム **2** の回転のパーセンテージを指定します。

例文

次の例で、"tBox" は "Box" というモデルのトランスフォームです。また、"tSphere" は "Sphere" というモデルのトランスフォームです。例の 3 行目は、"Box" のトランスフォームと "Sphere" のトランスフォームを中間で補間します。結果は、tBox に反映されます。

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tBox.interpolateTo(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model", i).transform;
// where i the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model",j).transform;
// where j is the number index for the model "Sphere"
tBox.interpolateTo(tSphere, 50);
```

関連事項

interpolate()

intersect()

使用法

```
rectangle1. Intersect(rectangle2)
intersect(rectangle1, rectangle2)
```

解説

2 つの矩形が交差する部分で形成される矩形を定義する関数です。

パラメータ

rectangle2 必須。交差テストの 2 番目の矩形を指定します。

例文

次のステートメントは、矩形 toolKit が矩形 Ramp と交差する部分に形成される矩形を、変数 newRectangle に割り当てます。

```
-- Lingo
newRectangle = toolKit.intersect(Ramp)

// Javascript
newRectangle = toolKit.intersect(Ramp);
```

関連事項

map(), rect(), union()

inverse()

使用法

```
member(whichCastmember).model(whichModel).transform.inverse()
member(whichCastmember).group(whichGroup).transform.inverse()
member(whichCastmember).camera(whichCamera).transform.inverse()
sprite(whichSprite).camera{(index)}.transform.inverse()
member(whichCastmember).light(whichLight).transform.inverse()
transformReference.inverse()
```

解説

3D transform メソッド。位置プロパティと回転プロパティを反転したトランスフォームのコピーを返します。

このメソッドは、オリジナルのトランスフォームは変更しません。オリジナルのトランスフォームを反転するには、invert() 関数を使用します。

パラメータ

なし

例文

次のステートメントは、" 椅子 " というモデルのトランスフォームのコピーを反転します。

```
-- Lingo
boxInv = member("3d world").model("Chair").transform.inverse()

// Javascript
var boxInv = member("3d world").getPropRef("model", a).transform.inverse();
// where a the number index for the model "Chair"
```

関連事項

invert()

invert()

使用法

```
member(whichCastmember).model(whichModel).transform.invert()
member(whichCastmember).group(whichGroup).transform.invert()
member(whichCastmember).camera(whichCamera).transform.invert()
sprite(whichSprite).camera{(index)}.transform.invert()
member(whichCastmember).light(whichLight).transform.invert()
transformReference.invert()
```

解説

3D transform メソッド。トランスフォームの位置プロパティと回転プロパティを反転します。

このメソッドは、オリジナルのトランスフォームを変更します。オリジナルのトランスフォームのコピーを反転するには、inverse() 関数を使用します。

パラメータ

なし

例文

```
-- Lingo
member("3d world").model("Box").transform.invert()

// Javascript
member("3d world").getPropRef("model", a).transform.invert();
// where a the number index for the model "Box"
```

関連事項

inverse()

isBusy()

使用法

```
-- Lingo syntax
soundChannelObjRef.isBusy()

// JavaScript syntax
soundChannelObjRef.isBusy();
```

解説

サウンドチャネル内でサウンドが再生されているかどうかを示すサウンドチャネルメソッドです。されている場合は TRUE、されていない場合は FALSE になります。

isBusy() を使用してサウンドチャネルをチェックする際には、再生ヘッドが動いたことを確認します。サウンド再生後もこの関数が FALSE を返し続ける場合は、再生ヘッドが動く前に updateStage() メソッドを使用してサウンドの再生を開始します。

このメソッドは実際のオーディオキャストメンバで占められているサウンドチャネルに有効です。QuickTime、Flash、および Shockwave Player オーディオなどのメディアタイプはサウンドの処理の仕方が異なるので、このメソッドは無効です。

isBusy() の代わりに、サウンドチャネルの status プロパティを使用することを考慮してください。status プロパティは、多くの状況で、より正確な状態が得られます。

パラメータ

なし

例文

次のステートメントは、サウンドチャネル 1 でサウンドが再生されているかどうかをチェックし、再生されている場合はそのフレーム内でループさせます。これにより、サウンドチャネル 1 の再生が終了するまで再生ヘッドは次のフレームに進みません。

```
-- Lingo syntax
if (sound(1).isBusy()) then
    _movie.go(_movie.frame)
end if

// JavaScript syntax
if (sound(1).isBusy()) {
    _movie.go(_movie.frame);
}
```

関連事項

status, サウンドチャネル

isCharSetInstalled

使用法

```
_system.isCharSetInstalled(strCharSet)
```

解説

文字セットメソッド。指定した文字セットがコンピュータにインストールされているかどうかを調べます。このメソッドはブール値を返します。

パラメータ

パラメータ	解説	必須 / オプション
strCharSet	確認する文字セット。例えば、"windows-1252" などです。	必須

例文

```
--Lingo syntax
put _system.isCharSetInstalled("windows-1252")

//JavaScript syntax
put(_system.isCharSetInstalled("windows-1252"));
```

isInWorld()

使用法

```
member(whichCastmember).model(whichModel).isInWorld()
member(whichCastmember).camera(whichCamera).isInWorld()
member(whichCastmember).light(whichLight).isInWorld()
member(whichCastmember).group(whichGroup).isInWorld()
```

解説

3D コマンド。モデル、カメラ、ライト、またはグループの親階層がワールド内で終わっている場合に TRUE の値を返します。isInWorld の値が TRUE の場合、モデル、カメラ、ライト、またはグループは、キャストメンバの 3D ワールド内で機能します。

モデル、カメラ、ライト、およびグループは 3D キャストメンバに格納できますが、キャストメンバの 3D ワールドでは使用できません。キャストメンバの 3D ワールドに対して、モデル、カメラ、ライト、およびグループを追加または削除するには、addToWorld コマンドおよび removeFromWorld コマンドを使用します。

パラメータ

なし

例文

次のステートメントは、"Teapot" というモデルが "TableScene" というキャストメンバの 3D ワールドに存在していることを示します。

```
--Lingo
put member("TableScene").model("Teapot").isInWorld()

// Javascript
put member("TableScene").getPropRef("model", a).isInWorld() ;
// where a is the member index for the Teapot model.
-- 1
```

関連事項

addToWorld, removeFromWorld, child (3D)

isPastCuePoint()

使用法

```
-- Lingo syntax
spriteObjRef.isPastCuePoint(cuePointID)

// JavaScript syntax
spriteObjRef.isPastCuePoint(cuePointID);
```

解説

メディア内の特定のキューポイントをスプライトまたはサウンドチャネルが通過したかどうかを調べるための関数です。この関数は、サウンド (WAV、AIFF、SND、SWA、AU)、QuickTime、またはキューポイントをサポートする Xtra ファイルで使用できます。

スプライトまたはサウンドには [スプライト番号] または [チャネル番号] を指定します。Shockwave Audio（SWA）サウンドはスプライトチャネルにスプライトとして表示されますが、そのサウンドはサウンドチャネルで再生されます。SWA サウンドスプライトは、サウンドチャネル番号ではなく、スプライトチャネル番号で参照することをお勧めします。

[キューポイント **ID**] は、キューポイントの参照で置き換えます。

- [キューポイント **ID**] が整数の場合、isPastCuePoint は、キューポイントを通過しているときは 1 を、通過していないときは 0 を返します。
- [キューポイント **ID**] が名前の場合、isPastCuePoint は、その名前を持つキューポイントを通過した回数を返します。

[キューポイント **ID**] に指定した値がスプライトまたはサウンドに存在しない場合、0 を返します。

isPastCuePoint が返す数値は、メディア内のキューの絶対位置を基準にしています。例えば、サウンドが " メイン " キューポイントを通過してからループして、再度 " メイン " キューポイントを通過した場合には、isPastCuePoint は 2 ではなく 1 を返します。

isPastCuePoint の値をブール値として扱う場合、[キューポイント **ID**] で示されるキューポイントを通過した場合は TRUE、通過していない場合は FALSE を返します。

パラメータ

cuePointID 必須。指定されたキューポイントの名前または番号を指定するストリングまたは整数です。

例文

次のステートメントは、" コーラス終了 " キューポイントを 3 回通過するまでサウンドを再生します。

```
-- Lingo syntax
if (sound(1).isPastCuePoint("Chorus End")=3) then
    sound(1).stop()
end if

// JavaScript syntax
var ce = sound(1).isPastCuePoint("Chorus End");
if (ce = 3) {
    sound(1).stop();
}
```

次の例は、サウンドチャネル 1 で再生している音楽に関する情報をキャストメンバ " フィールド 2" に表示します。音楽がまだキューポイント " クライマックス " を過ぎていない場合は、" フィールド 2" のテキストは " 先頭 " になります。そうでない場合は、" 終端 " になります。

```
--- Lingo syntax
if not sound(1).isPastCuePoint("climax") then
    member("field 2").text = "This is the beginning of the piece."
else
    member("field 2").text = "This is the end of the piece."
end if

// JavaScript syntax
var cmx = sound(1).isPastCuePoint("climax");
if (cmx != 1) {
    member("field 2").text = "This is the beginning of the piece.";
} else {
    member("field 2").text = "This is the end of the piece.";
}
```

ItemUpdate()

使用法

```
ItemUpdate(MUIObject, itemNumber, itemInputPropList)
```

解説

次のコマンドは、汎用目的ダイアログボックスのコンポーネントを更新します。ダイアログボックスが表示されている間にユーザのアクションに応じてダイアログボックスを更新するときに有用です。

- itemNumber は更新対象の項目の数を表します。
- itemInputPropList は、項目の新しいプロパティのリストを表します。

ItemUpdate コマンドは、多くの目的に使用できます。可能な使い方として、ボタンの有効化と無効化、ポップアップの範囲の変更、スライダ位置の更新、ユーザが無効な値を入力した場合の編集可能なテキスト項目の更新があります。

ダイアログボックスの個々の項目をユーザ入力やユーザ操作に応じて更新したり、基本のデータを表示したりできます。一般的には項目の #value を更新しますが、項目に関するその他すべても更新できます（ただしタイプを除く）。height、width、locH、および locV プロパティを -1 に設定して現行の値を維持します。

例文

以下のステートメントは、itemNum 値を持つダイアログボックスコンポーネントを更新します。

- 最初のステートメントは、項目定義の全体リストからコンポーネントの定義を取り込みます。
- 2 番目と 3 番目のステートメントは、コンポーネントの種類と属性プロパティを修正します。
- 最後のステートメントは、ItemUpdate コマンドを使用してコンポーネントの設定を更新します。

```
--Lingo
set baseItemList = getAt ( theItemList, itemNum )
set the type of baseItemList = #IntegerSliderH
set the attributes of baseItemList = [#valueRange :[#min:1, #max:8, #increment:1,
#jump:1, #acceleration:1]
ItemUpdate(MUIObj, itemNum, baseItemList)
on smileyUpdate
    -- declare globals
    global smileyIndex, gMuiSmileDialObj, itemNumSmile, itemNumSlide, smileItemList
-- validate dialog object
if ( objectP ( gMuiSmileDialObj ) ) then
    -- get a list to put in new/updated values
    set baseItemList = duplicate ( getAt ( smileItemList,itemNumSmile ) )
    -- metrics can be set to -1, this "keeps them the same"
    -- instead of updating.
    -- could also be set to a new value if you
    -- wanted to resize the item or relocate it.
    set the width of baseItemList = -1 -- keep previous
    set the height of baseItemList = -1 -- keep previous
    set the locH of baseItemList = -1 -- keep previous
    set the locV of baseItemList = -1 -- keep previous
    -- in this particular case, the value is
    -- the only thing that's changing
    set the value of baseItemList = string(smileyIndex)
    -- member name
    -- tell the dialog to update the item number
    -- with the new item list
    ItemUpdate(gMuiSmileDialObj, itemNumSmile, baseItemList)
    end if
end
```

keyPressed()

使用法

```
-- Lingo syntax
_key.keyPressed({keyCodeOrCharacter})

// JavaScript syntax
_key.keyPressed({keyCodeOrCharacter});
```

解説

最後に押されたキーに割り当てられている文字ストリング、または指定されたキーが押されたかどうか（オプション）を返すキーメソッドです。

keyCodeOrCharacter パラメータを省略した場合、このメソッドは最後に押されたキーに割り当てられている文字ストリングを返します。キーが押されていない場合、このメソッドは空のストリングを返します。

keyCodeOrCharacter を使用して押されるキーを指定した場合、そのキーが押されていれば TRUE を返し、押されていなければ FALSE を返します。

このメソッドは、repeat（Lingo）または for（JavaScript syntax）ループ内でユーザがキーを押したときに更新されます。この点では、repeat または for ループ内で更新されない key プロパティよりも優れています。

異なるキーボードでどの文字がどのキーに対応するのかをテストするには、「キーボードの Lingo」サンプルムービーを使用してください。

パラメータ

keyCodeOrCharacter オプション。テストするキーコードまたは ASCII 文字ストリングです。

例文

次のステートメントは、ユーザが Enter キー（Windows）または Return キー（Mac）を押したかどうかを調べ、押している場合には updateData ハンドラを実行します。

```
-- Lingo syntax
if (_key.keyPressed(RETURN)) then
    updateData
end if

// JavaScript syntax
if (_key.keyPressed(36)) {
    updateData();
}
```

次のステートメントは、**a** キーの keyCode を使用して、キーが押されているかどうかを調べ、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
if (_key.keyPressed(0)) then
    put("The key is down")
end if

// JavaScript syntax
if (_key.keyPressed(0)) {
    put("The key is down");
}
```

次のステートメントは、ASCII 文字列を使用して、"**a**" キーと "**b**" キーが押されているかどうかを調べ、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
if (_key.keyPressed("a") and _key.keyPressed("b")) then
    put("Both keys are down")
end if

// JavaScript syntax
if (_key.keyPressed("a") && _key.keyPressed("b")) {
    put("Both keys are down");
}
```

関連事項

キー , key, keyCode

label()

使用法

```
-- Lingo syntax
_movie.label(stringMarkerName)

// JavaScript syntax
_movie.marker(stringMarkerName);
```

解説

マーカラベルが設定されているフレームを示すムービーメソッドです。

パラメータ **stringMarkerName** は、現行ムービー内のラベルである必要があります。そうでない場合、このメソッドは 0 を返します。

パラメータ

stringMarkerName 必須。フレームに関連するマーカラベルの名前を指定するストリングです。

例文

次のステートメントは、" 開始 " というラベルが付いたフレームから 10 フレーム進んだフレームまで再生ヘッドを移します。

```
-- Lingo syntax
_movie.go(_movie.label("Start") + 10)

// JavaScript syntax
_movie.go(_movie.marker("Start") + 10);
```

関連事項

frameLabel, go(), labelList, ムービー

last()

使用法

```
the last chunk of (chunkExpression)
the last chunk in (chunkExpression)
```

解説

チャンク式内の最後のチャンクを識別する関数です。

チャンク式には、文字のコンテナとなる文字、単語、項目、行などを指定できます。このコンテナとは、フィールドキャストメンバ、ストリングを含む変数、特定の文字、単語、項目、行、コンテナ内の特定範囲などを指します。

パラメータ

chunkExpression 必須。最後のチャンクを含むチャンク式を指定します。

例文

次のステートメントは、"Adobe, the multimedia company" というストリングの最後の単語を識別し、結果をメッセージウィンドウに表示します。

```
-- Lingo
put the last word of "Adobe, the multimedia company"
```

結果は、**company** になります。

次のステートメントは、"Adobe, the multimedia company" というストリングの最後の文字を識別し、結果をメッセージウィンドウに表示します。

```
put the last char of ("Adobe, the multimedia company")
```

結果は、**y** になります。

関連事項

char...of, word...of

lastClick()

使用法

```
the lastClick
```

解説

最後にマウスボタンが押されたときから経過した時間をチック数（1/60 秒）で返す関数です。

この関数は、値を調べることはできますが、変更はできません。

パラメータ

なし

例文

次のステートメントは、最後にマウスがクリックされてから 10 秒以上経過しているかどうかを調べ、経過している場合は再生ヘッドを " クリックなし " マーカまで移動します。

```
if the lastClick > 10 * 60 then go to "No Click"
```

関連事項

lastEvent(), lastKey, lastRoll, milliseconds

lastEvent()

使用法

```
the lastEvent
```

解説

マウスクリック、ロールオーバー、キー操作などのイベントが最後に発生してから経過した時間をチック数 (1/60 秒) で返す関数です。

パラメータ

なし

例文

次のステートメントは、最後にマウスクリック、ロールオーバー、キー操作などの操作が行われてから 10 秒経過しているかどうかを調べ、経過している場合は再生ヘッドを " ヘルプ " マーカまで移動します。

```
if the lastEvent > 10 * 60 then go to "Help"
```

関連事項

lastClick(), lastKey, lastRoll, milliseconds

length()

使用法

```
string.length
length(string)
```

解説

[ストリング] で指定したストリングの文字数を返す関数です。TAB、RETURN などの制御文字やスペースも 1 文字として数えられます。

パラメータ

なし

例文

次のステートメントは、"Macro" と "media" というストリングの文字数を表示します。

```
put ("Macro" & "media").length
-- 10
```

次のステートメントは、フィールドキャストメンバ " ファイル名 " に 31 文字以上入力されているかどうかを調べ、入力されている場合はアラートを表示します。

```
-- Lingo syntax
if member("Filename").text.length > 31 then
    alert "That filename is too long."
end if

// JavaScript syntax
if (member("Filename").text.length > 31) {
    _player.alert("That filename is too long.");
}
```

関連事項

chars(), offset() (string function)

light()

使用法

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

解説

3D エレメント。ライトの放出元のベクトル位置にあるオブジェクトです。

ライトのプロパティおよびコマンドの完全な一覧については、Director のヘルプパネルの「Director ユーザガイド」を参照してください。

例文

次の例は、ライトを参照する 2 種類の方法を示します。1 行目では括弧内のストリングを使用し、2 行目では角括弧内の数字を使用します。ストリングはライトの名前で、数字はライトのキャストメンバリスト内でのライトの位置です。

```
-- Lingo
thisLight = member("3D World").light("spot01")
thisLight = member("3D World").light[2]

// Javascript
thisLight = member("3D World").light[1];
```

関連事項

newLight, deleteLight

lineHeight()

使用法

```
-- Lingo syntax
memberObjRef.lineHeight(lineNumber)

// JavaScript syntax
memberObjRef.lineHeight(lineNumber);
```

解説

指定したフィールドキャストメンバ内の指定行の高さをピクセル数で返す関数です。

パラメータ

lineNumber 必須。測定する行を指定する整数です。

例文

次のステートメントは、" 今日のニュース " フィールドキャストメンバの先頭行の高さをピクセル数で求め、その値を headline 変数に割り当てます。

```
--Lingo syntax
headline = member("Today's News").lineHeight(1)

// JavaScript syntax
var headline = member("Today's News").lineHeight(1);
```

linePosToLocV()

使用法

```
-- Lingo syntax
memberObjRef.linePosToLocV(lineNumber)

// JavaScript syntax
memberObjRef.linePosToLocV(lineNumber);
```

解説

フィールドキャストメンバの上端から指定の行までの距離をピクセル数で返す関数です。

パラメータ

lineNumber 必須。測定する行を指定する整数です。

例文

次のステートメントは、" 今日のニュース " フィールドキャストメンバの上端から 2 行目までの距離を測定し、結果を startOfString 変数に割り当てます。

```
--Lingo syntax
startOfString = member("Today's News").linePosToLocV(2)

// JavaScript syntax
var startOfString = member("Today's News").linePosToLocV(2);
```

linkAs()

使用法

```
castMember.linkAs()
```

解説

保存ダイアログボックスを開いて、スクリプトの内容を外部ファイルに保存できるようにするスクリプトキャストメンバ関数です。保存すると、スクリプトキャストメンバはそのファイルにリンクされます。

リンクしたスクリプトは、プロジェクタまたは Shockware コンテンツを含むムービーとして保存すると、ムービーに読み込まれます。これ以外のリンクしたメディアでは、明示的に読み込まない限り外部的なメディアのままなので、これとは異なります。

パラメータ

なし

例文

次のステートメントをメッセージウィンドウに入力すると、[保存] ダイアログボックスが開いて、スクリプト " ランダムモーション " を外部ファイルとして保存できます。

```
-- Lingo
member("Random Motion").linkAs()
importFileInto, linked

// Javascript
member("Random Motion").linkAs() ;
```

list()

使用法

```
-- Lingo syntax
list()
[]
list(stringValue1, stringValue2, ...)
[stringValue1, stringValue2, ...]

// JavaScript syntax
list();
list(stringValue1, stringValue2, ...);
```

解説

リニアリストを作成するトップレベル関数です。

パラメータを使用するか否かに関係なく、シンタックス list() を使用してリストを作成した場合、リスト値のインデックスは 1 から始まります。

パラメータを使用するか否かに関係なく、シンタックス [] を使用してリストを作成した場合、リスト値のインデックスは 0 から始まります。

実行可能なスクリプトの 1 行の長さは 256 文字までに制限されているため、list() を使用して大きなリストを作成することはできません。大量のデータを使用してリストを作成するには、データを角括弧（[]）で囲んでフィールドに入れ、そのフィールドを変数に割り当てます。変数の内容がデータのリストになります。

パラメータ

strigValue1, stringValue2 ... オプション。リスト内の内部値を指定するストリングのリストです。

例文

次のステートメントは、designers 変数に " ジー "、" ケイン "、" オーハシ " という値で構成されたリニアリストを割り当てます。

```
-- Lingo syntax
designers = list("Gee", "Kayne", "Ohashi") -- using list()
designers = ["Gee", "Kayne", "Ohashi"] -- using brackets

// JavaScript syntax
var designers = list("Gee", "Kayne", "Ohashi");
```

関連事項

propList()

listP()

使用法

```
listP(item)
```

解説

指定した項目がリスト、矩形、ポイントのいずれかであるか（1 または TRUE）、それ以外であるか（0 または FALSE）を示す関数です。

パラメータ

item 必須。テストする項目を指定します。

例文

次のステートメントは、designers 変数内のデータの形式がリスト、矩形、ポイントのいずれかであるかを調べ、結果をメッセージウィンドウに表示します。

```
put listP(designers)
```

結果は、TRUE を表す数値の 1 になります。

関連事項

ilk(), objectP()

loadFile()

使用法

```
member(whichCastmember).loadFile(fileName {, overwrite, generateUniqueNames})
```

解説

3D キャストメンバコマンド。W3D ファイルのアセットをキャストメンバに読み込みます。

loadFile コマンドを使用する前に、キャストメンバの state プロパティを -1 (エラー) または 4 (ロード) のいずれかに設定する必要があります。

パラメータ

fileName 必須。読み込むアセットを含む W3D ファイルを指定します。

overwrite オプション。W3D ファイルのアセットがキャストメンバのアセットを置き換えるか（TRUE）、またはキャストメンバのアセットに追加されるか（FALSE）を示します。[上書き] パラメータのデフォルト値は TRUE です。

generateUniqueNames オプション。TRUE に設定されている場合、キャストメンバ内のエレメントと同名の W3D ファイル内エレメントは、名前が変更されます。FALSE に設定されている場合、キャストメンバ内のエレメントは、同名の W3D 内エレメントで上書きされます。[生成される一意の名前] パラメータのデフォルト値は TRUE です。

例文

次のステートメントは、"Truck.W3d" というファイルの内容を " 道路 " というキャストメンバに読み込みます。"Truck.W3d" の内容が " 道路 " の内容に追加されます。読み込んだオブジェクトと " 道路 " 内のオブジェクトの名前が重複する場合、Director によって新しい名前が作成されます。

```
-- Lingo
member("Roadway").loadFile("Truck.W3d", FALSE, TRUE)

// Javascript
member("Roadway").loadFile("Truck.W3d",false,true);
```

次のステートメントは、"Chevy.W3d" というファイルの内容を " 道路 " というキャストメンバに読み込みます。"Chevy.W3d" は、ムービーから 1 レベル下の "Models" というフォルダにあります。" 道路 " の内容は "Chevy.W3d" の内容に置き換わります。2 番目のパラメータの値が TRUE なので、3 番目のパラメータは無関係です。

```
-- Lingo
member("Roadway").loadFile(the moviePath & "Models\Chevy.W3d", TRUE, TRUE)

// Javascript
member("Roadway").loadFile(_movie.Path + "Models\Chevy.W3d",true,true);
```

関連事項

state (3D)

loadPolicyFile()

使用法

```
_movie.loadPolicyFile(URL))
```

解説

URL パラメータにより指定される場所から、クロスドメインポリシーファイルをロードします。

例えば、Shockwave ムービー http://www.foo.com/crossdomain.dcr が http://www.Server1.com/data.txt からデータをロードしようとすると、http://www.Server1.com にあるクロスドメインポリシーファイルにより http://www.foo.com が Server1 からデータにアクセスできるかどうかを判定します。

http://www.Server1.com 内の次のポリシーファイルを使用すると、http://www.foo.com の Shockwave ムービーはサーバからデータにアクセスできます。

```
<cross-domain-policy>
    <allow-access-from domain="http://www.foo.com" to-ports="*" />
  </cross-domain-policy>
```

クロスドメインポリシーファイルは、次の形式で指定されます。

```
<cross-domain-policy>
    <allow-access-from domain="*" secure="true" to-ports="*"  />
  </cross-domain-policy>
```

ドメイン属性はサーバにアクセスできるドメインを指定します。domain="*" を指定すると、すべてのドメインにアクセスが許可されます。domain="www.bar.com/foo/*" を指定すると、www.bar.com/foo/ のすべてのドメインがサーバにアクセスできます。

secure 属性を False に設定すると、https ポリシーファイルを使用して HTTP ソースからの要求へのアクセスを許可できます。secure パラメータのデフォルト値は True です。この属性を False に設定することは推奨されません。

to-ports 属性を使用すると、サーバにアクセスできる所定のドメインのポートのリストまたは範囲を指定できます。to-ports = "*" の場合、または to-ports を <allow-access-from> タグ内で指定しない場合、ドメインのすべてのポートがアクセスを許可されます。<allow-access-from> タグでは、to-port および secure 属性は省略可能です。

クロスドメインポリシーファイルの例を次に示します。

```
<cross-domain-policy>
<allow-access-from domain="www.bar.com" to-ports="1222,8000-9000,9192" />
  </cross-domain-policy>
```

クロスドメインポリシーの詳細については、「Adobe® Director® の使用」ガイドで「クロスドメインポリシー」を検索してください。

詳細：

- loadPolicyFile() を含んでいない Shockwave ムービーがそれ以外のドメインからデータにアクセスすると、Shockwave Player はサーバのルートディレクトリからポリシーファイル、crossdomain.xml をロードしようとします。ルートディレクトリにポリシーファイルがない場合、Director ではセキュリティダイアログボックスが表示されます。
- loadPolicyFile() は、指定されたサーバの場所からポリシーファイルを 1 つだけロードできます。loadPolicyFile() の複数のインスタンスを指定した場合、シーケンスの最初のファイルがロードされます。

  次のコードスニペットでは、pf2.xml はロードされません。

  ```
  on prepareMovie
  _movie.loadPolicyFile("http://www.foo.com/bar/pf1.xml")
  _movie.loadPolicyFile("http://www.foo.com/bar/pf2.xml")
  end
  ```

  次のコードスニペットでは、ポリシーファイルは別々のサーバの場所からロードされるため、両方のポリシーファイルがロードされます。

  ```
  on prepareMovie
  _movie.loadPolicyFile("http://www.foo.com/pf1.xml")
  _movie.loadPolicyFile("http://www.foo.com/bar/pf2.xml")
  end
  ```

- 場所 http://www.foo.com/bar/pf.xml からロードされるポリシーファイルは http://www.foo.com/bar/* のすべてのサブドメインにアクセスできます。異なるアクセス権を持つ別のポリシーファイルを http://www.foo.com/bar/ のサブディレクトリからロードすると、これらのアクセス権のみがサブディレクトリに適用されます。

例えば、http://www.director.com の **policyfile.xml file** という次のクロスドメインポリシーファイルを考えてみましょう。

```
<cross-domain-policy>
<allow-access-from domain="www.bar.com" to-ports="1222,8000-9000,9192" />
  </cross-domain-policy>
```

URL http://www.swgame.com の Shockwave ムービーに次のコードスニペットがある場合、ダウンロードは失敗します。

```
on prepareMovie
_movie.loadPolicyFile(http://www.director.com/policyfile.xml)
getNetText("http://www.Director.com/data.txt")
getNetText("http://www.Director.com/foo/bar/data.txt")
end
```

次のコンテンツを含む別のポリシーファイル、policyfile2.xml を考えてみましょう。

```
<cross-domain-policy>
<allow-access-from domain="http://www.swgames.com" to-ports="*" />
  </cross-domain-policy>
```

URL http://www.swgame.com の Shockwave ムービーに次のコードスニペットがある場合、以下のシナリオでダウンロードが成功します。

```
on prepareMovie
_movie.loadPolicyFile(http://www.director.com/policyfile.xml)
_movie.loadPolicyFile(http://www.director.com/foo/bar/policyfile2.xml)
getNetText("http://www.Director.com/data.txt") -- download fails
getNetText("http://www.Director.com/foo/bar/data.txt") --download succeeds
end
```

- 次のケースでは、Shockwave でセキュリティダイアログが表示されます。
  - クロスドメインポリシーファイルのシンタックスが不適切です。
  - loadPolicyFile() は不適切な URL を指定しています。
  - ムービーで loadPolicyfile() が指定されていません。また、デフォルトの crossdomain.xml はサーバのルートディレクトリにありません。

パラメータ

パラメータ	解説	必須 / オプション
URL	ロードするポリシーファイルの場所を指定します。	必須

例文

```
--Lingo
_movie.loadPolicyFile("http://www.testServer.com/test/pf.xml")
//JavaScript
_movie.loadPolicyFile("http://www.example.com/dir/pf.xml");
```

locToCharPos()

使用法

```
-- Lingo syntax
memberObjRef.locToCharPos(location)

// JavaScript syntax
memberObjRef.locToCharPos(location);
```

解説

指定したフィールドキャストメンバ内において、フィールド内のポイントに最も近い文字の位置を返す関数です。

返される値は、ストリング内の最初の文字を 1 としてその文字が何字目かを示す整数になります。

パラメータ

location 必須。フィールドキャストメンバ内のポイントです。[場所] の値は、フィールドキャストメンバの左上隅を原点として測定されたポイントです。

例文

次のステートメントは、「今日のニュース」フィールドキャストメンバの左上隅から右に 100 ピクセル、下に 100 ピクセルのポイントに最も近い文字を調べ、結果を PageDesign 変数に割り当てます。

```
--Lingo syntax
pageDesign = member("Today's News").locToCharPos(point(100, 100))

// JavaScript syntax
var pageDesign = member("Today's News").locToCharPos(point(100, 100));
```

locVToLinePos()

使用法

```
-- Lingo syntax
memberObjRef.locVToLinePos(locV)

// JavaScript syntax
memberObjRef.locVToLinePos(locV);
```

解説

指定した垂直位置に表示された文字の行番号を返す関数です。

パラメータ

locV 必須。文字の行の垂直位置を指定します。この値はフィールドキャストメンバの上端からのピクセル数であって、現在ステージに表示されているフィールドキャストメンバの一部分ではありません。

例文

次のステートメントは、フィールドキャストメンバ " 今日のニュース " の上端から 150 ピクセルのところに表示される文字の行番号を調べ、その結果を変数 pageBreak に代入します。

```
--Lingo syntax
pageBreak = member("Today's News").locVToLinePos(150)

// JavaScript syntax
var pageBreak = member("Today's News").locVToLinePos(150);
```

log()

使用法

```
log(number)
```

解説

指定した値の自然対数を計算する関数です（Lingo のみ）。

JavaScript シンタックスでは、Math オブジェクトの log() 関数を使用します。

パラメータ

number 必須。自然対数を計算する数値です。この数値は、必ず 0 より大きな小数でなくてはなりません。

例文

次のステートメントは、10.5 の自然対数を Answer 変数に割り当てます。

```
-- Lingo
Answer = log(10.5)

// Javascript
Answer = Math.log(10.5);
```

例文

次のステートメントは、変数 Number の平方根の自然対数を計算し、結果を Answer 変数に割り当てます。

```
-- Lingo
Answer = log(Number.sqrt)

// Javascript
Answer = Math.log(Math.sqrt(Number));
```

makeList()

使用法

```
--Lingo syntax
parserObject.makeList()

// JavaScript syntax
parserObject.makeList();
```

解説

関数 ; parseString() または parseURL() で構文解析済み XML ドキュメントに基づいてプロパティリストを返します。

パラメータ

なし

例文

次のハンドラは XML ドキュメントを構文解析し、その結果を示すリストを返します。

```
-- Lingo syntax
on ConvertToList xmlString
    parserObject = new(xtra "xmlparser")
    errorCode = parserObject.parseString(xmlString)
    errorString = parserObject.getError()
    if voidP(errorString) then
        parsedList = parserObject.makeList()
    else
        alert "Sorry, there was an error" && errorString
        exit
    end if
    return parsedList
end

// JavaScript syntax
function ConvertToList(xmlString) {
    parserObject = new Xtra("xmlparser"); // check syntax
    errorCode = parserObject.parseString(xmlString);
    errorString = parserObject.getError();
    if (voidP(errorString)) {
        parsedList = parserObject.makeList();
    } else {
        alert("Sorry, there was an error" + errorString);
        return false;
    }
    return parsedList;
}
```

関連事項

makeSubList()

makeScriptedSprite()

使用法

```
-- Lingo syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc})

// JavaScript syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc});
```

解説

スプライトチャネルの制御をスコープからスクリプトに切り替え、オプションで指定したキャストメンバのスプライトをステージの指定した位置に配置するスプライトチャネル関数です。

スプライトチャネルの制御をスコープに戻すには、removeScriptedSprite() を呼び出します。

パラメータ

memberObjRef オプション。指定したスプライトを作成するキャストメンバへの参照です。このパラメータのみを指定すると、ステージの中央にスプライトが配置されます。

loc オプション。スクリプトが割り当てられたスプライトが配置されるステージ上の位置を指定するポイントです。

例文

次のステートメントは、kite という名前のキャストメンバからスクリプトが割り当てられたスプライトをスプライトチャネル 5 で作成し、ステージ上の特定のポイントに配置します。

```
--- Lingo syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70))

// JavaScript syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70));
```

関連事項

removeScriptedSprite(), スプライトチャネル

makeSubList()

使用法

```
XMLnode.makeSubList()
```

解説

関数 ; makeList() がリストフォーマットで XML ドキュメントのルートを返す方法と同じ方法で子ノードからプロパティリストを返します。

パラメータ

なし

例文

次の XML から始まる例 :

```
<e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
</e1>
```

次のステートメントは、タグ <e1> の最初の子のコンテンツから作成されたプロパティリストを返します。

```
put gparser.child[ 1 ].child[ 1 ].makeSubList()
-- ["tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]]]
```

関連事項

makeList()

map()

使用法

```
map(targetRect, sourceRect, destinationRect)
map(targetPoint, sourceRect, destinationRect)
```

解説

基準矩形と対象矩形の関係を基にして、矩形やポイントの位置とサイズを特定するための関数です。

[対象矩形] と [基準矩形] との関係によって、この関数の実行結果と [照合矩形] との関係が決まります。

パラメータ

targetRect 必須。関係における対象矩形です。

targetPoint 必須。関係における対象ポイントです。

sourceRect 必須。関係における基準矩形です。

destinationRect 必須。コピー先の矩形です。

例文

このビヘイビアーでは、スプライトはすべてドラッグ可能に設定してあります。スプライト 2b には小型のビットマップを含めてあります。スプライト 1s は、スプライト 2b を入れるのに十分な大きさの矩形スプライトです。スプライト 4b は、スプライト 2b のビットマップを大きくしたものです。スプライト 3s は、スプライト 1s で規定されている形状を大きくしたものです。スプライト 2b、またはスプライト 1s を移動すると、スプライト 4b が移動します。スプライト 2b をドラッグすると、その動きがスプライト 4b に反映されます。スプライト 1s をドラッグすると、スプライト 4b が反対方向に移動します。スプライト 2b、またはスプライト 1s のサイズを変更すると、別の興味深い結果を得ることができます。

```
on exitFrame
    sprite(4b).rect = map(sprite(2b).rect, sprite(1s).rect, sprite(3s).rect)
    go the frame
end
```

map (3D)

使用法

```
member(whichCastmember).motion(whichMotion).map(whichOtherMotion {, boneName})
```

解説

3D モーションコマンド。指定されたモーションを現在のモーションにマップして、ボーンおよびそのボーンのすべての子に適用します。このコマンドで、指定したボーンとその子にマップされていたすべてのモーションが置き換わります。モデルのプレイリストは変更されません。

パラメータ

whichOtherMotion 必須。マップするモーションの名前を指定するストリングです。

boneName オプション。マップされたモーションを適用するボーンの名前を指定するストリングです。省略した場合、ルートボーンが使用されます。

例文

次のステートメントは、"Neck" というボーンから始まる "SitDown" というモーションに、"LookUp" というモーションをマップします。モデルは、座るという動作と見上げるという動作を同時に行います。

```
member("Restaurant").motion("SitDown").map("LookUp", "Neck")
```

関連事項

motion(), duration (3D), cloneMotionFromCastmember

mapMemberToStage()

使用法

```
sprite(whichSpriteNumber). mapMemberToStage(whichPointInMember)
mapMemberToStage(sprite whichSpriteNumber, whichPointInMember)
```

解説

指定されたスプライトとポイントを使って、ステージ範囲内の等しいポイントを返す関数です。このプロパティは、quad または変形していない場合の矩形を使ったスプライトへの現行の変形とみなされます。

これはステージ上のスプライトへの主要な変形が行われた場合でも、キャストメンバ内の特定エリアがクリックされたかどうかを判断するのに便利です。

ステージ上の指定されたポイントがスプライト内にないときは、VOID が返されます。

パラメータ

whichPointInMember 必須。等しいポイントが返されるポイントです。

例文

次のステートメントは、指定された sprite(1) と point(10,10) を使用して、ステージ範囲内の等しいポイントを返します。

```
-- Lingo
sprite(1). mapMemberToStage(point(10,10))

// Javascript
sprite(1). mapMemberToStage(point(10,10)) ;
```

関連事項

map(), mapStageToMember()

mapStageToMember()

使用法

```
sprite(whichSpriteNumber). mapStageToMember(whichPointOnStage)
mapStageToMember(sprite whichSpriteNumber, whichPointOnStage)
```

解説

指定されたスプライトとポイントを使って、キャストメンバ範囲内の等しいポイントを返す関数です。quad を使用したスプライトの形状変更があるかどうか、形状変更がない場合は矩形そのものがわかります。

これはステージ上のスプライトへの主要な変形が行われた場合でも、キャストメンバ内の特定エリアがクリックされたかどうかを判断するのに便利です。

ステージ上の指定されたポイントがスプライト内にないときは、VOID が返されます。

パラメータ

whichPointOnStage 必須。等しいポイントが返されるポイントです。

関連事項

map(), mapMemberToStage()

marker()

使用法

```
-- Lingo syntax
_movie.marker(markerNameOrNum)

// JavaScript syntax
_movie.marker(markerNameOrNum);
```

解説

現行フレームの前後にあるマーカのフレーム番号を返すムービーメソッドです。

「次へ」、「戻る」などのボタンを実装するときや、アニメーションループを設定するときなどに役に立ちます。

パラメータ **markerNameOrNum** が整数の場合、任意の正の整数、負の整数、または 0 に評価されます。例えば、次のようになります。

- marker(2) は、現行フレームよりも 2 つ後のマーカのフレーム番号を返します。
- marker(1) は、現行フレームの次のマーカのフレーム番号を返します。
- marker(0) は、現行フレームにマーカが設定されている場合には現行フレームの番号を返し、マーカが設定されていない場合は 1 つ前のマーカのフレーム番号を返します。
- marker(-1) は、marker(0) の 1 つ前のマーカのフレーム番号を返します。
- marker(-2) は、marker(0) よりも 2 つ前のマーカのフレーム番号を返します。

パラメータ **markerNameOrNum** がストリングの場合、marker() は、マーカラベルがそのストリングと一致する最初のフレームの番号を返します。

パラメータ

markerNameOrNum 必須。マーカラベルを指定するストリング、またはマーカ番号を指定する整数です。

例文

次のステートメントは、現在のフレームにマーカが付いている場合は、再生ヘッドを現在のフレームの先頭に移動します。付いていない場合は、再生ヘッドを 1 つ前のマーカに移動します。

```
-- Lingo syntax
_movie.go(_movie.marker(0))

// JavaScript syntax
_movie.go(_movie.marker(0));
```

次のステートメントは、スコア内の次のマーカの番号を変数 nextMarker に割り当てます。

```
-- Lingo syntax
nextMarker = _movie.marker(1)

// JavaScript syntax
nextMarker = _movie.marker(1);
```

関連事項

frame, frameLabel, go(), label(), markerList, ムービー

matrixAddition()

使用法

```
<Matrix> matrixAddition(matrix1, matrix2)
```

解説

グローバル関数。2 つのマトリックスの加算を実行し、結果をマトリックスとして返します。マトリックス 1 とマトリックス 2 は同じ次元でなければなりません。

パラメータ

matrix1 必須。最初のマトリックス。

matrix2 必須。2 番目のマトリックス。

例文

次のコードは、2 つのマトリックスを作成し、matrixAddition() を使用してそれらを加算します。

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixAddition(mat1, mat2)
put(mat3.getVal(2,2))
-- 12.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixAddition(mat1, mat2);
put(mat3.getVal(2,2));
// 12.0000
```

関連事項

getVal(), setVal(), numRows(), numColumns(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

matrixMultiply()

使用法

```
<Matrix> matrixMultiply(matrix1, matrix2)
```

解説

グローバル関数。2 つのマトリックスの乗算を実行し、結果をマトリックスとして返します。マトリックス 1 の列の数は、マトリックス 2 の行の数と同じでなければなりません。

パラメータ

matrix1 必須。最初のマトリックス。

matrix2 必須。2 番目のマトリックス。

例文

次のコードは、2 つのマトリックスを作成し、matrixMultiply() を使用してそれらを乗算します。

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixMultiply(mat1, mat2)
put(mat3.getVal(2,2))
-- 50.0000

//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixMultiply(mat1, mat2);
put(mat3.getVal(2,2));
// 50.0000
```

関連事項

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

matrixMultiplyScalar()

使用法

```
<Matrix> matrixMultiplyScalar(matrix, scalarMultiplier)
```

解説

グローバル関数。マトリックスの各要素に指定したスカラーを乗算し、結果をマトリックスとして返します。

パラメータ

matrix 必須。スカラー乗算を行うマトリックス。

scalarMultiplier 必須。マトリックスに乗算するスカラー値。

例文

以下のコードスニペットは、2 つのマトリックスを作成し、matrixMultiply () を使用して追加します。

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixMultiplyScalar(mat1, 10)
put(mat2.getVal(2,2))
-- 40.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixMultiplyScalar(mat1, 10);
put(mat2.getVal(2,2));
// 40.0000
```

関連事項

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixTranspose(), newMatrix()

matrixTranspose()

使用法

```
<Matrix> matrixTranspose(matrix)
```

解説

グローバル関数。指定したマトリックスに転置操作を実行し、結果を別のマトリックスとして返します。

パラメータ

matrix 必須。転置するマトリックス。

例文

次のコードは、マトリックスを作成し、matrixTranspose() を使用してそれを転置します。

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixTranspose(mat1)
put(mat1.getVal(1,2))
put(mat2.getVal(1,2))
-- 2.0000
-- 3.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixTranspose(mat1);
put(mat1.getVal(1,2));
put(mat2.getVal(1,2));
// 2.0000
// 3.0000
```

関連事項

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), newMatrix()

max()

使用法

```
list.max()
max(list)
max(value1, value2, value3, ...)
```

解説

指定したリストまたは一連の値の中で最も大きな値を返す関数です（Lingo のみ）。

比較演算子（< および >）をストリングの評価に使用できるのと同様に、max 関数も ASCII 文字に対して使用できます。

パラメータ

value1, value2, value3, ... オプション。最も大きな値が選択される値のリストです。

例文

次のハンドラは、[#Castle:600, #Schmitz:750, #Wang:230] という内容のリスト Bids の最大値を変数 Winner に割り当て、結果を " おめでとう " フィールドキャストメンバのストリングに挿入します。

```
-- Lingo syntax
on findWinner Bids
    Winner = Bids.max()
    member("Congratulations").text =
    "You have won, with a bid of $" & Winner &"!"
end

// JavaScript syntax
function findWinner(Bids) {
    Winner = Bids.max();
    member("Congratulations").text = "You have won, with a bid of $" +
    Winner + "!");
}
```

maximize()

使用法

```
-- Lingo syntax
windowObjRef.maximize()

// JavaScript syntax
windowObjRef.maximize();
```

解説

ウィンドウを最大化するウィンドウメソッドです。

このメソッドは、カスタムタイトルバーを作成するときに使用します。

パラメータ

なし

例文

次のステートメントは、Artists という名前のウィンドウが最大化されていない場合に、このウィンドウを最大化します。

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

関連事項

minimize(), ウィンドウ

mci

使用法

```
mci "string"
```

解説

マルチメディアエクステンションを制御するための Windows 用のコマンドです。[ストリング] で指定したストリングを MCI (メディアコントロールインターフェイス) に送ります。

注意：Microsoft は、16 ビットの MCI インターフェイスの使用を推奨していません。この機能ではなく、代わりにサードパーティの Xtra 拡張を使用してください。

パラメータ

string 必須。MCI に渡されるストリングです。

例文

次のステートメントは、Windows でムービーを再生する場合にのみ "play cdaudio from 200 to 600 track 7" コマンドを実行します。

```
mci "play cdaudio from 200 to 600 track 7"
```

member()

使用法

```
-- Lingo syntax
member(memberNameOrNum {, castNameOrNum})

// JavaScript syntax
member(memberNameOrNum {, castNameOrNum});
```

解説

キャストメンバへの参照を作成し、オプションでそのメンバを含むキャストライブラリを指定するトップレベル関数です。

member() メソッドは、**memberNameOrNum** パラメータおよび **castNameOrNum** パラメータと共に使用された場合、キャストライブラリと、キャストライブラリ内のメンバの両方への参照です。

```
trace(sprite(1).member);
// (member 1 of castLib 1)
```

このメソッドは、スプライトの spriteNum プロパティとは異なります。後者は、常にキャストライブラリ内の整数指定位置ですが、キャストライブラリは指定しません。

```
trace(sprite(2).spriteNum);
// 2
```

メンバの number プロパティは、特定のキャストライブラリにある特定のメンバへの絶対参照でもあります。

```
trace(sprite(3).member.number)
// 3
```

パラメータ

memberNameOrNum 必須。参照するキャストメンバの名前を指定するストリング、または参照するキャストメンバのインデックス位置を指定する整数です。

castNameOrNum オプション。メンバが属するキャストライブラリ名を指定するストリング、またはメンバが属するキャストライブラリのインデックス位置を指定する整数です。省略した場合、member() は一致するものが見つかるまですべてのキャストライブラリを検索します。

例文

次のステートメントは、変数 memWings を、Transportation という名前のキャストライブラリに含まれている Planes という名前のキャストメンバに設定します。

```
-- Lingo syntax
memWings = member("Planes", "Transportation")

// JavaScript syntax
var memWings = member("Planes", "Transportation");
```

関連事項

メンバ , スプライト , spriteNum

mergeDisplayTemplate()

使用法

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList)

// JavaScript syntax
_movie.mergeDisplayTemplate(propList);
```

解説

任意の数の「テンプレートの表示」プロパティを、「テンプレートの表示」プロパティの既存のセットに一度にマージするムービーメソッドです。

パラメータ

propList 必須。「テンプレートの表示」プロパティの既存のセットにマージする「テンプレートの表示」プロパティを含むプロパティリストです。Lingo では、名前と値のペアのカンマ区切りリスト、またはシンボルと値のペアのカンマ区切りリストを **propList** に指定できます。JavaScript シンタックスでは、名前と値のペアのカンマ区切りリストのみを **propList** に指定できます。

例文

次のステートメントは、title プロパティの値を displayTemplate プロパティにマージします。

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList(#title, "Welcome!"))

// JavaScript syntax
_movie.mergeDisplayTemplate(propList("title", "Welcome!"))
```

関連事項

appearanceOptions, displayTemplate, ムービー , propList(), titlebarOptions

mergeProps()

使用法

```
-- Lingo syntax
windowObjRef.mergeProps(propList)

// JavaScript syntax
windowObjRef.mergeProps(propList);
```

解説

ウィンドウメソッドです。任意の数のウィンドウプロパティを、ウィンドウプロパティの既存のセットに一度にマージします。

パラメータ

propList 必須。ウィンドウプロパティの既存のセットにマージするウィンドウプロパティのセットです。これらのプロパティは、appearanceOptions プロパティおよび titlebarOptions プロパティによって指定されます。

- Lingo では、名前と値のペアのカンマ区切りリスト、またはシンボルと値のペアのカンマ区切りリストを **propList** に指定できます。
- JavaScript シンタックスでは、名前と値のペアのカンマ区切りリストのみを **propList** に指定できます。

例文

次のステートメントは、Cars という名前のウィンドウの様々なプロパティを設定します。

```
-- Lingo syntax
window("Cars").mergeProps([#title:"Car pictures", #resizable:FALSE, #titlebarOptions:[#closebox:TRUE,
#icon:member(2)], #appearanceOptions:[#border:#line, #shadow:TRUE]])

// JavaScript syntax
window("Cars").mergeProps(propList("title","Car pictures",
"resizable",false,"titlebarOptions",propList("closebox",true, "icon",member(2)),
"appearanceOptions",propList("border","line", "shadow",true)));
```

関連事項

appearanceOptions, titlebarOptions, ウィンドウ

mesh （プロパティ）

使用法

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].meshProperty
```

解説

3D コマンド。meshDeform モディファイアが割り当てられたモデルのメッシュプロパティにアクセスできます。mesh.count として指定すると、参照先モデル内のメッシュの合計数を返します。

アクセス可能な各メッシュのプロパティは次のとおりです。

- colorList。指定されたメッシュで使用されるカラーのリストを取得または設定します。
- vertexList。指定されたメッシュで使用される頂点のリストを取得または設定します。
- normalList。指定されたメッシュで使用される法線ベクトルのリストを取得または設定します。

- textureCoordinateList。指定されたメッシュの最初のテクスチャレイヤーで使用されるテクスチャ座標を取得または設定します。指定されたメッシュに含まれるほかのテクスチャレイヤーのテクスチャ座標を取得または設定するには、meshdeform.mesh[インデックス].texturelayer[インデックス].textureCoordinateList を使用します。
- textureLayer[index]。指定されたテクスチャレイヤーのプロパティへのアクセスを取得および設定します。
- face[index]。指定されたメッシュのフェイスで使用される頂点、法線、テクスチャ座標、カラー、およびシェーダを取得または設定します。
- face.count。指定されたメッシュ内にあるフェイスの合計数を取得します。

注意：これらのプロパティの詳細については、個別のエントリを参照してください（この後の「関連事項」に一覧があります）。

パラメータ

なし

例文

次の Lingo は、#meshDeform モディファイアを "thing1" というモデルに追加し、"thing1" モデル内の最初のメッシュの "vertexList" を表示します。

```
-- Lingo
member("newAlien").model("thing1").addModifier(#meshDeform)
put member("newalien").model("thing1").meshDeform.mesh[1].vertexList

// javascript
member("newAlien").getProp("model",1).addModifier(symbol("meshDeform"));
put (member("newalien").getProp("model",1).getPropRef("meshDeform").getPropRef("mesh",1).vertexList ;
-- [vector(239.0, -1000.5, 27.4), vector(162.5, -1064.7, 29.3), vector(115.3, -1010.8, -40.6),

vector(239.0, -1000.5, 27.4), vector(115.3, -1010.8, -40.6),
vector(162.5, -1064.7, 29.3), vector(359.0, -828.5, -46.3),
vector(309.9, -914.5, -45.3)]
```

次のステートメントは、" 航空機 " というモデル内のメッシュ数を表示します。

```
-- Lingo
put member("world").model("Aircraft").meshDeform.mesh.count
-- 4
```

関連事項

meshDeform （モディファイア）, colorList, textureCoordinateList, textureLayer, normalList, vertexList (mesh deform), face[ ]

meshDeform （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

解説

3D モディファイア。参照されたモデルのメッシュ構造の様々な属性を制御します。#meshDeform モディファイア (addModifier コマンドを使用) をモデルに追加すると、#meshDeform モディファイアの次のプロパティにアクセスできます。

注意：次のプロパティの詳細については、「関連事項」にある個別のエントリを参照してください。

- face.count。参照されたモデルのフェイスの合計数を返します。

- mesh.count。参照されたモデル内のメッシュの数を返します。
- mesh[index]。指定されたメッシュのプロパティにアクセスします。

パラメータ
なし

例文
次のステートメントは、"gbFace" というモデル内のフェイス数を表示します。

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.face.count
-- 432

// Javascript
member("3D World").getProp("model", a).getPropRef("meshDeform") ;
// where a refers to the number index of the model "gbFace"
```

次のステートメントは、"gbFace" というモデル内のメッシュの vertexList を表示します。

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.mesh[1].vertexList
-- 2

// Javscript
member("3D World").getProp("model", a).getPropRef("meshDeform").getPropRef("mesh" ,1).vertexList ;
```

次のステートメントは、"gbFace" というモデル内で 2 番目のメッシュのフェイス数を表示します。

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

関連事項
mesh （プロパティ）, addModifier

min

使用法

```
list.min
min(list)
min (a1, a2, a3...)
```

解説
リスト内の最小値を指定する関数です（Lingo のみ）。

パラメータ
a1, a2, a3, ... オプション。最も小さい値が選択される値のリストです。

例文
次のハンドラは、[#Castle:600, #Shields:750, #Wang:230] という内容のリスト bids の最小値を変数 vLowest に割り当て、結果を " 残念 " フィールドキャストメンバの一部に挿入します。

```
on findLowest bids
    vLowest = bids.min()
    member("Sorry").text = \
    "We're sorry, your bid of $" & vLowest && "is not a winner!"
end
```

関連事項

max()

minimize()

使用法

```
-- Lingo syntax
windowObjRef.minimize()

// JavaScript syntax
windowObjRef.minimize();
```

解説

ウィンドウを最小化するウィンドウメソッドです。

このメソッドは、カスタムタイトルバーを作成するときに使用します。

パラメータ

なし

例文

次のステートメントは、Artists という名前のウィンドウが最小化されていない場合に、このウィンドウを最小化します。

```
-- Lingo syntax
if (window("Artists").sizeState <> #minimized) then
    window("Artists").minimize()
end if

// JavaScript syntax
if (window("Artists").sizeState.toString() != symbol("minimized").toString())
{
    window("Artists").minimized();
}
```

関連事項

maximize(), ウィンドウ

model (3D)

使用法

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モデルで指定された名前のモデル、またはインデックスで指定されたインデックス位置にあるモデルを返します。指定されたパラメータに該当するモデルが存在しない場合は void を返します。model.count を指定すると、このコマンドは参照されるキャストメンバ内のモデルの数を返します。このコマンドを使用して、指定されたモデルのプロパティにアクセスすることもできます。

モデル名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモデルのインデックス位置が変更される場合があります。

指定された名前またはインデックス位置に該当するモデルがない場合は、void を返します。

パラメータ

whichModel オプション。返すモデルの名前を指定するストリングです。

例文

次のステートメントは、" 仮のプレーヤ " というモデルへの参照を変数 thismodel に格納します。

```
thismodel = member("3DWorld").model("Player Avatar")
```

次のステートメントは、"3D ワールド " というキャストメンバの 8 番目のモデルへの参照を変数 thismodel に格納します。

```
thismodel = member("3DWorld").model[8]
```

次のステートメントは、スプライト 1 のメンバに 4 つのモデルがあることを示します。

```
put sprite(1).member.model.count
-- 4
```

modelResource

使用法

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モデルリソースで指定された名前のモデルリソースか、またはインデックスで指定されるインデックス位置にあるモデルリソースを返します。指定されたパラメータに該当するモデルリソースが存在しない場合は void を返します。modelResource.count を指定すると、このコマンドは参照されるキャストメンバ内のモデルリソースの数を返します。このコマンドを使って、指定されたモデルリソースのプロパティにアクセスすることもできます。

モデルリソース名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモデルリソースのインデックス位置が変更される場合があります。

パラメータ

whichModelResource オプション。返すモデルリソースの名前を指定するストリングです。

例文

次のステートメントは、"HouseA" というモデルリソースへの参照を変数 thismodelResource に格納します。

```
thismodelResource = member("3DWorld").modelResource("HouseA")
```

次のステートメントは、"3DWorld" というキャストメンバの 14 番目のモデルリソースへの参照を変数 thismodelResource に格納します。

```
thismodelResource = member("3DWorld").modelResource[14]
```

次のステートメントは、スプライト 1 のメンバに 10 のモデルリソースがあることを示します。

```
put sprite(1).member.modelResource.count
--10
```

modelsUnderLoc

使用法

```
member(whichCastmember).camera(whichCamera).modelsUnderLoc(pointWithinSprite, optionsList)
```

解説

3D コマンド。参照されたカメラを使用するスプライトの矩形内の、指定されたポイントで見つかったモデルのリストを返します。モデルのリストを返す前に、オプションパラメータのセットと比較することもできます。

返されたリストにある最初のモデルは、観察者に最も近いモデルで、最後にあるモデルは、観察者から最も離れたモデルです。

モデルごとに 1 つの交差 (最も近い交差) のみが返されます。

指定されたポイントからモデルが見つからない場合は、空のリストを返します。

パラメータ

pointWithinSprite 必須。モデルのリストを検索するポイントです。このポイントには、スプライトの左上隅からの相対位置をピクセル単位で指定します。

optionsList オプション。返すモデルの最大数、情報の詳細レベル、キャストするモデルのリスト、およびレイを描く最大距離を指定するリストです。これらすべてのプロパティは省略可能です。

maxNumberOfModels オプション。返されるリストの最大長を指定する整数です。省略した場合、このコマンドは指定されたポイントで見つかったすべてのモデルの参照を含むリストを返します。

levelOfDetail オプション。返される情報の詳細レベルを指定するシンボルです。有効な値は以下のとおりです。

- #simple。そのポイントから見つかったモデルへの参照を含むリストを返します。これはデフォルト設定です。
- #detailed。交差するモデルを表すプロパティリストのリストを返します。各プロパティリストには以下のプロパティがあります。
  - #model。交差するモデルオブジェクトへの参照です。
  - #distance。カメラからモデルとの交差ポイントまでの距離です。
  - #isectPosition。交差ポイントのワールドスペース位置を表すベクトルです。
  - #isectNormal。交差ポイントのメッシュへのワールドスペース法線ベクトルです。
  - #meshID。交差するメッシュの meshID です。この ID は、meshDeform モディファイアのメッシュリストのインデックスとして使用できます。
  - #faceID。交差するフェイスのフェイス ID です。この ID は、meshDeform モディファイアのフェイスリストのインデックスとして使用できます。
  - #vertices。交差するフェイスの頂点のワールドスペース位置を表す、ベクトルの 3 つのエレメントリストです。

- #uvCoord。フェイスの u および v の重心座標を表す、#u プロパティおよび #v プロパティを持つプロパティリストです。

modelList オプション。指定したレイで見つかった場合に含まれるモデル参照のリストです。このリストに含まれないモデル参照は、指定したレイの下にある場合でも、無視されます。モデルのストリング名ではなく、このモデル参照を使用してください。含める各モデルを指定します。親モデル参照を追加しても、子モデル参照は自動的には含まれません。

例文

次のステートメントは、10 のモデルのリストを作成します。

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

次のステートメントは、最大で 10 のモデルを返し、簡単な詳細を含み、tModelList の結果を描画するオプションのリストを構築します。

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList]
```

オプションリストを構築したら、このハンドラの 1 行目で、カーソルの場所をステージ上のポイントからスプライト 5 内のポイントに移動します。2 行目で、modelsUnderLoc コマンドを使用して、そのポイントで見つかった先頭から 3 つのモデルを取得します。3 行目で、モデルについて返された情報をメッセージウィンドウに表示します。

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList)
    put m
    end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList);
    put(m);
}
```

関連事項

modelsUnderRay, modelUnderLoc

modelsUnderRay

使用法

```
member(whichCastmember).modelsUnderRay(locationVector, directionVector, optionsList)
```

解説

3D コマンド。指定された位置から開始される、指定された方向のレイが当たるモデルのリストを返します。どちらのベクトルもワールド座標で指定します。モデルのリストを返す前に、オプションパラメータのセットと比較することもできます。

返されたリストの最初のモデルは、位置ベクトルで指定された位置に最も近いモデルで、リストの最後のモデルは、その位置から最も遠いモデルです。

モデルごとに 1 つの交差 (最も近い交差) のみが返されます。

指定されたレイからモデルが見つからない場合は、空のリストを返します。

パラメータ

locationVector 必須。レイの開始位置であり、またモデルのリストが検索されるベクトルです。

directionVector 必須。レイが示している方向を指定するベクトルです。

optionsList オプション。返すモデルの最大数、情報の詳細レベル、キャストするモデルのリスト、およびレイを描く最大距離を指定するリストです。これらすべてのプロパティは省略可能です。

maxNumberOfModels オプション。返されるリストの最大長を指定する整数です。省略した場合、このコマンドは指定されたレイで見つかったすべてのモデルの参照を含むリストを返します。

levelOfDetail オプション。返される情報の詳細レベルを指定するシンボルです。有効な値は以下のとおりです。

- #simple。そのポイントから見つかったモデルへの参照を含むリストを返します。これはデフォルト設定です。
- #detailed。交差するモデルを表すプロパティリストのリストを返します。各プロパティリストには以下のプロパティがあります。
  - #model。交差するモデルオブジェクトへの参照です。
  - #distance。**locationVector** で指定されるワールド位置から、モデルの交差ポイントへの距離です。
  - #isectPosition。交差ポイントのワールドスペース位置を表すベクトルです。
  - #isectNormal。交差ポイントのメッシュへのワールドスペース法線ベクトルです。
  - #meshID。交差するメッシュの meshID です。この ID は、meshDeform モディファイアのメッシュリストのインデックスとして使用できます。
  - #faceID。交差するフェイスのフェイス ID です。この ID は、meshDeform モディファイアのフェイスリストのインデックスとして使用できます。
  - #vertices。交差するフェイスの頂点のワールドスペース位置を表す、ベクトルの 3 つのエレメントリストです。
  - #uvCoord。フェイスの u および v の重心座標を表す、#u プロパティおよび #v プロパティを持つプロパティリストです。

modelList オプション。指定したレイで見つかった場合に含まれるモデル参照のリストです。このリストに含まれないモデル参照は、指定したレイの下にある場合でも、無視されます。モデルのストリング名ではなく、このモデル参照を使用してください。含める各モデルを指定します。親モデル参照を追加しても、子モデル参照は自動的には含まれません。

maxDistance オプション。**locationVector** で指定されるワールド位置からの最大距離です。モデルの球体が指定した最大距離内である場合、そのモデルが含まれます。球体が範囲内の場合は、ポリゴンを範囲内に含み、したがって交差している可能性があります。

例文

次のステートメントは、10 のモデルのリストを作成します。

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

次のステートメントは、最大で 10 のモデルを返し、簡単な詳細を含み、tModelList の結果を描画し、最大レイ距離が 50 であるオプションのリストを構築します。

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList, #maxDistance: 50]
```

次のステートメントは、オプションリストを構築した後、vector (0, 0, 300) の位置から Z 軸の方向を指すレイの下に、そのリストを含めます。

```
put member("3d").modelsUnderRay(vector(0, 0, 300), vector(0, 0, -1), tOptionsList)
```

関連事項

modelsUnderLoc, modelUnderLoc

modelUnderLoc

使用法

```
member(whichCastmember).camera(whichCamera).modelUnderLoc(pointWithinSprite)
```

解説

3D コマンド。参照されたカメラを使用するスプライトの矩形内の、指定されたポイントで見つかった最初のモデルへの参照を返します。

指定されたポイントからモデルが見つからない場合は、void を返します。

指定されたポイントから見つかった全モデルのリスト、およびそのモデルの詳細については、「modelsUnderLoc」を参照してください。

パラメータ

pointWithinSprite 必須。最初のモデルを検索するポイントです。スプライト内のポイント で指定した位置は、スプライトの左上角を基準にしたものです。ピクセル単位で指定します。

例文

このハンドラの 1 行目で、カーソルの位置をステージ上のポイントからスプライト 5 内のポイントに変換します。2 行目で、そのポイントで見つかった最初のモデルを決定します。3 行目で、メッセージウィンドウに結果を表示します。

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelUnderLoc(pt)
    put m
end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelUnderLoc(pt);
    put(m);
}
```

関連事項

modelsUnderLoc, modelsUnderRay

motion()

使用法

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モーションで指定された名前のモーションか、またはインデックスで指定されるインデックス位置にあるモーションを返します。motion.count を指定すると、そのキャストメンバ内にあるモーションの合計数を返します。

オブジェクト名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモーションのインデックス位置が変更される場合があります。

指定された名前またはインデックス位置に該当するモーションがない場合は、void を返します。

例文

```
thisMotion = member("3D World").motion("Wing Flap")
thisMotion = member("3D World").motion[7]
put member("scene").motion.count
-- 2
```

関連事項

duration (3D), map (3D)

move()

使用法

```
-- Lingo syntax
memberObjRef.move({intPosn, castLibName})

// JavaScript syntax
memberObjRef.move({intPosn, castLibName});
```

解説

指定したキャストメンバを、包含するキャスト内の最初に空いている場所か、指定したキャスト内の指定した場所に移動するメンバ関数です。

move は通常ファイルで保存されるため、このメソッドはランタイムではなく、オーサリング環境で使用するのが適しています。キャストメンバの場所はエンドユーザの再生中のプレゼンテーションには影響しません。スプライトの内容を入れ替えるときや、ランタイム中の表示を変更するときには、スプライトのメンバを設定します。

パラメータ

intPosn オプション。メンバの移動先であるキャストライブラリ **castLibName** 内の位置を指定する整数です。

castLibName オプション。メンバの移動先であるキャストライブラリの名前を指定するストリングです。

例文

次のステートメントは、キャストウィンドウ内で最初に空いている場所にキャストメンバ " 神社 " を移動します。

```
-- Lingo syntax
member("shrine").move()

// JavaScript syntax
member("shrine").move();
```

次のステートメントは、" ビットマップ " キャストウィンドウ内の 20 番目の枠にキャストメンバ " 神社 " を移動します。

```
-- Lingo syntax
member("shrine").move(member(20, "Bitmaps"))

// JavaScript syntax
member("shrine").move(member(20, "Bitmaps"));
```

関連事項

メンバ

moveTo

使用法

```
soundObject.moveTo(#Mixer)
```

解説

サウンドオブジェクトメソッド。あるミクサから別のミクサへサウンドオブジェクトを移動します。

例文

次の例文は、指定したサウンドオブジェクトを mixer2 に移動します。

```
--Lingo syntax
on mouseUp me
     soundObjRef.moveTo(mixer2)
end

//JavaScript syntax
function mouseUp(){
     soundObjRef.moveTo(mixer2);
}
```

moveToBack()

使用法

```
-- Lingo syntax
windowObjRef.moveToBack()

// JavaScript syntax
windowObjRef.moveToBack();
```

解説

ウィンドウを他のすべてのウィンドウの後方に移動するウィンドウメソッドです。

パラメータ

なし

例文

次のステートメントは、windowList 内の最初のウィンドウを他のすべてのウィンドウよりも後方に移動します。

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToBack()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToBack();
```

移動したいウィンドウの名前が " デモウィンドウ " とわかっている場合、次のように記述することもできます。

```
-- Lingo syntax
window("Demo Window").moveToBack()

// JavaScript syntax
window("Demo Window").moveToBack();
```

関連事項

moveToFront(), ウィンドウ

moveToFront()

使用法

```
-- Lingo syntax
windowObjRef.moveToFront()

// JavaScript syntax
windowObjRef.moveToFront();
```

解説

ウィンドウを他のすべてのウィンドウよりも手前に移動するウィンドウメソッドです。

パラメータ

なし

例文

次のステートメントは、windowList 内の最初のウィンドウを他のウィンドウよりも手前に移動します。

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToFront()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToFront();
```

移動したいウィンドウの名前が " デモウィンドウ " とわかっている場合、次のように記述することもできます。

```
-- Lingo syntax
window("Demo Window").moveToFront()

// JavaScript syntax
window("Demo Window").moveToFront();
```

関連事項

moveToBack(), ウィンドウ

moveVertex()

使用法

```
-- Lingo syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange);
```

解説

ベクトルシェイプキャストメンバの頂点を別の場所に移動する関数です。

ムーブの座標は頂点ポイントの現在位置に、頂点ポイントの場所は元のベクトルシェイプメンバに基づいています。

頂点の場所を変更すると、エディタ内で頂点をドラッグするのと同様にシェイプに影響します。

パラメータ

vertexIndex 必須。移動する頂点のインデックス位置を指定します。

xChange 必須。頂点を水平方向に移動する量を指定します。

yChange 必須。頂点を垂直方向に移動する量を指定します。

例文

次のステートメントは、ベクトルシェイプ "Archie" の最初の頂点ポイントを現在位置から右へ 25 ピクセル、下へ 10 ピクセルシフトします。

```
-- Lingo syntax
member("Archie").moveVertex(1, 25, 10)

// JavaScript syntax
member("Archie").moveVertex(1, 25, 10);
```

関連事項

addVertex(), deleteVertex(), moveVertexHandle(), originMode, vertexList

moveVertexHandle()

使用法

```
-- Lingo syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange);
```

解説

ベクトルシェイプのキャストメンバの頂点ハンドルを別の場所に移動する関数です。

ムーブの座標は頂点ハンドルの現在位置に基づき、頂点ハンドルの場所は制御中の頂点ポイントに基づいています。

制御ハンドルの場所を変更すると、エディタ内で頂点をドラッグするのと同様にシェイプに影響します。

パラメータ

vertexIndex 必須。移動するハンドルを含む頂点のインデックス位置を指定します。

handleIndex 必須。移動するハンドルのインデックス位置を指定します。

xChange 必須。頂点ハンドルを水平方向に移動する量を指定します

。

yChange 必須。頂点ハンドルを垂直方向に移動する量を指定します。

例文

次のステートメントはベクトルシェイプ "Archie" の 2 番目の頂点ポイントの最初の制御ハンドルを現在位置から右へ 15 ピクセル、上へ 5 ピクセルシフトします。

```
-- Lingo syntax
moveVertexHandle(member("Archie"), 2, 1, 15, -5)

// JavaScript syntax
member("Archie").moveVertexHandle(2, 1, 15, -5);
```

関連事項

addVertex(), deleteVertex(), originMode, vertexList

multiply()

使用法

```
transform.multiply(transform2)
```

解説

3D コマンド。トランスフォーム **2** の位置、回転、スケールの各効果をオリジナルのトランスフォームに適用します。

パラメータ

transform2 必須。別のトランスフォームに適用する効果を含むトランスフォームを指定します。

例文

次のステートメントは、モデル「火星」のトランスフォームの位置、回転、スケールの各効果をモデル「冥王星」のトランスフォームに適用します。これは、1 フレームの間、「火星」を「冥王星」の親にするのと同様の効果があります。

```
-- Lingo
member("scene").model("Pluto").transform.multiply(member("scene").model("Mars").transform)

// Javascript
member("scene").getProp("model" ,i ).transform.multiply(member("scene").getProp("model" , j).transform) ;
```

mute （ミクサ）

使用法

```
mixer.Mute()
```

解説

ミクサ出力の volume プロパティをゼロに設定するミクサメソッドです。

例文

```
-- Lingo syntax
on mouseUp me
     mixerRef.mute() --Mutes the mixer.
end

//JavaScript syntax
function mouseUp(){
mixerRef.mute(); //Mutes the mixer.
}
```

関連事項

unmute（ミクサ）, ミクサ

mute（サウンドオブジェクト）

使用法

```
soundObject.mute()
```

解説

サウンドオブジェクトの volume プロパティをゼロに設定するサウンドオブジェクトメソッドです。

例文

```
-- Lingo syntax
on mouseUp me
     soundObjRef.mute() -- Mutes the sound object associated with soundObjRef.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.mute(); // Mutes the sound object associated with soundObjRef.
}
```

関連事項

unmute（サウンドオブジェクト）

neighbor

使用法

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face[index].neighbor[index]
```

解説

3D コマンド。フェイスの頂点の反対側にある、メッシュの特定のフェイスの隣接要素をまとめたリストを返す meshDeform コマンドです。頂点はインデックス (1,2,3) で指定します。リストが空の場合、フェイスはその方向に隣接要素を持っていないことを示します。リストの内容が複数ある場合、フェイスの辺に複数のフェイスが結合されている非多様体であることを示します。通常、リストには 4 つの整数値が格納されています。[meshIndex, faceIndex, vertexIndex, flipped]

meshIndex の値は、その隣接フェイスを含むメッシュのインデックスです。faceIndex の値は、そのメッシュ内の隣接フェイスのインデックスです。vertexIndex の値は、その隣接フェイスの共有されていない頂点のインデックスです。flipped の値は、そのフェイスの方向がオリジナルのフェイスの方向と同じ (1) か、または反対 (2) かを示します。

パラメータ
なし

関連事項
meshDeform（モディファイア）

netAbort

使用法

```
netAbort(URL)
netAbort(netID)
```

解説
結果を待たずにネットワーク操作をキャンセルするためのコマンドです。

ネットワーク ID を使うと最も効率的にネットワーク操作を終了できます。ネットワーク関数である getNetText() または postNetText() を使うと、ネットワーク ID が返されます。

ネットワーク ID がない場合でも、URL を使ってその URL へのデータ送信を停止することができます。URL は、ネットワーク操作を始めたときに使ったものである必要があります。データ送信が完了している場合には、このコマンドは無効になります。

パラメータ
URL 必須。キャンセルする URL を指定します。

netID オプション。キャンセルするネットワーク操作の ID を指定します。

例文
次のステートメントは、特定のネットワーク操作を中止するために network ID を netAbort に送ります。

```
-- Lingo syntax
on mouseUp
    netAbort(myNetID)
end

// JavaScript syntax
function mouseUp() {
    netAbort(myNetID);
}
```

関連事項
getNetText(), postNetText

netByteArrayResult

使用法

```
netByteArrayResult(netID)
```

解説

Net Lingo メソッド。クエリの結果をバイト配列としてフェッチします。

パラメータ

パラメータ	解説	必須 / オプション
netID	返すテキストを含むネットワーク操作の ID を指定します。	オプション

例文

```
--Lingo syntax
put netByteArrayResult(ID)

//JavaScript syntax
put(netByteArrayResult(ID);
```

netDone()

使用法

```
netDone()
netDone(netID)
```

解説

バックグラウンドでの読み込み操作（getNetText, preloadNetThing, gotoNetMovie、gotoNetPage、netTextResult など）が終了したかどうかを確認する関数です。終了したとき、またはブラウザのエラーにより操作が中断されたときは TRUE, を返します。読み込み操作が進行中のときは FALSE を返します。デフォルトは TRUE です。

- 最後に行われたネットワーク操作を調べるには、netDone() を使います。
- netID で指定したネットワーク操作を調べるには、netDone(**netID**) を使います。

バックグラウンドでの読み込み操作が進行中のとき、netDone 関数は 0 を返します。

パラメータ

netID オプション。テストするネットワーク操作の ID を指定します。

例文

次のハンドラは、netDone 関数を使用して、最後のネットワーク操作が終了しているかどうかを調べます。終了している場合は、netTextResult が返したテキストを "Display Text" フィールドキャストメンバに表示します。

```
-- Lingo syntax
on exitFrame
    if netDone() = 1 then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
function exitFrame() {
    if (netDone() == 1) {
        member("Display Text").text = netTextResult();
    }
}
```

次のハンドラは特定のネットワーク ID を netDone のパラメータとして使って、特定のネットワーク操作のステータスを調べます。

```
-- Lingo syntax
on exitFrame
    -- stay on this frame until the net operation is
    -- completed
    global mynetID
    if netDone(mynetID) = FALSE then
        go to the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    // stay on this frame until the net operation is completed
    global mynetID;
    if (!(netDone(mynetID))) {
        _movie.go(_movie.frame);
    }
}
```

関連事項

getNetText(), netTextResult(), gotoNetMovie, preloadNetThing()

netError()

使用法

```
netError()
netError(netID)
```

解説

ネットワーク操作でエラーが生じたかどうかを確認し、エラーが生じた場合はエラーメッセージに対応するエラー番号を返す関数です。操作が正常終了した場合、OK を意味するコードを返します。バックグラウンドでの読み込み操作が開始していない場合や操作が進行中の場合は、空白のストリングを返します。

- 最後のネットワーク操作を調べるには、netError() を使います。
- netID で指定したネットワーク操作を調べるには、netError(**netID**) を使います。

エラーコードには次のものがあります。

0	すべて正常です。
1	ローカルパスが存在しないディレクトリを示している場合に発生します。
4	MOA クラスが正しくありません。指定されたネットワークまたはネットワーク以外の Xtra が不適切にインストールされたか、またはインストールされていません。
5	MOA インターフェイスが正しくありません。「4」を参照してください。
6	URL または MOA クラスが正しくありません。指定されたネットワークまたはネットワーク以外の Xtra が不適切にインストールされたか、またはインストールされていません。
900	書き込み先のファイルは読み取り専用です。
903	ディスクに空き領域がありません。
905	ファイル仕様が正しくありません。
2018	postNetText エラーは、通常は URL で postNetText を パラメータと組み合わせて使用したときに発生します。M.Kloss は、2018 を避けるために、次のシンタックスを推奨しています。 pNetID = postNetText(myURL, myParamPropList)
4144	ネットワーク操作に失敗しました
4145	ネットワーク操作に失敗しました
4146	遠隔ホストと接続できませんでした。
4149	サーバからのデータは不適切なフォーマットです。
4150	接続が早く解除されました。
4154	タイムアウトのため操作が完了しませんでした。
4155	メモリ不足のため処理できません。
4156	リクエストへのプロトコルの応答がエラーを示しています。
4157	処理が認証されませんでした。
4159	URL が無効です。
4164	ソケットを作成できませんでした。
4165	指定されたオブジェクトが見つかりません。URL が間違っている可能性があります。
4166	プロキシの障害です。
4167	転送が、クライアントによって割り込まれました。
4242	ダウンロードが netAbort(url) によって停止されました。
4836	ダウンロード停止の理由は不明です。ネットワークエラーが発生したか、またはダウンロードが中断された可能性があります。
4153	ネットワーク操作に失敗しました
4154	タイムアウトのため操作が完了しませんでした
4155	メモリ不足のため処理できません
4157	処理が認証されませんでした
4159	URL が無効です
4160	ネットワーク操作に失敗しました

4161	ネットワーク操作に失敗しました
4162	ネットワーク操作に失敗しました
4163	ネットワーク操作に失敗しました
4164	ソケットを作成できませんでした
4165	指定されたオブジェクトが見つかりません。URL が間違っている可能性があります
4166	プロキシの障害です
4167	転送が、クライアントによって割り込まれました
4168	ネットワーク操作に失敗しました
4240	ネットワークエクストラが適切に初期化されていません。（Lewis Francis）
4242	ダウンロードが netAbort(url) によって停止されました。
4836	ダウンロード停止の理由は不明です。ネットワークエラーが発生したか、またはダウンロードが中断された可能性があります。

パラメータ

netID オプション。テストするネットワーク操作の ID を指定します。

例文

次のステートメントは、特定のネットワーク操作のエラーステータスを確認するために、netError にネットワーク ID を渡します。

```
--Lingo syntax
on exitFrame
    global mynetID
    if netError(mynetID)<>"OK" then beep
end

// JavaScript syntax
function exitFrame() {
    global mynetID;
    if (netError(mynetID) != "OK") {
        _sound.beep();
    }
}
```

netLastModDate()

使用法

```
netLastModDate()
```

解説

指定した項目の HTTP ヘッダから最後に変更された日付のストリングを返す関数です。ストリングは、グリニッジ標準時（GMT）を使用して、「**Ddd, nn Mmm yyyy hh:mm:ss** GMT」のフォーマットで表します。例えば、「Thu, 30 Jan 1997 12:00:00 AM GMT」のようになります。曜日や月がフルスペルで示されることもあります。このストリングは常に英語で表記されます。

netLastModDate 関数は、操作が正常に完了したことが netDone および netError によって報告された後にのみ呼び出せます。次の操作が開始されると、メモリを節約するために Director ムービーやプロジェクトが以前の操作の結果を削除します。

実際の日付のストリングはサーバーから渡される形式で HTTP ヘッダから直接取り出されますが、ストリングが渡されない場合があります。その場合、netLastModDate は EMPTY を返します。

パラメータ

なし

例文

次のステートメントは、インターネットからダウンロードされたファイルの日付を調べます。

```
-- Lingo syntax
if netDone() then
    theDate = netLastModDate()
    if theDate.char[6..11] <> "Jan 30" then
        alert "The file is outdated."
    end if
end if

// JavaScript syntax
if (netDone()) {
    theDate = netLastModDate();
    if (theDate.char[6..11] != "Jan 30") {
        alert("The file is outdated");
    }
}
```

関連事項

netDone(), netError()

netMIME()

使用法

```
netMIME()
```

解説

最後のネットワーク操作によって返された MIME タイプのインターネットファイル（最後にダウンロードされた HTTP または FTP 項目）を返す関数です。

netMIME 関数は、操作が正常に完了したことが netDone または netError によって報告された後にのみ呼び出せます。次の操作が開始されると、メモリを節約するために Director ムービーやプロジェクトが以前の操作の結果を削除します。

パラメータ

なし

例文

次のハンドラは、インターネットからダウンロードした項目の MIME タイプを調べて、それに応じた反応をします。

```
-- Lingo syntax
on checkNetOperation theURL
    if netDone (theURL) then
        set myMimeType = netMIME()
        case myMimeType of
            "image/jpeg": go frame "jpeg info"
            "image/gif": go frame "gif info"
            "application/x-director": goToNetMovie theURL
            "text/html": gotoNetPage theURL
            otherwise: alert "Please choose a different item."
        end case
    else
        go the frame
    end if
end

// JavaScript syntax
function checkNetOperation(theURL) {
    if (netDone(theURL)) {
        myMimeType = netMIME();
        switch (myMimeType) {
            case "image/jpeg":
                _movie.go("jpeg info");
                break;
            case "image/gif":
                _movie.go("gif info");
                break;
            case "application/x-director":
                goToNetMovie(theURL);
                break;
            case "text/html":
                gotoNetPage(theURL);
                break;
            default:
                alert("Please choose a different item.");
        }
    } else {
        _movie.go(_movie.frame);
    }
}
```

関連事項

netDone(), netError(), getNetText(), postNetText, preloadNetThing()

netStatus

使用法

```
netStatus msgString
```

解説

ブラウザウィンドウのステータス領域に、指定したストリングを表示するためのコマンドです。

netStatus コマンドは、プロジェクタでは使用できません。

パラメータ

msgString 必須。表示するストリングを指定します。

例文

次のステートメントは、ムービーを実行中のブラウザウィンドウのステータス領域に " これはテストです。" というストリングを表示します。

```
-- Lingo syntax
on exitFrame
    netStatus "This is a test"
end

// JavaScript syntax
function exitFrame() {
    _movie.netStatus("This is a test");
}
```

netTextResult()

使用法

```
netTextResult(netID)
netTextResult()
```

解説

ネットワーク操作で得たテキストを返す関数です。[ネットワーク ID] を指定しない場合、netTextResult は、最後に行われたネットワーク操作の結果を返します。

指定したネットワーク操作が getNetText() の場合、ネットワーク上のファイルのテキストが返されます。

指定したネットワーク操作が postNetText の場合、サーバからの応答が返されます。

次の操作が開始されると、メモリを節約するために Director は以前の操作の結果を削除します。

Director は、すべての再生環境での過去 6 か月間の操作の結果を維持します。netTextResult() を使用してデータを取得すると、結果はそのネットワーク要求に対して削除されます。

パラメータ

netID オプション。返すテキストを含むネットワーク操作の ID を指定します。

例文

次のハンドラは、最後に行われたネットワーク操作が成功したかどうかを "netDone 関数と netError" 関数を使用して調べます。終了している場合は、netTextResult が返したテキストを "Display Text" フィールドキャストメンバに表示します。

```
-- Lingo syntax
global gNetID
on exitFrame
    if (netDone(gNetID) = TRUE) and (netError(gNetID) = "OK") then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
global gNetID;
function exitFrame() {
    if (netDone(gNetID) && (netError(gNetID) == "OK")) {
        member("Display Text").text = netTextResult();
    }
}
```

関連事項

netDone(), netError(), postNetText

new()

使用法

```
new(type)
new(type, castLib whichCast)
new(type, member whichCastMember of castLib whichCast)
variableName = new(parentScript arg1, arg2, ...)
new(script parentScriptName, value1, value2, ...)
timeout("name").new(timoutPeriod, #timeoutHandler, {, targetObject})
new(xtra "xtraName")
```

解説

新規キャストメンバ、子オブジェクト、タイムアウトオブジェクト、エクストラインスタンスを作成し、子オブジェクトに個々のプロパティの値を割り当てられるようにする関数です。

キャストメンバの場合、[タイプ] パラメータがキャストメンバのタイプを特定します。利用できる定義済みの値は、既存のキャストメンバのタイプの #bitmap、#field などに対応します。new 関数ではエクストラキャストメンバタイプを作成することもでき、ユーザが任意に選択した名前で識別することができます。

カスタムカーソルエクストラを使用して新規のカラーカーソルを作成することもできます。そのためにはまず new(#cursor) を使って新規キャストメンバを作成してから castMemberList などのプロパティを設定します。

オプションの [キャストメンバ] と [キャスト] パラメータには、新規キャストメンバを入れるキャストメンバスロットとキャストウィンドウを指定することができます。キャストメンバスロットが指定されていない場合は、最初の空きスロットが使用されます。また、その場所は new 関数によって返されます。

new 関数のパラメータが親スクリプトの場合は、new 関数が子オブジェクトを作成します。この場合、親スクリプトには子オブジェクトの初期状態やプロパティ値を設定する on new ハンドラが必要です。このハンドラは子オブジェクトへの me 参照を返す必要があります。

子オブジェクトは、親スクリプトにあるすべてのハンドラを含みます。また、親スクリプト内で定義されているものと同じプロパティ変数名も含むことになりますが、個々の子オブジェクトはこれらのプロパティに対して独自の値を含むことができます。

子オブジェクトは 1 つの値なので、変数への代入やリストへの格納、またはパラメータとして渡すことができます。

他の変数と同様に、子オブジェクトに関する情報は、put コマンドを使ってメッセージウィンドウに表示させることができます。

new() を使用してタイムアウトオブジェクトを作成するとき、timeoutPeriod には、タイムアウトオブジェクトが送るタイムアウトイベント間の時間をミリ秒単位で指定します。[# タイムアウトハンドラ] は、タイムアウトイベントが発生するたびに呼び出されるハンドラを識別するシンボルです。[ターゲットオブジェクト] は、[# タイムアウトハンドラ] が含まれる子オブジェクト名を識別します。[ターゲットオブジェクト] が与えられなければ、[# タイムアウトハンドラ] はムービースクリプト内にあると仮定されます。タイムアウト作成シンタックスは、scriptExecutionStyle の設定によって異なる場合があります。

```
-- Lingo syntax when scriptExecutionStyle is set to 9
x = timeout(name).new(period,handler,targetData)

-- Lingo syntax when scriptExecutionStyle is set to 10
x = timeout().new(name, period, handler, targetData)
y = new timeout(name, period, handler, targetData)

// JavaScript syntax
x = new timeout(name, period, function, targetData)
```

タイムアウトオブジェクトを作成すると、[ターゲットオブジェクト] は prepareMovie、startMovie、stopMovie、prepareFrame、exitFrame などのシステムイベントを受けることが可能となります。この機能を活用するには、[ターゲットオブジェクト] がそれらのイベントのハンドラを含んでいる必要があります。イベントを渡さなくても、ムービーの残りの部分はそれらのイベントを使用できます。

注意：Lingo で作成したタイムアウトオブジェクトから JavaScript シンタックスの関数を呼び出すことが可能であり、またその逆も可能です。

完成したムービーの new() の実例は、Director アプリケーションフォルダの Learning\Lingo_Examples フォルダ内の parent scripts.dir と read and write text.dir を参照してください。

例文

次のシンタックスは、最初の空きスロットに新しいビットマップキャストメンバを作成し、

```
set newMember = new(#bitmap)
```

キャストメンバへのメンバ参照が newMember に格納されます。

```
put newMember
-- (member 1 of castLib 1)
```

JavaScript シンタックスを使用して新しいキャストメンバを作成している場合は、ムービーオブジェクトの newMember() メソッドを使用します。次のステートメントは、新しいビットマップキャストメンバを作成します。

```
var tMem = _movie.newMember(symbol("bitmap"))
```

次の startMovie スクリプトは、new コマンドを使用して新しい Flash キャストメンバを作成します。このキャストメンバの linked プロパティを設定することで、キャストメンバのアセットが外部ファイルに格納されるようにします。さらに、キャストメンバの pathName プロパティに WWW 上の Flash ムービーの位置を設定します。

```
on startMovie
    flashCastMember = new(#flash)
    member(flashCastMember).pathName = "http://www.someURL.com/myFlash.swf"
end
```

ムービーが始まると、このハンドラは新規にアニメーションカラーカーソルキャストメンバを作成し、そのキャストメンバの番号を変数 customCursor に格納します。この変数は、新規に作成されたカーソルの castMemberList プロパティを設定し、新しいカーソルに切り替えるために使用されます。

```
on startmovie
    customCursor = new(#cursor)
    member(customCursor).castMemberList = [member 1, member 2, member 3]
    cursor (member(customCursor))
end
```

次のステートメントは親スクリプトの一部で、子オブジェクトを作成する on new ハンドラが含まれます。親スクリプトは " 鳥 " という名のスクリプトキャストメンバで、これらのハンドラが含まれます。

```
on new me, nameForBird
    return me
end

on fly me
    put "I am flying"
end
```

次の最初のステートメントは、前の例で説明した親スクリプトの子オブジェクトを作成し、それを変数 myBird に割り当てます。2 番目のステートメントは親スクリプト "Bird" の fly ハンドラを呼び出し、実行します。

```
myBird = script("Bird").new()
myBird.fly()
```

次の例は、speed プロパティ変数を含む新たな親スクリプト " 鳥 " です。

```
property speed
on new me, initSpeed
    speed = initSpeed
    return me
end
on fly me
    put "I am flying at " & speed & "mph"
end
```

次のステートメントは、"myBird1" と "myBird2" という 2 つの子オブジェクトを作成し、それぞれの初期速度として 15 および 25 を設定します。それぞれの子オブジェクトの fly ハンドラが呼び出されると、オブジェクトの速度がメッセージウィンドウに表示されます。

```
myBird1 = script("Bird").new(15)
myBird2 = script("Bird").new(25)
myBird1.fly()
myBird2.fly()
```

次のメッセージがメッセージウィンドウに表示されます。

```
-- "I am flying at 15 mph"
-- "I am flying at 25 mph"
```

次のステートメントは、"intervalTimer" という新規のタイムアウトオブジェクトを作成します。このオブジェクトは、子オブジェクト "playerOne" 内の minuteBeep ハンドラに対して、60 秒ごとにタイムアウトイベントを送ります。

```
timeout("intervalTimer").new(60000, #minuteBeep, playerOne)
```

次のステートメントは、JavaScript 関数の例を作成します。

```
function sampleTimeout () { trace("hello"); }
```

ムービー内の任意の部分で、次のステートメントを使用して、JavaScript 関数を呼び出すタイムアウトオブジェクトを作成できます。

```
-- Lingo syntax
gTO = timeout().new("test",50,"sampleTimeout",0)

// JavaScript syntax
_global.gTO = new timeout("test",50,"sampleTimeout",0)
```

関連事項

on stepFrame, actorList, ancestor, end, type（メンバ）, timeout()

newCamera

使用法

```
member(whichCastmember).newCamera(newCameraName)
```

解説

3D コマンド。キャストメンバ内に新規カメラを作成します。

パラメータ

newCameraName 必須。新規カメラの名前を指定します。新規カメラの名前は、キャストメンバ内で一意でなければなりません。

例文

次のステートメントは、" 車内カメラ " というカメラを新規作成します。

```
--Lingo
member("3D World").newCamera("in-car camera")

// Javascript
member("3D World").newCamera("in-car camera") ;
```

newColorRatio

使用法

```
<ColorRatio> newColorRatio(color, ratio, alpha)
```

解説

グローバル関数。グラデーショングローまたはグラデーションベベルフィルタを使用するときに異なる円周レベルで適用されるカラープロパティを記述するのに使用される、新しい ColorRatio オブジェクトを作成します。

パラメータ

パラメータ	解説
color	必須。カラーの赤、緑、青のコンポーネントから構成されるカラーオブジェクト。
ratio	必須。0 ～ 255 の値で、グラデーションフィルタの ColorList 内の現在の ColorRatio オブジェクトに関するカラー配布の開始点を決定します。
alpha	オプション。カラーの透明度（0 ～ 255）。

例文

次のコードスニペットは、GradientGlowFilter を作成し、フィルタに様々なカラー比率を割り当てます。

```
--Lingo
on mouseUp me
     fil = filter(#GradientGlowFilter, [#distance:0, #blurX:10, #blurY:10, #strength:1.3, #inner:0])
     fil.colorList.append(newColorRatio(color(255,255,255),1,0))
     fil.colorList.append(newColorRatio(color(255,0,0),63,255))
     fil.colorList.append(newColorRatio(color(255,255,0),126,25))
     fil.colorList.append(newColorRatio(color(0,204,255),255,255))
     sprite(me.spriteNum).filterList.append(fil)
end
//Java Script
function mouseUp(me)
{
     fil = filter(symbol("GradientGlowFilter"), propList(symbol("distance"),0, symbol("blurX"), 10,
symbol("blurY"), 10, symbol("strength"), 1.3, symbol("inner"), 0));
     fil.colorList.append(newColorRatio(color(255,255,255),1,0));
     fil.colorList.append(newColorRatio(color(255,0,0),63,255));
     fil.colorList.append(newColorRatio(color(255,255,0),126,255));
     fil.colorList.append(newColorRatio(color(0,204,255),255,255));
     sprite(me.spriteNum).filterList.append(fil);
}
```

newCurve()

使用法

```
-- Lingo syntax
memberObjRef.newCurve(positionInVertexList)

// JavaScript syntax
memberObjRef.newCurve(positionInVertexList);
```

解説

ベクトルシェイプに新規のシェイプを追加する [ベクトルキャストメンバ] の vertexList に、#newCurve 記号を追加する関数です。一連の頂点の中間で newCurve() を呼び出すことにより、既存のシェイプを分割することができます。

パラメータ

positionInVertexList 必須。vertexList 内の #newCurve 記号が追加される位置を指定します。

例文

次のステートメントは、キャストメンバの vertexList の 3 番目に位置するキャストメンバ 2 に新規のカーブを追加します。例文の 2 行目では、カーブ 2 の内容をカーブ 3 の内容で置き換えます。

```
-- Lingo syntax
member(2).newCurve(3)
member(2).curve[2] = member(2).curve[3]

// JavaScript syntax
member(2).newCurve(3);
member(2).curve[2] = member(2).curve[3]
```

関連事項

curve, vertexList

newGroup

使用法

```
member(whichCastmember).newGroup(newGroupName)
```

解説

3D コマンド。新規グループを作成して、グループパレットに追加します。

パラメータ

newGroupName 必須。新規グループの名前を指定します。新規グループの名前は、グループパレット内で一意でなければなりません。

例文

次のステートメントは、キャストメンバ "Scene" 内に "gbGroup2" というグループを作成し、このグループへの参照を変数 ng に格納します。

```
-- Lingo
ng = member("Scene").newGroup("gbGroup2")
// Javascript
var ng = member("Scene").newGroup("gbGroup2") ;
```

newLight

使用法

```
member(whichCastmember).newLight(newLightName, #typeIndicator)
```

解説

3D コマンド。指定した種類の新規ライトを作成して、ライトパレットに追加します。

パラメータ

newLightName 必須。新規ライトの名前を指定します。新規ライトの名前は、ライトパレット内で一意でなければなりません。

typeIndicator 必須。新規ライトの種類を指定する記号です。有効な値は以下のとおりです。

- #ambient。3D ワールドの汎用ライトです。
- #directional。特定の方向からのライトです。
- #point。裸電球のような光源です。
- #spot。スポットライト効果です。

例文

次のステートメントは、"3D ワールド " というキャストメンバ内に新規のライトを作成します。これは " 室内用の周囲ライト " という名前の周囲ライトにします。

```
-- Lingo
member("3D World").newLight("ambient room light", #ambient)

// Javascript
member("3D World").newLight("ambient room light", symbol("ambient"));
```

newMatrix()

使用法

```
<Matrix> newMatrix(numRows, numColumns, {elementList})
```

解説

グローバル関数。指定した数の行と列を持つ新規マトリックスオブジェクトを作成します。行と列のインデックスは 1 から開始します。

注意：newmatrix を使用して作成されたマトリックスの値をデバッグモードで確認するには、getval(i,j) メソッドを使用して値を取得します。

パラメータ

numRows 必須。マトリックス内の行の数を指定します。

numColumns 必須。マトリックス内の列の数を指定します。

elementList 必須。浮動小数点数のリニアリストです。

例文

次のステートメントは、最初の行を [1,2,3]、2 番目の行を [4,5,6] として 2 つの行と 3 つの列を持つマトリックスを作成します。

```
-- Lingo
mat = newMatrix(2, 3, [1,2,3,4,5,6])

// Javascript
mat = newMatrix(2, 3, list(1,2,3,4,5,6));
```

関連事項

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose()

newMember()

使用法

```
-- Lingo syntax
_movie.newMember(symbol)
_movie.newMember(stringMemberType)

// JavaScript syntax
_movie.newMember(stringMemberType);
```

解説

新規キャストメンバを作成し、子オブジェクトに個々のプロパティの値を割り当てられるようにするムービー関数です。

キャストメンバの場合、**symbol** または **stringMemberType** パラメータの設定がキャストメンバのタイプを特定します。利用できる定義済みの値は、既存のキャストメンバのタイプの #bitmap、#field などに対応します。newMember() メソッドではエクストラキャストメンバタイプを作成することもでき、ユーザが任意に選択した名前で識別することができます。

カスタムカーソルエクストラを使用して新規のカラーカーソルを作成することもできます。newMember(#cursor) を使用し、作成されるキャストメンバのプロパティを設定して、それらを使用できるようにします。

newMember() を呼び出した場合、新しいキャストメンバは最初の空のキャストライブラリスロットに配置されます。

完成したムービーの newMember() の実例は、Director アプリケーションフォルダの Learning\Lingo_Examples フォルダ内の parent scripts.dir と read and write text.dir を参照してください。

パラメータ

symbol （Lingo のみ）必須。新規キャストメンバの種類を指定する記号です。

stringMemberType 必須。新規キャストメンバの種類を指定するストリングです。

例文

次のステートメントは、新規ビットマップキャストメンバを作成し、変数 newBitmap に割り当てます。

```
-- Lingo syntax
newBitmap = _movie.newMember(#bitmap) -- using a symbol
newBitmap = _movie.newMember("bitmap") -- using a string

// JavaScript syntax
var newBitmap = _movie.newMember(symbol("bitmap")) ; //using a symbol
var newBitmap = _movie.newMember("bitmap") ; // using a string
```

関連事項

ムービー , type （メンバ）

newMesh

使用法

```
member(whichCastmember).newMesh(name,numFaces, numVertices,
numNormals,numColors,numTextureCoordinates)
```

解説

新規メッシュモデルリソースを作成する 3D コマンドです。メッシュの作成後、ジオメトリを実際に生成するには、新規メッシュに対して build() コマンドを呼び出す前に、少なくとも vertexList および face[インデックス].vertices プロパティの値を設定する必要があります。

パラメータ

meshName 必須。新規メッシュモデルリソースの名前を指定します。

numFaces 必須。メッシュ内に必要な三角形の合計数を指定します。

numVertices 必須。すべての（三角形の）フェイスで使用される頂点の合計数を指定します。1 つの頂点を複数のフェイスで共有できます。

numNormals オプション。法線の合計数を指定します。1 つの法線を複数のフェイスで共有できます。三角形の角の法線は、どちらの方向が外側かを決めるもので、その角がライトでどのように照らされるかに影響を与えます。メッシュの generateNormals() コマンドを使用して法線を生成する場合は、このパラメータに 0 を指定するか、または省略します。

numColors オプション。すべてのフェイスで使用されるカラーの合計数を指定します。1 つのカラーを複数のフェイスで共有できます。フェイスのそれぞれの角に対して 1 つのカラーを指定できます。滑らかなカラーグラデーション効果を出すにはカラーを指定します。フェイスの角ごとにデフォルトのホワイトカラーを取得するには、このパラメータに 0 を指定するか、または省略します。

numTextureCoordinates オプション。すべてのフェイスで使用される、ユーザ指定のテクスチャ座標の数を指定します。プレーンマッピングで生成されたデフォルトのテクスチャ座標を取得するには、このパラメータに 0 を指定するか、または省略します (詳細については、shader.textureWrapMode の #planar の説明を参照してください)。テクスチャをメッシュのフェイスに正確にマッピングする必要がある場合は、テクスチャ座標を指定します。

例文

次の例は、#mesh というタイプのモデルリソースを作成して、プロパティを指定し、このモデルリソースから新規モデルを作成します。プロセスの概略は次のようになります。

1 行目では、6 つのフェイスを持ち、5 つの一意の頂点と 3 つの一意のカラーで構成されるメッシュを作成します。法線数と textureCoordinates の数は設定しません。法線は、generateNormals コマンドで作成されます。

2 行目では、メッシュのフェイスで使用される 5 つの一意の頂点を定義します。

3 行目では、メッシュのフェイスで使用される 3 つの一意のカラーを定義します。

4 ～ 9 行目では、Pyramid 内の各フェイスの角として使用する頂点を割り当てます。頂点の順序は時計回りです。GenerateNormals() は、時計回りの順であることを前提としています。

10 ～ 15 行目は、各フェイスの角にカラーを設定します。カラーはグラデーションを付けてフェイスに広がります。

16 行目では、generateNormals() コマンドを呼び出すことで、Triangle の法線を作成します。

17 行目では、build コマンドを呼び出してメッシュを構築します。

```
nm = member("Shapes").newMesh("pyramid",6 , 5, 0, 3)
nm.vertexList = [ vector(0,0,0), vector(40,0,0), vector(40,0,40), vector(0,0,40), vector(20,50,20) ]
nm.colorList = [ rgb(255,0,0), rgb(0,255,0), rgb(0,0,255) ]
nm.face[1].vertices = [ 4,1,2 ]
nm.face[2].vertices = [ 4,2,3 ]
nm.face[3].vertices = [ 5,2,1 ]
nm.face[4].vertices = [ 5,3,2 ]
nm.face[5].vertices = [ 5,4,3 ]
nm.face[6].vertices = [ 5,1,4 ]
nm.face[1].colors = [3,2,3]
nm.face[2].colors = [3,3,2]
nm.face[3].colors = [1,3,2]
nm.face[4].colors = [1,2,3]
nm.face[5].colors = [1,3,2]
nm.face[6].colors = [1,2,3]
nm.generateNormals(#flat)
nm.build()
nm = member("Shapes").newModel("Pyramid1", nm)
```

関連事項

newModelResource

newModel

使用法

```
member( whichCastmember ).newModel( newModelName {, whichModelResource } )
```

解説

3D コマンド。参照したキャストメンバ内に新規モデルを作成します。デフォルトでは、すべての新規モデルの resource プロパティは VOID に設定されています。

パラメータ

newModelName 必須。新規モデルの名前を指定します。新規モデルの名前は一意である必要があります。

whichModelResource オプション。モデルの作成元となるモデルリソースを指定します。

例文

次のステートメントは、キャストメンバ "3D ワールド " 内に " 新しい家 " というモデルを作成します。

```
-- Lingo
member("3D World").newModel("New House")

// Javascript
member("3D World").newModel("New House") ;
```

新規モデルのモデルリソースを、オプションの whichModelResource パラメータで設定することもできます。

```
member("3D World").newModel("New House", member("3D World").modelResource("bigBox"))
```

newModelResource

使用法

```
member(whichCastmember).newModelResource(newModelResourceName { ,#type, #facing })
```

解説

3D コマンド。特定のタイプおよびフェイシング（指定されている場合）に基づいて新規モデルリソースを作成し、このモデルリソースパレットに追加します。

facing パラメータを指定せずに、**type** パラメータに #box、#sphere、#particle または #cylinder を指定すると、表面のフェイスのみが生成されます。また、#plane を指定すると、表面と裏面の両方が生成されます。#plane タイプのモデルリソースには 2 つのメッシュが生成され (各面に 1 つ)、その結果、shaderList には 2 つのシェーダが設定されます。

#both のフェイシングを指定すると、2 倍のメッシュが作成され、その結果、shaderList には 2 倍のシェーダエントリが作成されます。プレーンと球体に 2 (それぞれのモデルの内側と外側)、キューブに 12 (外側に 6、内側に 6)、円柱に 6 (外側と内側の上面、側面、底面) となります。

パラメータ

newModelResourceName 必須。新規モデルリソースの名前を指定します。

type オプション。新規モデルリソースのプリミティブタイプを指定します。有効な値は以下のとおりです。

- #plane
- #box
- #sphere
- #cylinder
- #particle

facing オプション。新規モデルリソースのフェイスを指定します。有効な値は以下のとおりです。

- #front
- #back
- #both

例文

次のハンドラは、ボックスを作成します。ハンドラの 1 行目では、"box10" というモデルリソースを新規作成します。タイプは #box で、裏面のみを表示するように設定します。2 ～ 4 行目では、"box10" のサイズを設定し、最後の行では、モデルリソースとして "box10" を使用する新規モデルを作成します。

```
on makeBox
    nmr = member("3D").newModelResource("box10", #box, #back)
    nmr.height = 50
    nmr.width = 50
    nmr.length = 50
    aa = member("3D").newModel("gb5", nmr)
end
```

次のステートメントは、" 帽子箱 4" というボックス型のモデルリソースを作成します。

```
member("Shelf").newModelResource("hatbox4", #box)
```

関連事項

primitives

newMotion()

使用法

```
member(whichCastmember).newMotion(name)
```

解説

3D コマンド。参照されたキャストメンバ内に新規のモーションを作成し、その新規モーションへの参照を返します。新規のモーションを使用すると、map() コマンドによって、メンバのモーションリストにある既存のモーションを組み合わせることができます。

パラメータ

name 必須。新規モーションの名前を指定します。新規モーションの名前は、参照されたキャストメンバ内で一意でなければなりません。

例文

次の Lingo は、メンバ 1 内に runWithWave という新規のモーションを作成します。この新規モーションは、メンバのモーションリストにある " 走る " および " 振る " というモーションを組み合わせています。

```
runWithWave = member(1).newMotion("runWithWave")
runWithWave.map("run", "pelvisBone")
runWithWave.map("wave", "shoulderBone")
```

newObject()

使用法

```
-- Lingo syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....})

// JavaScript syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....});
```

解説

Flash スプライトコマンド。指定されたタイプの ActionScript オブジェクトを作成します。

次のシンタックスは、Flash スプライト内でオブジェクトを作成します。

```
flashSpriteReference.newObject("objectType" {, arg1, arg2 ....})
```

次のシンタックスは、グローバルオブジェクトを作成します。

```
newObject("objectType" {, arg1, arg2 ....})
```

注意：まだ Flash キャストメンバをまったく読み込んでいない場合は、Shockwave Player およびプロジェクタ内でグローバル Flash コマンドを正しく機能させるために、Flash Asset Xtra をムービーの Xtra リストに手動で追加する必要があります。Xtra を Xtra リストに追加するには、修正／ムービー／エクストラ を選択します。配布するムービーの Xtra 拡張機能の管理の詳細については、ヘルプパネルで「Director ユーザガイド」のトピックを参照してください。

パラメータ

objectType 必須。作成する新規オブジェクトのタイプを指定します。

arg1, arg2, ... オプション . オブジェクトに必要な初期化パラメータを指定します。各パラメータの間はカンマで区切る必要があります。

例文

次の Lingo は、スプライト 3 の Flash ムービー内の新しい "LocalConnection" オブジェクトへの参照を変数 tLocalConObject に設定します。

```
-- Lingo syntax
tLocalConObject = sprite(3).newObject("LocalConnection")

// JavaScript syntax
var tLocalConObject = sprite(3).newObject("LocalConnection");
```

次の Lingo は、スプライト 3 の Flash ムービー内の新しい Array オブジェクトへの参照を変数 tArrayObject に設定します。配列には、23、34、19 の 3 つの整数値が含まれています。

```
-- Lingo syntax
tArrayObject = sprite(3).newObject("Array",23,34,19)

// JavaScript syntax
var tArrayObject = sprite(3).newObject("Array",23,34,19);
```

関連事項

setCallback(), clearAsObjects()

newShader

使用法

```
member(whichCastmember).newShader(newShaderName, #shaderType)
```

解説

3D コマンド。参照したキャストメンバのシェーダリスト内に、指定されたシェーダの種類のシェーダを新規作成し、そのシェーダへの参照を返します。

どの種類のシェーダにも、固有のプロパティグループがあり、その種類のシェーダで使用できます。さらに、すべてのシェーダの種類は #standard シェーダのプロパティにアクセスできます。ただし、#standard シェーダのプロパティを別の種類のシェーダに設定しても、そのプロパティが視覚効果を持たないこともあります。これは、もし #standard プロパティを適用してしまうと本来のシェーダタイプの性質が無効になるような場合です。例えば、標準の diffuseLightMap シェーダプロパティは、#engraver、#newsprint、および #painter の各タイプのシェーダでは無視されます。

パラメータ

newShaderName 必須。新規シェーダの名前を指定します。新規シェーダの名前は、シェーダリスト内で一意でなければなりません。

shaderType 必須。シェーダが適用されるスタイルを判断する記号です。有効な値は以下のとおりです。

- #standard シェーダは写実的なスタイルで、次のプロパティがあります。ambient、blend、blendConstant、blendConstantList、blendFunction、blendFunctionList、blendSource、blendSourceList、diffuse、diffuseLightMap、emissive、flat、glossMap、ilk、name、region、renderStyle、silhouettes、specular、specularLightMap、texture、textureMode、textureModeList、textureRepeat、textureRepeatList、textureTransform、textureTransformList、transparent、useDiffuseWithTexture、wrapTransform、wrapTransformList。
- #painter シェーダでは滑らかに処理され、絵画のような外観となります。#standard の全プロパティに加え、colorSteps、hilightPercentage、hilightStrength、name、shadowPercentage、shadowStrength、style の各プロパティを持ちます。
- #engraver シェーダは彫刻のような線を描きます。#standard の全プロパティに加え、brightness、density、name、rotation の各プロパティを持ちます。
- #newsprint シェーダは新聞印刷のような点描風のスタイルです。#standard の全プロパティに加え、brightness、density、name の各プロパティを持ちます。

例文

次のステートメントは、"newPainter" という #painter シェーダを作成します。

```
-- Lingo
newPainter = member("3D World").newShader("newPainter",#painter)

// Javascript
var newPainter = member("3D World").newShader("newPainter",symbol("painter")) ;
```

関連事項

shadowPercentage, shader, shaderList, deleteShader

newTexture

使用法

```
member(whichCastmember).newTexture(newTextureName {,#typeIndicator, sourceObjectReference})
```

解説

3D コマンド。参照したメンバのテクスチャパレット内に新規のテクスチャを作成し、そのテクスチャへの参照を返します。キャストメンバテクスチャは、そのキャストメンバを newTexture コンストラクタで指定した場合にのみ機能します。

パラメータ

newTextureName 必須。新規テクスチャの名前を指定します。新規テクスチャの名前は、参照されたキャストメンバのテクスチャパレット内で一意でなければなりません。

typeIndicator オプション。新規テクスチャのタイプを指定します。省略した場合、新規テクスチャは特定のタイプを使用せずに作成されます。有効な値は以下のとおりです。

- #fromCastMember（キャストメンバ）
- #fromImageObject（Lingo 画像オブジェクト）

sourceObjectReference オプション。ソースキャストメンバまたは Lingo 画像オブジェクトへの参照を指定します。省略した場合、新規テクスチャは特定のソースを使用せずに作成されます。[**sourceObjectReference**] は、**typeIndicator** が #fromCastMember の場合はキャストメンバを参照し、**typeIndicator** が #fromImageObject の場合は Lingo 画像オブジェクトを参照する必要があります。

例文

次のステートメントの 1 行目は、castlib 1 のキャストメンバ 5 から、" 芝生 02" という新規のテクスチャを作成します。2 行目では、" ブランク " という新規のテクスチャを作成します。

```
-- Lingo
member("3D World").newTexture("Grass 02",#fromCastMember,member(5,1))
member("3D World").newTexture("Blank")

// Javascript
member("3D World").newTexture("Grass 02",symbol("fromCastMember"),member(5,1)) ;
member("3D World").newTexture("Blank") ;
```

関連事項

deleteTexture, texture

normalize

使用法

```
normalize(vector)
vector.normalize()
```

解説

3D コマンド。**x**、**y**、および **z** コンポーネントをベクトルの大きさで割ることによってベクトルを正規化します。正規化されたベクトルの大きさは、常に 1 です。

パラメータ

なし

例文

次のステートメントは、ベクトル MyVec の正規化の前後に、ベクトルの値を示します。

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
MyVec.normalize()
put MyVec
-- vector(-0.1199, 0.9928, 0.0000)
put MyVec.magnitude
-- 1.0000

// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
MyVec.normalize();
trace(MyVec);
// vector(-0.1199, 0.9928, 0.0000)
trace(MyVec.magnitude);
// 1.0000
```

次のステートメントは、ベクトル ThisVector を正規化する前後の値を示します。

```
-- Lingo
ThisVector = vector(-50.0000, 0.0000, 0.0000)
normalize(ThisVector)
put ThisVector
-- vector(-1.0000, 0.0000, 0.0000)

// Javascript
ThisVector = vector(-50.0000, 0.0000, 0.0000);
normalize(ThisVector);
trace(ThisVector);
// vector(-1.0000, 0.0000, 0.0000)
```

関連事項

getNormalized, randomVector(), magnitude

nothing

使用法

```
nothing
```

解説

このコマンドは、if...then ステートメントの論理をより明確にするときに有効です。ネストした if...then...else ステートメントが else 節に明示的なコマンドを含まない場合は、Lingo が else 節を前の if 節の一部として解釈するのを防ぐため、else nothing が必要になることがあります。

パラメータ

なし

例文

次のハンドラでは、ネストした if...then...else ステートメントで nothing コマンドを使い、ステートメント内の else 節が正しく解釈されるようにしています。

```
-- Lingo syntax
on mouseDown
    if the clickOn = 1 then
        if sprite(1).moveableSprite = TRUE then member("Notice").text = "Drag the ball"
        else nothing
    else member("Notice").text = "Click again"
    end if
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 1) {
        if (sprite(1).moveableSprite) {
            member("Notice").text = "Drag the ball";
        } else {
            // do nothing
        }
    } else {
        member("Notice").text = "Click again";
    }
}
```

次のハンドラは、マウスボタンが押されている間は、ムービーに何もしないように指示します。

```
-- Lingo syntax
on mouseDown
    repeat while the stillDown
        nothing
    end repeat
end mouseDown

// JavaScript syntax
function mouseDown() {
    do {
        // do nothing
    } while _mouse.stillDown;
}
```

関連事項

if

nudge()

使用法

```
-- Lingo syntax
spriteObjRef.nudge(#direction)

// JavaScript syntax
spriteObjRef.nudge(#direction);
```

解説

特定の QuickTime VR スプライトの視点を指定された方向に少しずつ移動する QuickTime VR コマンドです。

このコマンドで右へ少しずつ移動するということは、スプライトの画像が左へ移動することを意味します。nudge コマンドは値を返しません。

パラメータ

direction 必須。視点を少しずつ移動する方向を指定します。有効な値は以下のとおりです。

- #down
- #downLeft
- #downRight
- #left
- #right
- #up
- #upLeft
- #upRight

例文

次のハンドラは、スプライト上でマウスボタンが押されている間、QTVR スプライトのパースを左方向に移動します。

```
-- Lingo syntax
on mouseDown me
    repeat while the stillDown
        sprite(1).nudge(#left)
    end repeat
end

// JavaScript syntax
function mouseDown() {
    do {
        sprite(1).nudge(symbol("left"));
    } while _mouse.stillDown;
}
```

numColumns()

使用法

```
<matrixObject>.numColumns
```

解説

Matrix のプロパティ。マトリックスの列の数を取得します。このプロパティでは、numColumns の取得のみが可能であり、列数の変更はできません。

例文

次のコードスニペットは、マトリックスを作成し、numRows と numColumns を出力します。

```
--Lingo
mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)
-- 2
-- 3
//Java Script
mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);
// 2
// 3
```

関連事項

getVal(), setVal(), numRows(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

numRows()

使用法

```
<matrixObject>.numRows
```

解説

Matrix のプロパティ。マトリックスの行の数を取得します。このプロパティでは、numRows の取得のみが可能であり、行数の変更はできません。

例文

次のコードスニペットは、マトリックスを作成し、numRows と numColumns を出力します。

```
-- Lingo
mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)
-- 2
-- 3

// Javascript
mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);
// 2
// 3
```

関連事項

getVal(), setVal(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

numToChar()

使用法

```
numToChar(integerExpression)
```

解説

指定した式の値に相当する ASCII コード番号を持つ単一文字を含むストリングを表示する関数です。外部ソースからのデータで、実際の文字ではなくコード番号で示されているものを文字として解釈するときに有効です。

127 までの ASCII コード番号はすべてのコンピュータに共通しています。128 以上はコンピュータによって異なるキーを参照します。

パラメータ

integerExpression 必須。対応する文字が返される ASCII コード番号を指定します。

例文

次のハンドラは、あらゆる任意のストリングを受け取り、アルファベットとアルファベット大文字を削除し、小文字を大文字に変換して返します。

```
-- Lingo syntax
on ForceUppercase input
    output = EMPTY
    num = length(input)
    repeat with i = 1 to num
        theASCII = charToNum(input.char[i])
        if theASCII = min(max(96, theASCII), 123) then
            theASCII = theASCII - 32
            if theASCII = min(max(63, theASCII), 91) then
                put numToChar(theASCII) after output
            end if
        end if
    end repeat
    return output
end

// JavaScript syntax
function ForceUpperCase(input) {
    output = "";
    num = input.length;
    for (i=1;i<=num;i++) {
        theASCII = charToNum (input.getProp("char",i);
        if (theASCII == list(list(96, theASCII).max(), 123).min()) {
            theASCII = theASCII - 32;
            if (theASCII == list(list(63, theASCII).max(), 91).min()) {
                output = output + numToChar (theASCII);
            }
        }
    }
    return output;
}
```

関連事項

charToNum()

objectP()

使用法

```
objectP(expression)
```

解説

指定した式が、親スクリプト、エクストラ、ウィンドウで生成されたオブジェクトかどうかを示す関数です。オブジェクトの場合は TRUE で、オブジェクトでない場合は FALSE になります。

objectP の "**P**" は "**predicate** (属性)" を意味しています。

親スクリプトやエクストラインスタンスでオブジェクトを作成するときに、既に使用されている項目を objectP を使って調べてみると、この関数の機能がわかります。

完成したムービーでの objectP() の実例は、Director アプリケーションフォルダの Learning\Lingo フォルダ内の read and write text.dir を参照してください。

パラメータ

expression 必須。テストする式を指定します。

例文

次の Lingo は、グローバル変数 gDataBase にオブジェクトが割り当てられているかどうかを調べて、割り当てられていない場合はオブジェクトを割り当てます。繰り返さないムービーまたはセクションの開始時に初期化する場合に、このステートメントを使用できます。

```
-- Lingo syntax
if objectP(gDataBase) then
    nothing
else
    gDataBase = script("Database Controller").new()
end if

// JavaScript syntax
if (objectP(gDataBase)) {
    // do nothing
} else {
    gDataBase = script("Database Controller").new();
}
```

関連事項

floatP(), ilk(), integerP(), stringP(), symbolP()

offset() (string function)

使用法

```
offset(stringExpression1, stringExpression2)
```

解説

ストリングの最初の文字の、別のストリング内における位置を示す整数を返す関数です。最初のストリングが 2 番目のストリングの中に存在しない場合、結果は 0 になります。いずれのストリングにおいても、スペースも 1 文字として数えられます。

Mac では、大文字と小文字または発音符号の区別は行われません。例えば、Mac では **a** と Å は同一文字として処理されます。

パラメータ

stringExpression1 必須。**stringExpression2** 内で検索するサブストリングを指定します。

stringExpression2 必須。サブストリング **stringExpression1** を含むストリングを指定します。

例文

次のステートメントは、"kleptomedia" ストリング内における "media" ストリングの先頭位置をメッセージウィンドウに表示します。

```
put offset("media","kleptomedia")
```

この結果は 7 になります。

次のステートメントは、"Adobe" ストリングにおける "Micro" ストリングの先頭位置をメッセージウィンドウに表示します。

```
put offset("Micro", "Adobe")
```

"Adobe" 内に "Micro" というストリングは存在しないため、結果は 0 になります。

次のハンドラは、input で指定されたストリング内を検索します。stringToFind で指定されたストリングが登場するすべての該当箇所を検索し、stringToInsert で指定されたストリングに置き換えます。

```
-- Lingo syntax
on SearchAndReplace input, stringToFind, stringToInsert
    output = ""
    findLen = stringToFind.length - 1
    repeat while input contains stringToFind
        currOffset = offset(stringToFind, input)
        output = output & input.char [1..currOffset]
        delete the last char of output
        output = output & stringToInsert
        delete input.char [1.. (currOffset + findLen)]
    end repeat
    set output = output & input
    return output
end

// JavaScript syntax
function SearchAndReplace(input, stringToFind, stringToInsert) {
    output = "";
    findLen = stringToFind.length - 1;
    do {
        currOffset = offset(stringToFind, input);
        output = output + input.char[0..currOffset];
        output = output.substr(0,output.length-2);
        output = output + stringToInsert;
        input = input.substr(currOffset+findLen,input.length);
    } while (input.indexOf(stringToFind) >= 0);
    output = output + input;
    return output;
}
```

関連事項

chars(), length(), contains, starts

offset() （矩形関数）

使用法

```
rectangle.offset(horizontalChange, verticalChange)
offset (rectangle, horizontalChange, verticalChange)
```

解説

[矩形] で指定した矩形からオフセット矩形を生成するための関数です。

パラメータ

horizontalChange 必須。水平オフセットをピクセルで指定します。[水平オフセット] が 0 より大きいときはステージの右側の方にオフセットされ、0 より小さいときはステージの左側の方にオフセットされます。

verticalChange 必須。垂直オフセットをピクセルで指定します。[垂直オフセット] が 0 より大きいときはステージの上側の方にオフセットされ、0 より小さいときはステージの下側の方にオフセットされます。

例文

次のハンドラは、スプライト 1 を右へ 5 ピクセル、下へ 5 ピクセル移動します。

```
-- Lingo syntax
on diagonalMove
    newRect=sprite(1).rect.offset(5, 5)
    sprite(1).rect=newRect
end

// JavaScript syntax
function diagonalMove() {
    newRect = sprite(1).rect.offset(5,5);
    sprite(1).rect = newRect;
}
```

open() （プレーヤ）

使用法

```
-- Lingo syntax
_player.open({stringDocPath,} stringAppPath)

// JavaScript syntax
_player.open({stringDocPath,} stringAppPath);
```

解説

指定されたアプリケーションを開き、オプションでアプリケーションを開くときに指定されたファイルを開くプレーヤメソッドです。

stringDocPath または **stringAppPath** が現行ムービーとは異なるフォルダに含まれている場合、ファイルへのフルパス名を指定する必要があります。

Director と他のアプリケーションを同時に動作させるためには、コンピュータに十分なメモリが必要です。

これは、アプリケーションを開いたり、アプリケーションからファイルを開いたりする単純なメソッドです。より複雑な制御を行う場合は、サードパーティの Xtra 拡張のオプションを利用します。

パラメータ

stringDocPath オプション。**stringAppPath** によって指定されたアプリケーションが開くときに開くドキュメントを指定するストリングです。

stringAppPath 必須。開くアプリケーションへのパスを指定するストリングです。

例文

次のステートメントは、HD ドライブのアプリケーションフォルダに入っている "TextEdit" を起動し（Mac）、同時に " 物語 " という名前のファイルを開きます。

```
-- Lingo syntax
_player.open("Storyboards", "HD:Applications:TextEdit")

// JavaScript syntax
_player.open("Storyboards", "HD:Applications:TextEdit");
```

関連事項

プレーヤ

open() （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.open()

// JavaScript syntax
windowObjRef.open();
```

解説

ウィンドウを開き、他のすべてのウィンドウの前に配置するウィンドウメソッドです。

open() が呼び出されたウィンドウにムービーが割り当てられていない場合は、ファイルを開くダイアログボックスが表示されます。

ウィンドウオブジェクト **windowObjRef** への参照をムービーのファイル名で置き換えた場合、ウィンドウはそのファイル名をウィンドウ名として使用します。ただし、その後ウィンドウの fileName プロパティを使用してウィンドウにムービーを割り当てる必要があります。

ウィンドウオブジェクト **windowObjRef** への参照をウィンドウ名で置き換えた場合、ウィンドウはその名前を受け取ります。ただし、その後ウィンドウの fileName プロパティを使用してウィンドウにムービーを割り当てる必要があります。

URL のムービーを使うウィンドウを開くには、downloadNetThing() を使用してムービーのファイルをローカルディスクにダウンロードしてから、ディスク上でそのファイルを使用します。これによりムービーをダウンロードする待ち時間の問題を最小にすることができます。

ローカルムービーを使用するときは、open() を呼び出す前に、preloadMovie() を使用して少なくともムービーの先頭フレームをロードします。これにより、ムービーのロードで遅延が生じる可能性を抑えることができます。

現行では、ブラウザで再生中にウィンドウのムービーを開くことはできません。

パラメータ

なし

例文

次のステートメントは、" 制御パネル " のウィンドウを開き、ステージの最前面に表示します。

```
-- Lingo syntax
window("Control Panel").open()

// JavaScript syntax
window("Control Panel").open();
```

関連事項

close(), downloadNetThing, fileName （ウィンドウ）, preLoadMovie(), ウィンドウ

openFile()

使用法

```
-- Lingo syntax
fileioObjRef.openFile(stringFileName, intMode)

// JavaScript syntax
fileioObjRef.openFile(stringFileName, intMode)
```

解説

指定したファイルを指定したモードで開くファイル IO メソッドです。

パラメータ

stringFileName 必須。開くファイルのフルパスと名前を指定するストリングです。

intMode 必須。ファイルのモードを指定する整数です。有効な値は以下のとおりです。

- 0 - 読み取り / 書き込み可能
- 1 - 読み取り専用
- 2 - 書き込み可能

例文

次のステートメントは、ファイル c:\xtra.txt を読み取り / 書き込み権限で開きます。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
```

関連事項

Fileio

openXlib

使用法

```
openXlib whichFile
```

解説

指定した Xlibrary ファイルを開くコマンドです。

既に使用し終わったファイルは、随時閉じるようにします。ファイルが開いているときに openXlib コマンドを実行しても効果はありません。

openXlib コマンドは、ファイル参照として URL をサポートしていません。

Xlibrary ファイルには、Xtra 拡張が含まれています。openResFile とは異なり、openXlib は Xtra 拡張を Director に認識させます。

openXlib を使用してスクリプティング Xtra 拡張機能を開く場合、Director が拡張機能を使用し終わったときに、closeXlib を使用して拡張機能を閉じる必要があります。

Windows では拡張子 ".dll" の指定はオプションです。

注意：このコマンドは、Shockwave Player ではサポートされていません。

パラメータ

whichFile 必須。開く Xlibrary ファイルを指定します。ファイルが現行ムービーとは異なるフォルダ内に入っている場合、[ファイル] にはパス名を指定する必要があります。

例文

次のステートメントは、"Video Disc Xlibrary" という Xlibrary ファイルを開きます。

```
openXlib "Video Disc Xlibrary"
```

次のステートメントは、現行ムービーとは異なるフォルダに入っている " 転送エクストラ " Xlibrary ファイルを開きます。

```
openXlib "My Drive:New Stuff:Transporter Xtras"
```

関連事項

closeXlib, Interface()

param()

使用法

```
param(parameterPosition)
```

解説

ハンドラに渡されるパラメータの値を示す関数です。

ハンドラでのエラーを防ぐために、この関数を使用して特定のパラメータの種類を判定できます。

パラメータ

parameterPosition 必須。ハンドラに渡されるパラメータ内のパラメータの位置を指定します。

例文

次のハンドラは任意の数のパラメータを受け取り、パラメータとして渡されたすべての数字を加算し、合計値を返します。

```
--Lingo syntax
on AddNumbers
    sum = 0
    repeat with currentParamNum = 1 to the paramCount
        sum = sum + param(currentParamNum)
    end repeat
    return sum
end

// JavaScript syntax
function AddNumbers() {
    sum = 0;
    for (currentParamNum=0;currentParamNum<arguments.length;currentParamNum++){
        sum = sum + arguments[currentParamNum];
    }
    return sum;
}
```

このハンドラを使用するときは、以下のように加算する数値をカンマ区切りで渡します。

```
put AddNumbers(3, 4, 5, 6)
-- 18
put AddNumbers(5, 5)
-- 10
```

関連事項

getAt(), paramCount(), return （キーワード）

paramCount()

使用法

```
the paramCount
```

解説

現行ハンドラに送られるパラメータの数を示す関数です。

パラメータ

なし

例文

次のステートメントは、現行ハンドラに送られるパラメータ数を変数 counter に設定します。

```
set counter = the paramCount
```

parseByteArray

使用法

```
xmlParser.parseByteArray(byteArrayXML)
```

解説

XML Xtra メソッド。XML 宣言タグを使用して、XML ファイルの正しいエンコーディングを判別します。

パラメータ

パラメータ	解説	必須 / オプション
byteArrayXML	解析するバイト配列。	必須

例文

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gFileIO.openFile("test.xml",1)
gFileIO.setPosition(0)
bArray = gFileIO.readByteArray(gFileIO.getLength())
gXMLParser.parseByteArray(bArray)

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gFileIO.openFile("test.xml",1);
gFileIO.setPosition(0);
bArray = gFileIO.readByteArray(gFileIO.getLength());
gXMLParser.parseByteArray(bArray);
```

parseString()

使用法

```
parserObject.parseString(stringToParse)
```

解説

Director ムービーに既に使用されている XML ドキュメントの構文解析に使用される関数です。最初のパラメータは、パーサーオブジェクトを含む変数です。戻り値は、演算が成功した場合は <VOID> であり、演算が失敗した場合はエラーコード番号ストリングです。失敗は通常、XML のシンタックスまたは構造に問題があるためです。演算が完了すると、パーサーオブジェクトには構文解析済み XML データが取り込まれます。

URL 先の XML を構文解析するには、parseURL() を使用します。

パラメータ

stringToParse 必須。解析する XML データのストリングを指定します。

例文

次のステートメントは、テキストキャストメンバ "XMLtext" 内の XML データを構文解析します。演算が完了すると、変数 gParserObject には構文解析済み XML データが取り込まれます。

```
-- Lingo
errorCode = gParserObject.parseString(member("XMLtext").text)

// Javascript
errorCode = gParserObject.parseString(member("XMLtext").text);
```

関連事項

getError() (XML), parseURL()

parseString (XML Xtra)

使用法

```
xmlParser.parseString(strXML)
```

解説

XML Xtra メソッド。入力ストリングのデフォルトの型として UTF-8 を設定します。

パラメータ

パラメータ	解説	必須 / オプション
strXML	解析するストリング。	必須

例文

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gFileIO.openFile("test.xml",1)
gFileIO.setPosition(0)
b = gFileIO.readFile()
gXMLParser.parseString(b)

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gFileIO.openFile("test.xml",1);
gFileIO.setPosition(0);
b = gFileIO.readFile();
gXMLParser.parseString(b);
```

parseURL()

使用法

```
parserObject.parseURL(URLstring {,#handlerToCallOnCompletion} {, objectContainingHandler})
```

解説

関数 ; 外部インターネットロケーションに存在する XML ドキュメントを構文解析します。最初のパラメータは、XML Parser エクストラのインスタンスを含むパーサーオブジェクトです。

この関数は非同期に実行されます。そのため、タイミングによっては、URL 全体がまだ構文解析されていない場合があります。doneParsing() 関数を parseURL() と組み合わせて使用して、構文解析演算がいつ完了したかを判断することが重要です。

この演算は非同期であり、処理の完了までしばらく時間がかかる場合があるので、オプションのパラメータを使用して演算の終了時に特定のハンドラを呼び出すことができます。

戻り値は、演算が成功した場合は VOID であり、演算が失敗した場合はエラーコード番号ストリングです。

ローカルで XML を構文解析するには、parseString() を使用します。

パラメータ

URLstring 必須。XML データが存在する実際の URL を指定します。

handlerToCallOnCompletion オプション。URL を完全に解析した後に実行されるハンドラの名前を指定します。

objectContainingHandler オプション。ハンドラ **handlerToCallOnCompletion** を含むスクリプトオブジェクトの名前を指定します。省略した場合、ハンドラはムービーハンドラであると仮定されます。

例文

次のステートメントは、MyCompany.com でファイル sample.xml を構文解析します。構文解析演算がいつ完了したかを判断するには、doneParsing() を使用します。

```
--Lingo syntax
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml")

// JavaScript syntax
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml");
```

注意：次の例では、Xtra のインスタンスが既に作成されていて、そのインスタンスへの参照が gParserObject という名前のグローバル変数に格納されていると仮定しています。

次の Lingo はファイル sample.xml を構文解析し、on parseDone ハンドラを呼び出します。doneParsing() 関数ではスクリプトオブジェクトが与えられないので、on parseDone ハンドラはムービースクリプト内にあると仮定されます。

```
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone)
```

ムービースクリプトには、on parseDone ハンドラがあります。

```
on parseDone
    global gParserObject
    if voidP(gParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & gParserObject.getError()
    end if
end
```

次の JavaScript シンタックスは、ファイル sample.xml を解析し、parseDone 関数を呼び出します。doneParsing() 関数ではスクリプトオブジェクトが与えられないので、parseDone 関数はムービースクリプト内にあると仮定されます。

```
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml", symbol("parseDone"));
```

注意：次の例では、Xtra のインスタンスが既に作成されていて、そのインスタンスへの参照が gParserObject という名前のグローバル変数に格納されていると仮定しています。

ムービースクリプトには、on parseDone ハンドラがあります。

```
// JavaScript syntax
function parseDone () {
    if (_global.gParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + _global.gParserObject.getError());
    }
}
```

次の Lingo は MyCompany.com でドキュメント sample.xml を構文解析し、解析が終了すると親スクリプト TestScript の子であるスクリプトオブジェクト testObject 内の on parseDone ハンドラを呼び出します。

```
parserObject = new(xtra "XMLParser")
testObject = new(script "TestScript", parserObject)
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone, testObject)
```

次に、親スクリプト TestScript を示します。

```
property myParserObject

on new me, parserObject
    myParserObject = parserObject
end

on parseDone me
    if voidP(myParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & myParserObject.getError()
    end if
end
```

次の JavaScript シンタックスは MyCompany.com でドキュメント sample.xml を構文解析し、解析が終了すると定義済みの TestScript クラスのインスタンスであるオブジェクト testObject 内の parseDone 関数を呼び出します。

```
parserObject = new xtra("XMLParser");
testObject = new TestScript(parserObject);
errorCode = parserObject .parseURL("http://www.MyCompany.com/sam- ple.xml", symbol("parseDone"),
testObject)
```

次に、TestScript クラスの定義を示します。

```
TestScript = function (aParser) {
    this.myParserObject = aParser;
}
TestScript.prototype.parseDone = function () {
    if (this.myParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + this.myParserObject.getError());
    }
}
```

関連事項

getError() (XML), parseString()

parseURL (XML Xtra)

使用法

```
xmlParser.parseURL(strXML)
```

解説

XML Xtra メソッド。XML 宣言タグを使用して、XML ファイルの正しいエンコーディングを判別します。このメソッドは xmlParser.parseByteArray(byteArrayXML) に似ています。

パラメータ

パラメータ	解説	必須 / オプション
strXML	解析するストリング。	必須

例文

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gXMLParser.parseURL("http://www.bytearray.com/byte.xml")

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gXMLParser.parseURL("http://www.bytearray.com/byte.xml");
```

pass

使用法

```
pass
```

解説

イベントメッセージをメッセージ階層の次の場所に渡し、1 つのイベントに対して複数のハンドラを実行できるようにするためのコマンドです。

pass コマンドは、コマンドの実行と同時に残りのステートメントをスキップし、次の場所にジャンプします。ハンドラ内で pass コマンドに続く Lingo は実行されません。

デフォルトでは、イベントメッセージはそのイベントに対応するハンドラに最初に遭遇した場所で止まります。通常、その場所はスプライトレベルです。

pass コマンドをハンドラに入れると、本来ハンドラに妨げられるはずのイベントをメッセージ階層の別のオブジェクトに渡すことができます。

パラメータ

なし

例文

次のハンドラは入力されたキーを調べ、有効な文字の場合には編集可能なテキストスプライトへそれらを渡します。

```
-- Lingo syntax
on keyDown me
    legalCharacters = "1234567890"
    if legalCharacters contains the key then
        pass
    else
        beep
    end if
end

// JavaScript syntax
function keyDown() {
    legalCharacters = "1234567890";
    if (legalCharacters.indexOf(_key.key) >= 0) {
        pass();
    } else {
        _sound.beep();
    }
}
```

関連事項
stopEvent()

pasteClipBoardInto()

使用法

```
-- Lingo syntax
memberObjRef.pasteClipBoardInto()

// JavaScript syntax
memberObjRef.pasteClipBoardInto();
```

解説
指定したキャストメンバにクリップボードの内容をペーストし、既存のキャストメンバを消去するメンバメソッドです。

ペーストできるのは、Director がキャストメンバとして使用できる書式の項目です。

別のアプリケーションからストリングをコピーした場合、そのストリングの書式は維持されません。

このメソッドは、他のムービーやアプリケーションからキャストウィンドウにオブジェクトをコピーするのに使用できます。コピーしたキャストメンバは RAM に保存する必要があるため、メモリ容量が少ない環境で再生している場合、このコマンドは使わないようにします。

Shockwave Player で、またはオーサリング環境において safePlayer プロパティを TRUE に設定したプロジェクタで、このメソッドを使用すると、警告ダイアログボックスが表示され、ペースト操作をキャンセルできます。

パラメータ
なし

例文
次のステートメントは、クリップボードの内容を " 神社 " ビットマップキャストメンバにペーストします。

```
-- Lingo syntax
member("shrine").pasteClipBoardInto()

// JavaScript syntax
member("shrine").pasteClipBoardInto();
```

関連事項
メンバ , safePlayer

pause() (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.pause()

// JavaScript syntax
dvdObjRef.pause();
```

解説
再生を一時停止する DVD メソッドです。

パラメータ
なし

例文
次のステートメントは、再生を一時停止します。

```
-- Lingo syntax
member(1).pause()

// JavaScript syntax
member(1).pause();
```

関連事項
DVD

pause() （ミクサ）

使用法
```
Mixer.pause([soundobjectlist])
```

解説
ミクサメソッド。サウンドオブジェクトリストが渡されていない場合にミクサを一時停止したり、リスト内のサウンドオブジェクトのリストを一時停止します。

ミクサの操作	効果
一時停止 ()	ミクサを一時停止します。
pause([so1, so2])	参照 so1 and so2 を使用してミクサ内のサウンドオブジェクトを一時停止します。
pause(["so1", "so2"])	名前 so1 および so2 を使用してミクサ内のサウンドオブジェクトを一時停止します。

例文
```
-- Lingo syntax
on mouseUp me
     mixer1.pause() --Pauses mixer1.
end

// JavaScript syntax
function mouseup()
{
mixer1.pause(); //Pauses mixer1.

}
```

関連事項
play() （ミクサ）, ミクサ

pause (MP4Media/FLV)

使用法

```
sprite(1).pause()
member(1).pause()
```

解説

MP4Media/FLV メンバまたはスプライトメソッド。メディアストリームの再生を一時停止します。mediaStatus の値は #paused に変わります。

MP4Media/FLV ストリームの再生中にこのメソッドを呼び出しても、currentTime プロパティは変化しません。したがって、play コマンドはストリームを再バッファしないで再生を再開します。

例文

次の例は、MP4Media/FLV キャストメンバに属するスプライト 2 を一時停止します。

```
-- Lingo syntax
sprite(2).pause()
member("MP4Media/FLV").pause()

// JavaScript syntax
sprite(2).pause();
member("MP4Media/FLV").pause();
```

関連事項

play() (MP4Media/FLV), isPlayable (MP4Media/FLV), ミクサ

pause() (3D)

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.pause()
member(whichCastmember).model(whichModel).keyframePlayer.pause()
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D コマンド。モデルによって現在実行されているモーションを一時停止します。モーションを再開するには、play() コマンドを使用します。

このコマンドを使ってモデルのモーションを一時停止すると、モデルの bonesPlayer.playing プロパティが FALSE に設定されます。

パラメータ

なし

例文

次のステートメントは、Ant3 というモデルの現在のアニメーションを一時停止します。

```
member("PicnicScene").model("Ant3").bonesplayer.pause()
```

関連事項

play() (3D), playing (3D), playlist

pause() (RealMedia, SWA, Windows Media)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.pause()

// JavaScript syntax
memberOrSpriteObjRef.pause();
```

解説

RealMedia および Windows Media スプライトまたはキャストメンバメソッド。メディアストリームの再生を一時停止します。

mediaStatus の値は #paused になります。

RealMedia または Windows Media ストリームの再生中にこのメソッドを呼び出しても、currentTime プロパティは変更されず、メディアバッファも消去されません。このため、以降に play コマンドを実行すると、ストリームを再バッファせずに再生を再開できます。

パラメータ

なし

例文

次の例は、スプライト 2 およびキャストメンバ "Real" の再生を一時停止します。

```
-- Lingo syntax
sprite(2).pause()
member("Real").pause()

// JavaScript syntax
sprite(2).pause();
member("Real").pause();
```

関連事項

mediaStatus (RealMedia, Windows Media), play() (RealMedia, SWA, Windows Media), seek(), stop() (RealMedia, SWA, Windows Media)

pause() （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.pause()

// JavaScript syntax
soundChannelObjRef.pause();
```

解説

サウンドチャネルの現行サウンドの再生を一時停止するサウンドチャネルメソッドです。

一時停止後、play() メソッドを発行すると、再生が再開されます。

パラメータ
なし

例文
次のステートメントは、サウンドチャネル 1 のサウンドの再生を一時停止します。

```
-- Lingo syntax
sound(1).pause()

// JavaScript syntax
sound(1).pause();
```

関連事項
breakLoop(), play() （サウンドチャネル）, playNext() （サウンドチャネル）, queue(), rewind() （サウンドチャネル）サウンドチャネル , stop() （サウンドチャネル）

pause （サウンドオブジェクト）

使用法

```
soundObject.pause()
```

解説
サウンドオブジェクトメソッド。サウンドオブジェクトと関連付けられたオーディファイルの再生を一時停止します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.pause() --Pauses the sound object associated with soundObjRef.
end


// JavaScript syntax
function mouseUp(){
soundObjRef.pause(); //Pauses the sound object associated with soundObjRef.
}
```

関連事項
play （サウンドオブジェクト）

perlinNoise()

使用法

```
-- Lingo syntax
member(membername).Image.perlinNoise(baseX, baseY, numOctaves, seed, bStitch, bFractalNoise,
[#channelOptions:value, #grayscale:bolean, #offsets:listOfPoints])
// JavaScript syntax
```

解説

Perlin ノイズ生成アルゴリズムは、個々のランダムなノイズ関数（オクターブといいます）を、より自然に見えるランダムノイズを生成する単一の関数に補間および結合します。

パラメータ

プロパティ	解説	値の範囲	デフォルト
baseX：数値	作成されるパターンの x（サイズ）値を決定します。		
baseY：数値	作成されるパターンの y（サイズ）値を決定します。		
numOctaves：数値	このノイズを作成するために組み合わされるオクターブまたは個々のノイズ関数の数です。オクターブの数を増やすと、詳細な画像が作成されますが、処理時間は長くなります。		
randomSeed：数値	ランダムシード数です。		
bStitch：ブール値	true に設定した場合、このメソッドは画像のトランジションエッジをステッチ処理（スムーズ化）し、ビットマップの塗りつぶしとしてタイリングするためのシームレスなテクスチャを作成しようと試みます。	True/false	
bFractalNoise：ブール値	このパラメータは、メソッドによって生成されるグラデーションのエッジに関連します。true に設定した場合、このメソッドは効果のエッジをスムーズ化するフラクタルノイズを生成します。false に設定した場合は、乱流を生成します。乱流を含む画像は、グラデーション内に可視の不連続性を持ちます。そのため、炎や海の波などのシャープな視覚効果に向いています。	True/false	
channelOptions：数値（省略可能）	ノイズパターンが適用される（ビットマップの）カラーチャネルを指定します。この数値は、4 つのカラーチャネル RGBA（1、2、4 および 8）の任意の組み合わせにすることができます。デフォルト値は 7 です。		0
grayScale：ブール値（省略可能）	true に設定した場合は、randomSeed 値をビットマップピクセルに適用し、画像のすべての色を洗い落とします。デフォルト値は false です。	0 または 1	0
Offsets：リストまたはポイント（省略可能）	各オクターブの x オフセットと y オフセットに対応するポイントのリストまたはポイントを指定できます。オフセット値を操作することで、画像のレイヤーをスムーズにスクロールできます。オフセット List 内の各ポイントは、特定のオクターブノイズ関数に影響を与えます。デフォルト値は null です。ポイントを指定した場合、すべてのオクターブノイズ関数に対して同じ値が使用されます。		ポイント (0,0)

例文

次の例は、スプライト 2 およびキャストメンバ "Real" の再生を一時停止します。

```
-- Lingo syntax
myList = [point(0,1), point(5,5)] --List of Points
member("myMember").Image.perlinNoise(300, 300, 2, 2, true, true, [#channelOptions:7, #grayscale:false,
#offsets:myList])
// JavaScript syntax
myList = list(point(0,1), point(5,5)) //List of Points
member("myMember").image.perlinNoise(300, 300, 2, 2, true,
true, propList(symbol("channelOptions"),7, symbol("grayscale"),false,
symbol("offsets"),myList));
```

perpendicularTo

使用法

```
vector1.perpendicularTo(vector2)
```

解説

3D ベクトルコマンド。オリジナルのベクトルと 2 番目のベクトルの両方に対して垂直なベクトルを返します。このコマンドは、ベクトルコマンド crossProduct と同じです。

パラメータ

vector2 必須。2 番目のベクトルを指定します。

例文

次の例では、pos1 は x 軸上のベクトルで、pos2 は y 軸上のベクトルです。pos1.perpendicularTo(pos2) によって返される値は、vector(0.0000, 0.0000, 1.00000e4) です。最後の 2 行は、pos1 と pos2 の両方に対して垂直なベクトルを示します。

```
-- Lingo
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.perpendicularTo(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
pos1 = vector(100, 0, 0);
pos2 = vector(0, 100, 0);
trace(pos1.perpendicularTo(pos2));
// vector( 0.0000, 0.0000, 1.00000e4 )
```

関連事項

crossProduct(), cross

pictureP()

使用法

```
-- Lingo syntax
pictureP(pictureValue)

// JavaScript syntax
pictureP(pictureValue);
```

解説

指定したキャストメンバの picture メンバプロパティの値を、TRUE (1) または FALSE (0) で返す関数です。

pictureP は、キャストメンバがピクチャを含んでいるかどうかを直接調べるわけではないため、先にキャストメンバの picture メンバプロパティでピクチャをチェックする必要があります。

パラメータ

pictureValue 必須。キャストメンバのピクチャへの参照を指定します。

例文

最初のステートメントは、キャストメンバの " 神社 " の picture メンバプロパティの値 (ビットマップ) を変数 pictureValue に割り当てます。次のステートメントは、"pictureValue" に割り当てられた値をチェックして、"Shrine" がピクチャかどうかをチェックします。

```
-- Lingo syntax
pictureValue = member("Shrine").picture
put pictureP(pictureValue)

// JavaScript syntax
var pictureValue = member("Shrine").picture;
put(pictureP(pictureValue));
```

結果は、TRUE を表す数値の 1 になります。

play() (3D)

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.play()
member(whichCastmember).model(whichModel).keyframePlayer.play()
member(whichCastmember).model(whichModel).bonesPlayer.play(motionName {, looped, startTime, endTime,
scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.play(motionName {, looped, startTime, endTime,
scale, offset})
```

解説

3D #keyframePlayer および #bonesPlayer のコマンド。モーションの実行を開始または再開します。

このコマンドを使ってモデルのモーションを開始または再開すると、モデルの bonesPlayer.playing プロパティが TRUE に設定されます。

pause() コマンドで一時停止されているモーションの実行を再開するには、パラメータを指定しないで play() を使用します。

play() コマンドを使ってモーションを開始すると、そのモーションは、モディファイアのプレイリストの先頭に挿入されます。この操作によって別のモーションの再生が中断された場合、中断されたモーションは、プレイリストの、新たに開始されたモーションの次の位置に残ります。新たに開始されたモーションが終了するか (ループしない場合)、または playNext() コマンドが実行されると、中断されたモーションの再生が、中断されたところから再開されます。

パラメータ

motionName 必須。実行するモーションの名前を指定します。**motionName** が play() に渡す唯一のパラメータである場合、モディファイアの playRate プロパティに設定されている速度で、モーションがモデルによって始めから終わりまで 1 回実行されます。

looped オプション。FALSE を指定するとモーションが一度だけ再生され、TRUE を指定すると繰り返し再生されます。

startTime オプション。モーションの始めからミリ秒単位で計測されます。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後モーションが繰り返し再生される際には、開始時間で始まり、終了時間で終了します。

endTime オプション。モーションの始めからミリ秒単位で計測されます。ループが FALSE の場合、モーションは、オフセットで始まって終了時間で終了します。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後繰り返し再生される際には、開始時間で始まり、終了時間で終了します。モーションが終わりまで再生されるようにするには、終了時間を -1 に設定します。

playRate オプション。モーションの実際の再生速度を指定します。再生レートは、モデルの #keyframePlayer モディファイアまたは #bonesPlayer モディファイアの playRate プロパティと掛け算されます。その結果がモーションの実際の再生速度になります。

offset オプション。モーションの始めからミリ秒単位で計測されます。ループが FALSE の場合、モーションは、オフセットで始まって終了時間で終了します。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後繰り返し再生される際には、開始時間で始まり、終了時間で終了します。オフセットパラメータに #synchronized を指定すると、現在再生中のアニメーションの経過時間に合わせてずらした時間からモーションを開始できます。

例文

次のコマンドを実行すると、" 歩行者 " というモデルが " 落ちる " というモーションの再生を開始します。モーションの再生が完了すると、その前に再生していたモーションがあればそのモーションが再開されます。

```
sprite(1).member.model("Walker").bonesPlayer.play("Fall", 0, 0, -1, 1, 0)
```

次のコマンドを実行すると、"Walker" というモデルが "Kick" というモーションの再生を開始します。"Walker" によって現在実行されているモーションがある場合、そのモーションは "Kick" により中断され、"Kick" の一部がループによって繰り返し再生されます。ループの最初の回は、モーションの始めから 2000 ミリ秒のところから開始されます。その後ループによって繰り返し再生される際には、「Kick」の始めから 1000 ミリ秒のところから開始され、「Kick」の 5000 ミリ秒のところで終了します。再生レートは、モデルの bonesPlayer モディファイアの playRate プロパティに設定されている値の 3 倍になります。

```
sprite(1).member.model("Walker").bonesPlayer.play("Kick", 1, 1000, 5000, 3, 2000)
```

関連事項

queue() (3D), playNext() (3D), playRate (3D), playlist, pause() (3D), removeLast(), playing (3D)

play() (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.play()
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter)
dvdObjRef.play(beginTimeList, endTimeList)

// JavaScript syntax
dvdObjRef.play();
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter);
dvdObjRef.play(beginTimeList, beginTimeList);
```

解説

再生を開始または再開する DVD メソッドです。

パラメータを指定しない場合、このメソッドは、一時停止されていれば再生を再開し、停止されていればディスクの先頭または startTimeList プロパティによって指定された値から再生を開始します。stopTimeList プロパティが設定されている場合は、このプロパティによって指定される値まで再生が続きます。

beginTitle、**beginChapter**、**endTitle**、**endChapter** の各パラメータを指定した場合、このメソッドは指定されたタイトルまたはチャプタから再生を開始します。**endTitle** パラメータや **endChapter** パラメータが設定されている場合は、それらによって指定される値まで再生が続きます。

beginTimeList パラメータと **endTimeList** パラメータを指定した場合、このメソッドは beginTimeList パラメータによって指定された値から、endTimeList パラメータによって指定された値まで再生を行います。

beginTimeList および endTimeList に使用されるリスト形式：

```
[#title:1, #chapter:1, #hours:0, #minutes:1, #seconds:1]
```

または

```
[#title:1, #hours:0, #minutes:1, #seconds:1]
```

このメソッドは、成功すると 0 を返します。

パラメータ

beginTitle 特定のタイトルおよびチャプタから再生を開始する場合は必須。再生するチャプタを含むタイトルを指定する数値です。このパラメータは、メンバの startTimeList プロパティよりも優先されます。

beginChapter 特定のタイトルおよびチャプタから再生を開始する場合は必須。再生するチャプタを指定する数値です。このパラメータは、メンバの startTimeList プロパティよりも優先されます。

endTitle 特定のタイトルおよびチャプタで再生を停止する場合は必須。再生を停止するタイトルを指定する数値です。このパラメータは、メンバの stopTimeList プロパティよりも優先されます。

endChapter 特定のタイトルおよびチャプタで再生を停止する場合は必須。再生するチャプタを指定する数値です。このパラメータは、メンバの stopTimeList プロパティよりも優先されます。

beginTimeList 特定の開始時間から再生を開始する場合は必須。再生を開始する時間を指定するプロパティリストです。このパラメータは、メンバの startTimeList プロパティよりも優先されます。

endTimeList 特定の開始時間から再生を開始する場合は必須。再生を停止する時間を指定するプロパティリストです。このパラメータは、メンバの stopTimeList プロパティよりも優先されます。

例文

次のステートメントは、一時停止されたスプライトの再生を再開します。

```
-- Lingo syntax
member(12).play()

// JavaScript syntax
member(12).play();
```

次のステートメントは、タイトル 1 のチャプタ 2 から再生を開始し、チャプタ 4 で再生を終了します。

```
member(15).play([#title:1, #chapter:2], [#title:1, #chapter:4])
```

または

```
member(15).play(1,2,1,4)
```

次のステートメントは、チャプタ 2 内の 10 秒の位置から再生を開始し、17 秒の位置で再生を終了します。

```
member(15).play([#title:2, #seconds:10], [#title:2, #seconds:17])
```

関連事項
DVD, startTimeList, stopTimeList

play() （ミクサ）

使用法
```
Mixer.play([soundobjectlist])
```

解説
ミクサメソッド。サウンドオブジェクトリストが渡されていない場合にミクサ内のすべてのサウンドオブジェクトの再生を開始したり、リスト内のサウンドオブジェクトのリストを再生します。

ミクサの操作	効果
再生 ()	play() メソッドが呼び出される前に、作成されたサウンドオブジェクトを含むミクサを再生します。
再生 ([])	ミクサのみを再生します。サウンドオブジェクトは再生されません。
Play([so1, so2])	参照 so1 および so2 を持つ指定されたサウンドオブジェクトを含むミクサを再生します。
Play(["so1", "so2"])	名前 so1 および so2 を持つ指定されたサウンドオブジェクトを含むミクサを再生します。

例文
```
-- Lingo syntax
on mouseUp me
     mixer1.play() --Starts playing mixer1.
end

//JavaScript syntax
function mouseup()
{
mixer1.play(); //Starts playing mixer1.

}
```

関連事項
pause() （ミクサ）, ミクサ

play() (MP4Media/FLV)

使用法
```
sprite(1).play()
member(1).play()
```

解説
MP4Media/FLV メンバまたはスプライトメソッド。ビデオが一時停止または停止されている場合、ビデオの再生を開始します。mediaStatus の値は #playing に変わります。

例文

次の例は、MP4Media スプライトであるスプライト 2 の再生を開始します。

```
-- Lingo syntax
sprite(2).play()
member("MP4Media").play()

// JavaScript syntax
sprite(2).play();
member("MP4Media").play();
```

関連事項

pause (MP4Media/FLV)

play() (RealMedia, SWA, Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.play()
realMediaObjRef.play()

// JavaScript syntax
windowsMediaObjRef.play();
realMediaObjRef.play();
```

解説

Windows Media または RealMedia のキャストメンバを再生するか、またはステージ上のスプライトを再生する、Windows Media または RealMedia のキャストメンバメソッドまたはスプライトメソッドです。

キャストメンバの場合、音声のみがレンダリングされます（ムービー内に存在する場合）。キャストメンバが既に再生中の場合、このメソッドを呼び出しても効果はありません。

パラメータ

なし

例文

次の例は、スプライト 2 およびキャストメンバ Real のストリームのストリーミングプロセスを開始します。

```
-- Lingo syntax
sprite(2).play()
member("Real").play()

// JavaScript syntax
sprite(2).play();
member("Real").play();
```

関連事項

RealMedia, Windows Media

play() （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.play()
soundChannelObjRef.play(memberObjRef)
soundChannelObjRef.play(propList)

// JavaScript syntax
soundChannelObjRef.play();
soundChannelObjRef.play(memberObjRef);
soundChannelObjRef.play(propList);
```

解説

サウンドチャネル内でキューイングされたサウンドの再生、または特定のキャストメンバのキューイングと再生を行うサウンドチャネルメソッドです。

サウンドキャストメンバの再生を開始する前に RAM にロードするには時間がかかります。サウンドは再生前に queue() を使用してあらかじめキューに入れておき、このメソッドの第 1 のシンタックスを使用して再生することをお勧めします。第 2 および第 3 シンタックスの場合、queue() コマンドによるプリロードは利用できません。

オプショナルプロパティリストを使用すると、正確なサウンド再生設定ができます。

完成したムービーでの play() の実例は、Director アプリケーションフォルダの Learning\Lingo_Examples フォルダ内の sound control.dir を参照してください。

パラメータ

memberObjRef 特定のキャストメンバを再生する場合は必須。キューイングと再生を行うキャストメンバオブジェクトへの参照です。

propList サウンドの再生設定を指定する場合は必須。サウンドの再生設定を指定するプロパティリストです。次のオプション設定ができます。

プロパティ	解説
#member	キューに登録するサウンドキャストメンバ。このプロパティは指定する必要があります。これ以外はすべてオプションです。
#startTime	再生を開始するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの先頭です。「startTime」を参照してください。
#endTime	再生を終了するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの最後です。「endTime」を参照してください。
#loopCount	#loopStartTime と #loopEndTime で定義されたループを再生する回数。デフォルトは 1 です。「loopCount」を参照してください。
#loopStartTime	ループを開始するサウンド内の時間 (単位はミリ秒)。「loopStartTime」を参照してください。
#loopEndTime	ループを終了するサウンド内の時間 (単位はミリ秒)。「loopEndTime」を参照してください。
#preloadTime	再生前にバッファに溜めておくサウンドの量 (単位はミリ秒)。「preloadTime」を参照してください。

例文

次のステートメントは、サウンドチャネル 1 でキャストメンバ "introMusic" を再生します。

```
-- Lingo syntax
sound(1).play(member("introMusic"))

// JavaScript syntax
sound(1).play(member("introMusic"));
```

次のステートメントは、サウンドチャネル 2 でキャストメンバ "creditsMusic" を再生します。サウンド内の 4 秒の位置から再生が始まり、15 秒の位置で終了します。10.5 秒から 14 秒の間で 6 回ループ再生します。

```
-- Lingo syntax
sound(2).play([#member:member("creditsMusic"), #startTime:4000, #endTime:15000, #loopCount:6,
#loopStartTime:10500, #loopEndTime:14000])

// JavaScript syntax
sound(2).play(propList("member",member("creditsMusic"), "startTime",4000,"endTime",15000, "loopCount",6,
"loopStartTime",10500, "loopEndTime",14000));
```

関連事項

endTime （サウンドチャネル）, loopCount, loopEndTime （サウンドチャネル）, loopStartTime, pause() （サウンドチャネル）, preLoadTime, queue(), サウンドチャネル , startTime (サウンドチャネル), stop() （サウンドチャネル）

play （サウンドオブジェクト）

使用法

```
soundObject.play()
```

解説

サウンドオブジェクトメソッド。サウンドオブジェクトと関連付けられたオーディファイルの再生を開始します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.play() --Starts playing the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.play(); //Starts playing the sound object associated with soundObjRef.
}
```

関連事項

pause （サウンドオブジェクト）

playFile()

使用法

```
-- Lingo syntax
soundChannelObjRef.playFile(stringFilePath)

// JavaScript syntax
soundChannelObjRef.playFile(stringFilePath);
```

解説

サウンドチャネル内の AIFF、SWA、AU または WAV サウンドを再生するサウンドチャネルメソッドです。

サウンドを正しく再生するには、ムービーで正しい Mix Xtra が使用される必要があります。通常、この Xtra は、アプリケーションの Xtras フォルダにあります。

サウンドファイルがムービーとは異なるフォルダにある場合は、サウンドファイルへの完全なパスを **stringFilePath** で指定する必要があります。

URL から取り込んだサウンドを再生するには、まず、downloadNetThing() または preloadNetThing() を使用してファイルをローカルディスクにダウンロードするようにします。これによってムービーをダウンロードする間に起こる可能性のある問題を最小限に抑えます。

playFile() メソッドは RAM からではなく、サウンドをディスクから直接再生します。このため、デジタルビデオを再生するとき、またはキャストメンバをメモリにロードするときに playFile() を使用すると、コンピュータがディスク内の異なる場所を同時に読もうとして、競合が発生することがあります。

パラメータ

stringFilePath 必須。再生するファイルの名前を指定するストリングです。サウンドファイルが現在再生しているムービーとは異なるフォルダにある場合は、サウンドファイルへの完全なパスを **stringFilePath** で指定する必要があります。

例文

次のステートメントは " 雷 .wav" というフルパスで指定したファイルをチャネル 1 で再生します。

```
-- Lingo syntax
sound(1).playFile("Thunder.wav")

// JavaScript syntax
sound(1).playFile("Thunder.wav");
```

次のステートメントは " 雷 .wav" というフルパスで指定したファイルをチャネル 3 で再生します。

```
-- Lingo syntax
sound(3).playFile(_movie.path & "Thunder.wav")

// JavaScript syntax
sound(3).playFile(_movie.path + "Thunder.wav");
```

関連事項

play() （サウンドチャネル）, サウンドチャネル , stop() （サウンドチャネル）

playNext() （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.playNext()

// JavaScript syntax
soundChannelObjRef.playNext();
```

解説

チャネルサウンドで再生されている現在のサウンドの再生を即座に中断し、次にキューされているサウンドの再生を開始するサウンドチャネルメソッドです。

指定チャネルにキューされたサウンドがない場合は、サウンドは再生を停止します。

パラメータ

なし

例文

次のステートメントは、キューで待機している次のサウンドをサウンドチャネル 2 で再生します。

```
-- Lingo syntax
sound(2).playNext()

// JavaScript syntax
sound(2).playNext();
```

関連事項

pause() （サウンドチャネル）, play() （サウンドチャネル）, サウンドチャネル ,stop() （サウンドチャネル）

playNext() (3D)

使用法

```
member(whichMember).model(whichModel).bonesPlayer.playNext()
member(whichMember).model(whichModel).keyframePlayer.playNext()
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D コマンド。モデルの #keyframePlayer モディファイアまたは #bonesPlayer モディファイアのプレイリストにある次のモーションの再生を開始します。現在再生中のモーション (プレイリストの最初のエントリ) は中断され、プレイリストから削除されます。

モーションのブレンドがオンになっていて、プレイリストに複数のモーションがある場合は、playNext() を呼び出すと、現在のモーションとプレイリストの次のモーションとのブレンドが開始されます。

例文

次のステートメントは、モデル 1 によって現在実行されているモーションを中断し、プレイリストの次のモーションの再生を開始します。

```
-- Lingo
member("scene").model[1].bonesPlayer.playnext()

// Javascript
member("scene").getProp("model",1).bonesPlayer.playNext();
```

関連事項

blend (3D), playlist

playerParentalLevel()

使用法

```
-- Lingo syntax
dvdObjRef.playerParentalLevel()

// JavaScript syntax
dvdObjRef.playerParentalLevel();
```

解説
プレーヤのペアレンタルレベルを返す DVD メソッドです。

有効なペアレンタルレベルの範囲は 1 ～ 8 です。

パラメータ
なし

関連事項
DVD

point()

使用法

```
-- Lingo syntax
point(intH, intV)

// JavaScript syntax
point(intH, intV);
```

解説
トップレベル関数およびデータ型。指定された水平座標と垂直座標を持つポイントを返します。

ポイントには locH プロパティと locV プロパティがあります。

Lingo を使用する場合に限り、ポイントの座標値は演算によって変更することができます。例えば、Lingo を使用して次の 2 つのポイントを加算することは可能ですが、JavaScript シンタックスを使用すると NaN が返されます。

```
-- Lingo
pointA = point(10,10)
pointB = point(5,5)
put(pointA + pointB)
-- point(15,15)

// JavaScript syntax
var pointA = point(10,10);
var pointB = point(5,5);
trace(pointA + pointB);
// NaN
```

完成したムービーの point() の実例は、Director アプリケーションフォルダの Learning\Lingo_Examples フォルダ内の imaging.dir と vector shapes.dir を参照してください。

パラメータ
intH 必須。ポイントの水平座標を指定する整数です。

intV 必須。ポイントの垂直座標を指定する整数です。

例文
次のステートメントは、変数 lastLocation にポイント (250, 400) を設定します。

```
-- Lingo syntax
lastLocation = point(250, 400)

// JavaScript syntax
var lastLocation = point(250, 400);
```

次のステートメントは、変数 myPoint に割り当てられたポイントの水平座標に 5 ピクセルを加えます。

```
-- Lingo syntax
myPoint.locH = myPoint.locH + 5

// JavaScript syntax
myPoint.locH = myPoint.locH + 5;
```

次の 2 つのステートメントは、mouseH および mouseV に 10 ピクセルずつ加えて、スプライトのステージ座標として設定します（Lingo のみ）。どちらのステートメントでも同じ処理を実行します。

```
-- Lingo syntax
sprite(_mouse.clickOn).loc = point(_mouse.mouseH, _mouse.mouseV) + point(10, 10)
sprite(_mouse.clickOn).loc = _mouse.mouseLoc + 10
```

関連事項

locH, locV

pointAt

使用法

```
member(whichCastmember).model(whichModel).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).camera(whichCamera).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).light(whichLight).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).group(whichGroup).pointAt(vectorPosition{, vectorUp})
```

解説

3D コマンド。参照先オブジェクトの前方向ベクトルが指定したワールド基準の位置をポイントし、上方向ベクトルが指定したワールド基準のベクトルの向きになるように、このオブジェクトを回転させます。

オブジェクトの前方向ベクトルと上方向ベクトルは、オブジェクトの pointAtOrientation プロパティによって定義されます。

パラメータ

vectorPosition 必須。ワールド基準位置を指定します。この値にはノード参照も指定できます。

vectorUp オプション。オブジェクトの上方向ベクトルのポイント先を示すワールド基準のベクトルを指定します。このパラメータを指定しない場合、pointAt ではデフォルトでワールドの y 軸が上方向ベクトルのヒンティングベクトル（ポイント先を示すベクトル）として使用されます。オブジェクトの前方向ベクトルがワールドの y 軸と平行になるような位置のオブジェクトを指定した場合は、ワールドの x 軸が上方向ベクトルのヒンティングベクトルとして使用されます。オブジェクトの前方向に指定する方向とベクトル上方向で指定する方向は、互いに垂直である必要はありません。このコマンドがヒンティングベクトルとして使用するのは、ベクトル上方向パラメータだけです。

例文

次の例では、"MarsCam" というカメラ、"BrightSpot" というライト、および "BigGun" というモデルの 3 つのオブジェクトを "Mars" というモデルに向けています。

```
thisWorldPosn = member("Scene").model("Mars").worldPosition
member("Scene").camera("MarsCam").pointAt(thisWorldPosn)
member("Scene").light("BrightSpot").pointAt(thisWorldPosn)
member("Scene").model("BigGun").pointAt(thisWorldPosn, vector(0,0,45))
```

同じノード (たとえば、モデルなど) で不均等なスケーリングと独自の pointAtOrientation を使用している場合、pointAt は予期しない不均等なスケーリングを発生させる可能性が大きいので注意が必要です。これは、2 つの処理 (不均等なスケーリングおよび正しい向きへのノードの回転) を行う順序が原因で発生します。この問題を回避するには、次のいずれかを実行します。

- 同じノードで、不均等なスケーリングと、デフォルト以外の pointAtOrientation を同時に使用しないようにします。
- pointAt を使用する前にスケールを削除し、後からスケールを再適用します。

  例えば次のようになります。

  ```
  scale = node.transform.scale
  node.scale = vector( 1, 1, 1 )
  node.pointAt(vector(0, 0, 0)) -- non-default pointAtOrientation
  node.transform.scale = scale
  ```

関連事項

pointAtOrientation

pointInHyperlink()

使用法

```
-- Lingo syntax
spriteObjRef.pointInHyperlink(point)

// JavaScript syntax
spriteObjRef.pointInHyperlink(point);
```

解説

指定したポイントがテキストスプライトのハイパーリンクにあるかどうかを TRUE または FALSE で返すテキストスプライト関数です。通常、カーソル位置をポイントとして使います。カスタムカーソルの設定に使うことができます。

パラメータ

point 必須。テストするポイントを指定します。

例文

次のステートメントは、マウスが sprite(1) のハイパーリンク上にあるかどうかをチェックします。

```
-- Lingo
Put Sprite(1).pointInHyperLink(_mouse.mouseLoc)

// Javascript
trace(sprite(1).pointInHyperLink(_mouse.mouseLoc));
```

関連事項

cursor(), mouseLoc

pointToChar()

使用法

```
-- Lingo syntax
spriteObjRef.pointToChar(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToChar(pointToTranslate);
```

解説

指定した画面座標にあるテキストまたはフィールドスプライト内の文字位置を表す整数を返す関数です。当該のポイントがテキスト内に含まれない場合は -1 を返します。

この関数で、カーソル位置にどの文字があるかを調べることができます。

パラメータ

pointToTranslate 必須。テストする画面座標を指定します。

例文

次のステートメントは、クリックされた位置にある文字と、その文字がテキストの何文字目かを示す整数をメッセージウィンドウに表示します。

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    charNum = sprite(spriteNum).pointToChar(pointClicked)
    actualChar = currentMember.char[charNum]
    put("Clicked character" && charNum & ", the letter" && actualChar)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var charNum = sprite(this.spriteNum).pointToChar(pointClicked);
    var actualChar = currentMember.getProp("char", charNum);
    put("Clicked character " + charNum +", the letter " + actualChar);
}
```

関連事項

mouseLoc, pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToItem()

使用法

```
-- Lingo syntax
spriteObjRef.pointToItem(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToItem(pointToTranslate);
```

解説

指定した画面座標にあるテキストまたはフィールドスプライト内の項目位置を表す整数を返す関数です。当該のポイントがテキスト内に含まれない場合は -1 を返します。項目は、itemDelimiter プロパティで設定された区切り文字 (デフォルトはカンマ) で区切られます。

この関数で、カーソル位置にどの項目があるかを調べることができます。

パラメータ

pointToTranslate 必須。テストする画面座標を指定します。

例文

次のステートメントは、クリックされた項目の番号と、その項目のテキストをメッセージウィンドウに表示します。

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    itemNum = sprite(spriteNum).pointToItem(pointClicked)
    itemText = currentMember.item[itemNum]
    put("Clicked item" && itemNum & ", the text" && itemText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var itemNum = sprite(this.spriteNum).pointToItem(pointClicked);
    var itemText = currentMember.getProp("item",itemNum);
    trace( "Clicked item " + itemNum + ", the text " + itemText);
}
```

関連事項

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToLine(), pointToParagraph()

pointToLine()

使用法

```
-- Lingo syntax
spriteObjRef.pointToLine(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToLine(pointToTranslate);
```

解説

指定した画面座標にあるテキストまたはフィールドスプライト内の行位置を表す整数を返す関数です。当該のポイントがテキスト内に含まれない場合は -1 を返します。行はテキストキャストメンバまたはフィールドキャストメンバ内を改行文字で区切ります。

この関数で、カーソルがテキストの何行目にあるかを調べることができます。

パラメータ

pointToTranslate 必須。テストする画面座標を指定します。

例文

次のステートメントは、クリックされた行の番号と、その行のテキストをメッセージウィンドウに表示します。

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    lineNum = sprite(spriteNum).pointToLine(pointClicked)
    lineText = currentMember.line[lineNum]
    put("Clicked line" && lineNum & ", the text" && lineText)
end

// JavaScript syntax
functionmouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var lineNum = sprite(this.spriteNum).pointToLine(pointClicked);
    var lineText = currentMember.getProp("line", lineNum);
    put("Clicked line " + lineNum + ", the text " + lineText);
}
```

関連事項

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToParagraph()

pointToParagraph()

使用法

```
-- Lingo syntax
spriteObjRef.pointToParagraph(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToParagraph(pointToTranslate);
```

解説

指定した画面座標にあるテキストまたはフィールドスプライト内の段落番号を表す整数を返す関数です。当該のポイントがテキスト内に含まれない場合は -1 を返します。段落はテキストのブロック内を改行文字で区切ります。

この関数は、カーソルがテキストの何番目の段落にあるかを調べるために使用できます。

パラメータ

pointToTranslate 必須。テストする画面座標を指定します。

例文

次のステートメントは、クリックされた段落の番号と、その段落のテキストをメッセージウィンドウに表示します。

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    paragraphNum = sprite(spriteNum).pointToParagraph(pointClicked)
    paragraphText = currentMember.paragraph[paragraphNum]
    put("Clicked paragraph" && paragraphNum & ", the text" && paragraphText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var paragraphNum = sprite(this.spriteNum).pointToParagraph(pointClicked);
    var paragraphText = currentMember.getProp("paragraph", paragraphNum);
    trace("Clicked paragraph " + paragraphNum + ", the text " + paragraphText);
}
```

関連事項

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine()

pointToWord()

使用法

```
-- Lingo syntax
spriteObjRef.pointToWord(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToWord(pointToTranslate);
```

解説

指定した画面座標にあるテキストまたはフィールドスプライト内の単語の番号を表す整数を返す関数です。当該のポイントがテキスト内に含まれない場合は -1 を返します。単語はテキストのブロック内を半角スペースで区切ります。

この関数は、カーソルがテキストの何番目の単語にあるかを調べるために使用できます。

パラメータ

pointToTranslate 必須。テストする画面座標を指定します。

例文

次のステートメントは、クリックされた単語の番号と、その単語のテキストをメッセージウィンドウに表示します。

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    wordNum = sprite(spriteNum).pointToWord(pointClicked)
    wordText = currentMember.word[wordNum]
    put("Clicked word" && wordNum & ", the text" && wordText)
end

// JavaScript syntax
function mouseDown(me) {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var wordNum = sprite(this.spriteNum).pointToWord(pointClicked);
    var wordText = currentMember.getProp("word", wordNum);
    trace("Clicked word " + wordNum + ", the text " + wordText);
}
```

関連事項

itemDelimiter, mouseLoc, pointToChar(), pointToItem(), pointToLine(), pointToParagraph()

postNetByteArray

使用法

```
postNetByteArray(URL, byteArray, [propertyList])
postNetByteArray(URL, [propertyList which contains bytearray]))
```

解説

Net Lingo メソッド。バイト配列をポストします。このメソッドは postNetText と似ています。コンテンツタイプを "Content-Type: multipart/form-data" として使用します。このメソッドはバイト配列を返します。このメソッドの結果を取得するには、netByteArrayResult メソッドを使用します。

パラメータ

パラメータ	解説	必須 / オプション
URL	取得する byteArray が存在するファイルの URL。	必須
byteArray	ポストするバイト配列。	必須 . 注：propertyList を使用する場合はオプションです。
propertyList	CGI クエリに使用するプロパティリストを指定します。ここでバイト配列のリストを指定することもできます。	オプション

例文

```
--Lingo syntax
id = postNetByteArray("http://www.bytearray.com/byte.asp",bArray)

//JavaScript syntax
id = postNetByteArray("http://www.bytearray.com/byte.asp",[#name:"Director",#bArray:bArray])
```

postNetText

使用法

```
postNetText(url, propertyList {,serverOSString} {,serverCharSetString})
postNetText(url, postText {,serverOSString} {,serverCharSetString})
```

解説

URL で示される HTTP URL に、指定したデータを付けて POST リクエストを送信するコマンドです。

このコマンドは getNetText() と同様の機能です。getNetText() と同じように、サーバの応答は netTextResult(ネットワーク **ID**) 関数を介して返されます。この関数は、netDone(ネットワーク **ID**) が 1 を返し、netError(ネットワーク **ID**) が 0（OK を示す）を返した場合にのみ呼び出すことができます。

オプションのパラメータは、位置にかかわらず省略できます。

getNetText() に比べて、postNetText() クエリには長さが任意であるという利点があります。getNetText() クエリの場合、URL の長さ（ブラウザにより、1K または 4K）が制限されます。

注意：postNetText を使用して、ムービーを再生しているドメイン以外にデータを送信しようとすると、Shockwave Player の再生中にセキュリティに関する警告が表示されます。

完成したムービーで使用する postNetText の例については、Director アプリケーションフォルダの Learning\Lingo_Examples フォルダにある、Forms and Post.dir ムービーを参照してください。

パラメータ

url 必須。POST リクエストを送信する URL を指定します。

propertyList または **postText** 必須。リクエストと共に送信するデータを指定します。ストリングの代わりにプロパティリストを使うと、METHOD=POST を指定して HTML 形式でブラウザが処理するのと同じ形式で、情報を送信できます。この機能により、Director 内でこの形式のデータを構築し、送信できます。プロパティ名は HTML 形式のフィールド名に対応し、プロパティ値はフィールド値に対応します。

プロパティリストのプロパティ名は、ストリングまたはシンボルです。シンボルの場合、先頭の "#" を除いたストリングに自動的に変換されます。プロパティの値として数値が使用された場合も同様に、ストリングに変換されます。

注意：プロパティリストではなくストリングを使った場合、postText ストリングは、MIME タイプの「text/plain」を使用して HTTP POST リクエストとしてサーバに送信されます。この方法が便利な場合もありますが、HTML 形式の送信とは互換性がありません。例えば、PHP スクリプトでは常にプロパティリストを使用する必要があります。

serverOSString オプション。デフォルトでは UNIX ですが、Windows や Mac を設定することで、**postText** パラメータに含まれるすべての改行を、サーバで使用する形式の改行文字に変換できます。ほとんどのアプリケーションでは、この設定は必要ありません。フォームでの応答では通常、改行を使用しないためです。

serverCharSetString オプション。Shift-JIS（日本語）システムにのみ適用されます。選択できる設定としては、"JIS"、"EUC"、"ASCII"、"AUTO" があります。送信データを Shift-JIS から、名前を付けた文字セットへ変換します。返されたデータは、getNetText() の場合 (指定のキャラクタセットからシフト JIS へと変換) とまったく同様に処理されます。"AUTO" を指定した場合、ローカルのキャラクタセットでポストされたデータは変換されません。サーバから返されてきた結果は、getNetText() の場合と同じように変換されます。[サーバ文字セットストリング] の指定を省略した場合、"ASCII" が指定されたとみなされます。"ASCII" の場合、ポストや結果の変換は何も行われません。

例文

次のステートメントは、[サーバ文字セットストリング] の指定を省略しています。

```
-- Lingo
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win")

// Javascript
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win");
```

次の例は、ユーザ入力フィールドの " 姓 "、" 名前 "、" 得点 " からフォームを生成します。serverOSString と serverCharSetString は省略しています。

```
-- LingonetID = postNetText("www.mydomain.com/userbase.cgi", infoList);
lastName = member("Last Name").text
firstName = member("First Name").text
totalScore = member("Current Score").text
infoList = ["FName":firstName, "LName":lastName, "Score":totalScore]
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);

// Javascript
lastName = member("Last Name").text;
firstName = member("First Name").text;
totalScore = member("Current Score").text;
infoList = propList("FName",firstName, "LName",lastName, "Score",totalScore);
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);
```

関連事項

getNetText(), netTextResult(), netDone(), netError()

power()

使用法

```
power(base, exponent)
```

解説

指定した数の、指定した累乗を計算する算術関数です。

パラメータ

base 必須。基数を指定します。

exponent 必須。指数値を指定します。

例文

次のステートメントは、変数 vResult に 4 の 3 乗を設定します。

```
-- Lingo
set vResult = power(4,3)

// Javascript
Var vResult = Math.pow(4,3);
```

preLoad() （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.preLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.preLoad({toMemberObjRef});
```

解説

指定したフレームまたはフレーム範囲内のキャストメンバをメモリにプリロード（先読み）するメンバメソッドです。プリロードは、メモリがいっぱいになるか、または指定したすべてのキャストメンバが読み込まれた時点で終了します。

toMemberObjRef パラメータを指定しないと、preLoad() は現行フレームとムービーの最終フレームの間で使われているすべてのキャストメンバをプリロードします。

パラメータ

toMemberObjRef オプション。メモリにロードされるキャストメンバの範囲内の最後のキャストメンバへの参照です。範囲内の最初のキャストメンバは **memberObjRef** によって指定されます。

例文

次のステートメントは、QuickTime ムービー " 回転椅子 " がメモリにプリロードできるかどうかを、メッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Rotating Chair").preload())

// JavaScript syntax
put(member("Rotating Chair").preload());
```

次の startMovie ハンドラは、Flash ムービーキャストメンバをストリーミングするように設定し、その bufferSize プロパティを設定します。

```
-- Lingo syntax
on startMovie
    member("Flash Demo").preload = FALSE
    member("Flash Demo").bufferSize = 65536
end

// JavaScript syntax
function startMovie() {
    member("Flash Demo").preload = false;
    member("Flash Demo").bufferSize = 65536;
}
```

関連事項

メンバ

preLoad() (Movie)

使用法

```
-- Lingo syntax
_movie.preLoad({frameNameOrNum})
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum)

// JavaScript syntax
_movie.preLoad({frameNameorNum});
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum);
```

解説

ムービーメソッド。指定したフレームまたはフレーム範囲内のキャストメンバをメモリにプリロードします。プリロードは、メモリがいっぱいになるか、または指定したすべてのキャストメンバが読み込まれた時点で終了します。

- パラメータを指定しないと、このメソッドは現行フレームとムービーの最終フレームの間で使われているすべてのキャストメンバをプリロードします。
- **frameNameOrNum** パラメータのみを指定すると、このメソッドは現行フレームと指定したフレームの間で使われているすべてのキャストメンバをプリロードします。**frameNameOrNum** はフレーム番号またはラベル名で指定します。
- **fromFrameNameOrNum** および **toFrameNameOrNum** の 2 つのパラメータを指定すると、その範囲内のフレームで使われているすべてのキャストメンバがプリロードされます。**fromFrameNameOrNum** および **toFrameNameOrNum** はフレーム番号またはラベル名で指定します。

preLoad() メソッドでは、最後に正常に読み込んだフレームの番号を返すこともできます。この値を取得するには、result() メソッドを使用します。

パラメータ

frameNameOrNum オプション。あるフレームをプリロードすることを指定するストリング、またはあるフレームの番号がプリロードすることを指定する整数です。

fromFrameNameOrNum は、一定範囲のフレームをプリロードする場合に必要です。プリロードする一定範囲のフレーム内にある最初のフレームのラベル名を指定するストリング、またはプリロードする一定範囲のフレーム内にある最初のフレームの番号を指定する整数です。

toFrameNameOrNum は、一定範囲のフレームをプリロードする場合に必要です。プリロードする一定範囲のフレーム内にある最後のフレームのラベル名を指定するストリング、またはプリロードする一定範囲のフレーム内にある最後のフレームの番号を指定する整数です。

例文

次のステートメントは、現行フレームと次にマーカが設定されているフレームの間で使われているキャストメンバをプリロードします。

```
-- Lingo syntax
_movie.preLoad(_movie.marker(1))

// JavaScript syntax
_movie.preLoad(_movie.marker(1));
```

次のステートメントは、フレーム 10 ～ 50 の間で使われているキャストメンバをプリロードします。

```
-- Lingo syntax
_movie.preLoad(10, 50)

// JavaScript syntax
_movie.preLoad(10, 50);
```

関連事項

ムービー , result

preLoadBuffer()

使用法

```
-- Lingo syntax
memberObjRef.preLoadBuffer()

// JavaScript syntax
memberObjRef.preLoadBuffer();
```

解説

指定した Shockwave Audio (SWA) ファイルの一部をプリロードするコマンドです。プリロードする量は、preLoadTime プロパティで指定します。このコマンドは、SWA キャストメンバが停止しているときのみ動作します。

preLoadBuffer コマンドの実行が成功すると、state メンバプロパティの値は 2 になります。

SWA キャストメンバのほとんどのプロパティは、preLoadBuffer コマンドが成功した後にのみ調べることができます。このようなプロパティには、cuePointNames、cuePointTimes、currentTime、duration、percentPlayed、percentStreamed、bitRate、sampleRate、numChannels などがあります。

パラメータ

なし

例文

次のステートメントは、" メル・トーメ " SWA キャストメンバをメモリにロードします。

```
-- Lingo syntax
member("Mel Torme").preLoadBuffer()

// JavaScript syntax
member("Mel Torme").preLoadBuffer();
```

関連事項

preLoadTime

preLoadMember()

使用法

```
-- Lingo syntax
_movie.preLoadMember({memberObjRef})
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum)

// JavaScript syntax
_movie.preLoadMember({memberObjRef});
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum);
```

解説

キャストメンバをプリロードするムービーメソッドです。プリロードは、メモリがいっぱいになるか、指定したすべてのキャストメンバが読み込まれた時点で終了します。

このメソッドでは、読み込みが成功した最後のキャストメンバの番号を返します。この値を取得するには、result() メソッドを使用します。

パラメータを指定せずに preLoadMember() を使用すると、ムービー内のすべてのキャストメンバがプリロードされます。

memberObjRef を指定すると、preLoadMember() によってそのキャストメンバがプリロードされます。**memberObjRef** に整数を指定すると、最初のキャストライブラリが参照されます。**memberObjRef** に文字列を指定すると、その名前の最初のメンバが使用されます。

fromMemNameOrNum と **toMemNameOrNum** を指定すると、preLoadMember() は、キャストメンバ名か番号で指定した範囲のフレームで使われているすべてのキャストメンバをプリロードします。

パラメータ

memberObjRef オプション。プリロードするキャストメンバの参照です。

fromMenNameOrNum は、一定範囲のキャストメンバをプリロードする場合に必要です。プリロードするキャストメンバの範囲で最初のキャストメンバを指定するストリングまたは整数です。

toMemNameOrNum は、一定範囲のキャストメンバをプリロードする場合に必要です。プリロードするキャストメンバの範囲で最初のキャストメンバを指定するストリングまたは整数です。

例文

次のステートメントは、ムービー内の "SWF" という名前のメンバをプリロードします。

```
-- Lingo
_movie.preLoadMember(member("SWF"))

// Javascript
_movie.preLoadMember(member("SWF")) ;
```

関連事項

ムービー , preLoad() （メンバ）, result

preLoadMovie()

使用法

```
-- Lingo syntax
_movie.preLoadMovie(stringMovieName)

// JavaScript syntax
_movie.preLoadMovie(stringMovieName);
```

解説

ムービーメソッド。指定した [ムービー] の最初のフレームに関連づけられているキャストメンバをプリロードします。ムービーをプリロードしておくと、go() や play() メソッドでの開始がスムーズにできます。

URL からキャストメンバをプリロードするには、preloadNetThing() を指定して直接キャッシュに読み込むか、downloadNetThing() を指定してローカルディスクにダウンロードしてからメモリに読み込みます。後者は、ダウンロードの待ち時間を最小限にできます。

パラメータ

stringMovieName 必須。プリロードするムービーの名前を指定するストリングです。

例文

次のステートメントは、現行ムービーと同じフォルダにある " イントロダクション " ムービーをプリロードします。

```
-- Lingo syntax
_movie.preLoadMovie("Introduction")

// JavaScript syntax
_movie.preLoadMovie("Introduction");
```

関連事項

downloadNetThing, go(), ムービー , preloadNetThing()

preloadNetThing()

使用法

```
preloadNetThing (url)
```

解説

ファイルをインターネットからブラウザのキャッシュにプリロードし、後で使うときのダウンロードの遅滞を防ぐコマンドです。戻り値はネットワーク ID で、これを使用して処理の進行状況を監視することができます。

preloadNetThing() 関数は、現行ムービーの再生中にファイルをダウンロードします。ダウンロードが終了したかどうかは、netDone() を使用して確認します。

項目がプリロードされると、直ちに表示することができます。これは、その項目がネットワークからではなくローカルブラウザのキャッシュから取り込まれているためです。

複数のネットワーク操作を同時にアクティブにすることができますが、同時に行われる操作を 5 つ以上実行すると、通常、コンピュータのパフォーマンスを著しく低下させます。

キャッシュサイズの設定または [ドキュメントの確認] オプションのいずれも、preloadNetThing の機能には影響しません。

preloadNetThing() 関数は、Director ファイル内のリンクを解析しません。ダウンロードした Director ファイルがキャストやグラフィックファイルにリンクしていても、preloadNetThing() がダウンロードするのは Director ファイルのみです。リンクされたオブジェクトは個別にプリロードする必要があります。

パラメータ

url 必須。Director ムービー、グラフィック、FTP サーバの場所など、有効なインターネットファイルへのパスを指定します。

例文

次のステートメントは、preloadNetThing() を使用して、操作用のネットワーク ID を返します。

```
-- Lingo
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir")

// Javascript
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir");
```

ダウンロードが済んでからも、同じ URL でムービーにアクセスできます。キャッシュに蓄えられているので、そこから直接再生されます。

関連事項
netDone()

preMultiply

使用法

```
transform1.preMultiply(transform2)
```

解説

3D トランスフォームコマンド。別のトランスフォームの位置、回転、スケールの各効果をあらかじめ適用することによって、あるトランスフォームを変更します。

transform2 に X 軸を中心とした 90° の回転が設定され、**transform1** に Y 軸での 100 単位の平行移動が設定されている場合は、transform1.multiply(**transform2**) を実行すると、平行移動の後に回転を実行するようにこのトランスフォームが変更されます。ステートメント transform1.preMultiply(トランスフォーム **2**) を実行すると、平行移動の前に回転を実行するようにトランスフォーム 1 が変更されます。その結果、処理の順序が逆になります。

パラメータ

transform2 必須。別のトランスフォームに効果をあらかじめ適用するトランスフォームを指定します。

例文

次のステートメントは、モデル " 火星 " のトランスフォームをモデル " 冥王星 " のトランスフォームに適用します。

```
-- Lingo
member("scene").model("Pluto").transform.preMultiply(member("scene").model("Mars").transform)

// Javascript
member("scene").getPropRef("model" ,
i).transform.preMultiply(member("scene").getPropRef("model",j).transform) ;
// where i and j are the number index of the models "Pluto" and "Mars" respectively.
```

preRotate

使用法

```
transformReference.preRotate( xAngle, yAngle, zAngle )
transformReference.preRotate( vector )
transformReference.preRotate( positionVector, directionVector, angle )
member( whichCastmember ).node.transform.preRotate( xAngle, yAngle, zAngle )
member( whichCastmember ).node.transform.preRotate( vector )
member( whichCastmember ).node.transform.preRotate( positionVector, directionVector, angle )
```

解説

3D トランスフォームコマンド。参照先のトランスフォームオブジェクトが保持している現在の位置、回転、スケールの各オフセットより先に回転を適用します。適用する回転は、3 つの角度として指定できます。この 3 つの角度は、それぞれ 3 つの軸を中心とした回転の角度を指定します。これらの角度を指定するには、xAngle、yAngle、および zAngle で明示的に指定する形式と、ベクトルで指定する形式とがあります。ベクトルで指定する場合は、ベクトルの x コンポーネントが **x** 軸を中心とした回転に、y コンポーネントが **y** 軸を中心とした回転に、z コンポーネントが **z** 軸を中心とした回転に、それぞれ対応します。

また、適用する回転を、任意の軸を中心とした回転として指定することもできます。この場合の軸は、スペース内で、位置ベクトルと方向ベクトルによって定義します。この軸を中心にした回転の角度は、角度によって指定します。

ノードには、モデル、グループ、ライト、またはカメラのいずれかの参照を指定できます。

パラメータ

x の角度は、**x**、**y** および **z** 軸を指定して回転を適用する場合に必要です。**x** 軸を中心にした回転の角度を指定します。

y の角度は、**x**、**y** および **z** 軸を指定して回転を適用する場合に必要です。**y** 軸を中心にした回転の角度を指定します。

z の角度は、**x**、**y** および **z** 軸を使用して回転を適用する場合に必要です。**z** 軸を中心にした回転の角度を指定します。

ベクトルは、ベクトルを指定して回転を適用する場合に必要です。回転で角度を指定するベクトルを指定します。

位置ベクトルは、任意の軸を中心にした回転を適用する場合に必要です。位置オフセットを指定します。

方向ベクトルは、任意の軸を中心にした回転を適用する場合に必要です。方向オフセットを指定します。

角度は、任意の軸を中心にした回転を適用する場合に必要です。任意の軸を中心にした回転量を指定します。

例文

次のステートメントは、各軸を中心に 20° ずつの回転を実行します。モデルの transform プロパティは、モデルの親を基準にしたモデルの位置、回転、およびスケールのオフセットであり、preRotate による方向の変更はモデルのどの transform よりも先に適用されるため、このモデルは親の周りを回らずにその場で回転します。

```
-- Lingo
member("scene").model("bip01").transform.preRotate(20, 20, 20)

// Javascript
member("scene").getPropRef("model", i).transform.preRotate(20, 20, 20) ;
// where i is the number index of the model "bip01"
```

上の例は、次と同じになります。

```
member("scene").model("bip01").rotate(20,20,20).
// javascript
member("scene").getPropRef("model", i).rotate(20,20,20) ;
// where i is the number index of the model "bip01"
```

通常、preRotate() は、トランスフォーム変数を扱う場合以外には使用されません。次の行を実行すると、スペースのポイント (100, 0, 0) の周りを、y 軸を中心にカメラが 180° 回転します。

```
-- Lingo
t = transform()
t.position = member("scene").camera[1].transform.position
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180)
member("scene").camera[1].transform = t

// javascript
var t = transform() ;
t.position = member("scene").getPropRef("camera", 1).transform.position ;
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180) ;
member("scene").getPropRef("camera",1).transform = t ;
```

関連事項

rotate

preScale()

使用法

```
transformReference.preScale( xScale, yScale, zScale )
transformReference.preScale( vector )
member( whichCastmember ).node.transform.preScale( xScale, yScale, zScale )
member( whichCastmember ).node.transform.preScale( vector )
```

解説

3D トランスフォームコマンド。指定したトランスフォームの既存の位置、回転、スケールの各効果より先にスケールを適用します。

ノードには、モデル、グループ、ライト、またはカメラのいずれかの参照を指定できます。

パラメータ

x のスケールは、**x**、**y** および **z** 軸を指定してスケールを適用する場合に必要です。**x** 軸を中心にしたスケールを指定します。

y のスケールは、**x**、**y** および **z** 軸を指定してスケールを適用する場合に必要です。**y** 軸を中心にしたスケールを指定します。

z のスケールは、**x**、**y** および **z** 軸を指定してスケールを適用する場合に必要です。**z** 軸を中心にしたスケールを指定します。

ベクトルは、ベクトルを指定してスケールを適用する場合に必要です。適用するスケールを含むベクトルを指定します。

例文

次の Lingo の **1 行目**は、月 1 のトランスフォームの複製を作成します。モデルの transform プロパティは、参照によってアクセスされます。

2 行目は、トランスフォームの既存の位置や回転の効果より先にスケールを適用します。仮に、このトランスフォームが親の惑星を基準にした "Moon1" の位置のオフセットと回転の軌道を表していて、"Moon2" の親が "Moon1" の親と同じであるとした場合、preScale() の代わりに scale() を使用すると、"Moon2" は、"Moon1" の親から 2 倍の距離のところで、親の惑星の周りを " 月 1" の 2 倍回転することになります。これは、トランスフォームの既存の位置および回転のオフセットにスケールが適用されるためです。一方、preScale() を使用した場合は、これらの既存の位置および回転のオフセットが、適用されるサイズ変更によって影響を受けることはありません。

3 行目は、惑星の x 軸を中心とした 180° の回転を適用します。これにより、Moon2 は、Moon1 の軌道の反対側に移動します。ここで preRotate() を使用すると、"Moon2" は "Moon1" と同じ位置に残り、自分自身の x 軸を中心に 180　回転します。

4 行目は、この新しいトランスフォームを " 月 2" に割り当てます。

```
-- Lingo
t = member("scene").model("Moon1").transform.duplicate()
t.preScale(2,2,2)
t.rotate(180,0,0)
member("scene").model("Moon2").transform = t

// Javascript
var t = member("scene").getPropRef("model", i).transform.duplicate() ;
t.preScale(2,2,2) ;
t.rotate(180,0,0) ;
member("scene").getPropRef("model", i).transform = t ;
// where i the number index of model " Moon2".
```

preTranslate()

使用法

```
transformReference.preTranslate( xIncrement, yIncrement, zIncrement )
transformReference.preTranslate( vector )
member( whichCastmember ).node.transform.preTranslate(xIncrement, yIncrement, zIncrement)
member( whichCastmember ).node.transform.preTranslate( vector )

// Javascript
member( whichCastmember ).getProp("model",a).transform.preTranslate(xIncrement, yIncrement, zIncrement) ;
```

解説

3D トランスフォームコマンド。参照先のトランスフォームオブジェクトが保持している現在の位置、回転、スケールの各オフセットより先に平行移動を適用します。適用する平行移動は、3 つの軸に沿った 3 つの増分として指定できます。これらの増分は、**x** の増分、**y** の増分、および **z** の増分という形式で明示的に指定することも、ベクトルで指定することもできます。ベクトルで指定する場合は、ベクトルの X コンポーネントが X 軸に沿った増分に、Y コンポーネントが Y 軸に沿った増分に、Z コンポーネントが Z 軸に沿った増分に、それぞれ対応します。

一連のトランスフォーメーションが次の順番で行われると、モデルの親はワールドであるという仮定の下に、モデルのローカルの原点が (0, 0, -100) になります。

```
model.transform.identity()
model.transform.rotate(0, 90, 0)
model.transform.preTranslate(100, 0, 0)
```

preTranslate() の代わりに translate() を使用すると、モデルのローカルの原点が (100, 0, 0) になり、モデルは自身の Y 軸を中心に 90° 回転します。ステートメント model.transform.pretranslate(x, y, z) は、model.translate(x, y, z) と同じになります。通常、preTranslate() は、model.transform の参照ではなく、トランスフォーム変数を扱う場合に使用されます。

パラメータ

x 増分は、**x**、**y** および **z** 軸を指定して移動を適用する場合に必要です。**x** 軸を中心にした移動を指定します。

y 増分は、**x**、**y** および **z** 軸を指定して移動を適用する場合に必要です。**y** 軸を中心にした移動を指定します。

z 増分は、**x**、**y** および **z** 軸を指定して移動を適用する場合に必要です。**z** 軸を中心にした移動を指定します。

ベクトルは、ベクトルを指定して移動を適用する場合に必要です。移動で使用するベクトルを指定します。

例文

```
-- Lingo
    t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.preTranslate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(0.0000, 0.0000, -100.0000)

// Javascript

gbModel = member("scene").getProp("model" , a) ;
// where a is the number index for the mars model.
gbModel.transform.preTranslate(xIncrement , yIncrement, zIncrement) ;
member("scene").getProp("model" , a).transform.preTranslate(xIncrement , yIncrement, zIncrement) ;
```

print()

使用法

```
-- Lingo syntax
spriteObjRef.print({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.print({targetName, #printingBounds});
```

解説

コマンド。Flash 5 で ActionScript に導入された print コマンドを呼び出します。Flash ムービーのフレームのうち、#p のラベルが付いたフレームがすべてプリントされます。フレームにラベルが付いていない場合は、ムービー全体がプリントされます。

Flash ムービーのプリント方法は複雑なので、Flash 5 のマニュアルのプリントに関する項を参照してから、このスプライト関数を使用することをお勧めします。

パラメータ

targetName オプション。プリントされるターゲットムービーまたはムービークリップの名前を指定します。省略すると（ターゲットを 0 に指定すると）、メインの Flash ムービーがプリントされます。

printingBounds オプション。プリント境界のオプションを指定します。省略すると、ターゲットムービーの境界が使用されます。指定する場合、**printingBounds** は次のいずれかの値にする必要があります。

- #bframe。指定すると、各ページのプリント境界は、プリント対象のフレームと同じ大きさに変わります。
- #bmax。指定すると、プリント境界は、プリント対象のフレームがすべて収まる程度の大きさの矩形になります。

例文

次のステートメントは、"SWF" キャストメンバとして記述されている Flash ムービーをプリントします。

```
-- Lingo
member("SWF").print()
// javascript
member("SWF").print() ;
```

printAsBitmap()

使用法

```
-- Lingo syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds});
```

解説

Flash スプライトコマンド。機能的には、print コマンドと非常に似ていますが、Flash スプライトのみで機能します。printAsBitmap は、アルファチャネル情報を含んだオブジェクトをプリントできます。

printFrom()

使用法

```
-- Lingo syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux})

// JavaScript syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux});
```

解説

ムービーメソッド。開始フレームで指定したフレームでステージに表示される画像を印刷するコマンドです。**endFrame** を指定すれば、その範囲内のフレームすべてが印刷されます。また、印刷時の縮尺（**redux**）の値（100%、50% または 25%）を追加指定できます。

このコマンドは旧バージョン用に作成されたコマンドです。このコマンドは旧バージョン用に作成されたコマンドです。最新のプリンタドライバでの動作検証は行われておりませんので、広く配布するムービーには、PrintOMatic のようなサードパーティ製 Xtra をご使用ください。Windows 環境下の 256 色を超えるモニタ色深度でムービーが再生されている場合、多くのプリンタでは出力が正しく行われません。この場合は再生時の色深度を下げる必要があります。

パラメータ

startFrameNameOrNum 必須。出力する最初のフレームの名前または番号を指定するストリングまたは整数です。

endFrameNameOrNum オプション。出力する最後のフレームの名前または番号を指定するストリングまたは整数です。

redux オプション。縮尺の値を指定する整数です。有効な値は、100、50 または 25 です。

例文

このステートメントは、フレーム 1 のステージを印刷します。

```
-- Lingo syntax
_movie.printFrom(1)

// JavaScript syntax
_movie.printFrom(1);
```

次のステートメントは、フレーム 10 からフレーム 25 までの各フレーム内のステージの内容を 50% の縮尺で印刷します。

```
-- Lingo syntax
_movie.printFrom(10, 25, 50)

// JavaScript syntax
_movie.printFrom(10, 25, 50);
```

関連事項

ムービー

propList()

使用法

```
-- Lingo syntax
propList()
[:]
propList(string1, value1, string2, value2, ...)
propList(#symbol1, value1, #symbol2, value2, ...)
[#symbol1:value1, #symbol2:value2, ...]

// JavaScript syntax
propList();
propList(string1, value1, string2, value2, ...);
```

解説

トップレベルの関数。リスト内の各エレメントは名前と値のペアで構成されているプロパティリストを作成します。

シンタックス propList() または [:] （Lingo のみ）でプロパティリストを作成する場合、パラメータの有無とは無関係に、リストのインデックスは 1 から始まります。

実行可能なスクリプトの 1 行の長さは 256 文字までに制限されているため、propList() を使用して大きなプロパティリストを作成することはできません。データが大量にあるプロパティリストを作成するには、そのデータを角括弧（[]）で囲んでフィールドに入れ、そのフィールドを変数に割り当てます。変数の内容がデータのリストになります。

パラメータ

ストリング **1**、ストリング **2**、**...** オプション。リスト内のエレメントの名前部分を指定するストリングです。

値 **1**、値 **2**、**...** オプション。リスト内のエレメントの値部分を指定する値です。

#symbol1、**#symbol2**、**...** （Lingo のみ）オプション。リスト内のエレメントの名前部分を示すシンボルです。

例文

次のステートメントは、様々なプロパティおよび値で構成されるプロパティリストを作成し、メッセージウィンドウに様々なプロパティ値を表示します。

```
-- Lingo syntax
-- using propList()
colorList = propList(#top,"red", #sides,"blue", #bottom,"green")
-- using brackets
colorList = [#top:"red", #sides:"blue", #bottom:"green"]
put(colorList.top) -- "red"
put(colorList.sides) -- "blue"
put(colorList.bottom) -- "green"

// JavaScript syntax
var colorList = propList("top","red", "sides","blue", "bottom","green");
put(colorList.top); // red
put(colorList.sides); // blue
put(colorList.bottom); // green
```

関連事項

list()

proxyServer

使用法

```
proxyServer serverType, "ipAddress", portNum
proxyServer()
```

解説

次のように、FTP または HTTP プロキシサーバの値を設定するためのコマンドです。

パラメータがない場合、proxyServer() を使用すると、FTP または HTTP プロキシサーバー設定が返されます。

パラメータ

serverType オプション。プロキシサーバの種類を指定するシンボルです。値は #ftp または #http のいずれかになります。

ipAddress オプション。IP アドレスを指定するストリングです。

portNum オプション。ポート番号を指定する整数です。

例文

次のステートメントは、IP アドレス "197.65.208.157" のポート 5 を HTTP プロキシサーバに設定します。

```
-- Lingo
proxyServer #http,"197.65.208.157",5
// Javascript
proxyServer (symbol("http"),"197.65.208.157",5) ;
```

次のステートメントは、HTTP プロキシサーバーのポート番号を返します。

```
-- Lingo
put proxyServer(#http,#port)

// Javascript
put (proxyServer(symbol("http"),symbol("port"))) ;
```

サーバの種類が指定されていない場合、この関数は 1 を返します。

次のステートメントは、HTTP プロキシサーバの IP アドレスを文字列として返します。

```
-- Lingo
put proxyServer(#http)

// Javascript
put (proxyServer(symbol("http")) ;
```

次のステートメントは、FTP プロキシサーバをオフにします。

```
proxyServer #ftp,#stop
// Javascript
proxyServer(symbol("ftp"),symbol("stop")) ;
```

ptToHotSpotID()

使用法

```
-- Lingo syntax
spriteObjRef.ptToHotSpotID(point)

// JavaScript syntax
spriteObjRef.ptToHotSpotID(point);
```

解説

指定したポイントの位置にホットスポットがあればその ID を返す QuickTime VR 関数です。ホットスポットがなければ 0 を返します。

パラメータ

point 必須。テストするポイントを指定します。

puppetPalette()

使用法

```
-- Lingo syntax
_movie.puppetPalette(palette {, speed} {, frames})

// JavaScript syntax
_movie.puppetPalette(palette {, speed} {, frames});
```

解説

ムービーメソッド。パレットチャネルをパペットとして動作させます。スコアのパレットチャネルでのパレット設定を無効にして、パレットをムービーに割り当てることができます。

puppetPalette() メソッドは、[パレット] で指定したパレットキャストメンバを現行パレットに設定します。[パレット] がストリングの場合はパレットのキャストライブラリ名を指定するものになります。また、[パレット] が整数の場合はパレットのキャスト番号を指定するものになります。

puppetPalette() メソッドを対象フレームにジャンプする前に使用して、Director が実際に次のフレームを描画する前に目的のパレットに切り替えるようにします。

オプションとして [速度] に整数式を指定すると、パレットをフェードインさせることもできます (1 で最低速、60 で最高速)。また、[フレーム数] に整数式を指定すると、指定したフレーム数を使ってパレットをフェードインさせることもできます。

パレットチャネルのパペット状態は、_movie.puppetPalette(0) シンタックスでオフにするまで維持され、それまではスコアで設定されているパレットの変更は無視されます。

注意：ブラウザは Web ページ全体のパレットを制御します。したがって、Shockwave Player は常にブラウザのパレットを使用します。

パラメータ

palette 必須。新しいパレットの名前または番号を指定するストリングまたは整数です。

speed オプション。フェードの速度を指定する整数です。有効な値の範囲は 1 ～ 60 です。

frames オプション。フェードが動作するフレームの数を指定する整数です。

例文

次のステートメントは、「レインボー」をムービーの パレットにします。

```
-- Lingo syntax
_movie.puppetPalette("Rainbow")

// JavaScript syntax
_movie.puppetPalette("Rainbow");
```

次のステートメントは、" レインボー " をムービーのパレットにします。" レインボー " パレットへの移行は、20 のフレーム間で時間設定 15 で行われます。

```
-- Lingo syntax
_movie.puppetPalette("Rainbow", 15, 20)

// JavaScript syntax
_movie.puppetPalette("Rainbow", 15, 20);
```

関連事項

ムービー

puppetSprite()

使用法

```
-- Lingo syntax
_movie.puppetSprite(intSpriteNum, bool)

// JavaScript syntax
_movie.puppetSprite(intSpriteNum, bool);
```

解説

ムービーメソッド。スプライトチャネルが、パペットであるか（TRUE）、パペットでないか（FALSE）を決定します。パペットのときはスクリプトコントロールによって制御され、パペットではないときはスコアによって制御されます。

再生ヘッドが同一のスプライトにあるときに、puppetSprite(**intSpriteNum**, FALSE) というシンタックスでスプライトチャネルのパペット化をオフにすると、スプライトのプロパティはスコアで設定された値にリセットされます。

スプライトチャネルの初期プロパティには、puppetSprite() メソッドを実行した時点でのチャネル設定がそのまま受け継がれます。それ以降はスクリプトでスプライトプロパティを次のように変更できます。

- スプライトチャネルがパペットの場合、そのチャネルにあるスプライトのプロパティに対してスクリプトが加えた変更は、再生ヘッドがそのスプライトを抜け出した後も維持されます。
- スプライトチャネルがパペットでない場合は、そのチャネルにあるスプライトのプロパティに対してスクリプトが加えた変更は、そのスプライトが再生されている間のみ有効です。

puppetSprite() メソッドを使用する際は、指定したチャネル内にスプライトが入っている必要があります。

スプライトチャネルをパペットにすると、再生ヘッドがスプライトから抜け出した後でも、member、locH および width などの多くのスプライトプロパティをスクリプトから制御できます。

スクリプトからのスプライトチャネルの制御が必要なくなったときは、puppetSprite(**intSpriteNum**, FALSE) シンタックスを使用してスプライトチャネルのパペット状態をオフにします。これを怠ると、同じスプライトチャネルに配置された、本来パペットになっていないはずのスプライトがあるフレームに移動したときに、予期せぬ結果が生じることがあります。

注意：Director バージョン 6 には、自動パペット化機能が導入されました。これによって、ほとんどのケースで明示的なパペット化は不要になります。しかし、スプライトの再生後もチャネルの内容を完全に制御したいような場合には、やはり明示的にパペット化する必要があります。

パラメータ

intSpriteNum 必須。テストするスプライトチャネルを指定する整数です。

bool 必須。スプライトチャネルが、スクリプトコントロールによって制御されるか（TRUE）、スコアによって制御されるか（FALSE）を指定するブール値です。

例文

次のステートメントは、チャネル 15 のスプライトをパペットにします。

```
-- Lingo syntax
_movie.puppetSprite(15, TRUE)

// JavaScript syntax
_movie.puppetSprite(15, true);
```

次のステートメントは、チャネル (i + 1) のスプライトのパペット状態を解除します。

```
-- Lingo syntax
_movie.puppetSprite(i + 1, FALSE)

// JavaScript syntax
_movie.puppetSprite(i + 1, false);
```

関連事項

makeScriptedSprite(), ムービー , スプライトチャネル

puppetTempo()

使用法

```
-- Lingo syntax
_movie.puppetTempo(intTempo)

// JavaScript syntax
_movie.puppetTempo(intTempo);
```

解説

ムービーメソッド。テンポチャネルをパペット化し、テンポを指定のフレーム数に設定します。

テンポチャネルをパペットにすると、スクリプトがスコア内のテンポ設定よりも優先され、ムービーのテンポを変更できるようになります。

パペット化したテンポは、いちいち元の状態に戻す必要はありません。以降は、スコア内に設定されているテンポの変化がそのまま有効になります。

注意：フレームレートは、puppetTempo() メソッドで理論上の上限である 30000 fps まで設定できますが、非常に高速なコンピュータで、小さなアニメーションを扱う場合でもなければ、実際にこの値を使用することはありません。

パラメータ

intTempo 必須。テンポを指定する整数です。

例文

次のステートメントは、ムービーのテンポを 30 fps に設定します。

```
-- Lingo syntax
_movie.puppetTempo(30)

// JavaScript syntax
_movie.puppetTempo(30);
```

次のステートメントは、ムービーの元のテンポ値を 10 fps アップさせます。

```
-- Lingo syntax
_movie.puppetTempo(oldTempo + 10)

// JavaScript syntax
_movie.puppetTempo(oldTempo + 10);
```

関連事項

ムービー

puppetTransition()

使用法

```
-- Lingo syntax
_movie.puppetTransition(memberObjRef)
_movie.puppetTransition(int {, time} {, size} {, area})

// JavaScript syntax
_movie.puppetTransition(memberObjRef);
_movie.puppetTransition(int {, time} {, size} {, area});
```

解説

ムービーメソッド。現在のフレームと次のフレームとの間で指定したトランジションを実行します。

Xtra トランジションキャストメンバを使用するには、puppetTransition(**memberObjRef**) シンタックスを使用します。

Director 内蔵のトランジションを使用するときは、**int** に次の表のいずれかの値を指定します。[時間] には、トランジションに使う時間を 1/4 秒単位で指定します。最小値は 0、最大値は 120 (30 秒) です。[サイズ] には、トランジションのチャンクごとに変更されるピクセル数を指定します。最小値は 1、最大値は 128 です。値が小さいほどトランジションもスムーズになりますが、処理速度は遅くなります。

コード	トランジション	コード	トランジション
01	左から表示される	27	ランダム行
02	右から表示される	28	ランダム列
03	上から表示される	29	上から現れてカバーする
04	下から表示される	30	右上から現れてカバーする
05	中央から左右へ開く	31	左上から現れてカバーする
06	左右から水平に閉じる	32	右から現れてカバーする
07	中央から上下へ開く	33	左から現れてカバーする
08	上下から垂直に閉じる	34	下から現れてカバーする
09	中央から四隅に開く	35	右下から現れてカバーする
10	四隅から中央へ閉じる	36	左下から現れてカバーする
11	左へ押し出す	37	横ブラインド
12	右へ押し出す	38	チェッカーボード
13	下へ押し出す	39	右下からの縦ストライプ
14	上へ押し出す	40	左下からの縦ストライプ

コード	トランジション	コード	トランジション
15	上へ隠れていく	41	左上からの横ストライプ
16	右上へ隠れていく	42	左下からの横ストライプ
17	右へ隠れていく	43	右上からの横ストライプ
18	右下へ隠れていく	44	右下からの横ストライプ
19	下へ隠れていく	45	右上からの縦ストライプ
20	左下へ隠れていく	46	左上からの縦ストライプ
21	左へ隠れていく	47	ズームオープン
22	左上へ隠れていく	48	ズームクローズ
23	ピクセルごとに速くにじむ *	49	縦ブラインド
24	長方形でにじむ	50	ビットごとに速くにじむ *
25	正方形でにじむ	51	ピクセルごとににじむ *
26	パターンでにじむ	52	ビットごとににじむ *

アスタリスク (*) のマークがついているトランジションは、32 ビットに設定されたモニタ上では動作しません。

時間を短く設定することで、トランジションの高速化に直接結びつくようなことはありません。トランジションの実際のスピードは、[サイズ] と [時間] の設定によって決まってきます。例えば、[時間] がどんなに短く設定されていても、[サイズ] が 1 ピクセルだと、コンピュータ側での作業が多くなるためトランジションには時間がかかります。トランジションを高速化したければ、[時間] を短めに設定するのではなく、[滑らかさ] を大きめに設定します。

[範囲] には、変更範囲のみでトランジションが起こる（TRUE）か、ステージ全体で起こる（FALSE、デフォルト）かを設定します。[範囲] とは、スプライトが変更された範囲だけを対象にするかどうかということです。

パラメータ

memberObjRef は、Xtra トランジションキャストメンバを使用する場合に必要です。トランジションとして使用する Xtra キャストメンバの参照です。

int は、Director に組み込まれているトランジションを使用する場合に必要です。使用するトランジション数を指定する整数です。

time オプション。トランジションに使用する時間を 1/4 秒単位で指定する整数です。有効な値の範囲は 0 ～ 120 です。

size オプション。トランジションのチャンクごとに変更されるピクセル数を指定する整数です。有効な値の範囲は 1 ～ 128 です。

area オプション。変更範囲のみでトランジションが起こる（TRUE）か、ステージ全体で起こる（FALSE、デフォルト）かを指定するブール値です。

例文

次のステートメントは、" 左から表示される " トランジションを実行します。[範囲] が指定されていないので、デフォルト設定が使われ、変更された部分でのみトランジションが生じることになります。

```
-- Lingo syntax
_movie.puppetTransition(1)

// JavaScript syntax
_movie.puppetTransition(1);
```

次のステートメントは、" 右から現れる " トランジションを、時間 = 1 秒、滑らかさ = 20 の設定でステージ全体に行います。

```
-- Lingo syntax
_movie.puppetTransition(2, 4, 20, FALSE)

// JavaScript syntax
_movie.puppetTransition(2, 4, 20, false);
```

関連事項

ムービー

put()

使用法

```
-- Lingo syntax
put(value)

// JavaScript syntax
put(value);
```

解説

最上位レベルの関数。式を評価し、結果をメッセージウィンドウに表示します。

このメソッドの機能は、Lingo および JavaScript シンタックスの両方で使用できる最上位レベルの trace() メソッドと同じです。

このメソッドは、ムービー再生中に変数の値を監視するデバッグ用ツールとして使用できます。

パラメータ

value 必須。評価する式です。

例文

次のステートメントは、時刻 (PC ごとに出力フォーマットが異なる) をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_system.time())

// JavaScript syntax
put(_system.time());
```

次のステートメントは、変数 bid に割り当てられている値をメッセージウィンドウに表示します。

```
-- Lingo syntax
bid = "Johnson"
put(bid) -- "Johnson"

// JavaScript syntax
var bid = "Johnson";
put(bid); // Johnson
```

関連事項

trace()

qtRegisterAccessKey()

使用法

```
-- Lingo syntax
qtRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtRegisterAccessKey(categoryString, keyString);
```

解説

暗号化された QuickTime メディア用のキーの登録をするコマンドです。

キーとは、アプリケーションレベルのキーを指し、システムレベルのキーのことではありません。アプリケーションがキーの登録を解除したり、シャットダウンしたりすると、メディアにはアクセスできなくなります。

注意：セキュリティ上の理由により、すべての登録キーを表示することはできません。

関連事項

qtUnRegisterAccessKey()

qtUnRegisterAccessKey()

使用法

```
-- Lingo syntax
qtUnRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtUnRegisterAccessKey(categoryString, keyString);
```

解説

暗号化された QuickTime メディア用のキーの登録を解除するコマンドです。

キーとは、アプリケーションレベルのキーを指し、システムレベルのキーのことではありません。アプリケーションがキーの登録を解除すると、このキーで暗号化されたムービーのみが再生を続けます。他のメディアにはアクセスできません。

関連事項

qtRegisterAccessKey()

queue()

使用法

```
-- Lingo syntax
soundChannelObjRef.queue(memberObjRef)
soundChannelObjRef.queue(propList)

// JavaScript syntax
soundChannelObjRef.queue(memberObjRef);
soundChannelObjRef.queue(propList);
```

解説

サウンドチャネルメソッド。サウンドチャネルのキューにサウンドキャストメンバを追加します。

キューに登録されたサウンドは、play() メソッドを使用してすぐに再生できます。これは、play() メソッドの実行から再生開始までの遅延を短縮するため、キューに入った各サウンドをプリロードして実現しています。プリロードされるサウンドのデフォルトは、1500 ミリ秒です。queue() メソッドを使用して、パラメータを含むプロパティリストを送ってパラメータを変更できます。これらのパラメータは、setPlayList() メソッドでは渡すことはできません。

完成したムービーで使用する queue() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内のサウンドコントロールムービーを参照してください。

パラメータ

memberObjRef は、サウンドキャストメンバを指定する場合に必要です。キューに登録するサウンドキャストメンバの参照です。

propList は、プロパティリストをパラメータとして渡す場合に必要です。キューに登録するサウンドキャストメンバに適用するプロパティリストです。このようなプロパティには、

プロパティ	解説
#member	キューに登録するサウンドキャストメンバ。このプロパティは指定する必要があります。これ以外はすべてオプションです。
#startTime	再生を開始するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの先頭です。「startTime」を参照してください。
#endTime	再生を終了するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの最後です。「endTime」を参照してください。
#loopCount	#loopStartTime と #loopEndTime で定義されたループを再生する回数。デフォルトは 1 です。「loopCount」を参照してください。
#loopStartTime	ループを開始するサウンド内の時間 (単位はミリ秒)。「loopStartTime」を参照してください。
#loopEndTime	ループを終了するサウンド内の時間 (単位はミリ秒)。「loopEndTime」を参照してください。
#preloadTime	再生前にバッファに溜めておくサウンドの量 (単位はミリ秒)。「preloadTime」を参照してください。

例文

次のハンドラは、2 つのサウンドをキューに登録し、サウンドチャネル 2 で再生します。最初のサウンドであるキャストメンバ "Chimes" は、最初からすべて再生されます。2 番目のサウンドであるキャストメンバ "introMusic" は、3 秒目から再生された後、8 ～ 8.9 秒の部分が 5 回繰り返され、10 秒目で停止します。

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000, #loopCount:5,
#loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(member("Chimes"))
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000, "endTime",10000,
"loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    sound(2).play();
}
```

関連事項

endTime（サウンドチャネル）, loopCount, loopEndTime（サウンドチャネル）, loopStartTime, pause()（サウンドチャネル）, play()（サウンドチャネル）preLoadTime, setPlayList(), サウンドチャネル , startTime (サウンドチャネル), stop()（サウンドチャネル）

queue() (3D)

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.queue(motionName {, looped, startTime, endTime, scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.queue(motionName {, looped, startTime, endTime, scale, offset})
```

解説

keyframePlayer モディファイアおよび bonesPlayer モディファイアの 3D コマンド。指定したモーションをモディファイアの playList プロパティの最後に追加します。追加されたモーションは、プレイリストでそれよりも前に登録されているモーションの再生がすべて完了すると、モデルによって実行されます。

パラメータ

motionName 必須。追加するモーションの名前を指定します。

looped オプション。FALSE を指定するとモーションが一度だけ再生され、TRUE を指定すると繰り返し再生されます。

startTime オプション。モーションの始めからミリ秒単位で計測されます。ループが FALSE の場合、モーションは、オフセットで始まって終了時間で終了します。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後繰り返し再生される際には、開始時間で始まり、終了時間で終了します。

endTime オプション。モーションの始めからミリ秒単位で計測されます。ループが FALSE の場合、モーションは、オフセットで始まって終了時間で終了します。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後繰り返し再生される際には、開始時間で始まり、終了時間で終了します。モーションが終わりまで再生されるようにするには、終了時間を -1 に設定します。

scale オプション。モーションの実際の再生速度です。スケールは、モデルの #keyframePlayer モディファイアまたは #bonesPlayer モディファイアの playRate プロパティと掛け算され、モーションの実際の再生速度が求められます。

offset オプション。モーションの始めからミリ秒単位で計測されます。ループが FALSE の場合、モーションは、オフセットで始まって終了時間で終了します。ループが TRUE の場合、ループの最初の回は、オフセットで始まって終了時間で終了します。その後繰り返し再生される際には、開始時間で始まり、終了時間で終了します。

例文

次のステートメントは、"Walker" というモデルの bonesPlayer プレイリストの最後に、" 落ちる " というモーションを追加します。プレイリストで "Fall" より前のモーションがすべて実行されると、"Fall" が始めから終わりまで 1 回再生されます。

```
sprite(1).member.model("Walker").bonesPlayer.queue("Fall", 0, 0, -1, 1, 0)
```

次のステートメントは、"Walker" というモデルの bonesPlayer プレイリストの最後に、"Kick" というモーションを追加します。プレイリストで "Kick" より前のモーションがすべて実行されると、"Kick" がループによって繰り返し再生されます。ループの最初の回は、モーションの始めから 2000 ミリ秒のところから開始されます。その後ループによって繰り返し再生される際には、「Kick」の始めから 1000 ミリ秒のところから開始され、「Kick」の 5000 ミリ秒のところで終了します。再生レートは、モデルの bonesPlayer モディファイアの playRate プロパティに設定されている値の 3 倍になります。

```
sprite(1).member.model("Walker").bonesPlayer.queue("Kick", 1, 1000, 5000, 3, 2000)
```

関連事項
play() (3D), playNext() (3D), playRate (3D)

QuickTimeVersion()

使用法

```
-- Lingo syntax
QuickTimeVersion()

// JavaScript syntax
QuickTimeVersion();
```

解説
現在インストールされている QuickTime のバージョンを識別する浮動小数値を返す関数です。今までの QuickTimePresent 関数に代わるものです。

Windows では、QuickTime 3.0 以降の複数のバージョンがインストールされている場合、QuickTimeVersion() は、一番新しいバージョン番号を返します。QuickTime 3.0 以前のバージョンがインストールされている場合、QuickTimeVersion() はそのバージョン番号にかかわらず 2.1.2 という番号を返します。

パラメータ
なし

例文
次のステートメントは、QuickTimeVersion() を使用して、現在インストールされている QuickTime のバージョン番号をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(QuickTimeVersion())

// JavaScript syntax
put(QuickTimeVersion());
```

quit()

使用法

```
-- Lingo syntax
_player.quit()

// JavaScript syntax
_player.quit();
```

解説
プレーヤのメソッド。Director やプロジェクタを終了して、Windows の場合はデスクトップ、Mac の場合は Finder に戻るためのコマンドです。

パラメータ
なし

例文

次のステートメントは、ユーザが Windows 上で Ctrl+Q キー、または Mac 上で Command+Q キーを押したときに、処理を終了して Windows のデスクトップ、または Mac の Finder に戻るようにします。

```
-- Lingo syntax
if (_key.key = "q" and _key.commandDown) then
    _player.quit()
end if

// JavaScript syntax
if (_key.key == "q" && _key.commandDown) {
    _player.quit();
}
```

関連事項

プレーヤ

ramNeeded()

使用法

```
-- Lingo syntax
_movie.ramNeeded(intFromFrame, intToFrame)

// JavaScript syntax
_movie.ramNeeded(intFromFrame, intToFrame);
```

解説

ムービーメソッド。指定したフレーム範囲を表示するのに必要とするメモリ量をバイト単位で返します。例えば、32 ビットの画像を含むフレームのサイズを調べ、ramNeeded() が freeBytes() より大きい場合は、8 ビットの画像を含むフレームに移ることができます。KB（キロバイト）で示したいときは 1024 で割ります。

パラメータ

intFromFrame 必須。範囲内にある最初のフレームの番号を指定する整数です。

intToFrame 必須。範囲内にある最後のフレームの番号を指定する整数です。

例文

次のステートメントは、frameSize 変数をムービーのフレーム 100 ～ 125 の表示に必要なメモリ量に設定します。

```
-- Lingo syntax
frameSize = _movie.ramNeeded(100, 125)

// JavaScript syntax
var frameSize = _movie.ramNeeded(100, 125);
```

次のステートメントは、フレーム 100 ～ 125 を表示するのに必要なメモリ量が、利用可能なメモリ量よりも多いかどうかを調べ、多い場合は色深度の低いキャストメンバを使用したシーケンスに分岐させます。

```
-- Lingo syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) then
    _movie.go("8-bit")
end if

// JavaScript syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) {
    _movie.go("8-bit");
}
```

関連事項

freeBytes(), ムービー

random()

使用法

```
-- Lingo syntax
random(integerExpression)

// JavaScript syntax
random(integerExpression);
```

解説

最上位レベルの関数。1 から指定した値の間のランダムな整数を返します。この関数は、ゲームによってパスを変更する、乱数を割り当てる、スプライトの色や位置を変更するなど、ムービー内の値を変更するために使用します。

乱数の下限を 1 より小さくするには、random() 関数から適切な数値を引きます。例えば、random(n + 1) - 1 式では 0 ～ n までの数字の範囲が使えるようになります。

パラメータ

integerExpression 必須。乱数の最大値を指定します。

例文

次のステートメントは、変数 diceRoll にランダムな値を割り当てます。

```
-- Lingo syntax
diceRoll = (random(6) + random(6))

// JavaScript syntax
var diceRoll = (random(6) + random(6));
```

次のステートメントは、スプライト 10 のフォアカラーをランダムに変化させます。

```
-- Lingo syntax
sprite(10).foreColor = (random(256) - 1)

// JavaScript syntax
sprite(10).foreColor = (random(256) - 1);
```

次のハンドラは、2 つのムービーセグメントのどちらを再生するかをランダムに選択します。

```
-- Lingo syntax
on SelectScene
    if (random(2) = 2) then
        _movie.go("11a")
    else
        _movie.go("11b")
    end if
end

// JavaScript syntax
function SelectScene() {
    if (random(2) == 1) {
        _movie.go("11a");
    } else {
        _movie.go("11b");
    }
}
```

次のステートメントは、5 〜 100 の間のランダムな 5 の倍数を返します。

```
-- Lingo syntax
theScore = (5 * random(20))

// JavaScript syntax
var theScore = (5 * random(20));
```

randomVector()

使用法

```
-- Lingo syntax
randomVector()

// JavaScript syntax
randomVector();
```

解説

最上位レベルの関数。単位球体のサーフェス上でランダムに選択されたポイントを示す単位ベクトルを返します。

この関数は、常に randomVector() を使用した単位ベクトルが返されるという点で、vector(random(10)/10.0, random(10)/10.0, random(10)/10.0,) とは異なります。

単位ベクトルの長さは常に 1 です。

パラメータ

なし

例文

次のステートメントは、ランダムに定義された 2 つの単位ベクトルを作成し、メッセージウィンドウに表示します。

```
-- Lingo syntax
vec1 = randomVector()
vec2 = randomVector()
put(vec1 & RETURN & vec2)

// JavaScript syntax
var vec1 = randomVector();
var vec2 = randomVector();
put(vec1 + "\n" + vec2);
```

関連事項

vector()

randomVector

使用法

```
randomVector()
```

解説

3D コマンド。単位球体のサーフェス上でランダムに選択されたポイントを示す単位ベクトルを返します。このメソッドは、常に単位ベクトルが返されるという点で、vector(random(10)/10.0,random(10)/10.0,random(10)/10.0) とは異なります。

パラメータ

なし

例文

次のステートメントは、ランダムに定義された 2 つの単位ベクトルを作成し、メッセージウィンドウに表示します。

```
vec = randomVector()
put vec
-- vector(-0.1155, 0.9833, -0.1408)
vec2 = randomVector()
put vec2
-- vector(0.0042, 0.8767, 0.4810)
```

関連事項

getNormalized, generateNormals(), normalize

rawNew()

使用法

```
parentScript.rawNew()
rawNew(parentScript)
```

解説

on new ハンドラを呼び出さずに親スクリプトから子オブジェクトを作成する関数です。この関数により子オブジェクトのプロパティを初期化しないで、子オブジェクトの作成ができます。この関数は、特に後で多数の子オブジェクトを作成する場合に便利です。元の子オブジェクトを初期化するには、その on new ハンドラを呼び出します。

パラメータ
なし

例文
次のステートメントは、プロパティの初期化をしないで、親スクリプト CarParentScript から RedCar という子オブジェクトを作成します。

```
RedCar = script("CarParentScript").rawNew()
```

次のステートメントは、子オブジェクト RedCar のプロパティを初期化します。

```
RedCar.new()
```

関連事項
new(), script()

readBoolean

使用法
```
byteArrayObject.readBoolean()
```

解説
バイト配列メソッド。バイト配列からブール値を読み取ります。ブール値は 1 バイトを使用します。このメソッドはバイト値がゼロの場合に False を返し、それ以外の場合に TRUE を返します。

例文
```
--Lingo syntax
put bArray.readBoolean()

//JavaScript syntax
put(bArray.readBoolean());
```

readByteArray

使用法
```
readByteArray(intSize, [ByteArray], [intOffset])
```

解説
バイト配列メソッド。バイト配列の一部を別のバイト配列に読み取ります。既存のバイト配列が入力読み取りデータとして渡された場合、そのバイト配列の intOffset から始まる位置に追加されます。

パラメータ

パラメータ	解説	必須 / オプション
intSize	バイト配列のサイズ。	必須
ByteArray	バイト配列に追加する既存のバイト配列。	オプション
intOffset	開始オフセット位置。	オプション

例文

```
--Lingo syntax
bArray.readByteArray(10)

//JavaScript syntax
bArray.readByteArray(10);
```

readByteArray (FileIO Xtra)

使用法

```
FileIO.readByteArray(intSize,[byteArray],[intOffset])
```

解説

FileIO Xtra メソッド。intBytes により指定されたバイト数を読み取ります。既存のバイト配列が存在する場合、指定した intBytes は、intOffset により指定される位置で始まるバイト配列に追加されます。

パラメータ

パラメータ	解説	必須 / オプション
intSize	バイト配列のサイズ。	必須
ByteArray	バイトが追加される既存のバイト配列。	オプション
intOffset	開始オフセット位置。	オプション

例文

```
--Lingo syntax
FileIO.readByteArray(10,bArray,1)

//JavaScript syntax
FileIO.readByteArray(10,bArray,1);
```

readChar()

使用法

```
-- Lingo syntax
fileioObjRef.readChar()

// JavaScript syntax
fileioObjRef.readChar();
```

解説

FileIO メソッド。ファイルの次の文字を読み取り、返します。

文字の読み取りに readChar() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。Unicode ファイルを読み取る場合、readchar() は次のバイトのみを返し、Unicode 文字は返しません。Unicode 文字を読み取るには、readFile() メソッドを使用します。

パラメータ

なし

例文

次のステートメントは、読み取りおよび書き込みのアクセス権を持つ c:\xtra.txt ファイルを開き、文字「e」が見つかるまで、そのファイル内のすべての文字を読み取ります。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
repeat while(numToChar(objFileio.readChar()) <> 'e')
 put numToChar(objFileio.readChar())
end repeat

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
while(numToChar(objFileio.readChar()) != 'e')
{
trace(numToChar(objFileio.readChar()));
}
```

関連事項

Fileio, openFile()

readFile()

使用法

```
-- Lingo syntax
fileioObjRef.readFile()

// JavaScript syntax
fileioObjRef.readFile();
```

解説

FileIO メソッド。現在の位置から指定したファイルの最後まで読み取り、その結果をストリングで返します。

ファイルの読み取りに readFile() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。

readFile() を使用して UTF-8 または UTF-16 ファイルを読み取る場合、ファイルの先頭は想定どおりには取得されない場合があります。適切に読み取れるようにするには、スクリプトで BOM 文字を処理する必要があります。UTF-8 ファイルの場合、BOM 文字は EF BB BF です。UTF-16 ファイルの場合、BOM 文字は FF FE です。

パラメータ

なし

例文

次のステートメントは、読み取りおよび書き込みのアクセス権を持つ c:\xtra.txt ファイルを開き、そのファイルのコンテンツを読み取ります。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readFile()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
var contents = objFileio.readFile();
trace(contents);
```

関連事項

Fileio, openFile()

readFloat32

使用法

```
byteArrayObject.readFloat32()
```

解説

バイト配列メソッド。32 ビットの浮動小数点数値をバイト配列から読み取ります。

例文

```
--Lingo syntax
put bArray.readFloat32()

//JavaScript syntax
put(bArray.readFloat32());
```

readFloat64

使用法

```
byteArrayObject.readFloat64()
```

解説

バイト配列メソッド。64 ビットの浮動小数点数値をバイト配列から読み取ります。

例文

```
--Lingo syntax
put bArray.readFloat64()

//JavaScript syntax
put(bArray.readFloat64());
```

readInt8

使用法

```
byteArrayObject.readInt8()
```

解説

バイト配列メソッド。8 ビットの符号付き整数値をバイト配列から読み取ります。

例文

```
--Lingo syntax
put bArray.readInt8()

//JavaScript syntax
put(bArray.readInt8());
```

readInt16

使用法

```
byteArrayObject.readInt16()
```

解説

バイト配列メソッド。16 ビットの符号付き整数値をバイト配列から読み取ります。

例文

```
--Lingo syntax
put bArray.readInt16()

//JavaScript syntax
put(bArray.readInt16());
```

readInt32

使用法

```
byteArrayObject.readInt32()
```

解説

バイト配列メソッド。32 ビットの符号付き整数値をバイト配列から読み取ります。

例文

```
--Lingo syntax
put bArray.readInt32()

//JavaScript syntax
put(bArray.readInt32());
```

readLine()

使用法

```
-- Lingo syntax
fileioObjRef.readLine()

// JavaScript syntax
fileioObjRef.readLine();
```

解説

FileIO メソッド。次の RETURN を含むファイルの次の行を読み取り、ストリングで返します。

行の読み取りに readLine() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。

UNIX 形式のファイルについては、「setNewLineConversion()」を参照してください。

パラメータ

なし

例文

次のステートメントは、読み取りおよび書き込みのアクセス権で c:\xtra.txt ファイルを開き、そのファイルの最初の行を読み取ります。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents = objFileio.readLine()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readLine();
trace(contents);
```

読み取るべき追加の行が存在するかどうかを調べるには、次のコードを使用します。

```
-- Lingo:
if objFileio.getPosition() < objFileio.getLength() then -- There is at least one more
-- line available.
end if

// JavaScript syntax
if (objFileio.getPosition() < objFileio.getLength()) { // There is at least one more line
// available.
}
```

関連事項

Fileio, openFile(), setNewLineConversion()

readRawString

使用法

```
readRawString(intLen, [strCharSet])
```

解説
バイト配列メソッド。固定バイト数をストリングとして読み取ります。

パラメータ

パラメータ	解説	デフォルト値
intLen	バイト配列から読み取るストリングの長さ。	必須
strCharSet	必要なエンコーディング変換。	UTF-8

例文
```
--Lingo syntax
put bArray.readRawString(10)

//JavaScript syntax
put(bArray.readRawString(10));
```

readString

使用法
```
readString([strCharSet])
```

解説
バイト配列メソッド。バイト配列をストリングとして読み取ります。ストリングの BLOB の先頭 4 バイトに、ストリングの長さが格納されます。

パラメータ

パラメータ	解説	デフォルト値
strCharSet	必要なエンコーディング変換を指定します。	UTF-8

例文
```
--Lingo syntax
put bArray.readString()

//JavaScript syntax
put(bArray.readString());
```

readToken()

使用法
```
-- Lingo syntax
fileioObjRef.readToken(stringSkip, stringBreak)

// JavaScript syntax
fileioObjRef.readToken(stringSkip, stringBreak);
```

解説
FileIO メソッド。次のトークンを読み取り、ストリングで返します。

トークンの読み取りに readToken() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。

パラメータ

stringSkip 必須。トークン開始後に文字セットを指定するストリングです。返されるストリングには、**stringSkip** ストリングは含まれません。

stringBreak 必須。トークン終了前に文字セットを指定するストリングです。返されるストリングには、**stringBreak** ストリングは含まれません。

例文

次のステートメントは、読み取りおよび書き込みのアクセス権で c:\xtra.txt ファイルを開き、「Director is good.」間のトークンを取得します。スキップするストリングは「D」で、stringBreak は「g」です。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents = objFileio.readToken("D","g")
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readToken("D","g");
trace(contents);
```

出力は「irector is」です。

関連事項

Fileio, openFile()

readWord()

使用法

```
-- Lingo syntax
fileioObjRef.readWord()

// JavaScript syntax
fileioObjRef.readWord();
```

解説

FileIO メソッド。ファイルの次の単語を読み取り、ストリングで返します。

単語の読み取りに readWord() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。

パラメータ

なし

例文

次のステートメントは、読み取りおよび書き込みのアクセス権で c:\xtra.txt ファイルを開き、そのファイルの最初の単語を読み取ります。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readWord()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readWord();
trace(contents);
```

関連事項

Fileio, openFile()

realPlayerNativeAudio()

使用法

```
-- Lingo syntax
realPlayerNativeAudio()

// JavaScript syntax
realPlayerNativeAudio();
```

解説

RealMedia 関数。RealMedia キャストメンバのオーディオの部分が、RealPlayer® によって処理されるか（TRUE）、または Director によって処理されるか（FALSE）を示すグローバルフラグを取得および設定できます。この関数は、フラグの前の値を返します。

効率を上げるために、このフラグは RealPlayer の最初のロード (スコアで最初の RealMedia キャストメンバに遭遇するとき、または Lingo が RealMedia キャストメンバを最初に参照するとき) の前に設定する必要があります。RealPlayer がロードされた後は、フラグの変更はすべて無視されます。このフラグは、ムービースクリプトの prepareMovie イベントハンドラで実行する必要があります。このフラグの設定は、現在のムービーの間だけでなく、セッション全体で (Shockwave プレーヤの起動後、終了して再起動するまでの間) 有効です。

デフォルトでは、このフラグは FALSE に設定され、オーディオは Director によって処理されます。これにより、soundChannel プロパティを設定し、Lingo の標準のサウンドメソッドとプロパティを使用して、RealMedia スプライトのオーディオストリームを操作できます。この操作には、RealAudio® と Director のオーディオのミキシングも含まれます。このフラグを TRUE に設定すると、Lingo のサウンドチャネルの制御は処理されず、サウンドは RealPlayer によって処理されます。

パラメータ

なし

例文

次のコードは、realPlayerNativeAudio() 関数が FALSE に設定されていることを示します。これによって、RealMedia キャストメンバのオーディオが Director によって処理されます。

```
-- Lingo syntax
put(realPlayerNativeAudio())
-- 0

// JavaScript syntax
trace(realPlayerNativeAudio());
// 0
```

次のコードは、realPlayerNativeAudio() 関数を TRUE に設定します。これによって、RealMedia のオーディオが RealPlayer によって処理され、Lingo によるサウンドチャネルの制御はすべて無視されます。

```
-- Lingo syntax
realPlayerNativeAudio(TRUE)

// JavaScript syntax
realPlayerNativeAudio(1);
```

関連事項

soundChannel (RealMedia)

realPlayerPromptToInstall()

使用法

```
-- Lingo syntax
realPlayerPromptToInstall()

// JavaScript syntax
realPlayerPromptToInstall();
```

解説

RealMedia 関数。RealPlayer 8 の自動検出とアラートを有効にするか (TRUE)、または有効にしないか (FALSE) を示すグローバルフラグを取得および設定できます。

デフォルトでは、この関数は TRUE に設定されています。これにより、RealPlayer 8 を持っていないユーザが RealMedia を含むムービーをロードしようとすると、RealNetworks の Web サイトにアクセスして RealPlayer をインストールするかどうかを確認するためのメッセージが自動的に表示されます。realPlayerVersion() 関数とカスタムコードを使用して、検出とアラート用の独自のシステムを作成する場合は、このフラグを FALSE にします。このフラグを FALSE に設定して RealPlayer 8 の検出とアラート用の代替システムを実装しないと、RealPlayer を持っていないユーザは RealMedia キャストメンバを含むムービーをロードできても RealMedia スプライトを表示できません。

この関数は、ユーザのシステムにインストールされている RealPlayer のビルドナンバーを調べて、RealPlayer 8 がインストールされているかどうかを判断します。Windows システムでは、ビルド番号が 6.0.8.132 以降だと、RealPlayer 8 がインストールされていることになります。Mac システムでは、ビルド番号が 6.0.7.1001 以降の場合、RealPlayer 8 がインストールされていることになります。

このフラグは、ムービースクリプトの prepareMovie イベントハンドラで実行する必要があります。

この関数は、フラグの前の値を返します。

パラメータ

なし

例文

次のコードは、realPlayerPromptToInstall() 関数が TRUE に設定されていることを示します。これによって、RealPlayer を持っていないユーザは、RealPlayer のインストールを求められます。

```
-- Lingo syntax
put(realPlayerPromptToInstall()) -- 1

// JavaScript syntax
trace(realPlayerPromptToInstall()); // 1
```

次のコードは、realPlayerPromptToInstall() 関数を FALSE に設定します。この場合、検出とアラート用のシステムを作成しない限り、ユーザは RealPlayer のインストールを求められません。

```
-- Lingo syntax
realPlayerPromptToInstall(FALSE)

// JavaScript syntax
realPlayerPromptToInstall(0);
```

realPlayerVersion()

使用法

```
-- Lingo syntax
realPlayerVersion()

// JavaScript syntax
realPlayerVersion();
```

解説

RealMedia 関数。ユーザのシステムにインストールされている RealPlayer のビルド番号を示すストリングを返します。RealPlayer がインストールされていない場合は、空のストリングを返します。ユーザが RealMedia コンテンツを含む Director ムービーを見るには、RealPlayer 8 以上が必要です。Windows システムでは、ビルド番号が 6.0.8.132 以降だと、RealPlayer 8 がインストールされていることになります。Mac システムでは、ビルド番号が 6.0.7.1001 以降の場合、RealPlayer 8 がインストールされていることになります。

この関数の目的は、あらかじめ用意されている関数 realPlayerPromptToInstall() を使用せずに、RealPlayer の検出とアラート用の独自のシステムを作成できるようにすることです。

realPlayerVersion() 関数を使用して検出とアラート用の独自のシステムを作成するには、次の事項を実行する必要があります。

- RealMedia キャストメンバが Lingo で参照される前、またはスコアに出現する前に、realPlayerPromptToInstall(FALSE) を呼び出します。この関数はデフォルトで TRUE に設定されています。この関数は、ムービースクリプトの prepareMovie イベントハンドラで設定する必要があります。
- xtraList システムプロパティを使用して、RealMedia (RealMedia Asset.x32) 用の Xtra が [ムービーエクストラ] ダイアログボックスのリストに表示されていることを確認します。realPlayerVersion() 関数は、RealMedia 用の Xtra が存在しないと動作しません。

この関数が返すビルド番号は、RealPlayer に表示されるビルド番号と同じです。

Windows での RealPlayer ビルド番号の表示

1 RealPlayer を起動します。

2 [ヘルプ] メニューの [RealOne Player のバージョン情報] をクリックします。

ウィンドウが開き、ビルド番号が 2 行目の中央に表示されます。

Mac での RealPlayer ビルド番号の表示

1 RealPlayer を起動します。

2 Apple® メニューの RealOne Player のバージョン情報を選択します。

[RealOne Player のバージョン情報] ダイアログボックスが表示されます。上から 2 行目に表示されているビルド番号は正しくないので、無視します。

3 [バージョン情報] ボタンをクリックします。

[RealOne Player バージョン情報] ダイアログボックスが表示されます。

4 [インストール済みのコンポーネント] リストから [RealOne Player Core] を選択します。

RealPlayer Core コンポーネント (例えば、6.0.8.1649) のバージョン番号は、realPlayerVersion() 関数が返すージョン番号と同じです。

パラメータ

なし

例文

次のコードは、システムにインストールされている RealPlayer® のビルド番号が 6.0.9.357 であることを示します。

```
-- Lingo syntax
put(realPlayerVersion())

// JavaScript syntax
put(realPlayerVersion());
```

recordFont

使用法

```
recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {, userFontName})
```

解説

TrueType または Type 1 フォントをキャストメンバとして埋め込むコマンドです。これらのフォントが一旦埋め込まれると、作成者はシステムにインストールされている他のフォントと同じように使うことができます。

recordFont を使用する前に、new() コマンドを使用して空フォントのキャストメンバを作成する必要があります。

このコマンドは、[フォント] パラメータで名づけられたフォントを使用して Shock フォントを [キャストメンバ] 内に作成します。このコマンドで返される値により、動作が成功したかどうかがわかります。0 が表示されれば、成功です。

パラメータ

font 必須。記録するオリジナルフォントの名前を指定します。

face オプション。オリジナルフォントのフェイスを示すリストを指定します。使用できる値は #plain、#bold、#italic す。このパラメータに値を指定しないと、#plain が使用されます。値 [# bold, #italic] を指定した場合、これらが結合され、Bold-Italic スタイルを持つ新しいフォントが作成されます。

bitmapSizes オプション。ビットマップを記録するときのサイズを指定する整数のリストを指定します。このパラメータは空でも使用できます。このパラメータが省略されると、ビットマップは生成されません。ビットマップは、通常小さいポイントサイズ (14 ポイント以下) のほうが視覚的に優れていますが、より多くのメモリを必要とします

characterSubset オプション。エンコードする文字のストリングを指定します。指定された文字のみが、フォント内で利用可能となります。このパラメータが省略されると、すべての文字がエンコードされます。エンコードされている特定の文字以外が使われると、その文字は空白として表示されます。

userFontName オプション。新しく記録したフォントキャストメンバの名前として使用するストリングを指定します。

例文

次のステートメントは、2 つのパラメータ、キャストメンバおよび記録用フォントを使って、単純な Shocked フォントを作成します。

```
-- Lingo
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")

// Javascript
var myNewFontMember = new(symbol("font")) ;
myNewFontMember.recordFont( "Lunar Lander") ;
```

次のステートメントは、生成するビットマップサイズおよびフォントデータを作成する文字を指定します。

```
-- Lingo
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First Last Name")

// Javascript
var myNewFontMember = new(symbol("font")) ;
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First Last Name") ;
```

注意：recordFont はフォントデータを直接使用するというよりも、再合成されたものなのでプログラムの複製ではありません。ただし、ムービーを配布する際は、このような形態での使用がオリジナルフォントの使用許諾契約でカバーされているかどうかを確認することを強くお勧めします。

関連事項

newMember()

rect()

使用法

```
-- Lingo syntax
rect(intLeft, intTop, intRight, intBottom)

// JavaScript syntax
rect(intLeft, intTop, intRight, intBottom);
```

解説

最上位レベルの関数。矩形を定義します。

Lingo と JavaScript シンタックスの両方を使用して、矩形に対して演算を行うことができます。矩形に単一の値を追加した場合、Lingo および JavaScript シンタックスは矩形の各エレメントにその値を加えます。

矩形要素はリストシンタックスまたはプロパティシンタックスで参照することができます。例えば、次の割り当てで myRectWidth1 と myRectWidth2 の両方が 50 に設定されます。

```
// JavaScript syntax
var myRect = rect(40,30,90,70);
var myRectWidth1 = myRect.right - myRect.left; // 50
var myRectWidth2 = myRect[3] - myRect[1]; // 50
```

完成したムービーの rect() の実例は、Director アプリケーションフォルダの "Learning/Lingo" フォルダ内の "imaging.dir" を参照してください。

パラメータ

intLeft 必須。矩形の左辺がステージの左端になるピクセル数を指定する整数です。

intTop 必須。矩形の上辺がステージの上端になるピクセル数を指定する整数です。

intRight 必須。矩形の右辺がステージの左端になるピクセル数を指定する整数です。

intBottom 必須。矩形の下辺がステージの上端になるピクセル数を指定する整数です。

例文

次のステートメントは、変数 newArea に、左辺が 100、上辺が 150、右辺が 300、下辺が 400 の各ピクセル位置にある矩形を設定します。

```
-- Lingo syntax
newArea = rect(100, 150, 300, 400)

// JavaScript syntax
var newArea = rect(100, 150, 300, 400);
```

Lingo の場合のみ、次のステートメントは、newArea 変数に、firstPoint と secondPoint の 2 つのポイントで定義した矩形を設定します。

```
-- Lingo syntax
firstPoint = point(100, 150)
secondPoint = point(300, 400)
newArea = rect(firstPoint, secondPoint)
```

Lingo の場合のみ、次のステートメントは、矩形の値を加算および減算します。

```
-- Lingo syntax
put(rect(0, 0, 100, 100) + rect(30, 55, 120, 95)) -- rect(30, 55, 220, 195)
put(rect(0, 0, 100, 100) -rect(30, 55, 120, 95)) -- rect(-30, -55, -20, 5)
```

Lingo の場合のみ、次のステートメントは、矩形の各座標値に 80 を加えます。

```
-- Lingo syntax
put(rect(60, 40, 120, 200) + 80) -- rect(140, 120, 200, 280)
```

Lingo の場合のみ、次のステートメントは、矩形の各座標値を 3 で割ります。

```
-- Lingo syntax
put(rect(60, 40, 120, 200) / 3) -- rect(20, 13, 40, 66)
```

関連事項

point(), quad

registerByteArrayCallback

使用法

```
soundObject.registerByteArrayCallback(position, #callbackFunction, [castMemRef], [read-write/read-only
flag])
```

解説

バイト配列メソッド。#callbackFunction を使用して指定されたコールバックメソッドを呼び出します。バイト配列は、サウンドオブジェクトから受け取られたオーディオサンプルデータで満たされています。

パラメータ

パラメータ	解説	デフォルト値
position	このパラメータでは #preFilter および #postFilter を指定することができます。 #preFilter は、サウンドオブジェクトフィルタを適用する前にオーディオ PCM サンプルを指定します。 #postFilter は、サウンドオブジェクトフィルタを適用した後にオーディオ PCM サンプルを指定します。	必須
symCallback	サウンドオブジェクトのオーディオデータのコールバックメソッドとして callbackFunction を登録します。	必須
castMemRef	コールバックメソッドが親スクリプトのメンバである場合、キャストメンバ参照を指定します。コールバックメソッドが movieScript 内にある場合はパラメータを指定しないでください。	オプション
読み取り - 書き込み / 読み取り専用フラグ	コールバックが #readOnly であるか、#readWrite であるかを指定します。#readOnly - オーディオバイト配列データはコールバックメソッドで読み取り専用です。#readWrite - オーディオバイト配列データはコールバックメソッドで読み書きされます。そのため、データはコールバックで変更される可能性があります。	#readOnly

callbackFunction の実装は次のとおりです。

```
On callbackFunction byteArray, bitDepth, sampleRate, channelCount
    Put  byteArray --Audio output implementation
End
```

注意：コールバックハンドラの内部では、いずれかのスクリプトで実行時エラー（プロパティが見つからないなど）が発生した場合、エラーメッセージは表示されず、ハンドラの実行はエラーが発生した時点で中止されます。後続のすべてのステートメントは実行されません。

例文

```
-- BUTTON BEHAVIOR --
global gSound -- soundObject
global gMixer -- mixer object
global counter -- Counter to fill the silence
global fillSilence -- Flag to fill the silence
on mouseUp(me)
-------------------------------------------------------
-- ACTION: Creates a soundObject from the Cast member.
-- The callback handler HAS to be in a Parent Script.
-- See the Movie Script for callback details.
-------------------------------------------------------------------
gMixer = new(#mixer)
gSound = gMixer.createSoundObject("1", member(1))
--Create the sound object from cast member
sPre = new script("Output") -- Output is a parent script
sPost = new script("Output")
sPre.countLimit = 50 -- Setting the Property value for filling the silence
sPost.countLimit = 10
fl = gSound.filterlist
fl.append(audioFilter(#reverbFilter)) -- Adding Reverb Filter
gSound.registerByteArraycallback(#preFilter, #audioOutput,sPre,#readWrite)
-- Registering the callback with preFilter, so that the bytes will be available
--before applying filters
-- Specifying the script reference as sPre, can specify it as movie script if it is
--empty
-- Specifying readWrite, so that the bytes can be modified in the callback method
gSound.registerByteArrayCallback(#postFilter, #audioOutput,sPost,#readWrite)
-- Registering the callback with postFilter, so that the bytes will be available
--after applying filters
gMixer.bufferSize = 200 -- Setting the bufferSize of mixer to 200
gMixer.play()   -- Play the sound
end
```

コールバックメソッド（親スクリプト）

```
    -- PARENT SCRIPT named OUTPUT --
    global gSoundIOInstance -- instance of FileIO for reading sound file
    global gSound -- soundObject
    global counter -- Counter to  fill the  silence
    global fillSilence -- Flag to fill the silence
    property countLimit
    --Callback Method
    on audioOutput me, aByteArray, bitDepth, sampleFreq, numChannel
     ------------ --------- ---------------- -------
    -- SOURCE: Called back from gSound whenever gSound has finished
    --  playing its current contents.
    -- INPUT: <aByteArray> will be the byteArray with audio data which has to be
    -- modified and the byteArray to send back.
    -- ACTION: Transfers the (remaining) contents of the file to
    -- aByteArray
    -- OUTPUT: Returns TRUE if there was any data to pas to aByteArray
    -- FALSE if not.
  -------------------------------------------------------------------
    if fillSilence = 1 then -- Checking the fillSilence flag
    repeat with i = 1 to aByteArray.length -- Looping thru the bytes of audio data
    aByteArray[i] = 0 -- Setting the byte to silence
    --Modify the bytes before playing, here it is set to 0 means silence
    end repeat
    end if
    put aByteArray
    counter = counter + 1 -- Increasing the counter to keep filling the silence
    if(counter > countLimit)then -- If the countLimit is reached stop filling the silence
    counter = 0
    fillSilence = 1 - fillSilence -- Toggling between silence & play
    end if
    end
```

関連事項

unregisterByteArrayCallback

registerCuePointCallback

使用法

```
soundObject.registerCuePointCallback(#symCallback, [castMemRef])
```

解説

サウンドオブジェクトメソッド。現在のサウンドオブジェクトで再生しているサウンドストリームのキューポイントに対するコールバックメソッドとして、symCallback メソッドを登録します。

コールバックが親スクリプトのメンバに対するものであるときは、必要に応じて、キャストメンバの参照を渡すこともできます。

パラメータ

パラメータ	解説	必須 / オプション
symcallback	キューポイントにヒットした場合、シンボルを呼び出します。	必須
castMemRef	コールバックメソッドが親スクリプトのメンバである場合、キャストメンバ参照を渡します。	オプション

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.registerCuePointCallback(#callMe) -- Calls the callMe function when the
-- cue points of the sound object associated with soundObjRef are hit.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.registerCuePointCallback(#callMe); // Calls the callMe function when the cue
// points of the sound object associated with soundObjRef are hit.
}
```

関連事項

unregisterCuePointCallback

registerEndOfSpoolCallback()

使用法

```
soundObject.registerEndOfSpoolCallback(#callbackFun, #parentScriptRef)
```

解説

サウンドオブジェクトの読み取りが完了した場合、callbackFun をコールバック関数として登録します。このコールバックは、次に再生されるメンバを返します。オプションで、コールバック関数が親スクリプトの一部である場合、親スクリプトへの参照を渡すこともできます。

パラメータ

パラメータ	解説	必須 / オプション
callbackFun	サウンドオブジェクトの読み取りが完了した場合、シンボルを呼び出します。	必須
parentScriptRef	コールバックメソッドが親スクリプトのメンバである場合、キャストメンバ参照を渡します。	オプション

例文

```
 --Lingo syntax
on mous eUp mesoundObj Ref.r egist erEndOfSpoolCallback( #callMe)
-- Calls the callMe method when the sound objec t read is complete.
-- callme method should return the reference of next sound cast member
end
--The callme method is implemented as
On callme
bgcount = bgcount +1
return(member(bgcount))
End
// JavaScript syntax
function mouseUp(){
soundObjRef.registerEndOfSpoolCallback(#callMe); // Calls the callMe method when the
// sound object read is finished. callme method should return the reference of next sound cast member
}
```

関連事項
unregisterEndOfSpoolCallback()

registerForEvent()

使用法

```
member(whichCastmember).registerForEvent(eventName, handlerName, scriptObject {, begin, period,
repetitions})
```

解説

3D コマンド。指定したキャストメンバで指定したイベントが発生したときに呼び出されるハンドラとして、指定したハンドラを宣言します。

以下のパラメータの説明は、registerForEvent() コマンドと registerScript() コマンドの両方に当てはまります。

注意：registerScript() コマンドを使うと、スクリプトの登録をキャストメンバではなく特定のノードに関連付けることができます。

パラメータ

eventName 必須。イベントの名前を指定します。イベントには、以下の定義済みイベントのいずれか、または独自に定義したカスタムイベントを指定できます。

- #collideAny。衝突イベントです。
- #collideWith。現在のモデルが含まれる衝突イベントです。setCollisionCallback() コマンドは、#collideWith イベントで registerScript() コマンドを使用するためのショートカットです。
- #animationStarted と #animationEnded。ボーンアニメーションやキーフレームアニメーションの再生が開始または終了した際に発生する通知イベントです。ハンドラが受け取るパラメータは、イベント名、モーション、時間の 3 つです。イベント名パラメータには、#animationStarted または #animationEnded のいずれかを指定します。モーションパラメータは、再生を開始または終了したモーションの名前です。時間パラメータは、モーションの現在の時間です。
- アニメーションがループする場合、#animationStarted イベントは最初のループに対してのみ発生し、以降のループでは発生しません。2 つのアニメーションのブレンドの際には、ブレンドが開始されるときにこのイベントが送信されます。
- 一連のアニメーションがモデルに対してキューを作って待機しており、かつ、アニメーションの autoBlend プロパティが TRUE に設定されている場合は、指定したモーションが見かけ上終了する前に、#animationEnded イベントが発生する場合があります。この現象は、アニメーションが定義されたとおりに終了していても autoBlend プロパティによって、モーションが続行しているように見えるために発生します。
- #timeMS 。時間イベントです。最初の #timeMS イベントは、registerForEvent が呼び出されてから、開始パラメータに指定した時間 (単位はミリ秒) が経過したときに発生します。期間パラメータは、反復回数の値が 0 より大きい場合に、#timeMS イベント間の間隔をミリ秒単位で指定します。反復回数が 0 の場合、#timeMS イベントは無限に発生します。

handlerName 必須。ハンドラ名イベントが発生したときに呼び出されるハンドラの名前を指定します。このハンドラは、スクリプトオブジェクトで指定したスクリプトオブジェクトで検索されます。ハンドラには、以下のパラメータが渡されます。

- type。常に 0 です。
- delta。最後の #timeMS イベントが発生してからの経過時間 (単位はミリ秒) です。
- time。最初の #timeMS イベントが発生してからの経過時間 (単位はミリ秒) です。例えば、500 ミリ秒の間隔で 3 回の繰り返しがある場合は、最初の繰り返しの時間は 0 になり、2 回目は 500、3 回目は 1000 になります。

- duration。registerForEvent が呼び出されてから最後の #timeMS イベントが呼び出されるまでの時間の合計 (単位はミリ秒) です。例えば、500 ミリ秒の間隔で 5 回の繰り返しがある場合、duration は 2500 ミリ秒になります。無限に繰り返されるタスクの duration は 0 になります。
- systemTime。Director ムービーが開始されてからの絶対時間 (単位はミリ秒) です。

scriptObject 必須。**handlerName** ハンドラを含むスクリプトオブジェクトを指定します。スクリプトオブジェクトに 0 を指定すると、ムービースクリプト内で最初に見つかった、指定した名前のイベントハンドラが呼び出されます。

begin オプション。最初の #timeMS イベントを発生させる registerForEvent() を呼び出してからの経過時間をミリ秒単位で指定します。

period オプション。反復回数の値が 0 より大きい場合に、#timeMS イベント間の間隔をミリ秒単位で指定します。

反復回数はオプションです。#timeMS イベントの繰り返し回数を指定します。反復回数を 0 に設定すると、#timeMS イベントは無期限で発生します。

例文

次のステートメントは、ムービースクリプト内にある promptUser イベントハンドラを、5 秒の間隔で 2 回呼び出されるように登録します。

```
member("Scene").registerForEvent(#timeMS, #promptUser, 0, 5000, 5000, 2)
```

次のステートメントは、ムービースクリプト内にある promptUser イベントハンドラを、"Scene" というキャストメンバで衝突が発生するたびに呼び出されるように登録します。

```
member("Scene").registerForEvent(#collideAny, #promptUser, 0)
```

次のステートメントは、キャストメンバ "Scene" 内のモデル "Pluto" にオブジェクトが衝突するたびに promptUser ハンドラが呼び出されるように宣言します。このハンドラは、registerForEvent コマンドが含まれるスクリプトと同じスクリプト内にあります。

```
member("Scene").registerForEvent(#collideWith, #promptUser, me, member("Scene").model("Pluto"))
```

関連事項

setCollisionCallback(), registerScript(), play() (3D), playNext() (3D), autoblend, blendTime, sendEvent, unregisterAllEvents

registerScript()

使用法

```
member(whichCastmember).model(whichModel).registerScript(eventName, handlerName, scriptObject {, begin,
period, repetitions})
member(whichCastmember).camera(whichCamera).registerScript(eventName, handlerName, scriptObject {, begin,
period, repetitions})
member(whichCastmember).light(whichLight).registerScript(eventName, handlerName, scriptObject {, begin,
period, repetitions})
member(whichCastmember).group(whichGroup).registerScript(eventName, handlerName, scriptObject {, begin,
period, repetitions})
```

解説

3D コマンド。指定したハンドラを、指定したイベントが参照先のノードに対して発生したときに呼び出されるように登録します。

以下のパラメータの説明は、registerForEvent() コマンドと registerScript() コマンドの両方に当てはまります。

パラメータ

eventName 必須。イベントの名前を指定します。イベントには、以下の定義済みイベントのいずれか、または独自に定義したカスタムイベントを指定できます。

- #collideAny。衝突イベントです。
- #collideWith。現在のモデルが含まれる衝突イベントです。setCollisionCallback() コマンドは、#collideWith イベントで registerScript() コマンドを使用するためのショートカットです。
- #animationStarted と #animationEnded。ボーンアニメーションやキーフレームアニメーションの再生が開始または終了した際に発生する通知イベントです。ハンドラが受け取るパラメータは、イベント名、モーション、時間の 3 つです。イベント名パラメータには、#animationStarted または #animationEnded のいずれかを指定します。モーションパラメータは、再生を開始または終了したモーションの名前です。時間パラメータは、モーションの現在の時間です。

  アニメーションがループする場合、#animationStarted イベントは最初のループに対してのみ発生し、以降のループでは発生しません。2 つのアニメーションのブレンドの際には、ブレンドが開始されるときにこのイベントが送信されます。

  一連のアニメーションがモデルに対してキューを作って待機しており、かつ、アニメーションの autoBlend プロパティが TRUE に設定されている場合は、指定したモーションが見かけ上終了する前に、#animationEnded イベントが発生する場合があります。この現象は、アニメーションが定義されたとおりに終了していても autoBlend プロパティによって、モーションが続行しているように見えるために発生します。
- #timeMS 。時間イベントです。最初の #timeMS イベントは、registerForEvent が呼び出されてから、開始パラメータに指定した時間 (単位はミリ秒) が経過したときに発生します。期間パラメータは、反復回数の値が 0 より大きい場合に、#timeMS イベント間の間隔をミリ秒単位で指定します。反復回数が 0 の場合、#timeMS イベントは無限に発生します。

handlerName 必須。ハンドラ名イベントが発生したときに呼び出されるハンドラの名前を指定します。このハンドラは、スクリプトオブジェクトで指定したスクリプトオブジェクトで検索されます。ハンドラには、以下のパラメータが渡されます。

- type。常に 0 です。
- delta。最後の #timeMS イベントが発生してからの経過時間 (単位はミリ秒) です。
- time。最初の #timeMS イベントが発生してからの経過時間 (単位はミリ秒) です。例えば、500 ミリ秒の間隔で 3 回の繰り返しがある場合は、最初の繰り返しの時間は 0 になり、2 回目は 500、3 回目は 1000 になります。
- duration。registerForEvent が呼び出されてから最後の #timeMS イベントが呼び出されるまでの時間の合計 (単位はミリ秒) です。例えば、500 ミリ秒の間隔で 5 回の繰り返しがある場合、duration は 2500 ミリ秒になります。無限に繰り返されるタスクの duration は 0 になります。
- systemTime。Director ムービーが開始されてからの絶対時間 (単位はミリ秒) です。

scriptObject 必須。**handlerName** ハンドラを含むスクリプトオブジェクトを指定します。スクリプトオブジェクトに 0 を指定すると、ムービースクリプト内で最初に見つかった、指定した名前のイベントハンドラが呼び出されます。

begin オプション。最初の #timeMS イベントを発生させる registerForEvent() を呼び出してからの経過時間をミリ秒単位で指定します。

period オプション。反復回数の値が 0 より大きい場合に、#timeMS イベント間の間隔をミリ秒単位で指定します。

反復回数はオプションです。#timeMS イベントの繰り返し回数を指定します。反復回数を 0 に設定すると、#timeMS イベントは無期限で発生します。

例文

次のステートメントは、ムービースクリプト内にある messageReceived イベントハンドラを、" プレーヤ " というモデルが #message というユーザ定義のカスタムイベントを受け取ったときに呼び出されるように登録します。

```
member("Scene").model("Player").registerScript(#message, #messageReceived, 0)
```

次のステートメントは、registerScript コマンドと同じスクリプト内にある collisionResponder イベントハンドラを、#collision モディファイアを使用しているモデルが "Player" というモデルに衝突するたびに呼び出されるように登録します。

```
member("Scene").model("Player").registerScript(#collideWith, #collisionResponder, me)
```

関連事項

registerForEvent(), sendEvent, setCollisionCallback()

removeBackdrop

使用法

```
member(whichCastmember).camera(whichCamera).removeBackdrop(index)
```

解説

3D コマンド。指定した位置の背景を、カメラの背景のリストから削除します。

パラメータ

index 必須。背景のリストで背景のインデックス位置を指定します。

例文

次のステートメントは、" シーン " というメンバ内のカメラ 1 の背景リストから 3 番目の背景を削除します。このカメラを現在使用しているスプライトがある場合、この背景はステージから消えます。

```
-- Lingo
member("Scene").camera[1].removeBackdrop(3)

// Javascript
member("Scene").getProp("camera", 1).removeBackdrop(3) ;
```

関連事項

insertBackdrop, overlay, backdrop

removeFromWorld

使用法

```
member(whichCastmember).model(whichModel).removeFromWorld()
member(whichCastmember).light(whichLight).removeFromWorld()
member(whichCastmember).camera(whichCamera).removeFromWorld()
member(whichCastmember).group(whichGroup).removeFromWorld()
```

解説

3D コマンド。ワールドを親子階層のルートとして持つモデル、ライト、カメラ、またはグループに対してこのコマンドを発行すると、親が void に設定されるため、ワールドから削除されます。

親子階層のルートがワールドではないオブジェクトに対しては、このコマンドは何も実行しません。

パラメータ

なし

例文

次のコマンドは、"Scene" というキャストメンバの 3D ワールドから "gbCyl" というモデルを削除します。

```
-- Lingo
member("Scene").model("gbCyl").removeFromWorld()

// Javascript
member("Scene").getPropRef("model" , a).removeFromWorld() ;
// where a is the number index for gbCyl model.
```

removeLast()

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.removeLast()
member(whichCastmember).model(whichModel).keyframePlayer.removeLast()
```

解説

keyframePlayer モディファイアおよび bonesPlayer モディファイアの 3D コマンド。モディファイアの playlist の最後のモーションを削除します。

パラメータ

なし

例文

次のステートメントは、"Walker" というモデルの bonesPlayer モディファイアのプレイリストから最後のモーションを削除します。

```
member("MyWorld").model("Walker").bonesPlayer.removelast()
```

removeModifier

使用法

```
member(whichCastmember).model(whichModel).removeModifier.(#whichModifier)
```

解説

3D コマンド。指定したモデルから指定したモディファイアを削除します。

このコマンドは、正常に終了した場合は TRUE を返します。# モディファイアが有効なモディファイアではない場合、またはモディファイアがモデルに添付されていない場合は、FALSE を返します。

パラメータ

whichModifier 必須。削除するモディファイアを指定します。

例文

次のステートメントは、"Box" というモデルから #toon モディファイアを削除します。

```
-- Lingo
member("shapes").model("Box").removeModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , i).removeModifier(symbol("toon"));
// where "i" is the number index.
```

関連事項

addModifier, modifier, modifier[], modifiers

removeOverlay

使用法

```
member(whichCastmember).camera(whichCamera).removeOverlay(index)
```

解説

3D コマンド。指定した位置のオーバーレイを、カメラのオーバーレイのリストから削除します。

パラメータ

index 必須。オーバーレイのリストでオーバーレイのインデックス位置を指定します。

例文

次のステートメントは、スプライト 5 で使用されているカメラのオーバーレイリストから最初のオーバーレイを削除します。このオーバーレイはステージから消えます。

```
-- Lingo
sprite(5).camera.removeOverlay(1)

// Javascript
sprite(5).camera.removeOverlay(1) ;
```

関連事項

overlay, addOverlay

removeScriptedSprite()

使用法

```
-- Lingo syntax
spriteChannelObjRef.removeScriptedSprite()

// JavaScript syntax
spriteChannelObjRef.removeScriptedSprite();
```

解説

スプライトチャネル。スプライトチャネルのコントロールをスクリプトからスコアに切り替えます。

パラメータ

なし

例文

次のステートメントは、スプライトチャネル 5 からスクリプト化されたスプライトを削除します。

```
-- Lingo syntax
channel(5).removeScriptedSprite()

// JavaScript syntax
channel(5).removeScriptedSprite();
```

関連事項

makeScriptedSprite(), スプライトチャネル

replaceMember

使用法

```
soundObject.replaceMember(<member>, [startTime, endTime, loopCount, loopStartTime, loopEndTime,
preLoadTime])
```

解説

サウンドオブジェクトメソッド。サウンドオブジェクトと関連付けられているオーディオキャストメンバを動的に置き換えます。この置き換えは瞬時に行われます。

新しいキャストメンバと古いキャストメンバの形式が異なる場合、再生の前に、新しいキャストメンバは古いキャストメンバの形式に変換されます。そのため、サウンドオブジェクトは、作成時のキャストメンバの形式を使用して常に再生されます。

サウンドオブジェクトが停止され再生される場合、サウンドオブジェクトは作成時のキャストメンバを再生します。

注意：replaceMember() メソッドは、バイト 配列オーディオメンバ、およびビデオメンバのオーディオトラックに対しては機能しません。

パラメータ

パラメータ	解説
member	現在再生中のサウンドを置き換えるオーディオキャストメンバ。

replaceMember メソッドのパラメータの残りは、createSoundObject メソッドのパラメータに似ています。
createSoundObject を参照してください。

例文

以下の例では、gun サウンドオブジェクトメンバを、無限ループで再生されるメンバ laser で置き換えます。

```
-- Lingo syntax
on mouseUp me
gun.replaceMember(member("laser"),[#loopcount:0])
end

// JavaScript syntax
function mouseUp(){
gun.replacemember(member("laser");
}
```

関連事項
createSoundObject

reset （ミクサ）

使用法
```
mixer.reset()
```

解説
オーディオメソッド。最後の保存操作の前の状態にミクサを戻します。最後の保存の後でミクサが作成されている場合、このメソッドはそのミクサ（およびその中のサウンドオブジェクトとフィルタ）を空にします。

reset は、ミクサが停止状態のときにのみ呼び出してください。このメソッドは、成功の場合は 1、失敗の場合は 0 を返します。例えば、ミクサを再生しているときにこのメソッドを呼び出すと、0 を返します。

例文
```
-- Lingo syntax
on stopMovie
     mixer1.reset()
end

// JavaScript syntax
function stopMovie() {
     mixer1.reset();

}
```

関連事項
ミクサ

resetWorld

使用法
```
member(whichCastmember).resetWorld()
member(whichTextCastmember).resetWorld()
```

解説
3D コマンド。参照先の 3D キャストメンバのメンバのプロパティを、最初にメモリにロードしたときに格納された値にリセットします。このコマンドを使用する際には、メンバの state プロパティが、0（アンロード）、4（メディアロード済み）、または -1（エラー）のいずれかになっていないと、スクリプトエラーが発生します。

このコマンドは、使用する値を、メンバが最初に作成されたときの状態ではなく、最初にメモリにロードされたときの状態から取得するという点で、revertToWorldDefaults とは異なります。

パラメータ
なし

例文

次のステートメントは、" シーン " というキャストメンバのプロパティを、最初にメモリにロードしたときの値にリセットします。

```
-- Lingo
member("Scene").resetWorld()

// Javascript
member("Scene").resetWorld() ;
```

関連事項

revertToWorldDefaults

resolveA

使用法

```
collisionData.resolveA(bResolve)
```

解説

3D 衝突メソッド。collisionData.modelA の collision.resolve プロパティによって設定されている衝突時のビヘイビアーを上書きします。この関数を呼び出すのは、collision.resolve を使用して設定されている modelA のビヘイビアーを上書きする場合だけです。

パラメータ

bResolve 必須。modelA との衝突を解決するかどうかを指定します。modelA の衝突は、**bResolve** が TRUE の場合は解決され、**bResolve** が FALSE の場合は解決されません。

関連事項

collisionData, registerScript(), resolve, modelA, setCollisionCallback()

resolveB

使用法

```
collisionData.resolveB(bResolve)
```

解説

3D 衝突メソッド。collisionData.modelB の collision.resolve プロパティによって設定されている衝突時のビヘイビアーを上書きします。この関数を呼び出すのは、collision.resolve を使用して設定されている modelB のビヘイビアーを上書きする場合だけです。

パラメータ

bResolve 必須。modelB との衝突を解決するかどうかを指定します。modelB の衝突は、**bResolve** が TRUE の場合は解決され、**bResolve** が FALSE の場合は解決されません。

関連事項

collisionData, resolve, registerScript(), modelB, setCollisionCallback()

restart()

使用法

```
-- Lingo syntax
_system.restart()

// JavaScript syntax
_system.restart();
```

解説

システムメソッド。開いているすべてのアプリケーションを閉じて、コンピュータを再起動します。

パラメータ

なし

例文

次のステートメントは、ユーザが Command+R キーを押したとき（Mac の場合）または Ctrl+R キーを押したときとき（Windows の場合）にコンピュータを再起動します。

```
-- Lingo syntax
if (_key.key = "r" and _key.commandDown) then
    _system.restart()
end if

// JavaScript syntax
if (_key.key = "r" && _key.commandDown) {
    _system.restart();
}
```

関連事項

システム

restore()

使用法

```
-- Lingo syntax
windowObjRef.restore()

// JavaScript syntax
windowObjRef.restore();
```

解説

ウィンドウメソッド。最大化した後にウィンドウを復元します。

ウィンドウ内のムービー用のカスタムタイトルバーを作成する場合にこのメソッドを使用します。

パラメータ

なし

例文

次のステートメントは、コントロールパネルという名前の最大化されたウィンドウを復元します。

```
-- Lingo syntax
window("Control Panel").restore()

// JavaScript syntax
window("Control Panel").restore();
```

関連事項

maximize(), ウィンドウ

result

使用法

```
the result
```

解説

最後に実行されたハンドラ内の式の結果値を返す関数です。

result 関数は、ウィンドウ内で再生しているムービーから値を取得したり、ムービー再生中にハンドラの結果をメッセージウィンドウに表示させることで Lingo プロセスを監視するなどの目的で使うと有効です。

ハンドラから結果を返すには、結果を変数に割り当て、その変数の値を確認します。set myVariable = **function()** のようなステートメントを使用します。ここで **function()** は、指定された任意の関数名です。

パラメータ

なし

例文

次のステートメントは、2 つのサイコロのランダムな目の合計値を返します。

```
on diceRoll
    return random(6) + random(6)
end
```

次に、2 つのステートメントを示します。

```
diceRoll
roll = the result
```

上記のステートメントは、次のステートメントと同等です。

```
set roll = diceRoll()
```

roll = diceRoll というステートメントでは、diceRoll の後に括弧がないので、ハンドラは呼び出しません。ここでは diceRoll は変数参照であるとみなされます。

関連事項

return （キーワード）

resume()

使用法

```
-- Lingo syntax
animGifSpriteRef.resume()

// JavaScript syntax
animGifSpriteRef.resume();
```

解説

アニメーション GIF メソッド。スプライトが一時停止しているときに、現在のフレームの直後からスプライトの再生を再開します。アニメーション GIF スプライトが一時停止していない場合は、このコマンドは効力を持ちません。

パラメータ

なし

関連事項

rewind()（アニメーション GIF、Flash）

returnToTitle()

使用法

```
-- Lingo syntax
dvdObjRef.returnToTitle()

// JavaScript syntax
dvdObjRef.returnToTitle();
```

解説

DVD メソッド。メニューを表示した後に再生を再開します。

パラメータ

なし

例文

次のステートメントはメニューを表示した後に再生を再開します。

```
-- Lingo syntax
member(1).returnToTitle()

// JavaScript syntax
member(1).returnToTitle()
```

関連事項

DVD

revertToWorldDefaults

使用法

```
member(whichCastmember).revertToWorldDefaults()
```

解説

3D コマンド。参照先の 3D キャストメンバのプロパティを、メンバが最初に作成されたときに格納された値に戻します。このコマンドを使用するには、メンバの state プロパティが 4（ロード済み）または -1（エラー）のいずれかであることが必要です。この値になっていないと、スクリプトエラーが発生します。

このコマンドは、使用する値を、メンバが最初にメモリにロードされたときの状態ではなくメンバが最初に作成されたときの状態から取得するという点で、resetWorld とは異なります。

パラメータ

なし

例文

次のステートメントは、" シーン " というキャストメンバのプロパティを、最初の作成時に格納された値に戻します。

```
-- Lingo
member("Scene").revertToWorldDefaults()

// Javascript
member("Scene").revertToWorldDefaults() ;
```

関連事項

resetWorld

rewind() (MP4Media/FLV)

使用法

```
sprite(1).rewind()
```

解説：

MP4Media/FLV スプライトメソッド。ビデオの先頭をシークします。このメソッドを呼び出しても mediaStatus は変化しません。

例文

次の例は、MP4 ビデオスプライトを巻き戻します。

```
-- Lingo syntax
sprite(2).rewind()

// JavaScript syntax
sprite(2).rewind();
```

rewind() （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.rewind()

// JavaScript syntax
soundChannelObjRef.rewind();
```

解説

サウンドチャネルメソッド。サウンドチャネル内の現行サウンドの再生を中断して、startTime から再開します。

サウンドが一時停止している場合は、currentTime から startTime へ設定され停止したままになります。

パラメータ

なし

例文

次のステートメントは、サウンドチャネル 1 で再生中のサウンドキャストメンバを最初から再生を開始します。

```
-- Lingo syntax
sound(1).rewind()

// JavaScript syntax
sound(1).rewind();
```

関連事項

サウンドチャネル , startTime (サウンドチャネル)

rewind() (Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.rewind()

// JavaScript syntax
windowsMediaObjRef.rewind();
```

解説

Windows Media キャストメンバまたはスプライトメソッドです。Windows Media キャストメンバまたはスプライトの最初のフレームを巻き戻しします。

このメソッドを呼び出しても、mediaStatus に影響はありません。

パラメータ

なし

関連事項

mediaStatus (RealMedia, Windows Media), Windows Media

rewind() （アニメーション GIF、Flash）

使用法

```
-- Lingo syntax
animGifSpriteRef.rewind()

// JavaScript syntax
animGifSpriteRef.rewind();
```

解説

スプライトが停止したときまたは再生中に、Flash もしくはアニメーション GIF ムービースプライトをフレーム 1 に戻すコマンドです。

パラメータ

なし

例文

次フレームスクリプトは、ビヘイビアーの置かれているスプライト内の Flash ムービースプライトが再生中かどうかを確認し、再生中の場合は、現在のフレームでループを続行します。ムービーが終了すると、スプライトはムービーを巻き戻します (ムービーの最初のフレームがステージに表示されます)。そして、再生ヘッドを次のフレームに移します。

```
-- Lingo syntax
property spriteNum

    on exitFrame
    if sprite(spriteNum).playing then
        _movie.go(_movie.frame)
    else
        sprite(spriteNum).rewind()
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame(me) {
    var plg = sprite(this.spriteNum).playing;
    if (plg == 1) {
        _movie.go(_movie.frame);
    } else {
        sprite(this.spriteNum).rewind();
        _movie.updateStage();
    }
}
```

rollOver()

使用法

```
-- Lingo syntax
_movie.rollOver({intSpriteNum})

// JavaScript syntax
_movie.rollOver({intSpriteNum});
```

解説

ムービーメソッド。指定したスプライトの境界矩形上に、現在ポインタ（カーソル）が位置しているかどうかを示します。位置しているときは TRUE または 1 を返し、位置していないときは FALSE または 0 を返します。

通常、rollOver() メソッドはフレームスクリプト内で使用され、ユーザが特定のスプライト上にポインタを移動させたときの動作を定義するハンドラの作成に役に立ちます。

ユーザがマウスを動かしつづけると、スクリプトがハンドラを実行している間に rollOver() の値が変更されることがあります。ハンドラが一定のロールオーバー値を使用することを確実にするには、ハンドラが開始されてからすぐに rollOver() を変数に割り当てます。

以前スプライトが表示されていたステージの領域にポインタを置いたときも rollOver() は発生し、スプライトがあたかもそこに存在するかのように報告されます。このようなビヘイビアーを避けるには、このような位置でロールオーバーが生じないようにするか、スプライトを削除する前にメニューバーのさらに上方に移動させます。

パラメータ

intSpriteNum オプション。スプライト番号を指定する整数です。

例文

次のステートメントは、カーソルがスプライト 6 の上に位置するとき、フィールドキャストメンバ " メッセージ " の内容を " この場所です " に変更します。

```
-- Lingo syntax
if (_movie.rollOver(6)) then
    member("Message").text = "This is the place."
end if

// JavaScript syntax
if (_movie.rollOver(6)) {
    member("Message").text = "This is the place.";
}
```

次のハンドラは、カーソルがステージ上の特定のスプライトの上に位置するときに、再生ヘッドをそれぞれのマーカに送ります。まず、rollOver の値が変数に割り当てられます。これにより、ユーザーがマウスを動かし続けられるかどうかにかかわらず、ハンドラはロールオーバーが開始したときに有効である rollOver の値を利用できます。

```
-- Lingo syntax
on exitFrame
    currentSprite = _movie.rollOver()
    case currentSprite of
        1: _movie.go("Left")
        2: _movie.go("Middle")
        3: _movie.go("Right")
    end case
end exitFrame

// JavaScript syntax
function exitFrame() {
    var currentSprite = _movie.rollOver();
    switch (currentSprite) {
        case 1: _movie.go("Left");
            break;
        case 2: _movie.go("Middle");
            break;
        case 3: _movie.go("Right");
            break;
    }
}
```

関連事項
ムービー

rootMenu()

使用法

```
-- Lingo syntax
dvdObjRef.rootMenu()

// JavaScript syntax
dvdObjRef.rootMenu();
```

解説
DVD メソッド。ルートメニューを表示します。

パラメータ
なし

例文
次のステートメントはルートメニューを表示します。

```
-- Lingo syntax
member(1).rootMenu()

// JavaScript syntax
member(1).rootMenu();
```

関連事項
DVD

rotate

使用法

```
member(whichCastmember).node(whichNode).rotate(xAngle, yAngle, zAngle {, relativeTo})
member(whichCastmember).node(whichNode).rotate(rotationVector {, relativeTo})
member(whichCastmember).node(whichNode).rotate(position, axis, angle {, relativeTo})
transform.rotate(xAngle, yAngle, zAngle {, relativeTo})
transform.rotate(rotationVector {, relativeTo})
transform.rotate(position, axis, angle {, relativeTo})
```

解説
3D コマンド。ノードのトランスフォームオブジェクトまたは直接参照されているトランスフォームオブジェクトが保持している現在の位置、回転、スケールの各オフセットの後に回転を適用します。適用する回転は、3 つの角度として指定します。この 3 つの角度は、それぞれ 3 つの軸を中心とした回転の角度を指定します。これらの角度は、**x** の角度、**y** の角度、および **z** の角度という形式で明示的に指定することも、回転ベクトルで指定することもできます。ベクトルで指定する場合は、ベクトルの **x** コンポーネントが x 軸を中心とした回転に、**y** コンポーネントが y 軸を中心とした回転に、**z** コンポーネントが z 軸を中心とした回転に、それぞれ対応します。また、適用する回転を、スペース内のポイントを通過する任意の軸を中心とした回転として指定することもできます。

パラメータ

x の角度は、**x**、**y** および **z** 軸を指定して回転を適用する場合に必要です。**x** 軸を中心にした回転の角度を指定します。

y の角度は、**x**、**y** および **z** 軸を指定して回転を適用する場合に必要です。**y** 軸を中心にした回転の角度を指定します。

z の角度は、**x**、**y** および **z** 軸を使用して回転を適用する場合に必要です。**z** 軸を中心にした回転の角度を指定します。

回転ベクトルは、ベクトルを指定して回転を適用する場合に必要です。適用する角度を含むベクトルを指定します。

位置は、スペース内のポイントを通過する任意の軸を中心にした回転を適用する場合に必要です。スペース内のポイントを指定します。

軸は、スペース内のポイントを通過する任意の軸を中心にした回転を適用する場合に必要です。指定された位置、位置を通過する軸を指定します。

角度は、スペース内のポイントを通過する任意の軸を中心にした回転を適用する場合に必要です。軸、軸を中心にした回転量を指定します。

基準はオプションです。回転の変更を適用する際に使用する座標系の軸を指定します。基準パラメータの値を以下に示します。

- #self 。ノードのローカル座標系 (モデルの作成時に指定した X 軸、Y 軸、および Z 軸) を基準に増分が適用されます。ノード参照を指定して rotate コマンドを使用する場合に、基準パラメータを指定しないと、この値がデフォルトとして使用されます。
- #parent 。ノードの親の座標系に対して増分を適用します。トランスフォーム参照を指定して rotate コマンドを使用する場合に、基準パラメータを指定しないと、この値がデフォルトとして使用されます。
- #world 。ワールド座標系に対して増分を適用します。モデルの親がワールドである場合は、#parent を使用するのと同じになります。
- nodeReference。回転の基準にするノードを指定できます。指定したノードの座標系を基準に増分が適用されます。

例文

次の例は、" 月 " というモデルを、まずモデル自体の Z 軸を中心に回転します。これにより、" 月 " はその場で回転します。次に、親ノード (" 地球 " というモデル) を中心に " 月 " を回転します。これにより、" 月 " は " 地球 " の周りを回ります。

```
member("Scene").model("Moon").rotate(0,0,15)
member("Scene").model("Moon").rotate(vector(0, 0, 5), member("Scene").model("Moon"))
```

次の例は、" 棒 " というモデルがあるスペース内の位置を中心に " ボール " というモデルを回転させます。これにより、モデル " ボール " は、x-y 平面で " 棒 " の周りを回ります。

```
polePos = member("3d Scene").model("Pole").worldPosition
member("3d Scene").model("Ball").rotate(polePos, vector(0,0,1), 5, #world)
```

関連事項

pointAt, preRotate, rotation （トランスフォーム）, rotation (engraver shader), rotation (backdrop and overlay)preScale(), transform（プロパティ）

run

使用法

```
run(MUIObject)
```

解説

このコマンドは、MUI Xtra のインスタンスから作成された汎用目的のモーダルダイアログボックスを表示します。

Director がダイアログボックスを開くことができるように、Initialize コマンドを使用してダイアログボックスを使用します。

非モーダルダイアログボックスを開くには、WindowOperation コマンドに「#show」オプションを付けて使用します。このコマンドを使用すると、非モーダルダイアログボックスが開いている間に他の Lingo をムービーで実行できます。

例文

このハンドラは、オブジェクト MUIObject が存在するかどうかをチェックし、存在する場合は UIObject から汎用目的ダイアログボックスを表示します。

```
--Lingo syntax
on runDialog
    global MUIObject
    if objectP(MUIObject) then
        run(MUIObject)
    end if
end
```

runMode

使用法

```
_system.environmentPropList.runMode
```

解説

ムービーの再生モードを示すストリング返す関数です。使用できる値は以下のとおりです。

- Author - ムービーは、Director で再生されています。
- Projector - ムービーは、プロジェクタとして再生されています。
- BrowserPlugin - ムービーは、Shockwave Player のプラグインで再生されているか、または LiveConnect や ActiveX などの他のスクリプト環境で再生されています。

contains 演算子を使ってこのプロパティの特定の値を調べることが、一番安全な方法です。これによりエラーの発生を防ぎ、一部のみを一致させることができます。

パラメータ

なし

例文

次のステートメントは、外部パラメータが利用可能であるかどうか (Shockwave のプラグインで再生されているか) をチェックし、利用可能な場合はそれらを取り込みます。

```
--Lingo syntax
if _system.environmentPropList.runMode contains "Plugin" then
    -- decode the embed parameter
    if externalParamName(swURL) = swURL then
        put externalParamValue(swURL) into myVariable
    end if
end if

// JavaScript syntax
if (_system.environmentPropList.runMode.indexOf("Plugin") >=0) {
    // decode the embed parameter
    if (_player.externalParamName(swURL) == swURL) {
        myVariable = _player.externalParamValue(swURL);
    }
}
```

関連事項

environmentPropList, platform

save （ミクサ）

使用法

```
Mixer.save(filepath)
```

解説

ミクサメソッド。指定したローカルファイルパスにミックスされたサウンドを保存します。ファイルは WAV または MP4 形式で保存されます。

ファイルの保存中、play や pause などのミクサに対するその他の操作は失敗します。完了する前に保存操作を停止するには、mixer.stop() を使用します。

保存操作のステータスを確認するには、mixer.isSaving プロパティを使用します。

例文

```
-- Lingo
on mouseUp me
mixerRef.save("C:\audio.wav") -- Saves the mixer output at the given filepath as a WAV file.
end

// Javascript
function mouseUp(){
mixerRef.save("C:\audio.wav"); //Saves the mixer output at the given filepath as a WAV file.
}
```

関連事項

isSaving （ミクサ）, ミクサ

Save （サウンドオブジェクト）

使用法

```
soundObject.save(filePath)
```

解説

サウンドオブジェクトメソッド。サウンドストリームを、指定したファイルパスにサウンドオブジェクトとして保存します。ファイルは WAV または MP4 形式で保存されます。

ファイルの保存中、play や pause などのサウンドオブジェクトに対するその他の操作は失敗します。保存操作を停止するには、soundObject.stop() を使用します。

サウンドオブジェクトを現在保存中であるかどうかをテストするには、サウンドオブジェクトの isSaving プロパティを使用します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.save("C:\audio.wav") -- Saves the sound object associated with
-- soundObjRef at the given filepath in the .wav format.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.save("C:\audio.wav"); // Saves the sound object associated with
// soundObjRef at the given path in the .wav format.
}
```

関連事項

isSaving（サウンドオブジェクト）

save castLib

使用法

```
castLib(whichCast).save()
save castLib whichCast {,pathName&newFileName}
```

解説

コマンド。キャストの元ファイルまたは新しいファイルにキャストの変更を保存します。また、キャストへの操作や参照は、保存されたキャストメンバを使います。

このコマンドは、圧縮されたファイルでは動作しません。

save CastLib コマンドは、URL をファイル参照としてはサポートしません。

パラメータ

pathName&newFileName オプション。保存先のパスおよびファイルの名前を指定します。省略する場合、元のキャストをリンクする必要があります。

例文

次のステートメントは、修正した "Buttons" キャストを、同じフォルダ内に新たに "UpdatedButtons" という名前のファイルで保存します。

```
-- Lingo
castLib("Buttons").save(the moviePath & "UpdatedButtons.cst")

// Javascript
castLib("Buttons").save(_movie.path & "UpdatedButtons.cst") ;
```

関連事項

@ (パス名演算子)

saveMovie()

使用法

```
-- Lingo syntax
_movie.saveMovie({stringFilePath})

// JavaScript syntax
_movie.saveMovie({stringFilePath});
```

解説

ムービーメソッド。現在のムービーを保存します。

オプションのパラメータ **stringFilePath** を使用すると、指定したファイルにムービーが保存されます。このメソッドは、圧縮されたファイルでは動作しません。指定したファイル名には .dir ファイル拡張子を付ける必要があります。

saveMovie() メソッドは URL をファイル参照としてはサポートしません。

パラメータ

stringFilePath オプション。ムービーの保存先のパスとファイルの名前を指定します。

例文

次のステートメントは、現在のムービーを " 更新 " という名前のファイルに保存します。

```
-- Lingo syntax
_movie.saveMovie(_movie.path & "Update.dir")

// JavaScript syntax
_movie.saveMovie(_movie.path + "Update.dir");
```

関連事項

ムービー

scale （コマンド）

使用法

```
member(whichCastmember).node(whichNode).scale(xScale, yScale, zScale)
member(whichCastmember).node(whichNode).scale(uniformScale)
transform.scale(xScale, yScale, zScale)
transform.scale(uniformScale)
```

解説

3D トランスフォームコマンド。参照先のノードのトランスフォームまたは直接参照されているトランスフォームが保持している現在の位置、回転、スケールの各オフセットの後にスケールを適用します。スケールには、それぞれの軸に沿った 3 つのスケールを指定するか、またはすべての軸に沿って均等に適用する 1 つのスケールを指定します。個々のスケールを個別に指定する場合は、**x** のスケール、**y** のスケール、**z** のスケールの各パラメータを使用し、均等に適用するスケールを指定する場合は均等スケールパラメータを使用します。

ノードは、カメラ、グループ、ライト、またはモデルオブジェクトのいずれかです。scale コマンドを使用すると、参照先のノードの transform.scale プロパティが調整されます。ただし、ライトやカメラの場合には、ジオメトリがないため、目に見える効果はありません。

スケール値には 0 より大きい値を指定する必要があります。

パラメータ

x のスケールは、3 つのスケールを指定する場合に必要です。**x** 軸に沿ってスケールを指定します。

y のスケールは、3 つのスケールを指定する場合に必要です。**y** 軸に沿ってスケールを指定します。

z のスケールは、3 つのスケールを指定する場合に必要です。**z** 軸に沿ってスケールを指定します。

uniformScale は、1 つの均等なスケールを指定する場合に必要です。均等なスケールを指定します。

例文

次の例は、まず " 月 " というモデルの transform.scale プロパティを表示し、次に scale コマンドを使ってモデルにスケールを適用します。最後に、結果の transform.scale 値を表示します。

```
-- Lingo
put member("Scene").model("Moon").transform.scale

// Javascript
put member("Scene").getProp("model", i).transform.scale ;
-- vector( 1.0000, 1.0000, 1.0000)
```

次のステートメントは、" 冥王星 " というモデルに対して、3 つの軸に沿って均等に 0.5 のスケールを適用します。この結果、モデルは半分のサイズで表示されます。

```
-- Lingo
member("Scene").model("Pluto").scale(0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.5);
// where a is the number index of the model "Pluto"
```

次のステートメントは、Oval というモデルに対して、**x** 軸と **y** 軸に沿ったスケールは適用しないで、**z** 軸に沿ったスケールだけを適用します。この場合、上の例とは違って、スケールが不均等に適用されます。

```
-- Lingo
member("Scene").model("Pluto").scale(0.0, 0.0, 0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.0, 0.0, 0.5);
// where a is the number index of the model "Pluto"
```

関連事項

transform （プロパティ）, preScale(), scale （トランスフォーム）

script()

使用法

```
-- Lingo syntax
script(memberNameOrNum {, castNameOrNum})

// JavaScript syntax
script(memberNameOrNum {, castNameOrNum});
```

解説

最上レベルの関数。スクリプトを含む指定したキャストメンバへの参照を作成し、オプションでメンバを含むキャストライブラリを指定します。

指定したキャストメンバにスクリプトが含まれない場合、または指定したキャストメンバが存在しない場合、エラーが返されます。

パラメータ

memberNameOrNum 必須。スクリプトを含むキャストメンバの名前を指定するストリング、またはスクリプトを含むキャストメンバのインデックス位置を指定する整数です。

castNameOrNum オプション。**memberNameOrNum** メンバを含むキャストライブラリの名前を指定するストリング、または **memberName rNum** メンバを含むキャストライブラリのインデックス位置を指定する整数です。省略すると、script() が最初のキャストライブラリを検索します。

例文

Lingo の場合のみ、次のステートメントで、子オブジェクトが親スクリプト Warrior Ant のインスタンスかどうかを調べます。

```
-- Lingo syntax
if (bugObject.script = script("Warrior Ant")) then
    bugObject.attack()
end if
```

次のステートメントは、変数 actionMember をスクリプトキャストメンバの Actions に設定します。

```
-- Lingo syntax
actionMember = script("Actions")

// JavaScript syntax
var actionMember = script("Actions");
```

scrollByLine()

使用法

```
-- Lingo syntax
memberObjRef.scrollByLine(amount)

// JavaScript syntax
memberObjRef.scrollByLine(amount);
```

解説

指定したフィールドまたはテキストキャストメンバを指定した行数だけ上下にスクロールさせるためのコマンドです行とは、キャリッジリターンまたは折り返しによって定義される行を示します。

パラメータ

amount 必須。スクロールする行数を指定します。[行数] が正の数の場合、フィールドの下方向にスクロールさせます。[ページ数] が負の数の場合、フィールドは上方向にスクロールします。

例文

次のステートメントは、「今日のニュース」フィールドキャストメンバを 5 行下方向にスクロールさせます。

```
--Lingo syntax
member("Today's News").scrollbyline(5)

// JavaScript syntax
member("Today's News").scrollbyline(5);
```

次のステートメントは、" 今日のニュース " フィールドキャストメンバを 5 行上方向にスクロールさせます。

```
--Lingo syntax
member("Today's News").scrollByLine(-5)

// JavaScript syntax
member("Today's News").scrollByLine(-5);
```

scrollByPage()

使用法

```
-- Lingo syntax
memberObjRef.scrollByPage(amount)

// JavaScript syntax
memberObjRef.scrollByPage(amount);
```

解説

コマンド。指定したフィールドまたはテキストキャストメンバを指定した行数だけ上下にスクロールさせます。ページは、画面上に表示されるテキストの行数と同じです。

パラメータ

amount 必須。スクロールするページ数を指定します。[行数] が正の数の場合、フィールドの下方向にスクロールさせます。[ページ数] が負の数の場合、フィールドは上方向にスクロールします。

例文

次のステートメントは、フィールドキャストメンバ「本日のニュース」を 1 ページ下方向にスクロールします。

```
--Lingo syntax
member("Today's News").scrollbypage(1)

// JavaScript syntax
member("Today's News").scrollbypage(1);
```

次のステートメントは、" 本日のニュース " フィールドキャストメンバを 1 ページ上方向にスクロールさせます。

```
--Lingo syntax
member("Today's News").scrollbypage(-1)

// JavaScript syntax
member("Today's News").scrollbypage(-1);
```

関連事項

scrollTop

seek()

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.seek(milliseconds)

// JavaScript syntax
memberOrSpriteObjRef.seek(milliseconds);
```

解説

RealMedia スプライトまたはキャストメンバのメソッド。メディアストリームの再生位置をストリームの先頭からミリ秒で指定した位置に変更します。mediaStatus の値は、通常は #seeking になり、その後で #buffering に変わります。

このメソッドを使用すると、RealMedia ストリームの先頭以外の位置から再生を開始したり、ストリーム内で再生位置を前後にジャンプさせることができます。ミリ秒 には、ストリームの先頭からのミリ秒数を整数で指定します。このため、後ろにジャンプさせるときは、負の数ではなく、現在よりも小さな値を指定します。

mediaStatus が #paused のときに seek コマンドが呼び出されると、ストリームは再バッファされ、seek で指定した新しい位置で #paused の状態に戻ります。mediaStatus が #playing のときに seek コマンドが呼び出されると、ストリームは再バッファされ、ストリーム内の新しい位置から自動的に再生が開始されます。mediaStatus が #closed のときに seek が呼び出されたときは、何も行われません。

ストリームの duration 値を超える位置へジャンプしようとすると、指定した整数値は 0 からストリームの長さの値の範囲内に短縮されます。ライブコンテンツを再生している RealMedia スプライトへはジャンプできません。

x.seek(n) ステートメントは x.currentTime = n と同じ機能で、どちらを呼び出してもストリームは再バッファされます。

パラメータ

milliseconds 必須。ストリームの先頭からのミリ秒数を指定する整数です。

例文

次の例は、現在の再生位置を 10,000 ミリ秒 (10 秒) の位置に設定します。

```
-- Lingo syntax
sprite(2).seek(10000)
member("Real").seek(10000)

// JavaScript syntax
sprite(2).seek(10000);
member("Real").seek(10000);
```

関連事項

duration (RealMedia, SWA), currentTime (RealMedia), play() (RealMedia, SWA, Windows Media), pause() (RealMedia, SWA, Windows Media), stop() (RealMedia, SWA, Windows Media), mediaStatus (RealMedia, Windows Media)

seek(mSec) (MP4Media/FLV)

使用法

```
sprite(1).seek(3000) -- Seeks to the third second in the video.
member(1).seek(3000) -- Seeks to the third second of the video.
```

解説

MP4Media/FLV メンバまたはスプライトメソッド。ビデオ内の特定の位置をシークします。シークする位置はミリ秒で指定します。

パラメータ

パラメータ	解説
mSec	必須。このパラメータは、MP4 ビデオのストリームの開始からのミリ秒数でシーク位置を指定する整数値です。

例文

次の例は、現在の再生位置を 10,000 ミリ秒 (10 秒) の位置に設定します。

```
-- Lingo syntax
sprite(2).seek(10000)
member("MP4Media/FLV").seek(10000)

// JavaScript syntax
sprite(2).seek(10000);
member("MP4Media/FLV").seek(10000);
```

seek（サウンドオブジェクト）

使用法

```
so.seek(<milliseconds>)
```

解説

サウンドオブジェクトメソッド。オーディオ内の指定位置（ミリ秒単位）をシークします。

例文

```
-- Lingo syntax
on mouseUp me
soundObjectRef.seek = 7000  --  Seeks the seventh second within the audio.
end

// JavaScript syntax
function mouseUp(){
soundobjectref.seek= 7000  // Seeks the seventh second within the audio.
}
```

selectAtLoc()

使用法

```
-- Lingo syntax
dvdObjRef.selectAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.selectAtLoc(point(x, y));
```

解説

DVD メソッド。指定したポイントの下にあるボタンにフォーカスを移動します。

このメソッドにはボタン上に置いたマウスと同じ機能があります。

パラメータ

point(x, y) 必須。フォーカスをボタンとして指定した場所を指定するステージの座標のポイントです。

例文

次のステートメントは指定したポイントの下にあるボタンにフォーカスを移動します。

```
-- Lingo syntax
member(10).selectAtLoc(point(50, 75))

// JavaScript syntax
member(10).selectAtLoc(point(50, 75));
```

関連事項

DVD

selectButton()

使用法

```
-- Lingo syntax
dvdObjRef.selectButton(intButton)

// JavaScript syntax
dvdObjRef.selectButton(intButton);
```

解説

DVD メソッド。指定したボタンを選択します。

このメソッドは、成功すると 0 を返します。

注意：このメソッドは、Mac®-Intel® ではサポートされていません。

パラメータ

intButton 必須。指定したフォーカスであるボタンを指定する整数です。

例文

次のステートメントは、ボタン 5 を選択します。

```
-- Lingo syntax
sprite(11).selectButton(5)

// JavaScript syntax
sprite(11).selectButton(5);
```

関連事項

DVD

selectButtonRelative()

使用法

```
-- Lingo syntax
dvdObjRef.selectButtonRelative(direction)

// JavaScript syntax
dvdObjRef.selectButtonRelative(direction);
```

解説

DVD メソッド。メニュー内の現在のボタンの位置を基準にボタンを選択します。

パラメータ

direction 必須。現在のボタンの位置から移動する方向を指定するシンボル（Lingo）またはストリング（JavaScript シンタックス) です。有効な値は、left または right です。

注意：このメソッドは、Mac®-Intel® ではサポートされていません。

例文

次のステートメントは、現在のボタンの左側にボタンを指定します。

```
-- Lingo syntax
member(12).member.selectButtonRelative(#left)

// JavaScript syntax
member(12).member.selectButtonRelative("left");
```

関連事項

DVD

selection() （関数）

使用法

```
the selection
```

解説

現在の編集可能なフィールドでハイライト表示された部分を含むストリングを返す関数です。この関数は、ユーザがフィールドで選択した内容を調べるときに役に立ちます。

selection 関数は、どの文字のストリングが選択されているかを示すのみです。つまり、selection を使って、文字のストリングを選択することはできません。

パラメータ

なし

例文

次のステートメントは文字が選択されているかどうかを調べます。何も選択されていない場合は、「語句を選択してください。」という警告を表示します。：

```
if the selection = EMPTY then alert "Please select a word."
```

関連事項
selStart, selEnd

sendAllSprites()

使用法

```
-- Lingo syntax
_movie.sendAllSprites(stringEventMessage {, args})

// JavaScript syntax
_movie.sendAllSprites(stringEventMessage {, args});
```

解説
ムービーメソッド。イベントに含まれるスプライトのみでなく、すべてのスプライトに対してメッセージを送ります。stopEvent() メソッドが使用されていない限り、他のメッセージと同じく、このメッセージは現在のスプライトに割り当てられているすべてのスクリプトに伝達されます。

最善の結果を得るためには sendSprite() メソッドを使用してメッセージを適切に処理するスプライトにのみメッセージを送ります。すべてのスクリプトにメッセージが送られた場合、エラーは発生しませんが、パフォーマンスに影響があります。異なるスプライトが 1 つのビヘイビアー内で同じハンドルを持つと問題が発生することがあるので、競合を避けるために、全体に送信されるメッセージには一意の名前を使用します。

メッセージがすべてのビヘイビアーに送られた後で、イベントは通常のメッセージ階層に従い、キャストメンバスクリプト、フレームスクリプト、ムービースクリプトに送られます。

sendAllSprites() メソッドを使用するときは、次のように行います。

- **stringEventMessage** にメッセージを指定します。
- [パラメータ] でメッセージと共に送られるパラメータを指定します。

スプライトに指定されたハンドラを含むビヘイビアーがない場合は、sendAllSprites() は FALSE を返します。

パラメータ
stringEventMessage 必須。すべてのスクリプトに送信するメッセージを指定するストリングです。

args オプション。メッセージと共に送信するパラメータです。

例文
次のハンドラは、ユーザがマウスをクリックしたときに allSpritesShouldBumpCounter カスタムメッセージおよびパラメータ 2 をすべてのスプライトに送ります。

```
-- Lingo syntax
on mouseDown me
    _movie.sendAllSprites(#allspritesShouldBumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendAllSprites("allspritesShouldBumpCounter", 2);
}
```

関連事項
ムービー , sendSprite(), stopEvent()

sendEvent

使用法

```
member(whichCastmember).sendEvent(#eventName, arg1, arg2,...)
```

解説

3D コマンド。イベントと任意の数のパラメータを、そのイベントを受信するよう登録されたすべてのスクリプトに送信します。イベントにスクリプトを登録するには、registerForEvent() または setCollisionCallback() を使用します。

パラメータ

eventName 必須。送信するイベントの名前を指定します。

arg1, arg2, ... 必須。**eventName** イベントで送信するパラメータです。

例文

次の例の 1 行目は、"tester" という親スクリプトのインスタンスを作成します。2 行目は、スクリプトインスタンスのハンドラ jumpPluto を、#jump イベントが送信されたときに呼び出されるハンドラとして設定します。3 行目は、jumpMars というムービースクリプトハンドラを、#jump イベントが送信されたときに呼び出されるもう 1 つのハンドラとして登録します。4 行目は、#jump イベントを送信します。これにより、#jumpMars ムービースクリプトハンドラと #jumpPluto ハンドラが、#jump イベントに登録されているその他のハンドラと共に呼び出されます。この例の 3 行目のようにスクリプトインスタンスの値を 0 にすると、ビヘイビアーインスタンスのハンドラや親スクリプトの子ではなく、ムービースクリプトのハンドラが登録されます。

```
t = new (script "tester")
member("scene").registerForEvent(#jump, #jumpPluto, t)
member("scene").registerForEvent(#jump, #jumpMars, 0)
member("scene").sendEvent(#jump)
```

関連事項

registerScript(), registerForEvent(), setCollisionCallback()

sendSprite()

使用法

```
-- Lingo syntax
_movie.sendSprite(spriteNameOrNum, event {, args})

// JavaScript syntax
_movie.sendSprite(spriteNameOrNum, event {, args});
```

解説

ムービーメソッド。指定したスプライトに添付されたすべてのスクリプトにメッセージを送るコマンドです。

sendSprite() を使用して送られたメッセージは、スプライトに添付された各スクリプトに伝達されます。次にメッセージは、通常のメッセージ伝達順序に従って、キャストメンバスクリプト、フレームスクリプト、ムービースクリプトに送られます。

指定スプライトに指定されたハンドラを含むビヘイビアーがない場合は、sendSprite() は FALSE を返します。

パラメータ

spriteNameOrNum 必須。イベントを受信するスプライトの名前または数を指定するストリングまたは整数です。

event 必須。指定したスプライトに送信するイベントを指定するシンボルまたはストリングです。

args オプション。メッセージと共に送信するパラメータです。

例文

次のハンドラは、ユーザがマウスをクリックしたときに bumpCounter カスタムメッセージおよびパラメータ 2 をスプライト 1 に送ります。

```
-- Lingo syntax
on mouseDown me
    _movie.sendSprite(1, #bumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendSprite(1, "bumpCounter", 2);
}
```

関連事項

ムービー

setAlpha()

使用法

```
imageObject.setAlpha(alphaLevel)
imageObject.setAlpha(alphaImageObject)
```

解説

画像オブジェクトのアルファチャネルをフラット alphaLevel または既存のアルファ画像オブジェクトに設定する関数です。アルファレベルは、0 ～ 255 の数値であることが必要です。数値が小さいほど画像は透明になります。数値を大きくすると不透明になります。255 は 0 と同じ効果になります。**alphaLevel** の効果を有効にするには、画像オブジェクトの useAlpha() を TRUE に設定する必要があります。

画像オブジェクトは、32 ビットに設定します。アルファオブジェクトを指定する場合は、8 ビットに設定します。どちらも同じ画像サイズに設定します。この設定をしないと、setAlpha() の効果はなく、FALSE が返されます。成功した場合は TRUE が返されます。

例文

次の Lingo ステートメントは、ビットマップキャストメンバ "Foreground" の画像を不透明にして、そのアルファチャネルも無効にします。これは画像からアルファレイヤーを削除するには便利な方法です。

```
member("Foreground").image.setAlpha(255)
member("Foreground").image.useAlpha = FALSE
```

次の Lingo は、キャストメンバ " 日の出 " からアルファレイヤーを取得し、キャストメンバ " 日の入り " のアルファレイヤーへ配置します。

```
tempAlpha = member("Sunrise").image.extractAlpha()
member("Sunset").image.setAlpha(tempAlpha)
```

関連事項

useAlpha, extractAlpha()

setaProp

使用法

```
setaProp list, listProperty, newValue
setaProp (childObject, listProperty, newValue)
list.listProperty = newValue
list[listProperty] = newValue
childObject.listProperty = newValue
```

解説

[リストプロパティ] に割り当てられている値を、[新規値] で指定した値と置き換えるためのコマンドです。setaProp コマンドはプロパティリストおよび子オブジェクトとともに動作します。リニアリストを含む setaProp コマンドを使うと、スクリプトエラーが発生します。

- プロパティリストでは、setaProp を使って、[リスト] で指定したリストのプロパティを置き換えます。リスト内にプロパティが存在しない場合、Lingo は新しいプロパティと値を追加します。
- 子オブジェクトでは、setaProp を使って、子オブジェクトのプロパティを置き換えます。オブジェクト内にプロパティが存在しない場合、Lingo は新しいプロパティと値を追加します。
- setaProp コマンドは、ancestor プロパティを設定することもできます。

パラメータ

listProperty 必須。値が変化するプロパティの名前を指定するシンボル（Lingo のみ）またはストリングです。

newValue 必須。**listProperty** プロパティの変数の新しい値です。

例文

次のステートメントは新たにプロパティリストを作成し、そのリストに [#c:10] という項目を追加します。

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
setaProp newList, #c, 10
put newList
```

ドット演算子を使えば、既にリストにあるプロパティの値を setaProp を使用しないで変えることもできます。

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
newList.b = 99
put newList
-- [#a:1, #b:99]
```

注意：プロパティを変えるためにドット演算子を使う際には、プロパティが既にリスト、子オブジェクト、またはビヘイビアーの中に含まれている必要があります。

関連事項

ancestor, property, .（ドット演算子）（どっとえんざんし）

setAt

使用法

```
setAt list, orderNumber, value
list[orderNumber] = value
```

解説

[リスト] に指定したリスト内にある項目の中で、[順番] に指定した項目を、[値] に置き換えるためのコマンドです。[順番] の値がプロパティリストの項目数より大きい場合、setAt コマンドはスクリプトエラーを返します。[順番] の値がリニアリスト内の項目数より大きい場合、Director はリストの空白エントリを増やし、[順番] で指定した値に相当する番号を返します。

例文

次のハンドラは、[12, 34, 6, 7, 45] というリストに名前を与え、4 番目の値を 10 に置き換えてから結果をメッセージウィンドウに表示します。

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    setAt vnumbers, 4, 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers =list(12, 34, 6, 7, 45) ;
    vNumbers.setAt(4, 10) ;
    put (vNumbers);
}
```

このハンドラが実行されると、メッセージウィンドウには次のように表示されます。

```
[12, 34, 6, 10, 45]
```

次のようにブラケットを使用してリストへアクセスしても、同様の操作を実行できます。

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    vnumbers[4] = 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers = list(12, 34, 6, 7, 45);
    vnumbers[4] = 10 ;
    put (vNumbers);
}
```

このハンドラが実行されると、メッセージウィンドウには次のように表示されます。

```
[12, 34, 6, 10, 45]
```

関連事項

[ ] (ブラケットアクセス)

setCallback()

使用法

```
-- Lingo syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName, lingoScriptObject)

// JavaScript syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName, lingoScriptObject);
```

解説

スプライトまたはグローバルメソッドとして使用し、指定したオブジェクトによって生成された特定のイベントに対する Lingo のコールバックハンドラを定義する、Flash コマンドです。ActionScript のオブジェクトでイベントが発生すると、このイベントは指定された Lingo ハンドラへリダイレクトされます。このとき、パラメータもイベントと共に渡されます。

ActionScript オブジェクトが元々 Flash スプライト内で作成されていた場合は、**flashSpriteReference** というシンタックスを使用します。オブジェクトが元々グローバルとして作成されていた場合は、グローバルシンタックスを使用します。

注意：Flash キャストメンバを読み込んでいない場合は、グローバル Flash コマンドを正しく機能させるために、Flash Asset Xtra をムービーの Xtra リストに手動で追加する必要があります。Xtra を Xtra リストに追加するには、修正／ムービー／エクストラ を選択します。配布するムービーの Xtra 拡張機能の管理の詳細については、ヘルプパネルで「Director ユーザガイド」のトピックを参照してください。

パラメータ

actionScriptObject 必須。**ASEventName** イベントを含む ActionScript オブジェクトを指定します。

ASEventName 必須。発生する ActionScript イベントを指定します。

lingoHandlerName 必須。**ASEventName** イベントを扱う Lingo ハンドラを指定します。

lingoScriptObject 必須。**lingoHandlerName** ハンドラを含む Lingo スクリプトオブジェクトを指定します。

例文

次のステートメントは、Lingo スクリプトオブジェクト me 内で myOnStatus という名前の Lingo ハンドラを設定します。このハンドラは、スプライト 3 の Flash ムービーで、ActionScript オブジェクト tLocalConObject によって onStatus イベントが生成されたときに、呼び出されます。

```
Lingo syntax
sprite(3).setCallback(tLocalConObject, "onStatus", #myOnStatus, me)

// JavaScript syntax
sprite(3).setCallback(tLocalConObject, "onStatus", symbol("myOnStatus"), me);
```

次のステートメントは新規のグローバル XML オブジェクトを作成し、XML データを受け取ったときにこれを解析するコールバックハンドラを作成します。3 行目で XML ファイルをロードします。コールバックハンドラも含まれています。

```
-- Lingo syntax
gXMLCB = newObject("XML")
setCallback( gXMLCB, "onData", #dataFound, 0 )
gXMLCB.load( "myfile.xml")

-- Callback handler invoked when xml data arrives
on dataFound me, obj, source
    obj.parseXML(source)
    obj.loaded = 1
    obj.onload(TRUE)
end dataFound

// JavaScript syntax
gXMLCB = newObject("XML");
setCallback( gXMLCB, "onData", symbol("dataFound"), 0 );
gXMLCB.load( "myfile.xml" );

// Callback handler invoked when xml data arrives
function dataFound(me, obj, source) {
    obj.parseXML(source);
    obj.loaded = 1;
    obj.onload(1);
}
```

関連事項

newObject(), clearAsObjects()

setCharSet

使用法

```
FileIO.setCharSet(strCharSet)
```

解説

FileIO Xtra メソッド。FileIO がすべてのストリング読み書き関数に指定した文字セットを使用するようにします。

パラメータ

パラメータ	解説	デフォルト値
strCharSet	必要なエンコーディング変換を指定します。	UTF-8

例文

```
--Lingo syntax
FileIO.setCharSet("UTF-16")

//JavaScript syntax
FileIO.setCharSet("UTF-16");
```

関連事項

readChar(), readFile(), readLine(), readToken(), readWord(), writeChar(), writeString()

setCollisionCallback()

使用法

```
member(whichCastmember).model(whichModel).collision.setCollisionCallback (#handlerName, scriptInstance)
```

解説

3D 衝突コマンド。**whichModel** が衝突に巻き込まれたときに、指定したスクリプトインスタンス内で指定したハンドラが呼び出されるように登録します。

このコマンドが機能するのは、モデルの collision.enabled プロパティが TRUE に設定されている場合だけです。デフォルトのビヘイビアーは、collision.resolve の値によって決められます。デフォルトのビヘイビアーを上書きするには、collision.resolveA コマンドまたは collision.resolveB コマンドを使用します。指定したハンドラで updateStage コマンドを使用しないでください。

このコマンドを使用すると、衝突に対して registerScript コマンドを使用した場合と同じ結果を、より簡単に得ることができます。つまり、このコマンドは、registerScript コマンドの一部の機能を実行するコマンドであると言えます。

パラメータ

handlerName 必須。モデルが衝突に巻き込まれたときに、呼び出すハンドラを指定します。

scriptInstance 必須。**handlerName** によって指定されるハンドラを含むスクリプトインスタンスを指定します。

例文

次のステートメントは、キャストメンバ "colScript" の #bounce ハンドラを、"Sphere" というモデルが別のモデルと衝突したときに呼び出されるように設定します。

```
member("3d world").model("Sphere").collision.setCollisionCallback(#bounce, member("colScript"))
```

関連事項

collisionData, collision (modifier), resolve, resolveA, resolveB, registerForEvent(), registerScript(), sendEvent

setFilterMask()

使用法

```
-- Lingo syntax
fileioObjRef.setFilterMask(stringMask)

// JavaScript syntax
fileioObjRef.setFilterMask(stringMask);
```

解説

FileIO メソッド。ダイアログボックスの [ファイルの種類] フィールドのフィルタマスクを設定して、ダイアログボックスを開くときに表示されるファイルの種類を指定します。

パラメータ

stringMask 必須。フィルタマスクを指定するストリングです。

例文

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFilterMask(stringMask)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFilterMask(stringMask) ;
```

関連事項

Fileio

setFinderInfo()

使用法

```
-- Lingo syntax
fileioObjRef.setFinderInfo(stringAttrs)

// JavaScript syntax
fileioObjRef.setFinderInfo(stringAttrs)
```

解説

FileIO メソッド（Mac のみ）。開いたファイルの Finder 情報を設定します。

パラメータ

stringAttrs 必須。Finder 情報を指定するストリングです。

例文

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFinderInfo(stringAttrs)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFinderInfo(stringAttrs) ;
```

関連事項

Fileio

setFlashProperty()

使用法

```
-- Lingo syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue)

// JavaScript syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue);
```

解説

指定された Flash スプライトに対して、Flash アクションスクリプト関数 setProperty() を呼び出す関数です。setFlashProperty() 関数を使用すると、ムービークリップや Flash ムービー内のレベルのプロパティを設定できます。これは、Director にスプライトプロパティを設定する方法に似ています。

Flash スプライトのプロパティをグローバルに設定するには、空のストリングをターゲット名として送信します。グローバル Flash プロパティは、#focusRect と #spriteSoundBufferTime です。

これらのプロパティの詳細については、Flash のマニュアルを参照してください。

パラメータ

targetName 必須。プロパティを Flash スプライト内に設定するムービークリップまたはレベルの名前を指定します。

property 必須。設定するプロパティの名前を指定します。設定できるムービークリップパラメータは、#posX、#posY、#scaleX、#scaleY、#visible、#rotate、#alpha、#name です。

newValue 必須。新しい値を指定します。

例文

次のステートメントは、スプライト 3 の Flash メンバのムービークリップ "Star" の #rotate プロパティの値を 180 に設定します。

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate, 180)

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"), 180);
```

関連事項

getFlashProperty()

setNewLineConversion()

使用法

```
-- Lingo syntax
fileioObjRef.setNewLineConversion(intOnOff)

// JavaScript syntax
fileioObjRef.setNewLineConversion(intOnOff)
```

解説

FileIO メソッド（Mac のみ）。新しい行の文字の自動変換をオンまたはオフに指定します。このメソッドは、UNIX 形式の行末を持つファイルを読み取る際には有用ですが、Windows および Macintosh クラシック形式の行末を持つファイルを読み取る際には必要ありません。

パラメータ

intOnOff 必須。自動変換をオンまたはオフに指定する整数です。有効な値は、0 （オフ）または 25 （オン）です。デフォルト値は 0（オフ）です。

例文

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setNewLineConversion(intOnOff)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setNewLineConversion(intOnOff) ;
```

関連事項

Fileio

setPixel()

使用法

```
-- Lingo syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue)
imageObjRef.setPixel(point(x, y), colorObjOrIntValue)

// JavaScript syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue);
imageObjRef.setPixel(point(x, y), colorObjOrIntValue);
```

解説

画像メソッド。指定した画像の指定個所のピクセルのカラー値を設定します。

getPixel() を使用して、他のピクセルのカラーを大量のピクセルに設定する場合は、整数値として設定する方が速く処理できます。

カラーオブジェクトの最高性能を得るには、8 ビット以下の画像の場合はインデックスカラーオブジェクトを、16 ビット以上の画像の場合は RGB カラーオブジェクトを使用します。

指定したピクセルが指定画像の外側にある場合は、以下のメソッドは FALSE を返します。

完成したムービーのこのメソッドの実例は、Director アプリケーションフォルダの "Learning/Lingo" フォルダ内の "imaging.dir" を参照してください。

パラメータ

xx 座標と y 座標を使用してピクセルを指定する場合は必須。ピクセルの x 座標を指定する整数。

yx 座標と y 座標を使用してピクセルを指定する場合は必須。ピクセルの y 座標を指定する整数。

point**(x, y)** 点を使用してピクセルを指定する場合は必須。ピクセルを指定するポイントです。

colorObjOrIntValue は、カラーオブジェクトのカラーまたは整数の値を設定する場合に必要です。ピクセルのカラーを指定するカラーオブジェクトの参照、またはピクセルのカラー値を指定する整数です。

例文

次のステートメントは、メンバの画像ステージ内にあるポイント (20, 20) のピクセルカラーを赤に設定します。

```
-- Lingo
objImage = _movie.stage.image
objImage.setPixel(20, 20 , rgb(255,0,0))
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.setPixel(20, 20 , color(255,0,0)) ;
put (objImage) ;
```

関連事項

color(), draw(), fill(), getPixel(), image()

setPixels()

使用法

```
Image.setPixels(bytearray, #symbolImageFormat)
Possible values form symImageFormat are:
#bgra8888 => 32 bit BGRA
#argb8888=> 32 bit ARGB
#rgba8888=> 32 bit RGBA
#rgb888=> 24bit RGB
#bgr888=> 24bit BGR
```

解説

バイト配列メソッド。バイト配列の内容をコピーして画像データにコピーします。

注意：32 ビットの Lingo 画像オブジェクトのみがサポートされています。

例文

次のコードスニペットは、画像を赤色で塗りつぶし、アルファ値を 50 に設定します。

```
--Lingo syntax
on mouseUp me
i32=image(128,128,32) -- Creating a 32 bit image
i32.useAlpha=1 -- Setting the Alpha to 1
ba=i32.getPixels(#RGBA8888) -- Getting the pixels of the image using 32 bit RGBA format
ba2=byteArray(128*128*4,0) -- Creating the bytearray
i=0
repeat with r=1 to 128
repeat with c=1 to 128
ba2[i+1]=255 -- Value for RED
ba2[i+2]=0 -- Value for GREEN
ba2[i+3]=0 -- Value for BLUE
ba2[i+4]=50 -- Value for Alpha
i=i+4
end repeat
end repeat
i32.setPixels(ba2,#RGBA8888) -- Setting the pixels of the image using 32 bit RGBA format
member("test").image=i32
end
//JavaScript Syntax
function mouseUp(me)
{
```

```
var i=0;
i32=image(128,128,32);
i32.useAlpha=1;
ba=i32.getPixels(symbol("RGBA8888"));
ba2=byteArray(128*128*4,0);
i=0;
for(r=1;r<=128;r++)
{
for(c=1;c<=128;c++)
{
ba2[i+1]=255;
ba2[i+2]=0;
ba2[i+3]=0;
ba2[i+4]=50;
i=i+4;
}
}
i32.setPixels(ba2,symbol("RGBA8888"));
member("test").image=i32;
}
```

関連事項

getPixels()

setPlayList()

使用法

```
-- Lingo syntax
soundChannelObjRef.setPlayList(linearListOfPropLists)

// JavaScript syntax
soundChannelObjRef.setPlayList(linearListOfPropLists);
```

解説

サウンドチャネルメソッド。サウンドチャネルのプレイリストを設定またはリセットします。

このメソッドは、複数のサウンドを一度にキューに登録する場合に有効です。

完成したムービーで使用する setPlaylist() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内のサウンドコントロールムービーを参照してください。

パラメータ

linearListOfPropLists 必須。プレイリストのパラメータを指定するプロパティリストのリニアリストです。キューに登録する各サウンドに対し、次のプロパティ指定できます。

プロパティ	解説
#member	キューに登録するサウンドキャストメンバ。このプロパティは指定する必要があります。これ以外はすべてオプションです。
#startTime	再生を開始するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの先頭です。「startTime」を参照してください。
#endTime	再生を終了するサウンド内の時間 (単位はミリ秒)。デフォルトはサウンドの最後です。「endTime」を参照してください。

プロパティ	解説
#loopCount	#loopStartTime と #loopEndTime で定義されたループを再生する回数。デフォルトは 1 です。「loopCount」を参照してください。
#loopStartTime	ループを開始するサウンド内の時間 (単位はミリ秒)。「loopStartTime」を参照してください。
#loopEndTime	ループを終了するサウンド内の時間 (単位はミリ秒)。「loopEndTime」を参照してください。
#preloadTime	再生前にバッファに溜めておくサウンドの量 (単位はミリ秒)。「preloadTime」を参照してください。

例文

次のハンドラは、キャストメンバの "introMusic" をキューに登録しサウンドチャネル 2 で、3 秒目の位置から再生を開始します。次に、8 ～ 8.9 秒の部分を 5 回繰り返して再生し、10 秒目の位置で停止します。

```
-- Lingo syntax
on play Music
sound(2).setPlayList([]) -- empty the playlist sound(2).queue([#member:member("introMusic"),
#startTime:3000, #endTime:10000, #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
sound(2).setPlayList(propList()); sound(2).queue(propList("member",member("introMusic"),"startTime",3000,
"endTime",10000, "loopCount",5, "loopStartTime",8000,"loopEndTime",8900)); sound(2).play();
}
```

関連事項

endTime （サウンドチャネル）, getPlayList(), loopCount, loopEndTime （サウンドチャネル）, loopStartTime, メンバ , member, preLoadTime, queue(), サウンドチャネル , startTime (サウンドチャネル)

setPosition()

使用法

```
-- Lingo syntax
fileioObjRef.setPosition(intPosition)

// JavaScript syntax
fileioObjRef.setPosition(intPosition);
```

解説

FileIO メソッド。ファイルの位置を設定します。

パラメータ

intPosition 必須。ファイルの新しい位置を指定する整数です。

例文

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setPosition(intPosition)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setPosition(intPosition) ;
```

関連事項

Fileio

setPref()

使用法

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

解説

プレーヤメソッド。**prefString** で指定したストリングを、コンピュータのローカルディスクの **stringPrefName** で指定したファイルに書き込みます。

stringPrefName は有効なファイル名でなければなりません。ファイル名をすべてのプラットフォーム上で確実に有効にするために、ファイル名はアルファベット 8 文字以内にしてください。

ブラウザで再生されているムービーから setPref() メソッドが実行された場合は、"Prefs" フォルダがプラグインのサポートフォルダ内に作成されます。setPref() メソッドは、そのフォルダにのみ書き込みできます。

プロジェクタまたは Director で再生された場合、アプリケーションと同じ階層に、新たにフォルダが作成されます。このフォルダには "**Prefs**" という名前が付与されます。

このメソッドは、読み取り専用のメディアへの書き込みには使用しないでください。プラットフォームやシステムのバージョンによって、エラーまたはその他の問題が発生することがあるからです。

このメソッドはストリングデータの複雑な処理やそのフォーマット化は行いません。フォーマットその他の処理をする場合は getPref() を一緒に使用すると、setPref() を通してメモリ内で処理され、古いファイルに上書きされます。

ブラウザでは setPref() で記述されたデータはプライベートではありません。Shockwave コンテンツを含む任意のムービーからこの情報を読み取り、サーバにアップロードできます。極秘の情報は setPref() を使用して保存しないようにしてください。

Windows では、制限されているユーザの場合は、setPref() メソッドは失敗します。

完成したムービーでの setPref() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の read and write text.dir を参照してください。

パラメータ

stringPrefName 必須。**prefString** ストリングの書き込み先のファイルの名前を指定します。このファイルは標準のテキストファイルです。

prefString 必須。**stringPrefName** によって指定されるファイルに書き込むストリングです。

例文

次のハンドラは "Text Entry" フィールドキャストメンバのテキストを "DayWare" という名前のファイルに保存します。

```
-- Lingo syntax
on mouseUp me
    _player.setPref("CurPrefs", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("CurPrefs", member("Text Entry").text);
}
```

関連事項

getPref()（プレーヤ）, プレーヤ

setPref()（プレーヤ）

使用法

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

解説

プレーヤのメソッド。指定したストリングをコンピュータのローカルディスク上で指定したファイルに書き込みます。このファイルは標準のテキストファイルです。

ブラウザでムービーが再生されているときに setPref() コマンドが実行された場合は、"Prefs" フォルダがプラグインのサポートフォルダ内に作成されます。setPref() メソッドは、そのフォルダにのみ書き込みできます。

ファイルが作成される場所は次のとおりです。

Windows Projector/Director ..\Documents and Settings\< ユーザ名 >\Application Data\Adobe\Director< バージョン番号 >\Prefs\

Windows Shockwave ..\Documents and Settings\< ユーザ名 >\Application Data\Adobe\Shockwave Player < バージョン番号 >\Prefs\

MAC Projector ~/Library/Application Support/Adobe/Director < バージョン番号 >/Prefs

MAC Shockwave ~/Library/Application Support/Adobe/Shockwave Player< バージョン番号 >/Prefs

このメソッドは、読み取り専用のメディアへの書き込みには使用しないでください。プラットフォームやシステムのバージョンによって、エラーまたはその他の問題が発生することがあるからです。

ブラウザでは setPref() で記述されたデータはプライベートではありません。Shockwave コンテンツを含む任意のムービーからこの情報を読み取り、サーバにアップロードできます。極秘の情報は setPref() を使用して保存しないようにしてください。

Windows では、制限されているユーザの場合は、setPref() は失敗します。

完成したムービーでの setPref() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の read and write text.dir を参照してください。

パラメータ

prefName 必須。書き込み先のファイルを指定するストリングです。**prefName** パラメータは有効なファイル名でなければなりません。ファイル名をすべてのプラットフォーム上で確実に有効にするために、ファイル名はアルファベット 8 文字以内にしてください。

prefValue 必須。**prefName** ファイルに書き込むテキストを指定するストリングです。

例文

次のハンドラは "Text Entry" フィールドキャストメンバのテキストを "DayWare" という名前のファイルに保存します。

```
-- Lingo syntax
on mouseUp me
    _player.setPref("DayWare", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("DawWare", member("Text Entry").text);
}
```

関連事項

getPref()（プレーヤ）, プレーヤ

setProp

使用法

```
setProp list, property, newValue
list.listProperty = newValue
list[listProperty] = newValue
```

解説

コマンド。リストで、指定したプロパティに割り当てた値を新しい値に置き換えます。プロパティがリストに含まれていない場合、setProp はスクリプトエラーを返します。

setProp コマンドはプロパティリストでのみ利用できます。setProp をリニアリストで使うと、スクリプトエラーになります。

このコマンドは setaProp コマンドと似ていますが、setProp はリストにプロパティが存在しないときはエラーになる点が異なります。

パラメータ

property 必須。値が **newValue** で置き換えられるプロパティを指定するシンボル（Lingo のみ）またはストリングです。

newValue 必須。**property** によって指定されるプロパティの変数の新しい値です。

例文

次のステートメントは、プロパティリスト x の age プロパティに割り当てられている値を 11 に変更します。

```
--Lingo
setProp x, #age, 11

// Javascript
x[age] = 11
```

ドット演算子を使うと、既にリストにあるプロパティの値を同様に変えることができます。

```
x.age = 11
```

関連事項

setaProp

setScriptList()

使用法

```
spriteReference.setScriptList(scriptList)
sprite(whichSprite).setScriptList(scriptList)
```

解説

スプライトの scriptList を設定するコマンドです。scriptList はスプライトにどのスクリプトが含まれており、各スクリプトプロパティの設定がどうなっているのかを示します。このリストを設定することで、スプライトに含まれるビヘイビアーを変更したり、ビヘイビアープロパティの変更ができます。

リストの形式は次のとおりです。

```
[ [ (whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] ",
[(whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] " ] ]
```

このコマンドは、スコア生成セッションの間は使用できません。スコア生成セッション中に追加されたスプライトについては、スコア生成セッションの終了後に setScriptList() を使用します。

パラメータ

scriptList 必須。指定したスプライトのスクリプトを指定します。

関連事項

scriptList, value(), string()

settingsPanel()

使用法

```
-- Lingo syntax
spriteObjRef.settingsPanel({integerPanelIndex})

// JavaScript syntax
spriteObjRef.settingsPanel({integerPanelIndex});
```

解説

指定されたパネルインデックスで Flash プレーヤの設定パネルを開く、Flash スプライトコマンドです。これは、ブラウザ内で再生されている Flash ムービーを右クリック（Windows）または Ctrl+ クリック（Mac）したときに開かれるのと同じダイアログボックスです。

Flash スプライトの矩形が、設定パネルを収めるだけの十分なサイズがないときは、設定パネルは表示されません。

ユーザが右クリック（Windows）または Ctrl+ クリック（Mac）したときに、設定パネルを開くことによって Flash プレーヤをエミュレートする場合は、mouseDown ハンドラでこのコマンドを使用し、rightMouseDown プロパティや controlDown プロパティを調べます。

ブラウザで再生されている Director ムービーで、設定パネルを有効にすることで Flash プレーヤをエミュレートする場合は、最初に、Shockwave プレーヤのコンテキストメニューを最初に無効にする必要があります。このコンテキストメニューは、ブラウザ内で再生されている Shockwave コンテンツを含むムービーを右クリック（Windows）または Ctrl+ クリック（Mac）すると表示されます。このメニューを無効にする方法の詳細については、Director ヘルプパネルの「Director の使い方」トピックを参照してください。

パラメータ

integerPanelIndex オプション。ダイアログボックスを開くときに、どのパネルをアクティブにするのかを指定します。有効な値は、0、1、2 または 3 です。値が 0 の場合は [プライバシー] タブが表示されるダイアログボックスが開かれ、値が 1 の場合は [ローカルストレージ] タブが表示されます。また、値が 2 の場合は [マイク] タブが表示され、値が 3 の場合は [カメラ] タブが表示されます。デフォルト値は 0 です。

例文

次のステートメントは、Flash プレーヤの設定パネルを表示し、[ローカルストレージ] タブをアクティブにします。

```
-- Lingo syntax
sprite(3).settingsPanel(1)

// JavaScript syntax
sprite(3).settingsPanel(1);
```

関連事項

on mouseDown (イベントハンドラ), rightMouseDown, controlDown

setTrackEnabled()

使用法

```
-- Lingo syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse)

// JavaScript syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse);
```

解説

デジタルビデオの指定したトラックが再生可能かどうかを示すコマンドです。

- setTrackEnabled が TRUE のときは、指定したトラックは再生できます。
- setTrackEnabled が FALSE のときは、指定したトラックは再生できず、ミュートになります。したがって、トラックはこれ以上スクリーン上にアップデートされないことになります。

トラックが既に再生可能かどうかを調べるには、trackEnabled スプライトプロパティを調べてください。

パラメータ

whichTrack 必須。テストするトラックを指定します。

trueOrFalse 必須。デジタルビデオのトラックを有効（TRUE）、または無効（FALSE）に指定します。

例文

次のステートメントは、スプライトチャネル 8 に割り当てられているビデオのトラック 3 を再生できるようにします。

```
-- Lingo syntax
sprite(8).setTrackEnabled(3, TRUE)

// JavaScript syntax
sprite(8).setTrackEnabled(3, 1);
```

関連事項

trackEnabled

setVal()

使用法

```
<Void> Matrix.setVal(whichRow, whichColumn, valueToSet)
```

解説

マトリックスメソッド。指定されたマトリックス内の指定された要素の値を設定します。

パラメータ

行 必須。値を設定する対象の要素の行番号を指定します。

whichColumn 必須。値を設定する対象の要素の列番号を指定します。

valueToSet 必須。浮動小数 . マトリックスの行番号で指定した行と列番号で指定した列に対して設定する値を指定します。

例文

次の関数は 4 つの行と 5 つの列を持つマトリックスを作成し、マトリックス内の各要素の値としてランダムな値を設定します。

```
--Lingo
on randomMatrix()
     rows = 4
     cols = 5
     mat = newMatrix(rows,cols)
     repeat with i = 1 to rows
       repeat with j = 1 to cols
         mat.setVal(i,j,random(255))
       end repeat
     end repeat
     return mat
end
//Java Script
function randomMatrix()
{
     rows = 4;
     cols = 5;
     mat = newMatrix(rows,cols);
     for(i = 1 ; i <= rows;i++)
     for(j = 1 ; j <= rows;j++)
         mat.setVal(i,j,random(255));
     return mat;
}
```

関連事項

getVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

setVariable()

使用法

```
-- Lingo syntax
spriteObjRef.setVariable(variableName, newValue)

// JavaScript syntax
spriteObjRef.setVariable(variableName, newValue);
```

解説

Flash スプライトで変数に値を設定する関数です。Flash 変数は Flash バージョン 4 で導入されました。

パラメータ

variableName 必須。変数の名前を指定します。

newValue 必須。変数の新しい値を指定します。

例文

次のステートメントは、スプライト 3 の Flash キャストメンバの変数 currentURL を設定します。currentURL の新しい値は、"http://www.adobe.com/software/flash/" になります。

```
-- Lingo syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/")

// JavaScript syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/");
```

関連事項

hitTest(), getVariable()

showLocals()

使用法

```
-- Lingo syntax
showLocals()
```

解説

最上位レベルの関数（Lingo のみ）。すべてのローカル変数をメッセージウィンドウに表示します。このコマンドは、表示したいローカル変数を含むハンドラまたは親スクリプト内でのみ有効です。メッセージウィンドウで使われる変数はすべて自動的にグローバルになります。

ハンドラ内のローカル変数は、ハンドラが実行された後は使用できません。ハンドラに showLocals() ステートメントを挿入すると、すべてのローカル変数がメッセージウィンドウに表示されます。

このコマンドは、スクリプトのデバッグに役立ちます。

パラメータ

なし

関連事項

clearGlobals(), global, showGlobals()

showProps()

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.showProps()

// JavaScript syntax
memberOrSpriteObjRef.showProps();
```

解説

Flash ムービー、ベクトルメンバ、または現在再生中のサウンドの現在のプロパティ設定のリストをメッセージウィンドウに表示するコマンドです。このコマンドは、オーサリング環境でのみ有効で、プロジェクタまたは Shockwave コンテンツを含むムービーでは使えません。

パラメータ

なし

例文

次のハンドラはキャスト名をパラメータとして受け取り、Flash ムービーキャストメンバを割り当ててから、キャストメンバの名前、番号、およびプロパティをメッセージウィンドウに表示します。

```
-- Lingo syntax
on ShowCastProperties(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        castType = member(i, whichCast).type
        if (castType = #flash) OR (castType = #vectorShape) then
            put castType&&"cast member" && i & ":" && member(i, whichCast).name
            put RETURN
            member(i ,whichCast).showProps()
        end if
    end repeat
end

// JavaScript syntax
function ShowCastProperties(whichCast) {
    i = 1;
    while( i < (castLib(whichCast).member.count) +1 ) {
        castType = member(i, whichCast).type;
        if ((castType = "flash") || (castType = "vectorShape")) {
            trace (castType + " cast member " + i + ": " + member(i, whichCast).name) + \n;
            member(i ,whichCast).showProps();
i++;
        }
    }
}
```

関連事項

queue(), setPlayList()

showGlobals()

使用法

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

パラメータ

なし

解説

グローバルメソッド。すべてのローカル変数をメッセージウィンドウに表示します。

このメソッドは、スクリプトのデバッグに役立ちます。

例文

次のステートメントは、すべてのグローバル変数をメッセージウィンドウに表示します。

```
-- Lingo syntax
on mouseDown
    _global.showGlobals()
end

// JavaScript syntax
function mouseDown() {
    _global.showGlobals();
}
```

関連事項

グローバル

shutDown()

使用法

```
-- Lingo syntax
_system.shutDown()

// JavaScript syntax
_system.shutDown();
```

解説

システムメソッド。開いているすべてのアプリケーションを閉じて、コンピュータをオフにします。

パラメータ

なし

例文

次のステートメントはユーザが Ctrl+S キー（Windows）または Command+S キー（Mac）を押したかどうかをチェックし、押している場合はコンピュータの電源を切ります。

関連事項

システム

sin()

使用法

```
sin(angle)
```

解説

数学関数（Lingo のみ）。指定した角度のサインを計算します。角度は浮動小数のラジアンで指定する必要があります。

JavaScript シンタックスでは、数学オブジェクトの sin() 関数を使用します。

パラメータ

angle 必須。角度を指定します。

例文

次のステートメントは、pi/2 のサインを求めます。

```
put sin (PI/2.0)
-- 1
```

関連事項

PI

sort

使用法

```
list.sort()
sort list
```

解説

[リスト] で指定したリスト内の項目を、アルファベット順に並び替えるためのコマンドです。

- リニアリストの場合、リストは値でソートされます。
- プロパティリストの場合、リストはプロパティごとにアルファベット順にソートされます。

一度並べ替えたリストは、add コマンドを使って変数を追加しても順序を維持します。

パラメータ

なし

例文

次のステートメントは、[#a: 1, #d: 2, #c: 3] という内容の Values リストをアルファベット順に並べ替えます。結果はステートメントの下に示してあるとおりです。

```
put values
-- [#a: 1, #d: 2, #c: 3]
values.sort()
put values
--[#a: 1, #c: 3, #d: 2]
```

sound()

使用法

```
-- Lingo syntax
sound(intSoundChannel)

// JavaScript syntax
sound(intSoundChannel);
```

解説

最上位レベルの関数。指定したサウンドチャネルへの参照を返します。

このメソッドの機能は、サウンドオブジェクトの channel() メソッドと同じです。

パラメータ

intSoundChannel 必須。参照するサウンドチャネルを指定する整数。

例文

次の例は、サウンドチャネル 1 を変数 music に割り当てて、サウンドを再生します。

```
-- Lingo syntax
music = sound(1)
music.play(member("waltz1"))

// JavaScript syntax
var music = sound(1);
music.play(member("waltz1"));
```

関連事項

channel() （サウンド）, サウンドチャネル

sprite()

使用法

```
-- Lingo syntax
sprite(nameOrNum)

// JavaScript syntax
sprite(nameOrNum);
```

解説

最上位レベルの関数。スコアで指定したスプライトへの参照を返します。

ムービーの scriptExecutionStyle プロパティを 9 の値に設定している場合、その名前のスプライトが存在しない sprite("foo") を呼び出すと、スプライト 1 への参照が返されます。ムービーの scriptExecutionStyle プロパティを 10 の値に設定している場合、その名前のスプライトが存在しない sprite("foo") を呼び出すと、Lingo から呼び出されるときには VOID、JavaScript から呼び出されるときには undefined が返されます。

パラメータ

nameOrNum 必須。スプライトの名前またはインデックス位置を指定するストリングまたは整数です。

例文

次のステートメントは、変数 thisSprite を Cave という名前のスプライトに設定します。

```
-- Lingo syntax
thisSprite = sprite("Cave")

// JavaScript syntax
var thisSprite = sprite("Cave");
```

関連事項

スプライトチャネル

spriteSpaceToWorldSpace

使用法

```
sprite(whichSprite).camera.spriteSpaceToWorldSpace(loc)
sprite(whichSprite).camera(index).spriteSpaceToWorldSpace(loc)
```

解説

3D コマンド。参照先のスプライト内の位置に対応する、指定したカメラの投影平面上のワールドスペースの位置を返します。

投影平面は、カメラの X 軸と Y 軸によって定義され、カメラの前の、1 ピクセルが 1 ワールド単位になる距離にあります。ステージ上のスプライトの表示に使用されるのは、この投影平面です。

このコマンドの camera.spriteSpaceToWorldSpace() という形式は、camera(1).spriteSpaceToWorldSpace() のショートカットです。

参照先のスプライトで使用されるすべてのカメラは、spriteSpaceToWorldSpace コマンドに対して、表示用の矩形が スプライトと同じサイズであるかのように応答します。

パラメータ

loc 必須。参照したスプライトの場所を選択します。この位置には、スプライトの左上隅を基準にしたポイントが必要です。

例文

次のステートメントは、スプライト 5 のポイント (50, 50) が、スプライト 5 のカメラの投影平面上のベクトル (-1993.6699, 52.0773, 2263.7446) と等しいことを示します。

```
-- Lingo
put sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50))
-- vector(-1993.6699, 52.0773, 2263.7446)

// Javascript
put (sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50)) );
//<vector(-1993.6699, 52.0773, 2263.7446)>
```

関連事項
worldSpaceToSpriteSpace, rect (camera), camera

sqrt()

使用法

```
sqrt(number)
the sqrt of number
```

解説
数学関数（Lingo のみ）。指定した数の平方根を計算します。

指定する数値は、必ず 0 より大きな小数値でなくてはなりません。負の値は 0 になります。

JavaScript シンタックスでは、数学オブジェクトの sqrt() 関数を使用します。

パラメータ
number 必須。数を指定します。この数は浮動小数または最も近い整数に四捨五入された整数のいずれかです。

例文
次のステートメントは、3.0 の平方根をメッセージウィンドウに表示します。

```
put sqrt(3.0)
-- 1.7321
```

次のステートメントは、3 の平方根をメッセージウィンドウに表示します。

```
put sqrt(3)
-- 2
```

関連事項
floatPrecision

stageBottom

使用法

```
the stageBottom
```

解説
stageLeft、stageRight および stageTop と共に使うことによってデスクトップ上のステージの位置を示すための関数です。メイン画面の左上隅から見たステージの下端の相対座標値を返します。ピクセル数で表されるステージの高さは、the stageBottom - the stageTop という式で求めることができます。

ムービーがアプレットとして再生される場合、stageBottom プロパティはアプレットの高さをピクセルで表します。

この関数は、値を調べることはできますが、変更はできません。

パラメータ
なし

例文

次の 2 つのステートメントは、スプライト 3 をステージの下端から 50 ピクセルのところに位置させます。

```
stageHeight = the stageBottom - the stageTop
sprite(3).locV = stageHeight - 50
```

スプライト座標値は、ステージの左上隅から測った距離で表されます。詳しくは、Director ヘルプパネルの「Director の使い方」トピックを参照してください。

```
// Javascript
var stageHeight =_movie.stage.rect.Bottom -_movie.stage.rect.Top ;
sprite(3).locV = stageHeight - 50 ;
```

関連事項

stageLeft, stageRight, stageTop, locH, locV

stageLeft

使用法

```
the stageLeft
```

解説

stageRight、stageTop および stageBottom と共に使うことによって、デスクトップ上でのステージの位置を示すための関数です。値は、メイン画面の左上隅から見たステージの左端の相対座標値に相当します。ステージの左端がメイン画面の左端にちょうど一致している場合、この座標値は 0 になります。

ムービーがアプレットとして再生される場合、stageLeft プロパティの値は 0 になり、これがアプレットの左側の位置を表します。

このプロパティは、値を調べられますが、設定はできません。

スプライト座標値は、ステージの左上隅から測った距離で表されます。

パラメータ

なし

例文

次のステートメントは、ステージの左端が画面の左端を越えているかどうかをチェックし、越えている場合は leftMonitorProcedure ハンドラを呼び出します。

```
if the stageLeft < 0 then leftMonitorProcedure
```

関連事項

stageBottom, stageRight, stageTop, locH, locV

stageRight

使用法

```
the stageRight
```

解説

stageLeft、stageTop、stageBottom と共に使うことによって、デスクトップ上でのステージの位置を示すための関数です。メイン画面の左上隅から見たステージの右端の相対座標値を返します。ピクセル数で表されるステージの幅は、"the stageRight -the stageLeft" という式で求めることができます。

ムービーがアプレットとして再生される場合、stageRight プロパティはピクセル数で表されるアプレットの幅になります。

この関数は、値を調べることはできますが、変更はできません。

スプライト座標値は、ステージの左上隅から測った距離で表されます。

パラメータ

なし

例文

次の 2 つのステートメントは、スプライト 3 をステージの右端から 50 ピクセルのところに位置させます。

```
stageWidth = the stageRight - the stageLeft
    sprite(3).locH = stageWidth - 50
```

関連事項

stageLeft, stageBottom, stageTop, locH, locV

stageToFlash()

使用法

```
-- Lingo syntax
spriteObjRef.stageToFlash(pointOnDirectorStage)

// JavaScript syntax
spriteObjRef.stageToFlash(pointOnDirectorStage);
```

解説

Director ステージ上の指定した座標値に対応する、Flash ムービースプライト内の座標値を返すための関数です。この関数は、Director ステージの座標値を受け入れ Flash ムービーの座標値を、Director での位置の値として返します。例えば、point (300,300) です。

Flash ムービーの座標値は Flash ムービーピクセル単位で測ります。この座標値は、ムービーが Flash で作成されたときの元のサイズによって特定されます。Flash ムービーの point(0,0) は常に左上隅です（キャストメンバの originPoint プロパティは、ムービーの座標値の計算には使用されず、回転とスケーリングだけに使用されます）。

stageToFlash() 関数とこれに対応する flashToStage() 関数を使って、Director ステージ座標値に直接重なる Flash ムービー座標値を調べることができます。Flash および Director のどちらでも、point(0,0) は Flash ステージまたは Director ステージの左上の隅に位置します。Flash スプライトを伸張、拡大、縮小、または回転した場合は、Director ステージ上でこれらの座標値が一致しないことがあります。

パラメータ

pointOnDirectorStage 必須。Director ステージでポイントを指定します。

例文

次のハンドラは、マウスポインタ (位置は Director ステージの座標で計測) がチャネル 5 にある Flash ムービースプライトの特定の座標 (130,10) 上にあるかどうかをチェックします。ポインタがこの Flash ムービーの特定の座標の上にあると、スクリプトは Flash ムービーを停止します。

```
-- Lingo syntax
on checkFlashRollover
    if sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV)) = point(130,10) then
        sprite(5).stop()
    end if
end

// JavaScript syntax
function checkFlashRollover() {
    var stf = sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV));
    if (stf = point(130,10)) {
        sprite(5).stop();
    }
}
```

関連事項

flashToStage()

stageTop

使用法

```
the stageTop
```

解説

デスクトップ上でのステージの位置を示すための関数です (stageBottom、stageLeft、stageRight と一緒に使います)。メイン画面のデスクトップの左上隅から見たステージの上端の相対座標値を返します。ステージがメイン画面の左上隅に位置している場合、この座標値は 0 になります。

ムービーをアプレットとして再生するときは、stageTop プロパティは常に 0 になります。

この関数は、値を調べることはできますが、変更はできません。

スプライト座標値は、ステージの左上隅から測った距離で表されます。

パラメータ

なし

例文

次のステートメントは、ステージの上端が画面の上端を越えているかどうかをチェックし、越えている場合は upperMonitorProcedure ハンドラを呼び出します。

```
if the stageTop < 0 then upperMonitorProcedure
```

関連事項

stageLeft, stageRight, stageBottom, locH, locV

startSave （ミクサ）

使用法

```
mixer.startSave(bitdepth,samplingFreq,NumChannels,filePath)
```

解説

ミクサメソッド。指定したファイルにミクサの出力を保存します。出力は WAV または MP4 ファイルとして保存できます。startSave はミクサの再生中にのみ使用してください。この関数では、再生しているミクサ出力がキャプチャされて、指定したファイルに保存されます。

BitDepth、samplingFreq、NumChannels パラメータを指定しない場合は、元のミクサ出力のパラメータを使用してファイルが保存されます。

WAV 出力には 8、16、および 24 のビット深度がサポートされています。ただし、MP4 出力には、16 ビット深度のみがサポートされています。

注意：startSave で開始した保存操作を終了するには、stopSave() メソッドを使用します。

例文

```
-- Lingo
on mouseUp me
mixerRef.startSave(16,44100,2,"C:\audio.wav")
end
// Javascript
function mouseUp(){
mixerRef.startSave(16,44100,2,"C:\audio.wav");
}
```

関連事項

stopSave （ミクサ）, ミクサ

startSave （サウンドオブジェクト）

使用法

```
soundObject.startSave(bitdepth,samplingFreq,NumChannels,filename)
```

解説

サウンドオブジェクトメソッド。現在再生しているサウンドオブジェクトを指定したファイルに保存します。サウンドオブジェクトは WAV または MP4 ファイルとして保存できます。startSave はサウンドの再生中にのみ使用してください。この関数では、再生しているサウンドがキャプチャされて、指定したファイルに保存されます。

BitDepth、samplingFreq、NumChannels パラメータを指定しない場合は、元のサウンドオブジェクトのパラメータを使用してファイルが保存されます。

WAV 出力には 8、16、および 24 のビット深度がサポートされています。ただし、MP4 出力には、16 ビット深度のみがサポートされています。

注意：startSave で開始した保存操作を終了するには、stopsave() メソッドを使用します。

例文

次の例文では、現在のサウンドオブジェクトを、サンプリング周波数 48Khz およびビット深度 16 で、C:\audio.wav として保存します。

```
--Lingo syntax
on mouseUp me
     soundObjRef.startSave(16,48000,2,"C:\audio.wav")
end.

// JavaScript syntax
function mouseUp(){
soundObjRef.startSave(16,48000,2,"C:\audio.wav");
}
```

関連事項

stopSave（サウンドオブジェクト）

status()

使用法

```
-- Lingo syntax
fileioObjRef.status()

// JavaScript syntax
fileioObjRef.status();
```

解説

FileIO メソッド。最後に呼び出したメソッドのエラーコードを返します。

パラメータ

なし

例文

次のステートメントでは、サウンドチャネル 2 の現在のステータスをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

関連事項

Fileio

stop() (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.stop()

// JavaScript syntax
dvdObjRef.stop();
```

解説

DVD メソッド。再生を停止します。

このメソッドは成功すると、TRUE (1) を返します。

パラメータ

なし

例文

次のステートメントは、再生を停止します。

```
-- Lingo syntax
member(1).stop()

// JavaScript syntax
member(1).stop();
```

関連事項

DVD

stop() (Flash)

使用法

```
-- Lingo syntax
spriteObjRef.stop()

// JavaScript syntax
spriteObjRef.stop();
```

解説

現行フレームで再生中の Flash ムービースプライトを停止するための Flash コマンドです。

パラメータ

なし

例文

次のフレームスクリプトは、チャネル 5 ～ 10 で再生中の Flash ムービースプライトを停止します。

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).stop()
    end repeat
end

// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).stop();
        i++;
    }
}
```

関連事項
hold()

stop() （ミクサ）

使用法
```
Mixer.stop([soundobjectlist])
```

解説
ミクサメソッド。サウンドオブジェクトリストが渡されていない場合にミクサ内のすべてのサウンドオブジェクトの再生を停止したり、渡されたサウンドオブジェクトのリストを停止します。

ミクサの操作	効果
stop()	ミクサを停止します。
stop([so1, so2])	参照 so1 および so2 を使用してミクサ内の指定したサウンドオブジェクトを停止します。
stop(["so1", "so2"])	名前 so1 および so2 を持つ、ミクサ内の指定したサウンドオブジェクトを停止します。

例文
```
-- Lingo syntax
on mouseUp me
     mixer1.stop() --Stops mixer1.
end

// JavaScript syntax
function mouseUp()
{
mixer1.stop(); //Stops mixer1.

}
```

関連事項
ミクサ

stop() (MP4Media/FLV)

使用法
```
sprite(1).stop()
member("MP4Media/FLV").stop()
```

解説
MP4Media/FLV メンバまたはスプライトメソッド。MP4Media/FLV スプライトの再生を停止します。mediaStatus の値は #stopped に変わります。

例文
次の例は、スプライト 2 の再生を停止します。

```
-- Lingo syntax
sprite(2).stop()
member("MP4Media/FLV").stop()

// JavaScript syntax
sprite(2).stop();
member("MP4Media/FLV").stop();
```

stop() (RealMedia, SWA, Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.stop()
realMediaObjRef.stop()

// JavaScript syntax
windowsMediaObjRef.stop();
realMediaObjRef.stop();
```

解説

Windows Media または RealMedia のキャストメンバまたはスプライトメソッドです。Windows Media または RealMedia のキャストメンバまたはスプライトの再生を停止します。

パラメータ

なし

例文

次の例は、それぞれスプライト 2 とキャストメンバ "Real" の再生を停止します。

```
-- Lingo syntax
sprite(2).stop()
member("Real").stop()

// JavaScript syntax
sprite(2).stop();
member("Real").stop();
```

関連事項

RealMedia, Windows Media

stop() （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.stop()

// JavaScript syntax
soundChannelObjRef.stop();
```

解説

サウンドチャネルメソッド。サウンドチャネルで現在再生中のサウンドを停止します。

play() メソッドを発行すると、指定のサウンドチャネルのキューに残っている最初のサウンドの再生が開始されます。

完成したムービーで使用する stop() の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内のサウンドコントロールムービーを参照してください。

パラメータ

なし

例文

次のステートメントは、現在サウンドチャネル 1 で再生中のサウンドキャストメンバの再生を停止します。

```
-- Lingo syntax
sound(1).stop()

// JavaScript syntax
sound(1).stop();
```

関連事項

getPlayList(), pause() （サウンドチャネル）, play() （サウンドチャネル）, playNext() （サウンドチャネル）, rewind() （サウンドチャネル）, サウンドチャネル

stop （サウンドオブジェクト）

使用法

```
soundObject.stop()
```

解説

サウンドオブジェクトメソッド。サウンドオブジェクトと関連付けられたオーディファイルの再生を停止します。

例文

```
-- Lingo syntax
on mouseUp me
     soundObjRef.stop() -- Stops the playback of the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.stop();  //Stops the playback of the sound object associated with soundObjRef.
}
```

stop

使用法

```
stop(MUIObject, stopItem)
```

解説

この関数は、MUI Xtra のインスタンスから作成された汎用目的ダイアログを停止します。関数が呼び出された後、Lingo は stopItem パラメータの数値として結果を返します。

stopItem パラメータは run(MUIOject) 呼び出しから返されます。これを使用して、ダイアログボックスが停止された方法を示すパラメータを渡します。例えば、ユーザが「OK」をクリックした場合は 1 を、ユーザが「Cancel」をクリックした場合は 0 を返すことができます。

注意：非モーダルダイアログボックスを閉じるには、WindowOperation コマンドに「#hide」オプションを付けて使用します。

例文

このハンドラは、MUIObject から作成された汎用目的ダイアログボックスを停止します。stop コマンドの 2 番目のパラメータは 0 です。これは値の要件を満たしますが、それ以外の目的はありません。

```
--Lingo syntax
on stopDialog
    global MUIObject
    if ( objectP (MUIObject)) then
        stop(MUIObject, 0)
    end if
end stopDialog
```

stopEvent()

使用法

```
-- Lingo syntax
_movie.stopEvent()

// JavaScript syntax
_movie.stopEvent();
```

解説

ムービーメソッド。スクリプトがメッセージ階層内での次の階層にイベントメッセージを渡すのを防ぎます。

このメソッドは、スプライトスクリプトにも適用されます。

stopEvent() メソッドを使用して、初期イベントハンドラやスプライトスクリプト内のメッセージを停止すると、メッセージは以降の階層に渡されなくなります。

デフォルトでは、メッセージはまず初期イベントハンドラ (存在する場合) に渡され、その後にイベントに関連するスプライトに割り当てられたスクリプトに渡されます。スプライトに複数のスクリプトが割り当てられている場合、メッセージはスプライトの各スクリプトに渡されます。メッセージにスプライトスクリプトが反応しない場合は、メッセージはキャストメンバスクリプト、フレームスクリプト、ムービースクリプトに渡されます。

stopEvent() メソッドは、処理されている現行イベントにのみ適用でき、まだ起こっていないイベントに対しては効果がありません。stopEvent() メソッドが適用できるのは、初期イベントハンドラ、初期イベントハンドラから呼び出されるハンドラ、または複数のスプライトスクリプト内に限り、これら以外では効果はありません。

パラメータ

なし

例文

次のステートメントは、グローバル変数 grandTotal が 500 の場合、ビヘイビアーで mouseUp イベントが停止される例を次に示します。

```
-- Lingo syntax
global grandTotal
on mouseUp me
    if (grandTotal = 500) then
        _movie.stopEvent()
    end if
end

// JavaScript syntax
_global.grandTotal;
function mouseUp() {
    if (_global.grandTotal == 500) {
        _movie.stopEvent();
    }
}
```

この方法でイベントを停止すると、それ以降のスクリプトも、同じスプライトの他のビヘイビアーも、このイベントを受信しません。

関連事項

ムービー

stopSave （ミクサ）

使用法

```
mixer.stopSave()
```

解説

ミクサメソッド。startSave により開始された保存操作を終了します。

例文

```
--Lingo syntax
on mouseUp me
     mixerRef.stopSave() --Ends the save operation.
end

//JavaScript syntax
function mouseUp(){
     mixerRef.stopSave(); //Ends the save operation.
}
```

関連事項

startSave （ミクサ）, ミクサ

stopSave （サウンドオブジェクト）

使用法

```
soundObject.stopSave()
```

解説

サウンドオブジェクトメソッド。startSave により開始された保存操作を終了します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.stopSave() --Ends the save operation.
end

//JavaScript syntax
function mouseUp(){
     soundObjRef.stopSave(); //Ends the save operation.
}
```

関連事項

startSave（サウンドオブジェクト）

stream()

使用法

```
-- Lingo syntax
memberObjRef.stream(numberOfBytes)

// JavaScript syntax
memberObjRef.stream(numberOfBytes);
```

解説

指定した Flash ムービーキャストメンバの一部を、メモリに手動でストリームするためのコマンドです。

stream コマンドは、実際にストリームされたバイト数を返します。ネットワークのスピードや要求されたデータの使用可能度などの条件により、実際にストリームされるバイト数は、要求したバイト数より少なくなることがあります。

stream コマンドは、キャストメンバの streamMode プロパティにかかわりなく、キャストメンバに対して常に使うことができます。

パラメータ

numberOfBytes オプション。ストリームするバイト数を指定する整数です。[バイト数] パラメータを省略すると、Director はキャストメンバの bufferSize プロパティによって設定されたバイト数をストリームしようとします。

例文

次のフレームスクリプトは、リンクされた Flash ムービーキャストメンバの percentStreamed プロパティを調べることにより、ストリームがすべてメモリに格納されたかどうかを判断します。キャストメンバがメモリに完全にはロードされていない場合、このスクリプトは 32,000 バイトのムービーをメモリにストリームしようとします。

このスクリプトはまた、bytesReceived という変数に実際にストリームされたバイト数も保存します。実際にストリームされたバイト数が要求したバイト数と一致しない場合、このスクリプトは、実際に受信したバイト数を報告するためのテキストキャストメンバを更新します。キャストメンバのメモリへのロードが終了するまで、再生ヘッドを現在のフレームでループさせます。

```
-- Lingo syntax
on exitFrame
    if member(10).percentStreamed < 100 then
        bytesReceived = member(10).stream(32000)
        if bytesReceived < 32000 then
            member("Message Line").text = "Received only" && bytesReceived && "of 32,000 bytes requested."
            _movie.updateStage()
        else
            member("Message Line").text = "Received all 32,000 bytes."
        end if
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member(10).percentStreamed;
    if (pctStm < 100) {
        var bytesReceived = member(10).stream(32000);
        if (bytesReceived < 32000) {
            member("Message Line").text = "Received only " + bytesReceived + " of32,000 bytes requested.";
            _movie.updateStage();
        } else {
            member("Message Line").text = "Received all 32,000 bytes.";
        }
        _movie.go(_movie.frame);
    }
}
```

string()

使用法

```
string(expression)
```

解説

整数、浮動小数、オブジェクト参照、リスト、シンボルその他のストリング以外の [式] をストリング (文字列) に変換するための関数です。

パラメータ

expression 必須。ストリングに変換する式です。

例文

次のステートメントは、2.0+2.5 を計算して、結果を " 合計 " フィールドキャストメンバに挿入します。内容は "4.5" になります。

```
--Lingo
member("total").text = string(2.0 + 2.5)

// Javascript
member("total").text = (2.0 + 2.5).toString() ;
```

次のステートメントは、シンボル #red をストリングに変換し、これをフィールドキャストメンバの "Color" に挿入します。内容は "red" になります。

```
--Lingo
member("Color").text = string(#red)

// Javascript
member("Color").text = symbol("red").toString();
```

関連事項

value(), stringP(), float(), integer(), symbol()

stringP()

使用法

```
stringP(expression)
```

解説

[式] で指定した式がストリングか (TRUE)、ストリングでないか (FALSE) を確認するための関数です。

stringP の "**P**" は "**predicate** (属性)" を意味しています。

パラメータ

expression 必須。検査する式。

例文

次のステートメントは、"**3**" がストリングかどうかをチェックします。

```
put stringP("3")
```

結果は、TRUE を表す数値の 1 になります。

次のステートメントは、浮動小数 3.0 がストリングかどうかをチェックします。

```
put stringP(3.0)
```

3.0 は浮動小数でありストリングではないので、結果は FALSE を表す数値の 0 になります。

関連事項

floatP(), ilk(), integerP(), objectP(), symbolP()

subPictureType()

使用法

```
-- Lingo syntax
dvdObjRef.subPictureType(intStream)

// JavaScript syntax
dvdObjRef.subPictureType(intStream);
```

解説

DVD メソッド。指定したサブピクチャストリームの種類を選択します。

このメソッドは次の値を返す可能性があります。

Symbol	解説
#unknown	サブピクチャの種類は不明です。
#Language	サブピクチャには、ムービーのサブタイトルや他のテキストなど、言語に関連するコンテンツが含まれています。
#Other	サブピクチャには、"karaoke" のタイトルの跳ねるボールなど、言語に関係のないコンテンツが含まれています。

パラメータ

intStream 必須。テストするストリームを指定する整数です。

例文

次のステートメントは、ストリーム 2 のサブピクチャの種類を返します。

```
-- Lingo syntax
member(12).member.subPictureType(2)

// JavaScript syntax
member(12).member.subPictureType(2);
```

関連事項

DVD

substituteFont

使用法

```
TextMemberRef.substituteFont(originalFont, newFont)
substituteFont(textMemberRef, originalFont, newFont)
```

解説

テキストキャストメンバコマンド。テキストキャストメンバで、1 つのフォントのすべてのインスタントをテキストキャストメンバの別のフォントに置き換えます。

パラメータ

originalFont 必須。置き換えるフォントです。

newFont 必須。**originalFont** によって指定されたフォントを置き換える新しいフォントです。

例文

次のスクリプトは、"Bonneville" フォントをテキストキャストメンバで使えるかどうかをチェックし、使えない場合は "Arial" に置き換えます。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    currMember = sprite(spriteNum).member
    if currMember.missingFonts contains "Bonneville" then
        currMember.substituteFont("Bonneville", "Arial")
    end if
end

// JavaScript syntax
function beginSprite() {
    currMember = sprite(spriteNum).member;
    if (currMember.missingFonts contains "Bonneville") { //check syntax
        currMember.substituteFont("Bonneville", "Arial");
    }
}
```

関連事項

missingFonts

swing()

使用法

```
-- Lingo syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing)

// JavaScript syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing);
```

解説

QuickTime VR スプライト関数。VR Pano を含む QuickTime 3 スプライトを、新しい表示設定でスイングさせます。スイングは、カメラを滑らかに「移動撮影」（ドリーカメラ）する効果です。

whichQTVRSprite は QuickTime VR メンバを含むスプライトのスプライト番号です。

この関数はすぐに実行されますが、スプライトは最後の指定された状態に達するまで表示を変更し続けます。最後の設定変更までにかかる時間は、マシンタイプ、そのスプライト矩形のサイズ、スクリーンの色深度、その他、パフォーマンスに影響する各種の条件に応じて異なります。

スイングが完了したかどうかを調べるには、スプライトの pan プロパティが最後の値になっているかをチェックします。

パラメータ

pan 必須。新しいパンの位置を度数で指定します。

tilt 必須。新しい傾きを度数で指定します。

fieldOfView 必須。新しい視野を度数で指定します。

speedToSwing 必須。スイングを実行する速さを指定します。有効な値の範囲は 1 （遅い）～ 10 （速い）です。

例文

次の例では、QTVR スプライト 1 の表示をゆっくりと段階的に調整して、パン位置 300°、傾き -15°、視野 40° に設定します。

```
-- Lingo syntax
sprite(1).swing(300, -15, 40, 1)

// JavaScript syntax
sprite(1).swing(300, -15, 40, 1);
```

関連事項

pan（QTVR プロパティ）

symbol()

使用法

```
-- Lingo syntax
symbol(stringValue)

// JavaScript syntax
symbol(stringValue);
```

解説

最上位レベルの関数。ストリングを取得し、シンボルを返します。

パラメータ

stringValue 必須。シンボルに変換するストリングです。

例文

次のステートメントは、#hello シンボルを表示します。

```
--Lingo syntax
put(symbol("hello"))

// JavaScript syntax
put(symbol("hello"));
```

次のステートメントは、#goodbye シンボルを表示します。

```
--Lingo syntax
x = "goodbye"
put(symbol(x))

// JavaScript syntax
var x = "goodbye";
put(symbol(x));
```

関連事項

value(), string()

symbolP()

使用法

```
Expression.symbolP
symbolP(expression)
```

解説
関数。指定した式がストリングか（TRUE）、ストリングでないか（FALSE）を判断します。

symbolP の "**P**" は属性を意味しています。

パラメータ
expression 必須。テストする式を指定します。

例文
次のステートメントは、変数 myVariable がシンボルかどうかをチェックします。

```
put myVariable.symbolP
```

関連事項
ilk()

tan()

使用法
```
tan(angle)
```

解説
指定された角度のタンジェント (正接) を返す数学関数です。角度は浮動小数のラジアンで表されている必要があります。

JavaScript シンタックスでは、数学オブジェクトの tan() 関数を使用します。

パラメータ
angle 必須。タンジェントを求める角度を指定します。

例文
次の関数は、π/4 のタンジェントをメッセージウィンドウに表示します。

```
tan (PI/4.0) = 1
```

Lingo の式内では "π" 記号を使用できません。

関連事項
PI

tellStreamStatus()

使用法
```
tellStreamStatus(onOrOffBoolean)
```

解説
ストリーミングのステータスハンドラをオン (TRUE) またはオフ (FALSE) にするための関数です。

tellStreamStatus() という形式は、ハンドラのステータスを示します。

streamStatusHandler が TRUE の場合、インターネットストリーミングアクティビティによって、ムービースクリプトへの定期的な呼び出しが行われ、streamStatusHandler が起動します。このハンドラが実行されると、Director は、ダウンロードの進行状況に従って自動的にパラメータを指定します。

パラメータ

onOrOffBoolean オプション。ハンドラのステータスを指定します。

例文

次の on prepareMovie ハンドラは、ムービーの開始時に on streamStatus ハンドラをオンにします。

```
-- Lingo syntax
on prepareMovie
    tellStreamStatus(TRUE)
end

// JavaScript syntax
function prepareMovie() {
    tellStreamStatus(TRUE);
}
```

次のステートメントは、ストリーミングのステータスハンドラのステータスを示します。

```
-- Lingo syntax
on mouseDown
    put tellStreamStatus()
end

// JavaScript syntax
function mouseDown() {
    put(tellStreamStatus());
}
```

関連事項

on streamStatus

tellTarget()

使用法

```
-- Lingo syntax
spriteObjRef.tellTarget(targetName)

// JavaScript syntax
spriteObjRef.tellTarget(targetName);
```

解説

コマンド。Flash の beginTellTarget メソッドおよび endTellTarget メソッドと同じです。tellTarget() コマンドを使用すると、それ以降のスプライトコマンドが機能するターゲットとなるタイムラインを設定できます。Flash ムービークリップまたはロード済みの Flash ムービーを含んだレベルにターゲットを設定すると、メインのタイムラインではなく、ターゲットのコンポーネントで一定のコマンドが機能します。メインのタイムラインにフォーカスを戻すには、endTellTarget() を呼び出します。

tellTarget に指定できる有効なパラメータは、ターゲット名のみです。endTellTarget には、有効なパラメータはありません。

tellTarget の影響を受ける Flash 関数は、stop、play、getProperty、setProperty、gotoFrame、call(frame)、および find(label) です。また、現在のフレームを返すスプライトプロパティの frame も、tellTarget の影響を受けます。

パラメータ

targetName 必須。ターゲット名を指定します。

例文

次のコマンドは、スプライト 1 のムービークリップ "myMovieClip" をターゲットとして設定します。

```
-- Lingo syntax
sprite(1).tellTarget("myMovieClip")

// JavaScript syntax
sprite(1).tellTarget("myMovieClip");
```

次のコマンドは、ムービークリップを停止します。

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

次のコマンドは、"myMovieClip" を再生します。

```
-- Lingo syntax
sprite(1).play()

// JavaScript syntax
sprite(1).play();
```

次のコマンドは、フォーカスを切り替えて、制御をメインのタイムラインに戻します。

```
-- Lingo syntax
sprite(1).endTellTarget()

// JavaScript syntax
sprite(1).endTellTarget();
```

次のコマンドは、メインのムービーを停止します。

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

time() （システム）

使用法

```
-- Lingo syntax
_system.time()

// JavaScript syntax
_system.time();
```

解説

システムメソッド。システムクロックの現在の時間をストリングで返します。時間ストリングの形式はコンピュータの時間の設定によって異なります。

パラメータ

なし

例文

次のハンドラは、現在の時間をテキストフィールドに出力します。

```
-- Lingo syntax
on exitFrame
    member("clock").text = _system.time()
end

// JavaScript syntax
function exitFrame() {
    member("clock").text = _system.time();
}
```

関連事項

date() （システム）, システム

timeout()

使用法

```
-- Lingo syntax
timeout(timeoutObjName)

// JavaScript syntax
timeout(timeoutObjName);
```

解説

最上位レベルの関数。指定したタイムアウトオブジェクトを返します。

new() メソッドを使用すると、新しいタイムアウトオブジェクトを作成し、timeoutList に追加できます。

パラメータ

timeoutObjName 必須。返されるタイムアウトオブジェクトの名前を指定するストリングです。

例文

次のハンドラは、"Random Lightning" というタイムアウトオブジェクトを削除します。

```
-- Lingo syntax
on exitFrame
    timeout("Random Lightning").forget()
end

// JavaScript syntax
function exitFrame() {
    timeout("Random Lightning").forget();
}
```

関連事項

new(), timeoutList, timeoutHandler, time （タイムアウトオブジェクト）, name (timeout), period, persistent, target

titleMenu()

使用法

```
-- Lingo syntax
dvdObjRef.titleMenu()

// JavaScript syntax
dvdObjRef.titleMenu();
```

解説

DVD メソッド。タイトルメニューを表示します。

パラメータ

なし

例文

次のステートメントはタイトルメニューを表示します。

```
-- Lingo syntax
member(1).titleMenu()

// JavaScript syntax
member(1).titleMenu();
```

関連事項

DVD

toHexString

使用法

```
toHexString(intOffset, intLength)
```

解説

バイト配列メソッド。バイト配列の内容をストリングとして返します。(offset+length) がバイト配列の実際の長さを超える場合、バルト配列は (actualLength–offset) に切り詰められます。

このメソッドはメッセージウィンドウのみで使用し、デバッガでは使用しないでください。ストリングは、デバッガウォッチウィンドウでは切り詰められます。

パラメータ

パラメータ	解説	必須 / オプション
intOffset	開始オフセット位置。	必須
intLength	バイト配列の長さ。	必須

例文

```
--Lingo syntax
put bArray.toHexString(1,50)

//JavaScript syntax
put(bArray.toHexString(1,50));
```

top (3D)

使用法

```
modelResourceObjectReference.top
```

解説

3D コマンド。#box タイプのモデルリソースを指定すると、そのモデルリソースの top プロパティを取得および設定できます。

top プロパティでは、ボックスの上面を閉じた状態にするか (TRUE)、または開いた状態にするか (FALSE) を指定できます。デフォルト値は TRUE です。

パラメータ

なし

例文

次のステートメントは、スプライト 3 の上端がステージの上端を越えているかどうかをチェックし、越えている場合は offTopEdge ハンドラを呼び出します。

```
-- Lingo syntax
if (sprite(3).top < 0) then
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

関連事項

back, bottom (3D), front

topCap

使用法

```
modelResourceObjectReference.topCap
```

解説

3D コマンド。#cylinder というタイプのモデルリソースを指定すると、そのモデルリソースの topCap プロパティを取得および設定できます。

topCap プロパティは、円柱の上面が閉じているか（TRUE）、開いているか（FALSE）を示します。このプロパティのデフォルト値は FALSE です。

パラメータ
なし

例文
次のステートメントは、モデルリソース "Tube" の topCap プロパティを FALSE に設定します。この円柱の上面は開いた状態になります。

```
-- Lingo syntax
member("3D World").modelResource("Tube").topCap = FALSE

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topCap = false;
```

topRadius

使用法

```
modelResourceObjectReference.topRadius
```

解説
3D コマンド。#cylinder というタイプのモデルリソースを指定すると、そのモデルリソースの topRadius プロパティを浮動小数点値で取得および設定できます。

topRadius プロパティでは、円柱の上面の半径を指定できます。このプロパティの値は常に 0.0 以上でなければなりません。デフォルト値は 25.0 です。topRadius を 0.0 に設定すると、円錐になります。

パラメータ
なし

例文
次のステートメントは、モデルリソース「Tube」の topRadius プロパティを 0.0 に設定します。下部の半径が 0 より大きい場合、「Tube」を使用するモデルは円錐状になります。

```
-- Lingo syntax
member("3D World").modelResource("Tube").topRadius = 0.0

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topRadius = 0.0;
```

trace()

使用法

```
-- Lingo syntax
trace(value)

// JavaScript syntax
trace(value);
```

解説
最上位レベルの関数。式を評価し、結果をメッセージウィンドウに表示します。

このメソッドの機能は、Lingo および JavaScript シンタックスの両方で使用できる最上位レベルの put() メソッドと同じです。

このメソッドは、ムービー再生中に変数の値を監視するデバッグ用ツールとして使用できます。

パラメータ
value 必須。評価する式です。

例文
次のステートメントは、変数 counter の値をメッセージウィンドウに出力します。

```
-- Lingo syntax
counter = (_system.milliseconds / 1000)
trace(counter)

// JavaScript syntax
var counter = (_system.milliseconds / 1000);
trace(counter);
```

関連事項
put()

transform （コマンド）

使用法

```
transform()
```

解説
3D コマンド。このコマンドはアイデンティティトランスフォームと同じトランスフォームオブジェクトを作成します。アイデンティティトランスフォームは、vector(0,0,0) の位置コンポーネントと回転コンポーネント、および vector(1,1,1) のスケールコンポーネントを備えています。

トランスフォーム情報を保存してから再構築する必要がある場合、まずトランスフォームプロパティ（位置、回転およびスケール）を保存します。次に、アイデンティティトランスフォームを作成してから、保存したデータを使用して位置、回転およびスケールを設定し、トランスフォームを再構築します。

パラメータ
なし

例文
次のステートメントは、アイデンティティトランスフォームを作成し、変数 tTransform に格納します。

```
-- Lingo syntax
tTransform = transform()

// JavaScript syntax
tTransform = transform();
```

関連事項
transform （プロパティ）, preRotate, preTranslate(), preScale(), rotate, translate, scale （コマンド）

translate

使用法

```
member(whichCastmember).node(whichNode).translate(xIncrement, yIncrement, zIncrement {, relativeTo})
member(whichCastmember).node(whichNode).translate(translateVector {, relativeTo})
transform.translate(xIncrement, yIncrement, zIncrement {, relativeTo})
transform.translate(translateVector {, relativeTo})
```

解説

3D コマンド。参照されたノードのトランスフォームオブジェクト、または直接参照されたトランスフォームオブジェクトで保持される位置、回転、およびスケールのオフセットを適用した後に、平行移動を適用します。平行移動は、対応する 3 つの軸に沿った 3 つの増分のセットとして指定する必要があります。これらの増分は、**x** 増分、**y** 増分、および **z** 増分 の形式で、または平行移動ベクトル で明示的に指定します。平行移動ベクトルの x コンポーネントは x 軸に沿った平行移動に対応します。同様に、y コンポーネントは y 軸、z コンポーネントは z 軸に対応します。

ノードは、カメラ、モデル、ライト、またはグループオブジェクトのいずれかです。

パラメータ

xIncrement は、3 つの増分のセットを指定する場合に必要です。**x** 軸の増分を指定します。

yIncrement は、3 つの増分のセットを指定する場合に必要です。**y** 軸の増分を指定します。

zIncrement は、3 つの増分のセットを指定する場合に必要です。**z** 軸の増分を指定します。

translateVector は、ベクトルを指定する場合に必要です。**x**、**y** および **z** 成分を含むベクトルを指定します。

基準はオプションです。平行移動の変更を適用する際に使用する座標系の軸を示します。基準パラメータの値を以下に示します。

- #self 。ノードのローカル座標系 (モデルの作成時に指定した x 軸、y 軸、および z 軸) を基準に増分が適用されます。translate コマンドにノード参照を指定していて、基準パラメータが指定されていない場合、この値がデフォルトとして使用されます。
- #parent 。ノードの親の座標系に対して増分を適用します。translate コマンドにトランスフォーム参照を指定していて、基準パラメータが指定されていない場合、この値がデフォルトとして使用されます。
- #world 。ワールド座標系に対して増分を適用します。モデルの親がワールドである場合は、#parent を使用するのと同じになります。
- nodeReference。平行移動の基準となるノードを指定できます。このコマンドは、指定されたノードの座標系を基準にして平行移動を適用します。

例文

次の例は、transform コマンドを使用してトランスフォームを構築します。次に、スペースにおいてトランスフォームの位置と向きを初期化してから、「火星」というモデルにトランスフォームを設定します。最後に、モデルの結果の位置を表示します。

```
t =transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.translate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(100.0000, 0.0000, 0.0000)
```

次のステートメントは、モデル "Bip" を親ノードの x 軸に沿って 20 単位移動します。

```
put member("Scene").model("Bip").position
-- vector( -38.5000, 21.2500, 2.0000)
member("Scene").model("Bip").translate(20, 10, -0.5)
put member("Scene").model("Bip").position
-- vector( -18.5000, 31.2500, 1.5000)
```

関連事項

transform （プロパティ）, preTranslate(), scale （コマンド）, rotate

uncompress()

使用法

```
byteArrayObject.uncompress()
```

解説

バイト配列メソッド。標準の Zlib 圧縮解除を使用して、バイト配列の内容を圧縮解除します。圧縮解除後に、位置は 1 に設定されます。

例文

```
--Lingo syntax
bArray.uncompress()

//JavaScript syntax
bArray.uncompress();
```

union()

使用法

```
rect(1).union(rect(2))
union (rect1, rect2)
```

解説

関数。2 つの矩形を囲む最小矩形を返します。

パラメータ

rect2 必須。2 番目の矩形を指定します。

例文

次のステートメントは、指定した矩形を囲む矩形を返します。

```
-- Lingo syntax
put union (rect (0, 0, 10, 10), rect (15, 15, 20, 20))
-- rect (0, 0, 20, 20)
```

または

```
put rect(0, 0, 10, 10).union(rect(15, 15, 20, 20))
--rect (0, 0, 20, 20)

// JavaScript syntax
put ( rect (0, 0, 10, 10).union( rect (15, 15, 20, 20) ) );
// <rect(0, 0, 20, 20)>
```

関連事項

map(), rect()

unLoad() （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.unLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.unLoad({toMemberObjRef});
```

解説

メンバメソッド。指定したキャストメンバを強制的にメモリから解放します。

Director は、preLoad() メソッドまたは通常のキャストライブラリのロードに対応するために、使用されてから最も時間が経過しているキャストメンバを自動的にメモリから解放します。

- パラメータなしで使用した場合、unLoad() は現在のキャストメンバをメモリから解放します。
- **toMemberObjRef** パラメータなしで使用した場合、unLoad() は指定した範囲内のすべてのキャストメンバをメモリから解放します。

  キャストメンバがロードされていない新しいムービーでこのメソッドを使用すると、コマンドはエラーを返します。

オーサリング中に変更されたか、picture、pasteClipBoadInto() の設定などにより変更されたキャストメンバは、アンロードできません。

パラメータ

toMemberObjRef オプション。メモリから解放する範囲のサウンドキャストメンバの参照です。

例文

次のステートメントは、Ships というキャストメンバをメモリから解放します。

```
-- Lingo syntax
member("Ships").unLoad()

// JavaScript syntax
member("Ships").unLoad();
```

次のステートメントは、10 ～ 15 のキャストメンバをメモリから解放します。

```
-- Lingo syntax
member(10).unLoad(15)

// JavaScript syntax
member(10).unLoad(15);
```

関連事項

メンバ , unLoad() （ムービー） , unLoadMember(), unLoadMovie()

unLoad() （ムービー）

使用法

```
-- Lingo syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum})

// JavaScript syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum});
```

解説

ムービーメソッド。ムービーの指定フレームをメモリから解放します。

空きメモリが少なくなってきたときに、どのムービーを優先してメモリから解放するかを決めるときに使うことができます。

ムービーは、URL パスを使うことができます。

ムービーが RAM にまだ読み込まれていない場合、結果は -1 になります。

パラメータ

intFromFrameNum オプション。メモリからアンロードする範囲内にある最初のフレームの番号を指定する整数です。

intToFrameNum オプション。メモリからアンロードする範囲内にある最後のフレームの番号を指定する整数です。

例文

次のステートメントは、10 ～ 25 のフレームをメモリからアンロードします。

```
-- Lingo syntax
_movie.unLoad(10, 25)

// JavaScript syntax
_movie.unLoad(10, 25);
```

関連事項

ムービー , unLoad() （メンバ） , unLoadMember(), unLoadMovie()

unLoadMember()

使用法

```
-- Lingo syntax
_movie.unLoadMember({memberObjRef})
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum)

// JavaScript syntax
_movie.unLoadMember({memberObjRef});
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum);
```

解説

ムービーメソッド。特定のキャストメンバまたはキャストメンバの範囲を強制的にメモリから解放します。Director は、preLoad() メソッドまたは通常のキャストライブラリのロードに対応するために、使用されてから最も時間が経過しているキャストメンバを自動的にメモリから解放します。

- パラメータなしで使用した場合、unLoadMember() メソッドはムービーのすべてのフレームで使われているキャストメンバをメモリから解放します。
- **memberObjRef** パラメータのみを指定した場合、unLoadMember() メソッドは指定したキャストメンバをメモリから解放します。
- **fromMemberNameOrNum** および **toMemberNameOrNum** の 2 つのパラメータを指定した場合、unLoadMember() メソッドは指定した範囲内のすべてのキャストメンバをアンロードします。キャストメンバの範囲の指定には、メンバ番号とメンバ名のどちらも使用できます。

パラメータ

memberObjRef オプション。メモリからアンロードするキャストメンバの参照です。

fromMemberNameOrNum は、一定範囲のキャストメンバを解放する場合に必要です。メモリからアンロードする範囲内にある最初のキャストメンバの名前または番号を指定するストリングまたは整数です。

toMemberNameOrNum は、一定範囲のキャストメンバを解放する場合に必要です。メモリからアンロードする範囲内にある最後のキャストメンバの名前または番号を指定するストリングまたは整数です。

例文

次のステートメントは、" 画面 1" キャストメンバをメモリから解放します。

```
-- Lingo syntax
_movie.unLoadMember(member("Screen1"))

// JavaScript syntax
_movie.unLoadMember(member("Screen1));
```

次のステートメントは、キャストメンバ 1 から " ビッグムービー " キャストメンバまでのすべてのキャストメンバをメモリから解放します。

```
-- Lingo syntax
_movie.unLoadMember(member(1), member("Big Movie"))

// JavaScript syntax
_movie.unLoadMember(member(1), member("Big Movie));
```

関連事項

ムービー , unLoad() （メンバ）, unLoad() （ムービー）, unLoadMovie()

unLoadMovie()

使用法

```
-- Lingo syntax
_movie.unLoadMovie(stringMovieName)

// JavaScript syntax
_movie.unLoadMovie(stringMovieName);
```

解説

ムービーメソッド。指定したプリロードされているムービーをメモリから解放します。

空きメモリが少なくなってきたときに、どのムービーを優先してメモリから解放するかを決めるときに使うことができます。

ムービーは、URL パスを使うことができます。

ムービーが RAM にまだ読み込まれていない場合、結果は -1 になります。

パラメータ

stringMovieName 必須。メモリからアンロードするムービーの名前を指定するストリングです。

例文

次のステートメントは、メモリの連続する最大未使用ブロックが 100K 未満になっているかどうかをチェックし、100K 未満の場合は " パルジファル " ムービーをアンロードします。

```
-- Lingo syntax
if (_system.freeBlock < (100*1024)) then
    _movie.unLoadMovie("Parsifal")
end if

// JavaScript syntax
if (_system.freeBlock < (100*1024)) {
    _movie.unLoadMovie("Parsifal");
}
```

次のステートメントは、http://www.cbDemille.com/SunsetBlvd.dir にあるムービーをアンロードします。

```
-- Lingo syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir")

// JavaScript syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir");
```

関連事項

ムービー , unLoad() （メンバ）, unLoad() （ムービー）, unLoadMember()

unmute （ミクサ）

使用法

```
mixer.unmute()
```

解説

ミクサメソッド。ミクサ出力の volume プロパティを、mute() が呼び出される前の元の値に戻します。

例文

```
--Lingo syntax
on mouseUp me
     mixerRef.unmute() -- Unmutes the mixer.
end

// JavaScript syntax
function mouseUp(){
    mixerRef.unMute(); // Unmutes the mixer.
}
```

関連事項

mute（ミクサ）, ミクサ

unmute （サウンドオブジェクト）

使用法

```
soundObject.Unmute()
```

解説

サウンドオブジェクトメソッド。サウンドオブジェクトの volume プロパティを、mute() が呼び出される前の元の値に戻します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.unmute() -- Unmutes the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.unmute(); // Unmutes the sound object associated with soundObjRef.
}
```

関連事項

mute（サウンドオブジェクト）

unregisterAllEvents

使用法

```
-- Lingo syntax
member(whichMember).unregisterAllEvents()

// JavaScript syntax
member(whichMember).unregisterAllEvents();
```

解説

3D コマンド。すべてのイベント通知で参照されるキャストメンバの登録を解除します。このため、registerForEvent コマンドを使ってイベントに応答するよう登録していたすべてのハンドラは、そのイベントが発生しても起動しなくなります。

パラメータ

なし

例文

次のステートメントは、すべてのイベント通知に対して " シーン " というキャストメンバの登録を解除します。

```
-- Lingo syntax
member("Scene").unregisterAllEvents()

// JavaScript syntax
member("Scene").unregisterAllEvents();
```

関連事項

registerForEvent()

unregisterByteArrayCallback

使用法

```
soundObject.unRegisterByteArrayCallback(position)
```

パラメータ

パラメータ	解説	必須 / オプション
position	シンボルが渡されない場合にすべてのコールバックの登録を解除します。このパラメータで #preFilter または #postFilter シンボルを指定します。	オプション。

解説

バイト配列メソッド。バイト配列コールバックを登録解除します。

例文

```
--Lingo
so.unRegisterByteArrayCallback(#preFilter)
//JavaScript
so.unRegisterByteArrayCallback(#preFilter);
```

関連事項

registerByteArrayCallback

unregisterCuePointCallback

使用法

```
soundObject.unregisterCuePointCallback()
```

解説

サウンドオブジェクトメソッド。キューポイントコールバックを登録解除します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.unregisterCuePointCallback() -- Unregisters callback.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.unregisterCuePointCallback(); // Unregisters callback.
}
```

関連事項

registerCuePointCallback

unregisterEndOfSpoolCallback()

使用法

```
soundObject.unregisterEndOfSpoolCallback()
```

解説

サウンドオブジェクトメソッド。登録されている endofSpool コールバックを登録解除します。

例文

```
--Lingo syntax
on mouseUp me
     soundObjRef.unregisterEndOfSpoolCallback() -- Unregisters callback.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.unregisterEndOfSpoolCallback(); // Unregisters callback.
}
```

関連事項

registerEndOfSpoolCallback()

update

使用法

```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).model(whichModel).update();
```

解説

3D コマンド。モデル上のアニメーションをレンダリングせずに更新します。このコマンドを使用して、Lingo でアニメーションモデルの正確な位置を探すことができます。

パラメータ
なし

例文
```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).getPropRef("model",1).update();
```

updateFrame()

使用法
```
-- Lingo syntax
_movie.updateFrame()

// JavaScript syntax
_movie.updateFrame();
```

解説
ムービーメソッド。スコア生成セッション中でのみ動作する、スコアのレコード中に現行フレームに対して加えられた変更を確定して次のフレームに進みます。更新セッション開始時にフレーム内に存在していたすべてのオブジェクトが、そのままフレーム内に残ります。更新する各フレームごと updateFrame() メソッドを発行する必要があります。

パラメータ
なし

例文
次のハンドラで使われている updateFrame コマンドは、Lingo が繰り返しループの最後までくるたびに、現行フレームへの変更を確定して次のフレームに進みます。フレーム数はパラメータ numberOfFrames で定義されています。

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball").number
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

関連事項

beginRecording(), endRecording(), ムービー , scriptNum, tweened

updateStage()

使用法

```
-- Lingo syntax
_movie.updateStage()

// JavaScript syntax
_movie.updateStage();
```

解説

ムービーメソッド。フレーム間まで待たずに、ただちにステージを再描画します。

updateStage() メソッドは、スプライトの再描画、トランジションの実行、サウンドの再生、prepareFrame メッセージ（ムービースクリプトとビヘイビアースクリプトに影響）の送信、stepFrame メッセージ（actorList に影響）の送信の各処理を実行します。

パラメータ

なし

例文

次のハンドラは、スプライトの水平および垂直位置を変更し、ステージを再描画します。スプライトは、再生ヘッドが移動するのを待たずに、新たな位置に表示されます。

```
-- Lingo syntax
on moveRight(whichSprite, howFar)
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar
    _movie.updateStage()
end moveRight

// JavaScript syntax
function moveRight(whichSprite, howFar) {
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar;
    _movie.updateStage();
}
```

関連事項

actorList, ムービー , on prepareFrame, on stepFrame

URLEncode

使用法

```
URLEncode(proplist_or_string {, serverOSString} {, characterSet})
```

解説

最初のパラメータに対して URL でエンコードされたストリングを返す関数です。別のコマンドで CGI パラメータを使用することもできます。postNetText および getNetText() も、プロパティリストを指定すると同じ変換を行います。

パラメータ

propListOrString 必須。プロパティリストまたは URL でエンコードされたストリングを指定します。

serverOSString オプション。**propListOrString** のすべての改行文字をエンコードします。デフォルト値は "Unix" です。ただし、"Win" または "Mac" に設定することもできます。この場合、**propListOrString** に含まれる改行は、サーバ上で使用される文字に変換されます。ほとんどのアプリケーションでは、この設定は必要ありません。フォームでの応答では通常、改行を使用しないためです。

characterSet オプション。Shift-JIS（日本語）システムにのみ適用されます。選択できる設定としては、"JIS"、"EUC"、"ASCII"、"AUTO" があります。取り出されたデータは、シフト JIS から指定されたキャラクタセットに変換されます。返されたデータは、getNetText() の場合 (指定のキャラクタセットからシフト JIS へと変換) とまったく同様に処理されます。"AUTO" を指定した場合、ローカルのキャラクタセットでポストされたデータは変換されません。サーバから返されてきた結果は、getNetText() の場合と同じように変換されます。[キャラクタセット] を省略すると、"ASCII" がデフォルト設定になります。"ASCII" の場合、ポストや結果の変換は何も行われません。

例文

次の例では、URLEncode が、指定された位置にある CGI クエリに、URL でエンコードされたストリングを供給します。

```
URL = "http://aserver/cgi-bin/echoquery.cgi"
gotonetpage URL & "?" & URLEncode( [#name: "Ken", #hobby: "What?"] )
```

関連事項
getNetText(), postNetText

value()

使用法

```
value(stringExpression)
```

解説

ストリングの値を返す関数です。value() がコールされると、Lingo は与えられた [ストリング式] を解析して、その論理値を返します。

Lingo の式でメッセージウィンドウに put できるもの、または変数の値として指定できるものは、すべて value() でも使用できます。

次の 2 つの Lingo ステートメントは、同じ値になります。

```
put sprite(2).member.duration * 5
put value("sprite(2).member.duration * 5")
```

次の 2 つの Lingo ステートメントも、同じ値になります。

```
x = (the mouseH - 10) / (the mouseV + 10)
x = value("(the mouseH - 10) / (the mouseV + 10)")
```

Lingo が解析できない式を指定すると、予想できない結果が生じます。ただし、Lingo のエラーにはなりません。この場合の結果は、その式の先頭から、ストリング内の最初のシンタックスエラーまでの部分の値になります。

value() 関数は、エンドユーザがテキストフィールドに入力した式、Xtra 拡張から Lingo に渡されたストリング式、およびストリングから Lingo の値へ変換が必要な式を解析するときに役立ちます。

ユーザ入力に対して value() を使用すると危険な場合もあるので、注意が必要です。例えば、ユーザがフィールドにカスタムハンドラの名前を入力すると、それが value() に渡された時点で、そのハンドラが実行されてしまいます。

value 関数の行う操作を integer() 関数や float() 関数と混同しないように注意してください。

パラメータ

stringExpression 必須。値を返すストリングを指定します。このストリングは、Lingo で認識される式であればどのような式でもかまいません。

例文

次のステートメントは、"the sqrt of" && "2.0" というストリングの数値を表示します。

```
put value("the sqrt of" && "2.0")
```

この結果は 1.4142 になります。

次のステートメントは、"penny" というストリングの数値を表示します。

```
put value("penny")
```

"penny" には数値が含まれないので、メッセージウィンドウには "**VOID**" と表示されます。

次のシンタックスを使用すると、リストとしてフォーマットされたストリングを実際のリストに変換できます。

```
myString = "[" & QUOTE & "cat" & QUOTE & ", " & QUOTE & "dog" & QUOTE & "]"
myList = value(myString)
put myList
-- ["cat", "dog"]
```

これにより、リストがフィールドまたはテキストキャストメンバに置かれることによって、取り出して簡単にリストとしてフォーマットし直すことができます。

次のステートメントは "3 5" というストリングを解析して、Lingo が理解できる部分の値を返します。

```
put value("3 5")
-- 3
```

関連事項

string(), integer(), float()

vector()

使用法

```
-- Lingo syntax
vector()
vector(intX, intY, intZ)

// JavaScript syntax
vector();
vector(intX, intY, intZ);
```

解説

トップレベル関数およびデータ型。ベクトルは、3 つのパラメータで表される 3D スペース内のポイントです。これらのパラメータは、基準となる点から、それぞれ **x**、**y** および **z** 軸 上に沿った距離を示します。

ベクトルがワールドスペースにあるときは、基準点はワールドの原点の vector(0, 0, 0) になります。ベクトルがオブジェクトスペースにあるときは、基準点はオブジェクトの位置および向きになります。

このメソッドはベクトルオブジェクトを返します。

ベクトル値は、+、-、*、および / の各演算子を使用して演算できます。詳細については、個別の定義を参照してください。

パラメータ

intX オプション。**x** 軸のポイントを指定する整数です。

intY オプション。**y** 軸のポイントを指定する整数です。

intZ オプション。**z** 軸のポイントを指定する整数です。

例文

次のステートメントは、ベクトルを作成して、変数 myVector に設定します。

```
-- Lingo syntax
myVector = vector(10.0, -5.0, 0.0)

// JavaScript syntax
var myVector = vector(10.0, -5.0, 0.0);
```

Lingo の場合のみ、次のステートメントは、2 つのベクトルを足して、結果の値を変数 thisVector に設定します。

```
-- Lingo syntax
thisVector = vector(1.0, 0.0, 0.0) + vector(0.0, -12.5, 2.0)
```

version()

使用法

```
-- Lingo syntax
fileioObjRef.version()

// JavaScript syntax
fileioObjRef.version();
```

解説

FileIO メソッド。FileIO バージョンを表示してメッセージウィンドウの情報を構築します。

パラメータ

なし

例文

次のステートメントは、Director のバージョンをメッセージウィンドウに表示します。

```
-- Lingo
put(_player.productVersion)

// Javascript
trace(_player.productVersion)
```

関連事項

Fileio

voiceCount()

使用法

```
voiceCount()
```

解説

関数。音声読み上げエンジンで利用可能なインストール済みの音声の数を返します。戻り値は整数です。この数値は、特定の音声をアクティブにするために、voiceSet() と voiceGet() で使用できます。

パラメータ

なし

例文

次のステートメントは、音声読み上げで利用できる音声の数を変数 numVoices に設定します。

```
-- Lingo
numVoices = voiceCount()

// Javascript
Var numVoices = voiceCount();
```

関連事項

voiceInitialize(), voiceSet(), voiceGet()

voiceGet()

使用法

```
voiceGet()
```

解説

関数。音声読み上げで現在使用されている音声を示すプロパティリストを返します。このリストには、次のプロパティが含まれます。

- #name は、インストールされている音声の名前を示します。
- #age は、音声の年齢を示します。この値はストリングです。指定できる値は「Teen」、「Adult」、「Toddler」、および「Senior」と、「35」のような数値です。実際の値は、オペレーティングシステム、読み上げソフトウェア、およびインストールされている音声によって異なります。
- #gender は、音声が男性の声か女性の声かを示します。この値はストリングです。
- #index は、インストールされている音声のリストの中で特定の音声の位置を示します。voiceSet() コマンドを使用するときは、音声をインデックスで参照できます。

利用できる音声の数を知るには、voiceCount() を使用します。

パラメータ

なし

例文

次のステートメントは、現在の音声読み上げ用の音声を示すプロパティリストを変数 oldVoice に設定します。

```
-- Lingo
oldVoice = voiceGet()

// Javascript
Var oldVoice = voiceGet();
```

次のステートメントは、現在の音声読み上げ用の音声を示すプロパティリストを表示します。

```
-- Lingo
put voiceGet()
-- [#name: "Mary", #age: "teen", #gender: "female", #index: 5]

// Javascript
trace(voiceget())
// <[#name: "Mary", #age: "teen", #gender: "female", #index: 5]>
```

関連事項

voiceInitialize(), voiceCount(), voiceSet()

voiceGetAll()

使用法

```
voiceGetAll()
```

解説

関数。コンピュータにインストールされている利用可能な音声のリストを返します。リストは、利用できる各音声ごとのプロパティリストによって構成されます。

各プロパティリストには、次のプロパティが含まれます。

- #name は、インストールされている音声の名前を示します。
- #age は、音声の年齢を示します。この値はストリングです。指定できる値は「Teen」、「Adult」、「Toddler」、および「Senior」と、「35」のような数値です。実際の値は、オペレーティングシステム、読み上げソフトウェア、およびインストールされている音声によって異なります。
- #gender は、音声が男性の声か女性の声かを示します。
- #index は、インストールされている音声のリストの中で特定の音声の位置を示します。voiceSet() コマンドを使用するときは、音声をインデックスで参照できます。

利用できる音声の数を知るには、voiceCount() を使用します。

パラメータ

なし

例文

次のステートメントは、ユーザのコンピュータにインストールされている音声のリストを変数 currentVoices に設定します。

```
-- Lingo
currentVoices = voiceGetAll()

// Javascript
Var currentVoices = voiceGetAll();
```

次のステートメントは、現在インストールされている音声読み上げ用の各音声を示すプロパティリストを表示します。

```
-- Lingo
put voiceGetAll()
-- [[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age: "adult", #gender:
"male", #index: 2]]

// Javascript
trace(voiceGetAll());
// <[[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age: "adult", #gender:
"male", #index: 2]]>
```

関連事項

voiceInitialize(), voiceCount(), voiceSet(), voiceGet()

voiceGetPitch()

使用法

```
voiceGetPitch()
```

解説

関数。現在の音声の現在のピッチを整数で返します。有効な値の範囲は、オペレーティングシステムのプラットフォームと音声読み上げソフトウェアによって異なります。

パラメータ
なし

例文
次のステートメントは、現在の音声のピッチが 10 より大きいかどうかを調べ、大きいときは 10 に設定します。

```
-- Lingo syntax
if voiceGetPitch() > 10 then
    voiceSetPitch(10)
end if

// JavaScript syntax
if (voiceGetPitch() > 10) {
    voiceSetPitch(10);
}
```

関連事項
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetRate()

使用法
```
voiceGetRate()
```

解説
関数。音声読み上げエンジンの現在の再生レートを返します。戻り値は整数です。有効な値の範囲は、音声読み上げソフトウェアとオペレーティングシステムのプラットフォームによって異なります。一般的に、この値の範囲は -10 ～ 10 が適切です。

パラメータ
なし

例文
次のステートメントは、音声合成のレートが 50 より小さいかどうかを調べ、小さいときは 50 に設定します。

```
-- Lingo syntax
if voiceGetRate() < 50 then
    voiceSetRate(50)
end if

// JavaScript syntax
if (voiceGetRate() < 50) {
    voiceSetRate(50);
}
```

関連事項
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetVolume()

使用法

```
voiceGetVolume()
```

解説

関数。音声読み上げ合成の現在の音量を返します。戻り値は整数です。有効な値の範囲は、オペレーティングシステムのプラットフォームによって異なります。

パラメータ

なし

例文

次のステートメントは、音声合成の音量が 55 以上であるかどうかを調べ、55 未満の場合は 55 に設定します。

```
-- Lingo syntax
if voiceGetVolume() < 55 then
    voiceSetVolume(55)
end if

// JavaScript syntax
if (voiceGetVolume() < 55) {
    voiceSetVolume(55);
}
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceInitialize()

使用法

```
voiceInitialize()
```

解説

コマンド。コンピュータの音声読み上げエンジンをロードします。voiceInitialize() コマンドが 0 を返す場合は、音声読み上げソフトウェアが存在しないか、ロードに失敗したことを示します。

このコマンドは、ロードに成功すると 1 を返し、それ以外の場合は 0 を返します。

パラメータ

なし

例文

次のステートメントは、コンピュータの音声読み上げエンジンをロードします。次に、正しくロードされていることを調べ、voiceSpeak() コマンドを使用して "Shockwave にようこそ " というフレーズを読み上げます。

```
-- Lingo syntax
err = voiceInitialize()
if err = 1 then
    voiceSpeak("Welcome to Shockwave")
else
    alert "Text-to-speech software failed to load."
end if

// JavaScript syntax
err = voiceInitialize();
if (err == 1) {
    voiceSpeak("Welcome to Shockwave");
} else {
    alert("Text-to-speech software failed to load.");
}
```

関連事項

voiceCount(), voiceSet(), voiceGet()

voicePause()

使用法

```
voicePause()
```

解説

コマンド。コンピュータの音声読み上げエンジンへの音声出力を一時停止します。このコマンドは、停止に成功すると 1 を返し、それ以外の場合は 0 を返します。

パラメータ

なし

例文

次のステートメントは、ユーザがマウスをクリックしたときに、音声読み上げエンジンを一時停止させます。

```
-- Lingo syntax
on mouseUp
    voicePause()
end mouseUp

// JavaScript syntax
function mouseUp() {
    voicePause();
}
```

関連事項

voiceSpeak(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceResume()

使用法

```
voiceResume()
```

解説

コマンド。コンピュータの音声読み上げエンジンの音声出力を再開します。このコマンドは、停止に成功すると 1 を返し、それ以外の場合は 0 を返します。

パラメータ

なし

例文

次のステートメントは、再生ヘッドがスコア上の次のフレームに入ると、読み上げを再開します。

```
-- Lingo syntax
on exitFrame
    voiceResume()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceResume();
}
```

関連事項

voiceSpeak(), voicePause(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSet()

使用法

```
voiceSet(integer)
```

解説

関数。音声読み上げ合成の現在の音声を設定します。設定に成功すると、設定した新しい値が返されます。利用できる音声の数を知るには、voiceCount() を使用します。

パラメータ

integer 必須。関数。使用する音声読み上げ用の音声の番号を指定する整数です。有効な値の範囲は、ユーザのコンピュータにインストールされている音声の数によって異なります。範囲外の値を指定すると、それに最も近い有効な数値が使用されます。

例文

次のステートメントは、ユーザのコンピュータにインストールされている 3 番目の音声を音声読み上げの現在の音声として設定します。

```
voiceSet(3)
```

関連事項
voiceInitialize(), voiceCount(), voiceGet()

voiceSetPitch()

使用法
```
voiceSetPitch(integer)
```

解説
コマンド。音声読み上げエンジンの現在の音声のピッチを、指定した値に設定します。戻り値は、設定された新しいピッチの値です。

パラメータ
integer 必須。音声読み上げ用の音声のピッチを指定する整数です。有効な値の範囲は、オペレーティングシステムのプラットフォームと音声読み上げソフトウェアによって異なります。

例文
次のステートメントは、現在の音声のピッチを 75 に設定します。

```
-- Lingo
voiceSetPitch(75)

// Javascript
voiceSetPitch(75);
```

関連事項
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetRate()

使用法
```
voiceSetRate(integer)
```

解説
コマンド。音声読み上げエンジンの再生レートを、指定した整数値に設定します。このコマンドは、設定した新しい値を返します。

パラメータ
integer 必須。音声読み上げ用のエンジン使用の再生レートを指定する整数です。有効な値の範囲は、オペレーティングシステムのプラットフォームによって異なります。一般的に、ほとんどの音声読み上げ用ソフトウェアでは -10 ～ 10 の値が適切です。範囲外の値を指定すると、それに最も近い有効な数値が使用されます。

例文
次のステートメントは、音声読み上げエンジンの再生レートを 7 に設定します。

```
-- Lingo
voiceSetRate(7)

// Javascript
voiceSetRate(7);
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetVolume()

使用法

```
voiceSetVolume(integer)
```

解説

コマンド。音声読み上げ合成の音量を設定します。

パラメータ

integer 必須。音声読み上げ合成の音量を指定する整数です。有効な値の範囲は、オペレーティングシステムのプラットフォームによって異なります。設定に成功すると、設定した新しい値が返されます。無効な値を指定すると、それに最も近い有効な数値が使用されます。

例文

次のステートメントは、音声読み上げ合成の音量を 55 に設定します。

```
-- Lingo
voiceSetVolume(55)

// Javascript
voiceSetVolume(55);
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceState(), voiceWordPos()

voiceSpeak()

使用法

```
-- Lingo syntax
voiceSpeak("string")

// JavaScript syntax
voiceSpeak("string"); // documentation n/a
```

解説

コマンド。音声読み上げエンジンによって、指定したストリングを読み上げます。このコマンドを使用すると、実行中の読み上げは新しいストリングによって中断されます。

パラメータ

string 必須。音声読み上げエンジンによって、読み上げられるストリングです。

例文

次のステートメントは、音声読み上げエンジンによって "Shockwave にようこそ " というフレーズを読み上げます。

```
voiceSpeak("Welcome to Shockwave")
```

関連事項

voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceState()

使用法

```
-- Lingo syntax
voiceState()

// JavaScript syntax
voiceState(); // documentation n/a
```

解説

関数。音声の現在のステータスをシンボルで返します。戻り値は、#playing、#paused、または #stopped です。

パラメータ

なし

例文

次のステートメントは、音声読み上げエンジンが現在読み上げているかどうかを調べ、読み上げていないときは音声を 1 に設定します。

```
--Lingo syntax
if voiceState() <> #playing then
    voiceSet(1)
end if

// JavaScript syntax
if (voiceState() != symbol("playing")) {
    voiceSet(1);
}
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceWordPos(), voiceSpeak()

voiceStop()

使用法

```
-- Lingo syntax
voiceStop()

// JavaScript syntax
voiceStop(); // documentation n/a
```

解説

コマンド。コンピュータの音声読み上げエンジンへの音声出力を停止し、音声読み上げバッファを空にします。このコマンドは、停止に成功すると 1 を返し、それ以外の場合は 0 を返します。

パラメータ

なし

例文

次のステートメントは、再生ヘッドがスコア上の次のフレームに入ると、読み上げを停止します。

```
-- Lingo syntax
on exitFrame
    voiceStop()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceStop();
}
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos(), voiceSpeak()

voiceWordPos()

使用法

```
-- Lingo syntax
voiceWordPos()

// JavaScript syntax
voiceWordPos(); // documentation n/a
```

解説

関数。ストリング全体の中で、現在読み上げられている単語の位置を示す整数を返します。例えば、キャストメンバに 15 個の単語が含まれる場合、5 番目の単語を読み上げているときにこの関数を使用すると、戻り値は 5 になります。

パラメータ

なし

例文

次のステートメントは、" こんにちは、お元気ですか " という文を読み上げ、現在読み上げ中の単語の位置をメッセージウィンドウに表示します。voiceSpeak() コマンドの直後に voiceWordPos() 関数を呼び出しているので、戻り値は 1 になります。

```
-- Lingo syntax
voiceSpeak("Hello, how are you?")
put voiceWordPos()
-- 1

// JavaScript syntax
voiceSpeak("Hello, how are you?");
put(voiceWordPos());
// 1
```

関連事項

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceSpeak()

voidP()

使用法

```
-- Lingo syntax
voidP(variableName)

// JavaScript syntax
variableName == null
```

解説

関数。指定した変数に値があるかどうかを示します。変数に値がないか、変数の値が VOID である場合は、この関数は TRUE を返します。変数の値が VOID 以外である場合、この関数は FALSE を返します。

パラメータ

variableName 必須。テストする変数を指定します。

例文

次のステートメントは、変数 answer に初期値が与えられているかどうかをチェックしメッセージウィンドウに表示します。

```
-- Lingo syntax
put voidP(answer)

// JavaScript syntax
put(answer == null));
```

関連事項

ilk(), VOID

window()

使用法

```
-- Lingo syntax
window(stringWindowName)

// JavaScript syntax
window(stringWindowName);
```

解説

最上位レベルの関数。指定したウィンドウへの参照を返します。

指定したウィンドウに Director ムービーが含まれている必要があります。

ムービーを再生するウィンドウは、フローティングパレット、個別の制御パネル、異なる形状のウィンドウを作成するのに便利です。また、複数のムービーを同時に開いておき、相互にやり取りさせることもできます。

パラメータ

stringWindowName 必須。参照するウィンドウの名前を指定するストリングです。

例文

次のステートメントは、変数 myWindow を Collections という名前のウィンドウに設定します。

```
-- Lingo syntax
myWindow = window("Collections")

// JavaScript syntax
var myWindow = window("Collections");
```

関連事項

ウィンドウ

WindowOperation

使用法

```
WindowOperation(MUIObject, operation)
```

解説

このコマンドは、汎用目的ダイアログボックスのウィンドウを制御します。

operation パラメータをウィンドウが何を実行するかを決定する値に置換します。指定できる値とその結果は以下のとおりです。

使用できる値	結果
#show	非モーダルダイアログボックスのみを表示します（モーダルダイアログボックスを開くには、Run コマンドを使用します）。
#hide	非モーダルダイアログボックスを非表示にします（モーダルダイアログボックスを閉じるには、Stop コマンドを使用します）。
#center	ウィンドウをモニタ画面の中央に配置します。

使用できる値	結果
#zoom	ユーザがウィンドウ上のズームボックスをクリックしたことを示すメッセージを送信します。ユーザがズームボックスをクリックした後にウィンドウのサイズを変更する場合は、コールバックハンドラでダイアログボックスのサイズを変更する必要があります。
#tipsOn	ツールヒントをオンにします（今後のバージョンの MUI Xtra のために予約されています）。
#tipsOff	ツールヒントをオフにします（今後のバージョンの MUI Xtra のために予約されています）。

例文

次のハンドラは、MUIObject が存在するかどうかをチェックし、存在する場合はダイアログボックスを表示します。

```
--Lingo syntax
on showDialog
    global MUIObject
    if objectP( MUIObject ) then
        WindowOperation( MUIObject, #show )
    end if
end showDialog
```

このステートメントは、MUIObject から作成されたダイアログボックスを非表示にします。

```
WindowOperation(MUIObject, #hide)
```

windowPresent()

使用法

```
-- Lingo syntax
_player.windowPresent(stringWindowName)

// JavaScript syntax
_player.windowPresent(stringWindowName);
```

解説

プレーヤメソッド。**stringWindowName** で指定したオブジェクトがウィンドウ内ムービーとして機能しているか（TRUE）、ウィンドウ内ムービーではないか（FALSE）を示します。

開いているウィンドウがある場合、windowPresent() はそのウィンドウが windowList プロパティから削除されない限り、TRUE のままです。

stringWindowName パラメータには、windowList プロパティ内に示されるウィンドウ名を指定する必要があります。

パラメータ

stringWindowName 必須。テストするウィンドウの名前を指定するストリングです。

例文

次のステートメントは、"myWindow" オブジェクトがウィンドウ内ムービー (MIAW) かどうかをチェックし、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_player.windowPresent(myWindow))

// JavaScript syntax
put(_player.windowPresent(myWindow));
```

関連事項

プレーヤ , windowList

worldSpaceToSpriteSpace

使用法

```
-- Lingo syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector)

// JavaScript syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector);
```

解説

3D コマンド。指定されたワールド相対位置の、カメラの矩形内でのポイントを返します。このコマンドで返される位置は、カメラの矩形の左上隅を基準とします。

指定された位置がカメラの視野外にある場合は、void を返します。

パラメータ

vector 必須。表示されるワールド相対位置を指定します。

例文

次のステートメントは、vector (0, 0, 0) で指定されるワールドの原点が、カメラの矩形内の point (250,281) に表示されることを示します。

```
-- Lingo syntax
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0, 0, 0))
-- point(250, 281)

// JavaScript syntax
put(sprite(5).camera.worldSpaceToSpriteSpace(vector(0,0,0)));
```

関連事項

spriteSpaceToWorldSpace, rect (camera)

writeBoolean

使用法

```
byteArrayObject.writeBoolean(boolVal)
```

解説

バイト配列メソッド。バイト配列にブール値を書き込みます。ブール値は 1 バイトを使用します。バイト値がゼロ以外値の場合に 1 を書き込み、それ以外の場合にゼロを書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
boolVal	バイト配列に書き込むブール値。	必須

例文

```
--Lingo syntax
put bArray.writeBoolean(1)

//JavaScript syntax
put(bArray.writeBoolean(1));
```

writeByteArray

使用法

```
writeByteArray(byteArray, [intOffset],[ intLen])
```

解説

バイト配列メソッド。バイト配列またはその一部を別のバイト配列に書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
byteArray	既存のバイト配列。	必須
intOffset	開始オフセット位置。	オプション
intLen	バイト配列の長さ。	オプション

例文

```
--Lingo syntax
bArray.writeByteArray(gByteArray, 1, 10)

//JavaScript syntax
bArray.writeByteArray(gByteArray, 1, 10);
```

writeByteArray (FileIO Xtra)

使用法

```
FileIO.writeByteArray(byteArray, [intOffset],[intLen])
```

解説

FileIO Xtra メソッド。バイト配列またはその一部をファイルに書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
byteArray	既存のバイト配列。	必須
intOffset	開始オフセット位置。	オプション
intLen	バイト配列の長さ。	オプション

例文

```
--Lingo syntax
FileIO.writeByteArray(gByteArray, 1, 10)

//JavaScript syntax
FileIO.writeByteArray(gByteArray, 1, 10);
```

writeChar()

使用法

```
-- Lingo syntax
fileioObjRef.writeChar(stringChar)

// JavaScript syntax
fileioObjRef.writeChar(stringChar)
```

解説

FileIO メソッド。1 つの指定の ASCII 文字をファイルに書き込みます。

文字の書き込みに writeChar() を使用する前に、まず openFile() を呼び出してファイルを開いておく必要があります。

パラメータ

stringChar 必須。ファイルに書き込む ASCII 文字を指定します。

例文

次のステートメントは、読み取りおよび書き込みのアクセス権を持つ c:\xtra.txt ファイルを開き、そのファイルに「d」の文字を書き込み、ファイルを閉じます。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeChar("d")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeChar("d");
objFileio.closeFile();
```

関連事項

Fileio

writeFloat32

使用法

```
byteArrayObject.writeFloat32(floatVal)
```

解説

バイト配列メソッド。32 ビットの浮動小数点数値をバイト配列に書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
floatVal	バイト配列に書き込む浮動小数点数値。	必須

例文

```
--Lingo syntax
put bArray.writeFloat32(3.3)

//JavaScript syntax
put(bArray.writeFloat32(3.3));
```

writeFloat64

使用法

```
byteArrayObject.writeFloat64(floatVal)
```

解説

バイト配列メソッド。64 ビットの浮動小数点数値をバイト配列に書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
floatVal	バイト配列に書き込む浮動小数点数値。	必須

例文

```
--Lingo syntax
put bArray.writeFloat64(36666666.366666)

//JavaScript syntax
put(bArray.writeFloat64(36666666.366666));
```

writeInt8

使用法

```
byteArrayObject.writeInt8(intVal)
```

解説

バイト配列メソッド。8 ビットの符号付き整数値をバイト配列に書き込みます。入力値を 8 ビットで表現できない場合、メソッドは最下位バイトを処理します。

パラメータ

パラメータ	解説	必須 / オプション
intVal	バイト配列に書き込む整数値。	必須

例文

```
--Lingo syntax
put bArray.writeInt8(16)

//JavaScript syntax
put(bArray.writeInt8(16));
```

writeInt16

使用法

```
byteArrayObject.writeInt16(intVal)
```

解説

バイト配列メソッド。16 ビットの符号付き整数値をバイト配列に書き込みます。入力値を 16 ビットで表現できない場合、メソッドは最下位バイトから処理します。

パラメータ

パラメータ	解説	必須 / オプション
intVal	バイト配列に書き込む整数値。	必須

例文

```
--Lingo syntax
put bArray.writeInt16(1234)

//JavaScript syntax
put(bArray.writeInt16(1234));
```

writeInt32

使用法

```
byteArrayObject.writeInt32(intVal)
```

解説

バイト配列メソッド。32 ビットの符号付き整数値をバイト配列に書き込みます。

パラメータ

パラメータ	解説	必須 / オプション
intVal	バイト配列に書き込む整数値。	必須

例文

```
--Lingo syntax
put bArray.writeInt32(56781234)

//JavaScript syntax
put(bArray.writeInt32(56781234));
```

writeRawString

使用法

```
writeRawString(strValue, intLen,[strCharSet])
```

解説

バイト配列メソッド。固定バイト数を書き込みます。必要な場合は、NULL 文字がテキストに追加されます。

パラメータ

パラメータ	解説	デフォルト値
strValue	バイト配列に書き込むストリング。	必須
intLen	バイト配列から読み取るストリングの長さ。	必須
strCharSet	必要なエンコーディング変換を指定します。	UTF-8

例文

```
--Lingo syntax
bArray.writeRawString("Director",13)

//JavaScript syntax
bArray.writeRawString("Director",13);
```

writeReturn()

使用法

```
-- Lingo syntax
fileioObjRef.writeReturn(symbol(""))

// JavaScript syntax
fileioObjRef.writeReturn(symbol(""));
```

解説

FileIO メソッド。ファイルに改行を挿入します。

パラメータ

なし

例文

次のステートメントは、読み取りおよび書き込みのアクセス権を持つ c:\xtra.txt ファイルを開き、ファイルに「1 行目」の行を書き込み、その後に改行および 2 行目「2 行目」を続けてから、ファイルを閉じます。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeReturn(#windows)
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeReturn(Symbol("windows"));
objFileio.writeString("Second Line");
objFileio.closeFile();
```

関連事項

Fileio

writeString()

使用法

```
-- Lingo syntax
fileioObjRef.writeString(string)

// JavaScript syntax
fileioObjRef.writeString(string)
```

解説

FileIO メソッド。null 終端ストリングをファイルに書き込みます。

パラメータ

string 必須。ファイルに書き込むストリングです。

例文

次のステートメントは、読み取りおよび書き込みのアクセス権を持つ c:\xtra.txt ファイルを開き、ファイルに「1 行目」の行および「2 行目」の 2 行目を書き込んでから、ファイルを閉じます。

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeString("Second Line");
objFileio.closeFile();
```

関連事項

Fileio

writeString (byte array)

使用法

```
writeString(strValue, [strCharSet])
```

解説

バイト配列メソッド。バイト配列にストリングを書き込みます。ストリングの BLOB の先頭 4 バイトに、ストリングの長さが格納されています。

パラメータ

パラメータ	解説	デフォルト値
strValue	バイト配列に書き込むストリング。	必須
strCharSet	必要なエンコーディング変換を指定します。	UTF-8

例文

```
--Lingo syntax
bArray.writeString("Director")

//JavaScript syntax
bArray.writeString("Director");
```

xtra()

使用法

```
-- Lingo syntax
xtra(xtraNameOrNum)

// JavaScript syntax
xtra(xtraNameOrNum);
```

解説

最上位レベルの関数。指定した Xtra のインスタンスを返します。

指定した Xtra が検出されない場合は空のオブジェクトへの参照が返されます。

完成したムービーでの xtra の実例は、Director アプリケーションフォルダの Learning/Lingo フォルダ内の text.dir ムービーを参照してください。

パラメータ

xtraNameOrNum 必須。返される Xtra の名前を指定するストリング、または返される Xtra のインデックス位置を指定する整数です。ストリング名では大文字と小文字を区別しません。

例文

次のステートメントは、変数 myNetLingo を NetLingo Xtra 拡張機能に設定します。

```
-- Lingo syntax
myNetLingo = xtra("netlingo")

// JavaScript syntax
var myNetLingo = xtra("netlingo");
```

zoomBox

使用法

```
-- Lingo syntax
zoomBox startSprite, endSprite {,delayTicks}

// JavaScript syntax
zoomBox(startSprite, endSprite {,delayTicks}); // not yet documented
```

解説

Mac の Finder でウィンドウを開いたときのように、ズーム効果を加えるためのコマンドです。ズーム効果は、指定した開始スプライトの境界矩形から指定した最終スプライトの境界矩形までの間で実行されます。zoomBox 実行時には、次の論理が適用されます。

1 現在のフレーム内で [最終スプライト] を探す。それ以外の場合、

2 次のフレーム内で [最終スプライト] を探す。

ただし、[開始スプライト] と [最終スプライト] が同じチャネル内にあると、zoomBox コマンドは動作しません。

パラメータ

startSprite 必須。開始スプライトを指定します。

endSprite 必須。最終スプライトを指定します。

delayTicks オプション。ズーム矩形の各段階での動きの間隔をチック数で指定します。[ディレイチック数] を特に指定していない場合、ディレイは 1 になります。

例文

次のステートメントは、スプライト 7 ～ 3 の間でズーム効果を作成します。

```
-- Lingo syntax
zoomBox 7, 3

// JavaScript syntax
zoomBox(7, 3); // not yet documented
```

第 13 章：演算子

この項には、Director® で使用できるすべての演算子のアルファベット順リストが記載されています。

次に示す演算子の大部分は Lingo にのみ適用されます。JavaScript シンタックスには、この項に示されている Lingo の演算子に類似しているか同じである演算子がいくつか含まれています。そのため該当する場合は、Lingo の演算子の機能と、JavaScript シンタックスにある最も近い相当物を対応させることができるように、JavaScript シンタックスの使用法と例が記載されています。JavaScript シンタックスの演算子の詳細については、4 ページの「Director スクリプトに関する重要事項」を参照してください。

(シンボル)

使用法

```
--Lingo syntax
#symbolName

// JavaScript syntax
symbol("symbolName");
```

解説

シンボルを定義するためのシンボル演算子です。シンボルは、状態やフラグを表すために単独で使用することができます。[シンボル名] は先頭が英字で、その後に英数字が続きます。

シンボルを使った有効な操作は次のとおりです。

- 値が変数に割り当てられます。
- ストリング、整数、矩形、ポイント同士で比較されます。
- パラメータとしてハンドラまたはメソッドに渡されます。
- 値としてハンドラまたはメソッドから返されます。

シンボルは、ストリングに比べると必要なメモリが少なく、高速に処理されます。また、ストリングのようなキャラクタの集合ではなく、別のデータ型ですが、string 関数を使用すると、シンボルをストリングに変換して表示させることができます。

以下は、シンボルのシンタックスに関する規則です。

- 大文字と小文字を区別しません。
- 最初の文字には数字を使用できません。
- スペースは使用できませんが、アンダースコアで代用できます。
- シンボルには、128 個の ASCII 文字を使用できます。アクセント記号が付いた文字は、ベースの文字として扱われます。
- ピリオドは使用できません。

シンボル、グローバル変数、およびグローバル変数に割り当てられたパラメータ名は、すべて共通の参照テーブルに保存されます。

例文

次のステートメントは、変数 state に #Playing を割り当てます。

```
-- Lingo syntax
state = #Playing

// JavaScript syntax
var state = symbol("Playing");
```

関連事項

ilk(), string(), symbol(), symbolP()

.（ドット演算子）（どっとえんざんし）

使用法

```
-- Lingo syntax
objectReference.objectProperty
textExpression.objectProperty
object.commandOrFunction()

// JavaScript syntax
objectReference.objectProperty;
textExpression.objectProperty;
object.commandOrFunction();
```

解説

オブジェクトのプロパティの確認や設定、コマンドの発行、関数の実行に使う演算子です。キャストメンバ、スプライト、プロパティリスト、親スクリプトの子オブジェクト、ビヘイビアーなどをオブジェクトとして扱うことができます。

例文

次のステートメントは、チャネル 10 のスプライトに含まれる現行のメンバを表示します。

```
-- Lingo syntax
put(sprite(10).member)

// JavaScript syntax
put(sprite(10).member);
```

関数の呼び出しに、次のようなシンタックスも使用できます。

```
-- Lingo syntax
myColorObject = color(124, 22, 233)
put(myColorObject.ilk())
-- #color

// JavaScript syntax
var myColorObject = color(124, 22, 233);
put(myColorObject.ilk());
// #color
```

- (マイナス記号)

使用法

```
-- Lingo syntax
(Negation): -expression
(Subtraction): expression1 - expression2

// JavaScript syntax
(Negation): -expression
(Subtraction): expression1 - expression2
```

解説

符号反転に使用した場合、式の値の正と負の符号を反転させる演算子です。減算に使用した場合、2 つの数式の差を求める演算子で、式 **1** から 式 **2** を引きます。

符号反転に使用した場合、-（マイナス）演算子の評価優先度は 5 です。

減算に使用し、両方の数式が整数の場合には結果も整数になります。いずれか一方または両方の式が浮動小数の場合には結果が浮動小数になります。この場合、-（マイナス）演算子の評価優先度は 3 です。

例文

（符号反転）：次のステートメントは、式 2 + 3 で得られる解の正と負の符号を逆にします。

```
-- Lingo syntax
put(-(2 + 3))

// JavaScript syntax
put(-(2 + 3));
```

この結果は -5 になります。

（減算）: 次のステートメントは、整数 5 から 2 を引き、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(5 - 2)

// JavaScript syntax
put(5 - 2);
```

結果は整数の 3 になります。

（減算）: 次のステートメントは、浮動小数 3.25 から 1.5 を引き、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(3.25 - 1.5)

// JavaScript syntax
put(3.25 - 1.5);
```

結果は浮動小数の 1.75 になります。

-- (comment)

使用法

```
-- Lingo syntax
-- comment

// JavaScript syntax
// comment
```

解説

ハイフンを 2 つ続けたコメント記号は、コメントの開始を示します。コメント記号と行末の改行文字の間にあるものはすべてコメントとして解釈され、Lingo の処理からは無視されます。

例文

次のハンドラでは、2 つのハイフンを使って 2、3、4 行目にコメントを入れています。

```
-- Lingo syntax
on resetColors
    -- This handler resets the sprite's colors.
    sprite(1).forecolor = 35
    -- bright red
    sprite(1).backcolor = 36
    -- light blue
end

// JavaScript syntax
function resetColors() {
    // this handler resets the sprite's colors
    sprite(1).forecolor = 35;
    // bright red
    sprite(1).backcolor = 36;
    // light blue
}
```

&、+（連結演算子）

使用法

```
-- Lingo syntax
expression1 & expression2

// JavaScript syntax
expression1 + expression2
```

解説

2 つの式を 1 つのストリングに連結します。[式 **1**] または [式 **2**] が数値の場合には、まず数値がストリングに変換されます。結果はストリングになります。

このストリング演算子の評価優先度は 2 です。

パラメータを 1 つだけ取り込む Lingo コマンドや関数の中には、パラメータを括弧で囲まずに記述できるものがあります。通常は問題にはなりませんが、パラメータに演算子が含まれている場合は、最初の演算子の前にあるストリングのみがパラメータとして解釈されます。

このような問題を防ぐためには、演算子を含む文全体を括弧で囲みます。このように、括弧の位置によって Lingo が処理する演算子の優先順位を明確にし、演算子をはさんで記述されたパラメータを 1 つのパラメータとして処理することができるようになります。

例文

次のステートメントは、2 つのストリング「アブラ」と「カダブラ」を連結し、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put("abra" & "cadabra")

// JavaScript syntax
put("abra" + "cadabra");
```

結果は「アブラカダブラ」という 1 つのストリングになります。

次のステートメントは、ストリング「$」と変数 price の内容を連結して 1 つのストリングにした結果を「価格」というフィールドキャストメンバに割り当てます。

```
-- Lingo syntax
member("Price").text = "$" & price

// JavaScript syntax
member("Price").text = "$" + price;
```

関連事項

&&、+（連結演算子）

&&、+（連結演算子）

使用法

```
-- Lingo syntax
expression1 && expression2

// JavaScript syntax
expression1 + expression2
```

解説

2 つの式の間に半角スペース文字を挿入して 1 つのストリングに連結するためのストリング演算子です。**[式 1]** または **[式 2]** が数値の場合には、まず数値がストリングに変換されます。結果はストリングになります。

このストリング演算子の評価優先度は 2 です。

例文

次のステートメントは、2 つのストリング「アブラ」と「カダブラ」の間に半角スペースを挿入して連結します。

```
-- Lingo syntax
put("abra" && "cadabra")

// JavaScript syntax
put("abra " + "cadabra");
```

結果は「アブラカダブラ」という 1 つのストリングになります。

次のステートメントは、" 今日の日付 :" というストリングと実際の日付 (ロングバージョン) の間に半角スペースを挿入して連結します。

```
-- Lingo syntax
put("Today is" && date())

// JavaScript syntax
put("Today is " + Date());
```

関連事項

&、+（連結演算子）

() (括弧かっこ)

使用法

```
-- Lingo syntax
(expression)

// JavaScript syntax
(expression)
```

解説

式をグループ化する演算子で、式内の演算子の処理順序を制御するときに使用します。演算子の評価優先度に基づいた順序よりも優先され、括弧内にある式が先に処理されます。括弧がネストされている (括弧の中に別の括弧がある) 場合には、内側の括弧の式が先に処理されます。

この演算子の評価優先度は 5 です。

パラメータを 1 つだけ取り込む Lingo コマンドや関数の中には、パラメータを括弧で囲まずに記述できるものがあることに注意してください。通常は問題にはなりませんが、パラメータに演算子が含まれている場合は、最初の演算子の前にあるストリングのみがパラメータとして解釈されます。

例えば、open window コマンドがその例としてあげられます。このコマンドは、どのウィンドウを開くのかを指定するパラメータを 1 つだけ認識するため、パス名とファイル名を定義するときに演算子 & を使うと、& の手前にあるストリングのみがファイル名として解釈されます。例えば、open window the applicationPath & "theMovie" というステートメントは、(open window the applicationPath) & ("theMovie") というステートメントとして処理されます。このような問題を防ぐためには、演算子を含む文全体を括弧で囲みます。ステートメントを次のように記述すると正しい処理が実行されるようになります。

```
-- Lingo syntax
open window (the applicationPath & "theMovie")

// JavaScript syntax
window(the applicationPath + "theMovie").open();
```

例文

次のステートメントでは、グループ演算子を使って処理順序を変えたときの例です。結果は、各ステートメントの下に示されています。

```
-- Lingo syntax
put((2 + 3) * (4 + 5))
-- 45
put(2 + (3 * (4 + 5)))
-- 29
put(2 + 3 * 4 + 5)
-- 19

// JavaScript syntax
put((2 + 3) * (4 + 5));
// 45
put(2 + (3 * (4 + 5)));
// 29
put(2 + 3 * 4 + 5);
// 19
```

*（乗算）

使用法

```
-- Lingo syntax
expression1 * expression2

// JavaScript syntax
expression1 * expression2
```

解説

2 つの数式の積を求める演算子です。両方の式が整数の場合には積も整数になり、いずれか一方または両方の式が浮動小数の場合には積が浮動小数になります。

この演算子の評価優先度は 4 です。

例文

次のステートメントは、整数 2 と 3 を掛け合わせ、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(2 * 3)

// JavaScript syntax
put(2 * 3);
```

結果は整数の 6 になります。

次のステートメントは、浮動小数 2.0 と 3.1414 を掛け合わせ、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(2.0 * 3.1416)

// JavaScript syntax
put(2.0 * 3.1416);
```

結果は浮動小数の 6.2832 になります。

+ (加算)

使用法

```
-- Lingo syntax
expression1 + expression2

// JavaScript syntax
expression1 + expression2
```

解説

2 つの数式の和を求める演算子です。両方の式が整数の場合には和も整数になり、いずれか一方または両方の式が浮動小数の場合には結果が浮動小数になります。

この演算子の評価優先度は 4 です。

例文

次のステートメントは、整数 2 と 3 を足して、その結果の整数 5 をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(2 + 3)
// JavaScript syntax
put(2 + 3);
```

次のステートメントは、浮動小数 2.5 と 3.25 を足して、その結果の浮動小数 5.7500 をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(2.5 + 3.25)
// JavaScript syntax
put(2.5 + 3.25);
```

+（加算）（3D）

使用法

```
-- Lingo syntax
vector1 + vector2
vector + scalar
```

解説

3D ベクトル演算子。2 つのベクトルのコンポーネントを足します。または、ベクトルの各コンポーネントにスカラー値を足し、新しいベクトルを返します。

ベクトル **1** + ベクトル **2** は、ベクトル **1** のコンポーネントを ベクトル **2** の対応するコンポーネントに足して、新しいベクトルを返します。

ベクトル + スカラー は、ベクトルの各コンポーネントにスカラー値を足して、新しいベクトルを返します。

-（マイナス）（3D）

使用法

```
-- Lingo syntax
vector1 - vector2
vector - scalar
```

解説

3D ベクトル演算子。ベクトル **2** のコンポーネントを、対応する ベクトル **1** のコンポーネントから引きます。または、ベクトルの各コンポーネントからスカラー値を引いて、新しいベクトルを返します。

ベクトル **1** - ベクトル **2** は、ベクトル **1** の各コンポーネントから ベクトル **2** の対応するコンポーネントの値を引いて、新しいベクトルを返します。

ベクトル - スカラー は、ベクトルの各コンポーネントからスカラー値を引いて、新しいベクトルを返します。

*（乗算演算子（3D））

使用法

```
-- Lingo syntax
vector1 * vector2
vector * scalar
transform * vector
```

解説

3D ベクトル演算子。ベクトル **1** のコンポーネントにベクトル **2** の対応するコンポーネントを掛けて、内積を返します。または、ベクトルの各コンポーネントにスカラー値を掛けて、新しいベクトルを返します。

ベクトル **1** * ベクトル **2** は、2 つのベクトルの内積 (新しいベクトルではない) を返します。この演算は、ベクトル **1**.dotproduct(ベクトル **2**) と同じです。

ベクトル * スカラーは、ベクトルの各コンポーネントにスカラー値を掛けて、新しいベクトルを返します。

トランスフォーム * ベクトル は、トランスフォーム に ベクトルを掛けて、新しいベクトルを返します。新しいベクトルは、トランスフォーム で定義された位置および回転の変更を ベクトル に適用した結果です。ベクトル * トランスフォーム はサポートされていないことに注意してください。

関連事項

dotProduct()

/ (除算)

使用法

```
-- Lingo syntax
expression1 / expression2

// JavaScript syntax
expression1 / expression2
```

解説

2 つの数式の商を求める演算子で、[式 **1**] を [式 **2**] で割ります。両方の式が整数の場合には商も整数になり、いずれか一方または両方の式が浮動小数の場合には結果が浮動小数になります。

この演算子の評価優先度は 4 です。

例文

次のステートメントは、整数 22 を 7 で割り、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(22 / 7)

// JavaScript syntax
put(22 / 7);
```

結果は整数の 3 になります。実際には割り切れない値ですが、2 つの式が整数であるため、四捨五入されて整数値が返されます。

次のステートメントは、浮動小数 22.0 を 7.0 で割り、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(22.0 / 7.0)

// JavaScript syntax
put(22.0 / 7.0);
```

結果は浮動小数の 3.1429 になります。

/（除算）（3D）

使用法

```
-- Lingo syntax
vector / scalar
```

解説

3D ベクトル演算子。ベクトルの各コンポーネントをスカラー値で割り、新しいベクトルを返します。

< (より小さい)

使用法

```
-- Lingo syntax
expression1 < expression2

// JavaScript syntax
expression1 < expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] が [式 **2**] より小さい場合の結果は TRUE になり、[式 **1**] が [式 **2**] 以上の場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。矩形やポイントの比較では、エレメントがリストになっているものとして、エレメントごとに比較されます。

この演算子の評価優先度は 1 です。

<= (以下)

使用法

```
-- Lingo syntax
expression1 <= expression2

// JavaScript syntax
expression1 <= expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] が [式 **2**] 以下の場合の結果は TRUE) になり、[式 **1**] が [式 **2**] より大きい場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。矩形やポイントの比較では、エレメントがリストになっているものとして、エレメントごとに比較されます。

この演算子の評価優先度は 1 です。

<> (等しくない)

使用法

```
-- Lingo syntax
expression1 <> expression2

// JavaScript syntax
expression1 != expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] と [式 **2**] が等しくない場合の結果は TRUE になり、[式 **1**] と [式 **2**] が等しい場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。矩形やポイントの比較では、エレメントがリストになっているものとして、エレメントごとに比較されます。

この演算子の評価優先度は 1 です。

= (等号)

使用法

```
-- Lingo syntax
expression1 = expression2

// JavaScript syntax
expression1 = expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] と [式 **2**] が等しい場合の結果は TRUE になり、等しくない場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。

リストに含まれたエレメントの数をもとにリスト同士を比較します。エレメントの数が多い方のリストを、エレメントの少ない方のリストより大きいと見なします。

この演算子の評価優先度は 1 です。

> (より大きい)

使用法

```
-- Lingo syntax
expression1 > expression2

// JavaScript syntax
expression1 > expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] が [式 **2**] より大きい場合の結果は TRUE になり、[式 **1**] が [式 **2**] 以下の場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。矩形やポイントの比較では、エレメントがリストになっているものとして、エレメントごとに比較されます。

この演算子の評価優先度は 1 です。

>= (以上)

使用法

```
-- Lingo syntax
expression1 >= expression2

// JavaScript syntax
expression1 >= expression2
```

解説

2 つの式を比較する演算子です。[式 **1**] が [式 **2**] 以上の場合の結果は TRUE になり、[式 **1**] が [式 **2**] より小さい場合には FALSE になります。

この演算子は、ストリング、整数、浮動小数、矩形、ポイントを比較することができます。矩形やポイントの比較では、エレメントがリストになっているものとして、エレメントごとに比較されます。

この演算子の評価優先度は 1 です。

[ ] (ブラケットアクセス)

使用法

```
-- Lingo syntax
textExpression[chunkNumberBeingAddressed]
textExpression[firstChunk..lastChunk]
```

解説

チャンク式の内容を数字で指定して参照する演算子です。式の **n** 番目のチャンクを見つける場合に使用できます。単語、行、文字、段落などのテキストキャストメンバチャンクに適用できます。

例文

次のステートメントはテキストキャストメンバ " 名前 " の 3 行目にある 1 番目の単語を表示します。

```
-- Lingo syntax
put(member("First Names").text.line[3].word[1])

// JavaScript syntax
put(member("First Names").getPropRef("line", 1).getProp("word", 1));
```

[ ] (list)

使用法

```
[entry1, entry2, entry3, ...]
```

解説

角括弧（ブラケット）内の項目がリストであることを示す演算子です。次の 4 種類のリストがあります。

- 未ソートのリニアリスト
- ソート済みのリニアリスト
- 未ソートのプロパティリスト
- ソート済みのプロパティリスト

リニアリストの場合、各項目は関連付けられたプロパティを含まない単独の値です。一方、プロパティリスト内の各項目は、プロパティと値のペアで構成されています。この場合プロパティは値の前に入れ、コロンでプロパティと値を区切ります。リニアリストにはプロパティを含めることはできません。また、リスト内でストリングを項目として使用する場合は、ストリングを二重引用符で囲みます。

例えば、[6, 3, 8] はリニアリストで、これらの数字にはプロパティがありません。一方、[#gears:6, #balls:3,#ramps:8] はプロパティリストです。各数字は関連するプロパティがあり、この例では、機械装置の部品名になっています。プロパティリストは、機械構造のシミュレーションで、ステージ上に現在いくつの部品が表示されているかを確認するときなどに使用できます。プロパティは、1 つのプロパティリスト内で重複して使うことができます。

リストは、英数字順、および五十音順にソートすることができます。リニアリストはリスト内の値によってソートされ、プロパティリストはリスト内のプロパティ名によってソートされます。リストをソートするには、リニアリストまたはプロパティリストに対して適切なコマンドを実行します。

- リニアリストでは、シンボルとストリングの大文字と小文字が区別されます。
- プロパティリストでは、シンボルの大文字と小文字は区別されませんが、ストリングの大文字と小文字は区別されます。

リニアリストまたはプロパティリストには、必ずしも値を入れる必要はありません。2 つの角括弧 ([ ]) のみで構成される空白のリストもあります。リニアリストを作成または消去するには、そのリストに [ ] を割り当てます。プロパティリストを作成または消去するには [:] を割り当てます。

リスト内の項目は、それぞれ変更、確認、読み取りが可能です。

リストのインスタンスは、そのリストへの参照として扱われます。つまり、各インスタンスはデータそのものと同じで、変更すると元のデータが変更されます。リストの複製を作成するには duplicate コマンドを使います。

リストを削除するための記述は必要ありません。リストは、変数によって参照されなくなると自動的に削除されます。ただし、グローバル変数に格納されているリストはムービーが変わっても存続します。

リストは、on prepareMovie ハンドラ内ですばやく初期化することができます。リストをフィールドキャストメンバ内に記述し、それを変数に割り当てる方法もあります。このようにすると変数を操作することによってリストを処理できます。

PC 互換キーボードの中には角括弧がついていないものもあります。このような場合には、list 関数を使ってリニアリストを作成します。

プロパティリストを作成するには、ストリングとしてリストを作成し、これを実際のリストに変換します。

```
myListString = numToChar(91) & ":" & numToChar(93)
put myListString
-- "[:]"
myList = myListString.value
put myList
-- [:]
put myList.listP
-- 1
myList[#name] = "Brynn"
put myList
-- [#name: "Brynn"]
```

例文

次のステートメントは、変数 machinery にリストを割り当てることによってリストを定義しています。

```
-- Lingo syntax
machinery = [#gears:6, #balls:3, #ramps:8]

// JavaScript syntax
var machinery = propList("gears",6, "balls",3, "ramps",8);
```

次のハンドラは、まずパラメータ aList に渡されるリストをソートし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
on sortList aList
    alist.sort()
    put(aList)
end sortList

// JavaScript syntax
function sortList(aList) {
    aList.sort();
    put(aList);
}
```

ムービーが sortList machinery というステートメントを実行し、変数 machinery が上記の例文で定義したリストの場合、結果は [#balls:3, #gears:6, #ramps:8] になります。

次のステートメントは、空白のリニアリストを作成します。

```
-- Lingo syntax
x = [ ]
x = list()

// JavaScript syntax
var x = list();
```

次のステートメントは、空白のプロパティリストを作成します。

```
-- Lingo syntax
x = [:]
x = propList()

// JavaScript syntax
var x = propList();
```

関連事項

add, addVertex(), append, count(), deleteAt, duplicate()（リスト関数）, findPos, findPosNear, getProp(), getAt(), getLast(), getPos(), ilk(), list(), max(), min, setAt, setaProp, sort

@ (パス名演算子)

使用法

```
@pathReference
```

解説

この演算子は、現行ムービーがあるフォルダのパスを示します。パス名演算子を使って相対的に定義したパス名は、Windows® と Mac® の両方のコンピュータで認識することができます。

現行ムービーのフォルダを表すときには、"@" シンボルのすぐ後に次のパス指定記号のいずれかを挿入します。

- / (スラッシュ)
- \（バックスラッシュ、日本語フォントでは円記号）
- : (コロン)

ムービーの場所を照会すると、"@" シンボルを含む形式のストリングが返されます。

Director ムービー間を移動する場合や、リンクされたメディアキャストメンバのソースを変更する場合は、"@" シンボルのみを使います。@ シンボルは、Director の制御範囲外で処理される Fileio Xtra 拡張機能などの関数では使用できません。

このパス名をもとに、現行ムービーのフォルダより上または下の階層のフォルダを指定することができます。"@" は、プロジェクタの場所ではなく、現行ムービーの場所を表します。

- "@" シンボルのすぐ後にパス指定記号を追加してムービーフォルダの上の階層にあるフォルダを指定します。
- 現在のフォルダ名の後に（/、\、または : で区切って）フォルダ名およびファイル名を追加して、フォルダ内のサブフォルダとファイルを指定します。

相対的パス名を使用することで、ムービーのフォルダとは別のフォルダ内のファイルの位置を表すことができます。

例文

いずれの式も、現行ムービーのフォルダ内にある "bigFolder" サブフォルダを示しています。

```
@/bigFolder
@:bigFolder
@\bigFolder
```

いずれの式も、現行ムービーのフォルダ内にある bigFolder サブフォルダの linkedFile ファイルを示しています。

```
@:bigFolder:linkedFile
@\bigFolder\linkedFile
@/bigFolder/linkedFile
```

次の式は、現行ムービーのフォルダより 1 つ上の階層にあるファイル "linkedFile" を示しています。

```
@//linkedFile
```

次の式は、現行ムービーのフォルダより 2 つ上の階層にあるファイル "linkedFile" を示しています。

```
@:::linkedFile
```

いずれの式も、otherFolder フォルダ内にあるファイル linkedFile を示しています。otherFolder フォルダは現行ムービーのフォルダより 1 つ上の階層のフォルダ内にあります。

```
@::otherFolder:linkedFile
@\\otherFolder\linkedFile
@//otherFolder/linkedFile
```

関連事項

searchPathList, fileName （キャスト）, fileName （メンバ）, fileName （ウィンドウ）

and

使用法

```
-- Lingo syntax
logicalExpression1 and logicalExpression2

// JavaScript syntax
logicalExpression1 && logicalExpression2
```

解説

2 つの論理式が両方とも TRUE であるかどうかを調べるための論理演算子です。[論理式 **1**] と [論理式 **2**] の両方とも TRUE の場合、結果は TRUE (1) になり、両方または一方の式が FALSE の場合、結果は FALSE (0) になります。

この演算子の評価優先度は 4 です。

例文

次のステートメントは 2 つの論理式が両方とも TRUE であるかどうかを判断し、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(1 < 2 and 2 < 3)

// JavaScript syntax
put((1 < 2) && (2 < 3));
```

結果は、TRUE を表す数値の 1 になります。

次のステートメントの最初の論理式は TRUE で、2 番目の論理式は FALSE です。つまり、両方の論理式とも TRUE であるという条件には合わないので、FALSE であることを表す値 0 が結果として返されます。

```
-- Lingo syntax
put(1 < 2 and 2 < 1)
-- 0

// JavaScript syntax
put((1 < 2) && (2 < 1));
// 0
```

関連事項

not, or

contains

使用法

```
-- Lingo syntax
stringExpression1 contains stringExpression2

// JavaScript syntax
stringExpression1.indexOf(stringExpression2);
```

解説

2 つのストリングを比較する演算子で、[ストリング式 **1**] が [ストリング式 **2**] を含む場合、結果は TRUE になり、含まない場合、結果は FALSE になります。

contains 比較演算子の評価優先度は 1 です。

contains は、ユーザが特定の文字またはストリングを入力したかどうかを調べるときに使用できる比較演算子です。contains 演算子を使って 1 つまたは複数のフィールド内で特定のストリングを検索することもできます。

例文

次のステートメントでは、渡された文字が数字かどうかを判断しています。

```
-- Lingo syntax
on isNumber aLetter
    digits = "1234567890"
    if digits contains aLetter then
        return TRUE
    else
        return FALSE
    end if
end

// JavaScript syntax
function isNumber(aLetter) {
    var digits = "1234567890"
    if (digits.indexOf(aLetter) >= 0) {
        return true;
    } else {
        return false;
    }
}
```

注意：ストリング比較では、大文字小文字または発音記号は区別されません。このため、"a" と Å は同じものとして扱われます。

関連事項

offset() (string function), starts

mod

使用法

```
-- Lingo syntax
integerExpression1 mod integerExpression2

// JavaScript syntax
integerExpression1 % integerExpression2
```

解説

2 つの整数式間の除算で生じた余りを求める算術演算子です。演算は、[整数式 **1**] を [整数式 **2**] で割ります。

除算の余りの整数が式全体の結果値になり、常に [整数式 **1**] の記号を含んでいます。

この演算子の評価優先度は 4 です。

例文

次のステートメントは、7 を 4 で割り、余りをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(7 mod 4)

// JavaScript syntax
put(7 % 4);
```

この結果は 3 になります。

次のハンドラは、奇数番号のすべてのスプライトのインク効果を "copy" に設定します ("copy" インク効果の番号は 0 です)。まずスプライト番号を 2 で割って余りが 1 になるかどうかで、変数 mySprite に格納されたスプライトが奇数番号かどうか調べます。余りが 1 (つまり、奇数) であれば、インク効果を "copy" に設定します。

```
-- Lingo syntax
on setInk
    repeat with mySprite = 1 to _movie.lastChannel
        if (mySprite mod 2) = 1 then
            sprite(mySprite).ink = 0
        else
            sprite(mySprite).ink = 8
        end if
    end repeat
end setInk

// JavaScript syntax
function setInk() {
    for (mySprite=1; mySprite<=_movie.lastChannel; mySprite++) {
        if ((mySprite % 2) == 1) {
            sprite(mySprite).ink = 0;
        } else {
            sprite(mySprite).ink = 8;
        }
    }
}
```

次のハンドラは、いくつかのビットマップの中でスプライトのキャストメンバを定期的に一巡します。

```
-- Lingo syntax
on exitFrame
    global gCounter
    -- These are sample values for bitmap cast member numbers
    theBitmaps = [2,3,4,5,6,7]
    -- Specify which sprite channel is affected
    theChannel = 1
    -- This cycles through the list
    gCounter = 1 + (gCounter mod theBitmaps.count)
    sprite(theChannel).memberNum = theBitmaps[gCounter]
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    // these are sample values for bitmap cast member numbers
    theBitmaps = new Array(2,3,4,5,6,7);
    // specify which sprite channel is affected
    theChannel = 1;
    // this cycles through the list
    _global.gCounter = 1 + (_global.gCounter % theBitmaps.length);
    sprite(theChannel).memberNum = theBitmaps[_global.gCounter];
    _movie.go(_movie.frame);
}
```

not

使用法

```
-- Lingo syntax
not logicalExpression

// JavaScript syntax
! logicalExpression
```

解説

指定した論理式を否定する論理演算子です。TRUE 値を FALSE に、FALSE 値を TRUE にするのと同じ機能があります。既知の条件が満たされているかどうかを調べるのに使用できます。

この論理演算子の評価優先度は 5 です。

例文

次のステートメントは、1 が 2 より小さくないかどうかを評価します。

```
-- Lingo syntax
put(not (1 < 2))

// JavaScript syntax
put(!(1 < 2));
```

1 は 2 より小さいため、結果は 0、つまり FALSE になります。

次のステートメントは 1 が 2 より大きくないかどうかを評価します。

```
-- Lingo syntax
put(not (1 > 2))

// JavaScript syntax
put(!(1 > 2));
```

1 は 2 よりも大きくないため、結果は 1、つまり TRUE になります。

次のハンドラは " スタイル " メニュー内の " 太字 " の the checkMark メニュー項目プロパティを現行設定と逆のものにします。

```
-- Lingo syntax
on resetMenuItem
    menu("Style").menuItem("Bold").checkMark =
    not (menu("Style").menuItem("Bold").checkMark)
end resetMenuItem

// JavaScript syntax
function resetMenuItem() {
    menu("Style").menuItem("Bold").checkMark =
    !(menu("Style").menuItem("Bold").checkMark)
}
```

関連事項

and, or

or

使用法

```
-- Lingo syntax
logicalExpression1 or logicalExpression2

// JavaScript syntax
logicalExpression1 || logicalExpression2
```

解説

複数の論理式に論理 OR 演算を実行し、いずれかの式が TRUE かどうかを判別する演算子です。

この論理演算子の評価優先度は 4 です。

例文

次のステートメントは、1 < 2 と 1 > 2 のいずれかが TRUE になるかどうかをチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put((1 < 2) or (1 > 2))

// JavaScript syntax
put((1 < 2) || (1 > 2));
```

最初の式が TRUE になることによって、結果は TRUE を表す 1 になります。

次のステートメントは、フィールドキャストメンバ " 州 " の内容が "AK" または "HI" かどうかをチェックし、そうである場合は警告を表示します。

```
-- Lingo syntax
if member("State").text = "AK" or member("State").text = "HI" then
    _player.alert("You're off the map!")
end if

// JavaScript syntax
if (member("State").text == "AK" || member("State").text == "HI") {
    _player.alert("You're off the map!");
}
```

関連事項
and, not

ランダムアクセス演算子らんだむあくせすえんざんし

使用法

```
byteArrayObject[index]
```

解説

バイト配列演算子。バイト配列で記号なしバイトを取得または設定します。値が 255 より大きい場合は、最下位バイトが対象になります。

例文

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put bArray[1]

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(bArray[1]);
```

starts

使用法

```
-- Lingo syntax
string1 starts string2

// JavaScript syntax
string1.indexOf(string2) == 0;
```

解説

演算子。2 つのストリングを比較して、[ストリング **1**] の先頭が [ストリング **2**] と一致するかどうかを判定します。一致する場合は TRUE (または 1) になり、そうでない場合は FALSE (または 0) になります。

ストリング比較では、大文字小文字および発音符号は区別されません。このため、"**a**" と "**Å** " は同じものとして扱われます。

この演算子の評価優先度は 1 です。

例文

次のステートメントは、「**Macrostuff**」という単語が「Macro」で始まっているかどうかを調べ、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put("Macrostuff" starts "Macro")

// JavaScript syntax
var string1 = "Macrostuff";
put(string1.indexOf("Macro") == 0);
```

結果は、TRUE を表す数値の 1 になります。

関連事項

contains

String

使用法

```
String(byteArrayObject)
```

解説

バイト配列演算子。バイト配列を、読み取り可能な 16 進数形式に変換します。このメソッドは、メッセージウィンドウ、オブジェクトインスペクタ、デバッガなどにバイト配列を書き込むのに便利です。

例文

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put String(bArray)

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(String(bArray));
```

第 14 章：プロパティ

このセクションでは、Director® で使用できるすべてのプロパティをアルファベット順に示します。

_global

使用法

```
-- Lingo syntax
_global

// JavaScript syntax
_global;
```

解説

すべてのグローバル変数を格納する Global オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

すべてのグローバル変数は、Lingo および JavaScript シンタックスの両方にアクセス可能です。

例文

次のステートメントは、変数 objGlobal を _global プロパティに設定します。

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

次のステートメントは _global プロパティを直接使用して、すべてのグローバル変数をクリアします。

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

関連事項

グローバル

_key

使用法

```
-- Lingo syntax
_key

// JavaScript syntax
_key;
```

解説

ユーザのキーボード操作を監視するために使用される、Key オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

例文

次のステートメントは、変数 objKey を _key プロパティに設定します。

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

次のステートメントは直接 _key プロパティを使用して、key プロパティの値にアクセスします。

```
-- Lingo syntax
theKey = _key.key

// JavaScript syntax
var theKey = _key.key;
```

関連事項

キー

_mouse

使用法

```
-- Lingo syntax
_mouse

// JavaScript syntax
_mouse;
```

解説

マウスの動きとマウスクリックを含むユーザのマウス操作にアクセスする、Mouse オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

例文

次のステートメントは、変数 objMouse を _mouse プロパティに設定します。

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

次のステートメントは直接 _mouse プロパティを使用して、mouseH プロパティの値にアクセスします。

```
-- Lingo syntax
theMouseH = _mouse.mouseH

// JavaScript syntax
var theMouseH = _mouse.mouseH;
```

関連事項

マウス

_movie

使用法

```
-- Lingo syntax
_movie

// JavaScript syntax
_movie;
```

解説

Director プレーヤ内で現在アクティブなムービーを表す、Movie オブジェクトへの参照を示す最上位レベルのプロパティです。また、ムービーレベルで使用できるプロパティとメソッドにアクセスできるようにします。読み取り専用。

例文

次のステートメントは、変数 objMovie を _movie プロパティに設定します。

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

次のステートメントは直接 _movie プロパティを使用して、displayTemplate プロパティの値にアクセスします。

```
-- Lingo syntax
theTemplate = _movie.displayTemplate

// JavaScript syntax
var theTemplate = _movie.displayTemplate;
```

関連事項

ムービー

_player

使用法

```
-- Lingo syntax
_player

// JavaScript syntax
_player;
```

解説

ウィンドウ内のムービー（MIAW）を含むすべてのムービーを管理および実行する、Player オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

例文

次のステートメントは、変数 objPlayer を _player プロパティに設定します。

```
-- Lingo syntax
objPlayer = _player

// JavaScript syntax
var objPlayer = _player;
```

次のステートメントは直接 _player プロパティを使用して、xtraList プロパティの値にアクセスします。

```
-- Lingo syntax
theXtras = _player.xtraList

// JavaScript syntax
var theXtras = _player.xtraList;
```

関連事項

プレーヤ

_sound

使用法

```
-- Lingo syntax
_sound

// JavaScript syntax
_sound;
```

解説

使用できる 8 個のサウンドチャネルすべてのオーディオ再生を制御する、Sound オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

例文

次のステートメントは、変数 objSound を _sound プロパティに設定します。

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

次のステートメントは直接 _sound プロパティを使用して soundLevel プロパティにアクセスします。

```
-- Lingo syntax
theLevel = _sound.soundLevel

// JavaScript syntax
var theLevel = _sound.soundLevel;
```

関連事項

サウンド

_system

使用法

```
-- Lingo syntax
_system

// JavaScript syntax
_system;
```

解説

システムレベルのメソッドを含むシステム情報および環境情報にアクセスできるようにする、System オブジェクトへの参照を示す最上位レベルのプロパティです。読み取り専用。

例文

次のステートメントは、変数 objSystem を _system プロパティに設定します。

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

次のステートメントは直接 _system プロパティを使用して freeBytes プロパティにアクセスします。

```
-- Lingo syntax
theBytes = _system.freeBytes

// JavaScript syntax
var theBytes = _system.freeBytes;
```

関連事項

システム

aboutInfo

使用法

```
-- Lingo syntax
_movie.aboutInfo

// JavaScript syntax
_movie.aboutInfo;
```

解説

オーサリング時に、[ムービープロパティ] ダイアログボックスに入力されたストリングであり、ムービープロパティです。読み取り専用。

例文

次のステートメントは、メッセージウィンドウにムービー情報を表示します。

```
-- Lingo syntax
trace(_movie.aboutInfo)

// JavaScript syntax
trace(_movie.aboutInfo);
```

関連事項

copyrightInfo（ムービー）, ムービー

actionsEnabled

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.actionsEnabled

// JavaScript syntax
memberOrSpriteObjRef.actionsEnabled;
```

解説

Adobe® Flash® コンテンツでのアクションを有効にするかどうかを定義するキャストメンバプロパティおよびスプライトプロパティです。これを有効（TRUE,、デフォルト）にするか、または無効（FALSE）にするかをコントロールします。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、パラメータとしてスプライトの参照を受け入れ、そのスプライトの actionsEnabled プロパティを反転します。

```
-- Lingo syntax
on ToggleActions(whichSprite)
     sprite(whichSprite).actionsEnabled = not(sprite(whichSprite).actionsEnabled)
end

// JavaScript syntax
function ToggleActions(whichSprite) {
     switch(sprite(whichSprite).actionsEnabled){
        case 0:
             sprite(whichSprite).actionsEnabled = 1;
            break;
        case 1:
            sprite(whichSprite).actionsEnabled = 0;
            break;
    }
}
```

active3dRenderer

使用法

```
-- Lingo syntax
_movie.active3dRenderer

// JavaScript syntax
_movie.active3dRenderer;
```

解説

3D スプライトの描画にムービーが現在使用しているレンダリング方法を示すムービープロパティです。このプロパティは、getRendererServices().renderer プロパティと同じです。読み取り専用。

active3dRenderer プロパティで使用できる値は、#openGL、#directX7_0, #directx9、#directX5_2 および #software です。#openGL、#directX7_0、#directx9, および #directX5_2 はビデオカードドライバです。これらを利用できない場合は、ソフトウェアのレンダリング方法である #software が使用されますが、先の 3 つと比べるとパフォーマンスは劣ります。

このプロパティを設定するには、getRendererServices().renderer を使用します。

例文

次の例は、現在使用されているレンダリング方法を確認する 2 とおりの方法を示しています。

```
-- Lingo syntax
put(_movie.active3dRenderer)
put(getRendererServices().renderer)

// JavaScript syntax
put(_movie.active3dRenderer);
put(getRendererServices().renderer);
```

関連事項

ムービー , renderer

activeCastLib

使用法

```
-- Lingo syntax
_movie.activeCastLib

// JavaScript syntax
_movie.activeCastLib;
```

解説

最後にどのキャストライブラリがアクティブになったのかを示すムービープロパティです。読み取り専用。

activeCastLib プロパティの値はキャストライブラリの番号です。

activeCastLib プロパティは、Cast オブジェクトの selection プロパティを扱うときに役立ちます。どのキャストライブラリが参照されているのかを判断するときに使用します。

例文

次のステートメントは、最後にアクティブになったキャスト内で選択されたキャストメンバを変数 selectedMembers に割り当てます。

```
-- Lingo syntax
castLibOfInterest = _movie.activeCastLib
selectedMembers = castLib(castLibOfInterest).selection

// JavaScript syntax
var castLibOfInterest = _movie.activeCastLib;
var selectedMembers = castLib(castLibOfInterest).selection;
```

関連事項

プレーヤ , selection

activeWindow

使用法

```
-- Lingo syntax
_player.activeWindow

// JavaScript syntax
_player.activeWindow;
```

解説

どのムービーウィンドウが現在アクティブになっているかを示すプレーヤプロパティです。読み取り専用。

メインムービーの場合、activeWindow はステージになり、ウィンドウ内のムービー（MIAW）の場合、activeWindow はウィンドウ内のムービーそのものになります。

例文

次のステートメントは、クリックしたウィンドウのタイトルバーに " アクティブ " と表示し、それ以外の開いているウィンドウのタイトルバーには " 非アクティブ " と表示します。

```
-- Lingo syntax
on activateWindow
    clickedWindow = _player.windowList.getPos(_player.activeWindow)
    windowCount = _player.windowList.count
    repeat with x = 1 to windowCount
        if (x = clickedWindow) then
            _player.window[clickedWindow].title = "Active"
        else
            _player.windowList[x].title = "Inactive"
        end if
    end repeat
end activateWindow

// JavaScript syntax
function activateWindow() {
    var clickedWindow = _player.windowList.getPos(_player.activeWindow);
    var windowCount = _player.windowList.count;
    for (var x = 1; x <= windowCount; x++) {
        if (x == clickedWindow) {
            _player.window[clickedWindow].title = "Active"
        }
        else {
            _player.windowList[x].title = "Inactive"
        }
    }
}
```

関連事項

プレーヤ

actorList

使用法

```
-- Lingo syntax
_movie.actorList

// JavaScript syntax
_movie.actorList;
```

解説

明示的にリストに追加された子オブジェクトをリストしたムービープロパティです。読み取り / 書き込み可能。

再生ヘッドがフレームを移動するたびに、actorList に列挙されたオブジェクトに stepFrame メッセージが送られます。

the actorList にオブジェクトを追加するには、_movie.actorList.append(**newScriptObjRef**) を使用します。各フレームでオブジェクトの親またはアンセスタスクリプト内にある stepFrame ハンドラが自動的に実行されるようになります。

the actorList のオブジェクトをすべて消去するには、actorList に空白リスト [ ] を割り当てます。

別のムービーに分岐した場合でも actorList の内容は消去されないため、新しいムービー内で予期せぬ動作を引き起こすことがあります。現行ムービーの子オブジェクトが別のムービーでは適用されないようにするには、分岐先ムービーの prepareMovie ハンドラ内に actorList = [ ] というステートメントを記述します。

例文

次のステートメントは、親スクリプト " 動くボール " から子オブジェクトを作成します。3 つの数値は、スクリプトに必要なパラメータです。

次のステートメントは、actorList の内容をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.actorList)

// JavaScript syntax
put(_movie.actorList);
```

次のステートメントは、actorList からオブジェクトを消去します。

```
-- Lingo syntax
_movie.actorList = [] -- using brackets
_movie.actorList = list() -- using list()

// JavaScript syntax
_movie.actorList = list();
```

関連事項

ムービー , on prepareMovie, on stepFrame

alertHook

使用法

```
-- Lingo syntax
_player.alertHook

// JavaScript syntax
_player.alertHook;
```

解説

alertHook ハンドラを含む親スクリプトを指定するためのプレーヤプロパティです。読み取り / 書き込み可能。

alertHook で、ファイルエラーやスクリプトのエラーに関するアラートをどのように表示するか定義します。エラーが発生し、親スクリプトに alertHook が割り当てられていると、Director は親スクリプトの alertHook ハンドラを起動します。

alertHook ハンドラをムービースクリプトに入れることもできますが、alertHook をビヘイビアースクリプトまたは親スクリプトに入れることをお勧めします。こうすると、様々な場所で発生するエラーによってハンドラが起動し、エラーの原因と場所がわかりにくくなるのを防ぐことができます。

alertHook ハンドラはエラー発生時に起動するものなので、エラーを処理する目的以外で alertHook ハンドラを使用しないでください。例えば、alertHook ハンドラに go() ステートメントを記述するのは適切ではありません。

alertHook ハンドラには、1 つのインスタンスパラメータ、エラー内容を説明する 2 つのストリングパラメータ、およびハンドラを起動する追加イベントを指定する 1 つのオプションのパラメータが渡されます。

4 番目のパラメータは、次の 4 つのうちのいずれかの値です。

- #alert - alert() メソッドによってハンドラが起動するように指定します。
- #movie - go() コマンドの実行中にファイルが見つからないエラーが発生したときに、ハンドラが起動するように指定します。
- #script - スクリプトエラーによってハンドラが起動するように指定します。
- #safeplayer - safePlayer プロパティをオンにすることによってハンドラが起動するように指定します。

  ハンドラ内のスクリプトによりますが、alertHook ハンドラはエラーを無視したり、他の方法でエラーをレポートしたりできます。

例文

次のステートメントは、エラーが発生したときにアラートを表示するかどうかを判断するスクリプトとして親スクリプト "Alert" を指定します。エラーが発生すると、エラーストリングとメッセージストリングをフィールドキャストメンバ "Output" に割り当て、値 1 を返します。

```
-- Lingo syntax
on prepareMovie
    _player.alertHook = script("Alert")
end

-- "Alert" script
on alertHook me, err, msg
    member("Output").text = err && msg
    return 1
end

// JavaScript syntax
function prepareMovie() {
     _player.alertHook = "alert("Error type", "Error message");"
}

// alert handler
function alert(err, msg) {
    member("Output").text = err + " " + msg; return 1;
}
```

関連事項

プレーヤ , safePlayer

alignment

使用法

```
-- Lingo syntax
memberObjRef.alignment

// JavaScript syntax
memberObjRef.alignment;
```

解説

指定したキャストメンバ内のストリングを表示するときの揃え方を定義するキャストメンバプロパティです。このプロパティを使うには、フィールドおよびテキストキャストメンバに文字が必要です (スペースのみでも有効)。

フィールドキャストメンバの場合、プロパティの値は、"left"、"center"、"right" のいずれかを含むストリングになります。

テキストキャストメンバの場合、プロパティの値は、#left、#center、#right、#full のいずれかを含むシンボルになります。

[キャストメンバ] にはキャスト名またはキャスト番号を指定します。

このプロパティは、値を調べたり、設定することができます。テキストキャストメンバの場合、段落単位でプロパティを設定することができます。

例文

次のステートメントは、変数 characterAlign に、フィールドキャストメンバ " 六条の語り " の alignment プロパティの値を割り当てます。

```
--Lingo syntax
characterAlign = member("Rokujo Speaks").alignment

// JavaScript syntax
var characterAlign = member("Rokujo Speaks").alignment;
```

関連事項

text, font, lineHeight, fontSize, fontStyle, &、+（連結演算子）, &&、+（連結演算子）

allowCustomCaching

使用法

```
-- Lingo syntax
_movie.allowCustomCaching

// JavaScript syntax
_movie.allowCustomCaching;
```

解説

Director の将来のバージョンで、プライベートキャッシュに関する情報を含める予定のムービープロパティです。読み取り / 書き込み可能。

このプロパティのデフォルトは TRUE です。

関連事項

allowGraphicMenu, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, ムービー

allowGraphicMenu

使用法

```
-- Lingo syntax
_movie.allowGraphicMenu

// JavaScript syntax
_movie.allowGraphicMenu;
```

解説

Adobe Shockwave® 環境でムービーを再生するとき、コンテキストメニューを画像で表示するかどうかを設定するムービープロパティです。読み取り / 書き込み可能。

このプロパティを FALSE にすると、画像の代わりに文字でメニューを表示します。

このプロパティのデフォルトは TRUE です。

関連事項

allowCustomCaching, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, ムービー

allowSaveLocal

使用法

```
-- Lingo syntax
_movie.allowSaveLocal

// JavaScript syntax
_movie.allowSaveLocal;
```

解説

Shockwave® Player 環境でムービーを再生するとき、コンテキストメニューの保存コマンドを有効にするかどうかを設定するムービープロパティです。読み取り / 書き込み可能。

このプロパティは、Shockwave Player の将来のバージョンでの機能拡張のために用意されています。

このプロパティのデフォルトは TRUE です。

関連事項

allowCustomCaching, allowGraphicMenu, allowTransportControl, allowVolumeControl, allowZooming, ムービー

allowTransportControl

使用法

```
-- Lingo syntax
_movie.allowTransportControl

// JavaScript syntax
_movie.allowTransportControl;
```

解説

Shockwave Player の将来のバージョンでの機能拡張のために用意されているムービープロパティです。これは、Shockwave のパブリッシュ設定にある「トランスポートコントロール」チェックボックスの設定と一致します。読み取り / 書き込み可能。

このプロパティのデフォルトは TRUE です。

関連事項

allowCustomCaching, allowGraphicMenu, allowSaveLocal, allowVolumeControl, allowZooming, ムービー

allowVolumeControl

使用法

```
-- Lingo syntax
_movie.allowVolumeControl

// JavaScript syntax
_movie.allowVolumeControl;
```

解説

Shockwave Player 環境でムービーを再生するとき、コンテキストメニューの音量調整コマンドを有効にするかどうかを設定するムービープロパティです。読み取り / 書き込み可能。

このプロパティを TRUE にすると音量調節が有効になり、FALSE にすると無効になります。

このプロパティのデフォルトは TRUE です。

関連事項

allowCustomCaching, allowGraphicMenu, allowSaveLocal, allowTransportControl, allowZooming, ムービー

allowZooming

使用法

```
-- Lingo syntax
_movie.allowZooming

// JavaScript syntax
_movie.allowZooming;
```

解説

Shockwave Player でムービーを再生するとき、ズームしたりストレッチできるようにするかを設定するためのムービープロパティです。読み取り / 書き込み可能。

FALSE に設定すると、ブラウザでムービーのサイズを変更できなくなります。

このプロパティのデフォルトは TRUE です。

関連事項

allowCustomCaching, allowGraphicMenu, allowSaveLocal, allowTransportControl, allowVolumeControl, ムービー

alphaThreshold

使用法

```
-- Lingo syntax
memberObjRef.alphaThreshold

// JavaScript syntax
memberObjRef.alphaThreshold;
```

解説

ビットマップのアルファチャネル値の違いにより、マウスクリックを検出するかどうかを決めることができるビットマップキャストメンバプロパティです。32 ビットのビットマップ画像のアルファチャネル値と同じように、0 ～ 255 の値を取ります。

マウスをクリックしたときに、その点のアルファマップのピクセル値が alphaThreshold の値以上の場合に、マウスがクリックされたと判定します。alphaThreshold を 0 にすると、アルファチャネルの内容にかかわらず、常にクリックされたと判断されるようになります。

関連事項

useAlpha

ambient

使用法

```
member(whichCastmember).shader(whichShader).ambient
member(whichCastmember).model(whichModel).shader.ambient
member(whichCastmember).model(whichModel).shaderList{[index]}.ambient
```

解説

3D #standard シェーダプロパティ。キャストメンバ内の周囲ライトを構成する各カラーコンポーネントが、シェーダによってどのくらい反射されるのかを示します。

例えば、周囲ライトのカラーが rgb(255, 255, 255) で、シェーダの ambient プロパティの値が rgb(255, 0, 0) である場合、シェーダのカラーで反射されるライトの赤のコンポーネントがすべて反射されます。ライトの青および緑のコンポーネントは、シェーダのカラーにかかわらず反射されません。シーンに他のライトがない場合、シェーダの青と緑はライトを反射せず、黒になります。

このプロパティのデフォルト値は rgb(63,63,63) です。

例文

次のステートメントは、" 椅子 " というモデルの ambient プロパティを rgb(255, 255, 0) に設定します。" 椅子 " は、シーン内の周囲ライトの赤および緑のコンポーネントを完全に反射し、青のコンポーネントは完全に無視します。

```
-- Lingo syntax
member("Room").model("Chair").shader.ambient = rgb(255, 0, 0)

// JavaScript syntax
member("Room").getPropRef("model",1).shader.ambient = color(255,0,0);
```

関連事項

ambientColor, newLight, type (light), diffuse, specular (shader)

ambientColor

使用法

```
member(whichCastmember).ambientColor
```

解説

3D キャストメンバプロパティ。キャストメンバの、デフォルトの周囲ライトの RGB カラーを示します。

このプロパティのデフォルト値は rgb(0, 0, 0) です。シーンにライトは追加されません。

例文

次のステートメントは、" 部屋 " というキャストメンバの ambientColor プロパティを rgb(255, 0, 0) に設定します。キャストメンバのデフォルトの周囲ライトは赤になります。このプロパティは、プロパティインスペクタでも設定できます。

```
-- Lingo syntax
member("Room").ambientColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").ambientColor = color(255,0,0);
```

関連事項

directionalColor, directionalPreset, ambient

ancestor

使用法

```
property {optionalProperties} ancestor
```

解説

親スクリプトやビヘイビアーにはないハンドラを使用できるようにするためのオブジェクトプロパティです。

ancestor プロパティは、通常、複数の親スクリプトで共有されます。子オブジェクトやビヘイビアーが、アンセスタ (ancestor) から受け継いだ特定のビヘイビアーのみを共有し、親から受け継いだ他のビヘイビアーは違ったままにするときなどに使用します。

子オブジェクトの場合、通常は親スクリプトの on new ハンドラ内で ancestor プロパティが割り当てられます。対応する定義済みハンドラを含まない子オブジェクトにメッセージが送られると、そのメッセージは ancestor プロパティで定義されるスクリプトに受け渡されます。

ビヘイビアーにアンセスタがあるときは、アンセスタは mouseDown や mouseWithin などのマウスイベントを受けることができます。

ancestor プロパティを使用することで、多数のオブジェクトグループのビヘイビアーとプロパティをコマンド 1 つで変更することができます。

アンセスタスクリプトは、子オブジェクトからアクセスできる独立したプロパティ変数を含むことができます。アンセスタスクリプト内のプロパティ変数を参照するには、次のシンタックスを使います。

me.**propertyVariable =value**

例えば、次のステートメントでは、アンセスタスクリプト内のプロパティ変数 legCount に、値 4 を割り当てています。

```
me.legCount = 4
```

現行オブジェクト内に存在しないプロパティ変数にアクセスするには、the**variableName** of **scriptName** というシンタックスを使います。例えば、次のステートメントでは、子オブジェクト内の変数 myLegCount に、アンセスタスクリプト内のプロパティ変数 legCount の値を割り当てています。

```
set myLegCount to the legCount of me
```

例文

次のスクリプトはいずれもキャストメンバです。アンセスタスクリプト " 動物 " と親スクリプト " 犬 " および " 人間 " は、相互に作用してオブジェクトを定義します。

最初のスクリプト " 犬 " は、プロパティ変数 breed を " 忠犬 " に設定し、the ancestor of Dog を " 動物 " スクリプトに設定します。また、アンセスタスクリプトに保存されている変数 legCount を 4 に設定します。

```
property breed, ancestor

on new me
    set breed = "Mutt"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 4
    return me
end
```

2 番目のスクリプト " 人間 " では、プロパティ変数 race を " コーカサス人 " に設定し、the ancestor of Man を " 動物 " スクリプトに設定します。また、アンセスタスクリプトに保存されている変数 legCount を 2 に設定します。

```
property race, ancestor
on new me
    set race to "Caucasian"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 2
    return me
end
```

関連事項

new(), menu, property

angle (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.angle
```

解説

3D エミッタプロパティ。パーティクルシステムのパーティクルが放出される領域を表します。パーティクルシステムは、#particle というタイプのモデルリソースです。

パーティクル放出の基本的な方向は、エミッタの direction プロパティで設定されたベクトルです。ただし、パーティクルの放出方向は、0 からエミッタの angle プロパティ値の間のランダムな角度で、このベクトルからは外れます。

このプロパティの有効な範囲は 0.0 ～ 180.0 で、デフォルト値は 180.0 です。

例文

次のステートメントは、mrFount というモデルリソースの放出角度を 1 に設定します。放出されるパーティクルは、細い線になります。

```
member("fountain").modelResource("mrFount").emitter.angle = 1
```

関連事項
emitter, direction

angle (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.angle

// JavaScript syntax
dvdObjRef.angle;
```

解説
現在のカメラアングルの数値を返す DVD プロパティです。読み取り / 書き込み可能。

戻り値は整数です。

例文
次のステートメントは、現在のカメラアングルの数値を返します。

```
-- Lingo syntax
put(member(1).angle)-- 1

// JavaScript syntax
put(member(1).angle);// 1
```

関連事項
DVD

angleCount

使用法

```
-- Lingo syntax
dvdObjRef.angleCount

// JavaScript syntax
dvdObjRef.angleCount;
```

解説
現行タイトルで使用できるカメラアングルの数を返す DVD プロパティです。読み取り専用。

戻り値は、1 ～ 9 の整数です。

例文
次のステートメントは、現在使用できるカメラアングルの数を返します。

```
-- Lingo syntax
put(member(12).angleCount)-- 2

// JavaScript syntax
put(member(12).angleCount);// 2
```

関連事項
DVD

animationEnabled

使用法

```
member(whichCastmember).animationEnabled
```

解説

3D キャストメンバプロパティ。モーションを実行するか (TRUE)、無視するか (FALSE) を示します。このプロパティは、プロパティインスペクタでも設定できます。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" シーン " というキャストメンバのアニメーションを無効にします。

```
member("Scene").animationEnabled = FALSE
```

antiAlias

使用法

```
-- Lingo syntax
memberObjRef.antiAlias
// JavaScript syntax
memberObjRef.antiAlias;
```

解説

ベクトルシェイプまたは Flash® キャストメンバをレンダリングするときにアンチエイリアスを使用するかどうかを制御するキャストメンバプロパティです。アンチエイリアスを使用すると、描画品質が上がる代わりに、ムービーの再生速度が遅くなります。antiAlias プロパティのデフォルトは TRUE です。

ベクトルシェイプの場合、このプロパティに TRUE を設定することと、Flash asset の品質として #high を設定することは同じ効果があります。同様に、FALSE を設定することと #low を設定することは同じです。

antiAlias プロパティは、ベクトルシェイプスプライトの場合に限り、スプライトプロパティとして使用することもできます。

このプロパティは、値を調べたり、設定することができます。

注意：antiAlias プロパティは Director 11 のテキストメンバではサポートされていません。テキストメンバに対するアンチエイリアスには antiAliasType プロパティを使用します。

例文

次のビヘイビアーは、ムービーを再生するコンピュータの色深度を調べます。色深度が 8 ビット（256 色）以下に設定されている場合は、スプライトの antiAlias プロパティに FALSE を設定します。

```
--Lingo syntax
property spriteNum

on beginsprite me
    if _system.colorDepth <= 8 then
        member("text").antiAlias = FALSE
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        member("text").antiAlias = 0;
    }
}
```

関連事項

antiAliasThreshold, quality

antiAliasingEnabled

使用法

```
sprite(whichSprite).antiAliasingEnabled
```

解説

whichSprite というスプライトの 3D ワールドでアンチエイリアスが有効になっているかどうかを示します。このプロパティは、値を調べたり、変更することができます。デフォルト値は FALSE で、アンチエイリアスがオフであることを示します。antiAliasingEnabled プロパティが TRUE に設定されているとき、アンチエイリアスをサポートしない 3D レンダラに変更すると、このプロパティは FALSE に設定されます。ムービーの保存時には、このプロパティの値は保存されません。

アンチエイリアスが有効なスプライトは、アンチエイリアスが無効なスプライトよりも、多くのプロセッサパワーとメモリを使用します。アンチエイリアスを一時的にオフにすると、アニメーションとユーザインタラクションのパフォーマンスが改善されます。

例文

次の Lingo は、スプライト 2 に対して現在実行されている 3D レンダラがアンチエイリアスをサポートするかどうかを antiAliasingSupported プロパティを使用して調べます。アンチエイリアスがサポートされている場合は、2 番目のステートメントが antiAliasingEnabled プロパティを使用して、スプライトのアンチエイリアスをオンにします。

```
if sprite(2).antiAliasingSupported = TRUE then
    sprite(2).antiAliasingEnabled = TRUE
end if
```

関連事項

antiAliasingSupported, renderer, rendererDeviceList

antiAliasingSupported

使用法

```
sprite(whichSprite).antiAliasingSupported
```

解説

現行の 3D レンダラでアンチエイリアスがサポートされているかどうかを示す 3D スプライトプロパティです。このプロパティは、値を調べられますが、設定はできません。このプロパティは、TRUE または FALSE のいずれかを返します。

例文

次の Lingo は、スプライト 3 に対して現在実行されている 3D レンダラがアンチエイリアスをサポートするかどうかを調べます。アンチエイリアスがサポートされている場合は、2 番目のステートメントが antiAliasingEnabled プロパティを使用して、スプライトのアンチエイリアスをオンにします。

```
if sprite(3).antiAliasingSupported = TRUE then
    sprite(3).antiAliasingEnabled = TRUE
end if
```

関連事項

antiAliasingEnabled, renderer, rendererDeviceList

antiAliasThreshold

使用法

```
-- Lingo syntax
memberObjRef.antiAliasThreshold

// JavaScript syntax
memberObjRef.antiAliasThreshold;
```

解説

テキストキャストメンバが自動的にアンチエイリアスを適用する境界のポイントサイズを定義するテキストキャストメンバプロパティです。この定義は、テキストキャストメンバの antiAlias プロパティが TRUE の場合にのみ有効です。

値は、アンチエイリアスを使うフォントサイズ (ポイント単位) を表す整数です。

このプロパティのデフォルトは 14 ポイントです。

関連事項

antiAliasType

antiAliasType

使用法

```
-- Lingo syntax
memberObjRef.antiAliasType
// JavaScript syntax
memberObjRef.antiAliasType;
```

解説

キャストメンバプロパティです。テキストキャストメンバをレンダリングするときにアンチエイリアスを使用して描画品質を上げるかどうかを制御します。antiAliasType プロパティのデフォルトは #autoAlias です。

Director の以前のバージョンからアップグレードしている場合、「自動」オプションは antiAliasThreshold がゼロに設定された「指定値より大きい場合グレースケール」に対応します。「指定値以下ではオフ」オプションは対応するしきい値を持つ「指定値より大きい場合グレースケール」に対応します。

対応するアンチエイリアス処理のオプションを設定するには次のシンボルを使用します。

Auto #AutoAlias (Uses the font file information for anti-aliasing.)–

Grayscale All - #GrayScaleAllAlias (すべてのテキストメンバについてグレースケールのアンチエイリアス処理を有効にします。)

Subpixel All - #SubpixelAllAlias (すべてのテキストメンバについてサブピクセルのアンチエイリアス処理を有効にします。)

Grayscale Larger Than - #GrayscaleLargerThanAlias (フォントサイズが指定したしきい値よりも大きい場合にグレースケールのアンチエイリアス処理を有効にします。)

None - #NoneAlias (現在のキャストメンバのアンチエイリアス処理をオフにします。)

このプロパティは、値を調べたり、設定することができます。

例文

次のビヘイビアーは、ムービーを再生するコンピュータの色深度を調べます。色深度が 8 ビット（256 色）以下に設定されている場合は、スプライトの antiAlias プロパティに FALSE を設定します。

```
--Lingo syntax
property spriteNum

on beginsprite me
    if _system.colorDepth <= 8 then
        member("text").antiAliasType = #noneAlias
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        member("text").antiAliasType = symbol("noneAlias");
    }
}
```

関連事項

antiAliasThreshold

appearanceOptions

使用法

```
-- Lingo syntax
windowObjRef.appearanceOptions

// JavaScript syntax
windowObjRef.appearanceOptions;
```

解説

ウィンドウの外観オプションを保存するプロパティのリストを指定するウィンドウプロパティです。読み取り / 書き込み可能。

このプロパティリストには、次のプロパティが含まれます。

プロパティ	解説
#mask	ウィンドウのマスクとして使用する 1 ビットのキャストメンバを指定します。
#border	ウィンドウの境界線の種類を指定します。このプロパティの値は、次の 3 つの値のうちのいずれかです。 • #none。ウィンドウの周囲に境界線を指定しません。 • #line。ウィンドウの周囲に 1 ピクセルの黒の境界線を指定します。 #none プロパティおよび #line プロパティが有効であるのは、titlebarOptions.visible プロパティが FALSE に設定されている場合のみです。
#metal	（Macintosh のみ）ウィンドウに金属のような外観を設定するかどうかを指定します（TRUE）。FALSE の場合、ウィンドウは氷のような外観になります。
#dragRegionMask	ウィンドウの領域のマスクとして使用する 1 ビットのキャストメンバを指定します。
#shadow	（Macintosh のみ）ウィンドウに影を付けるかどうかを指定します。通常、Macintosh のウィンドウには影が付きます。
#liveresize	（Macintosh のみ）ユーザの操作中にウィンドウのサイズ変更を有効にするかどうかを指定します。TRUE の場合、サイズ変更が有効になります。FALSE の場合、サイズ変更が無効になります。

これらのプロパティは、ムービーオブジェクトの displayTemplate プロパティを使用してアクセスすることもできます。

例文

次のステートメントは、" 制御パネル " という名前のウィンドウの、現在のすべての外観オプションをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(window("Control Panel").appearanceOptions)

// JavaScript syntax
put(window("Control Panel").appearanceOptions);
```

次のステートメントは、border プロパティを設定し、" 制御パネル " という名前のウィンドウの周囲に 1 ピクセルの境界線を表示します。

```
-- Lingo syntax
window("Control Panel").appearanceOptions.border = #line

// JavaScript syntax
window("Control Panel").appearanceOptions.border = symbol("line");
```

関連事項

displayTemplate, titlebarOptions, visible, ウィンドウ

applicationName

使用法

```
-- Lingo syntax
_player.applicationName

// JavaScript syntax
_player.applicationName;
```

解説

オーサリング中の Director アプリケーションの実行コピーの名前、または実行中のプロジェクタファイルの名前を指定するプレーヤプロパティです。読み取り専用。

プロパティの値はストリングです。

Shockwave Player はこのプロパティをサポートしていません。

例文

次のステートメントは、Director アプリケーションである Director.exe の名前を表示します。

```
-- Lingo syntax
put(_player.applicationName)

// JavaScript syntax
put(_player.applicationName);
```

関連事項

applicationPath, プレーヤ

applicationPath

使用法

```
-- Lingo syntax
_player.applicationPath

// JavaScript syntax
_player.applicationPath;
```

解説

オーサリング中の Director アプリケーションの実行コピーを含むフォルダ、または実行中にプロジェクタを含むフォルダの場所またはパスを定義するためのプレーヤプロパティです。読み取り専用。

プロパティの値はストリングです。

applicationPath の後に & とサブフォルダのパス名を記述し、括弧で囲み、スクリプトが式を 1 つの文として認識できるようにします。

Shockwave Player はこのプロパティをサポートしていません。

例文

次のステートメントは、Director アプリケーションが入っているフォルダのパス名を表示します。

```
-- Lingo syntax
put(_player.applicationPath)

// JavaScript syntax
put(_player.applicationPath);
```

次のステートメントは、（Windows マシン上の）ウィンドウ内でムービー "Sunset Boulevard" を開きます。

```
-- Lingo syntax
window(_player.applicationPath & "Film Noir\Sunset Boulevard").open()

// JavaScript syntax
window(_player.applicationPath + "Film Noir\Sunset Boulevard").open();
```

関連事項

applicationName, プレーヤ

aspectRatio

使用法

```
-- Lingo syntax
dvdObjRef.aspectRatio

// JavaScript syntax
dvdObjRef.aspectRatio;
```

解説

DVD プロパティ。DVD キャストメンバの幅と高さを指定するプロパティリストを返します。読み取り専用。

幅と高さの両方を整数で返します。

例文

次のステートメントは、メンバ 1 の aspectRatio を返します。

```
-- Lingo syntax
trace(member(1).aspectRatio) -- [#width: 16, #height:9]

// JavaScript syntax
trace(member(1).aspectRatio); // ["width": 16, "height":9];
```

関連事項

DVD

attenuation

使用法

```
member(whichCastMember).light(whichLight).attenuation
```

解説

3D ライトプロパティ。スポットライトおよびポイントライトの定数、リニア、および二次減衰定数を示します。

このプロパティのデフォルト値は vector(1.0, 0.0, 0.0) です。

例文

次のステートメントは、"HouseLight" というライトの attenuation プロパティを vector (.5, 0, 0) に設定します。これにより、わずかに暗くなります。

```
-- Lingo syntax
member("3d world").light("HouseLight").attenuation = vector(.5, 0, 0)

// JavaScript syntax
member("3d world").getProp("light",1).attenuation = vector(.5, 0, 0);
```

関連事項

color (light)

attributeName

使用法

```
XMLnode.attributeName[ attributeNumber ]
```

解説

XML プロパティ。構文解析済み XML ドキュメントの指定された子ノードの名前を返します。

例文

次の XML から始まる例 :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

次の Lingo は、tagName と呼ぶタグの最初の属性の名前を返します。

```
put gParserObject.child[1].child[1].attributeName[1]
-- "attr1"
```

関連事項

attributeValue

attributeValue

使用法

```
XMLnode.attributeValue[ attributeNameOrNumber ]
```

解説

XML プロパティ。構文解析済み XML ドキュメントの指定された子ノードの値を返します。

例文

次の XML から始まる例 :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2">
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

次の Lingo は、tagName と呼ぶタグの最初の属性の値を返します。

```
put gParserObject.child[1].child[1].attributeValue[1]
-- "val1"
```

関連事項

attributeName

audio (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.audio

// JavaScript syntax
dvdObjRef.audio;
```

解説

DVD プロパティ。オーディオが有効（TRUE、デフォルト）であるか無効（FALSE）であるかを指定します。読み取り / 書き込み可能。

例文

次のステートメントはオーディオを無効にします。

```
-- Lingo syntax
member(14).audio = 0

// JavaScript syntax
member(14).audio = 0;
```

関連事項

DVD

audio (MP4Media/FLV)

使用法

```
member(1).audio = true
Sprite(1).audio = true
```

解説

MP4Media/FLV のプロパティ。再生中にオーディオが有効（True、デフォルト）か無効（False）かを指定します。このプロパティは読み書き可能です。

例文

次の例は、スプライト 2 とキャストメンバ MP4Media/FLV の audio プロパティを True に設定します。ムービーを実行すると、MP4Media/FLV ストリームのオーディオ部分が再生されます。

```
-- Lingo syntax
put(sprite(2).audio) -- 1
put(member("MP4Media/FLV").audio) -- 1

// JavaScript syntax
put(sprite(2).audio); // 1
put(member("MP4Media/FLV").audio); // 1
```

次の例は、スプライト 2 およびキャストメンバ MP4Media/FLV の audio プロパティを False に設定します。ムービーを実行すると、MP4Media/FLV ストリームのオーディオ部分は再生されません。

```
-- Lingo syntax
sprite(2).audio = FALSE
member("MP4Media/FLV").audio = FALSE

// JavaScript syntax
sprite(2).audio = 0;
member("MP4Media/FLV").audio = 0;
```

audio (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.audio

// JavaScript syntax
memberOrSpriteObjRef.audio;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。RealMedia ストリームでオーディオを再生 (TRUE) またはミュート (FALSE) します。このプロパティのデフォルト設定は TRUE (1) です。1 または 0 以外の整数値は、TRUE (1) として扱われます。realPlayerNativeAudio() メソッドが TRUE に設定されている場合は、このプロパティは無効です。

RealMedia キャストメンバが再生を開始したときに audio プロパティが FALSE に設定されていると、サウンドチャネルは割り当てられたままになり、再生中にサウンドのオンとオフを切り替えられます。

このとき、プロパティの設定に待ち時間が少しかかり、サウンドのオンとオフの切り替えが遅れることがあります。

例文

次の例は、スプライト 2 とキャストメンバ "Real" の audio プロパティが TRUE に設定されていることを示します。この場合、RealMedia ストリームのオーディオ部分は再生されます。

```
-- Lingo syntax
put(sprite(2).audio) -- 1
put(member("Real").audio) -- 1

// JavaScript syntax
put(sprite(2).audio); // 1
put(member("Real").audio); // 1
```

次の例は、スプライト 2 とキャストメンバ "Real" の audio プロパティが FALSE に設定されていることを示します。この場合、RealMedia ストリームのオーディオ部分は再生されません。

```
-- Lingo syntax
sprite(2).audio = FALSE
member("Real").audio = FALSE

// JavaScript syntax
sprite(2).audio = 0;
member("Real").audio = 0;
```

関連事項

soundChannel (RealMedia), video (RealMedia, Windows Media), sound （プレーヤ）

audio (Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.audio

// JavaScript syntax
windowsMediaObjRef.audio;
```

解説

Windows Media のプロパティ。再生中にオーディオが有効（TRUE、デフォルト）であるか無効（FALSE）であるかを指定します。読み取り / 書き込み可能。

例文

次のステートメントは、キャストメンバ 5 に対してオーディオが有効であるかどうかをメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(member(5).audio)

// JavaScript syntax
trace(member(5).audio);
```

関連事項

Windows Media

audioChannelCount

使用法

```
-- Lingo syntax
dvdObjRef.audioChannelCount

// JavaScript syntax
dvdObjRef.audioChannelCount;
```

解説

オーディオチャネルの数を返す DVD プロパティです。読み取り専用。

例文

次のステートメントは、オーディオチャネルの数を返します。

```
-- Lingo syntax
member(1).audioChannelCount

// JavaScript syntax
member(1).audioChannelCount;
```

関連事項

DVD

audioExtension

使用法

```
-- Lingo syntax
dvdObjRef.audioExtension

// JavaScript syntax
dvdObjRef.audioExtension;
```

解説

DVD プロパティ。オーディオストリームのオーディオ拡張機能がある場合は、それを示すシンボルを返します。読み取り専用。

戻り値は次のとおりです。

シンボル	解説
#caption	オーディオストリームにはキャプションが含まれています。
#lowvision	オーディオストリームには、目が不自由な人向けのコンテンツが含まれています。
#directorcomments1	オーディオストリームには、"Director コメント 1" が含まれています。
#directorcomments2	オーディオストリームには、"Director コメント 2" が含まれています。
#none	DVD では、このオーディオストリームに対してオーディオ拡張機能を指定していないか、拡張機能について判断できませんでした。

関連事項

DVD

audioFormat

使用法

```
-- Lingo syntax
dvdObjRef.audioFormat

// JavaScript syntax
dvdObjRef.audioFormat;
```

解説

DVD プロパティ。オーディオストリームの形式（エンコードモード）を示すシンボルを返します。読み取り専用。

戻り値は次のとおりです。

シンボル	解説
#AC3	オーディオ形式は Dolby AC-3 です。
#MPEG1	オーディオ形式は MPEG-1 です。
#MPEG1DRC	オーディオ形式はダイナミックレンジ制御の MPEG-1 です。
#MPEG2	オーディオ形式は MPEG-2 です。
#MPEG2DRC	オーディオ形式はダイナミックレンジ制御の MPEG-2 です。
#LPCM	オーディオ形式は LPCM（Linear Pulse Code Modulated）です。
#DTS	オーディオ形式は DTS（Digital Theater Systems）です。
#SDDS	オーディオ形式は SDDS（Sony Dynamic Digital Sound）です。

関連事項
DVD

audioSampleRate

使用法

```
-- Lingo syntax
dvdObjRef.audioSampleRate

// JavaScript syntax
dvdObjRef.audioSampleRate;
```

解説
オーディオストリームの周波数をヘルツ単位で返す DVD プロパティです。読み取り専用。

関連事項
DVD

audioStream

使用法

```
-- Lingo syntax
dvdObjRef.audioStream

// JavaScript syntax
dvdObjRef.audioStream;
```

解説
DVD プロパティ。現在アクティブなオーディオストリームを返します。読み取り / 書き込み可能。

有効な値の範囲は 1 ～ 8 です。

関連事項
DVD

audioStreamCount

使用法

```
-- Lingo syntax
dvdObjRef.audioStreamCount

// JavaScript syntax
dvdObjRef.audioStreamCount;
```

解説

現行タイトルで使用できるオーディオストリームの数を返す DVD プロパティです。読み取り専用。

使用できるオーディオストリームの数の範囲は、1 ～ 8 です。

関連事項

DVD

auto

使用法

```
member(whichCastmember).model(whichModel).lod.auto
```

解説

3D lod モディファイアプロパティ。モデルとカメラの距離の変化に応じて、モデルの詳細を削減する度合いを管理します。

auto プロパティが TRUE の場合は、モディファイアの bias プロパティの設定により、モディファイアがモデルの詳細を削除する度合いが決まります。

モディファイアは、モデルのレベルオブディテールを調整するときに level プロパティを更新します。auto プロパティが FALSE に設定されている場合は、level プロパティを設定しても効果はありません。

#lod モディファイアは、Director 以外の 3D モデリングプログラムで作成されたモデルに対してのみ追加できます。このようなモデルで使用されるモデルリソースの type プロパティは、#fromFile です。このモディファイアは、Director で作成したプリミティブには追加できません。

例文

次のステートメントは、" 宇宙船 " というモデルの lod モディファイアの auto プロパティを TRUE に設定します。モディファイアは、自動的にモデルのレベルオブディテールを設定します。

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.auto = TRUE

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).auto = true;
```

関連事項

lod （モディファイア）, bias, level

autoblend

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.autoblend
member(whichCastmember).model(whichModel).bonesPlayer.autoblend
```

解説

3D keyframePlayer および bonesPlayer モディファイアプロパティ。モディファイアが、前のモーションから現在再生中のモーションへのリニアトランジションを作成するか (TRUE)、または作成しないか (FALSE) を示します。autoBlend が TRUE の場合、トランジションの長さはモディファイアの blendTime プロパティによって設定されます。autoBlend が FALSE の場合、トランジションはモディファイアの blendFactor プロパティで制御され、blendTime は無視されます。

blendTime が 0 に設定され、autoBlend が TRUE に設定された場合、モーションのブレンドは完全に無効になります。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" エイリアン 3" というモデルの autoblend をオフにします。モデルの blendFactor 設定は、プレイリストの連続するモーションのブレンドに使用されます。

```
-- Lingo syntax
member("newaliens").model("Alien3").addModifier(#keyframeplayer)
member("newaliens").model("Alien3").keyframePlayer.autoblend = FALSE

// JavaScript syntax
member("newaliens").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("newaliens").getPropRef("model",1).getPropRef("keyframeplayer",1).autoBlend = false;
```

関連事項

blendFactor, blendTime

autoCameraPosition

使用法

```
member(whichTextCastmember).autoCameraPosition
```

解説

3D カメラプロパティ。3D テキストキャストメンバのカメラを、すべてのテキストを表示するように自動的に配置するか（TRUE）、自動で配置しないか（FALSE）を示します。このプロパティは、テキスト、フォント、フォントサイズ、およびキャストメンバのその他のプロパティを変更するときに役立ちます。

このプロパティは、他の種類の 3D キャストメンバでは有効でありません。

例文

次のステートメントは、" ヘッドライン " というキャストメンバの autoCameraPosition プロパティを FALSE に設定します。キャストメンバが 3D モードで表示されるときに、カメラは自動的には配置されません。

```
-- Lingo syntax
member("Headline").autoCameraPosition = FALSE

// JavaScript syntax
member("Headline").autoCameraPosition = false;
```

関連事項
displayMode

autoMask

使用法

```
member(whichCursorCastMember).autoMask
the autoMask of member whichCastMember
```

解説

アニメーションカラー [**whichCursorCastMember**] の白いピクセルを透過にするかどうかを設定するキャストメンバプロパティです。TRUE, (デフォルト) にすると透過になり、その部分の背景が透けて見えます。FALSE にすると不透明になります。

例文

次のスクリプトでは、キャストメンバ 5 のカスタムアニメーションカーソルがスプライトに入ったとき、autoMask を TRUE にし、白いピクセル部分のスプライト背景が透けて見えるようになります。カーソルがスプライトを離れると autoMask は FALSE になります。

```
-- Lingo syntax
on mouseEnter
    member 5.autoMask = TRUE
end

on mouseLeave
    member 5.autoMask = FALSE
end
```

従来からの Lingo シンタックスを使用する場合は、次のように記述します。

```
on mouseEnter
    set the autoMask of member 5 = TRUE
end

on mouseLeave
    set the autoMask of member 5 = FALSE
end
```

autoTab

使用法

```
-- Lingo syntax
memberObjRef.autoTab

// JavaScript syntax
memberObjRef.autoTab;
```

解説

ユーザが Tab キーを押したときに [キャストメンバ] で指定したフィールドキャストメンバの編集可能なフィールドやテキストキャストメンバを、アクティブ (TRUE) にするか非アクティブ (FALSE) にするかを定義するキャストメンバプロパティです。プロパティは、アクティブ (TRUE) または非アクティブ (FALSE) に設定可能です。タブ順はスプライト番号により決定され、スプライトのステージ上の位置には左右されません。

例文

次のステートメントは、ユーザが Tab キーを押したときにキャストメンバ " コメント " の次の編集可能なフィールドまたはテキストスプライトに挿入ポイントを移動します。

```
--Lingo syntax
member ("Comments").autotab = TRUE

// JavaScript syntax
member ("Comments").autotab = true;
```

axisAngle

使用法

```
member(whichCastmember).model(whichModel).transform.axisAngle
member(whichCastmember).camera(whichCamera).transform.axisAngle
member(whichCastmember).light(whichLight).transform.axisAngle
member(whichCastmember).group(whichGroup).transform.axisAngle
transformReference.axisAngle
```

解説

3D トランスフォームプロパティ。トランスフォームの回転を、軸と角度の組み合わせで示します。

axisAngle プロパティは、ベクトル (軸) と浮動小数点値 (角度) からなるリニアリストです。ベクトルはトランスフォームの回転軸です。小数点値は回転する角度を度数で表します。

このプロパティのデフォルト値は [vector(1.0000, 0.0000, 0.0000), 0.0000] です。

例文

次のステートメントは、" 郵便受け " というモデルの回転を axisAngle で示します。モデルは、y 軸を中心に 145.5° 回転します。

```
-- Lingo syntax
put member("Yard").model("Mailbox").transform.axisAngle
-- [vector( 0.0000, 1.0000, 0.0000 ), -145.5000]

// JavaScript syntax
put(member("Yard").getProp("model",1).transform.axisAngle);
// <[vector( 0.0000, 1.0000, 0.0000 ), -145.5000]>
```

関連事項

rotation （トランスフォーム）

back

使用法

```
member(whichCastmember).modelResource(whichModelResource).back
```

解説

3D #box モデルリソースプロパティ。ボックスの +Z 軸と交差する側面が、閉じた状態であるか (TRUE)、または開いた状態であるか (FALSE) を示します。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" 木箱 " というモデルリソースの back プロパティを FALSE に設定します。つまり、このボックスの裏面は開いた状態になります。

```
-- Lingo syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"))
member("3D World").modelResource("Crate").back = FALSE

// JavaScript syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"));
member("3D World").getProp("modelresource",10).back = false;
```

関連事項

bottom (3D), front, top (3D), left (3D), right (3D)

backColor

使用法

```
-- Lingo syntax
spriteObjRef.backColor

// JavaScript syntax
spriteObjRef.backColor;
```

解説

指定したスプライトのバックカラーを、カラー値で設定するスプライトプロパティです。読み取り / 書き込み可能。

backColor の設定は、ステージ上でスプライトを選択してツールパレットからバックカラーを選択するのと同じことです。スクリプトで設定した値を現行スプライトの終了後も維持するには、スプライトをスクリプトスプライトにしておく必要があります。バックカラーは、フィールド、ボタン、チェックボックス、ラジオの各キャストメンバに加えて、すべてのビットマップキャストメンバに適用されます。

backColor の値の範囲は、8 ビットカラーの場合は 0 ～ 255、4 ビットカラーの場合は 0 ～ 15 になります。番号は、現行パレット内のバックカラーのインデックス番号に相当します。各色のインデックス番号は、パレットウィンドウ内の任意の色をクリックすると、ウィンドウの左下に表示されます。

1 ビットより深いビットマップキャストメンバにこのプロパティを設定する場合で、ビットマップの背景が表示されないときは、backColor が表示されないことがあります。

スプライトのブレンドが 100 より小さく 0 より大きい場合、backColor は透明色とミックスされます。

注意：backColor プロパティの代わりに、新しい bgColor プロパティを使用することをお勧めします。

例文

次のステートメントは、変数 oldColor にスプライト 5 のバックカラーの値を設定します。

```
-- Lingo syntax
oldColor = sprite(5).backColor

// JavaScript syntax
var oldColor = sprite(5).backColor;
```

次のステートメントは、スプライト 11 ～ 13 までのランダムに指定したスプライトのバックカラーをカラー番号 36 に変更します。

```
-- Lingo syntax
sprite(10 + random(3)).backColor = 36

// JavaScript syntax
sprite(10 + random(3)).backColor = 36;
```

関連事項

スプライト

backdrop

使用法

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.backdrop[index].source
member(whichCastmember).camera(whichCamera).backdrop[index].source
sprite(whichSprite).camera{(index)}.backdrop[index].scale
member(whichCastmember).camera(whichCamera).backdrop[index].scale
sprite(whichSprite).camera{(index)}.backdrop[index].rotation
member(whichCastmember).camera(whichCamera).backdrop[index].rotation
sprite(whichSprite).camera{(index)}.backdrop[index].regPoint
member(whichCastmember).camera(whichCamera).backdrop[index].regPoint
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
```

解説

3D カメラプロパティ。カメラの投影平面上にレンダリングされる 2D 画像です。カメラのビューにあるモデルはすべて、backdrop (背景) の前に表示されます。

backdrop には次のプロパティがあります。

注意：これらのプロパティは、オーバーレイの取得、設定、および操作にも使用できます。詳細については、各プロパティのエントリを参照してください。

[loc (backdrop and overlay)] は、スプライトの左上隅から測定した背景の 2D 位置を示します。

[source] は、backdrop で使用されるテクスチャを示します。

[scale (backdrop and overlay)] は、backdrop のサイズを決定するために、テクスチャの高さと幅に掛ける数値です。

rotation (backdrop and overlay) は、背景が regPoint の周りを回転する量です。

[regPoint (3D)] は、backdrop のセンターポイントを示します。

[blend (3D)] は、backdrop の不透明性を示します。

[count (3D)] は、カメラの背景リストにある項目数を示します。

背景を作成および削除するには、次のコマンドを使用します。

[addBackdrop] は、テクスチャから背景（backdrop）を作成し、それをカメラの背景リストの最後に追加します。

[insertBackdrop] は、テクスチャから背景（backdrop）を作成し、それをカメラの背景リストの指定したインデックス位置に追加します。

[removeBackdrop] は背景（backdrop）を削除します。

例文

```
-- Lingo syntax
put sprite(2).camera.backdrop[1].scale
//JavaScript syntax
put (sprite(2).camera.getPropRef("backdrop",1).scale);
```

関連事項

overlay

insertBackdrop

removeBackdrop

backgroundColor

使用法

```
-- Lingo syntax
memberObjRef.backgroundColor

// JavaScript syntax
memberObjRef.backgroundColor;
```

解説

指定したキャストメンバまたはスプライトのバックカラーを、RGB カラー値で設定するベクトルシェイプキャストメンバプロパティです。

このプロパティは、値を調べたり、変更できます。

例文

```
-- Lingo syntax
member("Archie").backgroundColor= color(255,255,255)
// JavaScript syntax
member("Archie").backgroundColor= color(255,255,255);
```

関連事項

bgColor （ウィンドウ）

beepOn

使用法

```
-- Lingo syntax
_movie.beepOn

// JavaScript syntax
_movie.beepOn;
```

解説

アクティブなスプライトの外側をクリックしたときに警告音を鳴らす (TRUE) か鳴らさない (FALSE、デフォルト) かを定義するムービープロパティです。読み取り / 書き込み可能。

beepOn プロパティを設定するスクリプトは、フレームまたはムービースクリプトに置きます。

例文

次のステートメントは、beepOn プロパティを TRUE に設定します。

```
-- Lingo syntax
_movie.beepOn = TRUE

// JavaScript syntax
_movie.beepOn = true;
```

次のステートメントは、beepOn の設定を逆転します。

```
-- Lingo syntax
_movie.beepOn = not(_movie.beepOn)

// JavaScript syntax
_movie.beepOn = !(_movie.beepOn);
```

関連事項

ムービー

bevelDepth

使用法

```
member(whichTextCastmember).bevelDepth
member(which3DCastmember).modelResource(whichModelResource).bevelDepth
```

解説

3D テキストプロパティ。3D テキストのベベル量の度合いを示します。

メンバの displayMode プロパティが #mode3D に設定され、その bevelType プロパティが #miter または #round に設定されていない限り、テキストキャストメンバに対してこのプロパティは無効です。

3D キャストメンバの押し出しテキストの場合、モデルリソースの bevelType プロパティが #miter または #round に設定されていない限り、このプロパティは無効です。

このプロパティの範囲は 0.0 ～ 10.0 で、デフォルト値は 10.0 です。

例文

次の例で、" ロゴ " というキャストメンバはテキストキャストメンバです。次のステートメントで、ロゴの bevelDepth を 5.5 に設定します。ロゴが 3D モードで表示されるときに、bevelType プロパティが #miter または #round に設定されていれば、文字のエッジが斜めになります。

```
-- Lingo syntax
member("Logo").bevelDepth = 5.5

// JavaScript syntax
member("Logo").bevelDepth = 5.5;
```

次の例では、モデル " スローガン " のモデルリソースは押し出しテキストです。次のステートメントは、" スローガン " のモデルリソースの bevelDepth を 5 に設定します。" スローガン " の bevelType プロパティが #miter または #round に設定されている場合、文字のエッジは斜めになります。

```
-- Lingo syntax
member("scene").model("Slogan").resource.bevelDepth = 5
```

関連事項

bevelType, extrude3D, displayMode

bevelType

使用法

```
member(whichTextCastmember).bevelType
member(which3DCastmember).modelResource(whichModelResource).bevelType
```

解説

3D テキストプロパティ。3D テキストに適用されるベベルのスタイルを示します。

テキストキャストメンバの場合、これはメンバプロパティです。3D キャストメンバの押し出しテキストの場合、これはモデルリソースプロパティです。

bevelType プロパティに設定可能な値は次のとおりです。

- #none
- #miter (デフォルト)
- #round

例文

次の例で、" ロゴ " というキャストメンバはテキストキャストメンバです。次のステートメントは、" ロゴ " の bevelType を #round に設定します。

```
member("logo").beveltype = #round
```

次の例では、モデル " スローガン " のモデルリソースは押し出しテキストです。次のステートメントは、" スローガン " のモデルリソースの bevelType を #miter に設定します。

```
member("scene").model("Slogan").resource.bevelType = #miter
```

関連事項

bevelDepth, extrude3D, displayMode

bgColor （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.bgColor

// JavaScript syntax
windowObjRef.bgColor;
```

解説

ウィンドウのバックカラーを決定するウィンドウプロパティです。読み取り / 書き込み可能。

bgColor プロパティの設定は、ムービープロパティダイアログボックスで色を設定するのと同じ効果があります。

例文

次の例文は、" アニマル " という名前のウィンドウのカラーを RGB 値で設定します。

```
-- Lingo syntax
window("Animals").bgColor = color(255, 153, 0)

// JavaScript syntax
window("Animals").bgColor = color(255, 153, 0);
```

関連事項

ウィンドウ

bgColor （スプライト、3D メンバ）

使用法

```
sprite(whichSpriteNumber).bgColor
the bgColor of sprite whichSpriteNumber
the bgColor of the stage
(the stage).bgColor
member(which3dMember).bgcolor
```

解説

スプライトプロパティ、システムプロパティ、および 3D キャストメンバプロパティです。[スプライト番号] で指定されたスプライトのバックカラー、ステージのカラー、または 3D キャストメンバのバックカラーを定義します。bgColor の設定は、ステージでスプライトを選択し、ツールウィンドウでバックカラーを設定するのと同じ効果があります。ステージの bgColor プロパティの設定は、[ムービープロパティ] ダイアログボックスで色を設定するのと同じ効果があります。

このプロパティは、backColor と同じ機能ですが、スプライトがどのようなタイプであってもカラーオブジェクトとしてカラー値が返される点が異なります。

このプロパティは、値を調べたり、設定することができます。

例文

次の例文は、ステージのカラーを RGB 値で設定します。

ドットシンタックスの場合 :

```
(the stage).bgColor = rgb(255, 153, 0)
```

バーボーズ Lingo シンタックスの場合 :

```
set the bgColor of the stage = rgb(255, 153, 0)
```

関連事項

color(), backColor, backgroundColor

bias

使用法

```
member(whichCastmember).model(whichModel).lod.bias
```

解説

3D lod モディファイアプロパティ。auto プロパティが TRUE に設定されている場合に、モディファイアがモデルの詳細を削除する度合いを示します。モディファイアの auto プロパティが FALSE に設定されている場合、このプロパティは無効です。

このプロパティの範囲は、0.0 (すべてのポリゴンを削除) ～ +100.0 (ポリゴンを削除しない) です。デフォルトの設定は 100.0 です。

#lod モディファイアは、Director 以外の 3D モデリングプログラムで作成されたモデルに対してのみ追加できます。このようなモデルで使用されるモデルリソースの type プロパティは、#fromFile です。このモディファイアは、Director で作成したプリミティブには追加できません。

例文

次のステートメントは、" 宇宙船 " というモデルの lod モディファイアの bias プロパティを 10 に設定します。lod モディファイアの auto プロパティが TRUE に設定されている場合、" 宇宙船 " がカメラから離れていくときに、モディファイアはレベルオブディテールを急激に低くします。

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.bias = 10

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).bias = 10;
```

関連事項

lod （モディファイア）, auto, level

bitDepth（ミクサ）

使用法

```
mixer.bitDepth (Read-write)
```

解説

ミクサプロパティ。ミクサのビット深度を返すか設定します。

例文

```
-- Lingo syntax
on mouseUp me
mixerRef.bitdepth = 16 -- Sets the bit depth of the mixer to 16.
end

// JavaScript syntax
function mouseUp(){
mixerRef.bitdepth = 16; // Sets the bit depth of the mixer to 16.
}
```

関連事項

ミクサ

bitDepth（サウンドオブジェクト）

使用法

```
SoundObjectRef.bitDepth (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトのビット深度を返します。

例文

```
--Lingo syntax
on mouseUp me
put SoundObjectRef.bitdepth   -- Returns the bit depth of the sound object.
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.bitdepth); // Returns the bit depth of the sound object.
}
```

bitmapSizes

使用法

```
-- Lingo syntax
memberObjRef.bitmapSizes

// JavaScript syntax
memberObjRef.bitmapSizes;
```

解説

フォントキャストメンバが作成されたときに組み込まれたビットマップポイントサイズのリストを返すフォントキャストメンバです。

例文

次のステートメントは、フォントキャストメンバ 11 が作成されたときに組み込まれたビットマップポイントサイズを表示します。

```
-- Lingo syntax
put(member(11).bitmapSizes)

// JavaScript syntax
put(member(11).bitmapSizes);
```

関連事項

recordFont, characterSet, originalFont

bitRate

使用法

```
-- Lingo syntax
memberObjRef.bitRate

// JavaScript syntax
memberObjRef.bitRate;
```

解説

サーバからプリロードされた指定の Shockwave Audio(SWA) キャストメンバのビットレートを返すサウンドのキャストメンバプロパティです。値の単位は Kbps (キロバイト / 秒) です。

ストリーミングが開始されるまでの間は、bitRate メンバプロパティは 0 を返します。

例文

次のビヘイビアーは、スプライトを最初に検出したときに、SWA キャストメンバのビットレートを表示します。

```
-- Lingo syntax
property spriteNum

on beginSprite (me)
    memName = sprite(spriteNum).member.name
    put("The bitRate of member"&&memName&&"is"&&member(memName).bitRate)
end

// JavaScript syntax
function beginSprite() {
    var memName = sprite(spriteNum).member.name;
    put("The bitRate of member " + memName +" is " + member(memName).bitRate);
}
```

bitsPerSample

使用法

```
-- Lingo syntax
memberObjRef.bitsPerSample

// JavaScript syntax
memberObjRef.bitsPerSample;
```

解説

この Shockwave Audio (SWA) キャストメンバプロパティは、SWA にエンコードされたオリジナルのファイルのビット深度を示します。SWA サウンドの再生が開始されたか、または preLoadBuffer コマンドを使ってファイルがプリロードされた後にのみ、このプロパティは有効になります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、SWA ストリーミングのキャストメンバ "Paul Robeson" に使用されたファイルのオリジナルのビットレートを、フィールドキャストメンバ "How Deep" に割り当てます。

```
-- Lingo syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample

// JavaScript syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample;
```

blend (3D)

使用法

```
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.overlay[index].blend
member(whichCastmember).camera(whichCamera).overlay[index].blend
member(whichCastmember).shader(whichShader).blend
member(whichCastmember).model(whichModel).shader.blend
member(whichCastmember).model(whichModel).shaderList{[index]}.blend
```

解説

3D 背景、オーバーレイ、および #standard シェーダプロパティ。背景、オーバーレイ、シェーダの不透明度を示します。

シェーダの transparent プロパティが TRUE に設定されていない限り、シェーダの blend プロパティの設定は無効です。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 100 です。

例文

次のステートメントは、" 窓 " というモデルでシェーダの blend プロパティを 80 に設定します。" 窓 " のシェーダの transparent プロパティが TRUE に設定されていれば、モデルはわずかに透明になります。

```
-- Lingo syntax
member("House").model("Window").shader.blend = 80

// Java Script
member("House").getPropRef("model", 1).shaderList[1].blend = 80;
```

関連事項

bevelDepth, overlay, shadowPercentage, transparent

blend （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.blend

// JavaScript syntax
spriteObjRef.blend;
```

解説

スプライトのブレンド値を定義するスプライトプロパティです。ブレンド値は 0 ～ 100 の間で設定します。この値はスプライトプロパティダイアログボックス内のブレンド値と一致します。読み取り / 書き込み可能。

使用可能なカラーは、モニタの色深度とは関係なく、パレット内のカラーに依存します。

高品質なブレンドを得るには、8 ビット以上の色深度のブレンドインクを使用します。

例文

次のステートメントは、スプライト 3 のブレンド値を 40% に設定します。

```
-- Lingo syntax
sprite(3).blend = 40

// JavaScript syntax
sprite(3).blend = 40;
```

次のステートメントは、スプライト 3 のブレンド値をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(3).blend)

// JavaScript syntax
put(sprite(3).blend);
```

関連事項

blendLevel, スプライト

blendConstant

使用法

```
member(whichCastmember).shader(whichShader).blendConstant
member(whichCastmember).model(whichModel).shader.blendConstant
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstant
```

解説

3D #standard シェーダプロパティ。シェーダの最初のテクスチャレイヤーで使用されるブレンド率を示します。

シェーダの useDiffuseWithTexture プロパティが TRUE に設定されている場合、テクスチャはシェーダの diffuse プロパティで設定されているカラーとブレンドされます。useDiffuseWithTexture が FALSE の場合は、ブレンドに白が使用されます。

他のテクスチャレイヤーは、それぞれ下位のテクスチャレイヤーとブレンドされます。これらのテクスチャレイヤーのブレンドを制御するには、blendConstantList プロパティを使用します。

blendConstant プロパティが有効であるのは、シェーダの blendSource プロパティが #constant に設定されている場合のみです。詳細については、blendSource および blendSourceList を参照してください。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 50 です。

例文

次の例では、"MysteryBox" というモデルのシェーダリストに、6 つのシェーダがあります。次のステートメントは、2 番目のシェーダの blendConstant プロパティを 20 に設定します。このプロパティは、blendFunction、blendFunctionList、blendSource、および blendSourceList の各プロパティの設定の影響を受けます。

```
member("Level2").model("MysteryBox").shaderList[2].blendConstant = 20
```

関連事項

blendConstantList, blendFunction, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendConstantList

使用法

```
member(whichCastmember).shader(whichShader).blendConstantList
member(whichCastmember).model(whichModel).shader.blendConstantList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstantList{[index]}
```

解説

3D #standard シェーダプロパティ。シェーダのテクスチャレイヤーと、下位のテクスチャレイヤーとのブレンドに使用する比率を示します。

シェーダのテクスチャリストおよびブレンド定数リストには、どちらも 8 つのインデックス位置があります。ブレンド定数リストの各インデックス位置は、テクスチャリストの対応するインデックス位置にあるテクスチャのブレンドを制御します。オプションのインデックスパラメータを指定しない場合、リストのすべてのインデックス位置は一度に同じ値に設定されます。リストのインデックス位置を一度に 1 つずつ設定するには、インデックスパラメータを使用します。

blendConstantList プロパティは、対応するテクスチャレイヤーの blendSource プロパティが #constant に設定されている場合にのみ機能します。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 50 です。

例文

次の例では、"MysteryBox" というモデルのシェーダリストに、6 つのシェーダがあります。次のステートメントは、2 番目のシェーダで使用される各テクスチャの blendConstant プロパティを示します。このプロパティは、blendFunction、blendFunctionList、blendSource、および blendSourceList プロパティの設定の影響を受けます。

```
-- Lingo syntax
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]

// JavaScript syntax
put(member("Level2").getPropRef("model",3).shaderList[1].blendConstantList);
// <[20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]>
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]
```

関連事項

blendConstant, blendFunction, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendFactor

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.blendFactor
member(whichCastmember).model(whichModel).bonesPlayer.blendFactor
```

解説

3D keyframePlayer および bonesPlayer モディファイアプロパティ。前のモーションと組み合わせるモーションの量を示します。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 0 です。

BlendFactor は、モディファイアの autoblend プロパティが FALSE に設定されている場合にのみ機能します。blendFactor プロパティの値が 100 の場合、現在のモーションは前のモーションの特性をまったく持ちません。blendFactor の値が 0 の場合、現在のモーションは前のモーションの特性をすべて持ち、独自の特性は持ちません。blendFactor の値が 50 の場合、現在のモーションと前のモーションの特性が均等に合成されます。blendFactor の値は、モディファイアの autoblend プロパティが TRUE に設定されているときに作成されるリニアトランジションとは異なり、トランジションを作成するときに変化します。

例文

次のステートメントは、モデル " エイリアン 3" の blendFactor プロパティを 50 に設定します。モディファイアの autoblend プロパティが FALSE の場合、" エイリアン 3" の keyframePlayer プレイリストの各モーションは、前のモーションと均一に混合されます。

```
member("newaliens").model("Alien3").keyframePlayer.blendFactor = 50
```

関連事項

autoblend, keyframePlayer （モディファイア）

blendFunction

使用法

```
member(whichCastmember).shader(whichShader).blendFunction
member(whichCastmember).model(whichModel).shader.blendFunction
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunction
```

解説

3D #standard シェーダプロパティ。シェーダの最初のテクスチャレイヤーで使用されるブレンドのタイプを示します。

シェーダの useDiffuseWithTexture プロパティが TRUE に設定されている場合、テクスチャはシェーダの diffuse プロパティで設定されているカラーとブレンドされます。useDiffuseWithTexture が FALSE の場合は、ブレンドに白が使用されます。

他のテクスチャレイヤーは、それぞれ下位のテクスチャレイヤーとブレンドされます。これらのテクスチャレイヤーのブレンドを制御するには、blendFunctionList プロパティを使用します。

blendFunction プロパティは、次の値をとることができます。

#multiply は、テクスチャレイヤーの RGB 値と、ブレンドに使用されているカラーを乗算します (上記参照)。

#add は、テクスチャレイヤーの RGB 値を、ブレンドに使用されているカラーに加算します。値の上限は 255 です。

#replace は、シェーダの diffuse プロパティで設定されたカラーと、テクスチャがブレンドされないようにします。

#blend は、テクスチャレイヤーのカラーとブレンドに使用されているカラーを、blendConstant プロパティで設定された率で混合します。

このプロパティのデフォルト値は #multiply です。

例文

次の例では、"MysteryBox" というモデルのシェーダリストに、6 つのシェーダがあります。次のステートメントは、2 番目のシェーダの blendFunction プロパティを #blend に設定します。これにより、blendSource、blendSourceList、blendConstant、blendConstantList の各プロパティの設定が有効になります。

```
member("Level2").model("MysteryBox").shaderList[2].blendFunction = #blend
```

関連事項

blendConstant, blendConstantList, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendFunctionList

使用法

```
member(whichCastmember).shader(whichShader).blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shader.blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunctionList{[index]}
```

解説

3D #standard シェーダプロパティ。各テクスチャレイヤーとその下位にあるテクスチャレイヤーをブレンドする方法を示すリニアリストです。

シェーダのテクスチャリストおよびブレンド関数リストには、どちらも 8 つのインデックス位置があります。ブレンド関数リストの各インデックス位置は、テクスチャリストの対応するインデックス位置にあるテクスチャのブレンドを制御します。オプションのインデックスパラメータを指定しない場合、リストのすべてのインデックス位置は一度に同じ値に設定されます。リストのインデックス位置を一度に 1 つずつ設定するには、インデックスパラメータを使用します。

ブレンド関数リストの各インデックス位置には、次の値のいずれかを設定できます。

#multiply は、テクスチャレイヤーの RGB 値と、下位のテクスチャレイヤーの RGB 値を乗算します。

#add は、テクスチャレイヤーの RGB 値を、下位のテクスチャレイヤーの RGB 値に加算します。値の上限は 255 になります。

#replace は、テクスチャで下位のテクスチャレイヤーを覆います。ブレンドは発生しません。

#blend は、ブレンドを blendSource プロパティの値で制御します。これにより、アルファブレンドが許可されます。

このプロパティのデフォルト値は #multiply です。

例文

次の例では、MysteryBox というモデルの shaderList プロパティに、6 つのシェーダがあります。次のステートメントは、2 番目のシェーダの blendFunctionList プロパティの 4 番目のインデックス位置の値が #blend に設定されていることを示します。このモデルで 2 番目のシェーダ内の 4 番目にあるテクスチャレイヤーのブレンドは、blendSource、blendSourceList、blendConstant、blendConstantList、diffuse、diffuseColor、および useDiffuseWithTexture の各プロパティの設定によって制御されます。

```
put member("Level2").model("MysteryBox").shaderList[2].blendFunctionList[4]
-- #blend
```

関連事項

blendConstant, blendConstantList, blendFunction, blendSource, blendSourceList, diffuse, diffuseColor, useDiffuseWithTexture

blendLevel

使用法

```
sprite(whichSpriteNumber).blendLevel
the blendLevel of sprite whichSpriteNumber
```

解説

スプライトの現行のブレンド値を定義または参照するスプライトプロパティです。値は 0 ～ 255 の間で設定します。スプライトインスペクタは 0 ～ 100 の間で設定するので、範囲は異なりますが、結果は同じになります。

このプロパティは blend スプライトプロパティと同じものです。

例文

```
sprite(3).blendlevel = 99
```

関連事項

blend （スプライト）

blendRange

使用法

```
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
```

解説

3D プロパティ。#particle というタイプのモデルリソースで使用すると、モデルリソースのブレンド範囲の開始点と終了点を取得または設定できます。

システム内のパーティクルの不透明性は、blendRange.start から blendRange.end の間、各パーティクルの寿命が終わるまで一定の比率で補間されます。

このプロパティの値は、0.0 以上および 100.0 以下に設定する必要があります。このプロパティのデフォルト値は 100.0 です。

例文

次のステートメントは、#particle というタイプのモデルリソース "ThermoSystem" の、blendRange プロパティを設定します。

1 行目は開始点の値を 100 に設定し、2 行目は終了点の値を 0 に設定します。このステートメントによって、"ThermoSystem" のパーティクルは最初は完全な不透明で表示され、パーティクルの寿命の間に徐々に透明になります。

```
member("Heater").modelResource("ThermoSystem").blendRange.start = 100.0
member("Heater").modelResource("ThermoSystem").blendRange.end = 0.0
```

blendSource

使用法

```
member(whichCastmember).shader(whichShader).blendSource
member(whichCastmember).model(whichModel).shader.blendSource
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSource
```

解説

3D #standard シェーダプロパティ。シェーダのテクスチャリストにある最初のテクスチャレイヤーのブレンドが、テクスチャのアルファ情報と一定比率のどちらに基づくかを示します。

シェーダの useDiffuseWithTexture プロパティが TRUE に設定されている場合、テクスチャはシェーダの diffuse プロパティで設定されているカラーとブレンドされます。useDiffuseWithTexture が FALSE の場合は、ブレンドに白が使用されます。

他のテクスチャレイヤーは、それぞれ下位のテクスチャレイヤーとブレンドされます。これらのテクスチャレイヤーのブレンドを制御するには、blendSourceList プロパティを使用します。

blendSource プロパティが有効であるのは、シェーダの blendFunction プロパティが #blend に設定されている場合のみです。

このプロパティで使用できる値は以下のとおりです。

#alpha は、テクスチャのアルファ情報を使って、テクスチャの各ピクセルとブレンドに使用されているカラーのブレンド率を決定します (上記参照)。

#constant は、シェーダの blendConstant プロパティの値を、テクスチャのすべてのピクセルのブレンド率として使用するようにします。

このプロパティのデフォルト値は #constant です。

例文

次の例では、"MysteryBox" というモデルのシェーダリストに、6 つのシェーダがあります。次のステートメントは、2 番目のシェーダで使用される最初のテクスチャの blendSource プロパティを #constant に設定します。これにより、blendConstant および blendConstantList プロパティの設定が有効になります。

```
-- Lingo syntax
member("Level2").model("MysteryBox").shaderList[2].blendSource = #constant

// JavaScript syntax
member("Level2").getPropRef("model",3).shaderList[2].blendSource = symbol("constant");
```

関連事項

blendSourceList, blendFunction, blendFunctionList, blendConstant, blendConstantList, useDiffuseWithTexture, diffuse, diffuseColor

blendSourceList

使用法

```
member(whichCastmember).shader(whichShader).blendSourceList[index]
member(whichCastmember).model(whichModel).shader.blendSourceList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSourceList{[index]}
```

解説

3D #standard シェーダプロパティ。テクスチャレイヤーとその下位のテクスチャレイヤーのブレンドが、テクスチャのアルファ情報と一定比率のどちらに基づくかを示します。

シェーダのテクスチャリストおよびブレンドソースリストには、どちらも 8 つのインデックス位置があります。ブレンドソースリストの各インデックス位置は、テクスチャリストで対応するインデックス位置にあるテクスチャのブレンドを制御します。オプションのインデックスパラメータを指定しない場合、リストのすべてのインデックス位置は一度に同じ値に設定されます。リストのインデックス位置を一度に 1 つずつ設定するには、インデックスパラメータを使用します。

blendSourceList プロパティは、対応するテクスチャレイヤーの blendFunction プロパティが #blend に設定されている場合にのみ機能します。詳細については、blendFunction および blendFunctionList を参照してください。

このプロパティで使用できる値は以下のとおりです。

#alpha は、テクスチャのアルファ情報で、テクスチャレイヤーの各ピクセルと下位のレイヤーとのブレンド率を決定します。

#constant は、対応するテクスチャレイヤーの blendConstant プロパティの値を、テクスチャレイヤーのすべてのピクセルのブレンド率として使用するようにします。詳細については、blendConstant および blendConstantList を参照してください。

このプロパティのデフォルト値は #constant です。

例文

次の例では、"MysteryBox" というモデルのシェーダリストに、6 つのシェーダがあります。各シェーダには、8 つのテクスチャが含まれるテクスチャリストがあります。次のステートメントは、2 番目のシェーダ内で使用される 4 番目のテクスチャの blendSource プロパティを #constant に設定します。これにより、blendConstant、blendConstantList、useDiffuseWithTexture の各プロパティの設定が有効になります。

```
member("Level2").model("MysteryBox").shaderList[2].blendSourceList[4] = #constant
```

関連事項

blendSource, blendFunction, blendFunctionList, blendConstant, blendConstantList, useDiffuseWithTexture, diffuse, diffuseColor

blendTime

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.blendTime
member(whichCastmember).model(whichModel).bonesPlayer.blendTime
```

解説

3D keyframePlayer および bonesPlayer モディファイアプロパティ。モデルのモディファイアのプレイリストにあるモーション間の継続時間をミリ秒単位で決定します。

blendTime プロパティは、モディファイアの autoBlend プロパティと共に動作します。autoBlend が TRUE に設定されている場合、モディファイアは、前のモーションから現在再生中のモーションへのリニアトランジションを作成します。blendTime プロパティの値は、このトランジションの長さです。autoBlend が FALSE に設定されている場合、blendTime プロパティは無視されます。

このプロパティのデフォルト設定は 500 です。

例文

次のステートメントは、" エイリアン 5" というモデルのモディファイアで、プレイリスト中のモーション間のトランジションの長さを 1200 ミリ秒に設定します。

```
member("newaliens").model("Alien5").keyframePlayer.blendTime = 1200
```

関連事項

autoblend, blendFactor

bone

使用法

```
-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].worldTransform

// JavaScript Usage
member(whichCastmember).getProp("model", whichModelIndex).getPropRef("bonesplayer").getPropRef("bone",
whichBoneIndex).transform
member(whichCastmember).getProp("model", whichModelIndex).getPropRef("bonesplayer").getPropRef("bone",
whichBoneIndex).worldTransform
```

解説

3D エレメント。ボーン (bone) は、3D モデリングプログラムで作成されたモデルリソースの構造的なエレメントです。Director では、ボーンを作成、削除、または並べ替えできません。

ボーン（#bones）モーションは 3D モデリングプログラムで作成する必要がありますが、モデルリソースのボーン構造として動作し、Director の bonesPlayer モディファイアで管理されます。

関連事項

count (3D), bonesPlayer （モディファイア）, transform （プロパティ）, worldTransform

bonesPlayer （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.whichBonesPlayerProperty
```

解説

3D モディファイア。モデルによるモーションの使用を管理します。bonesPlayer モディファイアで管理されるモーションは、ボーンと呼ばれるモデルのセグメントをアニメーション化します。

これらを使用するモーションおよびモデルは、3D モデリングプログラムで作成し、W3D ファイルとして書き出した後、ムービーに読み込む必要があります。モーションは、Director で作成したモデルプリミティブには適用できません。

addModifier コマンドを使用してモデルに bonesPlayer モディファイアを追加すると、次の bonesPlayer モディファイアプロパティにアクセスできます。

playing (3D) は、モデルがモーションを実行しているかどうかを示します。

playlist は、プロパティリストのリニアリストです。モデル用に待機しているモーションの再生パラメータが含まれます。

currentTime (3D) は、現在再生中または一時停止されたモーションの、ミリ秒単位のローカルタイムを示します。

playRate (3D) は、play() または queue() コマンドの [スケール] パラメータを掛けた値であり、モーションの再生速度を決定します。

playlist.count (3D) は、現在再生リストで待機しているモーションの数を返します。

rootLock は、モーションの平行移動コンポーネントを使用するか無視するかを示します。

currentLoopState は、モーションを一度だけ再生するか、繰り返し再生するかを示します。

blendTime は、モディファイアの autoblend プロパティが TRUE に設定されているときに、モディファイアによって作成されたモーション間のトランジションの長さを示します。

autoblend は、前のモーションから現在再生中のモーションに、モディファイアがリニアトランジションを作成するかどうかを示します。

blendFactor は、モディファイアの autoBlend プロパティが FALSE に設定されているときに、モーション間のブレンドの度合いを示します。

bone[boneId].transform は、親ボーンを基準にしたボーンのトランスフォームを示します。モデルリソースの getBoneID プロパティを調べると、boneId の値を得られます。ボーンに設定したトランスフォームは、現在のモーションでは制御されなくなり、モーションの制御には戻せません。現在のモーションが終了すると、手動による制御はできなくなります。

bone[boneId].getWorldTransform は、ワールドを基準にしたボーンのトランスフォームを返します。

lockTranslation は、指定したプレーンからモデルを表示できるかどうかを示します。

positionReset は、モーションの終了後またはループの各反復後に、モデルを開始位置に戻すかどうかを示します。

rotationReset は、あるモーションから次のモーションへのトランジションまたは 1 つのモーションのループにおける回転エレメントを示します。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

bonesPlayer モディファイアは、次のコマンドを使用します。

pause() (3D) は、現在モデルによって実行されているモーションを停止します。

play() (3D) は、モーションの実行を開始または再開します。

playNext() (3D) は、プレイリスト内の次のモーションの再生を開始します。

queue() (3D) は、プレイリストの最後にモーションを追加します。

bonesPlayer モディファイアは、次のイベントを生成します。これらのイベントは、registerForEvent() コマンドおよび registerScript() コマンドで宣言されたハンドラによって使用されます。宣言されたハンドラの呼び出しには、イベントの種類（#animationStarted または #animationEnded）、モーションの名前、およびモーションの現在のタイムの、3 つの引数が必要です。通知イベントの詳細については、registerForEvent() を参照してください。

#animationStarted は、モーションの再生が開始されたときに送られます。モーション間でブレンドが使用されている場合は、トランジションが開始されたときにイベントが送られます。

#animationEnded は、モーションが終了したときに送られます。モーション間でブレンドが使用されている場合は、トランジションが終了したときにイベントが送られます。

関連事項

keyframePlayer（モディファイア）, addModifier, modifiers, modifier

border

使用法

```
-- Lingo syntax
memberObjRef.border

// JavaScript syntax
memberObjRef.border;
```

解説

指定したフィールドキャストメンバの境界線の幅をピクセル単位で示すフィールドキャストメンバプロパティです。

例文

次のステートメントは、フィールドキャストメンバ " タイトル " の境界線の幅を 10 ピクセルにします。

```
--Lingo syntax
member("Title").border = 10

// JavaScript syntax
member("Title").border = 10;
```

bottom

使用法

```
-- Lingo syntax
spriteObjRef.bottom

// JavaScript syntax
spriteObjRef.bottom;
```

解説

スプライトの境界矩形の底辺の垂直座標値を示すスプライトプロパティです。読み取り / 書き込み可能。

例文

次のステートメントは、(i + 1) という番号がついているスプライトの底辺の垂直座標値を変数 lowest に割り当てます。

```
-- Lingo syntax
lowest = sprite(i + 1).bottom

// JavaScript syntax
var lowest = sprite(i + 1).bottom;
```

関連事項

スプライト

bottom (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).bottom
```

解説

3D #box モデルリソースプロパティ。ボックスの -Y 軸と交差する側面が、閉じた状態であるか (TRUE)、開いた状態であるか (FALSE) を示します。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、"GiftBox" というモデルリソースの bottom プロパティを TRUE に設定します。このボックスの底面は閉じた状態になります。

```
-- Lingo syntax
nmr = member("3D World").newModelresource("GiftBox", #box)
member("3D World").modelResource("GiftBox").bottom = TRUE

// JavaScript syntax
nmr = member("3D World").newModelresource("GiftBox",symbol("box"));
member("3D World").getProp("modelresource",10).bottom = true;
```

関連事項

back, front, top (3D), left (3D), right (3D), bottomCap

bottomCap

使用法

```
member(whichCastmember).modelResource(whichModelResource).bottomCap
```

解説

3D #cylinder モデルリソースプロパティ。円柱の -Y 軸と交差する側面が、閉じた状態であるか (TRUE)、開いた状態であるか (FALSE) を示します。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" 円柱 11" というモデルリソースの bottomCap プロパティを FALSE に設定します。この円柱の底面は開いた状態になります。

```
-- Lingo syntax
nmr = member("3D World").newModelresource("Cylinder11", #cylinder);
member("3D World").modelResource("Cylinder11").bottomCap = FALSE

// JavaScript syntax
nmr = member("3D World").newModelresource("Cylinder11", symbol("cylinder"));
member("3D World").getPropRef("modelResource", 11).bottomCap = false;
```

関連事項

topCap, bottomRadius, bottom (3D)

bottomRadius

使用法

```
member(whichCastmember).modelResource(whichModelResource).bottomRadius
```

解説

3D #cylinder モデルリソースプロパティ。-Y 軸と交差する円柱の端の半径を、ワールド単位で示します。

このプロパティのデフォルト値は 25 です。

例文

次のステートメントは、" チューブ " というモデルリソースの bottomRadius プロパティを 38.5 に設定します。

```
member("3D World").modelResource("Tube").bottomRadius = 38.5
```

関連事項

topRadius, bottomCap

bottomSpacing

使用法

```
-- Lingo syntax
chunkExpression.bottomSpacing

// JavaScript syntax
chunkExpression.bottomSpacing;
```

解説

テキストキャストメンバのプロパティ。テキストキャストメンバの [チャンク式] の部分で、各段落の末尾に追加するスペースを示します。

値は整数で指定します。負の値の場合は段落間の間隔が狭くなり、正の値の場合は広くなります。

デフォルト値は 0 です。段落間の間隔はデフォルト値になります。

注意：テキストキャストメンバのすべてのプロパティと同様、このプロパティもドットシンタックスだけをサポートします。

例文

次の例では、News Items というキャストメンバ内の最初の段落に続けて行間を入れます。

```
--Lingo syntax
member("News Items").paragraph[1].bottomSpacing=20

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).bottomSpacing=20;
```

関連事項

top (3D)

boundary

使用法

```
member(whichCastmember).model(whichModel).inker.boundary
member(whichCastmember).model(whichModel).toon.boundary
```

解説

3D inker および toon モディファイアプロパティ。モデルの端に線を描画するかどうかを設定できます。

このプロパティのデフォルト設定は TRUE です。

例文

次のステートメントは、" ボックス " というモデルに適用される inker モディファイアの boundary プロパティを TRUE に設定します。モデル表面の端に線が描画されます。

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#inker)
member("shapes").model("Box").inker.boundary = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").boundary = true;
```

関連事項

lineColor, lineOffset, silhouettes, creases

boundingSphere

使用法

```
member(whichCastmember).model(whichModel).boundingSphere
member(whichCastmember).group(whichGroup).boundingSphere
member(whichCastmember).light(whichLight).boundingSphere
member(whichCastmember).camera(whichCamera).boundingSphere
```

解説

3D モデル、グループ、ライト、およびカメラのプロパティ。モデル、グループ、ライト、またはカメラと、その子を囲む球体を表します。

このプロパティの値は、球体の中心位置を表すベクトルと、球体の半径を表す浮動小数点値を含むリストです。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、メッセージウィンドウにライトを囲む球体を表示します。

```
-- Lingo syntax
put member("newAlien").light[5].boundingSphere
-- [vector(166.8667, -549.6362, 699.5773), 1111.0039]

// JavaScript syntax
put(member("newAlien").getProp("light",5).boundingSphere);
// <[vector(166.8667, -549.6362, 699.5773), 1111.0039]>
```

関連事項

debug

boxDropShadow

使用法

```
-- Lingo syntax
memberObjRef.boxDropShadow

// JavaScript syntax
memberObjRef.boxDropShadow;
```

解説

[キャストメンバ] に指定したフィールドキャストメンバのボックスの影のサイズをピクセル単位で定義するキャストメンバプロパティです。

例文

次のステートメントは " タイトル " フィールドキャストメンバの影の幅を 10 ピクセルにします。

```
--Lingo syntax
member("Title").boxDropShadow = 10

// JavaScript syntax
member("Title").boxDropShadow = 10;
```

boxType

使用法

```
-- Lingo syntax
memberObjRef.boxType

// JavaScript syntax
memberObjRef.boxType;
```

解説

指定したキャストメンバのテキストボックスの種類を定義するキャストメンバプロパティです。#adjust (合わせる)、#scroll (スクロール)、#fixed (固定)、#limit (制限) のいずれかの値を設定することができます。

例文

次のステートメントはフィールドキャストメンバ " 社説 " のボックスをスクロールフィールドにします。

```
--Lingo syntax
member("Editorial").boxType = #scroll

// JavaScript syntax
member("Editorial").boxType = symbol("scroll");
```

brightness

使用法

```
member(whichCastmember).shader(whichShader).brightness
member(whichCastmember).model(whichModel).shader.brightness
member(whichCastmember).model(whichModel).shaderList{[index]}.brightness
```

解説

3D #newsprint および #engraver シェーダプロパティ。シェーダにブレンドされる白の量を示します。

このプロパティの範囲は 1 ～ 100 で、デフォルト値は 0 です。

例文

次のステートメントは、"gbCyl2" というモデルで使用されているシェーダの明度を、その最大値の半分にします。

```
-- Lingo syntax
member("scene").model("gbCyl2").shader.brightness = 50

// JavaScript syntax
member("scene").getProp("shader",1).brightness = 50;
```

関連事項

newShader

broadcastProps

使用法

```
-- Lingo syntax
memberObjRef.broadcastProps

// JavaScript syntax
memberObjRef.broadcastProps;
```

解説

Flash メンバまたはベクトルシェイプキャストメンバに変更があったときの処理を設定するキャストメンバプロパティです。TRUE の場合、現在ステージにあるすべてのスプライトに変更を反映します。FALSE の場合、反映しません。

このプロパティが FALSE の場合、キャストメンバを変更しても、スプライトが今後作成されるときのデフォルト値が変わるだけで、ステージに作成済みのスプライトには反映されません。

デフォルトは TRUE です。このプロパティは、値を調べたり、変更することができます。

例文

次のフレームスクリプトは、"Navigation Movie" という Flash ムービーキャストメンバの broadcastProps プロパティが FALSE になっていたと想定しています。スクリプトは、直ちに Flash ムービーキャストメンバの変更をステージ上のスプライトに反映します。次に viewScale プロパティを設定し、変更をスプライト全体に反映させます。その後、スプライトに変更が反映されない設定に戻します。

```
-- Lingo syntax
on enterFrame
    member("Navigation Movie").broadcastProps = TRUE
    member("Navigation Movie").viewScale = 200
    member("Navigation Movie").broadcastProps = FALSE
end

// JavaScript syntax
function enterFrame() {
    member("Navigation Movie").broadcastProps = 1;
    member("Navigation Movie").viewScale = 200;
    member("Navigation Movie").broadcastProps = 0;
}
```

bufferSize

使用法

```
-- Lingo syntax
memberObjRef.bufferSize

// JavaScript syntax
memberObjRef.bufferSize;
```

解説

リンクされた Flash ムービーを一度に何バイトまでメモリにストリーミングできるかを設定する Flash キャストメンバプロパティです。bufferSize メンバプロパティは整数値を設定します。このプロパティは、キャストメンバの preload プロパティが FALSE に設定されている場合のみ有効です。

このプロパティは、値を調べたり、設定することができます。デフォルトは 32,768 バイトです。

例文

次の startMovie ハンドラは、Flash ムービーキャストメンバ "Flash Demo" がストリーミングするように設定し、次にその「bufferSize] プロパティを設定します。

```
-- Lingo syntax
on startMovie
    member.("Flash Demo").preload = FALSE
    member.("Flash Demo").bufferSize = 65536
end

// JavaScript syntax
function startMovie() {
    member.("Flash Demo").preload = 0;
    member.("Flash Demo").bufferSize = 65536;
}
```

関連事項

bytesStreamed, preLoadRAM, stream(), streamMode

bufferSize （ミクサ）

使用法

```
mixer.bufferSize
```

解説

サウンドオブジェクトまたはミキサプロパティ。オーディオエンジンで一度に処理されるサウンドオブジェクトまたはミクサ出力のチャンクのサイズをミリ秒単位で返します。ミクサの bufferSize は、ミクサ内のすべてのサウンドオブジェクトに適用されます。

この値を調整して、ムービーの待ち時間を調整できます。bufferSize は 10 の倍数で、デフォルト値は 100 です。

bufferSize は、ミクサが #stopped 状態のときにのみ設定できます。

例文

```
--Lingo syntax
on mouseUp me
    mixerRef.bufferSize = 40
end

// JavaScript syntax
function mouseUp(){
mixeRref.bufferSize = 40;
}
```

関連事項

ミクサ

buttonCount

使用法

```
-- Lingo syntax
dvdObjRef.buttonCount

// JavaScript syntax
dvdObjRef.buttonCount;
```

解説

現在の DVD メニューで使用できるボタンの数を返す DVD プロパティです。読み取り専用。

Macintosh では、現在サポートされていません。

関連事項

DVD

buttonsEnabled

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.buttonsEnabled

// JavaScript syntax
memberOrSpriteObjRef.buttonsEnabled;
```

解説

Flash ムービー内のボタンを有効にするかどうか設定する Flash キャストメンバプロパティです。TRUE, (デフォルト) にするとボタンはアクティブになり、FALSE にすると非アクティブになります。ボタンのアクションは、actionsEnabled プロパティが TRUE の場合にのみ起動します。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、パラメータとして whichSprite にスプライトの参照を受け取り、その buttonsEnabled プロパティを反転します。

```
-- Lingo syntax
on ToggleButtons(whichSprite)
    sprite(whichSprite).buttonsEnabled = not(sprite(whichSprite).buttonsEnabled)
end

// JavaScript syntax
function ToggleActions(whichSprite) {
    sprite(whichSprite).buttonsEnabled = !(sprite(whichSprite).buttonsEnabled);
}
```

関連事項

actionsEnabled

buttonStyle

使用法

```
-- Lingo syntax
_movie.buttonStyle

// JavaScript syntax
_movie.buttonStyle;
```

解説

マウスボタンを押している間のボタンの視覚的な反応と動作を定義するムービープロパティです。読み取り / 書き込み可能。

このプロパティは、ツールパレットのボタンツールで生成したボタンのみに適用できます。

buttonStyle プロパティには次の値を指定することができます。

- 0（リストスタイル：デフォルト）— 移動先のボタンがハイライトされます。そのボタンの上でマウスのボタンを離すと、移動先のボタンに割り当てられているスクリプトが実行されます。
- 1 (ダイアログスタイル) — 最初のボタンのみがハイライトされ、移動先のボタンはハイライトされません。クリックしたボタン以外のボタンにポインタを移動してマウスのボタンを離したときには、移動先のボタンに割り当てられているスクリプトは実行されません。

例文

次のステートメントは、buttonStyle プロパティを 1 (ダイアログスタイル) に設定します。

```
-- Lingo syntax
_movie.buttonStyle = 1

// JavaScript syntax
_movie.buttonStyle = 1;
```

次のステートメントは、buttonStyle を変数 buttonStyleValue に割り当てて buttonStyle プロパティの現行設定を記憶します。

```
-- Lingo syntax
buttonStyleValue = _movie.buttonStyle

// JavaScript syntax
var buttonStyleValue = _movie.buttonStyle;
```

関連事項

ムービー

buttonType

使用法

```
member(whichCastMember).buttonType
the buttonType of member whichCastMember
```

解説

指定したボタンキャストメンバの種類を定義するボタンキャストメンバプロパティです。#pushButton, #checkBox、#radioButton のいずれかの値を指定することができます。このプロパティは、ツールパレットのボタンツールで生成したボタンのみに適用できます。

例文

次のステートメントはボタンのキャストメンバ " 社説 " をチェックボックスにします。

```
--Lingo Dot syntax:
member("Editorial").buttonType = #checkBox

--Lingo Verbose syntax:
set the buttonType of member "Editorial" to #checkBox

// JavaScript syntax
member("Editorial").buttonType = symbol("checkBox");
```

byteArray

使用法

```
memberRef.byteArray
```

解説

バイト配列プロパティ。バイト配列のコンテンツを設定または取得します。

例文

次の例では、バイト配列をバイト配列キャストメンバに割り当てます。割り当てられたバイト配列は、ムービーの保存時に保存されます。

```
--Lingo syntax
ba_m ember=new(#byteArray)
ba = bytearray("Sample by tearray contents")
ba_member.byteArray = ba

//JavaScript syntax
ba_member=_movie.newMember(symbol("byteArray"))
ba = byteArray("Sample bytearray contents")
ba_member.byteArray = ba
```

BytesRemaining

使用法

```
byteArrayObject.bytesRemaining
```

解説

バイト配列プロパティ。バイト配列内の現在位置から配列の最後までの間で読み取ることのできるデータのバイト数を返します。

例文

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.bytesRemaining

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.bytesRemaining);
```

bytesStreamed

使用法

```
-- Lingo syntax
memberObjRef.bytesStreamed

// JavaScript syntax
memberObjRef.bytesStreamed;
```

解説

指定した [キャストメンバ] がメモリにロードされた総バイト数を示す Flash および Shockwave オーディオ (SWA) キャストメンバプロパティです。bytesStreamed プロパティは Director がムービーを再生しているときにのみ値を返します。値は整数値です。

このプロパティは、値を調べられますが、設定はできません。

例文

次のハンドラは、キャストメンバの参照をパラメータとして whichFlashMovie に受け取り、stream コマンドでメモリにロードします。Flash メンバのストリーミングが少しずつ進むごとに、bytesStreamed プロパティを使って、何バイト分進んだかをメッセージウィンドウに表示します。

```
-- Lingo syntax
on fetchMovie(whichFlashMovie)
    repeat while member(whichFlashMovie).percentStreamed < 100
        stream(member(whichFlashMovie))
        put("Number of bytes streamed:" && member(whichFlashMovie).bytesStreamed)
    end repeat
end

// JavaScript syntax
function fetchMovie(whichFlashMovie)
    var i = member(whichFlashMovie).percentStreamed;
    while(i < 100) {
        stream(member(whichFlashMovie));
        trace( "Number of bytes streamed: " + member(whichFlashMovie).bytesStreamed);
    }
}
```

関連事項

bufferSize, percentStreamed（メンバ）, stream()

bytesStreamed (3D)

使用法

```
member(whichCastMember).bytesStreamed
```

解説

3D キャストメンバプロパティ。最初のファイルを読み込むときの読み込み量、または最後に要求されたファイルをロードするときのロード量を示します。

例文

次のステートメントは、" シーン " というキャストメンバが 325,300 バイトロードされたことを示します。

```
put member("Scene").bytesStreamed
-- 325300
```

関連事項

streamSize (3D), state (3D)

camera

使用法

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

解説

3D エレメント。3D ワールドを表示するベクトル位置にあるオブジェクトです。

各スプライトにはカメラのリストがあります。リストの各カメラのビューは、インデックス の数が低いカメラのビューの上に表示されます。各カメラの rect (camera) プロパティを設定して、スプライト内に複数のビューを表示できます。

カメラは、キャストメンバのカメラパレットに保存されます。3D キャストメンバでカメラを作成および削除するには、newCamera および deleteCamera コマンドを使用します。

スプライトの camera プロパティは、スプライトのカメラリストにある最初のカメラです。sprite(スプライト).camera で参照されるカメラは、sprite(スプライト).camera(1) と同じです。3D スプライトのカメラリストを作成するには、addCamera および deleteCamera コマンドを使用します。

カメラプロパティとコマンドが記載されているリストについては、Director のヘルプパネルの「Director ユーザガイド」を参照してください。

例文

次のステートメントは、スプライト "1" のカメラを、" ピクニック " というキャストメンバの "TreeCam" というカメラに設定します。

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera("TreeCam")
```

次のステートメントは、スプライト "1" のカメラを、" ピクニック " というキャストメンバの 2 つ目のカメラに設定します。

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera[2]

// JavaScript syntax
sprite(1).camera = member("Picnic").getProp("camera", 2);
```

関連事項

bevelDepth, overlay, modelUnderLoc, spriteSpaceToWorldSpace, fog, clearAtRender

cameraPosition

使用法

```
member(whichCastMember).cameraPosition
sprite(whichSprite).cameraPosition
```

解説

3D キャストメンバおよびスプライトプロパティ。デフォルトのカメラの回転値を示します。

このプロパティのデフォルト値は vector(0, 0, 250) です。これが、新しく作成する 3D キャストメンバのデフォルトのカメラの位置になります。

例文

次のステートメントは、" ベビーランド " というキャストメンバのデフォルトのカメラの位置が vector (-117.5992, -78.9491, 129.0254) であることを示しています。

```
-- Lingo syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254)

// JavaScript syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254);
```

関連事項

cameraRotation, autoCameraPosition

cameraRotation

使用法

```
member(whichCastMember).cameraRotation
sprite(whichSprite).cameraRotation
```

解説

3D キャストメンバおよびスプライトプロパティ。デフォルトのカメラの回転値を示します。

このプロパティのデフォルト値は vector(0, 0, 0) です。これが、新しく作成する 3D キャストメンバのデフォルトのカメラの回転になります。

例文

次のステートメントは、" ベビーランド " というキャストメンバのデフォルトのカメラの回転が vector (82.6010, -38.8530, -2.4029) であることを示しています。

```
member("babyland").cameraRotation = vector(82.6010, -38.8530, -2.4029)
```

関連事項

cameraPosition, autoCameraPosition

castLib

使用法

```
-- Lingo syntax
_movie.castLib[castNameOrNum]

// JavaScript syntax
_movie.castLib[castNameOrNum];
```

解説

ムービーがアクティブであるかどうかとは無関係に、ムービーのキャストライブラリへのネームドアクセスまたはインデックスアクセスを提供するムービープロパティです。読み取り専用。

castNameOrNum のパラメータは、アクセスするムービーの名前を指定するストリング、またはアクセスするムービーの数を指定する整数とすることができます。

このプロパティは、最上位レベルの castLib() メソッドと同じような機能を提供しますが、castLib() メソッドは現在アクティブなムービーのみに適用される点が異なります。

例文

次のステートメントは " ボタン " キャストのキャスト番号をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.castLib["Buttons"].number)

// JavaScript syntax
put(_movie.castLib["Buttons"].number);
```

関連事項

castLib(), ムービー

castLibNum

使用法

```
-- Lingo syntax
memberObjRef.castLibNum

// JavaScript syntax
memberObjRef.castLibNum;
```

解説

キャストメンバが属しているキャストライブラリの数を定義するメンバプロパティです。読み取り専用。

例文

次のステートメントは、キャストメンバ " ジャズ " が属しているキャストの番号を調べます。

```
-- Lingo syntax
put(member("Jazz").castLibNum)

// JavaScript syntax
put(member("Jazz").castLibNum);
```

次のステートメントは、キャストを " 水曜日の予定 " に切り替えて、スプライト 5 に割り当てられているキャストメンバを変更します。

```
-- Lingo syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number

// JavaScript syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number;
```

関連事項

キャストライブラリ , メンバ

castMemberList

使用法

```
-- Lingo syntax
memberObjRef.castMemberList

// JavaScript syntax
memberObjRef.castMemberList;
```

解説

カーソルのフレームに使用するキャストメンバのリストを設定するカーソルキャストメンバプロパティです。[カーソルキャストメンバ] に、引用符で囲んでキャストメンバ名を指定するか、またはキャストメンバ番号を指定します。異なるキャストからキャストメンバを指定することもできます。

リストの最初のキャストメンバがカーソルの最初のフレーム、リストの 2 番目のキャストメンバがカーソルの 2 番目のフレームというようになります。

カーソルに使えないキャストメンバを指定すると、それを無視して残りのキャストメンバを使います。

このプロパティは、値を調べたり、設定することができます。

例文

次のコマンドは、"myCursor" というアニメーションカーソルキャストメンバに、4 つのキャストメンバのリストを設定します。

```
-- Lingo syntax
member("myCursor").castmemberList = [member(1), member(2), member(1, 2), member(2, 2)]

// JavaScript syntax
member("myCursor").castmemberList = list(member(1), member(2), member(1, 2), member(2, 2));
```

center

使用法

```
member(whichCastMember).center
the center of member whichCastMember
```

解説

crop キャストメンバプロパティと共に使用するキャストメンバプロパティです。

- crop プロパティ が FALSE の場合には、the center プロパティ の設定は効果がありません。
- crop が TRUE で、center も TRUE の場合には、デジタルビデオキャストメンバの中心を基準としてクロップが行われます。
- crop が TRUE で center が FALSE の場合には、デジタルビデオの右下隅を基準にクロップが行われます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、ビデオキャストメンバ " インタビュー " がスプライトの左上隅に表示されるようにします。

```
-- Lingo Dot syntax:
member("Interview").center = FALSE

-- Lingo Verbose syntax:
set the center of member "Interview" to FALSE

// JavaScript syntax
member("Interview").center = false
```

関連事項

crop, centerRegPoint, regPoint, scale （メンバ）

centerRegPoint

使用法

```
-- Lingo syntax
memberObjRef.centerRegPoint

// JavaScript syntax
memberObjRef.centerRegPoint;
```

解説

[キャストメンバ] のセンターポイントを、スプライトのリサイズ時に自動調整するかどうかを設定する Flash、ベクトルシェイプ、ビットマップキャストメンバプロパティです。TRUE, (デフォルト) にすると中心に自動調整されます。また、現行ポイント値に調整する場合は defaultRect または regPoint プロパティを設定し、centerRegPoint プロパティは FALSE にします。

このプロパティは、値を調べたり、設定することができます。

例文

次のスクリプトは、Flash ムービーの centerRegPoint プロパティが TRUE かどうか調べます。TRUE の場合、regPoint プロパティを使ってスプライトのセンターポイントを左上隅に再配置します。centerRegPoint プロパティを調べているのは、regPoint プロパティを使って前に設定されたセンターポイントを再配置しないようにするためです。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.centerRegPoint = TRUE then
        sprite(spriteNum).member.regPoint = point(0,0)
    end if
end

// JavaScript syntax
function beginSprite() {
    var ctrRg = sprite(this.spriteNum).member.centerRegPoint;
    if (ctrRg == 1) {
        sprite(this.spriteNum).member.regPoint = point(0,0);
    }
}
```

関連事項

regPoint

centerStage

使用法

```
-- Lingo syntax
_movie.centerStage

// JavaScript syntax
_movie.centerStage;
```

解説

ムービーがロードされるときにステージをモニタの中央に配置するかどうかを定義するムービープロパティです。TRUE,（デフォルト）の場合、ステージは中央に配置されます。FALSE の場合、中央には配置されません。読み取り / 書き込み可能。

このプロパティを含むステートメントは、制御するムービーより前にあるムービー内に配置します。

このプロパティは、ムービーをプロジェクタで再生する前にステージの配置を調べるときに有効です。

注意：プロジェクタで再生中のビヘイビアーは Windows と Macintosh で異なります。プロジェクタの生成中に設定によって、このプロパティは変更されます。

例文

次のステートメントは、ステージが中央に配置されていないときに、指定したフレームに移ります。

```
-- Lingo syntax
if (_movie.centerStage = FALSE) then
     _movie.go("Off Center")
end if

// JavaScript syntax
if (_movie.centerStage == false) {
    _movie.go("Off Center");
}
```

次のステートメントは centerStage プロパティを現在とは逆の値にします。

```
-- Lingo syntax
_movie.centerStage = not(_movie.centerStage)

// JavaScript syntax
_movie.centerStage = !(_movie.centerStage)
```

関連事項

fixStageSize, ムービー

changeArea

使用法

```
member(whichCastMember).changeArea
the changeArea of member whichCastMember
```

解説

トランジションをステージ上で変化する領域のみに適用するかどうかを定義するトランジションキャストメンバプロパティです。TRUE の場合、トランジションは変化する領域のみに適用されます。FALSE の場合、ステージ全体に適用されます。その効果は [フレームプロパティトランジション] ダイアログボックスにある [必要範囲のみ] オプションを選択したときと同じです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、トランジションのキャストメンバ " 波 " がステージ上で変化する領域のみに適用されるようにします。

```
-- Lingo Dot syntax:
member("Wave").changeArea = TRUE

-- Lingo Verbose syntax:
set the changeArea of member "Wave" to TRUE

// JavaScript syntax
member("Wave").changeArea = true;
```

channel

使用法

```
mixer.channel (Read-only)
```

解説

オーディオミクサプロパティ。ミクサに関連付けられたチャネルを返します。このプロパティはミクサの再生中のみ有効です。

Lingo または JavaScript で、_movie.channel[1] を使用して、チャネル 1 にアクセスできます。

例文

```
--Lingo syntax
on mouseUp me
    put mixer1.channel --Displays the channel that the mixer is associated with.
end

// JavaScript syntax
function mouseup()
{
put (mixer1.channel); //Displays the channel that the mixer is associated with.
}
```

関連事項

ミクサ

channelCount（ミクサ）

使用法

```
mixer.channelCount (Read-write)
```

解説

ミクサプロパティ。ミクサ内のチャネル数を返すか設定します。このプロパティは読み書きが可能です。

例文

```
--Lingo syntax
on mouseUp me
mixerRef.channelcount = 6 -- Sets the channelCount of the mixer to 6.
end

// JavaScript syntax
function mouseUp(){
mixerRef.channelcount = 6; // Sets the channelCount of the mixer to 6.
}
```

関連事項

ミクサ

channelCount （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.channelCount

// JavaScript syntax
soundChannelObjRef.channelCount;
```

解説

あるサウンドチャネルで、現在再生中または停止中のサウンドでのチャネルの個数を判定するサウンドチャネルプロパティです。読み取り専用。

このプロパティは、あるサウンドがモノラルかステレオかを調べるときに役立ちます。

例文

次のステートメントは、サウンドのキャストメンバ " ジャズ " 内のチャネル数を調べます。

```
-- Lingo syntax
put(member("Jazz").channelCount)

// JavaScript syntax
put(member("Jazz").channelCount);
```

次のステートメントでは、サウンドチャネル 2 で現在再生中のサウンドメンバのチャネルの個数をメッセージウィンドウに出力します。

```
-- Lingo syntax
put(sound(2).channelCount)

// JavaScript syntax
put(sound(2).channelCount);
```

関連事項

サウンドチャネル

channelCount （サウンドオブジェクト）

使用法

```
soundObj.channelCount (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトのチャネル数を返します。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.channelCount -- Displays the number of channels in the sound
-- object associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.channelCount) ; // Displays the number of channels in the sound object
// associated with soundobjectRef.
}
```

chapter

使用法

```
-- Lingo syntax
dvdObjRef.chapter

// JavaScript syntax
dvdObjRef.chapter;
```

解説

現行チャプタの数値を返す DVD プロパティです。読み取り / 書き込み可能。

例文

次のステートメントは、現行チャプタを返します。

```
-- Lingo syntax
trace (member(1).chapter)-- 1

// JavaScript syntax
trace (member(1).chapter);// 1
```

関連事項

DVD

chapterCount

使用法

```
-- Lingo syntax
dvdObjRef.chapterCount

// JavaScript syntax
dvdObjRef.chapterCount;
```

解説

タイトルで使用できるチャプタの数を返す DVD プロパティです。読み取り専用。

例文

次のステートメントは、現行タイトルのチャプタ数を返します。

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

関連事項

chapterCount(), DVD

characterSet

使用法

```
-- Lingo syntax
memberObjRef.characterSet

// JavaScript syntax
memberObjRef.characterSet;
```

解説

キャストメンバが作成されるときに読み込み用に含まれる文字を含むストリングを返すフォントキャストメンバです。元のフォントの文字がすべて含まれている場合、結果は、空白のストリングになります。

例文

次のステートメントは、キャストメンバ 11 が作成されたときに含まれた文字を表示します。読み込み時に含まれた文字は、数字とアルファベットです。

```
-- Lingo syntax
put(member(11).characterSet)

// JavaScript syntax
put(member(11).characterSet);
```

関連事項

recordFont, bitmapSizes, originalFont

charSpacing

使用法

```
-- Lingo syntax
chunkExpression.charSpacing

// JavaScript syntax
chunkExpression.charSpacing;
```

解説

テキストキャストメンバの [チャンク式] で示される範囲の各文字間に挿入する間隔を指定するテキストキャストメンバプロパティです。

負の値の場合は文字間の間隔が狭くなり、正の値の場合は広くなります。

デフォルトは 0 で、文字間隔はデフォルトのままになります。

例文

次のハンドラは、値 2 を割り当てることによって、テキストキャストメンバ myCaption の現在の 3 ～ 5 番目の単語の文字スペースを増やします。

```
--Lingo syntax
on myCharSpacer
    mySpaceValue = member("myCaption").word[3..5].charSpacing
    member("myCaption").word[3..5].charSpacing = (mySpaceValue + 2)
end

// JavaScript syntax
function myCharSpacer() {
    var i = 3;
    while (i < 6) {
        var mySpaceValue = member("myCaption").getPropRef("word", i).charSpacing;
        member("myCaption").getPropRef("word", i).charSpacing = (mySpaceValue + 2);
    }
}
```

checkMark

使用法

```
the checkMark of menuItem whichItem of menu whichMenu
```

解説

指定したカスタムメニュー項目に、チェックマークを付けて表示するかどうかを定義するメニュー項目プロパティです。TRUE の場合、カスタムメニュー項目の横にチェックマークが表示されます。FALSE, (デフォルト) の場合、チェックマークは表示されません。

[項目] には、メニュー項目名かメニュー項目番号のいずれかを指定し、[メニュー] には、メニュー名かメニュー番号のいずれかを指定します。

このプロパティは、値を調べたり、設定することができます。

注意：メニューは、Shockwave Player では使用できません。

例文

次のハンドラは、theMenu パラメータによって指定されたカスタムメニュー内でチェックされている項目のチェックマークを外します。例えば、次のサンプル unCheck "Format" では、Format メニュー内のすべての項目からチェックマークが消えます。

```
-- Lingo syntax
on unCheck theMenu
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the checkMark of menuItem i of menu theMenu to FALSE
    end repeat
end unCheck

// JavaScript syntax
function unCheck (theMenu) {
    var n = _menuBar.menu[theMenu].item.count; //the number of menuItems of menu theMenu
    for( i = 1 ; i <= n ; i++ )
        _menuBar.menu[theMenu].item[i].checkMark = false;
}
```

関連事項

installMenu, enabled, name （メニュー項目プロパティ）, number （メニュー項目）, script, menu

child (3D)

使用法

```
member(whichCastmember).model(whichParentNode).child(whichChildNodeName)
member(whichCastmember).model(whichParentNode).child[index]
```

解説

3D モデル、グループ、ライトおよびカメラのプロパティ。子ノードの名前という子ノード、または親ノードの子リストの指定したインデックス位置にある子ノードを返します。ノードは、モデル、グループ、カメラ、またはライトです。

ノードのトランスフォームは親を基準とします。親の位置を変更すると、その子も一緒に移動し、親との相対的な位置が維持されます。親の回転およびスケールプロパティの変更は、子にも同様に反映されます。

親の子リストに追加するには、親ノードの addChild メソッドを追加するか、子ノードの parent プロパティを設定します。子には親を 1 つしか設定できませんが、親にはいくつでも子を設定できます。また、子には自身の子も設定できます。

例文

次のステートメントは、" 車 " というモデルの 2 番目の子が " タイヤ " というモデルであることを示しています。

```
-- Lingo syntax
put member("3D").model("Car").child[2]
-- model("Tire")

// JavaScript syntax
put ( member("3D").getProp("model", 1).child[2] );
// model("Tire")
```

関連事項

addChild, parent

child (XML)

使用法

```
XMLnode.child[ childNumber ]
```

解説

XML プロパティ。構文解析済み XML ドキュメントのネストされたタグ構造の指定された子ノードに戻ります。子の番号は、1 から始まります。

例文

次の XML から始まる例 :

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

次の Lingo は、以上の XML の最初の子ノードの名前を返します。

```
put gParserObject.child[1].name
-- "e1"
```

chunkSize

使用法

```
member(whichCastMember).chunkSize
the chunkSize of member whichCastMember
```

解説

トランジションのチャンクサイズを定義するトランジションキャストメンバプロパティです。値は、トランジションで使う各チャンク内のピクセル数になり、1 ～ 128 までの間で指定します。効果はフレームプロパティトランジションダイアログボックス内の滑らかさスライダを調節するのと同じです。チャンクサイズが小さいほど滑らかにトランジションが表示されます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、トランジションキャストメンバ "霧" のチャンクサイズを 4 ピクセルに設定します。

ドットシンタックスの場合 :

```
member("Fog").chunkSize = 4
```

バーボーズシンタックスの場合 :

```
set the chunkSize of member "Fog" to 4
```

clearAtRender

使用法

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearAtRender
sprite(whichSprite).camera{(index)}.colorBuffer.clearAtRender
```

解説

3D プロパティ。各フレームの後にカラーバッファを消去するかどうかを示します。FALSE を設定するとバッファは消去されず、軌跡をオンにしたのと同じような結果になります。このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、カメラからのビューの過去の画像を消去しないように指示します。モデルの動いた跡がステージに残ります。

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearAtRender = 0

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearAtRender = 0;
```

関連事項

clearValue

clearValue

使用法

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearValue
sprite(whichSprite).camera{(index)}.colorBuffer.clearValue
```

解説

3D プロパティ。colorBuffer.clearAtRender が TRUE に設定されている場合に、カラーバッファの消去に使用するカラーを指定します。このプロパティのデフォルト設定は rgb(0, 0, 0) です。

例文

次のステートメントは、カメラの clearValue プロパティを rgb(255, 0, 0) に設定します。モデルのない 3D ワールドのスペースは赤で表示されます。

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearValue= rgb(255, 0, 0)

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearValue= color(255, 0, 0);
```

関連事項

clearAtRender

clickLoc

使用法

```
-- Lingo syntax
_mouse.clickLoc

// JavaScript syntax
_mouse.clickLoc;
```

解説

画面上でマウスが最後にクリックされた位置を返すマウスプロパティです。位置は point で示されます。読み取り専用。

例文

次の on mouseDown ハンドラは、最後にマウスがクリックされた位置を表示します。

```
-- Lingo syntax
on mouseDown
    put(_mouse.clickLoc)
end mouseDown

// JavaScript syntax
function mouseDown() {
    put(_mouse.clickLoc);
}
```

クリックされた位置がステージの左端から 50 ピクセル、上端から 100 ピクセルだった場合、メッセージウィンドウには次のように表示されます。

```
point(50, 100)
```

関連事項

clickOn, マウス

clickMode

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.clickMode

// JavaScript syntax
memberOrSpriteObjRef.clickMode;
```

解説

Flash キャストメンバおよびスプライトプロパティ。Flash ムービースプライトが、マウスクリックイベント (mouseUp および mouseDown) やロールオーバー (mouseEnter、mouseWithin、および mouseLeave) をいつ検出するかを指定します。clickMode プロパティは次の値を取り込むことができます。

- #boundingBox - スプライトの境界矩形内でマウスクリックイベントを検出します。また、スプライトの境界でロールオーバーを検出します。
- #opaque（デフォルト）- スプライトの不透明な領域にマウスポインタがあるときにのみマウスクリックイベントを検出します。また、インク効果が Background Transparent に設定されている場合は、スプライトの透明な部分では、ロールオーバーを検出しません。Background Transparent 以外の場合、この設定は #boundingBox と同じ効果があります。
- #object - マウスポインタがスプライトの塗りつぶされた（背景が見えていない）領域にある場合にのみマウスクリックイベントを検出します。また、塗りつぶされた領域の境界でロールオーバーを検出します。これはスプライトのインク効果にかかわらず動作します。

このプロパティは、値を調べたり、設定することができます。

例文

次のスクリプトは、インク効果が Background Transparent のスプライトが、常にステージの最前面に描画する設定かどうかを調べます。常に最前面に描画するのではない場合、clickMode を #opaque にします。常に最前面に描画する場合、Flash ムービースプライトではインク効果が無視されるため、clickMode を #boundingBox にします。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).directToStage = FALSE then
        sprite(spriteNum).clickMode = #opaque
    else
        sprite(spriteNum).clickMode = #boundingBox
    end if
end

// JavaScript syntax
function beginSprite(me){
    var dts = sprite(this.spriteNum).directToStage;
    if (dts == 0) {
        sprite(this.spriteNum).clickMode = symbol("opaque");
    } else {
        sprite(this.spriteNum).clickMode = symbol("boundingBox");
    }
}
```

clickOn

使用法

```
-- Lingo syntax
_mouse.clickOn

// JavaScript syntax
_mouse.clickOn;
```

解説

ユーザが最後にクリックしたアクティブスプライトを返すマウスプロパティです。読み取り専用。

アクティブスプライトとは、スプライトやキャストメンバにスクリプトが割り当てられているスプライトのことを意味します。

ユーザがステージをクリックすると、clickOn は 0 になります。ユーザがスクリプトのないスプライトをクリックしたかどうかを判定する場合には、マウスイベントスクリプトを割り当てて clickOn 関数で認識できるようにします。例えば次のようになります。

```
-- Lingo syntax
on mouseUp me
    ...
end
```

ボタン、チェックボックス、ラジオボタンは、スクリプトが割り当てられている場合でも、clickOn によって検出されます。

clickOn プロパティは、繰り返しループ内で調べることができますが、ハンドラの処理中には clickOn と clickLoc のいずれも値を更新しません。返される値は、ハンドラが処理される前の値です。

例文

次のステートメントは、スプライト 7 が最後にクリックされたアクティブスプライトかどうかを調べます。

```
-- Lingo syntax
if (_mouse.clickOn = 7) then
    _player.alert("Sorry, try again.")
end if

// JavaScript syntax
if (_mouse.clickOn == 7) {
    _player.alert("Sorry, try again.");
}
```

次のステートメントは、最後にクリックされたアクティブスプライトの foreColor プロパティをランダムカラーに設定します。

```
-- Lingo syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1)

// JavaScript syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1);
```

関連事項

clickLoc, マウス

closed

使用法

```
-- Lingo syntax
memberObjRef.closed

// JavaScript syntax
memberObjRef.closed;
```

解説

折れ線パスの最後の点を閉じるかどうかを設定する、ベクトルシェイプキャストメンバプロパティです。

ベクトルシェイプの中を塗りつぶすには、パスが閉じている必要があります。

次の値のいずれかです。

- TRUE - 最後の点を閉じます。
- FALSE - 最後の点を閉じないままにします。

closedCaptions

使用法

```
-- Lingo syntax
dvdObjRef.closedCaptions

// JavaScript syntax
dvdObjRef.closedCaptions;
```

解説

DVD プロパティ。クローズドキャプションが有効（TRUE）かどうか、または有効にできないか（FALSE）を判定します。Macintosh では、現在サポートされていません。読み取り / 書き込み可能。

例文

次のステートメントは、closedCaptions を TRUE に設定し、有効にできない場合はアラートを表示します。

```
-- Lingo syntax
member(3).closedCaptions = TRUE
if (member(3).closedCaptions = FALSE) then
    _player.alert("Closed captions cannot be enabled.")
end if

// JavaScript syntax
member(3).closedCaptions = true
if (member(3).closedCaptions == false) {
    _player.alert("Closed captions cannot be enabled.");
}
```

Macintosh では、現在サポートされていません。

関連事項

DVD

collision (modifier)

使用法

```
member(whichCastmember).model(whichModel).collision.collisionModifierProperty
```

解説

3D モディファイア。衝突の検出と解決を管理します。addModifier コマンドを使用してモデルに collision モディファイアを追加すると、次の collision モディファイアプロパティにアクセスできます。

enabled (collision) は、モデルとの衝突を検出するかどうかを示します。

resolve は、モデルとの衝突を解決するかどうかを示します。

immovable は、モデルをフレーム間で移動できるかどうかを示します。

mode (collision) は、衝突の検出に使用されるジオメトリを示します。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

衝突モディファイアは次のイベントを生成します。衝突イベントの使い方の詳細については、registerForEvent() のエントリを参照してください。

#collideAny イベントは、collision モディファイアが割り当てられているモデル間で衝突が発生すると生成されます。

#collideWith イベントは、collision モディファイアが割り当てられている特定のモデルとの衝突が発生すると生成されます。

#collideAny および #collideWith イベントでは、collisionData オブジェクトがパラメータとして送られます。プロパティの詳細については、collisionData エントリを参照してください。

関連事項

addModifier, removeModifier, modifiers

collisionData

使用法

```
on myHandlerName me, collisionData
```

解説

3D データオブジェクト。#collideWith イベントおよび #collideAny イベントで、registerForEvent、registerScript、および setCollisionCallback の各コマンドで指定したハンドラにパラメータとして送られます。collisionData オブジェクトには次のプロパティがあります。

modelA は、衝突を起こしているモデルの 1 つです。

modelB は、衝突を起こしているもう一方のモデルです。

pointOfContact は、衝突のワールド位置です。

collisionNormal は、衝突の方向です。

例文

次の例は、3 つの部分で構成されています。最初の部分はコードの 1 行目です。#collideAny イベントに #putDetails ハンドラを登録します。2 つ目の部分は #putDetails ハンドラです。キャストメンバ "MyScene" の 2 つのモデルが衝突している場合、#putDetails ハンドラが呼び出され、collisionData パラメータが送られます。このハンドラは、メッセージウィンドウに collisionData オブジェクトの 4 つのプロパティを表示します。この例の 3 番目の部分では、メッセージウィンドウの結果を示します。最初の 2 行は、"GreenBall" というモデルがモデル A で、"YellowBall" というモデルがモデル B であることを示します。3 行目は、2 つのモデルの接触位置を示します。最後の行は、衝突の方向を示します。

```
-- Lingo syntax

member("MyScene").registerForEvent(#collideAny, #putDetails, 0)

on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
    put collisionData.pointOfContact
    put collisionData.collisionNormal
end
-- model("GreenBall")
-- model("YellowBall")
-- vector( 24.800, 0.000, 0.000 )
-- vector( -1.000, 0.000, 0.000 )

// JavaScript syntax

member("MyScene").registerForEvent(symbol("collideAny"),symbol("putDetails"), 0);

function putDetails (me, collisionData)
{
    put (collisionData.modelA);
    put (collisionData.modelB);
    put (collisionData.pointOfContact);
    put (collisionData.collisionNormal);
}
// model("GreenBall")
// model("YellowBall")
// vector( 24.800, 0.000, 0.000 )
// vector( -1.000, 0.000, 0.000 )
```

関連事項

collisionData のプロパティ：modelA, modelB, pointOfContact, collisionNormal

collisionData メソッド：resolveA, resolveB, collision (modifier)

collisionNormal

使用法

```
collisionData.collisionNormal
```

解説

3D collisionData プロパティ。衝突の方向を示すベクトルです。

collisionData オブジェクトは、#collideWith および #collideAny イベントのパラメータとして、registerForEvent, registerScript, および setCollisionCallback の各コマンドによって指定されたハンドラに送られます。

衝突モディファイアが追加されているモデル間で衝突が発生すると、#collideWith イベントと #collideAny イベントが送信されます。このとき、モデルのモディファイアの resolve プロパティは、TRUE に設定されている必要があります。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、2 つの部分で構成されています。まずコードの 1 行目で、#collideAny イベントに #explode ハンドラを登録します。2 つ目は、#explode ハンドラです。キャストメンバ "MyScene" の 2 つのモデルが衝突している場合、#explode ハンドラが呼び出され、collisionData 引数が送られます。#explode ハンドラの最初の 10 行は、モデルリソース SparkSource を作成し、そのプロパティを設定します。このモデルリソースは、パーティクルの 1 回のバーストです。10 行目は、バーストの方向を collisionNormal に設定します。これは衝突の方向です。ハンドラの 11 行目は、モデルリソース SparkSource を使って "SparksModel" というモデルを作成します。ハンドラの最後の行は、"SparksModel" の位置を、衝突が発生した位置に設定します。全体の効果としては、接触した位置から衝突の方向に火花のスパークを飛ばします。

```
-- Lingo syntax
member("MyScene").registerForEvent(#collideAny, #explode, 0)
on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource", #particle)
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.angle = 45
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nmr.emitter.direction = vector(0,0,-1)
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal)
end

// JavaScript syntax
member("MyScene").registerForEvent(symbol("collideAny"), symbol("explode"), 0);
function explode (me, collisionData) {
    nmr = member("MyScene").newModelResource("SparkSource", symbol("particle"));
    nmr.getPropRef("emitter").mode = symbol("burst");
    nmr.getPropRef("emitter").loop = 0;
    nmr.getPropRef("emitter").minSpeed = 30;
    nmr.getPropRef("emitter").maxSpeed = 50;
    nmr.getPropRef("emitter").angle = 45;
    nmr.getPropRef("colorRange").start = color(0, 0, 255);
    nmr.getPropRef("colorRange").end = color(255, 0, 0);
    nmr.lifetime = 5000;
    nmr.getPropRef("emitter").direction = vector(0,0,-1);
    nm = member("MyScene").newModel("SparksModel", nmr);
    nm.transform.position = collisionData.pointOfContact;
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal);
}
```

関連事項

pointOfContact, modelA, modelB, resolveA, resolveB, collision (modifier)

color()

使用法

```
color(#rgb, redValue, greenValue, blueValue)
color(#paletteIndex, paletteIndexNumber)
rgb(rgbHexString)
rgb(redValue, greenValue, blueValue)
paletteIndex(paletteIndexNumber)
```

解説

オブジェクトの色を RGB 値または 8 ビットパレットのインデックス値で設定する関数とデータ型です。これらの値は、color メンバプロパティ、color スプライトプロパティ、bgColor メンバプロパティ、bgColor スプライトプロパティ、bgColor ステージプロパティに使われているものと同じです。

color 関数を使って、24 ビットまたは 8 ビットの色の値を扱い、キャストメンバ、スプライト、ステージに適用することができます。

RGB 値の各色要素の範囲は 0 ～ 255 で、それ以外の値を指定しても切り捨てられます。paletteIndex の範囲は、現行パレットのインデックスを示す 0 ～ 255 の整数で、それ以外の値は切り捨てられます。

例文

次のステートメントでは、パレットのインデックス番号を数値として計算しています。

```
palColorObj = paletteIndex(20)
put palColorObj
-- paletteIndex(20)
put palColorObj / 2
-- paletteIndex(10)
```

次のステートメントは、RGB 値をパレットインデックスに変換します。

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj
-- rgb(155, 0, 75)
newColorObj.colorType = #paletteIndex
put newColorObj
-- paletteIndex(106)
```

次のステートメントはパレットインデックスの色（32 番）を 16 進ストリングで表示します。

```
someColorObj = color(#paletteIndex, 32)
put someColorObj.hexString()
-- "#FF0099"
```

次のステートメントは、RGB の各色要素を独立に設定し、paletteIndex 値を求めます。

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj.green
-- 0
put newColorObj.paletteIndex
-- 106
newColorObj.green = 100
put newColorObj.paletteIndex
-- 94
put newColorObj
-- rgb(155, 100, 75)
newColorObj.paletteIndex = 45
put newColorObj
-- paletteIndex(45)
```

次のステートメントは、テキストメンバ myQuotes の 4 ～ 7 番目の文字の色を変更します。

```
member("myQuotes").char[4..7].color = rgb(200, 150, 75)
```

この Lingo は、メッセージウィンドウのスプライト 6 の色を表示し、新しい RGB 値にスプライト 6 の色を設定します。

```
put sprite(6).color
-- rgb( 255, 204, 102 )
sprite(6).color = rgb(122, 98, 210)
```

注意：RGB で設定したカラーオブジェクトに paletteIndex 値を設定すると、colorType が変わって、paletteIndex で表示するようになります。逆に、paletteIndex で設定したカラーオブジェクトに RGB 値で設定すると colorType が変わって RGB で表示するようになります。

関連事項

bgColor （ウィンドウ）

color (fog)

使用法

```
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
```

解説

3D プロパティ。カメラの fog.enabled プロパティが TRUE に設定されている場合に、カメラによってシーンに導入されるカラーを示します。

このプロパティのデフォルト設定は rgb(0, 0, 0) です。

例文

次のステートメントは、"BayView" というカメラのフォグのカラーを rgb(255, 0, 0) に設定します。カメラの fog.enabled プロパティが TRUE に設定されている場合、フォグ中のモデルは赤の色相を帯びます。

```
member("MyYard").camera("BayView").fog.color = rgb(255, 0, 0)
```

関連事項

fog

color (light)

使用法

```
member(whichCastmember).light(whichLight).color
```

解説

3D ライトプロパティ。ライトの RGB 値を示します。

このプロパティのデフォルト値は rgb(191,191,191) です。

例文

次のステートメントは、"RoomLight" というライトのカラーを rgb(255, 0, 255) に設定します。

```
member("Room").light("RoomLight").color = rgb(255,0,255)
```

関連事項

fog

colorBufferDepth

使用法

```
getRendererServices().colorBufferDepth
```

解説

3D rendererServices プロパティ。ユーザのシステムの、ハードウェア出力バッファのカラー精度を示します。値はハードウェアの設定に応じて 16 または 32 のいずれかです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、ユーザのビデオカードの colorBufferDepth 値が 32 であることを示します。

```
-- Lingo syntax
put getRendererServices().colorBufferDepth
-- 32

// JavaScript syntax
put (getRendererServices().colorBufferDepth);
// 32
```

関連事項

getRendererServices(), getHardwareInfo(), depthBufferDepth

colorDepth

使用法

```
-- Lingo syntax
_system.colorDepth

// JavaScript syntax
_system.colorDepth;
```

解説

コンピュータモニタの色深度を定義するシステムプロパティです。読み取り / 書き込み可能。

- Windows では、このプロパティを使って、モニタの 色深度の確認と設定ができます。ビデオカードとドライバの組み合わせによっては、colorDepth プロパティを設定することはできません。色深度の設定を変えたときには、必ず確認します。
- Macintosh では、このプロパティを使用して他のモニタの色深度を調べ、必要に応じて変更することができます。

このプロパティは次のいずれかの値になります。

1	白黒
2	4 色
4	16 色
8	256 色
16	32,768 または 65,536 色
32	16,777,216 色

モニタの色深度よりも高い色深度を割り当てようとしても、色深度は変化しません。

複数のモニタを接続している環境では、colorDepth プロパティはステージが置かれているモニタを対象とします。ステージが複数のモニタにわたっている場合、colorDepth プロパティは、その中で最も色深度が高いモニタの値を示します。また、colorDepth プロパティは、すべてのモニタを指定された色深度に設定しようとします。

例文

次のステートメントは、ムービー " フルカラー " を、モニタの色深度が 256 色に設定されているかどうかに基づいて開くようにします。

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    window("Full color").open()
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    window("Full color").open()
}
```

次のハンドラは、色深度を変更しようとします。変更できないときは、アラートが表示されます。

```
-- Lingo syntax
on tryToSetColorDepth(desiredDepth)
    _system.colorDepth = desiredDepth
    if (_system.colorDepth = desiredDepth) then
        return true
    else
        _player.alert("Please change your system to" && desiredDepth && "color depth and reboot.")
        return false
    end if
end

// JavaScript syntax
function tryToSetColorDepth(desiredDepth) {
    _system.colorDepth = desiredDepth;
    if (_system.colorDepth == desiredDepth) {
        return true;
    }
    else {
        _player.alert("Please change your system to " + desiredDepth + " color depth and reboot.");
        return false;
    }
}
```

関連事項

システム

colorList

使用法

```
member(whichCastmember).modelResource(whichModelResource).colorList
member(whichCastmember).modelResource(whichModelResource).colorList[index]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList[index]
```

解説

3D プロパティ。このプロパティを使用すると、メッシュで使用されるカラーを取得または設定できます。このコマンドは、#mesh というタイプのモデルリソースでのみアクセスできます。1 つのカラーをメッシュの複数の頂点 (面) で共有できます。代わりに、メッシュのフェイスのテクスチャ座標を指定し、このモデルリソースを使用するモデルにシェーダを適用できます。

このコマンドは、newMesh 呼び出しで指定したのと同じ番号の Lingo カラー値リストに設定する必要があります。

例文

次のステートメントは、モデルリソース " メッシュ 2" の colorList にある 3 番目のカラーが rgb(255, 0, 0) であることを示します。

```
-- Lingo syntax
put member("shapes").modelResource("mesh2").colorlist[3]
-- rgb(255,0,0)

// JavaScript syntax
put ( member("shapes").getProp("modelResource" , 1).colorlist[3] );
// color(255,0,0)
```

関連事項

face[ ], colors

colorRange

使用法

```
member(whichCastmember).modelResource(whichModelResource).colorRange.start
member(whichCastmember).modelResource(whichModelResource).colorRange.end
```

解説

3D #particle モデルリソースのプロパティ。パーティクルシステムの、パーティクルの開始カラーと終了カラーを示します。

start プロパティは、パーティクルが作成されるときのカラーです。end プロパティは、寿命が切れるときのパーティクルのカラーです。各パーティクルのカラーは、start から end の値まで、その寿命を通じて徐々に変化します。

start プロパティおよび end プロパティのデフォルト値は rgb(255, 255, 255) です。

例文

次のステートメントは、"ThermoSystem" というモデルリソースの colorRange プロパティを設定します。1 行目は開始点の値を rgb(255, 0, 0) に設定し、2 行目は終了点の値を rgb(0, 0, 255) に設定します。このステートメントの効果で、"ThermoSystem" のパーティクルは最初に表示されたときは赤で、寿命を通じて徐々に青に変化します。

```
member(8,2).modelResource("ThermoSystem").colorRange.start =rgb(255,0,0)
member(8,2).modelResource("ThermoSystem").colorRange.end = rgb(0,0,255)
```

関連事項

emitter, blendRange, sizeRange

colors

使用法

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].colors
```

解説

3D フェイスプロパティ。3 つの整数からなるリニアリストです。フェイスの 3 つの頂点で使用する、モデルリソースのカラーリストのインデックス位置を示します。カラーリストは、rgb の値からなるリニアリストです。

colors プロパティは、#mesh というタイプのモデルリソースのみで使用されます。

このプロパティを設定した後、変更を有効にするには、モデルリソースの build() コマンドを使用する必要があります。

例文

次の例は、#mesh というタイプのモデルリソースを作成し、そのプロパティを指定します。そして、モデルリソースを使用して新しいモデルを作成します。

1 行目は、newMesh() コマンドを使用して " 三角形 " という #mesh モデルリソースを作成します。これは 1 つの面、3 つの頂点、および最大で 3 つのカラーを持ちます。法線の数とテクスチャ座標の数は設定されていません。

2 行目は、vertexList プロパティを、3 つのベクトルのリストに設定します。

3 行目は、vertexList プロパティのベクトルを " 三角形 " の最初の面の頂点に割り当てます。

4 行目は、カラーリストに 3 つの rgb 値を設定します。

5 行目は、" 三角形 " の最初の面にカラーを割り当てます。カラーリストの 3 番目のカラーが " 三角形 " の最初の頂点に適用され、2 番目のカラーが 2 番目の頂点に割り当てられます。1 番目のカラーは 3 番目の頂点に割り当てられます。カラーは、" 三角形 " の最初の面全体にグラデーションで広がります。

6 行目は、generateNormals() コマンドで " 三角形 " の法線を作成します。

7 行目は、build() コマンドを使ってメッシュを構築します。

8 行目は、新しいメッシュを使用する "TriModel" という新しいモデルを作成します。

```
nm = member("Shapes").newMesh("Triangle",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)
```

関連事項

face, vertices, vertices, flat

colorSteps

使用法

```
member(whichCastmember).model(whichModel).toon.colorSteps
member(whichCastmember).model(whichModel).shader.colorSteps
member(whichCastmember).shader(whichShader).colorSteps
```

解説

3D toon モディファイアおよび painter シェーダプロパティ。toon モディファイアまたは painter シェーダで使用可能なカラーの最大数です。このプロパティの値は、2、4、8、または 16 です。colorSteps に他の値を設定した場合は、これらのいずれかの値に丸められます。

デフォルト値は 2 です。

例文

次のステートメントは、" ティーポット " というモデルの toon モディファイアで使用可能なカラーの数を 8 に制限します。ティーポットは、最大 8 色でレンダリングされます。

```
-- Lingo syntax
member("shapes").model("Teapot").toon.colorSteps = 8

// JavaScript syntax
member("shapes").getProp("model", 1).getPropRef("toon").colorSteps = 8;
```

関連事項

highlightPercentage, shadowPercentage

commandDown

使用法

```
-- Lingo syntax
_key.commandDown

// JavaScript syntax
_key.commandDown;
```

解説

Ctrl キー（Windows）または Command キー（Macintosh）が押されたかどうかを調べるキープロパティです。読み取り専用。

このプロパティは、Ctrl キーまたは Command キーが押された場合は TRUE を返し、そうでない場合は FALSE を返します。

commandDown を key プロパティと組み合わせて使用する、Ctrl キーまたは Command キーとその他のキーが同時に押されたかどうかを調べることができます。この機能を利用して、ユーザが Ctrl キーや Command キーと特定のキーを同時に押したときに実行するハンドラを作成することもできます。

Lingo または JavaScript シンタックスのカスタムメニューがインストールされている場合とプロジェクタから再生している場合を除き、ムービーの再生中は、Director のオーサリングメニューの Ctrl キーと Command キーショートカットが優先されます。

例文

次のステートメントは、再生ヘッドがフレームを移動し、ユーザが Ctrl+A（Windows）または Command+A（Macintosh）を押したときにプロジェクタの再生を一時停止します。

```
-- Lingo syntax
on enterFrame
    if (_key.commandDown and _key.key = "a") then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function enterFrame() {
    if (_key.commandDown && _key.key == "a") {
        _movie.go(_movie.frame);
    }
}
```

関連事項

キー , key

comments

使用法

```
-- Lingo syntax
memberObjRef.comments

// JavaScript syntax
memberObjRef.comments;
```

解説

メンバプロパティ。該当するキャストメンバの保持したいコメントまたはメンバに関連付けるその他のストリングの格納場所となります。読み取り / 書き込み可能。

このプロパティは、プロパティインスペクタの「メンバ」タブでも設定できます。

例文

次のステートメントは、メンバ Backdrop のコメントを "Still need to license this artwork" というストリングに設定します。

```
-- Lingo syntax
member("Backdrop").comments = "Still need to license this artwork"

// JavaScript syntax
member("Backdrop").comments = "Still need to license this artwork";
```

関連事項

メンバ

compressed

使用法

```
member(whichCastmember).texture(whichTexture).compressed
```

解説

3D テクスチャプロパティ。テクスチャのソースキャストメンバを圧縮するか（TRUE）、圧縮しないか（FALSE）を示します。テクスチャのレンダリングが必要な場合は、compressed プロパティの値は自動的に TRUE から FALSE に変更されます。あらかじめテクスチャを解凍しておく場合は、値を FALSE に設定します。解凍されたテクスチャをメモリから削除するには、値を TRUE に設定します。テクスチャで使用されるキャストメンバは、値が TRUE でも圧縮されません。ムービーを保存する場合にビットマップキャストメンバに対して使用される標準の圧縮は除きます。このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、テクスチャ " 冥王星の地図 " の compressed プロパティを TRUE に設定します。

```
-- Lingo syntax
member("scene").texture("Plutomap").compressed = TRUE

// Java Script
member("scene").getProp("texture",1).compressed = true;
```

関連事項

texture

connectionStatus (MP4Media/FLV)

使用法

```
member1.connectionStatus
sprite1.connectionStatus
```

解説

MP4Media/FLV キャストメンバおよびスプライトのプロパティ。MP4 アセットを RTMP ソースに接続したときの RTMP 接続の状態を返します。

connectionStatus は、次の 4 つのステータスのいずれかを返します。

- #idle— スプライトまたはメンバはサーバに接続されていません。ローカルおよび HTTP ソースは常にこのステータスを返します。
- #connecting— スプライトまたはメンバはサーバへの接続処理中です。サーバアドレスが無効の場合、またはサーバへの接続が失敗した場合、このステータスは #connecting のままです。
- #connected— スプライトまたはメンバはサーバに接続済みです。サーバアドレスが有効で、要求したビデオがサーバにない場合、このステータスは #connected のままです。
- #ready— スプライトまたはメンバは再生する準備が整っています。

例文

```
-- Lingo syntax
put(sprite("MP4MediaSprite").connectionStatus
put(member("MP4Media/FLV").connectionStatus

// JavaScript syntax
put(sprite("MP4MediaSprite").connectionStatus
put(member("MP4Media/FLV").connectionStatus
```

connectionStatus（サウンドオブジェクト）

使用法

```
soundobj.connectionStatus (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトに RTMP ソースがある場合の RTMP 接続のステータスを返します。

connectionStatus は、次の 4 つのステータスのいずれかを返します。

- #idle — サウンドオブジェクトはサーバに接続していません。確立する接続がないため、ローカルおよびリンクされたサウンドオブジェクトはこのステータスを返します。
- #connecting — サウンドオブジェクトはサーバへの接続処理中です。サーバアドレスが無効の場合、またはサーバへの接続が失敗した場合、このステータスは #connecting のままです。

- #connected — サウンドオブジェクトはサーバに接続しています。オーディオファイルが再生できないか利用不可である場合、ステータスは #connected のままになります。
- #ready — サウンドオブジェクトは再生の準備が完了しています。サウンドオブジェクトが停止した後、サウンドオブジェクトを再度再生するには、ステータスは #ready である必要があります。

例文

以下の例は、サウンドオブジェクトの connectionStatus を返します。

```
--Lingo syntax
on mouseUp me
put SoundObjectRef.connectionStatus
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.connectionStatus);
}
```

constraint

使用法

```
-- Lingo syntax
spriteObjRef.constraint

// JavaScript syntax
spriteObjRef.constraint;
```

解説

スプライトのセンターポイントが、別のスプライトの境界矩形内に制限されているかどうかを調べるスプライトプロパティです。制限されているときは、1 または TRUE を返し、制限されていないときは 0 または FALSE（デフォルト）を返します。読み取り / 書き込み可能。

constraint スプライトプロパティは、移動可能なスプライトの動きを別のスプライトの境界矩形内に制限するときに役に立ちます。

constraint プロパティは、移動可能なスプライトおよび locH と locV プロパティに影響を与えます。移動可能なスプライトの制限点は、制限スプライトの境界矩形外に動かすことはできません。ビットマップスプライトの制限点はセンターポイントです。シェイプスプライトの制限点は左上隅です。動きを制限されているスプライトでは、制限設定が locH と locV スプライトプロパティよりも優先されます。

例文

次のステートメントは、constraint スプライトプロパティの制限を解除します。

```
-- Lingo syntax
sprite(5).constraint = 0

// JavaScript syntax
sprite(5).constraint = 0;
```

次のステートメントは、スプライト（i + 1）の移動範囲をスプライト 14 の境界領域内に制限します。

```
-- Lingo syntax
sprite(i + 1).constraint = 14

// JavaScript syntax
sprite(i + 1).constraint = 14;
```

次のステートメントは、スプライト 3 の動きが制限されているかどうかを調べ、制限されている場合は、showConstraint ハンドラを呼び出します。

```
-- Lingo syntax
if (sprite(3).constraint <> 0) then
    showConstraint
end if

// JavaScript syntax
if (sprite(3).constraint != 0) {
    showConstraint();
}
```

関連事項

locH, locV, スプライト

controlDown

使用法

```
-- Lingo syntax
_key.controlDown

// JavaScript syntax
_key.controlDown;
```

解説

Ctrl キーが押されているかどうかを調べるキープロパティです。読み取り専用。

このプロパティは、Ctrl キーが押された場合は TRUE を返し、そうでない場合は FALSE を返します。

controlDown を key プロパティと組み合わせて使用すると、Ctrl キーとその他のキーが同時に押されたかどうかを調べることができます。この機能を利用して、ユーザが Ctrl キーと特定のキーを同時に押したときに実行するハンドラを作成することもできます。

Lingo または JavaScript シンタックスのカスタムメニューがインストールされている場合とプロジェクタから再生している場合を除き、ムービーの再生中は、Director のオーサリングメニューの Ctrl キーとショートカットが優先されます。

例文

次の on keyDown ハンドラは、Ctrl または Control キーが押されたかどうかを調べ、押されている場合は、on doControlKey ハンドラを呼び出します。パラメータには Ctrl または Control キーと一緒に押されたキー（_key.key）を渡します。

```
-- Lingo syntax
on keyDown
    if (_key.controlDown) then
        doControlKey(_key.key)
    end if
end

on doControlKey(theKey)
    trace("The " & theKey & " key is down")
end

// JavaScript syntax
function keyDown() {
    if (_key.controlDown) {
        doControlKey(_key.key);
    }
}

function doControlKey(theKey) {
    trace("The " & theKey & " key is down");
}
```

関連事項

キー , key

controller

使用法

```
member(whichCastMember).controller
the controller of member whichCastMember
```

解説

デジタルビデオムービーキャストメンバの制御パネルを表示するかどうかを定義するプロパティです。このプロパティを 1 に設定すると制御パネルが表示され、0 に設定すると表示されません。

controller メンバプロパティは、QuickTime デジタルビデオのみに適用できます。

- Video for Windows デジタルビデオに the controller メンバプロパティを設定しても、何の処理も実行されず、エラーメッセージも表示されません。
- Video for Windows デジタルビデオの controller メンバプロパティを調べると、常に FALSE が返されます。

制御パネルを表示するには、デジタルビデオが directToStage 再生モードになっている必要があります。

例文

次のステートメントは、QuickTime キャストメンバ " デモ " に制御パネルを表示します。

```
--Lingo Dot syntax:
member("Demo").controller = 1

--Lingo Verbose syntax:
set the controller of member "Demo" to 1

// JavaScript syntax
member("Demo").controller = 1;
```

関連事項
directToStage

copyrightInfo （ムービー）

使用法

```
-- Lingo syntax
_movie.copyrightInfo

// JavaScript syntax
_movie.copyrightInfo;
```

解説
オーサリング時に、[ムービープロパティ] ダイアログボックスに入力するストリングでムービープロパティです。このプロパティは、Shockwave Player の将来のバージョンでの機能拡張のために用意されています。読み取り専用。

関連事項
aboutInfo, ムービー

copyrightInfo（SWA）

使用法

```
-- Lingo syntax
memberObjRef.copyrightInfo

// JavaScript syntax
memberObjRef.copyrightInfo;
```

解説
このプロパティの内容は、Shockwave Audio (SWA) ファイルの著作権に関するテキストです。SWA サウンドの再生が開始されたか、または preLoadBuffer コマンドを使ってファイルがプリロードされた後にのみ、このプロパティは有効になります。

このプロパティは、値を調べたり、設定することができます。

例文
次のステートメントは、フィールドキャストメンバ " 著作権情報 " 内に Shockwave Audio ファイル "SWA ファイル " の著作権情報を表示します。

```
-- Lingo syntax
whatState = member("SWAfile").state
if whatState > 1 AND whatState < 9 then
    member("Info Display").text = member("SWAfile").copyrightInfo
end if

// JavaScript syntax
var whatState = member("SWAfile").state;
if (whatState > 1 && whatState < 9) {
    member("Info Display").text = member("SWAfile").copyrightInfo;
}
```

count

使用法

```
list.count
count (list)
count(theObject)
object.count
textExpression.count
```

解説

リニアリストまたはプロパティリストの項目数を返すプロパティ（Lingo のみ）です。親スクリプトのプロパティ数を数える場合、アンセスタスクリプトのプロパティや図、線または単語などテキスト表記のチャンクは数には入れません。

count コマンドはリニアリスト、プロパティリスト、親スクリプトで作成されたオブジェクトおよび the globals プロパティのいずれでも動作します。

完成したムービーの count() の実例は、Director アプリケーションフォルダの "Learning/Lingo_Example/" フォルダ内の "text.dir" を参照してください。

例文

次のステートメントは、リスト内の項目数 "3" を表示します。

```
put [10,20,30].count
-- 3
```

関連事項

globals

count (3D)

使用法

```
member(whichCastmember).light.count
member(whichCastmember).camera.count
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model.count
member(whichCastmember).group.count
member(whichCastmember).shader.count
member(whichCastmember).texture.count
member(whichCastmember).modelResource.count
member(whichCastmember).motion.count
member(whichCastmember).light.child.count
member(whichCastmember).camera.child.count
member(whichCastmember).model.child.count
member(whichCastmember).group.child.count
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
sprite(whichSprite).camera{(index)}.overlay.count
member(whichCastmember).camera(whichCamera).overlay.count
member(whichCastmember).model(whichModel).modifier.count
member(whichCastmember).model(whichModel).keyframePlayer.playlist.count
member(whichCastmember).model(whichModel).bonesPlayer.playlist.count
member(whichCastmember).modelResource(whichModelResource).face.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
member(whichCastmember).model(whichModel).meshDeform.mesh.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].face.count
```

解説

3D プロパティ。3D オブジェクトに関連付けられたリストのアイテム数を返します。どのオブジェクトにも使用できます。

face.count プロパティにより、#mesh というタイプのモデルリソースでメッシュ内の三角形の数を取得できます。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、"3D ワールド " という 3D キャストメンバ内の、様々な種類のオブジェクトの数を決定します。

```
numberOfCameras = member("3D World").camera.count
put member("3D World").light.count
-- 3
numberOfModels = member("3D World").model.count
numberOfTextures = member("3D World").texture.count
put member("3D World").modelResource("mesh2").face.count
-- 4
```

次のステートメントは、" 耳 " というモデルの最初のメッシュが 58 面で構成されていることを示します。

```
put member("Scene").model("Ear").meshdeform.mesh[1].face.count
-- 58
```

次のステートメントは、" 耳 " というモデルが 3 つのメッシュで構成されていることを示します。

```
put member("Scene").model("Ear").meshdeform.mesh.count
-- 3
```

次のステートメントは、" 耳 " というモデルの最初のメッシュに、2 つのテクスチャレイヤーがあることを示します。

```
put member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.count
-- 2
```

関連事項

cameraCount()

count (castLib)

使用法

```
-- Lingo syntax
castObjRef.count

// JavaScript syntax
castObjRef.count;
```

解説

ムービー内のキャストライブラリの数を返します。読み取り専用。

例文

次のステートメントは、ムービー内のキャストメンバの数をメッセージウィンドウに表示します。

```
-- Lingo syntax
  x = castLib(1)
  put x.count

// JavaScript syntax
  var cst = castLib(2);
  trace(cst.count);
```

このプロパティには、_movie.castLib.count 式を使用してアクセスすることもできます。

関連事項
globals

cpuHogTicks

使用法
```
the cpuHogTicks
```

解説
Director がバックグラウンドイベント処理 (他のアプリケーションのイベント、ネットワークイベント、時刻の更新、その他のキーボードイベントなど) のために CPU を解放する頻度を定義するシステムプロパティです。

デフォルト値は 20 チックです。バックグラウンドのイベントへの CPU リリース前の Director に与える時間を多くしたり、コンピュータのネットワークの操作に関する反応を制御したりするには、cpuHogTicks の値を大きく設定します。これにより、Director がバックグラウンドイベントのために CPU を解放するまでの時間が長くなります。

自動繰り返しキー入力の速度を早くし、アニメーションの速度を遅くするには、cpuHogTicks の値を低く設定します。ムービーでは、ユーザがキーを押し続けて自動繰り返しキー入力を高速で行うと、ほとんどの場合、Director はコンピュータのコントロールパネルの設定より低い頻度で自動繰り返しキー入力をチェックします。

cpuHogTicks プロパティは、Mac でのみ有効です。

例文
次のステートメントは、Director が CPU を 6 チック (1/10 秒) おきに解放するように設定します。

```
the cpuHogTicks = 6
```

関連事項
milliseconds

creaseAngle

使用法
```
member(whichCastmember).model(whichModel).inker.creaseAngle
member(whichCastmember).model(whichModel).toon.creaseAngle
```

解説
3D inker および toon モディファイアプロパティ。モデルのジオメトリにある折り目に対する、モディファイアの線の描画機能の感度を示します。設定値を大きくするほど、折り目に多くの (詳細な) 線が描画されます。

creaseAngle プロパティを有効にするには、モディファイアの creases プロパティを TRUE に設定する必要があります。

CreaseAngle の範囲は -1.0 ～ +1.0 です。デフォルトの設定は 0.01 です。

例文

次のステートメントは、" ティーポット " というモデルに適用される inker モディファイアの creaseAngle プロパティを 0.10 に設定します。このしきい値を超えるモデルの折り目すべてに、線が描画されます。この設定は、inker モディファイアの creases プロパティが TRUE に設定されている場合にのみ有効になります。

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creaseAngle = 0.10

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creaseAngle = 0.10;
```

関連事項

creases, lineColor, lineOffset, useLineOffset

creases

使用法

```
member(whichCastmember).model(whichModel).inker.creases
member(whichCastmember).model(whichModel).toon.creases
```

解説

3D toon および inker モディファイアプロパティ。モデルの表面の折り目で線を描画するかどうかを決定します。

このプロパティのデフォルト設定は TRUE です。

例文

次のステートメントは、" ティーポット " というモデルで inker モディファイアの creases プロパティを TRUE に設定します。inker モディファイアの creaseAngle プロパティで設定されたしきい値を超えるモデルの折り目すべてで、線が描画されます。

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creases = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creases = true;
```

関連事項

creaseAngle, lineColor, lineOffset, useLineOffset

creationDate

使用法

```
-- Lingo syntax
memberObjRef.creationDate

// JavaScript syntax
memberObjRef.creationDate;
```

解説

コンピュータのシステムクロックを使用して、キャストメンバが最初に作成された日付と時間を記録するメンバプロパティです。読み取り専用。

プロジェクトの予定にこのプロパティを使用できます。Director は、その他の目的にこのプロパティを使用しません。

例文

通常は、プロパティインスペクタまたはキャストウィンドウのリストビューで creationDate プロパティを調べますが、メッセージウィンドウでも確認できます。

```
-- Lingo syntax
put(member(1).creationDate)

// JavaScript syntax
put(member(1).creationDate);
```

関連事項

メンバ

crop

使用法

```
member(whichCastMember).crop
the crop of member whichCastMember
```

解説

デジタルビデオキャストメンバが表示されているスプライトの矩形内に正確に収まるように伸縮する（FALSE）か、伸縮せずにスプライトの矩形内に収まるようにキャストメンバを切り取る（TRUE）キャストメンバプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、" インタビュー " デジタルビデオキャストメンバをスプライトに合わせて切り取るように Lingo に指示します。

```
-- Lingo Dot syntax:
member("Interview").crop = TRUE

-- Lingo Verbose syntax:
set the crop of member "Interview" to TRUE

// JavaScript syntax
member("Interview").crop = true;
```

関連事項

center

cuePointNames

使用法

```
-- Lingo syntax
memberObjRef.cuePointNames

// JavaScript syntax
memberObjRef.cuePointNames;
```

解説

キューポイント名のリストを返すキャストメンバプロパティです。キューポイントに名前が付けられていない場合、代わりに空白のストリング ("") がリストに挿入されます。) キューポイント名は、サウンドや QuickTime、アニメーションの同期に使用できます。

このプロパティは、SoundEdit キャストメンバ、QuickTime デジタルビデオキャストメンバ、およびキューポイントが設定されている Xtra キャストメンバでサポートされています。実行時にキューポイントを生成する Xtra では、キューポイント名のリストを使用できないことがあります。

例文

次のステートメントは、キャストメンバの 3 番目のキューポイントを表示します。

```
-- Lingo syntax
put member("symphony").cuePointNames[3]

// JavaScript syntax
put(member("symphony").cuePointNames[3]);
```

関連事項

cuePointTimes, mostRecentCuePoint

cuePointTimes

使用法

```
-- Lingo syntax
memberObjRef.cuePointTimes

// JavaScript syntax
memberObjRef.cuePointTimes;
```

解説

指定したキャストメンバのキューポイント時間のリスト（単位はミリ秒）を返すキャストメンバプロパティです。キューポイント時間は、サウンドや QuickTime、アニメーションの同期するとき便利です。

このプロパティは、SoundEdit キャストメンバ、QuickTime デジタルビデオキャストメンバ、およびキューポイントをサポートする Xtra キャストメンバでサポートされています。実行時にキューポイントを生成する Xtra では、キューポイント名のリストを使用できないことがあります。

例文

次のステートメントは、サウンドキャストメンバの 3 番目のキューポイントを表示します。

```
-- Lingo syntax
put member("symphony").cuePointTimes[3]

// JavaScript syntax
put(member("symphony").cuePointTimes[3]);
```

関連事項

cuePointNames, mostRecentCuePoint

currentLoopState

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.currentLoopState
member(whichCastmember).model(whichModel).bonesPlayer.currentLoopState
```

解説

3D keyframePlayer および bonesPlayer モディファイアプロパティ。モデルで実行されているモーションを繰り返すか（TRUE）、最後まで再生してモディファイアのプレイリストにある次のモーションに置き換えるか（FALSE）を示します。

このプロパティのデフォルト設定は、モーションの再生を開始した play() コマンドのループパラメータの値か、モディファイアのプレイリストにモーションを追加した queue() コマンドの値です。currentLoopState プロパティを変更すると、モディファイアのプレイリストにあるモーションの #looped プロパティ値も変化します。

例文

次のステートメントは、" モンスター " というモデルによって実行されているモーションを繰り返します。

```
-- Lingo syntax
member("NewAlien").model("Monster").addModifier(#keyframeplayer)
member("NewAlien").model("Monster").keyframePlayer.currentLoopState = TRUE

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentLoopState = true;
```

関連事項

loop (3D), play() (3D), queue() (3D), playlist

currentSpriteNum

使用法

```
-- Lingo syntax
_player.currentSpriteNum

// JavaScript syntax
_player.currentSpriteNum;
```

解説

現在スクリプトが実行されているスプライトのチャネル番号を返すプレーヤプロパティです。読み取り専用。

このプロパティは、ビヘイビアーとキャストメンバスクリプトで有効です。フレームスクリプトやムービースクリプトで使用するとき、currentSpriteNum プロパティの値は 0 です。

currentSpriteNum プロパティは、スプライトオブジェクトの spriteNum プロパティと似ています。

注意：このプロパティはビヘイビアーが導入された Director 6 に、古いムービーを移行するときに使用されていました。これは、スクリプトを完全に書き替えることなく、ビヘイビアーのような機能を持っていたためです。ビヘイビアーでオーサリングする場合は必要なく、現在は以前ほど有用ではありません。

例文

キャストメンバまたはムービースクリプト内の次のハンドラは、mouseDown イベントが起こったスプライトに割り当てたキャストメンバを交換します。

```
-- Lingo syntax
on mouseDown
    sprite(_player.currentSpriteNum).member = member("DownPict")
end

// JavaScript syntax
function mouseDown() {
    sprite(_player.currentSpriteNum).member = member("DownPict");
}
```

関連事項

プレーヤ , spriteNum

currentTime (3D)

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.currentTime
member(whichCastmember).model(whichModel).bonesPlayer.currentTime
```

解説

3D keyframePlayer および bonesPlayer モディファイアプロパティ。モデルによって実行されているモーションのローカルタイムを示します。currentTime プロパティはミリ秒単位で計測されますが、モーションが元の速度で再生されるときの実際の時間にのみ対応します。

モデルによるモーションの再生は、play() コマンドまたは queue() コマンドによって指定されます。play() または queue() コマンドの scale パラメータは、モディファイアの playRate プロパティと掛け合わされます。結果の値とモーションの元の速度を掛けて、モデルがモーションを実行する速度と、モーションのローカルタイムの経過速度が決定されます。したがって、scale パラメータの値が 2 で、モディファイアの playRate プロパティの値が 3 の場合、モデルはモーションを元の速度の 6 倍で実行し、ローカルタイムは実際の時間の 6 倍の速さで経過します。

currentTime プロパティは、ループしたモーションの各繰り返しの開始時に、play() コマンドまたは queue() コマンドの cropStart パラメータの値をリセットします。

例文

次のステートメントは、" エイリアン 3" というモデルで実行されているモーションのローカルタイムを表示します。

```
-- Lingo syntax
member("NewAlien").model("Alien3").addModifier(#keyframeplayer)
put member("NewAlien").model("Alien3").keyframePlayer.play()
put member("NewAlien").model("Alien3").keyframePlayer.currentTime
-- 1393.8599

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").play();
put(member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentTime );
// 1393.8599
```

関連事項

play() (3D), queue() (3D), playlist

currentTime (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.currentTime

// JavaScript syntax
dvdObjRef.currentTime;
```

解説

経過時間をミリ秒単位で返す DVD プロパティです。読み取り / 書き込み可能。

例文

次のステートメントは経過時間を返します。

```
-- Lingo syntax
trace (member(1).currentTime)-- 11500

// JavaScript syntax
trace (member(1).currentTime);// 11500
```

次のステートメントは、currentTime を現行タイトルの特定のポイントに設定します。

```
-- Lingo syntax
member(1).currentTime = 22000

// JavaScript syntax
member(1).currentTime = 22000
```

関連事項

DVD

currentTime (QuickTime, AVI)

使用法

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

解説

Quicktime、AVI、または MP4 ビデオスプライトの現在の時間をミリ秒単位で返すデジタルビデオスプライトプロパティです。currentTime プロパティは、QuickTime と AVI の movieTime を置き換えます。

このプロパティの値は、QuickTime と AVI でテストおよび設定可能で、MP4 ではテストのみを行うことができます。seek メソッドを使用して、MP4 の再生時間を設定します。

完成したムービーの currentTime の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "lingo QT and Flash.dir" を参照してください。

例文

次のステートメントは、チャネル 9 上の QuickTime ムービーの現行時間をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(9).currentTime)
// JavaScript syntax
put(sprite(9).currentTime);
```

次のステートメントは、チャネル 9 の QuickTime ムービーの現在の時間を 2 秒に設定します。

```
-- Lingo syntax
sprite(9).currentTime = 2000
// JavaScript syntax
sprite(9).currentTime = 2000;
```

関連事項

duration （メンバ）

currentTime (MP4Media/FLV)

使用法

```
put sprite(1).currentTime
put member(1).currentTime
```

解説

再生中の MP4Media/FLV メンバまたはスプライトの現在の時間を返すか、または設定する、MP4Media/FLV メンバまたはスプライトのプロパティです。

currentTime には、0 からビデオメンバの合計の長さまでのミリ秒単位の整数値をとります。

- sprite.play() はビデオとサウンドトラックの両方を再生し、sprite.currentTime を更新します。
- member.play() はサウンドトラックのみを再生し、member.currentTime と member.mixer.soundObjectList[1].currentTime を更新します。更新には約 3 ミリ秒かかる場合があります。
- sprite.currentTime は整数で、member.currentTime は浮動小数点数です。

例文

次の例は、MP4 ビデオの現在の時間を返します。

```
-- Lingo syntax
put sprite(7).currentTime   -- 3000
put member("MP4Media/FLV").currentTime -- 3000

// JavaScript syntax
put sprite(7).currentTime  // 3000
put member("MP4Media/FLV").currentTime  // 3000
```

currentTime (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.currentTime

// JavaScript syntax
memberOrSpriteObjRef.currentTime;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。RealMedia ストリームの現在の時間をミリ秒単位で取得または設定します。RealMedia キャストメンバが再生されていない場合、このプロパティの値は 0 になります。0 はデフォルトの設定値です。このプロパティは、保存されることのない再生プロパティです。

currentTime プロパティを設定または変更するときにストリームが再生中のときは、シークアクションが発生し、ストリームが再バッファされ、新しい時間から再生が再開します。currentTime を設定または変更するときにストリームが一時停止している（#paused mediaStatus 値）場合、ストリームは新しい時間からフレームを再描画し、pausedAtStart が FALSE に設定されている場合は、再生を再開します。RealMedia ビューアが一時停止または停止しているときは、mediaStatus 値は #paused になります。Lingo の stop コマンドによってストリームが停止されたときは、mediaStatus 値は #closed になります。ストリームの mediaStatus 値が #closed である場合、このプロパティは無効です。整数値を設定するとき、値は 0 からストリームの長さの範囲に切り捨てられます。

currentTime の設定は、seek コマンドを起動することと同じです。つまり、x.seek(n) は x.currentTime = n と同じことになります。currentTime の変更または seek の呼び出しには、ストリームの再バッファが必要です。

例文

次のサンプルは、スプライト "2" とキャストメンバ "Real" の現在の時間が、ストリームの先頭から 15,534 ミリ秒 (15.534 秒) であることを表示します。

```
-- Lingo syntax
put(sprite(2).currentTime) -- 15534
put(member("Real").currentTime) -- 15534

// JavaScript syntax
put(sprite(2).currentTime) // 15534
put(member("Real").currentTime) // 15534
```

次のサンプルは、スプライト "2" とキャストメンバ "Real" の再生を 20,000 ミリ秒（20 秒）から開始させます。

```
-- Lingo syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000

// JavaScript syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000
```

関連事項

duration (RealMedia, SWA), seek(), mediaStatus (RealMedia, Windows Media)

currentTime（サウンドオブジェクト）

使用法

```
soundObject.currentTime (Read-only)
```

解説

現在のサウンドオブジェクトが再生した時間をミリ秒単位で返すサウンドオブジェクトプロパティです。このプロパティは読み取り専用です。

例文

```
--Lingo syntax
on mouseUp me
put soundObjRef.currentTime -- Gives the time, in milliseconds, that the
-- current sound object has been playing for.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.currentTime);
}
```

currentTime （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

解説

スプライトおよびサウンドチャネルプロパティ。サウンドスプライト、QuickTime デジタルビデオスプライト、MP4/F4V/FLV スプライト、またはキューポイントをサポートする Xtra 拡張機能の現在の再生時間をミリ秒で返します。サウンドチャネルについては、指定したサウンドチャネルで再生されているサウンドメンバの現在の再生時間を返します。

このプロパティを調べることはできますが、従来のサウンドキャストメンバ (WAV、AIFF、SND) でしか設定できません。このプロパティが設定されると、ゼロ (0) からメンバの duration (継続時間) までの値を指定できます。

Shockwave Audio（SWA）サウンドはスプライトチャネルにスプライトとして表示されますが、そのサウンドはサウンドチャネルで再生されます。SWA サウンドスプライトは、サウンドチャネル番号ではなく、スプライトチャネル番号で参照する必要があります。

例文

次のステートメントは、スプライトチャネル 10 のサウンドスプライトの現在の再生時間を秒単位で表示します。

```
-- Lingo syntax
member("time").text = string(sprite(10).currentTime/ 1000)

// JavaScript syntax
member("time").text = (sprite(10).currentTime / 1000).toString();
```

次のステートメントは、サウンドチャネル 2 で演奏されているサウンドをサウンドキャストメンバの始まりから 2.7 秒地点へスキップします。

```
-- Lingo syntax
sound(2).currentTime = 2700

// JavaScript syntax
sound(2).currentTime = 2700;
```

関連事項

duration（メンバ）

cursor

使用法

```
-- Lingo syntax
spriteObjRef.cursor

// JavaScript syntax
spriteObjRef.cursor;
```

解説

ポインタがスプライト上にあるときに使用するカーソルを定義するスプライトプロパティです。読み取り / 書き込み可能。

このプロパティは、カーソルを 0 に設定して解除するまで効果が維持されます。cursor プロパティを使用すると、画面上の指定された場所にマウスポインタが移動したときに、カーソルを変更することにより、ユーザがその場所をクリックするとアクションが可能になることを示すことができます。

所定のフレーム内で cursor スプライトプロパティを設定すると、Director は指定したスプライトの矩形を監視してカーソルの形状を変えるべきかどうかを判断します。この矩形は、そのチャネル用の cursor プロパティを 0 に設定しない限り、ムービーが別のフレームに移動した後も残ります。

- カーソルとそのマスクに使用するキャストメンバの番号を指定するには、次のシンタックスを使用します。

  ```
  -- Lingo syntax
  spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef]

  // JavaScript syntax
  spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef];
  ```

- デフォルトのシステムカーソルを指定するときは、次のシンタックスを使用します。

```
-- Lingo syntax
spriteObjRef.cursor = castMemberObjRef

// JavaScript syntax
spriteObjRef.cursor = castMemberObjRef;
```

cursor プロパティは、次のいずれかの整数値になります。

値	解説
-1, 0	矢印
1	I 字
2	十字
3	クロスバー
4	時計（Mac）または砂時計（Windows）
5	東西南北（NSEW）
6	南北（NS）
200	空白（カーソルは非表示）
254	ヘルプ
256	えんぴつ
257	消しゴム
258	選択
259	バケツ
260	ハンド
261	矩形ツール
262	角丸矩形ツール
263	円ツール
264	ラインツール
265	リッチテキストツール
266	テキストフィールドツール
267	ボタンツール
268	チェックボックスツール
269	ラジオボタンツール
270	配置ツール
271	センターポイントツール
272	投げ縄
280	指
281	スポイト
282	マウスボタン押下待ち 1

値	解説
283	マウスボタン押下待ち 2
284	垂直サイズ
285	水平サイズ
286	対角線サイズ
290	閉じたハンド
291	非ドロップハンド
292	コピー（閉じたハンド）
293	逆矢印
294	回転
295	傾斜
296	水平二重矢印
297	垂直二重矢印
298	南西北西の二重矢印
299	北西南東の二重矢印
300	にじみ / ソフトブラシ
301	エアブラシ
302	ズームイン
303	ズームアウト
304	ズームキャンセル
305	シェイプを開始
306	ポイントを追加
307	シェイプを閉じる
308	ズームカメラ
309	カメラを移動
310	カメラを回転
457	カスタム

カスタムカーソルを使用するときは、cursor スプライトプロパティを、カーソルとして使用するキャストメンバを含んだリストまたはシステムカーソルを指定する番号に設定してください。Windows では、カーソルはリソースではなくキャストメンバである必要があります。カーソルがリソースであるために使用できないときは、標準の矢印ポインタが代わりに使用されます。クロスプラットフォーム用のムービーを作成する場合は、カスタムカーソルを使用しないのがベストです。

カスタムカーソルのキャストメンバは、16 × 16 ピクセルより大きくすることはできず、1 ビットのビット深度である必要があります。

スプライトがビットマップの場合には、マットインクが適用され、カーソルがスプライトのマットインク上にあるときにだけ変わります。

カーソルが既にステージから消えているスプライトの位置に移動したときでも、ロールオーバーが生じます。この状況を防ぐには、このような位置でのロールオーバーをしないようにするか、削除する前にスプライトの位置をステージの外に移動しておきます。

Macintosh では、cursor をカーソルリソースの番号に設定することにより、現在開いているムービー内でその番号のカーソルリソースをカーソルとして使用することができます。

例文

次のステートメントは、スプライト 20 の上に表示されるカーソルを腕時計（Macintosh）または砂時計（Windows）カーソルに変更します。

```
-- Lingo syntax
sprite(20).cursor = 4

// JavaScript syntax
sprite(20).cursor = 4;
```

関連事項

スプライト

cursorSize

使用法

```
-- Lingo syntax
memberObjRef.cursorSize

// JavaScript syntax
memberObjRef.cursorSize;
```

解説

whichCursorCastMember というアニメーションカラーカーソルのキャストメンバのサイズを指定するカーソルキャストメンバプロパティです。

サイズの指定	カーソルの最大サイズ
16	16 × 16 ピクセル
32	32 × 32 ピクセル

指定されたサイズよりも小さいビットマップキャストメンバはフルサイズで表示され、大きいビットマップキャストメンバは指定サイズに合うように縮小されます。

デフォルト値は、Windows では 32、Macintosh では 16 です。無効な値を設定すると、コンパイル時ではなくムービーの実行時にエラーメッセージが表示されます。

このプロパティは、値を調べたり、設定することができます。

例文

次のコマンドは、キャストメンバ 20 に格納されているアニメーションカラーカーソルのサイズを 32 × 32 ピクセルに変更します。

```
-- Lingo syntax
member(20).cursorSize = 32

// JavaScript syntax
member(20).cursorSize = 32;
```

curve

使用法

```
-- Lingo syntax
memberObjRef.curve[curveListIndex]

// JavaScript syntax
memberObjRef.curve[curveListIndex];
```

解説

このプロパティには、ベクトルシェイプのキャストメンバから各曲線 (ベクトルシェイプ) の vertexList があります。ベクトルシェイプの特定の曲線の頂点を得るために、vertex プロパティに基づき、curve プロパティを使用できます。

vertexList は頂点のリストで、各頂点は、頂点の point である #vertex プロパティ、頂点の最初のコントロールポイントの point である #handle1 プロパティ、2 番目のコントロールポイントの point である #handle2 プロパティの中から最大 3 つまでのプロパティを含むプロパティリストになっています。「vertexList」を参照してください。

例文

次のステートメントは、ベクトルシェイプメンバ SimpleCurves の 3 番目の頂点のサンプルリストを表示します。

```
-- Lingo syntax
put(member("SimpleCurves").curve[3])
-- [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000), #handle2: point(-32.0000, -
10.0000)], [#vertex: point(164.0000, 56.0000)]]

// JavaScript syntax
put(member("SimpleCurves").curve[3]);
// [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000),#handle2: point(-32.0000, -
10.0000)], [#vertex: point(164.0000, 56.0000)]]
```

次のステートメントは、ベクトルシェイプの最初のカーブの最初の頂点を 10 ピクセルずつ右下に移動します。

```
-- Lingo syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10)

// JavaScript syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10);
```

次のコードは、スプライト 7 をベクトルシェイプの最初のカーブの最初の頂点の point に移動します。ベクトルシェイプの originMode はあらかじめ #topLeft に設定されている必要があります。

```
-- Lingo syntax
vertexLoc = member(1).curve[1].vertex[1]
spriteLoc = mapMemberToStage(sprite(3), vertexLoc)
sprite(7).loc = spriteLoc

// JavaScript syntax
var vertexLoc = member(1).curve[1].vertex[1];
var spriteLoc = mapMemberToStage(sprite(3), vertexLoc);
sprite(7).loc = spriteLoc;
```

debug

使用法

```
member(whichCastmember).model(whichModel).debug
```

解説

3D モデルプロパティ。モデルを囲む球体とモデルのローカル軸を表示するかどうかを示します。

例文

次のステートメントは、" 犬 " というモデルの debug プロパティを TRUE に設定します。

```
-- Lingo syntax
member("ParkScene").model("Dog").debug = TRUE

// JavaScript syntax
member("ParkScene").getProp("model", 1).debug = true;
```

関連事項

boundingSphere

debugPlaybackEnabled

使用法

```
-- Lingo syntax
_player.debugPlaybackEnabled

// JavaScript syntax
_player.debugPlaybackEnabled;
```

解説

Windows 環境において、Shockwave およびプロジェクタで、デバッグ用のメッセージウィンドウを開くプレーヤプロパティです。Macintosh 環境では、put ステートメントがデバッグ用のデータを出力できるようにするため、ログファイルが生成されます。読み取り / 書き込み可能。

Windows 環境では、このプロパティを Director アプリケーションで使用しても、効果はまったくありません。特定の Shockwave Player またはプロジェクタのセッションに対して開かれたメッセージウィンドウを一度閉じたら、再び開くことはできません。1 つのブラウザ内で、Shockwave コンテンツを含む複数のムービーがこの Lingo を使用している場合は、最初のムービーのメッセージウィンドウだけが開かれ、そのメッセージウィンドウは最初のムービーだけと関連付けられます。

Macintosh でメッセージウィンドウの項目を表示するには、\Library\Logs\ にある 1msgdump.txt ファイルを開きます。

このメッセージウィンドウを開くには、debugPlaybackEnabled プロパティを TRUE に設定します。このメッセージウィンドウを閉じるには、debugPlaybackEnabled プロパティを FALSE に設定します。

例文

次のステートメントは、Shockwave Player またはプロジェクタのいずれかでメッセージウィンドウを開きます。

```
-- Lingo syntax
_player.debugPlaybackEnabled = TRUE

// JavaScript syntax
_player.debugPlaybackEnabled = true;
```

関連事項

プレーヤ , put()

decayMode

使用法

```
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
```

解説

3D プロパティ。カメラの fog.enabled プロパティが TRUE に設定されているときに、フォグ密度を最小から最大の密度まで構築する方法を示します。

このプロパティで使用できる値は以下のとおりです。

- #linear：fog.near から fog.far まで、フォグは一定の比率で濃度を上げながら適用されます。
- #exponential：fog.far が飽和点となり、fog.near は無視されます。
- #exponential2：fog.near が飽和点となり、fog.far は無視されます。

このプロパティのデフォルト設定は #exponential です。

例文

次のステートメントは、Defaultview というカメラのフォグの decayMode プロパティを #linear に設定します。フォグの enabled プロパティが TRUE に設定されている場合、フォグの密度は near および far プロパティで設定された距離に対して一定の割合で増加します。near プロパティが 100 に、far プロパティが 1000 に設定されている場合、フォグはカメラの前 100 ワールド単位から始まり、カメラの前 1000 ワールド単位の距離まで徐々に増加します。

```
-- Lingo syntax
member("3d world").camera("Defaultview").fog.decayMode = #linear

// Java Script
member("3d world").getProp("camera",1).getPropRef("fog").decayMode = symbol("linear");
```

関連事項

fog, near (fog), far (fog), enabled (fog)

defaultRect

使用法

```
-- Lingo syntax
memberObjRef.defaultRect

// JavaScript syntax
memberObjRef.defaultRect;
```

解説

Flash ムービーやベクトルシェイプキャストメンバから作成されるすべての新しいスプライト用のデフォルトサイズを制御するキャストメンバプロパティです。defaultRect 設定は、ステージ上にストレッチされていない既存のすべてのスプライトにも適用します。プロパティの値は、rect(0, 0, 32, 32) のように Director の矩形として指定します。

defaultRect メンバプロパティは、キャストメンバの defaultRectMode メンバプロパティによって影響を受けます。ムービーがキャストに挿入されると、defaultRectMode プロパティは常に #Flash に設定されます。これはオリジナルの defaultRect 設定は、もともと Flash で作成されたムービーのサイズと常に同じであることを意味します。その後に defaultRect プロパティを設定すると、キャストメンバの defaultRectMode プロパティは暗黙的に #fixed になります。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、キャスト参照を受け取ります。次に、Flash キャストメンバとして指定されたキャストを検索し、defaultRect プロパティを指定された矩形として設定します。

```
-- Lingo syntax
on setDefaultFlashRect(whichCast, whichRect)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRect = whichRect
        end if
    end repeat
end

// JavaScript syntax
function setDefaultFlashRect(whichCast, whichRect) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRect = whichRect;
            i++;
        }
    }
}
```

関連事項

defaultRectMode, flashRect

defaultRectMode

使用法

```
-- Lingo syntax
memberObjRef.defaultRectMode

// JavaScript syntax
memberObjRef.defaultRectMode;
```

解説

Flash ムービーやベクトルシェイプキャストメンバから作成されるすべての新しいスプライト用のデフォルトサイズを制御するキャストメンバプロパティです。プロパティの値は、rect(0, 0, 32, 32) のように Director の矩形として指定します。

defaultRectMode プロパティは、Flash ムービーのデフォルト矩形の実サイズを設定することはなく、デフォルト矩形がどのように設定されているかを確認するだけです。defaultRectMode メンバプロパティには次のような値があります。

- #flash（デフォルト）- もともと Flash で作成されたムービーのサイズを使用してデフォルト矩形を設定します。
- #fixed - defaultRect メンバプロパティにより指定された固定サイズを使用してデフォルト矩形を設定します。

  defaultRect メンバプロパティは、キャストメンバの defaultRectMode メンバプロパティによって影響を受けます。ムービーがキャストに挿入されると、defaultRectMode プロパティは常に #flash に設定されます。これはオリジナルの defaultRect 設定は、もともと Flash で作成されたムービーのサイズと常に同じであることを意味します。その後に defaultRect プロパティを設定すると、キャストメンバの defaultRectMode プロパティは暗黙的に #fixed になります。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、キャスト参照を受け取ります。次に、Flash キャストメンバとして指定されたキャストを検索し、defaultRectMode プロパティを #fixed に、defaultRect プロパティを rect(0, 0, 320, 240) に設定します。

```
-- Lingo syntax
on setDefaultRectSize(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRectMode = #fixed
            member(i, whichCast).defaultRect = rect(0,0,320,240)
        end if
    end repeat
end

// JavaScript syntax
function setDefaultRectSize(whichCast) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRectMode = symbol("fixed");
            member(i, whichCast).defaultRect = rect(0,0,320,240);
            i++;
        }
    }
}
```

関連事項

flashRect, defaultRect

density

使用法

```
member(whichCastmember).shader(whichShader).density
member(whichCastmember).model(whichModel).shader.density
member(whichCastmember).model(whichModel).shaderList{[index]}.density
```

解説

3D #engraver および #newsprint シェーダプロパティ。これらの特殊なシェーダの効果を生成するために使用する、線またはドットの数を調整します。値を大きくすると、線またはドットが多くなります。

#engraver シェーダでは、density プロパティは画像を作成するために使用する線の数を調整します。値の範囲は 0 ～ 100 で、デフォルト値は 40 です。

#newsprint シェーダでは、density このプロパティは画像を作成するために使用するドットの数を調整します。値は 0 ～ 100 で、デフォルト値は 45 です。

例文

次のステートメントは、"EngShader" というシェーダの density プロパティを 10 に設定します。この #engraver シェーダで画像を作成するために使用される線は、粗く疎になります。

```
-- Lingo syntax
member("scene").shader("EngShader").density = 10
// JavaScript syntax
member("scene").getProp("shader",1).density = 10;
```

次のステートメントは、シェーダ "gbShader" の density プロパティを 100 に設定します。この #newsprint シェーダで画像を作成するために使用されるドットは、非常に細かく密になります。

```
-- Lingo syntax
member("scene").shader("gbShader").density = 100
// JavaScript syntax
member("scene").getProp("shader", 2).density = 100;
```

関連事項

newShader

depth (3D)

使用法

```
member(whichCastmember).model(whichModel).sds.depth
```

解説

3D サブディビジョンサーフェス (sds) モディファイアプロパティ。sds モディファイアを使用するときに、モデルで表示可能な最大の解像度レベルを指定します。

sds モディファイアの error および tension 設定が低い場合、depth プロパティの値を増加させると、モデルのジオメトリへの効果がより明白になります。

sds モディファイアは、inker モディファイアや toon モディファイアと一緒に使用することはできません。また、sds モディファイアは、lod モディファイアと一緒に使用する際には注意する必要があります。

例文

次のステートメントは、" 赤ちゃん " モデルで sds モディファイアの depth プロパティを 3 に設定します。sds モディファイアの error および tension 設定が低い場合、" 赤ちゃん " のジオメトリへの効果は明らかになります。

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.depth = 3

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model", 2).getPropRef("sds").depth = 3;
```

関連事項

sds （モディファイア）, error, tension

depth （ビットマップ）

使用法

```
imageObject.depth
member(whichCastMember).depth
the depth of member whichCastMember
```

解説

指定した画像オブジェクトまたは、ビットマップキャストメンバの色深度を表示する画像オブジェクトまたはビットマップキャストメンバプロパティです。

Depth	カラー数
1	白黒
2	4 色
4, 8	16 または 256 パレット色またはグレイ色深度
16	数千色
32	数百万色

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、変数 newImage に格納されている画像オブジェクトの色深度を表示します。出力は、メッセージウィンドウに表示されます。

```
-- Lingo syntax
put(newImage.depth)

// JavaScript syntax
trace(newImage.depth);
```

次のステートメントは、メッセージウィンドウのキャストメンバ " 神社 " の色深度を表示します。

```
-- Lingo syntax
put(member("Shrine").depth)

// JavaScript syntax
put(member("Shrine").depth);
```

depthBufferDepth

使用法

```
getRendererServices().depthBufferDepth
```

解説

3D rendererServices プロパティ。ユーザのシステムの、ハードウェアの深度バッファの精度を示します。値はハードウェアの設定に応じて 16 または 24 のいずれかです。

例文

次のステートメントは、ユーザのビデオカードの depthBufferDepth 値が 16 であることを示します。

```
-- Lingo syntax
put getRendererServices().depthBufferDepth
-- 16

// JavaScript syntax
put (getRendererServices().depthBufferDepth);
// 16
```

関連事項

getRendererServices(), getHardwareInfo(), colorBufferDepth

deskTopRectList

使用法

```
-- Lingo syntax
_system.deskTopRectList

// JavaScript syntax
_system.deskTopRectList;
```

解説

コンピュータに接続されているモニタのサイズとデスクトップでの位置を示すシステムプロパティです。読み取り専用。

ウィンドウ、スプライト、ポップアップウィンドウなどのオブジェクトが完全にモニタ画面に収まるかどうかを調べるときに使うことができます。

結果として、個々のモニタの領域を示す rect 値のリストが返されます。rect (矩形) の座標値はすべてモニタ 1 の左上隅を原点 (0,0) とした値になります。リストの最初にはメインモニタの矩形の座標値が、次に 2 台目のモニタが使われているときは、その矩形の座標値が最初のモニタと同じ原点からの相対値で返されます。

例文

次のステートメントは、コンピュータに接続されているモニタのサイズを調べ、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_system.deskTopRectList)

// JavaScript syntax
put(_system.deskTopRectList);
```

次のハンドラは、現行のシステムに接続されているモニタの台数を返します。

```
-- Lingo syntax
on countMonitors
    return _system.deskTopRectList
end

// JavaScript syntax
function countMonitors() {
    return _system.deskTopRectList;
}
```

関連事項

システム

diffuse

使用法

```
member(whichCastmember).shader(whichShader).diffuse
member(whichCastmember).model(whichModel).shader.diffuse
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuse
```

解説

3D #standard シェーダプロパティ。次の最初の条件に一致し、さらに 2 番目または 3 番目の条件に一致する場合に、シェーダの最初のテクスチャにブレンドされるカラーを示します。

- シェーダの useDiffuseWithTexture プロパティが TRUE に設定されている。
- シェーダの blendFunction プロパティが #add または #multiply に設定されている。
- シェーダの blendFunction プロパティが #blend に設定され、blendSource プロパティが #constant に設定されている。さらに、シェーダの blendConstant プロパティの値が 100 未満である。

このプロパティのデフォルト値は rgb(255, 255, 255) です。

例文

次のステートメントは、" 天体 " というシェーダの diffuse プロパティを rgb(255, 0, 0) に設定します。

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").diffuse = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).diffuse = color(255, 0, 0);
```

関連事項

diffuseColor, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseColor

使用法

```
member(whichCastmember).diffuseColor
```

解説

3D キャストメンバプロパティ。次の最初の条件に一致し、さらに 2 番目または 3 番目の条件に一致する場合に、キャストメンバの最初のシェーダの最初のテクスチャにブレンドされるカラーを示します。

- シェーダの useDiffuseWithTexture プロパティが TRUE に設定されている。
- シェーダの blendFunction プロパティが #add または #multiply に設定されている。
- シェーダの blendFunction プロパティが #blend に設定され、blendSource プロパティが #constant に設定されている。さらに、シェーダの blendConstant プロパティの値が 100 未満である。

diffuseColor プロパティのデフォルト値は rgb(255, 255, 255) です。

例文

次のステートメントは、" 部屋 " というキャストメンバの diffuseColor プロパティを rgb(255, 0, 0) に設定します。

```
-- Lingo syntax
member("Room").diffuseColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").diffuseColor = color(255, 0, 0);
```

関連事項

diffuse, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseLightMap

使用法

```
member(whichCastmember).shader(whichShader).diffuseLightMap
member(whichCastmember).model(whichModel).shader.diffuseLightMap
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuseLightMap
```

解説

3D #standard シェーダプロパティ。拡散ライトマッピングに使用するテクスチャを指定します。

このプロパティを設定すると、次のプロパティが自動的に設定されます。

- シェーダの 2 番目のテクスチャレイヤーが、指定したテクスチャに設定されます。
- textureModeList[2] の値が #diffuse に設定されます。
- blendFunctionList[2] の値が #multiply に設定されます。
- blendFunctionList[1] の値が #replace に設定されます。

例文

次のステートメントは、" 楕円 " というテクスチャを、"GlassBox" というモデルで使用されるシェーダの diffuseLightMap プロパティとして設定します。

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.diffuseLightMap = member("3DPlanet").texture("Oval")

// JavaScript syntax
member("3DPlanet").getProp("model", 1).shaderList[1].diffuseLightMap =
member("3DPlanet").getprop("texture", 1);
```

関連事項

blendFunctionList, textureModeList, glossMap, region, specularLightMap

digitalVideoTimeScale

使用法

```
-- Lingo syntax
_player.digitalVideoTimeScale

// JavaScript syntax
_player.digitalVideoTimeScale;
```

解説

システムがデジタルビデオキャストメンバをトラッキングするために使用するタイムスケール（秒単位）を定義するプレーヤプロパティです。読み取り / 書き込み可能。

digitalVideoTimeScale プロパティには、任意の値を設定できます。

このプロパティの値によって、ビデオのトラッキングに使用される秒単位の分母が決められます。例を以下に示します。

- 100 ⊠ タイムスケールの単位は 1/100 秒（1 秒あたりムービーの 100 ユニットをトラッキング）
- 500 ⊠ タイムスケールの単位は 1/500 秒（1 秒あたりムービーの 500 ユニットをトラッキング）
- 0 ⊠ 現在再生中のムービーに使われているタイムスケールを適用

システムのタイムスケール単位をデジタルビデオのタイムスケール単位の倍数にするように digitalVideoTimeScale を設定すると、正確にビデオキャストメンバをトラッキングできます。ビデオの再生を詳細に制御するときは、digitalVideoTimeScale プロパティに大きな値を設定します。

例文

次のステートメントは、システムがデジタルビデオを測定するタイムスケールを 1 秒間に 600 ユニットに設定します。

```
-- Lingo syntax
_player.digitalVideoTimeScale = 600

// JavaScript syntax
_player.digitalVideoTimeScale = 600;
```

関連事項

プレーヤ

digitalVideoType

使用法

```
member(whichCastMember).digitalVideoType
the digitalVideoType of member whichCastMember
```

解説

指定したデジタルビデオのフォーマットを示すキャストメンバプロパティです。設定値は、#quickTime または #videoForWindows です。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、キャストメンバ " 本日のイベント " が QuickTime であるのか AVI（Audio-Video Interleaved）デジタルビデオであるのかを調べて、その結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put member("Today's Events").digitalVideoType

// JavaScript syntax
put ( member("Today's Events").digitalVideoType );
```

関連事項

QuickTimeVersion()

direction

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.direction
```

解説

3D エミッタプロパティ。パーティクルシステムのパーティクルが放出される向きを示すベクトルです。パーティクルシステムは、#particle というタイプのモデルリソースです。

パーティクル放出の基本的な方向は、エミッタの direction プロパティで設定されたベクトルです。ただし、パーティクルの放出方向は、3 からエミッタの angle (3D) プロパティ値の間のランダムな角度で、このベクトルからは外れます。

direction を vector(0,0,0) に設定すると、パーティクルはすべての方向に放出されます。

このプロパティのデフォルト値は vector(1,0,0) です。

例文

次の例で、"ThermoSystem" は #particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" のエミッタの direction プロパティを vector(1, 0, 0) に設定します。"ThermoSystem" のパーティクルは、3D ワールドの X 軸を軸とする円錐形の領域に放出されます。

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.direction = vector(1,0,0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").direction = vector(1,0,0);
```

関連事項

emitter, angle (3D)

directionalColor

使用法

```
member(whichCastmember).directionalColor
```

解説

3D キャストメンバプロパティ。キャストメンバの、デフォルトの指向性ライトの RGB カラーを示します。

このプロパティのデフォルト値は rgb(255, 255, 255) です。

例文

次のステートメントは、" 部屋 " というキャストメンバの directionalColor プロパティを rgb(0, 255, 0) に設定します。キャストメンバのデフォルトの指向性ライトは緑になります。このプロパティは、プロパティインスペクタでも設定できます。

```
-- Lingo syntax
member("Room").directionalcolor = rgb(0, 255, 0)

// JavaScript syntax
member("Room").directionalcolor = color(0, 255, 0);
```

関連事項

directionalPreset

directionalPreset

使用法

```
member(whichCastmember).directionalPreset
```

解説

3D キャストメンバプロパティ。デフォルトの指向性ライトが照らす方向を、スプライトのカメラを基準に示します。

このプロパティの値を変更すると、ライトのトランスフォームの position および rotation プロパティが変更されます。

directionalPreset に設定できる値は、以下のとおりです。

- #topLeft
- #topCenter
- #topRight
- #middleLeft
- #middleCenter
- #middleRight
- #bottomLeft
- #bottomCenter
- #bottomRight
- #None

このプロパティのデフォルト値は #topCenter です。

例文

次のステートメントは、" 部屋 " というキャストメンバの directionalPreset プロパティを #middleCenter に設定します。これにより、" 部屋 " のデフォルトライトで、スプライトのカメラの現在のビューの中央を照らします。このプロパティは、プロパティインスペクタでも設定できます。

```
-- Lingo syntax
member("Room").directionalpreset = #middleCenter

// JavaScript syntax
member("Room").directionalpreset = symbol("middleCenter");
```

関連事項
directionalColor

directToStage

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.directToStage

// JavaScript syntax
memberOrSpriteObjRef.directToStage;
```

解説

デジタルビデオ、アニメーション GIF、ベクトルシェイプ、3D、Windows Media または Flash Asset の各キャストメンバを再生するレイヤーを決めるキャストメンバおよびスプライトプロパティです。

このプロパティが 1（TRUE）に設定されている場合は、ステージの最前面でキャストメンバが再生され、インク効果は適用されません。

このプロパティが 0（FALSE）に設定されている場合は、ステージの任意のアニメーションレイヤーにキャストメンバを表示でき、スプライトの外観にインク効果が適用されます。

- デジタルビデオまたはアニメーション GIF には、member(**whichCastMember**).directToStage というシンタックを使用します。
- Flash またはベクトルシェイプの場合は、sprite(**whichSprite**).directToStage というシンタックスを使います。
- 3D キャストメンバには、どちらかのシンタックスを使います。

このプロパティを使用すると、キャストメンバまたはスプライトを再生するときのパフォーマンスが向上します。

directToStage スプライトが必ず最前面に表示されます。また、directToStage スプライトには、インク効果は適用されません。

スプライトの directToStage プロパティが TRUE の場合、Director はオフスクリーンのバッファで合成せずに、スプライトを直接画面に書き込みます。結果は、ステージの trails インク効果のようになります。

軌跡の残っている部分をリフレッシュするには、directToStage プロパティをオフにしてからオンにするか、フルスクリーンのトランザクションを実行するか、別のスプライトで領域を " ワイプ " します。(Windows では、ビデオを完全に消去せずに別の類似画面にジャンプすることもできます。)

完成したムービーの directToStage の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "lingo QT and Flash.dir" を参照してください。

例文

次のステートメントは、QuickTime ムービー "The Residents" を常に再前面で再生します。

```
-- Lingo syntax
member("The Residents").directToStage = 1

// JavaScript syntax
member("The Residents").directToStage = 1;
```

directToStage (MP4Media/FLV)

使用法

```
member(1).directToStage = true
put sprite(1).directToStage
```

解説

キャストメンバおよびスプライトのプロパティ。ステージ上で MP4Media/FLV キャストメンバが再生するレイヤーを指定します。このプロパティは読み書き可能です。

directToStage が 1（True）の場合、キャストメンバはステージの最前面で再生されます。この場合、インク効果はスプライトの外観に影響を与えません。

このプロパティが 0 （False）の場合は、ステージの任意のアニメーションレイヤーにキャストメンバを表示できます。この場合、インク効果がスプライトの外観に適用されます。

例文

この構文は、キャストメンバ MP4Media/FLV をステージの最上位レイヤーで常に再生します。

```
-- Lingo syntax
member("MP4Media/FLV").directToStage = TRUE
put sprite("MP4Media/FLV").directToStage   -- 1

// JavaScript syntax
member("MP4Media/FLV").directToStage = true;
put sprite("MP4Media/FLV").directToStage   // 1
```

disableImagingTransformation

使用法

```
-- Lingo syntax
_player.disableImagingTransformation

// JavaScript syntax
_player.disableImagingTransformation;
```

解説

ステージの画像をキャプチャするときに、ステージのスクロール操作やズーム率を自動的に調整するかどうかを指定するプレーヤプロパティです。読み取り / 書き込み可能。

TRUE の場合、このプロパティにより、ステージの画像を取得するために image プロパティを使用するときに、ステージのスクロール操作やズーム率を自動的に調整しないようにします。ステージのズーム率とスクロール操作は、image を使用してキャプチャした画像の外観に影響します。

FALSE の場合、ステージの画像は常に、ステージウィンドウのズーム率が 100% でステージウィンドウの中央からスクロールアウトしないようにキャプチャされます。FALSE がデフォルト値です。

例文

次のステートメントは、disableImagingTransformation を TRUE に設定します。

```
-- Lingo syntax
_player.disableImagingTransformation = TRUE

// JavaScript syntax
_player.disableImagingTransformation = true;
```

関連事項

image （画像）, プレーヤ

displayFace

使用法

```
member(whichTextCastmember).displayFace
member(which3DCastmember).modelResource(whichModelResource).displayFace
```

解説

3D テキストプロパティ。表示する 3D テキストのフェイスを示すリニアリストです。使用できる値は、#front、#tunnel、および #back です。フェイスの任意の組み合わせを表示できます。リストの順序は任意です。

このプロパティのデフォルト値は [#front, #back, #tunnel] です。

テキストキャストメンバの場合、これはメンバプロパティです。3D キャストメンバの押し出しテキストの場合、これはモデルリソースプロパティです。

例文

次の例では、"Rugsign" というキャストメンバはテキストキャストメンバです。次のステートメントは、"Rugsign" の displayFace プロパティを [#tunnel] に設定します。"Rugsign" が 3D モードで表示される場合、表面と裏面は表示されません。

```
-- Lingo syntax
member("Rugsign").displayFace = [#tunnel]

// JavaScript syntax
member("Rugsign").displayFace = list( symbol("tunnel") );
```

次の例では、モデル " スローガン " のモデルリソースは押し出しテキストです。次のステートメントは、" スローガン " のモデルリソースの displayFace プロパティを [#back, #tunnel] に設定します。" スローガン " の表面は描画されません。

```
-- Lingo syntax
member("scene").model("Slogan").resource.displayFace = [#back, #tunnel]
```

関連事項

extrude3D, displayMode

displayMode

使用法

```
member(whichTextCastmember).displayMode
```

解説

テキストキャストメンバプロパティ。テキストを 2D テキストまたは 3D テキストのどちらでレンダリングするかを指定します。

このプロパティを #Mode3D に設定すると、テキストは 3D で表示されます。テキストの 3D プロパティ (displayFace や bevelDepth) と、通常のテキストプロパティ (text や font) を設定できます。このキャストメンバを含むスプライトは 3D スプライトになります。

このプロパティを #ModeNormal に設定すると、テキストは 2D で表示されます。

このプロパティのデフォルト値は #ModeNormal です。

例文

次の例で、" ロゴ " というキャストメンバはテキストキャストメンバです。次のステートメントは、" ロゴ " を 3D で表示します。

```
-- Lingo syntax
member("Logo").displayMode = #mode3D

// JavaScript syntax
member("Logo").displayMode = symbol("mode3D");
```

関連事項

extrude3D

displayRealLogo

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.displayRealLogo

// JavaScript syntax
memberOrSpriteObjRef.displayRealLogo;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。RealNetworks® ロゴが表示される（TRUE）か、表示されない（FALSE）かを取得または設定できます。このプロパティを TRUE に設定すると、ビデオを停止または巻き戻したときに、ストリーミングの先頭で RealMedia ビューアに RealNetworks のロゴが表示されます。

このプロパティのデフォルト値は TRUE (1) です。1 または 0 以外の整数値は、TRUE として扱われます。

例文

次の例は、スプライト "2" とキャストメンバ "Real" の displayRealLogo プロパティが TRUE に設定されていることを示します。この場合、ムービーの再生が開始されるとき、および停止または巻き戻しをしたときに、RealNetworks ロゴが表示されます。

```
-- Lingo syntax
put(sprite(2).displayRealLogo) -- 1
put(member("Real").displayRealLogo) -- 1

// JavaScript syntax
trace(sprite(2).displayRealLogo); // 1
put(member("Real").displayRealLogo); // 1
```

次の例は、スプライト "2" とキャストメンバ "Real" の displayRealLogo プロパティを FALSE に設定します。これによって、RealNetworks ロゴは表示されなくなります。

```
-- Lingo syntax
sprite(2).displayRealLogo = FALSE
member("Real").displayRealLogo = FALSE

// JavaScript syntax
sprite(2).displayRealLogo = 0;
member("Real").displayRealLogo = 0;
```

displayTemplate

使用法

```
-- Lingo syntax
_movie.displayTemplate

// JavaScript syntax
_movie.displayTemplate;
```

解説

ムービーを再生中のウィンドウに適用されているプロパティのリストにアクセスできるようにするムービープロパティです。読み取り / 書き込み可能。

displayTemplate プロパティは、デフォルトのウィンドウ設定を指定するために使用される Window オブジェクトのプロパティにアクセスできるようにします。そのため、appearanceOptions および titlebarOptions プロパティを Window オブジェクトで使用するのと同じ方法で、displayTemplate を Movie オブジェクトで使用してデフォルトのウィンドウ設定を返すか設定します。

displayTemplate プロパティは次のプロパティにアクセスできるようにします。

プロパティ	解説
appearanceOptions	ウィンドウの外観オプションを保存するプロパティリストです。外観オプションには、mask、border、metal、dragRegionMask、shadow および liveresize があります。詳細については、「appearanceOptions」を参照してください。
dockingEnabled	ウィンドウ内のムービー（MIAW）が、オーサリング中に開いたときにドッキング可能かどうかを指定します。TRUE の場合、ウィンドウはドッキングできます。FALSE の場合、ウィンドウはドッキングできません。デフォルト値は FALSE です。詳細については、「dockingEnabled」を参照してください。
resizable	ウィンドウのサイズ変更が可能かどうかを指定します。TRUE の場合、ウィンドウはサイズ変更が可能です。FALSE の場合、ウィンドウはサイズ変更が可能ではありません。デフォルト値は TRUE です。詳細については、「resizable」を参照してください。
title	テンプレートの表示のタイトルを返すか設定します。詳細については、「title」を参照してください。
titlebarOptions	ウィンドウのタイトルバーを保存するプロパティリストです。タイトルバーのオプションには、icon、visible、closebox、minimizebox、maximizebox および sideTitlebar があります。詳細については、「titlebarOptions」を参照してください。
systemTrayIcon	（Microsoft Windows のみ）ウィンドウには、ユーザのデスクトップのシステムトレイに関連するアイコンがあるかどうかを指定します。
systemTrayTooltip	（Microsoft Windows のみ）システムトレイアイコンのヒントポップアップに表示される文字列を指定します。
type	ウィンドウのタイプを返すか設定します。ウィンドウのタイプを設定すると、そのウィンドウタイプに関連するすべてのプロパティも設定されます。ウィンドウのタイプには、tool、document、dialog があります。詳細については、「type」を参照してください。

例文

次のステートメントは、displayTemplate のプロパティと対応する値をメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(_movie.displayTemplate)

// JavaScript syntax
trace(_movie.displayTemplate);
```

次のステートメントは、displayTemplate の様々なプロパティを設定します。

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE
_movie.displayTemplate.resizable = FALSE
_movie.displayTemplate.appearanceOptions.mask = member("mask")
_movie.displayTemplate.titlebarOptions.sideTitlebar = TRUE

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true;
_movie.displayTemplate.resizable = false;
_movie.displayTemplate.appearanceOptions.mask = member("mask");
_movie.displayTemplate.titlebarOptions.sideTitlebar = true;
```

関連事項

appearanceOptions, dockingEnabled, ムービー , resizable, systemTrayIcon, title （ウィンドウ）, titlebarOptions, type （ウィンドウ）, ウィンドウ

distribution

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
```

解説

3D エミッタプロパティ。パーティクルシステムのパーティクルを作成するときに、エミッタの領域にどのように配布するかを示します。このプロパティで使用できる値は #gaussian または #linear です。デフォルト値は #linear です。

例文

次の例で、"ThermoSystem" は #particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" のエミッタの distribution プロパティを #linear に設定します。"ThermoSystem" のパーティクルは、生成時に発生領域に均一に配布されます。

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.distribution = #linear

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").distribution = symbol("linear");
```

関連事項

emitter, region

dither

使用法

```
-- Lingo syntax
memberObjRef.dither

// JavaScript syntax
memberObjRef.dither;
```

解説

8 ビット以下 (256 色) の色深度でキャストメンバを表示するときに、そのキャストメンバをディザリングするビットマップキャストメンバプロパティです。TRUE を設定すると、キャストメンバにはない色のグラデーションが表示され、FALSE を設定すると、現在のパレットで利用できる最も近い色が選択されます。

パフォーマンスと画質の両方の理由から、より高度な画質が必要な場合にのみ、dither を TRUE に設定します。ディザの速度はリマップより遅く、ディザをかけられた画像をアニメーションにすると画像が粗くなることがあります。

色深度が 8 ビットより大きい場合は、このプロパティは効力を持ちません。

関連事項

depth（ビットマップ）

dockingEnabled

使用法

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled
windowObjRef.dockingEnabled

// JavaScript syntax
_movie.displayTemplate.dockingEnabled;
windowObjRef.dockingEnabled;
```

解説

ウィンドウ内のムービー（MIAW）が、オーサリング中に開いたときにドッキング可能かどうかを指定するムービープロパティおよびウィンドウプロパティです。読み取り / 書き込み可能。

このプロパティは、Movie オブジェクトから直接アクセスすることはできません。Movie オブジェクトの displayTemplate プロパティからアクセスします。

このプロパティのデフォルト値は FALSE です。この場合、オーサリング中に開いたときに MIAW はドッキング可能でないことを指定します。このプロパティを TRUE に設定した場合、Window オブジェクトの type プロパティにより、オーサリング中にウィンドウがどのように表示されるかが決まります。

- dockingEnabled が TRUE に設定され、type が #document に設定されている場合、MIAW は Director のドキュメントウィンドウのように表示され、機能します。ウィンドウは「ビューポート」領域に表示され、ステージウィンドウ、スコアウィンドウ、キャストウィンドウ、メディアエディタ、メッセージウィンドウとドッキングできます。ただし、ウィンドウをこれらのウィンドウのいずれかとグループ化することはできません。
- dockingEnabled が TRUE で、type が #tool に設定されている場合、MIAW は Director でツールウィンドウのような外観と動作になります。ウィンドウは、プロパティインスペクタおよびツールパレットを除くすべてのツールウィンドウとグループ化できます。

- dockingEnabled が TRUE に設定され、type が #dialog に設定されている場合、タイプは無視され、ウィンドウはオーサリングウィンドウになります。

このプロパティはプロジェクタで無視されます。

例文

次のステートメントは、dockingEnabled プロパティを TRUE に設定します。

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE -- from the Movie object
window("Instructions").dockingEnabled = TRUE -- from the Window object

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true; // from the Movie object
window("Instructions").dockingEnabled = true; // from the Window object
```

関連事項

appearanceOptions, displayTemplate, ムービー , titlebarOptions, type （ウィンドウ）, ウィンドウ

domain

使用法

```
-- Lingo syntax
dvdObjRef.domain

// JavaScript syntax
dvdObjRef.domain;
```

解説

現在のドメインを示すシンボルを返す DVD プロパティです。読み取り専用。

例文

次のステートメントは、現在のドメインを返します。

```
-- Lingo syntax
trace (member(1).domain)-- #title

// JavaScript syntax
trace (member(1).domain);// #title
```

関連事項

DVD

doubleClick

使用法

```
-- Lingo syntax
_mouse.doubleClick

// JavaScript syntax
_mouse.doubleClick;
```

解説

マウスのボタンのクリックがダブルクリックだったかどうかを返すマウスプロパティです。マウスのボタンが 2 回シングルクリックされたのではなく、時間内にダブルクリックされたと認識された場合、TRUE を返します。マウスのボタンが時間内に 2 回シングルクリックされたと認識された場合に、FALSE を返します。読み取り専用。

例文

次のステートメントは、ユーザがマウスのボタンをダブルクリックしたときに再生ヘッドを "Enter Bid" フレームに移動させます。

```
-- Lingo syntax
if (_mouse.doubleClick) then
    _movie.go("Enter Bid")
end if

// JavaScript syntax
if (_mouse.doubleClick) {
    _movie.go("Enter Bid");
}
```

関連事項

clickLoc, clickOn, マウス

drag

使用法

```
member(whichCastmember).modelResource(whichModelResource).drag
```

解説

3D #particle モデルリソースプロパティ。シミュレーションステップごとに失われる、各パーティクルの速度のパーセンテージを示します。このプロパティの範囲は 0 (減速なし) ～ 100 (パーティクルが停止するまで減速) です。デフォルト値は 0 です。

例文

次の例で、"ThermoSystem" は #particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の drag プロパティを 5 に設定しています。"ThermoSystem" のパーティクルのモーションに抵抗を適用して、遠くまで移動しないようにします。

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").drag = 5

// JavaScript syntax
member("Fires").getProp("modelResource", 1).drag = 5;
```

関連事項

wind, gravity

drawRect

使用法

```
-- Lingo syntax
windowObjRef.drawRect

// JavaScript syntax
windowObjRef.drawRect;
```

解説

ウィンドウ内で再生されるムービーの表示領域（ステージの矩形座標）を定義するウィンドウプロパティです。読み取り / 書き込み可能。

座標の値は rect 値で、左、上、右、下の順に示されます。

drawRect of window は、ムービーをパンまたは拡大縮小するときなどに便利なプロパティです。ただし、フィールドキャストメンバとテキストキャストメンバ内の文字のサイズには影響しません。また、ビットマップのサイズ変更は処理速度を低下させます。

例文

次のステートメントは、" 制御パネル " というタイトルのムービーウィンドウの現在の表示領域の座標を求めます。

```
-- Lingo syntax
put(window("Control Panel").drawRect)

// JavaScript syntax
put(window("Control Panel").drawRect);
```

次のステートメントは、ムービーの rect 値を movieRectangle という rect 値に設定します。矩形内のムービーの一部は、ウィンドウに表示されます。

```
-- Lingo syntax
movieRectangle = rect(10, 20, 200, 300)
window("Control Panel").drawRect = movieRectangle

// JavaScript syntax
var movieRectangle = rect(10, 20, 200, 300);
window("Control Panel").drawRect = movieRectangle;
```

次の行は、メインモニタ領域に再生されるステージを作成します。

```
-- Lingo syntax
_movie.stage.drawRect = _system.deskTopRectList[1]
_movie.stage.rect = _system.deskTopRectList[1]

// JavaScript syntax
_movie.stage.drawRect = _system.deskTopRectList[1];
_movie.stage.rect = _system.deskTopRectList[1];
```

関連事項

rect(), ウィンドウ

dropShadow

使用法

```
-- Lingo syntax
memberObjRef.dropShadow

// JavaScript syntax
memberObjRef.dropShadow;
```

解説

フィールドキャストメンバ内の文字の影のサイズを定義するキャストメンバプロパティです。単位はピクセルです。

例文

次のステートメントは、" コメント " フィールドキャストメンバ内の文字の影を 5 ピクセルに設定します。

```
--Lingo syntax
member("Comment").dropShadow = 5

// JavaScript syntax
member("Comment").dropShadow = 5;
```

duration (3D)

使用法

```
member(whichCastmember).motion(whichMotion).duration
motionObjectReference.duration
```

解説

3D プロパティ。モーションパラメータで指定したモーションの、再生が完了するまでにかかる時間をミリ秒単位で取得します。このプロパティは常に 0 以上です。

例文

次のステートメントは、モーション " 蹴る " の長さをミリ秒単位で示します。

```
-- Lingo syntax
put member("GbMember").motion("Kick").duration
-- 5100.0000

// JavaScript syntax
put( member("GbMember").getProp("motion", 1).duration );
// 5100.0000
```

関連事項

motion, currentTime (3D), play() (3D), queue() (3D)

duration (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.duration

// JavaScript syntax
dvdObjRef.duration;
```

解説

タイトルの合計時間をミリ秒単位で返す DVD プロパティです。読み取り専用。

例文

次のステートメントは、現行タイトルの継続時間を返します。

```
--Lingo syntax
trace (member(1).duration)-- 1329566

// JavaScript syntax
trace (member(1).duration);// 1329566
```

関連事項

DVD

duration （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.duration

// JavaScript syntax
memberObjRef.duration;
```

解説

指定した SWA（Shockwave Audio）、トランジション、Windows Media または QuickTime キャストメンバの継続時間を示すキャストメンバプロパティです。

- [キャストメンバ] がストリーミングサウンドファイルの場合、このプロパティはサウンドの継続時間を示します。ストリーミングが開始されるまでの間は、duration プロパティは 0 を返します。preLoadTime を 1 秒に設定すると、ビットレートは実際の継続時間を返します。
- キャストメンバがデジタルビデオキャストメンバの場合、このプロパティはデジタルビデオの継続時間を示します。値はチック数で指定します。
- キャストメンバがトランジションキャストメンバの場合、このプロパティはトランジションの継続時間を示します。値の単位は、ミリ秒です。再生中には、この設定は [フレームトランジション] ダイアログボックスの [速度] 設定と同じ効果になります。

このプロパティは値を調べることはできますが、変更できるのはトランジションキャストメンバの場合のみです。

完成したムービーの duration の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "lingo QT and Flash.dir" を参照してください。

例文

SWA キャストメンバ " ルイ・プリマ " が事前に読み込まれている場合、次のステートメントにより、サウンドの長さがフィールドキャストメンバ " 演奏時間 " に表示されます。

```
-- Lingo syntax
on exitFrame
    if member("Louie Prima").state = 2 then
        member("Duration Displayer").text = string(member("Louie Prima").duration)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (member("Louie Prima").state == 2) {
        member("Duration Displayer").text = member("Louie Prima").duration.toString()
    }
}
```

duration (MP4Media/FLV)

使用法

```
put member(1).duration
```

解説

MP4Media/FLV スプライトまたはキャストメンバのプロパティ。MP4Media/FLV キャストメンバの継続時間を秒単位で返します。このプロパティは読み取り専用です。取得できますが設定はできません。

例文

次の例では、スプライト 2 の キャストメンバ MP4Media/FLV の継続時間が 100.009 秒と表示されます。

```
-- Lingo syntax
put(sprite(2).duration) - 100.009
put(member("MP4Media/FLV ").duration) - 100.009

// JavaScript syntax
put(sprite(2).duration); // 100.009
put(member("MP4Media/FLV ").duration); // 100.009
```

duration (RealMedia, SWA)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.duration

// JavaScript syntax
memberOrSpriteObjRef.duration;
```

解説

RealMedia または Shockwave オーディオスプライトまたはキャストメンバプロパティ。RealMedia または Shockwave Audio ストリーミングの継続時間をミリ秒で返します。ストリーミングの長さは、キャストメンバの再生が始まらない限りわかりません。ライブフィードからのストリーミングの場合、またはストリーミングが再生されていない場合は、このプロパティの値は 0 になります。このプロパティの値は調べることはできますが、設定できません。

例文

次の例は、スプライト "2" およびキャストメンバ "Real" の RealMedia ストリーミングの継続時間が 100,500 ミリ秒（100.500 秒）であることを示します。

```
-- Lingo syntax
put(sprite(2).duration) -- 100500
put(member("Real").duration) -- 100500

// JavaScript syntax
put(sprite(2).duration); // 100500
put(member("Real").duration); // 100500
```

関連事項

play() (RealMedia, SWA, Windows Media), seek(), currentTime (RealMedia)

editable

使用法

```
-- Lingo syntax
spriteObjRef.editable

// JavaScript syntax
spriteObjRef.editable;
```

解説

指定したスプライトがステージ上で編集可能（TRUE）か、そうでないか（FALSE）を定義するスプライトプロパティです。読み取り / 書き込み可能。

キャストメンバプロパティが設定されている場合は、指定したフィールドを含むすべてのスプライトに適用されます。

このプロパティが設定されている場合は、指定したスプライトのみに適用されます。

フィールドスプライトは、フィールドキャストメンバプロパティダイアログの「編集可能」オプションを使用しても編集可能になります。

フィールドスプライトは、スコアの [編集可能] オプションを使って編集可能にすることができます。

スクリプトで設定した値を現行スプライトの終了後も維持するには、スプライトをスクリプトスプライトにしておく必要があります。

例文

次のハンドラは、まず、スプライトチャネルをパペットにします。次に、フィールドスプライトを編集可能にします。

```
-- Lingo syntax
on myNotes
    _movie.puppetSprite(5, TRUE)
    sprite(5).editable = TRUE
end

// JavaScript syntax
function myNotes() {
    _movie.puppetSprite(5, true);
    sprite(5).editable = true;
}
```

次のステートメントは、フィールドスプライトが編集可能かどうかを調べ、編集可能な場合はメッセージを表示します。

```
-- Lingo syntax
if (sprite(13).editable = TRUE) then
    member("Notice").text = "Please enter your answer below."
end if

// JavaScript syntax
if (sprite(13).editable == true) {
    member("Notice").text = "Please enter your answer below.";
}
```

関連事項

スプライト

editShortCutsEnabled

使用法

```
-- Lingo syntax
_movie.editShortCutsEnabled

// JavaScript syntax
_movie.editShortCutsEnabled;
```

解説

現行ムービーでのカット、コピー、ペースト操作 (およびキーボードショートカット) を有効にするかどうかを定義するムービープロパティです。読み取り / 書き込み可能。

TRUE を設定すると、これらの操作を実行できます。FALSE を設定すると、これらの操作を実行できません。Director 8 以上で作成されたムービーでは、このプロパティのデフォルトは TRUE です。8 以前のバージョンの Director で作成されたムービーでは、このプロパティのデフォルトは FALSE です。

例文

次のステートメントは、カット、コピー、ペースト操作を無効にします。

```
-- Lingo syntax
_movie.editShortCutsEnabled = 0

// JavaScript syntax
_movie.editShortCutsEnabled = 0;
```

関連事項

ムービー

elapsedTime

使用法

```
-- Lingo syntax
soundChannelObjRef.elapsedTime

// JavaScript syntax
soundChannelObjRef.elapsedTime;
```

解説

チャネル番号で現行のサウンドメンバが演奏されている時間（ミリ秒単位）を返す、サウンドチャネルプロパティです。読み取り専用。

経過時間は、演奏が開始されたときに 0 からスタートし、ループ、currentTime の設定や他のどのような操作にも影響されずに、演奏が続く限り増え続けます。サウンド中の現在の絶対時間を得るには、currentTime を使用してください。

このプロパティの値は浮動小数点値になっており、サウンドの演奏をミリ秒未満の単位まで測定できます。

例文

次の idle ハンドラは、アイドル中ステージ上のフィールド "time" にチャネル 4 のサウンドの演奏時間を表示します。

```
-- Lingo syntax
on idle
    member("time").text = string(sound(4).elapsedTime)
end idle

// JavaScript syntax
function idle() {
    member("time").text = sound(4).elapsedTime.toString();
}
```

関連事項

currentTime（スプライト）, サウンドチャネル

elapsedTime（ミクサ）

使用法

```
mixerRef.elapsedTime (Read-only)
```

解説

現在のミクサが再生した時間をミリ秒単位で返すミクサプロパティです。elapsedTime は読み取り専用のプロパティです。

例文

```
--Lingo syntax
on mouseUp me
put mixerRef.elapsedTime -- Gives the time, in milliseconds, that the
-- current mixer object has played for.
end

// JavaScript syntax
function mouseUp(){
put (mixerRef.elapsedTime); // Gives the time, in milliseconds, that the current
// mixer object has played for.
}
```

関連事項

ミクサ

elapsedTime （サウンドオブジェクト）

使用法

```
soundObject.elapsedTime (Read-only)
```

解説

現在のサウンドオブジェクトが再生した時間をミリ秒単位で返すサウンドオブジェクトプロパティです。elapsedTime は読み取り専用のプロパティです。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.elapsedTime -- Gives the time, in milliseconds, that the
-- current sound object has played for.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.elapsedTime) ; // Gives the time, in milliseconds, that the current
// sound object has played for.
}
```

emissive

使用法

```
member(whichCastmember).shader(whichShader).emissive
member(whichCastmember).model(whichModel).shader.emissive
member(whichCastmember).model(whichModel).shaderList{[index]}.emissive
```

解説

3D #standard シェーダプロパティです。シーン内のライティングのシェーダごとにライトを追加します。例えば、emissive プロパティが rgb(255, 255, 255) に設定されているシェーダを使用するモデルは、シーンにライトがない場合でも、白のライトで照らされたように表示されます。ただし、そのモデルが他のモデルを照らしたり、シーンに光を放射することはありません。

このプロパティのデフォルト値は rgb(0, 0, 0) です。

例文

次のステートメントは、" 天体 " というシェーダの emissive プロパティを rgb(255, 0, 0) に設定します。このシェーダを使用するモデルは、赤いライトで照らされたように表示されます。

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").emissive = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).emissive = color(255, 0, 0);
```

関連事項

silhouettes

emitter

使用法

```
--Lingo Usage
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
member(whichCastmember).modelResource(whichModelResource).emitter.mode
member(whichCastmember).modelResource(whichModelResource).emitter.loop
member(whichCastmember).modelResource(whichModelResource).emitter.direction
member(whichCastmember).modelResource(whichModelResource).emitter.region
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
member(whichCastmember).modelResource(whichModelResource).emitter.angle
member(whichCastmember).modelResource(whichModelResource).emitter.path
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed

// JavaScript usage
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").numPart
icles
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").mode
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").loop
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").directi
on
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").region
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").distrib
ution
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").angle
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").path
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").pathStr
ength
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").minSpee
d
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emitter").maxSpee
d
```

解説

3D パーティクルシステムエレメント。#particle というタイプのモデルリソースの、パーティクルの初期推進力を制御します。

エミッタプロパティのリストについては、「関連事項」を参照してください。詳細については、各プロパティのエントリを参照してください。

関連事項

numParticles, loop (emitter), direction, distribution, region, angle (3D), path (3D), pathStrength, minSpeed, maxSpeed

emulateMultibuttonMouse

使用法

```
-- Lingo syntax
_player.emulateMultibuttonMouse

// JavaScript syntax
_player.emulateMultibuttonMouse;
```

解説

Macintosh で Control キーを押しながらクリックする操作を、Windows でマウスの右ボタンをクリックする操作と同等に扱うかどうかを定義するプレーヤプロパティです。TRUE の場合、これらの操作は同等になります。FALSE（デフォルト）の場合、これらの操作は別の操作として扱われます。読み取り / 書き込み可能。

Macintosh には、Windows での右クリックに相当する操作がありません。

このプロパティを TRUE に設定すると、クロスプラットフォームで再生するムービーのマウスボタン対応に一貫性を持たせることができます。

例文

次のステートメントは、emulateMultibuttonMouse プロパティを TRUE に設定します。

```
-- Lingo syntax
_player.emulateMultibuttonMouse = TRUE

// JavaScript syntax
_player.emulateMultibuttonMouse = true;
```

関連事項

プレーヤ

enabled

使用法

```
the enabled of menuItem whichItem of menu whichMenu
```

解説

[項目] によって指定されたメニュー項目をテキストで標準表示 (選択可能) にするか、淡色表示 (選択不可能) にするかを定義するメニュー項目プロパティです。TRUE, (デフォルト) のときは、メニュー項目が標準表示 (選択可能) になります。FALSE のときは、メニュー項目が淡色表示 (選択不可能) になります。

式 [項目] にはメニュー項目名またはメニュー項目番号を、式 [メニュー] にはメニュー名またはメニュー番号を指定します。

このプロパティは、値を調べたり、変更できます。

注意：メニューは、Shockwave Player では使用できません。

例文

次のハンドラは、指定されたメニュー内のすべての項目を選択可能または選択不可能にします。パラメータ theMenu は、メニュー項目を指定し、パラメータ Setting は、TRUE または FALSE を指定します。例えば、呼び出しステートメント ableMenu ("Special", FALSE) は、スペシャルメニュー内の全項目を選択不可能にします。

```
-- Lingo syntax
on ableMenu theMenu, vSetting
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the enabled of menuItem i of menu theMenu to vSetting
    end repeat
end ableMenu
// JavaScript syntax
function ableMenu (theMenu, vSetting){
    n = _menuBar.menu[theMenu].item.count;
    for( i = 1 ; i <= n ; i++)
        _menuBar.menu[theMenu].item[i].enabled = vSetting;
}
```

関連事項

name （メニュープロパティ）, number （メニュー）, checkMark, script, number （メニュー項目）

enabled (collision)

使用法

```
member(whichCastmember).model(whichModel).collision.enabled
```

解説

3D 衝突プロパティ。このプロパティを使用すると、モデルで衝突を検出するか (TRUE)、検出しないか (FALSE) を取得または設定できます。このプロパティを FALSE に設定すると、collision モディファイアを、モデルから削除することなく一時的に無効にできます。

このプロパティのデフォルト設定は TRUE です。

例文

次のステートメントは、モデル " ボックス " の collision モディファイアをアクティブにします。

```
-- Lingo syntax
member("3d world").model("box").collision.enabled = TRUE

// JavaScript syntax
member("3d world").getProp("model",1).getPropRef("collision").enabled = true;
```

関連事項

addModifier, collision (modifier), modifier

enabled（フィルタ）

使用法

```
audioFilter(#NameOfFilter, [#enabled:isEnabled])
```

解説

enabled プロパティはすべてのフィルタに使用できます。フィルタを有効または無効にするために使用できます。

プロパティ	解説	範囲	デフォルト
enabled：ブール値	フィルタを有効（enabled:true）または無効（enabled:false）にします。	True, false	True

例文

```
--Lingo Syntax
audioFilter(#EchoFilter, [#enabled:true])
//Javascript Syntax:
audioFilter(symbol("EchoFilter"),propList(symbol("enabled"),true));
```

enabled (fog)

使用法

```
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
```

解説

3D カメラプロパティ。カメラからのビューにフォグを追加するかどうかを示します。このプロパティのデフォルト設定は FALSE です。

例文

次のステートメントは、"BayView" というカメラからのビューにフォグを作成します。

```
member("MyYard").camera("BayView").fog.enabled = TRUE
```

関連事項

fog

enabled (sds)

使用法

```
member(whichCastmember).model(whichModel).sds.enabled
```

解説

3D sds モディファイアプロパティ。モデルに割り当てられた sds モディファイアが、モデルで使用されるかどうかを示します。

このプロパティのデフォルト設定は TRUE です。

既に inker または toon モディファイアが割り当てられているモデルに、sds モディファイアを追加しようとすると失敗します。エラーメッセージは表示されません。同様に、既に sds モディファイアが割り当てられているモデルに、inker または toon モディファイアを追加しようとしても失敗します。エラーメッセージは表示されません。sds モディファイアを lod モディファイアと共に使用する場合は注意してください。詳細については、sds (modifier) のエントリを参照してください。

例文

次のステートメントは、モデル " 赤ちゃん " の sds モディファイアを有効にします。

```
member("Scene").model("Baby").sds.enabled = TRUE
```

関連事項

sds （モディファイア）, modifier, addModifier

enableFlashLingo

使用法

```
-- Lingo syntax
_movie.enableFlashLingo

// JavaScript syntax
_movie.enableFlashLingo;
```

解説

Flash getURL() メソッドを使用中に、Flash コンテンツを含むスプライトが、直接的なスクリプトコールバックを行えるかどうかを定義するムービープロパティです。読み取り / 書き込み可能。

Flash getURL() メソッドは、新しい URL を空白のブラウザウィンドウにロードします。

enableFlashLingo が TRUE に設定されている場合、Flash コンテンツを含むスプライトは、getURL() が呼び出されたときに、（標準の Shockwave Player で有効なルールに従っている）有効な任意のスクリプトコマンドを実行できます。

enableFlashLingo を FALSE に設定すると、Flash コンテンツを含むスプライトは、getURL() が呼び出されたときにスクリプトコマンドを実行できなくなります。このプロパティのデフォルト値は FALSE です。

このプロパティが便利なのは、不明な場所からの Flash コンテンツを表示するムービーを作成する場合です。例えば、システムフォルダで SWF ファイルを参照するプロジェクタや、エンドユーザから SWF ファイルの URL を受け取る Shockwave コンテンツを含むムービーなどです。

例文

次のステートメントは enableFlashLingo プロパティを TRUE に設定します。

```
-- Lingo syntax
_movie.enableFlashLingo = TRUE

// JavaScript syntax
_movie.enableFlashLingo = true;
```

関連事項

ムービー

endAngle

使用法

```
member(whichCastmember).modelResource(whichModelResource).endAngle
```

解説

3D #cylinder または #sphere モデルリソースプロパティ。球体または円柱をどこまで描画するかを示します。

球体のサーフェスは、Y 軸の startAngle から endAngle まで 2 次元の半円弧を描いて作成します。startAngle が 0 に、endAngle が 360 に設定されている場合は、完全な球体になります。球体の一部を描画する場合は、endAngle を 360 未満に設定します。

円柱のサーフェスは、startAngle から endAngle まで Y 軸の周囲に 2 次元の線を描いて作成します。startAngle が 0 に、endAngle が 360 に設定されている場合は、完全な円柱になります。円柱の一部を描画する場合は、endAngle を 360 未満に設定します。

このプロパティのデフォルト値は 360 です。

例文

次の例では、"MyMember" というキャストメンバに " 球体 4" というモデルリソースを使用するモデルが含まれているとします。このとき、endAngle の値は 310 で、50° 開いたままになります。ハンドラ closeSphere は、引き戸を閉めるように球体の空いている部分を閉じます。繰り返しループで、球体の endAngle の値は 1 度ずつ変化します。繰り返しループ中の updateStage コマンドは、値が 1 度増加するごとにステージを再描画します。

```
-- Lingo syntax
on closeSphere
    MyAngle = member("MyMember").modelresource("Sphere4").endAngle
    repeat with r = 1 to 50
        MyAngle = MyAngle + 1
        member("MyMember").modelresource("Sphere4").endAngle = MyAngle
        updateStage
    end repeat
end

// JavaScript syntax
function closeSphere(){
    var MyAngle = member("MyMember").getProp("modelresource", 1).endAngle;
    for( r = 1; r <= 50; r++)
    {
        MyAngle++;
        member("MyMember").getProp("modelresource", 1).endAngle = MyAngle;
        _movie.updateStage();
    }
}
```

関連事項

state (3D)

endColor

使用法

```
-- Lingo syntax
memberObjRef.endColor

// JavaScript syntax
memberObjRef.endColor;
```

解説

ベクトルシェイプキャストメンバプロパティ。RGB 値として指定したグラデーションシェイプの塗りつぶしの終わりの色を示します。

endColor は、fillMode が #gradient に設定されている場合にのみ有効です。また、塗りつぶしを始める色は fillColor を使って指定します。

このプロパティは、値を調べたり、設定することができます。

完成したムービーの endColor の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

関連事項

color(), fillColor, fillMode

endFrame

使用法

```
-- Lingo syntax
spriteObjRef.endFrame

// JavaScript syntax
spriteObjRef.endFrame;
```

解説

スプライトの範囲にある最後のフレームの番号を返すスプライトプロパティです。読み取り専用。

このプロパティは、特定のスプライトのスコアで範囲を決めるために使います。

この関数は、スプライトを含むフレーム内でのみ使用できます。ムービーの別のフレームのスプライトに適用することはできません。

例文

次のステートメントは、チャネル 5 にあるスプライトの終了フレームをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(5).endFrame)

// JavaScript syntax
put(sprite(5).endFrame);
```

関連事項

スプライト , startFrame

endian

使用法

```
byteArrayObject.endian
```

解説

バイト配列での整数および浮動小数点数の読み書きに使用されるバイト配列プロパティです。

例文

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.endian

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.endian);
```

endTime （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.endTime

// JavaScript syntax
soundChannelObjRef.endTime;
```

解説

再生中、一時停止中またはキューで待機中のいずれかのサウンドについて、指定された終了時刻を示すサウンドチャネルプロパティです。読み取り / 書き込み可能。

終了時刻とは、そのサウンドメンバが再生を終了する時刻のことです。この値は浮動小数点値なので、サウンドの再生時間をミリ秒未満の単位まで測定および制御できます。デフォルト値は、通常のサウンド終了時刻です。

このプロパティは、queue() メソッドまたは setPlayList() メソッドのパラメータとして渡す場合にのみ、通常のサウンド終了時刻以外の値を設定できます。

例文

次のステートメントは、サウンドメンバ " ジングル " が、サウンドチャネル 1 で最初から最後までサウンド全体を再生するように設定されているかどうかを確認します。

```
-- Lingo syntax
if (sound(1).startTime > 0 and sound(1).endTime < member("Jingle").duration) then _player.alert("Not playing
the whole sound.")
end if

// JavaScript syntax
if (sound(1).startTime > 0 && sound(1).endTime < member("Jingle").duration) {
    _player.alert("Not playing the whole sound.");
}
```

関連事項

queue(), setPlayList(), サウンドチャネル

endTime （サウンドオブジェクト）

使用法

```
soundObject.endTime
```

解説

現在のサウンドオブジェクトの終了時間をミリ秒単位を返すサウンドオブジェクトプロパティです。

例文

```
soundObject.endTime = 46890
--Lingo syntax
on mouseUp me
    put soundObjRef.endTime -- Displays the end time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.endTime) ; // Displays the end time for the sound object
// associated with soundobjectRef.
}
```

関連事項

startTime（サウンドオブジェクト）

environmentPropList

使用法

```
-- Lingo syntax
_system.environmentPropList

// JavaScript syntax
_system.environmentPropList;
```

解説

Director コンテンツを現在実行している環境に関する情報のリストを含むシステムプロパティです。読み取り専用。

この設計は、将来的に environmentPropList プロパティの情報を追加しても既存のムービーに影響しないようにするためのものです。

情報は、その領域に関するプロパティ名と値のペアという形式になっています。

#platform	"Macintosh,PowerPC" または "Windows,32" を含むストリングです。ムービーが実行されている現在の OS およびハードウェア環境によって決まります。
#runMode	"Author"、"Projector"、または "Plugin" を含むストリングです。ムービーが現在どのアプリケーションで実行されているかによって決まります。
#colorDepth	ステージが表示されるモニタのビット深度を表す整数です。1、2、4、8、16、32 が値として有効です。
#internetConnected	ムービーが再生されているコンピュータがインターネットに接続されているかどうかを表すシンボルです。使用できる値は、#online と #offline です。
#uiLanguage	プレーヤがユーザインターフェイスの表示に使用している言語を表すストリングです。

#osLanguage	コンピュータのオペレーティングシステムが使用している第一言語です。
#osVersion	#osVersion の値はストリングです。 Windows では、#osVersion プロパティに、GetVersionEx() システム呼び出しで取得した情報が入力されます。次に示すストリング内の値は、OSVERSIONINFO 構造からのものです。 "Windows CE" または "Windows NT" または "Windows 2000" または Windows XP" または "Windows 95" または "Windows 98" または "Windows ME" dwMajorVersion dwMinorVersion dwOSVersionInfoSize dwPlatformId szCSDVersion Macintosh では、次に示すストリングの値は Gestalt(gestaltSystemVersion) 呼び出しからのものです。 "Mac OS" major version minor version sub-version
#productBuildVersion	再生アプリケーションの内部ビルド番号を表すストリングです。

これらのプロパティには、同じ名前のプロパティまたは関数とまったく同じ情報を含みます。

例文

次のステートメントは、メッセージウィンドウに環境リストを表示します。

```
-- Lingo syntax
put(_system.environmentPropList)

// JavaScript syntax
put(_system.environmentPropList);
```

関連事項

システム

error

使用法

```
member(whichCastmember).model(whichModel).sds.error
```

解説

3D #sds モディファイアプロパティ。モデルのジオメトリの詳細を合成するときに、モディファイアで許容されるエラーのパーセンテージを示します。

このプロパティが有効なのは、モディファイアの subdivision プロパティが #adaptive に設定されている場合のみです。モディファイアの tension および depth (3D) プロパティと error プロパティを組み合わせて、モディファイアによるサブディビジョンの量を制御します。

例文

次のステートメントは、" 赤ちゃん " というモデルの #sds モディファイアの error プロパティを 0 に設定します。このモディファイアの tension 設定が低く、depth 設定が高く、subdivision 設定が #adaptive の場合、このステートメントは非常にきわだった効果を " 赤ちゃん " のジオメトリに与えます。

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.subdivision = #adaptive
member("Scene").model("Baby").sds.error = 0

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model",2).getPropRef("sds").subdivision = symbol("adaptive");
member("Scene").getProp("model",2).getPropRef("sds").error = 0;
```

関連事項

sds（モディファイア）, subdivision, depth (3D), tension

eventPassMode

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.eventPassMode

// JavaScript syntax
memberOrSpriteObjRef.eventPassMode;
```

解説

Flash のキャストメンバプロパティおよびスプライトプロパティです。Flash ムービーから、Flash スプライトに割り当てられたビヘイビアーへのマウスイベントの送信を制御します。eventPassMode プロパティは、次のいずれかの値になります。

- #passAlways（デフォルト）- 常にマウスイベントを送ります。
- #passButton - Flash ムービーのボタンがクリックされた場合にのみマウスイベントを送ります。
- #passNotButton - ボタン以外のオブジェクトがクリックされた場合にのみマウスイベントを送ります。
- #passNever - マウスイベントを送りません。

このプロパティは、値を調べたり、設定することができます。

例文

次のフレームスクリプトは、Flash ムービースプライトのボタンが現時点で選択可能かどうかをチェックし、選択可能な場合は eventPassMode を #passNotButton に設定します。選択不可能な場合は、eventPassMode を #passAlways に設定します。このスクリプトの効果は次のとおりです。

- ボタン以外のオブジェクト上のマウスイベントは常にスプライトスクリプトに送られます。
- ボタンが選択不可能な場合、ボタンオブジェクト上のマウスイベントはスプライトスクリプトへ送られます。選択可能な場合、ボタンのマウスイベントは停止します。

```
-- Lingo syntax
on enterFrame
    if sprite(5).buttonsEnabled = TRUE then
        sprite(5).eventPassMode= #passNotButton
    else
        sprite(5).eventPassMode = #passAlways
    end if
end

// JavaScript syntax
function enterFrame() {
    var btEn = sprite(5).buttonsEnabled;
    if (btEn == 1) {
        sprite(5).eventPassMode= symbol("passNotButton");
    } else {
        sprite(5).eventPassMode = symbol("passAlways");
    }
}
```

exitLock

使用法

```
-- Lingo syntax
_movie.exitLock

// JavaScript syntax
_movie.exitLock;
```

解説

ユーザがプロジェクタを終了して Windows のデスクトップまたは Macintosh の Finder に戻ることができるかどうかを定義するためのムービープロパティです。TRUE または FALSE（デフォルト値）で定義します。読み取り / 書き込み可能。

Windows では、Ctrl+ ピリオド（.）、Ctrl+Q、Ctrl+W または Escape キーを押して終了し、デスクトップに戻ります。Macintosh では、Command+ ピリオド（.）、Command+Q、または Command+W を押します。

例文

次のステートメントは exitLock プロパティを TRUE に設定します。

```
-- Lingo syntax
_movie.exitLock = TRUE

// JavaScript syntax
_movie.exitLock = true;
```

exitLock が TRUE の場合、Windows で Ctrl+ ピリオド（.）、Ctrl+Q、Ctrl+W または Esc キーを押しても終了してデスクトップに戻ることができません。Macintosh の Command+ ピリオド（.）、Command+Q、または Command+W も無効です。次のハンドラは、キーボード入力で終了のショートカットキーが押されたかどうかを調べ、押された場合は " 終了シーケンス " を再生します。

```
-- Lingo syntax
on checkExit
    if ((_key.commandDown) and (_key.key = "." or _key.key = "q") and (_movie.exitLock = TRUE)) then
_movie.go("quit sequence")
end checkExit

// JavaScript syntax
function checkExit() {
    if ((_key.commandDown) && (_key.key == "." || _key.key == "q") && (_movie.exitLock == true)) {
        _movie.go("quit sequence");
    }
}
```

関連事項

ムービー

externalParamCount

使用法

```
-- Lingo syntax
_player.externalParamCount

// JavaScript syntax
_player.externalParamCount;
```

解説

HTML の <EMBED> タグまたは <OBJECT> タグから Shockwave コンテンツに渡されるパラメータの数を返すプレーヤプロパティです。読み取り専用。

このプロパティは、ブラウザ内で再生される Shockwave コンテンツを含むムービーのみに有効です。オーサリング環境またはプロジェクタで再生するムービーでは機能しません。

外部パラメータについて詳しくは、externalParamName() および externalParamValue() を参照してください。

例文

次のハンドラは、<OBJECT> タグまたは <EMBED> タグが外部パラメータを Shockwave コンテンツを含むムービーに渡すかどうかを調べ、渡す場合には Lingo ステートメントを実行します。

```
-- Lingo syntax
if (_player.externalParamCount > 0) then
    -- perform some action
end if

// JavaScript syntax
if (_player.externalParamCount > 0) {
    // perform some action;
}
```

関連事項

externalParamName(), externalParamValue(), プレーヤ

face

使用法

```
\-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).face.count
member(whichCastmember).modelResource(whichModelResource).face[index].colors
member(whichCastmember).modelResource(whichModelResource).face[index].normals
member(whichCastmember).modelResource(whichModelResource).face[index].shader
member(whichCastmember).modelResource(whichModelResource).face[index].textureCoordinates
member(whichCastmember).modelResource(whichModelResource).face[index].vertices
member(whichCastmember).model(whichModel).meshdeform.face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex].neighbor{[neighborIn
dex]}

// JavaScript Usage
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face", index).colors
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).normals
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face", index).shader
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).textureCoordinates
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).vertices
member(whichCastmember).getProp("model", whichModelIndex).getPropRef("meshdeform").getPropRef("mesh",
meshIndex).face[faceIndex]
member(whichCastmember).getProp("model", whichModelIndex).getPropRef("meshdeform").getPropRef("mesh",
meshIndex).face[faceIndex].neighbor{[neighborIndex]}
```

解説

3D #mesh モデルリソースおよび meshdeform モディファイアプロパティ。モデルリソースはすべて、三角形で構成されたメッシュです。各三角形はフェイスです。

#mesh というタイプのモデルリソースの、フェイスのプロパティにアクセスできます。これらのプロパティに対する変更は、build() コマンドを呼び出すまでは有効になりません。

注意：以下のプロパティの詳細については、プロパティ個別のエントリを参照してください。

- count は、メッシュ内の三角形ポリゴン (フェイス) の数を示します。
- colors は、モデルリソースのカラーリストのどのインデックスを、フェイスの各頂点に使用するかを示します。
- normals は、モデルリソースの法線リストのどのインデックスを、フェイスの各頂点に使用するかを示します。
- shadowPercentage は、フェイスをレンダリングするときに使用するシェーダを示します。
- textureCoordinates は、モデルリソースのテクスチャ座標リストのどのインデックスを、フェイスの各頂点に使用するかを示します。
- vertices は、モデルリソースの頂点リストのどのインデックスを使用して、フェイスを定義するかを示します。

フェイスのプロパティの詳細については、meshDeform のエントリを参照してください。

関連事項

build(), newMesh, meshDeform （モディファイア）

face[ ]

使用法

```
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
```

解説

3D meshdeform モディファイアプロパティ。フェイスの定義に使用する、モデルリソースの頂点リストのインデックスを示します。

このプロパティは、値を調べられますが、設定はできません。#mesh モデルリソースのフェイスの頂点は、vertexList および vertices プロパティを設定してから build コマンドを呼び出すことで指定できます。

例文

次のステートメントは、" 床 " というモデルの最初のメッシュの最初のフェイスが、" 床 " で使用されているモデルリソースの頂点リストの、最初の 3 つのベクトルで定義されていることを示します。

```
put member("Scene").model("Floor").meshdeform.mesh[1].face[1]
-- [1, 2, 3]
```

関連事項

meshDeform （モディファイア）, face, vertexList (mesh deform), vertices

far (fog)

使用法

```
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
```

解説

3D カメラプロパティ。カメラの fog.enabled プロパティが TRUE に設定されているときに、カメラからフォグが最大密度に達する位置までの距離をワールド単位で示します。

このプロパティのデフォルト値は 1000 です。

例文

次のステートメントは、"BayView" というカメラのフォグの far プロパティを 5000 に設定します。フォグの enabled プロパティが TRUE に設定されている場合、フォグの密度はカメラの前方 5000 ワールド単位の位置で最大になります。

```
-- Lingo syntax
member("MyYard").camera("BayView").fog.far = 5000

// JavaScript syntax
member("MyYard").getProp("camera",1).getPropRef("fog").far = 5000;
```

関連事項

fog, near (fog)

fieldOfView

使用法

```
-- Lingo syntax
spriteObjRef.fieldOfView

// JavaScript syntax
spriteObjRef.fieldOfView;
```

解説

特定スプライトの現在のフィールドビューを角度で表す QTVR スプライトプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、1 番目のカメラの fieldOfView プロパティを 90 に設定します。

```
-- Lingo syntax
member("3d world").camera[1].fieldOfView = 90

// JavaScript syntax
member("3d world").getProp("camera",1).fieldOfView = 90;
```

fieldOfView (3D)

使用法

```
member(whichCastmember).camera(whichCamera).fieldOfView
sprite(whichSprite).camera{(index)}.fieldOfView
```

解説

3D カメラプロパティ。カメラと投影平面の上辺を結ぶ線、およびカメラと投影平面の下辺を結ぶ線の 2 つの線が形成する角度を示します。

3D ワールドでのモデルの画像は、投影平面上にマッピングされます。これは、映写機の前にあるスクリーンのように、カメラの前に位置する平面です。3D スプライトで表示されるものが投影平面です。投影平面の上部と下部は、fieldOfView プロパティで定義されます。ただし、fieldOfView プロパティの値を変更しても、スプライトのサイズは変更されません。スプライトのサイズに合わせて、投影平面上の画像が拡大縮小されます。

このプロパティの値は、カメラの projection プロパティの値が #perspective に設定されているときだけ意味を持ちます。projection プロパティが #orthographic に設定されているときは、カメラの orthoHeight プロパティを使用して、投影平面の上辺と下辺を定義します。

このプロパティのデフォルト値は 30.0 です。

例文

次のステートメントは、1 番目のカメラの fieldOfView プロパティを 90 に設定します。

```
member("3d world").camera[1].fieldOfView = 90
```

関連事項

orthoHeight

fileFreeSize

使用法

```
-- Lingo syntax
_movie.fileFreeSize

// JavaScript syntax
_movie.fileFreeSize;
```

解説

現行ムービー内で、キャストライブラリやキャストメンバへの変更が原因で、未使用になっているメモリ量をバイト数で返すムービープロパティです。読み取り専用。

Save、Compact、および Save As の各コマンドはファイルを書き直して、空きスペースを削除します。

未使用スペースが存在しない場合、fileFreeSize は 0 を返します。

例文

次のステートメントは、現行ムービー内の未使用のメモリ量をバイト数で表示します。

```
-- Lingo syntax
put(_movie.fileFreeSize)

// JavaScript syntax
put(_movie.fileFreeSize);
```

関連事項

ムービー

fileName （キャスト）

使用法

```
-- Lingo syntax
castObjRef.fileName

// JavaScript syntax
castObjRef.fileName;
```

解説

キャストライブラリのファイル名を返すか設定するキャストライブラリプロパティです。内部キャストライブラリの場合は読み取り専用で、外部キャストライブラリの場合は読み取りと書き込みが可能です。

外部キャストライブラリの場合、fileName はキャストのフルパス名とファイル名を返します。

内部キャストライブラリの場合、fileName は、指定されている内部キャストライブラリに応じて、値を返します。

- 最初の内部キャストライブラリが指定されている場合、fileName はムービーの名前を返します。
- その他の内部キャストライブラリが指定されている場合、fileName は空の文字列を返します。

このプロパティには URL 参照を設定できますが、インターネット上のキャストライブラリを使用するときには、downloadNetThing() または preloadNetThing() メソッドでキャストファイルをローカルディスクにダウンロードしてから fileName でそのファイル名を設定します。このようにすると、ダウンロードの待ち時間を減らすことができます。

例文

次のステートメントは、外部キャスト " ボタン " のパス名およびファイル名をメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(castLib("Buttons").fileName)

// JavaScript syntax
trace(castLib("Buttons").fileName);
```

次のステートメントは、外部キャスト " ボタン " が参照するファイルを "Content.cst" に設定します。

```
-- Lingo syntax
castLib("Buttons").fileName = _movie.path & "Content.cst"

// JavaScript syntax
castLib("Buttons").fileName = _movie.path + "Content.cst";
```

このように設定すると、ムービーは外部キャストファイル "Content.cst" を " ボタン " キャストとして使用するようになります。

次のステートメントは、URL から外部キャストを Director アプリケーションフォルダにダウンロードし、そのファイルを外部キャスト " キャスト集 " として設定します。

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath & "Thousands.cst")
castLib("Cast of Thousands").fileName = _player.applicationPath & "Thousands.cst"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath + "Thousands.cst");
castLib("Cast of Thousands").fileName = _player.applicationPath + "Thousands.cst";
```

関連事項

キャストライブラリ , downloadNetThing, preloadNetThing()

fileName （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.fileName

// JavaScript syntax
memberObjRef.fileName;
```

解説

リンクキャストメンバに割り当てられているファイルの名前を示すメンバプロパティです。読み取り / 書き込み可能。

このプロパティは、ムービーの再生中に、キャストメンバに割り当てられた外部リンクファイルを切り替える場合に便利です。キャストメンバの切り替えと同様のことができます。リンクされているファイルがムービーとは別のフォルダに入っている場合には、ファイルのパス名も指定する必要があります。

リンクしているメディアをサポートするメンバタイプのファイル名を設定することによって読み込んだメディアをリンクさせることもできます。

このプロパティには URL 参照を指定することもできますが、URL からのファイルを使用するときには、downloadNetThing() または preloadNetThing() メソッドでファイルをローカルディスクにダウンロードしてから fileName でそのファイル名を設定します。このようにすると、ダウンロードの待ち時間を減らすことができます。

ファイル名を設定すると、以後そのキャストメンバを使う際はそのファイルが使われるようになります。

例文

次のステートメントは QuickTime ムービー "ChairAnimation" をキャストメンバ 40 にリンクします。

```
-- Lingo syntax
member(40).fileName = "ChairAnimation"

// JavaScript syntax
member(40).fileName = "ChairAnimation";
```

次のステートメントは、URL から外部ファイルをダウンロードして Director と同じフォルダに入れ、そのファイルをサウンドキャストメンバ " ノーマ・デズモンドが語る " というメディアに設定します。

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath & "Talkies.AIF")
member("Norma Desmond Speaks").fileName = _player.applicationPath & "Talkies.AIF"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath + "Talkies.AIF");
member("Norma Desmond Speaks").fileName = _player.applicationPath + "Talkies.AIF";
```

関連事項

downloadNetThing, メンバ , preloadNetThing()

fileName (MP4Media/FLV)

使用法

```
put(sprite(1).fileName)
Member(1).fileName = "C:\downloads\movie.mp4"
```

解説

MP4Media/FLV メンバプロパティ。MP4Media/FLV メンバのファイル名を定義します。このプロパティは、リンクキャストメンバに割り当てられているファイルの名前を指定します。

fileName は、読み取り書き込み可能プロパティです。

例文

次の例は、スプライト 7 の fileName プロパティを設定します。

```
-- Lingo syntax
sprite(7).fileName = "C:\Documents and Settings\user1\Desktop\DEMO_VIDEO\Ratatoullie.mp4"

// JavaScript syntax
sprite(7).fileName = "C:\Documents and Settings\user1\Desktop\DEMO_VIDEO\Ratatoullie.mp4";
```

fileName （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.fileName

// JavaScript syntax
windowObjRef.fileName;
```

解説

ウィンドウに割り当てられているムービーのファイル名を示すウィンドウプロパティです。読み取り / 書き込み可能。

リンクされているファイルがムービーとは別のフォルダに入っている場合には、ファイルのパス名も指定する必要があります。

ウィンドウ内でムービーを再生するには、fileName ウィンドウプロパティにムービーのファイル名を設定します。

fileName プロパティには URL 参照を設定することもできますが、URL からのムービーファイルを使用するときには、downloadNetThing() または preloadNetThing() メソッドでムービーファイルをローカルディスクにダウンロードしてから fileName でそのファイル名を設定します。このようにすると、ダウンロードの待ち時間を減らすことができます。

例文

次のステートメントは " 制御パネル " という名前のファイルを " ツールボックス " ウィンドウに割り当てます。

```
-- Lingo syntax
window("Tool Box").fileName = "Control Panel"

// JavaScript syntax
window("Tool Box").fileName = "Control Panel";
```

次のステートメントは " ナビゲータ " ウィンドウに割り当てられているファイルの名前を表示します。

```
-- Lingo syntax
trace(window("Navigator").fileName)

// JavaScript syntax
trace(window("Navigator").fileName);
```

次のステートメントは、URL からムービーファイルをダウンロードして Director と同じフォルダに入れ、そのファイルを " クローズアップ " ウィンドウに割り当てます。

```
-- Lingo syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath & "Finale.DIR")
window("My Close Up").fileName = _player.applicationPath & "Finale.DIR"

// JavaScript syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath + "Finale.DIR");
window("My Close Up").fileName = _player.applicationPath + "Finale.DIR";
```

関連事項

downloadNetThing, preloadNetThing(), ウィンドウ

fileSize

使用法

```
-- Lingo syntax
_movie.fileSize

// JavaScript syntax
_movie.fileSize;
```

解説

ディスクに保存された現行ムービーのバイト数を返すムービープロパティです。読み取り専用。

Windows のファイル／プロパティまたは Macintosh の「情報を見る」を使用したときと同じ結果になります。

例文

次のステートメントは、現行ムービーのバイト数を表示します。

```
-- Lingo syntax
put(_movie.fileSize)

// JavaScript syntax
put(_movie.fileSize);
```

関連事項

ムービー

fileVersion

使用法

```
-- Lingo syntax
_movie.fileVersion

// JavaScript syntax
_movie.fileVersion;
```

解説

最後にムービーが保存されたときの Director のバージョンをストリングとして示すムービープロパティです。読み取り専用。

例文

次のステートメントは、現行ムービーが保存されたときの Director のバージョンを表示します。

```
-- Lingo syntax
put(_movie.fileVersion)

// JavaScript syntax
put(_movie.fileVersion);
```

関連事項

ムービー

fillColor

使用法

```
-- Lingo syntax
memberObjRef.fillColor

// JavaScript syntax
memberObjRef.fillColor;
```

解説

シェイプの塗りつぶし色を RGB 値で指定する、ベクトルシェイプキャストメンバプロパティです。

シェイプの fillMode プロパティが #solid または #gradient に設定されているときは、fillColor を使用することができますが、#none に設定されているときはできません。fillMode が #gradient に設定されている場合、fillColor グラデーションの最初の色を指定し、endColor 最後の色を指定します。

このプロパティは、値を調べたり、設定することができます。

完成したムービーの fillColor の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントは、メンバ "Archie" の塗りつぶし色を新しい RGB 値に設定します。

```
-- Lingo syntax
member("Archie").fillColor = color( 24, 15, 153)

// JavaScript syntax
member("Archie").fillColor = color( 24, 15, 153);
```

関連事項

endColor, fillMode

fillCycles

使用法

```
-- Lingo syntax
memberObjRef.fillCycles

// JavaScript syntax
memberObjRef.fillCycles;
```

解説

グラデーションの塗りつぶしサイクル番号を示すベクトルシェイプキャストメンバプロパティです。塗りつぶし番号は 1 ～ 7 の整数で指定されます。

シェイプの fillMode プロパティが #gradient に設定されているときのみ有効ですが、値を調べたり、設定できます。

このプロパティは、値を調べたり、設定することができます。

完成したムービーの fillCycles の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントは、メンバ "Archie" の fillCycles を 3 に設定します。

```
-- Lingo syntax
member("Archie").fillCycles = 3

// JavaScript syntax
member("Archie").fillCycles = 3;
```

関連事項

endColor, fillColor, fillMode

fillDirection

使用法

```
-- Lingo syntax
memberObjRef.fillDirection

// JavaScript syntax
memberObjRef.fillDirection;
```

解説

シェイプの塗りつぶし方向を角度で指定するベクトルシェイプキャストメンバプロパティです。

シェイプの fillMode プロパティが #gradient に設定されているときのみ有効です。

このプロパティは、値を調べたり、設定することができます。

完成したムービーの fillDirection の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のビヘイビアーは、ベクトルシェイプの fillMode を 'gradient' に変更し、ベクトルシェイプの 'fillDirection' を連続的に変更して、シンプルなアニメーションを作成します。

```
-- Lingo syntax
on beginSprite(me)
    member("VectorShape").fillMode = #gradient
end

on exitFrame(me)
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) mod \ 360
end

// JavaScript syntax
function beginSprite(me)
{
    member("VectorShape").fillMode = symbol("gradient");
}

function exitFrame(me)
{
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) % 360;
}
```

関連事項

fillMode

filled

使用法

```
member(whichCastMember).filled
the filled of member whichCastMember
```

解説

指定のキャストメンバをパターンで塗りつぶすかどうかを指定するシェイプキャストメンバプロパティです。TRUE の場合は塗りつぶされ、FALSE の場合は塗りつぶされません。

例文

次のステートメントはシェイプキャストメンバ " ターゲットエリア " を塗りつぶし、そこにパターン番号 1 (単色) を割り当てます。

```
-- Lingo syntax
member("Target Area").filled = TRUE
member("Target Area").pattern = 1

// Java Script
member("Target Area").filled = true;
member("Target Area").pattern = 1;
```

関連事項

fillColor, fillMode

fillMode

使用法

```
-- Lingo syntax
memberObjRef.fillMode

// JavaScript syntax
memberObjRef.fillMode;
```

解説

シェイプの塗りつぶし方法を示すベクトルシェイプキャストメンバプロパティです。次の値が有効です。

- #none - シェイプは透明です。
- #solid - シェイプは単一の塗りつぶしカラーを使用します。
- #gradient - シェイプは 2 色間のグラデーションを使用します。

シェイプが閉じているときは、このプロパティの値を調べたり、変更できます。シェイプが開いているときは塗りつぶされません。

完成したムービーの fillMode の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントは、メンバ "Archie" の fillMode をグラデーションに設定します。

```
-- Lingo syntax
member("Archie").fillMode = #gradient

// JavaScript syntax
member("Archie").fillMode = symbol("gradient");
```

関連事項

endColor, fillColor

fillOffset

使用法

```
-- Lingo syntax
memberObjRef.fillOffset

// JavaScript syntax
memberObjRef.fillOffset;
```

解説

シェイプの塗りつぶしをオフセットするため、defaultRect スペース内の水平方向と垂直方向のサイズをピクセル数で定義するベクトルシェイプキャストメンバプロパティです。

このプロパティは、シェイプの fillMode プロパティが #gradient に設定されているときのみ有効ですが、値を調べたり、設定したりできます。

完成したムービーの fillOffset の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントはベクトルシェイプキャストメンバ "miette" の塗りつぶしオフセットを水平方向 33 ピクセル、垂直方向 27 ピクセルのオフセットに変更します。

```
-- Lingo syntax
member("miette").fillOffset = point(33, 27)

// JavaScript syntax
member("miette").fillOffset = point(33, 27);
```

関連事項

defaultRect, fillMode

fillScale

使用法

```
-- Lingo syntax
memberObjRef.fillScale

// JavaScript syntax
memberObjRef.fillScale;
```

解説

シェイプで塗りつぶす範囲の度合いを指定するベクトルシェイプキャストメンバプロパティですこのプロパティはベクトルシェイプウィンドウでは "spread" と呼ばれます。

このプロパティは、シェイプの fillMode プロパティが #gradient に設定されているときのみ有効ですが、値を調べたり、設定したりできます。

完成したムービーの fillScale の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントは、メンバ "Archie" の fillScale を 33 に設定します。

```
-- Lingo syntax
member("Archie").fillScale = 33.00

// JavaScript syntax
member("Archie").fillScale = 33.00;
```

関連事項

fillMode

filterlist

使用法

```
-- Lingo syntax
spriteObjRef.filterlist
// JavaScript syntax
spriteObjRef.filterlist;
```

解説

スプライトプロパティ。スプライトにビットマップフィルタを適用するかどうかを指定します。これはリストであり、スプライトをリストの最後に追加することによって、スプライトにビットマップフィルタを適用できます。

注意：duplicate() メソッドを使用してフィルタリストを複製することはできません。

例文

最初のステートメントは、myFilter という変数にぼかしフィルタを設定します。次の行は、ぼかしフィルタを sprite(1) に設定します。

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)
// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

関連事項

Bitmap filters in Using Director.

filterList （ミクサ）

使用方法

```
mixer.filterList
```

解説

ミクサに現在適用されているオーディオフィルタのリストを返すオーディオミクサプロパティです。

例文

```
--Lingo syntax
on mouseUp me
    put mixer1.filterlist -- Returns the filters currently applied to mixer1.
end

// JavaScript syntax
function mouseup()
{
put (mixer1.filterList); // Returns the filters currently applied to mixer1.
}
```

関連事項

ミクサ

filterList （サウンドオブジェクト）

使用法

```
soundObject.filterList
```

解説

サウンドオブジェクトに適用されるオーディオフィルタのリストを示すサウンドオブジェクトプロパティです。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.filterlist -- Displays the filter list for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.filterList) ; //Displays the filter list for the sound object
// associated with soundobjectRef.

}
```

firstIndent

使用法

```
-- Lingo syntax
chunkExpression.firstIndent

// JavaScript syntax
chunkExpression.firstIndent;
```

解説

[チャンク式] の最初のインデントが [チャンク式] の左マージンからオフセットされるピクセル数を示すキャストメンバプロパティです。

値は整数で、0 以下はハンギングインデント、0 はインデントがないこと、0 以上は通常のインデントを意味します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、メンバ " 机 " の先頭行を 0 ピクセルだけインデントします。

```
--Lingo syntax
member("Desk").firstIndent = 0

// JavaScript syntax
member("Desk").firstIndent = 0;
```

関連事項

leftIndent, rightIndent

fixedLineSpace

使用法

```
-- Lingo syntax
chunkExpression.fixedLineSpace

// JavaScript syntax
chunkExpression.fixedLineSpace;
```

解説

テキストキャストメンバの [チャンク式] 部分の行の高さを制御するキャストメンバプロパティです。

各行の高さを整数のピクセル数で表します。

デフォルト値は 0 で、行の自然な高さを意味します。

例文

次のステートメントは、メンバ " 机 " の各行の高さを 24 ピクセルに設定します。

```
--Lingo syntax
member("Desk").fixedLineSpace = 24

// JavaScript syntax
member("Desk").fixedLineSpace = 24;
```

fixedRate

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.fixedRate

// JavaScript syntax
memberOrSpriteObjRef.fixedRate;
```

解説

Flash ムービーまたはアニメーション GIF のフレームレートを制御するキャストメンバプロパティ、スプライトプロパティです。fixedRate プロパティには整数値を設定できます。デフォルト値は 15 です。

スプライトの playbackMode プロパティが #fixed 以外の場合、このプロパティは無視されます。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは Flash ムービースプライトのフレームレートを調整します。ハンドラはパラメータとして、スプライトの参照、Flash ムービーの速度の増減についての指定、調節する速度の増減を受け入れます。

```
-- Lingo syntax
on adjustFixedRate(whichSprite, adjustType, howMuch)
    case adjustType of
        #faster:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch
        #slower:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch
    end case
end

// JavaScript syntax
function adjustFixedRate(whichSprite, adjustType, howMuch) {
    switch(adjustType) {
        case "faster":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch;
            break;
        case "slower":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch;
            break;
    }
}
```

関連事項

playBackMode

fixStageSize

使用法

```
-- Lingo syntax
_movie.fixStageSize

// JavaScript syntax
_movie.fixStageSize;
```

解説

別のムービーを読み込むときに、ムービーで保存するステージサイズや centerStage の設定にかかわらず現行ステージのサイズを維持するかどうかを定義するプロパティです。現行ステージのサイズを維持する場合は TRUE（デフォルト）で、維持しない場合は FALSE です。デフォルト値は TRUE です。読み取り / 書き込み可能。

fixStageSize では、現在再生中のムービーのステージサイズを変更することはできません。

例文

次のステートメントは、fixStageSize プロパティがオンになっているかどうかを調べます。fixStageSize が FALSE の場合は、再生ヘッドが指定されたフレームに戻されます。

```
-- Lingo syntax
if (_movie.fixStageSize = FALSE) then
    _movie.go("proper size")
end if

// JavaScript syntax
if (_movie.fixStageSize == false) {
    _movie.go("proper size");
}
```

次のステートメントは現行の fixStageSize プロパティの設定を逆にします。

```
-- Lingo syntax
_movie.fixStageSize = not(_movie.fixStageSize)

// JavaScript syntax
_movie.fixStageSize = !(_movie.fixStageSize);
```

関連事項

centerStage, ムービー

flashRect

使用法

```
-- Lingo syntax
memberObjRef.flashRect

// JavaScript syntax
memberObjRef.flashRect;
```

解説

Flash ムービーまたはベクトルシェイプキャストメンバの元のサイズを定義するキャストメンバプロパティです。プロパティの値は Director の矩形として、rect(0,0,32,32) のように示されます。

Flash キャストメンバがリンクしている場合、キャストメンバのヘッダがメモリへの読み込みを終了している場合にのみ、FlashRect メンバプロパティは有効値を返します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のスプライトスクリプトは Flash ムービースプライトのサイズを変更し、Flash ムービーキャストメンバの元のサイズと同じにします。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).rect = sprite(spriteNum).member.FlashRect
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).rect = sprite(this.spriteNum).member.FlashRect;
}
```

関連事項

defaultRect, defaultRectMode, state (Flash, SWA)

flat

使用法

```
member(whichCastmember).shader(whichShader).flat
member(whichCastmember).model(whichModel).shader.flat
member(whichCastmember).model(whichModel).shaderList{[index]}.flat
```

解説

3D #standard シェーダプロパティ。メッシュをフラットシェーディングでレンダリングするか (TRUE)、グーローシェーディングでレンダリングするか (FALSE) を示します。

フラットシェーディングは、メッシュのフェイスごとにカラーを 1 つ使います。フェイスに使用するカラーは、最初の頂点のカラーです。フラットシェーディングは、グーローシェーディングよりも高速です。

グーローシェーディングは、フェイスの各頂点にカラーを割り当て、フェイス全体のカラーがグラデーションになるようにカラーを補間します。グーローシェーディングには時間と計算が必要ですが、サーフェスはより滑らかになります。

このプロパティのデフォルト値は FALSE です。

例文

次のステートメントは、" 壁 " というシェーダの flat プロパティを TRUE に設定します。このシェーダを使用するモデルのメッシュは、単一カラーのフェイスでレンダリングされます。

```
-- Lingo syntax
member("MysteryWorld").shader("Wall").flat = TRUE

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).flat = true;
```

関連事項

mesh （プロパティ）, colors, vertices, generateNormals()

flipH

使用法

```
-- Lingo syntax
spriteObjRef.flipH

// JavaScript syntax
spriteObjRef.flipH;
```

解説

スプライトの画像がステージ上で左右反転しているかを示すスプライトプロパティです。左右反転している場合は TRUE、左右反転していない場合は FALSE となります。読み取り専用。

画像自体がセンターポイントを中心として反転されます。

つまり、回転や傾斜はそのままの状態で、画像データのみが反転します。

例文

次のステートメントはスプライト 5 の flipH を表示します。

```
-- Lingo syntax
put(sprite(5).flipH)

// JavaScript syntax
put(sprite(5).flipH);
```

関連事項

flipV, rotation, skew, スプライト

flipV

使用法

```
-- Lingo syntax
spriteObjRef.flipV

// JavaScript syntax
spriteObjRef.flipV;
```

解説

スプライトの画像がステージ上で上下反転しているかを示すスプライトプロパティです。上下反転している場合は TRUE、上下反転していない場合は FALSE となります。読み取り専用。

画像自体がセンターポイントを中心として反転されます。

つまり、回転や傾斜はそのままの状態で、画像データのみが反転します。

例文

次のステートメントは、スプライト 5 の flipV を TRUE に設定します。

```
-- Lingo syntax
put(sprite(5).flipV)

// JavaScript syntax
put(sprite(5).flipV);
```

関連事項

flipH, rotation, skew, スプライト

floatPrecision

使用法

```
the floatPrecision
```

解説

浮動小数の小数点を指定の桁数に四捨五入するシステムプロパティです。floatPrecision の値は整数であることが必要です。桁数の最大値は 15 です。デフォルト値は 4 です。

floatPrecision は浮動小数を表示する桁数を定義するプロパティであり、計算に使用する桁数は変更しません。

- floatPrecision が 1 ～ 15 の値の場合は、浮動小数は指定の桁数で表示されます。小数点以下の "0" も表示されます。
- floatPrecision が 0 の場合、浮動小数は最も近い整数に四捨五入され、小数点は表示されません。
- floatPrecision が負の値の場合、浮動小数は小数点以下の指定桁数の絶対値に四捨五入されます。小数点以下の "0" は表示されません。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、3.0 の平方根を小数点以下 3 桁に四捨五入します。

```
the floatPrecision = 3
x = sqrt(3.0)
put x
-- 1.732
```

次のステートメントは、3.0 の平方根を小数点以下 8 桁に四捨五入します。

```
the floatPrecision = 8
put x
-- 1.73205081
```

fog

使用法

```
\-- Lingo Usage
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
member(whichCastmember).camera(whichCamera).fog.near
sprite(whichSprite).camera{(index)}.fog.near
// JavaScript Usage
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").color
sprite(whichSprite).camera.getPropRef("fog").color
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").decayMode
sprite(whichSprite).camera.getPropRef("fog").decayMode
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").enabled
sprite(whichSprite).camera.getPropRef("fog").enabled
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").far
sprite(whichSprite).camera.getPropRef("fog").far
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").near
sprite(whichSprite).camera.getPropRef("fog").near
```

解説

3D カメラプロパティ。fog プロパティにより、カメラからの距離に応じて増加する着色とぼかしがモデルに導入されます。効果は現実の霧と似ていますが、任意のカラーを設定できる点が違います。

関連事項

color (fog), decayMode, enabled (fog), far (fog), near (fog)

folder

使用法

```
-- Lingo syntax
dvdObjRef.folder

// JavaScript syntax
dvdObjRef.folder;
```

解説

DVD プロパティ。再生中の DVD があるフォルダのパス名を調べます。読み取り / 書き込み可能。

パス名はストリングです。

folder プロパティは、プロパティインスペクタまたはスクリプトのいずれかを使用して設定できます。現在の実装では、次の要件があります。

Windows：

- ターゲットのローカル DVD メディアのファイルパスの最後には、video_ts を指定する必要があります。例えば、C:\video_ts or C:\myLocalDVDContent\video_ts のようにします。

  Macintosh：

- フォルダプロパティパスの値は、/Volumes/ で始まる必要があります。
- フォルダプロパティの値に指定したパスへの video_ts の追加は、現在はオプションです。例えば、DVD video_ts フォルダが起動ドライブのルートにある場合、フォルダプロパティの値は、次の 2 とおりの方法で入力できます。
  - /Volumes/Mac HD/myLocalDVDContent/video_ts

    または

  - /Volumes/Mac HD/myLocalDVDContent

プロパティインスペクタでのフォルダプロパティ値の編集

1 DVD キャストメンバを選択し、リストビューモードでプロパティインスペクタの「DVD」タブをアクティブにします。
2 「再生プロパティ」セクションで、フォルダプロパティの編集可能な値フィールドを選択し、ターゲットの DVD メディアの場所を示すファイルパスを入力します。

次の例を参考にして、スクリプトを使用してフォルダのプロパティを設定してください。次のステートメントは、DVD フォルダプロパティのパス名を設定します。

例文

Windows：

```
-- Lingo syntax
member(2).folder = "C:\myLocalDVDContent\video_ts"

// JavaScript syntax
member(2).folder = "C:\\myLocalDVDContent\video_ts";
```

Macintosh：

```
-- Lingo syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent"

// JavaScript syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent";
```

注意：最初の DVD キャストメンバが作成されたときに video_ts フォルダが見つからない場合、"DVD ボリュームが見つかりません " というエラー警告が表示されます。この警告はセッションで 1 回のみ表示されます。この時点で、新しく作成した DVD メンバに名前を付けて、そのフォルダプロパティを有効な video_ts フォルダが含まれるフォルダの場所に設定できます。

Macintosh DVD フォルダパス名に関する問題

Macintosh コンピュータでは、フォルダプロパティのパス名のフォーマットに、標準の Macintosh 区切り記号のコロン（:）の代わりに、パスの区切り記号としてスラッシュ（/）を使用する必要があります。さらに、DVD フォルダのパス名の先頭に /volumes/ を連結する必要があります。例えば、DVD フォルダがブートドライブのルートにある場合、パスは次のようになります。

```
member (2).folder = "/Volumes/Mac HD/Test_DVD/video_ts"
```

Mac 上のプロジェクタやムービーのパスを取得するために _movie.path コマンドを使用するときは、このコマンドにスラッシュ（/）の代わりにコロン（:）が含まれます。DVD フォルダのパス名にコロンを使用すると、エラーが生じます。この問題の対処法として、開発者はスクリプトを使用してパス名のコロン文字をスラッシュに置き換えることができます。

関連事項

DVD

font

使用法

```
-- Lingo syntax
memberObjRef.font

// JavaScript syntax
memberObjRef.font;
```

解説

テキストキャストメンバおよびフィールドキャストメンバのプロパティです。指定されたキャストメンバの表示に使用するフォントを定義します。キャストメンバには、文字が含まれることが必要です。**whichCastMember** パラメータにはキャストメンバの名前または番号のいずれかを指定します。

font メンバプロパティは、値を調べたり、変更できます。

完成したムービーの font の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "text.dir" を参照してください。

例文

次のステートメントは、oldFont という変数に、フィールドキャストメンバ " 六条の語り " の font の現在の設定を代入します。

```
-- Lingo syntax
oldFont = member("Rokujo Speaks").font

// JavaScript syntax
var oldFont = member("Rokujo Speaks").font;
```

関連事項

text, alignment, fontSize, fontStyle, lineHeight

fontSize

使用法

```
-- Lingo syntax
memberObjRef.fontSize

// JavaScript syntax
memberObjRef.fontSize;
```

解説

指定したフィールドキャストメンバの表示に使用するフォントサイズを定義するためのフィールドキャストメンバプロパティです。キャストメンバ内にスペースのみが含まれる場合、文字が必要です。**whichCastMember** パラメータにはキャストメンバの名前または番号のいずれかを指定します。

このプロパティは、値を調べたり、設定することができます。値を調べると、フィールド内の最初の直線の高さが返されます。値を設定すると、フィールド内のすべての直線が設定されます。

完成したムービーの fontSize の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "text.dir" を参照してください。

例文

次のステートメントは、フィールドキャストメンバ " 六条の語り " の fontSize を oldSize 変数に設定します。

```
--Lingo syntax
oldSize = member("Rokujo Speaks").fontSize

// JavaScript syntax
var oldSize = member("Rokujo Speaks").fontSize;
```

次のステートメントは、テキストキャストメンバ "myMenu" の 3 行目 を 24 ポイントに設定します。

```
member("myMenu").fontSize = 12

// JavaScript syntax
member("myMenu").fontSize = 12;
```

関連事項

text, alignment, font, fontStyle, lineHeight

fontStyle

使用法

```
-- Lingo syntax
memberObjRef.fontStyle
memberObjRef.char[whichChar].fontStyle
memberObjRef.line[whichLine].fontStyle
memberObjRef.word[whichWord].fontStyle

// JavaScript syntax
memberObjRef.fontStyle;
memberObjRef.getPropRef("char", whichChar).fontStyle
memberObjRef.getPropRef("line", whichLine).fontStyle;
memberObjRef.getPropRef("word", whichWord).fontStyle;
```

解説

指定したフィールドキャストメンバ、文字、行、単語などのチャンク式を表示するフォントに適用するスタイルを決定するキャストメンバプロパティです。フィールドキャストメンバ内にスペースのみが含まれる場合、文字が必要です。

プロパティの値は、スタイルのストリングをカンマで区切って記述します。Lingo は指定されているフォントにこれらのスタイルを組み合わせてテキストを表示します。使用できるスタイルは plain （標準）、bold （太字）、italic （斜体）、underline （下線）、shadow （影文字）、outline （袋文字）、extended （文字間広く）です。Macintosh では condensed （文字間狭く）も指定できます。

現在設定されているスタイルをすべて削除するには、標準 (plain) スタイルを使用します。**whichCastMember** パラメータにはキャストメンバの名前または番号のいずれかを指定します。

このプロパティは、値を調べたり、設定することができます。

完成したムービーの fontStyle の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "text.dir" を参照してください。

例文

次のステートメントは、フィールドキャストメンバ " 六条の語り " の現在の fontStyle 設定を変数 oldStyle に設定します。

```
--Lingo syntax
oldStyle = member("Rokujo Speaks").fontStyle

// JavaScript syntax
var oldStyle = member("Rokujo Speaks").fontStyle;
```

次のステートメントは、フィールドキャストメンバ " ポエム " の fontStyle メンバプロパティを太字と斜体（bold/italic）に設定します。

```
--Lingo syntax
member("Poem").fontStyle = "bold, italic"

// JavaScript syntax
member("Poem").fontStyle = "bold, italic";
```

次のステートメントはキャストメンバ " 息子の名前 " の 3 番目の単語の fontStyle プロパティを斜体（italic）に設定します。

```
--Lingo syntax
member("Son's Names").word[3].fontStyle = "italic"

// JavaScript syntax
member("Son's Names").getPropRef("word", 3).fontStyle = "italic";
```

次のステートメントは、テキストキャストメンバの fontStyle プロパティを太字と斜体（bold/italic）に設定します。

```
--Lingo syntax
member("text").fontStyle=[#bold]
or
member("text").fontStyle=[#bold, #italic]

// JavaScript syntax
member("text").fontStyle = list(symbol("bold"));
```

関連事項

text, alignment, fontSize, font, lineHeight

foreColor

使用法

```
-- Lingo syntax
spriteObjRef.foreColor

// JavaScript syntax
spriteObjRef.foreColor;
```

解説

スプライトのフォアカラーを返すか、または設定するスプライトプロパティです。読み取り / 書き込み可能。

このプロパティを 1 ビットより色深度の大きいビットマップキャストメンバに適用すると結果の予測が難しく、あまり適切な方法ではありません。

foreColor プロパティの代わりに、新しい color プロパティを使用することをお勧めします。

例文

次のステートメントは、変数 "oldColor" にスプライト 5 のフォアカラーを設定します。

```
-- Lingo syntax
oldColor = sprite(5).foreColor

// JavaScript syntax
var oldColor = sprite(5).foreColor;
```

次のステートメントは、スプライト 11 ～ 13 のうちいずれかのランダムなスプライトについて、フォアカラーの番号を 36 に設定します。

```
-- Lingo syntax
sprite(10 + random(3)).foreColor = 36

// JavaScript syntax
sprite(10 + random(3)).foreColor = 36;
```

関連事項

backColor, color(), スプライト

frame

使用法

```
-- Lingo syntax
_movie.frame

// JavaScript syntax
_movie.frame;
```

解説

ムービーの現行フレーム番号を返すムービープロパティです。読み取り専用。

例文

次のステートメントは、再生ヘッドを現行フレームの前のフレームに戻します。

```
-- Lingo syntax
_movie.go(_movie.frame - 1)

// JavaScript syntax
_movie.go(_movie.frame - 1);
```

関連事項

go(), ムービー

frameCount

使用法

```
-- Lingo syntax
memberObjRef.frameCount

// JavaScript syntax
memberObjRef.frameCount;
```

解説

Flash ムービーキャストメンバのフレーム数を示す Flash キャストメンバです。frameCount メンバプロパティの値は整数です。

このプロパティは、値を調べられますが、設定はできません。

例文

次のスプライトスクリプトは、チャネル番号と Flash ムービーのフレーム数をメッセージウィンドウに表示します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    put(""The Flash movie in channel"" && spriteNum && has"" && /sprite(spriteNum).member.frameCount &&
""frames."")
end

// JavaScript syntax
function beginSprite() {
    trace("The Flash movie in channel " + (this.spriteNum) + " has " +
sprite(this.spriteNum).member.frameCount + " frames.");
}
```

frameLabel

使用法

```
-- Lingo syntax
_movie.frameLabel

// JavaScript syntax
_movie.frameLabel;
```

解説

現行フレームに割り当てられているラベルを識別するムービープロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

現行フレームにラベルが割り当てられていない場合、frameLabel プロパティの値は 0 になります。

例文

次のステートメントは、現行フレームのラベルを調べます。この例では、現行の frameLabel は " 開始 " です。

```
-- Lingo syntax
put(_movie.frameLabel)

// JavaScript syntax
put(_movie.frameLabel);
```

関連事項

labelList, ムービー

framePalette

使用法

```
-- Lingo syntax
_movie.framePalette

// JavaScript syntax
_movie.framePalette;
```

解説

現行フレームで使われているパレット（現行パレットまたは現行フレームに設定されているパレット）のキャストメンバ番号を調べるムービープロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

例文

次のステートメントは、現行フレームで使われているパレットを調べます。この例では、パレットはキャストメンバ 45 です。

```
-- Lingo syntax
put(_movie.framePalette)

// JavaScript syntax
put(_movie.framePalette);
```

次のステートメントは、パレットキャストメンバ "45" を現行フレームのパレットにします。

```
-- Lingo syntax
_movie.framePalette = 45

// JavaScript syntax
_movie.framePalette = 45;
```

関連事項

ムービー

frameRate

使用法

```
-- Lingo syntax
memberObjRef.frameRate

// JavaScript syntax
memberObjRef.frameRate;
```

解説

指定したデジタルビデオの再生フレームレートを指定するキャストメンバプロパティ、または Flash ムービーキャストメンバです。

デジタルビデオメンバのフレームレートに設定できる値は、デジタルビデオ再生オプションを選択するラジオボタンに対応して変化します。

- frameRate メンバプロパティが 1 ～ 255 に設定されている場合、デジタルビデオムービーは各フレームをそのフレームレートで再生します。frameRate メンバプロパティに 255 より大きい値を設定することはできません。全フレームをそのフレームレートで再生します。
- frameRate メンバプロパティが -1 または 0 に設定されている場合、デジタルビデオムービーは各フレームをムービーに設定されている標準レートで再生します。これにより、ビデオとサウンドトラックとを同期させることができます。frameRate を -1 または 0 以外に設定すると、デジタルビデオのサウンドトラックは再生されません。
- frameRate メンバプロパティが -2 に設定されている場合、デジタルビデオムービーは各フレームを最高速度で再生します。

Flash ムービーキャストメンバの場合、プロパティは Flash で作成されたムービーのフレームレートを示します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、QuickTime デジタルビデオキャストメンバ " 回転椅子 " のフレームレートを 30 フレーム / 秒に設定します。

```
-- Lingo syntax
member("Rotating Chair").frameRate = 30

// JavaScript syntax
member("Rotating Chair").frameRate = 30;
```

次のステートメントは、QuickTime デジタルビデオキャストメンバ " 回転椅子 " の全フレームを最高速度で再生します。

```
-- Lingo syntax
member("Rotating Chair").frameRate = -2

// JavaScript syntax
member("Rotating Chair").frameRate = -2;
```

次のスプライトスクリプトは、スプライトのキャストメンバが Flash で最初に作成されたときのフレームレートが 15 フレーム / 秒より小さいかどうかを調べます。ムービーのフレームレートが 15 フレーム / 秒より小さい場合、スクリプトは、他のレートに設定できるようにスプライトに playBackMode プロパティを設定します。次にスクリプトはスプライトの fixedRate プロパティを 15 フレーム / 秒に設定します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.frameRate < 15 then
        sprite(spriteNum).playBackMode = #fixed
        sprite(spriteNum).fixedRate = 15
    end if
end

// JavaScript syntax
function beginSprite () {
    var fr = sprite(this.spriteNum).member.frameRate;
    if (fr < 15) {
        sprite(this.spriteNum).playBackMode = symbol("fixed");
        sprite(this.spriteNum).fixedRate = 15;
    }
}
```

関連事項

fixedRate, playRate（QuickTime、AVI、MP4、FLV）, currentTime (QuickTime, AVI), playBackMode

frameRate (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.frameRate

// JavaScript syntax
dvdObjRef.frameRate;
```

解説

DVD プロパティ。DVD の frameRate 値を返します。読み取り専用。

次のいずれかの浮動小数が frameRate 値として返されます。

浮動小数	解説
0.0	frameRate 値がタイトルドメインにないか、タイトルが 1 つの連続したビデオタイトルでないため、値を特定できませんでした。
25.0	DVD は 25 フレーム / 秒で再生するようにオーサリングされます。
30.0	DVD は 30 フレーム / 秒で再生するようにオーサリングされます。
29.97	DVD は 29.97 フレーム / 秒で再生するようにオーサリングされます。

関連事項

DVD

frameRate (MP4Media/FLV)

使用法

```
put member(1).frameRate
```

解説

キャストメンバのプロパティ。指定した MP4Media/FLV キャストメンバの再生フレームレートを指定します。

例文

次の構文は、元のビデオ MP4Media/FLV キャストメンバのフレームレートを、1 秒間のフレーム数で表示します。

```
-- Lingo syntax
put member("MP4Media/FLV").frameRate

// JavaScript syntax
put member("MP4Media/FLV").frameRate;
```

frameScript

使用法

```
-- Lingo syntax
_movie.frameScript

// JavaScript syntax
_movie.frameScript;
```

解説

現行フレームに割り当てられたフレームスクリプトの一意のキャストメンバ番号を含むムービープロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

スコア生成セッション中には、frameScript プロパティを設定して現行フレームにフレームスクリプトを割り当てることもできます。

現行フレームにフレームスクリプトが割り当てられていない場合は、0 が返されます。

例文

次のステートメントは、現行フレームに割り当てられているスクリプト番号を表示します。この例では、スクリプト番号は 25 です。

```
-- Lingo syntax
put(_movie.frameScript)

// JavaScript syntax
put(_movie.frameScript);
```

次のステートメントは、スクリプトキャストメンバ " ボタンの反応 " を現行フレームのフレームスクリプトに設定します。

```
-- Lingo syntax
_movie.frameScript = member("Button responses")

// JavaScript syntax
_movie.frameScript = member("Button responses");
```

関連事項

ムービー

frameSound1

使用法

```
-- Lingo syntax
_movie.frameSound1

// JavaScript syntax
_movie.frameSound1;
```

解説

現行フレームの 1 つ目のサウンドチャネルに割り当てられているキャストメンバの番号を定義するためのムービープロパティです。読み取り / 書き込み可能。

また、このプロパティは、スコア録画セッション中に設定することもできます。

例文

スコア録画セッション中に、次のステートメントを使うと、サウンドキャストメンバ " ジャズ " を 1 つ目のサウンドチャネルに割り当てることができます。

```
-- Lingo syntax
_movie.frameSound1 = member("Jazz").number

// JavaScript syntax
_movie.frameSound1 = member("Jazz").number;
```

関連事項

frameSound2, ムービー

frameSound2

使用法

```
-- Lingo syntax
_movie.frameSound2

// JavaScript syntax
_movie.frameSound2;
```

解説

現行フレームの 2 つ目のサウンドチャネルに割り当てられているキャストメンバの番号を定義するためのムービープロパティです。読み取り / 書き込み可能。

また、このプロパティは、スコア録画セッション中に設定することもできます。

例文

スコア録画セッション中に、次のステートメントを使うと、サウンドキャストメンバ " ジャズ " を 2 つ目のサウンドチャネルに割り当てることができます。

```
-- Lingo syntax
_movie.frameSound2 = member("Jazz").number

// JavaScript syntax
_movie.frameSound2 = member("Jazz").number;
```

関連事項

frameSound1, ムービー

frameTempo

使用法

```
-- Lingo syntax
_movie.frameTempo

// JavaScript syntax
_movie.frameTempo;
```

解説

現行フレームに割り当てられているテンポを示すムービープロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

例文

次のステートメントは、現行フレームで使われているテンポを調べます。この例ではテンポが 15 フレーム / 秒になっています。

```
-- Lingo syntax
put(_movie.frameTempo)

// JavaScript syntax
put(_movie.frameTempo);
```

関連事項

ムービー , puppetTempo()

frameTransition

使用法

```
-- Lingo syntax
_movie.frameTransition

// JavaScript syntax
_movie.frameTransition;
```

解説

現行フレームに割り当てられているトランジションキャストメンバの番号を定義するためのムービープロパティです。スコア録画セッション中のみトランジションを指定用に読み取り / 書き込み可能。

例文

次のステートメントは、スコア録画セッションで現在 Lingo が録画しているフレームのトランジションにキャストメンバ " 霧 " を設定します。

```
-- Lingo syntax
_movie.frameTransition = member("Fog")

// JavaScript syntax
_movie.frameTransition = member("Fog");
```

関連事項

ムービー

front

使用法

```
member(whichCastmember).modelResource(whichModelResource).front
```

解説

3D #box モデルリソースプロパティ。ボックスの -Z 軸と交差する側面が、閉じた状態であるか (TRUE)、開いた状態であるか (FALSE) を示します。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" 木箱 " というモデルリソースの front プロパティを FALSE に設定します。つまり、このボックスの表面は開いた状態になります。

```
member("3D World").modelResource("Crate").front = FALSE
```

関連事項

back, bottom (3D), top (3D), left (3D), right (3D)

frontWindow

使用法

```
-- Lingo syntax
_player.frontWindow

// JavaScript syntax
_player.frontWindow;
```

解説

現在画面上で最前面にあるウィンドウ (MIAW) 内のムービーを調べるためのプレーヤプロパティです。読み取り専用。

ステージが最前面にある場合、frontWindow はステージになり、メディアエディタやフローティングパレットが手前にある場合は、frontWindow は VOID（Lingo）または null（JavaScript シンタックス）を返します。

例文

次のステートメントは、ウィンドウ " 音楽 " が最前面にあるかどうかを調べ、最前面にある場合は " テスト " ウィンドウを手前に表示します。

```
-- Lingo syntax
if (_player.frontWindow = "Music") then
    window("Try This").moveToFront()
end if

// JavaScript syntax
if (_player.frontWindow == "Music") {
    window("Try This").moveToFront();
}
```

関連事項

プレーヤ

fullScreen

使用法

```
-- Lingo syntax
dvdObjRef.fullScreen

// JavaScript syntax
dvdObjRef.fullScreen;
```

解説

DVD プロパティ。DVD をフルスクリーンモードで再生するかどうかを指定します。読み取り / 書き込み可能。

Esc キーを押すと、ディスプレイモードはフルスクリーン以外のモードに戻り、プロパティは false に設定されます。

Macintosh では、現在サポートされていません。

例文

次のステートメントは DVD の再生をフルスクリーンモードにします。

```
-- Lingo syntax
member("DVDMember").fullScreen = TRUE

// Java Script
member("DVDMember").fullScreen = true;
```

関連事項

DVD

getBoneID

使用法

```
memberReference.modelResource.getBoneID("boneName")
```

解説

3D モデルリソースプロパティ。モデルリソースのボーン名というボーンのインデックス番号を返します。指定した名前のボーンがない場合は、0 が返されます。

例文

次のステートメントは、ボーン "ShinL" の ID 番号を返します。

```
put member("ParkScene").modelResource("LittleKid").getBoneId("ShinL")
-- 40
```

関連事項

bone

globals

使用法

```
the globals
```

解説

VOID 以外の値を含む現行のグローバル変数すべての特殊なプロパティリストを含むシステムプロパティです。各グローバル変数はリストのプロパティで、2 つの値が割り当てられています。

globals では、次のリスト操作を実行できます。

- count() - リストのエントリ数を返します。
- getPropAt(n) - **n** 番目のエントリの名前を返します。
- getProp(x) - 指定した名前を持つエントリの値を返します。
- getAProp(x) - 指定した名前を持つエントリの値を返します。

注意：globals プロパティには、実行している Director のバージョンを示すプロパティ #version が自動的に格納されます。つまりグローバル変数がまだ定義されていなくても、リストには少なくとも 1 個のエントリが常にあります。

the globals プロパティは、メッセージウィンドウ以外の文脈でも使用できる点が showGlobals とは異なります。the globals をメッセージウィンドウに表示するには、showGlobals を使用します。

関連事項

showGlobals(), clearGlobals()

glossMap

使用法

```
member(whichCastmember).shader(whichShader).glossMap
member(whichCastmember).model(whichModel).shader.glossMap
member(whichCastmember).model(whichModel).shaderList{[index]}.glossMap
```

解説

3D #standard シェーダプロパティ。グロスマッピングで使用するテクスチャを指定します。

このプロパティを設定すると、次のプロパティが自動的に設定されます。

- シェーダの 4 番目のテクスチャレイヤーが、指定したテクスチャに設定されます。
- textureModeList[4] の値が #none に設定されます。
- blendFunctionList[4] の値が #multiply に設定されます。

例文

次のステートメントは、" 楕円 " というテクスチャを、"GlassBox" というモデルで使用されるシェーダの glossMap の値として設定します。

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.glossMap = member("3DPlanet").texture("Oval")

// Java Script
member("House").getPropRef("model", 1).shaderList[1].glossMap = member("House").getPropRef("texture",1);
```

関連事項

blendFunctionList, textureModeList, region, specularLightMap, diffuseLightMap

gravity

使用法

```
member(whichCastmember).modelResource(whichModelResource).gravity
```

解説

3D パーティクルモデルリソースプロパティ。#particle というタイプのモデルリソースを指定すると、リソースの gravity プロパティをベクトルで取得または設定できます。

このプロパティを使用すると、各シミュレーションステップ内のすべてのパーティクルに適用する重力を定義できます。

このプロパティのデフォルト値は vector(0,0,0) です。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の gravity プロパティを vector (0, -.1, 0) に設定します。"thermoSystem" のパーティクルは、y 軸の下方向に引っ張られます。

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").gravity = vector(0, -.1, 0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).gravity = vector(0, -.1, 0);
```

関連事項

drag, wind

gradientType

使用法

```
-- Lingo syntax
memberObjRef.gradientType

// JavaScript syntax
memberObjRef.gradientType;
```

解説

キャストメンバを塗りつぶすグラデーションの方式を設定するベクトルシェイプキャストメンバプロパティです。

#linear (線状) または #radial (放射状) のいずれかの値を設定します。gradientType は、fillMode が #gradient の場合にのみ有効です。

このプロパティは、値を調べたり、設定することができます。

例文

このハンドラは、キャストメンバ " バックドロップ " のグラデーションの方式が線状グラデーションならば、放射状グラデーションに切り替えます。また、放射状グラデーションならば線状グラデーションに切り替えます。

```
-- Lingo syntax
on mouseUp me
    if member("backdrop").gradientType = #radial then
        member("backdrop").gradientType = #linear
    else
        member("backdrop").gradientType = #radial
    end if
end

// JavaScript syntax
function mouseUp() {
    var gt = member("backdrop").gradientType;
    if (gt == "radial") {
        member("backdrop").gradientType = symbol("linear");
    } else {
        member("backdrop").gradientType = symbol("radial");
    }
}
```

関連事項

fillMode

group

使用法

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

解説

3D エレメント。名前、トランスフォーム、親、および子のプロパティのみを持つ 3D ワールドのノードです。

各 3D キャストメンバには、"World" というデフォルトのグループがあります。このグループは削除できません。3D ワールドに存在するモデル、ライト、カメラ、およびグループすべての親階層は、最後が group("world") で終わります。

例文

次の例の 1 行目はキャストメンバ "3D オブジェクト " の 2 番目のグループを示します。2 行目はキャストメンバ "3D オブジェクト " の "RB コレクション 01" グループを示します。

```
-- Lingo syntax
put member("3Dobjects").group("RBCollection01")
put member("3Dobjects").group[2]

// Javascript
put(member("3Dobjects").getPropRef("group",2));
```

関連事項

newGroup, deleteGroup, child (3D), parent

height

使用法

```
-- Lingo syntax
imageObjRef.height
memberObjRef.height
spriteObjRef.height

// JavaScript syntax
imageObjRef.height;
memberObjRef.height;
spriteObjRef.height;
```

解説

画像、メンバ、およびスプライトのプロパティ。ベクトルシェイプ、Flash、アニメーション GIF、RealMedia、Windows Media、ビットマップ、およびシェイプキャストメンバに対して使用し、キャストメンバがステージに描画されるときの高さをピクセル単位で示します。キャストメンバおよび画像オブジェクトについては読み取り専用、スプライトについては読み書き可能です。

例文

次のステートメントは、キャストメンバ " ヘッドライン " の高さを変数 vHeight に割り当てます。

```
-- Lingo syntax
vHeight = member("Headline").height

// JavaScript syntax
var vHeight = member("Headline").height;
```

次のステートメントは、スプライト 10 の高さを 26 ピクセルに設定します。

```
-- Lingo syntax
sprite(10).height = 26

// JavaScript syntax
sprite(10).height = 26;
```

関連事項

メンバ , スプライト , width

height (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).height
member(whichCastmember).texture(whichTexture).height
```

解説

3D #box モデルリソース、#cylinder モデルリソース、およびテクスチャプロパティ。オブジェクトの高さを示します。

#box または #cylinder モデルリソースの高さはワールド単位で測定され、値を調べたり、変更できます。このプロパティのデフォルト値は 50 です。

テクスチャの高さはピクセル単位で測定されます。このプロパティの値は調べられますが、設定できません。テクスチャの高さは、テクスチャのソースの高さから、最も近い 2 の累乗に丸められます。

例文

次のステートメントは、" タワー " というモデルリソースの高さを 225.0 ワールド単位に設定します。

```
member("3D World").modelResource("Tower").height = 225.0
```

次のステートメントは、" 火星の地図 " というテクスチャの高さが 512 ピクセルであることを示します。

```
put member("scene").texture("Marsmap").height
-- 512
```

関連事項

length (3D), width (3D)

height (MP4Media/FLV)

使用法

```
put member(1).height
Sprite(1).height = 30
```

解説

メンバおよびスプライトのプロパティ。MP4Media/FLV キャストメンバの場合、このメソッドは作成者が作成した元の MP4 ビデオのステージ上での高さを、ピクセルの単位で指定します。このプロパティは、MP4Media/FLV キャストメンバの場合は読み取り専用で、MP4Media/FLV スプライトの場合は読み書き可能です。

例文

次の例は、キャストメンバ MP4Media/FLV の高さを変数 vHeight に代入します。

```
-- Lingo syntax
vHeight = member("MP4Media/FLV").height

// JavaScript syntax
var vHeight = member("MP4Media/FLV").height;
```

次の例は、MP4Video スプライト 10 の高さを 26 ピクセルに設定します。

```
-- Lingo syntax
sprite(10).height = 26

// JavaScript syntax
sprite(10).height = 26;
```

heightVertices

使用法

```
member(whichCastmember).modelResource(whichModelResource).\heightVertices
```

解説

3D #box モデルリソースプロパティ。ボックスの高さに沿ったメッシュの頂点の数を示します。この値を大きくするとフェイスの数が増加し、メッシュの緻密さも増します。

ボックスの高さは、Y 軸に沿って測定されます。

モデルのシェーダの renderStyle プロパティを #wire に設定すると、メッシュのフェイスが表示されます。renderStyle プロパティを #point に設定すると、メッシュの頂点のみが表示されます。

このプロパティの値は 2 以上にする必要があります。デフォルト値は 4 です。

例文

次のステートメントは、" タワー " というモデルリソースの heightVertices プロパティを 10 に設定します。Z 軸に沿って、9 つのポリゴンを使ってモデルリソースのジオメトリを定義します。したがって、頂点の数は 10 になります。

```
member("3D World").modelResource("Tower").heightVertices = 10
```

関連事項

height (3D)

highlightPercentage

使用法

```
member(whichCastmember).model(whichModel).toon.highlightPercentage
member(whichCastmember).model(whichModel).shader.highlightPercentage
member(whichCastmember).shader(whichShader).highlightPercentage
```

解説

3D toon モディファイアおよび #painter シェーダプロパティ。ライトによってハイライトが生成されているモデルの、サーフェス領域で利用可能なカラーのパーセンテージを示します。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 50 です。

モデルの toon モディファイアおよび #painter シェーダが使用するカラーの数は、モデルの toon モディファイアまたは #painter シェーダの colorSteps プロパティによって決まります。

例文

次の例は、" 球体 " モデルの toon モディファイアの highlightPercentage プロパティを設定します。

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightPercentage = 25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightPercentage = 25;
```

関連事項

highlightStrength, brightness

highlightStrength

使用法

```
member(whichCastmember).model(whichModel).toon.highlightStrength
member(whichCastmember).model(whichModel).shader.highlightStrength
member(whichCastmember).shader(whichShader).highlightStrength
```

解説

3D toon モディファイアおよび #painter シェーダプロパティ。ライトによってハイライトが生成されているモデルの、サーフェイス領域の明度を示します。

このプロパティのデフォルト値は 1.0 です。

例文

次の例は、" 球体 " モデルの toon モディファイアの highlightStrength プロパティを設定します。

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightStrength = 0.25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightStrength = 0.25;
```

関連事項

highlightPercentage, brightness

hilite

使用法

```
-- Lingo syntax
memberObjRef.hilite

// JavaScript syntax
memberObjRef.hilite;
```

解説

ボタンツールで作成されたチェックボックスやラジオボタンが選択されている（TRUE）か、選択されていない（FALSE デフォルト）かを示すためのメンバプロパティです。読み取り / 書き込み可能。

例文

次のステートメントは、"Sound on" というボタンが選択されたかどうかを調べ、選択されている場合はチャネル 1 の音量を最大にします。

```
-- Lingo syntax
if (member("Sound On").hilite = TRUE) then
    sound(1).volume = 255
end if

// JavaScript syntax
if (member("Sound On").hilite == true) {
    sound(1).volume = 255;
}
```

次のステートメントは、ボタンキャストメンバ "powerSwitch" の hilite メンバプロパティを TRUE に設定することにより、このキャストメンバを選択します。

```
-- Lingo syntax
member("powerSwitch").hilite = TRUE

// JavaScript syntax
member("powerSwitch").hilite = true;
```

関連事項

メンバ

hinting

使用法

```
-- Lingo syntax
member("text").hinting

// JavaScript syntax
member("text").hinting;
```

解説

Text キャストメンバプロパティ。フォントヒンティングオプションを有効にします。このプロパティは次の値をとることができます。

- #Auto— ヒンティングにフォントファイルの情報を使用します。この値は、デフォルトで使用されます。
- #Algorithmic— ヒンティング操作にテキストエンジンを使用します。
- #TVMode— 文字を横または縦に対称に配置し、ヒンティングにテキストエンジンを使用します。
- #None— 現在のキャストメンバでヒンティングをオフにします。

例文

```
-- Lingo syntax
member("text").hinting = #Auto

// JavaScript syntax
member("text").hinting = symbol("Auto");
```

hither

使用法

```
member(whichCastmember).camera(whichCamera).hither
sprite(whichSprite).camera{(index)}.hither
```

解説

3D カメラプロパティ。カメラからモデルの描画を開始する位置までの距離をワールド単位で示します。hither よりもカメラに近いオブジェクトは描画されません。

このプロパティの値は 1.0 以上とする必要があります。デフォルト値は 5.0 です。

例文

次のステートメントは、カメラ 1 の hither プロパティを 1000 に設定します。カメラからの距離が 1000 ワールド単位未満のモデルは表示されません。

```
member("SolarSystem").camera[1].hither = 1000
```

関連事項

yon

hotSpot

使用法

```
-- Lingo syntax
memberObjRef.hotSpot

// JavaScript syntax
memberObjRef.hotSpot;
```

解説

[whichCursorCastMember] で示されるアニメーションカラーカーソルキャストメンバのホットスポットの座標をピクセル数で設定するカーソルキャストメンバプロパティです。Director はこの座標を使用して画面上のカーソルの位置を追跡します。例えば、mouseH や mouseV 関数の値を返すときにこの値を使用します。また、mouseEnter メッセージで示したロールオーバーが発生した位置を特定します。

カーソルの左上隅座標が (0,0) で、これが hotSpot のデフォルト値です。カーソル境界の外側のポイントを指定するとエラーになります。座標は (1,1) ではなく (0,0) が基準であるため、例えば、16 × 16 ピクセルのカーソルで、(16,16) をホットスポットに指定しようとするとエラーになります。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、キャストメンバが変数 cursorNum で示される 32 × 32 ピクセルのカーソルの中央に、ホットスポットを設定します。

```
-- Lingo syntax
on startMovie
    member(cursorNum).hotSpot = point(16,16)
end

// JavaScript syntax
function startMovie() {
    member(cursorNum).hotSpot = point(16,16);
}
```

hotSpotEnterCallback

使用法

```
-- Lingo syntax
spriteObjRef.hotSpotEnterCallback

// JavaScript syntax
spriteObjRef.hotSpotEnterCallback;
```

解説

ステージ上に見えている QTVR のホットスポットにカーソルが入ったときに起動するハンドラ名を設定する QuickTime VR スプライトプロパティです。QuickTime VR スプライトが、このメッセージを最初に受け取ります。メッセージは、次の 2 つのパラメータを受け取ります。1 つは me パラメータで、もう 1 つはカーソルが入ったホットスポットの ID です。

コールバックを解除するには、このプロパティに "0" を設定します。

処理性能を考慮し、コールバックプロパティは必要な場合にのみ設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次の例は、ステージ上に見えている QuickTime VR のホットスポットにカーソルが入ったときに起動するハンドラ名を示します。

```
-- Lingo syntax
put sprite("multinode").hotSpotEnterCallback

// Javascript
put(sprite("multinode").hotSpotEnterCallback);
```

関連事項

hotSpotExitCallback, nodeEnterCallback, nodeExitCallback, triggerCallback

hotSpotExitCallback

使用法

```
-- Lingo syntax
spriteObjRef.hotSpotExitCallback

// JavaScript syntax
spriteObjRef.hotSpotExitCallback;
```

解説

ステージ上に見えている QuickTime VR のホットスポットからカーソルが出たときに起動するハンドラ名を設定する QuickTime VR スプライトプロパティです。QuickTime VR スプライトが、このメッセージを最初に受け取ります。メッセージは、次の 2 つのパラメータを受け取ります。1 つは me パラメータで、もう 1 つはカーソルが入ったホットスポットの ID です。

コールバックを解除するには、このプロパティに "0" を設定します。

処理性能を考慮し、コールバックプロパティは必要な場合にのみ設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次の例は、ステージ上に見えている QuickTime VR のホットスポットからカーソルが出たときに起動するハンドラ名を示します。

```
-- Lingo syntax
put sprite("multinode").hotSpotExitCallback

// Javascript
put(sprite("multinode").hotSpotExitCallback);
```

関連事項

hotSpotEnterCallback, nodeEnterCallback, nodeExitCallback, triggerCallback

HTML

使用法

```
-- Lingo syntax
memberObjRef.HTML

// JavaScript syntax
memberObjRef.HTML;
```

解説

HTML 形式のテキストキャストメンバ内のテキストとそのテキストのレイアウトを制御するタグを設定するキャストメンバプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、テキストキャストメンバ " ホームページ " に埋めこまれている HTML 書式情報を、メッセージウィンドウに表示します。

```
--Lingo syntax
put(member("Home Page").HTML)

// JavaScript syntax
trace(member("Home Page").HTML);
```

関連事項

importFileInto(), RTF

hyperlink

使用法

```
-- Lingo syntax
chunkExpression.hyperlink

// JavaScript syntax
chunkExpression.hyperlink;
```

解説

テキストキャストメンバのチャンク式とハイパーリンクストリングを返す、テキストキャストメンバプロパティです。

このプロパティは、値を調べたり、変更できます。

このプロパティを検索すると、[チャンク式] の最初の文字を含むリンクが使用されます。

ハイパーリンクはオーバーラップできません。既存のリンクに一部でもオーバーラップするようなリンクを設定しようとすると、既存のリンクは新しいリンクに置き換えられます。

ハイパーリンクを空ストリングに設定すると、削除されます。

例文

次のハンドラは、テキストキャストメンバ "MacroLink" の最初の単語にハイパーリンクを設定します。テキストは、Adobe の Web サイトにリンクされます。

```
--Lingo syntax
on startMovie
    member("MacroLink").word[1].hyperlink = "http://www.adobe.com"
end

// JavaScript syntax
function startMovie() {
    member("MacroLink").getPropRef("word", 1).hyperlink = "http://www.adobe.com";
}
```

関連事項

hyperlinkRange, hyperlinkState

hyperlinkRange

使用法

```
-- Lingo syntax
chunkExpression.hyperlinkRange

// JavaScript syntax
chunkExpression.hyperlinkRange;
```

解説

[チャンク式] の最初の文字を含むハイパーリンクの範囲を返すテキストキャストメンバプロパティです。

このプロパティは、値を調べられますが、設定はできません。

hyperLink や hyperLinkState と同様、[チャンク式] の最初の文字を含むリンクの範囲が返されます。

例文

次の例は、[チャンク式] の最初の文字を含むハイパーリンクの範囲を表示します。

```
-- Lingo syntax
put member("MyText").hyperlinkRange

// Javascript
put(member("MyText").hyperlinkRange);
```

関連事項

hyperlink, hyperlinkState

hyperlinks

使用法

```
-- Lingo syntax
chunkExpression.hyperlinks

// JavaScript syntax
chunkExpression.hyperlinks;
```

解説

テキストキャストメンバの指定されたチャンクのハイパーリンク範囲が含まれるリニアリストを返す、テキストキャストメンバプロパティです。各範囲は、リストの最初の文字と最後の文字の位置を表す、エレメント数 2 のリニアリストとして示されます。

例文

次のステートメントは、テキストキャストメンバ "Glossary" のすべてのリンクを、メッセージウィンドウに表示します。

```
--Lingo syntax
put(member("Glossary").hyperlinks) -- [[3, 8], [10, 16], [41, 54]]

// JavaScript syntax
trace(member("Glossary").hyperlinks); // [[3, 8], [10, 16], [41, 54]]
```

hyperlinkState

使用法

```
-- Lingo syntax
chuckExpression.hyperlinkState

// JavaScript syntax
chuckExpression.hyperlinkState;
```

解説

ハイパーリンクの現在の状態を示すテキストキャストメンバプロパティです。状態を示す値は、#normal、#active、#visited のいずれかです。

このプロパティは、値を調べたり、設定することができます。

hyperLink や hyperLinkRange と同様、[チャンク式] の最初の文字を含むリンクの範囲が返されます。

例文

次のハンドラは、クリックされたハイパーリンクが Web アドレスかどうかを調べます。Web アドレスであった場合は、hyperlinkTextState を #visited に設定し、ムービーをその Web アドレスに分岐します。

```
--Lingo syntax
property spriteNum

on hyperlinkClicked me, data, range
    if data starts "http://" then
        currentMember = sprite(spriteNum).member
        currentMember.word[4].hyperlinkState = #visited
        gotoNetPage(data)
    end if
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st == ht) {
        currentMember = sprite(spriteNum).member;
        currentMember.getPropRef("word", 4).hyperlinkState = symbol("visited");
        gotoNetPage(data);
    }
}
```

関連事項

hyperlink, hyperlinkRange

idleHandlerPeriod

使用法

```
-- Lingo syntax
_movie.idleHandlerPeriod

// JavaScript syntax
_movie.idleHandlerPeriod;
```

解説

ムービーが idle メッセージを送るまでの最大チック数を定義するムービープロパティです。読み取り / 書き込み可能。

デフォルト値 1 の場合、1 秒間に 60 回 idle ハンドラメッセージが送られます。

再生ヘッドがフレームに入ると、Director はタイマをスタートさせ、必要なスプライトをステージ上で再描画してから enterFrame イベントを発行します。この時点でテンポ設定で指定されている時間が経過している場合、Director は exitFrame イベントを生成し、次に指定されたフレームに移行します。このフレームに設定されている時間が経過していない場合、Director は時間が過ぎるまで待ちます。この待ち時間の間、Director は定期的に idle メッセージを生成します。この idle イベント間の間隔は、idleHandlerPeriod プロパティによって定義されます。

idleHandlerPeriod に設定できる値は、次のとおりです。

- 0 - 可能な限り頻繁に idle イベントを生成
- 1 - 1 秒間に 60 回以下
- 2 - 1 秒間に 30 回以下
- 3 - 1 秒間に 20 回以下

- **n** - 1 秒間に 60/**n** 回以下

フレームあたりの idle イベントの回数は、ムービーのフレームレートの他、スクリプトが実行されているかなど、他の状況によっても変化します。テンポが 1 秒あたり 60 フレーム (60 fps) で、idleHandlerPeriod の値が 1 の場合は、フレームあたり 1 回の idle イベントが発生します。テンポが 20 fps の場合、フレームあたり 3 回の idle イベントが発生することになります。アイドル時間は、実行する現行タスクがない時間に生じるものなので、イベントを生成することはできません。

反対に、idleHandlerPeriod プロパティが 0 で、テンポが非常に遅い場合は、莫大な回数の idle イベントが発生します。

このプロパティのデフォルト値は 1 です。

例文

次のステートメントは、1 秒あたり最大 1 回の idle メッセージをムービーに送るように設定します。

```
-- Lingo syntax
_movie.idleHandlerPeriod = 60

// JavaScript syntax
_movie.idleHandlerPeriod = 60;
```

関連事項

on idle, idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, ムービー

idleLoadMode

使用法

```
-- Lingo syntax
_movie.idleLoadMode

// JavaScript syntax
_movie.idleLoadMode;
```

解説

preLoad() メソッドと preLoadMember() メソッドが、アイドル中のどのタイミングでキャストメンバの読み込みを行うかを定義するムービープロパティです。読み取り / 書き込み可能。

アイドル間隔は、次のいずれかの値になります。

- 0 - アイドル中の読み込み、つまりアイドルロードは行いません。
- 1 - フレーム間に空き時間があるときに読み込みを実行します。
- 2 - idle イベントの発生中に読み込みを実行します。
- 3 - 可能な限り頻繁に読み込みを実行します。

idleLoadMode プロパティを設定するだけでは何も実行されません。このプロパティは preLoad() および preLoadMember() メソッドと共に使用したときのみ有効になります。

アイドル中に読み込まれたキャストメンバは、ムービーが使うまでは圧縮されたままになっています。ムービーの再生が始まる前に、キャストメンバを解凍する間しばらく停止することがあります。

例文

次のステートメントは、preLoad または preLoadMember コマンドによってプリロードが実行されるときに、キャストメンバの読み込みを可能な限り頻繁に試みるように指示します。

```
-- Lingo syntax
_movie.idleLoadMode = 3

// JavaScript syntax
_movie.idleLoadMode = 3;
```

関連事項

on idle, ムービー , preLoad() (Movie), preLoadMember()

idleLoadPeriod

使用法

```
-- Lingo syntax
_movie.idleLoadPeriod

// JavaScript syntax
_movie.idleLoadPeriod;
```

解説

アイドルロード待ちのキャストメンバの読み込みを試みるまでに、Director が待つ時間をチック数（1/60 秒）で定義するムービープロパティです。読み取り / 書き込み可能。

idleLoadPeriod がデフォルト値の 0 の場合、Director は可能な限り頻繁に読み込もうとします。

例文

次のステートメントは、キャストメンバのアイドルロードを 1/2 秒 (30 チック) おきに試みるように Director に指示します。

```
-- Lingo syntax
_movie.idleLoadPeriod = 30

// JavaScript syntax
_movie.idleLoadPeriod = 30;
```

関連事項

on idle, ムービー

idleLoadTag

使用法

```
-- Lingo syntax
_movie.idleLoadTag

// JavaScript syntax
_movie.idleLoadTag;
```

解説

このムービープロパティを使用することにより、キャストメンバをアイドルロードするときに、ロードを開始したキャストメンバに任意のタグを設定することができます。読み取り / 書き込み可能。

idleLoadTag プロパティは、プリロードするグループ内のキャストメンバを識別するときに使用でき、値には任意の番号を設定できます。

例文

次のステートメントはアイドルロードタグとして 10 を設定します。

```
-- Lingo syntax
_movie.idleLoadTag = 10

// JavaScript syntax
_movie.idleLoadTag = 10;
```

関連事項

on idle, ムービー

idleReadChunkSize

使用法

```
-- Lingo syntax
_movie.idleReadChunkSize

// JavaScript syntax
_movie.idleReadChunkSize;
```

解説

Director がロード待ちのキャストメンバを読み込むときに、一度に読み込む最大バイト数を定義する ムービープロパティです。読み取り / 書き込み可能。

idleReadChunkSize のデフォルト値は 32 KB です。

例文

次のステートメントは、Director がロード待ちのキャストメンバを読み込むときに、一度に読み込む最大サイズを約 500 KB に設定します。

```
-- Lingo syntax
_movie.idleReadChunkSize = (500 * 1024)

// JavaScript syntax
_movie.idleReadChunkSize = (500 * 1024);
```

関連事項

on idle, ムービー

image （画像）

使用法

```
-- Lingo syntax
imageObjRef.image

// JavaScript syntax
imageObjRef.image;
```

解説

画像のプロパティ。ステージまたはウィンドウのビットマップまたはテキストキャストメンバの画像オブジェクトを参照します。キャストメンバの画像では読み取り / 書き込み可能、ステージ またはウィンドウの画像の場合は読み取り専用。

キャストメンバの image プロパティを設定すると、メンバの内容がただちに変更されます。ただし、メンバまたはウィンドウの画像を調べる場合は、指定したメンバまたはウィンドウの画像への参照が作成されます。ウィンドウに変更を加えると、キャストメンバまたはウィンドウの内容がただちに変更されます。

画像プロパティに多数の変更を加える場合は、duplicate() メソッドを使用して image プロパティを新規の画像オブジェクトにコピーし、変更内容を新規の画像オブジェクトに適用し、もとの画像を新規の画像オブジェクトに設定する方が高速です。非ビットマップメンバの場合は、duplicate() メソッドを使用した方が常に高速です。

例文

次のステートメントは、キャストメンバ "originalFlower" の画像をキャストメンバ "newFlower" に設定します。

```
-- Lingo syntax
member("newFlower").image = member("originalFlower").image

// JavaScript syntax
member("newFlower").image = member("originalFlower").image;
```

次のステートメントは、ステージの画像への参照を変数 myImage に格納し、その画像をキャストメンバ Flower に設定します。

```
-- Lingo syntax
myImage = _movie.stage.image
member("flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
member("flower").image = myImage;
```

関連事項

copyPixels(), draw(), duplicate() （画像）, fill(), image(), setPixel()

image (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.image

// JavaScript syntax
memberOrSpriteObjRef.image;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。RealMedia ビデオストリームの現行フレームを含む Lingo 画像オブジェクトを返します。このプロパティを使用すると、RealVideo® を 3D モデル上にマッピングできます（以下の例を参照してください）。

例文

次のステートメントは、RealMedia キャストメンバ "Real" の現在のフレームをビットマップキャストメンバ "Still" にコピーします。

```
-- Lingo syntax
member("Still").image = member("Real").image

// JavaScript syntax
member("Still").image = member("Real").image;
```

image (MP4Media/FLV)

使用法

```
img = sprite(1).image
img = member(1).image
```

解説

再生中の MP4Media/FLV メンバまたはスプライトの現在のフレームの Lingo 画像オブジェクトを返す、MP4Media/FLV メンバまたはスプライトプロパティです。

例文

次の例は、MP4 ビデオの currentFrame の Lingo 画像オブジェクトを返します。

```
-- Lingo syntax
img = sprite(1).image
img = member(1).image

// JavaScript syntax
img = sprite(1).image;
img = member(1).image;
```

image （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.image

// JavaScript syntax
windowObjRef.image;
```

解説

ウィンドウの画像オブジェクトを参照するウィンドウプロパティです。読み取り専用。

ウィンドウの画像を調べる場合は、指定したウィンドウの画像への参照が作成されます。画像に変更を加えると、ウィンドウの内容がただちに変更されます。

image プロパティに多数の変更を加える場合は、メンバオブジェクトの duplicate() メソッドを使用して image プロパティを新規の画像オブジェクトにコピーし、変更内容を新規の画像オブジェクトに適用し、元の画像を新規の画像オブジェクトに設定する方が高速です。非ビットマップメンバの場合は、duplicate() メソッドを使用した方が常に高速です。

例文

次のステートメントは、ステージの画像への参照を変数 myImage に格納し、その画像を "Flower" というウィンドウに設定します。

```
-- Lingo syntax
myImage = _movie.stage.image
window("Flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
window("Flower").image = myImage;
```

関連事項

duplicate()（メンバ）, ウィンドウ

imageCompression

使用法

```
-- Lingo syntax
_movie.imageCompression
memberObjRef.imageCompression

// JavaScript syntax
_movie.imageCompression;
memberObjRef.imageCompression;
```

解説

ムービーおよびビットマップキャストメンバのプロパティ。Shockwave 形式でムービーを保存する際に内蔵（リンクされていない）ビットマップメンバに適用される圧縮方式を示します。読み取り / 書き込み可能。

imageCompression の有効な値は、以下のとおりです。

値	意味
#standard	Director 標準の内部圧縮フォーマットを使用します。
#movieSetting	_movie.imageCompression プロパティに格納されている、ムービーの圧縮設定を使用しま す。これは、標準圧縮に限定されない画像フォーマットのデフォルト値です。
#jpeg	JPEG 圧縮を使用します。imageQuality 参照。

通常、このプロパティは Director の [パブリッシュ設定] ダイアログボックスで設定します。

例文

次のステートメントは、現在再生中のムービーに適用されている imageCompression をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.imageCompression)

// JavaScript syntax
put(_movie.imageCompression);
```

関連事項

imageQuality, ムービー

imageEnabled

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.imageEnabled

// JavaScript syntax
memberOrSpriteObjRef.imageEnabled;
```

解説

Flash ムービーまたはベクトルシェイプの画像が可視（TRUE、デフォルト）か不可視（FALSE）かを設定するキャストメンバプロパティとスプライトプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次の beginSprite スクリプトは、リンクされた Flash ムービーが最初にステージに表示されたときにその画像を隠し、メモリへのストリーミングを開始します。そして、後でスコアのフレームスクリプトで使えるように、スプライト番号をグローバル変数 StreamingSprite に保存します。

```
-- Lingo syntax
global gStreamingSprite

on beginSprite me
    gStreamingSprite = me.spriteNum
    sprite(gStreamingSprite).imageEnabled = FALSE
end

// JavaScript syntax
function beginSprite() {
    _global.gStreamingSprite = this.spriteNum;
    sprite(_global.gStreamingSprite).imageEnabled = 0;
}
```

次のフレームスクリプトは、前のスクリプトの実行後、グローバル変数 StreamingSprite で示されている Flash ムービースプライトのメモリへのストリーミングが 100% 完了しているかどうかを調べます。完了していない場合は、ムービーのメモリへのストリーミングが完了するまで、再生ヘッドを現行フレームでループさせます。次に、imageEnabled プロパティを TRUE に設定して画像を表示し、再生ヘッドをスコアの次のフレームから続行させます。

```
-- Lingo syntax
global gStreamingSprite

on exitFrame me
    if sprite(gStreamingSprite).member.percentStreamed < 100 then
        _movie.go(_movie.frame)
    else
        sprite(gStreamingSprite).imageEnabled = TRUE
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame() {
    var stmSp = sprite(_global.gStreamingSprite).member.percentStreamed;
    if (stmSp < 100) {
        _movie.go(_movie.frame);
    } else {
        sprite(_global.gStreamingSprite).imageEnabled = 1;
        _movie.updatestage();
    }
}
```

imageQuality

使用法

```
-- Lingo syntax
_movie.imageQuality
memberObjRef.imageQuality

// JavaScript syntax
_movie.imageQuality;
memberObjRef.imageQuality;
```

解説

ムービーおよびビットマップキャストメンバのプロパティ。ムービーの imageCompression プロパティを #jpeg に設定する場合に、使用する圧縮レベルを指定します。オーサリング中のみ読み取り / 書き込み可能。

取り得る値の範囲は 0 ～ 100 です。0 の場合は画像画質が最低で圧縮率が最高となり、100 の場合は画像画質が最高で圧縮率が最低となります。

このプロパティはオーサリング中のみ設定でき、ムービーが Shockwave Player 形式で保存されるまで影響がありません。

例文

次のステートメントは、現在再生中のムービーに適用されている imageQuality をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.imageQuality)

// JavaScript syntax
put(_movie.imageQuality);
```

関連事項

imageCompression, ムービー

immovable

使用法

```
member(whichCastmember).model(whichModel).collision.immovable
```

解説

3D #collision モディファイアプロパティ。アニメーション中に、衝突の結果としてモデルを移動させるかどうかを示します。TRUE を指定するとモデルは固定され、FALSE を指定するとモデルは移動可能になります。移動しないモデルについては Lingo で衝突を確認する必要がないため、このプロパティはアニメーション中のパフォーマンスを向上させる方法として有効です。

このプロパティのデフォルト値は FALSE です。

例文

次のステートメントは、"3D オブジェクト " というキャストメンバの 2 番目のモデルに割り当てられた collision モディファイアの immovable プロパティを TRUE に設定します。

```
-- Lingo syntax
member("3Dobjects").model[2].collision.immovable = TRUE

// Javascript
member("3Dobjects").getPropRef("model",2).collision.immovable = 1;
```

関連事項

collision (modifier)

ink

使用法

```
-- Lingo syntax
spriteObjRef.ink

// JavaScript syntax
spriteObjRef.ink;
```

解説

スプライトに適用するインク効果を定義するスプライトプロパティです。読み取り / 書き込み可能。

ink の有効な値は次のとおりです。

0Copy—	32Blend—
1Transparent—	33Add pin—
2Reverse—	34Add—
3Ghost—	35Subtract pin—
4Not copy—	36Background transparent —
5Not transparent—	37Lightest—
6Not reverse—	38Subtract—

7Not ghost—	39Darkest—
8Matte—	40Lighten—
9Mask—	41Darken—

36 （Background transparent）インクの場合は、スコア内でスプライトを選択しておき、ツールウィンドウ内のバックカラーチップを使って透明になる色を設定します。backColor プロパティを設定するという方法もあります。

再生ヘッドが動いていないときにスクリプト内でこのプロパティを設定する場合は、ムービーオブジェクトの updateStage() メソッドを使ってステージを描き直す必要があります。複数のスプライトプロパティまたはスプライトを変更する場合は、すべての変更を加えた後に updateStage() メソッドを一度だけ実行します。

例文

次のステートメントは、スプライト（3）のインク効果の値を変数 currentInk に設定します。

```
-- Lingo syntax
currentInk = sprite(3).ink

// JavaScript syntax
var currentInk = sprite(3).ink;
```

次のステートメントは、sprite プロパティのインク効果を 8 に設定してスプライト（i + 1）に matte インク効果を設定します。

```
-- Lingo syntax
sprite(i + 1).ink = 8

// JavaScript syntax
sprite(i + 1).ink = 8;
```

関連事項

backColor, スプライト , updateStage()

inker （モディファイア）

使用方法

```
member(whichCastmember).modelResource(whichModelResource).inker.inkerModifierProperty
modelResourceObjectReference.inker.inkerModifierProperty
```

解説

3D モディファイア。addModifier を使用してモデルリソースに #inker モディファイアを追加すると、#inker モディファイアプロパティを取得および設定できます。

#inker モディファイアは、既存のモデルにシルエット、折り目、および境界のエッジを追加します。#inker プロパティを使用すると、これらの プロパティの定義と強調を制御できます。

#inker モディファイアを #toon モディファイアと共に使用すると、レンダリング効果は累積され、最初に適用されたモディファイアに応じて変化します。modifier プロパティによって返されるモディファイアのリストには、#inker または #toon のうち最初に追加されたほうが含められます。両方が含められることはありません。#inker モディファイアは、#sds モディファイアと組み合わせて使用することはできません。

#inker モディファイアには以下のプロパティがあります。

- lineColor。インカーによって描画される線のカラーを取得または設定できます。

- silhouettes。モデルの輪郭をなぞるように境界に沿ってエッジを定義する線を描画するかどうかを取得または設定できます。
- creases。折り目で線を描画するかどうかを取得または設定できます。
- creaseAngle を使用すると、インカーの折り目の角度を検出する感度を取得または設定できます。
- boundary。サーフェスの境界線を描画するかどうかを取得または設定できます。
- lineOffset を使用すると、シェードされたサーフェスおよびカメラを基準に線を描画するかどうかを取得または設定できます。
- useLineOffset。lineOffset がオンかオフかを取得または設定できます。

注意：以下のプロパティの詳細については、プロパティ個別のエントリを参照してください。

例文

次のステートメントは inker モディファイアを "3Dobjects" というキャストメンバの 2 番目のモデルに追加します。addModifier を使用してモデルリソースに #inker モディファイアを追加すると、#inker モディファイアプロパティを取得および設定できます。

```
-- Lingo syntax
member("3Dobjects").model[2].addModifier(#inker)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).addModifier(symbol("inker"));
```

関連事項

addModifier, modifiers, toon （モディファイア）, shadowPercentage

inlineImeEnabled

使用法

```
-- Lingo syntax
_player.inlineImeEnabled

// JavaScript syntax
_player.inlineImeEnabled;
```

解説

Director 11 では、inlineImeEnabled プロパティは常に TRUE であり、スクリプトを使用して TRUE または FALSE に設定しても変わりません。このプロパティは前のバージョンとの互換性のためだけに残されています。

例文

次のステートメントは、プレーヤの inlineImeEnabled プロパティを TRUE に設定します。

```
-- Lingo syntax
_player.inlineImeEnabled=TRUE

// JavaScript syntax
_player.inlineImeEnabled=1;
```

関連事項

プレーヤ

interval

使用法

```
-- Lingo syntax
memberObjRef.interval

// JavaScript syntax
memberObjRef.interval;
```

解説

[カーソルキャストメンバ] で示されるアニメーションカラーカーソルの各フレーム間の間隔を、ミリ秒 (ms) 単位で指定するカーソルキャストメンバプロパティです。-1 を指定すると、静止したカーソル、つまりアニメーションカーソルの先頭フレームのみを表示します。これはアニメーションを停止する効果があります。デフォルトは 100 ミリ秒です。

カーソルの間隔は、テンポチャネルに使うムービーのフレームレートや、puppetTempo コマンドの設定とは独立しています。

このプロパティは、値を調べたり、設定することができます。

例文

次のスプライトスクリプトでは、" 蝶 " というキャストメンバに格納されたアニメーションカラーカーソルがスプライトに入ると、間隔が 50 ミリ秒に設定されアニメーションの速度が上がります。カーソルがスプライトから離れると、間隔が再度 100 ミリ秒に設定されアニメーションの速度は低下します。

```
-- Lingo syntax
on mouseEnter
    member("Butterfly").interval = 50
end

on mouseLeave
    member("Butterfly").interval = 100
end

// JavaScript syntax
function mouseEnter() {
    member("Butterfly").interval = 50;
}

function mouseLeave() {
    member("Butterfly").interval = 100;
}
```

invertMask

使用法

```
-- Lingo syntax
memberObjRef.invertMask

// JavaScript syntax
memberObjRef.invertMask;
```

解説

Director が QuickTime ムービーを描画する場合に、白いピクセルのマスクで描画する（TRUE）か、黒いピクセルで描画する（FALSE、デフォルト）かを設定する QuickTime キャストメンバプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、" 星の爆発 " という QuickTime ムービーの invertMask プロパティの設定を反転します。

```
-- Lingo syntax
on toggleMask
    member("Starburst").invertMask = not(member("Starburst").invertMask)
end

// JavaScript syntax
function toggleMask() {
    member("Starburst").invertMask = !(member("Starburst").invertMask);
}
```

関連事項

mask

isPlayable (MP4Media/FLV)

使用法

```
member("MP4Media").isPlayable = true
member("FLV").isPlayable = true
sprite("MP4Media").isPlayable = true
sprite("FLV").isPlayable = true
```

解説

Director でサポートされるオーディオまたはビデオコンテンスが MP4Media/FLV エレメントに含まれるかどうかを示すブール値を返す、MP4Media/FLV メンバまたはスプライトのプロパティです。このプロパティは読み取り専用です。

例文

次の例は、スプライト 1 の MP4Media エレメントまたは FLV ビデオを Director で再生できるかどうかを示します。

```
-- Lingo syntax
put(sprite(1).isPlayable) -- 1
put(member(1).isPlayable) -- 1

// JavaScript syntax
put(sprite(1).isPlayable); // 1
put(member(1).isPlayable); // 1
```

isSaving （ミクサ）

使用法

```
mixer.isSaving(ReadOnly)
```

解説

オーディオミクサプロパティ。ミクサ出力が保存されているときは 1 を返し、それ以外は 0 を返します。

注意：startSave() メソッドと stopSave() メソッドでは、このプロパティを使用しないでください。

例文

```
-- Lingo syntax
on exitFrame me
    put mixerRef.issaving --Returns 0 if the mixer associated with mixerRef is not being
-- saved. Returns 1 otherwise.
end

// JavaScript syntax
function exitframe(){
put mixerRef.isSaving) ; //Returns 0 if the mixer associated with mixerRef is not being
// saved. Returns 1 otherwise.

}
```

関連事項

ミクサ

isSaving （サウンドオブジェクト）

使用法

```
soundObj.isSaving(ReadOnly)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトが保存されているときは 1 を返し、それ以外は 0 を返します。

注意：startSave() メソッドと stopSave() メソッドでは、このプロパティを使用しないでください。

例文

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.issaving -- Returns 0 if the sound object associated
-- with soundobjectRef is not being saved. Returns 1 otherwise.
end

// JavaScript syntax
function exitframe(){
put (soundObjRef.isSaving) ; // Returns 0 if the sound object associated
// with soundobjectRef is not being saved. Returns 1 otherwise.

}
```

isVRMovie

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.isVRMovie

// JavaScript syntax
memberOrSpriteObjRef.isVRMovie;
```

解説

キャストメンバやスプライトが、まだダウンロードされていない QuickTime VR ムービーである（TRUE）か、QuickTime VR ムービーでない（FALSE）かを示す QuickTime キャストメンバおよびスプライトプロパティです。

タイプが #quickTimeMedia 以外のアセットでこのプロパティを調べると、エラーメッセージが表示されます。

このプロパティは、値を調べられますが、設定はできません。

例文

このハンドラは、スプライトのメンバが QuickTime ムービーであるかどうかを調べます。QuickTime ムービーの場合は、QTVR ムービーであるかどうかをさらに調べます。どのような場合でもアラートが表示されます。

```
-- Lingo syntax
on checkForVR(theSprite)
    if sprite(theSprite).member.type = #quickTimeMedia then
        if sprite(theSprite).isVRMovie then
            _player.alert("This is a QTVR asset.")
        else
            _player.alert("This is not a QTVR asset.")
        end if
    else
        _player.alert("This is not a QuickTime asset.")
    end if
end

// JavaScript syntax
function checkForVR(theSprite) {
    var memType = sprite(theSprite).member.type;
    if (memType == "quickTimeMedia") {
        var isType = sprite(theSprite).isVRMovie;
        if (isType == 1) {
            _player.alert("This is a QTVR asset.");
        } else {
            _player.alert("This is not a QTVR asset.");
        } else {
            _player.alert("This is not a QuickTime asset.");
        }
    }
}
```

itemDelimiter

使用法

```
_player.itemDelimiter
```

解説

項目を区切るために使用する特殊文字を規定するプレーヤプロパティです。

itemDelimiter を Windows の場合は円記号（\）、Macintosh の場合にはコロン（:）に設定すると、itemDelimiter を使用してファイルのパス名を解析できます。通常の操作では itemDelimiter をカンマ (,) に戻します。

このプロパティは、値を調べたり、変更できます。

例文

次のハンドラは、Macintosh でのパス名の最後の要素を調べます。ハンドラは、最初に現在の区切り文字を記憶してからコロン (:) に変更します。コロンが区切り文字として使われている場合は、Lingo は the last item of を使って Macintosh のパス名を構成するチャンク内の最後の項目を認識します。ハンドルを抜け出す前に、区切り文字が元の値にリセットされます。

```
on getLastComponent pathName
    save = _player.itemDelimiter
    _player.itemDelimiter = ":"
    f = the last item of pathName
    _player.itemDelimiter = save
    return f
end
```

関連事項

プレーヤ

kerning

使用法

```
-- Lingo syntax
memberObjRef.kerning

// JavaScript syntax
memberObjRef.kerning;
```

解説

[テキストメンバ] の内容が変わったときに、自動的にカーニングするかどうかを設定するテキストキャストメンバプロパティです。

TRUE に設定すると自動的にカーニングが行われ、FALSE に設定するとカーニングは行われません。

このプロパティのデフォルトは TRUE です。

例文

次のステートメントはテキストキャストメンバの kerning プロパティを TRUE に設定します。

```
-- Lingo syntax
member("MyText").kerning=TRUE

// JavaScript syntax
member("MyText").kerning=1;
```

関連事項

kerningThreshold

kerningThreshold

使用法

```
-- Lingo syntax
memberObjRef.kerningThreshold

// JavaScript syntax
memberObjRef.kerningThreshold;
```

解説

テキストキャストメンバ内で自動的にカーニングを実行するフォントサイズを設定するテキストキャストメンバプロパティです。この設定は、テキストキャストメンバの kerning プロパティが TRUE の場合にのみ有効です。

プロパティの値は、カーニングを行う最小のポイントサイズを表す整数値です。

このプロパティのデフォルトは 14 ポイントです。

例文

次のステートメントはテキストキャストメンバの kerningThreshold プロパティのフォントサイズを 14 に設定します。

```
-- Lingo syntax
member("MyText").kerningThreshold=14

// JavaScript syntax
member("MyText").kerningThreshold=14;
```

関連事項

kerning

key

使用法

```
-- Lingo syntax
_key.key

// JavaScript syntax
_key.key;
```

解説

最後に押されたキーを返すキープロパティです。読み取り専用。

返される値はコード番号ではなく、キーに割り当てられている米国国家規格協会（ANSI）値になります。

key プロパティは、ユーザが特定のキーを押したときに対応するアクションを実行するためのハンドラで使用できます。初期イベントハンドラ内で使うと、指定したアクションが最優先で実行されます。

注意：Lingo または JavaScript シンタックスがループを繰り返しているときにキーが押された場合には、key の値は更新されません。

異なるキーボードでどの文字がどのキーに対応するのかをテストするには、「キーボードの Lingo」サンプルムービーを使用してください。

例文

次のステートメントは、ユーザが q キーを押したときにムービーをメインメニューマーカの位置まで戻します。keyDownScript プロパティを checkKey に設定してあるので、キーが押されると on prepareMovie ハンドラでは on checkKey ハンドラが最初に実行されるイベントハンドラになります。on checkKey ハンドラは、q キーが押されたかどうかを調べ、押された場合はメインメニューマーカへ移動させます。

```
-- Lingo syntax
on prepareMovie
    keyDownScript = "checkKey"
end

on checkKey
    if (_key.key = "q") then _movie.go("Main Menu")
    end if
end

// JavaScript syntax
function prepareMovie() {
    keyDownScript = checkKey();
}

function checkKey() {
    if (_key.key == "q") {
         _movie.go("Main Menu");
    }
}
```

次の on keyDown ハンドラは、最後に押されたキーが z キーかどうかを調べ、z キーの場合は on addNumbers ハンドラを呼び出します。

```
-- Lingo syntax
on keyDown
    if (_key.key = "z") then addNumbers
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "z") {
        addNumbers();
    }
}
```

関連事項

commandDown, キー

keyboardFocusSprite

使用法

```
-- Lingo syntax
_movie.keyboardFocusSprite

// JavaScript syntax
_movie.keyboardFocusSprite;
```

解説

画面上にある特定のテキストスプライトにキーボード入力のフォーカスを設定するムービープロパティです。カーソルの挿入ポイントは制御しません。読み取り / 書き込み可能。

これは、メンバの autoTab プロパティが設定されているときに Tab キーを押すのと同じ機能です。

keyboardFocusSprite に -1 を設定すると、キーボードフォーカス制御がスコアに戻ります。0 を設定すると、編集可能スプライトに対するキーボード入力が無効になります。

例文

次のステートメントは編集可能スプライトに対するキーボード入力を無効にします。

```
-- Lingo syntax
_movie.keyboardFocusSprite=0

// JavaScript syntax
_movie.keyboardFocusSprite=0;
```

関連事項

ムービー

keyCode

使用法

```
-- Lingo syntax
_key.keyCode

// JavaScript syntax
_key.keyCode;
```

解説

最後に押されたキーのコード番号を返すキープロパティです。読み取り専用。

この戻り値はキーに割り当てられている番号のことで、米国国家規格協会（ANSI）値とは異なります。

keyCode プロパティを使用すると、key プロパティでは特定できない矢印キーやファンクションキーなどを押したときも検知できます。

異なるキーボードでどの文字がどのキーに対応するのかをテストするには、「キーボードの Lingo」サンプルムービーを使用してください。

例文

次のハンドラは、キーが押されるたびに相応するキーコードをメッセージウィンドウに表示します。

```
-- Lingo syntax
on enterFrame
    keyDownScript = put(_key.keyCode)
end

// JavaScript syntax
function enterFrame() {
    keyDownScript = put(_key.keyCode);
}
```

次のステートメントは、上向き矢印キー（コード番号 = 126）が押されたかどうかを調べ、押されている場合には 1 つ前のマーカに戻ります。

```
-- Lingo syntax
if (_key.keyCode = 126) then
    _movie.goPrevious()
end if

// JavaScript syntax
if (_key.keyCode == 126) {
    _movie.goPrevious();
}
```

次のハンドラは、矢印キーのいずれかが押されたかどうかを調べ、押されている場合は、キーに応じた処理を行います。

```
-- Lingo syntax
on keyDown
    case (_key.keyCode) of
        123: TurnLeft
        126: GoForward
        125: BackUp
        124: TurnRight
    end case
end keyDown

// JavaScript syntax
function keyDown() {
    switch (_key.keyCode) {
        case 123: TurnLeft();
            break;
        case 126: GoForward();
            break;
        case 125: BackUp();
            break;
        case 124: TurnRight();
            break;
    }
}
```

関連事項

キー , key

keyDownScript

使用法

```
the keyDownScript
```

解説

キーが押されたときに実行する Lingo を定義するためのシステムプロパティです。二重引用符で囲んだストリングとして、簡単なステートメントやハンドラを呼び出すスクリプトを記述します。

keyDownScript が定義されているときにキーが押されると、Lingo は最優先で keyDownScript プロパティに定義されているスクリプトを実行します。keyDown メッセージをムービー内の他のオブジェクトに渡すことができる pass コマンドがスクリプトに含まれていると、他の場所の on keyDown ハンドラが実行されます。

keyDownScript プロパティを設定すると、Director の旧バージョンで使われていた when keyDown then コマンドと同じ機能になります。

keyDownScript プロパティで指定した命令が不要になった場合は、set the keyDownScript to EMPTY というステートメントを使ってプロパティをオフにします。

例文

次のステートメントはキーが押されたときに実行するスクリプトを指定します。

```
-- Lingo syntax
    the keyDownScript = "go to the frame"

// JavaScript syntax
    _system.keyDownScript = "_movie.go(_movie.frame)";
```

関連事項

on keyDown, keyUpScript, mouseDownScript, mouseUpScript

keyframePlayer （モディファイア）

使用方法

```
member(whichCastmember).model(whichModel).keyframePlayer.keyframePlayerModifierProperty
```

解説

3D モディファイア。モデルによるモーションの使用を管理します。keyframePlayer モディファイアで管理されるモーションは、モデル全体が一度にアニメーション化されます。ボーンと呼ばれるモデルのセグメントをアニメーション化する、ボーンプレーヤモーションとは異なります。

これらを使用するモーションおよびモデルは、3D モデリングプログラムで作成し、W3D ファイルとして書き出した後、ムービーに読み込む必要があります。モーションは、Director で作成したモデルプリミティブには適用できません。

addModifier コマンドを使ってモデルに keyframePlayer モディファイアを追加すると、次の keyframePlayer モディファイアプロパティにアクセスできます。

- playing は、モデルがモーションを実行しているかどうかを示します。
- playList は、プロパティリストのリニアリストです。モデル用に待機しているモーションの再生パラメータが含まれます。
- currentTime は、現在再生中または一時停止されたモーションの、ミリ秒単位のローカルタイムを示します。
- playRate は、play() または queue() コマンドの **scale** パラメータを掛けた値であり、モーションの再生速度を決定します。
- playlist.count は、現在再生リストで待機しているモーションの数を返します。
- rootLock は、モーションの平行移動コンポーネントを使用するか無視するかを示します。
- currentLoopState は、モーションを一度だけ再生するか、繰り返し再生するかを示します。
- blendTime は、モディファイアの autoblend プロパティが TRUE に設定されているときに、モディファイアによって作成されたモーション間のトランジションの長さを示します。
- autoBlend は、前のモーションから現在再生中のモーションに、モディファイアがリニアトランジションを作成するかどうかを示します。
- blendFactor は、モディファイアの autoBlend プロパティが FALSE に設定されているときに、モーション間のブレンドの度合いを示します。
- lockTranslation は、指定したプレーンからモデルを表示できるかどうかを示します。
- positionReset は、モーションの終了後またはループの各反復後に、モデルを開始位置に戻すかどうかを示します。
- rotationReset は、あるモーションから次のモーションへのトランジションまたは 1 つのモーションのループにおける回転エレメントを示します。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

keyframePlayer モディファイアは、次のコマンドを使用します。

- pause は、現在モデルによって実行されているモーションを停止します。
- play() は、モーションの実行を開始または再開します。
- playNext() は、プレイリスト内の次のモーションの再生を開始します。
- queue() は、プレイリストの最後にモーションを追加します。

keyframePlayer モディファイアは、次のイベントを生成します。これらのイベントは、registerForEvent() コマンドおよび registerScript() コマンドで宣言されたハンドラによって使用されます。宣言されたハンドラの呼び出しには、イベントの種類（#animationStarted または #animationEnded）、モーションの名前、およびモーションの現在のタイムの、3 つの引数が必要です。通知イベントの詳細については、「registerForEvent()」を参照してください。

#animationStarted は、モーションの再生が開始されたときに送られます。モーション間でブレンドが使用されている場合は、トランジションが開始されたときにイベントが送られます。

#animationEnded は、モーションが終了したときに送られます。モーション間でブレンドが使用されている場合は、トランジションが終了したときにイベントが送られます。

関連事項

addModifier, modifiers, bonesPlayer （モディファイア）, motion

keyUpScript

使用法

```
the keyUpScript
```

解説

キーが離されたときに実行する Lingo を定義するためのシステムプロパティです。二重引用符で囲んだストリングとして、簡単なステートメントやハンドラを呼び出すスクリプトを記述します。

keyUpScript プロパティが定義されているときにキーが離されると、Lingo は最優先で keyUpScript に定義されたスクリプトを実行します。keyUp メッセージをムービー内の他のオブジェクトに渡すことができる pass コマンドがスクリプトに含まれていると、他の場所の on keyUp ハンドラが実行されます。

keyUpScript プロパティに指定した命令が不要になった場合、set the keyUpScript to empty というステートメントを使ってプロパティを解除します。

例文

次のステートメントはキーが押されたときに実行するスクリプトを指定します。

```
-- Lingo syntax
    the keyUpScript = "go to the frame +1 "

// JavaScript syntax
    _system.keyUpScript = "_movie.go(_movie.frame+1)";
```

関連事項

on keyUp

labelList

使用法

```
the labelList
```

解説

現行ムービー内のフレームラベルをリストするシステムプロパティです。結果は各行にラベルを 1 つ含み、改行で区切られたストリングになります (リストではありません)。ラベルはスコア内の順番に従ってリストされます。エントリは改行で区切られているため、ストリングの最終行は、最後の改行の後の空白行になります。必要に応じてこの行を削除します。

例文

次のステートメントは現行ムービー内のフレームラベルをリストします。結果は各行にラベルを 1 つ含み、改行で区切られたストリングになります（リストではありません）。

```
put the labelList
```

関連事項

frameLabel, label(), marker()

lastChannel

使用法

```
-- Lingo syntax
_movie.lastChannel

// JavaScript syntax
_movie.lastChannel;
```

解説

[ムービープロパティ] ダイアログボックスに入力された、ムービーの最大のチャネル番号を示すムービープロパティです。読み取り専用。

完成したムービーの lastChannel の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "lingo QT and Flash.dir" を参照してください。

例文

次のステートメントは、ムービーの最大のチャネル番号をメッセージウィンドウ上に表示します。

```
-- Lingo syntax
put(_movie.lastChannel)

// JavaScript syntax
put(_movie.lastChannel);
```

関連事項

ムービー

lastClick

使用法

```
-- Lingo syntax
_player.lastClick

// JavaScript syntax
_player.lastClick;
```

解説

最後にマウスボタンが押されたときから経過した時間をチック数（1/60 秒）で返すプレーヤのプロパティです。読み取り専用。

例文

次のステートメントは、最後にマウスがクリックされてから 10 秒以上経過しているかどうかを調べ、経過している場合は再生ヘッドを " クリックなし " マーカまで移動します。

```
-- Lingo syntax
if (_player.lastClick > (10 * 60)) then
    _movie.go("No Click")
end if

// JavaScript syntax
if (_player.lastClick > (10 * 60)) {
    _movie.go("No Click");
}
```

関連事項

lastEvent, lastKey, lastRoll, プレーヤ

lastError

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.lastError

// JavaScript syntax
memberOrSpriteObjRef.lastError;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。RealPlayer® から Lingo のシンボルとして返された最後のエラーシンボルを取得します。RealPlayer から返されたエラーシンボルは簡単な英語のストリングで、トラブルシューティングプロセスの開始ポイントを示します。このプロパティは、再生中はダイナミックに値が変わります。値を調べられますが設定はできません。

#PNR_OK という値は、すべてが正常に機能していることを示します。

例文

次の例は、スプライト 2 について RealPlayer から返された最後のエラーを表示し、キャストメンバ "Real" は #PNR_OUTOFMEMORY であったことを示します。

```
-- Lingo syntax
put(sprite(2).lastError) -- #PNR_OUTOFMEMORY
put(member("Real").lastError) -- #PNR_OUTOFMEMORY

// JavaScript syntax
trace(sprite(2).lastError); // #PNR_OUTOFMEMORY
put(member("Real").lastError); // #PNR_OUTOFMEMORY
```

lastEvent

使用法

```
-- Lingo syntax
_player.lastEvent

// JavaScript syntax
_player.lastEvent;
```

解説

マウスクリック、ロールオーバー、キー操作などのイベントが最後に発生してから経過した時間をチック数 (1/60 秒) で返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、最後にマウスクリック、ロールオーバー、キー操作などの操作が行われてから 10 秒経過しているかどうかを調べ、経過している場合は再生ヘッドを " ヘルプ " マーカまで移動します。

```
-- Lingo syntax
if (_player.lastEvent > (10 * 60)) then
    _movie.go("Help")
end if

// JavaScript syntax
if (_player.lastEvent > (10 * 60)) {
    _movie.go("Help");
}
```

関連事項

lastClick, lastKey, lastRoll, プレーヤ

lastFrame

使用法

```
-- Lingo syntax
_movie.lastFrame

// JavaScript syntax
_movie.lastFrame;
```

解説

ムービー内の最終フレーム番号を表示するムービープロパティです。読み取り専用。

例文

次のステートメントは、ムービーの最終フレームの番号をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.lastFrame)

// JavaScript syntax
put(_movie.lastFrame);
```

関連事項

ムービー

lastKey

使用法

```
-- Lingo syntax
_player.lastKey

// JavaScript syntax
_player.lastKey;
```

解説

最後にキーが押されたときから経過した時間をチック数（1/60 秒）で返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、最後にキーが押されてから 10 秒経過しているかどうかを調べ、経過している場合は再生ヘッドを " キーなし " マーカまで移動します。

```
-- Lingo syntax
if (_player.lastKey > (10 * 60)) then
    _movie.go("No Key")
end if

// JavaScript syntax
if (_player.lastKey > (10 * 60)) {
    _movie.go("No Key");
}
```

関連事項

lastClick, lastEvent, lastRoll, プレーヤ

lastRoll

使用法

```
-- Lingo syntax
_player.lastRoll

// JavaScript syntax
_player.lastRoll;
```

解説

マウスが最後に移動したときから経過した時間をチック数（1/60 秒）で返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、マウスが最後に移動したときから 45 秒経過したかどうかを調べ、経過している場合は再生ヘッドを " ループをひき付ける " マーカまで移動します。

```
-- Lingo syntax
if (_player.lastRoll > (45 * 60)) then
    _movie.go("Attract Loop")
end if

// JavaScript syntax
if (_player.lastRoll > (45 * 60)) {
    _movie.go("Attract Loop");
}
```

関連事項

lastClick, lastEvent, lastKey, プレーヤ

left

使用法

```
-- Lingo syntax
spriteObjRef.left

// JavaScript syntax
spriteObjRef.left;
```

解説

スプライトの境界矩形の左端の水平座標値を示すスプライトプロパティです。読み取り / 書き込み可能。

スプライト座標値は、ステージの左上隅 (0,0) より開始するピクセル数で測定されます。

例文

次のステートメントは、スプライトの左端がステージの左端より左側にあるかどうかを調べます。スプライトの左端がステージより左にある場合は、スクリプトは offLeftEdge ハンドラを実行します。

```
-- Lingo syntax
if (sprite(3).left < 0) then
    offLeftEdge()
end if

// JavaScript syntax
if (sprite(3).left < 0) {
    offLeftEdge();
}
```

次のステートメントは、（i + 1）の番号のスプライトの左端水平座標値を測定し、その値を vLowest という変数に割り当てます。

```
-- Lingo syntax
vLowest = sprite(i + 1).left

// JavaScript syntax
var vLowest = sprite(i + 1).left
```

関連事項

bottom, height, locH, locV, right, スプライト , top, width

left (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).left
```

解説

3D #box モデルリソースプロパティ。X 軸と交差するボックスの側面を閉じた状態にするか (TRUE)、または開いた状態にするか (FALSE) を指定します。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" 木箱 " という名前のモデルリソースの left プロパティを FALSE に設定し、ボックスの左面を開いた状態にします。

```
member("3D World").modelResource("crate").left = FALSE
```

関連事項

back, front, bottom (3D), top (3D), right (3D)

leftIndent

使用法

```
chunkExpression.leftIndent
```

解説

[チャンク式] の左マージンがテキストキャストメンバの左側からどれくらい離れているかをピクセル数で表すテキストキャストメンバプロパティです。

このプロパティの値は、0 以上の整数になります。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントはテキストキャストメンバ "myText" を 10 ピクセルだけインデントします。

```
-- Lingo syntax
member("myText").leftIndent=10

// JavaScript syntax
member("myText").leftIndent=10;
```

関連事項

firstIndent, rightIndent

length (byte array)

使用法

```
byteArrayObject.length
```

解説

バイト配列のサイズをバイト数で返す、バイト配列プロパティです。

例文

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.length

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.length);
```

length (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).length
vectorReference.length
```

解説

3D #box モデルリソース、#plane モデルリソース、およびベクトルプロパティ。ボックスまたはプレーンの長さをワールド単位で示します。

ボックスの長さは Z 軸に沿って測定されます。ボックスのデフォルトの長さは 50 です。

プレーンの長さは Y 軸に沿って測定されます。プレーンのデフォルトの長さは 1 です。

ベクトルの長さは vector(0, 0, 0) からの距離で、ワールド単位で表します。これは、ベクトルの magnitude プロパティと同じです。

例文

次のステートメントは、変数 myBoxLength に "GiftBox" というモデルリソースの長さを設定します。

```
myBoxLength = member("3D World").modelResource("GiftBox").length
```

関連事項

height (3D), width (3D), magnitude

lengthVertices

使用法

```
member(whichCastmember).modelResource(whichModelResource).lengthVertices
```

解説

3D #box および #plane モデルリソースプロパティ。ボックスまたはプレーンの長さに沿ったメッシュの頂点の数を示します。この値を大きくするとフェイスの数が増加し、メッシュの緻密さも増します。

ボックスの長さは Z 軸に沿って測定されます。プレーンの長さは Y 軸に沿って測定されます。

モデルのシェーダの renderStyle プロパティを #wire に設定すると、メッシュのフェイスが表示されます。renderStyle プロパティを #point に設定すると、メッシュの頂点のみが表示されます。

このプロパティの値は 2 以上にする必要があります。デフォルト値は 4 です。

例文

次のステートメントは、" タワー " というモデルリソースの lengthVertices プロパティを 10 に設定します。9 個の三角形を使用してモデルリソースのジオメトリを Y 軸に沿って定義し、10 個の頂点を形成します。

```
member("3D World").modelResource("Tower").lengthVertices = 10
```

関連事項

length (3D)

level

使用法

```
member(whichCastmember).model(whichModel).lod.level
```

解説

3D lod モディファイアプロパティ。モディファイアの auto プロパティが FALSE のときに、モディファイアによって削除されるディテールの量を示します。このプロパティの値の範囲は 0.0 ～ 100.00 です。

モディファイアの auto プロパティが TRUE に設定されると、level プロパティの値はダイナミックに更新されますが、設定はできません。

#lod モディファイアは、Director 以外の 3D モデリングプログラムで作成されたモデルに対してのみ追加できます。このようなモデルで使用されるモデルリソースの type プロパティは、#fromFile です。このモディファイアは、Director で作成したプリミティブには追加できません。

例文

次のステートメントは、モデル " 宇宙船 " の lod モディファイアの level プロパティを 50 に設定します。lod モディファイアの auto プロパティが FALSE の場合は、" 宇宙船 " は中程度のレベルオブディテールで描画されます。lod モディファイアの auto プロパティが TRUE に設定されている場合は、このコードを実行しても何の影響もありません。

```
member("3D World").model("Spaceship").lod.level = 50
```

関連事項

lod （モディファイア）, auto, bias

lifetime

使用法

```
member(whichCastmember).modelResource(modelResource).lifetime
```

解説

3D #particle モデルリソースプロパティ。パーティクルシステム内にある各パーティクルの作成から消滅までの期間をミリ秒で示します。

このプロパティのデフォルト値は 10,000 です。

例文

次のステートメントは、"3D オブジェクト " というキャストメンバでパーティクルの lifetime プロパティを 100 ミリ秒に設定します。

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").lifetime = 100.0

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).lifetime = 100.0;
```

関連事項

emitter

light

使用法

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

解説

3D エレメント。ライトの放出元のベクトル位置にあるオブジェクトです。

例文

次の例は "3D オブジェクト " というキャストメンバでの最初のライトを示します。

```
-- Lingo syntax
putmember("3Dobjects").light(1)

// JavaScript syntax
put( member("3Dobjects").getPropRef("light",1));
```

関連事項

newLight, deleteLight

lineColor

使用法

```
member(whichCastmember).model(whichModel).inker.lineColor
member(whichCastmember).model(whichModel).toon.lineColor
```

解説

3D toon および inker モディファイアプロパティ。モディファイアによってモデルに描画される線のカラーを示します。このプロパティを有効にするには、モディファイアの creases、silhouettes、または boundary のいずれかのプロパティを TRUE に設定する必要があります。

このプロパティのデフォルト値は rgb(0, 0, 0) です。

例文

次のステートメントは、2 番目のモデルの toon モディファイアによって描画されるすべての線のカラーを rgb(255, 0, 0)、つまり赤に設定します。

```
-- Lingo syntax
member("3Dobjects").model[2].toon.lineColor = rgb(255, 0, 0)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).toon.lineColor =color(100,0,0);
```

関連事項

creases, silhouettes, boundary, lineOffset

lineCount

使用法

```
-- Lingo syntax
memberObjRef.lineCount

// JavaScript syntax
memberObjRef.lineCount;
```

解説

ステージ上のフィールドキャストメンバに表示される行数を示すキャストメンバプロパティです。実際の改行の数ではなく、ストリングの自動回り込みによる行数を返します。

例文

次のステートメントは、" 今日のニュース " フィールドキャストメンバがステージに登場したときに何行になるかを調べ、その値を numberOfLines 変数に割り当てます。

```
--Lingo syntax
numberOfLines = member("Today's News").lineCount

// JavaScript syntax
var numberOfLines = member("Today's News").lineCount;
```

lineDirection

使用法

```
member(whichCastMember).lineDirection
```

解説

描かれた線の傾斜を 0 または 1 で示すシェイプメンバプロパティです。

線が左から右に上がっているときは、このプロパティの値は 1 に設定され、左から右に下がっているときは 0 に設定されます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、キャストメンバ "theLine" のラインの傾斜を切り替えて、シーソー効果を演出します。

```
-- Lingo syntax
member("theLine").lineDirection = not member("theLine").lineDirection

// JavaScript syntax
member("theLine").lineDirection = ! (member("theLine").lineDirection);
```

lineHeight

使用法

```
member(whichCastMember).lineHeight
the lineHeight of member whichCastMember
```

解説

指定したフィールドキャストメンバを表示するときに使う行間隔を定義するためのキャストメンバプロパティです。**whichCastMember** パラメータにはキャストメンバの名前または番号のいずれかを指定します。

定義した lineHeight メンバプロパティは、ムービーが閉じるまで一時的にシステムの設定よりも優先されます。ムービー全体で任意の行間隔を使う場合には、lineHeight メンバプロパティを on prepareMovie ハンドラ内で定義します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントはキャストメンバ "myText" 内の行の高さを設定します。

```
-- Lingo syntax
member("myText").lineHeight=20

// JavaScript syntax
member("myText").lineHeight=20;
```

関連事項

text, alignment, font, fontSize, fontStyle

lineOffset

使用法

```
member(whichCastmember).model(whichModel).toon.lineOffset
member(whichCastmember).model(whichModel).inker.lineOffset
```

解説

3D toon および inker ディファイアプロパティ。モディファイアによって線が描画されたモデルのサーフェスからの見かけ上の距離を示します。このプロパティを有効にするには、モディファイアの useLineOffset プロパティを TRUE に設定し、creases、silhouettes、または boundary の各プロパティのうち 1 つまたは複数を TRUE に設定する必要があります。

このプロパティの範囲は -100.00 ～ +100.00 です。デフォルトの設定は -2.0 です。

例文

次のステートメントは、" ティーポット " というモデルの toon モディファイアの lineOffset プロパティを 10 に設定します。トゥーンモディファイアによってモデルのサーフェスに描画される線は、デフォルト設定の -2 のときよりも目立つようになります。

```
member("shapes").model("Teapot").toon.lineOffset = 10
```

関連事項

creases, silhouettes, boundary, useLineOffset, lineColor

lineSize

使用法

```
member(whichCastMember).lineSize
the lineSize of member whichCastMember
sprite whichSprite.lineSize
the lineSize of sprite whichSprite
```

解説

ステージ上に表示されている指定されたシェイプキャストメンバの境界線の幅をピクセル数で定義するためのシェイプキャストメンバプロパティです。矩形以外のシェイプの場合には、境界線は境界矩形ではなく、そのシェイプの境界線になります。

スプライトの oflineSize 設定は、メンバの oflineSize 設定よりも優先されます。ステージ上にスプライトがあるときに、Lingo がメンバの lineSize 設定を変更した場合、スプライトが終了するまでスプライトの lineSize 設定は維持されます。

Lingo が設定した値を現行スプライト終了後も維持するには、スクリプトを使用したスプライトにする必要があります。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、シェイプキャストメンバ "myShape" の幅を 5 ピクセルに設定します。

```
--Lingo syntax
member("myShape").lineSize=5

// JavaScript syntax
member("myShape").lineSize=5;
```

linked

使用法

```
-- Lingo syntax
memberObjRef.linked

// JavaScript syntax
memberObjRef.linked;
```

解説

スクリプト、Flash ムービー、またはアニメーション化された GIF ファイルを外部ファイルに格納するか（TRUE）、Director キャストライブラリに格納するか（FALSE）を制御するメンバプロパティです。デフォルトは TRUE です。スクリプト、Flash、またはアニメーション化された GIF メンバでは読み取り / 書き込み可能、他のすべてのメンバタイプの場合は読み取り専用。

データをリンクされた外部のファイルに格納する場合は、キャストメンバの pathName プロパティで、ムービーファイルのある場所を指定する必要があります。

例文

次のステートメントは、Flash キャストメンバ "homeBodies" を、リンクされたメンバから内部的に格納されたメンバに変換します。

```
-- Lingo syntax
member("homeBodies").linked = 0

// JavaScript syntax
member("homeBodies").linked = 0;
```

関連事項

メンバ

loaded

使用法

```
-- Lingo syntax
memberObjRef.loaded

// JavaScript syntax
memberObjRef.loaded;
```

解説

指定したキャストメンバがメモリ内に読み込まれているかどうかを調べるためのメンバプロパティです。読み込まれているときは TRUE を返し、読み込まれていないときは FALSE を返します。読み取り専用。

キャストメンバの種類によって、読み込み方法が多少異なります。

- シェイプとスクリプトキャストメンバは、常にメモリ内に読み込まれます。
- ムービーキャストメンバは、メモリから解放されることはありません
- デジタルビデオキャストメンバは、その使用状況にかかわらず個別にプリロードやメモリからの解放を指示することができますデジタルビデオキャストメンバの再生は、メモリからの方がディスクからよりも高速になります。

例文

次のステートメントは、" デモムービー " キャストメンバがメモリ内に読み込まれているかどうかを調べ、読み込まれていない場合は他のムービーを再生します

```
-- Lingo syntax
if member("Demo Movie").loaded = FALSE then
    _movie.go(1, "Waiting.dir")
end if

// JavaScript syntax
if (member("Demo Movie").loaded == false) {
    _movie.go(1, "Waiting.dir")
}
```

関連事項

メンバ

loc (backdrop and overlay)

使用法

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.overlay[index].loc
member(whichCastmember).camera(whichCamera).overlay[index].loc
```

解説

3D 背景およびオーバーレイプロパティ。スプライトの左上隅から計測した際の、背景またはオーバーレイの 2D 位置を示します。

このプロパティは、背景またはオーバーレイを作成する addBackdrop、addOverlay、insertBackdrop、または insertOverlay の各コマンドのパラメータとして初期設定されます。

例文

次のステートメントは、スプライト 2 のカメラの最初の背景の位置を設定します。

```
sprite(2).camera.backdrop[1].loc = point(120, 120)
```

関連事項

bevelDepth, overlay, regPoint (3D)

locH

使用法

```
-- Lingo syntax
spriteObjRef.locH

// JavaScript syntax
spriteObjRef.locH;
```

解説

スプライトのセンターポイントの水平位置を示すスプライトプロパティです。読み取り / 書き込み可能。

スプライトの座標値はステージの左上隅を原点として測定されます。

値を現行スプライト終了後も維持するには、スプライトをスクリプトされたものにする必要があります。

例文

次のステートメントは、スプライト 15 の水平位置をマウスがクリックされた位置に揃えます。

```
-- Lingo syntax
sprite(15).locH = _mouse.mouseH

// JavaScript syntax
sprite(15).locH = _mouse.mouseH;
```

関連事項

bottom, height, left, locV, point(), right, スプライト , top, updateStage()

lockTranslation

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.lockTranslation
member(whichCastmember).model(whichModel).keyframePlayer.lockTranslation
```

解説

3D #bonesPlayer および #keyframePlayer モディファイアプロパティ。指定されたプレーンから移動されないようにします。ただし、モーションデータの絶対平行移動は除きます。手動による移動や、累積エラーにより発生した移動はすべて削除されます。#none、#x、#y、#z、#xy、#yz、#xz、および #all の値によって、各フレームで 3 つの平行移動コンポーネントのいずれを制御するのかを決めます。軸に対するロックがオンになっている場合、その軸に沿った平行移動は一旦格納され、アニメーションの基準となる固定移動の後に使用されます。軸ロックを非アクティブにするか、オブジェクトを移動するか、またはその軸ロックを再びアクティブにすることで、この移動をリセットできます。

言い換えると、モーションの再生時に無視する平行移動の軸を定義します。上方向を **z** 軸に沿って指定したモデルをグラウンドプレーンのレベルにロックしておくには、lockTranslation を #z に設定します。このプロパティのデフォルト値は #none です。

例文

次のステートメントは、" 歩行者 " というモデルの lockTranslation プロパティを #z に設定します。

```
member("ParkScene").model("Walker").bonesPlayer.lockTranslation = #z
```

関連事項

immovable

locV

使用法

```
-- Lingo syntax
spriteObjRef.locV

// JavaScript syntax
spriteObjRef.locV;
```

解説

スプライトのセンターポイントの垂直位置を示すスプライトプロパティです。読み取り / 書き込み可能。

スプライトの座標値はステージの左上隅を原点として測定されます。

値を現行スプライト終了後も維持するには、スプライトをスクリプトされたものにする必要があります。

例文

次のステートメントは、スプライト 15 の垂直位置をマウスがクリックされた位置に揃えます。

```
-- Lingo syntax
sprite(15).locV = _mouse.mouseV

// JavaScript syntax
sprite(15).locV = _mouse.mouseV;
```

関連事項

bottom, height, left, locH, point(), right, スプライト , top, updateStage()

locZ

使用法

```
-- Lingo syntax
spriteObjRef.locZ

// JavaScript syntax
spriteObjRef.locZ;
```

解説

スプライトチャネルやプロパティの操作を必要とせずにレイヤー処理を制御する、Z 方向のスプライトの順序を指定するスプライトプロパティです。読み取り / 書き込み可能。

このプロパティには、負の 20 億から正の 20 億までの整数を持たせることができます。大きい番号のスプライトは、小さい番号のスプライトの手前に表示されます。同じ locZ 値のスプライトが 2 つある場合は、チャネル番号を優先して、最終的な表示順序を決定します。これは、例え locZ 値が同じでも、小さいチャネル番号のスプライトは大きいチャネル番号の後ろに表示されるということです。

デフォルトでは、スプライトの locZ 値とチャネル番号は同じです。

ヒット検知やマウスイベントなどレイヤーに依存する操作では、スプライトの locZ 値に従います。したがって、スプライトの locZ 値を変更すると、スプライト同士を部分的または完全に見分けるのが難しくなり、ユーザがスプライトをクリックできない可能性があります。

この関数以外の関数は、スプライトのこのような locZ 順序を守っていません。生成されるイベントは、チャネル 1 から始まり、そこから連続的に増加し、スプライトの Z 順序には左右されません。

例文

このハンドラは、gHighestSprite というグローバル変数を使用します。gHighestSprite は、スプライト番号の設定で使用する startMovie ハンドラに初期設定されています。スプライトをクリックすると、そのスプライトの locZ が gHighestSprite + 1 に設定され、ステージの最前面に移動します。移動すると、gHighestSprite が 1 だけ増えて、その次の mouseUp の実行に備えます。

```
-- Lingo syntax
on mouseUp me
    global gHighestSprite
    sprite(me.spriteNum).locZ = gHighestSprite + 1
    gHighestSprite = gHighestSprite + 1
end

// JavaScript syntax
function mouseUp() {
    _global.gHighestSprite;
    sprite(this.spriteNum).locZ = _global.gHighestSprite + 1
    _global.gHighestSprite = _global.gHighestSprite + 1
}
```

関連事項

locH, locV, スプライト

lod （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).lod.lodModifierProperty
```

解説

3D モディファイア。モデルがカメラから遠ざかるに従って、モデルからディテールをダイナミックに削除します。

このモディファイアは、Director 以外の 3D モデリングプログラムで作成されたモデルに対してのみ追加できます。このようなモデルで使用されるモデルリソースの type プロパティは、#fromFile です。このタイプのモデルでは、lod モディファイアが割り当てられているかどうかにかかわらず、ディテールの削除が実行されます。このモディファイアを割り当てると、ディテール削除のプロパティを制御できます。このモディファイアは、Director で作成したプリミティブには追加できません。

3D モデリングプログラムにより、すべてのモデルに対する lod モディファイアデータが生成されます。userData プロパティに "sw3d_no_lod = true" を設定すると、ストリーミングの完了時に lod モディファイアデータとメモリを解放するよう指定できます。

sds モディファイアと lod モディファイアは互いに反対の機能を持つため、一緒に使用する場合は注意する必要があります。sds モディファイアが詳細なジオメトリを追加するのに対し、lod モディファイアはジオメトリの詳細を削除します。sds モディファイアを追加する前に、lod.auto モディファイアプロパティを無効にし、lod.level モディファイアプロパティを使用して解像度を最大に設定しておくことをお勧めします。

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

lod モディファイアには以下のプロパティがあります。

- auto。モディファイアは、モデルとカメラ間の距離の変化に応じて、ディテール削除のレベルを設定できます。モディファイアの level プロパティの値は更新されますが、auto プロパティが TRUE に設定されない限り、level プロパティの設定は有効になりません。
- bias。モディファイアの auto プロパティが TRUE に設定されるときの、モデルからディテールを削除する度合いを示します。このプロパティの範囲は、0.0 (すべてのポリゴンを削除) ～ 100.0 (ポリゴンを削除しない) です。このプロパティのデフォルト値は 100.0 です。
- level。モディファイアの auto プロパティが FALSE に設定されるときのディテールの削除量を示します。このプロパティの範囲は 0.0 ～ 100.00 です。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

例文

次の例は "3D オブジェクト " というキャストメンバからモデル " 球体 01" を動的に削除します。

```
--Lingo
member("3Dobjects").model("Sphere01").lod.auto=1
member("3Dobjects").model("Sphere01").lod.bias=0

// Javascript
member("3Dobjects").getPropRef("model",2).lod.auto=1;
member("3Dobjects").getPropRef("model",2).lod.bias=0;
```

関連事項

sds （モディファイア）, auto, bias, level, addModifier

loop (3D)

使用法

```
member(whichCastmember).loop
```

解説

3D キャストメンバプロパティ。キャストメンバ内の最初のモデルに対して、モーションを連続的に適用するか (TRUE)、または 1 回再生して停止するか (FALSE) を示します。

このプロパティのデフォルト設定は TRUE です。

例文

次のステートメントは、" 歩行者 " というキャストメンバの loop プロパティを TRUE に設定します。" 歩行者 " の最初のモデルで実行中のモーションは連続して繰り返されます。

```
member("Walkers").loop = TRUE
```

関連事項

motion, play() (3D), queue() (3D), animationEnabled

loop (emitter)

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.loop
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、寿命が終了するパーティクルに対する処理を取得および設定できます。値が TRUE の場合、パーティクルは寿命の終了時に、エミッタの region プロパティで定義されたエミッタ位置で再生されます。値が FALSE の場合、パーティクルは寿命の終了時に消滅します。このプロパティのデフォルト設定は TRUE です。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の emitter.loop プロパティを 1 に設定し、"ThermoSystem" のパーティクルが連続して放出されるようにします。

```
member("Fires").modelResource("ThermoSystem").emitter.loop = 1
```

関連事項

emitter

loop （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.loop

// JavaScript syntax
memberObjRef.loop;
```

解説

指定したデジタルビデオ、サウンド、Flash ムービーキャストメンバにループを設定するかどうかを指定するためのキャストメンバプロパティです。ループを設定するときは TRUE を指定し、設定しないときは FALSE を指定します。

例文

次のステートメントは、" デモ " QuickTime ムービーキャストメンバにループを設定します。

```
-- Lingo syntax
member("Demo").loop = 1

// JavaScript syntax
member("Demo").loop = 1;
```

loop (MP4Media/FLV)

使用法

```
member(1).loop = true
sprite(1).loop = true
```

解説

スプライトおよびキャストメンバのプロパティ。指定した MP4Media/FLV キャストメンバをループに設定するか（True）、しないか（False）を指定します。

例文

以下の例は、MP4Media/FLV キャストメンバの Demo をループに設定します。

```
-- Lingo syntax
member("Demo").loop = 1

// JavaScript syntax
member("Demo").loop = 1;
```

loop (Flash)

使用法

```
sprite(whichFlashSprite).loop
the loop of sprite whichFlashSprite
member (whichFlashMember).loop
the loop of member whichFlashMember
```

解説

Flash ムービーを連続するループで再生するかどうかを制御する Flash スプライトおよびメンバプロパティです。連続再生する場合は TRUE であり、一度再生してから停止するときは FALSE です。

このプロパティは、値を調べたり、変更できます。

例文

次のフレームスクリプトは、percentStreamed プロパティを使って、リンクしている Flash キャストメンバ "flashObj" のストリーミングのステータスを調べます。flashObj をダウンロードしている間、ムービーは現行フレームでループします。flashObj がダウンロードを完了すると、その次のフレームに進み、チャネル 1 の Flash ムービーの loop プロパティが FALSE に設定されるので、再生が最後まで続いてから停止します（flashObj のダウンロード中このスプライトがループすると考えてください）。

```
-- Lingo syntax
on exitFrame me
if member("flashObj").percentStreamed = 100 then
    member(1).loop = FALSE
    go the frame + 1
else
    go to the frame
end if

end
// JavaScript syntax
function exitFrame(me) {
if(member("flashObj").percentStreamed == 100)
    {
        member(1).loop = 0;
        _movie.go(_movie.frame+1);
    }
    else
    {
        _movie.go(_movie.frame);
    }
}
```

loop (Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.loop

// JavaScript syntax
windowsMediaObjRef.loop;
```

解説

Windows Media のプロパティ。完了時にムービーをループさせる（TRUE、デフォルト）か、ループさせない（FALSE）かを定義します。読み取り / 書き込み可能。

例文

次のステートメントはキャストメンバ " クラシカル " が完了時にムービーをループさせるよう指定します。

```
-- Lingo syntax
member("Classical").loop = TRUE

// JavaScript syntax
member("Classical").loop = true;
```

関連事項

Windows Media

loopBounds

使用法

```
-- Lingo syntax
spriteObjRef.loopBounds

// JavaScript syntax
spriteObjRef.loopBounds;
```

解説

QuickTime キャストメンバまたはスプライトの内部ループポイントを設定する QuickTime スプライトプロパティです。ループポイントは、[開始時間 , 終了時間] という Director リストとして指定されます。

[開始時間] および [終了時間] パラメータは次の条件を満たす必要があります。

- 両方のパラメータは、Director のチック数で時間を指定する整数です。
- 値は、0 から QuickTime キャストメンバの継続時間までの数値です。
- 開始時間には、終了時間よりも早い時間を設定します。

これらの条件をすべて満たしていない場合は、QuickTime ムービーは継続時間が終了するまでループします。

ムービーの loop プロパティが FALSE に設定されていると、loopBounds プロパティの効果はありません。ムービー再生中に loop プロパティが TRUE に設定されていると、ムービーの再生が継続されます。次のようなルールに基づいて、どのようにムービーをループさせるかが決定されます。

- loopBounds により指定されている終了時間になると、ムービーは開始時間までループバックします。
- ムービーが最後まで再生されると、ムービーは先頭までループバックします。

ムービー再生中に loop プロパティが消されると、ムービーの再生が継続されます。Director は、ムービーが最後まで再生されると、再生を停止します。

このプロパティは、値を調べたり、設定することができます。デフォルト設定は [0,0] です。

例文

次のスプライトスクリプトは、QuickTime スプライト内のループの開始時間および終了時間を設定します。時間を秒数で指定し、次にその値に 60 をかけてチック数に換算しています。

```
-- Lingo syntax
on beginSprite me
    sprite(me.spriteNum).loopBounds = [(16 * 60),(32 * 60)]
end

// JavaScript syntax
function beginSprite() {
    sprite(me.spriteNum).loopBounds = list((16 * 60),(32 * 60));
}
```

loopCount

使用法

```
-- Lingo syntax
soundChannelObjRef.loopCount

// JavaScript syntax
soundChannelObjRef.loopCount;
```

解説

サウンドチャネルプロパティです。ループを設定されたサウンドチャネルの現在のサウンドをループさせる総回数を指定します。読み取り専用。

このプロパティのデフォルト値は 1 で、この場合はループなしで単にキューに登録されます。

サウンドの一部分をループさせるには、queue() メソッドまたは setPlayList() メソッドに、loopStartTime パラメータ、loopEndTime、および loopCount の各パラメータを渡します。これらのパラメータ以外でこのプロパティを設定する方法はありません。

loopCount を 0 に設定すると、無限ループになります。サウンドのキャストメンバの loop プロパティを TRUE に設定すると、loopCount は 0 を返します。

例文

次のハンドラは、2 つのサウンドをサウンドチャネル 2 で待機させ、再生します。最初のサウンドであるキャストメンバの "introMusic" は、8 ～ 8.9 秒の部分を 5 回ループします。2 番目のサウンドであるキャストメンバ "creditsMusic" は、3 回ループします。#loopStartTime と #loopEndTime を指定しないと、これらの値が #startTime と #endTime のデフォルトになります。

```
-- Lingo syntax
on playMusic
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #loopCount:5, #loopStartTime:8000,
#loopEndTime:8900])
    sound(2).queue([#member:member("creditsMusic"), #startTime:3000, #endTime:3000, #loopCount:3])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
    sound(2).queue(propList("member",member("creditsMusic"), "startTime",3000,"endTime",3000,
"loopCount",3]);
    sound(2).play();
}
```

関連事項

loopEndTime （サウンドチャネル）, loopStartTime, queue(), setPlayList(), サウンドチャネル

loopCount （サウンドオブジェクト）

使用法

```
soundObject.loopCount
```

解説

サウンドオブジェクトが再生する回数を返すサウンドオブジェクトプロパティです。

例文

```
soundObject.loopCount = 8
--Lingo syntax
on mouseUp me
    put soundObjRef.loopcount -- Displays the number of times the sound object
-- associated with soundobjectRef plays.
end

//JavaScript syntax
function mouseUp(){
put (soundObjRef.loopCount) ; // Displays the number of times the sound object
// associated with soundobjectRef plays.
}
```

loopEndTime （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.loopEndTime

// JavaScript syntax
soundChannelObjRef.loopEndTime;
```

解説

サウンドチャネルで現在再生されているサウンドに設定したループ終了時間をミリ秒単位で指定するサウンドチャネルプロパティです。読み取り専用。

このプロパティの値は浮動小数点なので、サウンド再生をミリ秒未満まで測定および制御することができます。

このプロパティは、queue() コマンドまたは setPlaylist() コマンドのプロパティとして渡すことによって設定します。これ以外の方法では設定できません。

例文

次のハンドラは、サウンドチャネル 2 のサウンドキャストメンバ "introMusic" を再生します。再生では、8 秒のポイントから 8.9 秒のポイントまでが 5 回ループされます。

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5 \ #loopStartTime:8000,
#loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

関連事項

queue(), setPlayList(), サウンドチャネル

loopEndTime （サウンドオブジェクト）

使用法

```
soundObject.loopEndTime
```

解説

サウンドオブジェクトのループする部分の終了位置を設定するサウンドオブジェクトプロパティです。loopEndTime は、現在のサウンドオブジェクトの開始からのミリ秒数で指定します。

loopStartTime が 100、loopEndTime が 200、loopCount が 3 の場合は、サウンドオブジェクトの開始から 100 ミリ秒と 200 ミリ秒の間の部分が 3 回再生されます。

例文

```
soundObject.loopEndTime = 46890
--Lingo syntax
on mouseUp me
    put soundObjRef.loopEndTime -- Displays the end time for the sound portion
-- that loops.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopEndTime) ; // Displays the end time for the sound portion
// that loops.
}
```

関連事項

loopStartTime （サウンドオブジェクト）

loopsRemaining

使用法

```
-- Lingo syntax
soundChannelObjRef.loopsRemaining

// JavaScript syntax
soundChannelObjRef.loopsRemaining;
```

解説

サウンドチャネルで現在再生されているサウンドの残りループ回数を指定するサウンドチャネルプロパティです。読み取り専用。

サウンドを待機させた時点でループを指定しなかった場合、このプロパティは 0 です。このプロパティは、サウンド再生開始直後に調べると、queue() メソッドまたは setPlayList() メソッドの #loopCount プロパティで定義したループ回数より 1 回少ない回数を返します。

例外

次の例はサウンドチャネル 2 で残っているループを示します。

```
-- Lingo syntax
put sound(2).loopsRemaining

// JavaScript syntax
put(sound(2).loopsRemaining);
```

関連事項

loopCount, queue(), setPlayList(), サウンドチャネル

loopsRemaining （サウンドオブジェクト）

使用法

```
soundObject.loopsRemaining
```

解説

再生しているサウンドオブジェクトで保留中の再生ループの数を返すサウンドオブジェクトプロパティです。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.loopsRemaining -- Displays the number of remaining playback loops.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopsRemaining) ; // Displays the number of remaining playback loops.
}
```

loopStartTime

使用法

```
-- Lingo syntax
soundChannelObjRef.loopStartTime

// JavaScript syntax
soundChannelObjRef.loopStartTime;
```

解説

サウンドチャネルで現在再生されているサウンドのループ開始時間をミリ秒単位で指定するサウンドチャネルプロパティです。読み取り専用。

値は浮動小数点なので、サウンド再生をミリ秒未満まで測定および制御することができます。ループを設定しないと、startTime がデフォルトになります。

このプロパティは、queue() メソッドまたは setPlaylist() メソッドのプロパティとして渡すことによって設定します。これ以外の方法では設定できません。

例文

次のハンドラは、サウンドチャネル 2 のサウンドのキャストメンバ "introMusic" を再生します。8 ～ 8.9 秒の間を 5 回ループします。

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5 \#loopStartTime:8000,
#loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

関連事項

queue(), setPlayList(), サウンドチャネル , startTime (サウンドチャネル)

loopStartTime （サウンドオブジェクト）

使用法

```
soundObject.loopStartTime
```

解説

サウンドオブジェクトのループする部分の開始位置を設定するサウンドオブジェクトプロパティです。loopStartTime は、現在のサウンドオブジェクトの開始からのミリ秒数で指定します。

例文

```
soundObject.loopStartTime = 6890
--Lingo syntax
on mouseUp me
    put soundObjRef.loopStartTime -- Displays the loop start time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopStartTime) ; // Displays the loop start time for the sound object
// associated with soundobjectRef.

}
```

関連事項

loopEndTime （サウンドオブジェクト）

magnitude

使用法

```
whichVector.magnitude
```

解説

3D プロパティ。ベクトルの大きさを返します。値は浮動小数点数で表します。ベクトルの大きさとはベクトルの長さであり、常に 0.0 以上です (vector (0, 0, 0) = 0)。

例文

次のステートメントは、MyVec1 の大きさが 100 であることを示します。

```
-- Lingo syntax
MyVec1 = vector(100, 0, 0)
put MyVec1.magnitude

// JavaScript syntax
MyVec1 = vector(100, 0, 0);
put(MyVec1.magnitude);
```

関連事項

length (3D), identity()

margin

使用法

```
-- Lingo syntax
memberObjRef.margin

// JavaScript syntax
memberObjRef.margin;
```

解説

フィールドボックス内のマージンをピクセル数で定義するためのフィールドキャストメンバプロパティです。

例文

次のステートメントは、フィールドキャストメンバ " 今日のニュース " のボックス内のマージンを 15 ピクセルに設定します。

```
--Lingo syntax
member("Today's News").margin = 15

// JavaScript syntax
member("Today's News").margin = 15;
```

markerList

使用法

```
-- Lingo syntax
_movie.markerList

// JavaScript syntax
_movie.markerList;
```

解説

スコア内のマーカ群をスクリプトのプロパティリストにして表示するムービープロパティです。読み取り専用。

リストの形式を次に示します。

```
frameNumber: "markerName"
```

例文

次のステートメントは、マーカ群のリストをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_movie.markerList)

// JavaScript syntax
put(_movie.markerList);
```

関連事項

ムービー

mask

使用法

```
-- Lingo syntax
memberObjRef.mask

// JavaScript syntax
memberObjRef.mask;
```

解説

ステージに、直接レンダリングされるメディア用のマスクとして、マスクのピクセルが黒の領域にメディアが表示されるように白黒（1 ビット）のキャストメンバを指定するキャストメンバプロパティです。mask プロパティを使うと、QuickTime ムービーを矩形領域以外で再生しても、ステージ最前面に置くオプションを利用したデジタルビデオのパフォーマンスの利点を得ることができます。mask プロパティは、ステージ最前面にないキャストメンバには効果がありません。

Director は常に、マスクキャストメンバのセンターポイントを QuickTime ムービースプライトの左上隅に合わせます。デフォルトでは、ビットマップのセンターポイントは中央に合うように設定されているので、左上隅に再設定します。[QuickTime メンバ] のセンターポイントは、左上隅以外に再設定することはできません。マスクキャストメンバを移動することはできません。また、マスクキャストメンバは関連するキャストメンバの center および crop プロパティから影響を受けません。

最善の結果を得るには、QuickTime キャストメンバの mask プロパティを、onbeginSprite イベントハンドラによってスプライトがステージ上に表示される前に設定します。キャストメンバがステージ上にあるときに mask プロパティを設定または変更すると、予想できない結果を引き起こすことがあります。例えば、mask プロパティが有効になった瞬間に、マスクがデジタルビデオの残像として張り付くことがあります。

マスキングは高度な機能であるため、練習が必要です。

このプロパティは、値を調べたり、設定することができます。マスクを削除するには、mask プロパティを 0 に設定します。

例文

次のフレームスクリプトは、Director がフレームを描きはじめる前に、QuickTime スプライト用のマスクを設定します。

```
-- Lingo syntax
on prepareFrame
    member("Peeping Tom").mask = member("Keyhole")
end

// JavaScript syntax
function prepareFrame() {
    member("Peeping Tom").mask = member("Keyhole");
}
```

関連事項

invertMask

maxInteger

使用法

```
the maxInteger
```

解説

システムが対応している最大整数値を返すためのプロパティです。ほとんどのパーソナルコンピュータでは、この値は "2,147,483,647"（2 の 31 乗から 1 を引いた値）です。

ループ前の境界変数の初期化や、限界テストなどに便利なプロパティです。

処理できる整数値の範囲よりも大きい数値を使うには、浮動小数を使います。整数値に比べると処理速度が低下しますが、より広範囲の値に対応できます。

例文

次のステートメントは、特定の桁数の 2 進数で表すことができる 10 進数の最大値を示す表を生成し、メッセージウィンドウに表示します。

```
on showMaxValues
    b = 31
    v = the maxInteger
    repeat while v > 0
        put b && "-" && v
        b = b-1
        v = v/2
    end repeat
end
```

maxSpeed

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのパーティクルの最高放出速度を取得または設定できます。各パーティクルの初期速度は、エミッタの minSpeed プロパティと maxSpeed プロパティの間でランダムに選択されます。

値は浮動小数点数で、0.0 より大きくなければなりません。

例文

次の例では、"3D オブジェクト " というキャストメンバでパーティクルの最高速度を 15 に設定します。

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.maxSpeed=15

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.maxSpeed=15;
```

関連事項

minSpeed, emitter

media

使用法

```
-- Lingo syntax
memberObjRef.media

// JavaScript syntax
memberObjRef.media;
```

解説

指定したキャストメンバを表す数字を返すメンバプロパティです。読み取り / 書き込み可能。

このプロパティを設定すると大量のメモリを消費するため、オーサリング環境でのみ使用してください。

media プロパティを使用すると、キャストメンバの内容を別のキャストメンバにコピーできます。コピーするには、コピー元のメンバの media の値をコピー先のメンバの media の値として設定します。

フィルムループキャストメンバの場合、media プロパティはスコア内のフレームやチャネルの選択範囲を特定します。

プロジェクト内でメディアを入れ替えるには、member スプライトプロパティを設定する方が効率的です。

例文

次のステートメントは、キャストメンバ " 日の出 " の media の値をキャストメンバ " 夜明け " の media に設定することにより、" 日の出 " の内容を " 夜明け " にコピーします。

```
-- Lingo syntax
member("Dawn").media = member("Sunrise").media

// JavaScript syntax
member("Dawn").media = member("Sunrise").media;
```

関連事項

メンバ

mediaReady

使用法

```
-- Lingo syntax
memberObjRef.mediaReady

// JavaScript syntax
memberObjRef.mediaReady;
```

解説

キャストメンバ、ムービーまたはキャストライブラリファイル、リンクされたキャストメンバのいずれかがインターネットからダウンロードされているかどうか、ローカルディスクで使用可能か（TRUE）、そうでないか（FALSE）を確認するメンバプロパティです。読み取り専用。

このプロパティは、ムービーやキャストライブラリファイルをストリーミングするときに有効です。ムービーのストリーミングを設定するには、[設定] メニューを使って [ムービー再生] プロパティに [ダウンロードされた時点でメディアを使用] を選択します。これは、デフォルトの設定です。

mediaReady プロパティの例については、Director ヘルプのサンプルムービー「Streaming Shockwave」を参照してください。

例文

次のステートメントは、必要なキャストメンバがダウンロードされてローカルディスクに存在していれば、キャストメンバをそれに変更します。

```
-- Lingo syntax
if member("background").mediaReady = TRUE then sprite(2).member = member("background").number
end if

// JavaScript syntax
if (member("background").mediaReady == true) {
    sprite(2).member = member("background").number;
}
```

関連事項

メンバ

mediaStatus (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.mediaStatus

// JavaScript syntax
dvdObjRef.mediaStatus;
```

解説

DVD プロパティ。DVD プレーヤの現在の状態を示すシンボルを返します。読み取り専用。

返されるシンボルは次のとおりです。

シンボル	解説
#stopped	DVD は停止中です。
#playing	DVD は再生中です。
#paused	DVD は一時停止中です。
#scanning	DVD はスキャン中です。
#uninitialized	DVD は初期化されていません。
#volumeInvalid	指定された DVD は有効ではありません。
#volumeUknown	DVD が存在しないか、DVD ドライブにディスクが挿入されていません。
#systemSoftwareMissing	DVD デコーダがインストールされていません。
#systemSoftwareBusy	DVD の再生に必要なシステムソフトウェアが別のアプリケーションに使用されています。

関連事項

DVD

mediaStatus (MP4Media/FLV)

使用法

```
put sprite(1).mediaStatus
put member(1).mediaStatus
```

解説

MP4Media/FLV メンバまたはスプライトのプロパティ。MP4Media/FLV ストリームの状態を表すシンボルを取得します。このプロパティは読み取り専用であり、その値は再生中に変化することがあります。

このプロパティの有効な値は以下のとおりです。

値	プロパティ
#playing	MP4Media/FLV ストリームが現在再生中であることを示します。
#paused	再生が中断されたことを示します。原因としては、ユーザが MP4Media/FLV ビューアで一時停止ボタンを押したこと、またはスクリプトが pause() メソッドを実行したことが考えられます。
#stopped	MP4Media/FLV エディタウィンドウの Stop ボタンをクリックすることで、またはスクリプトが stop() メソッドを実行することで、ストリームが停止されたことを示します。

例文

次の例は、スプライト 1 の MP4Media/FLV エレメントが再生中であることを示します。

```
-- Lingo syntax
put(sprite(1).mediaStatus)  -- #playing
put(member(1).mediaStatus) -- #playing

// JavaScript syntax
put(sprite(1).mediaStatus);  // #playing
put(member(1).mediaStatus); // #playing
```

mediaStatus (RealMedia, Windows Media)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.mediaStatus

// JavaScript syntax
memberOrSpriteObjRef.mediaStatus;
```

解説

RealMedia および Windows Media のキャストメンバおよびスプライトプロパティ。RealMedia または Windows Media ストリームの状態を表すシンボルを取得します。読み取り専用。

このプロパティの値は、再生中に変化することがあります。

このプロパティの有効な値は以下のとおりです。

- #closed は、RealMedia または Windows Media キャストメンバがアクティブでないことを示します。再生が開始されるまでは、mediaStatus の値は #closed のままです。
- #connecting は、RealMedia または Windows Media ストリームへの接続が確立されていることを示します。
- #opened は、RealMedia または Windows Media ストリームへの接続が確立されて開かれたことを示します。これは一時的な状態で、すぐに #buffering に移行します。
- #buffering は、RealMedia または Windows Media ストリームが再生バッファにダウンロードされていることを示します。バッファへのダウンロードが完了する（percentBuffered の値が 100 になる）と、pausedAtStart プロパティが FALSE の場合はストリームの再生が開始されます。詳細については、percentBuffered を参照してください。
- #playing は、RealMedia または Windows Media が現在実行中であることを示します。
- #seeking は、再生が seek コマンドによって中断されたことを示します。
- #paused は、再生が中断されたことを示します。原因として、ユーザが RealMedia または Windows Media ビューで [停止] ボタンを押した、またはスクリプトが pause() メソッドを呼び出したことが考えられます。
- #error は、何らかの原因で、ストリームへの接続、バッファリング、または再生ができなかったことを示します。lastError プロパティに実際のエラー内容が報告されます。

キャストメンバの state (RealMedia) の値に応じて、mediaStatus プロパティに返される値が異なります。1 つの state 値には 1 つの mediaStatus 値だけが対応します。

例文

次の例は、スプライト "1" およびキャストメンバ "Real" の RealMedia エレメントが再生中であることを示します。

```
-- Lingo syntax
put(sprite(1).mediaStatus)
put(member("RealDemo").mediaStatus)

// JavaScript syntax
put(sprite(1).mediaStatus);
put(member("RealDemo").mediaStatus);
```

関連事項

state (RealMedia), percentBuffered, lastError

mediaXtraList

使用法

```
-- Lingo syntax
_player.mediaXtraList

// JavaScript syntax
_player.mediaXtraList;
```

解説

プレーヤのプロパティ。Director プレーヤが使用可能なすべてのメディアエクストラ拡張機能のリニアリストを返します。読み取り専用。

例文

次のステートメントは、Director プレーヤが使用可能なすべてのメディアエクストラ拡張機能をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(_player.mediaXtraList)

// JavaScript syntax
put(_player.mediaXtraList);
```

関連事項

メディアタイプ , プレーヤ , scriptingXtraList, toolXtraList, transitionXtraList, xtraList （プレーヤ）

member

使用法

```
member(whichCastmember).texture(whichTexture).member
member(whichCastmember).model(whichModel).shader.texture.member
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].textureList[textureListIndex].membe
r
```

解説

3D テクスチャプロパティ。テクスチャのタイプが #fromCastMember のとき、このプロパティはテクスチャのソースとして使用されるキャストメンバを示します。

このプロパティは、値を調べたり、設定することができます。

テクスチャのタイプが #importedFromFile である場合、このプロパティの値は void となり、設定できません。テクスチャのタイプが #fromImageObject の場合は、このプロパティの値は void となりますが、設定はできます。

例文

次の Lingo は、新規テクスチャを追加します。2 番目のステートメントは、"gbTexture" というテクスチャの作成に使用されるキャストメンバが、キャスト 1 のメンバ 16 であることを示します。

```
member("scene").newTexture("gbTexture", #fromCastmember, member(16, 1))
put member("scene").texture("gbTexture").member
-- (member 16 of castLib 1)
```

member （キャスト）

使用法

```
-- Lingo syntax
castObjRef.member[memberNameOrNum]

// JavaScript syntax
castObjRef.member[memberNameOrNum]
```

解説

キャストライブラリのメンバにインデックスまたは名前でアクセスできるキャストライブラリプロパティです。読み取り専用。

memberNameOrNum パラメータには、キャストメンバの名前を表すストリング、またはキャストメンバの番号を表す整数を指定できます。

例文

次の例では、" 内部 " というキャストライブラリの 2 番目のキャストメンバにアクセスします。

```
-- Lingo syntax
myMember = castLib("Internal").member[2]

// JavaScript syntax
var myMember = castLib("Internal").member[2];
```

関連事項

キャストライブラリ

member （ムービー）

使用法

```
-- Lingo syntax
_movie.member[memberNameOrNum]

// JavaScript syntax
_movie.member[memberNameOrNum];
```

解説

ムービーのキャストライブラリのメンバにインデックスまたは名前でアクセスできるムービープロパティです。読み取り専用。

memberNameOrNum パラメータには、キャストメンバの名前を表すストリング、またはキャストメンバの番号を表す整数を指定できます。

例文

次のステートメントは、名前と番号が付いているアクセスを使用してキャストメンバにアクセスし、結果を変数 myMember に設定します。

```
-- Lingo syntax
myMember = _movie.member[2] -- using numbered access
myMember = _movie.member["Athlete"] -- using named access

// JavaScript syntax
var myMember = _movie.member[2]; // using numbered access;
var myMember = _movie.member["Athlete"]; // using named access;
```

関連事項

ムービー

member （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.member

// JavaScript syntax
soundChannelObjRef.member;
```

解説

サウンドチャネルで再生中のサウンドキャストメンバを指定するサウンドチャネルプロパティです。読み取り専用。

サウンドが再生されていない場合は、VOID（Lingo）または null（JavaScript シンタックス）を返します。

例文

次のステートメントは、サウンドチャネル 2 で再生中のサウンドのメンバの情報をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sound(2).member)

// JavaScript syntax
put(sound(2).member);
```

関連事項

サウンドチャネル

member （サウンドオブジェクト）

使用法

```
soundObj.member (Read-only)
```

解説

現在のサウンドオブジェクトのキャストメンバの参照を返すサウンドオブジェクトプロパティです。ファイルパスを使用してサウンドオブジェクトを作成した場合は、メンバは NULL を示します。

例文

```
-- Lingo syntax
on mouseUp me
    put soundObjRef.  -- Displays the reference of the cast member of the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.mixer) ; // Displays the reference of the cast member of the sound
// object associated with soundobjectRef.

}
```

member （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.member

// JavaScript syntax
spriteObjRef.member;
```

解説

スプライトのキャストメンバとキャストライブラリを指定するスプライトプロパティです。読み取り / 書き込み可能。

member スプライトプロパティは、キャストライブラリを特定せずにキャスト内のメンバ位置を示す spriteNum スプライトプロパティとは異なります。member プロパティは、スプライトのキャストライブラリを示さないマウスオブジェクトの mouseMember プロパティとも異なります。

スプライトの member プロパティを割り当てる場合は、次のフォーマットを使用することができます。

- キャストメンバとキャストライブラリの指定（スプライトオブジェクト参照 .member = member(メンバ番号 {, キャストライブラリ名または番号 })）
- キャストメンバ名の指定スプライトオブジェクト参照 .member = member(" メンバ名 ")）
- すべてのキャストライブラリを含み、mouseMember プロパティに対応する一意の整数値の指定（スプライトオブジェクト参照 .member = 132）

キャストメンバ名のみを使う場合、すべての現行キャストライブラリの中でその名前と一致する最初のキャストメンバを探します。重複した名前がある場合は、最初の名前のみが認識されます。

複数のキャストが存在する場合は、キャストメンバを番号で指定します。この場合には、スプライトのキャストライブラリには影響を与えずにキャストライブラリ内のメンバの場所を変更する memberNum スプライトプロパティを使用します（**spriteObjRef**.memberNum = 10）。

スプライトチャネルに割り当てられたキャストメンバは、そのスプライトのプロパティの 1 つにすぎません。その他のプロパティは、スプライトのスコア内のチャネルにあるメディアエレメントの種類によって異なります。例えば、member スプライトプロパティを設定して、ビットマップを塗りつぶされていないシェイプに置き換えた場合、シェイプスプライトの lineSize プロパティは自動的に変更されないために、シェイプが表示されない可能性があります。

フィールドスプライトのメンバをビデオに変更した場合にも、同じようなスプライトプロパティの不一致が起きることがあります。一般的に、キャストメンバを同じようなキャストメンバに置き換える方が有効であり、確実です。例えば、ビットマップスプライトをビットマップキャストメンバに置き換えます。

例文

次のステートメントは、キャスト番号 4 のキャストメンバ 3 をスプライト 15 に割り当てます。

```
-- Lingo syntax
sprite(15).member = member(3, 4)

// JavaScript syntax
sprite(15).member = member(3, 4);
```

次のハンドラは、mouseMember 関数と sprite.member プロパティを使用して、マウスが特定のスプライトに置かれているかどうかを判断します。

```
-- Lingo syntax
on exitFrame
    mm = _mouse.mouseMember
    target = sprite(1).member
    if (target = mm) then
        put("Above the hotspot.")
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var mm = _mouse.mouseMember;
    var target = sprite(1).member;
    if (target == mm) {
        put("Above the hotspot.");
        _movie.go(_movie.frame);
    }
}
```

関連事項

lineSize, mouseMember, スプライト , spriteNum

memorySize

使用法

```
the memorySize
```

解説

使用中であるなしに関わらず、プログラムに割り当てられているメモリの容量を返すシステムプロパティです。最低限必要なメモリ量を確認するときに使うと便利です。この値の単位はバイトです。

Windows では、値は使用可能な全物理メモリであり、Mac では、値はアプリケーションに割り当てられているパーティション全体です。

例文

次のステートメントは、プログラムに割り当てられているメモリ量が 500K バイトより多いかどうか調べ、多い場合はアラートを表示します。

```
-- Lingo syntax
if the memorySize > 500 * 1024 then alert "There is enough memory to run this movie."

// JavaScript syntax
if ( _system.memorySize > 500 * 1024)
{
    _player.alert( "There is enough memory to run this movie.");
}
```

関連事項

freeBlock(), freeBytes(), ramNeeded(), size

meshDeform （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

解説

3D モディファイア。参照されたモデルのメッシュ構造の様々な属性を制御します。#meshDeform モディファイア (addModifier コマンドを使用) をモデルに追加すると、#meshDeform モディファイアの次のプロパティにアクセスできます。

注意：次のプロパティの詳細については、「関連事項」にある個別のエントリを参照してください。

- face.count。参照されたモデルのフェイスの合計数を返します。
- mesh.count。参照されたモデル内のメッシュの数を返します。
- mesh[index]。指定されたメッシュのプロパティにアクセスします。

例文

次のステートメントは、"gbFace" というモデル内のフェイス数を表示します。

```
put member("3D World").model("gbFace").meshDeform.face.count
-- 432
```

次のステートメントは、"gbFace" というモデル内のメッシュ数を表示します。

```
put member("3D World").model("gbFace").meshDeform.mesh.count
-- 2
```

次のステートメントは、"gbFace" というモデル内で 2 番目のメッシュのフェイス数を表示します。

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

関連事項

mesh （プロパティ）, addModifier

milliseconds

使用法

```
-- Lingo syntax
_system.milliseconds

// JavaScript syntax
_system.milliseconds;
```

解説

現在の時間をミリ秒 (1/1000 秒) 単位で返すシステムプロパティです。読み取り専用。

時間の計算は、コンピュータが起動した時点から始まります。

例文

次のステートメントは、ミリ秒の数を 1000 で割ってから 60 で割ることにより、ミリ秒を秒と分に変換し、その結果を変数 currentMinutes に割り当てます。

```
-- Lingo syntax
currentSeconds = _system.milliseconds/1000
currentMinutes = currentSeconds/60

// JavaScript syntax
var currentSeconds = _system.milliseconds/1000;
var currentMinutes = currentSeconds/60;
```

結果の精度は、コンピュータおよびオペレーティングシステムによって異なります。

次のハンドラは、milliseconds をカウントし、操作時間が長すぎると警告を表示します。

```
-- Lingo syntax
on idle
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) then
        _player.alert("Take a break")
    end if
end

// JavaScript syntax
function idle() {
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) {
        _player.alert("Take a break");
    }
}
```

関連事項

システム

minSpeed

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのパーティクルが放出される最低速度を取得または設定できます。各パーティクルの初期速度は、エミッタの minSpeed プロパティと maxSpeed プロパティの間でランダムに選択されます。

値は浮動小数点数で、0.0 より大きくなければなりません。

例文

次の例では、"3D オブジェクト " というキャストメンバでパーティクルの最低速度を 4 に設定します。

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.minSpeed=4

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.minSpeed=4;
```

関連事項

maxSpeed, emitter

missingFonts

使用法

```
member(textCastMember).missingFonts
```

解説

このメンバプロパティには、テキストで参照されるフォントの名前のリストが含まれます。ただし、現在、システムでは使用できません。

このプロパティを使用すると、開発者が実行時に特定のフォントを使用できるかどうかを確認することができます。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例ではテキストで参照されるフォントの名前のリストを示します。ただし、現在、システムでは使用できません。

```
-- Lingo syntax
put member(4).missingFonts

// JavaScript syntax
put(member(4).missingFonts);
```

関連事項

substituteFont

mixer

使用法

```
soundObject.mixer (Read-only)
```

解説

サウンドオブジェクトのミクサを返すサウンドオブジェクトプロパティです。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.mixer -- Displays the sound object's mixer.
end

//JavaScript syntax
function mouseUp(){
put (soundObjRef.mixer) ; // Displays the sound object's mixer.
}
```

mode (emitter)

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.mode
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのリソースのパーティクルエミッタの mode プロパティを取得および設定できます。

このプロパティの値は、#burst または #stream になります。デフォルト値は #stream です。mode プロパティの値が #burst の場合は、すべてのパーティクルが同時に生成されます。一方、値が #stream の場合は、パーティクルのグループがフレームごとに生成されます。各フレームで放出されるパーティクルの数は、次の式で決定されます。

```
particlesPerFrame = resourceObject.emitter.numParticles (resourceObject.lifetime x
millisecondsPerRenderedFrame)
```

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の emitter.mode プロパティを #burst に設定して、"ThermoSystem" のパーティクルがバースト内に表示されるようにします。パーティクルの単一のバーストを作成するには、emitter.mode = #burst および emitter.loop= 0 を設定します。

```
member("Fires").modelResource("ThermoSystem").emitter.mode = #burst
```

関連事項

emitter

mode (collision)

使用法

```
member(whichCastmember).model(whichModel).collision.mode
```

解説

3D collision モディファイアプロパティ。衝突検出アルゴリズムで使用されるジオメトリを示します。球体のようなシンプルなジオメトリを使用すると、パフォーマンスが向上します。このプロパティは、次のいずれかの値になります。

- #mesh。モデルリソースの実際のメッシュジオメトリを使用します。これにより 1 つの三角形の精度が指定されます。通常は、#box または #sphere よりも低速です。
- #box。モデルを囲むボックスを使用します。球体よりもボックスにフィットするオブジェクト（壁など）を使用するときに、役に立ちます。
- #sphere。モデルを囲むの球体を使用するので最速モードになります。このプロパティのデフォルト値です。

例文

次のステートメントは、任意のモデルという名前のモデルに衝突モディファイアを追加し、mode プロパティを #mesh に設定します。

```
member("3d").model("yourModel").addModifier(#collision)
member("3d").model("yourModel").collision.mode = #mesh
```

model

使用法

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モデルで指定された名前のモデル、またはインデックスで指定されたインデックス位置にあるモデルを返します。指定されたパラメータに該当するモデルが存在しない場合は void を返します。model.count を指定すると、このコマンドは参照されるキャストメンバ内のモデルの数を返します。このコマンドを使用して、指定されたモデルのプロパティにアクセスすることもできます。

モデル名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモデルのインデックス位置が変更される場合があります。

指定された名前またはインデックス位置に該当するモデルがない場合は、void を返します。

例文

次のステートメントは、" 仮のプレーヤ " というモデルへの参照を変数 thismodel に格納します。

```
thismodel = member("3DWorld").model("Player Avatar")
```

次のステートメントは、"3D ワールド " というキャストメンバの 8 番目のモデルへの参照を変数 thismodel に格納します。

```
thismodel = member("3DWorld").model[8]
```

次のステートメントは、スプライト 1 のメンバに 4 つのモデルがあることを示します。

```
put sprite(1).member.model.count
-- 4
```

modelA

使用法

```
collisionData.modelA
```

解説

3D collisionData プロパティ。衝突を起こしたモデルのうち、片方のモデルを示します。もう一方のモデルは modelB になります。

collisionData オブジェクトは、#collideWith および #collideAny イベントのパラメータとして、registerForEvent, registerScript、および setCollisionCallback の各コマンドによって指定されたハンドラに送られます。

衝突モディファイアが追加されているモデル間で衝突が発生すると、#collideWith イベントと #collideAny イベントが送信されます。このとき、モデルのモディファイアの resolve プロパティは、TRUE に設定されている必要があります。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、3 つの部分で構成されています。最初の部分はコードの 1 行目です。#collideAny イベントに #putDetails ハンドラを登録します。2 つ目の部分は #putDetails ハンドラです。"MyScene" というキャストメンバ内の 2 つのモデルが衝突した際に、#putDetails ハンドラが呼び出され、collisionData パラメータが渡されます。このハンドラは、collisionData オブジェクトの modelA および modelB プロパティをメッセージウィンドウに表示します。この例の 3 番目の部分では、メッセージウィンドウの結果を示します。これにより、"GreenBall" というモデルが衝突の modelA で、"YellowBall" というモデルが modelB であったことが示されます。

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

関連事項

registerScript(), registerForEvent(), sendEvent, modelB, setCollisionCallback()

modelB

使用法

```
collisionData.modelB
```

解説

3D collisionData プロパティ。衝突を起こしたモデルのうち、片方のモデルを示します。もう一方のモデルは modelA になります。

collisionData オブジェクトは、#collideWith および #collideAny イベントのパラメータとして、registerForEvent, registerScript、および setCollisionCallback の各コマンドによって指定されたハンドラに送られます。

衝突モディファイアが追加されているモデル間で衝突が発生すると、#collideWith イベントと #collideAny イベントが送信されます。このとき、モデルのモディファイアの resolve プロパティは、TRUE に設定されている必要があります。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、3 つの部分で構成されています。最初の部分はコードの 1 行目です。#collideAny イベントに #putDetails ハンドラを登録します。2 つ目の部分は #putDetails ハンドラです。"MyScene" というキャストメンバ内の 2 つのモデルが衝突した際に、#putDetails ハンドラが呼び出され、collisionData パラメータが渡されます。このハンドラは、collisionData オブジェクトの modelA および modelB プロパティをメッセージウィンドウに表示します。この例の 3 番目の部分では、メッセージウィンドウの結果を示します。これにより、"GreenBall" というモデルが衝突の modelA で、"YellowBall" というモデルが modelB であったことが示されます。

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

関連事項

registerScript(), registerForEvent(), sendEvent, modelA, collisionNormal, setCollisionCallback()

modelResource

使用法

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モデルリソースで指定された名前のモデルリソースか、またはインデックスで指定されるインデックス位置にあるモデルリソースを返します。指定されたパラメータに該当するモデルリソースが存在しない場合は void を返します。modelResource.count を指定すると、このコマンドは参照されるキャストメンバ内のモデルリソースの数を返します。このコマンドを使用して、指定されたモデルリソースのプロパティにアクセスすることもできます。

モデルリソース名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモデルリソースのインデックス位置が変更される場合があります。

例文

次のステートメントは "3D オブジェクト " というキャストメンバの最初のモデルリソースを示します。

```
-- Lingo syntax
putmember("3Dobjects").modelResource[1]

// JavaScript syntax
put( member("3Dobjects").getPropRef("modelResource",1));
```

modified

使用法

```
-- Lingo syntax
memberObjRef.modified

// JavaScript syntax
memberObjRef.modified;
```

解説

キャストメンバがムービーファイルから読み込まれた後に修正されているかどうかを示すメンバプロパティです。読み取り専用。

- modified プロパティが TRUE (1) のときは、キャストメンバはムービーファイルから読み込まれた後に修正されています。
- modified プロパティが FALSE (0) のときは、キャストメンバはムービーファイルから読み込まれた後に修正されていません。

例文

次のステートメントは、" イントロダクション " キャストメンバがムービーファイルから読み込まれた後に修正されているかどうかを返します。

```
-- Lingo syntax
if (member("Introduction").modified) then
    _player.alert("Introduction has been modified")
else
    _player.alert("Introduction has not been modified")
end if

// JavaScript syntax
if (member("Introduction").modified) {
    _player.alert("Introduction has been modified");
}
else {
    _player.alert("Introduction has not been modified");
}
```

関連事項

メンバ

modifiedBy

使用法

```
-- Lingo syntax
memberObjRef.modifiedBy

// JavaScript syntax
memberObjRef.modifiedBy;
```

解説

キャストメンバを最後に編集したユーザの名前を記録するメンバプロパティです。読み取り専用。

ユーザの名前は、Director のインストール時に入力されたユーザ名情報から得られます。この情報は、Director の [環境設定 一般] ダイアログボックスで変更できます。

このプロパティは、複数の作成者で Director のプロジェクトを作成するときに、作成者を管理するのに役に立ちます。このプロパティは、プロパティインスペクタの「メンバ」タブでも確認できます。

例文

次のステートメントは、キャストメンバ 1 を最後に修正したユーザを表示します。

```
-- Lingo syntax
put(member(1).modifiedBy)

// JavaScript syntax
put(member(1).modifiedBy);
```

関連事項

メンバ

modifiedDate

使用法

```
-- Lingo syntax
memberObjRef.modifiedDate

// JavaScript syntax
memberObjRef.modifiedDate;
```

解説

キャストメンバが最後に変更された日時を表示するメンバプロパティです。表示には、オーサリング時のコンピュータのシステム時間を使用します。読み取り専用。

このプロパティは、Director プロジェクトを追跡および調整するときに役に立ちます。また、このプロパティは、プロパティインスペクタの「メンバ」タブおよびキャストウィンドウのリストビューで確認できます。

例文

次のステートメントは、キャストメンバ 1 に最後に変更が行われた日付を表示します。

```
-- Lingo syntax
put(member(1).modifiedDate)

// JavaScript syntax
put(member(1).modifiedDate);
```

関連事項

メンバ

modifier

使用法

```
member(whichCastmember).model(whichModel).modifier
member(whichCastmember).model(whichModel).modifier.count
```

解説

3D モデルプロパティ。指定されたモデルに割り当てられている全モディファイアのリストを返します。modifier.count を指定すると、このコマンドはモデルに割り当てられたモディファイア数を返します。

toon および inker モディファイアの両方がモデルに適用される場合、モデルに最初に追加されたモディファイアが返されます。

このプロパティは、値を調べられますが、設定はできません。モディファイアをモデルに追加およびモデルから削除するには、addModifier コマンドおよび removeModifier コマンドを使用します。

例文

次のステートメントは、"3D オブジェクト " というキャストメンバの " 球体 01" というモデルにどのモディファイアが割り当てられているかを示します。

```
-- Lingo syntax
put member("3Dobjects").model("Sphere01").modifier

// JavaScript syntax
put( member("3Dobjects").getPropRef("model",2).modifier);
```

関連事項

modifier[], modifiers, addModifier, removeModifier

modifier[]

使用法

```
member(whichCastmember).model(whichModel).modifier[index]
```

解説

3D モデルプロパティ。モデルに割り当てられたモディファイアリストから、**index** で指定された位置にあるモディファイアのタイプを返します。返される値はシンボルです。

指定された位置にモディファイアがない場合、このプロパティの値は void になります。

モデルに割り当てられたモディファイアについての情報を取得するには、modifier プロパティを使用します。

このコマンドでは、添付されたモディファイアのプロパティに直接アクセスすることはできません。

例文

```
put member("3d world").model("box").modifier[1]
-- #lod
```

関連事項

modifier, modifiers, addModifier, removeModifier

modifiers

使用法

```
getRendererServices().modifiers
```

解説

グローバル 3D プロパティ。3D キャストメンバ内のモデルで使用できるモディファイアのリストを返します。

例文

次のステートメントは、現在使用できるすべてのモディファイアのリストを返します。

```
-- Lingo syntax
put getRendererServices().modifiers

// JavaScript syntax
put(getRendererServices().modifiers);
```

関連事項

getRendererServices(), addModifier

mostRecentCuePoint

使用法

```
-- Lingo syntax
spriteObjRef.mostRecentCuePoint

// JavaScript syntax
spriteObjRef.mostRecentCuePoint;
```

解説

スプライトまたはサウンドで最後に通過したキューポイントを識別するサウンドチャネルおよびスプライトプロパティです。キューポイントをサポートするサウンドスプライト、QuickTime デジタルビデオ、および Xtra 拡張機能で使用できます。値はキューポイントの順番を示します。キューポイントを通過していない場合は、値は 0 になります。

Shockwave Audio（SWA）サウンドはスプライトチャネルにスプライトとして表示されますが、そのサウンドはサウンドチャネルで再生されます。SWA サウンドスプライトは、サウンドチャネル番号ではなく、スプライトチャネル番号で参照することをお勧めします。

例文

次のステートメントは、スプライトチャネル 1 で最後に通過したキューポイントの番号をメッセージウィンドウに表示します。

```
-- Lingo syntax
put sprite(1).mostRecentCuePoint

// JavaScript syntax
put(sprite(1).mostRecentCuePoint);
```

次のステートメントは、サウンドチャネル 2 で再生中に直前に通過したキューポイントの順番を示す値を返します。

```
-- Lingo syntax
put sound(2).mostRecentCuePoint

// JavaScript syntax
put(sound(2).mostRecentCuePoint);
```

関連事項

cuePointNames, isPastCuePoint(), cuePointTimes, on cuePassed

mostRecentCuePoint （サウンドオブジェクト）

使用法

```
soundobj.mostRecentCuePoint (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトがヒットした最新のキューポイントを返します。

例文

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.mostRecentCuePoint -- Returns the most recent cue point hit by
-- the sound object.
end

// JavaScript syntax
function exitFrame(){
put (soundObjRef.mostRecentCuePoint) ; // Returns the most recent cue point hit by
// the sound object.

}
```

motion

使用法

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

解説

3D コマンド。参照されたキャストメンバに含まれる、モーションで指定された名前のモーションか、またはインデックスで指定されるインデックス位置にあるモーションを返します。motion.count を指定すると、そのキャストメンバ内にあるモーションの合計数を返します。

オブジェクト名のストリング比較では大文字と小文字は区別されません。数が小さいインデックス位置に存在するオブジェクトを削除すると、特定のモーションのインデックス位置が変更される場合があります。

指定された名前またはインデックス位置に該当するモーションがない場合は、void を返します。

例文

次のステートメントは、参照したキャストメンバ内の最初のモーションを表示します。

```
-- Lingo syntax
put member("3Dobjects").motion[1]

// JavaScript syntax
put(member("3Dobjects").getPropRef("motion",1));
```

関連事項

duration (3D), map (3D)

motionQuality

使用法

```
-- Lingo syntax
spriteObjRef.motionQuality

// JavaScript syntax
spriteObjRef.motionQuality;
```

解説

QuickTime VR スプライトをクリックし、ドラッグしたときに使用されるコーデックの品質を設定する QuickTime VR スプライトプロパティです。このプロパティの値は、#minQuality、#maxQuality、#normalQuality のいずれかです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、スプライト 1 の motionQuality を #minQuality に設定します。

```
-- Lingo syntax
sprite(1).motionQuality = #minQuality

// JavaScript syntax
sprite(1).motionQuality = symbol("minQuality");
```

mouseChar

使用法

```
-- Lingo syntax
_mouse.mouseChar

// JavaScript syntax
_mouse.mouseChar;
```

解説

フィールドスプライト用のマウスプロパティで、プロパティが呼ばれたときにポインタの下に位置している文字の番号を返します。読み取り専用。

番号はフィールドの先頭から数えた文字数で、マウスポインタがフィールド上になかったり、フィールド内の文字以外の場所に位置していた場合、結果は -1 になります。

mouseChar の値は、ハンドラ内やループ内で変化することがあります。ハンドラやループがこのプロパティを何度も使用する場合、プロパティを 1 回呼び出し、その値をローカル変数に割り当てることをお勧めします。

例文

次のステートメントは、ポインタがフィールドスプライト上に位置しているかどうかを調べ、位置していない場合は、" 指示 " フィールドキャストメンバの内容を「文字を指してください。」に変更します。

```
-- Lingo syntax
if (_mouse.mouseChar = -1) then
    member("Instructions").text = "Please point to a character."
end if

// JavaScript syntax
if (_mouse.mouseChar == -1) {
    member("Instructions").text = "Please point to a character.";
}
```

次のステートメントは、ポインタの指すフィールドでポインタの下に位置している文字を変数 currentChar に割り当てます。

```
-- Lingo syntax
currentChar = member(_mouse.mouseMember).char[_mouse.mouseChar]

// JavaScript syntax
var currentChar = member(_mouse.mouseMember).getProp("char",_mouse.mouseChar);
```

関連事項

マウス , mouseItem, mouseLine

mouseDown

使用法

```
-- Lingo syntax
_mouse.mouseDown

// JavaScript syntax
_mouse.mouseDown;
```

解説

マウスボタンが現在押されているかどうかを示すマウスプロパティです。TRUE のときは、ボタンは押されています。FALSE のときは、ボタンは押されていません。読み取り専用。

例文

スプライトに割り当てられている次の mouseEnter ハンドラは、マウスがスプライト上に位置したときに マウスボタンが押されていない場合は 1 つのハンドラを呼び出し、マウスがスプライト上に位置したときに マウスボタンが押されていない場合は別のハンドラを呼び出します。

```
-- Lingo syntax
on mouseEnter
    if (_mouse.mouseDown) then
        runMouseDownScript
    else
        runMouseUpScript
    end if
end

// JavaScript syntax
function mouseEnter() {
    if (_mouse.mouseDown) {
        runMouseDownScript();
    }
    else {
        runMouseUpScript();
    }
}
```

関連事項

マウス , on mouseDown (イベントハンドラ), mouseH, mouseUp, on mouseUp (イベントハンドラ), mouseV

mouseDownScript

使用法

```
the mouseDownScript
```

解説

マウスボタンが押されたときに実行する Lingo を指定するためのシステムプロパティです。Lingo は、引用符で囲まれたストリングとして記述します。簡単なステートメント、またはハンドラを呼び出すスクリプトなどを指定することができます。デフォルトの値は EMPTY です。この場合、mouseDownScript プロパティに割り当てられている Lingo はありません。

mouseDownScript プロパティが定義されているときにマウスボタンが押されると、Lingo は mouseDownScript プロパティの設定を優先して実行します。mouseDown メッセージをムービー内の他のオブジェクトに渡すことができる pass コマンドが含まれている場合、その他の on mouseDown ハンドラが実行されます。

mouseDownScript プロパティを設定することは、旧バージョンの Director で使われていた when keyDown then コマンドを使うのと同じことになります。

mouseDownScript プロパティで指定した指示を解除するには、set the mouseDownScript to empty ステートメントを使用します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、ユーザがマウスボタンをクリックすると実行されるスクリプトを指定します。

```
-- Lingo syntax
the mouseDownScript = "go to the frame"

// JavaScript syntax
_system.mouseDownScript = "_movie.go(_movie.frame)";
```

関連事項

stopEvent(), mouseUpScript, on mouseDown (イベントハンドラ), on mouseUp (イベントハンドラ)

mouseH

使用法

```
-- Lingo syntax
_mouse.mouseH

// JavaScript syntax
_mouse.mouseH;
```

解説

マウスポインタの水平位置を調べるためのマウスプロパティです。読み取り専用。

mouseH の値は、ステージの左端からカーソル位置までのピクセル数になります。

mouseH プロパティは、スプライトをマウスポインタの水平位置まで移動するときや、マウスポインタがステージの範囲内にあるかどうかを調べるときに使用すると便利です。mouseH プロパティと mouseV プロパティを組み合わせて使用すると、カーソルの正確な位置を把握することができます。

例文

次のハンドラは、スプライト 10 をマウスポインタの位置まで移動し、ユーザがマウスボタンをクリックした時点でステージを更新します。

```
-- Lingo syntax
on mouseDown
    sprite(10).locH = _mouse.mouseH
    sprite(10).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(10).locH = _mouse.mouseH;
    sprite(10).locV = _mouse.mouseV;
}
```

次のステートメントは、ポインタが開始点から左右 10 ピクセルの範囲よりも遠くにあるかどうかを調べ、遠くにある場合は変数 Far を TRUE に設定します。

```
-- Lingo syntax
startH = 7
if (abs(_mouse.mouseH - startH) > 10) then
    Far = TRUE
end if

// JavaScript syntax
var startH = 7;
if (Math.abs(_mouse.mouseH - startH) > 10) {
    var Far = true;
}
```

関連事項

locH, locV, マウス , mouseLoc, mouseV

mouseItem

使用法

```
-- Lingo syntax
_mouse.mouseItem

// JavaScript syntax
_mouse.mouseItem;
```

解説

ポインタがフィールドスプライト上にある状態でプロパティが呼び出されたときに、ポインタの下にある項目の番号を返すマウスプロパティです。読み取り専用。

the itemDelimiter プロパティで設定された区切り文字で区切られた一連の文字が項目として扱われます。番号はフィールドの先頭から数えたものになり、マウスポインタがフィールド上に位置していないときに呼び出すと、結果は -1 になります。

mouseItem プロパティの値は、ハンドラ内やループ内で変化することがあります。ハンドラやループの開始時に mouseItem の初期値を必要とする場合は、このプロパティを 1 回呼び出し、その値をローカル変数に割り当てます。

例文

次のステートメントは、ポインタがフィールドスプライト上に位置しているかどうかを調べ、位置していない場合は、" 指示 " フィールドキャストメンバの内容を「項目を指定してください。」に変更します。

```
-- Lingo syntax
if (mouse.mouseItem = -1) then
    member("Instructions").text = "Please point to an item."
end if

// JavaScript syntax
if (_mouse.mouseItem == -1) {
    member("Instructions").text = "Please point to an item.";
}
```

次のステートメントは、ポインタの指すフィールド内でポインタの下にある項目を変数 currentItem に割り当てます。

```
-- Lingo syntax
currentItem = member(_mouse.mouseMember).item[_mouse.mouseItem]

// JavaScript syntax
var currentItem = member(_mouse.mouseMember).getProp("item",_mouse.mouseItem);
```

関連事項

itemDelimiter, マウス , mouseChar, mouseLine, mouseWord

mouseLevel

使用法

```
-- Lingo syntax
spriteObjRef.mouseLevel

// JavaScript syntax
spriteObjRef.mouseLevel;
```

解説

Director が [QuickTime スプライト] 上でのマウスクリックを QuickTime に渡す方法を管理する QuickTime スプライトプロパティです。スプライトの境界矩形内のマウスクリックを渡す機能は、QuickTime VR などのインタラクティブなメディアに対して使用できます。mouseLevel スプライトプロパティには、次のような値を設定できます。

- #controller - ムービーコントローラでのクリックだけを QuickTime に渡します。Director はコントローラの外側で行ったマウスクリックのみに応答します。これは、QuickTime VR 以外の QuickTime スプライトの標準的な挙動です。
- #all - スプライトの境界矩形内部でのすべてのマウスクリックを QuickTime に渡します。他の Lingo ハンドラに渡されるクリックはありません。
- #none - QuickTime にはマウスクリックを渡しません。Director はすべてのマウスクリックに応答します。
- #shared - QuickTime VR スプライトの境界矩形内でのすべてのマウスクリックを QuickTime に渡し、次に、これらのイベントを Lingo ハンドラに渡します。これは QuickTime VR のデフォルト値です。

このプロパティは、値を調べたり、設定することができます。

例文

このフレームスクリプトはチャネル 5 の QuickTime の名前にストリング「QTVR」が含まれるかどうかを調べます。含まれる場合、このスクリプトは、mouseLevel を #all に設定し、含まれない場合は、mouseLevel を #none に設定します。

```
on prepareFrame
    if sprite(5).member.name contains "QTVR" then
        sprite(5).mouseLevel = #all
    else
        sprite(5).mouseLevel = #none
    end if
end

// JavaScript syntax
function prepareFrame() {
    var nm = sprite(5).member.name;
    var nmStr = nm.indexOf("QTVR");
    if (nmStr != -1) {
        sprite(5).mouseLevel = symbol("all");
    } else {
        sprite(5).mouseLevel = symbol("none");
    }
}
```

mouseLine

使用法

```
-- Lingo syntax
_mouse.mouseLine

// JavaScript syntax
_mouse.mouseLine;
```

解説

ポインタがフィールドスプライト上にある状態でプロパティが呼び出されたときに、ポインタの下にある行の番号を返すマウスプロパティです。読み取り専用。

番号はフィールドの先頭から数えます。実際の改行コードの数であり、ストリングの自動回り込みによる改行は数えません。ポインタがフィールドスプライト上に位置していないときに呼び出すと、結果は -1 になります。

mouseLine プロパティの値は、ハンドラ内やループ内で変化することがあります。ハンドラやループがこのプロパティを何度も使用する場合、プロパティを 1 回呼び出し、その値をローカル変数に割り当てることをお勧めします。

例文

次のステートメントは、ポインタがフィールドスプライト上に位置しているかどうかを調べ、位置していない場合は、" 指示 " フィールドキャストメンバの内容を「行を指してください。」に変更します。

```
-- Lingo syntax
if (_mouse.mouseLine = -1) then
    member("Instructions").text = "Please point to a line."
end if

// JavaScript syntax
if (_mouse.mouseLine == -1) {
    member("Instructions").text = "Please point to a line.";
}
```

次のステートメントは、ポインタの指すフィールドでポインタの下に位置している行の内容を変数 currentLine に割り当てます。

```
-- Lingo syntax
currentLine = member(_mouse.mouseMember).line[_mouse.mouseLine]

// JavaScript syntax
var currentLine = member(_mouse.mouseMember).getProp("line",_mouse.mouseLine);
```

関連事項

マウス , mouseChar, mouseItem, mouseWord

mouseLoc

使用法

```
-- Lingo syntax
_mouse.mouseLoc

// JavaScript syntax
_mouse.mouseLoc;
```

解説

point() としてマウスの現在の位置を返すマウスプロパティです。読み取り専用。

ポイントの位置は、水平位置、垂直位置の順で示される 2 つの座標で表されます。

例文

次のステートメントは、マウスの現在の位置を示します。

```
-- Lingo syntax
trace(_mouse.mouseLoc)

// JavaScript syntax
trace(_mouse.mouseLoc);
```

関連事項

マウス , mouseH, mouseV

mouseMember

使用法

```
-- Lingo syntax
_mouse.mouseMember

// JavaScript syntax
_mouse.mouseMember;
```

解説

プロパティが呼び出されたときに、ポインタの下にあるスプライトに割り当てられているキャストメンバを返すマウスプロパティです。読み取り専用。

ポインタがスプライト上にない場合、結果は VOID（Lingo）または null（JavaScript シンタックス）になります。

このプロパティは特定のキャストメンバを使うスプライトの上にポインタが移動したときに、ムービーに特定の動作を実行させるときなどに有効です。

mouseMember プロパティの値は頻繁に変化することがあります。この関数が返す値をハンドラで何度も使うには、ハンドラが開始したときに mouseMember の値をローカル変数に割り当て、その変数を使います。

例文

次のステートメントは、キャストメンバ " 立ち入り禁止 " が、ポインタの下にあるスプライトに割り当てられているかどうかを調べ、割り当てられている場合は警告を表示します。この例は、スプライトに割り当てられたキャストメンバに基づいて動作を指定する方法を示しています。

```
-- Lingo syntax
if (_mouse.mouseMember = member("Off Limits")) then
    _player.alert("Stay away from there!")
end if

// JavaScript syntax
if (_mouse.mouseMember == member("Off Limits")) {
    _player.alert("Stay away from there!");
}
```

次のステートメントは、ポインタの下にあるスプライトのキャストメンバを変数 lastMember に割り当てます。

```
-- Lingo syntax
lastMember = _mouse.mouseMember

// JavaScript syntax
var lastMember = _mouse.mouseMember;
```

関連事項

castLibNum, マウス

mouseOverButton

使用法

```
-- Lingo syntax
spriteObjRef.mouseOverButton
// JavaScript syntax
spriteObjRef.mouseOverButton;
```

解説

マウスポインタが、[**Flash** スプライト] パラメータで指定された Flash ムービースプライト内のボタン上にあるかどうかを調べる Flash スプライトプロパティです。ボタン上にある場合は、TRUE になります。マウスポインタがスプライトの領域外にあるか、または領域内にあるがバックグラウンドなどのボタン以外のオブジェクトの上にある場合は、FALSE になります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のフレームスクリプトは、マウスポインタがスプライト 3 の Flash ムービー内のナビゲーションボタン上にあるかどうかを調べます。ボタン上にある場合、スクリプトは適切なメッセージでテキストフィールドを更新し、ない場合はメッセージを消去します。

```
-- Lingo syntax
on enterFrame
    case sprite(3).mouseOverButton of
        TRUE:
        member("Message Line").text = "Click here to go to the next page."
        FALSE:
        member("Message Line").text = " "
    end case
    _movie.updatestage()
end

// JavaScript syntax
function enterFrame() {
    switch(sprite(3).mouseOverButton)
    {
        case 1:
            member("Message Line").text = "Click here to go to the next page.";
            break;
        case 0:
            member("Message Line").text = " ";
            break;
    }
    _movie.updatestage();
}
```

mouseUp

使用法

```
-- Lingo syntax
_mouse.mouseUp

// JavaScript syntax
_mouse.mouseUp;
```

解説

マウスボタンが離されたかどうかを確認するためのマウスプロパティです。離されているときは TRUE、押されているときは FALSE になります。読み取り専用。

例文

次のハンドラは、ユーザがマウスボタンを押している限りムービーを実行します。ユーザがマウスボタンを離すと再生ヘッドが停止します。

```
-- Lingo syntax
on exitFrame me
    if (_mouse.mouseUp) then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
     if (_mouse.mouseUp) {
        _movie.go(_movie.frame);
    }
}
```

次のステートメントは、ユーザがマウスボタンを離した時点で、繰り返しループやハンドラを終了します。

```
-- Lingo syntax
if (_mouse.mouseUp) then exit

// JavaScript syntax
if (_mouse.mouseUp) {
    return;
}
```

関連事項

マウス , mouseDown, mouseH, mouseV

mouseUpScript

使用法

```
the mouseUpScript
```

解説

マウスボタンが離されたときに実行する Lingo を定義するシステムプロパティです。二重引用符で囲んだストリングとして、簡単なステートメントやハンドラを呼び出すスクリプトを記述します。

mouseUpScript が定義されているときにキーが離されると、Lingo は mouseUpScript の指定を最優先します。mouseUp メッセージをムービー内の他のオブジェクトに渡すことができる pass コマンドが命令に含まれない限り、その他の on mouseUp ハンドラは実行されません。

mouseUpScript プロパティに指定した命令が不適切になった場合、set the mouseUpScript to empty ステートメントを使用してこれを解除します。

mouseUpScript プロパティを設定することは、以前のバージョンの Director で使われていた when mouseUp then コマンドを使うのと同じことになります。

このプロパティは、値を調べたり、設定することができます。デフォルト値は EMPTY です。

例文

次のステートメントが有効なときにムービーが一時停止した場合、ユーザがマウスボタンを離すとムービーが常に継続されます。

```
the mouseUpScript = "go to the frame +1"
```

次のステートメントは、ユーザがステージ上のどこかをクリックしてからマウスボタンを離すとムービーがピープ音を鳴らすようにします。

```
the mouseUpScript = "if the clickOn = 0 then beep"
```

次のステートメントは、mouseUpScript にカスタムハンドラ **myCustomHandler** を設定します。Lingo のカスタムハンドラは、mouseUpScript プロパティで使用されるときは二重引用符で囲む必要があります。

```
the mouseUpScript = "myCustomHandler"
```

関連事項

stopEvent(), mouseDownScript, on mouseDown (イベントハンドラ), on mouseUp (イベントハンドラ)

mouseV

使用法

```
-- Lingo syntax
_mouse.mouseV

// JavaScript syntax
_mouse.mouseV;
```

解説

マウスポインタの垂直位置をステージ上部からのピクセル数で示すマウスプロパティです。読み取り専用。

このプロパティの値はカーソルが下がると増加し、上がると減少します。

mouseV プロパティは、スプライトをカーソルの垂直位置まで移動するときや、マウスポインタがステージの範囲内にあるかどうかを調べるときに使うと便利です。mouseH プロパティと mouseV プロパティを組み合わせて使用すると、カーソルの正確な位置を把握することができます。

例文

次のハンドラは、スプライト 1 をマウスポインタの位置まで移動し、ユーザがマウスボタンをクリックした時点でステージを更新します。

```
-- Lingo syntax
on mouseDown
    sprite(1).locH = _mouse.mouseH
    sprite(1).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(1).locH = _mouse.mouseH;
    sprite(1).locV = _mouse.mouseV;
}
```

次のステートメントは、ポインタが開始点から上下 10 ピクセルの範囲よりも遠くにあるかどうかを調べ、遠くにある場合は vFar 変数を TRUE に設定します。

```
-- Lingo syntax
startV = 7
if (abs(_mouse.mouseV - startV) > 10) then
    vFar = TRUE
end if

// JavaScript syntax
var startV = 7
if (Math.abs(_mouse.mouseV - startV) > 10) {
    var vFar = true;
}
```

関連事項

locH, locV, マウス , mouseH, mouseLoc

mouseWord

使用法

```
-- Lingo syntax
_mouse.mouseWord

// JavaScript syntax
_mouse.mouseWord;
```

解説

ポインタがフィールドスプライト上にある状態でプロパティが呼び出されたときに、ポインタの下にある単語の位置を返すマウスプロパティです。読み取り専用。

番号はフィールドの先頭から数えたものになり、ポインタがフィールド上に位置していないときに呼び出すと、結果は -1 になります。

mouseWord プロパティの値は、ハンドラ内やループ内で変化することがあります。ハンドラやループがこのプロパティを何度も使用する場合、関数を 1 回呼び出し、その値をローカル変数に割り当てることをお勧めします。

例文

次のステートメントは、ポインタがフィールドスプライト上に位置しているかどうかを調べ、位置していない場合は、" 指示 " フィールドキャストメンバの内容を「単語を指してください。」に変更します。

```
-- Lingo syntax
if (_mouse.mouseWord = -1) then
    member("Instructions").text = "Please point to a word."
else
    member("Instructions").text = "Thank you."
end if

// JavaScript syntax
if (_mouse.mouseWord == -1) {
    member("Instructions").text = "Please point to a word.";
}
else {
    member("Instructions").text = "Thank you.";
}
```

次のステートメントは、指定したフィールド内でポインタの下にある単語の番号を currentWord 変数に割り当てます。

```
-- Lingo syntax
currentWord = member(_mouse.mouseMember).word[_mouse.mouseWord]

// JavaScript syntax
var currentWord = member(_mouse.mouseMember).getProp("word",_mouse.mouseWord);
```

関連事項

マウス , mouseChar, mouseItem

moveableSprite

使用法

```
sprite(whichSprite).moveableSprite
the moveableSprite of sprite whichSprite
```

解説

スプライトが移動可能かどうかを示すスプライトプロパティです。ユーザがスプライトを移動できる場合は TRUE、移動できない場合は FALSE になります。

スコア内の [移動可能] オプションを使ってスプライトを移動可能にすることもできますが、Lingo を使って制御すると、ムービー内で必要な状況に応じて自由にこの設定のオンやオフを切り替えることができます。例えば、必要に応じて moveableSprite のスプライトプロパティのオン / オフを切り替えることで、ユーザがステージ上のスプライトを 1 つずつドラッグできるようにし、ある特定の領域に置いた時点で移動できなくなるようにすることもできます。

注意：元の座標へのスナップバックや、ドラッグ中のアニメーション化など、よりカスタマイズ化された機能は、ビヘイビアーを作って処理します。

このプロパティは、値を調べたり、設定することができます。

例文

次のハンドラは、チャネル 5 内のスプライトを移動可能にします。

```
on spriteMove
     sprite(5).moveableSprite = TRUE
end
```

次のステートメントは、スプライトを移動できるかどうかを調べ、移動できない場合はメッセージを表示します。

```
if sprite(13).moveableSprite = FALSE thenmember("Notice").text = "You can't drag this item by using the
mouse."
```

関連事項

mouseLoc

movie

使用法

```
-- Lingo syntax
windowObjRef.movie

// JavaScript syntax
windowObjRef.movie;
```

解説

指定されたウィンドウで再生中のムービーオブジェクトへの参照を返すウィンドウプロパティです。読み取り専用。

例文

次のステートメントは、帝国というウィンドウで再生しているムービーオブジェクトをメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(window("Empires").movie)

// JavaScript syntax
trace(window("Empires").movie);
```

関連事項

ウィンドウ

multiSound

使用法

```
the multiSound
```

解説

システムが複数のサウンドチャネルをサポートしているかを示すシステムプロパティです。Windows の場合、マルチチャネル対応のサウンドカードがインストールされているときは TRUE、インストールされていないときは FALSE になります。

例文

次のステートメントは、コンピュータが複数のサウンドチャネルをサポートしているときに、サウンドチャネル 2 で " 音楽 .wav" サウンドファイルを再生します。

```
if the multiSound then sound playFile 2, "Music.wav"
```

name

使用法

```
-- Lingo syntax
castObjRef.name
memberObjRef.name
_movie.name
windowObjRef.name

// JavaScript syntax
castObjRef.name;
memberObjRef.name;
_movie.name;
windowObjRef.name;
```

解説

キャスト、メンバ、ムービー、およびウィンドウのプロパティ。オブジェクト名を返すか、設定します。キャスト、メンバ、およびウィンドウのオブジェクトでは読み取り / 書き込み可能、ムービーオブジェクトの場合は読み取り専用。

例文

次のステートメントはキャストメンバ 1 の名前を " 新しい名前 " に変更します。

```
-- Lingo syntax
member(1).name="Newname"

// JavaScript syntax
member(1).name="Newname";
```

関連事項

キャストライブラリ , メンバ , ムービー , ウィンドウ

name (3D)

使用法

```
member(whichCastmember).texture(whichTexture).name
member(whichCastmember).shader(whichShader).name
member(whichCastmember).motion(whichMotion).name
member(whichCastmember).modelResource(whichModelResource).name
member(whichCastmember).model(whichModel).name
member(whichCastmember).light(whichLight).name
member(whichCastmember).camera(whichCamera).name
member(whichCastmember).group(whichGroup).name
node.name
```

解説

3D プロパティ。オブジェクト参照とともに使用すると、参照されたオブジェクトの名前を取得できます。このプロパティでは、名前の取得のみが可能であり、名前の変更はできません。

名前はすべて一意である必要があります。Lingo で作成された場合、コンストラクタ関数に指定された名前が返されます。3D オーサリングプログラムで作成された場合、モデルの名前が返されることがあります。

例文

次のステートメントは、キャストメンバ "TableScene" 内の 5 番目のカメラ名を "BirdCam" に設定します。

```
member("TableScene").camera[5].name = "BirdCam"
```

name （メニュープロパティ）

使用法

```
the name of menu(whichMenu)
the name of menu whichMenu
```

解説

指定したメニュー番号の名前を含むストリングを返すメニュープロパティです。

このプロパティは、値を調べられますが、設定はできません。カスタムメニューバーを設定するには、installMenu コマンドを使用します。

注意：メニューは、Shockwave Player では使用できません。

例文

次のステートメントは、メニュー番号 1 の名前を変数 firstMenu に割り当てます。

```
firstMenu = menu(1).name
```

次のハンドラは、メニュー名のリストを 1 行に 1 つずつ返します。

```
on menuList
    theList = []
    repeat with i = 1 to the number of menus
        theList[i] = the name of menu i
    end repeat
    return theList
end menuList
```

関連事項

number （メニュー）, name （メニュー項目プロパティ）

name （メニュー項目プロパティ）

使用方法

```
the name of menuItem(whichItem) of menu(whichMenu)
the name of menuItem whichItem of menu whichMenu
```

解説

[メニュー] で指定したメニューの、[項目] で指定したメニュー項目に表示されるテキストを定義するメニュープロパティです。[項目] パラメータにはメニュー項目名またはメニュー項目番号を指定でき、[メニュー] にはメニュー名またはメニュー番号を指定できます。

このプロパティは、値を調べたり、設定することができます。

注意：メニューは、Shockwave Player では使用できません。

例文

次のステートメントは、編集メニューの 8 番目の項目名を変数 itemName に設定します。

```
set itemName = the name of menuItem(8) of menu("Edit")
```

次のステートメントは、ファイルメニューにある「開く」という項目名に特定のファイル名を追加します。

```
the name of menuItem("Open") of menu("fileMenu") = "Open" && fileName
```

関連事項

name （メニュープロパティ）, number （メニュー項目）

name （ミクサ）

使用法

```
mixer.name
```

解説

ミクサキャストメンバの名前を設定または返すオーディオミクサプロパティです。このプロパティは読み書き可能です。

例文

```
--Lingo syntax
on mouseUp me
    mixerRef.name = "BackgroundMixer"
    put mixerRef.name -- Displays the name of the mixer associated with mixerRef.
end

// JavaScript syntax
function mouseUp(){
    mixerRef.name = "BackgroundMixer"
    put (mixerRef.name) ; // Displays the name of the mixer associated with mixerRef.
}
```

関連事項

ミクサ

name （サウンドオブジェクト）

使用法

```
soundObject.name (Read-write)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトの名前を返します。このプロパティは読み書き可能です。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.name -- Displays the name of the sound object associated with
-- soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.name) ; // Displays the name of the sound object associated with
// soundobjectRef.
}
```

name （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.name

// JavaScript syntax
spriteObjRef.name;
```

解説

スプライトの名前を定義するスプライトプロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

backColor、blend などのスプライト表示プロパティとは異なり、スプライト name をスクリプトしたスプライトにすることはできません。このため、name の設定は、スコア録画セッション中、つまりムービーオブジェクトの beginRecording() メソッド呼び出しと endRecording() メソッド呼び出しの間にのみ行うことができます。また、スプライトを含むスコア内のフレームがくるまでに beginRecording() を呼び出した場合にのみ、name を設定できます。

注意：beginRecording() を使用してスコア録画セッションを開始すると、すべてのスクリプトしたスプライトとスプライトチャネルのプロパティがリセットされます。

スクリプトを使用してスコア録画セッション中に新しいスプライトを作成し、updateFrame() を使用して スプライトデータをそのセッションに適用した場合は、スプライトを作成したフレームに戻らない限り、スプライト名を設定できません。特定のフレームに戻るには、go() などのメソッドを使用します。

例文

次のステートメントはスプライト 5 の名前を " 効果音 " に変更します。

```
-- Lingo syntax
sprite(5).name = "Background Sound"

// JavaScript syntax
sprite(5).name = "Background Sound";
```

関連事項

beginRecording(), endRecording(), go(), スプライト , updateFrame()

name （スプライトチャネル）

使用法

```
-- Lingo syntax
spriteChannelObjRef.name

// JavaScript syntax
spriteChannelObjRef.name;
```

解説

スプライトチャネルの名前を定義するスプライトチャネルプロパティです。スコア録画セッション中のみ読み取り / 書き込み可能。

スプライトチャネルの name は、スコア録画セッション中、つまりムービーオブジェクトの beginRecording() メソッド呼び出しと endRecording() メソッド呼び出しの間に設定します。

注意：beginRecording() を使用してスコア録画セッションを開始すると、すべてのスクリプトしたスプライトとスプライトチャネルのプロパティがリセットされます。

Sprite Channel オブジェクトの name プロパティは、スプライトがスコア内に表示されるフレーム以降で設定可能なスプライトオブジェクトの name プロパティとは異なり、空のチャネル上で設定できます。つまり、スプライトチャネルの name を設定する前に updateFrame() を呼び出す必要はありません。

スクリプトを使用してスプライトチャネルの名前を変更しても、スコアウィンドウには反映されません。

例文

次のステートメントは、スコア録画セッション中にスプライトチャネル 6 の名前を " 凧糸 " に設定します。

```
-- Lingo syntax
on mouseDown
    _movie.beginRecording()
    channel(6).name = "Kite string"
    _movie.endRecording()
end

// JavaScript syntax
function mouseDown() {
    _movie.beginRecording();
    channel(6).name = "Kite string";
    _movie.endRecording();
}
```

関連事項

beginRecording(), endRecording(), スプライトチャネル

name (timeout)

使用法

```
timeoutObject.name
```

解説

このタイムアウトプロパティは、オブジェクト作成の際に定義されたタイムアウトオブジェクト名です。タイムアウトオブジェクトの作成には new() コマンドを使用します。

例文

次のタイムアウトハンドラは、イベントを送ったタイムアウトの名前をアラートによって示します。

```
on handleTimeout timeoutObject
    alert "Timeout:" && timeoutObject.name
end
```

関連事項

forget() (Timeout), new(), period, persistent, target, time （タイムアウトオブジェクト）, timeout(), timeoutHandler, timeoutList

name (XML)

使用法

```
XMLnode.name
```

解説

指定された XML ノードの名前を返す XML プロパティです。

例文

次の XML から始まる例 :

```
<?xml version="1.0"?>
    <e1>
<tagName attr1="val1" attr2="val2"/>
<e2> element 2</e2>
<e3>element 3</e3>
    </e1>
```

次の Lingo は、タグ <e1> 内にネストされた第 2 のタグの名前を返します。

```
put gParserObject.child[1].child[2].name
-- "e2"
```

関連事項

attributeName

near (fog)

使用法

```
member(whichCastmember).camera(whichCamera).fog.near
cameraReference.fog.near
member(whichCastmember).camera(whichCamera).fog.far
cameraReference.fog.far
```

解説

3D プロパティ。fog.enabled が TRUE のとき、このプロパティを使用して、カメラの前面からフォグの開始ポイントまでの距離を取得または設定できます。

このプロパティのデフォルト値は 0.0 です。

例文

次のステートメントは、" デフォルトビュー " というカメラのフォグの near プロパティを 100 に設定します。

```
-- Lingo syntax
member("3dobjects").camera("defaultview").fog.near = 100.0

// JavaScript syntax
member("3dobjects").getPropRef("camera",1).fog.near = 100.00;
```

関連事項

fog, far (fog), enabled (fog), decayMode

nearFiltering

使用法

```
member(whichCastmember).texture(whichTexture).nearFiltering
member(whichCastmember).shader(whichShader).texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shader.texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].texture(whichTexture).nearFiltering
```

解説

3D テスクチャプロパティ。このプロパティを使用すると、オリジナルのテクスチャソースよりも大きなスクリーンスペースをカバーする投影テクスチャマップをレンダリングするときに、バイリニアフィルタリングを使用するかどうかを取得または設定できます。バイリニアフィルタリングを使用すると、テクスチャ全体のエラーを目立たなくすることでテクスチャの外観を良くすることができます。バイリニアフィルタリングでは、2 次元におけるエラーが目立たなくなります。トリリニアフィルタリングでは、3 次元におけるエラーが目立たなくなります。フィルタリングを使用すると、外観を良くすることができますが、その反面、パフォーマンスは低下します。バイリニアの方がトリリニアに比べ、パフォーマンスへの影響は小さくなります。

バイリニアフィルタリングを使用する場合は、プロパティの値を TRUE に設定します。バイリニアフィルタリングを使用しない場合は、値を FALSE に設定します。デフォルトは TRUE です。

例文

次のステートメントは、キャストメンバ "3D オブジェクト " 内の最初のテクスチャに対するバイリニアフィルタリングをオフにします。

```
-- Lingo syntax
member("3dobjects").texture[1].nearFiltering = FALSE

// JavaScript syntax
member("3dobjects").getPropRef("texture",1).nearFiltering = 0;
```

netPresent

使用法

```
-- Lingo syntax
_player.netPresent

// JavaScript syntax
_player.netPresent;
```

解説

インターネットへのアクセスに必要な Xtra 拡張が使用可能かどうかを示すプレーヤプロパティです。ただし、インターネット接続が現在アクティブかどうかは報告しません。読み取り専用。

ネットサポート Xtra 拡張が使用できない場合、netPresent は正しく機能しますが、netPresent() を使うとスクリプトエラーが発生します。

例文

次のステートメントは、ネットサポート Xtra 拡張が使用できないときにアラートを発生させます。

```
-- Lingo syntax
if (not(_player.netPresent)) then
    _player.alert("Sorry, the Network Support Xtras could not be found.")
end if

// JavaScript syntax
if (!(_player.netPresent)) {
    _player.alert("Sorry, the Network Support Xtras could not be found.");
}
```

関連事項

プレーヤ

netThrottleTicks

使用法

```
-- Lingo syntax
_player.netThrottleTicks

// JavaScript syntax
_player.netThrottleTicks;
```

解説

Macintosh のオーサリング環境でネットワーク操作に対するサービス頻度を制御するプレーヤプロパティです。読み取り / 書き込み可能。

デフォルトは 15 です。大きい値を設定すると、再生やアニメーションはスムーズになりますが、ネットワーク作業に費やす時間は短くなります。小さい値を設定すると、ネットワーク操作に費やす時間は増えますが、逆に再生やアニメーションのパフォーマンスに影響します。

このプロパティは、オーサリング環境と Macintosh のプロジェクタに影響します。Windows や Macintosh の Shockwave Player では無視されます。

例文

次のステートメントはプレーヤの netthrottleticks の値を示します。

```
-- Lingo syntax
put _player.netThrottleTicks

// JavaScript syntax
    put( _player.netThrottleTicks);
```

関連事項

プレーヤ

node

使用法

```
-- Lingo syntax
spriteObjRef.node

// JavaScript syntax
spriteObjRef.node;
```

解説

スプライトが表示する現行のノード ID を示す QuickTime VR スプライトプロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントはスプライトによって表示される現在のノード ID を示します。

```
-- Lingo syntax
put sprite(3).node

// JavaScript syntax
put( sprite(3).node);
```

nodeEnterCallback

使用法

```
-- Lingo syntax
spriteObjRef.nodeEnterCallback

// JavaScript syntax
spriteObjRef.nodeEnterCallback;
```

解説

QuickTime VR ムービーがステージ上で新しいアクティブなノードに切り替わった後に実行するハンドラの名前を含む QuickTime VR スプライトプロパティです。このメッセージのパラメータは、me パラメータと、これから表示されるノードの ID です。

QuickTime VR スプライトが、このメッセージを最初に受け取ります。

コールバックを解除するには、このプロパティに "0" を設定します。

処理性能を考慮し、コールバックプロパティは必要な場合にのみ設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは QuickTime VR ムービーがステージ上で新しいアクティブなノードに切り替わった後に実行するハンドラの名前を表示します。

```
-- Lingo syntax
put sprite(3).nodeEnterCallback

// JavaScript syntax
put( sprite(3).nodeEnterCallback);
```

nodeExitCallback

使用法

```
-- Lingo syntax
spriteObjRef.nodeExitCallback

// JavaScript syntax
spriteObjRef.nodeExitCallback;
```

解説

QuickTime VR ムービーがステージ上で新しいアクティブなノードに切り替わるときに実行するハンドラの名前を含む QuickTime VR スプライトプロパティです。このメッセージのパラメータは、me パラメータ、ムービーの切り替え前のノード ID、ムービーの切り替え後のノード ID です。

ハンドラが返す値によって、ムービーが次のノードに移るかどうかが決まります。ハンドラが #continue を返した場合、QuickTime VR スプライトは通常のノードトランジションで継続します。ハンドラが #cancel を返した場合、トランジションは実行されずに、ムービーは元のノードに留まります。

コールバックを解消するには、このプロパティを 0 に設定します。

QuickTime VR スプライトが、このメッセージを最初に受け取ります。

処理性能を考慮し、コールバックプロパティは必要な場合にのみ設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは QuickTime VR ムービーがステージ上で新しいアクティブなノードに切り替わる際に実行するハンドラの名前を表示します。

```
-- Lingo syntax
put sprite(3).nodeExitCallback

// JavaScript syntax
put( sprite(3).nodeExitCallback);
```

nodeType

使用法

```
-- Lingo syntax
spriteObjRef.nodeType

// JavaScript syntax
spriteObjRef.nodeType;
```

解説

指定したスプライトについて、ステージ上に現在あるノードの種類を示す QuickTime VR スプライトプロパティです。有効な値は、#object、#panorama、#unknown です。#unknown は、QuickTime VR スプライト以外のスプライトの値です。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、指定したスプライトについて、ステージ上に現在あるノードの種類を示します。

```
-- Lingo syntax
put sprite(3).nodeType

// JavaScript syntax
put( sprite(3).nodeType);
```

normalList

使用法

```
member(whichCastmember).modelResource(whichModelResource).normalList
model.meshDeform.mesh[index].normalList
```

解説

3D プロパティ。#mesh タイプのモデルリソースと使用し、そのモデルリソースオブジェクトの normalList プロパティを取得または設定します。

normalList プロパティは、ベクトルのリニアリストです。メッシュのフェイスを構築するときに、このリストから頂点の法線を指定できます。

このプロパティには、newMesh() 呼び出しで指定されたベクトル数とまったく同じ数のリストを設定する必要があります。

また、normalList プロパティは、メッシュモデルリソースの generateNormals() メソッドで生成することもできます。

meshDeform モディファイアと共に使用するときは、normalList プロパティもベクトルのリニアリストになります。メッシュの変形時には、ここから頂点の法線を指定できます。

フェイス法線と頂点法線の詳細については、「normals」を参照してください。

例文

次のステートメントは "3D オブジェクト " というキャストメンバの " ピラミッド " というモデルリソースの normalList プロパティを示します。

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").normalList[2]

// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).normalList[2]);
```

関連事項

face, meshDeform （モディファイア）

normals

使用法

```
member(whichCastmember).modelResource(whichModelResource).face[index].normals
```

解説

3D フェイスプロパティ。#mesh タイプのモデルリソースの場合 (newMesh コマンドで作成)、このプロパティを使用すると、インデックスパラメータで指定されたフェイス用の法線ベクトルのリストを取得および設定できます。

このプロパティには、整数のリニアリストを設定します。このリニアリストの整数は、モデルリソースの normalList プロパティに含まれる各頂点の法線のインデックス位置に対応したものです。

このプロパティは、face[index].verticies リストと同じ長さに設定する必要がありますが、空のリスト [ ] にすることもできます。

generateNormals() コマンドを使って法線ベクトルを生成する場合は、このプロパティには値を設定しないでください。

このプロパティに変更を加えたり、generateNormals() コマンドを使用する場合は、メッシュを再構築するために build() コマンドを呼び出す必要があります。

例文

次のステートメントは "3D オブジェクト " というキャストメンバの " ピラミッド " というモデルリソースの 5 番目のフェイスの normals プロパティを示します。

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").face[5].normals

// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).face[5].normals);
```

関連事項

face, normalList, vertices

number （キャスト）

使用法

```
-- Lingo syntax
castObjRef.number

// JavaScript syntax
castObjRef.number;
```

解説

指定したキャストライブラリの番号を返すキャストライブラリプロパティです。読み取り専用。

例文

次の繰り返しループは、ムービーの各キャスト内のキャストメンバの数をメッセージウィンドウに表示します。

```
-- Lingo syntax
repeat with n = 1 to _movie.castLib.count
    put(castLib(n).name && "contains" && castLib(n).member.count && "cast members.")
end repeat

// JavaScript syntax
for (var n=1; n<=_movie.castLib.count; n++) {
    put(castLib(n).name + " contains " + castLib(n).member.count + " cast members.")
}
```

関連事項

キャストライブラリ

number （文字）

使用法

```
the number of chars in chunkExpression
```

解説

チャンク式内の文字数を返すチャンク関数です。

チャンク式には、文字が格納されているコンテナ内のすべての文字 (空白とタブ、改行などの制御文字を含む)、単語、項目、行を指定できます。フィールドキャストメンバ、ストリングを格納した変数、特定の文字、単語、項目、行、コンテナ内の範囲などは、すべてコンテナです。

注意：count() 関数は、チャンク式の文字数をより効率的に返します。

例文

次のステートメントは、「Adobe, the Multimedia Company」というストリングの文字数をメッセージウィンドウに表示します。

```
put the number of chars in "Adobe, the Multimedia Company"
```

この結果は 29 になります。

次のステートメントは、" 名前 " キャストメンバの I 個目の単語の文字数を変数 charCounter に設定します。

```
charCounter = the number of chars in member("Names").word[i]
```

次のドットシンタックスでも、同じことを実行できます。

```
charCounter = member("Names").word[i].char.count
```

関連事項

length(), char...of, count(), number （項目）, number （行）, number （単語）

number （項目）

使用法

```
the number of items in chunkExpression
```

解説

チャンク式内の項目の数を返すチャンク関数です。カンマで区切られた一連の文字それぞれが項目にあたります。

文字コンテナ内の任意の文字、単語、項目、行をチャンク式として指定できます。フィールド (フィールドキャストメンバ)、ストリングを含む変数、特定の文字、単語、項目、行、コンテナ内の特定範囲などはすべて文字のコンテナです。

注意：別の方法として、count() 関数を使用すると、チャンク式内の項目の数をより効率的に返すことができます。

例文

次のステートメントは、「Adobe, the Multimedia Company」というストリングの項目数をメッセージウィンドウに表示します。

```
put the number of items in "Adobe, the Multimedia Company"
```

この結果は 2 になります。

次のステートメントは、" 名前 " フィールド内の項目数を変数 itemCounter に設定します。

```
itemCounter = the number of items in member("Names").text
```

次のドットシンタックスでも、同じことを実行できます。

```
itemCounter = member("Names").item.count
```

関連事項

item...of, count(), number （文字）, number （行）, number （単語）

number （行）

使用法

```
the number of lines in chunkExpression
```

解説

チャンク式内の行数を返すチャンク関数です。行末で折り返し処理された行ではなく、改行で区切られた行を指します。

文字コンテナ内の任意の文字、単語、項目、行をチャンク式として指定できます。フィールドキャストメンバ、ストリングを格納した変数、特定の文字、単語、項目、行、コンテナ内の範囲などは、すべてコンテナです。

注意：別の方法として、count() 関数を使用すると、チャンク式の行数をより効率的に返すことができます。

例文

次のステートメントは、「Adobe, the Multimedia Company」というストリングの行数をメッセージウィンドウに表示します。

```
put the number of lines in "Adobe, the Multimedia Company"
```

この結果は 1 になります。

次のステートメントは " 名前 " フィールド内の行数を変数 lineCounter に設定します。

```
lineCounter = the number of lines in member("Names").text
```

テキストキャストメンバに次のシンタックスを使って同じことができます。

```
lineCounter = member("Names").line.count
```

関連事項

line...of, count(), number （文字）, number （項目）, number （単語）

number （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.number

// JavaScript syntax
memberObjRef.number;
```

解説

指定されたキャストメンバのキャストライブラリの番号を示すメンバプロパティです。読み取り専用。

このプロパティの値は、キャストライブラリ内の位置を 1 つの整数値で表すキャストメンバに固有の識別子です。

例文

次のステートメントは、" 電源スイッチ " キャストメンバのキャストメンバ番号を変数 **whichCastMember** に割り当てます。

```
-- Lingo syntax
whichCastMember = member("Power Switch").number

// JavaScript syntax
var whichCastMember = member("Power Switch").number;
```

次のステートメントは、" 赤い風船 " キャストメンバをスプライト 1 のメンバに指定します。

```
-- Lingo syntax
sprite(1).member = member("Red Balloon").number

// JavaScript syntax
sprite(1).member = member("Red Balloon").number;
```

次の例では、スプライトのキャストメンバを切り替える前に、キャストメンバが実際に存在するかを検証します。

```
-- Lingo syntax
property spriteNum

on mouseUp me
    if (member("Mike's face").number > 0) then
        sprite(spriteNum).member = "Mike's face"
    end if
end

// JavaScript syntax
function mouseUp() {
    if (member("Mike's face").number > 0) {
        sprite(this.spriteNum).member = "Mike's face"
    }
}
```

関連事項

castLib(), メンバ

number （メニュー）

使用法

```
the number of menus
```

解説

現行ムービーにインストールされているメニューの数を示すメニュープロパティです。

このメニュープロパティは値を調べられますが、変更はできません。カスタムメニューバーを設定するには、installMenu コマンドを使用します。

注意：メニューは、Shockwave Player では使用できません。

例文

次のステートメントは、ムービーにカスタムメニューがインストールされているかどうかを調べ、メニューがインストールされていない場合は " メニューバー " メニューをインストールします。

```
if the number of menus = 0 then installMenu "Menubar"
```

次のステートメントは現行ムービー内のメニューの数をメッセージウィンドウに表示します。

```
put the number of menus
```

関連事項

installMenu, number （メニュー項目）

number （メニュー項目）

使用法

```
the number of menuItems of menu whichMenu
```

解説

[メニュー] で指定したカスタムメニュー内のメニュー項目の数を確認するためのメニュープロパティです。[メニュー] パラメータにはメニュー名またはメニュー番号を指定します。

このメニュープロパティは値を調べられますが、変更はできません。カスタムメニューバーを設定するには、installMenu コマンドを使用します。

注意：メニューは、Shockwave Player では使用できません。

例文

次のステートメントは、カスタムのファイルメニュー内のメニュー項目の数を変数 fileItems に設定します。

```
fileItems = the number of menuItems of menu "File"
```

次のステートメントは、メニュー番号が変数 i と同じになるカスタムメニュー内のメニュー項目の数を変数 itemCount に設定します。

```
itemCount = the number of menuItems of menu i
```

関連事項

installMenu, number （メニュー）

number （スプライトチャネル）

使用法

```
-- Lingo syntax
spriteChannelObjRef.number

// JavaScript syntax
spriteChannelObjRef.number;
```

解説

スプライトチャネルの番号を返すスプライトチャネルプロパティです。読み取り専用。

例文

次のステートメントは、指定した名前のスプライトチャネルの番号をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(channel("Kite String").number)

// JavaScript syntax
put(channel("Kite String").number);
```

関連事項

スプライト

number （システム）

使用法

```
the number of castLibs
```

解説

現行ムービー内のキャストの数を返すシステムプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次の繰り返しループは、ムービーの各キャスト内のキャストメンバの数をメッセージウィンドウに表示します。

```
repeat with n = 1 to the number of castLibs
    put castLib(n).name && "contains" && the number of members of castLib(n) && "cast members."
end repeat
```

number （単語）

使用法

```
the number of words in chunkExpression
```

解説

[チャンク式] で指定したチャンク式内の単語数を返す関数です。

文字コンテナ内の任意の文字、単語、項目、行をチャンク式として指定できます。フィールドキャストメンバ、ストリングを格納した変数、特定の文字、単語、項目、行、コンテナ内の範囲などは、すべてコンテナです。

テキストキャストメンバで単語数を数えるには、count を参照してください。

注意：別の方法として、count() 関数を使用すると、チャンク式の単語数をより効率的に返すことができます。

例文

次のステートメントは、「Adobe, the multimedia company」というストリング内の単語数をメッセージウィンドウに表示します。

```
put the number of words in "Adobe, the multimedia company"
```

この結果は 4 になります。

次のハンドラは、wordList パラメータで指定したストリング単語の並び順を逆にします。

```
on reverse wordList
    theList = EMPTY
    repeat with i = 1 to the number of words in wordList
        put word i of wordList & " " before theList
    end repeat
    delete theList.char[thelist.char.count]
    return theList
end
```

関連事項

count(), number （文字）, number （項目）, number （行）, word...of

number of members

使用法

```
the number of members of castLib whichCast
```

解説

指定したキャスト内で最後のキャストメンバの番号を示すキャストメンバプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" 中心キャスト " 内の各キャストメンバのタイプをメッセージウィンドウに表示します。castLib プロパティのメンバの数を使用して、ループを何回繰り返すのかを決めています。

```
repeat with i = 1 to the number of members of castLib("Central Casting")
    put "Cast member" && i && "is a" && member(i, "Central Casting").type
end repeat
```

number of xtras

使用法

```
the number of xtras
```

解説

ムービーで使用できるスクリプティング Xtra 拡張の数を返すシステムプロパティです。Xtra 拡張の数には openxlib コマンドで開いたものや、Configuration\Xtras フォルダ内に入っているものが含まれます。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、ムービーで使用できるスクリプティング Xtra 拡張の数をメッセージウィンドウに表示します。

```
put the number of xtras
```

numBuffersToPreload

使用法

```
mixer.numBuffersToPreload
```

解説

play 呼び出しが返す前に計算されたバッファ数を定義するオーディオプロパティです。デフォルトでは、このプロパティは 0 に設定されます。このプロパティは、ミクサが #stopped 状態のときにのみ設定できます。

例文

```
--Lingo syntax
on mouseUp me
    mixerRef.numBuffersToPreload = 2
end

// JavaScript syntax
function mouseUp(){
mixerRef.numBuffersToPreload = 2;
}
```

関連事項

ミクサ

numChannels

使用法

```
-- Lingo syntax
memberObjRef.numChannels

// JavaScript syntax
memberObjRef.numChannels;
```

解説

指定した SWA ストリーミングキャストメンバ内のチャネルの数を返す Shockwave Audio (SWA) キャストメンバプロパティです。返される値の 1 はモノラルを 2 はステレオを表します。

このプロパティは、SWA ストリーミングキャストメンバが再生を開始した後、または preLoadBuffer コマンドを使ってファイルがプリロードされた後にのみ使用できます。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は " デューク・エリントン "SWA ストリーミングキャストメンバのサウンドチャネルがステレオかモノラルかを " チャネル表示 " フィールドキャストメンバに表示します。

```
-- Lingo syntax
myVariable = member("Duke Ellington").numChannels
if myVariable = 1 then
    member("Channel Display").text = "Mono"
else
    member("Channel Display").text = "Stereo"
end if

// JavaScript syntax
var myVariable = member("Duke Ellington").numChannels;
if (myVariable == 1) {
    member("Channel Display").text = "Mono";
} else {
    member("Channel Display").text = "Stereo";
}
```

numParticles

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
modelResourceObjectReference.emitter.numParticles
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのリソースのパーティクルエミッタの numParticles プロパティを取得または設定できます。この値は、0 より大きく 100000 以下である必要があります。デフォルト設定は 1000 です。

例文

次のステートメントは、"3D オブジェクト " というキャストメンバで内のパーティクル数を 50000 に設定します。

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.numParticles = 50000
// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.numParticles = 50000;
```

関連事項

emitter

numSegments

使用法

```
member(whichCastmember).modelResource(whichModelResource).numSegments
```

解説

3D プロパティ。#cylinder というタイプのモデルリソースを指定すると、そのモデルリソースの numSegments プロパティを取得または設定できます。

numSegments プロパティでは、円柱の上面から底面までのセグメント数を指定できます。このプロパティは、デフォルト値の 2 以上にする必要があります。

円柱のサーフェスの滑らかさは、このプロパティに指定された値によって変わります。プロパティの値が大きくなるほど、円柱のサーフェスはより滑らかに見えます。

例文

次のステートメントは、" 円柱 01" というモデルリソースの numSegments プロパティを 10 に設定します。

```
-- Lingo syntax
member("3Dobjects").modelResource("Cylinder01").numSegments = 10

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",11).numSegments = 10;
```

obeyScoreRotation

使用法

```
member(flashMember).obeyScoreRotation
```

解説

Flash ムービースプライトがスコアからの回転情報を使用するか、それとも Flash アセットの以前の回転プロパティを使用するかを指定する Flash キャストメンバプロパティです。前者の場合は TRUE を、後者の場合は FALSE を設定します。

バージョン 7 より前の Director で作成されたムービーでは、このプロパティは自動的に FALSE に設定されます。これにより、この Flash キャストメンバを含むすべてのスプライトのメンバ回転プロパティを使用する古いバージョンの機能を利用できます。

バージョン 7 以降で作成された新しいアセットでは、このプロパティは自動的に TRUE に設定されます。

このプロパティが TRUE の場合は、メンバの回転プロパティは無視され、代わりにスコア回転設定に従います。

例文

次のスクリプトは、キャストメンバ "FlashObj" の obeyScoreRotation プロパティを、1（TRUE）に設定し、そのキャストメンバを含むスプライトを 180° 回転させます。

```
-- Lingo syntax
member("FlashObj").obeyScoreRotation = 1
sprite(1).rotation = sprite(1).rotation + 180

// JavaScript syntax
member("FlashObj").obeyScoreRotation = 1;
sprite(1).rotation = sprite(1).rotation + 180;
```

関連事項

rotation

optionDown

使用法

```
-- Lingo syntax
_key.optionDown

// JavaScript syntax
_key.optionDown;
```

解説

ユーザが Alt キー（Windows）または Option キー（Macintosh）を押しているかどうかを調べるキープロパティです。読み取り専用。

Alt キーまたは Option キーが押されているときは、TRUE が返されます。それ以外の場合は、FALSE が返 されます。

Windows の場合、修飾キー以外のキーを押さずに Alt キーを押したときは optionDown はプロジェクタでは動作しません。Windows プロジェクタとしてムービーを配布するときや、修飾キーのみが押されていることを検出するときは、optionDown は使わず、controlDown や shiftDown を使います。

Macintosh では Option キーを押すと key の値が変化します。このため、代わりに keyCode を使用します。

例文

次のハンドラは、ユーザが Alt キーまたは Option キーを押しているかどうかをチェックし、押している場合は doOptionKey というハンドラを実行します。

```
-- Lingo syntax
on keyDown
    if (_key.optionDown) then
        doOptionKey(_key.key)
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.optionDown) {
        doOptionKey(_key.key);
    }
}
```

関連事項

controlDown, キー , key, keyCode, shiftDown

organizationName

使用法

```
-- Lingo syntax
_player.organizationName

// JavaScript syntax
_player.organizationName;
```

解説

Director のインストールの際に入力された会社名を格納するプレーヤプロパティです。読み取り専用。

このプロパティはオーサリング環境でのみ使用できます。ユーザ情報を表示するようにカスタマイズできる Movie In A Window（MIAW）ツールのムービーで使用できます。

例文

次のハンドラは、ウィンドウ内のムービー (MIAW) のムービースクリプトに記述します。このハンドラは、ウィンドウが開いたときに表示されるフィールドにユーザ名とシリアル番号を表示します。

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN & _player.serialNumber
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" + _player.serialNumber;
    member("User Info").text = displayString;
}
```

関連事項

プレーヤ

originalFont

使用法

```
-- Lingo syntax
memberObjRef.originalFont

// JavaScript syntax
memberObjRef.originalFont;
```

解説

フォントキャストメンバプロパティ。指定したキャストメンバが作成されるときに、読み込まれたオリジナルフォント名を正確に返します。

例文

次のステートメントは、キャストメンバ 11 が作成されるときに、読み込まれたフォント名を表示します。

```
-- Lingo syntax
put(member(11).originalFont)

// JavaScript syntax
put(member(11).originalFont);
```

関連事項

recordFont, bitmapSizes, characterSet

originH

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.originH

// JavaScript syntax
memberOrSpriteObjRef.originH;
```

解説

Flash ムービーとベクトルシェイプの原点の水平方向の座標をピクセル単位で制御するキャストメンバおよびスプライトプロパティです。値には浮動小数を使用できます。

原点は、Flash ムービーやベクトルシェイプでの拡大縮小と回転の基準となる座標です。原点は、originH プロパティと originV プロパティを個別に使用して浮動小数の精度で設定できます。また、原点は、originPoint プロパティを使用して整数の精度で設定することもできます。

originPoint プロパティを使用した場合は、整数の精度で設定できます。originH プロパティは、originMode プロパティが #point に設定されている場合にのみ設定できます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 0 です。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のスプライトスクリプトは、originMode プロパティを使って Flash ムービースプライトを設定し、Flash ムービースプライトの原点を指定した点に設定できるようにします。次に、水平方向と垂直方向の原点を設定します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

関連事項
originV, originMode, originPoint, scaleMode

originMode

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.originMode

// JavaScript syntax
memberOrSpriteObjRef.originMode;
```

解説
拡大縮小や回転の基準となる原点を次のように設定するキャストメンバプロパティとスプライトプロパティです。

- #center (デフォルト) - 原点は Flash ムービーの中央になります。
- #topleft - 原点は Flash ムービーの左上隅になります。
- #point - 原点は originPoint、originH、および originV の各プロパティで指定した点となります。

このプロパティは、値を調べたり、設定することができます。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文
次のスプライトスクリプトは、originMode プロパティを使って Flash ムービースプライトを設定し、Flash ムービースプライトの原点を指定した点に設定できるようにします。次に、水平方向と垂直方向の原点を設定します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

関連事項
originH, originV, originPoint, scaleMode

originPoint

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.originPoint

// JavaScript syntax
memberOrSpriteObjRef.originPoint;
```

解説

Flash ムービーやベクトルシェイプでの拡大縮小と回転の基準となる原点を制御するキャストメンバやスプライトプロパティです。

originPoint プロパティは Director の点を示す値として指定します。例えば、point(100,200) と指定します。originPoint プロパティを使って Flash ムービーやベクトルシェイプの原点を設定するのは、originH と originV プロパティを個別に設定することと同じです。例えば、originPoint プロパティを (50,75) に設定するのは、originH プロパティを 50 に originV を 75 に設定するのと同じです。

originPoint プロパティには Director の点の値として整数値のみを指定しますが、originH と originV には浮動小数値を指定できます。originPoint プロパティを調べると、原点の値が整数値になっていることがわかります。一般に、精度が要求されるときは originH プロパティと originV プロパティを使用し、早く簡単に作成するときは originPoint プロパティを使用します。

originPoint プロパティは、originMode プロパティが #point に設定されている場合にのみ設定できます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 0 です。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のスプライトスクリプトは、originMode プロパティを使って Flash ムービースプライトを設定し、Flash ムービースプライトの原点を指定した点に設定できるようにします。次に原点を設定します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originPoint = point(100, 80)
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originPoint = point(100, 80);
}
```

関連事項

originH, originV, scaleMode

originV

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.originv

// JavaScript syntax
memberOrSpriteObjRef.originV;
```

解説

Flash ムービーとベクトルシェイプでの拡大縮小や回転の基準となる原点の垂直方向の座標を、ピクセルで制御するキャストメンバとスプライトプロパティです。値には浮動小数を使用できます。

原点は、originH プロパティと originV プロパティを個別に使用して浮動小数の精度で設定できます。また、原点は、originPoint プロパティを使用して整数の精度で設定することもできます。

originPoint プロパティを使用した場合は、整数の精度で設定できます。originV プロパティは、originMode プロパティが #point に設定されている場合にのみ設定できます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 0 です。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のスプライトスクリプトは、originMode プロパティを使って Flash ムービースプライトを設定し、Flash ムービースプライトの原点を指定した点に設定できるようにします。次に、水平方向と垂直方向の原点を設定します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

関連事項

originH, originPoint, scaleMode

orthoHeight

使用法

```
member(whichCastmember).camera(whichCamera).orthoHeight
member(whichCastmember).camera[cameraindex].orthoHeight
sprite(whichSprite).camera.orthoHeight
```

解説

3D プロパティ。camera.projection が #orthographic に設定されているとき、camera.orthoHeight の値は、スプライトに垂直にフィットする垂直ワールド単位の数を示します。ワールド単位は、特定の 3D ワールドの計測単位です。この単位は、内部的には一定ですが、任意に選択されるので、3D ワールドごとに異なる場合があります。

スプライトの最初のカメラにアクセスする場合は、カメラのインデックス（[カメラ]）を指定する必要はありません。

このプロパティのデフォルト値は 200.0 です。

例文

次のステートメントは、スプライト 1 のカメラの orthoHeight を 200 に設定します。この場合、スプライトに垂直にフィットするワールド単位の数は 200 になります。

```
-- Lingo syntax
sprite(1).camera.orthoheight = 200.0

// JavaScript syntax
sprite(1).camera.orthoheight = 200.0;
```

関連事項

projection

overlay

使用法

```
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].propertyName
member(whichCastmember).camera(whichCamera).overlay.count
```

解説

3D カメラプロパティ。このプロパティを使用すると、カメラの表示するオーバーレイのリストに含まれているオーバーレイのプロパティへのアクセスを取得および設定できます。overlay.count として使用すると、カメラの表示するオーバーレイのリストに含まれているオーバーレイの総数が返されます。

オーバーレイとは、カメラの表示に表示されるすべてのモデルの前面に表示されるテクスチャです。オーバーレイは、カメラのオーバーレイリストに表示されている順に描画されます。したがって、リストの最初のオーバーレイが一番後ろに表示され、最後のオーバーレイが一番手前に表示されます。

カメラのオーバーレイリスト内の各オーバーレイには、次のようなプロパティがあります。

- loc。カメラの矩形の左上角を基準にしたオーバーレイの regPoint の明確な位置を取得または設定できます。
- source。オーバーレイのソース画像として使用するテクスチャを取得または設定できます。
- scale。オーバーレイが使用するスケール値を取得または設定できます。scale プロパティは、オーバーレイの倍率を決定します。このプロパティのデフォルト値は 1.0 です。
- rotation。オーバーレイの回転 (単位は度) を取得または設定できます。
- regPoint。テクスチャの左上角を基準にしたオーバーレイのセンターポイントを取得または設定できます。
- blend。0 ～ 100 の整数で表されるオーバーレイのブレンドを取得または設定できます。この値は、オーバーレイの透明 (0) または不透明 (100) の度合いを表します。

例文

次のステートメントは、スプライトのカメラのオーバーレイのうち最初のオーバーレイの scale プロパティを表示します。

```
-- Lingo syntax
put sprite(2).camera.overlay[1].scale
```

関連事項

addOverlay, removeOverlay

pageHeight

使用法

```
-- Lingo syntax
memberObjRef.pageHeight

// JavaScript syntax
memberObjRef.pageHeight;
```

解説

ステージ上に表示されている部分のフィールドキャストメンバの高さをピクセル数で返すフィールドキャストメンバプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、フィールドキャストメンバ " 今日のニュース " の表示されている部分の高さをメッセージウィンドウに表示します。

```
--Lingo syntax
trace(member("Today's News").pageHeight)

// JavaScript syntax
trace(member("Today's News").pageHeight);
```

palette

使用法

```
-- Lingo syntax
memberObjRef.palette

// JavaScript syntax
memberObjRef.palette;
```

解説

[キャストメンバ] で指定したキャストメンバと関連づけられているパレットを特定するためのキャストメンバプロパティで、ビットマップキャストメンバのみに適用されます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、" 葉 " キャストメンバに割り当てられているパレットをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Leaves").palette)

// JavaScript syntax
put(member("Leaves").palette);
```

paletteMapping

使用法

```
-- Lingo syntax
_movie.paletteMapping

// JavaScript syntax
_movie.paletteMapping;
```

解説

キャストメンバのパレットが現行ムービーのパレットと異なる場合、ムービーがキャストメンバのパレットをリマップするかどうかを決めるムービープロパティです。TRUE の場合はリマップし、FALSE, の場合はリマップしません。デフォルトは FALSE です。読み取り / 書き込み可能。

このプロパティはムービープロパティダイアログボックスにある「必要時パレットをリマップする」チェックボックスと同じような働きをします。

パレットが異なるビットマップを同時に表示するには、paletteMapping を TRUE に設定します。このとき、Director は画面上の各キャストメンバの参照パレット（キャストメンバプロパティダイアログボックスで割り当てられているパレット）を確認し、これが現行パレットと異なる場合は、新規パレット内の各ピクセルに最も近いものを探します。

一致しないビットマップカラーは、オリジナルカラーに比較的近いものになります。

リマップには処理時間を要するので、通常は、前もってビットマップのパレットを調整しておくとよいでしょう。

この機能のために、望ましくない結果になることがあります。スプライトの範囲の途中でパレットが変更されると、直ちにビットマップが新規パレットにリマップされて間違ったカラーで表示されます。ただし、トランジションやステージを横切るスプライトなどで画面がリフレッシュされた場合は、画面上で影響を受けた矩形はリマップされたカラーで表示されます。

例文

次のステートメントは、必要なときにムービーのパレットをリマップします。

```
-- Lingo syntax
_movie.paletteMapping = TRUE

// JavaScript syntax
_movie.paletteMapping = true;
```

関連事項

ムービー

paletteRef

使用法

```
member(whichCastMember). paletteRef
the paletteRef
```

解説

ビットマップキャストメンバに関連づけられているパレットを決めるためのビットマップキャストメンバプロパティです。Director 内蔵パレットはシンボル (#systemMac、#rainbow など) によって示され、キャストメンバになっているパレットはキャストメンバとして参照されます。キャストパレットを正の数値で、Director 内蔵パレットを負の数値で表す palette メンバプロパティとは、この点で異なります。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、ビットマップキャストメンバ " 貝殻 " に Macintosh のシステムパレットを割り当てます。

```
member("Shell").paletteRef = #systemMac
```

pan

使用法

```
-- Lingo syntax
soundChannelObjRef.pan

// JavaScript syntax
soundChannelObjRef.pan;
```

解説

サウンドチャネルで再生中のサウンドの左右バランスを示すサウンドチャネルプロパティです。読み取り / 書き込み可能。

値の範囲は -100 から 100 までです。-100 は左のチャネルのみが聞こえることを示します。100 は右のサウンドのみが聞こえることを示します。値 0 は左右のバランスが釣り合っていることを示し、音源が中央にあるように聞こえます。モノラルサウンドの場合は、pan は、どちらのスピーカー (左か右か) からサウンドが再生されているかによって影響を受けます。

サウンドオブジェクトのパンはいつでも変更できますが、サウンドチャネルがフェードを実行中の場合は、フェードが終了するまではパン設定が反映されません。

完成したムービーでの pan の使用例については、Director アプリケーションフォルダ内の Learning\\Lingo_Examples フォルダにある sound control.dir を参照してください。

例文

次のステートメントは、チャネル 2 のサウンドを左のチャネルから右のチャネルへパンします。

```
-- Lingo syntax
repeat with x = -100 to 100
    sound(2).pan = x
end repeat

// JavaScript syntax
for (var x = -100; x <= 100; x++) {
    sound(2).pan = x;
}
```

関連事項

サウンドチャネル

pan （QTVR プロパティ）

使用法

```
-- Lingo syntax
spriteObjRef.pan

// JavaScript syntax
spriteObjRef.pan;
```

解説

QuickTime VR ムービーの現行パンを示す QuickTime VR スプライトプロパティです。値は度で示されます。

このプロパティは、値を調べたり、設定することができます。

panMatrix、toChannels、および useMatrix（ミクサ）

使用法

```
mixerRef.toChannels
mixerRef.useMatrix
mixerRef.panMatrix
```

解説

ミクサプロパティ。パンは、様々な方向のオーディオ信号の強度変化を参照します。既存のチャネル数だけでなく、追加のチャネル数についても（合計で最高 5.1 チャネル）、オーディオ信号の強度をパンできます。

例えば、オーディオ信号のチャネル数を 2 から 5.1 または 5.1 からモノに増やすことができます。

パンでサポートされているオーディオチャネルはモノ、ステレオ、2.1、3.1、4.1 および 5.1 です。

- 2.1 には、Left、Right および LFE（低周波数効果）チャネルがあります。
- 3.1 には、Left、Right、Center および LFE チャネルがあります。
- 4.1 には、Left、Right、Left Surround、Right Surround および LFE チャネルがあります。
- 5.1 には、Left、Right、Center、LFE、Left Surround および Right Surround チャネルがあります。

プロパティの概要

プロパティ	解説	範囲	デフォルト
パンマトリクス	パンの値を持つ 2 次元配列。panMatrix は n X m マトリクスで、n はパン前のオーディオデータのチャネル数、m はパン後のオーディオデータ内のチャネル数です。panMatrix[i][j] により、出力オーディオの j 番目チャネル（スピーカー）を生成する入力オーディオの i 番目チャネル（スピーカー）の寄与に関する情報が得られます。	0-5	ソースおよびターゲットのチャネルに応じて 0 または 1 のいずれかになります。
toChannels	パン後のチャネル数。	1-6	0（パンなし）。
ユースマトリクス	デフォルトでは、このフラグは消去されます。これが有効で、パンマトリクスを提供しない場合、デフォルトのマトリクスが使用されます。	True/False	False

例文

```
-- Lingo syntax
-- Panning  mixer output to 3.1 channels.
on mouseUp me
 gmixer = new(#mixer)   -- Creates a mixer.
 so = gmixer.createSoundObject("so1", member(4),[#loopcount:0])
 gmixer.toChannels = 4  -- Sets the channel property of the mixer to 4.
 gmixer.usematrix = true
 myPanMatrix = newMatrix(2,4)   -- Creates a 2x4 myPanmatrix. The stereo file is
-- panned to 3.1 channels.
  repeat with i = 1 to 2
  repeat with j = 1 to 4
   myPanMatrix.setval(i,j,1)  -- Updates all values of myPanmatrix to 1.
  end repeat
 end repeat
gmixer.panMatrix = myPanMatrix   -- Assigns myPanmatrix to the mixer's PanMatrix.

gmixer.play()
end
```

関連事項

ミクサ

panMatrix、toChannels、および useMatrix（サウンドオブジェクト）

使用法

```
soundObj.toChannels
soundObj.useMatrix
soundObj.panMatrix
```

解説

サウンドオブジェクトプロパティ。パンは、様々な方向のオーディオ信号の強度変化を参照します。既存のチャネル数だけでなく、追加のチャネル数についても（合計で最高 5.1 チャネル）、オーディオ信号の強度をパンできます。

例えば、オーディオ信号のチャネル数を 2 から 5.1 または 5.1 からモノに増やすことができます。

パンでサポートされているオーディオチャネルはモノ、ステレオ、2.1、3.1、4.1 および 5.1 です。

- 2.1 には、Left、Right および LFE（低周波数効果）チャネルがあります。
- 3.1 には、Left、Right、Center および LFE チャネルがあります。
- 4.1 には、Left、Right、Left Surround、Right Surround および LFE チャネルがあります。
- 5.1 には、Left、Right、Center、LFE、Left Surround および Right Surround チャネルがあります。

プロパティの概要

プロパティ	解説	範囲	デフォルト
パンマトリクス	パンの値を持つ 2 次元配列。panMatrix は n X m マトリクスで、n はパン前のオーディオデータのチャネル数、m はパン後のオーディオデータ内のチャネル数です。panMatrix[i][j] により、出力オーディオの j 番目チャネル（スピーカー）を生成する入力オーディオの i 番目チャネル（スピーカー）の寄与に関する情報が得られます。	0-5	ソースおよびターゲットのチャネルに応じて 0 または 1 のいずれかになります。
toChannels	パン後のチャネル数。	1-6	0（パンなし）。
ユースマトリクス	デフォルトでは、このフラグは消去されます。これが有効で、パンマトリクスを提供しない場合、デフォルトのマトリクスが使用されます。	True/False	False

注意：パンを適切に行うため、ミクサの channelCount はサウンドオブジェクトの tochannels 以上である必要があります。

例文

```
-- Lingo syntax
-- Panning to 3.1 channels.
on mouseUp me
 gmixer = new(#mixer)   -- Creates a mixer.
 gmixer.channelCount = 4 -- Sets the channelCount of the mixer to 4. This channelCount
-- value should be greater than or equal to the value of the sound object's
-- toChannels property.
 so = gmixer.createsoundobject("so1", member(4),[#loopcount:0])
 so.toChannels = 4  -- Sets the channel property of the sound object to 4.
 so.usematrix = true
 myPanMatrix = newMatrix(2,4)   -- Creates a 2x4 myPanmatrix. The stereo file is panned
-- to 3.1 channels.
  repeat with i = 1 to 2
   repeat with j = 1 to 4
    myPanMatrix.setval(i,j,1)  -- Updates all values of myPanmatrix to 1.
   end repeat
  end repeat
 so.panMatrix = myPanMatrix   -- Assigns myPanmatrix to the sound object's panMatrix.

gmixer.play()
end
```

paragraph

使用法

```
chunkExpression.paragraph[whichParagraph]
chunkExpression.paragraph[firstParagraph..lastParagraph]
```

解説

チャンク式によってテキストキャストメンバ内の異なる段落にアクセスできるテキストキャストメンバプロパティです。

段落とは、キャリッジリターンで区切られている部分のことです。

```
put member("AnimText").paragraph[3]
```

例文

次のステートメントはテキストキャストメンバ "myText" の第 2 段落を返します。

```
-- Lingo syntax
put member("myText").paragraph[2]
```

関連事項

line...of

parent

使用法

```
member(whichCastmember).model(whichModel).parent
member(whichCastmember).camera(whichCamera).parent
member(whichCastmember).light(whichLight).parent
member(whichCastmember).group(whichGroup).parent
```

解説

3D プロパティ。モデル、カメラ、ライト、またはグループの参照を指定して、参照先のオブジェクトの親ノードを取得または設定します。親ノードは、他のモデル、カメラ、ライト、またはグループのいずれかのオブジェクトです。

オブジェクトの transform プロパティは、親オブジェクトを基準にしたオブジェクトのスケール、位置、および向きを定義します。

オブジェクトの parent プロパティを VOID に設定すると、removeFromWorld() コマンドを使ってオブジェクトをワールドから削除するのと同じことになります。

オブジェクトの parent プロパティを World グループオブジェクト（group("World")）に設定すると、addToWorld() コマンドを使ってオブジェクトをワールドに追加するのと同じことになります。

addChild コマンドを使ってこのプロパティの値を変更することもできます。

例文

次のステートメントは、" タイヤ " というモデルの parent プロパティを設定します。" 車 " というモデルが " タイヤ " の親に設定されます。

```
-- Lingo syntax
member("3Dobjects").model("Tire").parent = member("3Dobjects").model("Car")

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).parent = member("3Dobjects").getPropRef("model",3);
```

関連事項

child (3D), addChild

password

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.password

// JavaScript syntax
memberOrSpriteObjRef.password;
```

解説

RealMedia スプライトおよびキャストメンバプロパティ。保護されている RealMedia ストリームへのアクセスに必要なパスワードを設定します。セキュリティ上の理由により、このプロパティを使用して指定した前のパスワードは、このプロパティでは取得できません。前にパスワードが設定されていた場合は、このプロパティの値は "********" というストリングになります。パスワードが設定されていない場合は、このプロパティの値は空のストリングになります。

例文

次の例は、キャストメンバ "Real" およびスプライト 2 の RealMedia ストリームにパスワードが設定されていることを示します。

```
-- Lingo syntax
put(sprite(2).password) -- "********"
put(member("Real").password) -- "********"

// JavaScript syntax
put(sprite(2).password); // "********"
put(member("Real").password); // "********"
```

次の例は、キャストメンバ "Real" またはスプライト "2" の RealMedia ストリームにパスワードが設定されていないことを示します。

```
-- Lingo syntax
put(sprite(2).password) -- ""
put(member("Real").password) -- ""

// JavaScript syntax
put(sprite(2).password); // ""
put(member("Real").password); // ""
```

次の例は、スプライト 2 およびキャストメンバ "Real" の RealMedia ストリームに "abracadabra" というパスワードを設定します。

```
-- Lingo syntax
sprite(2).password = "abracadabra"
member("Real").password = "abracadabra"

// JavaScript syntax
sprite(2).password = "abracadabra";
member("Real").password = "abracadabra";
```

関連事項

userName (RealMedia)

path （ムービー）

使用法

```
-- Lingo syntax
_movie.path

// JavaScript syntax
_movie.path;
```

解説

現行ムービーが保存されているフォルダのパス名を返すムービープロパティです。読み取り専用。

パス名を Windows と Macintosh の両方で使用するには、"@" パス名演算子を使います。

完成したムービーの path の実例は、Director アプリケーションフォルダの Learning/Lingo_Examples フォルダ内の read and write text.dir を参照してください。

例文

次のステートメントは、現行ムービーが保存されているフォルダのパス名を表示します。

```
-- Lingo syntax
trace(_movie.path)

// JavaScript syntax
trace(_movie.path);
```

次のステートメントは、現行ムービーのフォルダの Sounds サブフォルダに保存されているサウンドファイル Crash.aif を再生します。

```
-- Lingo syntax
sound(1).playFile(_movie.path & "Sounds\Crash.aif")

// JavaScript syntax
sound(1).playFile(_movie.path + "Sounds\\Crash.aif");
```

関連事項

ムービー

path (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.path
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのリソースのパーティクルエミッタの path プロパティを取得または設定できます。

このプロパティは、寿命が終わるまでパーティクルが従うパスを定義するベクトルのリストです。デフォルト値は空のリスト [] です。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" のパーティクルが従うパスをベクトルのリストで指定します。

```
member("Fires").modelResource("ThermoSystem").emitter.path = [vector(0,0,0), vector(15,0,0),
vector(30,30,-10)]
```

関連事項

pathStrength, emitter

pathName （Flash メンバ）

使用法

```
member(whichFlashMember).pathName
the pathName of member whichFlashMember
```

解説

Flash ムービーキャストメンバのアセットが格納されている外部ファイルの位置を制御するキャストメンバプロパティです。Flash ムービーをローカルドライブまたはネットワークドライブのパスや URL で指定できます。

リンクされていないキャストメンバのパスを設定すると、リンクされたキャストメンバに変換されます。

このプロパティは、値を調べたり、設定することができます。リンクされていないメンバの pathName プロパティには空のストリングが格納されます。

このプロパティは、ほかのメンバタイプに関しては fileName プロパティと同じで、fileName を pathName の代わりに使うことができます。

例文

次のステートメントは、キャストメンバの pathName プロパティを WWW 上の Flash ムービーの位置に変更します。

```
-- Lingo syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf"

// JavaScript syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf";
```

関連事項

fileName（メンバ）, linked

pathStrength

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
```

解説

3D プロパティ。#particle タイプモデルリソースを指定すると、エミッタの path プロパティで指定したパスにパーティクルが接近する度合いを設定できます。0.0 (パーティクルはパスに引き寄せられない) から無限大までの値を指定できます。デフォルト値は 0.1 です。pathStrength を 0.0 に設定すると、パスを一時的にオフにすることができます。

pathStrength に大きな値を設定するほど、パーティクルシステム全体の動きが滑らかではなくなります。一般に、pathStrength の値を大きくすると、パーティクルは高速ではね回ります。ただし、パーティクルの drag プロパティなど、抑制する力が適用されていると、この効果は低下します。

このプロパティは、値を調べたり、設定することができます。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の pathStrength プロパティを 0.97 に設定します。"ThermoSystem" の emitter.path プロパティでパスが指定されていると、パーティクルはそのパスに緊密に従います。

```
member("Fires").modelResource("ThermoSystem").emitter.pathStrength = 0.97
```

関連事項

path (3D), emitter

pattern

使用法

```
member(whichCastMember).pattern
the pattern of member whichCastMember
```

解説

指定したシェイプに関連づけられているパターンを決めるキャストメンバプロパティです。値はツールウィンドウのパターンパレットのサンプルに対応する数値になります。シェイプキャストメンバが塗りつぶされていない場合、パターンはキャストメンバの外側のエッジに適用されます。

Shockwave コンテンツを含むムービーでこのプロパティを使用すると、シェイプに適用されたタイルを変更して画像を変えることにより、より大きなビットマップが必要とするメモリを節約できます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、シェイプキャストメンバ "myShape" に塗りつぶしを適用し、1 番のパターン（単色塗りつぶし）を割り当てます。

```
-- Lingo syntax
member("myShape").filled = TRUE
member("myShape").pattern = 1

// JavaScript syntax
member("myShape").filled = 1;
member("myShape").pattern = 1;
```

pausedAtStart （Flash、デジタルビデオ）

使用法

```
member(whichFlashOrDigitalVideoMember).pausedAtStart
the pausedAtStart of member whichFlashOrDigitalVideoMember
```

解説

スプライトがステージに表示されたときにデジタルビデオや Flash ムービーを再生するかどうかを制御するキャストメンバプロパティです。TRUE の場合、デジタルビデオや Flash ムービーは、スプライトが表示されたときに自動で再生されません。FALSE の場合、スプライトが表示されるとすぐに再生されます。

デジタルビデオキャストメンバの場合、このプロパティはデジタルビデオキャストメンバプロパティダイアログボックス内の「一時停止」チェックボックスがチェックされているかどうかを示します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、" 回転椅子 " QuickTime ムービーに対して [デジタルビデオキャストメンバ情報] ダイアログボックスの [一時停止] チェックボックスをオンにします。

```
member("Rotating Chair").pausedAtStart = TRUE
```

pausedAtStart (MP4Media/FLV)

使用法

```
member(1).pausedAtStart = true
sprite(1).pausedAtStart = true
```

解説

MP4Media/FLV スプライトまたはキャストメンバのプロパティ。MP4Media/FLV の再生を自動的に開始するか、または一時停止したままにするかを指定できます。ムービーを自動的に再生するには、MP4Media/FLV の値を False または 0 に設定します。ムービーを一時停止するには、この値を True または 1 に設定します。

pausedAtStart には True または False として評価される式を設定できます。1 または 0 以外の整数値は、True として扱われます。このプロパティのデフォルト設定は False です。

pausedAtStart を False に設定した場合、ユーザは MP4Media/FLV ビューアの再生ボタン（またはムービー内に作成した同じ機能を持つボタン）をクリックする必要があります。または、play() メソッドを呼び出してステージ上のスプライトを再生することもできます。

このプロパティはスコアベースの再生だけに影響します。MP4Media/FLV ビューアでの再生には影響しません。

例文

次の例は、スプライト 2 およびキャストメンバ MP4Media/FLV の pausedAtStart プロパティを False に設定します。この場合、MP4Media/FLV ストリームは自動的に再生を開始します。

```
-- Lingo syntax
put(sprite(2).pausedAtStart) -- 0
put(member("MP4Media/FLV").pausedAtStart) -- 0

// JavaScript syntax
put(sprite(2).pausedAtStart); // 0
put(member("MP4Media/FLV").pausedAtStart); // 0
```

次の例は、スプライト 2 およびキャストメンバ MP4Media/FLV の pausedAtStart プロパティを True に設定します。この場合、MP4Media/FLV ストリームは play コマンドを呼び出さない限り再生されません。

```
-- Lingo syntax
sprite(2).pausedAtStart = TRUE
member("MP4Media/FLV").pausedAtStart = TRUE

// JavaScript syntax
sprite(2).pausedAtStart = 1;
member("MP4Media/FLV").pausedAtStart = 1;
```

pausedAtStart (RealMedia, Windows Media)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.pausedAtStart

// JavaScript syntax
memberOrSpriteObjRef.pausedAtStart;
```

解説

RealMedia および Windows Media スプライトまたはキャストメンバプロパティ。バッファリングか完了したときに、ステージ上の RealMedia または Windows Media ストリームを自動的に開始するか（FALSE または 0）、一時停止したままにするか（TRUE または 1）を取得または設定できます。読み取り / 書き込み可能。

このプロパティには、TRUE または FALSE として評価される式を設定できます。1 または 0 以外の整数値は、TRUE として扱われます。このプロパティのデフォルト設定は FALSE です。プロパティインスペクタのグラフィックビューで [一時停止] を選択すると、このプロパティは TRUE に設定されます。

このプロパティが FALSE に設定されている場合は、バッファリングが完了したときにスプライトの再生を開始するには、ユーザが RealMedia または Windows Media ビューアで再生ボタン（または、ムービー内に作成した同じ機能を持つボタン）をクリックするか、play() メソッドを呼び出す必要があります。

このプロパティはスコアベースの再生だけに影響し、RealMedia または Windows Media ビューアでの再生には影響しません。

例文

次の例は、スプライト 2 とキャストメンバ "Real" の pausedAtStart プロパティが FALSE に設定されていることを示します。この場合、バッファリングが完了すると、RealMedia ストリームの再生が自動的に開始されます。

```
-- Lingo syntax
put(sprite(2).pausedAtStart) -- 0
put(member("Real").pausedAtStart) -- 0

// JavaScript syntax
put(sprite(2).pausedAtStart); // 0
put(member("Real").pausedAtStart); // 0
```

次の例は、スプライト 2 とキャストメンバ "Real" の pausedAtStart プロパティを TRUE に設定します。この場合、play コマンドが呼び出されるまでは、RealMedia ストリームの再生は開始されません。

```
-- Lingo syntax
sprite(2).pausedAtStart = TRUE
member("Real").pausedAtStart = TRUE

// JavaScript syntax
sprite(2).pausedAtStart = 1;
member("Real").pausedAtStart = 1;
```

次の例は、pausedAtStart プロパティを使用して RealMedia スプライトをステージ外でバッファリングし、バッファリングが完了すると、そのスプライトをステージ上で再生します。この例では、RealMedia メンバの pausedAtStart プロパティは TRUE に設定されています。スプライトチャネル 1 で、このメンバのインスタンスはステージ外に位置しています。次のフレームスクリプトはスプライトスパンに置く必要があります。

```
-- Lingo syntax
on exitFrame me
    if sprite(1).state > 3 then -- check to see if buffering is complete
        sprite(1).locH = 162
        sprite(1).locV = 118
        sprite(1).play() -- position and play the sprite
    end if
end

// JavaScript syntax
function exitFrame() {
    var st = sprite(1).state;
    if (st > 3) { // check to see if buffering is complete
        sprite(1).locH = 162;
        sprite(1).locV = 118;
        sprite(1).play(); // position and play the sprite
    }
}
```

この RealMedia スプライトはステージ外でバッファリングされ、バッファリングが完了した直後にステージに表示され、再生されます。

percentBuffered

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.percentBuffered

// JavaScript syntax
memberOrSpriteObjRef.percentBuffered;
```

解説

RealMedia スプライトまたはキャストメンバプロパティ。ローカルファイルまたはサーバからロードしている RealMedia ストリームによってバッファが満たされたパーセンテージを返します。この値が 100 に達すると、バッファはフルの状態になり、pausedAtStart プロパティが TRUE に設定されていない場合は、RealMedia ストリームの再生が開始されます。このプロパティの値は、再生中に動的に変わります。値を設定することはできません。

バッファはメモリキャッシュの一種で、ムービー全体のうち再生直前の部分 (通常は数秒分) を蓄えておくものです。RealPlayer が受信するのに従ってストリームはバッファに入っていき、RealPlayer がクリップを再生するにつれてバッファから出ていきます。バッファがあることにより、ビューアがファイル全体をダウンロードしなくても再生できるようになります。また、ネットワークの混雑や使用可能な帯域幅の減少によってストリームの再生が中断されることも、バッファによって防止できます。

バッファリングプロセスは play コマンドによって開始されます。バッファがフルの状態 (100%) になると、バッファ内にあるストリームの再生が開始されます。最初のバッファリングプロセスには数秒かかるため、play コマンドを呼び出してから実際にストリームの再生が開始されるまで、遅延が生じます。この問題は pausedAtStart コマンドを使用して回避できます。ステージ外で play コマンドを呼び出してバッファリング処理を行い、実際に再生が開始された時点で RealMedia ストリームをステージに表示することができます。詳細については、pausedAtStart (RealMedia, Windows Media) エントリの例を参照してください。

例文

次の例は、スプライト 2 およびキャストメンバ "Real" の RealMedia ストリームの 56% がバッファリングされたことを示します。

```
-- Lingo syntax
put(sprite(2).percentBuffered) -- 56
put(member("Real").percentBuffered) -- 56

// JavaScript syntax
put(sprite(2).percentBuffered); // 56
put(member("Real").percentBuffered); // 56
```

関連事項

mediaStatus (RealMedia, Windows Media), pausedAtStart (RealMedia, Windows Media), state (RealMedia)

percentPlayed

使用法

```
member(whichCastMember).percentPlayed
the percentPlayed of member whichCastMember
```

解説

Shockwave Audio (SWA) キャストメンバプロパティ。指定した SWA ファイルが実際に再生された量の比率 (パーセント) を返します。

このプロパティの値は、SWA サウンドの再生が開始された後、または preLoadBuffer コマンドによってプリロードされた後にのみ調べることができます。このプロパティを変更することはできません。

例文

次のハンドラは、SWA ストリーミングキャストメンバ " フランク・シナトラ " の再生率を表示し、その値を " 再生率 " フィールドキャストメンバに入れます。

```
on exitFrame
    whatState = member("Frank Sinatra").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Played").text = string(member("Frank Sinatra").percentPlayed)
    end if
end
```

関連事項

percentStreamed （メンバ）

percentStreamed (3D)

使用法

```
member(whichCastMember).percentStreamed
```

解説

3D プロパティ。このプロパティを使用すると、ストリーミングされた 3D キャストメンバのパーセンテージを取得できます。このプロパティで取得できる値は、最初のファイルの読み込みの際の値か、または要求した最後のファイルのロードの際の値です。戻り値は 0 ～ 100 の整数で、デフォルト値はありません。

例文

次のステートメントは、キャストメンバ "PartyScene" のロードが完了したことを示します。

```
put member("PartyScene").percentStreamed
-- 100
```

percentStreamed （メンバ）

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.percentStreamed

// JavaScript syntax
memberOrSpriteObjRef.percentStreamed;
```

解説

Shockwave Audio (SWA) キャストメンバプロパティ、Flash キャストメンバプロパティ、および QuickTime スプライトプロパティ。

SWA ストリーミングサウンドの場合は、HTTP サーバまたは FTP サーバから既にストリーミングされた SWA ファイルの比率（パーセント）を取得します。SWA ファイルについては、percentPlayed プロパティとは異なり、バッファに保存されておりまだ再生されていないデータも数値に含まれます。このプロパティの値は、SWA サウンドの再生が開始された後、または preLoadBuffer コマンドによってプリロードされた後にのみ調べることができます。

Flash ムービーキャストメンバの場合は、このプロパティはメモリにストリームされた Flash ムービーの比率 (パーセント) を取得します。

QuickTime スプライトの場合は、このプロパティは再生された QuickTime ファイルの比率 (パーセント) を取得します。

このプロパティは、0 ～ 100% の範囲内の値になります。ローカルディスク上のファイルの場合は、このプロパティの値は 100 になります。インターネットからストリームされるファイルの場合は、受信したバイト数に応じて percentStreamed の値が増加していきます。このプロパティを変更することはできません。

例文

次の例は、" レイ・チャールズ " SWA ストリーミングキャストメンバのストリーミングの割合を表示し、その値をフィールドに入れます。

```
-- Lingo syntax
on exitFrame
    whatState = member("Ray Charles").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Streamed Displayer").text = string(member("Ray Charles").percentStreamed)
    end if
end

// JavaScript syntax
function exitFrame() {
    var whatState = member("Ray Charles").state;
    var pcStm = new String(member("Ray Charles").percentStreamed);
    if (whatState > 1 && whatState < 9) {
        member("Percent Streamed Displayer").text = pcStm;
    }
}
```

次のフレームスクリプトは、" スプラッシュスクリーン " という Flash ムービーがメモリに 60% ストリーミングされるまで、再生ヘッドを現在のフレームでループさせます。

```
-- Lingo syntax
on exitFrame
    if member("Splash Screen").percentStreamed < 60 then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var ssStrm = member("Splash Screen").percentStreamed;
    if (ssStrm < 60) {
        _movie.go(_movie.frame);
    }
}
```

関連事項

percentPlayed

percentStreamed (MP4Media/FLV)

使用法

```
sprite(1).percentStreamed
member(1).percentStreamed
```

解説

MP4Media/FLV メンバおよびスプライトのプロパティ。RTMP URL からビデオストリーミングでストリーミングされるデータの量を表します。percentStreamed はストリーミングされる合計データの割合（パーセント）です。

このプロパティは読み取り専用です。

例文

```
-- Lingo syntax
put(sprite("Mp4MediaSprite").percentStreamed
put(member("MP4Media/FLV").percentStreamed

// JavaScript syntax
put(sprite("Mp4MediaSprite").percentStreamed
put(member("MP4Media/FLV").percentStreamed
```

percentStreamed（サウンドオブジェクト）

使用法

```
soundobj.percentStreamed (Read-only)
```

解説

サウンドオブジェクトプロパティ。ストリーミングされたデータの継続時間を、ストリーミングされるファイルの合計の継続時間のパーセンテージとして返します。

このプロパティは読み取り専用です。

例文

以下の例では、サウンドオブジェクトのストリーミングされるデータのパーセンテージを返します。

```
--Lingo syntax
on mouseUp me
put SoundObjectRef.percentStreamed
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.percentStreamed);
}
```

period

使用法

```
timeoutObject.period
```

解説

timeoutObject によってタイムアウトハンドラへ送信されるタイムアウトイベントの間隔を示すオブジェクトプロパティです。単位はミリ秒です。

このプロパティは、値を調べたり、設定することができます。

例文

次のタイムアウトハンドラは、最小の間隔の 2 秒（2000 ミリ秒）に達するまでの間は、タイムアウトが発生するたびにその period を 1 秒短縮します。

```
on handleTimeout timeoutObject
    if timeoutObject.period > 2000 then
        timeoutObject.period = timeoutObject.period - 1000
    end if
end handleTimeout
```

関連事項

name (timeout), persistent, target, time （タイムアウトオブジェクト）, timeout(), timeoutHandler, timeoutList

persistent

使用法

```
timeoutObject.persistent
```

解説

現行ムービーの再生が停止したときに、指定した [タイムアウトオブジェクト] が the timeoutList から除外されるかどうかを示すオブジェクトプロパティです。[タイムアウトオブジェクト] がアクティブな状態を続けている場合は TRUE、ムービーの再生が停止したときにタイムアウトオブジェクトが削除される場合は FALSE、デフォルト値は FALSE です。

次のプロパティを TRUE に設定すると、timeoutObject は他のムービーでタイムアウトイベントの作成を続けます。この機能は、go to movie コマンドを使用して、あるムービーを別のムービーへ分岐する場合に便利です。

例文

次のステートメントはタイムアウトオブジェクトを作成し、持続的なものとします。

```
-- Lingo syntax
gTO = timeout().new("test",50000,"sampleTimeout",0)
gTO.persistent = TRUE

// JavaScript syntax
_global.gTO = new timeout("test",50000,"sampleTimeout",0)
_global.gTO.persistent = 1;
```

関連事項

name (timeout), period, target, time （タイムアウトオブジェクト）, timeout(), timeoutHandler, timeoutList

picture （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.picture

// JavaScript syntax
memberObjRef.picture;
```

解説

ビットマップ、テキスト、PICT などのキャストメンバに関連づけられている画像を示すキャストメンバプロパティです。キャストメンバのセンターポイントの変更や、fileName プロパティを使って再リンクした後の画像の変更などを更新するときは、次のステートメントを使用します。

```
member(whichCastMember).picture = member(whichCastMember).picture
```

[キャストメンバ] には、対象となるキャストメンバの名前または番号を指定します。

キャストメンバへの変更は RAM に保存されるため、このプロパティはオーサリング時に使うと最適です。プロジェクタ内ではこのプロパティを設定しないでください。

プロパティは、調べたり、変更できます。

例文

次のステートメントは、変数 pictHolder に " 夕焼け " というキャストメンバ内の画像を設定します。

```
-- Lingo syntax
pictHolder = member("Sunset").picture

// JavaScript syntax
var pictHolder = member("Sunset").picture;
```

関連事項

type (sprite)

picture （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.picture

// JavaScript syntax
windowObjRef.picture;
```

解説

ステージウィンドウまたはウィンドウ内のムービー（MIAW）の現在の内容の画像を取得するウィンドウプロパティです。読み取り専用。

取得したビットマップデータは、既存のビットマップに適用できます。また、新規ビットマップの作成にも使用できます。

画像が存在しない場合は、VOID（Lingo）または null（JavaScript シンタックス）を返します。

例文

次のステートメントは、ステージの現在の内容をビットマップキャストメンバに取り込みます。

```
-- Lingo syntax
member("Stage image").picture = _movie.stage.picture

// JavaScript syntax
member("Stage image").picture = _movie.stage.picture;
```

関連事項

ウィンドウ

platform

使用法

```
the platform
```

解説

プロジェクタが作成された対象プラットフォームの種類を示すシステムプロパティです。

このプロパティは、値を調べられますが、設定はできません。

このプロパティは次のいずれかの値になります。

使用できる値	プラットフォーム
Mac,PowerPC	PowerPC Mac
Windows,32	Windows 95 または Windows NT

将来プラットフォームが追加された場合に互換性を保つために、contains 演算子を使ってプラットフォームを調べることをお勧めします。

例文

次のステートメントは、ムービーが作成されたプラットフォームを返します。

```
-- Lingo syntax
if the platform contains "Windows,32" then
    alert ("This movie has been created using Windows")
end if
```

関連事項

runMode

playBackMode

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.playBackMode

// JavaScript syntax
memberOrSpriteObjRef.playBackMode;
```

解説

Flash ムービーやアニメーション GIF キャストメンバのテンポを次のいずれかの値で制御するキャストメンバおよびスプライトプロパティです。

- #normal (デフォルト) - Flash ムービーまたはアニメーション GIF ファイルを可能な限り元のテンポに近い速 で再生します。
- #lockStep - Flash ムービーまたはアニメーション GIF ファイルのフレームを Director ムービーのフレームに合わせて再生します。
- #fixed - Flash ムービーまたはアニメーション GIF ファイルを fixedRate プロパティで指定されたレートで再生します。

このプロパティは、値を調べたり、設定することができます。

例文

次のスプライトスクリプトは、Flash ムービースプライトのフレームレートを Director ムービーに合わせます。

```
-- Lingo syntax
property spriteNum

on beginSprite(me)
    sprite(spriteNum).playBackMode = #lockStep
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).playBackMode = symbol("lockStep");
}
```

関連事項

fixedRate

playing

使用法

```
-- Lingo syntax
spriteObjRef.playing

// JavaScript syntax
spriteObjRef.playing;
```

解説

Flash ムービーが再生中（TRUE）か、停止中か（FALSE）を示す Flash スプライトプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のフレームスクリプトは、チャネル 5 の Flash ムービースプライトが再生中かどうか調べ、再生中でないときは再生を開始します。

```
-- Lingo syntax
on enterFrame
    if not sprite(5).playing then
        sprite(5).play()
    end if
end

// JavaScript syntax
function enterFrame() {
    var plg = sprite(5).playing;
    if (plg == 0) {
        sprite(5).play();
    }
}
```

playing (3D)

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.playing
member(whichCastmember).model(whichModel).bonesPlayer.playing
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D プロパティ。モディファイアのアニメーション再生エンジンが実行中であるか（TRUE）停止中であるか（FALSE）を示します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" エイリアン 3" というモデルの #keyframePlayer アニメーション再生エンジンが現在実行中であることを示します。

```
put member("newaliens").model("Alien3").keyframePlayer.playing
-- 1
```

関連事項

play() (3D), pause() (3D), playlist, queue() (3D)

playlist

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.playlist
member(whichCastmember).model(whichModel).bonesPlayer.playlist
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D プロパティ。プロパティリストのリニアリストを返します。各プロパティリストは、モディファイアによる再生をキューで待機しているモーションを表します。

各プロパティリストには以下のプロパティがあります。

- #name。再生されるモーションの名前です。
- #loop。モーションの再生がループするかどうかを示します。
- #startTime。アニメーションの再生を開始する時間です (単位はミリ秒)。
- #endTime。アニメーションの再生を終了する時間、またはモーションがループに入る時間です（単位はミリ秒)。負の値の場合は、モーションは終わりまで再生されます。
- #scale。モーションの再生レートです。この値にモディファイアの playRate プロパティの値を掛け算した値によって、モーションの実際の再生速度が決まります。

playlist プロパティは、値を調べることはできますが、設定できません。この値を操作するには、queue()、play()、playNext()、および removeLast() の各コマンドを使用します。

例文

次のステートメントは、モデル " 散歩する人 " の現行キューで待機しているモーションをメッセージウィンドウに表示します。

```
-- Lingo syntax
put member("3Dobjects").model("Stroller").bonesPlayer.playList
// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).bonesPlayer.playList);
```

関連事項

play() (3D), playNext() (3D), removeLast(), queue() (3D)

playRate (3D)

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.playRate
member(whichCastmember).model(whichModel).keyframePlayer.playRate
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D プロパティ。再生されるモーションのローカルタイムのスケールの乗数です。モーションがモデルによって実行される速度は、このプロパティだけでは決まりません。

モデルによるモーションの再生は、play() コマンドまたは queue() コマンドによって指定されます。play() コマンドまたは queue() コマンドの scale パラメータを、モディファイアの playRate プロパティと掛け算して得られた結果が、モーションの再生速度になります。

例文

次のステートメントは、"GreenAlien" というモデルの keyframePlayer モディファイアの playRate プロパティを 3 に設定します。

```
member("newAliens").model("GreenAlien").keyframePlayer.playRate = 3
```

関連事項

play() (3D), queue() (3D), playlist, currentTime (3D)

playRate (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.playRate

// JavaScript syntax
dvdObjRef.playRate;
```

解説

現在の位置から DVD の再生または逆再生をするレートを定義する DVD プロパティです。読み取り / 書き込み可能。

負の値の場合は DVD を逆再生し、正の値の場合は DVD を再生します。

関連事項

DVD

playRate（QuickTime、AVI、MP4、FLV）

使用法

```
-- Lingo syntax
spriteObjRef.playRate

// JavaScript syntax
spriteObjRef.playRate;
```

解説

指定したチャネル内のデジタルビデオの再生速度を制御するためのデジタルビデオスプライトプロパティです。

playRate はデジタルビデオの再生速度を指定する値です。設定値 1 で通常の再生、-1 で逆再生、0 で停止になり、その他の値を設定することもできます。値を設定して、ビデオを通常よりも速くまたは遅く再生できます。0.5 では通常の 2 分の 1 の速度でビデオが再生されます。2 では通常の 2 倍の速度でビデオが再生されます。playRate プロパティを 1 を超える値に設定すると、ムービーを再生するときに使用するコンピュータのパフォーマンスやデジタルビデオスプライトなどの状況によって、フレームがドロップすることもあります。

注意：MP4Media/FLV メンバの場合、playRate の有効な値は 0 ～ 4 です。MP4Media/FLV メンバでは、playRate を負の値にすることはできません。ただし、Quicktime および AVI メンバでは、playRate に負の値を指定してビデオを戻すことができます。

例文

次のステートメントは、スプライトチャネル 9 上のビデオの再生速度を通常の速度に設定します。

```
-- Lingo syntax
sprite(9).playRate = 1

// JavaScript syntax
sprite(9).playRate = 1;
```

次のステートメントは、スプライトチャネル 9 上のビデオを逆再生します。

```
-- Lingo syntax
sprite(9).playRate = -1

// JavaScript syntax
sprite(9).playRate = -1;
```

関連事項

duration （メンバ）, currentTime (QuickTime, AVI)

playRate（サウンドオブジェクト）

使用法

```
soundobj.playRate
```

解説

サウンドオブジェクトプロパティ。再生の速度を変更します。このプロパティはオーディオのピッチにも影響します。

playRate は、オリジナルのオーディオ再生レートを基準として再生される必要があるオーディオのレートを決定します。再生レートの精度は 0.01 です。

playRate は 0.25 から 4 までの範囲の値を取ることができます。デフォルト値は 1 です。

例文

```
on mouseUp me
soundobjectref.playrate = 0.50  --  Changes the play rate of the sound object to 0.50.
end

// JavaScript syntax
function mouseUp(){
soundobjectRef.playrate = 0.25;  // Changes the play rate of the sound object to 0.25.
}
```

playRate (Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.playRate

// JavaScript syntax
windowsMediaObjRef.playRate;
```

解説

Windows Media のプロパティ。Windows Media のキャストメンバの再生速度を定義します。読み取り専用。

例文

次のステートメントは、キャストメンバ 10 の再生速度をメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(member(10).playRate)

// JavaScript syntax
trace(member(10).playRate);
```

関連事項

Windows Media

pointAtOrientation

使用法

```
member(whichCastmember).model(whichModel).pointAtOrientation
member(whichCastmember).group(whichGroup).pointAtOrientation
member(whichCastmember).light(whichLight).pointAtOrientation
member(whichCastmember).camera(whichCamera).pointAtOrientation
```

解説

3D モデル、ライト、グループ、およびカメラプロパティ。このプロパティを使用すると、参照先オブジェクトの pointAt コマンドに対する応答方法を取得または設定できます。このプロパティは、2 つのオブジェクト基準ベクトルのリニアリストです。リストの最初のベクトルはオブジェクトの前方向となる方向を定義し、2 番目のベクトルはオブジェクトの上方向となる方向を定義します。

オブジェクトの前方向と上方向は、互いに垂直である必要はありませんが、互いに平行であってはいけません。

例文

次のステートメントは、"bip01" というモデルのオブジェクト基準の前方向ベクトルと上方向ベクトルを表示します。

```
put member("scene").model("bip01").pointAtOrientation
-- [vector(0.0000, 0.0000, 1.0000), vector(0.0000, 1.0000, 0.0000)]
```

関連事項

pointAt

pointOfContact

使用法

```
collisionData.pointOfContact
```

解説

3D collisionData プロパティ。2 つのモデルが衝突したポイントを示すベクトルを返します。

collisionData オブジェクトは、#collideWith および #collideAny イベントのパラメータとして、registerForEvent, registerScript, および setCollisionCallback の各コマンドによって指定されたハンドラに送られます。

衝突モディファイアが追加されているモデル間で衝突が発生すると、#collideWith イベントと #collideAny イベントが送信されます。このとき、モデルのモディファイアの resolve プロパティは、TRUE に設定されている必要があります。

このプロパティは、値を調べられますが、設定はできません。

例文

次の例は、2 つの部分で構成されています。まずコードの 1 行目で、#collideAny イベントに #explode ハンドラを登録します。2 つ目は、#explode ハンドラです。キャストメンバ "MyScene" の 2 つのモデルが衝突すると、#explode ハンドラが呼び出され、このハンドラに対して collisionData パラメータが送信されます。#explode ハンドラの最初の 9 行は、"SparkSource" というモデルリソースを作成し、そのプロパティを設定します。このモデルリソースは、パーティクルの 1 回のバーストです。ハンドラの 10 行目は、モデルリソース "SparkSource" を使って、"SparksModel" というモデルを作成します。ハンドラの最後の行は、"SparksModel" の位置を、衝突が発生した位置に設定します。この例全体から、衝突によって火花が散る (バースト) 効果が得られます。

```
member("MyScene").registerForEvent(#collideAny, #explode, 0)

on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource",#particle)
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.direction = vector(0, 0, 1)
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
end
```

関連事項

modelA, modelB

position (byte array)

使用法

```
byteArrayObject.position
```

解説

バイト数で表された ByteArray オブジェクト内でのファイルポインタの現在位置を移動または取得する、バイト配列プロパティです。

例文

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put bArray.position
bArray.position=1

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(bArray.position);
bArray.position=1;
```

position （トランスフォーム）

使用法

```
member(whichCastmember).node(whichNode).transform.position
member(whichCastmember).node(whichNode).getWorldTransform().position
transform.position
```

解説

3D プロパティ。このプロパティを使用すると、トランスフォームの位置コンポーネントを取得または設定できます。トランスフォームは、特定の基準に基づくスケール、位置、および回転を定義します。このプロパティのデフォルト値は vector(0,0,0) です。

ノードは、カメラ、グループ、ライト、またはモデルオブジェクトのいずれかです。ノードのトランスフォームの position を設定すると、トランスフォームの基準に基づくオブジェクトの位置を定義できます。getWorldTransform().position を使って、オブジェクトのワールド基準のトランスフォームの position プロパティを設定すると、ワールドの原点を基準にしたオブジェクトの位置を定義できます。transform.position を使って、オブジェクトの親基準のトランスフォームの position プロパティを設定すると、親ノードを基準にしたオブジェクトの位置を定義できます。

モデル、ライト、カメラ、またはグループオブジェクトの worldPosition プロパティを使用すると、オブジェクトに対して getWorldTransform().position を使用するよりも簡単にこのプロパティを設定できます。

例文

次のステートメントは、" 球体 01" というモデルの親を基準にした位置を表示します。

```
-- Lingo syntax
    put(member("3Dobjects").model("Sphere01").transform.position)

// JavaScript syntax
    put(member("3Dobjects").getPropRef("model",2).transform.position);
```

関連事項

transform （プロパティ）, getWorldTransform(), rotation （トランスフォーム）, scale （トランスフォーム）

positionReset

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.positionReset
member(whichCastmember).model(whichModel).keyframePlayer.positionReset
```

解説

keyframePlayer モディファイアおよび bonesPlayer モディファイアの 3D プロパティ。モーションの終了後にモデルが開始位置に戻るかどうかを示します。TRUE の場合には戻り、FALSE の場合には戻りません。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、モデル " モンスター " がモーションの完了後に元の位置に戻らないように設定します。

```
member("NewAlien").model("Monster").keyframePlayer.positionReset = FALSE
```

関連事項

currentLoopState

posterFrame

使用法

```
-- Lingo syntax
memberObjRef.posterFrame

// JavaScript syntax
memberObjRef.posterFrame;
```

解説

サムネイル画像として使用する Flash ムービーキャストメンバのフレームを制御する Flash キャストメンバプロパティです。フレーム番号を示す整数で指定します。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 1 です。

例文

次のハンドラは、Flash ムービーキャストメンバの参照とフレーム番号をパラメータとして、指定したフレーム番号を指定したムービーのサムネイルに設定します。

```
-- Lingo syntax
on resetThumbnail(whichFlashMovie, whichFrame)
    member(whichFlashMovie).posterFrame = whichFrame
end

// JavaScript syntax
function resetThumbnail(whichFlashMovie, whichFrame) {
    member(whichFlashMovie).posterFrame = whichFrame;
}
```

preferred3dRenderer

使用法

```
-- Lingo syntax
_movie.preferred3dRenderer

// JavaScript syntax
_movie.preferred3dRenderer;
```

解説

ムービープロパティ。このプロパティを使用すると、特定のムービー内で 3D スプライトの描画に使用するデフォルトのレンダラを取得または設定できます。ただし、このレンダラがクライアントマシンで使用できる必要があります。読み取り / 書き込み可能。

指定したレンダラがクライアントマシンで使用できない場合は、使用できる中で最も適したレンダラがムービーによって選択されます。

このプロパティでは、以下の値を使用できます。

- #openGL。ハードウェアアクセラレーターの openGL ドライバを指定します。Macintosh と Windows の両方のプラットフォームで使用できます。
- #directX9。ハードウェアアクセラレーターの DirectX® 9 ドライバを指定します。Windows プラットフォームだけで使用できます。#auto は Windows のレンダラを DirectX 9 に設定します。Mac® -Intel® では、#OpenGL レンダラのみが使用可能です。プロパティインスペクタの「ムービー」タブで使用する 3D レンダラメニューに DirectX 9 が追加されます。DirectX 9 を使用する際、既存の機能に変化はありません。ただし、パフォーマンスは強化されています。
- #directX7_0。ハードウェアアクセラレーターの DirectX 7 ドライバを指定します。Windows プラットフォームだけで使用できます。
- #directX5_2。ハードウェアアクセラレーターの DirectX 5.2 ドライバを指定します。Windows プラットフォームだけで使用できます。
- #software。Director の内蔵ソフトウェアレンダラを指定します。Macintosh と Windows の両方のプラットフォームで使用できます。
- #auto。最適なレンダリング方法が自動的に選択されます。このプロパティのデフォルト値です。

このプロパティに設定した値は、レンダラサービスオブジェクトの renderer プロパティのデフォルトとして使用されます。

このプロパティは、getRendererServices() オブジェクトの renderer プロパティとは異なります。preferred3dRenderer は、優先して使用するレンダラを指定するのに対し、getRendererServices() オブジェクトの renderer プロパティは、ムービーによって実際に使用されているレンダラを示します。

Shockwave Player のユーザは、Shockwave Player の 3D レンダラコンテキストメニューによってレンダラを指定できます。ユーザが「コンテンツの設定に従う」を選択した場合は、renderer プロパティまたは preferred3DRenderer プロパティに設定されているレンダラによってムービーが描画されます（ユーザのシステムで使用できる場合）。それ以外の場合は、ユーザが選択したレンダラが使用されます。

例文

次のステートメントは、ユーザのシステムで利用可能な最適の 3D レンダラをムービーが選択できるように設定します。

```
-- Lingo syntax
_movie.preferred3dRenderer = #auto

// JavaScript syntax
_movie.preferred3dRenderer = "auto";
```

関連事項

getRendererServices(), ムービー , renderer

preLoad (3D)

使用法

```
member(whichCastmember).preload
memberReference.preload
```

解説

3D プロパティ。再生前にデータをプリロードするか (TRUE)、または再生中にデータをストリームするか (FALSE) を取得または設定します。このプロパティは、リンクされたファイルでのみ使用できます。デフォルト値は FALSE です。

Director では、preLoad プロパティを TRUE に設定すると、再生が開始する前にキャストメンバのロードが完了されます。Shockwave Player では、preLoad プロパティを TRUE に設定すると、ムービーの再生が始まるときにキャストメンバのストリーミングが開始されます。ストリーミングされる 3D キャストメンバに Lingo を使用した操作を実行する前に、キャストメンバの state プロパティが 2 以上であることを確認してください。

例文

次のステートメントは、キャストメンバ "PartyScene" の preload プロパティを FALSE に設定します。これにより、外部からリンクされたメディアは、再生中に "PartyScene" にストリームされます。

```
member("PartyScene").preload = FALSE
member("3D world").preload
```

関連事項

state (3D)

preLoad （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.preLoad

// JavaScript syntax
memberObjRef.preLoad;
```

解説

[キャストメンバ] で指定したデジタルビデオキャストメンバをメモリ内にプリロードできるか（TRUE)、またはプリロードできないか（FALSE, デフォルト）を決めるキャストメンバプロパティです。TRUE の場合は、デジタルビデオキャストメンバプロパティダイアログボックスの「プリロードを可能にする」オプションを選択することと同じ結果になります。

Flash ムービーキャストメンバについては、スプライトの最初のフレームを表示する前に、全体を RAM にロードしておく必要がある (TRUE) か、再生の進行に合わせてストリーミングすればよい (FALSE、デフォルト) かを設定するプロパティです。ただし、リンクされた Flash ムービーのアセットが外部ファイルに格納されている場合にのみ有効です。streamMode および bufferSize プロパティで、キャストメンバをメモリにストリーミングする方法を設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、QuickTime ムービー " 回転椅子 " がメモリにプリロードできるかどうかを、メッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Rotating Chair").preload)

// JavaScript syntax
put(member("Rotating Chair").preload);
```

関連事項

bufferSize, streamMode

preLoadEventAbort

使用法

```
-- Lingo syntax
_movie.preLoadEventAbort

// JavaScript syntax
_movie.preLoadEventAbort;
```

解説

ムービープロパティ。キーの押下またはマウスのクリックによって、キャストメンバのプリロードを停止できるかどうかを示します。停止できるときは TRUE、停止できないときは FALSE （デフォルト）です。読み取り / 書き込み可能。

プロパティの変更は現在のムービーに適用されます。

例文

次のステートメントは、キーの押し下げやマウスのクリックによって、ユーザがキャストメンバのプリロードを停止できるようにします。

```
-- Lingo syntax
_movie.preLoadEventAbort = TRUE

// JavaScript syntax
_movie.preLoadEventAbort = true;
```

関連事項

ムービー

preLoadMode

使用法

```
-- Lingo syntax
castObjRef.preLoadMode

// JavaScript syntax
castObjRef.preLoadMode;
```

解説

指定したキャストライブラリのプリロードモードを定義するキャストライブラリプロパティです。読み取り / 書き込み可能。

preLoadMode の有効な値は次のとおりです。

- 0。必要に応じてキャストライブラリを読み込みます。デフォルト値です。
- 1。フレーム 1 の前にキャストライブラリを読み込みます。
- 2。フレーム 1 の後にキャストライブラリを読み込みます。

このプロパティを設定すると、キャストプロパティダイアログボックスにある「キャストのロード」を設定し た場合と同じ効果があります。

例文

次のステートメントは、ムービーがフレーム 1 に入る前に " ボタン " というキャストのメンバを Director にプリロードさせます。

```
-- Lingo syntax
castLib("Buttons").preLoadMode = 1

// JavaScript syntax
castLib("Buttons").preLoadMode = 1;
```

関連事項

キャストライブラリ

preLoadRAM

使用法

```
the preLoadRAM
```

解説

デジタルビデオのプリロード用に使える RAM の量を設定するプロパティです。このプロパティは、値を調べたり、変更することができます。

デジタルビデオキャストメンバのプリロードに一定以上のメモリを割り当てないことで、その他の種類のキャストメンバもプリロードできるようにするというように、メモリを管理するのに使用できるプロパティです。preLoadRAM が FALSE に設定されていると、ビデオキャストメンバのプリロードに利用可能なすべてのメモリが割り当てられることになります。

どれだけの RAM を割り当てればよいのか予測することは一般には不可能です。メモリの必要量は、圧縮率、キーフレームの個数、シーンチェンジなどによっても異なります。

通常、最初の 2 秒分をプリロードしてからストリーミングを始めるようにすると安全です。

ムービーのデータレートがわかっていれば、the preLoadRAM の設定を見積もることができます。例えば、データレートが毎秒 300K ならば、2 秒分をプリロードするには、preLoadRAM を 600K と設定します。おおまかな見積もりですが、ほとんどの状況でうまく動作します。

例文

次のステートメントは、preLoadRAM を 600K に設定して、毎秒 300K のレートのムービーを 2 秒分プリロードします。

```
--Lingo
set the preLoadRAM = 600

// Javascript
_system.preLoadRAM = 600;
```

関連事項

loop （キーワード）, next

preLoadTime

使用法

```
-- Lingo syntax
memberObjRef.preLoadTime

// JavaScript syntax
memberObjRef.preLoadTime;
```

解説

キャストメンバおよびサウンドチャネルのプロパティ。キャストメンバの場合は、再生を開始する前、または preLoadBuffer コマンドを使用する際に、Shockwave Audio (SWA) ストリーミングキャストメンバをダウンロードする量を秒単位で指定します。デフォルトは 5 秒です。

このプロパティは、SWA ストリーミングキャストメンバが停止したときにのみ設定できます。

サウンドチャネルに対しては、この値はキュー内の指定されたサウンドに適用されます。指定がない場合は、現在再生されているサウンドに適用されます。

例文

次のハンドラは、SWA ストリーミングキャストメンバ " ルイ・アームストロング " のプリロードのダウンロード時間を 6 秒に設定します。実際のプリロードは、「preLoadBuffer」または「play」コマンドが発行されたときに行われます。

```
-- Lingo syntax
on mouseDown
    member("Louis Armstrong").stop()
    member("Louis Armstrong").preLoadTime = 6
end

// JavaScript syntax
function mouseDown() {
    member("Louis Armstrong").stop();
    member("Louis Armstrong").preLoadTime = 6;
}
```

次のステートメントは、サウンドチャネル 1 で再生中のサウンドの preLoadTime を返します。

```
-- Lingo syntax
put sound(1).preLoadTime

// JavaScript syntax
trace(sound(1).preLoadTime);
```

関連事項

preLoadBuffer()

primitives

使用法

```
getRendererServices().primitives
```

解説

3D 関数。新しいモデルリソースの作成に使用できるプリミティブのタイプのリストを返します。

例文

次のステートメントは、使用できるプリミティブのタイプを表示します。

```
--Lingo
put getRendererServices().primitives

// Javascript
put( getRendererServices().primitives);
```

関連事項

getRendererServices(), newModelResource

productName

使用法

```
-- Lingo syntax
_player.productName

// JavaScript syntax
_player.productName;
```

解説

Director アプリケーションの名前を返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、Director アプリケーションの名前をメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(_player.productName)

// JavaScript syntax
trace(_player.productName);
```

関連事項

プレーヤ

productVersion

使用法

```
-- Lingo syntax
_player.productVersion

// JavaScript syntax
_player.productVersion;
```

解説

Director アプリケーションのバージョン番号を返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、Director アプリケーションのバージョンをメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(_player.productVersion)

// JavaScript syntax
trace(_player.productVersion);
```

関連事項

プレーヤ

projection

使用法

```
sprite(whichSprite).camera.projection
camera(whichCamera).projection
member(whichCastmember).camera(whichCamera).projection
```

解説

3D プロパティ。このプロパティを使用すると、カメラの投影スタイルを取得または設定できます。使用できる値は、#perspective (デフォルト) および #orthographic です。

projection プロパティが #perspective の場合、カメラに近い位置にあるオブジェクトは、カメラから遠い位置にあるオブジェクトより大きく見えます。この場合は、projectionAngle プロパティまたは fieldOfView プロパティで指定する垂直方向の投影角度によって、表示されるワールドの範囲が決まります。水平方向の投影角度は、カメラの rect プロパティのアスペクト比によって決まります。

projection プロパティが #orthographic の場合、オブジェクトの見た目の大きさはカメラからの距離に影響されません。この場合は、orthoHeight プロパティで指定する、スプライトに垂直にフィットするワールド単位の数によって、表示されるワールドの範囲が決まります。垂直方向の投影の幅は、カメラの rect プロパティのアスペクト比によって決まります。

例文

次のステートメントは、スプライト 5 のカメラの projection プロパティを #orthographic に設定します。

```
sprite(5).camera.projection = #orthographic
```

関連事項

fieldOfView (3D), orthoHeight, fieldOfView (3D)

purgePriority

使用法

```
-- Lingo syntax
memberObjRef.purgePriority

// JavaScript syntax
memberObjRef.purgePriority;
```

解説

キャストメンバのメモリ解放順を指定するメンバプロパティです。読み取り / 書き込み可能。

キャストメンバのメモリ解放順は、メモリがいっぱいになったときに、どのキャストメンバをメモリから解放するかを Director が判断する際に基準とするものです。解放順が高いキャストメンバほど、先に解放されることになります。purgePriority で使える値は次のとおりです。

- 0— 解放しない
- 1- 最後に解放する
- 2- 次に解放する
- 3 - 標準（デフォルト）

デフォルトの「標準」に設定すると、キャストメンバはランダムにメモリから解放されます。「次に開放する」、「最後に開放する」、「開放しない」を設定すると、メモリ解放順をユーザがある程度制御することができますが、多くのキャストメンバが「最後に開放する」や「開放しない」に指定されていると、ムービーそのものがメモリ不足に陥る可能性が高くなってしまいます。

ムービーで使用するキャストライブラリのサイズが利用可能なメモリ総量よりも大きいときに、キャストメンバに purgePriority を設定してメモリ管理をすると便利です。一般的に、頻繁に登場するキャストメンバのメモリ解放順を低い値に設定しておくと、再読み込みする回数が少なくなり、キャストの読み込み時に生じる一時停止を最小限に抑えることができます。

例文

次のステートメントは、" 背景 " キャストメンバのメモリ解放順を 3 に設定し、メモリに余裕がなくなってくると、早い段階でメモリから解放されるようにします。

```
-- Lingo syntax
member("Background").purgePriority = 3

// JavaScript syntax
member("Background").purgePriority = 3;
```

関連事項

メンバ

quad

使用法

```
-- Lingo syntax
spriteObjRef.quad

// JavaScript syntax
spriteObjRef.quad;
```

解説

ステージ上のスプライトの角の点を示す浮動小数点値の 4 つのポイントのリストを含むスプライトプロパティです。読み取り / 書き込み可能。

このポイントは、左上、右上、右下、左下の順に構成されています。

ポイントそのものは、パースやその他の画像のひずみを作成するために操作できます。

スプライトの矩形を操作した後は、puppetSprite（[スプライト番号],FALSE) を使用してスクリプトされたスプライトをオフにすることにより、スプライトをスコア値にリセットすることができます。スプライトの矩形が無効になると、スプライトに回転や傾斜を実行することはできません。

例文

次のステートメントは、スプライトを表す典型的なリストを表示します。

```
-- Lingo syntax
put(sprite(1).quad)

// JavaScript syntax
put(sprite(1).quad);
```

スプライトプロパティの quad を変更するときは、値を変更した後で、ポイントのリストをリセットする必要があります。これは、プロパティの値を変数に設定すると、リストそのものではなく、リストのコピーを変数に格納することになるからです。変更した内容を有効にするには、次のようなシンタックスを使います（Lingo にのみ適用）。

```
-- Lingo syntax
currQuadList = sprite(5).quad
currQuadList[1] = currQuadList[1] + point(50, 50)
sprite(5).quad = currQuadList
```

関連事項

point(), puppetSprite(), スプライト

quality

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.quality

// JavaScript syntax
memberOrSpriteObjRef.quality;
```

解説

Flash ムービースプライトをレンダリングするときに、アンチエイリアスを使うかどうかを制御するキャストメンバおよびスプライトプロパティです。アンチエイリアスを使うと、画質が向上する代わり、ムービーの再生速度が遅くなります。quality プロパティは、次のいずれかの値になります。

- #autoHigh - Director がアンチエイリアスを使用してスプライトのレンダリングを開始します。ムービーの実際のフレームレートが指定されたフレームレートに満たないときは、Director はアンチエイリアスをオフにします。この設定により、視覚的な品質よりも再生速度が優先されます。
- #autoLow - Director はアンチエイリアスなしで、ムービーのレンダリングを開始します。Flash プレーヤは、コンピュータプロセッサで処理可能と判断した場合に、アンチエイリアスを起動します。この設定では、常に視覚的品質が優先されます。
- #high（デフォルト）- ムービーは常にアンチエイリアスを使用して再生されます。
- #low - ムービーは常にアンチエイリアスを使用せずに再生されます。

quality プロパティは、値を調べることも、設定することもできます。

例文

次のスプライトスクリプトは、ムービーを再生しているコンピュータの色深度をチェックします。色深度が 8 ビット以下 (256 色) に設定されている場合、スクリプトはチャネル 1 のスプライトの quality を #low と設定します。

```
-- Lingo syntax
on beginSprite me
    if _system.colorDepth <= 8 then
        sprite(1).quality = #low
    end if
end

// JavaScript syntax
function beginSprite() {
    var clrDp = _system.colorDepth;
    if (clrDp <= 8) {
        sprite(1).quality = symbol("low");
    }
}
```

quality (3D)

使用法

```
member(whichCastmember).texture(whichTexture).quality
member(whichCastmember).shader(whichShader).texture(whichTexture).quality
member(whichCastmember).model(whichModel).shader.texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shader.texturelist[TextureListIndex].quality
member(whichCastmember).model(whichModel).shaderList[shaderListIndex]. texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shaderList[ shaderListIndex ]. texturelist[ TextureListIndex
].quality
```

解説

3D テクスチャプロパティ。このプロパティを使用すると、テクスチャに適用するミップマッピングのレベルを制御することによって、テクスチャの画質を取得または設定できます。ミップマッピングとは、オリジナルの画像サイズよりも小さい複数のサイズで、テクスチャ画像の追加バージョンを作成するプロセスです。3D Xtra 拡張機能は、画面に表示されているモデルの現在のサイズに最も適した画像のバージョンを使用し、必要に応じて使用する画像のバージョンを変更します。トリリニアミップマッピングはバイリニアミップマッピングに比べ高画質ですが、より多くのメモリを必要とします。

ミップマッピングとフィルタリングは、テクスチャの外観を良くするという点では同じですが、本質的には異なります。フィルタリングは、テクスチャ領域におけるエラーを分散させて、エラーが偏らないようにします。ミップマッピングは、再サンプリングを実行して、画像を適切なサイズにします。

このプロパティは次の値をとることができます。

- #low。オフと同じです。テクスチャに対してミップマッピングは使用されません。
- #medium。テクスチャに対して低画質 (バイリニア) ミップマッピングが使用されます。
- #high。テクスチャに対して高画質 (トリリニア) ミップマッピングが使用されます。

デフォルトは #low です。

例文

次のステートメントは、" 火星の地図 " というテスクチャの quality プロパティを #medium に設定します。

```
member("scene").texture("Marsmap").quality = #medium
```

関連事項
nearFiltering

radius

使用法

```
modelResourceObjectReference.radius
member(whichCastmember).modelResource(whichModelResource).radius
```

解説
3D モデルプロパティ。#sphere または #cylinder というタイプのモデルリソースを指定すると、モデルの半径を取得または設定できます。

radius プロパティでは、モデルリソースの作成に使用する描画の半径を指定できます。このプロパティには、常に 0.0 より大きい値を設定する必要があります。デフォルト値は 25.0 です。

例文
次の例は、モデルリソース " 球体 01" の半径を示します。

```
--Lingo
put member("3Dobjects").modelResource("Cylinder01").radius

// Javascript
put(member("3Dobjects").getPropRef("modelResource",11).radius);
```

randomSeed

使用法

```
the randomSeed
```

解説
random() 関数を使用して乱数を生成するためのシード値を指定するためのシステムプロパティです。

同じシードを使うと、乱数のシーケンスも同じになります。プログラム作成中にデバッグをするときに使うと便利です。連続使用中に ticks 値が重複することはまずあり得ないので、ticks プロパティを使うと、一意なランダムシードを容易に作ることができます。

このプロパティは、値を調べたり、設定することができます。

例文
次のステートメントは、randomSeed 値をメッセージウィンドウに表示します。

```
--Lingo
put the randomSeed

// Javascript
put(_system.randomSeed);
```

関連事項
random(), milliseconds

recordFont

使用法

```
recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {, userFontName})
```

解説

TrueType または Type 1 フォントをキャストメンバとして埋め込むコマンドです。これらのフォントが一旦埋め込まれると、作成者はシステムにインストールされている他のフォントと同じように使うことができます。

recordFont を使用する前に、new() コマンドを使用して空フォントのキャストメンバを作成する必要があります。

- [フォント] - 記録するオリジナルフォントの名前。
- [スタイル] - オリジナルフォントのフェースを示すシンボルのリスト。#plain、#bold、#italic を指定できます。このパラメータが省略されると、#plain が使われます。
- [ビットマップサイズ] - ビットマップを記録するときのサイズを指定する整数のリスト。このパラメータは空でも使用できます。このパラメータが省略されると、ビットマップは生成されません。ビットマップは、通常小さいポイントサイズ (14 ポイント以下) のほうが視覚的に優れていますが、より多くのメモリを必要とします
- [サブセットキャラクタ] - エンコードする文字ストリング。指定された文字のみが、フォント内で利用可能となります。このパラメータが省略されると、すべての文字がエンコードされます。エンコードされている特定の文字以外が使われると、その文字は空白として表示されます。
- [新規フォント名] - 新しく記録したフォントキャストメンバの名前として使用するストリング。

このコマンドは、[フォント] パラメータで名づけられたフォントを使って Shock フォントを [キャストメンバ] 内に作成します。このコマンドで返される値により、動作が成功したかどうかがわかります。0 が表示されれば、成功です。

例文

次のステートメントは、2 つのパラメータ、キャストメンバおよび記録用フォントを使って、単純な Shocked フォントを作成します。

```
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")
```

次のステートメントは、生成するビットマップサイズおよびフォントデータを作成する文字を指定します。

```
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First Last Name")
```

注意：recordFont はフォントデータを直接使用するというよりも、再合成されたものなのでプログラムの複製ではありません。ただし、ムービーを配布する際は、このような形態での使用がオリジナルフォントの使用許諾契約でカバーされているかどうかを確認することを強くお勧めします。

関連事項

new()

rect (camera)

使用法

```
sprite(whichSprite).camera(whichCamera).rect
```

解説

3D カメラプロパティ。このプロパティを使用すると、カメラのサイズおよび位置を制御する矩形を取得または設定できます。この矩形は、実際のカメラのファインダー越しに見える矩形のようになっています。

rect プロパティのデフォルト値は rect(0,0,1,1) であるため、すべてのカメラは、設定を変えない限り見えません。ただし、sprite.camera(1) がレンダリングされると、rect プロパティが rect(0, 0, sprite(スプライト).width, sprite(スプライト).height) にリセットされるため、カメラが画面全体を満たすことになります。すべてのカメラの rect プロパティの座標は、スプライトの左上隅を基準に設定されます。

カメラが 1 より大きい場合は、スプライトにスケールを適用する際に rect にはスケールが適用されません。したがって、必要に応じてスクリプトで管理する必要があります。

カメラが 1 より大きい場合は、rect.top プロパティと rect.left プロパティを、sprite.camera(1) の rect.top と rect.left に設定されている値以上の値に設定する必要があります。

例文

次のステートメントは、ビットマップキャストメンバ "1" の座標値を表示します。

```
-- Lingo syntax
put(member(1).rect)

// JavaScript syntax
put(member(1).rect);
```

関連事項

cameraPosition, cameraRotation

rect （画像）

使用法

```
-- Lingo syntax
imageObjRef.rect

// JavaScript syntax
imageObjRef.rect;
```

解説

画像のプロパティ。設定された画像のサイズを記述した矩形を返します。読み取り専用。

画像の左上隅に対して、矩形の座標値が返されます。したがって、矩形の左と上の値は常に 0 になり、右と下の値はキャストメンバの幅と高さになります。

例文

次のステートメントは、メッセージウィンドウにキャストメンバ " 日の出 " の矩形 300 x 400 ピクセルを返します。

```
-- Lingo syntax
member("Sunrise").image.rect -- rect(0, 0, 300, 400)

// JavaScript syntax
member("Sunrise").image.rect; // rect(0, 0, 300, 400)
```

次の Lingo は、最初のキャストメンバ 50 の中から、ビットマップ式のキャストメンバのそれぞれの矩形および名前を表示します。

```
-- Lingo syntax
on showAllRects
    repeat with x = 1 to 50
    if member(x).type = #bitmap then
        put member(x).image.rect && "-" && member(x).name
    end if
    end repeat
end

// JavaScript syntax
function showAllRects() {
    var x = 1;
    while(x < 51) {
        var tp = member(x).type;
        if (tp == "bitmap") {
            trace(member(x).image.rect + " - " + member(x).name);
            i++;
        }
    }
}
```

関連事項

height, image(), width

rect （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.rect

// JavaScript syntax
memberObjRef.rect;
```

解説

ビットマップ、シェイプ、ムービー、デジタルビデオなどのグラフィックキャストメンバの矩形の上下左右の座標値を示すメンバプロパティです。すべてのキャストメンバに対して読み取り専用で、フィールドキャストメンバについてのみ、読み取りと書き込みが可能です。

ビットマップの rect プロパティは、ペイントウィンドウのイーゼルの左上隅からではなく、ビットマップの左上隅から測定されます。

Xtra 拡張キャストメンバの rect プロパティは、左上隅の座標値が（0,0）の矩形になります。

例文

次のステートメントは、ビットマップキャストメンバ "20" の座標値を表示します。

```
-- Lingo syntax
put(member(20).rect)

// JavaScript syntax
put(member(20).rect);
```

次のステートメントは、ビットマップキャストメンバ " 旗 " の座標値を設定します。

```
-- Lingo syntax
member("Banner").rect = rect(100, 150, 300, 400)

// JavaScript syntax
member("Banner").rect = rect(100, 150, 300, 400);
```

関連事項

メンバ

rect （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.rect

// JavaScript syntax
spriteObjRef.rect;
```

解説

ビットマップ、シェイプ、ムービー、デジタルビデオなどのグラフィックスプライトの矩形の上下左右の座標値を示すスプライトプロパティです。座標値は矩形で表されます。読み取り / 書き込み可能。

例文

次のステートメントは、ビットマップスプライト 20 の座標値を表示します。

```
-- Lingo syntax
put(sprite(20).rect)

// JavaScript syntax
put(sprite(20).rect);
```

関連事項

rect(), スプライト

rect （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.rect

// JavaScript syntax
windowObjRef.rect;
```

解説

ウィンドウの上下左右の座標値を指定するウィンドウプロパティです。読み取り / 書き込み可能。

指定された矩形のサイズが、ムービーが作成されたステージのサイズよりも小さい場合は、ムービーのサイズが変更されるのではなく、ウィンドウに合うようクロップされます。

ウィンドウで再生中のムービーにパンまたはスケールを適用するときは、ウィンドウの drawRect プロパティまたは sourceRect プロパティを設定します。

例文

次のステートメントは、"Control_panel" という名前のウィンドウの座標値を表示します。

```
-- Lingo syntax
put(window("Control_panel").rect)

// JavaScript syntax
put(window("Control_panel").rect);
```

関連事項

drawRect, sourceRect, ウィンドウ

ref

使用法

```
chunkExpression.ref
```

解説

テキストチャンク式プロパティです。テキストキャストメンバ内のチャンク式を簡単に参照することができます。

例文

参照がない場合は、ステートメントは次のように記述する必要があります。

```
member(whichTextMember).line[whichLine].word[firstWord..lastWord].font = "Palatino"
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontSize = 36
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontStyle = [#bold]
```

しかし、ref プロパティを使用すると、次の方法で同じチャンクを参照できます。

```
myRef = member(whichTextMember).line[whichLine].word[firstWord..lastWord].ref
```

変数 myRef が、チャンク式全体の短縮形となります。次のように記述することもできます。

```
put myRef.font
-- "Palatino"
```

また、次のようにチャンクのプロパティを設定することもできます。

```
myRef.fontSize = 18
myRef.fontStyle = [#italic]
```

次の参照のテキストプロパティを使って、その参照により参照されているストリングへアクセスすることもできます。

```
put myRef.text
```

次の結果は、ストリングについての情報ではなく、実際のストリングデータとなります。

reflectionMap

使用法

```
member(whichCastmember).shader(whichShader).reflectionMap
```

解説

3D シェーダプロパティ。このプロパティを使用すると、モデルのサーフェス上に反射を作成するときに使用するテクスチャを取得および設定できます。このテクスチャは、シェーダの 3 番目のテクスチャレイヤーに適用されます。toon モディファイアがモデルリソースに適用される場合は、このプロパティは無視されます。

このヘルパープロパティによって、反射マッピングの一般的な使用をセットアップする簡単なインターフェイスが提供されます。次のプロパティを設定することにより、同じ効果が得られます。

```
shader.textureModeList[3] = #reflection
shader.blendFunctionList[3] = #blend
shader.blendSourceList[3] = #constant
shader.blendConstantList[3] = 50.0
```

テストを実行すると、このプロパティによって、モデルの 3 番目のテクスチャレイヤーに関連付けられたテクスチャを取得できます。デフォルトは void です。

例文

次のステートメントでは、" 球体と 01" いうモデルのサーフェスにデフォルトのテクスチャを反映する表示が設定されます。

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").shader.reflectionMap=member("3Dobjects").texture[1]

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).shader.reflectionMap=member("3Dobjects").getPropRef("texture",1
);
```

関連事項

textureModeList, blendFunctionList, blendConstantList

reflectivity

使用法

```
member(whichCastmember).reflectivity
```

解説

3D シェーダプロパティ。このプロパティを使用すると、参照先のメンバのデフォルトシェーダの輝きを取得または設定できます。値は 100.00 ～ 0.0 の浮動小数点値で、デフォルトシェーダを使用するモデルのサーフェスに反射するライトのパーセンテージを表します。デフォルト値は 0.0 です。

例文

次のステートメントは、" シーン " というキャストメンバのデフォルトシェーダの輝きを 50% に設定します。

```
-- Lingo syntax
member("Scene").reflectivity = 50

// JavaScript syntax
member("Scene").reflectivity = 50;
```

region

使用法

```
member(whichCastmember).modelResource(whichModelResource).emitter.region
modelResourceObjectReference.emitter.region
```

解説

3D エミッタプロパティ。#particle というタイプのモデルリソースを指定すると、そのリソースのパーティクルエミッタの region プロパティを取得および設定できます。

region プロパティは、パーティクルの放出元となる位置を定義します。この値が 1 つのベクトルの場合は、そのベクトルを使用して、パーティクルが放出される 3D ワールドのポイントが定義されます。

この値が 2 つのベクトルのリストである場合、それらのベクトルを使用して、パーティクルが放出される線セグメントの終点が定義されます。

この値が 4 つのベクトルのリストである場合、それらのベクトルを使用して、パーティクルが放出される四辺形の頂点が定義されます。

このプロパティのデフォルト値は vector(0,0,0) です。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" のパーティクルの放出元となる矩形の 4 隅を指定します。

```
member("Fires").modelResource("ThermoSystem").emitter.region = [vector(20,90,100), vector(30,90,100),
vector(30,100,100), vector(20,100,100)]
```

関連事項

emitter

regPoint

使用法

```
-- Lingo syntax
memberObjRef.regPoint

// JavaScript syntax
memberObjRef.regPoint;
```

解説

キャストメンバのセンターポイントを示すメンバプロパティです。読み取り / 書き込み可能。

センターポイントは、point(**horizontal, vertical**) 形式で、ポイントの水平垂直座標値としてリストされます。サウンドなどの視覚的でないメンバには、regPoint プロパティはあまり役に立たない場合があります。

regPoint プロパティを使用して、フィルムループ内の個々のグラフィックをアニメートするには、フィルムループの位置をステージ上のその他のオブジェクトと相対的に変更します。

regPoint を使って、スプライト上で使われているマスクの位置を調整することもできます。

Flash ムービーキャストメンバが最初にキャストライブラリに挿入されると、センターポイントが中央となり、centerRegPoint プロパティは TRUE に設定されます。regPoint プロパティを連続使用してセンターポイントの位置を再設定すると、centerRegPoint プロパティは自動的に FALSE に設定されます。

例文

次のステートメントは、" 机 " ビットマップキャストメンバのセンターポイントの位置をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Desk").regPoint)

// JavaScript syntax
put(member("Desk").regPoint);
```

次のステートメントは、" 机 " ビットマップキャストメンバのセンターポイントをリスト内の座標値の位置に変更します。

```
-- Lingo syntax
member("Desk").regPoint = point(300, 400)

// JavaScript syntax
member("Desk").regPoint = point(300, 400);
```

関連事項

メンバ , スプライト

regPoint (3D)

使用法

```
sprite(whichSprite).camera.backdrop[backdropIndex].regPoint
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].regPoint
```

解説

3D 背景およびオーバーレイのプロパティ。このプロパティを使用すると、背景またはオーバーレイのセンターポイントを取得または設定できます。センターポイントは、3D 空間の背景またはオーバーレイの中心になる **x**、**y**、および **z** 座標を表します。このプロパティのデフォルト値は point(0,0) です。

例文

次のステートメントは、スプライト 13 のカメラの、最初の背景のセンターポイントを変更します。背景のセンターポイントは、背景の左上隅から計測した point (50, 0) になります。

```
sprite(13).camera.backdrop[1].regPoint = point(50, 0)
```

関連事項

loc (backdrop and overlay)

regPointVertex

使用法

```
-- Lingo syntax
memberObjRef.regPointVertex

// JavaScript syntax
memberObjRef.regPointVertex;
```

解説

キャストメンバプロパティ。ベクトルキャストメンバの頂点が、そのキャストメンバのセンターポイントとして使用されているかどうかを示します。値が 0 の場合は、centerRegPoint プロパティと regPoint プロパティを使用して、センターポイントは通常どおりに設定されます。値が 0 以外の場合は、このプロパティは、登録ポイントとして使用する頂点の vertextList 内の位置を示します。centerRegPoint は FALSE に設定され、regPoint はその頂点の位置に設定されます。

例文

次のステートメントは、ベクトルシェイプキャストメンバ Squiggle のセンターポイントを 3 番目の頂点の位置に設定します。

```
-- Lingo syntax
member("squiggle").regPointVertex=3

// JavaScript syntax
member("squiggle").regPointVertex=3;
```

関連事項

centerRegPoint, regPoint

renderer

使用法

```
getRendererServices().renderer
```

解説

3D プロパティ。このプロパティを使用すると、ムービーが現在使用しているレンダラを取得または設定できます。このプロパティの値の範囲は、レンダラサービスオブジェクトの rendererDeviceList プロパティによって返される使用可能なレンダラのリストによって決められます。

Shockwave Player のユーザは、Shockwave Player の 3D レンダラコンテキストメニューによってレンダラを指定できます。ユーザが「コンテンツの設定に従う」を選択した場合は、renderer プロパティまたは preferred3DRenderer プロパティで指定されているレンダラでムービーが描画されます（そのレンダラがユーザのシステムで使用できるとき）。それ以外の場合は、ユーザが選択したレンダラが使用されます。

このプロパティのデフォルト値は、preferred3DRenderer プロパティによって決まります。

このプロパティは、ムービープロパティ the active3dRenderer が返す値と同じ値を返します。

例文

次のステートメントは、ユーザのシステムで現在使用されているレンダラを示します。

```
-- Lingo syntax
put getRendererServices().renderer

// JavaScript syntax
put( getRendererServices().renderer);
```

関連事項

getRendererServices(), preferred3dRenderer, rendererDeviceList, active3dRenderer

rendererDeviceList

使用法

```
getRendererServices().rendererDeviceList
```

解説

3D レンダラプロパティ。クライアントマシンで使用できるレンダラを示すシンボルのリストを返します。このリストの内容によって、renderer プロパティと preferred3DRenderer プロパティに指定できる値の範囲が決まります。このプロパティは、値を調べられますが、設定はできません。

このプロパティのリストで使用される値は次のとおりです。

- #openGL。ハードウェアアクセラレーターの openGL ドライバを指定します。Mac と Windows の両方のプラットフォームで使用できます。
- #directX9。ハードウェアアクセラレーターの DirectX 9 ドライバを指定します。Windows プラットフォームだけで使用できます。#auto は、レンダラを DirectX 9 に設定します。Mac-Intel では、#OpenGL レンダラのみを使用できます。
- #directX7_0。ハードウェアアクセラレーターの DirectX 7 ドライバを指定します。Windows プラットフォームだけで使用できます。
- #directX5_2。ハードウェアアクセラレーターの DirectX 5.2 ドライバを指定します。Windows プラットフォームだけで使用できます。
- #software。Director の内蔵ソフトウェアレンダラを指定します。このレンダラは、Mac と Windows の両方のプラットフォームで機能します。

例文

次のステートメントは、現在のシステムで使用できるレンダラを示します。

```
-- Lingo syntax
put getRendererServices().rendererDeviceList

// JavaScript syntax
put( getRendererServices().rendererDeviceList);
```

関連事項

getRendererServices(), renderer, preferred3dRenderer, active3dRenderer

renderFormat

使用法

```
member(whichCastmember).texture(whichTexture).renderFormat
member(whichCastmember).texture[index].renderFormat
member(whichCastmember).shader(whichShader).texture.renderFormat
member(whichCastmember).model(whichModel).shader.texture.renderFormat
member(whichCastmember).model(whichModel).shader.textureList[index].renderFormat
member(whichCastmember).model(whichModel).shaderList[index].texture(whichTexture).renderFormat
member(whichCastmember).model(whichModel).shaderList[index].textureList[index].renderFormat
```

解説

3D プロパティ。このプロパティを使用すると、特定のテクスチャの textureRenderFormat を取得または設定できます。以下のいずれかの値を指定できます。

#default。getRendererServices().textureRenderFormat によって返された値を使用します。

```
#rgba8888
#rgba8880
#rgba5650
#rgba5550
#rgba5551
#rgba4444
```

これらの値の詳細については、「textureRenderFormat」を参照してください。

個々のテクスチャに対してこのプロパティを設定すると、textureRenderFormat を使って設定したグローバル設定が変更されます。

renderFormat プロパティは、指定したテクスチャをレンダリングする際にレンダリング方法で使用するピクセルフォーマットを決定します。各ピクセルフォーマットには複数の数字があり、それぞれ、赤、緑、青、およびアルファに使用される色深度を示します。選択した値により、カラーの忠実度の精度（オプションのアルファチャネルの精度を含む）とビデオカードで使用されるメモリ量が決まります。カラーの忠実度を向上させるか、またはビデオカード上により多くのテクスチャを読み込ませるかといういずれかの目的に合わせて値を選択できます。メモリ容量が同じであれば、32 ビットテクスチャの約 2 倍の数の 16 ビットテクスチャを読み込ませることができます。

例文

次のステートメントは、"TexPic" というテスクチャの renderFormat プロパティを #rgba4444 に設定します。テクスチャの赤、青、緑、アルファの各成分は、それぞれ 4 ビットの情報を使って描画されます。

```
-- Lingo syntax
member("3Dobjects").texture[1].renderFormat = #rgba4444
put(member("3Dobjects").texture[1].renderFormat )
```

関連事項

textureRenderFormat, getHardwareInfo()

renderStyle

使用法

```
member(whichCastmember).shader(whichShader).renderStyle
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、シェーダの renderStyle を取得または設定できます。renderStyle は、シェーダが適用されるモデルリソースのジオメトリによって決まります。このプロパティの値は以下のとおりです。

#fill。モデルリソースのサーフェス領域全体にシェーダが描画されるように指定します。

#wire。モデルリソースのフェイスのエッジだけにシェーダが描画されるように指定します。

#point。モデルリソースの頂点だけにシェーダが描画されるように指定します。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

次のステートメントは、最初のシェーダがモデルリソースの頂点だけにレンダリングされるように指定します。

```
-- Lingo syntax
member("3Dobjects").shader[1].renderStyle = #point

// JavaScript syntax
    member("3Dobjects").getPropRef("shader",1).renderStyle = symbol("point");
```

resizable

使用法

```
-- Lingo syntax
windowObjRef.resizable

// JavaScript syntax
windowObjRef.resizable;
```

解説

ウィンドウプロパティ。ウィンドウがサイズ変更可能であるか（TRUE、デフォルト）、サイズ変更可能でないか（FALSE）を示します。

読み取り / 書き込み可能。

例文

次のステートメントは、ウィンドウがサイズ変更可能な場合 "Empire" という名前のウィンドウを最大化します。

```
-- Lingo syntax
if (window("Empire").resizable = TRUE) then
    window("Empire").maximize()
end if

// JavaScript syntax
if (window("Empire").resizable == true) {
    window("Empire").maximize();
}
```

関連事項

ウィンドウ

resolution (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).resolution
```

解説

3D プロパティ。このプロパティを使用すると、#sphere または #cylinder というタイプのモデルリソースの resolution プロパティを取得または設定できます。

resolution プロパティでは、モデルリソースのジオメトリの作成に使用するポリゴンの数を指定できます。値が大きくなるほど、数多くのポリゴンが作成され、その結果、サーフェスがより滑らかになります。このプロパティのデフォルト値は 20 です。

例文

次のステートメントは、" 球体 01" というモデルリソースの resolution プロパティを 10.0 に設定します。

```
member("3D World").modelResource("sphere01").resolution = 10.0
```

resolution (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.resolution

// JavaScript syntax
dvdObjRef.resolution;
```

解説

DVD プロパティ。x 軸（幅）および y 軸（高さ）のソース解像度を含むプロパティリストを返します。読み取り専用。

例文

次のステートメントは、解像度のサンプルプロパティリストを返します。

```
-- Lingo syntax
trace(member(1).resolution) -- [#width: 720, #height: 480]

// JavaScript syntax
trace(member(1).resolution); // ["width": 720, "height": 480]
```

関連事項

DVD

resolve

使用法

```
member(whichCastmember).model(whichModel).collision.resolve
```

解説

3D 衝突プロパティ。このプロパティを使用すると、2 つのモデルが衝突した際に衝突が解決されるかどうかを取得または設定できます。衝突に含まれる両方のモデルでこのプロパティが TRUE に設定されていると、いずれのモデルも衝突したポイントで停止します。一方のモデルの resolve プロパティだけが TRUE に設定されている場合は、そのモデルだけが停止します。resolve プロパティが設定されていないかまたは FALSE に設定されているもう一方のモデルは、そのまま動き続けます。このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" ボックス " というモデルに割り当てられている衝突モディファイアの resolve プロパティを TRUE に設定します。" ボックス " が別のモデルと衝突し、相手のモデルに #collision モディファイアが割り当てられていた場合、" ボックス " は停止します。

```
member("3d world").model("Box").collision.resolve = TRUE
```

関連事項

collisionData, collisionNormal, modelA, modelB, pointOfContact

resource

使用法

```
member(whichCastmember).model(whichModel).resource
```

解説

3D プロパティ。このプロパティを使用すると、参照されたモデルリソースのジオメトリを定義する resource プロパティを取得または設定できます。また、このプロパティを使用して、参照先のモデルのリソースオブジェクトおよびそれに関連付けられているプロパティにアクセスすることもできます。

例文

次のステートメントは、2 番目のモデルが使用するモデルリソースの radius プロパティを表示します。

```
-- Lingo syntax
put member("3Dobjects").model[2].resource.radius

// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).resource.radius);
```

right

使用法

```
-- Lingo syntax
spriteObjRef.right

// JavaScript syntax
spriteObjRef.right;
```

解説

スプライトの右端からステージの左端までの距離をピクセル数で示すスプライトプロパティです。読み取り / 書き込み可能。

スプライト座標値は、ステージの左上隅から測った距離で表されます。

例文

次のステートメントは、スプライトの右端の距離を返します。

```
-- Lingo syntax
put(sprite(6).right)

// JavaScript syntax
put(sprite(6).right);
```

関連事項

bottom, height, left, locH, locV, スプライト , top, width

right (3D)

使用法

```
member(whichCastmember).modelResource
(whichModelResource).right
modelResourceObjectReference.right
```

解説

3D プロパティ。このプロパティを使用すると、#box というタイプのモデルリソースの right プロパティを取得または設定できます。

right プロパティを使用すると、ボックスの右側面を閉じた状態にするか (TRUE)、または開いた状態にするか (FALSE) を指定できます。デフォルト値は TRUE です。

例文

次のステートメントは、モデルリソース " 木箱 " の right プロパティを TRUE に設定します。これにより、このボックスの右面は閉じた状態になります。

```
member("3D World").modelResource("crate").right = TRUE
```

関連事項

bottom (3D), left (3D), top (3D)

rightIndent

使用法

```
chunkExpression.rightIndent
```

解説

次のステートメントは、テキストキャストメンバ "myText" を 10 ピクセル右インデントします。

```
-- Lingo syntax
member("myText").rightIndent=10

// JavaScript syntax
member("myText").rightIndent=10;
```

関連事項

firstIndent, leftIndent

rightMouseDown

使用法

```
-- Lingo syntax
_mouse.rightMouseDown

// JavaScript syntax
_mouse.rightMouseDown;
```

解説

マウスプロパティ。Windows ではマウスの右ボタンが現在押されているかどうか、Mac ではマウスボタンと Control キーが押されているかどうかを示します。押されている場合は（TRUE）を返し、押されていないときは（FALSE）を返します。読み取り専用。

Mac では、emulateMultiButtonMouse プロパティが TRUE であるときのみ rightMouseDown は TRUE を返します。

例文

次のステートメントは、マウスの右ボタンが押されたかどうかをチェックし、押されている場合は、サウンドチャネル 2 の "Oops" サウンドを再生します。

```
-- Lingo syntax
if (_mouse.rightMouseDown) then
    sound(2).play(member("Oops"))
end if

// JavaScript syntax
if (_mouse.rightMouseDown) {
    sound(2).play(member("Oops"));
}
```

関連事項

emulateMultibuttonMouse, マウス

rightMouseUp

使用法

```
-- Lingo syntax
_mouse.rightMouseUp

// JavaScript syntax
_mouse.rightMouseUp;
```

解説

マウスプロパティ。Windows ではマウスの右ボタンが現在離されているかどうかを示し、Mac ではマウスボタンと Control キーが離されているかどうかを示します。離されている場合は（TRUE）を返し、離されていないときは（FALSE）を返します。読み取り専用。

Mac では、emulateMultiButtonMouse プロパティが TRUE であるときのみ rightMouseUp は TRUE を返します。

例文

次のステートメントは、Windows のマウスの右ボタンが離されたかどうかをチェックし、離されているときは、"Click Me" サウンドを再生します。

```
-- Lingo syntax
if (_mouse.rightMouseUp) then
    sound(2).play(member("Click Me"))
end if

// JavaScript syntax
if (_mouse.rightMouseUp) {
    sound(2).play(member("Click Me"));
}
```

関連事項

emulateMultibuttonMouse, マウス

romanLingo

使用法

```
the romanLingo
```

解説

Lingo が 1 バイトインタープリタと 2 バイトインタープリタのどちらを使うかを指定するためのシステムプロパティです。romanLingo が TRUE のときは、Lingo は 1 バイトインタープリタを使い、FALSE のときは、2 バイトインタープリタを使います。

Lingo インタープリタの処理速度は 1 バイト文字セット用を使うほうが高速です。Mac システムソフトウェアの中には、漢字 Talk のように 2 バイト文字を使用するものもあります。英語のシステムソフトウェアは 1 バイト文字セットを使っています。通常、romanLingo は、Director 起動時に設定され、その設定は使用している Macintosh システムソフトウェアのローカルバージョンにより決まります。

非ローマンスクリプトのシステムを使用していても、スクリプト内で 2 バイト文字を使用しない場合は、このプロパティを TRUE に設定すると、Lingo スクリプトの実行速度を速めることができます。

例文

次のステートメントは、romanLingo を TRUE に設定し、Lingo が 1 バイトの文字セットを使うようにします。

```
-- Lingo
set the romanLingo to TRUE
```

関連事項

inlineImeEnabled

rootLock

使用法

```
member(whichCastmember).model(whichModel).keyframePlayer.rootLock
member(whichCastmember).model(whichModel).bonesPlayer.rootLock
```

解説

#keyframePlayer モディファイアおよび #bonesPlayer モディファイアの 3D プロパティ。モーションの平行移動コンポーネントが使用されるか (FALSE)、無視されるか (TRUE) を示します。

このプロパティのデフォルト値は FALSE です。

例文

次のステートメントは、最初のモデルであるエイリアン 3 というモデルに対して、モーションを実行する間そのまま開始位置にとどまるように指定します。これにより、このモデルは、モーションの開始位置で足踏みを続けます。

```
-- Lingo
member("newalien").model("Alien3").keyframePlayer.rootLock = 1

// Javascript
member("newalien").getPropRef("model",1).keyframePlayer.rootLock=1
```

rootNode

使用法

```
member(whichCastmember).camera(whichCamera).rootNode
sprite(whichSprite).camera.rootNode
```

解説

3D プロパティ。このプロパティを使用すると、スプライト内に表示されるオブジェクトを取得または設定できます。カメラを最初に作成するときには、ワールド内のすべてのノードが表示されます。rootNode プロパティでは、これとは異なるデフォルト表示を作成して、特定のノードおよびその子のみを表示するようにできます。

例えば、ライト C はモデル A の子である場合、rootNode プロパティを camera("defaultView").rootNode=model(A) と設定すると、スプライトにはライト C によって照らされたモデル A のみが表示されます。デフォルトの group("world") は、すべてのノードが使用されることを意味します。

例文

次のステートメントは、スプライト 5 のカメラの rootNode プロパティをモデル " 冥王星 " に設定します。スプライト 5 には、" 冥王星 " とその子だけが表示されます。

```
-- Lingo
sprite(5).camera.rootNode = member("Scene").model("Pluto")

// Javascript
sprite(5).getPropRef("camera",1).rootNode=member("Scene").getPropRef("model",1);
```

rotation

使用法

```
-- Lingo syntax
spriteObjRef.rotation

// JavaScript syntax
spriteObjRef.rotation;
```

解説

スプライトのプロパティ。スプライトの境界矩形内の QuickTime ムービー、アニメーション GIF、Flash ムービー、またはビットマップスプライトの回転を管理します。QuickTime の場合には、矩形またはスプライトのコントローラを回転しません。読み取り / 書き込み可能。

実際、スプライトの境界矩形は、Flash または QuickTime ムービーを表示できるウィンドウのように動作します。ビットマップとアニメーション GIF の境界矩形は、回転画像に応じて変化します。

Flash ムービーのスコアの回転は、obeyScoreRotation が TRUE に設定されている場合のみできます。

Flash ムービーは、その originMode プロパティで指定されたように元の点を軸に回転します。QuickTime ムービーは、スプライトの境界矩形の中心を軸に回転します。ビットマップは、画像の中点を軸に回転します。

QuickTime メディアの場合、スプライトの crop プロパティが TRUE のときは、スプライトを回転すると、表示可能領域から画像の一部が頻繁に外に移動します。スプライトの crop プロパティが FALSE のときは、画像は境界矩形内に収まるように大きさが調整されます。調整によって、画像にゆがみが生じる場合があります。

回転度を浮動小数点で指定します。

スコアは画像を回転するための情報を +21,474,836.47° ～ -21,474,836.48° まで保持できます。これにより、どちらの方向にも 59,652 回フルに回転できます。

回転が限界に達したとき、つまり 59,652 回転を過ぎると、回転は +116.47° または -116.48° にリセットされ、0.00° にはリセットされません。これは、+21,474,836.47° が +116.47° に等しく、-21,474,836.48° が -116.48°（または +243.12°）に等しいからです。リセット状態を避けるには、スクリプトを使用して、連続回転の角度を ±360° に強制します。

このプロパティのデフォルト値は 0 です。

例文

次のビヘイビアーは、再生ヘッドが進むたびに、スプライトを 2° ずつ連続して回転させます。角度の上限は 360° です。

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).rotation = integer(sprite(spriteNum).rotation + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
        sprite(this.spriteNum).rotation = parseInt(sprite(this.spriteNum).rotation + 2) % 360;
}
```

次のフレームスクリプトは、チャネル 5 の QuickTime スプライトを 16° ずつの増分で 360° まで回転する間、再生ヘッドを現在のフレーム内でループさせます。スプライトが 360° 回転すると、再生ヘッドは次のフレームに移ります。

```
-- Lingo syntax
on rotateMovie(whichSprite)
    repeat with i = 1 to 36
        sprite(whichSprite).rotation = i * 10
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function rotateMovie(whichSprite) {
    for (var i = 1; i <= 36; i++) {
        sprite(whichSprite).rotation = i * 10;
        _movie.updateStage();
    }
}
```

関連事項

obeyScoreRotation, originMode, スプライト

rotation (backdrop and overlay)

使用法

```
sprite(whichSprite).camera.backdrop[backdropIndex].rotation
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].rotation
sprite(whichSprite).camera.overlay[overlayIndex].rotation
member(whichCastmember).camera[cameraIndex].overlay
[overlayIndex].rotation
```

解説

3D プロパティ。このプロパティを使用すると、デフォルトのカメラ方向への背景またはオーバーレイの回転を取得または設定できます。このプロパティのデフォルト値は 0.0 です。

例文

次のステートメントは、センターポイントを中心に背景を 60° 回転します。

```
sprite(4).camera.backdrop[1].rotation = 60.0
```

関連事項

bevelDepth, transform （プロパティ）

rotation (engraver shader)

使用法

```
member(whichCastmember).shader(whichShader).rotation
member(whichCastmember).model(whichModel).shader.rotation
member(whichCastmember).model(whichModel).shaderList[index].rotation
```

解説

3D エングレーバーシェーダプロパティ。このプロパティを使用すると、エングレーブされた線の 2D 回転オフセットを示す度単位の角度を浮動小数点数で取得または設定できます。このプロパティのデフォルト値は 0.0 です。

例文

次のステートメントは、モデル "gbCyl3" のエングレーバーシェーダの描画に使用する線を現在よりも 1　回転します。

```
member("scene").model("gbCyl3").shader.rotation = \ member("scene").model("gbCyl3").shader.rotation + 1
```

関連事項

transform （プロパティ）

rotation （トランスフォーム）

使用法

```
member(whichCastmember).node(whichNode).transform.rotation
member(whichCastmember).node(whichNode).getWorldTransform().rotation
transform.rotation
```

解説

3D プロパティ。このプロパティを使用すると、トランスフォームの回転コンポーネントを取得または設定できます。トランスフォームは、特定の基準に基づくスケール、位置、および回転を定義します。このプロパティのデフォルト値は vector(0,0,0) です。

ノードは、カメラ、グループ、ライト、またはモデルオブジェクトのいずれかです。ノードのトランスフォームの rotation を設定すると、トランスフォームの基準に基づくオブジェクトの回転を定義できます。getWorldTransform().rotation を使用して、オブジェクトのワールド基準のトランスフォームの rotation プロパティを設定すると、ワールドの原点を基準にしたオブジェクトの回転を定義できます。transform.rotation を使用して、オブジェクトの親基準のトランスフォームの rotation プロパティを設定すると、親ノードを基準にしたオブジェクトの回転を定義できます。

トランスフォームの向きを変更する場合は、このプロパティを設定する代わりに、rotate メソッドと prerotate メソッドを使用することをお勧めします。

例文

次のステートメントは、メンバの最初のカメラの親を基準にした回転を vector(0,0,0) に設定します。

```
member("Space").camera[1].transform.rotation = vector(0, 0, 0)
```

次の例は、まず、" 月 " というモデルの親を基準にした回転を表示します。次に、rotate コマンドを使ってモデルの向きを調整します。最後に、その結果を、ワールドを基準にしたモデルの回転として表示します。

```
put member("SolarSys").model("Moon").transform.rotation
-- vector( 0.0000, 0.0000, 45.0000)
member("SolarSys").model("Moon").rotate(15,15,15)
put member("SolarSys").model("Moon").getWorldTransform().rotation
--vector( 51.3810, 16.5191, 65.8771 )
```

関連事項

getWorldTransform(), preRotate, rotate, transform （プロパティ）, position （トランスフォーム）, scale （トランスフォーム）

rotationReset

使用法

```
member(whichCastmember).model(whichModel).bonesPlayer.rotationReset
member(whichCastmember).model(whichModel).keyframePlayer.rotationReset
```

解説

keyframePlayer モディファイアおよび bonesPlayer モディファイアの 3D プロパティ。回転の変化が維持される軸を示します。この軸を中心にした回転は、モーションが終了してから次のモーションが開始されるまでの間、またはループするモーションの繰り返しの間、維持されます。

このプロパティで使用できる値は、#none、#x、#y、#z、#xy、#yz、#xz、および #all です。デフォルト値は #all です。

例文

次のステートメントは、" モンスター " というモデルの rotationReset プロパティを **z** 軸に設定します。これによって、現在再生中のモーションまたはループの終了時に、このモデルは自身の **z** 軸を中心にした回転を維持します。

```
member("NewAlien").model("Monster").bonesPlayer.rotationReset = #z
```

関連事項

positionReset, bonesPlayer （モディファイア）

RTF

使用法

```
-- Lingo syntax
memberObjRef.RTF

// JavaScript syntax
memberObjRef.RTF;
```

解説

キャストメンバプロパティ。このプロパティを使用すると、リッチテキスト形式のテキストを含むテキストキャストメンバ内で、テキストのレイアウトを管理するテキストおよびタグにアクセスできます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、テキストキャストメンバ "Resume" に埋め込まれている RTF 形式の情報をメッセージウィンドウに表示します。

```
--Lingo syntax
put(member("Resume").RTF)

// JavaScript syntax
trace(member("Resume").RTF);
```

関連事項

HTML, importFileInto()

safePlayer

使用法

```
-- Lingo syntax
_player.safePlayer

// JavaScript syntax
_player.safePlayer;
```

解説

Director の安全機能がオンになっているかどうかを管理するプレーヤのプロパティです。読み取り / 書き込み可能。

Shockwave コンテンツのあるムービーで、このプロパティを調べられますが、変更はできません。Shockwave Player では、このプロパティは常に TRUE です。

オーサリング環境やプロジェクタでデフォルト値は FALSE です。このプロパティは返すことができますが、プロパティが TRUE に設定されている場合にかぎります。一旦、TRUE に設定すると、Director またはプロジェクタを再起動しないと FALSE へ戻すことはできません。

safePlayer が TRUE のときは、次のような安全機能がオンになります。

- 安全な Xtra のみが使用できます。
- safePlayer プロパティはリセットできません。

- pasteClipBoardInto() メソッドを使って、クリップボードの内容をペーストすると、警告ダイアログが表示され、ユーザがキャンセルできるようになります。
- スクリプトを使用してムービーまたはキャストを保存できません。
- printFrom() メソッドを使用して印刷できません。
- open() メソッドを使用してアプリケーションを開くことができません。
- restart() または shutDown() メソッドを使用してアプリケーションまたはエンドユーザのコンピュータを停止することはできません。
- "dswmedia" フォルダの外側にあるファイルを開くことはできません。
- ローカルのファイル名を探すことはできません。
- getNetText()、postNetText() を使用するなどして、同じドメインではない URL にアクセスすると、セキュリティダイアログが表示されます。

例文

次のステートメントは、オーサリング環境で、safePlayer プロパティに設定されている値を確認します。

```
-- Lingo
put( _player.safePlayer)
--0

// Javascript
trace(_player.safePlayer)
// 0
```

関連事項

プレーヤ

sampleCount （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.sampleCount

// JavaScript syntax
soundChannelObjRef.sampleCount;
```

解説

サウンドチャネルで再生中のサウンドサンプル数を指定するサウンドチャネルプロパティです。読み取り専用。

サンプルの総数で、sampleRate およびサウンドの duration により異なります。サウンドの channelCount には依存しません。

秒 44.1 KHz サウンドには、44,100 のサンプルが含まれます。

例文

次のステートメントは、サウンドチャネル 1 で再生中のキャストメンバの名前と sampleCount をメッセージウィンドウに表示します。

```
-- Lingo syntax
put("Sound cast member" && sound(1).member.name && "contains" && sound(1).sampleCount && "samples.")

// JavaScript syntax
put("Sound cast member " + sound(1).member.name + " contains " +sound(1).sampleCount + " samples.");
```

関連事項

sampleRate（サウンドチャネル）, サウンドチャネル

sampleCount（サウンドオブジェクト）

使用法

```
soundObject.sampleCount (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトのサンプル数数を返します。

例文

```
--Lingo syntax
on mouseUp me
    put soundObjRef.sampleCount -- Displays the number of samples in the sound
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.sampleCount); // Displays the number of samples in the sound
// associated with soundobjectRef.
}
```

sampleRate (ミクサ)

使用法

```
mixer.sampleRate (Read-write)
```

解説

ミクサプロパティ。ミクサの毎秒のサンプル数を返します。sampleRate は任意の標準サンプルレートに設定できます。

例文

```
--Lingo syntax
on mouseUp me
mixerRef.sampleRate = 44100 -- Sets the sample rate of the mixer to 44100.
end

// JavaScript syntax
function mouseUp(){
mixerRef.sampleRate = 44100; // Sets the sample rate of the mixer to 44100.
}
```

関連事項

ミクサ

sampleRate （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.sampleRate

// JavaScript syntax
soundChannelObjRef.sampleRate;
```

解説

サウンドキャストメンバのサンプルレートまたは Shockwave Audio（SWA）形式にコード化された元々のファイルのサンプルレートを返すサウンドチャネルプロパティです。読み取り専用。

SWA サウンドの再生が開始されたか、または preLoadBuffer() メソッドを使用してファイルがプリロードされた後にのみ、このプロパティは有効になります。サウンドチャネルが設定されている場合は、設定されたサウンドチャネルで再生中のサウンドキャストメンバのサンプルレートが結果となります。

このプロパティの代表的な値は、8000、11025、16000、22050、および 44100 です。

サウンドチャネルのキューに複数のサウンドが登録されている場合、Director は、登録されたすべてのサウンドに対して最初のサウンドの channelCount、sampleRate、および sampleSize を適用し、再サンプリング処理します。これによって、すべてのサウンドが滑らかに再生されます。このプロパティはチャネルのサウンドキューがすべて完了した場合、または stop() メソッドを使用した場合のみリセットができます。次にキューに登録されるサウンド、または再生されるサウンドによって、新しい設定が決められます。

例文

次のステートメントは、" ポール・ローブソン " SWA ストリーミングキャストメンバで使われたファイルの元のサンプルレートを " 音質 " フィールドキャストメンバに割り当てます。

```
-- Lingo syntax
member("Sound Quality").text = string(member("Paul Robeson").sampleRate)

// JavaScript syntax
member("Sound Quality").text = member("Paul Robeson").sampleRate.toString();
```

次のステートメントは、サウンドチャネル 1 で再生されているサウンドのサンプルレートをメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(sound(1).sampleRate)

// JavaScript syntax
trace(sound(1).sampleRate);
```

関連事項

channelCount （サウンドチャネル）, sampleSize, preLoadBuffer(), サウンドチャネル , stop() （サウンドチャネル）

sampleRate （サウンドオブジェクト）

使用法

```
soundObject.sampleRate (Read-only)
```

解説

現在のサウンドオブジェクトのサンプリングレート（1 秒間のサンプル数）を返すサウンドオブジェクトプロパティです。

例文

```
-- Lingo syntax
on mouseUp me
    put soundObjRef.sampleRate -- Displays the samples per second of the
-- sound object associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.sampleRate) ; // Displays the samples per second of the
// sound object associated with soundobjectRef.
}
```

sampleSize

使用法

```
-- Lingo syntax
memberObjRef.sampleSize

// JavaScript syntax
memberObjRef.sampleSize;
```

解説

指定したキャストメンバのサンプルサイズを示すサウンドキャストメンバプロパティです。結果は通常 8 ビットか 16 ビットになります。サウンドチャネルが設定されている場合は、設定されたサウンドチャネルで再生中のサウンドメンバの値が結果となります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" 語り " サウンドキャストメンバのサンプルサイズをチェックし、その値を変数 soundSize に割り当てます。

```
-- Lingo syntax
soundSize = member("Voice Over").sampleSize

// JavaScript syntax
var soundSize = member("Voice Over").sampleSize;
```

このステートメントは、サウンドチャネル 1 で再生中のサウンドのサンプルサイズをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sound(1).sampleSize)

// JavaScript syntax
put(sound(1).sampleSize);
```

savew3d

使用法

```
--Lingo syntax
member(whichcastmember).savew3d(Absolute path (optional argument))
// JavaScript syntax
member(whichcastmember).savew3d(Absolute path (optional argument));
```

解説

プロジェクタから 3D ワールドを保存します。

このメソッドのビヘイビアーは以下のとおりです。

- 絶対パスが指定された場合、このメソッドはパス名が指定されたファイルにその時点の 3D シーンを保存します（ただし、パスが正しい場合)。これはオーサリングモードおよびプロジェクタモード両方のビヘイビアーです。
- 絶対パスが指定されていない場合、つまり呼び出しが membername.savew3d() である場合、ビヘイビアーは以下のとおりです。
  - w3d メンバが外部にリンクしており、savew3d がプロジェクタモードで呼び出された場合、外部にリンクしている w3d ファイルにその時点の 3D シーンが保存されます。このときファイルは上書きされます。
  - 3d メンバが内部メンバであり、savew3d がプロジェクタモードで呼び出された場合、このメソッド呼び出しは無視されます。

オーサリングモードでは、savew3d メソッドのビヘイビアーは saveWorld メソッドのビヘイビアーと同じです。メンバが書き込み対象としてマークされ、Director ファイルを保存した後に 3D 保存が行われます。

例文

このステートメントは、メンバ（"3dworld"）に対する変更を savedworld.w3d という名前の別ファイルに保存します。

```
-- Lingo syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d")

// JavaScript syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d");
```

saveWorld

使用法

```
--Lingo syntax
member(whichcastmember).saveWorld()
// JavaScript syntax
member(whichcastmember).saveWorld();
```

解説

ムービー保存後に既存の 3D ワールドを 3D メンバに保存します。

例文

次のステートメントは編集済みとしてメンバにマークを付けます。これにより、メンバは Director ムービーを保存したときにファイルに書き込まれます。このステートメントは、メンバを別のファイルに保存するわけではありません。

```
-- Lingo syntax
member("3dworld").saveWorld()

// JavaScript syntax
member("3dworld").saveWorld();
```

scale (3D)

使用法

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

解説

3D プロパティ。このプロパティを使用すると、参照先のカメラの表示するオーバーレイまたは背景のリスト内の特定のオーバーレイまたは背景で使用するスケール値を取得または設定できます。背景またはオーバーレイの幅と高さはスケール値に掛け合わされます。このプロパティのデフォルト値は 1.0 です。

例文

次のステートメントは、背景のサイズを 2 倍にします。

```
-- Lingo
sprite(25).camera.backdrop[1].scale = 2.0

// Javascript
sprite(25).getPropRef("camera",1).getProp("backDrop",1).scale=2.0;
```

関連事項

bevelDepth, overlay

scale (backdrop and overlay)

使用法

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

解説

3D プロパティ。このプロパティを使用すると、参照先のカメラの表示するオーバーレイまたは背景のリスト内の特定のオーバーレイまたは背景で使用するスケール値を取得または設定できます。背景またはオーバーレイの幅と高さはスケール値に掛け合わされます。このプロパティのデフォルト値は 1.0 です。

例文

次のステートメントは、背景のサイズを 2 倍にします。

```
sprite(25).camera.backdrop[1].scale = 2.0
```

関連事項

bevelDepth, overlay

scale （メンバ）

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.scale

// JavaScript syntax
memberOrSpriteObjRef.scale;
```

解説

QuickTime、ベクトルシェイプ、または Flash ムービースプライトのスケーリングを管理するキャストメンバプロパティおよびスプライトプロパティです。

QuickTime の場合、このプロパティは、スプライトの境界矩形、またはスプライトのコントローラをスケールしません。代わりに、境界矩形内にある画像のセンターポイントを中心に画像をスケールします。スケーリングは、浮動小数点で保存された 2 つの割合を含む Director リストで指定されます。

```
[xPercent, yPercent]
```

xPercent パラメータは、横のスケーリング量を指定します。**yPercent** パラメータは縦のスケーリング量を指定します。

スプライトの crop プロパティが TRUE に設定されていると、scale プロパティと共に使用してスプライトの境界矩形内でズームのように使用することができます。スプライトの crop プロパティが FALSE に設定されていると、scale プロパティは無視されます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は [1.0000,1.0000] です。

Flash ムービーまたはベクトルシェイプキャストメンバの場合、スケールは浮動小数点の値になります。ムービーはその元の点からスケールされ、その originMode プロパティで指定されます。

注意：このプロパティは、scaleMode プロパティが #autoSize に設定されていたりスプライトが正しく表示されないときは、初期設定に戻す必要があります。戻さないとスプライトが正しく表示されません。

例文

次のハンドラは、Flash ムービースプライトへの参照をパラメータとして受け取り、ムービーのスケールを 0% に縮小して見えなくします。次に、再度 5% ずつ拡大して、もう一度 100% のサイズにします。

```
-- Lingo syntax
on scaleMovie whichSprite
    sprite(whichSprite).scale = 0
    _movie.updatestage()
    repeat with i = 1 to 20
        sprite(whichSprite).scale = i * 5
        _movie.updatestage()
    end repeat
end

// JavaScript syntax
function scaleMovie(whichSprite) {
    sprite(whichSprite).scale = 0;
    _movie.updateStage();
    var i = 1;
    while (i < 21) {
        sprite(whichSprite).scale = i * 5;
        _movie.updateStage();
        i++;
    }
}
```

関連事項
scaleMode, originMode

scale （トランスフォーム）

使用法

```
member(whichCastmember).node(whichNode).transform.scale
member(whichCastmember).node(whichNode).getWorldTransform().scale
transform.scale
```

解説
3D プロパティ。このプロパティを使用すると、トランスフォームのスケールコンポーネントを取得または設定できます。トランスフォームは、特定の基準に基づくスケール、位置、および回転を定義します。scale プロパティを使用すると、3 つそれぞれの軸に沿ったトランスフォームのスケールの度合いを取得および設定できます。このプロパティのデフォルト値は vector(1.0,1.0,1.0) です。

ノードは、カメラ、グループ、ライト、またはモデルオブジェクトのいずれかです。ただし、ライトやカメラにはジオメトリがないため、このコマンドから目に見える効果は得られません。ノードのトランスフォームの scale プロパティを設定すると、トランスフォームの基準に基づく、X、Y、および Z 軸に沿ったオブジェクトのスケールを定義できます。getWorldTransform().scale を使用して、オブジェクトのワールド基準のトランスフォームの scale プロパティを取得すると、ワールドの原点を基準にしたオブジェクトのスケールが返されます。transform.scale を使用して、オブジェクトの親基準のトランスフォームの scale プロパティを設定すると、親ノードを基準にしたオブジェクトのスケールを定義できます。

例文
次のステートメントは、" 月 " というモデルのトランスフォームの scale プロパティを vector(2,5,3) に設定します。

```
member("Scene").model("Moon").transform.scale = vector(2,5,3)
```

関連事項
transform （プロパティ）, getWorldTransform(), position （トランスフォーム）, rotation （トランスフォーム）, scale （コマンド）

scaleMode

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.scaleMode

// JavaScript syntax
memberOrSpriteObjRef.scaleMode;
```

解説
Flash ムービーまたはベクトルシェイプが、スプライトの境界矩形内をスケールする方法をコントロールするキャストメンバプロパティおよびスプライトプロパティです。scale および viewScale プロパティを設定して Flash ムービースプライトをスケールする場合は、スプライト自体がスケールされず、スプライト内のムービーの表示のみがスケールされます。scaleMode プロパティは次の値を設定できます。

- #showAll （バージョン 7 以前の Director ムービーのデフォルト）- 元の Flash ムービーキャストメンバの縦横の比率を保ちます。横または縦の寸法に隙間があれば、バックカラーを使って必要に応じて塗りつぶします。

- #noBorder - 元の Flash ムービーキャストメンバの縦横の比率を保ちます。横または縦の寸法を必要に応じてクロップします。
- #exactFit - 元の Flash ムービーキャストメンバの縦横の比率を保ちません。Flash ムービーを拡大して、スプライトの寸法に正しく合わせます。
- #noScale - ステージ上でのスプライトのサイズにかかわらず、Flash メディアの元のサイズを保持します。スプライトが元の Flash ムービーよりも小さくなっている場合、スプライトに表示されるムービーは、スプライトの境界に合うようにクロップされます。
- #autoSize (デフォルト) - スプライトの矩形のサイズと位置を、rotation、skew、flipH、および flipV によって、自動的に決めます。つまり、Flash スプライトが回転すると、Director の旧バージョンとしてクロップされません。#autoSize 設定は、scale、viewScale、originPoint、および viewPoint がそれぞれデフォルト値のときにのみ正常に機能します。

このプロパティは、値を調べたり、設定することができます。

例文

このスプライトスクリプトは Director ムービーのステージの色をチェックします。ステージの色が現在のパレットで 0 の位置を示す場合、Flash ムービースプライトの scaleMode プロパティを #showAll に設定します。そうでない場合は、scaleMode プロパティを #noBorder に設定します。

```
-- Lingo syntax
property spriteNum

on beginsprite me
    if _movie.stage.bgColor = 0 then
        sprite(spriteNum).scaleMode = #showAll
    else
        sprite(spriteNum).scaleMode = #noBorder
    end if
end

// JavaScript syntax
function beginsprite() {
    var stgClr = _movie.stage.bgColor;
    if (stgClr == 0) {
        sprite(this.spriteNum).scaleMode = symbol("showAll");
    } else {
        sprite(this.spriteNum).scaleMode = symbol("noBorder");
    }
}
```

関連事項

scale （メンバ）

score

使用法

```
-- Lingo syntax
_movie.score

// JavaScript syntax
_movie.score;
```

解説

現在のムービーにどのスコアが関連づけられているかを特定するためのムービープロパティです。読み取り / 書き込み可能。

現在のスコアの内容を削除し新規作成したり、フィルムループに割り当てたりする前にスコアの内容を保存するときに便利なプロパティです。

例文

次のステートメントは、フィルムループキャストメンバの " 滝 " を現在のムービーのスコアに割り当てます。

```
-- Lingo syntax
_movie.score = member("Waterfall").media

// JavaScript syntax
_movie.score = member("Waterfall").media;
```

関連事項

ムービー

scoreColor

使用法

```
sprite(whichSprite).scoreColor
the scoreColor of sprite whichSprite
```

解説

[スプライト] で指定したスプライトに割り当てられているスコアカラーを示すスプライトプロパティです。このプロパティの値は、現在のパレット内のカラーチップ 0 ～ 5 に対応します。

このプロパティは、値を調べたり、設定することができます。このプロパティは、オーサリング環境およびスコアレコーディングのときにのみ設定できます。

例文

次のステートメントは、スプライト 7 に割り当てられているスコアカラーの値をメッセージウィンドウに表示します。

```
put sprite(7).scorecolor
```

scoreSelection

使用法

```
-- Lingo syntax
_movie.scoreSelection

// JavaScript syntax
_movie.scoreSelection;
```

解説

スコアウィンドウ内でどのチャネルが選択されているかを特定するムービープロパティです。読み取り / 書き込み可能。

情報は、リニアリストのリニアリストとして形式設定されています。選択範囲は、先頭チャネル番号、末尾チャネル番号、先頭フレーム番号、末尾フレーム番号で構成されたリストで表されます。スプライトチャネルはそのチャネル番号で指定し、その他のチャネルは次の値でそれぞれ表します。

チャネル：	値：
フレームスクリプトチャネル	0
サウンドチャネル 1	-1
サウンドチャネル 2	-2
トランジションチャネル	-3
パレットチャネル	-4
テンポチャネル	-5

チャネルやフレームは個別に選択することができます。

例文

次のステートメントは、フレーム 100 ～ 200 の間のスプライトチャネル 15 ～ 25 を選択します。

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200));
```

次のステートメントは、フレーム 100 ～ 200 の間のスプライトチャネル 15 ～ 25 と 40 ～ 50 を選択します。

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200], [40, 50, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200), list(40, 50, 100, 200));
```

次のステートメントは、フレーム 100 ～ 200 の間のフレームスクリプトを選択します。

```
-- Lingo syntax
_movie.scoreSelection = [[0, 0, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(0, 0, 100, 200));
```

関連事項

ムービー

script

使用法

```
-- Lingo syntax
_movie.script[scriptNameOrNum]

// JavaScript syntax
_movie.script[scriptNameOrNum];
```

解説

ムービーのスクリプトキャストメンバへのインデックスまたは名前によるアクセスを提供するムービープロパティです。読み取り専用。

引数 **scriptNameOrNum** は、スクリプトキャストメンバの名前を指定するストリング、またはスクリプトキャストメンバの数を指定する整数を指定します。

- **scriptNameOrNum** がストリングの場合、script プロパティは、メンバが含まれるキャストライブラリに関係なく、スクリプトキャストメンバへのアクセスを提供します。
- **scriptNameOrNum** が整数の場合、script プロパティは参照されたムービーの最初のキャストライブラリ内で検索されたスクリプトキャストメンバへのアクセスのみを提供します。最初のキャストライブラリ以外のキャストライブラリにインデックスを使用してアクセスすることはできません。

例文

次のステートメントは、名前で指定されたスクリプトにアクセスします。

```
-- Lingo syntax
bugScript = _movie.script["Warrior Ant"]

// JavaScript syntax
var bugScript = _movie.script["Warrior Ant"];
```

関連事項

ムービー

scripted

使用法

```
-- Lingo syntax
spriteChannelObjRef.scripted

// JavaScript syntax
spriteChannelObjRef.scripted;
```

解説

スプライトチャネルをスクリプトの制御下に置くか（TRUE）、スコアの制御の下に置くか（FALSE）を指定するスプライトチャネルプロパティです。読み取り専用。

例文

次のステートメントは、チャネルがスクリプトの制御下に置かれていない場合、スプライトチャネル 5 にキャストメンバ " 凧 " からスクリプト化されたスプライトを作成します。

```
-- Lingo syntax
if (channel(5).scripted = FALSE) then
    channel(5).makeScriptedSprite(member("kite"))
end if

// JavaScript syntax
if (channel(5).scripted == false) {
    channel(5).makeScriptedSprite(member("kite"));
}
```

関連事項

スプライトチャネル

scriptingXtraList

使用法

```
-- Lingo syntax
_player.scriptingXtraList

// JavaScript syntax
_player.scriptingXtraList;
```

解説

Director プレーヤで使用可能なすべてのスクリプト作成用 Xtra 拡張機能のリニアリストを返すプレーヤプロパティです。読み取り専用。

Xtra 拡張機能は、アプリケーションの Configuration\Xtras フォルダにあります。

例文

次のステートメントは、メッセージウィンドウに使用可能なすべてのスクリプト作成用 Xtra 拡張機能を表示します。

```
-- Lingo syntax
trace(_player.scriptingXtraList)

// JavaScript syntax
trace(_player.scriptingXtraList);
```

関連事項

mediaXtraList, プレーヤ , スクリプトオブジェクト , toolXtraList, transitionXtraList, xtraList （プレーヤ）

scriptInstanceList

使用法

```
sprite(whichSprite).scriptInstanceList
the scriptInstanceList of sprite whichSprite
```

解説

スプライトに割り当てられたスクリプト参照のリストを返すスプライトプロパティです。このプロパティは実行時にのみ使用できます。ムービーが実行されていないと、空のリストが返されます。リストを変更してもスコアに変更されません。このプロパティは次のような場合に有効です。

- 実行中にビヘイビアーをスプライトに割り当てる場合
- どのビヘイビアーがスプライトに割り当てられているかを確認する場合
- sendSprite コマンドで使うためにビヘイビアーまたはスクリプト参照を検索する場合

このプロパティは、値を調べたり、設定することができます。(既にスプライトがあり、少なくとも 1 つのビヘイビアーのインスタンスが割り当てられている場合にのみ変更できます)。

例文

次のハンドラは、スプライトに割り当てられたスクリプト参照のリストを表示します。

```
-- Lingo
on showScriptRefs spriteNum
    put sprite(spriteNum).scriptInstanceList
end

// Javascript
function showScriptRefs(spriteNum)
{
    trace(sprite(spriteNum).scriptInstanceList);
}
```

次のステートメントは " 騒音 " スクリプトをスプライト 5 に割り当てます。

```
x = script("Big Noise").new()
sprite(5).scriptInstanceList.add(x)
```

関連事項

scriptNum, sendSprite()

scriptList

使用法

```
sprite(whichSprite).scriptList
the scriptList of sprite whichSprite
```

解説

指定されたスプライトに添付されたビヘイビアーとプロパティのリストを返すスプライトプロパティです。このプロパティは、setScriptList() を使用した場合のみ設定できます。スコアの録音中は設定できません。

例文

次のステートメントは、スプライト 1 に添付されたスクリプトのリストをメッセージウィンドウに表示します。

```
put sprite(1).scriptList
-- [[(member 2 of castLib 1), "[#myRotateAngle: 10.0000, #myClockwise: 1, #myInitialAngle: 0.0000]"],
[(member 3 of castLib 1), "[#myAnglePerFrame: 10.0000, #myTurnFrames: 10, #myHShiftPerFrame: 10,
#myShiftFrames: 10, #myTotalFrames: 60, #mySurfaceHeight: 0]"]]
```

関連事項

setScriptList(), value()

scriptNum

使用法

```
sprite(whichSprite). scriptNum
scriptNum of sprite whichSprite
```

解説

[スプライト] で指定したスプライトに添付されているスクリプトの番号を示すスプライトプロパティです。スプライトに複数のスクリプトが添付されている場合、scriptNum は、最初のスクリプトの番号を返します。スプライトに添付されたスクリプトの完全なリストを参照するには、ビヘイビアインスペクタ内でそのスクリプトのビヘイビアのリストを参照してください。

このプロパティの値は、スコアレコーディング中に調べたり、設定することができます。

例文

次のステートメントは、スプライト 4 に添付されているスクリプトの番号を表示します。

```
-- Lingo
put sprite(4).scriptNum

// Javascript
trace(sprite(4).scriptNum);
```

関連事項

scriptInstanceList

scriptsEnabled

使用法

```
-- Lingo syntax
memberObjRef.scriptsEnabled

// JavaScript syntax
memberObjRef.scriptsEnabled;
```

解説

リンクされたムービー内のスクリプトをオンにするかどうかを特定する Director ムービーキャストメンバプロパティです。TRUE または 1 のときは、スクリプトはオンになります。FALSE または 0 のときは、スクリプトはオフになります。

このプロパティは、Director ムービーキャストメンバに対してのみ有効です。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、" ジャズの歴史 " リンクムービー内のスクリプトをオフにします。

```
-- Lingo syntax
member("Jazz Chronicle").scriptsEnabled = FALSE

// JavaScript syntax
member("Jazz Chronicle").scriptsEnabled = 0;
```

scriptSyntax

使用法

```
member whichScript.scriptSyntax
```

解説

メンバのスクリプトシンタックスのシンボルを返すキャストメンバプロパティです。使用できる値は #lingo または #javaScript です。読み取り専用。

例文

次のステートメントは、メンバ Main Script のスクリプトシンタックスタイプを表示します。

```
-- Lingo
put member("Main Script").scriptSyntax
// JavaScript
put(member("Main Script").scriptSyntax);
```

scriptText

使用法

```
-- Lingo syntax
memberObjRef.scriptText

// JavaScript syntax
memberObjRef.scriptText;
```

解説

キャストメンバに割り当てられているスクリプトの内容（存在する場合）を示すためのメンバプロパティです。

読み取り / 書き込み可能。

ムービーがプロジェクタに変換されたり、保護されたり、Shockwave Player 用に圧縮された場合は、スクリプトのテキストは消去されます。このようなムービーは、scriptText プロパティの値を失ってしまうので、したがって、ムービーがプロジェクタで再生されているときに、ムービーの scriptText プロパティの値を取得することはできません。ただし、Director はプロジェクタ内で scriptTex プロパティの新規の値を設定することができます。このように新しく割り当てられたスクリプトは自動的にコンパイルされるので、すばやく実行できます。

例文

次のステートメントは、フィールドキャストメンバ "20" の内容をキャストメンバ "30" のスクリプトにします。

```
-- Lingo syntax
member(20).text = member(30).scriptText

// JavaScript syntax
member(20).text = member(30).scriptText;
```

関連事項

メンバ

scriptType

使用法

```
member whichScript.scriptType
the scriptType of member whichScript
```

解説

指定したスクリプトの種類を特定するためのキャストメンバプロパティです。値は、#movie、#score、および #parent です。

例文

次のステートメントは、" メインスクリプト " スクリプトメンバをムービースクリプトにします。

```
-- Lingo
member("Main Script").scriptType = #movie

// Javascript
member("Main Script").scriptType = Symbol("movie");
```

scrollTop

使用法

```
-- Lingo syntax
memberObjRef.scrollTop

// JavaScript syntax
memberObjRef.scrollTop;
```

解説

フィールドキャストメンバの上端から現在スクロールボックス内に表示されているフィールドの上端までの距離をピクセル数で示すキャストメンバプロパティです。ムービー再生中に scrollTop の値を変更すると、スクロールフィールド内に表示されるフィールドのセクションを変更することができます。

同じ方法でテキストおよびフィールドメンバのスクロールビヘイビアーを設定することができます。

例えば、次の Lingo では、sliderVal の変数の値によって、" クレジット " フィールドキャストメンバをフィールドボックス内で上下に移動できます。

```
global sliderVal

on prepareFrame
    newVal = sliderVal * 100
    member("Credits").scrolltop = newVal
end
```

グローバル変数 sliderVal を使うと、ユーザがスライダをどれぐらいドラックしたか測定できます。ステートメント set newVal = sliderVal * 100 は sliderVal を増加させて、値はユーザがスライダをドラッグする距離より大きい値になります。sliderVal が正の数の場合、テキストは上方向に移動し、sliderVal が負の数の場合、テキストは下方向に移動します。

例文

次の繰り返しループは、scrollTop メンバプロパティの値を増やし続けることで [クレジット] フィールドのスクロールを自動で行うことができます。

```
--Lingo syntax
on wa
    member("Credits").scrollTop = 1
    repeat with count = 1 to 150
        member("Credits").scrollTop = member("Credits").scrollTop + 1
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function wa() {
    member("Credits").scrollTop = 1;
    for (var count = 1; count <= 150; count++) {
        member("Credits").scrollTop = member("Credits").scrollTop + 1;
        _movie.updateStage();
    }
}
```

sds （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).sds.whichProperty
```

解説

3D モディファイア。モデルに詳細なジオメトリを追加します。モデルがカメラに近付くと、追加した詳細を合成してカーブを滑らかにします。addModifier() を使用してモデルに sds モディファイを追加すると、sds モディファイアのプロパティを設定できます。

sds モディファイアは、直接モデルリソースに影響します。sds モディファイアと lod モディファイアは互いに反対の機能を持つため、一緒に使用する場合は注意する必要があります。sds モディファイアが詳細なジオメトリを追加するのに対し、lod モディファイアはジオメトリの詳細を削除します。sds モディファイアを追加する際には、事前に lod.auto モディファイアプロパティを FALSE に設定し、lod.level モディファイアプロパティを目的の解像度に設定しておくことをお勧めします。例えば次のようになります。

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

sds モディファイアは、inker モディファイアや toon モディファイアを既に使用しているモデルには使用できません。

モデルリソースに sds モディファイアを追加すると、以下のプロパティを取得または設定できます。

enabled。サブディビジョンサーフェスの機能が有効 (TRUE) か無効 (FALSE) かを示します。このプロパティのデフォルト設定は TRUE です。

depth。sds モディファイアを使用する際にモデルが表示できる解像度の最大レベル数を指定します。

error。サブディビジョンサーフェスの機能のエラー許容度のレベルを示します。このプロパティは、sds.subdivision プロパティが #adaptive に設定されている場合にのみ適用できます。

subdivision。サブディビジョンサーフェスモディファイアの処理モードを示します。使用できる値は以下のとおりです。

- #uniform。メッシュは細部にわたって均一に拡大され、各フェイスが同じ倍率で細分化されます。
- #adaptive。サーフェスの向きに大きな変更があった場合に、現在表示可能なメッシュの領域にのみ詳細が追加されます。

注意：これらのプロパティの詳細については、各プロパティのエントリを参照してください。

例文

次のステートメントは、" 地形 " というモデルの sds.depth プロパティの値を表示します。

```
-- Lingo
put member("3D").model("Terrain").sds.depth
-- 2
// Javascript
trace(member("3D").getPropRef("model",1).sds.depth
// 2
```

関連事項

lod （モディファイア）, toon （モディファイア）, inker （モディファイア）, depth (3D), enabled (sds), error, subdivision, addModifier

searchCurrentFolder

使用法

```
-- Lingo syntax
_player.searchCurrentFolder

// JavaScript syntax
_player.searchCurrentFolder;
```

解説

ファイルを名前で検索するときに、Director が現在のフォルダを探すかどうかを特定するためのプレーヤプロパティです。読み取り / 書き込み可能。

- searchCurrentFolder プロパティが TRUE (1) の場合は、ファイル名を解決するときに現在のフォルダを検索します。
- searchCurrentFolder プロパティが FALSE (0) の場合は、ファイル名を解決するときに現在のフォルダを検索しません。

デフォルトでは、このプロパティは、TRUE です。

注意：_player.searchCurrentFolder は、オーサリングおよびプロジェクタの runModes でのみ使用可能です。プラグインの runMode では使用できません。

例文

次のステートメントは、searchCurrentFolder プロパティのステータスをメッセージウィンドウに表示します。結果は、TRUE を表す数値の 1 になります。

```
-- Lingo syntax
put(_player.searchCurrentFolder)

// JavaScript syntax
put(_player.searchCurrentFolder);
```

関連事項

プレーヤ

searchPathList

使用法

```
-- Lingo syntax
_player.searchPathList

// JavaScript syntax
_player.searchPathList;
```

解説

Director がデジタルビデオ、GIF、ビットマップ、またはサウンドファイルなど、リンクされたメディアを検索するときに使用するパス名のリストを返すプレーヤプロパティです。読み取り / 書き込み可能。

パスのリストの各項目は、実行時に現在のプラットフォームに表示されるときの完全パス名です。

searchPathList は、通常のリストと同じ構成になっていて、add()、addAt()、append()、deleteAt()、および setAt() などのリスト用コマンドを使って操作することができます。デフォルト値は空のリストです。

searchPaths のパスとして URL を使うことはできません。

searchPaths リスト内のパスの数が多くなると、それに応じて検索の時間がかかることになります。リスト内のパスの数を最小限に抑えるようにします。

注意：このプロパティを設定すると、その後のすべてのムービー上で機能します。現在のムービーのアセットは既にロードされているので、設定変更しても、これらのアセットには影響しません。

例文

次のステートメントは、Director がファイル名を解決するときに使うパスを表示します。

```
-- Lingo syntax
trace(_player.searchPathList)

// JavaScript syntax
trace(_player.searchPathList);
```

次のステートメントは、Windows で 2 つのフォルダを searchPaths に割り当てます。

```
-- Lingo syntax
_player.searchPathList = ["C:\Director\Projects\", "D:\CDROM\Sources\"]

// JavaScript syntax
_player.searchPathList = list("C:\\Director\Projects\","D:\CDROM\Sources\");
```

次のステートメントは、Mac で 2 つのフォルダを searchPaths に割り当てます。

```
-- Lingo syntax
_player.searchPathList = ["Hard Drive:Director:Projects:", "CDROM:Sources:"]

// JavaScript syntax
_player.searchPathList = list("Hard Drive:Director:Projects:","CDROM:Sources:");
```

関連事項

プレーヤ , searchCurrentFolder

selectedButton

使用法

```
-- Lingo syntax
dvdObjRef.selectedButton

// JavaScript syntax
dvdObjRef.selectedButton;
```

解説

現在フォーカスのあるボタンのインデックスを返す DVD プロパティです。読み取り専用。

例文

次のステートメントは、DVD オブジェクトで選択されたボタンを表示します。

```
-- Lingo
put sprite(1).selectedButton
--1

// Javascript
trace(sprite(1).selectedButton)
//1
```

関連事項
DVD

selectedText

使用法

```
-- Lingo syntax
memberObjRef.selectedText

// JavaScript syntax
memberObjRef.selectedText;
```

解説
1 つのオブジェクト参照として現在選択されているテキストのチャンクを返すテキストキャストメンバです。フォント文字と実際の文字のストリング情報にアクセスできます。

例文
次のハンドラは、ローカル変数オブジェクトに置かれていて、現在選択されているテキストを表示します。次にそのオブジェクトは、テキストの様々な文字を参照するために使われます。その詳細は、メッセージウィンドウに表示されます。

```
--Lingo syntax
property spriteNum

on mouseUp(me)
    mySelectionObject = sprite(spriteNum).member.selectedText
    put(mySelectionObject.text)
    put(mySelectionObject.font)
    put(mySelectionObject.fontSize)
    put(mySelectionObject.fontStyle)
end

// JavaScript syntax
function mouseUp() {
    var mySelectionObject = sprite(this.spriteNum).member.selectedText;
    trace(mySelectionObject.text);
    trace(mySelectionObject.font);
    trace(mySelectionObject.fontSize);
    trace(mySelectionObject.fontStyle);
}
```

selection

使用法

```
-- Lingo syntax
castObjRef.selection

// JavaScript syntax
castObjRef.selection;
```

解説
指定したキャストウィンドウで選択したキャストメンバを返すキャストライブラリプロパティです。読み取り / 書き込み可能。

例文

次のステートメントは、castLib 1 のキャストメンバ 1 ～ 10 を選択します。

```
-- Lingo syntax
castLib(1).selection = [[1, 10]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10) );
```

次のステートメントは、castLib 1 のキャストメンバ 1 ～ 10 と、30 ～ 40 を選択します。

```
-- Lingo syntax
castLib(1).selection = [[1, 10], [30, 40]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10), list(30, 40) );
```

関連事項

キャストライブラリ

selection （テキストキャストメンバプロパティ）

使用法

```
member(whichTextMember).selection
```

解説

テキスト / フィールドキャストメンバプロパティ。テキストキャストメンバで選択された先頭の文字および末尾の文字のリストを返します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、スプライトテキストメンバ myAnswer の 6 文字目から 10 文字目までを選択し、ハイライトさせます。

```
member("myAnswer").selection = [6, 10]
```

関連事項

color(), selStart, selEnd

selEnd

使用法

```
-- Lingo syntax
_movie.selEnd

// JavaScript syntax
_movie.selEnd;
```

解説

選択範囲の末尾の文字を特定するためのムービープロパティです。このプロパティは、selStart と共に使用され、現在の編集フィールドにおける選択範囲を先頭の文字から数えた値で示します。selEnd の最大値は、現在の編集フィールドの文字数です。このプロパティはフィールドメンバスプライトに適用されますが、テキストメンバスプライトには適用されません。selStart に selEnd より大きい値を割り当てると、selEnd の値は selStart の値にリセットされます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 0 です。

例文

次のステートメントは、「abcdefg」という内容のフィールドから「cde」を選択します。

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5
// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

次のステートメントは、選択範囲の長さを 20 文字にします。

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20
// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

関連事項

editable, hilite （コマンド）, selection() （関数）, selStart, text

selStart

使用法

```
-- Lingo syntax
_movie.selStart
// JavaScript syntax
_movie.selStart;
```

解説

選択範囲の先頭の文字より前の位置を指定するためのムービープロパティです。このプロパティは、selEnd と共に使用され、現在の編集フィールドにおける選択範囲を示します。値 0 は最初の文字より前の位置を示します。selStart の最大値は、現在の編集フィールドの文字数です。このプロパティはフィールドメンバスプライトに適用されますが、テキストメンバスプライトには適用されません。selStart に selEnd より大きい値を割り当てると、selEnd の値は selStart の値にリセットされます。

このプロパティは、値を調べたり、設定することができます。デフォルト値は 0 です。

例文

次のステートメントは、「abcdefg」という内容のフィールドから「cde」を選択します。

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5

// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

次のステートメントは、選択範囲の長さを 20 文字にします。

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20

// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

関連事項

selection() （関数）, selEnd, text

serialNumber

使用法

```
-- Lingo syntax
_player.serialNumber
// JavaScript syntax
_player.serialNumber;
```

解説

ムービープロパティ。Director のインストール中に入力されたシリアル番号を含むストリングを返します。読み取り専用。

このプロパティはオーサリング環境でのみ使用できます。このプロパティは、ユーザ情報を表示するようにカスタマイズできるウィンドウ内ムービー（MIAW）ツールで使用できます。

例文

次のハンドラは MIAW のムービースクリプトに置かれて、ウィンドウが開いたときに表示されるフィールドにユーザ名とシリアル番号を表示します。

```
-- Lingo syntax
on prepareMovie
displayString = _player.userName & RETURN & _player.organizationName & RETURN & _player.serialNumber
member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
var displayString = _player.userName + "\n" + _player.organizationName + "\n" + _player.serialNumber;
member("User Info").text = displayString;
}
```

関連事項

プレーヤ

shader

使用法

```
member(whichCastmember).shader(whichShader)
member(whichCastmember).shader[index]
member(whichCastmember).model(whichModel).shader
member(whichCastmember).modelResource(whichModelResource).face[index].shader
```

解説

3D エレメント、モデルプロパティ、およびフェイスプロパティ。モデルのサーフェスの外観を定義するオブジェクトです。シェーダとは、モデルによって使用されるモデルリソースを覆う「皮膚」のようなものです。

シェーダ自体は画像ではありません。シェーダの表示コンポーネントは、最大 8 レイヤーの texture を使って作成されます。これら 8 つのテクスチャレイヤーは、Director 内のビットマップキャストメンバか画像オブジェクトから作成されるか、または 3D モデリングプログラムのモデルを使って読み込みます。詳細については、「texture」を参照してください。

各モデルには、shaderlist と呼ばれるシェーダのリニアリストがあります。リストのエントリの数は、モデルによって使用されているモデルリソースのメッシュの数と同じです。各メッシュに適用できるシェーダは、それぞれ 1 つだけです。

3D キャストメンバには、DefaultShader というデフォルトのシェーダがあります。このシェーダは削除できません。このシェーダは、モデルにシェーダが割り当てられていない場合や、モデルが使用するシェーダが削除された場合に使用されます。

member(キャストメンバ).model(モデル).shader というシンタックスを使うと、モデルの shaderlist の最初のシェーダにアクセスできます。このシンタックスは、member(キャストメンバ).model(モデル).shaderList[1] と同じです。

シェーダの作成や削除には、それぞれ newShader() コマンドと deleteShader() コマンドを使用します。

シェーダは、3D キャストメンバのシェーダパレットに格納されます。格納されているシェーダは、名前 (シェーダ) で参照することも、パレットのインデックス (シェーダインデックス) で参照することもできます。1 つのシェーダを使用できるモデルの数に制限はありません。シェーダに変更を加えると、そのシェーダを使用しているすべてのモデルに影響します。

シェーダには 4 つの種類があります。

#standard シェーダは、テクスチャを写実的に適用します。

#painter シェーダ、#engraver シェーダ、および #newsprint シェーダは、それぞれ、絵画、版画、および新聞印刷の効果をテクスチャに対して適用します。この 3 つのシェーダには、#standard シェーダのプロパティの他に特別なプロパティがあります。

シェーダのプロパティについて詳しくは、Director ヘルプパネルの「Director の使い方」トピックを参照してください。

#mesh プリミティブの個々のフェイスに使用するシェーダは、member(**whichCastmember**).modelResource(**whichModelResource**).face[**index**].shader というシンタックスを使用して設定します。このプロパティを変更するには、build() コマン を呼び出す必要があります。

例文

次のステートメントは、" 壁 " というモデルの shader プロパティを、"WallSurface" というシェーダに設定します。

```
member("Room").model("Wall").shader = member("Room").shader("WallSurface")
```

関連事項

shaderList, newShader, deleteShader, face, texture

shaderList

使用法

```
member(whichCastmember).model(whichModel).shaderList
member(whichCastmember).model(whichModel).shaderList[index]
```

解説

3D モデルプロパティ。モデルに適用される shadowPercentage のリニアリストです。リストのエントリの数は、モデルによって使用されているモデルリソースのメッシュの数と同じです。各メッシュに適用できるシェーダは、それぞれ 1 つだけです。

shaderlist の **index** で指定した位置のシェーダを設定するには、次のシンタックスを使用します。member(キャストメンバ).model(モデル).shaderList[インデックス] = shaderReference

3D テキストでは、各文字が個別のメッシュになります。シェーダを設定する文字の番号を index に設定します。

shaderList のすべてのインデックス位置を同じシェーダに設定するには、次のシンタックスを使用します。この場合、shaderList のインデックスは指定しません。member(キャストメンバ).model(モデル).shaderList = shaderReference

shaderlist 内のシェーダのプロパティを設定するには、次のシンタックスを使用します。member(キャストメンバ).model(モデル).shaderList[インデックス] プロパティ = プロパティ値

モデルのすべてのシェーダのプロパティを同じ値に設定するには、次のシンタックスを使用します。この場合、shaderList のインデックスは指定しません。

```
member(whichCastmember).model(whichModel).shaderList.whichProperty = propValue
```

例文

次のステートメントは、" バンパー " というモデルの shaderList の 2 番目のシェーダを " クロム " というシェーダに設定します。

```
member("Car").model("Bumper").shaderList[2] = member("Car").shader("Chrome")
```

次のステートメントは、" バンパー " というモデルの shaderList のすべてのシェーダを " クロム " というシェーダに設定します。

```
member("Car").model("Bumper").shaderList = member("Car").shader("Chrome")
```

関連事項

shadowPercentage, shader, deleteShader, newShader

shadowPercentage

使用法

```
member(whichCastmember).model(whichModel).toon.shadowPercentage
member(whichCastmember).model(whichModel).shader.shadowPercentage
member(whichCastmember).shader(whichShader).shadowPercentage
```

解説

3D toon モディファイアおよび painter シェーダプロパティ。ライトによってハイライトが作成されていないモデルのサーフェス領域で使用できるカラーのパーセンテージを示します。

このプロパティの範囲は 0 ～ 100 で、デフォルト値は 50 です。

モデルの toon モディファイアおよび painter シェーダが使用するカラーの数は、モデルの toon モディファイアまたは painter シェーダの colorSteps プロパティによって決まります。

例文

次のステートメントは、" ティーポット " というモデルの toon モディファイアの shadowPercentage プロパティを 50 に設定します。これにより、このモデルの toon モディファイアで使用できるカラーの半分が、モデルのサーフェスのシャドウ領域に使用されます。

```
-- Lingo
member("shapes").model("Teapot").toon.shadowPercentage = 50

// Javascript
member("shapes").getPropRef("model",1).getProp("toon").shadowPercentage=50;
```

関連事項

colorSteps, shadowStrength

shadowStrength

使用法

```
member(whichCastmember).model(whichModel).toon.shadowStrength
member(whichCastmember).model(whichModel).shader.shadowStrength
member(whichCastmember).shader(whichShader).shadowStrength
```

解説

3D toon モディファイアおよび #painter シェーダプロパティ。ライトによってハイライトが作成されていないモデルのサーフェス領域の明度を示します。

このプロパティのデフォルト値は 1.0 です。

例文

次のステートメントは、" 球体 " というモデルの toon モディファイアの shadowStrength プロパティを 0.1 に設定します。これにより、このモデルのサーフェスのハイライト部分以外の領域は非常に暗くなります。

```
member("Shapes").model("Sphere").toon.shadowStrength = 0.1
```

shapeType

使用法

```
member(whichCastMember).shapeType
the shapeType of member whichCastMember
```

解説

指定したシェイプの種類を示すシェイプキャストメンバプロパティです。種類は #rect、#roundRect、#oval、#line のいずれかになります。このプロパティは、Lingo で作成したシェイプキャストメンバに対して、後からシェイプの種類を特定したいときなどに便利です。

例文

次のステートメントは、新たに 100 のシェイプキャストメンバを作成し、種類を楕円 (oval) に設定します。

```
-- Lingo
new(#shape, member 100)
member(100).shapeType = #oval

// Javascript
var t = _movie.newMember(symbol("shape"));
t.shapeType=symbol("oval");
```

shiftDown

使用方法

```
-- Lingo syntax
_key.shiftDown

// JavaScript syntax
_key.shiftDown;
```

解説

ユーザが Shift キーを押しているかどうかを示すキープロパティです読み取り専用。

このプロパティは、ユーザが Shift キーを押していると TRUE を返し、押していないと FALSE を返します。

このプロパティは、他のキーと関連して調べる必要があります。

例文

次のステートメントは、Shift キーが押されたかどうかをチェックし、押されている場合は doCapitalA ハンドラを、パラメータ the key を付けて呼び出します。

```
-- Lingo syntax
if (_key.shiftDown) then
    doCapitalA(_key.key)
end if

// JavaScript syntax
if (_key.shiftDown) {
    doCapitalA(_key.key);
}
```

関連事項

controlDown, キー , key, keyCode, optionDown

shininess

使用法

```
member(whichCastmember).shader(whichShader).shininess
member(whichCastmember).model(whichModel).shader.shininess
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].shininess
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、サーフェスの輝きを取得または設定できます。輝きは、ハイライト部分に割り当てられたシェーダサーフェスのパーセンテージで定義されます。値は 0 ～ 100 の整数であり、デフォルトは 30 です。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

次のステートメントは、モデル "gbCyl3" のシェーダリスト内にある最初のシェーダの shininess プロパティを 60 に設定します。これにより、このシェーダのサーフェスの 60% がハイライト部分に割り当てられます。

```
-- Lingo
member("Scene").model("gbCyl3").shader.shininess = 60

// Javascript
member("Scene").getPropRef("model",1).getProp("shader").shininess=60;
```

silhouettes

使用法

```
member(whichCastmember).model(whichModel).inker.silhouettes
member(whichCastmember).model(whichModel).toon.silhouettes
```

解説

toon モディファイアおよび inker モディファイアの 3D プロパティ。モデルの表示されるエッジにモディファイアが線を描画するかどうかを示します。(TRUE) の場合は描画され、(FALSE) の場合は描画されません。

シルエットラインは、カメラの投影平面上のモデルの 2D 画像の周りに描画されます。メッシュ上に描画される折り目や境界線とは異なり、シルエットラインとモデルのメッシュとの関係は定まっていません。

シルエットラインは、子供のぬり絵の輪郭線のようなものです。

このプロパティのデフォルト値は TRUE です。

例文

次のステートメントは、" 球体 " というモデルの inker モディファイアの silhouettes プロパティを FALSE に設定します。この場合、モデルの輪郭線は描画されません。

```
-- Lingo
member("Shapes").model("Sphere").inker.silhouettes = FALSE

// Javascript
member("Shapes").getPropRef("model",1).inker.silhouetess = false;
```

size

使用法

```
-- Lingo syntax
memberObjRef.size

// JavaScript syntax
memberObjRef.size;
```

解説

指定したキャストメンバのメモリ内のサイズをバイト数で返すメンバプロパティです。読み取り専用。

合計を 1024 で割ると、キロバイトに変換できます。

例文

次のステートメントは、キャストメンバ " 神社 " のサイズを " 大きさ " というフィールドに出力します。

```
-- Lingo syntax
member("How Big").text = string(member("shrine").size)

// JavaScript syntax
member("How Big").text = member("shrine").size.toString();
```

関連事項

メンバ

sizeRange

使用法

```
member(whichCastmember).modelResource
(whichModelResource).sizeRange.start
modelResourceObjectReference.sizeRange.start
member(whichCastmember).modelResource
(whichModelResource).sizeRange.end
modelResourceObjectReference.sizeRange.end
```

解説

3D プロパティ。#particle というタイプのモデルリソースを指定すると、そのモデルリソースの sizeRange の start および end プロパティを取得または設定できます。パーティクルは、ワールド単位で計測されます。

システムのパーティクルのサイズは、sizeRange.start から sizeRange.end の間、各パーティクルの寿命が終わるまで一定の比率で補間されます。

このプロパティは、0 より大きい整数に設定する必要があります。デフォルト値は 1 です。

例文

この例の mrFount は、#particle というタイプのモデルリソースです。次のステートメントは、"mrFount" の sizeRange プロパティを設定します。1 行目は開始値を 4 に設定し、2 行目は終了値を 1 に設定します。これにより、"mrFount" のパーティクルは、最初にサイズ 4 で表示され、寿命が終わるまでの間に徐々にサイズ 1 まで縮小されていきます。

```
-- Lingo
member("fountain").modelResource("mrFount").sizeRange.start = 4
member("fountain").modelResource("mrFount").sizeRange.end = 1

// Javascript
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").start=4;
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").end=4;
```

sizeState

使用法

```
-- Lingo syntax
windowObjRef.sizeState

// JavaScript syntax
windowObjRef.sizeState;
```

解説

ウィンドウのサイズの状態を返すウィンドウプロパティです。読み取り専用。

返されるサイズの状態は、次のいずれかの値になります。

サイズの状態	解説
#minimized	現在ウィンドウが最小化されていることを示します。
#maximized	現在ウィンドウが最大化されていることを示します。
#normal	現在ウィンドウが最小化も最大化もされていないことを示します。

例文

次のステートメントは、Artists という名前のウィンドウが最大化されていない場合に、このウィンドウを最大化します。

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

関連事項

ウィンドウ

skew

使用法

```
-- Lingo syntax
spriteObjRef.skew

// JavaScript syntax
spriteObjRef.skew;
```

解説

スプライトの垂直エッジが垂直線から傾く (skew) 角度を小数点以下第 2 位のフロート値として返すプロパティです。読み取り / 書き込み可能。

負の値のときは左に傾き、正の値のときは右に傾きます。値が 90° 以上になると、画像は上下反転します。

スコアは +21,474,836.47° ～ -21,474,836.48° までの間で画像を傾斜させ、左右両方向に 59,652 回のフル回転ができます。

傾斜の限界に達すると（59,652 回目の回転から少し経ったとき）、傾斜角度は +116.47° または -116.48° にリセットされ、0.00° にはリセットされません。これは、+21,474,836.47° が +116.47° に等しく、-21,474,836.48° が -116.48°（または +243.12°）に等しいからです。このリセット条件を避けるには、連続的に傾斜させるときに、角度を両方向とも ±360° 以内に制限します。

例文

次のビヘイビアーは角度を 360° 以内に保ちつつ、再生ヘッドが進むたびにスプライトを 2° ずつ傾けます。

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).skew = integer(sprite(spriteNum).skew + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
    sprite(this.spriteNum).skew = parseInt(sprite(this.spriteNum).skew + 2) % 360;
}
```

関連事項

flipH, flipV, rotation, スプライト

smoothness

使用法

```
member(whichTextmember).smoothness
member(whichCastMember).modelResource(whichExtruderModelResource).smoothness
```

解説

押し出しモデルリソースおよびテキストの 3D プロパティ。このプロパティを使用すると、3D テキストキャストメンバの作成に使うセグメントの数を制御する整数値を取得または設定できます。値が大きいほど、テキストは滑らかに表示されます。このプロパティの範囲は 1 ～ 10 で、デフォルト値は 5 です。

押し出しモデルリソースおよびテキストキャストメンバの使い方の詳細については、extrude3D を参照してください。

例文

この例のキャストメンバ " ロゴ " はテキストキャストメンバです。次のステートメントは、" ロゴ " の smoothness プロパティを 8 に設定します。これにより、3D モードで " ロゴ " を表示した場合に、文字のエッジが非常に滑らかに表示されます。

```
-- Lingo
member("Logo").smoothness = 8

// Javascript
member("Logo").smoothness = 8;
```

次の例で、モデル " スローガン " のモデルリソースは押し出しテキストです。次のステートメントは、" スローガン " のモデルリソースの smoothness プロパティを 1 に設定します。これにより、" スローガン " は非常に角張った文字になります。

```
-- Lingo
member("Scene").model("Slogan").resource.smoothness = 1

// Javascript
member("Scene").getPropRef("model",1).getProp("resource").smoothness=1;
```

関連事項

extrude3D

sound （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.sound

// JavaScript syntax
memberObjRef.sound;
```

解説

ムービー、デジタルビデオ、Flash ムービーのサウンドを再生するかどうかを指定するキャストメンバプロパティです。再生する場合は TRUE（デフォルト）、しない場合は FALSE です。読み取り / 書き込み可能。

フラッシュメンバを新しく設定した場合は、現在再生中のサウンドが終了した時点で有効となります。

完成したムービーでの sound の使用例については、Director アプリケーションフォルダ内の Learning\Lingo Examples フォルダにある sound control.dir を参照してください。

例文

次のハンドラは、メンバ参照を受け取り、メンバの sound プロパティのオンとオフを切り替えます。

```
-- Lingo syntax
on ToggleSound(whichMember)
    member(whichMember).sound = not(member(whichMember).sound)
end

// JavaScript syntax
function ToggleSound(whichMember) {
    member(whichMember).sound = !(member(whichMember).sound);
}
```

関連事項

Flash ムービー

sound （プレーヤ）

使用法

```
-- Lingo syntax
_player.sound[intSoundChannelNum]

// JavaScript syntax
_player.sound[intSoundChannelNum];
```

解説

プレーヤプロパティを使用して、サウンドチャネルオブジェクトへのインデックスを使用したアクセスを提供するプレーヤプロパティです。読み取り専用。

引数 **intSoundChannelNum** は、アクセスするサウンドチャネルの数を指定する整数です。

このプロパティの機能は、sound() メソッドのトップレベルと同じです。

例文

次のステートメントは変数 mySound をサウンドチャネル 3 のサウンドに設定します。

```
-- Lingo syntax
mySound = _player.sound[3]

// JavaScript syntax
var mySound = _player.sound[3];
```

関連事項

プレーヤ , sound(), サウンドチャネル

soundChannel (SWA)

使用法

```
-- Lingo syntax
memberObjRef.soundChannel

// JavaScript syntax
memberObjRef.soundChannel;
```

解説

SWA サウンドが再生されるサウンドチャネルを特定するための Shockwave Audio (SWA) キャストメンバプロパティです。

チャネル番号を指定していない、または 0 を指定した場合は、SWA ストリーミングキャストメンバは、使われていない最も番号の大きいサウンドチャネルにサウンドを割り当てます。

Shockwave Audio ストリーミングサウンドは、スプライトチャネル内でスプライトとして表示することもできますが、サウンドはサウンドチャネル内で再生されます。SWA サウンドスプライトは、サウンドチャネル番号でなくスプライトチャネル番号で参照することをお勧めします。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、SWA ストリーミングキャストメンバ " フランク・ザッパ " をサウンドチャネル 3 で再生します。

```
-- Lingo syntax
member("Frank Zappa").soundChannel = 3

// JavaScript syntax
member("Frank Zappa").soundChannel = 3;
```

soundChannel (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.soundChannel

// JavaScript syntax
memberOrSpriteObjRef.soundChannel;
```

解説

RealMedia ストリームのオーディオを再生するサウンドチャネルを取得または設定する、RealMedia スプライトまたはキャストメンバプロパティです。このプロパティを設定すると、Lingo のサウンド用のメソッドとプロパティを使用して、RealMedia ストリームのオーディオを制御できます。このプロパティに 0 ～ 8 以外の値を設定すると、Lingo エラーが発生します。realPlayerNativeAudio() が TRUE に設定されている場合は、このプロパティは無効です。

このプロパティのデフォルト設定は 0 で、RealMedia オーディオは最も大きい番号のサウンドチャネルで再生されます。このプロパティの値は、使用されているチャネルに応じて再生中に変わります。RealMedia キャストメンバの再生中は、このプロパティの値は現在使用されているサウンドチャネルを示します。RealMedia キャストメンバの再生が停止すると、このプロパティは 0 に戻ります。

このプロパティに指定したチャネル（1 ～ 8）が現在再生中（ムービーのほかの部分）のときは、そのサウンドが停止され、RealMedia オーディオがそのチャネルで再生されます。

RealMedia キャストメンバの同時再生はサポートされていません。同時に再生される RealMedia キャストメンバがムービーに含まれる場合は、これらのサウンドは同じチャネルで同時に再生されます。

例文

次の例は、スプライト 2 およびキャストメンバ "Real" の RealMedia ストリームのサウンドがサウンドチャネル 2 で再生されることを示しています。

```
-- Lingo syntax
put(sprite(2).soundChannel) -- 2
put(member("Real").soundChannel) -- 2

// JavaScript syntax
put(sprite(2).soundChannel); // 2
put(member("Real").soundChannel); // 2
```

次の例は、サウンドチャネル 1 をスプライト 2 およびキャストメンバ "Real" の RealMedia ストリームに割り当てます。

```
-- Lingo syntax
sprite(2).soundChannel = 1
member("Real").soundChannel = 1

// JavaScript syntax
sprite(2).soundChannel = 1;
member("Real").soundChannel = 1;
```

関連事項

realPlayerNativeAudio()

soundDevice

使用法

```
-- Lingo syntax
_sound.soundDevice

// JavaScript syntax
_sound.soundDevice;
```

解説

ムービーの再生中に、サウンドミキシングデバイスを設定するサウンドプロパティです。読み取り / 書き込み可能。

soundDeviceList にリストされているデバイスが、soundDevice に設定可能です。

Windows のサウンドデバイスはそれぞれ違った利点があるので、いくつかのサウンドデバイスを参照してください。

- MacroMix (Windows®) - Windows の再生における共通項です。このデバイスはどの Windows においても機能しますが、他のデバイスと違って待ち時間がかかります。
- QT3Mix（Windows）- サウンドを QuickTime オーディオとミックスします。DirectSound を使っている場合は、他のアプリケーションでもミックスすることができます。このデバイスは QuickTime がインストールされていることが条件で、待ち時間は MacroMix よりも短くなっています。
- DirectSound®（Windows）- QT3Mix に似ていますが、待ち時間は短くなります。
- MacSoundManager（Mac）- Mac で使用できる唯一のサウンドデバイスです。

例文

次のステートメントは、サウンドデバイスを Windows の MacroMix に設定します。新しく割り当てられたデバイスが動作しない場合は、soundDevice プロパティは変更されません。

```
-- Lingo syntax
_sound.soundDevice = "MacroMix"

// JavaScript syntax
_sound.soundDevice = "MacroMix";
```

関連事項

サウンド , soundDeviceList

soundDeviceList

使用法

```
-- Lingo syntax
_sound.soundDeviceList

// JavaScript syntax
_sound.soundDeviceList;
```

解説

現在のマシンで利用できるサウンドデバイスのリニアリストを作成するサウンドプロパティです。読み取り専用。

リストの中で、Mac に有効なデバイスは MacSoundManager のみです。

例文

次のステートメントは Windows の代表的なサウンドデバイスリストを表示します。

```
-- Lingo syntax
trace(_sound.soundDeviceList)

// JavaScript syntax
trace(_sound.soundDeviceList);
```

関連事項

サウンド , soundDevice

soundEnabled

使用法

```
-- Lingo syntax
_sound.soundEnabled

// JavaScript syntax
_sound.soundEnabled;
```

解説

サウンドプロパティ。サウンドがオンであるか（TRUE、デフォルト）、オフであるか（FALSE）を示します。読み取り / 書き込み可能。

このプロパティを FALSE に設定すると、サウンドはオフになります。ただし、音量の設定は変更されません。

例文

次のステートメントは soundEnabled を現在の設定と逆の設定にします (オンの場合はオフに、オフの場合はオンにします)。

```
-- Lingo syntax
_sound.soundEnabled = not(_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

関連事項

サウンド

soundKeepDevice

使用法

```
-- Lingo syntax
_sound.soundKeepDevice

// JavaScript syntax
_sound.soundKeepDevice;
```

解説

Windows のみに有効で、サウンドの再生のたびに、サウンドドライバをアンロードおよびリロードするかを指定するサウンドプロパティです。読み取り / 書き込み可能。

このプロパティのデフォルト値は TRUE で、サウンド再生のたびに、サウンドデバイスがアンロードおよびリロードされないようにします。

サウンドが終了したあとに、サウンドデバイスが確実にアンロードされ、またマシン上の他のアプリケーションやプロセスに使用できるようにするため、サウンドを再生する前にこのプロパティを FALSE に設定しておく必要があります。

サウンド再生が Director のアプリケーションで頻繁に行われる場合、このプロパティを FALSE に設定すると、パフォーマンスに悪影響を及ぼします。

例文

次のステートメントは soundKeepDevice プロパティを FALSE に設定します。

```
-- Lingo syntax
_sound.soundKeepDevice = FALSE

// JavaScript syntax
_sound.soundKeepDevice = false;
```

関連事項

サウンド

soundLevel

使用法

```
-- Lingo syntax
_sound.soundLevel

// JavaScript syntax
_sound.soundLevel;
```

解説

内蔵スピーカーから流れるサウンドの音量を返す、または設定するサウンドプロパティです。読み取り / 書き込み可能。

設定範囲は 0 (ミュート) ～ 7 (最大音量) までとなります。デフォルト値は 7 に設定されています。

Windows では、システムサウンド設置は外部スピーカーの音量調節とコンバインします。したがって、サウンドレベル設定を変えれば、実際の音量が変えられます。設定結果をユーザが受け入れることを確認してから、soundLevel プロパティの設定を行ってください。個々のチャネルやスプライトのボリュームは、サウンドチャネルオブジェクトの volume プロパティで設定する方が適切です。

これらの値は Mac のサウンド制御パネルの設定と一致します。このプロパティを使用すると、スクリプトは直接音量を変えることができ、またサウンドが特定レベルのときに、他の動作を行うこともできます。

完成したムービーの soundLevel の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "sound control.dir" を参照してください。

例文

次のステートメントは変数 oldSound を現行音量と同等のものに設定します。

```
-- Lingo syntax
oldSound = _sound.soundLevel

// JavaScript syntax
var oldSound = _sound.soundLevel;
```

次のステートメントは音量を 5 に設定します。

```
-- Lingo syntax
_sound.soundLevel = 5

// JavaScript syntax
_sound.soundLevel = 5;
```

関連事項

サウンド , volume (Windows Media)

soundMixMedia

使用法

```
-- Lingo syntax
_sound.soundMixMedia

// JavaScript syntax
_sound.soundMixMedia;
```

解説

Flash キャストメンバのサウンドとスコアサウンドチャネルのサウンドをミックスするかどうかを指定するサウンドプロパティです。読み取り / 書き込み可能。

Director 7 以降で作成されたムービーでは、デフォルトは TRUE です。それ以前のバージョンで作成されたムービーでは、デフォルトは FALSE です。このプロパティは、Windows でのみ有効です。

このプロパティを TRUE に設定すると、Flash キャストメンバのサウンドとスコアサウンドチャネルのサウンドをミックスできます。Director が Flash キャストメンバのサウンドのミックスと再生を実行します。

Flash サウンド再生の場合は、微妙な差が生じることがあります。Flash サウンドを Flash で再生されるように正確に聞くには、このプロパティを FALSE に設定します。

FALSE に設定すると、Flash サウンドをミックスすることはできず、別々に再生する必要があります。

例文

次のステートメントは、オーサリング環境で、soundMixMedia プロパティのデフォルト値を示します。

```
-- Lingo
put _sound.soundMixMedia
-- 1

// Javascript
trace( _sound.soundMixMedia);
// 1
```

関連事項

サウンド

soundObjectList

使用法

```
Mixer.soundObjectList
```

解説

ミクサ内のサウンドオブジェクトのリストを返すオーディオミクサプロパティです。

例文

```
--Lingo syntax
on mouseUp me
put mixer1.soundObjectList.count -- Displays the count of the sound objects in mixer1.
end

// JavaScript syntax
function mouseUp()
{
put (mixer1.soundObjectList.count ); // Displays the count of the sound objects
// in mixer1.


}
```

関連事項

ミクサ

source

使用法

```
sprite(whichSprite).camera.backdrop[backdropIndex].source
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].source
sprite(whichSprite).camera.overlay[overlayIndex].source
member(whichCastmember).camera(whichCamera).overlay
[overlayIndex].source
```

解説

3D バックドロップおよびオーバーレイのプロパティ。このプロパティを使用すると、バックドロップまたはオーバーレイのソース画像として使用するテクスチャを取得または設定できます。

例文

次のステートメントは、バックドロップの 1 つ目のソースをテクスチャ " 杉 " に設定します。

```
sprite(3).camera.backdrop[1].source =
sprite(3).member.texture("Cedar")
```

関連事項

bevelDepth, overlay

sourceFileName

使用法

```
flashCastMember.sourceFileName
```

解説

Flash キャストメンバプロパティ。起動して編集する操作で使用する FLA ソースファイルのパス名を指定します。このプロパティは、値を調べたり、設定することができます。デフォルトは空のストリングです。

例文

次のステートメントは、Flash キャストメンバ「SWF」の sourceFileName を C:\FlashFiles\myFile.fla: に設定します。

```
-- Lingo
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla"

// Javascript
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla";
```

sourceRect

使用法

```
-- Lingo syntax
windowObjRect.sourceRect

// JavaScript syntax
windowObjRect.sourceRect;
```

解説

ウィンドウプロパティ。ウィンドウ内で再生されるムービーの元のステージ座標を示します。読み取り専用。

このプロパティを使用して、ウィンドウをドラッグや rect の設定をして変更した後に、ウィンドウを元のサイズと位置に戻すことができます。

例文

次のステートメントは、"Control_panel" という名前のステージの元の座標値をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(window("Control_panel").sourceRect)

// JavaScript syntax
put(window("Control_panel").sourceRect);
```

関連事項

ウィンドウ

specular (light)

使用法

```
member(whichCastmember).light(whichLight).specular
```

解説

3D ライトプロパティ。このプロパティを使用すると、光沢をオン (TRUE) にするか、またはオフ (FALSE) にするかを取得または設定できます。光沢とは、モデル上にハイライト部分を表示させる機能です。このハイライト部分では、モデルを照らすライトがカメラに向かって反射します。モデルのシェーダの輝きによって、ハイライトの光沢部分の大きさが決まります。環境ライトの場合、このプロパティの値は無視されます。このプロパティのデフォルト値は TRUE です。

注意：このプロパティをオフにするとパフォーマンスが向上する場合があります。

例文

次のステートメントは、"omni2" というライトの specular プロパティを FALSE に設定します。このライトは、ハイライト部分を発生させません。現在シーンを照らしているライトが他にない場合は、シーンのどのシェーダにも光沢ハイライトがないことになります。

```
-- Lingo
member("3d world").light("omni2").specular = FALSE

// Javascript
member("3d world").getPropRef("light",1).specular = false;
```

関連事項

silhouettes, specularLightMap

specular (shader)

使用法

```
member(whichCastmember).shader(whichShader).specular
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、指定したシェーダの光沢カラーを取得または設定できます。光沢カラーとは、光沢がオンになっている場合に作成されるハイライトのカラーのことです。specular プロパティが TRUE に設定されているライトがシーンにないと、このプロパティによる効果は得られません。光沢カラーは、オブジェクトを照らすライトのカラーによる影響を受けます。光沢カラーが白の場合に、ライトのカラーが赤であると、オブジェクト上に表示される光沢ハイライトは赤になります。このプロパティのデフォルト値は rgb(255, 255, 255) です。この値は白を表します。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

```
-- Lingo
put member("scene").shader("plutomat").specular
--rgb(11, 11, 11)

// Javascript
trace(member("scene").getPropRef("shader",1).specular
// <rgb(11,11,11)>
```

関連事項

silhouettes, specular (light), specularColor, emissive

specularColor

使用法

```
member(whichCastmember).specularColor
```

解説

3D キャストメンバプロパティ。このプロパティを使用すると、キャストメンバ内の最初のシェーダの光沢カラーの RGB 値を取得または設定できます。キャストメンバのシェーダパレットの最初のシェーダは、常にデフォルトのシェーダです。このプロパティに限らず、3D キャストメンバのすべてのプロパティは、キャストメンバと一緒に保存され、次にムービーを開いたときに復元されます。このプロパティのデフォルト値は rgb(255, 255, 255) です。この値は白を表します。

例文

次のステートメントは、キャストメンバ " シーン " の最初のシェーダの光沢カラーを rgb(255, 0, 0) に設定します。これは、member(" シーン ").shader[1].specular = rgb(255, 0, 0) と同じです。ただし、このシンタックスを使用すると、ムービーの保存時に、新しい値がキャストメンバと共には保存されません。メンバのみ。specularColor は新しい色の値を保存します。

```
-- Lingo
member("Scene").specularColor = rgb(255, 0, 0)

// Javascript
member("Scene").specularColor = rgb(255, 0, 0);
```

関連事項

silhouettes, specular (light), specular (shader)

specularLightMap

使用法

```
member(whichCastmember).shader(whichShader).specularLightMap
member(whichCastmember).model(whichModel).shader.specularLightMap
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].specularLightMap
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、指定した標準シェーダの 5 番目のテクスチャレイヤーを取得または設定できます。toon モディファイアがモデルリソースに適用される場合は、このプロパティは無視されます。

設定できる値は以下のとおりです。

- textureModeList[5] = #specular
- blendFunctionList[5] = #add
- blendFunctionList[1] = #replace
- default = void

このヘルパープロパティは、光沢ライトマッピングの一般的な使用をセットアップするためのより簡単なインターフェイスを提供します。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

次のステートメントは、テクスチャ " 楕円 " を、モデル "GlassBox" によって使用されるシェーダの specularLightMap として設定します。

```
-- Lingo
member("3DPlanet").model("GlassBox").shader.specularLightMap = \ member("3DPlanet").texture("Oval")

// Javascript
member("3DPlanet").getPropRef("model",1).getProp("shader").specularLightMap
=member("3DPlanet").getPropRef("texture",1);
```

関連事項

diffuseLightMap

spotAngle

使用法

```
member(whichCastmember).light(whichLight).spotAngle
```

解説

3D プロパティ。このプロパティを使用すると、ライトの投影範囲 (円錐形) の角度を取得または設定できます。このプロパティに指定されている角度の外側を照らすライトは、輝度に影響しません。このプロパティの値は 0.0 ～ 180.00 の浮動小数点値であり、デフォルト値は 90.0 です。指定する浮動小数点値は、実際の角度の半分に相当します。したがって、角度を 90° にするには、45.0 を指定します。

例文

次のステートメントは、ライト "unidirectional" の spotAngle プロパティを 8 に設定します。このライトの投影範囲 (円錐形) の角度は 16 度になります。

```
-- Lingo
member("3d world").light("unidirectional").spotAngle = 8

// Javascript
member("3d world").getPropRef("light",1).spotAngle = 8;
```

spotDecay

使用法

```
member(whichCastmember).light(whichLight).spotDecay
```

解説

3D ライトプロパティ。このプロパティを使用すると、カメラからの距離に応じてスポットライトの輝度を落とすかどうかを取得または設定できます。このプロパティのデフォルト値は FALSE です。

例文

次のステートメントは、ライト 1 の spotDecay プロパティを TRUE に設定します。これにより、ライト 1 から遠い位置にあるモデルは、ライト 1 に近い位置にあるモデルより暗くなります。

```
-- Lingo
member("Scene").light[1].spotDecay = TRUE

// Javascript
member("Scene").getPropRef("light",1).spotDecay = true;
```

sprite （ムービー）

使用法

```
-- Lingo syntax
_movie.sprite[spriteNameOrNum]

// JavaScript syntax
_movie.sprite[spriteNameOrNum];
```

解説

ムービースプライトへのインデックスまたは名前によるアクセスを提供するムービープロパティです。読み取り専用。

引数 **spriteNameOrNum** は、スプライトの名前を指定するストリング、またはスプライトの数を指定する整数を指定します。

例文

次のステートメントは、変数 sportSprite にスプライト 5 のムービーを設定します。

```
-- Lingo syntax
sportSprite = _movie.sprite[5]

// JavaScript syntax
var sportSprite = _movie.sprite[5];
```

関連事項

ムービー

sprite （スプライトチャネル）

使用法

```
-- Lingo syntax
spriteChannelObjRef.sprite

// JavaScript syntax
spriteChannelObjRef.sprite;
```

解説

スプライトチャネルの現行フレームでスプライトへの参照を返すスプライトチャネルプロパティです。読み取り専用。

例文

次のステートメントは、スプライトチャネル " リボン " 内のスプライトチャネルを変数 mySprite に設定します。

```
-- Lingo syntax
mySprite = channel("Ribbon").sprite

// JavaScript syntax
var mySprite = channel("Ribbon").sprite;
```

関連事項

スプライトチャネル

spriteNum

使用法

```
-- Lingo syntax
spriteObjRef.spriteNum

// JavaScript syntax
spriteObjRef.spriteNum;
```

解説

ビヘイビアーのスプライトが割り当てられたチャネル番号を定義し、その番号をどのビヘイビアーでも利用できるようにするためのスプライトプロパティです。読み取り専用。

このプロパティは、ビヘイビアーが使う他のプロパティと共に、ビヘイビアーのトップで簡単にプロパティを知らせます。

new() ハンドラを使用してビヘイビアーのインスタンスを作成した場合は、スクリプトの new() ハンドラ内で明示的に spriteNum プロパティにスプライトの番号を設定する必要があります。これによって、スクリプトが添付されているスプライトを識別できます。new() ハンドラを呼び出すときは、スプライト番号をパラメータとして new() ハンドラに渡す必要があります。

例文

次のハンドラでは、spriteNum プロパティは、システムにより新しく作られたスクリプトインスタンスに自動的に設定されます。

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on mouseDown me
    sprite(spriteNum).member = member("DownPict")
end

// JavaScript syntax
function mouseDown() {
    sprite(this.spriteNum).member = member("DownPict");
}
```

次のハンドラは、新しいプロパティ変数 pMySpriteRef へのスプライト参照を割りあてるため、便宜上 spriteNum プロパティへインサートされた自動値を使用します。

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on beginSprite me
    pMySpriteRef = sprite(me.spriteNum)
end

// JavaScript syntax
function beginSprite() {
    this.pMySpriteRef = sprite(this.spriteNum);
}
```

この方法で参照 pMySpriteRef を、スクリプトの後半で次のハンドラと共に使うようにできます。

```
-- Lingo syntax
currMember = pMySpriteRef.member

// JavaScript syntax
var currMember = pMySpriteRef.member
```

またはこれより少し長くなりますが、同じ意味を含む次のシンタックスを使うこともできます。

```
Lingo syntax
currMember = sprite(spriteNum).member

// JavaScript syntax
var currMember = sprite(this.spriteNum).member
```

このように 2 つの方法がありますが、機能的には同じです。

関連事項

new(), スプライト

stage

使用法

```
-- Lingo syntax
_movie.stage

// JavaScript syntax
_movie.stage;
```

解説

メインムービーを参照するためのムービープロパティです。読み取り専用。

子ムービーからメインムービーへメッセージを送るときなどに使うことができます。

例文

次のステートメントは、ステージの現行設定を表示します。

```
-- Lingo syntax
put(_movie.stage.rect)

// JavaScript syntax
put(_movie.stage.rect);
```

関連事項

ムービー

startAngle

使用法

```
member(whichCastmember).modelResource(whichModelResource).
startAngle
modelResourceObjectReference.startAngle
```

解説

3D プロパティ。#cylinder または #sphere タイプのモデルリソースを指定すると、そのモデルリソースの startAngle プロパティを取得および設定できます。値は 360.0 ～ 0.0 の浮動小数点値です。このプロパティのデフォルト値は 0.0 です。

startAngle プロパティは、モデルリソースの描画の開始角度を決定します。球体や円柱の描画の際に、endAngle プロパティと共に使用されます。例えば、半球体を作成するには、startAngle を 0.0 に設定し、endAngle を 180.0 に設定します。

例文

次のステートメントは、モデルリソース " 球体 01" の startAngle を 0.0 に設定します。endAngle が 90 に設定されている場合、このモデルリソースを使用するモデルは、全体の 1/4 しか表示されません。

```
-- Lingo
put member("3D World").modelResource(1).startAngle
-- 0.0

// Javascript
trace(member("3D World").getPropRef("modelResource",1).startAngle);
// 0
```

関連事項

endAngle

startFrame

使用法

```
-- Lingo syntax
spriteObjRef.startFrame

// JavaScript syntax
spriteObjRef.startFrame;
```

解説

スプライト範囲の開始フレームのフレーム番号を返すスプライトプロパティです。読み取り専用。

このプロパティは、スコア上で特定のスプライトがカバーする範囲を調べるのに役立ちます。指定したスプライトが含まれるフレームでのみ使用できます。ムービーの他のフレームにあるスプライトには適用できません。

例文

次のステートメントは、チャネル 5 のスプライトの開始フレームをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(5).startFrame)

// JavaScript syntax
put(sprite(5).startFrame);
```

関連事項

endFrame, スプライト

startTime (サウンドチャネル)

使用法

```
-- Lingo syntax
soundChannelObjRef.startTime

// JavaScript syntax
soundChannelObjRef.startTime;
```

解説

現在再生中または一時停止中のサウンドがキューに登録されたとき設定されていた開始時刻を示すサウンドチャネルプロパティです。読み取り専用。

サウンドをキューに登録した後でこのプロパティを設定することはできません。サウンドがキューに登録されるときに値が指定されなかった場合は、このプロパティはゼロを返します。

例文

次のステートメントは、チャネル 5 内のデジタルビデオスプライトの再生を、デジタルビデオの 100 チックの場所から開始します。

```
-- Lingo syntax
sprite(5).startTime = 100

// JavaScript syntax
sprite(5).startTime = 100;
```

関連事項

サウンドチャネル

startTime （サウンドオブジェクト）

使用法

```
soundObject.startTime
```

解説

現在のサウンドオブジェクトの開始時間をミリ秒単位で返すサウンドオブジェクトプロパティです。

例文

```
soundObject.startTime = 6890
--Lingo syntax
on mouseUp me
    put soundObjRef.startTime -- Displays the start time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.startTime) ; // Displays the start time for the sound object
// associated with soundobjectRef.
}
```

関連事項

endTime （サウンドオブジェクト）

startTimeList

使用法

```
-- Lingo syntax
dvdObjRef.startTimeList

// JavaScript syntax
dvdObjRef.startTimeList;
```

解説

DVD プロパティ。再生が開始する時間またはチャプタを指定するプロパティリストです。

読み取り / 書き込み可能。

startTimeList は、チャプタベースまたはタイムベースのプロパティリストです。

チャプタベースの startTimeList には、次のプロパティが含まれています。

- title。再生するチャプタを含むタイトルを指定します。
- chapter。再生するチャプタを指定します。

この startTimeList は、タイトル 1 のチャプタ 2 で再生を開始します。

```
[#title:1, #chapter:2]
```

タイムベースの startTimeList には、次のプロパティが含まれています。

- title タイトルを指定します。
- hours。再生が開始する時間数を指定します。
- min。再生が開始する分数を指定します。
- sec。再生が開始する秒数を指定します。
- frames。再生が開始するフレームを指定します。

この startTimeList は、タイトル 1 の特定の時間に再生を開始します。

```
[#title:1, #hours:0, #minutes:45, #seconds:15, #frames:15]
```

この startTimeList は、1 回限りのパラメータのみをリストします。

```
[#title:1, #seconds:15]
```

startTimeList は、0 に設定して消去できます。

例文

次のステートメントは、DVD オブジェクトの startTimeList を 0 に設定します。

```
-- Lingo
sprite(1).startTimeList = 0

// Javascript
sprite(1).startTimeList = 0;
```

関連事項

DVD, play() (DVD), stopTimeList

state (3D)

使用法

```
member(whichCastmember).state
```

解説

3D プロパティ。参照先のメンバの、ストリーミングプロセスやロードプロセスにおける現在の状態を返します。このプロパティで取得できる値は、最初のファイルの読み込みの際の値か、または要求した最後のファイルのロードの際の値です。

メンバの state プロパティによって、キャストメンバに対して実行できる 3D Lingo があるかどうか、ある場合にはどの 3D Lingo を実行できるのかが決まります。

このプロパティは次の値をとることができます。

- 0。現在、メンバがロードされていないので、使用できる 3D メディアがないことを示します。この場合、メンバに対して実行できる 3D Lingo はありません。
- 1。メディアのロードが開始されたことを示します。
- 2。メンバの初期ロードセグメントがロードされたことを示します。モデルファイルの作成時にストリームの優先順位が 0 に設定されたすべてのオブジェクトは、初期ロードセグメントに含まれるため、この時点でロードされます。ロードの優先順位が 0 に設定されているオブジェクトに関連するほとんどの 3D Lingo を実行できます。loadFile コマンドと resetWorld コマンドは、この状態では使用しないでください。
- 3。メンバのその他のすべてのメディアがロードおよび解凍されていることを示します。この時点では、ほとんどの 3D Lingo を実行できます。loadFile コマンドと resetWorld コマンドは、この状態では使用しないでください。
- 4。メンバのメディアがすべてロードされ、解凍が完了したことを示します。キャストメンバに対してすべての 3D Lingo を実行できます。
- -1。メディアのストリーミングプロセスの間に、未定義のエラーが発生したことを示します。ロードプロセスのどの時点でエラーが発生したのかがわからないため、キャストメンバの状態は信頼できません。

一般的に、現在の状態が 3 よりも低い 3D キャストメンバに対しては、Lingo を使用しないでください。

例文

次のステートメントは、"PartyScene" というキャストメンバのロードが完了し、再生の準備が整ったかどうかをメッセージウィンドウに表示します。

```
-- Lingo
put member("PartyScene").state
-- 4

// Javascript
trace(member("PartyScene").state);
// 4
```

state (Flash, SWA)

使用法

```
-- Lingo syntax
memberObjRef.state

// JavaScript syntax
memberObjRef.state;
```

解説

キャストメンバプロパティ。Shockwave Audio（SWA）ストリーミングキャストメンバおよび Flash ムービーキャストメンバについて、ストリーミングファイルの現在の状態を示します。streamName、URL、preLoadTime などのプロパティは、SWA サウンドが停止したときにのみ変更ができます。

SWA ファイル用の cuePointNames、cuePointTimes、currentTime、duration、percentPlayed、percentStreamed、bitRate、sampleRate、numChannels などのプロパティは、ファイルのストリーミングが開始した後でのみ、意味のある情報を返します。

SWA ストリーミングキャストメンバの値は次のとおりです。

- 0 - キャストのストリーミングが停止しました。
- 1 - キャストメンバは再ロード中です。
- 2 - プリロードが正常に終了しました。
- 3 - キャストメンバは再生中です。
- 4 - キャストメンバは一時停止中です。
- 5 - キャストメンバがストリーミングを終了しました。
- 9 - エラーが起こりました。
- 10 - CPU のスペースが不十分です。

Flash ムービーキャストメンバの場合、このプロパティは、Director ムービーが実行中の場合のみ以下の有効な値を返します。

- 0 - キャストメンバがメモリ内にありません。
- 1 - ヘッダは現在ロード中です。
- 2 - ヘッダのロードが終了しました。
- 3 - キャストメンバのメディアは現在ロード中です。
- 4 - キャストメンバのメディアのロードが終了しました。
- -1 - エラーが起こりました。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、SWA ストリーミングキャストメンバ "Ella Fitzgerald" でエラーが検出されたときにアラートを発行します。

```
-- Lingo syntax
on mouseDown
    if member("Ella Fitzgerald").state = 9 then
        _player.alert("Sorry, can't find an audio file to stream.")
    end if
end

// JavaScript syntax
function mouseDown() {
    var ellaSt = member("Ella Fitzgerald").state;
    if (ellaSt == 9) {
        _player.alert("Sorry, can't find an audio file to stream.");
    }
}
```

関連事項

clearError(), getError() (Flash, SWA)

state (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.state

// JavaScript syntax
memberOrSpriteObjRef.state;
```

解説

RealMedia スプライトまたはキャストメンバのプロパティ。RealMedia ストリームの現在の状態を 1 ～ 4 の整数で返します。状態を表す各整数の値は、ストリーミングのプロセスの特定のポイントに対応します。このプロパティは、再生中はダイナミックに値が変わります。値を調べられますが設定はできません。

ストリーミングプロセスが開始するのは、再生ヘッドがスコア内の RealMedia スプライトの範囲に入ったとき、RealMedia スプライトまたはキャストメンバに対して play メソッドが呼び出されたとき、または RealMedia ビューアでユーザが「再生」ボタンをクリックしたとき、のいずれかです。このプロパティを呼び出すと、RealMedia キャストメンバのストリーミングプロセスの状態を示す整数値が返されます。各状態には、1 つまたは複数の対応する mediaStatus (RealMedia, Windows Media) プロパティの値があります。それぞれの mediaStatus 値は、特定の状態でのみ検出されます。例えば、mediaStatus プロパティの値 #seeking と #buffering は、state の値が 3 のときにのみ存在します。

state プロパティの値は、キャストメンバに対する Lingo の実行に関して重要な情報を提供します。例えば、member.state が 2 よりも小さいときは、いくつかの Lingo プロパティが無効な値になります。この結果、プロパティデータに依存する Lingo も不正になります。member.state が 2 以上で 4 未満のとき、RealMedia キャストメンバは表示されません。しかし、すべての Lingo プロパティとメソッドは正しく定義された値を持つので、これらを使用して Lingo の処理をキャストメンバに適用できます。

ストリーミングプロセスが開始すると、エラー (-1) が発生しない限り、state プロパティは循環的に次の状態を遷移します。エラーが発生すると、ストリーミングプロセスが停止します。

-1 (エラー) は、不具合があることを示します。前の RealMedia ストリームのエラーが残っている可能性があります。より詳しい情報を得るには、lastError プロパティを調べます。この状態は、mediaStatus プロパティの #error と等価です。

0 (クローズ) は、ストリーミングプロセスが開始されていない、キャストメンバプロパティが開始状態にある、またはキャストメンバプロパティが前のキャストメンバの再生からのコピーである、のいずれかを示します。この状態は、mediaStatus プロパティの #closed と等価です。

1 (接続中) は、ストリーミングは開始しているが、サーバへの接続の非常に早い段階にあることを示します。十分な情報がないので、ローカルではキャストメンバに操作を実行できません。この状態は、mediaStatus プロパティの #connecting と等価です。

2 (オープン) は、Lingo のプロパティが実際のストリームによって更新されたことを示します。state が 2 以上のときは、RealMedia ストリームの height、width、および duration の各プロパティの値が確定しています。この状態は一時的なもので、すぐに状態 3 に移行します。この状態は、mediaStatus プロパティの #opened と等価です。

3（シークまたはバッファ中）は、RealMedia キャストメンバに関する Lingo のすべてのプロパティが更新されたことを示します。ただし、キャストメンバはまだ再生の準備は整っていません。ステージビューアまたは RealMedia には、黒い矩形または RealNetworks のロゴが表示されます。この状態が、ネットワークの混雑によるリバッファリングの結果のときは、state の値はすぐに 4 (再生中) に戻ります。この状態は、mediaStatus プロパティの #buffering または #seeking と等価です。

4 (再生中) は、RealMedia ストリームが、問題やエラーのない状態で再生されている (または、一時停止中である) ことを示します。正常に再生されている間は、状態はこの値をとります。この状態は、mediaStatus プロパティの #playing または #paused と等価です。

例文

次の例は、スプライト 2 およびキャストメンバ "Real" のストリームの状態が 0、つまりクローズされていることを示します。

```
-- Lingo syntax
put(sprite(2).state) -- 0
put(member("Real").state) -- 0

// JavaScript syntax
put(sprite(2).state); // 0
put(member("Real").state); // 0
```

関連事項

mediaStatus (RealMedia, Windows Media), percentBuffered, lastError

static

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.static

// JavaScript syntax
memberOrSpriteObjRef.static;
```

解説

ムービーにアニメーションが含まれるかどうかによって、Flash ムービースプライトの再生パフォーマンスを制御するキャストメンバプロパティおよびスプライトプロパティです。ムービーにアニメーションが含まれる場合（FALSE、デフォルト）、このプロパティはスプライトをフレームごとに描き直します。ムービーにアニメーションが含まれない場合（TRUE）、このプロパティは、移動するかサイズを変更したときのみスプライトを描き直します。

このプロパティは、値を調べたり、設定することができます。

注意：Flash ムービースプライトが他の移動 Director スプライトと交差しない場合のみ、static プロパティを TRUE に設定します。Flash ムービーが移動 Director スプライトと交差していると、スプライトは正しく描き直されません。

例文

次のスプライトスクリプトは、メッセージウィンドウに Flash ムービースプライトのチャネル番号を表示し、Flash ムービーにアニメーションが含まれるかどうかを示します。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).static then
        animationType = "does not have animation."
    else
        animationType = "has animation."
    end if
    put("The Flash movie in channel" && spriteNum && animationType)
end

// JavaScript syntax
function beginSprite() {
    var st = sprite(this.spriteNum).static;
    if (st == 1) {
        animationType = "does not have animation.";
    } else {
        animationType = "has animation.";
    }
    trace("The Flash movie in channel " + this.spriteNum + animationType);
}
```

staticQuality

使用法

```
-- Lingo syntax
spriteObjRef.staticQuality

// JavaScript syntax
spriteObjRef.staticQuality;
```

解説

パノラマ画像が静止状態のときに使うコーデックの質を指定するための QuickTime VR スプライトプロパティです。値は、#minQuality、#maxQuality、#normalQuality のいずれかになります。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは QuickTime スプライトの staticQuality をメッセージウィンドウに表示します。

```
-- Lingo
put sprite(1).staticQuality
-- #normalQuality

// Javascript
trace(sprite(1).staticQuality)
// symbol(normalQuality)
```

status

使用法

```
-- Lingo syntax
soundChannelObjRef.status

// JavaScript syntax
soundChannelObjRef.status;
```

解説

サウンドチャネルプロパティで、サウンドチャネルのステータスを示します。読み取り専用。

使用できる値は以下のとおりです。

Status	名前	意味
0	アイドル	待ち状態、または再生中のサウンドはありません。
1	ロード中	待ち状態のサウンドのプリロード中ですが、まだ再生していません。
2	待ち状態	サウンドチャネルでは待ち状態のサウンドのプリロードを完了しましたが、まだそのサウンドを再生してはいません。
3	再生中	サウンドを再生中です。
4	一時停止中	サウンドは一時停止中です。

例文

次のステートメントでは、サウンドチャネル 2 の現在のステータスをメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

関連事項

サウンドチャネル

status （ミクサ）

使用法

```
mixer.status (Read-only)
```

解説

ミクサのステータスを返すオーディオプロパティです。このプロパティは次のいずれかの値をとることができます。

- #playing
- #paused
- #stopping
- #stopped

ミクサが #stopping 状態のときは、play や stop などの呼び出しはサウンドオブジェクトとその中のミクサで無視されます。

例文

次の例は、ミクサが停止した場合に、exit フレームの mixer1 の再生を開始します。

```
--Lingo syntax
on exitFrame me
     if (mixer1.status = #stopped)then
       mixer1.play()
     end if
end

// JavaScript syntax
function exitFrame() {
if (mixer1.status = #stopped)then
     mixer1.play();
end if
}
```

関連事項

ミクサ

status （サウンドオブジェクト）

使用法

```
soundObj.status (Read-only)
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトのステータスを返します。このプロパティは次のいずれかの値をとります。

- #playing
- #paused
- #stopping
- #stopped

サウンドオブジェクトが #stopping 状態のときは、play および stop の呼び出しは無視されます。

例文

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.status -- Returns the current status of the sound object.
end

// JavaScript syntax
function exitframe(){
put (soundObjRef.status) ; // Returns the current status of the sound object.

}
```

stillDown

使用法

```
-- Lingo syntax
_mouse.stillDown

// JavaScript syntax
_mouse.stillDown;
```

解説

ユーザがマウスボタンを押しているか（TRUE）、押していないか（FALSE）を確認するためのマウスプロパティです。読み取り専用。

この関数は、mouseDown スクリプト内で mouseUp 関数の後に特定のイベントを実行したいときに使うと便利です。

stillDown がループ内で使用されていると、スクリプトはこの関数の状況を調べることができません。代わりにループ内で mouseDown 関数を使用してください。

例文

次のステートメントは、マウスボタンが押されているかどうかをチェックし、押されている場合は dragProcedure ハンドラを呼び出します。

```
-- Lingo syntax
if (_mouse.stillDown) then
    dragProcedure
end if

// JavaScript syntax
if (_mouse.stillDown) {
    dragProcedure();
}
```

関連事項

マウス , mouseDown, mouseUp

stopTime

使用法

```
sprite(whichSprite).stopTime
the stopTime of sprite whichSprite
```

解説

指定したデジタルビデオスプライトの停止時間を特定するためのスプライトプロパティです。stopTime の値はチック数で測ります。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、チャネル 5 内のデジタルビデオスプライトを、デジタルビデオの 100 チックの場所で停止します。

```
-- Lingo
sprite(5).stopTime = 100

// Javascript
sprite(5).stopTime = 100;
```

stopTimeList

使用法

```
-- Lingo syntax
dvdObjRef.stopTimeList

// JavaScript syntax
dvdObjRef.stopTimeList;
```

解説

DVD プロパティ。再生が停止する時間またはチャプタを指定するプロパティリストです。

読み取り / 書き込み可能。

stopTimeList は、チャプタベースまたはタイムベースのプロパティリストです。

チャプタベースの stopTimeList には、次のプロパティが含まれています。

- title。タイトルを指定します。
- chapter。チャプタを指定します。このチャプタが再生された後に、再生が停止します。

この stopTimeList は、タイトル 1 のチャプタ 4 で再生を停止します。

```
[#title:1, #chapter:4]
```

タイムベースの stopTimeList には、次のプロパティが含まれています。

- BULLET_ITEM
- title タイトルを指定します。
- hours。再生が停止する時間数を指定します。
- min。再生が停止する分数を指定します。
- sec。再生が停止する秒数を指定します。
- frames。再生が停止するフレームを指定します。

この stopTimeList は、タイトル 1 の特定の時間に再生を停止します。

```
[#title:1, #hours:0, #minutes:55, #seconds:45, #frames:15]
```

この stopTimeList は、1 回限りのパラメータのみをリストします。

```
[#title:1, #seconds:45]
```

stopTimeList は、0 に設定して消去できます。

関連事項

DVD, play() (DVD), startTimeList

streamMode

使用法

```
-- Lingo syntax
memberObjRef.streamMode

// JavaScript syntax
memberObjRef.streamMode;
```

解説

リンクされた Flash ムービーキャストメンバをメモリにストリームする方法を制御するための Flash キャストメンバプロパティです。制御の方法には以下のものがあります。

- #frame (デフォルト) - スプライトがステージにある間、フレームが進むたびにキャストメンバの一部をストリームします。
- #idle - スプライトがステージにある間、idle イベントが生成されるたび、または Director の各フレームごとに少なくとも 1 回、キャストメンバの一部をストリームします。
- #manual - キャストメンバに stream コマンドが発行されたときのみ、そのキャストメンバの一部をメモリにストリームします。

このプロパティは、値を調べたり、設定することができます。

例文

次の startMovie スクリプトは、内蔵キャストから Flash ムービーキャストメンバを検索し、それらのキャストメンバの streamMode プロパティを #manual に設定します。

```
-- Lingo syntax
on startMovie
    repeat with i = 1 to castLib(1).member.count
    if member(i, 1).type = #flash then
        member(i, 1).streamMode = #manual
    end if
    end repeat
end

// JavaScript syntax
function startMovie() {
    i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, 1).streamMode = symbol("manual");
            i++;
        }
    }
}
```

streamName

使用法

```
-- Lingo syntax
memberObjRef.streamName

// JavaScript syntax
memberObjRef.streamName;
```

解説

Shockwave Audio（SWA）キャストメンバプロパティ。ストリーミングキャストメンバの URL またはファイル名を示します。このプロパティは、URL メンバ ロパティと同じ機能を持ちます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、BigBand.swa というファイルを SWA ストリーミングキャストメンバにリンクします。MyDisk ディスクの Sounds フォルダにリンクファイルがあります。

```
-- Lingo syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa"

// JavaScript syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa";
```

streamSize

使用法

```
-- Lingo syntax
memberObjRef.streamSize

// JavaScript syntax
memberObjRef.streamSize;
```

解説

指定したキャストメンバのストリーム内の合計バイト数を表す整数値を調べるためのキャストメンバプロパティです。streamSize プロパティは、Director ムービーが再生中の場合のみ、値を返します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のフレームスクリプトは、Flash ムービーキャストメンバの "Intro Movie" が、メモリへのストリーミングを終了したかどうかをチェックします。終了していない場合、このスクリプトは、ストリームされたバイト数 (bytesStreamed メンバプロパティを使用) と、キャストメンバの合計バイト数 (streamSize メンバプロパティを使用) とを表すようにフィールドキャストメンバ "Message Line" を更新します。このスクリプトは、ムービーのメモリへのロードが完了するまで、再生ヘッドを現在のフレームでループさせます。

```
-- Lingo syntax
on exitFrame
    if member("Intro Movie").percentStreamed < 100 then
        member("Message Line").text = string(member("Intro Movie").bytesStreamed) && "of" \ &&
string(member("Intro Movie").streamSize) && "bytes have downloaded so far."
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member("Intro Movie").percentStreamed;
    var strSs = new String(member("Intro Movie").streamSize);
    var strIm = new String(member("Intro Movie").bytesStreamed);
    if (pctStm < 100) {
        member("Message Line").text = strStm + " of " + strSS+ " bytes have downloaded so far.";
        _ movie.go(_movie.frame);
    }
}
```

streamSize (3D)

使用法

```
member(whichCastmember).streamSize
```

解説

3D プロパティ。ダウンロードするデータストリームのサイズを示します。値の範囲は、0 ～ maxInteger です。このコマンドで取得できる値は、最初のファイルの読み込みの際の値か、または要求した最後の loadFile() の値です。

例文

次のステートメントは、キャストメンバ " シーン " に関連してロードされたファイルの合計サイズが 325300 バイトであることを示します。

```
put member("Scene").streamSize
-- 325300
```

関連事項

bytesStreamed (3D), percentStreamed (3D), state (3D), preLoad (3D)

strokeColor

使用法

```
-- Lingo syntax
memberObjRef.strokeColor

// JavaScript syntax
memberObjRef.strokeColor;
```

解説

シェイプのフレームストロークの RGB によるカラーを表すための、ベクトルシェイプキャストメンバプロパティです。

完成したムービーの strokeColor の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次の例は、キャストメンバ "line" の strokeColor を赤に設定します。

```
-- Lingo syntax
member("line").strokeColor = color(255, 0, 0)

// JavaScript syntax
member("line").strokeColor = color(255, 0, 0);
```

関連事項

color(), fillColor, endColor, backgroundColor

strokeWidth

使用法

```
-- Lingo syntax
memberObjRef.strokeWidth

// JavaScript syntax
memberObjRef.strokeWidth;
```

解説

シェイプのフレームストロークのピクセルによる幅を表すための、ベクトルシェイプキャストメンバプロパティです。

この値は 0 ～ 100 までの浮動小数で、調べたり、変更することができます。

完成したムービーの strokeWidth の実例は、Director アプリケーションフォルダの "Learning/Lingo_Examples/" フォルダ内の "vector shapes.dir" を参照してください。

例文

次のステートメントは、キャストメンバ "line" の strokeWidth を 10 ピクセルに設定します。

```
-- Lingo syntax
member("line").strokeWidth = 10

// JavaScript syntax
member("line").strokeWidth = 10;
```

style

使用法

```
member(whichCastmember).model(whichModel).toon.style
member(whichCastmember).model(whichModel).shader.style
member(whichCastmember).shader(whichShader).style
```

解説

toon モディファイアおよび painter シェーダの 3D プロパティ。toon モディファイアと painter シェーダがモデルにカラーを適用する方法を示します。使用できる値は以下のとおりです。

- #toon。カラーの間のシャープトランジションを使用します。
- #gradient。カラーの間のソフトトランジションを使用します。デフォルト値です。
- #blackAndWhite。黒と白の 2 色を使用します。

toon モディファイアおよび painter シェーダが使用するカラーの数は、モディファイアまたはシェーダの colorSteps プロパティによって設定されます。

例文

次のステートメントは、" ティーポット " というモデルの toon モディファイアの style プロパティを #blackAndWhite に設定します。これにより、このモデルは黒と白の 2 色でレンダリングされます。

```
member("Shapes").model("Teapot").toon.style = #blackAndWhite
```

subdivision

使用法

```
member(whichCastmember).model(whichModel).sds.subdivision
```

解説

3D sds モディファイアプロパティ。このプロパティを使用すると、処理のサブディビジョンサーフェスモードを取得または設定できます。使用できる値は以下のとおりです。

- #uniform 。メッシ は細部にわたって均一に拡大され、各フェイスが同じ倍率で細分化されます。
- #adaptive。サーフェスの向きに大きな変更があった場合に、現在表示可能なメッシュの領域にのみ詳細が追加されます。

sds モディファイアは、inker モディファイアまたは toon モディファイアと共には使用できません。また、sds モディファイアを lod モディファイアと共に使用するときは、注意が必要です。詳細については、sds モディファイアのエントリを参照してください。

例文

次のステートメントは、" 赤ちゃん " というモデルの sds モディファイアの subdivision プロパティを #adaptive に設定します。" 赤ちゃん " のジオメトリは不均等に変更されます。

```
-- Lingo
member("Scene").model(1).sds.subdivision = #adaptive

// Javascript
Member("Scene").getPropRef("model",1).sds.subdivision = symbol("adaptive");
```

関連事項

sds （モディファイア）, error, enabled (sds), depth (3D), tension

subPicture

使用法

```
-- Lingo syntax
dvdObjRef.subPicture

// JavaScript syntax
dvdObjRef.subPicture;
```

解説

DVD プロパティ。サブピクチャがある場合、現在のサブピクチャを指定します。読み取り / 書き込み可能。

subPicture の値は整数です。0 を指定すると subPicture は無効になります。

関連事項

DVD

subPictureCount

使用法

```
-- Lingo syntax
dvdObjRef.subPictureCount

// JavaScript syntax
dvdObjRef.subPictureCount;
```

解説

DVD プロパティ。利用可能なサブピクチャの数を返します。読み取り専用。

関連事項

DVD

suspendUpdates

使用法

```
sprite(which3dSprite).suspendUpdates
```

解説

3D スプライトプロパティ。TRUE の場合、そのスプライトは通常の画面再描画処理で更新されなくなります。これによって、ムービー再生のパフォーマンスが改善することがあります。他のウィンドウをドラッグしてスプライトの上に重ねる操作など、画面更新の種類によっては、このプロパティの値にかかわらずスプライトに影響を与えます。suspendUpdates プロパティが FALSE に設定されている場合は、スプライトは通常どおり再描画されます。

3D スプライト内にアニメーションするエレメントがある場合は、suspendUpdates プロパティを FALSE にしておくことが重要です。

例文

次のステートメントは、スプライト（3）の更新を一時停止します。

```
-- Lingo
sprite(3).suspendUpdates = FALSE

// Javascript
sprite(3).suspendUpdates = false;
```

switchColorDepth

使用法

```
-- Lingo syntax
_player.switchColorDepth

// JavaScript syntax
_player.switchColorDepth;
```

解説

ムービーをロードするときに、ステージを表示するモニタの色深度を Director によって切り替えるか（TRUE）、切り替えないか（FALSE）定義するためのプレーヤプロパティです。読み取り / 書き込み可能。

例え switchColorDepth が TRUE になっていても、新たなムービーがロードされるまで何も起こりません。

モニタの色深度をムービーに合わせて切り替えられると、次のような利点があります。

- モニタの色深度がムービーの色深度よりも低い場合は、ムービーと同じ色深度に再設定することで（設定する色深度にモニタが対応していることが前提）、ムービー本来の色で再生できます。
- モニタの色深度がムービーの色深度よりも高い場合は、モニタの色深度を下げることで、最小限のメモリでムービーを再生できます。また、キャストメンバのロードが効率化され、アニメーションの開始が早くなります。

このプロパティの値は、環境設定一般ダイアログボックスの「ムービーの色深度にモニタをリセット」オプションを使用して設定することもできます。

例文

次のステートメントは、変数 switcher を、switchColorDepth の現行設定と同じものに設定します。

```
-- Lingo syntax
switcher = _player.switchColorDepth

// JavaScript syntax
var switcher = _player.switchColorDepth;
```

次のステートメントは、現行の色深度が 8 ビットになっているかどうかをチェックし、8 ビットの場合は switchColorDepth プロパティをオンにします。

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    _player.switchColorDepth = TRUE
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    _player.switchColorDepth = true;
}
```

関連事項

colorDepth, プレーヤ

systemTrayIcon

使用法

```
-- Lingo syntax
_movie.displayTemplate.systemTrayIcon
windowObjRef.systemTrayIcon

// JavaScript syntax
_movie.displayTemplate.systemTrayIcon;
windowObjRef.systemTrayIcon;
```

解説

ムービーおよび Windows のプロパティです（Microsoft Windows のみ）。ユーザのデスクトップのシステムトレイのウィンドウにアイコンを使用するかどうかを指定します。読み取り / 書き込み可能。

systemTrayIcon を TRUE に設定すると、システムトレイにウィンドウのアイコンが配置されます。

systemTrayIcon を FALSE に設定すると、システムトレイにウィンドウのアイコンは配置されません。

例文

次のステートメントは、ムービーの再生中にシステムトレイアイコンを表示します。

```
-- Lingo
_movie.displayTemplate.systemTrayIcon = TRUE

// Javascript
_movie.displayTemplate.systemTrayIcon = true;
```

関連事項

displayTemplate, ムービー , systemTrayTooltip, ウィンドウ

systemTrayTooltip

使用法

```
-- Lingo syntax
_movie.displayTemplate.systemTrayTooltip
windowObjRef.systemTrayTooltip

// JavaScript syntax
_movie.displayTemplate.systemTrayTooltip;
windowObjRef.systemTrayTooltip;
```

解説

ムービーおよび Windows のプロパティです（Microsoft Windows のみ）。システムトレイアイコンのヒントに表示されるストリングを指定します。読み取り / 書き込み可能。

このプロパティは、systemTrayIcon プロパティが TRUE に設定されているときにのみ適用可能です。systemTrayIcon を TRUE に設定すると、システムトレイアイコン上にユーザがマウスを置いたときに、ヒントが表示されます。

systemTrayTooltip のデフォルト値は、ウィンドウのタイトルです。

関連事項

displayTemplate, ムービー , systemTrayIcon, ウィンドウ

tabCount

使用法

```
chunkExpression.tabCount
```

解説

テキストキャストメンバの指定されたチャンク式にある一意のタブ停止位置の個数を表すテキストキャストメンバプロパティです。

この値は 0 以上の整数で、値は調べられますが、変更できません。

例文

次のステートメントは、テキストフィールド内のタブの数を示します。

```
-- Lingo
put member(1).tabCount
-- 3

// Javascript
trace(member(1).tabCount);
// 3
```

tabs

使用法

```
member(whichTextMember).tabs
```

解説

テキストキャストメンバプロパティ。このプロパティには、テキストキャストメンバに設定されているすべてのタブ停止位置のリストが含まれます。

リストの各エレメントには、そのテキストキャストメンバのタブに関する情報が入っています。リストにあるプロパティは次のとおりです。

#type	#left、#center、#right、#decimal のいずれか
#position	タブの位置をポイントで表す整数値

このプロパティは取り出して設定することができます。tabs を設定する場合、type プロパティはオプションです。type を指定しない場合のデフォルトは #left です。

例文

次のステートメントは、テキストキャストメンバ "Intro credits" のすべてのタブを取得し、メッセージウィンドウに表示します。

```
-- Lingo
put member("Intro credits").tabs
-- [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right, #position: 216]]

// Javascript
trace(member("Intro credits").tabs)
// < [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right, #position: 216]]>
```

target

使用法

```
timeoutObject.target
```

解説

指定された [タイムアウトオブジェクト] がタイムアウトイベントを送信する宛先になる子オブジェクトを示します。ターゲットプロパティが VOID であるタイムアウトオブジェクトは、イベントをムービースクリプトにあるハンドラに送信します。

このプロパティは、タイムアウトオブジェクトを使用するビヘイビアーおよび親スクリプトのデバッグに役立ちます。

例文

次のステートメントでは、"timerOne" という名前のタイムアウトオブジェクトからタイムアウトイベントを受け取る子オブジェクトの名前をメッセージウィンドウに表示します。

```
-- Lingo
put timeout("timerOne").target

// Javascript
trace(timeout("timerOne").target)
```

関連事項

name (timeout), timeout(), timeoutHandler, timeoutList

targetFrameRate

使用法

```
sprite(which3dSprite).targetFrameRate
```

解説

3D スプライト。3D をレンダリングするときに使用する 1 秒あたりのフレーム数を示します。デフォルトの値は、1 秒間あたり 30 フレームです。useTargetFrameRate が TRUE に設定されている場合は、targetFrameRate プロパティだけが使用されます。useTargetFrameRate プロパティが TRUE に設定されている場合は、指定されたフレームレートを維持するために、Director はモデルのポリゴン数を減らします。

例文

次のステートメントは、スプライト 3 の targetFrameRate プロパティを 45 に設定します。また、スプライトの useTargetFrameRate プロパティを TRUE に設定することにより、このフレームレートを強制します。

```
-- Lingo
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// Javascript
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

関連事項

useTargetFrameRate

tension

使用法

```
member(whichCastmember).model(whichModel).sds.tension
```

解説

3D サブディビジョンサーフェスプロパティ。このプロパティを使用すると、新しく作成したサーフェスをオリジナルのサーフェスと一致させる精度を示す浮動小数の比率 (0.0 ～ 100.0) を取得または設定できます。この値が大きいほど、細分化されたサーフェスが元のサーフェスに一致する度合いが高くなります。デフォルトは 65.0 です。

例文

次のステートメントは、モデル " 赤ちゃん " の sds モディファイアの tension プロパティを 35 に設定します。sds モディファイアの error 設定が低く、depth 設定が高い場合は、次のステートメントは " 赤ちゃん " のジオメトリに非常に大きな効果を与えます。

```
-- Lingo
member("scene").model("Baby").sds.tension = 35

// Javascript
member("scene").getPropRef("model",1).sds.tension = 35;
```

関連事項

sds（モディファイア）, error, depth (3D)

text

使用法

```
-- Lingo syntax
memberObjRef.text

// JavaScript syntax
memberObjRef.text;
```

解説

[キャストメンバ] で指定したフィールドキャストメンバに含まれる文字ストリングを特定するためのテキストキャストメンバプロパティです。

text キャストメンバプロパティは、メッセージを表示したり、ユーザの入力内容を記録するときに役立ちます。

このプロパティは、値を調べたり、設定することができます。

キャストメンバ内のテキスト全体を Lingo によって変更すると、個々の単語や行に適用していた書式設定は失われます。text キャストメンバプロパティを変更すると、グローバル書式が適用されることになります。テキスト内の行、単語、または項目を指定することにより、テキストの特定部分のみを変更することができます。

ムービーをアプレットとして再生する場合、テキストをまだストリーミングしていないフィールドキャストメンバに対して、このプロパティの値は ""（空白のストリング）になります。

完成したムービーの text の実例は、Director アプリケーションフォルダの "Learning\\Lingo_Examples\\" フォルダ内の "forms and post.dir" と "text.dir" を参照してください。

例文

次のステートメントは、" ありがとうございました。" というフレーズを " 応答 " という空のキャストメンバに挿入します。

```
--Lingo syntax
if (member("Response").text = EMPTY) then
    member("Response").text = "Thank You."
end if

// JavaScript syntax
if (member("Response").text == " ") {
    member("Response").text = "Thank You.";
}
```

次のステートメントは、キャストメンバ " メッセージ " の内容を " 正解です !" に設定します。

```
--Lingo syntax
member("Notice").text = "You have made the right decision!"

// JavaScript syntax
member("Notice").text = "You have made the right decision!";
```

関連事項

selEnd, selStart

texture

使用法

```
member(whichCastmember).texture(whichTexture)
member(whichCastmember).texture[index]
member(whichCastmember).shader(whichShader).texture
member(whichCastmember).model(whichModel).shader.texture
member(whichCastmember).model(whichModel).shaderList.texture
member(whichCastmember).model(whichModel).shaderList[index].texture
member(whichCastmember).modelResource(whichParticleSystemModelResource).texture
```

解説

3D エレメントおよびシェーダのプロパティ。モデルのサーフェスの外観を定義するシェーダで使用される画像オブジェクトです。画像は、シェーダによってモデルのジオメトリ上にラップされます。

シェーダの表示コンポーネントは、最大 8 レイヤーのテクスチャを使って作成されます。これら 8 つのテクスチャレイヤーは、Director 内のビットマップキャストメンバか画像オブジェクトから作成されるか、または 3D モデリングプログラムのモデルを使って読み込みます。

テクスチャの作成と削除は、newTexture() および deleteTexture() コマンドで行います。

テクスチャは、3D キャストメンバのテクスチャパレットに格納されます。テクスチャは、名前 (テクスチャ) またはパレットインデックス (テクスチャインデックス) で参照できます。1 つのテクスチャを複数のシェーダで使用できます。テクスチャへの変更は、そのテクスチャを使用するすべてのシェーダに反映されます。

テクスチャには以下の 3 つの種類があります。

#fromCastmember。テクスチャは、newTexture() コマンドを使用してビットマップキャストメンバから作成されます。

#fromImageObject。テクスチャは、newTexture() コマンドを使って Lingo 画像オブジェクトから作成されます。

#importedFromFile。テクスチャは、3D モデリングプログラムからモデルと共に読み込まれます。

テクスチャのプロパティについて詳しくは、Director ヘルプパネルの「Director の使い方」トピックを参照してください。

パーティクルシステムのテクスチャは、#particle というタイプのモデルリソースのプロパティです。

例文

次のステートメントは、"WallSurface" というシェーダの texture プロパティを "BluePaint" というテクスチャに設定します。

```
-- Lingo
member("Room").shader("WallSurface").texture = member("Room").texture("BluePaint")

// Javascript
member("Room").getPropRef("shader",1).texture = member("Room").getPropRef("texture",1);
```

関連事項

newTexture, deleteTexture

textureCoordinateList

使用法

```
member(whichCastmember).modelResource(whichmodelResource).
textureCoordinateList
modelResourceObjectReference.textureCoordinateList
```

解説

3D プロパティ。#mesh タイプのモデルリソース、またはモデルに添付された meshDeform モディファイアと共に使用すると、そのモデルリソースの textureCoordinateList プロパティを取得または設定できます。

textureCoordinateList プロパティは、テクスチャを三角形にマッピングする場合に使用する、画像の位置を示すサブリストのリストです。各サブリストは、テクスチャマップ内の位置を示す 2 つの値で構成されます。値は 0.0 ～ 1.0 にする必要があります。任意のサイズのテクスチャマップにスケールできます。デフォルトは空のリストです。

textureCoordinates とメッシュフェイスの角の間のマッピングを変更するには、modelResource.face[index].textureCoordinates または model.meshdeform.mesh[index].face[index] を操作します。

例文

```
-- Lingo
put member(5).modelResource("mesh square").textureCoordinateList
--[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3], [0.2, 0.3], [0.3,
0.3] ]

// Javascript
trace(member(5).getPropRef("modelResource",1).textureCoordinateList;
// <[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3], [0.2, 0.3], [0.3,
0.3] ]>
```

関連事項

face, texture, meshDeform （モディファイア）

textureCoordinates

使用法

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].textureCoordinates
modelResourceObject.face[faceIndex].textureCoordinates
```

解説

3D プロパティ。textureCoordinateList 内のどのエレメントを「フェイスインデックス」で指定したフェイスに使用するかを示します。このプロパティは、textureCoordinateList のインデックスを示す 3 つの整数のリストであることが必要です。これらは、メッシュフェイスのそれぞれのコーナーで使用する textureCoordinates に対応します。

関連事項

face, textureCoordinateList

textureLayer

使用法

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer.add()
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer[index].textureCoordinateLis
t.
```

解説

3D meshdeform モディファイアプロパティ。これらのプロパティを使用すると、指定されたメッシュのテクスチャレイヤーに関する情報を取得および設定できます。

1 つのシェーダには 8 までのテクスチャレイヤーを設定できます。各レイヤーが保有するテクスチャは 1 つのみですが、同じテクスチャを複数のレイヤーに指定できます。テクスチャレイヤーは、シェーダで使用されるテクスチャのレイヤーです。

テクスチャレイヤーをアクセスおよび操作するには、以下のプロパティを使用します。

meshdeform.mesh[index].texturelayer.count。指定されたメッシュのテクスチャレイヤー数を返します。

model.meshdeform.mesh[index].texturelayer.add()。指定されたメッシュに空のテクスチャレイヤーを追加します。

model.meshdeform.mesh[index].texturelayer[index].texturecoordinatelist。指定されたメッシュの特定のレイヤーに textureCoordinates のリストを設定または取得します。また、このプロパティを使って、次のようにテクスチャレイヤー間でテクスチャ座標をコピーできます。

```
model.meshdeform.texturelayer[a].texturecoordinatelist =
model.meshdeform.texturelayer[b].texturecoordinatelist
```

関連事項

meshDeform （モディファイア）, mesh （プロパティ）, textureCoordinateList, add (3D texture), count, texture, textureModeList

textureList

使用法

```
member(whichMember).model(whichModel).shader(whichShader).textureList
member(whichMember).model(whichModel).shader(whichShader).textureList[index]
```

解説

3D シェーダプロパティ。シェーダに適用されるテクスチャのリストを示します。1 つのシェーダは、最大 8 レイヤーのテクスチャを使用できます。値を調べると、このプロパティはテクスチャオブジェクトのリニアリストを返します。インデックスを省略して設定すると、このプロパティはテクスチャオブジェクトをすべてのレイヤーに適用するように設定します。textureList プロパティを VOID に設定すると、すべてのレイヤーでテクスチャが無効になります。デフォルト値は VOID です。

特定のテクスチャレイヤーのテクスチャオブジェクトを取得または設定するには、インデックス値を指定します。

例文

次のステートメントは、"WallSurface" というシェーダの 3 番目のテクスチャレイヤーを "BluePaint" というテクスチャに設定します。このテクスチャは、キャストメンバ " 部屋 " に含まれています。

```
-- Lingo
member(3).model("Car").shader("WallSurface").textureList[3] = member("Room").texture("BluePaint")

// Javascript
member(3).getPropRef("model",1).getPropRef("shader",1).textureList[3] =
member("Room").getPropRef("texture",1);
```

関連事項

textureModeList

textureMember

使用法

```
member(whichCastmember).textureMember
```

解説

3D キャストメンバプロパティ。3D キャストメンバのデフォルトテクスチャのソースとして使用されるビットマップキャストメンバの名前を示します。

textureMember プロパティを有効にするために、3D キャストメンバの textureType プロパティを #member に設定する必要があります。

例文

次のステートメントは、"YardScene" というキャストメンバの textureMember プロパティを "Fence" に設定します。"YardScene" の textureType プロパティが #member に設定されていれば、キャストメンバ「フェンス」が、"YardScene" のデフォルトテクスチャのビットマップになります。

```
-- Lingo
member("YardScene").textureMember = "Fence"

// Javascript
member("YardScene").textureMember = "Fence";
```

関連事項

textureType

textureMode

使用法

```
member(whichCastmember).shader(whichShader).textureMode
member(whichCastmember).model(whichModel).shader.textureMode
member(whichCastmember).model(whichModel).shaderList{[index]}.textureMode
```

解説

3D #standard シェーダプロパティ。最初のテクスチャレイヤーがモデルのサーフェス上にどのようにマップされるかを指定します。最初のレイヤー以外のレイヤーのテクスチャを指定するには、textureModeList プロパティを使用します。#toon モディファイアがモデルリソースに適用される場合、このプロパティは無視されます。

このプロパティは、#none、#wrapPlanar、#wrapCylindrical、#wrapSpherical、#reflection、#diffuseLight、および #specularLight のいずれかの値になります。詳細については、「textureModeList」を参照してください。

例文

次のステートメントは、" ボール " というモデルのシェーダの最初のテクスチャレイヤーの textureMode プロパティの値を #wrapSpherical に設定します。

```
-- Lingo
member("scene").model("Ball").shader.textureMode = #wrapSpherical

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureMode = symbol("wrapSpherical");
```

関連事項

textureModeList

textureModeList

使用法

```
member(whichCastmember).shader(whichShader).textureModeList
member(whichCastmember).shader(whichShader).
textureModeList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureModeList
member(whichCastmember).model(whichModel).shader.
textureModeList[textureLayerndex]
```

解説

3D 標準シェーダプロパティ。textureLayer がモデルのサーフェス上にどのようにマップされるかを変更できます。#toon モディファイアがモデルリソースに適用される場合、このプロパティは無視されます。使用できる値は以下のとおりです。

- #none。モデルリソースに元々定義されていたテクスチャ座標値を使用します。この設定によって、wrapTransform and wrapTransformList[textureLayerIndex] が無効になります。
- #wrapPlanar。オーバーヘッドプロジェクタによって投影されているかのように、モデルサーフェスをテクスチャで覆います。シェーダの wrapTransformList[textureLayerIndex] は、テクスチャ座標がモデルスペースで生成される前に、マッピングスペースに適用されます。wrapTransformList[textureLayerIndex] （デフォルト）を指定すると、テクスチャの上方向が Y 軸に沿った状態で、Z 軸に沿ってテクスチャが押し出されるように平面マッピングが実行されます。
- #wrapCylindrical。サーフェスがテクスチャの中央に置かれており、テクスチャが円柱を形成するようにサーフェスに巻き付けらているかのように、サーフェスをテクスチャで覆います。wrapTransformList[textureLayerIndex] は、テクスチャ

座標がモデルスペースで生成される前に、マッピングスペースに適用されます。恒等の wrapTransformList[textureLayerIndex] (デフォルト) を指定すると、-Y 軸から +X 軸の方へ Z 軸を中心にテクスチャで覆うように円柱マッピングがテクスチャの左端から実行されます。テクスチャの上方向は、+Z 軸の方向に向いています。

- #wrapSpherical。サーフェスがテクスチャの中央に置かれており、テクスチャの 4 つのコーナーが最上部で合わさるように引き上げられているかのように、サーフェスをテクスチャで覆います。wrapTransformList[textureLayerIndex] は、テクスチャ座標がモデルスペースで生成される前に、マッピングスペースに適用されます。恒等の wrapTransformList[textureLayerIndex] を指定すると、球体マッピングは、モデルスペースの原点に配置され、-Y 軸から +X 軸の方へ Z 軸を中心にテクスチャで覆うようにテクスチャの左端から実行されます。テクスチャの上方向は、+Z 軸の方向に向いています。
- #reflection。#wrapSpherical と似ていますが、新しいテクスチャ座標は一定の方向からサーフェスに連続的に投影されます。モデルが回転しても、テクスチャ座標は同時に回転しません。環境ライトによってオブジェクトに反射されるライトをシミュレートします。この設定によって、wrapTransform が無効になります。
- #diffuseLight。各頂点に対して拡散ライトマッピングテクスチャ座標値を作成し、その結果を参照されたメッシュ内に格納します。この設定によって、wrapTransform が無効になります。
- #specularLight。各頂点に対して光沢ライトマッピングテクスチャ座標値を作成し、その結果を参照されたメッシュ内に格納します。この設定によって、wrapTransform が無効になります。

例文

次の例では、シェーダは反射するガーデンボールをシミュレートするよう設定されます。シェーダの最初の textureLayer は球体マッピングに設定され、3 番目の textureLayer は #refection スタイルマッピングを使用するよう設定されます。シェーダの textureList[3] エントリは、このシェーダを使用するすべてのモデルに環境ライトから反射するように表示されます。

```
-- Lingo
member("scene").shader("GardenBall).textureList[1] = member("scene").texture("FlatShinyBall")
member("scene").shader("GardenBall").textureModeList[1] = #wrapSpherical
member("scene").shader("GardenBall").textureList[3] = member("scene").texture("GardenEnvironment")
member("scene").shader("GardenBall").textureModeList[3] = #reflection

// Javascript
member("scene").getPropRef("shader",1).textureList[1] = member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[1] = symbol("wrapSpherical");
member("scene").getPropRef("shader",1).textureList[3] = member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[3] = symbol("reflection");
```

関連事項

textureTransformList, wrapTransform

textureRenderFormat

使用法

```
getRenderServices().textureRenderFormat
```

解説

3D rendererServices プロパティ。すべての 3D キャストメンバ内のすべてのテクスチャで使用されるデフォルトのビットフォーマットを取得または設定できます。テクスチャの textureRenderFormat プロパティを使用して、特定のテクスチャのみについてこの設定を無効にします。ビットフォーマットのサイズが小さい（#rgba5551 のような 16 ビットバリアント）場合は、ハードウェアアクセラレーターのビデオ RAM 使用量が少なくてすみます。このため、ソフトウェアレンダリングに強制的に切り替わることなく利用できるテクスチャの数が多くなります。一般に、ビットフォーマットのサイズが大きいほ

ど（#rgba8888 のような 32 ビットバリアント）、外観が良くなります。テクスチャでアルファ透明度を使用するには、最終ビットは 0 以外であることが必要です。滑らかな透明度グラデーションを得るには、アルファチャネルに 2 ビット以上の精度が必要です。

各ピクセルフォーマットは 4 桁で構成され、各桁は、赤、緑、青、およびアルファの精度の度合いを示します。選択した値により、カラーの忠実度の精度 (アルファチャネルの精度) ハードウェアテクスチャバッファで使用されるメモリ量が決まります。カラーの忠実度を向上させるか、またはカード上により多くのテクスチャを読み込ませるかといういずれかの目的に合わせて値を選択できます。メモリ容量が同じであれば、32 ビットテクスチャの倍の数の 16 ビットテクスチャを読み込ませることができます。カードに 1 度で読み込めるよりも多くのテクスチャをムービーで使用すると、Director によって自動的に #software レンダリングに切り替えられます。

textureRenderFormat には、次の値を指定できます。

- #rgba8888：32 ビットカラーモード。赤、緑、青、アルファに 8 ビットずつ。
- #rgba8880：アルファなしの 32 ビットカラーモード。赤、緑、青に 8 ビットずつ。
- #rgba5650：アルファなしの 16 ビットカラーモード。赤に 5 ビット、緑に 6 ビット、青に 5 ビット。
- #rgba5550：アルファなしの 16 ビットカラーモード。アルファ指標なしで赤、緑、青に 5 ビットずつ。
- #rgba5551：16 ビットカラーモード。赤、緑、青に 5 ビットずつ、アルファに 1 ビット。
- #rgba4444：16 ビットカラーモード。赤、緑、青、アルファに 4 ビットずつ。

デフォルト値は #rgba5551 です。

例文

次のステートメントは、3D メンバのグローバル textureRenderFormat を #rgba8888 に設定します。このムービーの各テクスチャは、そのテクスチャを除き、32 ビットカラーでレンダリングされます。renderFormat プロパティは、#default 以外の値に設定されます。

```
-- Lingo
getRendererServices().textureRenderFormat = #rgba8888

// Javascript
getRendererServices().textureRenderFormat = symbol("rgba8888");
```

関連事項

renderer, preferred3dRenderer, renderFormat, getRendererServices()

textureRepeat

使用法

```
member(whichCastmember).shader(whichShader).textureRepeat
member(whichCastmember).model(whichModel).shader.textureRepeat
member(whichCastmember).model(whichModel).shaderList{[index]}.textureRepeat
```

解説

3D #standard シェーダプロパティ。シェーダの最初のテクスチャレイヤーのテクスチャクランピングの方法を制御します。最初のレイヤー以外のテクスチャレイヤーに対してこのプロパティを指定するには、textureRepeatList プロパティを使用します。

textureRepeat が TRUE に設定され、shaderReference.textureTransform.scale の x コンポーネントや y コンポーネントの値が 1 未満である場合、テクスチャはモデルのサーフェスをタイル状に (反復的に) 覆います。

textureRepeat が FALSE に設定されている場合、テクスチャはタイル状になりません。shaderReference.textureTransform.scale の x または y コンポーネントの値が 1 未満の場合は、そのテクスチャで覆われないモデルの領域は黒になります。shaderReference.textureTransform.scale の x または y コンポーネントの値が 1 を超えている場合は、テクスチャ座標範囲を超えて拡張され、余りのテクスチャは切り取られます。

このプロパティのデフォルト値は TRUE です。#software レンダラを使用するときは、このプロパティは常に TRUE になります。

例文

次のステートメントは、"gbCyl3" というモデルで使用される最初のシェーダの textureRepeat プロパティを TRUE に設定します。シェーダの textureTransform または textureTransformList プロパティの x または y コンポーネントの値が 1 未満の場合は、そのシェーダの最初のテクスチャはタイル状になります。

```
-- Lingo
member("scene").model("gbCyl3").shader.textureRepeat = TRUE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureRepeat = true;
```

関連事項

textureTransform, textureTransformList

textureRepeatList

使用法

```
shaderReference.textureRepeatList[textureLayerIndex]
member(whichCastmember).shader(whichShader).textureRepeatList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].
textureRepeatList[textureLayerIndex]
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、任意のテクスチャレイヤーのテクスチャクランピングの方法を取得または設定できます。TRUE (デフォルト) の場合、textureLayerIndex のテクスチャはモデルサーフェスを何度もタイル状 (反復的) に覆います。これは、shaderReference.textureTransform[textureLayerIndex].scale の x または y を 1 未満に設定することで実現できます。shaderReference.textureTransform[textureLayerIndex].scale の x または y が 1 未満である場合、この値を FALSE に設定すると、テクスチャはモデルサーフェスをタイル状に覆うよりも小さな部分に適用されます。元の画像のフレーム内でソース画像が縮小し、そのすき間が黒で埋まっていると考えてください。同様に、shaderReference.textureTransform[textureLayerIndex].scale の x または y が 1 を超える場合、画像はテクスチャ座標範囲を超えて拡張され、超えた部分のテクスチャは切り取られます。

例文

次のコードは、球体 "mySphere" 全体のサーフェスを「花崗岩」テクスチャで 4 回繰り返し textureMap し、サーフェスの 4 分の 1 をカバーするロゴ画像をテクスチャマップします。

```
-- Lingo
m = member(2).model("mySphere")
f = member(2).newTexture("granite", #fromCastmember, member("granite"))
g = member(2).newTexture("logo", #fromCastmember, member("logo"))
s = member(2).newShader("s", #standard)
s.textureList[1] = g
s.textureList[2] = f
s.textureRepeatList[2] = false
s.textureRepeatList[1] = true
s.textureTransformList[1].scale(0.5,0.5,1.0)
s.textureTransformList[2].scale(0.5,0.5,1.0)
s.textureModeList[2] = #wrapPlanar
s.blendFunctionList[2] = #add
m.shaderList = s

// Javascript
m = member(2).getPropRef("model","1");
f = member(2).newTexture("granite", symbol("fromCastmember"), member("granite"));
g = member(2).newTexture("logo", symbol("fromCastmember"), member("logo"));
s = member(2).newShader("s", symbol("standard"));
s.textureList[1] = g;
s.textureList[2] = f;
s.textureRepeatList[2] = false;
s.textureRepeatList[1] = true;
s.textureTransformList[1].scale(0.5,0.5,1.0);
s.textureTransformList[2].scale(0.5,0.5,1.0);
s.textureModeList[2] = symbol("wrapPlanar");
s.blendFunctionList[2] = symbol("add");
m.shaderList = s;
```

textureTransform

使用法

```
member(whichCastmember).shader(whichShader).textureTransform
member(whichCastmember).model(whichModel).shader.textureTransform
member(whichCastmember).model(whichModel).shaderList{[index]}.textureTransform
```

解説

3D #standard シェーダプロパティ。シェーダの最初のテクスチャレイヤーのテクスチャ座標マッピングを修正するトランスフォームにアクセスできます。テクスチャをモデルのサーフェスに適用する前に、このトランスフォームを操作してテクスチャをタイル化、回転、または移動できます。テクスチャ自体は影響を受けません。トランスフォームは、シェーダのテクスチャへの適用方法をほとんど変更しません。textureTransform プロパティは、textureMode プロパティの設定に関係なく、すべてのテクスチャ座標に適用されます。これは、テクスチャ座標がレンダラに送られる前の最後の変更です。textureTransform プロパティは、textureImage スペースのテクスチャに対して機能するマトリックスです。TextureImage スペースは、X、Y 平面にのみ存在するように定義されます。

水平軸に沿って画像を 2 回タイル状にするには、shaderReference.textureTransform.scale(0.5, 1.0, 1.0) を使用します。Z 軸に対するスケーリングは無視されます。

point(xOffset,yOffset) で画像をオフセットするには、shaderReference.textureTransform.translate(xOffset,yOffset,0.0) を使用します。シェーダの textureRepeat プロパティが TRUE の場合は、整数による移動は効果がありません。この場合は、テクスチャの幅と高さが 0.0 ～ 1.0 の値に設定されるからです。

テクスチャレイヤーに回転を適用するには、shaderReference.textureTransform.rotate(0,0,angle) を使用します。Z 軸を中心とした回転は、テクスチャの左上隅にマッピングされた 2D 画像ポイント (0,0) を中心として実行されます。X 軸および Y 軸を中心とした回転は無視されます。

モデルのトランスフォームに関してのみ、textureTransform の変更は重複できます。point(0,0) の代わりに point(xOffset,yOffset) を中心としてテクスチャを回転させるには、まず point(0 - xOffset, 0 - yOffset) に移動し、回転してから、point(xOffset, yOffset) に移動します。textureTransform は、シェーダの wrapTransform プロパティと似ていますが、次の違いがあります。3D ワールドスペースではなく、2D 画像スペースで適用されます。結果として、Z 軸を中心とした回転と、X 軸および Y 軸に対する移動とスケールは有効です。トランスフォームは、shaderReference.textureMode 設定とは無関係に適用されます。一方、wrapTransform は、textureMode が #wrapPlanar、#wrapCylindrical、または #wrapSpherical の場合にのみ有効です。

例文

次のステートメントは、モデル "gbCyl3" で使用される最初のシェーダの最初のテクスチャの textureTransform を表示します。

```
-- Lingo
put member("Scene").model("gbCyl3").shader.textureTransform
-- transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000,
0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("Scene").getPropRef("model",1).getProp("shader").textureTransform);
// <transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000,
0.0000, 0.0000, 0.0000, 1.0000)>
```

次のステートメントは、"gbCyl3" というシェーダで使用される最初のテクスチャの高さと幅を半分にします。gbCyl3 の textureRepeat プロパティが TRUE に設定されている場合は、テクスチャの 4 つのコピーがタイル状にシェーダを覆います。

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.scale = vector(0.5, 0.5, 1)

// Javascript
member("Scene").getPropRef("shader",1).textureTransform.scale = vector(0.5,0.5,1);
```

次のステートメントは、シェーダ "gbCyl3" で使用される最初のテクスチャを vector(0, 0, 0) から 90° 回転させます。

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.rotation = vector(0, 0, 90)

// Javascript
member("Scene").getPropRef("shader",1).textureTransform.rotation = vector(0,0,90);
```

textureTransformList

使用法

```
shaderReference .textureTransformList[textureLayerIndex]
member(whichCastmember).shader(ShaderName).textureTransformList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shader.textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shaderList[shaderListIndex].
textureTransformList[textureLayerIndex]
```

解説

3D 標準シェーダプロパティ。このプロパティを使用して、テクスチャレイヤーのテクスチャ座標マッピングを変更するトランスフォームにアクセスできます。テクスチャ画像をモデルのサーフェスに適用する前に、このトランスフォームを操作してテクスチャをタイル化、回転、または移動できます。テクスチャ自体は影響を受けません。トランスフォームは、シェーダのテクスチャへの適用方法をほとんど変更しません。

水平軸に沿って画像を 2 回タイル状に表示するには、textureTransformList[whichTextureLayer].scale(0.5, 1.0, 1.0) を使用します。画像は本来 2D であるため、Z 方向のスケールは無視されます。0.0 のスケールは、テクスチャ全体を無効にするので、使用しないよう注意が必要です (Z 方向も同様)。

point(xOffset,yOffset) で画像をオフセットするには、textureTransformList[whichTextureLayer].translate(xOffset,yOffset,0.0) を使用します。テクスチャレイヤーの textureRepeat プロパティが TRUE の場合は、整数による移動は効果がありません。この場合は、テクスチャの幅と高さが 0.0 ～ 1.0 までの値に設定されるからです。

テクスチャレイヤーに回転を適用するには、textureTransformList[whichTextureLayer].rotate(0,0,angle) を使用します。Z 軸を中心とした回転は、テクスチャの左上隅にマッピングされた 2D 画像ポイント (0,0) を中心として実行されます。画像は本来 2D であるため、X 軸および Y 軸を中心とした回転は無視されます。

モデルのトランスフォームと同様に、textureTransform の修正は重複できます。point(0,0) の代わりに point(xOffset,yOffset) を中心として画像を回転させるには、まず point(0 - xOffset, 0 - yOffset) に移動し、回転してから、point(xOffset, yOffset) に移動します。

textureTransformList は、シェーダの wrapTransformList プロパティと似ていますが、次の違いがあります。

3D ワールドスペースではなく、2D 画像スペースで適用されます。結果として、Z 軸のみを中心とした回転と、X 軸および Y 軸に対する移動とスケールは有効です。

トランスフォームは、shaderReference.textureModeList[index] 設定とは無関係に適用されます。一方、wrapTransform は、textureMode が #wrapPlanar、#wrapCylindrical、または #wrapSpherical の場合にのみ有効です。

例文

次のステートメントは、モデル "gbCyl3" で使用される最初のシェーダの 3 番目のテクスチャの textureTransform プロパティを表示します。

```
-- Lingo
put member("scene").model("gbCyl3").shader.textureTransformList[3]
-- transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000,
0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("scene").getPropRef("model",1).getProp("shader").textureTransformList[3])
//< transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000,
0.0000, 0.0000, 0.0000, 0.0000, 1.0000)>
```

次のステートメントは、"gbCyl3" というシェーダで使用される 5 番目のテクスチャの高さと幅を半分にします。gbCyl3 の textureRepeatList[5] の値が TRUE に設定されている場合、テクスチャの 4 つのコピーがタイル状にシェーダを覆います。

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[5].scale = vector(0.5, 0.5, 1)

// Javascript
member("scene").getPropRef("shader","1").textureTransformList[5].scale = vector(0.5, 0.5, 1);
```

次のステートメントは、シェーダ "gbCyl3" で使用される 4 番目のテクスチャを vector(0, 0, 0) から 90　回転させます。

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[4].rotation = vector(0, 0, 90)

// Javascript
member("scene").getPropRef("shader","1 ").textureTransformList[4].rotation = vector(0, 0, 90);
```

次のステートメントは、シェーダ "gbCyl3" で使用される 3 番目のテクスチャを、その中央を中心に 90° 回転します。textureList[3] は 128 × 128 サイズのテクスチャであるものとします。

```
-- Lingo
s = member("scene").shader("gbCyl3")
s.textureTransformList[3].translate(-64,-64,0)
s.textureTransformList[3].rotate(0,0,90)
s.textureTransformList[3].translate(64,64,0)

// Javascript
s = member("scene").getPropRef("shader","1");
s.textureTransformList[3].translate(-64,-64,0);
s.textureTransformList[3].rotate(0,0,90);
s.textureTransformList[3].translate(64,64,0);
```

textureType

使用法

```
member(whichCastmember).textureType
```

解説

3D テクスチャプロパティ。デフォルトテクスチャのテクスチャタイプを取得または設定できます。使用できる値は以下のとおりです。

- #none。テクスチャタイプがないことを指定します。
- #default。元のシェーダのテクスチャをテクスチャとして使用します。
- #member。指定されたキャストメンバの画像をテクスチャとして使用します。

このプロパティのデフォルト値は #default です。textureMember プロパティを使用するには、このプロパティの #member を指定する必要があります。

例文

次のステートメントは、キャストメンバ " シーン " の textureType プロパティを #member に設定します。

```
member("Scene").textureType = #member
```

次のステートメントは、textureMember プロパティを設定することで、ビットマップキャストメンバをデフォルトテクスチャのソースとして使用できるようにします。ビットマップキャストメンバの名前は " 芝生 " です。

```
member("Scene").textureMember = "grass"
```

関連事項

textureMember

thumbNail

使用法

```
-- Lingo syntax
memberObjRef.thumbNail

// JavaScript syntax
memberObjRef.thumbNail;
```

解説

キャストウィンドウでキャストメンバのプレビューに使用する画像を含むメンバプロパティです。オーサリング中のみ読み取り / 書き込み可能。

この画像は、すべてのキャストメンバ用にカスタマイズできます。

例文

次のステートメントは、" プレースホルダー " キャストメンバを使用して、別のサムネイルをステージに表示する方法を示します。" プレースホルダー " キャストメンバをステージに配置し、次に、メンバ 10 のサムネイルをそのメンバのピクチャに設定します。これにより、スケールやその他のグラフィック操作をしなくても、縮小した画像を表示できます。

```
-- Lingo syntax
member("Placeholder").picture = member(10).thumbNail

// JavaScript syntax
member("Placeholder").picture = member(10).thumbNail;
```

関連事項

メンバ

tilt

使用法

```
-- Lingo syntax
spriteObjRef.tilt

// JavaScript syntax
spriteObjRef.tilt;
```

解説

QuickTime VR スプライトプロパティ。QuickTime VR ムービーの現行のチルトの傾斜角度を度数で表します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントはスプライト（3）にある現行の QuickTime ムービーのチルトを取得する方法を示します。

```
-- Lingo
putsprite(3).tilt

// Javascript
trace(sprite(3).tilt);
```

time （タイムアウトオブジェクト）

使用法

```
timeoutObject.time
```

解説

タイムアウトオブジェクトプロパティで、指定された [タイムアウトオブジェクト] が次回のタイムアウトイベント送信するシステム時間を、ミリセカンド単位で示します。

これは次回のイベントまでの経過時間ではなく、次回のタイムアウトイベントの絶対時刻です。

例文

次のハンドラは、"Update" タイムアウトオブジェクトが次回のタイムアウトイベントを送信するまでの残り時間を示します。この値は、オブジェクトの time プロパティと現在の時刻の差をミリ秒で求めて計算します。そして、その結果を "Time Until" フィールドに表示します。

```
-- Lingo
on prepareFrame
    msBeforeUpdate = timeout("Update").time - the milliseconds
    secondsBeforeUpdate = msBeforeUpdate / 1000
    minutesBeforeUpdate = secondsBeforeUpdate / 60
    member("Time Until").text = string(minutesBeforeUpdate) && "minutes before next \
    timeout"
end

// Javascript
Function prepareFrame()
{
    var msBeforeUpdate = timeout("Update").time - _system.milliseconds;
    var secondsBeforeUpdate = msBeforeUpdate / 1000;
    var minutesBeforeUpdate = secondsBeforeUpdate / 60;
    member("Time Until").text = string(minutesBeforeUpdate) + " minutes before next timeout";
}
```

関連事項

milliseconds, period, persistent, target, timeout(), timeoutHandler

timeoutHandler

使用法

```
timeoutObject.timeoutHandler
```

解説

システムプロパティ。指定された [タイムアウトオブジェクト] からタイムアウトメッセージを受け取るハンドラの名前を返します。このプロパティの値は、#timeExpiredHandler のようなシンボルです。timeoutHandler は、常に、タイムアウトオブジェクトの target オブジェクト内に置かれます。タイムアウトオブジェクトで target が指定されていない場合は、このハンドラはムービースクリプト内に置かれます。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントでは、"Quiz Timer" タイムアウトオブジェクトの timeoutHandler をメッセージウィンドウに表示します。

```
-- Lingo
put timeout("Quiz Timer").timeoutHandler

// Javascript
trace(timeout("Quiz Timer").timeoutHandler);
```

関連事項

target, timeout(), timeoutList

timeoutList

使用法

```
-- Lingo syntax
_movie.timeoutList

// JavaScript syntax
_movie.timeoutList;
```

解説

ムービープロパティ。現時点でアクティブなタイムアウトオブジェクトすべてを含んだリニアリストです。読み取り専用。

タイムアウトオブジェクトを削除するには、forget() メソッドを使用します。

timeoutList にタイムアウトオブジェクトを追加するには、new() メソッドを使用します。

例文

次のステートメントでは、タイムアウトリストの 3 番目のタイムアウトオブジェクトを削除します。

```
-- Lingo syntax
_movie.timeoutList[3].forget()

// JavaScript syntax
_movie.timeoutList[3].forget();
```

関連事項

forget() （ウィンドウ）, ムービー , new(), forget() (Timeout), timeout()

timeScale

使用法

```
member(whichCastMember).timeScale
the timeScale of member whichCastMember
```

解説

デジタルビデオのフレームレートの測定に使用する秒あたりの時間の単位を返すキャストメンバプロパティです。例えば、QuickTime デジタルビデオでは、1/600 秒が測定単位になります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライト（3）の QuickTime デジタルビデオのタイムスケールを表示します。

```
-- Lingo
put(sprite(3).timeScale)

// Javascript
trace(sprite(3).timeScale)
```

関連事項

digitalVideoTimeScale

title (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.title

// JavaScript syntax
dvdObjRef.title;
```

解説

現行タイトルを指定する DVD プロパティです。読み取り / 書き込み可能。

このプロパティは、現行タイトル番号を指定する整数を返します。

例文

次のステートメントは、現行タイトルを返します。

```
-- Lingo syntax
trace (member(1).title)-- 1

// JavaScript syntax
trace (member(1).title);// 1
```

関連事項

DVD

title （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.title

// JavaScript syntax
windowObjRef.title;
```

解説

ウィンドウにタイトルを割り当てるウィンドウプロパティです。読み取り / 書き込み可能。

例文

次のステートメントは、" 惑星 " というタイトルを 5 番目のウィンドウに割り当てます。

```
-- Lingo syntax
_player.windowList[5].title = "Planets"

// JavaScript syntax
_player.windowList[5].title = "Planets";
```

関連事項

ウィンドウ

titlebarOptions

使用法

```
-- Lingo syntax
windowObjRef.titlebarOptions

// JavaScript syntax
windowObjRef.titlebarOptions;
```

解説

ウィンドウプロパティ。ウィンドウのタイトルバーのオプションを保存するプロパティのリストを指定します。読み取り / 書き込み可能。

このプロパティリストには、次のプロパティが含まれます。

プロパティ	解説
#icon	タイトルバーで使用するキャストメンバアイコンを指定します。このプロパティは、タイトルバーが表示される場合のみ使用可能です（#visible プロパティが TRUE に設定されている場合）。
#visible	タイトルバーを表示するかどうかを指定します。FALSE に設定すると、タイトルバーは表示されませんが、その他すべてのタイトルバーやウィンドウのプロパティに影響はありません。TRUE に設定すると、タイトルバーが表示され、ウィンドウにはその他すべてのタイトルバーとウィンドウのプロパティの状態が維持されます。デフォルト値は TRUE です。
#closebox	クローズボックスをウィンドウの右上隅に表示するかどうかを指定します。TRUE に設定すると、クローズボックスが表示されます。FALSE に設定すると、クローズボックスは表示されません。デフォルト値は TRUE です。
#minimizebox	最小化ボックスをウィンドウの右上隅に表示するかどうかを指定します。TRUE に設定すると、最小化ボックスが表示されます。FALSE に設定すると、最小化ボックスは表示されません。デフォルト値は TRUE です。
#maximizebox	最大化ボックスをウィンドウの右上隅に表示するかどうかを指定します。TRUE に設定すると、最大化ボックスが表示されます。FALSE に設定すると、最大化ボックスは表示されません。デフォルト値は TRUE です。
#sideTitlebar	（Mac のみ）タイトルバーをウィンドウの横に表示するかどうかを指定します。TRUE を指定すると、タイトルバーがウィンドウの横に表示されます。FALSE を指定すると、タイトルバーはウィンドウの横に表示されません。デフォルト値は FALSE です。

これらのプロパティは、ムービーオブジェクトの displayTemplate プロパティを使用してアクセスすることもできます。

例文

次のステートメントは、メッセージウィンドウに " エレメント " という名前のウィンドウに対して使用可能なタイトルバーのオプションを表示します。

```
-- Lingo syntax
trace(window("Elements").titlebarOptions)

// JavaScript syntax
trace(window("Elements").titlebarOptions);
```

これらのステートメントは、smallIcon という名前のビットマップキャストメンバに icon プロパティを設定します。

```
-- Lingo syntax
window("Elements").titlebarOptions.icon = member("smallIcon")

// JavaScript syntax
window("Elements").titlebarOptions.icon = member("smallIcon");
```

関連事項

appearanceOptions, displayTemplate, ウィンドウ

titleCount

使用法

```
-- Lingo syntax
dvdObjRef.titleCount

// JavaScript syntax
dvdObjRef.titleCount;
```

解説

使用可能なタイトルの番号を返す DVD プロパティです。読み取り専用。

使用可能なタイトルの番号の範囲は 1 ～ 99 です。

例文

次のステートメントは、DVD オブジェクトのタイトルの数を返します。

```
-- Lingo
put sprite(1).titleCount

// Javascript
trace(sprite(1).titleCount)
```

関連事項

DVD

toolXtraList

使用法

```
-- Lingo syntax
_player.toolXtraList

// JavaScript syntax
_player.toolXtraList;
```

解説

Director プレーヤで使用可能なすべてのツール Xtra 拡張機能のリニアリストを返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、メッセージウィンドウに使用可能なすべてのツール Xtra 拡張機能を表示します。

```
-- Lingo syntax
put(_player.toolXtraList)

// JavaScript syntax
put(_player.toolXtraList);
```

関連事項

mediaXtraList, プレーヤ , scriptingXtraList, transitionXtraList, xtraList （プレーヤ）

toon （モディファイア）

使用法

```
member(whichCastmember).model(whichModel).toon.toonModifierProperty
```

解説

3D モディファイア。#toon モディファイアをモデルに追加すると、#toon モディファイアプロパティを取得および設定できます。

トゥーンモディファイアは、少数のカラーを使用してモデルを描画し、モデルのサーフェスのレンダリングをカートゥーンスタイルで行います。#toon モディファイアを適用すると、モデルのシェーダの texture、reflectionMap、diffuseLightMap、specularLightMap、および glossMap の各プロパティは無視されます。

#toon モディファイアを #inker モディファイアと同時に使用すると、レンダリングされる効果は累積し、どちらのモディファイアが先に適用されたかによって結果が異なります。modifier プロパティによって返されるモディファイアのリストには、#inker または #toon のうち最初に追加されたほうが含められます。両方が含められることはありません。トゥーンモディファイアは、#sds モディファイアと同時に使用することはできません。

#toon モディファイアには以下のプロパティがあります。

注意：以下のプロパティの詳細については、プロパティ個別のエントリを参照してください。

- style。カラートランジションに適用されるスタイルを取得または設定できます。使用できる値は以下のとおりです。

  #toon。使用可能なカラー間でのシャープトランジションを指定します。

  #gradient。使用可能なカラー間でのソフトトランジションを指定します。

  #blackAndWhite。黒と白の間のシャープトランジションを指定します。

- colorSteps。ライティングの計算に使用するカラー数を取得または設定できます。この値を設定すると、カラー数は最も近い 2 の累乗に丸められます。使用できる値は、2、4、8、および 16 です。デフォルトは 2 です。
- shadowPercentage。ライティング用に定義され、モデルサーフェスのシャドウ部分のレンダリングに使用されるカラー（colorSteps）のパーセンテージを取得または設定できます。この値の範囲は 0 ～ 100 で、デフォルトは 50 です。
- shadowStrength。モデルサーフェスのシャドウ部分に適用される暗度を取得または設定できます。この値は、負の数以外の浮動小数点数です。デフォルト値は 1.0 です。
- highlightPercentage。ライティング用に定義され、モデルサーフェスのハイライト部分のレンダリングに使用されるカラー（colorSteps）のパーセンテージを取得または設定できます。この値の範囲は 0 ～ 100 で、デフォルトは 50 です。
- highlightStrength。モデルサーフェスのハイライト部分に適用される明度を取得または設定できます。この値は、負の数以外の浮動小数点数です。デフォルト値は 1.0 です。
- lineColor。インカーによって描画される線のカラーを取得または設定できます。この値には、有効な Lingo カラーオブジェクトを使用できます。デフォルト値は rgb (0, 0, 0)、つまり黒です。
- creases。折り目で線を描画するかどうかを取得または設定できます。これはブール値であり、デフォルト値は True です。
- creaseAngle。creases が TRUE に設定されているとき、トゥーンモディファイアの線描画の機能が、折り目でどのように処理されるかを取得または設定できます。
- boundary。サーフェスの境界線を描画するかどうかを取得または設定できます。これはブール値であり、デフォルト値は True です。
- lineOffset。シェーディングされたサーフェスとカメラに対して線を描画する位置を取得または設定できます。負の値の場合、線はカメラの方に移動します。正の値の場合、線はカメラから離れます。この値は、-100.0 ～ 100.0 の浮動小数点数です。デフォルト値は -2.0 です。

- useLineOffset。lineOffset がオンかオフかを取得または設定できます。これはブール値であり、デフォルト値は False です。
- silhouettes。モデルの輪郭をなぞるように境界に沿ってエッジを定義する線を描画するかどうかを取得または設定できます。これはブール値であり、デフォルト値は True です。

関連事項

addModifier, modifiers, sds （モディファイア）, inker （モディファイア）

top

使用法

```
-- Lingo syntax
spriteObjRef.top

// JavaScript syntax
spriteObjRef.top;
```

解説

スプライトプロパティ。ステージの左上隅からのピクセル数で、スプライトの境界矩形の上端の垂直座標を返すか設定します。読み取り / 書き込み可能。

例文

次のステートメントは、スプライト 3 の上端がステージの上端を越えているかどうかをチェックし、越えている場合は offtopedge ハンドラを呼び出します。

```
-- Lingo syntax
if (sprite(3).top < 0) then
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

関連事項

bottom, height, left, locH, locV, right, スプライト , width

topSpacing

使用法

```
chunkExpression.topSpacing
```

解説

テキストキャストメンバのプロパティ。テキストキャストメンバの [チャンク式] の部分で、各段落の上に追加するスペースを示します。

このプロパティの値は整数で指定します。負の値の場合は段落間の間隔が狭くなり、正の値の場合は広くなります。

デフォルト値は 0 です。段落間の間隔はデフォルト値になります。

例文

次のステートメントでは、テキストキャストメンバ myText 内の第 2 段落の topSpacing を 20 に設定します。

```
-- Lingo
member(1).paragraph[2].topSpacing = 20

// Javascript
member(1).getPropRef("paragraph",2).topSpacing = 20;
```

関連事項

bottomSpacing

traceLoad

使用法

```
-- Lingo syntax
_movie.traceLoad

// JavaScript syntax
_movie.traceLoad;
```

解説

キャストメンバのロード時に表示する情報の量を示すムービープロパティです。読み取り / 書き込み可能。

traceLoad の有効な値は、以下のとおりです。

- 0 - 情報を表示しません（デフォルト）。
- 1 — キャストメンバ名を表示します。
- 2 — キャストメンバ名、現行フレーム番号、ムービー名、およびファイル検索のオフセット（ドライブがメディアをロードするために移動する必要があった相対量）を表示します。

例文

次のステートメントでは、ムービーのキャストメンバがロードされるときに、その名前を表示します。

```
-- Lingo syntax
_movie.traceLoad = 1

// JavaScript syntax
_movie.traceLoad = 1;
```

関連事項

ムービー

traceLogFile

使用法

```
-- Lingo syntax
_movie.traceLogFile

// JavaScript syntax
_movie.traceLogFile;
```

解説

ムービープロパティ。メッセージウィンドウに表示される内容を書き込むファイルの名前を示します。読み取り / 書き込み可能。

traceLogFile プロパティを EMPTY（Lingo）または空の文字列 ""（JavaScript シンタックス）に設定することにより、ファイルを閉じることができます。メッセージウィンドウに表示される出力はすべてこのファイルに書き出されます。このプロパティは、プロジェクタでムービーを再生するときや、オーサリングを行うときのデバッグに利用できます。

例文

次のステートメントは、メッセージウィンドウの内容を現在のムービーと同じフォルダにある "Messages.txt" ファイルに書き出します。

```
-- Lingo syntax
_movie.traceLogFile = _movie.path & "Messages.txt"

// JavaScript syntax
_movie.traceLogFile = _movie.path + "Messages.txt";
```

次のステートメントは、メッセージウィンドウに表示されている内容を書き出すファイルを閉じます。

```
-- Lingo syntax
_movie.traceLogFile = ""

// JavaScript syntax
_movie.traceLogFile = "";
```

関連事項

ムービー

traceScript

使用法

```
-- Lingo syntax
_movie.traceScript

// JavaScript syntax
_movie.traceScript;
```

解説

ムービープロパティ。ムービーの trace 機能がオンになっているか（TRUE）、オフになっているか（FALSE）を示します。

読み取り / 書き込み可能。

traceScript 機能がオンのときは、実行されるスクリプトの各行がメッセージウィンドウに表示されます。

例文

次のステートメントは、traceScript プロパティをオンにします。

```
-- Lingo syntax
_movie.traceScript = TRUE

// JavaScript syntax
_movie.traceScript = true;
```

関連事項

ムービー

trackCount （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.trackCount()

// JavaScript syntax
memberObjRef.trackCount();
```

解説

指定したデジタルビデオキャストメンバのトラック数を返すデジタルビデオキャストメンバプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" ジャズの歴史 " デジタルビデオキャストメンバのトラック数をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Jazz Chronicle").trackCount())

// JavaScript syntax
trace(member("Jazz Chronicle").trackCount());
```

trackCount （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.trackCount()

// JavaScript syntax
spriteObjRef.trackCount();
```

解説

指定したデジタルビデオスプライトのトラック数を返すデジタルビデオスプライトプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、チャネル 10 に割り当てられているデジタルビデオスプライトのトラック数をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(10).trackCount())

// JavaScript syntax
trace(sprite(10).trackCount());
```

trackEnabled

使用法

```
-- Lingo syntax
spriteObjRef.trackEnabled(whichTrack)

// JavaScript syntax
spriteObjRef.trackEnabled(whichTrack);
```

解説

デジタルビデオ内の指定したトラックのステータスを示すデジタルビデオスプライトプロパティです。このプロパティが TRUE のときは、指定したトラックは使用可能で、再生できます。このプロパティが FALSE のときは、指定したトラックは使用不能で、再生できなくなっているか、更新されていません。

このプロパティを変更することはできません。代わりに setTrackEnabled プロパティを使ってください。

例文

次のステートメントでは、デジタルビデオスプライト 1 のトラック 2 が使用できるかどうかをチェックします。

```
-- Lingo syntax
put(sprite(1).trackEnabled(2))

// JavaScript syntax
put(sprite(1).trackEnabled(2));
```

関連事項

setTrackEnabled()

trackInfo

使用法

```
put member(1).trackInfo
put sprite(1).trackInfo
```

解説

MP4Media/FLV キャストメンバおよびスプライトのプロパティ。指定したビデオキャストメンバに含まれるビデオとオーディオのトラックのリストを返します。このプロパティは、取得できますが設定はできません。

トラックのリストは次の形式です。

```
[[#trackNum: #trackType]…]
#trackNum is 0, 1, 2…
#trackType is #video or #audio
Output: [[#tracknum:0 , #trackType:#video],[#tracknum:1, #trackType: #audio]]
```

注意：MP4 ビデオキャストメンバでサポートされるビデオトラックは 1 つだけです。ただし、ビデオキャストメンバは任意の数のオーディオトラックを持つことができます。

例文

```
-- Lingo syntax
put member(1).trackInfo -- [[#tracknum:0 , #trackType:#video],[#tracknum:1,
-- #trackType:#audio]]
put sprite(1).trackInfo -- [[#tracknum:0 , #trackType:#video],[#tracknum:1,
-- #trackType:#audio]]

// JavaScript syntax
put member(1).trackInfo; // [[#tracknum:0 , #trackType:#video],[#tracknum:1,
// #trackType:#audio]]
put sprite(1).trackInfo; // [[#tracknum:0 , #trackType:#video],[#tracknum:1,
// #trackType:#audio]]
```

trackNextKeyTime

使用法

```
-- Lingo syntax
spriteObjRef.trackNextKeyTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackNextKeyTime(whichTrack);
```

解説

指定したデジタルビデオトラック内で、現在の時間よりも後に存在するキーフレームの時間を示すデジタルビデオスプライトプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライトチャネル 15 に割り当てられているデジタルビデオのトラック 5 内で、次のキーフレームの時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(15).trackNextKeyTime(5))

// JavaScript syntax
put(sprite(15).trackNextKeyTime(5));
```

trackNextSampleTime

使用法

```
-- Lingo syntax
spriteObjRef.trackNextSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackNextSampleTime(whichTrack);
```

解説

デジタルビデオの現在の時間よりも後に存在するサンプルフレームの時間を示すデジタルビデオスプライトプロパティです。このプロパティは、デジタルビデオ内のテキストトラックを検索するときに役立ちます。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライト 15 に割り当てられているデジタルビデオのトラック 5 内で、現在の時間に続く、次のサンプルの時間を表示します。

```
-- Lingo syntax
put(sprite(15).trackNextSampleTime(5))

// JavaScript syntax
put(sprite(15).trackNextSampleTime(5));
```

trackPreviousKeyTime

使用法

```
-- Lingo syntax
spriteObjRef.trackPreviousKeyTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousKeyTime(whichTrack);
```

解説

現在の時間よりも前のキーフレームの時間を示すデジタルビデオスプライトプロパティです。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、チャネル 15 に割り当てられているデジタルビデオスプライトのトラック 5 内で、現在の時間よりも前のキーフレームの時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(15).trackPreviousKeyTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousKeyTime(5));
```

trackPreviousSampleTime

使用法

```
-- Lingo syntax
spriteObjRef.trackPreviousSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousSampleTime(whichTrack);
```

解説

デジタルビデオ内で、現在の時間よりも前にあるサンプルフレームの時間を示すデジタルビデオスプライトプロパティです。このプロパティは、デジタルビデオ内のテキストトラックを検索するときに役立ちます。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、チャネル 15 に割り当てられているデジタルビデオスプライトのトラック 5 内で、現在の時間よりも前にあるサンプルの時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(15).trackPreviousSampleTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousSampleTime(5));
```

trackStartTime （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.trackStartTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStartTime(whichTrack);
```

解説

デジタルビデオキャストメンバプロパティ。指定したデジタルビデオキャストメンバの指定したトラックの開始時間を返します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" ジャズの歴史 " デジタルビデオキャストメンバのトラック 5 の開始時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStartTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStartTime(5));
```

trackStartTime （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.trackStartTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStartTime(whichTrack);
```

解説

デジタルビデオスプライトプロパティ。指定したスプライトチャネル内のデジタルビデオムービーの開始時間を示します。trackStartTime の値はチック数で測定されます。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライトチャネル 10 内のトラック 5 が再生を開始する時間をメッセージウィンドウに表示します。開始時間は、トラックの 120 チック (2 秒) の場所です。

```
-- Lingo syntax
put(sprite(10).trackStartTime(5))

// JavaScript syntax
put(sprite(10).trackStartTime(5))
```

関連事項

duration （メンバ）, playRate（QuickTime、AVI、MP4、FLV）, currentTime (QuickTime, AVI)

trackStopTime （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.trackStopTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStopTime(whichTrack);
```

解説

デジタルビデオキャストメンバプロパティ。指定したデジタルビデオキャストメンバの指定したトラックの停止時間を返します。このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、" ジャズの歴史 " デジタルビデオキャストメンバのトラック 5 の停止時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStopTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStopTime(5));
```

trackStopTime （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.trackStopTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStopTime(whichTrack);
```

解説

デジタルビデオスプライトプロパティ。指定したデジタルビデオスプライトの指定したトラックの停止時間を示します。

デジタルビデオムービーが再生されているときは、再生が中断したり、ループ (loop プロパティがオンになっている場合) した場所が、trackStopTime になります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライト 6 に割り当てられているデジタルビデオ内のトラック 5 の停止時間をチェックし、結果をメッセージウィンドウに表示します。

```
-- Lingo syntax
put(sprite(6).trackStopTime(5))

// JavaScript syntax
put(sprite(6).trackStopTime(5));
```

関連事項

playRate（QuickTime、AVI、MP4、FLV）, currentTime (QuickTime, AVI), trackStartTime （メンバ）

trackText

使用法

```
-- Lingo syntax
spriteObjRef.trackText(whichTrack)

// JavaScript syntax
spriteObjRef.trackText(whichTrack);
```

解説

デジタルビデオスプライトプロパティ。デジタルビデオの指定したトラック内で、現在の時間に存在しているテキストを示します。結果はストリング値で、最大 32K 分の長さの文字になります。このプロパティは、テキストトラックのみに適用されます。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、スプライト 20 に割り当てられているデジタルビデオのトラック 5 内で現在の時間に位置しているテキストを " アーカイブ " フィールドキャストメンバに割り当てます。

```
-- Lingo syntax
member("Archives").text = string(sprite(20).trackText(5))

// JavaScript syntax
member("Archives").text = sprite(20).trackText(5).toString();
```

trackType （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.trackType(whichTrack)

// JavaScript syntax
memberObjRef.trackType(whichTrack);
```

解説

指定したキャストメンバの指定したトラック内のメディアタイプを示すためのデジタルビデオキャストメンバプロパティです。値は、#video、#sound、#text、#music のいずれかになります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のハンドラは、デジタルビデオキャストメンバの " 今日のニュース " のトラック 5 がテキストトラックかどうかをチェックし、テキストトラックの場合は textFormat ハンドラを実行します。

```
-- Lingo syntax
on checkForText
    if member("Today's News").trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = member("Today's News").trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trackType （スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.trackType(whichTrack)

// JavaScript syntax
spriteObjRef.trackType(whichTrack);
```

解説

指定したスプライトの指定したトラック内のメディアタイプを示すデジタルビデオスプライトプロパティです。値は、#video、#sound、#text、#music のいずれかになります。

このプロパティは、値を調べられますが、設定はできません。

例文

次のハンドラは、チャネル 10 に割り当てられているデジタルビデオスプライトのトラック 5 がテキストトラックかどうかをチェックし、テキストトラックの場合は textFormat ハンドラを実行します。

```
-- Lingo syntax
on checkForText
    if sprite(10).trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = sprite(10).trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trails

使用法

```
sprite(whichSprite).trails
the trails of sprite whichSprite
```

解説

スプライトプロパティ。[スプライト] で指定したスプライトの軌跡インク効果をオン（1 または TRUE）またはオフ（0 または FALSE）にします。Lingo が設定した値を現行スプライト終了後も維持するには、スクリプトを使用したスプライトにする必要があります。

軌跡を消去するには、これらのピクセル上で別のスプライトをアニメートさせるか、トランジションを使います。

例文

次のステートメントは、スプライト 7 の軌跡をオンにします。

```
-- Lingo
sprite(7).trails = 1

// Javascript
sprite(7).trails = 1;
```

関連事項

directToStage

transform （プロパティ）

使用法

```
member(whichCastmember).node(whichNode).transform
member(whichCastmember).node(whichNode).transform.transformProperty
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform.transformProperty
```

解説

3D プロパティおよびコマンド。特定のノード、または bonesPlayer モディファイアを使用し、モデル内の特定のボーンに関連付けられたトランスフォームを取得または設定できます。コマンドとして transform を使用すると、トランスフォームオブジェクトの各種コマンドおよびプロパティにアクセスできます。ノードは、カメラ、グループ、ライト、またはモデルオブジェクトのいずれかです。

ノードオブジェクトの場合、このプロパティのデフォルトはアイデンティティトランスフォームです。ノードのトランスフォームは、親オブジェクトを基準としてノードの位置、回転、およびスケールを定義します。ノードの親がワールドグループオブジェクトである場合、ノードの transform プロパティは、getWorldTransform() コマンドで返されるのと同じ値です。

bonesPlayer モディファイアを使用してモデル内のボーンを指定する場合は、このプロパティのデフォルト値は、モデルファイルの作成時にそのボーンに割り当てられたトランスフォームになります。ボーンのトランスフォームは、親ボーンを基準としたボーンの回転、および元のジョイント位置を基準とした位置を表します。元のジョイント位置は、モデルファイルの作成時に決定されます。

ノードオブジェクトの transform プロパティでは、次のトランスフォームコマンドとプロパティを使用できます。

注意：ここでは概要のみを紹介します。詳細については、個別のエントリを参照してください。

- preScale。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用する前にスケーリングを適用します。
- preTranslate。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用する前に平行移動を適用します。
- preRotate。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用する前に回転を適用します。
- scale (コマンド)。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用した後にスケールを適用します。
- scale (トランスフォーム)。トランスフォームのスケールの度合いを取得または設定できます。
- translate。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用した後に平行移動を適用します。
- rotate。トランスフォームで保持される現在の位置、回転、およびスケールのオフセットを適用した後に回転を適用します。
- position (トランスフォーム)。トランスフォームの位置オフセットを取得または設定できます。
- rotation (トランスフォーム)。トランスフォームの回転オフセットを取得または設定できます。

モデル内のボーンの transform プロパティを変更するには、ボーンの元のトランスフォームのコピーを保存し、保存したコピーを上記のコマンドとプロパティで変更します。次に、ボーンの transform プロパティを、変更したトランスフォームと等しくなるようにリセットします。例えば次のようになります。

```
t = member("character").model("biped").bonesPlayer.bone[38].transform.duplicate()
t.translate(25,0,-3)
member("character").model("biped").bonesPlayer.bone[38].transform = t
```

パラメータ

なし

例文

次のステートメントは、モデル " ボックス " のトランスフォームと、トランスフォームの position プロパティおよび rotation プロパティを示します。

```
put member("3d world").model("box").transform
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("box").transform.position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("box").transform.rotation
--vector(0.0000, 0.0000, 0.0000)
```

関連事項

interpolateTo(), scale （トランスフォーム）, rotation （トランスフォーム）, position （トランスフォーム）, bone, worldTransform, preRotate, preScale(), preTranslate()

transitionType

使用法

```
member(whichCastMember).transitionType
the transitionType of member whichCastMember
```

解説

トランジションキャストメンバプロパティ。トランジションの種類を番号で示します。ここで使用される番号の意味は、puppetTransition コマンドの各トランジションに割り当てられているコード番号と同じです。

例文

次のステートメントは、キャストメンバ 3 のトランジションを 51 に設定します。これは、" ピクセルごとににじむ " というトランジションの種類です。

```
member(3).transitionType = 51
```

transitionXtraList

使用法

```
-- Lingo syntax
_player.transitionXtraList

// JavaScript syntax
_player.transitionXtraList;
```

解説

Director プレーヤで使用可能なすべてのトランジション Xtra 拡張機能のリニアリストを返すプレーヤプロパティです。読み取り専用。

例文

次のステートメントは、メッセージウィンドウに使用可能なすべてのトランジション Xtra 拡張機能を表示します。

```
-- Lingo syntax
put(_player.transitionXtraList)

// JavaScript syntax
put(_player.transitionXtraList);
```

関連事項

mediaXtraList, プレーヤ , scriptingXtraList, toolXtraList, xtraList （プレーヤ）

translation

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.translation

// JavaScript syntax
memberOrSpriteObjRef.translation;
```

解説

「QuickTime スプライト」の境界ボックス内の画像開始位置を制御する、QuickTime キャストメンバおよびスプライトプロパティです。

この開始位置は、center プロパティによって設定されたスプライトのデフォルトの位置を基準として表されます。center が TRUE に設定されている場合、画像開始位置は境界矩形の中心を基準として決められます。center が FALSE に設定されている場合、開始位置は境界矩形の左上隅を基準として決められます。

開始位置は正または負の整数としてピクセル単位で指定し、Director リストとして [**xTrans**,**yTrans**] と指定します。**xTrans** パラメータはスプライトのデフォルト位置から見た左右の開始位置を、**yTrans** パラメータは上下の開始位置を表します。デフォルト設定は [0,0] です。

スプライトの crop プロパティが TRUE に設定されていると、translation プロパティを使用して QuickTime ムービーの一部を境界矩形の外側に移動することにより、この部分を隠すことができます。crop プロパティが FALSE に設定されている場合は、translation プロパティは無視され、スプライトは常にその矩形の左上隅に位置します。

このプロパティは、値を調べたり、設定することができます。

例文

次のフレームスクリプトでは、ピクセル幅 320 の QuickTime スプライトのキャストメンバがチャネル 5 にあるものとします。この center プロパティは FALSE に設定され、crop プロパティは TRUE に設定されています。ムービーの水平方向の平行移動ポイントが 10 ピクセルずつ移動し、スプライトの右端に至るまで、このフレームスクリプトは再生ヘッドを現行フレームに保ちます。これには右方へのワイプ効果があり、スプライトを表示の右方の外側に移動します。スプライトが表示外に移動すると、再生ヘッドは次のフレームに進みます。

```
-- Lingo syntax
on exitFrame
    horizontalPosition = sprite(5).translation[1]
    if horizontalPosition < 320 then
        sprite(5).translation = sprite(5).translation + [10, 0]
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var horizontalPosition = sprite(5).translation[1];
    if (horizontalPosition < 320 ) {
        sprite(5).translation = sprite(5).translation + list(10, 0);
        _movie.go(_movie.frame);
    }
}
```

transparent

使用法

```
member(whichCastmember).shader(whichShader).transparent
member(whichCastmember).model(whichModel).shader.transparent
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].transparent
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、アルファ値を使用してモデルをブレンドするか（TRUE）、または不透明としてレンダリングするか（FALSE）を取得または設定できます。このプロパティのデフォルト値は TRUE（アルファブレンドする）です。

shader.blend の機能は、このプロパティに依存します。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

次のステートメントは、モデル " 冥王星 " を不透明にレンダリングします。このモデルのシェーダに対する blend プロパティの設定は、効果がありません。

```
-- Lingo
member("scene").model("Pluto").shader.transparent = FALSE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").transparent = false;
```

関連事項

blendFactor, blend (3D)

triggerCallback

使用法

```
-- Lingo syntax
spriteObjRef.triggerCallback

// JavaScript syntax
spriteObjRef.triggerCallback;
```

解説

ユーザが QuickTime VR ムービー内のホットスポットをクリックしたときに実行されるハンドラの名前を含む QuickTime VR スプライトプロパティです。このハンドラには、me パラメータと、ユーザがクリックしたホットスポットの ID という 2 つのパラメータが渡されます。

このハンドラが返す値は、ムービーによるホットスポットの処理方法を表します。ハンドラが #continue を返す場合、QuickTime VR スプライトはホットスポットの処理を通常の方法で続行します。ハンドラが #cancel を返す場合は、ホットスポットのデフォルトのビヘイビアーが取り消されます。

コールバックを解消するには、このプロパティを 0 に設定します。

QuickTime VR スプライトが、このメッセージを最初に受け取ります。

パフォーマンス上の違反を避けるため、triggerCallback は必要なときのみ設定します。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、再生ヘッドが最初にスプライト範囲に入ったときに、QuickTime VR スプライトのコールバックハンドラを MyHotSpotCallback というハンドラに設定します。このホットスポットが呼び出されるたびに MyHotSpotCallback ハンドラが実行されます。再生ヘッドがスプライト範囲を抜け出すと、コールバックはキャンセルされます。

```
-- Lingo syntax
property pMySpriteNum, spriteNum

on beginSprite(me)
    pMySpriteNum = spriteNum
    sprite(pMySpriteNum).triggerCallback = #MyHotSpotCallback
end

on MyHotSpotCallback(me, hotSpotID)
    put "Hotspot" && hotSpotID && "was just triggered"
end

on endSprite me
    sprite(pMySpriteNum).triggerCallback = 0
end

// JavaScript syntax
function beginSprite() {
    pMySpriteNum = this.spriteNum;
    sprite(this.pMySpriteNum).triggerCallback = symbol("MyHotSpotCallback");
}

function MyHotSpotCallback(hotSpotID) {
    trace("Hotspot " + hotSpotID + " was just triggered");
}

function endSprite() {
    sprite(pMySpriteNum).triggerCallback = 0;
}
```

trimWhiteSpace

使用法

```
-- Lingo syntax
memberObjRef.trimWhiteSpace

// JavaScript syntax
memberObjRef.trimWhiteSpace;
```

解説

キャストメンバプロパティ。ビットマップキャストメンバのエッジの周辺にある白いピクセルを消去するか、そのままにしておくのかを示します。このプロパティは、メンバが読み込まれるときに設定されます。このプロパティを変更するには、Lingo を使用するか、またはプロパティインスペクタの [ビットマップ] タブを使用します。

例文

次のステートメントは、テキストメンバ内のホワイトスペースをトリミングします。

```
-- Lingo
member(1).trimWhiteSpace;

// Javascript
member(1).trimWhiteSpace;
```

tunnelDepth

使用法

```
member(whichTextmember).tunnelDepth
member(whichCastMember).modelResource(whichExtruderModelResource).tunnelDepth
```

解説

3D 押し出しリソースプロパティ、およびテキストキャストメンバプロパティ。このプロパティを使用すると、1.0D テキストモデルリソースの押し出し深度 (表面と裏面の間の距離) を取得または設定できます。値は 100.0 ～ 50.0 の浮動小数点数です。デフォルト値は 50.0 です。

押し出しモデルリソースとテキストキャストメンバの操作の詳細については、「extrudeToMember」エントリを参照してください。

例文

この例のキャストメンバ " ロゴ " はテキストキャストメンバです。次のステートメントは、ロゴの tunnelDepth プロパティを 5 に設定します。ロゴ が 3D モードで表示されるときは、文字の深度が非常に浅くなります。

```
-- Lingo
member("logo").tunnelDepth = 5

// Javascript
member("logo").tunnelDepth = 5;
```

次の例で、モデル " スローガン " のモデルリソースは押し出しテキストです。次のステートメントは、" スローガン " のモデルリソースの tunnelDepth を 1000 に設定します。" スローガン " の文字の深度は非常に深くなります。

```
-- Lingo
member("scene").model("Slogan").resource.tunnelDepth = 1000

// Javascript
member("scene").getPropRef("model",1).getProp("resource").tunnelDepth = 1000;
```

関連事項

extrude3D

tweened

使用法

```
sprite(whichSprite).tweened
the tweened of sprite whichSprite
```

解説

新規スプライトの最初のフレームのみをキーフレームにするか (TRUE)、新規スプライトのすべてのフレームをキーフレームにするか (FALSE) を決めるスプライトプロパティです。

このプロパティは、再生には何の影響もなく、スコア生成セッションにのみ有効です。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントが発行されると、チャネル 25 内の新規スプライトの範囲の 1 つ目のフレームのみがキーフレームになります。

```
-- Lingo
sprite(25).tweened = 1

// Javascript
sprite(25).tweened = 1;
```

tweenMode

使用法

```
member(whichCastmember).modelResource(whichModelResource).tweenMode
modelResourceObjectReference.tweenMode
```

解説

3D パーティクルプロパティ。パーティクルのカラーが速度と期間のどちらに応じて変化するかを取得または設定できます。tweenMode プロパティの値を以下に示します。

- #velocity。パーティクルの速度に基づいて、colorRange.start と colorRange.end の間のパーティクルのカラーを変更します。
- #age。パーティクルの寿命が終わるまで、colorRange.start と colorRange.end の間のカラーを一定の比率で変更します。このプロパティのデフォルト設定です。

例文

この例の "ThermoSystem" は、#particle というタイプのモデルリソースです。次のステートメントは、"ThermoSystem" の tweenMode を #velocity に設定します。これによって、高速のパーティクルは colorRange.end で指定したカラーに達しますが、低速のパーティクルはそのカラーには達しません。

```
-- Lingo
member(8).modelResource("thermoSystem").tweenMode = #velocitytype

// Javascript
member(8).getPropRef("modelResource",1).tweenMode =symbol("velocitytype");
```

type (light)

使用法

```
member(whichCastmember).light(whichLight).type
```

解説

3D ライトプロパティ。参照されたライトの種類です。このプロパティの使用できる値は以下のとおりです。

- #ambient タイプのライトは、すべてのサーフェスに均等にライトを当てます。周囲ライトの輝度は、光源からの距離の影響を受けません。
- #directional タイプは指向性ライトです。特定の方向を照らしますが、#spot タイプほど光が集中しません。指向性ライトの輝度は、光源から離れるにしたがって低下します。
- #point タイプのライトは、3D ワールド内の指定された場所にあり、すべての方向を照らします。この効果は、部屋の天井に吊ってある裸電球が光を照らすようすに似ています。ポイントライトの輝度は、光源から離れるにしたがって低下します。

- #spot タイプのライトは、特定のポイントから、ライトの前方向および spotAngle プロパティで定義される円錐形状に光を当てます。スポットライトの輝度は、attenuation プロパティで定義された値を使用して、光源から離れるにつれて低下します。

例文

次のステートメントは、"MainLight" というライトの type プロパティを表示します。

```
-- Lingo
put member("3D").motion("MainLight").type
-- #spot

// Javascript
trace(member("3D").getPropRef("motion",1).type);
// symbol("spot")
```

関連事項

spotAngle, attenuation

type （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.type

// JavaScript syntax
memberObjRef.type;
```

解説

キャストメンバの種類を示すメンバプロパティです。読み取り専用。

type プロパティは、次のいずれかの値になります。

#animgif	#palette
#bitmap	#physics
#button	#picture
#cursor	#QuickTimeMedia
#digitalVideo	#realMedia
#DVD	#script
#empty	#shape
#field	#shockwave3D
#filmLoop	#sound
#flash	#swa
#flashcomponent	#text
#font	#transition

#havok	#vectorShape
#movie	#windowsMedia
#ole	#MP4Asset

上のリストには、Director および標準で付属する Xtra で使用できる種類のキャストメンバが記載されています。カスタムキャストメンバでは、キャストメンバの種類を独自に定義することもできます。

Director のバージョン 5、6、および 6.5 で作成されたムービーでは、type プロパティは #field をフィールドキャストメンバとして、#richText をテキストキャストメンバとして返します。Director 7 以降のすべてのバージョンでは、#field がフィールドキャストメンバとして、#text がテキストキャストメンバとして返されます。

例文

次のハンドラは、キャストメンバの " 今日のニュース " がフィールドキャストメンバかどうかをチェックし、そうでない場合はアラートを表示します。

```
-- Lingo syntax
on checkFormat
    if (member("Today's News").type <> #field) then
        _player.alert("Sorry, this cast member must be a field.")
    end if
end

// JavaScript syntax
function checkFormat() {
    if (member("Today's News").type != "field") {
        _player.alert("Sorry, this cast member must be a field.");
    }
}
```

関連事項

メンバ

type （モデルリソース）

使用法

```
member(whichCastmember).modelResource(whichModelResource).type
```

解説

3D モデルリソースプロパティ。参照されるモデルリソースのリソースの種類です。このプロパティは、次のいずれかの値になります。

- #box 。このモ ルリソースが、newModelResource コマンドを使って作成されたプリミティブボックスリソースであることを示します。
- #cylinder 。このモデルリソースが、newModelResource コマンドを使って作成されたプリミティブ円柱リソースであることを示します。
- #extruder 。extrude3d コマンドを使って作成されたプリミティブテキスト押し出しリソースであることを示します。
- #mesh 。このモデルリソースが、newMesh コマンドを使って作成されたプリミティブメッシュジェネレータリソースであることを示します。

- #particle 。このモデルリソースが、newModelResource コマンドを使って作成されたプリミティブパーティクルシステムリソースであることを示します。
- #plane 。このモデルリソースが、newModelResource コマンドを使って作成されたプリミティブ平面リソースであることを示します。
- #sphere 。このモデルリソースが、newModelResource コマンドを使って作成されたプリミティブ球体リソースであることを示します。
- #fromFile 。このモデルリソースが Director の外部で作成され、外部ファイルまたはキャストメンバからロードされたことを示します。

例文

次のステートメントは、" 螺旋 " というモデルリソースの type プロパティを表示します。

```
put member("helix models").modelResource("Helix").type
-- #fromFile
```

関連事項

newModelResource, newMesh, extrude3D

type (motion)

使用法

```
member(whichCastmember).motion(whichMotion).type
```

解説

3D モーションプロパティ。参照されたモーションオブジェクトのモーションの種類です。このプロパティは、次のいずれかの値になります。

- #bonesPlayer 。このモーションがボーンベースのアニメーションであり、再生には #bonesPlayer モディファイアを使用する必要があることを示します。
- #keyFramePlayer 。このモーションがキーフレームアニメーションであり、再生には #keyFramePlayer モディファイアを使用する必要があることを示します。
- #none 。このモーションにはマッピングされた動きがなく、再生には #bonesPlayer モディファイアまたは #keyFramePlayer モディファイアのいずれかを使用できることを示します。すべての 3D キャストメンバに含まれている、デフォルトモーションオブジェクトです。

例文

次のステートメントは、" 走る " というモーションの type プロパティを表示します。

```
put member("scene").motion("Run").type
-- #bonesPlayer
```

次のステートメントは、"DefaultMotion" というモーションの type プロパティを表示します。

```
put member("scene").motion("DefaultMotion").type
-- #none
```

関連事項

bonesPlayer （モディファイア）, keyframePlayer （モディファイア）

type (shader)

使用法

```
member(whichCastmember).shader(whichShader).type
```

解説

3D シェーダプロパティ。参照されたシェーダオブジェクトのシェーダの種類です。このプロパティは、次のいずれかの値になります。

- #standard 。標準シェーダであることを示します。
- #painter 。ペイン ーシェーダであることを示します。
- #newsprint 。ニュースプリントシェーダであることを示します。
- #engraver 。エングレーバーシェーダであることを示します。

例文

次のステートメントは、" ボックス 2" というモデルで使用されるシェーダがペインターシェーダであることを示します。

```
put member("Scene").model("box2").shader.type
-- #painter
```

関連事項

newShader

type (sprite)

使用法

```
sprite(whichSprite).type
the type of sprite whichSprite
```

解説

スコア生成セッションでこの type スプライトプロパティのチャネルの値を 0 に設定することにより、そのスプライトチャネルをクリアできるスプライトプロパティです。

注意：メンバタイプを切り替えるときにスプライトのプロパティが変更されるのを防ぐため、スプライトのメンバは、同じタイプのメンバにのみ切り替えてください。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、スコア生成セッション時に実行すると、スプライトチャネル 1 をクリアします。

```
sprite(1).type = 0
```

関連事項

beginRecording(), updateFrame()

type (texture)

使用法

```
member(whichCastmember).shader(whichShader).type
```

解説

3D テクスチャプロパティ。参照されたテクスチャオブジェクトのテクスチャの種類です。このプロパティは、次のいずれかの値になります。

- #fromCastMember 。このテクスチャが newTexture コマンドを使って、image プロパティをサポートする Director キャストメンバから作成されたことを示します。
- #fromImageObject 。このテクスチャが、newTexture コマンドを使って画像オブジェクトから作成されたことを示します。
- #importedFromFile 。このテクスチャが Director 外部で作成され、ファイルの読み込みかキャストメンバのロードによって作成されたことを示します。

例文

次のステートメントと実行結果は、" 冥王星 " というモデルのシェーダで使用されるテクスチャが、画像オブジェクトから作成されたことを示します。

```
put member("scene").model("Pluto").shader.texture.type
-- #fromImageObject
```

関連事項

newTexture

type （ウィンドウ）

使用法

```
-- Lingo syntax
windowObjRef.type

// JavaScript syntax
windowObjRef.type;
```

解説

ウィンドウのタイプを指定するウィンドウプロパティです。読み取り / 書き込み可能。

type プロパティを設定すると、新しいウィンドウに関連するすべてのプロパティがそれに応じて設定されます。

このプロパティは次のいずれかの値をとることができます。

プロパティ	解説
#document	ウィンドウに標準のタイトルバー、クローズボックス、および最小化と最大化のボックスが表示されるように指定します。これらの種類のウィンドウは移動できます。
#tool	ウィンドウが短いタイトルバーと右上隅の小さなクローズボックスのみと一緒に表示されるように指定します。#tool ウィンドウは常にアクティブなので、これらの種類のウィンドウはアクティブ化および非アクティブ化のイベントは受け付けません。これらの種類のウィンドウは、常に #document ウィンドウの上に重なって表示されます。
#dialog	ウィンドウが標準のタイトルバー、クローズボックス、アイコンなしで表示されるように指定します。これらの種類のウィンドウはモーダルで、常にすべてのウィンドウの一番手前に表示されます。

これらのプロパティは、ムービーオブジェクトの displayTemplate プロパティを使用してアクセスすることもできます。

ウィンドウのビヘイビアーは、type プロパティとムービーオブジェクトの dockingEnabled プロパティの値によっても異なります。

- dockingEnabled が TRUE に設定され、type が #document に設定されている場合、MIAW は Director のドキュメントウィンドウのように表示され、機能します。ウィンドウは「ビューポート」領域に表示され、ステージウィンドウ、スコアウィンドウ、キャストウィンドウ、メディアエディタ、メッセージウィンドウとドッキングできます。ただし、ウィンドウをこれらのウィンドウのいずれかとグループ化することはできません。
- dockingEnabled が TRUE に設定され、type が #tool に設定されている場合、MIAW は Director のツールウィンドウのように表示され、機能します。ウィンドウは、プロパティインスペクタおよびツールパレットを除くすべてのツールウィンドウとグループ化できます。
- dockingEnabled が TRUE に設定され、type が #fullscreen または #dialog に設定されている場合、type は無視され、ウィンドウはオーサリングウィンドウになります。

例文

次のステートメントは、" 惑星 " という名前のウィンドウのタイプを #tool に設定します。

```
-- Lingo syntax
window("Planets").type = #tool

// JavaScript syntax
window("Planets").type = "tool";
```

関連事項

appearanceOptions, displayTemplate, dockingEnabled, titlebarOptions, ウィンドウ

updateLock

使用法

```
-- Lingo syntax
_movie.updateLock

// JavaScript syntax
_movie.updateLock;
```

解説

ムービープロパティ。スコア録画時にステージを更新するか（FALSE）、更新しないか（TRUE）を示します。読み取り / 書き込み可能。

スコア録画時にステージの表示を一定に保つには、スクリプトがスコアを更新する前に updateLock ムービープロパティを TRUE に設定します。updateLock が FALSE に設定されている場合は、フレームに入るたびにステージが更新されて、新しいフレームが表示されます。

updateLock は、他のフレーム内のプロパティを検証するなどの理由で、フレームを一時的に離れる際に、予期しないスコアの更新が行われないようにする目的で使うこともできます。

このプロパティは、ランタイム中にフレームへの変更を隠すために使うこともできますが、その場合フィールドキャストメンバの変更は内容が変更された時点ですぐに表示されます。他のスプライトで、場所やメンバが変更されてもそれらの変更はこのプロパティがオフになるまで更新されないのとは異なっています。

例文

次のステートメントは、updateLock プロパティの値をメッセージウィンドウに表示します。

```
-- Lingo
put _movie.updateLock

// Javascript
trace(_movie.updateLock);
```

関連事項

ムービー

updateMovieEnabled

使用法

```
the updateMovieEnabled
```

解説

ムービーが別のムービーに分岐する際に、現在のムービーに加えられた変更を自動的に保存するか（TRUE）保存しないか（FALSE、デフォルト）を指定するムービープロパティです。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、別のムービーに分岐する際に、現在のムービーに加えられた変更を Director が自動的に保存するようにします。

```
-- Lingo
_movie.updateMovieEnabled = TRUE

// Javascript
_movie.updateMovieEnabled = true;
```

URL

使用法

```
-- Lingo syntax
memberObjRef.URL

// JavaScript syntax
memberObjRef.URL;
```

解説

キャストメンバプロパティ。Shockwave Audio（SWA）キャストメンバおよび Flash ムービーキャストメンバの URL を示します。

Flash ムービーメンバの場合、このプロパティは pathName メンバプロパティと同じ意味になります。詳細および例については、pathName メンバプロパティを参照してください。

この URL プロパティは、値を調べたり、変更できます。SWA メンバの場合は、SWA ストリーミングキャストメンバが停止したときにのみ設定することができます。

例文

次のステートメントは、インターネットサーバ上のファイルを、" ベニー・グッドマン " SWA キャストメンバの URL に設定します。

```
-- Lingo syntax
on mouseDown
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
end

// JavaScript syntax
function mouseDown() {
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
}
```

useAlpha

使用法

```
-- Lingo syntax
memberObjRef.useAlpha
imageObjRef.useAlpha

// JavaScript syntax
memberObjRef.useAlpha;
imageObjRef.useAlpha;
```

解説

ビットマップキャストメンバおよび画像オブジェクトのプロパティ。アルファチャネル情報を持つ 32 ビットのキャストメンバおよび画像オブジェクトに対し、Director がステージ上に画像を描画するときに、アルファ情報を使用するか（TRUE）、または無視するか（FALSE）を示します。

例文

次の例は、キャストメンバ "foreground" のアルファチャネルのオンとオフを切り替えます。

```
-- Lingo syntax
member("foreground").useAlpha = not(member("foreground").useAlpha)

// JavaScript syntax
switch(member("foreground").useAlpha) {
    case 0:
        member("foreground").useAlpha = 1;
        break;
    case 1:
        member("foreground").useAlpha = 0;
        break;
    }
}
```

useDiffuseWithTexture

使用法

```
member(whichCastmember).shader(whichShader).useDiffuseWithTexture
```

解説

3D 標準シェーダプロパティ。テクスチャの調整に拡散カラーを使用するか (TRUE)、または使用しないか (FALSE) を取得または設定します。

TRUE に設定すると、このプロパティは blendFunction プロパティおよび blendConstant プロパティと共に機能します。blendFunction が #blend に設定されている場合、拡散カラーはテクスチャカラーで重み付けされて、最終的なカラーが決定されます。例えば、blendFunction が #blend に設定されていて、blendConstant が 100.0 に設定されている場合、最終的なカラーは純粋なテクスチャカラーとなります。blendConstant を 0.0 に変更すると、最終的なカラーは拡散カラーになります。blendConstant を 10.0 に変更すると、最終的なカラーは 10% がテクスチャカラーで、90% が拡散カラーになります。

このプロパティのデフォルト値は FALSE です。

すべてのシェーダは #standard シェーダプロパティにアクセスできます。これらの標準シェーダプロパティに加えて、#engraver、#newsprint、および #painter というタイプのシェーダには、それぞれ独自のプロパティがあります。詳細については、「newShader」を参照してください。

例文

次の例で、モデル "MysteryBox" の shaderList には 6 つのシェーダが含まれています。各シェーダには、最大で 8 つのテクスチャをもつテクスチャリストが備わっています。キャストメンバ (Level2) の diffuseColor プロパティは rgb(255, 0, 0) に設定されます。6 つのすべてのシェーダでは、blendFunction プロパティが #blend に設定され、blendConstant プロパティが 80 に設定されます。次のステートメントは、MysteryBox で使用されるすべてのシェーダの useDiffuseWithTexture プロパティを TRUE に設定します。少量の赤色がモデルのサーフェスにブレンドされます。このプロパティは、blendFunction、blendFunctionList、blendSource、blendSourceList、blendConstant、および blendConstantList の各プロパティの設定による影響を受けます。

```
-- Lingo
member("Level2").model("MysteryBox").shaderlist.useDiffuseWithTexture = TRUE

// Javascript
member("Level2").getPropRef("model",1).getProp("shaderlist").useDiffuseWithTexture = true;
```

関連事項

blendFunction, blendConstant

useFastQuads

使用法

```
-- Lingo syntax
_movie.useFastQuads

// JavaScript syntax
_movie.useFastQuads;
```

解説

4 点演算で高速（TRUE）を使用するか、低速（FALSE、デフォルト）を使用するかを指定するムービープロパティです。読み取り / 書き込み可能。

TRUE に設定すると、Director は処理速度が速く精度の劣る方法で 4 点演算を行います。単純な図形回転や傾斜操作には、この高速演算が向いています。

FALSE に設定すると、速度が遅く精度の高い 4 点演算が使用されます。ゆがみや任意の効果に 4 点演算を行うときは、この演算の方が視覚的に良い結果が得られます。

単純なスプライト回転や傾斜操作では、この設定とは無関係にいつでも高速の 4 点演算方式が採用されます。こうした単純な操作では、useFastQuads を TRUE に設定しても、操作速度が向上するわけではありません。

例文

次のステートメントは、このムービーの中のすべての 4 点演算で高速の演算を使用するよう設定します。

```
- Lingo syntax
_movie.useFastQuads = TRUE

// JavaScript syntax
_movie.useFastQuads = true;
```

関連事項

ムービー

useHypertextStyles

使用法

```
-- Lingo syntax
memberObjRef.useHypertextStyles

// JavaScript syntax
memberObjRef.useHypertextStyles;
```

解説

テキストキャストメンバプロパティ。テキストキャストメンバ内のハイパーテキストリンクの表示を制御します。

useHypertextStyles が TRUE に設定されていると、すべてのリンクが自動的に下線つきで青く表示されます。ポインタ (カーソル) は、リンクの上に置かれると、人差し指の形状に変わります。

このプロパティを FALSE に設定すると、自動的にフォーマットしてポインタを変更する機能がオフになります。

例文

次のビヘイビアーは、テキストキャストメンバ "myText" 内のハイパーテキストのフォーマットのオンとオフを切り替えます。

```
--Lingo syntax
on mouseUp
    member("myText").usehypertextStyles = not(member("myText").usehypertextStyles)
end

// JavaScript syntax
function mouseUp() {
    member("myText").usehypertextStyles = !(member("myText").usehypertextStyles)
}
```

useLineOffset

使用法

```
member(whichCastmember).model(whichModel).toon.useLineOffset
member(whichCastmember).model(whichModel).inker.useLineOffset
```

解説

3D toon モディファイアおよび inker モディファイアのプロパティ。モデルのサーフェスに線を描画するときに、そのモディファイアの lineOffset プロパティが使用されるかどうかを示します。

このプロパティのデフォルト値は FALSE です。

例文

次のステートメントは、" ティーポット " というモデルの toon モディファイアの useLineOffset プロパティを FALSE に設定します。toon モディファイアの lineOffset プロパティの効力がなくなります。

```
-- Lingo syntax
member("tp").model("Teapot").toon.useLineOffset = FALSE

// JavaScript syntax
member("tp").getPropRef("model",1).getProp("toon").useLineOffset = false;
```

関連事項

lineOffset

userData

使用法

```
member(whichCastmember).model(whichModel).userData
member(whichCastmember).light(whichLight).userData
member(whichCastmember).camera(whichCamera).userData
member(whichCastmember).group(whichCamera).userData
```

解説

3D プロパティ。モデル、グループ、カメラ、またはライトの userData プロパティのリストを返します。Director の外部で作成されたオブジェクトの場合、このプロパティのデフォルト値は、3D モデリングツールでモデルの userData プロパティに設定された全プロパティのリストです。Director 内で作成されたオブジェクトの場合、このプロパティのデフォルト値は、空のプロパティリスト [:] です。ただし、複製コマンドで作成されたオブジェクトを除きます。複製コマンドを使用してオブジェクトを作成した場合、作成されたオブジェクトの userData プロパティのデフォルトは、元のソースオブジェクトのプロパティと同じ値に設定されます。

このリストのエレメントを変更するには、addProp コマンドおよび deleteProp コマンドを使用する必要があります。これらのコマンドは、メインの Lingo 辞書に記載されています。

例文

次のステートメントは、" 新しいボディ " というモデルの userData プロパティを表示します。

```
put member("Car").model("New Body").userData
-- [#driver: "Bob", #damage: 34]
```

次のステートメントは、" プレーヤ " というモデルの userData プロパティリストに、値 100 が設定されたプロパティ #health を追加します。

```
member("scene").model("Player").userData.addProp(#health,100)
```

userName

使用法

```
-- Lingo syntax
_player.userName

// JavaScript syntax
_player.userName;
```

解説

プレーヤプロパティ。Director をインストールしたときに入力したユーザ名を含むストリングです。読み取り専用。

このプロパティはオーサリング環境でのみ使用できます。このプロパティは、ユーザ情報を表示するようにカスタマイズできるウィンドウ内ムービー（MIAW）ツールで使用できます。

例文

次のハンドラは、ウィンドウが開いたときに、表示フィールドにユーザ名とシリアル番号を入力します。このハンドラを置く場所としては、MIAW 内のムービースクリプトが適しています。

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN & _player.serialNumber
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" + _player.serialNumber;
    member("User Info").text = displayString;
}
```

関連事項

プレーヤ

userName (RealMedia)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.userName

// JavaScript syntax
memberOrSpriteObjRef.userName;
```

解説

RealMedia スプライトおよびキャストメンバプロパティ。保護されている RealMedia ストリームへのアクセスに必要なユーザ名を設定します。セキュリティ上の理由により、このプロパティでは、既に指定されているユーザ名を取得することはできません。ユーザ名が設定されている場合は、このプロパティの値は "********" というストリングになります。このプロパティのデフォルト値は空のストリングで、ユーザ名が指定されていないことを意味します。

例文

次の例は、キャストメンバ "Real" またはスプライト "2" の RealMedia ストリームにユーザ名が設定されていることを示します。

```
-- Lingo syntax
put(sprite(2).userName) -- "********"
put(member("Real").userName) -- "********"

// JavaScript syntax
put(sprite(2).userName); // "********"
put(member("Real").userName); // "********"
```

次の例は、キャストメンバ "Real" またはスプライト "2" の RealMedia ストリームにユーザ名が設定されていないことを示します。

```
-- Lingo syntax
put(sprite(2).userName) -- ""
put(member("Real").userName) -- ""

// JavaScript syntax
put(sprite(2).userName); // ""
put(member("Real").userName); // ""
```

次の例は、キャストメンバ "Real" またはスプライト "2" の RealMedia ストリームのユーザ名を Marcelle に設定します。

```
-- Lingo syntax
member("Real").userName = "Marcelle"
sprite(2).userName = "Marcelle"

// JavaScript syntax
member("Real").userName = "Marcelle";
sprite(2).userName = "Marcelle";
```

関連事項

password

useTargetFrameRate

使用法

```
sprite(which3dSprite).useTargetFrameRate
```

解説

3D スプライトプロパティ。スプライトの targetFrameRate プロパティが強制されているかどうかを示します。useTargetFrameRate プロパティが TRUE に設定されていると、指定されたフレームレートを維持するために必要な場合に、Director はモデルのポリゴン数を減らします。

例文

次のステートメントは、スプライト 3 の targetFrameRate プロパティを 45 に設定します。また、スプライトの useTargetFrameRate プロパティを TRUE に設定することにより、このフレームレートを強制します。

```
-- Lingo syntax
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// JavaScript syntax
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

関連事項

targetFrameRate

vertex

使用法

```
-- Lingo syntax
memberObjRef.vertex[whichVertexPosition]

// JavaScript syntax
memberObjRef.vertex[whichVertexPosition];
```

解説

ベクトルシェイプキャストメンバの頂点リストの一部に直接アクセスできるようにするチャンク式です。

このチャンク式を使用すると、頂点リストの異なるチャンクのシンタックスを解析する必要がありません。このタイプのチャンク式を使用すると、頂点リストの値を調べたり、設定することができます。

例文

次のコードは、メンバ内の頂点の数を決める方法を示します。

```
-- Lingo syntax
put(member("Archie").vertex.count) -- 2

// JavaScript syntax
put(member("Archie").vertex.count); // 2
```

メンバの 2 番目の頂点を取得するには、次のシンタックスを使います。

```
-- Lingo syntax
put(member("Archie").vertex[2]) -- point(66.0000, -5.0000)

// JavaScript syntax
put(member("Archie").vertex[2]); // point(66.0000, -5.0000)
```

次の方法で、制御ハンドルに値を設定することもできます。

```
-- Lingo syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000)

// JavaScript syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000);
```

関連事項

vertexList

vertexList

使用法

```
-- Lingo syntax
memberObjRef.vertexList

// JavaScript syntax
memberObjRef.vertexList;
```

解説

ベクトルシェイプの各頂点に 1 つずつ、プロパティリストが含まれる線形リストを返すキャストメンバプロパティです。このプロパティリストには、頂点の位置と制御ハンドルの位置が入っています。位置が (0,0) の場合、制御ハンドルはありません。

各頂点には、その頂点と隣接する頂点との間の曲線を決定する 2 つの制御ハンドルがあります。vertexList では、頂点の制御ハンドルの座標値は、シェイプの座標システムでの絶対位置ではなく、常にその頂点から見た位置になります。頂点の 1 つ目の制御ハンドルがその頂点から 10 ピクセル左方にあると、その位置は (-10, 0) として保存されます。したがって、頂点の位置が Lingo によって変更されると、制御ハンドルはその頂点とともに移動するため、更新の必要はありません (ユーザが特にハンドルの位置またはサイズを変更したい場合は別です)。

このプロパティを変更するときは、値のいずれかを変更した後でリストの内容をリセットする必要があります。これは、このプロパティの値に変数を設定する場合、その変数内でリストそのものではなくリストのコピーを設定していることになるからです。変更を有効にするには、次のようなコードを記述します。

```
- Get the current property contents
currVertList = member(1).vertexList
-- Add 25 pixels to the horizontal and vertical positions of the first vertex in the list
currVertList[1] .vertex = currVertList[1] .vertex + point(25, 25)
-- Reset the actual property to the newly computed position
member(1).vertexList = currVertList
```

例文

次のステートメントは、2 つの頂点を持つ弓形の線を描く vertextList の値を表示します。

```
-- Lingo syntax
put(member("Archie").vertexList)-- [[#vertex: point(-66.0000, 37.0000), #handle1: point(-70.0000, -
36.0000), #handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -5.0000), #handle1:
point(121.0000, 56.0000), #handle2: point(11.0000, -66.0000)]]

// JavaScript syntax
put(member("Archie").vertexList);//[[#vertex: point(-66.0000, 37.0000),#handle1: point(-70.0000, -
36.0000),#handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -5.0000),#handle1: point(121.0000,
56.0000), #handle2: point(11.0000, -66.0000)]]
```

関連事項

addVertex(), count(), deleteVertex(), moveVertex(), moveVertexHandle(), originMode, vertex

vertexList (mesh generator)

使用法

```
member(whichCastmember).modelResource(whichModelResource).vertexList
```

解説

3D プロパティ。#mesh というタイプのモデルリソースを指定すると、そのモデルリソースの vertexList プロパティを取得または設定できます。

vertexList は、メッシュで使用される各頂点のリニアリストです。1 つの頂点をメッシュの多数のフェイスで共有できます。このプロパティには任意のサイズのリストを指定できます。しかし、このプロパティに格納されるのは、newMesh() コマンドを使用して #mesh モデルリソースを作成するときに指定されたアイテム数だけです。

例文

次のステートメントは、" 三角形 " というモデルリソースの vertexList を設定します。

```
member("Shapes").modelResource("Triangle").vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

関連事項

newMesh, face, vertices

vertexList (mesh deform)

使用法

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].vertexList
```

解説

3D プロパティ。#meshDeform モディファイアが割り当てられたモデルを指定すると、そのモデル内で指定されたメッシュの vertexList プロパティを取得または設定できます。

vertexList は、指定されたメッシュで使用される各頂点のリニアリストです。1 つの頂点をメッシュの多数のフェイスで共有できます。

モデルが #meshDeform モディファイアに加えて #sds モディファイアまたは #lod モディファイアを利用する場合は、#sds モディファイアまたは #lod モディファイアの影響で、このプロパティの値が変化することに注意してください。

例文

次のステートメントは、" 三角形 " というモデルの最初のメッシュについて、#meshDeform モディファイアの vertexList を表示します。

```
put member("Shapes").model("Triangle").meshDeform.mesh[1].vertexList
-- [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

関連事項

face, vertices, mesh （プロパティ）

vertices

使用法

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].vertices
```

解説

3D フェイスプロパティ。#mesh というタイプのモデルリソースを指定すると、フェイスインデックスで指定されたメッシュフェイスに対し、リソースの vertexList のどの頂点を使用するかを取得または設定できます。

このプロパティは、3 つの整数によるリニアリストです。各整数は、指定されたフェイスを構成するメッシュの vertexList プロパティに含まれる 3 つの頂点のインデックス位置に対応しています。

外向きのサーフェス法線を作成するには、反時計回りの順序でこれらの頂点をリストに指定する必要があります。

このプロパティを変更する場合、または generateNormals() コマンドを使用する場合は、build() コマンドを呼び出してメッシュを再構築する必要があります。

例文

次の例は、メッシュモデルリソース "SimpleSquare" の vertexList を表示します。次に、そのメッシュの 2 番目のフェイスの vertices プロパティを表示します。

```
put member("3D").modelResource("SimpleSquare").vertexList
-- [vector( 0.0000, 0.0000, 0.0000), vector( 0.0000, 5.0000, 0.0000), vector( 5.0000, 0.0000, 0.0000),
vector( 5.0000, 5.0000, 0.0000)]
put member("3D").modelResource("SimpleSquare").face[1].vertices
-- [3, 4, 1]
```

関連事項

face, vertexList (mesh deform), generateNormals()

video (QuickTime, AVI)

使用法

```
member(whichCastMember).video
the video of member whichCastMember
```

解説

指定したデジタルビデオキャストメンバのグラフィック画像を、再生するか (TRUE または 1)、再生しないか (FALSE または 0) を指定するためのデジタルビデオキャストメンバプロパティです。

このプロパティは、デジタルビデオキャストメンバのビジュアルエレメントにのみ作用します。例えば、video が FALSE に設定されている場合は、デジタルビデオのサウンドトラック（存在する場合）の再生が続行されます。

例文

次のステートメントは、キャストメンバ " インタビュー " に関連するビデオの再生をオフにします。

```
-- Lingo syntax
member("Interview").video = FALSE

// JavaScript syntax
member("Interview").video = false;
```

関連事項

setTrackEnabled(), trackEnabled

video (MP4Media/FLV)

使用法

```
member(1).video = true
sprite(1).video = true
```

解説

MP4Media/FLV キャストメンバおよびスプライトのプロパティ。指定した MP4Media/FLV のビデオキャストメンバを再生するか（True または 1）しないか（False または 0）を指定します。

このプロパティは、デジタルビデオキャストメンバのビジュアルエレメントにのみ作用します。デジタルビデオのサウンドトラックの再生が続行されます。

例文

以下の例は、キャストメンバ Interview に関連付けられたビデオをオフにします。

```
-- Lingo syntax
member("Interview").video = FALSE

// JavaScript syntax
member("Interview").video = false;
```

video (RealMedia, Windows Media)

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.video

// JavaScript syntax
memberOrSpriteObjRef.video;
```

解説

RealMedia と Windows Media のプロパティ。スプライトまたはキャストメンバがビデオをレンダリングするか（TRUE または 1）、サウンドだけを再生するか（FALSE または 0）を取得または設定できます。読み取り / 書き込み可能。

1 または 0 以外の整数値は、TRUE として扱われます。

RealMedia、または Windows Media キャストメンバのオーディオコンポーネントの再生中にビデオをオフにする、または再生中のビデオのオンとオフを切り替えるときに、このプロパティを使用します。

例文

次の例は、スプライト "2" およびキャストメンバ "Real" の video プロパティを TRUE に設定します。

```
-- Lingo syntax
put(sprite(2).video) -- 1
put(member("Real").video) -- 1

// JavaScript syntax
put(sprite(2).video); // 1
put(member("Real").video); // 1
```

次の例は、スプライト 2 およびキャストメンバ Real の RealMedia ビデオエレメントの video プロパティを FALSE に設定します。

```
-- Lingo syntax
sprite(2).video = FALSE
member("Real").video = FALSE

// JavaScript syntax
sprite(2).video = 0;
member("Real").video = 0;
```

videoFormat

使用法

```
-- Lingo syntax
dvdObjRef.videoFormat

// JavaScript syntax
dvdObjRef.videoFormat;
```

解説

DVD プロパティ。ビデオフォーマットを示す記号を返します。読み取り専用。

返されるシンボルは次のとおりです。

シンボル	解説
#MPEG1	ビデオフォーマットは MPEG-1 です。
#MPEG2	ビデオフォーマットは MPEG-2 です。
#unknown	ビデオフォーマットは不明です。

例文

次のステートメントは、DVD オブジェクトスプライトのビデオフォーマットを返します。

```
-- Lingo syntax
put sprite(1).videoFormat

// JavaScript syntax
trace(sprite(1).videoFormat)
```

関連事項

DVD

videoForWindowsPresent

使用法

```
the videoForWindowsPresent
```

解説

システムプロパティ。Video for Windows（AVI）がコンピュータにインストールされているかどうかを示します。

このプロパティは、値を調べられますが、設定はできません。

例文

次のステートメントは、Video for Windows がコンピュータにインストールされているかどうかを調べ、インストールされていない場合は、" 代替場面 " マーカーの位置まで再生ヘッドを移動させます。

```
if the videoForWindowsPresent= FALSE then go to "Alternate Scene"
```

関連事項

QuickTimeVersion()

viewH

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.viewH

// JavaScript syntax
memberOrSpriteObjRef.viewH;
```

解説

Flash ムービーとベクトルシェイプの表示位置の左右の座標値（ピクセル単位で指定）を制御するキャストメンバおよびスプライトプロパティです。この値には、浮動小数を指定できます。デフォルト値は 0 です。

Flash ムービーの表示位置は原点を基準に設定されます。

viewH に正数を設定すると、ムービーはスプライト内の左方に移動します。負数を設定すると、ムービーは右方に移動します。したがって、viewH プロパティを変更することにより、ムービーの端を切り取ったり、ムービー全体を表示から削除する効果を出すことができます。

このプロパティは、値を調べたり、設定することができます。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のハンドラは、スプライト参照をパラメータとして取り、Flash ムービースプライトの表示を、そのスプライトの境界矩形内で左方から右方へ移動します。

```
-- Lingo syntax
on panRight whichSprite
    repeat with i = 120 down to -120
        sprite(whichSprite).viewH = i
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panRight(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewH = i;
        _movie.updateStage();
        i--;
    }
}
```

関連事項

scaleMode, viewV, viewPoint, viewScale

viewPoint

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.viewPoint

// JavaScript syntax
memberOrSpriteObjRef.viewPoint;
```

解説

スプライトの境界矩形の中心に表示される Flash ムービーまたはベクトルシェイプ内の位置をピクセル単位で制御するための、キャストメンバプロパティおよびスプライトプロパティです。値は整数で指定します。

キャストメンバの表示位置を変更しても、スプライトの境界矩形内のムービーの表示が変更されるだけで、ステージ上でのスプライトの位置は変わりません。表示位置は、スプライトの境界矩形の中心に表示されるキャストメンバ内部の座標値で、常にムービーの原点（originPoint、originH、originV プロパティによって設定される）から見た位置として表されます。例えば、Flash ムービーの表示位置を Point(100,100) に設定すると、スプライトの中心は、Flash ムービー内の、原点から右方に 100 Flash ムービーピクセル単位、下方に 100 Flash ムービーピクセル単位の位置になります。これは、原点をどこに移動しても変わりません。

viewPoint プロパティは Director の点を示す値として指定します。例えば、point (100,200) と指定します。Flash ムービーの表示位置を viewPoint プロパティによって指定することは、viewH と viewV プロパティをそれぞれ設定することと同じです。例えば、viewPoint プロパティを point (50,75) に設定することは、viewH プロパティを 50、viewV プロパティを 75 に設定することと同じです。

viewPoint プロパティに指定する Director のポイント値は整数に限られますが、viewH と viewV には浮動小数を指定できます。viewPoint プロパティを調べる場合、ポイント値は整数に切り捨てられます。一般的には、viewH と viewV プロパティを使う方が正確です。スピードと便利さを優先する場合は、originPoint プロパティを使ってください。

このプロパティは、値を調べたり、設定することができます。デフォルト値は point (0,0) に設定されています。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のハンドラは、指定した Flash ムービースプライトを、下方および右方に 5 Flash ムービーピクセル単位で増分しながら移動します。

```
-- Lingo syntax
on panAcross(whichSprite)
    repeat with i = 1 to 10
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -5)
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panAcross(whichSprite) {
    var i = 1;
    while(i < 11) {
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -5);
        _movie.updateStage();
        i++
    }
}
```

関連事項

scaleMode, viewV, viewH, viewScale

viewScale

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.viewScale

// JavaScript syntax
memberOrSpriteObjRef.viewScale;
```

解説

スプライトの境界矩形内にある Flash ムービーまたはベクトルシェイプスプライトの表示のスケーリングの総量を設定するための、キャストメンバプロパティおよびスプライトプロパティです。この総量は、浮動小数を使ってパーセンテージで指定します。デフォルト値は 100 です。

スプライトの矩形そのものはスケーリングされず、矩形内部のキャストメンバの表示のみがスケーリングされます。スプライトの viewScale プロパティの設定は、カメラのレンズを選択するのに似ています。viewScale の値が小さくなると、スプライト内部のムービーの表示サイズは大きくなり、値が大きくなるとサイズは小さくなります。例えば、viewScale プロパティを 200% に設定すると、スプライト内部の表示は 1 回表示された大きさの 2 倍のサイズで表示され、スプライト内部のキャストメンバは元のサイズの半分のサイズで表示されます。

viewScale と scale プロパティでの大きな相違点は、viewScale は常にスプライトの境界矩形の中心からスケーリングしますが、scale は Flash ムービーの originMode プロパティによって特定される位置からスケーリングすることです。

このプロパティは、値を調べたり、設定することができます。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のスプライトスクリプトは、Flash ムービースプライトを設定し、その表示の大きさを 2 倍にします。

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).viewScale = 200
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).viewScale = 200;
}
```

関連事項

scaleMode, viewV, viewPoint, viewH

viewV

使用法

```
-- Lingo syntax
memberOrSpriteObjRef.viewV

// JavaScript syntax
memberOrSpriteObjRef.viewV;
```

解説

Flash ムービーとベクトルシェイプの表示位置の上下の座標値（ピクセル単位で指定）を制御するキャストメンバおよびスプライトプロパティです。この値には、浮動小数を指定できます。デフォルト値は 0 です。

Flash ムービーの表示位置は原点を基準に設定されます。

viewV に正数を設定すると、ムービーはスプライト内の上方に移動します。負数を設定すると、ムービーは下方に移動します。このため、viewV プロパティを変更することにより、ムービーの端を切り取ったり、ムービー全体を表示から削除するなどの効果を得られます。

このプロパティは、値を調べたり、設定することができます。

注意：scaleMode プロパティが #autoSize に設定されているとき、このプロパティにはデフォルト値を設定する必要があります。これ以外の値を設定すると、スプライトが正しく表示されません。

例文

次のハンドラは、スプライト参照をパラメータとして取り、Flash ムービースプライトの表示を、そのスプライトの境界矩形内で上端から下端へ移動します。

```
-- Lingo syntax
on panDown(whichSprite)
    repeat with i = 120 down to -120
        sprite(whichSprite).viewV = i
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panDown(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewV = i;
        _movie.updateStage();
        i--;
    }
}
```

関連事項

scaleMode, viewPoint, viewH

visible

使用法

```
-- Lingo syntax
windowObjRef.visible

// JavaScript syntax
windowObjRef.visible;
```

解説

ウィンドウプロパティ。ウィンドウが表示可能であるか（TRUE）、表示可能でないか（FALSE）を示します。読み取り / 書き込み可能。

例文

次のステートメントは、" 制御パネル " というウィンドウを表示します。

```
-- Lingo syntax
window("Control_Panel").visible = TRUE

// JavaScript syntax
window("Control_Panel").visible = true;
```

関連事項

ウィンドウ

visible （スプライト）

使用法

```
sprite(whichSprite).visible
the visible of sprite whichSprite
```

解説

[スプライト] で指定したスプライトが表示されているか (TRUE)、表示されていないか (FALSE) を調べるためのスプライトプロパティです。このプロパティは、チャネル内のすべてのスプライトに、それらのスプライトのスコアでの位置にかかわらず影響します。

注意：スプライトチャネルの visible プロパティを FALSE に設定すると、そのスプライトは表示されなくなり、マウス関連のイベントのみがそのチャネルに送信されなくなります。beginSprite、endSprite、prepareFrame、enterFrame、exitFrame などのイベントはスプライトの表示レベルにかかわらず送信されます。ただし、スコアのチャネル上で [ミュート] ボタンをクリックすると、visible プロパティが FALSE に設定され、さらにすべてのイベントがそのチャネルに送信されなくなります。ミューティングを使用するとチャネルが使用不能になりますが、スプライトの visible プロパティを FALSE に設定しても、グラフィカルプロパティに影響するだけです。

このプロパティは、値を調べたり、設定することができます。このプロパティが FALSE に設定されていると、スプライトが終了しても自動的に TRUE にリセットされません。チャネルを使う他のメンバを表示するには、スプライトの visible プロパティを TRUE に設定しておく必要があります。

例文

次のステートメントは、スプライト 8 が表示されるようにします。

```
sprite(8).visible = TRUE
```

visibility

使用法

```
member(whichCastmember).model(whichModel).visibility
modelObjectReference.visibility
```

解説

3D プロパティ。このプロパティを使用すると、参照されたモデルの visibility プロパティを取得または設定できます。このプロパティを使用して、モデルのジオメトリを描画する方法を指定できます。このプロパティは、次のいずれかの値になります。

- #none。ポリゴンは描画されず、モデルは表示されません。
- #front。カメラに向いているポリゴンのみが描画されます。この方法は陰面消去と呼ばれ、レンダリング速度を最適化します。このプロパティのデフォルト設定です。
- #back。カメラとは異なる方向に向いているポリゴンのみが描画されます。この設定は、モデルの内側を描画するときに使用します。また、法線の計算に異なる簡易値を用いるファイル形式からモデルを読み込んだ場合など、モデルが正常に描画されないときもこの設定を使用します。
- #both。すべてのポリゴンの両面が描画されます。見る方向に関係なく平面を見たい場合や、正常に描画されていないモデルには、この設定を使用します。

例文

次のステートメントは、モデル "Monster02" の visibility プロパティが #none に設定されていることを示します。モデルは表示されません。

```
-- Lingo syntax
put member("3D").model("Monster02").visibility
-- #none

// JavaScript syntax
trace(member("3D").getPropRef("model",1).visibility)
// symbol("none")
```

volume (DVD)

使用法

```
-- Lingo syntax
dvdObjRef.volume

// JavaScript syntax
dvdObjRef.volume;
```

解説

DVD プロパティ。現在の DVD のサウンドの音量を指定します。読み取り / 書き込み可能。

音量は 0（静か）から 100（フルボリューム）の範囲の整数に設定する必要があります。

Windows の音量の基準は対数です。Mac では基準はリニアです。

例文

次のステートメントは、DVD メンバ音量を設定します。

```
-- Lingo syntax
member(1).volume = 20

// JavaScript syntax
member(1).volume = 20;
```

関連事項

DVD

volume （メンバ）

使用法

```
-- Lingo syntax
memberObjRef.volume

// JavaScript syntax
memberObjRef.volume;
```

解説

指定した Shockwave Audio (SWA) ストリーミングキャストメンバの音量を特定するための SWA キャストメンバプロパティです。値の範囲は 0 ～ 255 です。

このプロパティは、値を調べたり、設定することができます。

例文

次のステートメントは、SWA ストリーミングキャストメンバ ”SWAfile” の音量を最大値の半分に設定します。

```
-- Lingo syntax
member("SWAfile").volume = 128

// JavaScript syntax
member("SWAfile").volume = 128;
```

volume （ミクサ）

使用法

```
mixer.volume
```

解説

ミクサの音量を指定するオーディオミクサプロパティです。値 255 は最大音量を示し、0 は無音（ミュート）を示します。

例文

```
--Lingo syntax
on mouseUp me
    mixerRef.volume = 120
end

// JavaScript syntax
function mouseUp(){
mixerRef.volume = 120;
}
```

関連事項

mute （ミクサ）, unmute （ミクサ）, ミクサ

volume (MP4Media/FLV)

使用法

```
sprite(1).volume
```

解説

MP4Media/FLV スプライトのプロパティ。MP4Media/FLV スプライトの音量を指定します。

このプロパティの値の範囲は、0（無音）～ 255（大）の整数です。

注意： このメソッドを使用する前に、audio プロパティを True に設定します。

例文

以下の例は、スプライト 7 の volume を 200 に設定します。

```
-- Lingo syntax
sprite(7).volume = 200

// JavaScript syntax
sprite(7).volume = 200;
```

volume （サウンドチャネル）

使用法

```
-- Lingo syntax
soundChannelObjRef.volume

// JavaScript syntax
soundChannelObjRef.volume;
```

解説

サウンドチャネルの音量を指定するサウンドチャネルプロパティです。読み取り / 書き込み可能。

サウンドチャネルには、1、2、3、というように 8 までの番号が付いています。チャネル 1 と 2 はスコア内に表示されます。

volume プロパティの値の範囲は、0（無音）～ 255（最大）です。255 は、マシン上で実現できる最大の音量です。音量はサウンドオブジェクトの soundLevel プロパティで制御し、255 より低い値は、総音量に対してスケーリングされます。このプロパティを使って、複数のチャネルに、使用できる範囲内でそれぞれ別個の設定を割り当てることができます。

volume サウンドプロパティの値が低いほど、静電気音や雑音がより増えます。soundLevel を使うと雑音が減りますが、volume サウンドプロパティほどの制御は行えません。

完成したムービーでの volume の使用例については、Director アプリケーションフォルダ内の Learning\\Lingo_Examples フォルダにある sound control.dir を参照してください。

例文

次のステートメントは、サウンドチャネル 2 の音量を 130 (中間音量) に設定します。

```
-- Lingo syntax
sound(2).volume = 130

// JavaScript syntax
sound(2).volume = 130;
```

関連事項

サウンドチャネル , soundLevel

volume （サウンドオブジェクト）

使用法

```
soundObject.volume
```

解説

サウンドオブジェクトプロパティ。サウンドオブジェクトの音量（0 ～ 255 の範囲）を返すか設定します。値 255 は最大音量を示し、0 は無音（ミュート）を示します。

このプロパティは読み書き可能です。

例文

```
soundObject.volume = 120
--Lingo syntax
on mouseUp me
    put soundObjRef.volume -- Displays the volume of the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.volume) ; // Displays the volume of the sound object
// associated with soundobjectRef.
}
```

関連事項

mute（サウンドオブジェクト）, unmute（サウンドオブジェクト）

volume（スプライト）

使用法

```
-- Lingo syntax
spriteObjRef.volume

// JavaScript syntax
spriteObjRef.volume;
```

解説

名前または番号で指定したデジタルビデオムービーまたは Windows Media キャストメンバの音量を制御する際に使用するスプライトプロパティです。値の範囲は 0 ～ 256 です。0 以下の値を設定すると無音になります。値が 256 を超えると、大きな音量になり、音にかなりのひずみが生じます。

例文

次のステートメントは、スプライトチャネル 7 で再生する QuickTime ムービーの音量を最大の 256 に設定します。

```
-- Lingo syntax
sprite(7).volume = 256

// JavaScript syntax
sprite(7).volume = 256;
```

関連事項

soundLevel

volume (Windows Media)

使用法

```
-- Lingo syntax
windowsMediaObjRef.volume

// JavaScript syntax
windowsMediaObjRef.volume;
```

解説

Windows Media のスプライトプロパティ。Windows Media スプライトの音量を指定します。

このプロパティの値の範囲は、0（無音）～ 7（大）の整数です。

Director の制御／音量メニューを使用して、このプロパティを設定することもできます。

例文

次のステートメントは、スプライト の値を 7 から 2 に設定します。

```
-- Lingo syntax
sprite(7).volume = 2

// JavaScript syntax
sprite(7).volume = 2;
```

関連事項

Windows Media

warpMode

使用法

```
-- Lingo syntax
spriteObjRef.warpMode

// JavaScript syntax
spriteObjRef.warpMode;
```

解説

パノラマ上で実行するねじりのタイプを特定するための QuickTime VR スプライトプロパティです。

値は、#full、#partial、#none のいずれかになります。

このプロパティは、値を調べたり、設定することができます。値を調べる場合に、スタティックモードとダイナミックモードの値が異なるときは、このプロパティの値は現行モードに設定された値になります。値を設定する場合は、このプロパティはスタティックモードとダイナミックモードの両方のねじりを設定します。

例文

次の例では、スプライト 1 の warpMode を #full に設定します。

```
-- Lingo syntax
sprite(1).warpMode = #full

// JavaScript syntax
sprite(1).warpMode = symbol("full");
```

width

使用法

```
-- Lingo syntax
memberObjRef.width
imageObjRef.width
spriteObjRef.width

// JavaScript syntax
memberObjRef.width;
imageObjRef.width;
spriteObjRef.width;
```

解説

ベクトルシェイプ、Flash、アニメーション GIF、RealMedia、Windows Media、ビットマップ、シェイプキャストメンバに対して使用し、キャストメンバの幅をピクセル単位で示すメンバ、画像、およびスプライトプロパティです。キャストメンバおよび画像オブジェクトについては読み取り専用、スプライトについては読み書き可能です。

このプロパティは、フィールドキャストメンバおよびボタンキャストメンバには影響しません。

例文

次のステートメントは、メンバ 50 の幅を変数 theHeight に割り当てます。

```
-- Lingo syntax
theHeight = member(50).width

// JavaScript syntax
var theHeight = member(50).width;
```

次のステートメントは、スプライト 10 の幅を 26 ピクセルに設定します。

```
-- Lingo syntax
sprite(10).width = 26

// JavaScript syntax
sprite(10).width = 26;
```

次のステートメントは、スプライト i + 1 の幅を変数 howWide に割り当てます。

```
-- Lingo syntax
howWide = sprite(i + 1).width

// JavaScript syntax
var howWide = sprite(i + 1).width;
```

関連事項

height, image （画像）, メンバ, スプライト

width (3D)

使用法

```
member(whichCastmember).modelResource(whichModelResource).width
modelResourceObjectReference.width
```

解説

3D プロパティ。#box または #plane タイプのモデルリソースに対し、プレーンの幅を取得または設定できます。このプロパティは 0.0 より大きな値であることが必要です。デフォルト設定は 1.0 です。#box タイプのオブジェクトの場合、width のデフォルト値は 50.0 です。#plane タイプのオブジェクトの場合、デフォルト値は 1.0 です。width は X 軸に沿って測定されます。

例文

次のステートメントは、モデルリソース " 芝生のプレーン " の幅を 250.0 に設定します。

```
-- Lingo syntax
member("3D World").modelResource("Grass plane").width = 250.0

// JavaScript syntax
Member("3D World").getPropRef("modelResource",1).width = 250.0;
```

width (MP4Media/FLV)

使用法

```
put member(1).width
Sprite(1).width = 50
```

解説

メンバおよびスプライトのプロパティ。MP4Media/FLV キャストメンバの場合、このメソッドは作成者が作成した元のビデオのステージ上での幅を、ピクセルの単位で指定します。このプロパティは、キャストメンバの場合は読み取り専用で、スプライトの場合は読み書き可能です。

例文

次のシンタックスは、MP4Media/FLV のキャストメンバであるメンバ 50 の幅を、変数 theHeight: に代入します。

```
-- Lingo syntax
theHeight = member(50).width

// JavaScript syntax
var theHeight = member(50).width;
```

次の構文は、スプライト 10 の幅を 26 ピクセルに設定します。

```
-- Lingo syntax
sprite(10).width = 26

// JavaScript syntax
sprite(10).width = 26;
```

widthVertices

使用法

```
member(whichCastmember).modelResource(whichModelResource).
widthVertices
modelResourceObjectReference.widthVertices
```

解説

3D プロパティ。#box または #plane タイプのモデルリソースの X 軸上の頂点の数を整数で取得または設定できます。このプロパティの値は 2 以上であることが必要です。デフォルト値は 2 です。

例文

次のステートメントは、モデルリソース " タワー " の widthVertices プロパティを 10 に設定します。X 軸に沿った 18 個のポリゴン（2 * (10-1) 個の三角形）によって、モデルリソースのジオメトリが定義されます。

```
member("3D World").modelResource("Tower").widthVertices = 10
```

wind

使用法

```
member(whichCastmember).modelResource(whichModelResource).wind
modelResourceObjectReference.wind
```

解説

3D プロパティ。#particle というタイプのモデルリソースの wind プロパティをベクトルで取得または設定できます。

wind プロパティでは、各シミュレーションステップ内のすべてのパーティクルに適用する風の方向と強さを定義できます。このプロパティのデフォルト値は vector(0, 0, 0) です。これは風がまったく適用されていないことを示します。

例文

```
-- Lingo syntax
put member("3D").modelResource("fog bank").wind
-- vector(10.5,0,0)

// JavaScript syntax
trace(member("3D").getPropRef("modelResource","1").wind)
// vector(10.5,0,0)
```

window

使用法

```
-- Lingo syntax
_player.window[windowNameOrNum]

// JavaScript syntax
_player.window[windowNameOrNum];
```

解説

Director プレーヤ により作成されたウィンドウオブジェクトへのインデックスまたは名前によるアクセスを提供するプレーヤおよびムービープロパティです。読み取り専用。

引数 **windowNameOrNum** は、アクセスするウィンドウの名前を指定するストリング、またはアクセスするウィンドウのインデックスの位置を指定する整数です。

このプロパティの機能は、window() メソッドのトップレベルと同じです。

例文

次のステートメントは、変数 myWindow に 3 番目のウィンドウオブジェクトを設定します。

```
-- Lingo syntax
myWindow = _player.window[3]

// JavaScript syntax
var myWindow = _player.window[3];
```

関連事項

プレーヤ , window()

windowBehind

使用法

```
-- Lingo syntax
windowObjRef.windowBehind

// JavaScript syntax
windowObjRef.windowBehind;
```

解説

すべての他のウィンドウの後ろにあるウィンドウへの参照を返すウィンドウプロパティです。読み取り専用。

例文

これらのステートメントは、すべてのウィンドウの後ろにあるウィンドウに変数 backWindow 設定し、次にそのウィンドウを手前に移動します。

```
-- Lingo syntax
backWindow = _player.windowList[5].windowBehind
backWindow.moveToFront()

// JavaScript syntax
var backWindow = _player.windowList[5].windowBehind;
backWindow.moveToFront();
```

関連事項

moveToBack(), moveToFront(), ウィンドウ , windowInFront, windowList

windowInFront

使用法

```
-- Lingo syntax
windowObjRef.windowInFront

// JavaScript syntax
windowObjRef.windowInFront;
```

解説

すべての他のウィンドウの前にあるウィンドウへの参照を返すウィンドウプロパティです。読み取り専用。

例文

これらのステートメントは、すべてのウィンドウの前にあるウィンドウに変数 frontWindow 設定し、次にそのウィンドウを後ろに移動します。

```
-- Lingo syntax
frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack()

// JavaScript syntax
var frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack();
```

関連事項

moveToBack(), moveToFront(), ウィンドウ , windowBehind, windowList

windowList

使用法

```
-- Lingo syntax
_player.windowList

// JavaScript syntax
_player.windowList;
```

解説

プレーヤプロパティ。すべての既知のムービーウィンドウへの参照を含むリストを示します。読み取り専用。

ステージもウィンドウと見なされます。

例文

次のステートメントは、すべてのムービーウィンドウのリストをメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(_player.windowList)

// JavaScript syntax
trace(_player.windowList);
```

関連事項

プレーヤ

wordWrap

使用法

```
-- Lingo syntax
memberObjRef.wordWrap

// JavaScript syntax
memberObjRef.wordWrap;
```

解説

キャストメンバプロパティ。行を折り返すか（TRUE）、折り返さないか（FALSE）を決定します。

例文

次のステートメントは、フィールドキャストメンバ " 六条 " での行の折り返しをオフにします。

```
--Lingo syntax
member("Rokujo").wordWrap = FALSE

// JavaScript syntax
member("Rokujo").wordWrap = false;
```

worldPosition

使用法

```
member(whichCastmember).model(whichModel).worldPosition
member(whichCastmember).light(whichLight).worldPosition
member(whichCastmember).camera(whichCamera).worldPosition
member(whichCastmember).group(whichGroup).worldPosition
```

解説

3D プロパティ。このプロパティを使用すると、指定したノードの位置をワールド座標で取得できます。ただし、設定はできません。ノードは、モデル、グループ、カメラ、またはライトのいずれかです。このプロパティを設定するのは、getWorldTransform().position コマンドを使用するのと同じです。ノードの位置は、ベクトルオブジェクトで表されます。

例文

次のステートメントは、" 火星 " というモデルの位置が、ワールド座標で vector (-1333.2097, 0.0000, -211.0973) であることを示します。

```
--Lingo syntax
put member("scene").model("Mars").worldPosition
-- vector(-1333.2097, 0.0000, -211.0973)

// JavaScript syntax
trace(member("scene").getProp("model",1).worldPosition)
// vector(-1333.2097, 0.0000, -211.0973)
```

関連事項

getWorldTransform(), position （トランスフォーム）

worldTransform

使用法

```
member(whichMember).model(whichModel).bonesPlayer.bone[index].worldTransform
```

解説

3D ボーンプレーヤプロパティ。特定のボーンのワールドを基準としたトランスフォームを取得できます。ボーンの親を基準としたトランスフォームを返す transform プロパティを使用するのとは対照的です。worldTransform プロパティは、ボーンプレーヤが変更されたモデルでのみ使用できます。

例文

次のステートメントは、ボーンのワールドを基準としたトランスフォームを変数 finalTransform に格納します。

```
--Lingo syntax
finalTransform = member("3D").model("biped").bonesPlayer.bone[3].worldTransform

// JavaScript syntax
finalTransform = member("3D").getPropRef("model",1).getProp("bonesPlayer").bone[3].worldTransform;
```

関連事項

bone, getWorldTransform(), transform （プロパティ）

wrapTransform

使用法

```
member( whichCastmember ).shader( ShaderName ).wrapTransform
member( whichCastmember ).shader[ ShaderIndex ].wrapTransform
member( whichCastmember ).model[modelName].shader.wrapTransform
member( whichCastmember ).model.shaderlist[ shaderListIndex ].wrapTransform
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、シェーダの texture にマップされるテクスチャ座標を変更するトランスフォームにアクセスできます。テクスチャがモデルサーフェスに投影される度合いを変更するには、このトランスフォームを回転します。テクスチャに影響はなく、トランスフォームはシェーダによるテクスチャの適用方向のみを変更します。

注意：注意：このコマンドは、シェーダの textureModeList が #planar、#spherical、または #cylindrical に設定されている場合にのみ有効です。

例文

次のステートメントは、"shad2" というシェーダの transformMode を #wrapCylindrical に設定します。次に、その円柱投影を x 軸を中心に 90 度回転して、円柱マッピングが **z** 軸ではなく、**y** 軸の回りに適用されるようにします。

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureMode= #wrapCylindrical
s.wrapTransform.rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureMode = symbol("wrapCylindrical");
s.wrapTransform.rotate(90.0,0.0,0.0);
```

wrapTransformList

使用法

```
member( whichCastmember ).shader( ShaderName ).wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).shader[ shaderListIndex ].wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shader.wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shaderList[ shaderListIndex ]. wrapTransformList[
textureLayerIndex ]
```

解説

3D 標準シェーダプロパティ。このプロパティを使用すると、指定されたテクスチャレイヤーにマップされるテクスチャ座標を変更するトランスフォームにアクセスできます。テクスチャがモデルサーフェスに投影される度合いを変更するには、このトランスフォームを回転します。テクスチャ自体に影響はなく、トランスフォームはシェーダによるテクスチャの適用方向のみを変更します。

注意：wrapTransformList[textureLayerIndex] は、textureModeList[textureLayerIndex] が #planar、#spherical、または #cylindrical に設定されている場合にのみ有効です。

例文

次の例では、まず 2 行目で、"shad2" というシェーダの 3 番目のテクスチャレイヤーの transformMode を #wrapCylindrical に設定します。次に、3 行目で、その円柱投影を x 軸を中心に 90° 回転して、円柱マッピングが **z** 軸ではなく、**y** 軸を中心に適用されるようにします。

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureModeList[3] = #wrapCylindrical
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureModeList[3] = symbol("wrapCylindrical");
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0);
```

関連事項

newShader, textureModeList

x （ベクトル）

使用法

```
member(whichCastmember).vector.x
member(whichCastmember).vector[1]
```

解説

3D プロパティ。このプロパティを使用すると、ベクトルの **x** コンポーネントを取得または設定できます。

例文

次のステートメントは、ベクトルの **x** コンポーネントを示します。

```
vec = vector(20, 30, 40)
put vec.x
-- 20.0000
```

xAxis

x5796 | Lingo_3d_x_axis_help

使用法

```
member(whichCastmember).transform.xAxis
```

解説

3D トランスフォームプロパティ。このプロパティを使用すると、トランスフォームスペースにおける、トランスフォームの標準形の **x** 軸を表すベクトルを取得できます。ただし、設定はできません。

例文

次の例の 1 行目は、モデル "ModCylinder" のトランスフォームをアイデンティティトランスフォームに設定します。2 行目と 3 行目では、"ModCylinder" の **x** 軸が vector (1.0000, 0.0000, 0.0000) であることを示します。つまり、"ModCylinder" の **x** 軸はワールドの **x** 軸と揃っていることになります。4 行目では、"ModCylinder" を y 軸を中心に 90° 回転します。これにより "ModCylinder" の軸も回転します。5 行目と 6 行目は、"ModCylinder" の **x** 軸が vector (0.0000, 0.0000, -1.0000) になったことを示します。つまり、ModCylinder の **x** 軸はワールドの負の **z** 軸に揃っていることになります。

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

xtra

使用法

```
-- Lingo syntax
_player.xtra[xtraNameOrNum]

// JavaScript syntax
_player.xtra[xtraNameOrNum];
```

解説

Director プレーヤが使用可能な Xtra 拡張機能へのインデックスまたは名前によるアクセスを提供するプレーヤプロパティです。読み取り専用。

引数 **xtraNameOrNum** は、アクセスする Xtra 拡張機能の名前を指定するストリング、またはアクセスする Xtra 拡張機能のインデックスの位置を指定する整数です。

このプロパティの機能は、xtra() メソッドのトップレベルと同じです。

例文

次のステートメントは変数 myXtra を Speech Xtra 拡張機能に設定します。

```
-- Lingo syntax
myXtra = _player.xtra["SpeechXtra"]

// JavaScript syntax
var myXtra = _player.xtra["SpeechXtra"];
```

関連事項

プレーヤ , xtra()

xtraList （ムービー）

使用法

```
-- Lingo syntax
_movie.xtraList

// JavaScript syntax
_movie.xtraList;
```

解説

[ムービー / エクストラ] ダイアログボックス内の、ムービーに追加されているすべての Xtra 拡張のリニアプロパティリストを表示するムービープロパティです。読み取り専用。

xtraList には 2 つのプロパティが表示されます。

- #filename - その Xtra の現行プラットフォームにおけるファイル名を指定します。Xtra 拡張機能が 1 つのプラットフォームにのみ存在する場合など、リストに #filename エントリが含まれないこともあります。
- #packageurl - #packagefiles により指定されるダウンロードパッケージの場所を URL として指定します。
- #packagefiles - Xtra 拡張機能がダウンロード用にマークされている場合のみ設定されます。このプロパティの値は、現行プラットフォーム用のダウンロードパッケージにある各ファイルのプロパティリストを含む別のリストです。このサブプロパティリストのプロパティは #name および #version で、これらは xtraList (Player) と同じ情報を含みます。

例文

次のステートメントでは、メッセージウィンドウに xtraList を表示します。

```
-- Lingo syntax
put(_movie.xtraList)

// JavaScript syntax
put(_movie.xtraList);
```

関連事項

ムービー , xtraList （プレーヤ）

xtraList （プレーヤ）

使用法

```
-- Lingo syntax
_player.xtraList

// JavaScript syntax
_player.xtraList;
```

解説

プレーヤプロパティ。使用できるすべての Xtra 拡張機能とそれらのファイルバージョンのリニアプロパティリストを示します。読み取り専用。

このプロパティは、ムービーの機能が Xtra の特定のバージョンによって異なる場合に使用できます。Xtra 拡張機能には、次の 2 つのプロパティが含まれます。

xtraList には 2 つのプロパティが表示されます。

- #filename - その Xtra の現行プラットフォームにおけるファイル名を指定します。Xtra 拡張機能が 1 つのプラットフォームにのみ存在する場合など、リストに #filename エントリが含まれないこともあります。
- #version - デスクトップ上でファイルが選択されたとき、プロパティダイアログボックス（Windows）または情報ウィンドウ（Mac）に表示されるものと同じバージョン番号を示します。結果として、Xtra 拡張機能にはバージョン番号を付けられません。

例文

次のステートメントは、Director プレーヤで使用できるすべての Xtra 拡張機能をメッセージウィンドウに表示します。

```
-- Lingo syntax
trace(_player.xtraList)

// JavaScript syntax
trace(_player.xtraList);
```

関連事項

mediaXtraList, プレーヤ , scriptingXtraList, toolXtraList, transitionXtraList

y （ベクトル）

使用法

```
member(whichCastmember).vector.y
member(whichCastmember).vector[2]
```

解説

3D プロパティ。このプロパティを使用すると、ベクトルの **y** コンポーネントを取得または設定できます。

例文

次のステートメントは、ベクトルの y コンポーネントを示します。

```
vec = vector(20, 30, 40)
put vec.y
-- 30.0000
```

yAxis

使用法

```
member(whichCastmember).transform.yAxis
```

解説

3D トランスフォームプロパティ。このプロパティを使用すると、トランスフォームスペースにおける、トランスフォームの標準形の **y** 軸を表すベクトルを取得できます。ただし、設定はできません。

例文

次の例の 1 行目は、モデル "ModCylinder" のトランスフォームをアイデンティティトランスフォームに設定します。2 行目と 3 行目では、"ModCylinder" の Y 軸が vector (0.0000, 1.0000, 0.0000) であることを示します。つまり、ModCylinder の **y** 軸はワールドの **y** 軸に揃っていることになります。4 行目では、"ModCylinder" を **x** 軸を中心に 90° 回転します。これにより "ModCylinder" の軸も回転します。5 行目と 6 行目は、"ModCylinder" の **y** 軸が vector (0.0000, 0.0000, 1.0000) になったことを示します。つまり、ModCylinder の **y** 軸はワールドの正の **z** 軸に揃っていることになります。

```
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 1.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(90, 0, 0)
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 0.0000, 1.0000 )
```

yon

使用法

```
member(whichCastmember).camera(whichCamera).yon
```

解説

3D プロパティ。このプロパティを使用すると、表示フラスタムをクリッピングするカメラの Z 軸上の場所を定義する、カメラからの距離を取得または設定できます。yon よりも離れた距離にあるオブジェクトは描画されません。

このプロパティのデフォルト値は 3.40282346638529e38 です。

例文

次のステートメントは、カメラ 1 の yon プロパティを 50000 に設定します。

```
-- Lingo syntax
member("3d world").camera[1].yon = 50000

// JavaScript syntax
member("3d world").getPropRef("camera",1).yon = 50000
```

関連事項

hither

z （ベクトル）

使用法

```
member(whichCastmember).vector.z
member(whichCastmember).vector[3]
```

解説

3D プロパティ。このプロパティを使用すると、ベクトルの **z** コンポーネントを取得または設定できます。

例文

次のステートメントは、ベクトルの z コンポーネントを示します。

```
vec = vector(20, 30, 40)
put vec.z
-- 40.0000
```

zAxis

使用法

```
member(whichCastmember).transform.zAxis
```

解説

3D トランスフォームプロパティ。このプロパティを使用すると、トランスフォームスペースにおける、トランスフォームの標準形の **z** 軸を表すベクトルを取得できます。ただし、設定はできません。

例文

次の例の 1 行目は、モデル "ModCylinder" のトランスフォームをアイデンティティトランスフォームに設定します。2 行目と 3 行目では、"ModCylinder" の **z** 軸が vector (0.0000, 1.0000, 0.0000) であることを示します。つまり、"ModCylinder" の **z** 軸はワールドの **z** 軸に揃っていることになります。4 行目では、"ModCylinder" を **y** 軸を中心に 90° 回転します。これにより "ModCylinder" の軸も回転します。5 行目と 6 行目は、"ModCylinder" の **z** 軸が vector (0.0000, 0.0000, 1.0000) になったことを示します。つまり、"ModCylinder" の **z** 軸はワールドの **x** 軸に揃っていることになります。

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.zAxis)
//vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.zAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

第 15 章：物理エンジン

物理 Xtra は、オブジェクトが相互に作用する 3D ワールドを作成する開発者のための高パフォーマンスなツールです。この Xtra は、質量、速度、回転などのオブジェクトのプロパティを計算に入れて、衝突の結果を決定する計算を実行します。力を適用し、制限を与えてオブジェクトどうしを接続できます。使用できる制限は、6 自由度ジョイント、リニアジョイント、角ジョイント、およびスプリングジョイントです。

また、地形およびレイキャストもサポートされています。地形とは、2 次元が無限で第 3 次元に沿って高度を定義する、凹凸のあるプレーンのようなものです。レイキャスティングとは、レイによる衝突検出のメカニズムです。レイキャスティングは、すべてのタイプの剛体および地形に対して実行できます。

この Xtra を使用すると、開発者はゲームの進行とユーザ操作に集中でき、Lingo スクリプトを使用したリアルタイム物理エンジンの作成について考慮する必要はありません。

物理（力学）Xtra は、完全に統合された Adobe® Director® 用の剛体物理シミュレーションエンジンです。Windows および Mac プラットフォームでサポートされています。

Director 用の物理エンジンには次の機能があります。詳細については、この章の説明を参照してください。

物理ワールドのプロパティ

angularDamping

使用法

```
world.angularDamping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

デフォルト値： 0.000

解説

角の力が減少する減衰量を表します。

例文

```
--Lingo Syntax
member("PhysicsWorld").angularDamping = 10.0
//Javascript syntax
member("PhysicsWorld").angularDamping = 10.0;
```

関連事項

linearDamping

contactTolerance

使用法

```
world.contactTolerance
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

衝突が検出されるための剛体間の浸透深度。

注意：ボディ間の衝突をシミュレートするときに思ったとおりの結果を得るには、衝突耐性値を調整します。

例文

このステートメントは、衝突が検出されるための剛体間の浸透深度を設定します。

```
--Lingo Syntax
member("PhysicsWorld").contactTolerance = 0.01
//JavaScript Syntax
member("PhysicsWorld").contactTolerance = 0.01;
```

注意：

- 理想的には、衝突耐性は剛体寸法の 2% とします。
- 各剛体の衝突耐性を指定しない場合、ワールドに対して設定されている値が使用されます。
- 衝突耐性の値を大きくすると、数値エラーが発生し、不安定になります。

friction

使用法

```
world.friction
```

アクセス： 取得および設定

タイプ： 浮動小数点数

値 0 - 1

解説

ワールド内のすべての剛体に適用される摩擦係数のデフォルト値を表します。friction プロパティには 0 ～ 1 の値を指定します。

例文

```
--Lingo Syntax
member("PhysicsWorld").friction = 0.5
//Javascript syntax
member("PhysicsWorld").friction = 0.5;
```

関連事項

restitution

gravity

使用法

```
world.gravity
```

アクセス： 取得および設定

タイプ： ベクトル

解説

物理ワールド内のすべての剛体が受ける重力による加速を表します。

例文

```
--Lingo Syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0)
//Javascript syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0);
```

IsInitialized

使用法

```
world.isInitialized
```

アクセス： 取得のみ

タイプ： ブール値

解説

ワールドの、現在の初期化の状態。ワールドに対する init() メソッドが正常に終了した場合、このプロパティは 1 に設定されます。

例文

```
--Lingo Syntax
If not member("PhysicsWorld").isInitalized then
-- Initialize the physics world here
end if
//JavaScript Syntax
If (member("PhysicsWorld").isInitialized == 0) {
    // Initialize World here
}
```

関連事項

init()

linearDamping

使用法

```
world.linearDamping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

デフォルト値： 0.000

解説
時間ステップの終わりのリニア減速を表します。

例文
```
--Lingo Syntax
member("PhysicsWorld").linearDamping = 10.0
//Javascript syntax
member("PhysicsWorld").linearDamping = 10.0;
```

関連事項
angularDamping

restitution

使用法
```
world.restitution
```

アクセス： 取得および設定

タイプ： 浮動小数点数

値 0 - 1

解説
ワールド内のすべての剛体に適用される反発係数のデフォルト値を表します。restitution プロパティには 0 ～ 1 の値を指定します。

例文
```
--Lingo Syntax
member("PhysicsWorld").restitution = 0.5
//Javascript syntax
member("PhysicsWorld").restitution = 0.5;
```

関連事項
friction

scalingFactor

使用法
```
world.scalingFactor
```

アクセス： 取得のみ

タイプ： ベクトル

解説
3D ワールドと物理ワールドの単位が異なる場合に、変換に使用するスケールを表します。scalingFactor は、長さ、質量、および時間のスケールを受け取るベクトルです。

注意：Director 11 では、このプロパティの値は設定できません。この機能は将来のリリースで利用可能になります。

例文

```
--Lingo Syntax
put member("PhysicsWorld").scalingFactor
//Javascript syntax
put(member("PhysicsWorld').scalingFactor)
```

関連事項

init()

sleepMode

使用法

```
world.sleepMode
```

アクセス： 取得および設定

値： #energy、#linearvelocity

デフォルト値： #energy

解説

剛体のスリープモードを取得または設定します。

#energy スリープ状態を確認するのに、剛体の運動エネルギーが使用されます。

#linearvelocity スリープ状態を確認するのに、剛体のリニア速度が使用されます。

例文

```
--Lingo Syntax
member("PhysicsWorld").sleepMode = #energy
//Javascript syntax
member("PhysicsWorld").sleepMode = symbol("energy");
```

関連事項

sleepThreshold

sleepThreshold

使用法

```
world.sleepThreshold
```

アクセス： 取得および設定

タイプ： 浮動小数点数

デフォルト値： 0.000

解説

その値を下回ると剛体が非アクティブになるしきい値を表します。値は、sleepMode プロパティに設定されている値に応じて、剛体の運動エネルギーまたはリニア速度のいずれかです。動的オブジェクトのみに適用されます。

例文

```
--Lingo Syntax
member("PhysicsWorld").sleepThreshold = 10.0
//Javascript syntax
member("PhysicsWorld").sleepThreshold = 10.0;
```

関連事項

sleepMode

subSteps

使用法

```
world.subSteps
```

アクセス： 取得および設定

タイプ： 整数

解説

各シミュレーション呼び出しのサブステップ数を表します。

例文

```
--Lingo Syntax
member("PhysicsWorld").subSteps = 5
//Javascript syntax
member("PhysicsWorld").subSteps = 5;
```

関連事項

init(),timeStep,timeStepMode

timeStep

使用法

```
world.timeStep
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

等分時間ステップモードでのみ使用され、物理ワールドのステップ時間を指定します。

例文

```
--Lingo Syntax
member("PhysicsWorld"). timeStep = 0.0167
//Javascript syntax
member("PhysicsWorld"). timeStep = 0.0167;
```

関連事項

init(),subSteps,timeStepMode

timeStepMode

使用法

```
world.timeStepMode
```

アクセス： 取得および設定

解説

ワールドへの init() 呼び出しに指定された現在の時間ステップモードを返します。#automatic または #equal のいずれかです。

#equal - init 呼び出しの timeStep パラメータに指定した時間に従って物理ワールドの進度を調整します。

#automatic - 実際の経過時間に従って物理ワールドの進度を調整します。

例文

このステートメントは、物理ワールドの現在の時間ステップモードを「equal」に設定します。

```
--Lingo Syntax
member("PhysicsWorld").timeStepMode = #equal
//JavaScript Syntax
member("PhysicsWorld").timeStepMode = symbol("equal");
```

関連事項

init(),timeStep,subSteps

物理ワールドのメソッド

destroy()

使用法

```
<void>world.destroy()
```

解説

ワールドのシミュレーションを停止し、ワールド内のすべてのリソースを解放します。シミュレーションを再開するには、ワールドに対して Init() を呼び出す必要があります。

パラメータ

なし

例：

```
--Lingo Syntax
member("PhysicsWorld").destroy()
//Javascript syntax
member("PhysicsWorld").destroy();
```

注意： destroy() を呼び出さずに Director Authoring でムービーを再生すると、物理オブジェクトを作成するときにエラーが発生します。

関連事項
init()

getSimulationTime()

使用法
```
<float>world.getSimulationTime()
```

解説
物理ワールドのシミュレーションの経過時間を返します。

パラメータ
なし

例文
```
--Lingo Syntax
nFloatTime = member("PhysicsWorld").getSimulationTime()
//Javascript syntax
var nFloatTime = member("PhysicsWorld").getSimulationTime();
```

関連事項
simulate()

init()

使用法
```
world.init(string 3dmembername, vector scalingFactor, symbol timeStepMode, float timeStep, int
substepCount).
world.init(3dmemberref, vector scalingFactor, symbol timeStepMode,float timeStep, int substepCount).
```

解説
この関数は、指定された 3D キャストメンバを使用して物理ワールドを初期化します。各物理キャストメンバは、「3dmembername」によって表される一意の 3D キャストメンバに関連付けられている必要があります。

- init() 関数は、別の init() 関数を呼び出す前に world.destroy() で終了する必要があります。
- 物理シミュレーションの init() メソッドと destroy() メソッドの間に resetworld() メソッドを使用しないでください。
- キーフレームアニメーションまたはボーンプレーヤアニメーションを持つモデルには、物理を適用しないでください。

注意：物理オブジェクトのメソッドまたはプロパティにアクセスする前にワールドが初期化されていない場合、-2 エラーが発生します。

パラメータ

パラメータ	解説
3dmembername	必須。一意の 3D キャストメンバを指定するストリングまたはメンバ参照。
ScalingFactor	必須。物理シミュレーションでの 3d ワールドのスケール係数を指定するベクトル。

パラメータ	解説
Timestep	ミリ秒単位の値。
Timestepmode	必須。 #equal - init 呼び出しの timeStep パラメータに指定した時間に従って物理ワールドの進度を調整します。 #automatic - 実際の経過時間に従って物理ワールドの進度を調整します。
subStepcount	必須。正確を期すために計算される整数値。

例：

```
--Lingo Syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#automatic,0,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#automatic,0,5)
//Javascript syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("equal"),0.167,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("equal"),0.33, 5);
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("automatic"),0,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("automatic"),0, 5);
```

関連事項

destroy(),subSteps,timeStep,timeStepMode

PauseSimulation()

使用法

```
world.PauseSimulation()
```

解説

この物理ワールドでの物理シミュレーションを一時停止します。ResumeSimulation() 呼び出しが行われるまで、simulate() 呼び出しは無効です。

パラメータ

なし

例文

```
--Lingo Syntax
member("PhysicsWorld").PauseSimulation()
//Javascript syntax
member("PhysicsWorld").PauseSimulation();
```

関連事項

ResumeSimulation ()

ResumeSimulation ()

使用法

```
<void>world.ResumeSimulation ()
```

解説

world.PauseSimulation() 呼び出しによって一時停止していた物理ワールドのシミュレーションを再開します。

パラメータ

なし

例文

```
--Lingo Syntax
member("PhysicsWorld").ResumeSimulation ()
//Javascript syntax
member("PhysicsWorld").ResumeSimulation ();
```

関連事項

PauseSimulation()

simulate()

使用法

```
world.simulate()
```

解説

呼び出しの物理ワールドを一定の時間だけ進行させます。このステップ時間は、init 呼び出し時に設定される timeStepMode によって異なります。

timeStepMode には次の 2 つの値を指定できます。

#equal simulate() 呼び出しは、init 呼び出しの timeStep パラメータに指定した時間に従って物理ワールドの進度を調整します。

#automatic simulate() 呼び出しは、実際の経過時間に従って物理ワールドの進度を調整します。

- 物理ワールドに加えたすべての変更は、simulate() メソッドを呼び出したときにのみ反映されます。例えば、applyimpulse() メソッドを使用して剛体にインパルスを適用すると、simulate() メソッドを呼び出すまで、linearVelocity の値は変更されません。
- 正しいシミュレーションのために、すべての exitFrame() で simulate() メソッドを呼び出すことをお勧めします。

注意：シミュレート呼び出しが失敗した場合、-3 エラーが表示されます。

パラメータ

なし

例文

```
--Lingo Syntax
member("PhysicsWorld").simulate()
//Javascript syntax
member("PhysicsWorld").simulate();
```

関連事項

init(),subSteps,timeStep,timeStepMode

剛体の管理メソッド

注意：剛体と地形には別々の名前を使用してください。剛体と地形に同じ名前を使用すると、無効なパラメータを示すエラー -4 が返されます。

addProxyTemplate()

使用法

```
physicsWorld.addProxyTemplate(3dmember, proxyTemplate, string proxyname)
```

解説

createproxyTemplate() メソッドを使用して作成されるプロキシテンプレートを指定した 3d メンバに追加します。

パラメータ

パラメータ	解説
3dmember	必須。3d キャストメンバへの参照。
proxyTemplate	必須。proxytemplate は createProxyTemplate() を使用して作成されます。
proxyName	必須。プロキシを保存する名前。

戻り値は、成功の場合は 1、失敗の場合は 0 です。

このメソッドは、createProxyTemplate() メソッドを使用して作成されるプロキシテンプレートを指定した 3d メンバに追加します。

#physicsmesh タイプの新しいモデルリソースが 3d メンバに追加されます。

注意：addProxyTemplate() メソッドは、指定した 3D ワールドメンバで newModelResource #physicsmesh としてプロキシテンプレートのみを追加します。新しく追加したプロキシを保存するには、saveWorld() または saveW3d() メソッドを呼び出す必要があります。

例文

```
--Lingo syntax
member("PhysicsWorld").addProxyTemplate(member("3dWorld") , ProxyTemplate, "saved_proxyTemplateName")
--creates a modelresource of type #physicsmesh in the 3d world specified.
--modelresource("saved_proxyTemplateName").meshtype gives the type of physicsmesh . i.e #concave and
#convex.
--Use saveWorld() or saveW3d() methods to save the proxy added to the 3d world.
-- member("3dWorld").saveW3d(the moviepath & "proxyWorld.w3d")
// JavaScript syntax
member("PhysicsWorld").addProxyTemplate(member("3dWorld") , ProxyTemplate,"saved_proxyTemplateName");
```

注意：メッセージウィンドウを使用してプロキシテンプレートを保存することをお勧めします。

関連事項

createRigidBodyFromProxy(),createProxyTemplate(),loadProxyTemplate()

createRigidBody()

使用法

```
<RigidBody> world.createRigidBody(string rigidbodyname, string 3Dmodelname,symbolBodyProxy,symbol
bodyType,symbol flipNormals)
```

解説

このメソッドは、指定したプロキシシェイプの剛体を作成します。

剛体を作成する前に #meshdeform モディファイアをモデルに追加する必要があります。meshdeform モディファイアが 3D モデルに追加されていない場合、-21 エラーが表示されます。

注意：動的な凹型の剛体の作成方法については、「CreateRigidBodyFromProxy()」を参照してください。

パラメータ

パラメータ	解説
3dModelName	必須。剛体を作成する 3d モデルの名前を指定するストリング。
rigidBodyName	必須。剛体の名前を指定するストリング。
symbolBodyProxy	必須。次の値を指定するストリング。 #box #Sphere #convexShape #concaveShape
bodyType	必須。 #static - 静的または固定されたボディを作成する必要がある場合に指定します。 **注意：**位置を変更して静的なボディを移動すると、剛体の位置は変化しますが、3D モデルの位置は変わりません。 #dynamic - 移動可能なボディを作成する必要がある場合に指定します。
flipNormals	オプション。 このパラメータはフェイス法線を反転します。 #concaveshape プロキシを使用するボディの場合は、フェイス法線の正方向に衝突が検出されます。浸透は反対側から検出されます。この動作を逆にするには、FlipNormals を使用します。 このパラメータは、#convexshape プロキシと #concaveshape プロキシのみに使用できます。他のプロキシでは、エラーが発生します。

剛体への参照が返されます。作成が失敗した場合は、void が返されます。

注意：#concaveshape プロキシを使用する剛体は、#static である必要があります。動的な凹型のボディを作成しようとすると、エラー -28 が発生します。動的な凹型の剛体の作成方法については、「createRigidBodyFromProxy()」を参照してください。

例文

```
--Lingo syntax
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#box,#dynamic)
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#convexshape,#dynamic)
objRigidBody
=member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#concaveshape,#static,#flipNormals) --The
last parameter,#flipNormals,is optional.
//JavaScript Syntax
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("sphere"),symbol("static"));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("box"),symbol("dynamic"));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("convexshape"),symbol("dynamic'));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("concaveshape"),symbol("static"),symb
ol("flipNormals"));//The last parameter, symbol("flipNormals"),is optional.
```

関連事項

deleteRigidBody()

createRigidBodyFromProxy()

使用法

```
world.createRigidBodyFromProxy(string rigidbodyname, string 3Dmodelname,symbol
bodyType,ProxyTemplate,symbol flipNormals)
```

解説

このメソッドは、指定したプロキシシェイプを使用して剛体を作成します。

剛体を作成する前には、モデルに #meshdeform モディファイアを追加します。次のいずれかの手順により、プロキシから剛体を作成できます。

- プロキシテンプレートを作成し、それを直接使用して実行時に剛体を作成する。
- オーサリング時にモデルリソースのプロキシテンプレートを作成してそれを 3D ワールド（w3d）に保存し、実行時にこのテンプレートを使用して実際の剛体を作成する。この方法は、凹型および凸型の剛体を作成する場合にお勧めします。この方法では、その他のプロキシの種類はサポートされていません。

  剛体の作成時には、拡大 / 縮小およびモデルのトランスフォームが自動的に適用されます。同じモデルリソースからモデリングされる剛体を作成する場合は、プロキシテンプレートを再利用することができます。

  オーサリング中にプロキシを作成し、後で使用するために 3D メンバとして保存すると、効率が向上します。

  プロキシテンプレートが無効の場合、または動的な凹型剛体の作成時に #concave プロキシテンプレートが渡された場合、エラー - 30 が発生します。

パラメータ

パラメータ	解説
3dModelName	必須。剛体を作成する 3d モデルの名前を指定するストリング。
rigidBodyName	必須。剛体の名前を指定するストリング。
bodyType	必須。 #static - 静的または固定されたボディを作成する必要がある場合に指定します。 **注意：**位置を変更して静的なボディを移動すると、剛体の位置は変化しますが、3D モデルの位置は変わりません。 #dynamic - 移動可能なボディを作成する必要がある場合に指定します。
ProxyTemplate	必須。メソッド createProxyTemplate() から取得されたプロキシテンプレートオブジェクト。 **注意：**プロキシの作成と保存については、createProxyTemplate()、addProxyTemplate() および loadProxyTemplate() の各メソッドを参照してください。
flipNormals	オプション。 このパラメータはフェイス法線を反転します。 #concaveshape プロキシを使用するボディの場合は、フェイス法線の正方向に衝突が検出されます。浸透は反対側から検出されます。この動作を逆にするには、FlipNormals を使用します。 このパラメータは、#convexshape プロキシと #concaveshape プロキシのみに使用できます。他のプロキシでは、エラーが発生します。

剛体への参照が返されます。作成が失敗した場合は、void が返されます。

注意：動的な凹型の剛体に #concave 型のプロキシテンプレートを指定するとエラーになります。createProxyTemplate() および addProxyTemplate() はメッセージウィンドウと共に使用する必要があります。これらを、ランタイムコードの一部にはしないでください。この方法により、動的な凹型の剛体の凸型分解のオーバーヘッドを回避することができます。また、独立した 3D ワールドでプロキシを保存すると、モデルリソースの混乱を回避することができます。

例文

```
--Lingo syntax
--During Author time save the proxy template to a 3D world using steps 1 to 3
--Step1: Create a proxy template from a model resource
ProxyTemplate = member("PhysicsWorld").createProxyTemplate(modelResource Reference
,#convexDecomposed,[#concavity:2, #depth:8,#mergeVolume:2])
--Step2: Add the proxy template to the 3D world
member("PhysicsWorld").addProxyTemplate(member("3DproxyWorld"), ProxyTemplate, "proxyName")
--Step3: Save the 3D world
member("3DproxyWorld").saveWorld()
--At runtime load the saved proxy template and create the rigid body.
--Load the proxy template from the saved 3D world
Saved_proxy = member("PhysicsWorld").loadProxyTemplate("proxyName",member("3DproxyWorld"))
--Create the rigid body from proxy template
objRB = member("PhysicsWorld").createRigidBodyFromProxy("RigidBodyA","ModelA", #dynamic, Saved_proxy)
//Javascript syntax
Var objRB = member("PhysicsWorld").createRigidBodyFromProxy("RigidBodyA"," ModelA", symbol("dynamic"),
Saved_proxy) ;
```

関連事項

createProxyTemplate(),loadProxyTemplate(),addProxyTemplate()

createProxyTemplate()

使用法

```
physicsWorld.createProxyTemplate( ModelResourceRef, #proxyTemplateSymbol, decomposition parameter list)
```

解説

このメソッドは、指定したプロキシシェイプを使用して剛体を作成します。

パラメータ

パラメータ	解説
ModelResourceRef	必須。動的な凹型のプロキシを作成する対象であるモデルの参照。
proxyTemplateSymbol	必須。次のいずれかを取るシンボル： #convexDecomposed - 凸型メッシュ分解方式を使用して、動的な凹型のプロキシテンプレートを作成します。 #concave - 静的な凹型のプロキシテンプレートを作成します。 #convex - 凸型のメッシュプロキシテンプレートを作成します。
動的な凹型の剛体用の ConvexDecomposition	オプション。プロパティリスト #depth - デフォルト値：3、範囲：0-10。 #concavity - デフォルト値：10. 範囲：0-100。 #mergeVolume - デフォルト値：10. 範囲：0-100。

指定したモデルリソースに対して作成されたプロキシへの参照。作成が失敗した場合は、void が返されます。

注意：使用されているモデルリソースが無効である場合は、-31 エラーが発生します。重複するモデルリソースがある場合は、エラー -29 が発生します。

返されるプロキシは、モデルから作成されたプロキシのプロパティが含まれるプロパティリストです。

numMeshes 凹シェイプを作成するために使用された凸型メッシュの数。

vertexList 各メッシュの頂点が含まれるリスト。

indexList 各メッシュのインデックスが含まれるリスト。

meshType #concave または #convex を返すシンボル。

proxyTemplate シンボル	proxyTemplate.meshType	proxyTemplate を使用して作成可能な剛体
#convexdecomposed	#convex	凹型 - 動的、凹型 - 静的
#concave	#concave	凹型 - 静的
#convex	#convex	凸型 - 静的、凸型 - 動的

#mesh 以外のモデルリソースタイプの Director 3D プリミティブは、このメソッドを使用したプロキシテンプレートの作成には使用できません。エラー -31 が発生します。

例文

```
--Lingo syntax
ProxyTemplate = member("PhysicsWorld").createProxyTemplate(modelResource Reference
,#convexDecomposed,[#concavity:4, #depth:6,#mergeVolume:4])
// JavaScript syntax
var ProxyTemplate = member("PhysicsWorld").
createProxyTemplate(member("3dWorld").getPropRef("modelResource",2)
,symbol("convexDecomposed"),propList(symbol("concavity"),2, symbol("depth"),8, symbol("mergeVolume"),2));
```

関連事項

createRigidBodyFromProxy(),loadProxyTemplate(),addProxyTemplate()

deleteRigidBody()

使用法

```
<void>world.deleteRigidBody(string rigidbodyname)
<void>world.deleteRigidBody(RigidBodyref r)
```

解説

物理ワールドから剛体を削除します。

注意：剛体を削除する前に、剛体に関連付けられている 3D モデルを削除しないでください。

パラメータ

次のパラメータのいずれか 1 つを指定する必要があります。

パラメータ	解説
Rigidbodyname	剛体名を指定するストリング。
Rigibodyref	createrigidbody() を使用して作成した剛体への参照。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody("RigidBodyA")
```

or

```
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody(objRB);
//Javascript Syntax
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody("RigidBodyA");
```

or

```
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody(objRB);
```

関連事項

createRigidBody()

getRigidBody()

使用法

```
<RigidBody>world.getRigidBody(string rigidBodyName)
```

解説

剛体名を指定した剛体を返します。指定した名前の剛体が存在しない場合は、void を返します。

パラメータ

パラメータ	解説
Rigidbodyname	必須。剛体名を指定するストリング。

剛体への参照を返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
```

getRigidBodies()

使用法

```
<list of RigidBodies>world.getRigidBodies()
```

解説

物理ワールドに現在存在している剛体のリストを返します。

パラメータ

なし

物理ワールドの剛体への参照を含むリストを返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBodies()
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBodies();
```

getSleepingBodies()

使用法

```
<list of RigidBodies> world.getSleepingBodies()
```

解説

スリープしている剛体のリストを返します。

パラメータ

なし

物理ワールドのスリープしている剛体への参照を含むリストを返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getSleepingBodies()
//Javascript Syntax
var objRB = member("PhysicsWorld").getSleepingBodies ();
```

loadProxyTemplate()

使用法

```
physicsWorld.loadProxyTemplate( string proxyname, 3dmember)
```

解説

このメソッドは、createproxyTemplate() および addProxyTemplate() メソッドを使用して作成されるプロキシテンプレートをロードします。

パラメータ

パラメータ	解説
proxyName	必須。プロキシを保存する名前。
3dmember	必須。3D キャストメンバへの参照。

#physicsmesh モデルリソースへの参照を返します。

例文

```
--Lingo syntax
Saved_proxyTemplate  =
member("PhysicsWorld").loadProxyTemplate("saved_proxyTemplateName",member("3dWorld"))
//JavaScript syntax
var Saved_proxyTemplate  =
member("PhysicsWorld").loadProxyTemplate("saved_proxyTemplateName",member("3dWorld"));
```

関連事項

createRigidBodyFromProxy(),createProxyTemplate(),addProxyTemplate()

剛体のプロパティ

angularDamping

使用法

```
r.angularDamping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

ボディの角減衰を取得および設定します。これはリニア減衰に対する角減衰です。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.angularDamping
objRB.angularDamping = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.angularDamping);
objRB.angularDamping = 0.5;
```

関連事項

linearDamping

angularMomentum

使用法

```
r.angularMomemtum
```

アクセス： 取得および設定

タイプ： ベクトル

解説

剛体の角運動量を取得および設定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.angularMomentum = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearVelocity);
objRB.angularMomentum =  vector(-100,0,0);
```

関連事項

linearMomemtum

angularVelocity

使用法

```
r.angularVelocity
```

アクセス： 取得および設定

タイプ： ベクトル

解説

剛体の角速度ベクトルを取得および設定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB. angularVelocity
objRB.angularVelocity = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. angularVelocity);
objRB.angularVelocity =  = vector(-100,0,0);
```

関連事項

linearVelocity

axisAffinity

使用法

```
r.axisAffinity
```

アクセス： 取得および設定

タイプ： ブール値

解説

動的な凹型の剛体のプロパティ。ボディーに力を適用した場合、軸に対する剛体のアフィニティを制御します。

剛体には常に、ワールドの軸に対するアフィニティがあります。剛体の axisAffinity プロパティを False に設定した場合、ワールドの軸に対するこのアフィニティが補正されます。

注意： axisAffinity プロパティのデフォルト値は True です。これは、この値を False に設定すると、多くの計算リソースを消費するためです。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.axisAffinity
objRB.axisAffinity = TRUE

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.axisAffinity);
objRB.axisAffinity = true;
```

centerOfMass

使用法

```
r.centerOfMass
```

アクセス： 取得および設定

タイプ： ベクトル

解説

剛体座標での剛体の質量の中心を表します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.centerOfMass
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.centerOfMass);
```

関連事項

mass

contactTolerance

使用法

```
r.contacttolerance
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

剛体が別のボディと衝突するときの相互浸透深度を設定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.contactTolerance = 0.01
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.contactTolerance = 0.01;
```

friction

使用法

```
r.friction
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

剛体の摩擦係数を取得および設定します。friction プロパティには 0 ～ 1 の値を指定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.friction
objRB.friction= 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.friction);
objRB.friction= 0.5;
```

関連事項
restitution

isPinned

使用法
r.ispinned

アクセス： 取得および設定

タイプ： ブール値

解説
ボディの固定状態を取得および設定します。true の場合、ボディは現在の向きでその位置に固定されます。外部の力によってボディの位置または向きは変更されません。

例文
```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isPinned
objRB.isPinned = TRUE
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.isPinned);
objRB.isPinned = true;
```

isSleeping

使用法
r.isSleeping

アクセス： 取得および設定

タイプ： ブール値

解説
剛体の状態をスリープ中として返すか、設定します。

注意：剛体の休止状態はそのシェイプによって異なります。例えば、ボックスプロキシを使用して作成した剛体は、球体に比べてスリープ時間が長くなります。

例文
```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isSleeping
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB. isSleeping)
```

linearDamping

使用法

```
r.linearDamping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

ボディのリニア減衰の現在値を取得および設定します。これは、ワールド内を直線的に移動するボディの減速量です。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearDamping
objRB.linearDamping = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearDamping);
objRB.linearDamping = 0.5;
```

関連事項

angularDamping

linearMomemtum

使用法

```
r.linearMomemtum
```

アクセス： 取得および設定

タイプ： ベクトル

解説

剛体の直線運動量を取得および設定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearMomentum
objRB.linearMomentum = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearMomentum);
objRB.linearMomentum =  vector(-100,0,0);
```

関連事項

angularMomentum

linearVelocity

使用法

```
r.linearVelocity
```

アクセス： 取得および設定

タイプ： ベクトル

解説

剛体のリニア速度ベクトルを取得および設定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.linearVelocity = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. linearVelocity);
objRB.linearVelocity = vector(-100,0,0);
```

関連事項

angularVelocity

mass

使用法

```
r.mass
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

剛体の質量を設定または取得します。質量のデフォルト値は 0 です。このプロパティには 0 より大きい値を設定します。値を指定しないとシミュレーションは不正確になります。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.mass
objRB.mass = objRB.mass + 20
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.mass);
objRB.mass = objRB.mass + 20;
```

関連事項

centerOfMass

model

使用法

```
r.model
```

アクセス： 取得のみ

解説

この剛体に関連付けられている 3D モデルを返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objModel = objRB.model
put objModel.name
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
var objModel = objRB.model
put(objModel.name)
```

name

使用法

```
r.name
```

アクセス： 取得のみ

タイプ： ストリング

解説

剛体の名前を返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.name
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.name);
```

orientation

使用法

```
r.orientation
```

アクセス： 取得および設定

タイプ： リスト

解説

物理ワールドでの剛体の向きを取得および設定します。orientation プロパティは、方向ベクトルおよび剛体の軸とワールドの軸の間の角度を指定したリストを表します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.orientation = [vector(10,0,0),45]
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.orientation = list(vector(10,0,0),45);
```

注意：静的な剛体の位置または向きを変更しても、モデルの位置は変わりません。モデルの位置または向きは明示的に変更する必要があります。

position

使用法

```
r.position
```

アクセス： 取得および設定

タイプ： ベクトル

解説

物理ワールドでの剛体の位置を取得および設定します。位置はワールド座標での (x,y,z) のベクトルを表します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.position = vector(0,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.position = vector(0,0,0);
```

注意：静的な剛体の位置または向きを変更しても、モデルの位置は変わりません。モデルの位置または向きは明示的に変更する必要があります。

プロパティ

使用法

```
r.properties
```

アクセス： 取得のみ

タイプ： プロパティリスト

解説

この剛体のプロパティを表すプロパティリストを返します。rb.properties は、rigidbody シェイプのタイプに基づき、プロパティが含まれるリストを返します。

rb.shape	rb.properties
#box	[#length, #width, #height, #center]

rb.shape	rb.properties
#sphere	[#radius, #center]
#convex	[#numvertices, #numfaces, #vertexlist, #face
#concave	[#numvertices, #numfaces, #vertexlist, #face]

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.properties
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.properties);
```

restitution

使用法

```
r.restitution
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

剛体の反発係数を取得および設定します。restitution プロパティには 0 ～ 1 の値を指定します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.restitution
objRB.restitution = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.restitution);
objRB.restitution = 0.5;
```

シェイプ

使用法

```
r.shape
```

アクセス： 取得のみ

値 #sphere、#box、#convexshape、または #concaveshape

解説

剛体のシェイプを返します。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
    put objRB.shape
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.shape);
```

sleepMode

使用法

```
r.sleepMode
```

アクセス： 取得および設定

解説

剛体のスリープモードを取得または設定します。#energy または #linearvelocity のいずれかです。デフォルト値は #energy です。

#energy スリープ状態を確認するのに、剛体の運動エネルギーが使用されます。

#linearvelocity スリープ状態を確認するのに、剛体のリニア速度が使用されます。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy
//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
```

関連事項

sleepThreshold

sleepThreshold

使用法

```
r.sleepThreshold
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

その値を下回ると剛体が非アクティブになるしきい値を表します。値は、sleepMode プロパティに設定されている値に応じて、剛体の運動エネルギーまたはリニア速度のいずれかです。動的オブジェクトのみに適用されます。デフォルト値は 0.000 です。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy
objRB.sleepThreshold = 10.0
//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
objRB.sleepThreshold = 10.0;
```

関連事項

sleepMode

type

使用法

```
r.type
```

アクセス： 取得のみ

解説

剛体の種類を返します。#static または #dynamic のいずれかです。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.type
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.type);
```

userData

使用法

```
r.userData
```

アクセス： 取得および設定

タイプ： オブジェクト

解説

このプロパティを使用すると、ユーザ定義のデータを取得および設定し、そのデータをこの剛体と関連付けることができます。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
--Assigning Integer Value
objRB.userData = 0.5
--Assigning String Value
objRB.userData = "My User data Value"
--Assigning Reference Value
objRB.userData = member("PhysicsWorld').getRigidBody("RigidBodyB")
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
--Assigning Integer Value
objRB.userData = 0.5;
--Assigning String Value
objRB.userData = "My User data Value";
--Assigning Reference Value
objRB.userData = member("PhysicsWorld').getRigidBody("RigidBodyB");
```

剛体のメソッド

applyForce()

使用法

```
<int>r.applyForce(vector vForce, vector vposition)
```

解説

与えられた力を、ベクトル位置によって指定された剛体座標内の位置に適用します。位置が指定されていない場合は、剛体の質量の中心に力が適用されます。

パラメータ

パラメータ	解説
vForce	必須。適用する力の大きさと向きを指定するベクトル値。
vPosition	オプション。力が適用される点を指定するベクトル。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyForce(vector(-100,0,0),vector(0,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyForce(vector(-100,0,0),vector(0,0,0));
```

applyLinearImpulse()

使用法

```
< int >r.applyLinearImpulse(vector vImpulse, vector vposition)
```

解説
与えられたリニアインパルスを、ベクトル位置によって指定された剛体座標内の位置に適用します。位置が指定されていない場合は、剛体の質量の中心にインパルスが適用されます。

パラメータ

パラメータ	解説
vImpulse	必須。適用する力の大きさと向きを指定するベクトル値。
vPosition	オプション。力が適用される点を指定するベクトル。

例文
```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0));
```

applyAngularImpulse()

x5848 | Lingo_r_linearimpulse

使用法
```
< int >r.applyAngularImpulse(vector vImpulse)
```

解説：
与えられた角インパルスを質量の中心に適用します。

パラメータ

パラメータ	解説
vImpulse	必須。適用される角インパルスを指定するベクトル。

例文
```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyAngularImpulse(vector(-100,100,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyAngularImpulse (vector(-100,0,0));
```

applyTorque()

使用法
```
< int >r.applyTorque(vector vTorque)
```

解説
指定したトルクを剛体の質量の中心に適用します。

パラメータ

パラメータ	解説
vTorque	必須。適用されるトルクを指定するベクトル。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyTorque(vector(-100,100,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyTorque(vector(-100,0,0));
```

attemptMoveTo()

使用法

```
< int >r.attemptMoveTo(vector vposition)
```

解説

この呼び出しは、衝突がない場合、指定した位置に剛体を移動します。指定した位置で衝突が検出された場合、オブジェクトは移動されません。

パラメータ

パラメータ	解説
vPosition	必須。オブジェクトを移動する位置を指定するベクトル。

例文

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.attemptMoveTo(vector(-100,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
objRB.atemptMoveTo(vector(-100,0,0));
```

制限のメソッド

ConstraintDesc

使用法

```
ConstraintDesc(sname,ObjectA,ObjectB,vPointA,vPointB,fStiffness,fDamping)
```

vPointA と vPointB は、剛体の内部になければなりません。これらはオブジェクト座標で指定します。点がボディ上にない場合、動作は未定義です。

リニアジョイントでは、オブジェクト A に関連しているオブジェクト B の整列が createLinearJoint メソッドの orientation パラメータによって指定されます。

解説

スプリング、角ジョイント、またはリニアジョイントを作成する前に、制限記述子オブジェクトを作成する必要があります。

注意：既存の名前を持つ ConstraintDesc を作成しようとすると、-5 エラーが表示されます。

パラメータ

すべての制限は制限記述子を使用して作成されます。型と内容は次のとおりです。

パラメータ	解説
string Name	制限の名前
ObjectA	制限を追加する剛体 A
ObjectB	制限を追加する剛体 B
vector PoCA	剛体 A に制限を付加する点
vector PoCB	剛体 B に制限を付加する点
float Stiffness	制限の剛性の値
float damping	制限の減衰の値

例文

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
-- Linear Joint constraint Descriptor between a rigid body b1 and point in world.
LJointDesc = ConstraintDesc("LJoinDesc",b1,void,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
```

注意：

- ワールド内の点と剛体の間に制限を作成するには、ObjectB の値を void と設定します。
- 点と静的な剛体の間、2 つの静的な剛体の間、またはワールド内の 2 つの点の間に制限記述子を作成することはできません。

関連事項

createAngularJoint(),createLinearJoint(),createSpring(),createD6Joint

createAngularJoint()

使用法

```
<AngularJoint> world.createAngularJoint(ConstraintDesc desc , float length)
```

解説

角ジョイントを作成します。角ジョイントは、角の方向に自由に移動できます。リニア方向に制限されます。制限された長さとして保持される長さは、第 2 のパラメータとして指定します。

パラメータ

パラメータ	解説
ConstraintDesc	必須。制限記述子オブジェクトへの参照。
restlength	必須。ジョイントの静止長を指定する浮動小数点数。

例文

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
AJointDesc = ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a angular joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint(AJointDesc, 5.0)
//JavaScript Syntax
//Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
var AJointDesc = ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a angular Joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint (AJointDesc,5.0);
```

関連事項

ConstraintDesc, createLinearJoint(), createSpring(), createD6Joint

createLinearJoint()

使用法

```
<LinearJoint> world.createLinearJoint(ConstraintDesc desc , list orientation)
```

解説：

リニアジョイントを作成します。リニアジョイントは、直線方向に自由に移動できます。角方向に制限されます。制限された角度として保持される角度は、第 2 のパラメータとして指定します。

パラメータ

パラメータ	解説
ConstraintDesc	必須。制限記述子オブジェクトへの参照。
Orientation	必須。ジョイントの向きを指定するリスト値。

例文

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
LJointDesc = ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint(LJointDesc,[vector(1,1,0),45])
//JavaScript Syntax
//Linear Joint constraint Descriptor between two rigid bodies' b1 and b2.
var LJointDesc = ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint (LJointDesc, list(vector(1,1,0),45));
```

関連事項

createAngularJoint() , ConstraintDesc, createSpring(), createD6Joint

createSpring()

使用法

```
<Spring> world.createSpring( ConstraintDesc desc, symbol forceExertionMode , float restLength)
```

解説

制限記述子に指定した詳細を使用してストリングを作成します。

パラメータ

パラメータ	解説
ConstraintDesc	必須。制限記述子オブジェクトへの参照。
forceexertionmode	次のシンボル値を指定します。 #kDuringCompression - ObjectA と ObjectB の間において、実際の長さが静止長より小さい場合にのみ復元力が適用されます。 #kDuringExpansion - objectA と objectB の間において、実際の長さが静止長より長い場合にのみ復元力が適用されます。 #kBoth - これらの両方の場合に復元力が適用されます。
restlength	必須。スプリングの静止長を指定する浮動小数点数。

例文

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
```

関連事項

createAngularJoint(),createLinearJoint(),ConstraintDesc,createD6Joint

deleteConstraint()

使用法

```
<int>deleteConstraint(string constraintname)
<int>deleteConstraint(Constraint constraintref)
```

解説

制限の名前を使用して指定した制限を削除します。

パラメータ

パラメータ	解説
ConstraintName	必須。制限のストリングによる名前です。

or

パラメータ	解説
Constraintref	必須。制限オブジェクトへの参照。

例文

```
--Lingo Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc")
//JavaScript Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc");
```

deleteSpring()

使用法

```
<int>deleteSpring(string springname)
<int>deleteSpring(Spring spring)
```

解説

スプリングの名前を使用して指定したストリングを削除します。

パラメータ

パラメータ	解説
Springname	必須。作成されるスプリングの名前を指定するストリング値。

例文

```
--Lingo Syntax
-- Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)
--Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc")
//JavaScript Syntax
//Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc = ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
//Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc");
```

getConstraint()

使用法

```
<AngularJoint / LinearJoint>getConstraint(string constraintname)
```

解説

指定した名前の制限を返します。

パラメータ

パラメータ	解説
ConstraintName	必須。制限のストリングによる名前です。

ジョイントへの参照を返します。

例文

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc")
//JavaScript Syntax
var objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc");
```

関連事項

ConstraintDesc

getConstraints()

使用法

```
<list of Constraints> world.getConstraints(symbol constraintType)
```

解説

指定した種類のすべての制限を返します。#linearjoint、#angularjoint、または #d6joint のいずれかになります。

パラメータ

パラメータ	解説
constraintType	オプション。制限の種類を指定するシンボル。指定されない場合、物理ワールドにおけるスプリングを除くすべての制限を返します。

例文

```
--Lingo Syntax
--Returns all the constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints()
--Returns all the Linear joint constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints(#linearjoint)
//JavaScript Syntax
var lstJoint = member("PhysicsWorld").getConstraints(symbol("angularjoint"));
```

関連事項

ConstraintDesc

getSpring()

使用法

```
<Spring > world.getSpring(string springname)
```

解説

指定した名前のスプリングオブジェクトを返します。

パラメータ

パラメータ	解説
Springname	必須。作成されるスプリングの名前を指定するストリング値。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringName")
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringName");
```

関連事項

ConstraintDesc

getSprings()

使用法

```
<list of Springs> world.getSprings();
```

解説

物理ワールド内のすべてのストリングのリストを返します。

衝突の関数：デフォルトでは、ワールド内のすべてのボディ間の衝突が有効になっています。これらの関数を使用して、ボディの組の間の衝突を有効または無効にすることができます。

パラメータ

なし

例文

```
--Lingo Syntax
lSprings= member("PhysicsWorld").getSprings()
//JavaScript Syntax
var lSprings = member("PhysicsWorld").getSprings();
```

制限のプロパティ

constraintType

使用法

```
c.constraintType
```

アクセス： 取得のみ

解説

制限の種類を返します。#linearjoint、#angularjoint、または #6djoint になります。

例文

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.constraintType
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.constraintType);
```

damping

使用法

```
c.damping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

制限の減衰を返します。

例文

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.damping
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.damping);
```

name

使用法

```
c.name
```

アクセス： 取得のみ

タイプ： ストリング

解説

制限の名前を返します。

例文

```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("ConstraintA")
put objConstraint.name
//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("ConstraintA");
put(objConstraint.name);
```

objectA

使用法

```
s.objectA
```

アクセス： 取得のみ

タイプ： 剛体

解説

ジョイントに接続されている剛体 "A" への参照を返します。

例文

```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectA
//JavaScript Syntax.
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectA);
```

objectB

使用法

```
s.objectB
```

アクセス： 取得のみ

タイプ： 剛体

解説

スプリングに接続されている剛体 "B" への参照を返します。

例文

```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint ("lJoint01")
put objJoint.objectB
--Change one end of the Joint to a worldPoint
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectB
//JavaScript Syntax.
--Change one end of the joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectB);
```

pointA

使用法

```
s.pointA
```

アクセス： 取得のみ

タイプ： ベクトル

解説
ジョイントがオブジェクト A に接続されている点を返します。

例文
```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointA
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointA);
```

pointB

使用法
```
s.pointB
```

アクセス： 取得のみ

タイプ： ベクトル

解説
ジョイントがオブジェクト B に接続されている点を返します。

例文
```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointB
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointB);
```

プロパティ

使用法
```
c.properties
```

アクセス： 取得および設定

タイプ： プロパティリスト

解説
ジョイントのプロパティを表すプロパティリストを返します。例えば、リニアジョイントは向きを返し、角ジョイントは長さを返します。

例文
```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.properties
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.properties);
```

stiffness

使用法

```
s.stiffness
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

制限の剛性を返します。

例文

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.stiffness
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.stiffness);
```

スプリングのプロパティ

damping

使用法

```
s.damping
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

スプリングの減衰を表します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.damping
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.damping);
```

flagsflags

使用法

```
s.flags
```

アクセス： 取得および設定

解説

スプリングに適用される伸縮力を表します。#kDuringCompression、#kDuringExpansion、または #kBoth のいずれかを返します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.flags
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.flags);
```

name

使用法

```
s.name
```

アクセス： 取得のみ

タイプ： ストリング

解説

スプリングの名前を返します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringA")
put objSpring.name
//Javascript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringA");
put(objSpring.name);
```

objectA

使用法

```
s.objectA
```

アクセス： 取得および設定

タイプ： 剛体

解説

スプリングに接続されている剛体「A」の名前を返します。

例文

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = rigidbodyX
--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = void
//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectA);
objSpring.objectA = rigidbodyX;
```

objectB

使用法

```
s.objectB
```

アクセス： 取得および設定

タイプ： 剛体

解説

スプリングに追加されている剛体「B」の名前を返します。

例文

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = rigidbodyX

--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = void
//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectB);
objSpring.objectB = rigidbodyX;
```

pointA

使用法

```
s.pointA
```

アクセス： 取得および設定

タイプ： ベクトル

解説

スプリングがオブジェクト A に接続されている点を返します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointA
//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointA);
```

pointB

使用法

```
s.pointB
```

アクセス： 取得および設定

タイプ： ベクトル

解説

スプリングがオブジェクト B に接続されている点を返します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointB
//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointB);
```

restLength

使用法

```
s.restLength
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

静止時のスプリングの長さを表します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.restLength
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.restLength);
```

stiffness

使用法

```
s.stiffNess
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

スプリングの剛性を表します。

例文

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.stiffness
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.stiffness);
```

衝突および衝突コールバックのメソッド

disableCollision()

使用法

```
<int>world .disableCollision(RigidBody a, RigidBody b)
<int>world .disableCollision(string rigidbodyname1, string rigidbodyname2)
```

解説

A と B が void の場合は、ワールド内のすべての剛体間の衝突が無効になります。

B が void の場合、剛体 A とすべてのボディとの衝突が無効になります。

A と B を指定すると、これらの 2 つのボディ間の衝突は、以前に有効になっている場合は無効になります。

注意：disableCollision( ) はすべてのコールバックを無効にします。ただし、enableCollision( ) を使用してもコールバックは自動的に有効になりません。コールバックを有効にするための enableCollisionCallback( ) を、個別に呼び出す必要があります。

注意：

パラメータ

衝突を無効にする必要がある剛体の数によって異なります。

例文

```
--Lingo Syntax
--All collisions are disabled.
member("PhysicsWorld").disableCollision()
--Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA)
--Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collisions are disabled.
member("PhysicsWorld").disableCollision();
//Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA);
//Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB);
```

関連事項

enableCollision()

disableCollisionCallback()

使用法

```
<int>world .disableCollisionCallback(RigidBody a, RigidBody b)
<int>world . disableCollisionCallback (string rigidbodyname1, string rigidbodyname2)
```

解説

パラメータを渡さない場合、この呼び出しより前に作成されたすべての剛体に対して、衝突コールバックレポートが無効になります。

このメソッドにパラメータを渡すと、衝突コールバックは指定した剛体に対してのみ無効になります。

パラメータ

衝突コールバックを無効にする必要がある剛体の数によって異なります。

例文

```
--Lingo Syntax
--All collisions callbacks are disabled for rigid bodies created before this call.
member("PhysicsWorld").disableCollisionCallback()
--Collisions callback disabled for rigid body A with all other bodies.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA)
--Collisions  callback disabled between rigidbody A and rigidbody B.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collision callbacks are disabled.
member("PhysicsWorld").disableCollisionCallback();
//Collision callbacks disabled for rigid body A
member("PhysicsWorld").disableCollisionCallback(rigidbodyA);
//Collision callbacks disabled between rigidbody A and rigidbody B
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB);
```

関連事項

enableCollisionCallback(),getCollisionCallbackDisabledPairs(),getCollisionDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

enableCollision()

使用法

```
<int>world .enableCollision(RigidBody a, RigidBody b)
<int>world .enableCollision(string rigidbodyname1, string rigidbodyname2)
```

解説

パラメータを渡さない場合、すべての衝突が有効になります。それ以外の場合、衝突は指定された剛体に対してのみ有効になります。

パラメータ

衝突を有効にする必要がある剛体の数によって異なります。

例文

```
--Lingo Syntax
--All collisions are enabled.
member("PhysicsWorld").enableCollision()
--Collisions enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollision(rigidbodyA)
--Collisions enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collisions are enabled.
member("PhysicsWorld").enableCollision();
//Collisions enabled for rigid body A
member("PhysicsWorld").enableCollision(rigidbodyA);
//Collisions enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB);
```

関連事項

disableCollision()

enableCollisionCallback()

使用法

```
<int>world .enableCollisionCallback(RigidBody a, RigidBody b)
<int>world . enableCollisionCallback(string rigidbodyname1, string rigidbodyname2)
```

解説

パラメータを渡さない場合、この呼び出しより前に作成されたすべての剛体に対して、衝突コールバックレポートが有効になります。

このメソッドにパラメータを渡すと、衝突コールバックは指定した剛体に対してのみ有効になります。

パラメータ

衝突コールバックを有効にする必要がある剛体の数によって異なります。

例文

```
--Lingo Syntax
--All collisions callbacks are enabled for rigid bodies created before this call.
member("PhysicsWorld").enableCollisioncallback()
--Collisions callback enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA)
--Collisions  callback enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collision callbacks are enabled.
member("PhysicsWorld").enableCollisioncallback();
//Collision callbacks enabled for rigid body A
member("PhysicsWorld").enableCollisioncallback(rigidbodyA);
//Collision callbacks enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB);
```

関連事項

disableCollisionCallback(),getCollisionDisabledPairs(),getCollisionCallbackDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

getCollisionCallbackDisabledPairs()

使用法

```
<[[RigidBody,RigidBody]…]>world .getCollisionCallbackDisabledPairs()
```

解説

衝突コールバックが無効になっている剛体の組すべてのリストを返します。

パラメータ

なし

例文

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs()
lstPair1 = lstPairs[1]
//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs();
var lstPair1 = lstPairs(0);
```

関連事項

enableCollisionCallback(),disableCollisionCallback(),getCollisionDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

getCollisionDisabledPairs()

使用法

```
<[[RigidBody,RigidBody]…]>world .getCollisionDisabledPairs()
```

解説

衝突が無効になっている剛体の組すべてのリストを返します。

パラメータ

なし

例文

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionDisabledPairs()
lstPair1 = lstPairs[1]
//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionDisabledPairs();
var lstPair1 = lstPairs(0);
```

関連事項

enableCollisionCallback(),disableCollisionCallback(),getCollisionCallbackDisabledPairs(), registerCollisionCallback(),removeCollisionCallback()

registerCollisionCallback()

使用法

```
<int>world .registerCollisionCallback( #collisionCallback,<script reference>)
```

解説

この関数は、衝突を報告するために呼び出されるコールバック関数を登録します。物理シーンに対して衝突コールバックハンドラは 1 つだけ存在することができます。

ILK(<rigidBody_handler>) メソッドを使用すると、衝突に関与する剛体の種類を調べることができます。

衝突に関する情報は、衝突レポートリストとしてこの関数に渡されます。このレポートに含まれる情報は次のとおりです。

パラメータ

パラメータ	解説
CollisionCallback	必須。衝突コールバックハンドラの名前を指定します。
スクリプト参照	オプション。ビヘイビアーまたは親スクリプトインスタンスです。このパラメータを省略すると、コールバック関数はムービースクリプトで定義されると仮定されます。

衝突レポートの構造：

ObjectA	オブジェクト A が衝突
ObjectB	オブジェクト B が衝突
リスト (ベクトル < 浮動小数点数 , 浮動小数点数 , 浮動小数点数 >)	オブジェクト A とオブジェクト B の間の接触点
リスト (ベクトル << 浮動小数点数 ,, 浮動小数点数 ,, 浮動小数点数 >	上記の各点の接触法線

注意：衝突コールバックハンドラの内部では、いずれかのスクリプトで実行時エラー（プロパティが見つからないなど）が発生した場合、エラーメッセージは表示されず、ハンドラの実行はエラーが発生した時点で中止されます。後続のすべてのステートメントは実行されません。

resetworld()、deleteModel()、deleterigidBody() などの衝突コールバックハンドラの内部では、3D シーンを変更しないでください。

例文

```
--Lingo Syntax
member("PhysicsWorld").registercollisionCallback(#collisioncallback)
--Movie Script
On collisionCallback collisionReport
CollisionA = collisionReport[1]
R1 = CollisionA.objectA
R2 = CollisionA.objectB
lstPoints = CollisionA.contactPoints
lstNormals = CollisionA.contactNormals
end
//JavaScript Syntax
member("PhysicsWorld").registerCallback(symbol("collisioncallback"));
//Movie Script
function collisionCallback(collisionreport)
{
var CollisionA = collisionReport(0);
var R1 = CollisionA.objectA;
var R2 = CollisionA.objectB;
var lstPoints = CollisionA.contactPoints;
var lstNormals = CollisionA.contactNormals;
}
```

関連事項

enableCollisionCallback(), disableCollisionCallback(), getCollisionDisabledPairs(), getCollisionDisabledPairs(), removeCollisionCallback()

removeCollisionCallback()

使用法

```
<int>world .removeCollisionCallback()
```

解説：

この関数は登録済みのコールバックを削除します。

パラメータ

なし

例文

```
--Lingo Syntax
member("PhysicsWorld").removeCollisionCallback()
//JavaScript Syntax
member("PhysicsWorld").removeCollisionCallback();
```

関連事項

enableCollisionCallback(), disableCollisionCallback(), getCollisionDisabledPairs(), getCollisionCallbackDisabledPairs(), registerCollisionCallback()

地形のメソッド

注意：剛体と地形には別々の名前を使用してください。剛体と地形に同じ名前を使用すると、無効なパラメータを示すエラー -4 が返されます。

createTerrain

使用法

```
createTerrain(terrainName, terrainDesc, position, orientation, rowScale, columnScale, heightScale)
```

解説

このメソッドは、指定した高さマップ情報を使用し、地形を静的な剛体として作成します。

注意：高さマップ画像を使用してメッシュと高さマップを作成するときのパフォーマンスを高めるには、メッシュを細分化し、ステッチ処理で元に戻すようにすると効果的です。

パラメータ

パラメータ	解説
terrainName	必須。地形の名前となるストリング。
terrainDesc	必須。地形記述子への参照。
position	必須。地形の位置を設定するベクトル。
orientation	必須。地形の方向を設定する、軸と角度のリスト（[vector(1,0,0), 45] など）。
rowScale	必須。行スケール係数を指定する浮動小数点値です。
columnScale	必須。列スケール係数を指定する浮動小数点値です。
heightScale	必須。高さスケール係数を指定する浮動小数点数。

例文

```
--Lingo Syntax
objTerrain = member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1)
//JavaScript Syntax
var objTerrain= member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1);
```

関連事項

createTerrainDesc,terrainDesc

createTerrainDesc

使用法

```
createTerrainDesc(elevationMatrix,friction,restitution)
```

解説

地形を作成する前に、地形記述子オブジェクトを作成する必要があります。

パラメータ

パラメータ	解説	値
ElevationMatrix	必須。地形の高さマップ情報を表す Lingo マトリックス。	
Friction	必須。摩擦係数を表す浮動小数点数。	0-1
Restitution	必須。反発係数を表す浮動小数点数。	0-1

例文

```
--Lingo Syntax
tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5);
```

関連事項

createTerrain,terrainDesc

deleteTerrain

使用法

```
world.deleteterrain(String terrainName)
world.deleteterrain(terrain refTerrain)
```

解説

このメソッドは、物理シーンから地形を削除します。パラメータとして地形の名前または参照を受け入れます。

パラメータ

パラメータ	解説
terrainName	必須。作成済みの地形の名前を指定するストリング値。

or

パラメータ	解説
refTerrain	必須。作成済みの地形オブジェクトへの参照。

例文

```
--Lingo Syntax
member("PhysicsWorld").deleteTerrain("myterrain")
//JavaScript Syntax
member("PhysicsWorld").deleteTerrain("myTerrain");
```

getTerrain

使用法

```
<terrain refTerrain> world.getterrain(String "myterrain")
```

解説
このメソッドは、指定した名前の地形オブジェクトを返します。

パラメータ

パラメータ	解説
myTerrain	：必須。地形の名前を指定するストリング。

例文
```
--Lingo Syntax
objTerrain = member("PhysicsWorld").getTerrain("myTerrain")
//JavaScript Syntax
var objTerrain = member("PhysicsWorld").getTerrain("myTerrain");
```

関連事項
createTerrainDesc,terrainDesc

getTerrains

使用法
```
<list lstTerrain> world.getterrains()
```

解説
このメソッドは、物理ワールド内の地形オブジェクトのリストを返します。

パラメータ
パラメータはなし。

例文
```
--Lingo Syntax
lstTerrain = member("PhysicsWorld").getTerrains()
//JavaScript Syntax
var lstTerrain = member("PhysicsWorld").getTerrains();
```

地形のプロパティ

columnScale

使用方法
```
t.columnScale
```

アクセス： 取得のみ

解説
地形の列スケール係数を表します。

例文

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.columnScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.columnScale);
```

contactTolerance

使用方法

```
t.contactTolerance
```

アクセス： 取得および設定

タイプ： 浮動小数点数

解説

衝突が検出されるための地形と他の剛体や地形との間の浸透深度です。

例文

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.contactTolerance
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.contactTolerance);
```

heightScale

使用方法

```
t.heightScale
```

アクセス： 取得のみ

解説

地形の高さスケール係数を表します。

例文

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.heightScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.heightScale);
```

name

使用法

```
t.name
```

アクセス： 取得のみ

タイプ： ストリング

解説
地形の名前を返します。

例文
```
--Lingo Syntax
put objTerrain.name
//Javascript Syntax
put(objTerrain.name);
```

orientation

使用方法
```
t.orientation
```

アクセス： 取得のみ

解説
地形の方向を表す、軸と角度のリスト（[vector(1,0,0), 45] など）。

例文
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.orientation
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.orientation);
```

position

使用方法
```
t.position
```

アクセス： 取得のみ

解説
地形の位置を表すベクトル。

例文
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.position
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.position);
```

rowScale

使用方法

```
t.rowScale
```

アクセス： 取得のみ

解説：地形の行スケール係数を表します。

例文

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.rowScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.rowScale);
```

terrainDesc

使用方法

```
t.terrainDesc
```

アクセス： 取得のみ

解説

地形記述子オブジェクトを表します。

地形記述子には以下の属性があります。

- terraindesc.elevationmatrix
- terraindesc.friction
- terraindesc.restitution
- terraindesc.numrows
- terraindesc.numcolumns

注意：numrows と numcolumns は高度マトリックスの行と列に対応しています。

例文

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.terrainDesc
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.terrainDesc);
```

関連事項

createTerrainDesc,createTerrain

6DOF ジョイントのメソッド

createD6Joint

設定されるジョイント軸とドライブ値に関連して、剛体が 6DOF ジョイントへの入力として提供される順序は重要です。例えば、順序が rb,rb2 でドライブの位置がベクトル（10,0,0）である場合、剛体がその位置に到達するために移動する方向は、順序を rb2,rb1 とした場合に移動する方向とは完全に逆です。

createD6Joint

使用法

```
<d6joint> world.createD6Joint(jointName, RigidBody rb1 ,Rigidbody rb2, vector globalanchor)
```

解説

このメソッドは、2 つの剛体の間またはワールド内の点と剛体の間に、6DOF ジョイントを作成します。

注意：D6 ジョイントはリニアジョイントや角ジョイントのような制限の種類です。D6 ジョイントの constraintType は #d6joint です。

D6 ジョイントを削除するには、deleteConstraint() を使用します。

D6 ジョイントに固有の追加のプロパティについては、D6 ジョイントのプロパティの項に記載されています。

パラメータ

パラメータ	解説
jointName	必須。ジョイントの名前を指定するストリング。
rb1	必須。ジョイントに関与している剛体への参照。点と剛体の間にジョイントを作成する必要がある場合は、このパラメータを void にする必要があります。
rb2	必須。ジョイントに関与している剛体への参照。
globalAnchor	必須。ジョイントのアンカーポイントを指定するベクトル。

例文

```
--Lingo Syntax
--Creates a joint between rigid bodies rb1 and rb2.
ObjJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1))
-- Creates a joint between the anchor point and rb2.
objJoint = member("PhysicsWorld").createD6Joint("myJoint",void,rb2,vector(1,1,1))
//JavaScript Syntax
//Creates a joint between rigid bodies rb1 and rb2.
var objJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1));
```

関連事項

createAngularJoint(), createLinearJoint(), createSpring(), ConstraintDesc, globalAnchor

6DOF のプロパティ

axisDrive

使用方法

```
d6joint.axisDrive = [type, spring, damping, forceLimit]
```

アクセス： 取得および設定

解説

ジョイントをジョイントの軸に沿って指定の位置まで駆動する、ジョイントのリニアドライブを指定するリスト値。

リストには次の属性が含まれます。

type #position または #velocity

stiffness 駆動中に適用するスプリング（> 0）

damping スプリングの減衰（> 0）

forceLimit 指定の位置または速度まで駆動するための力またはトルク（> 0）

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.axisDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.axisDrive = [#position,100,0.1,100];
```

関連事項

biNormalDrive, normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition

axisMotion

使用方法

```
D6joint.axisMotion
```

アクセス： 取得および設定

解説

ジョイントの軸に沿ったリニア自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.axisMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.axisMotion);
```

関連事項

biNormalMotion, normalMotion, swing1Motion, swing2Motion, twistMotion

biNormalDrive

使用方法

```
d6joint.binormalDrive = [type, spring, damping, forceLimit]
```

アクセス： 取得および設定

解説

ジョイントをジョイントの従法線軸に沿って指定の位置まで駆動する、ジョイントのリニアドライブを指定するリスト値。

リストには次の属性が含まれます。

type #position または #velocity

stiffness 駆動中に適用するスプリング（> 0）

damping スプリングの減衰（> 0）

forceLimit 指定の位置または速度まで駆動するための力またはトルク（> 0）

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.binormalDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.binormalDrive = [#position,100,0.1,100];
```

関連事項

normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive

biNormalMotion

使用方法

```
D6joint.binormalMotion
```

アクセス： 取得および設定

解説

ジョイントの従法線軸に沿ったリニア自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.biNormalMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.biNormalMotion);
```

関連事項
normalMotion, swing1Motion,swing2Motion, twistMotion, axisMotion

constraintType

使用方法
```
d6joint.constraintType
```

アクセス： 取得のみ

解説
D6 ジョイントはリニアジョイントや角ジョイントのような制限の種類です。D6 ジョイントの constraintType は #d6joint です。

例文
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.constraintType
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
put(objJoint.constraintType);
```

driveAngularVelocity

使用方法
```
d6joint.driveAngularVelocity = vector velocity
```

アクセス： 取得および設定

解説
swingDrive または twistDrive のドライブタイプが #velocity と指定されているときに、目的の角速度を指定するベクトル値。

例文
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveAngularVelocity = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveAngularVelocity = vector(20,10,0);
```

関連事項
normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveLinearVelocity

使用方法
```
d6joint.driveLinearVelocity = vector velocity
```

アクセス： 取得および設定

解説

axisDrive、normalDrive、または binormalDrive のドライブタイプが #velocity であるときに、目的のリニア速度を指定するベクトル値。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveLinearVelocity = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveLinearVelocity = vector(20,10,0);
```

関連事項

normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveOrientation

使用方法

```
d6joint.driveOrientation = list orientation
```

アクセス： 取得および設定

解説

swingDrive または twistDrive のドライブタイプが #position と指定されているときに、目標の向きを指定するリスト値。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveOrientation = [vector(20,10,0),45]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveOrientation = [vector(20,10,0),45];
```

関連事項

normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveAngularVelocity, drivePosition, axisDrive, biNormalDrive

drivePosition

使用方法

```
d6joint.drivePosition = vector position
```

アクセス： 取得および設定

解説

axisDrive、normalDrive、および binormalDrive のドライブタイプが #position と指定されているときに、目標の位置を指定するベクトル値。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.drivePosition = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.drivePosition = vector(20,10,0);
```

関連事項

normalDrive, swingDrive,twistDrive, driveLinearVelocity, driveAngularVelocity, driveOrientation, axisDrive, biNormalDrive

globalAnchor

使用方法

```
d6joint.globalAnchor = vector position
```

アクセス： 取得および設定

解説

ジョイントのアンカーポイントを指定するベクトル値。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.globalAnchor = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.globalAnchor = vector(20,10,0);
```

関連事項

localAnchorA, localAnchorB

linearLimit

使用方法

```
D6joint.linearlimit = [limitvalue, stiffness, damping, restitution]
```

アクセス： 取得および設定

解説

直線運動が制限付きであるときに、ジョイントの動作を指定するリスト。剛体は制限値に達すると振動します。これはすべての直線運動に適用可能です。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.linearLimit =  [2, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.linearLimit = [2, 100, 0.01, 0];
```

注意：制限を有効にするには、まず制限値を設定してから運動パラメータ（#free、#limited など）を設定します。

関連事項
swing1Limit, swing2Limit, twistLimit

localAnchorA

使用方法
```
D6joint.localAnchorA
```

アクセス： 取得および設定

解説
objectA のスペースにおけるジョイントの結合ポイント。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

注意：ジョイントを安定させるには、localAnchorA と localAnchorB に整合性のある値を指定して、ワールドスペースで両方が同じ点に対応するようにします。

例文
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorA = vector(0,5,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorA = vector(0,5,0);
```

関連事項
localAnchorB, globalAnchor

localAnchorB

使用方法
```
D6joint.localAnchorB
```

アクセス： 取得および設定

解説
objectB のスペースにおけるジョイントの結合ポイント。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

例文
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorB = vector(0,-5,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorB = vector(0,-5,0);
```

関連事項
localAnchorA, globalAnchor

localAxisA

使用方法
```
D6joint.localAxisA
```

アクセス： 取得および設定

解説
これは D6Joint の objectA のローカル座標スペースにおけるジョイントの基本軸です。localAxisA と localNormalA は、互いに正しい角度になければなりません。

剛体が静的である場合、ジョイントの軸に沿った方向に向けることができないため、ボディに対して設定したローカル軸がグローバルなものと見なされます。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

- ジョイントの主軸を中心にひねりが行われ、ジョイントの主軸に沿って AxisMotion が行われます。
- ジョイントの法線軸を中心に swing1 が行われ、ジョイントの法線軸に沿って NormalMotion が行われます。
- ジョイントの従法線軸を中心に swing2 が行われ、ジョイントの従法線軸に沿って BinormalMotion が行われます。

ジョイントの軸を任意の時点で変更するには、以下のプロパティを適切に設定します。

- localAxisA
- localNormalA
- localAxisB
- localNormalB

例文
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

関連事項
localAxisB, localNormalA, localNormalB

localAxisB

使用方法

```
D6joint.localAxisB
```

アクセス： 取得および設定

解説

これは D6Joint の objectB のローカル座標スペースにおけるジョイントの基本軸です。localAxisB と localNormalB は、互いに正しい角度になければなりません。この軸を中心にひねりの回転が定義され、この軸に沿って AxisMotion が定義されます。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

関連事項

localAxisA, localNormalA, localNormalB

localNormalA

使用方法

```
D6joint.localNormalA
```

アクセス： 取得および設定

解説

これは D6Joint の objectA のローカル座標スペースにおけるジョイントの上下軸です。localAxisA と localNormalA は、互いに正しい角度になければなりません。この軸を中心に swing1 の回転が定義され、この軸に沿って NormalMotion が定義されます。

剛体が静的である場合、ジョイントの軸に沿った方向に向けることができないため、ボディに対して設定したローカル軸がグローバルなものと見なされます。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

関連事項

localAxisB, localAxisA, localNormalB

localNormalB

使用方法

```
D6joint.localNormalB
```

アクセス： 取得および設定

解説

これは D6Joint の objectB のローカル座標スペースにおけるジョイントの上下軸です。localAxisB と localNormalB は、互いに正しい角度になければなりません。この軸を中心に swing1 の回転が定義され、この軸に沿って NormalMotion が定義されます。

プロパティ localAxisA、localAxisB、localNormalA、localNormalB、localAnchorA、および localAnchorB の値は、これらのプロパティに明示的に値を設定して初めて使用できるようになります。値を設定していない場合は、void ベクトルが返されます。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

関連事項

localAxisA, localAxisB, localNormalA

name

使用法

```
d6joint.name
```

アクセス： 取得のみ

タイプ： ストリング

解説

6DOF ジョイントの名前を返します。

例文

```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("d6joint")
put objConstraint.name
//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("d6joint");
put(objConstraint.name);
```

normalDrive

使用方法

```
d6joint.normalDrive = [type, spring, damping, forceLimit]
```

アクセス： 取得および設定

解説

ジョイントをジョイントの法線軸に沿って指定の位置まで駆動する、ジョイントのリニアドライブを指定するリスト値。

リストには次の属性が含まれます。

type #position または #velocity

注意：position を選択すると、forcelimit の値は無視され、spring の値のみが考慮されます。同様に、velocity を選択すると、spring の値は無視され、forcelimit の値のみが考慮されます。

stiffness 駆動中に適用するスプリング（> 0）

damping スプリングの減衰（> 0）

forceLimit 指定の位置または速度まで駆動するための力またはトルク（> 0）

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.normalDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.normalDrive = [#position,100,0.1,100];
```

関連事項

swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

normalMotion

使用方法

```
D6joint.normalMotion
```

アクセス： 取得および設定

解説

ジョイントの法線軸に沿ったリニア自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.normalMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.normalMotion);
```

関連事項

swing1Motion, swing2Motion, twistMotion, axisMotion, biNormalMotion

objectA

使用法

```
d6joint.objectA
```

アクセス： 取得のみ

タイプ： 剛体

解説
点に接続されている剛体「A」の名前を返します。

例文
```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectA
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectA);
```

関連事項
objectB

objectB

使用法
```
d6joint.objectB
```

アクセス： 取得のみ

タイプ： 剛体

解説
スプリングに追加されている剛体「B」の名前を返します。

例文
```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectB
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectB);
```

関連事項
objectA

swing1Limit

使用方法
```
D6joint.swing1limit = [limitvalue, stiffness, damping, restitution]
```

アクセス： 取得および設定

解説
ジョイントの法線軸を中心にした角運動が制限付きであるときに、ジョイントの動作を指定するリスト。剛体は制限値に達すると振動します。

制限値がとることができる値は 3.14 ～ -3.14（円周率）です。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing1limit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing1limit = [3.14*0.5, 100, 0.01, 0];
```

注意：制限を有効にするには、まず制限値を設定してから運動パラメータ（#free、#limited など）を設定します。

関連事項

swing2Limit,twistLimit,linearLimit

swing1Motion

使用方法

```
D6joint.swing1Motion
```

アクセス： 取得および設定

解説

ジョイントの法線軸を中心とした角自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing1Motion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing1Motion);
```

関連事項

normalMotion, swing2Motion, twistMotion, axisMotion, biNormalMotion

swing2Limit

使用方法

```
D6joint.swing2limit = [limitvalue, stiffness, damping, restitution]
```

アクセス： 取得および設定

解説

ジョイントの従法線軸を中心にした角運動が制限付きであるときに、ジョイントの動作を指定するリスト。剛体は制限値に達すると振動します。

制限値がとることができる値は 3.14 ～ -3.14（円周率）です。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing2limit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing2limit = [3.14*0.5, 100, 0.01, 0];
```

注意：制限を有効にするには、まず制限値を設定してから運動パラメータ（#free、#limited など）を設定します。

関連事項

swing1Limit, twistLimit, linearLimit

swing2Motion

使用方法

```
D6joint.swing2Motion
```

アクセス： 取得および設定

解説

ジョイントの従法線軸を中心とした角自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing2Motion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing2Motion);
```

関連事項

normalMotion, swing1Motion, twistMotion, axisMotion, biNormalMotion

swingDrive

使用方法

```
d6joint.swingDrive = [type, spring, damping, forceLimit]
```

アクセス： 取得および設定

解説

ジョイントをジョイントの法線軸と従法線軸を中心に指定の向きまで駆動する、ジョイントの角ドライブを指定するリスト値。

リストには次の属性が含まれます。

type #position または #velocity

stiffness 駆動中に適用するスプリング（> 0）

damping スプリングの減衰（> 0）

forceLimit 指定の位置または速度まで駆動するための力またはトルク（> 0）

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swingDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swingDrive = [#position,100,0.1,100];
```

関連事項

normalDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

twistDrive

使用方法

```
d6joint.twistDrive = [type, spring, damping, forceLimit]
```

アクセス： 取得および設定

解説

ジョイントをジョイントの軸を中心に指定の向きまで駆動する、ジョイントの角ドライブを指定するリスト値。

リストには次の属性が含まれます。

type #position または #velocity

stiffness 駆動中に適用するスプリング（> 0）

damping スプリングの減衰（> 0）

forceLimit 指定の位置または速度まで駆動するための力またはトルク（> 0）

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistDrive = [#position,100,0.1,100];
```

関連事項

normalDrive, swingDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

twistLimit

使用方法

```
D6joint.twistLimit = [limitvalue, stiffness, damping, restitution]
```

アクセス： 取得および設定

解説

ジョイントの軸に沿った角運動が制限付きであるときに、ジョイントの動作を指定するリスト。剛体は制限値に達すると振動します。

制限値がとることができる値は 3.14 ～ -3.14（円周率）です。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistLimit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistLimit = [3.14*0.5, 100, 0.01, 0];
```

注意：制限を有効にするには、まず制限値を設定してから運動パラメータ（#free、#limited など）を設定します。

関連事項

swing2Limit, swing1Limit, linearLimit

twistMotion

使用方法

```
D6joint.twistMotion
```

アクセス： 取得および設定

解説

ジョイントの主軸を中心とした角自由度を設定するシンボル。

このプロパティは以下の値をとります。

#Locked この DOF に沿った移動は行えません。

#Limited この DOF に沿って制限付きの移動が可能です。

#Free この DOF に沿った移動に制限はありません。

例文

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.twistMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.twistMotion);
```

関連事項

normalMotion, swing2Motion, swing1Motion, axisMotion, biNormalMotion

レイキャスティングのメソッド

rayCastAll

使用法

```
<list <list>> world.rayCastAll(vector origin, vector direction, sorted list)
```

解説

このメソッドは、指定された原点および指定された方向からレイに沿って発見される、すべての剛体または地形の参照を返します。また、接触点、接触法線、およびレイの原点からの距離も返します。

パラメータ

パラメータ	解説
origin	必須。レイキャスティングの原点を指定するベクトル。
direction	必須。レイキャスティングの方向を指定するベクトル。
ソート済み	オプション。距離に基づいてソートされた剛体のリストを返します。

このメソッドは、以下の情報を含むリストを返します。

- 剛体または地形の参照
- 接触点
- 接触法線
- レイの原点から剛体または地形までの距離

例文

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1), [#sorted:#distance])
repeat with i = 1 to lstraycast.count
    raycstEntry = lstraycast[i]
    put "Name:" & raycstEntry[1].name
    put "Contact Point:" & raycstEntry[2]
    put "Contact Normal:" & raycstEntry[3]
    put "Distance:" & raycstEntry[4]
end repeat
//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1),
propList(symbol("sorted"), symbol("distance"));
for(i = 1; i <= lstraycast.count ; i++)
{
    raycstEntry = lstraycast[i];
    put("Name:" + raycstEntry[1].name);
    put"Contact Point:" & raycstEntry[2]);
    put("Contact Normal:" & raycstEntry[3]);
    put("Distance:" & raycstEntry[4]);
}
```

関連事項

rayCastClosest

rayCastClosest

使用法

```
<list> world.rayCastClosest(vector origin, vector direction)
```

解説

このメソッドは、指定された原点および指定された方向からレイに沿って発見される、最も近い剛体または地形の参照を返します。また、接触点、接触法線、およびレイの原点からの距離も返します。

パラメータ

パラメータ	解説
origin	必須。レイキャスティングの原点を指定するベクトル。
direction	必須。レイキャスティングの方向を指定するベクトル。

このメソッドは、以下の情報を含むリストを返します。

- 剛体または地形の参照
- 接触点
- 接触法線
- レイの原点から剛体または地形までの距離

例文

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1))
put "Name:"  & lstraycast[1].name
put "Contact Point:"  & lstraycast[2]
put "Contact Normal:"  & lstraycast[3]
put "Distance:"  & lstraycast[4]
//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1));
put("Name:"  + lstraycast[0].name);
put"Contact Point:"  & lstraycast[1]);
put("Contact Normal:"  & lstraycast[2]);
put("Distance:"  & lstraycast[3]);
```

関連事項

rayCastAll

エラーコード

エラーコードと各エラーの理由を次の表に示します。

エラーコード	エラーの理由
-1	不明なエラー カメラとモデルが非常に離れていると、剛体を作成したときにエラー -1 が発生する場合があります。これは、最適化によってメッシュカウントが 0 になるためです。
-2	ワールド未初期化
-3	シミュレーション失敗
-4	無効なパラメータ
-5	制限の重複
-6	制限が無効、または制限が存在しません。
-8	剛体が無効、または剛体が存在しません。
-11	エレメントが見つかりません。削除された物理オブジェクトが使用されています。
-18	無効な 3D モデル
-19	3D エラー
-20	メモリ不足
-21	モデルにメッシュ変形が追加されていません。
-28	#concaveshape プロキシを使用する剛体が、#dynamic になっている場合
-29	重複するモデルリソース。
-30	プロキシテンプレートが無効である場合。
-31	モデルリソースが無効である場合。

索引

記号

_global プロパティ _global ぷろぱてぃ 705
_key プロパティ key ぷろぱてぃ 705
_mouse プロパティ _mouse ぷろぱてぃ 706
_movie プロパティ movie ぷろぱてぃ 707
_player プロパティ _player ぷろぱてぃ 707
_sound プロパティ sound ぷろぱてぃ 708
_system プロパティ _system ぷろぱてぃ 709
.（ドット演算子）（どっとえんざんし）684
"（二重引用符）"（にじゅういんようふ）167
"（ストリング定数）"（すとりんぐていすう）164
() (括弧演算子 () かっこえんざんし) 688
--（コメント記号）--（こめんときごう）686
-（マイナス記号（3D））-（まいなすきごう（3D））691
-（マイナス記号）685, 690
[]（ブラケットアクセス演算子）695
[]（リストブラケット）695
「コメントを解除」ボタンこめんとをかいじょぼたん 68
「コメントを設定」ボタンこめんとをせっていぼたん 68
「ステップ」ボタン、デバッガウィンドウ「すてっぷ」ぼたん、でばっがうぃんどう 89
@（パス名演算子）@（ぱすめいえんざんし）697
（乗算演算子（3D））（じょうざんえんざんし（3D））691
*（乗算演算子）689
/（除算演算子）/（じょざんえんざんし）691, 692
\ 692, 693
\ (等しくない) 693
\\（継続シンボル）214
\> (より大きい) 694
\>= (以上) 694
&（連結演算子）&（れんけつえんざんし）686
&&（連結演算子）&&（れんけつえんざんし）687
683
#（シンボル）データ型 #（しんぼる）でーたがた 10
#（シンボル演算子）#（しんぼるえんざんし）683
+（加算演算子）689, 690
=（代入演算子）=（だいにゅうえんざんし）693

数字

3D オブジェクト 43, 154
6DOF ジョイントのメソッド
  createD6joint 1293
6DOF のプロパティ
  axisDrive 1293
  axisMotion 1294
  biNormalDrive 1295
  biNormalMotion 1295
  driveAngularVelocity 1296
  driveLinearVelocity 1296
  driveOrientation 1297
  drivePosition 1297
  linearLimit 1298
  localAnchorA 1299
  localAnchorB 1299
  localAxisA 1300
  localAxisB 1301
  localNormalA 1301
  localNormalB 1302
  normalDrive 1303
  normalMotion 1304
  Swing1Limit 1305
  swing1Motion 1306
  Swing2Limit 1306
  swing2Motion 1307
  swingDrive 1307
  TwistDrive 1308
  twistLimit 1309
  twistMotion 1309

A

abort メソッド abort めそっど 239
aboutInfo プロパティ aboutInfo ぷろぱてぃ 709
abs() メソッド abs() めそっど 240
actionsEnabled プロパティ actionsEnabled ぷろぱてぃ 710
activateApplication ハンドラ activateApplication はんどら 171
activateAtLoc() メソッド activateAtLoc() めそっど 241
activateButton() メソッド activateButton() めそっど 241
active3dRenderer プロパティ active3dRenderer ぷろぱてぃ 710
activeCastLib プロパティ activeCastLib ぷろぱてぃ 711
activeWindow プロパティ activeWindow ぷろぱてぃ 712
actorList プロパティ actorList ぷろぱてぃ 51, 713
add (3D texture) メソッド add (3D texture) めそっど 243
add メソッド add めそっど 242
addAt メソッド addAt めそっど 243
addBackdrop メソッド addBackdrop めそっど 244
addCamera メソッド addCamera めそっど 245
addChild メソッド addChild めそっど 245
addModifier メソッド addModifier めそっど 246
addOverlay メソッド addOverlay めそっど 247
addProp メソッド addProp めそっど 248

addProxyTemplate()addProxyTemplate() 1246
addToWorld メソッド addToWorld めそっど 249
addVertex メソッド addVertex めそっど 249
adobeindexterm 1178
Alert() メソッド Alert() めそっど 251
alert() メソッド alert() めそっど 250
alertHook プロパティ alertHook ぷろぱてぃ 714
alignment プロパティ alignment ぷろぱてぃ 715
allowCustomCaching プロパティ allowCustomCaching ぷろぱてぃ 715
allowGraphicMenu プロパティ allowGraphicMenu ぷろぱてぃ 716
allowSaveLocal プロパティ allowSaveLocal ぷろぱてぃ 716
allowTransportControl プロパティ allowTransportControl ぷろぱてぃ 717
allowVolumeControl プロパティ allowVolumeControl ぷろぱてぃ 717
allowZooming プロパティ allowZooming ぷろぱてぃ 717
alphaThreshold プロパティ alphaThreshold ぷろぱてぃ 718
ambient プロパティ ambient ぷろぱてぃ 718
ambientColor プロパティ ambientColor ぷろぱてぃ 719
AmplifierFilter（オーディオ）136
ancestor プロパティ ancestor ぷろぱてぃ 719
and 論理演算子 and ろんりえんざんし 698
angle (3D) プロパティ angle (3D) ぷろぱてぃ 720
angle (DVD) プロパティ angle (DVD) ぷろぱてぃ 721
angleCount プロパティ angleCount ぷろぱてぃ 721
angularDamping（剛体）プロパティ angularDamping（ごうたい）ぷろぱてぃ 1253
angularDamping（ワールド）プロパティ angularDamping（わーるど）ぷろぱてぃ 1236
angularMomentum（剛体）プロパティ angularMomentum（ごうたい）ぷろぱてぃ 1254
angularVelocity（剛体）プロパティ angularVelocity（ごうたい）ぷろぱてぃ 1254
animationEnabled プロパティ animationEnabled ぷろぱてぃ 722
antiAlias プロパティ antiAlias ぷろぱてぃ 722
antiAliasingEnabled プロパティ antiAliasingEnabled ぷろぱてぃ 723
antiAliasingSupported プロパティ antiAliasingSupported ぷろぱてぃ 724
antiAliasThreshold プロパティ antiAliasThreshold ぷろぱてぃ 724
antiAliasType 724
API、定義 API、ていぎ 1
appearanceOptions プロパティ appearanceOptions ぷろぱてぃ 726
append メソッド append めそっど 252
applicationName プロパティ applicationName ぷろぱてぃ 726
applicationPath プロパティ applicationPath ぷろぱてぃ 727
applyAngularImpulse()（剛体）関数 applyAngularImpulse()（ごうたい）かんすう 1266
applyForce()（剛体）関数 applyForce()（ごうたい）かんすう 1265
applyLinearImpulse()（剛体）関数 applyLinearImpulse()（ごうたい）かんすう 1266
applyTorque()（剛体）関数 applyTorque()（ごうたい）かんすう 1266
appMinimize() メソッド appMinimize() めそっど 254
ASCII コード 275, 480
aspectRatio プロパティ aspectRatio ぷろぱてぃ 728
atan() メソッド atan() めそっど 254
attemptMoveTo()（剛体）関数 attemptMoveTo()（ごうたい）かんすう 1267
attenuation プロパティ attenuation ぷろぱてぃ 728
attributeName プロパティ attributeName ぷろぱてぃ 729
attributeValue プロパティ attributeValue ぷろぱてぃ 729
audio (DVD) プロパティ audio (DVD) ぷろぱてぃ 730
audio (RealMedia) プロパティ audio (RealMedia) ぷろぱてぃ 730
audio (Windows Media) プロパティaudio (Windows Media) ぷろぱてぃ 732
audio MP4Media/FLV プロパティ 730
audioChannelCount プロパティ audioChannelCount ぷろぱてぃ 732
audioFormat プロパティ audioFormat ぷろぱてぃ 733
audioLangExt プロパティ audioLangExt ぷろぱてぃ 733
audioSampleRate プロパティ audioSampleRate ぷろぱてぃ 734
audioStream プロパティ audioStream ぷろぱてぃ 734
audioStreamCount プロパティ audioStreamCount ぷろぱてぃ 735
auto プロパティ auto ぷろぱてぃ 735
autoblend プロパティ autoblend ぷろぱてぃ 735
autoCameraPosition プロパティ autoCameraPosition ぷろぱてぃ 736
autoMask プロパティ autoMask ぷろぱてぃ 737
autoTab プロパティ autoTab ぷろぱてぃ 738
axisAffinity プロパティ axisAffinity ぷろぱてぃ 1255
axisAngle プロパティ axisAngle ぷろぱてぃ 738
axisDrive（6dof）プロパティ axisDrive（6dof）ぷろぱてぃ 1293
axisMotion（6dof）プロパティ axisMotion（6dof）ぷろぱてぃ 1294

B

back プロパティ back ぷろぱてぃ 738
backColor プロパティ backColor ぷろぱてぃ 739
backdrop プロパティ backdrop ぷろぱてぃ 740
backgroundColor プロパティ backgroundColor ぷろぱてぃ 741
BACKSPACE 文字定数 BACKSPACE もじていすう 164
BandPassFilter（オーディオ）141
BandStopFilter（オーディオ）142
beep() メソッド beep() めそっど 255
beepOn プロパティ beepOn ぷろぱてぃ 742
beginRecording() メソッド beginRecording() めそっど 256
bevelDepth プロパティ bevelDepth ぷろぱてぃ 742

bevelType プロパティ bevelType ぷろぱてぃ 743
bgColor（ウィンドウ）プロパティ bgColor（うぃんどう）ぷろぱてぃ 744
bgColor（スプライト、3D メンバ）プロパティ bgColor（すぷらいと、3D めんば）ぷろぱてぃ 744
bias プロパティ bias ぷろぱてぃ 745
biNormalDrive（6dof）プロパティ biNormalDrive（6dof）ぷろぱてぃ 1295
biNormalMotion（6dof）プロパティ biNormalMotion（6dof）ぷろぱてぃ 1295
bitAnd() メソッド bitAnd() めそっど 257
bitDepth サウンドオブジェクトプロパティ 746
bitDepth ミクサプロパティ 745
bitmapSizes プロパティ bitmapSizes ぷろぱてぃ 745
bitNot() メソッド bitNot() めそっど 258
bitOr() メソッド bitOr() めそっど 259
bitRate プロパティ bitRate ぷろぱてぃ 747
bitsPerSample プロパティ bitsPerSample ぷろぱてぃ 748
bitXor() メソッド bitXor() めそっど 260
blend (3D) プロパティ blend (3D) ぷろぱてぃ 748
blend プロパティ blend ぷろぱてぃ 749
blendConstant プロパティ blendConstant ぷろぱてぃ 749
blendConstantList プロパティ blendConstantList ぷろぱてぃ 750
blendFactor プロパティ blendFactor ぷろぱてぃ 751
blendFunction プロパティ blendFunction ぷろぱてぃ 751
blendFunctionList プロパティ blendFunctionList ぷろぱてぃ 752
blendLevel プロパティ blendLevel ぷろぱてぃ 753
blendRange プロパティ blendRange ぷろぱてぃ 753
blendSource プロパティ blendSource ぷろぱてぃ 754
blendSourceList プロパティ blendSourceList ぷろぱてぃ 754
blendTime プロパティ blendTime ぷろぱてぃ 755
bone プロパティ bone ぷろぱてぃ 756
bonesPlayer（モディファイア）プロパティ bonesPlayer（もでぃふぁいあ）ぷろぱてぃ 756
border プロパティ border ぷろぱてぃ 758
bottom (3D) プロパティ bottom (3D) ぷろぱてぃ 759
bottom プロパティ bottom ぷろぱてぃ 758
bottomCap プロパティ bottomCap ぷろぱてぃ 759
bottomRadius プロパティ bottomRadius ぷろぱてぃ 760
bottomSpacing プロパティ bottomSpacing ぷろぱてぃ 760
boundary プロパティ boundary ぷろぱてぃ 761
boundingSphere プロパティ boundingSphere ぷろぱてぃ 761
boxDropShadow プロパティ boxDropShadow ぷろぱてぃ 762
boxType プロパティ boxType ぷろぱてぃ 762
breakLoop サウンドオブジェクトメソッド breakLoop さうんどおぶじぇくとめそっど 261
breakLoop() メソッド breakLoop() めそっど 260
brightness プロパティ brightness ぷろぱてぃ 763
broadcastProps プロパティ broadcastProps ぷろぱてぃ 763
browserName() メソッド browserName() めそっど 261
bufferSize プロパティ bufferSize ぷろぱてぃ 764
bufferSize ミクサプロパティ 764
build() メソッド build() めそっど 263
buttonCount プロパティ buttonCount ぷろぱてぃ 765
buttonsEnabled プロパティ buttonsEnabled ぷろぱてぃ 765
buttonStyle プロパティ buttonStyle ぷろぱてぃ 766
buttonType プロパティ buttonType ぷろぱてぃ 767
byteArray プロパティ 767
ByteArray メソッド ByteArray めそっど 262
ByteArray(str) メソッド ByteArray(str) めそっど 263
BytesRemaining プロパティ 768
bytesStreamed (3D) プロパティ bytesStreamed (3D) ぷろぱてぃ 769
bytesStreamed プロパティ bytesStreamed ぷろぱてぃ 768

C

C++ 用語 C++ ようご 46
cacheDocVerify() メソッド cacheDocVerify() めそっど 264
cacheSize() メソッド cacheSize() めそっど 265
call メソッド call めそっど 266
callAncestor メソッド callAncestor めそっど 268
callFrame() メソッド callFrame() めそっど 269
camera プロパティ camera ぷろぱてぃ 769
camera() メソッド camera() めそっど 270
cameraCount() メソッド cameraCount() めそっど 271
cameraPosition プロパティ cameraPosition ぷろぱてぃ 770
cameraRotation プロパティ cameraRotation ぷろぱてぃ 770
cancelIdleLoad() メソッド cancelIdleLoad() めそっど 271
case キーワード case きーわーど 214
castLib プロパティ castLib ぷろぱてぃ 771
castLib() メソッド castLib() めそっど 272
castLibNum プロパティ castLibNum ぷろぱてぃ 771
castMemberList プロパティ castMemberList ぷろぱてぃ 772
center プロパティ center ぷろぱてぃ 773
centerOfMass（剛体）プロパティ centerOfMass（ごうたい）ぷろぱてぃ 1255
centerRegPoint プロパティ centerRegPoint ぷろぱてぃ 773
centerStage プロパティ centerStage ぷろぱてぃ 774
CGI クエリ CGI くえり 362
changeArea プロパティ changeArea ぷろぱてぃ 775
channel ミクサプロパティ 776
channel()（サウンド）メソッド channel()（さうんど）めそっど 273
channel()（トップレベル）メソッド channel()（とっぷれべる）めそっど 272
channelCount サウンドオブジェクトプロパティ 777
channelCount プロパティ channelCount ぷろぱてぃ 777
channelCount ミクサプロパティ 776
chapter プロパティ chapter ぷろぱてぃ 777
chapterCount プロパティ chapterCount ぷろぱてぃ 778
chapterCount() メソッド chapterCount() めそっど 274

characterSet プロパティ characterSet ぷろぱてぃ 779
charPosToLoc() メソッド charPosToLoc() めそっど 274
chars() メソッド chars() めそっど 275
charSpacing プロパティ charSpacing ぷろぱてぃ 779
charToNum() メソッド charToNum() めそっど 275
checkMark プロパティ checkMark ぷろぱてぃ 780
child (3D) プロパティ child (3D) ぷろぱてぃ 781
child (XML) プロパティ child (XML) ぷろぱてぃ 781
chunkSize プロパティ chunkSize ぷろぱてぃ 782
clearAsObjects() メソッド clearAsObjects() めそっど 277
clearAtRender プロパティ clearAtRender ぷろぱてぃ 782
clearCache メソッド clearCache めそっど 277
clearError メソッド clearError めそっど 278
clearFrame() メソッド clearFrame() めそっど 279
clearGlobals() メソッド clearGlobals() めそっど 280
clearValue プロパティ clearValue ぷろぱてぃ 783
clickLoc プロパティ clickLoc ぷろぱてぃ 783
clickMode プロパティ clickMode ぷろぱてぃ 784
clickOn プロパティ clickOn ぷろぱてぃ 785
clone メソッド clone めそっど 281
cloneDeep メソッド cloneDeep めそっど 281
cloneModelFromCastmember メソッド cloneModelFromCastmember めそっど 282
cloneMotionFromCastmember メソッド cloneMotionFromCastmember めそっど 283
close() メソッド close() めそっど 283
closed プロパティ closed ぷろぱてぃ 786
closedCaptions プロパティ closedCaptions ぷろぱてぃ 786
closeFile() メソッド closeFile() めそっど 284
closeXlib メソッド closeXlib めそっど 284
collision (modifier) プロパティ collision (modifier) ぷろぱてぃ 787
collisionData プロパティ collisionData ぷろぱてぃ 788
collisionNormal プロパティ collisionNormal ぷろぱてぃ 789
color (fog) プロパティ color (fog) ぷろぱてぃ 791
color (light) プロパティ color (light) ぷろぱてぃ 792
color() プロパティ color() ぷろぱてぃ 790
color() メソッド color() めそっど 285
colorBufferDepth プロパティ colorBufferDepth ぷろぱてぃ 792
colorDepth プロパティ colorDepth ぷろぱてぃ 793
colorList プロパティ colorList ぷろぱてぃ 794
colorRange プロパティ colorRange ぷろぱてぃ 795
colors プロパティ colors ぷろぱてぃ 795
colorSteps プロパティ colorSteps ぷろぱてぃ 796
columnscale（地形）プロパティ columnscale（ちけい）ぷろぱてぃ 1289
commandDown プロパティ commandDown ぷろぱてぃ 797
comments プロパティ comments ぷろぱてぃ 798
compress() メソッド compress() めそっど 286
compressed プロパティ compressed ぷろぱてぃ 798
connectionStatus プロパティ 799
constrainH() メソッド constrainH() めそっど 286
constraint プロパティ constraint ぷろぱてぃ 800
ConstraintDesc（制限）関数 ConstraintDesc（せいげん）かんすう 1267
constraintType（6dof）プロパティ constraintType（6dof）ぷろぱてぃ 1296
constraintType（制限）プロパティ constraintType（せいげん）ぷろぱてぃ 1273
constrainV() メソッド constrainV() めそっど 287
contactTolerance（剛体）プロパティ contactTolerance（ごうたい）ぷろぱてぃ 1256
contactTolerance（地形）プロパティ contactTolerance（ちけい）ぷろぱてぃ 1290
contactTolerance（ワールド）プロパティ contactTolerance（わーるど）ぷろぱてぃ 1236
contains 演算子 contains えんざんし 698
controlDown プロパティ controlDown ぷろぱてぃ 801
controller プロパティ controller ぷろぱてぃ 802
copyPixels() メソッド copyPixels() めそっど 288
copyrightInfo (SWA) プロパティ copyrightInfo (SWA) ぷろぱてぃ 803
copyrightInfo（ムービー）プロパティ copyrightInfo（むーびー）ぷろぱてぃ 803
copyToClipBoard() メソッド copyToClipBoard() めそっど 290
cos() メソッド cos() めそっど 291
count (3D) プロパティ count (3D) ぷろぱてぃ 804, 806
count プロパティ count ぷろぱてぃ 804, 805
count() メソッド count() めそっど 291
CPU、バックグラウンドイベントの処理 CPU、ばっくぐらうんどいべんとのしょり 806
cpuHogTicks プロパティ cpuHogTicks ぷろぱてぃ 806
creaseAngle プロパティ creaseAngle ぷろぱてぃ 806
creases プロパティ creases ぷろぱてぃ 807
createAngularJoint()（制限）関数 createAngularJoint()（せいげん）かんすう 1268
createD6joint（6dof）メソッド createD6joint（6dof）めそっど 1293
createFile() メソッド createFile() めそっど 292
createLinearJoint()（制限）関数 createLinearJoint()（せいげん）かんすう 1269
createMask() メソッド createMask() めそっど 293
createMatte() メソッド createMatte() めそっど 293
createProxyTemplate()createProxyTemplate() 1250
createRigidBody()createRigidBody() 1246
createRigidBodyFromProxy()createRigidBodyFromProxy() 1248
createSoundObject メソッド createSoundObject めそっど 294
createSpring()（制限）関数 createSpring()（せいげん）かんすう 1270
creationDate プロパティ creationDate ぷろぱてぃ 808
crop プロパティ crop ぷろぱてぃ 808
crop() (メンバコマンド) メソッド crop() (めんばこまんど) めそっど 296

crop() メソッド crop() めそっど 295
cross メソッド cross めそっど 296
crossProduct() メソッド crossProduct() めそっど 297
cuePointNames プロパティ cuePointNames ぷろぱてぃ 809
cuePointTimes プロパティ cuePointTimes ぷろぱてぃ 809
currentLoopState プロパティ currentLoopState ぷろぱてぃ 810
currentSpriteNum プロパティ currentSpriteNum ぷろぱてぃ 810
currentTime (3D) プロパティ currentTime (3D) ぷろぱてぃ 811
currentTime (DVD) プロパティ currentTime (DVD) ぷろぱてぃ 812
currentTime (RealMedia) プロパティ currentTime (RealMedia) ぷろぱてぃ 813
currentTime （スプライト）プロパティ currentTime （すぷらいと）ぷろぱてぃ 815
currentTime MP4Media/FLV プロパティ 813
currentTime サウンドオブジェクトプロパティ 815
cursor プロパティ cursor ぷろぱてぃ 816
cursor() メソッド cursor() めそっど 298
cursorSize プロパティ cursorSize ぷろぱてぃ 819
curve プロパティ curve ぷろぱてぃ 820

D

damping（制限）プロパティ damping（せいげん）ぷろぱてぃ 1274
damping（スプリング）プロパティ damping（すぷりんぐ）ぷろぱてぃ 1277
date() （システム）メソッド date() （しすてむ）めそっど 302
date() （フォーマット）メソッド date() （ふぉーまっと）めそっど 301
deactivateApplication ハンドラ deactivateApplication はんどら 175
debug プロパティ debug ぷろぱてぃ 821
debugPlaybackEnabled プロパティ debugPlaybackEnabled ぷろぱてぃ 821
decayMode プロパティ decayMode ぷろぱてぃ 822
defaultRect プロパティ defaultRect ぷろぱてぃ 823
defaultRectMode プロパティ defaultRectMode ぷろぱてぃ 823
delay() メソッド delay() めそっど 303
Delete キー（Macintosh）Delete きー（Macintosh）164
delete() メソッド delete() めそっど 304
deleteAt メソッド deleteAt めそっど 306
deleteCamera メソッド deleteCamera めそっど 307
deleteConstraint()（制限）関数 deleteConstraint()（せいげん）かんすう 1270
deleteFile() メソッド deleteFile() めそっど 305
deleteFrame() メソッド deleteFrame() めそっど 307
deleteGroup メソッド deleteGroup めそっど 308
deleteLight メソッド deleteLight めそっど 309
deleteModel メソッド deleteModel めそっど 309
deleteModelResource メソッド deleteModelResource めそっど 309
deleteMotion メソッド deleteMotion めそっど 310
deleteOne メソッド deleteOne めそっど 311
deleteProp メソッド deleteProp めそっど 311
deleteRigidBody() 1251
deleteShader メソッド deleteShader めそっど 312
deleteSoundObject メソッド deleteSoundObject めそっど 312
deleteSpring()（制限）関数 deleteSpring()（せいげん）かんすう 1271
deleteTexture メソッド deleteTexture めそっど 313
deleteVertex() メソッド deleteVertex() めそっど 314
density プロパティ density ぷろぱてぃ 824
depth (3D) プロパティ depth (3D) ぷろぱてぃ 825
depth（ビットマップ）プロパティ depth（びっとまっぷ）ぷろぱてぃ 826
depthBufferDepth プロパティ depthBufferDepth ぷろぱてぃ 827
deskTopRectList プロパティ deskTopRectList ぷろぱてぃ 827
destroy()（ワールド）関数 destroy()（わーるど）かんすう 1242
diffuse プロパティ diffuse ぷろぱてぃ 828
diffuseColor プロパティ diffuseColor ぷろぱてぃ 829
diffuseLightMap プロパティ diffuseLightMap ぷろぱてぃ 829
digitalVideoTimeScale プロパティ digitalVideoTimeScale ぷろぱてぃ 830
digitalVideoType プロパティ digitalVideoType ぷろぱてぃ 830
direction プロパティ direction ぷろぱてぃ 831
directionalColor プロパティ directionalColor ぷろぱてぃ 831
directionalPreset プロパティ directionalPreset ぷろぱてぃ 832
directToStage MP4Media/FLV プロパティ 834
directToStage プロパティ directToStage ぷろぱてぃ 833
disableCollision()（衝突）メソッド disableCollision()（しょうとつ）めそっど 1281
disableCollisionCallback()（衝突）メソッド disableCollisionCallback()（しょうとつ）めそっど 1282
disableImagingTransformation プロパティ disableImagingTransformation ぷろぱてぃ 833
displayFace プロパティ displayFace ぷろぱてぃ 835
displayMode プロパティ displayMode ぷろぱてぃ 836
displayOpen() メソッド displayOpen() めそっど 314
displayRealLogo プロパティ displayRealLogo ぷろぱてぃ 836
displaySave() メソッド displaySave() めそっど 315
displayTemplate プロパティ displayTemplate ぷろぱてぃ 837
DistortionFilter（オーディオ）134
distribution プロパティ distribution ぷろぱてぃ 838
dither プロパティ dither ぷろぱてぃ 839
do メソッド do めそっど 315
dockingEnabled プロパティ dockingEnabled ぷろぱてぃ 839
domain プロパティ domain ぷろぱてぃ 840
doneParsing() メソッド doneParsing() めそっど 316
dot() メソッド dot() めそっど 316
dotProduct() メソッド dotProduct() めそっど 317
doubleClick プロパティ doubleClick ぷろぱてぃ 841
downloadNetThing メソッド downloadNetThing めそっど 317

drag プロパティ drag ぷろぱてぃ 841
draw() メソッド draw() めそっど 318
drawRect プロパティ drawRect ぷろぱてぃ 842
driveAngularVelocity（6dof）プロパティ driveAngularVelocity（6dof）ぷろぱてぃ 1296
driveLinearVelocity（6dof）プロパティ driveLinearVelocity（6dof）ぷろぱてぃ 1296
driveOrientation（6dof）プロパティ driveOrientation（6dof）ぷろぱてぃ 1297
drivePosition（6dof）プロパティ drivePosition（6dof）ぷろぱてぃ 1297
dropShadow プロパティ dropShadow ぷろぱてぃ 843
duplicate()（画像）メソッド duplicate()（がぞう）めそっど 319
duplicate()（メンバ）メソッド duplicate()（めんば）めそっど 321
duplicate()（リスト関数）メソッド duplicate()（りすとかんすう）めそっど 320
duplicateFrame() メソッド duplicateFrame() めそっど 322
duration (3D) プロパティ duration (3D) ぷろぱてぃ 843
duration (DVD) プロパティ duration (DVD) ぷろぱてぃ 844
duration（メンバ）プロパティ duration（めんば）ぷろぱてぃ 844
duration MP4Media/FLV プロパティ 845
duration（RealMedia）プロパティ duration（RealMedia）ぷろぱてぃ 845
DVD オブジェクト DVD おぶじぇくと 116
DVDeventNotification ハンドラ DVDeventNotification はんどら 176

E

e 対数関数 e たいすうかんすう 428
EchoFilter（オーディオ）132
editable プロパティ editable ぷろぱてぃ 846
editShortCutsEnabled プロパティ editShortCutsEnabled ぷろぱてぃ 847
elapsedTime サウンドオブジェクトプロパティ 849
elapsedTime プロパティ elapsedTime ぷろぱてぃ 848
elapsedTime ミクサプロパティ 848
emissive プロパティ emissive ぷろぱてぃ 849
emitter プロパティ emitter ぷろぱてぃ 849
EMPTY 定数 EMPTY ていすう 165
EMPTY 文字定数 EMPTY もじていすう 165
emulateMultibuttonMouse プロパティ emulateMultibuttonMouse ぷろぱてぃ 851
enableCollision()（衝突）メソッド enableCollision()（しょうとつ）めそっど 1283
enableCollisionCallback()（衝突）メソッド enableCollisionCallback()（しょうとつ）めそっど 1283
enabled (collision) プロパティ enabled (collision) ぷろぱてぃ 852
enabled (fog) プロパティ enabled (fog) ぷろぱてぃ 853
enabled (sds) プロパティ enabled (sds) ぷろぱてぃ 853
enabled プロパティ enabled ぷろぱてぃ 851
enabled（フィルタ）プロパティ 852
enableFlashLingo プロパティ enableFlashLingo ぷろぱてぃ 854
enableHotSpot メソッド enableHotSpot めそっど 322
enableSoundTrack メソッド enableSoundTrack めそっど 323
end case キーワード end case きーわーど 217
end キーワード end きーわーど 216
endAngle プロパティ endAngle ぷろぱてぃ 854
endColor プロパティ endColor ぷろぱてぃ 855
endFrame プロパティ endFrame ぷろぱてぃ 856
endian プロパティ 856
endRecording() メソッド endRecording() めそっど 323
endTime サウンドオブジェクトプロパティ 858
endTime プロパティ endTime ぷろぱてぃ 857
ENTER 文字定数 ENTER もじていすう 165
Enter キー Enter きー 165
EnvelopeFilter（オーディオ）136
environmentPropList プロパティ environmentPropList ぷろぱてぃ 858
erase() メソッド erase() めそっど 324
error プロパティ error ぷろぱてぃ 859
error() メソッド error() めそっど 325
eventPassMode プロパティ eventPassMode ぷろぱてぃ 860
exit repeat キーワード exit repeat きーわーど 218
exit キーワード exit きーわーど 217
exitLock プロパティ exitLock ぷろぱてぃ 861
externalEvent() メソッド externalEvent() めそっど 326
externalParamCount プロパティ externalParamCount ぷろぱてぃ 862
externalParamName() メソッド externalParamName() めそっど 328
externalParamValue() メソッド externalParamValue() めそっど 329
extractAlpha() メソッド extractAlpha() めそっど 331
extrude3D メソッド extrude3D めそっど 327

F

F4VF4V 122
face プロパティ face ぷろぱてぃ 863
face[ ] プロパティ 864
fadeIn() メソッド fadeIn() めそっど 331
FadeInFilter（オーディオ）138
fadeOut() メソッド fadeOut() めそっど 332
FadeOutFilter（オーディオ）138
fadeTo() メソッド fadeTo() めそっど 333
FadeToFilter（オーディオ）139
FALSE 論理定数 FALSE ろんりていすう 166
FALSE キーワード 20
far (fog) プロパティ far (fog) ぷろぱてぃ 864
fieldOfView (3D) プロパティ fieldOfView (3D) ぷろぱてぃ 865
fieldOfView プロパティ fieldOfView ぷろぱてぃ 865
fileFreeSize プロパティ fileFreeSize ぷろぱてぃ 866

Fileio オブジェクト Fileio おぶじぇくと 148
fileName（キャスト）プロパティ fileName（きゃすと）ぷろぱてぃ 866
fileName（メンバ）プロパティ fileName（めんば）ぷろぱてぃ 867
fileName プロパティ fileName ぷろぱてぃ 868
fileName() メソッド fileName() めそっど 333
FileOpen() メソッド FileOpen() めそっど 334
fileSave() メソッド fileSave() めそっど 335
fileSize プロパティ fileSize ぷろぱてぃ 869
fileVersion プロパティ fileVersion ぷろぱてぃ 870
fill() メソッド fill() めそっど 336
fillColor プロパティ fillColor ぷろぱてぃ 870
fillCycles プロパティ fillCycles ぷろぱてぃ 871
fillDirection プロパティ fillDirection ぷろぱてぃ 872
filled プロパティ filled ぷろぱてぃ 873
fillMode プロパティ fillMode ぷろぱてぃ 873
fillOffset プロパティ fillOffset ぷろぱてぃ 874
fillScale プロパティ fillScale ぷろぱてぃ 874
filter() メソッド filter() めそっど 337
filterList サウンドオブジェクトプロパティ 876
filterlist プロパティ filterlist ぷろぱてぃ 875
filterList ミクサプロパティ 875
findEmpty() メソッド findEmpty() めそっど 338
findLabel() メソッド findLabel() めそっど 338
findPos メソッド findPos めそっど 339
findPosNear メソッド findPosNear めそっど 340
finishIdleLoad() メソッド finishIdleLoad() めそっど 340
firstIndent プロパティ firstIndent ぷろぱてぃ 876
fixedLineSpace プロパティ fixedLineSpace ぷろぱてぃ 877
fixedRate プロパティ fixedRate ぷろぱてぃ 877
fixStageSize プロパティ fixStageSize ぷろぱてぃ 878
flags（スプリング）プロパティ flags（すぷりんぐ）ぷろぱてぃ 1277
FlangeFilter（オーディオ）133
Flash Asset Xtra, endTellTarget コマンド Flash Asset Xtra, endTellTarget こまんど 639
Flash コンポーネントオブジェクト Flash こんぽーねんとおぶじぇくと 119
Flash ムービー
　プロパティを設定する 602
　変数を設定する 613
Flash ムービーオブジェクト Flash むーびーおぶじぇくと 120
flashRect プロパティ flashRect ぷろぱてぃ 879
flashToStage() メソッド flashToStage() めそっど 341
flat プロパティ flat ぷろぱてぃ 880
flipH プロパティ flipH ぷろぱてぃ 880
flipV プロパティ flipV ぷろぱてぃ 881
float() メソッド float() めそっど 342
floatP() メソッド floatP() めそっど 342
floatPrecision プロパティ floatPrecision ぷろぱてぃ 881
flushInputEvents() メソッド flushInputEvents() めそっど 343
FLVFLV 122
fog プロパティ fog ぷろぱてぃ 882
folder プロパティ folder ぷろぱてぃ 883
font プロパティ font ぷろぱてぃ 884
fontSize プロパティ fontSize ぷろぱてぃ 884
fontStyle プロパティ fontStyle ぷろぱてぃ 886
foreColor プロパティ foreColor ぷろぱてぃ 886
forget() (Timeout) メソッド forget() (Timeout) めそっど 344
forget()（ウィンドウ）メソッド forget()（うぃんどう）めそっど 344
frame プロパティ frame ぷろぱてぃ 887
frameCount プロパティ frameCount ぷろぱてぃ 888
frameLabel プロパティ frameLabel ぷろぱてぃ 889
framePalette プロパティ framePalette ぷろぱてぃ 889
frameRate (DVD) プロパティ frameRate (DVD) ぷろぱてぃ 891
frameRate MP4Media/FLV プロパティ 892
frameRate プロパティ frameRate ぷろぱてぃ 889
frameReady() (Movie) メソッド frameReady() (Movie) めそっど 346
frameScript プロパティ frameScript ぷろぱてぃ 891
frameSound1 プロパティ frameSound1 ぷろぱてぃ 893
frameSound2 プロパティ frameSound1 ぷろぱてぃ 893
frameStep() メソッド frameStep() めそっど 347
framesToHMS() メソッド framesToHMS() めそっど 345
frameTempo プロパティ frameTempo ぷろぱてぃ 894
frameTransition プロパティ frameTransition ぷろぱてぃ 894
freeBlock() メソッド freeBlock() めそっど 347
freeBytes() メソッド freeBytes() めそっど 348
friction（剛体）プロパティ friction（ごうたい）ぷろぱてぃ 1256
friction（ワールド）プロパティ friction（わーるど）ぷろぱてぃ 1237
front プロパティ front ぷろぱてぃ 895
frontWindow プロパティ frontWindow ぷろぱてぃ 895
FTP プロキシサーバー、値を設定する 532
fullScreen プロパティ fullScreen ぷろぱてぃ 896

G

generateNormals() メソッド generateNormals() めそっど 349
getaProp メソッド getaProp めそっど 349
getAt メソッド getAt めそっど 350
getBoneID プロパティ getBoneID ぷろぱてぃ 374, 897
getCharSet メソッド getCharSet めそっど 351
getCollisionCallbackDisabledPairs()（衝突）メソッド getCollisionCallbackDisabledPairs()（しょうとつ）めそっど 1284
getCollisionDisabledPairs（衝突）メソッド getCollisionDisabledPairs（しょうとつ）めそっど 1284
getConstraint()（制限）関数 getConstraint()（せいげん）かんすう 1271

GetConstraints()（制限）関数 GetConstraints()（せいげん）かんすう 1272
getError() (XML) メソッド getError() (XML) めそっど 354
getError() メソッド getError() めそっど 352
getErrorString() メソッド getErrorString() めそっど 354
getFinderInfo() メソッド getFinderInfo() めそっど 355
getFlashProperty() メソッド getFlashProperty() めそっど 356
getFrameLabel() メソッド getFrameLabel() めそっど 357
getHardwareInfo() メソッド getHardwareInfo() めそっど 358
getHotSpotRect() メソッド getHotSpotRect() めそっど 358
getInstalledCharSets メソッド getInstalledCharSets めそっど 359
GetItemPropListt() メソッド GetItemPropListt() めそっど 359
getLast() メソッド getLast() めそっど 360
getLatestNetID メソッド getLatestNetID めそっど 361
getLength() メソッド getLength() めそっど 361
getNetByteArray メソッド getNetByteArray めそっど 362
getNetText() メソッド getNetText() めそっど 362
getNormalized メソッド getNormalized めそっど 364
getNthFileNameInFolder() メソッド getNthFileNameInFolder() めそっど 364
getOne() メソッド getOne() めそっど 366
getOSDirectory() メソッド getOSDirectory() めそっど 367
getPixel() メソッド getPixel() めそっど 367
getPixels() メソッド 368
getPlayList() メソッド getPlayList() めそっど 369
getPos() メソッド getPos() めそっど 370
getPosition() メソッド getPosition() めそっど 370
getPref() メソッド getPref() めそっど 372
getProp() メソッド getProp() めそっど 373
getPropAt() メソッド getPropAt() めそっど 374, 378, 381, 382
getRendererServices() メソッド getRendererServices() めそっど 375
getRigidBodies() 1252
getRigidBody() 1251
getSimulationTime()（ワールド）関数 getSimulationTime()（わーるど）かんすう 1243
getSleepingBodies() 1252
getSoundObject() メソッド getSoundObject() めそっど 376
getSoundObjectList メソッド getSoundObjectList めそっど 377
getSpring()（制限）関数 getSpring()（せいげん）かんすう 1272
getSprings()（制限）関数 getSprings()（せいげん）かんすう 1273
getStreamStatus() メソッド getStreamStatus() めそっど 377
getSystemCharSet メソッド getSystemCharSet めそっど 378
getTerrain（地形）メソッド getTerrain（ちけい）めそっど 1289
getTerrains（地形）メソッド getTerrains（ちけい）めそっど 1289
getURL() メソッド getURL() めそっど 378
getVal() 379
getVariable() メソッド getVariable() めそっど 380
GetWidgetList() メソッド GetWidgetList() めそっど 381
GetWindowPropList() メソッド GetWindowPropList() めそっど 382
getWorldTransform() メソッド getWorldTransform() めそっど 382
global キーワード global きーわーど 219
globalAnchor（6dof）プロパティ globalAnchor（6dof）ぷろぱてぃ 1298
globals プロパティ globals ぷろぱてぃ 897
glossMap プロパティ glossMap ぷろぱてぃ 897
go() メソッド go() めそっど 383
goLoop() メソッド goLoop() めそっど 385
goNext() メソッド goNext() めそっど 385
goPrevious() メソッド goPrevious() めそっど 386
goToFrame() メソッド goToFrame() めそっど 387
gotoNetMovie メソッド gotoNetMovie めそっど 387
gotoNetPage メソッド gotoNetPage めそっど 388
gradientType プロパティ gradientType ぷろぱてぃ 899
gravity プロパティ gravity ぷろぱてぃ 898
gravity（ワールド）プロパティ gravity（わーるど）ぷろぱてぃ 1237
group のプロパティ group のぷろぱてぃ 899
group() メソッド group() めそっど 388

H

halt() メソッド halt() めそっど 389
handler() メソッド handler() めそっど 390
handlers() メソッド handlers() めそっど 390
height (3D) プロパティ height (3D) ぷろぱてぃ 901
height MP4Media/FLV プロパティ 901
height プロパティ height ぷろぱてぃ 900
heightscale（地形）プロパティ heightscale（ちけい）ぷろぱてぃ 1290
heightVertices プロパティ heightVertices ぷろぱてぃ 902
highlightPercentage プロパティ highlightPercentage ぷろぱてぃ 902
highlightStrength プロパティ highlightStrength ぷろぱてぃ 903
HighPassFilter（オーディオ）140
hilite （コマンド）メソッド hilite （こまんど）めそっど 391
hilite （メンバ）プロパティ hilite （めんば）ぷろぱてぃ 903
hilite メンバプロパティ 903
hinting プロパティ 904
hither プロパティ hither プロパティ 905
hitTest() メソッド hitTest() めそっど 391
HMStoFrames() メソッド HMStoFrames() めそっど 393
hold() メソッド hold() めそっど 393
hotSpot プロパティ hotSpot ぷろぱてぃ 905
hotSpotEnterCallback プロパティ hotSpotEnterCallback ぷろぱてぃ 905
hotSpotExitCallback プロパティ hotSpotExitCallback ぷろぱてぃ 907
HTML プロパティ HTML ぷろぱてぃ 907
HTTP プロキシサーバー、値を設定する 532
HTTP ヘッダ、最後に変更された日付 459

hyperlink プロパティ hyperlink ぷろぱてぃ 908
hyperlinkRange プロパティ hyperlinkRange ぷろぱてぃ 908
hyperlinks プロパティ hyperlinks ぷろぱてぃ 909
hyperlinkState プロパティ hyperlinkState ぷろぱてぃ 909

I

identity() メソッド identity() めそっど 395
idle ハンドラ idle はんどら 188
idleHandlerPeriod プロパティ idleHandlerPeriod ぷろぱてぃ 910
idleLoadDone() メソッド idleLoadDone() めそっど 396
idleLoadMode プロパティ idleLoadMode ぷろぱてぃ 911
idleLoadPeriod プロパティ idleLoadPeriod ぷろぱてぃ 912
idleLoadTag プロパティ idleLoadTag ぷろぱてぃ 912
idleReadChunkSize プロパティ idleReadChunkSize ぷろぱてぃ 913
if キーワード if きーわーど 220
if...then...else ステートメント 220, 477
ignoreWhiteSpace() メソッド ignoreWhiteSpace() めそっど 396
ilk (3D) メソッド ilk (3D) めそっど 399
ilk() メソッド ilk() めそっど 397
image (RealMedia) プロパティ image (RealMedia) ぷろぱてぃ 914
image （画像）プロパティ image （画像）ぷろぱてぃ 914
image （ウィンドウ）プロパティ image （ウィンドウ）ぷろぱてぃ 915
image MP4Media/FLV プロパティ 915
image() メソッド image() めそっど 400
imageCompression プロパティ imageCompression ぷろぱてぃ 916
imageEnabled プロパティ imageEnabled ぷろぱてぃ 916
imageQuality プロパティ imageQuality ぷろぱてぃ 917
immovable プロパティ immovable ぷろぱてぃ 919
importByteArrayInto() 394
importFileInto() メソッド importFileInto() めそっど 401
INF キーワード INF きーわーど 221
init()（ワールド）関数 init()（わーるど）かんすう 1243
initializeinitialize 403
ink プロパティ ink ぷろぱてぃ 919
inker （モディファイア）プロパティ inker （もでぃふぁいあ）ぷろぱてぃ 920
inlineImeEnabled プロパティ inlineImeEnabled ぷろぱてぃ 921
insertBackdrop メソッド insertBackdrop めそっど 403
insertFrame() メソッド insertFrame() めそっど 404
insertOverlay メソッド insertOverlay めそっど 405
inside() メソッド inside() めそっど 406
installMenu メソッド installMenu めそっど 406
integer() メソッド integer() めそっど 407
integerP() メソッド integerP() めそっど 408
interface() メソッド interface() めそっど 408
interpolate() メソッド interpolate() めそっど 409
interpolateTo() メソッド interpolateTo() めそっど 409
intersect() メソッド intersect() めそっど 410
interval プロパティ interval ぷろぱてぃ 922
inverse() メソッド inverse() めそっど 411
invert() メソッド invert() めそっど 411
invertMask プロパティ invertMask ぷろぱてぃ 922
isCharSetInstalled メソッド isCharSetInstalled めそっど 413
isInitialized（ワールド）プロパティ isInitialized（わーるど）ぷろぱてぃ 1238
isInWorld() メソッド isInWorld() めそっど 413
isPastCuePoint() メソッド isPastCuePoint() めそっど 414
isPinned（剛体）プロパティ isPinned（ごうたい）ぷろぱてぃ 1257
isPlayable プロパティ 923
isSaving サウンドオブジェクトプロパティ 924
isSaving ミクサプロパティ 923
isSleeping（剛体）プロパティ isSleeping（ごうたい）ぷろぱてぃ 1257
isVRMovie プロパティ isVRMovie ぷろぱてぃ 925
item... of キーワード item... of きーわーど 221
itemDelimiter プロパティ itemDelimiter ぷろぱてぃ 925

J

Java 用語 Java ようご 46
JavaScript
C++ および Java との比較 46
Lingo との比較 40
一般的なスクリプト項目の挿入 67
インスタンス変数 58
インスタンスメソッド 59
演算子 18, 683
大文字と小文字の区別 9
オブジェクトインスタンス 56
オブジェクト指向プログラミング 45, 54
オブジェクトの継承 57
カスタムクラス 55
キーワード 214
クラスの定義 60
クラス変数 59
クラスメソッド 60
繰り返しループ 24
グローバル変数 17
コメント 7, 68
コンストラクタ関数 56
順序、実行 21
条件一致、式 23
シンタックス 7
シンボル 14
数値 12
スクリプトの種類 4
ストリング 12

スペース 9
セミコロン 9
データ型 10
定数 13
ドットシンタックス 41
配列 37
プロトタイプオブジェクト 58, 61
プロパティ 11
用語集 5
リスト 30
論理条件、テスト 22

K

kerning プロパティ kerning ぷろぱてぃ 926
kerningThreshold プロパティ kerningThreshold ぷろぱてぃ 927
key プロパティ key ぷろぱてぃ 927
keyboardFocusSprite プロパティ keyboardFocusSprite ぷろぱてぃ 929
keyCode プロパティ keyCode ぷろぱてぃ 929
keyDown ハンドラ keyDown はんどら 189
keyDownScript プロパティ keyDownScript ぷろぱてぃ 930
keyframePlayer（モディファイア）プロパティ keyframePlayer（もでぃふぁいあ）ぷろぱてぃ 931
keyPressed() メソッド keyPressed() めそっど 415
keyUp ハンドラ keyUp はんどら 191
keyUpScript プロパティ keyUpScript ぷろぱてぃ 932

L

label() メソッド label() めそっど 417
labelList プロパティ labelList ぷろぱてぃ 933
last() メソッド last() めそっど 418
lastChannel プロパティ lastChannel ぷろぱてぃ 933
lastClick プロパティ lastClick ぷろぱてぃ 934
lastClick() メソッド lastClick() めそっど 419
lastError プロパティ lastError ぷろぱてぃ 934
lastEvent プロパティ lastEvent ぷろぱてぃ 935
lastEvent() メソッド lastEvent() めそっど 419
lastFrame プロパティ lastFrame ぷろぱてぃ 935
lastKey プロパティ lastKey ぷろぱてぃ 936
lastRoll プロパティ lastRoll ぷろぱてぃ 937
left (3D) プロパティ left (3D) ぷろぱてぃ 938
left プロパティ left ぷろぱてぃ 937
leftIndent プロパティ leftIndent ぷろぱてぃ 938
length (3D) プロパティ length (3D) ぷろぱてぃ 939
length バイト配列プロパティ 939
length() メソッド length() めそっど 420
lengthVertices プロパティ lengthVertices ぷろぱてぃ 940
level プロパティ level ぷろぱてぃ 940
lifetime プロパティ lifetime ぷろぱてぃ 941
light のプロパティ 941
light() メソッド light() めそっど 420
linearDamping（剛体）プロパティ linearDamping（ごうたい）ぷろぱてぃ 1258
lineardamping（ワールド）プロパティ lineardamping（わーるど）ぷろぱてぃ 1238
linearLimit（6dof）プロパティ linearLimit（6dof）ぷろぱてぃ 1298
linearMomentum（剛体）プロパティ linearMomentum（ごうたい）ぷろぱてぃ 1258
linearVelocity（剛体）プロパティ linearVelocity（ごうたい）ぷろぱてぃ 1259
lineColor プロパティ lineColor ぷろぱてぃ 942
lineCount プロパティ lineCount ぷろぱてぃ 942
lineDirection プロパティ lineDirection ぷろぱてぃ 942
lineHeight プロパティ lineHeight ぷろぱてぃ 943
lineHeight()（関数）メソッド lineHeight()（かんすう）めそっど 421
lineOffset プロパティ lineOffset ぷろぱてぃ 943
linePosToLocV() メソッド linePosToLocV() めそっど 421
lineSize プロパティ lineSize ぷろぱてぃ 944
Lingo
C++ および Java との比較 46
JavaScript との比較 40
Xtra 拡張 1014
一般的なスクリプト項目の挿入 67
演算子 18, 683
大文字と小文字の区別 9
オブジェクト指向プログラミング 45
キーワード 214
繰り返しループ 24
グローバル変数 16
コメント 7, 68
順序、実行 21
条件一致、式 23
シンタックス 7
シンボル 14
数値 12
スクリプト、種類 4
ストリング 12
スペース 9
データ型 10
定数 13
ドットシンタックス 41
プロパティ 11
用語集 5
リスト 30
論理条件、テスト 22
linkAs() メソッド linkAs() めそっど 422
linked プロパティ linked ぷろぱてぃ 945
list() メソッド list() めそっど 423

listP() メソッド listP() めそっど 423
loaded プロパティ loaded ぷろぱてぃ 945
loadFile() メソッド loadFile() めそっど 424
loadPolicyFile() メソッド loadPolicyFile() めそっど 424
loadProxyTemplate() 1253
loc (backdrop and overlay) プロパティloc (backdrop and overlay) ぷろぱてぃ 946
loc（モディファイア）プロパティ loc（もでぃふぁいあ）ぷろぱてぃ 949
localAnchorA（6dof）プロパティ localAnchorA（6dof）ぷろぱてぃ 1299
localAnchorB（6dof）プロパティ localAnchorB（6dof）ぷろぱてぃ 1299
localAxisA（6dof）プロパティ localAxisA（6dof）ぷろぱてぃ 1300
localAxisB（6dof）プロパティ localAxisB（6dof）ぷろぱてぃ 1301
localNormalA（6dof）プロパティ localNormalA（6dof）ぷろぱてぃ 1301
localNormalB（6dof）プロパティ localNormalB（6dof）ぷろぱてぃ 1302
locH プロパティ locH ぷろぱてぃ 946
lockTranslation プロパティ lockTranslation ぷろぱてぃ 947
locToCharPos() メソッド locToCharPos() めそっど 427
locV プロパティ locV ぷろぱてぃ 947
locVToLinePos() メソッド locVToLinePos() めそっど 427
locZ プロパティ locZ ぷろぱてぃ 948
log() メソッド log() めそっど 428
loop (3D) プロパティ loop (3D) ぷろぱてぃ 950
loop (emitter) プロパティ loop (emitter) ぷろぱてぃ 950
loop MP4Media/FLV プロパティ 951
loop（Flash プロパティ）プロパティ loop（Flash ぷろぱてぃ）ぷろぱてぃ 952
loop（メンバ）プロパティ loop（めんば）ぷろぱてぃ 951
loopBounds プロパティ loopBounds ぷろぱてぃ 953
loopCount サウンドオブジェクトプロパティ 955
loopCount プロパティ loopCount ぷろぱてぃ 953
loopEndTime サウンドオブジェクトプロパティ 956
loopEndTime プロパティ loopEndTime ぷろぱてぃ 955
loopMovie プロパティ loopMovie ぷろぱてぃ 953
loopsRemaining サウンドオブジェクトプロパティ 957
loopsRemaining プロパティ loopsRemaining ぷろぱてぃ 957
loopStartTime（サウンドオブジェクトプロパティ 958
loopStartTime プロパティ loopStartTime ぷろぱてぃ 958
LowPassFilter（オーディオ）140

M

magnitude プロパティ magnitude ぷろぱてぃ 959
makeList() メソッド makeList() めそっど 428
makeScriptedSprite() メソッド makeScriptedSprite() めそっど 429
makeSubList() メソッド makeSubList() めそっど 430
map (3D) メソッド map (3D) めそっど 431
map() メソッド map() めそっど 430
mapMemberToStage() メソッド mapMemberToStage() めそっど 431
mapStageToMember() メソッド mapStageToMember() めそっど 432
margin プロパティ margin ぷろぱてぃ 959
marker() メソッド marker() めそっど 433
markerList プロパティ markerList ぷろぱてぃ 960
mask プロパティ mask ぷろぱてぃ 960
mass（剛体）プロパティ mass（ごうたい）ぷろぱてぃ 1259
matrixAddition() 434
matrixMultiply() 434
matrixMultiplyScalar() 435
matrixTranspose 436
max() メソッド max() めそっど 436
maximize() メソッド maximize() めそっど 437
maxInteger プロパティ 961
maxSpeed プロパティ maxSpeed ぷろぱてぃ 962
mci 438
mci メソッド mci めそっど 438
me 変数 49
media プロパティ media ぷろぱてぃ 962
mediaReady プロパティ mediaReady ぷろぱてぃ 963
mediaStatus (DVD) プロパティ mediaStatus (DVD) ぷろぱてぃ 963
mediaStatus MP4Media/FLV プロパティ 964
mediaStatus プロパティ mediaStatus ぷろぱてぃ 965
mediaXtraList プロパティ mediaXtraList ぷろぱてぃ 966
member（キャスト）プロパティ member（きゃすと）ぷろぱてぃ 967
member（サウンドチャネル）プロパティ member（さうんどちゃねる）ぷろぱてぃ 968
member（スプライト）プロパティ member（すぷらいと）ぷろぱてぃ 969
member（ムービー）プロパティ member（むーびー）ぷろぱてぃ 967
member サウンドオブジェクトプロパティ 968
member プロパティ member ぷろぱてぃ 966
member() メソッド member() めそっど 438
memorySize プロパティ memorySize ぷろぱてぃ 970
menu キーワード menu きーわーど 224
mergeDisplayTemplate() メソッド mergeDisplayTemplate() めそっど 439
mergeProps() メソッド mergeProps() めそっど 440
mesh（プロパティ）メソッド mesh（ぷろぱてぃ）めそっど 440
meshDeform（モディファイア）プロパティ meshDeform（もでぃふぁいあ）ぷろぱてぃ 971
meshDeform（モディファイア）メソッド meshDeform（もでぃふぁいあ）めそっど 441
milliseconds プロパティ milliseconds ぷろぱてぃ 971

MIME ファイル 388, 460
min メソッド min めそっど 442
minimize() メソッド minimize() めそっど 443
minSpeed プロパティ minSpeed ぷろぱてぃ 973
missingFonts プロパティ missingFonts ぷろぱてぃ 973
mixer プロパティ 974
mod (剰余) 演算子 700
mode (collision) プロパティ mode (collision) ぷろぱてぃ 975
mode (emitter) プロパティ mode (emitter) ぷろぱてぃ 974
model プロパティ model ぷろぱてぃ 975
model メソッド model めそっど 444
model（剛体）プロパティ model（ごうたい）ぷろぱてぃ 1260
modelA プロパティ modelA ぷろぱてぃ 976
modelB プロパティ modelB ぷろぱてぃ 976
modelResource プロパティ modelResource ぷろぱてぃ 977
modelResource メソッド modelResource めそっど 444
modelsUnderLoc メソッド modelsUnderLoc めそっど 445
modelsUnderRay メソッド modelsUnderRay めそっど 446
modelUnderLoc メソッド modelUnderLoc めそっど 448
modified プロパティ modified ぷろぱてぃ 978
modifiedBy プロパティ modifiedBy ぷろぱてぃ 978
modifiedDate プロパティ modifiedDate ぷろぱてぃ 979
modifier プロパティ modifier ぷろぱてぃ 980
modifier[] プロパティ 980
modifiers プロパティ modifiers ぷろぱてぃ 981
mostRecentCuePoint サウンドオブジェクトプロパティ 982
mostRecentCuePoint プロパティ mostRecentCuePoint ぷろぱてぃ 981
motion プロパティ motion ぷろぱてぃ 982
motion() メソッド motion() めそっど 448
motionQuality プロパティ motionQuality ぷろぱてぃ 983
mouseChar プロパティ mouseChar ぷろぱてぃ 983
mouseDown プロパティ mouseDown ぷろぱてぃ 984
mouseDownScript プロパティ mouseDownScript ぷろぱてぃ 985
mouseH プロパティ mouseH ぷろぱてぃ 986
mouseItem プロパティ mouseItem ぷろぱてぃ 987
mouseLevel プロパティ mouseLevel ぷろぱてぃ 988
mouseLine プロパティ mouseLine ぷろぱてぃ 988
mouseLoc プロパティ mouseLoc ぷろぱてぃ 989
mouseMember プロパティ mouseMember ぷろぱてぃ 990
mouseOverButton プロパティ mouseOverButton ぷろぱてぃ 991
mouseUp プロパティ mouseUp ぷろぱてぃ 991
mouseUpScript プロパティ mouseUpScript ぷろぱてぃ 992
mouseV プロパティ mouseV ぷろぱてぃ 993
mouseWord プロパティ mouseWord ぷろぱてぃ 994
move() メソッド move() めそっど 449
moveableSprite プロパティ moveableSprite ぷろぱてぃ 995
moveTo メソッド moveTo めそっど 450
moveToBack() メソッド moveToBack() めそっど 450
moveToFront() メソッド moveToFront() めそっど 451
moveVertex() メソッド moveVertex() めそっど 451
moveVertexHandle() メソッド moveVertexHandle() めそっど 452
movie プロパティ movie ぷろぱてぃ 996
movieRate プロパティ movieRate ぷろぱてぃ 1047
movieTime プロパティ movieTime ぷろぱてぃ 813
MP4MediaMP4Media 122
multiply() メソッド multiply() めそっど 453
multiSound プロパティ multiSound ぷろぱてぃ 996
mute サウンドオブジェクトメソッド mute さうんどおぶじぇくとめそっど 454
mute ミクサメソッド mute みくさめそっど 454

N

name (3D) プロパティ name (3D) ぷろぱてぃ 997
name (timeout) プロパティ name (timeout) ぷろぱてぃ 1001
name (XML) プロパティ name (XML) ぷろぱてぃ 1001
name （スプライト）プロパティ name （すぷらいと）ぷろぱてぃ 999
name（スプライトチャネル）プロパティ name（すぷらいとちゃねる）ぷろぱてぃ 1000
name（メニュー項目プロパティ）プロパティ name（めにゅーこうもくぷろぱてぃ）ぷろぱてぃ 998
name （メニュープロパティ）プロパティ name （めにゅーぷろぱてぃ）ぷろぱてぃ 997
name サウンドオブジェクトプロパティ 999
name プロパティ name ぷろぱてぃ 996
name ミクサプロパティ 998
name（6dof）プロパティ name（6dof）ぷろぱてぃ 1303
name（剛体）プロパティ name（ごうたい）ぷろぱてぃ 1260
name（制限）プロパティ name（せいげん）ぷろぱてぃ 1274
name（地形）プロパティ name（ちけい）ぷろぱてぃ 1290
name（スプリング）プロパティ name（すぷりんぐ）ぷろぱてぃ 1278
NAN キーワード NAN きーわーど 225
near (fog) プロパティ near (fog) ぷろぱてぃ 1002
nearFiltering プロパティ nearFiltering ぷろぱてぃ 1002
neighbor メソッド neighbor めそっど 454
netAbort メソッド netAbort めそっど 455
netByteArrayResult メソッド netByteArrayResult めそっど 456
netDone() メソッド netDone() めそっど 456
netError() メソッド netError() めそっど 457
netLastModDate() メソッド netLastModDate() めそっど 459
NetLingo オブジェクト NetLingo おぶじぇくと 150
netMIME() メソッド netMIME() めそっど 460
netPresent プロパティ netPresent ぷろぱてぃ 1003
netStatus メソッド netStatus めそっど 461
netTextResult() メソッド netTextResult() めそっど 462
netThrottleTicks プロパティ netThrottleTicks ぷろぱてぃ 1004
new() メソッド new() めそっど 463
newCamera メソッド newCamera めそっど 466

newColorRatio 466
newCurve() メソッド newCurve() めそっど 466
newGroup メソッド newGroup めそっど 468
newLight メソッド newLight めそっど 468
newMatrix() 469
newMember() メソッド newMember() めそっど 469
newMesh メソッド newMesh めそっど 470
newModel メソッド newModel めそっど 471
newModelResource メソッド newModelResource めそっど 472
newMotion() メソッド newMotion() めそっど 473
newObject() メソッド newObject() めそっど 474
newShader メソッド newShader めそっど 474
newTexture メソッド newTexture めそっど 475
next キーワード 226
node プロパティ node ぷろぱてぃ 1004
nodeEnterCallback プロパティ nodeEnterCallback ぷろぱてぃ 1005
nodeExitCallback プロパティ nodeExitCallback ぷろぱてぃ 1005
nodeType プロパティ nodeType ぷろぱてぃ 1006
normalDrive（6dof）プロパティ normalDrive（6dof）ぷろぱてぃ 1303
normalize メソッド normalize めそっど 476
normalList プロパティ normalList ぷろぱてぃ 1007
normalMotion（6dof）プロパティ normalMotion（6dof）ぷろぱてぃ 1304
normals 349, 440, 476
normals プロパティ normals ぷろぱてぃ 1007
not 論理演算子 701
nothing メソッド nothing めそっど 477
nudge メソッド nudge めそっど 478
number（行）プロパティ number（ぎょう）ぷろぱてぃ 1010
number（項目）プロパティ number（こうもく）ぷろぱてぃ 1009
number（単語）プロパティ number（たんご）ぷろぱてぃ 1013
number（文字）プロパティ number（もじ）ぷろぱてぃ 1008
number（キャスト）プロパティ number（きゃすと）ぷろぱてぃ 1008
number（システム）プロパティ number（しすてむ）ぷろぱてぃ 1013
number（スプライトチャネル）プロパティ number（すぷらいとちゃねる）ぷろぱてぃ 1012
number（メニュー）プロパティ number（めにゅー）ぷろぱてぃ 1011
number（メニュー項目）プロパティ number（めにゅーこうもく）ぷろぱてぃ 1012
number（メンバ）プロパティ number（めんば）ぷろぱてぃ 1010
number of members プロパティ number of members ぷろぱてぃ 1013
number of xtras プロパティ number of xtras ぷろぱてぃ 1014
numBuffersToPreload プロパティ 1014
numChannels プロパティ numChannels ぷろぱてぃ 1015
numColumns() 479
numParticles プロパティ numParticles ぷろぱてぃ 1015
numRows() 480
numSegments プロパティ numSegments ぷろぱてぃ 1016
numToChar() メソッド numToChar() めそっど 480

O

obeyScoreRotation プロパティ obeyScoreRotation ぷろぱてぃ 1016
objectA（6dof）プロパティ objectA（6dof）ぷろぱてぃ 1304
objectA（制限）プロパティ objectA（せいげん）ぷろぱてぃ 1275
objectA（スプリング）プロパティ objectA（すぷりんぐ）ぷろぱてぃ 1278
objectB（6dof）プロパティ objectB（6dof）ぷろぱてぃ 1305
objectB（制限）プロパティ objectB（せいげん）ぷろぱてぃ 1275
objectB（スプリング）プロパティ objectB（すぷりんぐ）ぷろぱてぃ 1279
objectP() メソッド objectP() めそっど 481
offset() (string function) メソッド offset() (string function) めそっど 482
offset()（矩形関数）メソッド offset()（くけいかんすう）めそっど 483
on activateWindow ハンドラ on activateWindow はんどら 172
on beginSprite ハンドラ on beginSprite はんどら 172
on closeWindow ハンドラ on closeWindow はんどら 173
on cuePassed ハンドラ on cuePassed はんどら 174
on deactivateWindow ハンドラ on deactivateWindow はんどら 176
on endSprite ハンドラ on endSprite はんどら 179
on enterFrame ハンドラ on enterFrame はんどら 180
on EvalScript ハンドラ on EvalScript はんどら 181
on exitFrame ハンドラ on exitFrame はんどら 183
on getBehaviorDescription ハンドラ on getBehaviorDescription はんどら 184
on getBehaviorTooltip ハンドラ on getBehaviorTooltip はんどら 185
on getPropertyDescriptionList ハンドラ on getPropertyDescriptionList はんどら 186
on hyperlinkClicked ハンドラ on hyperlinkClicked はんどら 187
on idle ハンドラ on idle はんどら 188
on isOKToAttach ハンドラ on isOKToAttach はんどら 188
on keyDown ハンドラ on keyDown はんどら 189
on keyUp ハンドラ on keyUp はんどら 191
on mouseDown ハンドラ on mouseDown はんどら 192
on mouseEnter ハンドラ on mouseEnter はんどら 193
on mouseLeave ハンドラ on mouseLeave はんどら 194
on mouseUp ハンドラ on mouseUp はんどら 195
on mouseUpOutside ハンドラ on mouseUpOutside はんどら 197
on mouseWithin ハンドラ on mouseWithin はんどら 197
on moveWindow ハンドラ 198
on new ハンドラ、作成する 48
on openWindow ハンドラ on openWindow はんどら 199
on prepareFrame ハンドラ on prepareFrame はんどら 199

on prepareMovie ハンドラ on prepareMovie はんどら 200
on resizeWindow ハンドラ on resizeWindow はんどら 201
on rightMouseDown ハンドラ on rightMouseDown はんどら 202
on rightMouseUp ハンドラ on rightMouseUp はんどら 202
on runPropertyDialog ハンドラ on runPropertyDialog はんどら 202
on savedLocal ハンドラ on savedLocal はんどら 204
on sendXML ハンドラ on sendXML はんどら 205
on startMovie ハンドラ on startMovie はんどら 206
on stepFrame ハンドラ on stepFrame はんどら 207
on stopMovie ハンドラ on stopMovie はんどら 208
on streamStatus ハンドラ on streamStatus はんどら 209
on timeOut ハンドラ on timeOut はんどら 210
on zoomWindow ハンドラ on zoomWindow はんどら 213
on キーワード 227
open()（ウィンドウ）メソッド open()（うぃんどう）めそっど 485
open()（プレーヤ）メソッド open()（ぷれーや）めそっど 484
openFile() メソッド openFile() めそっど 486
openXlib メソッド openXlib めそっど 486
optionDown プロパティ optionDown ぷろぱてぃ 1017
or 論理演算子 702
organizationName プロパティ organizationName ぷろぱてぃ 1018
orientation（剛体）プロパティ orientation（ごうたい）ぷろぱてぃ 1260
orientation（地形）プロパティ orientation（ちけい）ぷろぱてぃ 1291
originalFont プロパティ originalFont ぷろぱてぃ 1018
originH プロパティ originH ぷろぱてぃ 1019
originMode プロパティ originMode ぷろぱてぃ 1020
originPoint プロパティ originPoint ぷろぱてぃ 1020
originV プロパティ originV ぷろぱてぃ 1021
orthoHeight プロパティ orthoHeight ぷろぱてぃ 1023
otherwise キーワード 227
overlay プロパティ overlay ぷろぱてぃ 1023

P

pageHeight プロパティ pageHeight ぷろぱてぃ 1024
palette のプロパティ 1024
paletteMapping プロパティ paletteMapping ぷろぱてぃ 1025
paletteRef プロパティ paletteRef ぷろぱてぃ 1025
pan（QTVR プロパティ）プロパティ pan（QTVR ぷろぱてぃ）ぷろぱてぃ 1027
pan プロパティ pan ぷろぱてぃ 1026
panMatrix サウンドオブジェクトプロパティ 1029
panMatrix ミクサプロパティ 1027
paragraph プロパティ paragraph ぷろぱてぃ 1030
param() メソッド param() めそっど 487
paramCount() メソッド paramCount() めそっど 488
ParamEqFilter（オーディオ）143
parent プロパティ parent ぷろぱてぃ 1030
parseByteArray メソッド parseByteArray めそっど 488
parseString XML Xtra メソッド parseString XML Xtra めそっど 490
parseString() メソッド parseString() めそっど 489
parseURL XML Xtra メソッド parseURL XML Xtra めそっど 492
parseURL() メソッド parseURL() めそっど 490
pass メソッド pass めそっど 492
password プロパティ password ぷろぱてぃ 1031
pasteClipBoardInto() メソッド pasteClipBoardInto() めそっど 494
path (3D) プロパティ path (3D) ぷろぱてぃ 1033
path（ムービー）プロパティ path（むーびー）ぷろぱてぃ 1032
pathName（Flash メンバ）プロパティ pathName（Flash めんば）ぷろぱてぃ 1033
pathStrength プロパティ pathStrength ぷろぱてぃ 1034
pattern プロパティ pattern ぷろぱてぃ 1034
pause MP4Media/FLV メソッド pause MP4Media/FLV めそっど 496
pause サウンドオブジェクトメソッド pause さうんどおぶじぇくとめそっど 498
pause() (3D) メソッド pause() (3D) めそっど 496
pause() (DVD) メソッド pause() (DVD) めそっど 495
pause()（サウンドチャネル）メソッド pause()（さうんどちゃねる）めそっど 497
pause() ミクサメソッド pause() みくさめそっど 495
pause（RealMedia、Windows Media）メソッド pause（RealMedia、Windows Media）めそっど 497
pausedAtStart (RealMedia, Windows Media) プロパティ pausedAtStart (RealMedia, Windows Media) ぷろぱてぃ 1036
pausedAtStart MP4Media/FLV プロパティ 1035
pausedAtStart（Flash、デジタルビデオ）プロパティ pausedAtStart（Flash、でじたるびでお）ぷろぱてぃ 1035
PauseSimulation（ワールド）関数 PauseSimulation（わーるど）かんすう 1244
percentBuffered プロパティ percentBuffered ぷろぱてぃ 1038
percentPlayed プロパティ 1038
percentStreamed (3D) プロパティ percentStreamed (3D) ぷろぱてぃ 1039
percentStreamed サウンドオブジェクトプロパティ 1041
percentStreamed プロパティ percentStreamed ぷろぱてぃ 1039
percentStreamer MP4Media/FLV プロパティ 1040
period プロパティ period ぷろぱてぃ 1041
perlinNoise メソッド perlinNoise めそっど 499
perpendicularTo メソッド perpendicularTo めそっど 500
persistent プロパティ persistent ぷろぱてぃ 1042
PI 定数 PI ていすう 167
picture（ウィンドウ）プロパティ picture（うぃんどう）ぷろぱてぃ 1043
picture（メンバ）プロパティ picture（めんば）ぷろぱてぃ 1042
pictureP() メソッド pictureP() めそっど 501

PitchShiftFilter（オーディオ）143
platform プロパティ platform ぷろぱてぃ 1044
play MP4Media/FLV メソッド play MP4Media/FLV めそっど 504
play() (3D) メソッド play() (3D) めそっど 501
play() (DVD) メソッド play() (DVD) めそっど 502
play()（サウンドチャネル）メソッド play()（さうんどちゃねる）めそっど 504
play() サウンドオブジェクトメソッド play() さうんどおぶじぇくとめそっど 507
play() ミクサメソッド play() みくさめそっど 504
play()（RealMedia、Windows Media）メソッド play()（RealMedia、Windows Media）めそっど 505
playBackMode プロパティ playBackMode ぷろぱてぃ 1044
playBackRate プロパティ playBackRate ぷろぱてぃ 1049
playerParentalLevel() メソッド playerParentalLevel() めそっど 510
playFile() メソッド playFile() めそっど 505
playing (3D) プロパティ playing (3D) ぷろぱてぃ 1046
playing プロパティ playing ぷろぱてぃ 1045
playlist プロパティ playlist ぷろぱてぃ 1046
playNext() (3D) メソッド playNext() (3D) めそっど 509
playNext()（サウンドチャネル）メソッド playNext()（さうんどちゃねる）めそっど 508
playRate (3D) プロパティ playRate (3D) ぷろぱてぃ 1047
playRate (DVD) プロパティ playRate (DVD) ぷろぱてぃ 1047
playRate サウンドオブジェクトプロパティ 1048
playrate（QuickTime、AVI、MP4）プロパティ 1048
point() メソッド point() めそっど 510
pointA（制限）プロパティ pointA（せいげん）ぷろぱてぃ 1275
pointA（スプリング）プロパティ pointA（すぷりんぐ）ぷろぱてぃ 1279
pointAt メソッド pointAt めそっど 511
pointAtOrientation プロパティ pointAtOrientation ぷろぱてぃ 1049
pointB（制限）プロパティ pointB（せいげん）ぷろぱてぃ 1276
pointB（スプリング）プロパティ pointB（すぷりんぐ）ぷろぱてぃ 1280
pointInHyperlink() メソッド pointInHyperlink() めそっど 512
pointOfContact プロパティ pointOfContact ぷろぱてぃ 1050
pointToChar() メソッド pointToChar() めそっど 512
pointToItem() メソッド pointToItem() めそっど 514
pointToLine() メソッド pointToLine() めそっど 514
pointToParagraph() メソッド pointToParagraph() めそっど 515
pointToWord() メソッド pointToWord() めそっど 516
position（トランスフォーム）プロパティ position（とらんすふぉーむ）ぷろぱてぃ 1051
position バイト配列プロパティ 1050
position（剛体）プロパティ position（ごうたい）ぷろぱてぃ 1261
position（地形）プロパティ position（ちけい）ぷろぱてぃ 1291
positionReset プロパティ positionReset ぷろぱてぃ 1052
posterFrame プロパティ posterFrame ぷろぱてぃ 1052
postNetByteArray メソッド postNetByteArray めそっど 517
postNetText メソッド postNetText めそっど 518
power() メソッド power() めそっど 519
preferred3dRenderer プロパティ preferred3dRenderer ぷろぱてぃ 1053
preLoad (3D) プロパティ preLoad (3D) ぷろぱてぃ 1054
preLoad（メンバ）プロパティ preLoad（めんば）ぷろぱてぃ 1054
preLoad() (Movie) メソッド preLoad() (Movie) めそっど 521
preLoad()（メンバ）メソッド preLoad()（めんば）めそっど 520
preLoadBuffer メンバメソッド preLoadBuffer めんば めそっど 522
preLoadEventAbort プロパティ preLoadEventAbort ぷろぱてぃ 1055
preLoadMember() メソッド preLoadMember() めそっど 522
preLoadMode プロパティ preLoadMode ぷろぱてぃ 1055
preLoadMovie() メソッド preLoadMovie() めそっど 523
preloadNetThing() メソッド preloadNetThing() めそっど 524
preLoadRAM プロパティ preLoadRAM ぷろぱてぃ 1056
preLoadTime プロパティ preLoadTime ぷろぱてぃ 1057
preMultiply メソッド preMultiply めそっど 525
preRotate メソッド preRotate めそっど 525
preScale() メソッド preScale() めそっど 527
preTranslate() メソッド preTranslate() めそっど 528
primitives プロパティ primitives ぷろぱてぃ 1058
print() メソッド print() めそっど 529
printAsBitmap() メソッド printAsBitmap() めそっど 529
printFrom() メソッド printFrom() めそっど 530
productName プロパティ productName ぷろぱてぃ 1058
productVersion プロパティ productVersion ぷろぱてぃ 1058
projection プロパティ projection ぷろぱてぃ 1059
properties（剛体）プロパティ properties（ごうたい）ぷろぱてぃ 1261
properties（制限）プロパティ properties（せいげん）ぷろぱてぃ 1276
property キーワード property きーわーど 227
propList() メソッド propList() めそっど 531
proxyServer メソッド proxyServer めそっど 532
ptToHotSpotID() メソッド ptToHotSpotID() めそっど 533
puppetPalette() メソッド puppetPalette() めそっど 533
puppetSprite() メソッド puppetSprite() めそっど 534
puppetTempo() メソッド puppetTempo() めそっど 535
puppetTransition() メソッド puppetTransition() めそっど 536
purgePriority プロパティ purgePriority ぷろぱてぃ 1060
put() メソッド put() めそっど 538

Q

qtRegisterAccessKey メソッド qtRegisterAccessKey めそっど 539
qtUnRegisterAccessKey メソッド qtUnRegisterAccessKey めそっど 539
quad プロパティ quad ぷろぱてぃ 1060
quality (3D) プロパティ quality (3D) ぷろぱてぃ 1062

quality プロパティ quality ぷろぱてぃ 1061
queue() (3D) メソッド queue() (3D) めそっど 541
queue() メソッド queue() めそっど 540
QuickTime 3 マスク 960
QuickTime オブジェクト QuickTime おぶじぇくと 123
quickTimeVersion() メソッド quickTimeVersion() めそっど 542
quit() メソッド quit() めそっど 542

R

radius のプロパティ 1063
ramNeeded() メソッド ramNeeded() めそっど 543
random() メソッド random() めそっど 544
randomSeed プロパティ randomSeed ぷろぱてぃ 1063
randomVector メソッド randomVector めそっど 546
randomVector() メソッド randomVector() めそっど 545
rawNew() メソッド rawNew() めそっど 546
raycastAll（レイキャスティング）メソッド raycastAll（れいきゃすてぃんぐ）めそっど 1310
readBoolean メソッド readBoolean めそっど 547
readByteArray FileIO Xtra メソッド readByteArray FileIO Xtra めそっど 548
readByteArray メソッド readByteArray めそっど 547
readChar() メソッド readChar() めそっど 548
readFile() メソッド readFile() めそっど 549
readFloat32 メソッド readFloat32 めそっど 550
readFloat64 メソッド readFloat64 めそっど 550
readInt16 メソッド readInt16 めそっど 551
readInt32 メソッド readInt32 めそっど 551
readInt8 メソッド readInt8 めそっど 551
readLine() メソッド readLine() めそっど 552
readRawString メソッド readRawString めそっど 553
readString メソッド readString めそっど 553
readToken() メソッド readToken() めそっど 553
readWord() メソッド readWord() めそっど 554
RealMedia オブジェクト RealMedia おぶじぇくと 125
realPlayerNativeAudio() メソッド realPlayerNativeAudio() めそっど 555
realPlayerPromptToInstall() メソッド realPlayerPromptToInstall() めそっど 556
realPlayerVersion() メソッド realPlayerVersion() めそっど 557
recordFont プロパティ recordFont ぷろぱてぃ 1064
recordFont メソッド recordFont めそっど 558
rect (camera) プロパティ rect (camera) ぷろぱてぃ 1064
rect（画像）プロパティ rect（がぞう）ぷろぱてぃ 1065
rect（ウィンドウ）プロパティ rect（うぃんどう）ぷろぱてぃ 1067
rect（スプライト）プロパティ rect（すぷらいと）ぷろぱてぃ 1067
rect（メンバ）プロパティ rect（めんば）ぷろぱてぃ 1066
rect() メソッド rect() めそっど 559
ref プロパティ ref ぷろぱてぃ 1068
reflectionMap プロパティ reflectionMap ぷろぱてぃ 1068
reflectivity 1069
reflectivity プロパティ reflectivity ぷろぱてぃ 1069
RegExp データ型 RegExp でーたがた 11
region プロパティ region ぷろぱてぃ 1070
registerByteArrayCallback メソッド registerByteArrayCallback めそっど 561
registerCollisionCallback（衝突）メソッド registerCollisionCallback（しょうとつ）めそっど 1285
registerCuePointCallback メソッド registerCuePointCallback めそっど 563
registerEndOfSpoolCallback() 564
registerForEvent() メソッド registerForEvent() めそっど 565
registerScript() メソッド registerScript() めそっど 566
regPoint (3D) プロパティ regPoint (3D) ぷろぱてぃ 1071
regPoint プロパティ regPoint ぷろぱてぃ 1070
regPointVertex プロパティ regPointVertex ぷろぱてぃ 1072
removeBackdrop メソッド removeBackdrop めそっど 568
removeFromWorld メソッド removeFromWorld めそっど 568
removeLast() メソッド removeLast() めそっど 569
removeModifier メソッド removeModifier めそっど 569
removeOverlay メソッド removeOverlay めそっど 570
removeScriptedSprite() メソッド removeScriptedSprite() めそっど 570
renderer プロパティ renderer ぷろぱてぃ 1072
rendererDeviceList プロパティ rendererDeviceList ぷろぱてぃ 1073
renderFormat プロパティ renderFormat ぷろぱてぃ 1073
renderStyle プロパティ renderStyle ぷろぱてぃ 1074
repeat while キーワード repeat while きーわーど 231
repeat ループ
  next repeat キーワード 226
  repeat with キーワード 232
  repeat with...down to キーワード 233
  repeat with...in list キーワード 234
  終了 218
replaceMember メソッド replaceMember めそっど 571
Reset ミクサメソッド Reset みくさめそっど 572
resetWorld メソッド resetWorld めそっど 572
resolution 1075, 1076
resolution (3D) プロパティ resolution (3D) ぷろぱてぃ 1075
resolution (DVD) プロパティ resolution (DVD) ぷろぱてぃ 1076
resolve プロパティ resolve ぷろぱてぃ 1076
resolveA メソッド resolveA めそっど 573
resolveB メソッド resolveB めそっど 573
resource プロパティ resource ぷろぱてぃ 1077
restart() メソッド restart() めそっど 574
restitution（剛体）プロパティ restitution（ごうたい）ぷろぱてぃ 1262
restitution（ワールド）プロパティ restitution（わーるど）ぷろぱてぃ 1239
restLength（スプリング）プロパティ restLength（すぷりんぐ）ぷろぱてぃ 1280

restore() メソッド restore() めそっど 574
result メソッド result めそっど 575
ResumeSimulation（ワールド）関数 ResumeSimulation（わーるど）かんすう 1245
return 関数、ハンドラ return かんすう、はんどら 29
return キーワード return きーわーど 234
returnToTitle() 576
ReverbFilter（オーディオ）142
revertToWorldDefaults メソッド revertToWorldDefaults めそっど 577
rewind() (Windows Media) メソッド rewind() (Windows Media) めそっど 578
rewind() （サウンドチャネル）メソッド rewind() （サウンドチャネル）めそっど 578
rewind() MP4Media/FLV メソッド rewind() MP4Media/FLV めそっど 577
right (3D) プロパティ right (3D) ぷろぱてぃ 1078
right プロパティ right ぷろぱてぃ 1077
rightIndent プロパティ rightIndent ぷろぱてぃ 1078
rightMouseDown プロパティ rightMouseDown ぷろぱてぃ 1079
rightMouseUp プロパティ rightMouseUp ぷろぱてぃ 1079
rollOver テスト rollOver てすと 419
rollOver() メソッド rollOver() めそっど 579
romanLingo プロパティ romanLingo ぷろぱてぃ 1080
rootLock プロパティ rootLock ぷろぱてぃ 1080
rootMenu() メソッド rootMenu() めそっど 581
rootNode プロパティ rootNode ぷろぱてぃ 1081
rotate メソッド rotate めそっど 581
rotation 1084
rotation (backdrop and overlay) プロパティ rotation (backdrop and overlay) ぷろぱてぃ 1083
rotation (engraver shader) プロパティ rotation (engraver shader) ぷろぱてぃ 1083
rotation （トランスフォーム）プロパティ rotation （とらんすふぉーむ）ぷろぱてぃ 1084
rotation プロパティ rotation ぷろぱてぃ 1081
rotationReset プロパティ rotationReset ぷろぱてぃ 1084
rowscale（地形）プロパティ rowscale（ちけい）ぷろぱてぃ 1292
RTF プロパティ RTF ぷろぱてぃ 1084
run メソッド run めそっど 582
runMode メソッド runMode めそっど 582, 583

S

safePlayer プロパティ safePlayer ぷろぱてぃ 1085
sampleCount サウンドオブジェクトプロパティ 1087
sampleCount プロパティ sampleCount ぷろぱてぃ 1086
sampleRate サウンドオブジェクトプロパティ 1089
sampleRate プロパティ sampleRate ぷろぱてぃ 1088
sampleRate ミクサプロパティ 1087
sampleSize プロパティ sampleSize ぷろぱてぃ 1089
save castLib メソッド save castLib めそっど 585
save サウンドオブジェクトメソッド save さうんどおぶじぇくとめそっど 585
save ミクサメソッド save みくさめそっど 584
saveMovie() メソッド saveMovie() めそっど 586
savew3d プロパティ savew3d ぷろぱてぃ 1090
saveWorld プロパティ saveWorld ぷろぱてぃ 1090
scale (3D) プロパティ scale (3D) ぷろぱてぃ 1090
scale (backdrop and overlay) プロパティ scale (backdrop and overlay) ぷろぱてぃ 1091
scale （コマンド）メソッド scale （コマンド）メソッド 586
scale （トランスフォーム）プロパティ scale （とらんすふぉーむ）ぷろぱてぃ 1093
scale プロパティ scale ぷろぱてぃ 1091
scaleMode プロパティ scaleMode ぷろぱてぃ 1093
scalingFactor（ワールド）プロパティ scalingFactor（わーるど）ぷろぱてぃ 1239
score プロパティ score ぷろぱてぃ 1094
scoreColor プロパティ scoreColor ぷろぱてぃ 1095
scoreSelection プロパティ scoreSelection ぷろぱてぃ 1095
script プロパティ script ぷろぱてぃ 1096
script() メソッド script() めそっど 588
scripted プロパティ scripted ぷろぱてぃ 1097
scriptingXtraList プロパティ scriptingXtraList ぷろぱてぃ 1098
scriptInstanceList プロパティ 51, 1098
scriptList プロパティ scriptList ぷろぱてぃ 1099
scriptNum プロパティ scriptNum ぷろぱてぃ 1099
scriptsEnabled プロパティ scriptsEnabled ぷろぱてぃ 1100
scriptText プロパティ scriptText ぷろぱてぃ 1100
scriptType プロパティ scriptType ぷろぱてぃ 1101
scrollByLine メソッド scrollByLine めそっど 588
scrollByPage メソッド scrollByPage めそっど 589
scrollTop プロパティ scrollTop ぷろぱてぃ 1102
sds （モディファイア）プロパティ sds （もでぃふぁいあ）ぷろぱてぃ 1102
searchCurrentFolder プロパティ searchCurrentFolder ぷろぱてぃ 1104
searchPathList プロパティ searchPathList ぷろぱてぃ 1104
seek MP4Media/FLV メソッド seek MP4Media/FLV めそっど 591
seek サウンドオブジェクトメソッド seek さうんどおぶじぇくとめそっど 591
seek() メソッド seek() めそっど 590
selectAtLoc() メソッド selectAtLoc() めそっど 591
selectButton() メソッド selectButton() めそっど 592
selectButtonRelative() メソッド selectButtonRelative() めそっど 593
selectedButton プロパティ selectedButton ぷろぱてぃ 1105
selectedText プロパティ selectedText ぷろぱてぃ 1106
selection プロパティ selection ぷろぱてぃ 1106
selection() （関数）メソッド selection() （かんすう）めそっど 593
selection（テキストキャストメンバプロパティ）プロパティ selection（てきすときゃすとめんばぷろぱてぃ）ぷろぱてぃ 1107

selEnd プロパティ selEnd ぷろぱてぃ 1108
selStart プロパティ selStart ぷろぱてぃ 1108
sendAllSprites() メソッド sendAllSprites() めそっど 594
sendEvent メソッド sendEvent めそっど 595
sendSprite() コマンド sendSprite() こまんど 595
serialNumber プロパティ serialNumber ぷろぱてぃ 1109
setAlpha() メソッド setAlpha() めそっど 596
setaProp メソッド setaProp めそっど 597
setAt メソッド setAt めそっど 598
setCallback() メソッド setCallback() めそっど 599
setCharSet メソッド setCharSet めそっど 600
setCollisionCallback() メソッド setCollisionCallback() めそっど 600
setFilterMask() メソッド setFilterMask() めそっど 601
setFinderInfo() メソッド setFinderInfo() めそっど 602
setFlashProperty() メソッド setFlashProperty() めそっど 602
setNewLineConversion() メソッド setNewLineConversion() めそっど 603
setPixel() メソッド setPixel() めそっど 604
setPixels() (Bytearray) 605
setPlayList() メソッド setPlayList() めそっど 606
setPosition() メソッド setPosition() めそっど 607
setPref() メソッド setPref() めそっど 608
setProp メソッド setProp めそっど 610
setScriptList() メソッド setScriptList() めそっど 611
settingsPanel() メソッド settingsPanel() めそっど 611
setTrackEnabled メソッド setTrackEnabled めそっど 612
setVal() 613
setVariable() メソッド setVariable() めそっど 613
shader のプロパティ 1110
shaderList プロパティ shaderList ぷろぱてぃ 1111
shadowPercentage プロパティ shadowPercentage ぷろぱてぃ 1111
shadowStrength プロパティ shadowStrength ぷろぱてぃ 1112
shape（剛体）プロパティ shape（ごうたい）ぷろぱてぃ 1262
shapeType プロパティ shapeType ぷろぱてぃ 1112
shiftDown プロパティ shiftDown ぷろぱてぃ 1112
shininess 1114
shininess プロパティ shininess ぷろぱてぃ 1113
Shockwave 3D オブジェクト Shockwave 3D おぶじぇくと 126
Shockwave Audio キャストメンバ
  preLoadBuffer メンバコマンド 522
  エラー 352, 354
  状況 1137
  メンバの percentPlayed プロパティ 1038
  メンバの percentStreamed プロパティ 1039
Shockwave オーディオオブジェクト Shockwave おーでぃおおぶじぇくと 126
Shockwave ムービー
  インターネットから再生する 387
  デバッグ 90
  開く 388
Shockwave、グローバル変数 Shockwave、ぐろーばるへんすう 16
showGlobals() メソッド showGlobals() めそっど 615
showLocals メソッド showLocals めそっど 614
showProps() メソッド showProps() めそっど 615
shutDown() メソッド shutDown() めそっど 616
silhouettes プロパティ silhouettes ぷろぱてぃ 1114
simulate()（ワールド）関数 simulate()（わーるど）かんすう 1245
sin() メソッド sin() めそっど 617
size プロパティ size ぷろぱてぃ 1114
sizeRange プロパティ sizeRange ぷろぱてぃ 1115
sizeState プロパティ sizeState ぷろぱてぃ 1115
skew プロパティ skew ぷろぱてぃ 1116
sleepMode（剛体）プロパティ sleepMode（ごうたい）ぷろぱてぃ 1263
sleepMode（ワールド）プロパティ sleepMode（わーるど）ぷろぱてぃ 1240
sleepThreshold（剛体）プロパティ sleepThreshold（ごうたい）ぷろぱてぃ 1263
sleepThreshold（ワールド）プロパティ sleepThreshold（わーるど）ぷろぱてぃ 1240
smoothness 1117
smoothness プロパティ smoothness ぷろぱてぃ 1117
sort メソッド sort めそっど 617
sound() メソッド sound() めそっど 618
soundBusy() メソッド soundBusy() めそっど 412
soundChannel (RealMedia) プロパティ soundChannel (RealMedia) ぷろぱてぃ 1119
soundChannel (SWA) プロパティ soundChannel (SWA) ぷろぱてぃ 1119
soundDevice プロパティ soundDevice ぷろぱてぃ 1120
soundDeviceList プロパティ soundDeviceList ぷろぱてぃ 1121
soundEnabled プロパティ soundEnabled ぷろぱてぃ 1122
soundKeepDevice プロパティ soundKeepDevice ぷろぱてぃ 1122
soundLevel プロパティ soundLevel ぷろぱてぃ 1123
soundMixMedia プロパティ soundMixMedia ぷろぱてぃ 1124
soundObjectList プロパティ 1124
source プロパティ source ぷろぱてぃ 1125
sourceFileName プロパティ sourceFileName ぷろぱてぃ 1125
sourceRect プロパティ sourceRect ぷろぱてぃ 1126
SPACE 定数 SPACE ていすう 168
specular (light) プロパティ specular (light) ぷろぱてぃ 1126
specular (shader) プロパティ specular (shader) ぷろぱてぃ 1127
specularColor プロパティ specularColor ぷろぱてぃ 1128
specularLightMap プロパティ specularLightMap ぷろぱてぃ 1128
SpeechXtra オブジェクト SpeechXtra おぶじぇくと 151
spotAngle プロパティ spotAngle ぷろぱてぃ 1129
spotDecay プロパティ spotDecay ぷろぱてぃ 1129
sprite（スプライトチャネル）プロパティ sprite（すぷらいとちゃねる）ぷろぱてぃ 1130
sprite（ムービー）プロパティ sprite（むーびー）ぷろぱてぃ 1130
sprite...intersects キーワード sprite...intersects きーわーど 236

sprite...within キーワード sprite...within きーわーど 237
sprite() メソッド sprite() めそっど 618
spriteNum プロパティ spriteNum ぷろぱてぃ 1131
spriteSpaceToWorldSpace メソッド spriteSpaceToWorldSpace めそっど 619
sqrt() メソッド sqrt() めそっど 620
stage プロパティ stage ぷろぱてぃ 1132
stageBottom メソッド stageBottom めそっど 620
stageLeft メソッド stageLeft めそっど 621
stageRight メソッド stageRight めそっど 622
stageToFlash() メソッド stageToFlash() めそっど 622
stageTop メソッド stageTop めそっど 623
startAngle プロパティ startAngle ぷろぱてぃ 1132
startFrame プロパティ startFrame ぷろぱてぃ 1133
starts 比較演算子 starts ひかくえんざんし 703
startSave サウンドオブジェクトメソッド startSave さうんどおぶじぇくとめそっど 624
startSave ミクサメソッド startSave みくさめそっど 624
startTime サウンドオブジェクトプロパティ 1134
startTime プロパティ startTime ぷろぱてぃ 1134
startTimeList プロパティ startTimeList ぷろぱてぃ 1135
state (Flash, SWA) プロパティ state (Flash, SWA) ぷろぱてぃ 1137
state (RealMedia) プロパティ state (RealMedia) ぷろぱてぃ 1138
static プロパティ static ぷろぱてぃ 1139
staticQuality プロパティ staticQuality ぷろぱてぃ 1140
status サウンドオブジェクトプロパティ 1142
status プロパティ status ぷろぱてぃ 1141
status ミクサプロパティ 1141
status() メソッド status() めそっど 625
stiffness（制限）プロパティ stiffness（せいげん）ぷろぱてぃ 1277
stiffness（スプリング）プロパティ stiffness（すぷりんぐ）ぷろぱてぃ 1281
stillDown プロパティ stillDown ぷろぱてぃ 1143
stop (Flash) メソッド stop (Flash) めそっど 626
stop メソッド stop めそっど 629
stop() (DVD) メソッド stop() (DVD) めそっど 625
stop() (RealMedia, Windows Media) メソッド stop() (RealMedia, Windows Media) めそっど 627
stop() （サウンドチャネル）メソッド stop() （さうんどちゃねる）めそっど 628
stop() MP4Media/FLV メソッド stop() MP4Media/FLV めそっど 627
stop() サウンドオブジェクトメソッド stop() さうんどおぶじぇくとめそっど 629
stop() ミクサメソッド stop() みくさめそっど 627
stopEvent() メソッド stopEvent() めそっど 630
stopSave サウンドオブジェクトメソッド stopSave さうんどおぶじぇくとめそっど 631
stopSave ミクサメソッド stopSave みくさめそっど 631
stopTime プロパティ stopTime ぷろぱてぃ 1143
stopTimeList プロパティ stopTimeList ぷろぱてぃ 1144
stream メソッド stream めそっど 632
streamMode プロパティ streamMode ぷろぱてぃ 1145
streamName プロパティ streamName ぷろぱてぃ 1146
streamSize (3D) プロパティ streamSize (3D) ぷろぱてぃ 1147
streamSize プロパティ streamSize ぷろぱてぃ 1146
String バイト配列演算子 String ばいとはいれつえんざんし 704
string() メソッド string() めそっど 633
stringP() メソッド stringP() めそっど 634
strokeColor プロパティ strokeColor ぷろぱてぃ 1147
strokeWidth プロパティ strokeWidth ぷろぱてぃ 1148
style プロパティ style ぷろぱてぃ 1149
subdivision プロパティ subdivision ぷろぱてぃ 1149
subPicture プロパティ subPicture ぷろぱてぃ 1150
subPictureCount プロパティ subPictureCount ぷろぱてぃ 1150
subPictureType() メソッド subPictureType() めそっど 634
subSteps（ワールド）プロパティ subSteps（わーるど）ぷろぱてぃ 1241
substituteFont メソッド substituteFont めそっど 635
suspendUpdates プロパティ suspendUpdates ぷろぱてぃ 1150
swing() メソッド swing() めそっど 636
Swing1Limit（6dof）プロパティ Swing1Limit（6dof）ぷろぱてぃ 1305
swing1Motion（6dof）プロパティ swing2Motion（6dof）ぷろぱてぃ 1306
swing2Motion（6dof）プロパティ swing2Motion（6dof）ぷろぱてぃ 1307
swingDrive（6dof）プロパティ swingDrive（6dof）ぷろぱてぃ 1307
switchColorDepth プロパティ switchColorDepth ぷろぱてぃ 1151
symbol() メソッド symbol() めそっど 637
symbolP() メソッド symbolP() めそっど 638
systemTrayIcon プロパティ systemTrayIcon ぷろぱてぃ 1152
systemTrayTooltip プロパティ systemTrayTooltip ぷろぱてぃ 1152

T

TAB 文字定数 TAB もじていすう 169
Tab キー、スクリプトの再フォーマットに使用 69
Tab キー Tab きー 169
tabCount プロパティ tabCount ぷろぱてぃ 1153
tabs プロパティ tabs ぷろぱてぃ 1153
tan() メソッド tan() めそっど 638
target プロパティ target ぷろぱてぃ 1154
targetFrameRate プロパティ targetFrameRate ぷろぱてぃ 1154
tellStreamStatus() メソッド tellStreamStatus() めそっど 638
tellTarget() メソッド tellTarget() めそっど 639
tension プロパティ tension ぷろぱてぃ 1155
terraindesc（地形）プロパティ terraindesc（ちけい）ぷろぱてぃ 1292
text プロパティ text ぷろぱてぃ 1155
texture プロパティ texture ぷろぱてぃ 1156

textureCoordinateList プロパティ textureCoordinateList ぷろぱてぃ 1157
textureCoordinates プロパティ textureCoordinates ぷろぱてぃ 1157
textureLayer プロパティ textureLayer ぷろぱてぃ 1158
textureList プロパティ textureList ぷろぱてぃ 1159
textureMember プロパティ textureMember ぷろぱてぃ 1159
textureMode プロパティ textureMode ぷろぱてぃ 1160
textureModeList プロパティ textureModeList ぷろぱてぃ 1160
textureRenderFormat プロパティ textureRenderFormat ぷろぱてぃ 1161
textureRepeat プロパティ textureRepeat ぷろぱてぃ 1162
textureRepeatList プロパティ textureRepeatList ぷろぱてぃ 1163
textureTransform プロパティ textureTransform ぷろぱてぃ 1163
textureTransformList プロパティ textureTransformList ぷろぱてぃ 1165
textureType プロパティ textureType ぷろぱてぃ 1167
thumbNail プロパティ thumbNail ぷろぱてぃ 1168
tilt プロパティ tilt ぷろぱてぃ 1168
time（タイムアウトオブジェクト）プロパティ time（たいむあうとおぶじぇくと）ぷろぱてぃ 1168
time()（システム）メソッド time()（しすてむ）めそっど 640
timeout オブジェクト
子オブジェクトにイベントを送る 1154
作成 52
システムイベントの中継 53
名前の設定 1001
timeOut ハンドラ timeOut はんどら 210
timeout() メソッド timeout() めそっど 641
timeoutHandler プロパティ timeoutHandler ぷろぱてぃ 1169
timeOutList プロパティ timeOutList ぷろぱてぃ 53
timeoutList プロパティ timeoutList ぷろぱてぃ 1170
timeScale プロパティ timeScale ぷろぱてぃ 1170
timeStep（ワールド）プロパティ timeStep（わーるど）ぷろぱてぃ 1241
timeStepMode（ワールド）プロパティ timeStepMode（わーるど）ぷろぱてぃ 1242
title (DVD) プロパティ title (DVD) ぷろぱてぃ 1171
title（ウィンドウ）プロパティ title（うぃんどう）ぷろぱてぃ 1171
titlebarOptions プロパティ titlebarOptions ぷろぱてぃ 1172
titleCount プロパティ titleCount ぷろぱてぃ 1173
titleMenu() メソッド titleMenu() めそっど 641
toChannels サウンドオブジェクトプロパティ 1029
toChannels ミクサプロパティ 1027
toHexString メソッド toHexString めそっど 642
toolXtraList プロパティ toolXtraList ぷろぱてぃ 1173
toon（モディファイア）プロパティ toon（もでぃふぁいあ）ぷろぱてぃ 1174
top (3D) メソッド top (3D) めそっど 643
top プロパティ top ぷろぱてぃ 1175
topCap メソッド topCap めそっど 643
topRadius メソッド topRadius めそっど 644
topSpacing プロパティ topSpacing ぷろぱてぃ 1175
trace() メソッド trace() めそっど 644
traceLoad プロパティ traceLoad ぷろぱてぃ 1176
traceScript プロパティ traceScript ぷろぱてぃ 1177
trackCount（キャストメンバ）プロパティ trackCount（きゃすとめんば）ぷろぱてぃ 1178
trackCount（スプライトプロパティ）trackCount（すぷらいとぷろぱてぃ）1178
trackEnabled プロパティ trackEnabled ぷろぱてぃ 1179
trackInfo プロパティ 1179
trackNextKeyTime プロパティ trackNextKeyTime ぷろぱてぃ 1180
trackNextSampleTime プロパティ trackNextSampleTime ぷろぱてぃ 1180
trackPreviousKeyTime プロパティ trackPreviousKeyTime ぷろぱてぃ 1181
trackPreviousSampleTime プロパティ trackPreviousSampleTime ぷろぱてぃ 1181
trackStartTime（キャストメンバ）プロパティ trackStartTime（きゃすとめんば）ぷろぱてぃ 1182
trackStartTime（スプライト）プロパティ trackStartTime（すぷらいと）ぷろぱてぃ 1182
trackStopTime（キャストメンバ）プロパティ trackStopTime（きゃすとめんば）ぷろぱてぃ 1183
trackStopTime（スプライト）プロパティ trackStopTime（すぷらいと）ぷろぱてぃ 1183
trackText プロパティ trackText ぷろぱてぃ 1184
trackType（スプライト）プロパティ trackType（すぷらいと）ぷろぱてぃ 1185
trackType（キャストメンバ）プロパティ trackType（きゃすとめんば）ぷろぱてぃ 1184
trails プロパティ trails ぷろぱてぃ 1186
transform（コマンド）メソッド transform（コマンド）メソッド 645
transform（プロパティ）プロパティ transform（ぷろぱてぃ）ぷろぱてぃ 1186
transitionType プロパティ transitionType ぷろぱてぃ 1188
transitionXtraList プロパティ transitionXtraList ぷろぱてぃ 1188
translate メソッド translate めそっど 645
translation プロパティ translation ぷろぱてぃ 1189
transparent プロパティ transparent ぷろぱてぃ 1189
trayIconMouseDoubleClick ハンドラ trayIconMouseDoubleClick はんどら 210
trayIconMouseDown ハンドラ trayIconMouseDown はんどら 211
trayIconRightMouseDown ハンドラ trayIconRightMouseDown はんどら 212
triggerCallback プロパティ triggerCallback ぷろぱてぃ 1190
trimWhiteSpace プロパティ trimWhiteSpace ぷろぱてぃ 1191
TRUE 論理定数 TRUE ろんりていすう 169
TRUE キーワード 20
tunnelDepth プロパティ tunnelDepth ぷろぱてぃ 1192
tweened プロパティ tweened ぷろぱてぃ 1192

tweenMode プロパティ tweenMode ぷろぱてぃ 1192
TwistDrive（6dof）プロパティ TwistDrive（6dof）ぷろぱてぃ 1308
twistLimit（6dof）プロパティ twistLimit（6dof）ぷろぱてぃ 1309
twistMotion（6dof）プロパティ twistMotion（6dof）ぷろぱてぃ 1309
type (light) プロパティ type (light) ぷろぱてぃ 1193
type (motion) プロパティ type (motion) ぷろぱてぃ 1196
type (shader) プロパティ type (shader) ぷろぱてぃ 1197
type (sprite) プロパティ type (sprite) ぷろぱてぃ 1197
type (texture) プロパティ type (texture) ぷろぱてぃ 1198
type （ウィンドウ）プロパティ type （うぃんどう）ぷろぱてぃ 1198
type （メンバ）プロパティ type （めんば）ぷろぱてぃ 1194
type （モデルリソース）プロパティ type （もでるりそーす）ぷろぱてぃ 1195
type（剛体）プロパティ type（ごうたい）ぷろぱてぃ 1264

U

uncompress() メソッド uncompress() めそっど 647
union() メソッド union() めそっど 647
unLoad()（ムービー）メソッド unLoad()（むーびー）めそっど 649
unLoad() （メンバ）メソッド unLoad() （めんば）めそっど 648
unLoadMember() メソッド unLoadMember() めそっど 650
unLoadMovie() メソッド unLoadMovie() めそっど 651
unmute サウンドオブジェクトメソッド unmute さうんどおぶじぇくとめそっど 652
unmute ミクサメソッド unmute みくさめそっど 651
unregisterAllEvents メソッド unregisterAllEvents めそっど 652
unregisterByteArrayCallback メソッド unregisterByteArrayCallback めそっど 653
unregisterCuePointCallback メソッド unregisterCuePointCallback めそっど 653
unregisterEndOfSpoolCallback() 654
update メソッド update めそっど 653
updateFrame() メソッド updateFrame() めそっど 655
updateLock プロパティ updateLock ぷろぱてぃ 1199
updateMovieEnabled プロパティ updateMovieEnabled ぷろぱてぃ 1200
updateStage() メソッド updateStage() めそっど 656
URL プロパティ URL ぷろぱてぃ 1200
URLEncode メソッド URLEncode めそっど 657
useAlpha プロパティ useAlpha ぷろぱてぃ 1201
useDiffuseWithTexture プロパティ useDiffuseWithTexture ぷろぱてぃ 1202
useFastQuads プロパティ useFastQuads ぷろぱてぃ 1202
useHypertextStyles プロパティ useHypertextStyles ぷろぱてぃ 1203
useLineOffset プロパティ useLineOffset ぷろぱてぃ 1204
useMatrix サウンドオブジェクトプロパティ 1029
useMatrix ミクサプロパティ 1027
userData プロパティ userData ぷろぱてぃ 1204
userData（剛体）プロパティ userData（ごうたい）ぷろぱてぃ 1264
userName (RealMedia) プロパティ userName (RealMedia) ぷろぱてぃ 1205
userName プロパティ userName ぷろぱてぃ 1204
useTargetFrameRate プロパティ useTargetFrameRate ぷろぱてぃ 1206

V

value() メソッド value() めそっど 658
vector() メソッド vector() めそっど 659
version キーワード version きーわーど 237
version() メソッド version() めそっど 660
vertex プロパティ vertex ぷろぱてぃ 1207
vertexList (mesh deform) プロパティ vertexList (mesh deform) ぷろぱてぃ 1209
vertexList (mesh generator) プロパティ vertexList (mesh generator) ぷろぱてぃ 1208
vertexList プロパティ vertexList ぷろぱてぃ 1207
vertices プロパティ vertices ぷろぱてぃ 1209
video (QuickTime, AVI) プロパティ video (QuickTime, AVI) ぷろぱてぃ 1210
video (RealMedia, Windows Media) プロパティ video (RealMedia, Windows Media) ぷろぱてぃ 1211
video MP4Media/FLV プロパティ 1210
videoFormat プロパティ videoFormat ぷろぱてぃ 1212
videoForWindowsPresent プロパティ videoForWindowsPresent ぷろぱてぃ 1212
viewH プロパティ viewH ぷろぱてぃ 1212
viewPoint プロパティ viewPoint ぷろぱてぃ 1213
viewScale プロパティ viewScale ぷろぱてぃ 1215
viewV プロパティ viewV ぷろぱてぃ 1215
visibility プロパティ visibility ぷろぱてぃ 1217
visible （スプライト）プロパティ visible （すぷらいと）ぷろぱてぃ 1217
visible プロパティ visible ぷろぱてぃ 1216
voiceCount() メソッド voiceCount() めそっど 660
voiceGet() メソッド voiceGet() めそっど 661
voiceGetAll() メソッド voiceGetAll() めそっど 662
voiceGetPitch() メソッド voiceGetPitch() めそっど 662
voiceGetRate() メソッド voiceGetRate() めそっど 663
voiceGetVolume() メソッド voiceGetVolume() めそっど 664
voiceInitialize() メソッド voiceInitialize() めそっど 664
voicePause() メソッド voicePause() めそっど 665
voiceResume() メソッド voiceResume() めそっど 666
voiceSet() メソッド voiceSet() めそっど 666
voiceSetPitch() メソッド voiceSetPitch() めそっど 667
voiceSetRate() メソッド voiceSetRate() めそっど 667
voiceSetVolume() メソッド voiceSetVolume() めそっど 668
voiceSpeak() メソッド voiceSpeak() めそっど 668

voiceState() メソッド voiceState() めそっど 669
voiceStop() メソッド voiceStop() めそっど 670
voiceWordPos() メソッド voiceWordPos() めそっど 670
VOID 定数 VOID ていすう 170
VOID データ型 VOID でーたがた 11
voidP() メソッド voidP() めそっど 671
volume (DVD) プロパティ volume (DVD) ぷろぱてぃ 1218
volume（キャストメンバ）プロパティ volume（きゃすとめんば）ぷろぱてぃ 1218
volume（サウンドチャネル）プロパティ volume（さうんどちゃねる）ぷろぱてぃ 1220
volume MP4Media/FLV プロパティ 1219
volume サウンドオブジェクトプロパティ 1220
volume プロパティ volume ぷろぱてぃ 1221
volume ミクサプロパティ 1219
volume（スプライト）プロパティ volume（すぷらいと）ぷろぱてぃ 1221

W

warpMode プロパティ warpMode ぷろぱてぃ 1222
Web ページ、リフレッシュする Web ぺーじ、りふれっしゅする 264
Web ページをリフレッシュする Web ぺーじをりふれっしゅする 264
width（3D）プロパティ width（3D）ぷろぱてぃ 1223
width MP4Media/FLV プロパティ 1224
width プロパティ width ぷろぱてぃ 1223
widthVertices プロパティ widthVertices ぷろぱてぃ 1225
wind プロパティ wind ぷろぱてぃ 1225
window プロパティ window ぷろぱてぃ 1225
window() メソッド window() めそっど 672
windowBehind プロパティ windowBehind ぷろぱてぃ 1226
windowInFront プロパティ windowInFront ぷろぱてぃ 1226
windowList プロパティ windowList ぷろぱてぃ 1227
WindowOperation メソッド WindowOperation めそっど 672
windowPresent() メソッド windowPresent() めそっど 673
Windows Media オブジェクト Windows Media おぶじぇくと 130
Windows ソフトウェア用のビデオ Windows そふとうぇあようのびでお 1212
wordWrap プロパティ wordWrap ぷろぱてぃ 1228
worldPosition プロパティ worldPosition ぷろぱてぃ 1228
worldSpaceToSpriteSpace メソッド worldSpaceToSpriteSpace めそっど 674
worldTransform プロパティ worldTransform ぷろぱてぃ 1228
wrapTransform プロパティ wrapTransform ぷろぱてぃ 1229
wrapTransformList プロパティ wrapTransformList ぷろぱてぃ 1229
writeBoolean メソッド writeBoolean めそっど 674
writeByteArray FileIO Xtra メソッド writeByteArray FileIO Xtra めそっど 675
writeByteArray メソッド writeByteArray めそっど 675
writeChar() メソッド writeChar() めそっど 676
writeFloat32 メソッド writeFloat32 めそっど 676
writeFloat64 メソッド writeFloat64 めそっど 677
writeInt16 メソッド writeInt16 めそっど 678
writeInt32 メソッド writeInt32 めそっど 678
writeInt8 メソッド writeInt8 めそっど 677
writeRawString メソッド writeRawString めそっど 679
writeReturn() メソッド writeReturn() めそっど 679
writeString バイト配列メソッド writeString ばいとはいれつめそっど 681
writeString() メソッド writeString() めそっど 680

X

x（ベクトル）プロパティ x（べくとる）ぷろぱてぃ 1230
xAxis プロパティ xAxis ぷろぱてぃ 1230
XCMD および XFCN (Macintosh) 486
XCOD リソース XCOD りそーす 486
Xlibrary ファイル
　閉じる 284
　開く 486
XML Parser オブジェクト XML Parser おぶじぇくと 151
XML ハンドラ XML はんどら 205
Xtra 拡張
　ムービーで使用可能 1014
Xtra 拡張機能
　scriptingXtraList プロパティ 1098
　作成 463
　作成されるオブジェクト 481
　使用可能、リストする 1173, 1188
　スクリプトオブジェクト 42, 132
　スクリプト作成 68
　ストリングの数値 658
xtra プロパティ xtra ぷろぱてぃ 1231
xtra() メソッド xtra() めそっど 681
xtraList（プレーヤ）プロパティ xtraList（ぷれーや）ぷろぱてぃ 1232
xtraList（ムービー）プロパティ xtraList（むーびー）ぷろぱてぃ 1232

Y

y（ベクトル）プロパティ y（べくとる）ぷろぱてぃ 1233
yAxis プロパティ yAxis ぷろぱてぃ 1233
yon プロパティ yon ぷろぱてぃ 1234

Z

z（ベクトル）プロパティ z（べくとる）ぷろぱてぃ 1234
zAxis プロパティ zAxis ぷろぱてぃ 1235
zoomBox メソッド zoomBox めそっど 681
圧縮 798

以下 (\ 693
以上 (\>=) 694
一般的なスクリプト項目いっぱんてきなすくりぷとこうもく 67
影、フィールドキャストメンバ 762
円柱 643
階層、オブジェクトかいそう、おぶじぇくと 43
割り当てる、ビヘイビアー 72
括弧演算子 () かっこえんざんし () 688
括弧かっこ 7
環境設定、スクリプトウィンドウかんきょうせってい、すくりぷとうぃんどう 66
関数データ型かんすうでーたがた 10
距離、行 421
境界線、シェイプキャストメンバきょうかいせん、しぇいぷきゃすとめんば 944
矩形、識別する 423
矩形データ型くけいでーたがた 11
矩形とポイントの位置を特定する 430
矩形とポイントのサイズを特定する 430
空きメモリ 347, 348, 970
継承 47, 57
継続シンボル（\\）けいぞくしんぼる（\\）69
検索けんさく検索を参照
検索する、ファイル名 364
交差、スプライト 236
光沢 1126
行の折り返し 1228
行を折り返す 1228
項目、区切る 925
項目を区切る 925
再開スプライトメソッドさいかいすぷらいと めそっど 576
削除する
　リストから値 311
削除する . 削除を参照
算術演算子 19
指数 519
字体 884, 886
自然対数関数 428
自動ポーリングじどうぽーりんぐ 85
式の論理否定 701
終了、ハンドラしゅうりょう、はんどら 217
終了する、ハンドラしゅうりょうする、はんどら 239
除算、余り 700
除算演算子（/）じょざんえんざんし（/）691
除算演算子（/）じょざんえんざんし（/）(3D) 692
除算の余り 700
小数しょうすう 12
衝突、解決しょうとつ、かいけつ 1076
条件一致、式じょうけんいっち、しき 23
色深度 826
色分け、スクリプトいろわけ、すくりぷと 66
親 1030
数学データ型すうがくでーたがた 11
数値データ型すうちでーたがた 11
整数、シンタックス 12
整数データ型せいすうでーたがた 10
相対パス、パス名演算子（@）そうたいぱす、ぱすめいえんざんし（@）697
揃える、メンバフィールドプロパティ 715
多次元配列たじげんはいれつ 39
多次元リストたじげんりすと 36
対数関数 428
代入演算子（=）だいにゅうえんざんし（=）693
代入演算子だいにゅうえんざんし 20
値、リテラルの表記 12
定数データ型ていすうでーたがた 10
登録 564, 565
等しくない (\) 693
二重引用符（"）にじゅういんようふ（"）164, 167
日付データ型ひづけでーたがた 10
背景 244, 247, 403
配列データ型はいれつでーたがた 10
配列はいれつリストを参照
反転された整数値 454
比較演算子 19
標準形の X 軸ひょうじゅんけいの X じく 1231
標準形の Y 軸ひょうじゅんけいの Y じく 1233, 1235
標準形の Z 軸ひょうじゅんけいの Z じく 1235
評価する、式 229, 230, 235, 315
不動小数点数データ型ふどうしょうすうてんすうでーたがた 10
浮動小数点数 342, 881
浮動小数を四捨五入する 881
文字、フィールドキャストメンバに位置する 427
文字スペースもじすぺーす 9
平行移動 645
並べ替えリスト 617
未定義データ型みていぎでーたがた 11
優先順位、演算子 19
連結、ストリング 21
論理演算子ろんりえんざんし 20
論理式 698
論理条件、テストろんりじょうけん、てすと 22

あ

アイデンティティトランスフォーム 395
アスタリスク、乗算演算子（*）689, 691
値
　オブジェクトインスペクタ 85
　比較する 19
　変数、格納と更新 15
アニメーション GIF オブジェクトあにめーしょん GIF おぶじぇくと 112

アプリケーション
開始する 484
再開する 574
ハンドラ 171, 175
アプリケーション、起動する 484
アンセスタスクリプト、メッセージを送る 268
アンセスタスクリプトあんせすたすくりぷと 47
アンパサンド演算子（& または &&）686, 687

い

位置
キャストメンバの 274
スプライト 236, 237
デスクトップ上のステージ 620, 621, 622, 623
イベント、定義 5
イベントハンドラ . ハンドラを参照
イベントメッセージ 492
順序、実行 25
リスト 171
インスタンス変数いんすたんすへんすう 58
インスタンスメソッドいんすたんすめそっど 59
インターネット
ファイルをプリロードする 317, 524
インターネットから
Shockwave ムービーを再生する 387
インターネットファイル、MIME タイプ 460
インデント、スクリプトいんでんと、すくりぷと 69
イントゥーボタン、用途 90

う

ウィンドウ
forget メソッド 344
minimize() メソッド 443
on activateWindow ハンドラ 172
on closeWindow ハンドラ 173
on deactivateWindow ハンドラ 176
on moveWindow ハンドラ 198
on openWindow ハンドラ 199
on resizeWindow ハンドラ 201
on zoomWindow ハンドラ 213
open メソッド 485
picture プロパティ 1043
rect プロパティ 1067
ブラウザウィンドウにストリングを表示する 461
ウィンドウオブジェクトうぃんどうおぶじぇくと 109
ウォッチャーペイン、デバッガウィンドウうぉっちゃーぺいん、でばっがうぃんどう 88

え

エラー
Shockwave Audio キャストメンバ 352, 354
ネットワーク操作の間 457
エラーコードえらーこーど 1311
エレメント、定義 5
演算子
JavaScript 683
Lingo 683
算術 19
種類 18
ストリング 21
代入 20
定義 6
比較 19
優先順位 19
論理 20

お

オーディオフィルタ 132
オートカラーオプション 66
オーバーレイ
削除 570
挿入する 405
追加する 247
大文字 9, 12
大文字と小文字の区別 9, 12
オブジェクト
3D 154
インスタンス、JavaScript 56
階層 43
継承 57
コア 92
削除 312, 568
種類 42
スクリプト 132
デバッガウィンドウ 88
デバッグ 83
プロトタイプ 58, 61
メディアタイプ 112
オブジェクト指向プログラミングおぶじぇくとしこうぷろぐらみんぐ 45, 54
オブジェクトインスペクタおぶじぇくといんすぺくた 83
オブジェクトデータ型おぶじぇくとでーたがた 11
オプション改行文字（\\）214
親スクリプト
ancestor プロパティ 719
C++ および Java での対応する用語 46
オブジェクト指向プログラミング 46
記述 48
基本的な概念 46

作成されるオブジェクト 481
定義 4
プロパティ変数 48
編集 72

か

カーソルオブジェクトかーそるおぶじぇくと 115
カーブ、追加する 466
改行シンボル（\\）214
開始する
アプリケーション 484
スコア更新セッション 256
選択範囲の文字 1108
角括弧（[]）695
加算演算子（+）689, 690
数
浮動小数点数 881
カスタムクラス、JavaScript かすたむくらす、JavaScript 55
カスタムメッセージとハンドラかすたむめっせーじとはんどら 27
数える
ストリングの文字数 420
ハンドラに送信されたパラメータ 488
リストの項目 291, 804, 805
括弧（[]）695
カメラ
削除 307
新規 466
追加する 245
カメラオブジェクトかめらおぶじぇくと 154
カラーデータ型からーでーたがた 10
カラーパレットオブジェクトからーぱれっとおぶじぇくと 114
関数
括弧 7
定義 5

き

キー
BackSpace キー（Windows）164
Delete キー（Macintosh）164
Enter キー 165
lastEvent 関数 419
RETURN 文字定数 168
Tab キー 169
キーオブジェクトきーおぶじぇくと 94
キーフレーム
trackNextKeyTime プロパティ 1180
trackPreviousKeyTime プロパティ 1181
キーボードの文字、テスト 13
キーワード
Lingo 214
定義 6
キャスト、変更を保存するきゃすと、へんこうをほぞんする 585
キャストウィンドウ 4, 5
キャストメンバ
locToCharPos 関数 427
locVToLinePos 関数 427
pictureP 関数 501
Shockwave Audio 352, 354
関連付けられたパレット 1024, 1025
キャストメンバトラック内のメディア 1184
行 942
境界線 944
行間隔 943
継続時間 844
コピー 290
作成 463
テキストボックス 762
表示するフォント 884
プリロード 1054
キャストメンバの行間隔 943
キャストメンバの制御パネルの非表示 802
キャストメンバのボックス 762
キャストライブラリオブジェクトきゃすとらいぶらりおぶじぇくと 92
キャッシュ
Web ページをリフレッシュする 264
ブラウザでクリアする 277
ブラウザでのサイズの設定 265
キューポイント
時間のリスト 809
スプライト 414, 981
名前のリスト 809
行
距離 421
継続シンボル（\\）214
高さ 421

く

区切り文字
項目に対して定義する 925
コメント記号（--）686
矩形
inside 関数 406
intersect 関数 410
union rect 関数 647
位置指定とサイズ調整 430
オフセットを指定する 483
タイプ 397
定義 1064

クラス、JavaScript
カスタム 55
定義 60
クラス変数くらすへんすう 59
クラスメソッドくらすめそっど 60
繰り返しループ
シンタックス 24
クリップボード、キャストメンバのコピーくりっぷぼーど、きゃすとめんばのこぴー 290
グループ化演算子 () ぐるーぷかえんざんし () 688
グループオブジェクトぐるーぷおぶじぇくと 156
グローバル変数 230
グローバルオブジェクトぐろーばるおぶじぇくと 93
グローバルプロパティ、cpuHogTicks ぐろーばるぷろぱてぃ、cpuHogTicks 806
グローバル変数 14, 16

け

継続シンボル（\\）214
検索
スクリプト内のハンドラとテキスト 70
検索する
ファイル名 364
ムービー 364
減算、マイナス記号（-）685, 690

こ

コールスタックこーるすたっく 87
コアオブジェクトこあおぶじぇくと 42, 92
剛体の関数
applyAngularImpulse() 1266
子オブジェクト
ancestor プロパティ 719
C++ および Java との比較 46
アンセスタにメッセージを送る 268
オブジェクト指向プログラミング 46
基本的な概念 46
削除 51
作成 48, 49, 463
システムイベントの中継 53
スプライトへのビヘイビアーの追加 51
ビヘイビアー、比較 47
プロパティの確認 50
メッセージ 51
コピー
キャストメンバ 290
リスト 35, 38, 320
コメント 75
コメント記号（--）686
コメントこめんと 7, 68
小文字 9, 12
コンストラクタ関数、JavaScript こんすとらくたかんすう、JavaScript 56

さ

サーバー、プロキシ 532
サイズ
キャストメンバ 782
メモリの空きブロック 347
メンバのプロパティの chunkSize 782
メンバプロパティの lineSize 944
再生する
インターネットからの Shockwave ムービー 387
デジタルビデオトラック 612
サウンドオブジェクトさうんどおぶじぇくと 102, 127
サウンドチャネル 104
サウンドプロパティ
multiSound 996
スプライトの音量 1221
サウンドプロパティさうんどぷろぱてぃ 1118
削除
オブジェクト 85, 568
子オブジェクト 51
配列アイテム 37
ビヘイビアー 72
変数 61
リストの項目 35, 306, 311
作成
Xtra 拡張機能 463
キャストメンバ 463
矩形 1064
項目に対して定義する 925
子オブジェクト 49, 463
ビヘイビアー 71
サブディビジョンサーフェス（SDS）
プロパティ 1102
モディファイア 1102
サンプリング
trackNextSampleTime プロパティ 1180
trackPreviousSampleTime プロパティ 1181

し

シェーダ、Lingo しぇーだ、Lingo 1111
シェーダオブジェクトしぇーだおぶじぇくと 161
シェイプ
境界線 944
種類 1112
パターン 873

式
FALSE 式 166
item... of キーワード 221
シンボルとしての 637, 638
ストリングとしての 634
ストリングに変換する 633
整数としての 408
定義 5
評価 229, 230, 235, 315
浮動小数点数 342
浮動小数点数に変換する 342
論理否定 701
システムイベント、子オブジェクトへの中継 53
システムオブジェクトしすてむおぶじぇくと 108
システムプロパティ
floatPrecision 881
multiSound 996
実行順序
イベント、メッセージ、およびハンドラ 25
スクリプト 21
シャープ記号（#）しゃーぷきごう（#）683
順序、実行
イベント、メッセージ、およびハンドラ 25
スクリプト 21
条件
end case キーワード 217
if...then ステートメント 220
otherwise キーワード 227
repeat while キーワード 231
テストと設定 20
分岐
otherwise キーワード 227
乗算演算子（*）689
シンタックス
JavaScript 56
Lingo における大文字と小文字の区別 12
エラー 77
演算子 18
大文字と小文字の区別 9
オブジェクト指向プログラミング 45
規則 7
繰り返しループ 24
条件一致、式 23
シンボル 14
数値 12
ストリング 12
整数 12
定数 13
ドット 41
トラブルシューティング 76
配列 37
ハンドラ 27
リスト 30
シンボル
式 638
シンボル演算子（#）683
ストリング 637
定義 14
用途 14
シンボル演算子（#）しんぼるえんざんし（#）683

す

ズームハンドラずーむはんどら 213
数値
指数 519
システムでサポートされている最大値 961
小数 12
スプライトに割り当てられたスクリプト番号 1099
浮動小数点 12
浮動小数点数 342
スクリプト
Lingo と JavaScript 40
アンセスタ 47
一般的な項目の挿入 67
一般的な作業 71
インデント 69
オブジェクト指向プログラミング 45
親スクリプトによって作成されるオブジェクト 481
改行の挿入 69
記述 62, 75
コメント 7, 75
コメント記号（--）686
削除 72
種類 4
順序、実行 21
シンタックスの色分け 66
スプライトに添付された 611
スプライトに割り当てられたスクリプト番号 1099
タイプの変更 73
トラブルシューティング 75
ハンドラとテキストの検索 70
ハンドラを呼び出す 266
編集 69
ムービー 72
メッセージウィンドウ 81
リンク 73
リンクされた 422
リンクされたムービー 1100
スクリプト API、定義すくりぷと API、ていぎ 1
スクリプト、キャストメンバ
解説 5

スクリプト作成 Xtras すくりぷとさくせい Xtras 68
スクリプトウィンドウ 62, 75, 78
スクリプトオブジェクト 42, 132
スクリプトシンタックスメニューすくりぷとしんたっくすめにゅー 67
スクリプトの読み込みすくりぷとのよみこみ 73
スコア
　記録する 256
　更新する 256
　スプライトに割り当てられたカラー 1095
スコアを更新するすこあをこうしんする 256
スタティック変数すたてぃっくへんすう 59
スタティックメソッドすたてぃっくめそっど 60
ステートメント
　順序、実行 21
　定義 6
ステートメント、if...then...else すてーとめんと、if...then...else 220
ストリーミング、Lingo すとりーみんぐ、Lingo 1039
ストリーミングステータスハンドラすとりーみんぐすてーたすはんどら 209
ストリーミングのステータスレポート 638
ストリーミングのステータスレポート作成を有効にする 638
ストリーミングのステータスレポートを無効にするすとりーみんぐのすてーたすれぽーとをむこうにする 638
ストリング
　ASCII コード 275, 480
　char...of キーワード 215
　chars 関数 275
　do コマンド 315
　EMPTY 文字定数 165
　field キーワード 218
　integer 関数 407
　last 関数 418
　length 関数 420
　line...of キーワード 222
　offset 関数 482
　put...after コマンド 229
　put...before コマンド 230
　put...into コマンド 230
　starts 比較演算子 703
すとりんぐ
　行を数える 1010
　項目を数える 1009
ストリング
　最後に変更された日付 459
　式 634
　式の変換 633
　シンタックス 12
　数値 658
　選択する 593
　選択範囲の先頭の文字 1108
　チャンク式内の単語を数える 1013
　二重引用符と 164
　ハイライトする 391
　比較 698
　ファイルに書き込む 608
　フィールドキャストメンバ内 1155
　ブラウザウィンドウに表示する 461
　ブラウザに送信する 326
ストリング演算子 21
ストリング定数（"）すとりんぐていすう（"）164
ストリングデータ型すとりんぐでーたがた 11
スプライト
　on beginSprite ハンドラ 172
　on endSprite ハンドラ 179
　on isOKToAttach ハンドラ 188
　位置 236, 237
　移動 995
　軌跡効果 1186
　キューポイント 414, 981
　伸縮する 633
　スプライトチャネルでのムービーの開始時間 1182
　デジタルビデオスプライトトラック内のメディア 1185
　デジタルビデオスプライトのトラック数を数える 1178
　添付されたスクリプト 611
　トラックの停止時間 1183
　トラックを再生する 1179
　ビヘイビアー 72
　ビヘイビアーの動的な追加 51
　ビヘイビアーの割り当て 72
　表示 1217
　割り当てられたスクリプト番号 1099
スプライトオブジェクトすぷらいとおぶじぇくと 105, 162
スプライトチャネルオブジェクトすぷらいとちゃねるおぶじぇくと 107
スプライトの移動すぷらいとのいどう 995
スプライトの伸縮 633
スプライトメソッドの巻き戻しすぷらいと めそっどのまきもどし 579
スペースすぺーす 9
スラッシュ（/）すらっしゅ（/）691, 692
スラッシュ記号（/）すらっしゅきごう（/）691, 692

せ

整数
　maxInteger プロパティ 961
　ランダム 544
セミコロンせみころん 9
線
　キャストメンバ内 942
センターポイント 1071

た

高さ
lineHeight 関数 421
フィールドキャストメンバの行 421
タブ順、メンバのプロパティの autoTab たぶじゅん、めんばのぷろぱてぃの autoTab 738

ち

地形のプロパティ
contactTolerance 1290
heightscale 1290
orientation 1291
position 1291
rowscale 1292
terraindesc 1292
地形のメソッド
createTerrain 1287
deleteTerrain 1288
チック
lastClick 関数 419
movieTime of sprite プロパティ 813
チャンク式内の単語 1013
チャンク式
char...of キーワード 215
field キーワード 218
item... of キーワード 221
last 関数 418
line...of キーワード 222
put...after コマンド 229
put...before コマンド 230
put...into コマンド 230
word...of キーワード 238
行を数える 1010
項目を数える 1009
単語の選択 238
単語を数える 1013
ハイライトする 391
ブラケットアクセス演算子 695

つ

追加
スプライトへのビヘイビアーの 51
配列アイテム 37
リストアイテム 35
追加する
プロパティリストに 249
リニアリストに 242, 249, 252

て

データ型でーたがた 10
定義
エレメント 5
オブジェクト指向プログラミング 46, 55
定数
BACKSPACE 164
EMPTY 165
ENTER 165
FALSE 166
PI 167
QUOTE 167
RETURN 168
SPACE 168
TAB 169
TRUE 169
VOID 170
シンタックス 8, 13
定義 5
テキスト
ASCII コード 275, 480
char...of キーワード 215
charPosToLoc 関数 274
chars 関数 275
do コマンド 315
EMPTY 文字定数 165
field キーワード 218
last 関数 418
length 関数 420
line...of キーワード 222
offset 関数 482
put...after コマンド 229
put...before コマンド 230
put...into コマンド 230
starts 比較演算子 703
行を数える 1010
項目を数える 1009
式のストリングへの変換 633
スクリプト内で検索 70
スクリプトの編集 69
ストリングとしての式 634
ストリングの数値 658
ストリングの比較 698
選択 593
選択範囲の先頭の文字 1108
単語の選択 238
チャンク式内の単語を数える 1013
ネットワークサーバ上のファイルから読み取る 362
ネットワーク操作により返される 462
ハイライトする 391

フィールドキャストメンバ内のストリング 1155
連結演算子（& または &&）686, 687
テキストオブジェクトてきすとおぶじぇくと 128
テクスチャ 441, 1156
テクスチャオブジェクトてくすちゃおぶじぇくと 163
デジタルビデオ
継続時間 844
トラックを再生する 612
フォーマット 830
デジタルビデオキャストメンバ
再生をオンまたはオフにする 1210, 1217
トラック数を数える 1178
トラック内のメディア 1184
トラックの開始時間 1182
トラックの停止時間 1183
メモリにプリロードする 1054
デジタルビデオキャストメンバプロパティ
trackStartTime （メンバ）1182
trackStopTime（メンバ）1183
trackType（メンバ）1184
スプライトの音量 1221
メンバの digitalVideoType 830
メンバの frameRate 889
メンバの loop 951
メンバの制御パネル 802
メンバの中心 773
メンバのビデオ 1210, 1217
デジタルビデオスプライト
トラック数を数える 1178
トラック内のテキスト 1184
トラック内のメディア 1185
トラックを再生する 1179
デジタルビデオスプライトプロパティ
trackNextKeyTime 1180
trackPreviousKeyTime 1181
trackPreviousSampleTime 1181
trackText 1184
trackType （スプライト）1185
デバッガウィンドウ 75, 86
デバッグ
1 行ずつ 89
Shockwave ムービーおよびプロジェクタ 90
オブジェクト 88
オブジェクトインスペクタ、使用 83
高度な方法 91
シンタックスエラー 77
スクリプトウィンドウ、使用 78
説明 76
デバッガウィンドウ、使用 86
メッセージウィンドウ、使用 79
点
タイプ 397

と

トゥイーンモード 1192
ドキュメント、アプリケーションで開く 484
ドキュメントどきゅめんと 2
ドット演算子（.）どっとえんざんし（.）684
ドットシンタックス 41
トラック、再生する 612
トラブルシューティング
1 行ずつ 89
Shockwave ムービーおよびプロジェクタ 90
オブジェクト 88
オブジェクトインスペクタ、使用 83
高度な方法 91
シンタックスエラー 77
スクリプトウィンドウ、使用 78
スクリプトのステップ実行 90
説明 75
ツール 75
デバッガウィンドウ 86
デバッグ 76
メッセージウィンドウ、使用 79
トランジション
種類 1188
メンバのプロパティの transitionType 1188
トランジションキャストメンバ
継続時間 844
プロパティ、chunkSize of member 782
トランスフォーム
反転または逆転する 411
角度 738

な

並べ替えリスト 36, 38
ナンバー記号（#）なんばーきごう（#）683

ぬ

ヌルデータ型ぬるでーたがた 11

ね

ネットワーク操作をキャンセルする 455
ネットワークサーバー、サーバー上のファイルからテキストを読み取る 362
ネットワークサーバー上のファイルからテキストを読み取る 362
ネットワーク操作
MIME タイプ 460
エラー 457

返されるテキスト 462
キャンセルする 455

の

ノードの削除のーどのさくじょ 307

は

パーティクルシステム
distribution 838
gravity 898
ハードウェア情報、取得 358
ハイパーリンク、ハンドラはいぱーりんく、はんどら 187
ハイライトされたテキスト 391
バイリニアフィルタリング 1003
場所
キャストメンバの 274
スプライト 236, 237
デスクトップ上のステージ 620, 621, 622, 623
パス、ブラウザぱす、ぶらうざ 261
パス名 697
パス名演算子（@）ぱすめいえんざんし（@）697
パターン、シェイプキャストメンバを塗りつぶす 873
バックグラウンドイベント、処理 806
パラメータ
on getPropertyDescriptionList ハンドラ 186
on runPropertyDialog ハンドラ 202
シンタックス 7
定義 6
ハンドラ 28
ハンドラに送信されたパラメータを数える 488
リストの 487
パレット
キャストメンバに割り当てる 1024, 1025
スプライトに割り当てられたスコアカラー 1095
パレットをキャストメンバに割り当てるぱれっとをきゃすとめんばにわりあてる 1024, 1025
ハンドラ
ancestor プロパティ 719
on キーワード 227
result 関数 575
return キーワード 234
親スクリプトへの追加 48
カスタム 27
括弧 7
結果 29
検索 70
コールスタック 87
終了 217, 239
終了を示す 216
順序、実行 25
送信されたパラメータをカウントする 488
定義 5
デバッグ 89
配置 27
パラメータ 28
メッセージ、受信 28
呼び出す 266
リスト 171
ハンドラの終了を示す 216

ひ

比較演算子
contains 698
sprite...within 237
starts 703
以下（\ 693
以上（\>=）694
代入演算子（=）693
等しくない（\）693
より大きい（\>）694
より小さい（\ 692
ビットマップ
メンバのプロパティの picture 501
余白のトリミング 1191
ビットマップオブジェクトびっとまっぷおぶじぇくと 113
ビットマップキャストメンバ
色深度 826
関連付けられたパレット 1025
ビヘイビアー
on getBehaviorDescription ハンドラ 184
on getBehaviorTooltip 185
on getPropertyDescriptionList ハンドラ 186
on isOKToAttach ハンドラ 188
on runPropertyDialog ハンドラ 202
関連付け 72
子オブジェクトとの比較 47
削除 72
スプライト 72
スプライトへの動的な追加 51
定義び 4
フレーム 71
編集 4, 72
ビヘイビアーのスプライトへの割り当てびへいびあのすぷらいとへのわりあて 188
開く
MIME ファイル 388
Shockwave ムービー 388
アプリケーション 484

ふ

ブールデータ型ぶーるでーたがた 10
ファイル
　インターネットからプリロードする 317, 524
　ストリングを書き込む 608
ファイルから
　ネットワーク経由でテキストを読み取る 362
ファイル名 MP4Media/FLV プロパティ 868
フィールドオブジェクトふぃーるどおぶじぇくと 117
フィールドキャストメンバ
　field キーワード 218
　lineHeight 関数 421
　locToCharPos 関数 427
　locVToLinePos 関数 427
　行の位置 421
　行の高さ 421
　スクロール 588, 589, 1102
　ストリング 1155
　定義されているメニューをインストールする 406
　テキストボックス 762
　メンバプロパティの lineHeight 943
フィールドキャストメンバ、スクロールする 588, 589, 1102
フィールドキャストメンバで定義されたメニューのインストール 406
フィールドキャストメンバのプロパティ
　メンバの autoTab 738
　メンバの boxDropShadow 762
　メンバの lineCount 942
　メンバの margin 959
　メンバのプロパティの wordWrap 1228
フィールドキャストメンバのボックスふぃーるどきゃすとめんばのぼっくす 959
フィールドプロパティ
　selStart 1108
　メンバの font 884
　メンバの fontSize 884
　メンバの fontStyle 886
　メンバの lineHeight 943
　メンバを揃える 715
フィルムループオブジェクトふぃるむるーぷおぶじぇくと 118
フォーマット、スクリプト 69
フォント 884, 886
フォントオブジェクトふぉんとおぶじぇくと 121
ふぉんとさいず] 884
ふぉんとすたいる] 886
浮動小数点数 12
ブラウザ
　インターネットからファイルをプリロードする 524
　キャッシュサイズの設定 265
　キャッシュをクリアする 277
　ストリングを送信する 326
　ストリングを表示する 461
　場所 261
ブラウザにストリングを送信する 326
ブラケットアクセス演算子（[]）695
プラス記号（+）689, 690
フレーム
　framesToHMS 関数 345
　HMStoFrames 関数 393
　on enterFrame ハンドラ 180
　on exitFrame ハンドラ 183
　on prepareFrame ハンドラ 199
　on stepFrame ハンドラ 207
　再生時間に変換する 345
　再生時間を変換する 393
　ビヘイビアーの作成 71
　ビヘイビアーの割り当て 72
　リスト、フレームラベル 933
フレームプロパティ
　trackNextKeyTime 1180
　trackPreviousKeyTime 1181
　メンバの frameRate 889
プレーヤオブジェクトぷれーやおぶじぇくと 100
ブレンド範囲、開始点と終了点 753
プロキシサーバー、値を設定する 532
プロジェクタ、デバッグぷろじぇくた、でばっぐ 90
プロトタイプオブジェクトぷろとたいぷおぶじぇくと 58, 61
プロパティ
　オブジェクトインスペクタ 85
　シンタックス 11
　定義 6
　リスト 705
プロパティ変数 230
プロパティ変数、宣言 48
プロパティリスト
　findPos コマンド 339
　findPosNear コマンド 340
　setaProp コマンド 597
　setProp コマンド 610
　値を削除 311
　シンタックス 30
　追加する 249
　プロパティの値を置き換える 597, 610
　プロパティの位置 339, 340
フロントスラッシュ（/）ふろんとすらっしゅ（/）691, 692
分岐
　end case キーワード 217
　if...then ステートメント 220
　repeat while キーワード 231

へ

ベクトルシェイプオブジェクトべくたーしぇいぷおぶじぇくと 129
ベクトルデータ型べくとるでーたがた 11
変換する
ASCII コードを文字に 480
再生時間をフレームに 393
浮動小数点数に変換する式 342
フレームを再生時間に 345
文字を ASCII コードに 275
編集
親スクリプト 72
スクリプト 69
デバッガモード 90
ビヘイビアー 72
変数
global 219
me 49
property キーワード 227
voidP プロパティ 671
値、格納と更新 15
インスタンス 58
クラス 59
グローバル 16
コールスタック 87
削除 61
種類 14
シンタックス 8
データ型 10
定義 7
デバッガウィンドウ 87
名前の設定 76
ローカル 17
ローカル変数 614

ほ

ポイント
位置指定とサイズ調整 430
識別する 423
ポイントデータ型ぽいんとでーたがた 11
保存
ムービー 1200
保存する
キャストへの変更 585
ボタンオブジェクトぼたんおぶじぇくと 114

ま

マーカ
loop キーワード 223
next キーワード 226
マイナス記号（-）（3D）まいなすきごう（-）（3D）691
マイナス記号（-）685, 690
マウスオブジェクトまうすおぶじぇくと 97
マウスクリック
lastClick 関数 419
lastEvent 関数 419
マウスハンドラ
on mouseDown 192
on mouseEnter 193
on mouseLeave 194
on mouseUp 195
on mouseUpOutside 197
on mouseWithin 197
on rightMouseDown 202
on rightMouseUp 202
trayIconMouseDown 211
trayIconRightMouseDown 212

み

ミップマッピング 1062

む

ムービー
on prepareMovie ハンドラ 200
on startMovie ハンドラ 206
on stopMovie ハンドラ 208
Shockwave 387
検索する 364
時間内の長さ 813
実行モード 583
使用可能な Xtra 拡張 1014
保存 1200
メニューの数 1014
ムービーオブジェクトむーびーおぶじぇくと 98
ムービースクリプト 4, 72
ムービープロパティ
updateMovieEnabled 1200
videoForWindowsPresent 1212
メンバの scriptsEnabled 1100
メンバの制御パネル 802
メンバの中心 773

め

メソッド
インスタンス 59
括弧 7
クラス 60
定義 6
ハンドラの終了を示す 216
リスト 239

メッシュカラーめっしゅからー 794
メッセージ
childxd5 のアンセスタに送る 268
イベント 171
エラー 352, 354
カスタム 27
子オブジェクト 51
順序、実行 25
定義 6
ハンドラ受信 28
ハンドラを呼び出す 266
渡す 492
メッセージウィンドウ 75, 79
メディアタイプのオブジェクトめでぃあたいぷのおぶじぇくと 42, 112
メニュー
name of menu プロパティ 997
インストールする 406
現行ムービー 1014
メニュー項目
選択 851
テキストを設定する 998
メニュー項目プロパティ
enabled of menuItem 851
name of menuItem 998
メニュープロパティ
メニューの数 1014
メニューの名前 997
メモリ
空き 347, 348, 970
空きブロックのサイズ 347
キャストメンバをプリロードする 1054
プログラムに割り当てられている 970
メンバオブジェクトめんばおぶじぇくと 95, 157

も

モーションオブジェクトもーしょんおぶじぇくと 160
文字ストリング
ASCII コード 275, 480
char...of キーワード 215
chars 関数 275
do コマンド 315
EMPTY 文字定数 165
field キーワード 218
integer 関数 407
last 関数 418
length 関数 420
line...of キーワード 222
offset 関数 482
put...after コマンド 229
put...before コマンド 230
put...into コマンド 230
starts 比較演算子 703
文字すとりんぐ
行を数える 1010
文字ストリング
項目を数える 1009
式 634
式の変換 633
数値 658
選択する 593
選択範囲の先頭の文字 1108
単語の選択 238
チャンク式内の単語を数える 1013
ハイライトする 391
比較 698
フィールドキャストメンバ内 1155
文字定数
BACKSPACE 164
EMPTY 165
ENTER 165
QUOTE 167
RETURN 168
TAB 169
モディファイア
サブディビジョンサーフェス 1102
レベルオブディテール（LOD）949
モデルオブジェクトもでるおぶじぇくと 159
モデルリソースオブジェクトもでるりそーすおぶじぇくと 160

よ

用語
オブジェクト指向プログラミング 46, 55
用語集
エレメント 5
より小さい (\ 692
より大きい (\>) 694

ら

ライト
attenuation プロパティ 728
指向性ライト 832
周囲ライト 828
ライトオブジェクトらいとおぶじぇくと 157
ラベル 933
ランダムアクセス演算子らんだむあくせすえんざんし 703
ランダムな整数 544

り

リスト
count 関数 291, 804, 805
findPos コマンド 339
findPosNear コマンド 340
getaProp コマンド 349
getAt コマンド 350
getLast コマンド 360
getOne コマンド 366
getPos コマンド 370
getProp コマンド 373
getPropAt コマンド 374
ilk 関数 397
setaProp コマンド 597
setAt コマンド 598
setProp コマンド 610
確認、内容 34, 37
キューポイント時間 809
キューポイント名 809
項目数を数える 291
項目または値を削除 306, 311
項目を数える 804, 805
項目を識別する 349, 350, 360, 366, 370, 373, 374
コピー 35, 38, 320
最小値 442
最大値 436
削除、アイテム 37
シンタックス 30
設定と取得、アイテム 32
タイプ 397, 423
多次元 36, 39
追加、アイテム 35, 37
追加する 252
定義 6
並べ替え 36, 38
並べ替える 617
パラメータの値 487
プロパティの値を置き換える 597, 610
プロパティリストでのプロパティの位置 339, 340
プロパティリストに追加する 249
リスト演算子 695
リニアリストに追加する 242, 249
リスト、フレームラベル 933
リスト演算子（[]）695
リストデータ型りすとでーたがた 11
リストブラケット（[]）695
リソースファイル、開くりそーすふぁいる、ひらく 486
リテラル二重引用符（"）りてらるにじゅういんようふ（"）167
リテラル値
解説 12
小数と浮動小数点数 12
ストリング 12
整数 12
リニアリスト
値を削除 311
シンタックス 30
追加する 242, 249, 252
リンク
スクリプト 73, 422
ムービー 1100
リンクされたムービーオブジェクトりんくされたむーびーおぶじぇくと 122

る

ループ
loop キーワード 223
next repeat キーワード 226
repeat with キーワード 232
repeat with...down to キーワード 233
repeat with...in list キーワード 234
終了 218
シンタックス 24

れ

レイキャスティングのメソッド
raycastClosest 1311
レベルオブディテール（LOD）モディファイアのプロパティれべるおぶでぃてーる（LOD）もでぃふぁいあのぷろぱてぃ 949
連結演算子（& または &&）686, 687
レンダラサービスオブジェクトれんだらさーびすおぶじぇくと 161
レンダリング 710, 1072

ろ

ローカル変数 230, 614, 14, 17
論理定数
FALSE 166
TRUE 169

わ

ワールド単位 1023