Method	Description	Returns
`rotate (position, axis, angle, relativeTo)`	Rotates the model around the axis vector in the specified position the specified number of degrees. The *relativeTo* argument is optional and defaults to `#self`.	Nothing
`scale(uniformScale)`	Scales the model the same amount in all directions.	Nothing
`scale(x, y, z)`	Scales the model by a factor of *x* in the *x* dimension, *y* in the *y* dimension, and *z* in the *z* dimension. Scaling is applied in object-relative space.	Nothing
`pointAt(worldPosition, worldUp)`	Points the node's "front" at the world position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *pointAtOrientation* property. Both the object-relative axes are defined by the *pointAtOrientation* property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this model into world-relative positions.	A transform object

Moving models

Because the 3D world has no absolute frame of reference, moving and rotating is much more complex than in 2D, where all movement is in relation to screen position.

In 3D, everything is drawn relative to the camera's frame of reference. If the camera is behind an object, when the object moves to the left relative to the center of the world, or *world origin*, it appears to move toward the right of the screen.

Each piece of position and orientation information can be expressed relative to one or more frames of reference. A model's transform property, for instance, expresses its position and rotation relative to the model's parent. In general, there are four frames of reference to consider: relative to the object (model, light, camera) itself, relative to the object's parent, relative to the world, and relative to some other object.

- Object-relative: When you create a model in a 3D modeling program, you build it relative to its own frame of reference. For instance, when you create a model of a car, the front of the car may be pointed along its *z*-axis and the antenna may be pointed along its *y*-axis. To move such a car forward (along its *z*-axis) regardless of which direction it is pointing relative to the camera or the world, use `car.translate(0,0,10)`. To turn the car left, use `car.rotate(0,45,0)`.

  The car model might have wheel models as children. To rotate the wheel of a car relative to itself, rather than relative to its parent (the car), use the following script:

  ```
  wheel.rotate(0,10,0)
  ```


Downloaded from www.Manualslib.com manuals search engine

or

```
car.child[1].rotate(0,10,0, #self)
```

where the fourth parameter of the `rotate` method is the object the rotation should be relative to.

- Parent-relative: A model's `transform` property expresses its position and rotation relative to the model's parent. If you want the wheels of the car to move outward regardless of how the wheels are turned, use `car.child[1].translate(10,0,0,#parent)` or `car.child[1].transform.translate(10,0,0)`. If you want a planet model that is a child of the sun to orbit around the sun, use `planet.rotate(0,5,0, #parent)`.
- World-relative: If you want the car to move along the world's *x*-axis regardless of which way it is facing, use `model.translate(10,0,0,#world)`. If you want to rotate the car 20° around the world *y*-axis, with the rotation taking place at the world location vector (10, 10, 10), use `model.rotate(vector(10,10,10), vector(0,1,0), 20, #world)`.
- Relative to another object: If you want to move an object so that it goes toward the right edge of the screen, use `model.translate (vector(10,0,0), sprite(1).camera)`. If you want to rotate the object parallel to the camera and around the center of the screen, use `model.rotate(vector(0,0,0), vector(0,0,1), 20, sprite(1).camera)`.

Shaders

A model resource defines a model's shape, and shaders define the model's surface colors and reflectivity. You can use just one shader or more than one. Each mesh in a model resource can have its own shader. For example, a box might have six different shaders, one for each mesh (a box is actually composed of 6 plane meshes carefully arranged). If you do not specify a shader, the default `#standard` shader is used. If the shader's properties are modified, the change affects all models that use that shader.

Models that are created with script are assigned the standard shader. You can replace the default shader of a model with any of the other types of shaders.

Properties of the standard shader

The standard shader makes the surface of a model appear in a photorealistic style. Use these properties to work with the standard shader:

Property Name	Access	Description	Default
`name`	Get	The string name of this shader.	None
`ambient`	Get and set	A color object describing the surface's reaction to ambient light.	`color(63,63, 63)`
`diffuse`	Get and set	A color object describing the surface's reaction to diffuse light. Ambient and diffuse color objects together describe a model resource's base color.	`color(255, 255,255)`
`specular`	Get and set	A color object describing the surface's specular highlight color. This setting has an effect only if there are lights in the scene whose specular property is `TRUE (1)`.	`color(255, 255,255)`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
shininess	Get and set	An integer between 0 and 100 indicating how shiny a surface is.	30.0
emissive	Get and set	A color object describing the color of light this object seems to give off. This does not turn the surface using this shader into a light source; it just gives it the appearance of being one.	color(0,0,0)
blend	Get and set	An integer between 0 and 100 indicating how transparent (0) or opaque (100) this surfaces is. Unlike with a texture that includes alpha information, this setting affects the entire surface uniformly.	100
transparent	Get and set	This property controls whether or not the model is blended using alpha values or rendered as opaque. The default is TRUE (1) (alpha blended). The functionality of shader.blend is dependent on shader.transparent.	TRUE (1)
renderStyle	Get and set	This property can take the following values: #fill #wire #point When shader.renderStyle = #fill, the polygons of the mesh are filled. When shader.renderStyle = #wire, the polygon edges of the mesh are rendered. When shader.renderStyle = #point, the vertices of the mesh are rendered, provided that #Fill is supported by the #software renderer. The #point and #wire values do not work with the software renederer.	#fill
flat	Get and set	When shader.flat = TRUE (1), the mesh should be rendered with flat shading instead of Gouraud shading, which shades each polygon separately. Flat shading shades the mesh as a whole.	FALSE (0)
textureList	Get and set	A shader can use up to eight layers of textures. This eight-element list defines which texture is used for which layer. Get: Returns a list of texture objects, one per layer. Set: Specifies a texture object to be applied to all layers. An argument of void disables texturing for all layers.	void

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
`textureList [index]`	Get and set	A shader can use up to eight layers of textures. This property gives access to the texture at the indicated index position.	`void`
`texture`	Get and set	This property allows access to the texture for the first layer. It is equivalent to `textureList[1]`. An argument of `void` can be used to disable texturing for the first layer.	`void`

`reflectionMap`	Get and set	Get: Returns the texture associated with the third layer.
		Set: Specifies a texture to be used in the third layer and applies the following values:
		`textureModeList[3] = #reflection`
		`blendFunctionList[3] = #blend`
		`blendSourceList[3] = #constant`
		`blendConstantList[3] = 50.0`
			`void`

diffuseLightMap	Get and set	Get: Returns the texture associated with the second layer.
		Set: Specifies a texture to be used in the second layer and applies the following values:
		`textureModeList[2] = #diffuse`
		`blendFunctionList[2] = #multiply`
		`blendFunctionList[1] = #replace`
			`void`

`specularLight Map`	Get and set	Get: Returns the texture associated with the fifth layer.
		Set: Specifies a texture to be used in the fifth layer and applies the following values:
		`textureModeList[5] = #specular`
		`blendFunctionList[5] = #add`
		`blendFunctionList[1] = #replace`
			`void`

`glossMap`	Get and set	Get: Returns the texture associated with the fourth layer.
		Set: Specifies a texture to be used in the fourth layer and applies the following values:
		`textureModeList[4] = #none`
		`blendFunctionList[4] = #multiply`
			`void`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default

`textureMode List[index]`	Get and set	This property allows access to the texture coordinate generation method used for a texture at the texture level and then to allows you to change how textures are applied to a model's surface. The property can take the following values:
		`#none`
		`#wrapPlanar`
		`#wrapCylindrical`
		`#wrapSpherical`
		`#reflection`
		`#diffuseLight`
		`#specularLight`
			`#none`

`textureMode List`	Get and set	Get: Returns a list of texture coordinate generation methods, one per layer.
		Set: Specifies texture coordinate generation modes to be applied to all layers.
		Possible values are as follows:
		`#none`
		`#wrapPlanar`
		`#wrapCylindrical`
		`#wrapSpherical`
		`#reflection`
		`#diffuseLight`
		`#specularLight`
			`#none`

`textureMode`	Get and set	Access to the texture coordinate generation method for the first layer.
		Possible values are as follows:
		`#none`
		`#wrapPlanar`
		`#wrapCylindrical`
		`#wrapSpherical`
		`#reflection`
		`#diffuseLight`
		`#specularLight`
			`#none`

`wrapTransform List[index]`	Get and set	Access to the texture coordinate generation transform associated with a specified layer. This transformation has effect only if the `textureModeList[index]` is `#wrapPlanar`, `#wrapSpherical`, or `#wrapCylindrical`. Controls the orientation of texture coordinate generation in model-relative space. Use this property to change the orientation, offset, and scale of how the `wrapTransformList[index]` is applied on the model.	`transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000, 50.0000,0.0000, 0.0000,0.0000, 0.0000,1.0000)`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
wrapTransform List	Get and set	Controls the orientation of UV generation in model space. Get: Returns a list of texture coordinate generation transforms, one per layer. Set: Specifies a texture coordinate generation transform to be applied to all layers.	transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000,50.0000,0.0000, 0.0000,0.0000,0.0000,1.0000)
wrapTransform	Get and set	Access to the texture coordinate generation transform for the first layer. Controls the orientation of the UV generation in model space.	transform(50.0000,0.0000,0.0000,0.0000, 0.0000,50.0000,0.0000,0.0000, 0.0000,0.0000,50.0000,0.0000, 0.0000,0.0000,0.0000,1.0000)
texture TransformList	Get and set	Access to the list of texture coordinate modifier transforms, one per texturing layer. The textureTransform is applied to all texture coordinates regardless of the textureMode property setting. This is the last modification of the texture coordinates before they are sent to the renderer. Allows you to manipulate the scale, orientation, and positional offsets of the source image before it's wrapped. WrapTransformList changes the projection of the transformed texture. The textureTransform matrix operates on the texture in textureImage space, which is defined to exist only on the x,y plane. Rotations about the *z*-axis are rotated around the (0,0) point, which maps to the upper left corner of the texture. Translating by integers when textureRepeat is TRUE (1) has no effect, because the width and height of the textures are defined to be 1.0 in textureImage space. Care must be taken not to scale any dimension (even *z*) by 0.	Identity transform
textureTransform List[*index*]	Get and set	Access to the texture coordinate modifier transform associated with a specified layer.	Identity transform
texture Transform	Get and set	Access to the texture coordinate modifier transform for the first layer.	Identity transform

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default

`blendFunctionList[index]`	Get and set	Access to the blending method associated with a texture layer at the position indicated by *index*, which must be a positive integer smaller than or equal to 8.
		Possible values are as follows:
		`#replace`
		`#multiply`
		`#add`
		`#blend`
		`#alpha`
		`#constant`
		For detailed information about all of these options, see the Scripting Reference topics in the Director Help Panel.
			`#multiply`

`blendFunction`	Get and set	Access to the list of blending methods, `#multiply`, `#replace`, `#blend`, and `#add`, for the first layer.	`#multiply`
`blendFunctionList`	Get and set	Access to the list of blending methods, `#multiply`, `#replace`, `#blend`, and `#add`, for all layers.	`#multiply`
`blendSourceList[index]`	Get and set	Access to the blending source associated with a specified layer. When the `blendFunction` property is set to `#blend` for the ‹index›th layer, this results in the ‹index›th texture being combined with the result of the previous layers for the entire texture using a single blending ratio. The blending ratio, in this case, is the value of `blendConstant` for layer ‹*index*›. For example, if the layer at that index position's `blendConstant` value is 0.9, the resultant texture will be 90% of the texture at that index position and 10% of the result of the previous texture layers Possible values are `#constant` and `#alpha`.	`#constant`
`blendSourceList`	Get and set	Access to the blending sources for each layer, providing that the blend method is set to `#blend`. Possible values are `#constant` and `#alpha`.	`#constant`
`blendSource`	Get and set	Access to the blending sources for the first layer, providing that the blend method is set to `#blend`. Possible values are `#constant` and `#alpha`.	`#constant`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
`blendConstantList[index]`	Get and set	The blending ratio used for a specific layer when the blend method is set to `#blend` and `blendSourceList[index]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`blendConstantList`	Get and set	The blending ratio used for any layer when the blend method is set to `#blend` and `blendSourceList[index]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`blendConstant`	Get and set	The blending ratio used for the first layer when the blend method is set to `#blend` and `blendSourceList[index]` is set to `#constant`. Returns a floating-point value from `0.0` to `100.0`.	`50.0`
`textureRepeatList[index]`	Get and set	Allows you to get or set the texture clamping behavior associated with a specified layer. Texture clamping refers to how a texture "clamps" to its shader. If the ratio of the texture to the shader is less than 1 to 1 and `textureRepeatList` is set to `TRUE (1)`, the texture tiles over the shader. If `textureRepeatList` is set to `FALSE (0)`, the texture isn't repeated but appears only once in one part of the shader. If the ratio of the texture to the shader is greater than 1 to 1 and `textureRepeatList` is set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range.	`TRUE (1)`
`textureRepeatList`	Get and set	Access to the list of texture clamping behaviors, one per layer. When set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range. Get: Returns a list of texture clamping behaviors, one per layer. Set: Specifies a texture clamping behavior to be applied to all layers.	`TRUE (1)`
`textureRepeat`	Get and set	Access to the texture clamping behavior for the first layer. When set to `FALSE (0)`, the border of the texture is extended past the unit UV coordinate range.	`TRUE (1)`

Downloaded from www.Manualslib.com manuals search engine

Properties of the painter shader

The #painter shader gives the model a painted effect. Use these properties to work with the painter shader:

Property	Access	Description	Default
name	Get and set	Name of shader	None
style	Get and set	Possible values: #toon: sharp transitions between available colors #gradient: smooth transitions between available colors #blackAndWhite: sharp transitions between black and white	#gradient
colorSteps	Get and set	Number of color steps used for lighting calculations	2
shadow Percentage	Get and set	Percentage of lighting intensity that is the threshold between highlight and shadow	50
highlight Percentage	Get and set	Percentage of lighting steps to be treated as highlight	50
shadowStrength	Get and set	Factor controlling darkness of shadowed areas	1.0
highlight Strength	Get and set	Factor controlling brightness of highlighted areas	1.0

Properties of the newsprint shader

The #newsprint shader creates a dithering effect similar to a newspaper photograph. Use these properties to work with the newsprint shader:

Property	Access	Description	Default
name	Get and set	Name of shader	None
brightness	Get and set	Value controlling amount of white in shader	0.0
density	Get and set	Value controlling density of dots in newsprint image	45.0

Downloaded from www.Manualslib.com manuals search engine

Properties of the engraver shader

The #engraver shader gives the effect of an engraved metal surface. You can control the size and number of etched lines by adjusting the brightness and density properties, respectively. Use these properties to work with the engraver shader:

Property	Access	Description	Default
name	Get and set	Name of shader	None
brightness	Get and set	Value controlling amount of white in shader	0.0
density	Get and set	Value controlling number of lines used to shade an area	40.0
rotation	Get and set	Angle describing 2D rotational offset for engraving lines	0.0

Textures

Each shader can have textures applied to it. Textures are 2D images drawn on the surface of a model. The appearance of the model's surface is the combined effect of the shader and textures applied to it. If you do not specify a texture, a default red-and-white bitmap is used.

The pixel height and width of the 2D images you use as textures should be powers of 2 (that is, 2, 4, 8, 16, 32, and so on). This is because most video cards scale images to powers of 2. If the image used does not have pixel dimensions that are a power of 2 (values including 2, 4, 8, 16, etc.), there will be a decrease in both rendering performance and visual quality. In addition, all the textures used in a 3D scene must be able to fit in the computer's video RAM at the same time. If not, Director switches to software rendering, which slows performance.

Be aware of the limitations of your video RAM and that of your intended audience. Some video cards have as little as 4 megabytes of video RAM. Carefully budget your total texture size when designing your 3D world.

Texture properties

Use these properties to work with textures:

Property	Access	Description	Default
name	Get and set	Name of texture.	None
type	Get	Possible values: #fromfile: bitmap defined as part of 3D import #fromCastmember: bitmap derived from Director cast member #fromImageObject: the texture was created from a Director image object	None
member	Get and set	If the type is #castmember, this property identifies the source of the bitmap. If the type if #fromfile, this property is void.	None

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
`width`	Get	Width, in pixels.	None
`height`	Get	Height, in pixels.	None
`quality`	Get and set	Property with the following possible values: `#low`: texture is not mipmapped `#medium`: mipmapping is at a low bilinear setting (default) `#high`: the mipmapping is at a high trilinear setting For more information, see the Scripting Reference topics in the Director Help Panel.	`#medium`
`nearFiltering`	Get and set	Determines whether bilinear filtering is used when rendering a projected texture map that covers more screen space than the original. For more information, see the Scripting Reference topics in the Director Help Panel.	`TRUE (1)`

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
compressed	Get and set	The property can be TRUE (1) or FALSE (0): TRUE (1): the texture is compressed FALSE (0): the texture is not compressed The value changes automatically from TRUE (1) to FALSE (0) when the texture is to be rendered. The value can be set to FALSE (0) to decompress or to TRUE (1) to remove the decompressed representation from memory.	TRUE (1)
renderFormat	Get and set	This property determines how many bits are used to render the texture. It takes the following values: #default: the texure is rendered based on the value of the getRendererServices().textureRenderFormat property. #rgbaWXYZ: renders the texture using W bits for the red channel, X bits for the green channel, Y bits for the blue channel and Z bits for the alpha channel. This is available using the following possible combinations: #rgba8888 #rgba8880 #rgba5650 #rgba5550 #rgba5551 #rgba4444	#default

Texture methods

The pixel height and width of the 2D images you use as textures should be powers of 2 (that is, 2, 4, 8, 16, 32, and so on). If not, the image will be resized to a dimension that is a power of 2. The scaleDown() method lets you retain manual control over this procedure at the texture level.

Method	Description	Returns
scaleDown()	Reduces the height of the texture to the next lowest power of 2. This is useful for dynamically resizing textures to fit on a client machine.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Groups

Groups have many of the same properties and methods as models, except that you need to substitute the word `group` for the word `model` when writing scripts. A group is a node that can have a parent and/or children. These can be models, lights, cameras, or other groups.

Group properties

The properties of the group determine its particular appearance and relationship to the rest of the 3D world.

Use these properties to work with groups:

Property	Access	Description	Value
`name`	Get	Unique string name.	Any string.
`parent`	Get and set	This group's parent; either another object or the 3D cast member itself.	An object or cast member.
`child.count`	Get	Number of children (but not grandchildren) of a given group.	An integer.
`transform`	Get and set	Script transform object representing this group's position and orientation relative to its parent's position and orientation. `transform.position` gives the relative position. `transform.rotation` gives the relative rotation.	Set: a transform object. Get: reference to a transform object.
`userData`	Get and set	A property list containing all properties assigned to the group. Users can add, remove, get, and set properties on this list.	The default list includes the properties assigned in the 3D modeling tool. User properties are then added.
`bounding Sphere`	Get	A list containing a vector and a floating-point value. The vector represents the position of the group in world space. The floating-point value represents the radius of the bounding sphere that contains the group and its children.	A list, with the default value of `[vector (0,0,0), 0.0]`.
`world Position`	Get	Position of the group in world coordinates. A quick shortcut for `group.getWorldTransform().position`	A vector object.
`pointAt Orientation`	Get and set	A list of two orthogonal vectors, `[objectRelativeDirection, objectRelativeUp]`, that control how the groups `pointAt()` method works.	A vector list.

Downloaded from www.Manualslib.com manuals search engine

Group methods

Use these methods to work with groups:

Method	Description	Returns
`addChild(aNode, preserveWorld)`	Adds `aNode` to this group's list of children. An equivalent operation is to set `aNode.parent` to equal `this group`. The `preserveWorld` argument is optional. It can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the world transform of the child being added remains intact. If the value is `#preserveParent`, the child's existing transform is interpreted as parent-relative.	Nothing
`child[index]`	Returns the child at the specified position in the index.	Script group object
`child(name)`	Returns the child named `name`.	Script group object
`clone(name)`	Clones a group named `name`, adds it to group's parent's child list, and adds it to world All children of the group are automatically cloned. This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the group palette, has no parent, and has no children. This option lets you quickly create temporary group instances.	Script group object
`cloneDeep(name)`	Clones both the group and the parent used by the group's children. Modifications to the clones' resource don't affect the parent. This is a more memory-intensive operation than `clone(name)`.	Script group object
`addtoWorld()`	Adds the group to the currently active 3D world, setting its parent as "world." Equivalent to `group.parent=member("scene").group("world")`. All newly created groups are added to the world by default, without it being necessary to use this method.	Nothing
`remove from world`	For groups whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing
`isInWorld()`	For groups whose parent hierarchy terminates in the world, the value is `TRUE (1)`.	`TRUE (1)` or `FALSE (0)`

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns
`registerScript (eventName, handlerName, scriptInstance)`	Registers a handler named *handlerName* that is called in the *scriptInstance* when the member method *sendEvent()* is called with *eventName* as an argument. If *scriptInstance* is 0, a movie script handler is called. The user defines what *eventName* is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (xIncrement, yIncrement, zIncrement, relativeTo)`	Moves the group by *xIncrement* along the *x*-axis, *yIncrement* along the *y*-axis, and *zIncrement* along the *z*-axis. The *relativeTo* parameter is optional. It determines how arguments are interpreted. The possible values are as follows: #self: the default. Increments are applied relative to the group's local coordinate system. #parent: increments are relative to the group's parent's coordinate system. #world: increments are relative to the world coordinate system. Equivalent to #parent if parent is the world. node (group, light, camera, or group): increments are relative to the coordinate system of the argument object.	Nothing
`translate (directionVector, relativeTo)`	Moves the group `directionVector.length()` in the direction of the vector *directionVector*. The *relativeTo* argument is optional and defaults to #self.	Nothing
`translate (x,y,z, relativeTo)`	Moves the group distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *relativeTo* argument is optional and defaults to #self. This method can also be written as `translate(vector(x,y,z) relativeTo)`.	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the group by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The *relativeTo* argument is optional and defaults to #self. If included, it defines the coordinate space of the axes. This method can also be written as `rotate(vector(x,y,z) relativeTo)`.	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the group around the axis vector in the specified position the specified number of degrees. The *relativeTo* argument is optional and defaults to #self.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns
`pointAt(worldPosition, worldUp)`	Rotates the model until it points at the world-relative position worldPosition. The optional *worldUp* argument gives the general position of the model's Up axis. The exact position can't be determined using this method. Both the object-relative axes are defined by the `pointAtOrientation` property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this group into world-relative positions.	A transform object

Modifiers

Modifiers control how a model is rendered or how it animates. They are attached to a model with the `addModifier()` method. Once a modifier has been attached, its properties can be manipulated with script. The tables that follow detail the properties of the modifiers included with Director.

Level of detail (LOD) modifier properties

The level of detail (LOD) modifier provides per-model control over the number of polygons used to render a model, by allowing you to reduce the number to a lower value, based on the model's distance from the camera. This modifier is attached to all imported models.

The LOD modifier can work in one of two ways: automatically using the distance from the camera or manually. To use manual mode, disable auto mode and then set the properties yourself.

Downloaded from www.Manualslib.com manuals search engine

Use these properties to work with the level of detail modifier:

Property	Access	Description	Default
whichModel.lod.auto	Get and set	TRUE (1) means that polygons are automatically reduced based on the distance from the camera. The fewer polygons that are drawn, the faster performance will be. The lod.bias property controls how aggressively this takes place. FALSE (0) means that you can control polygon reduction on a per-model basis, provided you've attached the level of detail modifier to the model. The level of detail modifier lets you override the default settings. To release level of detail data from memory once the model has been streamed in, set the userData property sw3D to TRUE (1).	TRUE (1).
whichModel.lod.bias	Get and set	Aggressiveness with which the level of detail is reduced when in automatic mode. A value of 0.0 is most aggressive and removes all polygons. A value of 100.00 should result in no visible degradation of the geometry. A middle level can be used to remove polygons at runtime that were not removed during authoring.	A value between 0.0 and 100.0. The default is 100.0.
whichModel.lod.level	Get and set	The percentage of the model resource mesh resolution to use when the automatic mode is FALSE (0).	A value between 0.0 and 100.0. The default is 100.0.

Downloaded from www.Manualslib.com manuals search engine

Toon modifier properties

The toon modifier changes a model's rendering to imitate a cartoon drawing style. Only a few colors are used, and the model's shader, texture, and related properties are ignored.

Use these properties to work with the toon modifier:

Property	Access	Description	Default
whichModel.toon.style	Get and set	The following are the possible values: #toon: sharp transitions between available colors #gradient: smooth transitions between available colors #black and white: sharp transitions between black and white	#gradient
whichModel.toon.colorSteps	Get and set	Maximum number of colors available, rounded to the nearest power of 2, with a limit of 16	2
whichModel.toon.shadowPercentage	Get and set	The percentage of color steps to be used in shadows	50
whichModel.toon.highlightPercentage	Get and set	The percentage of color steps to be used in highlight	50
whichModel.toon.shadowStrength	Get and set	A floating-point value that determines shadow darkness	1.0
whichModel.toon.highlightStrength	Get and set	A floating-point value that determines highlight brightness	1.0
whichModel.toon.lineColor	Get and set	Color object describing line color	Black (rgb 0,0,0)
whichModel.toon.silhouettes	Get and set	TRUE (1) or FALSE (0) value indicating presence or absence of lines around silhouettes	TRUE (1)
whichModel.toon.creases	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn when mesh boundaries meet at a crease	TRUE (1)
whichModel.toon.creaseAngle	Get and set	A floating-point value controlling crease angle detection	0.01
whichModel.toon.boundary	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn at boundary of surface	TRUE (1)

Inker modifier properties

The inker modifier adds silhouette, crease, and boundary edges to an existing model. Silhouettes are edges along the border of a model. Crease edges are created when the angle between two areas of the mesh exceeds a given threshold.

Downloaded from www.Manualslib.com manuals search engine

Use these properties to work with the inker modifier:

Property	Access	Description	Default
whichModel.inker.lineColor	Get and set	Color object describing line color	Black (rgb 0,0,0)
whichModel.inker.silhouettes	Get and set	TRUE (1) or FALSE (0) value indicating presence or absence of lines around silhouettes	TRUE (1)
whichModel.inker.creases	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn when mesh boundaries meet at a crease	TRUE (1)
model.inker.creaseAngle	Get and set	A floating-point value controlling crease angle detection	0.01
whichModel.inker.boundary	Get and set	TRUE (1) or FALSE (0) value indicating whether lines are drawn at boundary of surface	TRUE (1)

Subdivision surfaces modifier properties

The subdivision surfaces (SDS) modifier causes the model to be rendered with additional geometric detail in the area of the model that the camera is currently looking at. The additional detail must be created in a third-party modeling application (usually by having the 3D application render the extra polygons during export of the model) and imported into Director along with the 3D cast member. The SDS modifier is available only for models created outside of Director. The SDS modifier should not be combined with the level of detail or toon modifier on the same model.

Use these properties to work with the SDS modifier:

Property	Access	Description	Default
whichModel.sds.enabled	Get and set	Enables/disables subdivision surfaces modifier functionality.	TRUE (1)
whichModel.sds.subdivision	Get and set	The following are the possible values: #uniform: mesh is uniformly scaled up in detail, with each face subdivided the same number of times #adaptive: additional detail is added only when there are major orientation changes and only to those areas of the mesh that are currently visible	#uniform
whichModel.sds.depth	Get and set	Maximum recursion depth, with a range of 0 to 5, to which the subdivision surfaces modifier is applied. At a value of 0, no change occurs.	1

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
whichModel.sds.tension	Get and set	Percentage of matching between modified and original surfaces.	65
whichModel.sds.error	Get and set	Percentage of error tolerance. This property applies only if sds.subdivision equals #adaptive.	0.0

Collision modifier properties

The collision modifier allows a model to be notified of and respond to collisions. You can access a model's collision modifier properties using syntax such as model.collision.*whichProperty*.

Detecting collisions and responding to collisions are separate tasks. If the enabled property is set to TRUE, and a script has been registered to be notified of collisions using the setCollisionCallback() method, that script instance receives a callback. However, the collision isn't resolved unless the resolve property is also set to TRUE.

This separation is deliberate and valuable: it can be important for a collision to be registered. In a game, for example, a projectile could strike a wall and the player's score could be increased. In that same game, however, you might not want the projectile to bounce off the wall. In that case, you'd set the enabled property to TRUE and set the resolve property to FALSE.

Collision notification can also be implemented by using registerScript() on a specific model in addition to using the setCollisionCallback() technique.

Use these properties to work with the collision modifier:

Property	Access	Description	Default
whichModel.collision.enabled	Get and set	TRUE (1) or FALSE (0) value indicating whether collisions between this model and other models will trigger a collision event.	TRUE (1)
whichModel.collision.resolve	Get and set	TRUE (1) or FALSE (0) value indicating whether collisions are automatically resolved. If the value is TRUE (1) and if the other model has the collision modifier applied and has enabled set to TRUE (1), the models will be moved back to the position of their original contact.	TRUE (1)

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
whichModel.collision.immovable	Get and set	TRUE (1) or FALSE (0) value indicating whether the model can be moved. If a model cannot be moved, the 3D Xtra can save time by not checking it for collisions with other models that also have their immovable property set to TRUE.	FALSE (0)
whichModel.collision.mode	Get and set	Indicates the geometry to be used in the collision detection algorithm. Using simpler geometry such as the bounding sphere leads to better performance. The possible values for this property are: #mesh: uses the actual mesh geometry of the model's resource. This gives one-triangle precision and is usually slower than #box or #sphere. #box: uses the bounding box of the model. This is useful for objects that can fit more tightly in a box than in a sphere, such as a wall. #sphere: is the fastest mode, because it uses the bounding sphere of the model. This is the default value for this property.	

Collision modifier methods

Use this method to work with the collision modifier:

Method	Description	Returns
whichModel.collision.setCollisionCallback (#*handlerName, scriptObjectName*)	If collision.enabled is set to TRUE (1), this method registers the script instance to receive an event when a collision occurs. If collision.enabled is set to FALSE (0), no event occurs. What happens when a collision occurs depends on the value assigned to the resolve property. You can override this value by using the collisionData.resolveA() or collisionData.resolveB() methods. The collisionData object will be the second argument passed to #*handlerName* in the specified script object *scriptObjectName*.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Collision modifier events

These events are generated when collisions occur:

Event Name	Description
`#collideAny`	The first event called when any collision occurs.
`#collideWith`	The event called whenever a collision with a specified model occurs. It is implicitly registered for when `setCollisionCallback()` is called.

Animation modifiers

Once you've created animations in your modeling software, you apply animation modifiers to models to play them back in Director.

Director supports both model keyframe and character bone animations, and modifiers are available to enable both. Keyframe animations modify a model's `transform` properties over time. Bones animations modify the model's geometry over time. Creating bones animation in a 3D modeling application can be complex, but it results in more natural-looking movements.

The two animation types can be combined. You might, for example, combine a "run in place" bones animation with a "move around the room" keyframe animation.

Bones animations use the Bones player modifier. Keyframe animations use the Keyframe player modifier. Most methods and properties are available to both players. The bones and keyframe player modifiers, like all modifiers, can only be attached to geometric nodes (models). This is only really an issue for the keyframe player because you might think to keyframe animate a camera or a light but this isn't possible.

- Motions: A 3D cast member contains a set of motions authored in your 3D modeling application. For bones animation, each motion contains a list of tracks, and each track contains the keyframes for a particular bone. A bone is a segment of the skeleton of a model. For example, track 14 of the motion named `Run` could be named `RtKneeTrack` and move a bone named `RtKnee`. These names are defined in the 3D modeling application.
- Play list: The Bones player manages a queue of motions. The first motion in the play list is the motion that is currently playing or paused. When that motion finishes playing, it's removed from the play list and the next motion begins. Motions can be added with *`bonesPlayerOrKeyframePlayer`*`.play("run")`, which adds the motion to the top of the list, or *`bonesPlayerOrKeyframePlayer`*`.queue("motion")`, which adds it to the end of the list. Using the `play` method starts the motion immediately. The motion previously at the beginning of the play list is halted unless `autoBlend` is turned on. When you use `queue`, the motion is added to the end of the play list. Motions are removed from the play list automatically when they are complete, or you can remove them explicitly using `bonesPlayer.playNext()`.
- Motion blending: If `autoblend` is `TRUE`, an ending motion blends smoothly into the next motion using the *`bonesPlayerOrKeyframePlayer`*`.blendTime` property to determine how long the blend should be. You can control this manually by setting `bonesPlayerOrKeyframePlayer.autoBlend` to `FALSE` and using *`bonesPlayerOrKeyframePlayer`*`.blendFactor` to control the blend frame by frame.
- Motion mapping: You can create new motions by combining existing motions. For example, a walking motion could be combined with a shooting motion to produce a walk-and-shoot motion. This is available only with Bones player animations.

Downloaded from www.Manualslib.com manuals search engine

You can add the Keyframe player modifier at runtime to a model created in Director, but you cannot add the Bones player modifier at runtime. The Bones player modifier is automatically attached to models with bones animation exported from a 3D modeling application. Since the required bones information can't be assigned in Director, it has to exist before the model is imported into Director.

Bones player methods

Use these methods to work with bones animations:

Method	Description	Returns
whichModel.bonesPlayer.play(*"name", looped, startTime, endTime, playRate, timeOffset*)	Plays the motion named *name* starting at the time *timeOffset*, with the currently playing motion being pushed down the play list. If *looped* is FALSE (0), the preceding motion begins again when this motion completes. *StartTime* can be an integer number of milliseconds, or it can be the symbol #synchronized. Use #synchronized to start this new motion at the same relative time offset to it's total duration as the currently playing motion is to it's total duration. The *playRate* parameter indicates how fast to play the motion. A value of 2 doubles the speed of the motion. This value is multiplied by the value of the bonesPlayer.playRate property. If blending is enabled, blending begins the instant play() is called.	Nothing
whichModel.bonesPlayer.playNext()	Ends the currently playing motion, removes it from the play list, and begins the next motion. if blending is enabled, blending begins the instant playNext() is called.	Nothing
whichModel.bonesPlayer.queue(*"name", looped, startTime, endTime, playRate, timeOffset*)	Adds the specified motion to the end of the play list. The parameters are same as those for the play() method.	Nothing
whichModel.bonesPlayer.removeLast()	Removes the most recently added motion from the play list. The motion will be removed from the play list even if it is also the currently playing motion.	Nothing
whichModel.bonesPlayer.pause()	Pauses the Bones player.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Bones player properties

Use these properties to work with bones animations:

Property	Access	Description	Default
whichModel.bonesPlayer.playing	Get	TRUE (1)= playing; FALSE (0)= paused.	TRUE (1)
whichModel.bonesPlayer.playList	Get	A linear list of property lists, where each property list yields the parameters for the currently playing and queued animations. For example, [[#name: "Walk_rt_turn", #loop: 0, #startTime: 0, #endTime: 4000, #scale: 1.0000], [#name: "Walk", #loop: 1, #startTime: 0, #endTime: -1, #scale: 1.0000]].	Empty list []
whichModel.bonesPlayer.currentTime	Get and set	Current local time of motion at the top of the play list, in milliseconds. The motion's duration property tells you how long the animation lasts.	0
whichModel.bonesPlayer.playRate	Get and set	A value indicating how quickly or slowly to play back the motion. For example, a value of 2.0 doubles the speed of the motion; a value of 0.5 halves the speed of the motion. This value is multiplied by the value of the *playRate* parameter of the play or queue method.	1.0
whichModel.bonesPlayer.playList.count	Get	Current linear list of property lists, with each property list containing the name of a motion and its playback properties.	0
whichModel.bonesPlayer.rootLock	Get and set	TRUE means the model's root bone remains at its current position. The root bone is the central bone from which all other bones branch. If this property is set to TRUE during a walking motion, the model appears to walk in place.	FALSE
whichModel.bonesPlayer.currentLoopState	Get and set	A value of TRUE means the top motion in the play list loops. A value of FALSE turns off looping for the motion at the top of the play list.	FALSE
whichModel.bonesPlayer.blendTime	Get and set	Length in milliseconds of the period when blending takes place between motions. The blendTime property is linked to motion duration. Motion blending is disabled if blendTime = 0 and autoBlend = TRUE.	500

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
whichModel. bonesPlayer. autoBlend	Get and set	If TRUE, automatic linear blending (from 0.0 to 100.0) is applied over the blend time. Otherwise, blendTime is ignored, and the amount of blending is user-determined by the blendFactor property.	TRUE (1)
whichModel. bonesPlayer. blendFactor	Get and set	The degree of blending between motions, expressed as a floating-point value between 0.0 and 100.0. A value of 0.0 uses all the previous motion. A value of 100.0 uses all of the next motion in the play list. The blend factor can be changed frame by frame to create custom blending effects.	0
whichModel. bonesPlayer. bone[*boneID*] transform	Get and set	A transform relative to the parent bone. You can get and set the entire transform, but you can't call any methods of this property.	Depends on the bone
whichModel. bonesPlayer. bone[*boneID*] worldTransform	Get and set	A transform relative to the world coordinates. You can get and set the entire transform to move a bone.	Depends on the bone
whichModel. bonesPlayer. positionReset	Get and set	TRUE (1) = object returns to starting position at end of animation. FALSE (0) = object remains at final animation position after motion completes.	TRUE (1)

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
whichModel.bonesPlayer.rotationReset	Get and set	Normally a model snaps back to its original rotation after a motion finishes playing. This property maintains any or all of the rotational changes after playing is complete. The values are as follows: #none #x #y #z #xy #xz #all	#all

whichModel.bonesPlayer.lockTranslation	Get and set	Defines the axis of translation to ignore when playing back a motion.
		The values are as follows:
		#none
		#x
		#y
		#z
		#xy
		#xz
		#all
		To keep a model locked to a ground plane with the top pointing along the *z*-axis, set lockTranslation to #z.
		lockTranslation = #all is equivalent to rootLock = TRUE (1).
			#none

Bones player events

These events are generated by bones animations:

Event name	Description
#animationStarted	This is a system-defined notification event triggered when a motion begins playing. If looping is on, this event is triggered only by the first playthrough. During a blend of two animations, this event is triggered as the blend begins.
#animationEnded	This is a system-defined notification event triggered when a motion ends. If looping is on, this event is triggered only by the first playthrough. If blending is on, this event is generated for the first animation when the blend is complete. There may be some latency because of the overhead of scheduling all other Director events.

Downloaded from www.Manualslib.com manuals search engine

Keyframe player methods

Use these methods to work with keyframe animations:

Method	Description	Returns
whichModel.keyframePlayer.play ("*name*", *looped, startTime, endTime, playRate, timeOffset*)	Plays the motion named *name* starting at the time *startTime*, with the currently playing motion being pushed down the play list. If *looped* is FALSE (0), the preceding motion begins again when this motion completes. The *startTime* parameter can be an integer number of milliseconds, or it can be the symbol #synchronized. Use #synchronized to start this new motion at the same relative time offset to its total duration as the currently playing motion is to its total duration. The *playRate* parameter indicates how fast to play the motion. A value of 2 doubles the speed of the motion. This value is multiplied by the value of the keyframePlayer.playRate property. If blending is enabled, blending begins the instant play() is called.	Nothing
whichModel.keyframePlayer.playNext()	Ends the currently playing motion, removes it from the play list, and begins the next motion. If blending is enabled, blending begins the instant playNext() is called.	Nothing
whichModel.keyframePlayer.queue("*name*", *looped, startTime, endTime, playRate, timeOffset*)	Adds the specified motion to the end of the play list. The parameters are same as those for the play() method.	Nothing
whichModel.keyframePlayer.removeLast()	Removes the most recently added motion from the play list. The motion will be removed from the play list even if it is also the currently playing motion.	Nothing
whichModel.keyframePlayer.pause()	Pauses the Keyframe player.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Keyframe player properties

Use these properties to work with keyframe animations:

Property	Access	Description	Returns
whichModel.keyframePlayer.playing	Get	TRUE (1)= playing; FALSE (0)= paused.	TRUE (1)
whichModel.keyframePlayer.playList	Get	A linear list of property lists, where each property list yields the parameters for the currently playing and queued animations. For example, [[#name: "Walk_rt_turn", #loop: 0, #startTime: 0, #endTime: 4000, #scale: 1.0000], [#name: "Walk", #loop: 1, #startTime: 0, #endTime: -1, #scale: 1.0000]].	Empty list []
whichModel.keyframePlayer.currentTime	Get and set	Current local time of motion at top of play list, in milliseconds.	0
whichModel.keyframePlayer.playRate	Get and set	A value indicating how quickly or slowly to play back the motion. For example, a value of 2.0 doubles the speed of the motion; a value of 0.5 halves the speed of the motion. This value is multiplied by the value of the *playRate* parameter of the play or queue method.	1.0
whichModel.keyframePlayer.playList.count	Get	Current number of motions in the play list.	0
whichModel.keyframePlayer.rootLock	Get and set	TRUE (1)= root translational component of the model remains at its referenced unanimated position (and therefore cannot disappear offstage).	FALSE (0)
whichModel.keyframePlayer.currentLoopState	Get and set	TRUE (1)= animation loops; FALSE (0)= animation plays through once.	FALSE (0)
whichModel.keyframePlayer.blendTime	Get and set	Length in milliseconds of the period when blending takes place between motions. The blendTime property is linked to motion duration. Motion blending is disabled if blendTime = 0 and autoBlend = TRUE.	500
whichModel.keyframePlayer.autoBlend	Get and set	If TRUE, automatic linear blending (from 0.0 to 100.0) is applied over the blend time. Otherwise, blendTime is ignored, and the amount of blending is user-determined by the blendFactor property.	TRUE (1)

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Returns
whichModel.keyframePlayer.blendFactor	Get and set	The degree of blending between motions, expressed as a floating-point value between 0.0 and 100.0. A value of 0.0 uses all the previous motion. A value of 100.0 uses all of the next motion. The blendFactor can be changed frame by frame to create custom blending effects.	0
whichModel.keyframePlayer.positionReset	Get and set	TRUE (1) = object returns to starting position at end of animation; FALSE (0) = object remains at final animation position, and begins again from there if looping is on.	TRUE (1)
whichModel.keyframePlayer.rotationReset	Get and set	Normally a model snaps back to its original rotation after a motion finishes playing. This property maintains any or all of the rotational changes after playing is complete. The values are as follows: #none #x #y #z #xy #xz #all	#all

whichModel.keyframePlayer.lockTranslation	Get and set	Defines the axis of translation to ignore when playing back a motion.
		The values are as follows:
		#none
		#x
		#y
		#z
		#xy
		#xz
		#all
		To keep a model locked to a ground plane with the top pointing along the *z*-axis, set lockTranslation to #z.
		LockTranslation = #all is equivalent to rootLock = TRUE (1).
			#none

Downloaded from www.Manualslib.com manuals search engine

Keyframe player events

These events are generated by keyframe animations:

Event name	Description
`#animation Started`	This is a system-defined notification event triggered when a motion begins playing. If looping is on, this event is triggered only by the first playthrough. During a blend of two animations, this event will be triggered as the blend begins.
`#animationEnded`	This is a system-defined notification event triggered when a motion ends. If looping is on, this event is triggered only by the first playthrough. If blending is on, this event will be generated for the first animation when the blend is complete. There may be some latency because of the overhead of scheduling all other Director events.

Mesh deform modifier properties

The mesh deform modifier lets you alter an existing model resource's geometry at runtime. You can create twist, bend, and ripple effects. Unlike other modifiers, the mesh deform modifier directly affects model resources as well as the models that use those resources. For example, if three car models share the same model resource, adding this modifier to one model and then deforming it will deform all the car models.

The mesh deform modifier is complex and is primarily useful for users with a thorough understanding of 3D geometry. However, you can take advantage of much of the modifier's potential by using only the `vertexList` property.

Use these properties to work with the mesh deform modifier:

Property	Access	Description
`whichModel``.meshDeform.mesh.count`	Get	Returns the number of meshes in a model.
`whichModel``.meshDeform.mesh[`*`index`*`].vertexList`	Get and set	Returns a list of the vertices for the specified mesh. To modify the vertices in this mesh, set this property to a list of modified vertex positions, or modify individual vertices using bracket analysis.
`whichModel``.meshDeform.mesh[`*`index`*`].normalList`	Get and set	Returns a list of the normals for the specified mesh.
`whichModel``.meshDeform.mesh[`*`index`*`].textureCoordinateList`	Get and set	Returns a list of the texture coordinates for the specified mesh.
`whichModel``.meshDeform.mesh[`*`index`*`].face.count`	Get	Returns the number of triangular faces in a given mesh.

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description
whichModel.meshDeform.mesh[*index*].face[*index*]	Get	Returns a list of three indexes into the vertex, normal, texture coordinate, and color lists. These indexes correspond to the corners of the face for the specified mesh.
whichModel.meshDeform.mesh[*index*].face[*index*].neighbor[*index*]	Get	Returns a list of lists describing the neighbors of a particular face of a mesh opposite the face corner specified by the neighbor index (1,2,3). If the list is empty, the face has no neighbors in that direction. If the list contains more than one list, the mesh is nonmanifold. This is rare. Usually the list contains four integer values. The first value is for the index into the mesh[] list, where the neighbor face lives. The second is FaceIndex, the index of the neighbor face in that mesh. The third is vertexIndex, the index within the neighbor face. The last is for Flipped, which describes whether the neighbor face is oriented in the same (0) or the opposite (1) way as the original face.
whichModel.meshDeform.face.count	Get	Returns the total number of faces in the model, which is equivalent to the sum of all the *model*.meshDeform.mesh[*index*].face.count properties in a given model.

Motions

Motions are simply animations that have been predefined in a 3D modeling application. They are included in the file that's exported from the 3D application and imported into Director.

Motions can be reused on any model in the 3D cast member, as long as the motion is appropriate to the geometry of the model. The properties and commands that follow can be used with either keyframe or bones motions.

Motion properties

Use these properties to work with motions:

Property	Access	Returns
name	Get	Name of motion.
duration	Get	Time in milliseconds motion needs to play to completion.
type	Get	The type of motion with the following values: #keyFrame: suitable for use with the Keyframe player #bones: suitable for use with the Bones player #none: no mapping has been made for this motion The default is #none.

Downloaded from www.Manualslib.com manuals search engine

Motion methods

Use this method to work with motions:

Method	Description
`map(motion, "bone name")`	Maps the given motion into the current motion beginning at the named bone. If no bone name is specified, the mapping begins at the root bone. The `map()` method will replace any motion tracks mapped previously to the specified bone and all of its children. Motion mapping has no effect on motions that are already on a play list. The `map()` method does not work with keyframe animations.

About lights and cameras

Lights illuminate the 3D world and the models in it. Without lights, the world exists, and actions can take place, but users see nothing. You can add lights to your 3D world in your 3D modeling application or with the Property inspector. For information about the Property inspector, see Chapter 14, "3D Basics," on page 303. You can also add and remove lights, change their color or position, and manipulate their parent-child relationships using methods and properties. Those methods and properties are detailed here. You can find the same lighting methods and properties, with more detailed syntax and coding examples, in the Scripting Reference topics in the Director Help Panel.

Cameras act as windows into a 3D world. Each camera that exists in a 3D cast member offers a different view into it, and each sprite that uses a 3D cast member uses one of these cameras. A camera's position can be moved with the Property inspector or the Shockwave 3D window. You can also use the Director 3D behaviors or script to manipulate camera positions. For information about the Property inspector and the Shockwave 3D window, see Chapter 14, "3D Basics," on page 303. For information about behaviors, see Chapter 15, "The 3D Cast Member, 3D Text, and 3D Behaviors," on page 315. More complex manipulations require the use of methods and properties. Accessing the properties and methods of a light or camera requires that the member be on the Stage or explicitly and completely loaded with the `preLoad()` method. When using the `preLoad()` method, you can verify that the load is complete by testing whether `member.state = 4` (loaded).

Lights and cameras have the same `transform` methods and parent-child properties as models and groups. Lights and cameras can be added, deleted, cloned, moved, and rotated in the same ways as models and groups. You can access their names, parents, children, and other properties in the same way you would with models and groups. However, there are some important differences, which arise from the specific roles that lights and cameras play in the 3D world.

Downloaded from www.Manualslib.com manuals search engine

Light properties

Use these properties to work with lights:

Property Name	Access	Description	Default
`name`	Get	Unique name of this light. If the light was exported from a 3D modeling package, the name is the name assigned there.	None
`parent`	Get and set	The model, light, camera, or group that is this light's parent. If the light has no parent, it cannot contribute light.	`Group ("World")`
`child.count`	Get	Number of immediate children (no grandchildren) that the light has.	0
`transform`	Get and set	Transform object representing light's position relative to its parent's transform. The `transform.position` gives the relative position; `transform.rotation` gives the relative rotation.	Whichever transform is required to represent the light's position and orientation in space.
`userData`	Get and set	A property list associated with this light. The list defaults to the properties assigned in the 3D modeling tool, but users can add or delete properties at any time.	Properties assigned in 3D modeling tool
`type`	Get and set	The kind of light this is. Must be one of the following: `#ambient`: applied to all sides of the model `#directional`: applied to those parts of the light facing the light's direction. Distance to the light isn't important. `#point`: Like a bare light bulb, omnidirectional and illuminating all parts of the model facing the light `#spot`: Like a spotlight, casting light on model parts that face it, with brighter illumination the closer the model is. Similar to `#directional`, except that the apparent distance from the light is taken into account.	None
`color`	Get and set	Color object defining color and intensity. Ranges from color(255,255,255), which is pure white to color(0,0,0), which is no light at all.	`color(191,191, 191)`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
spotAngle	Get and set	Angle of the light's projection cone. If type equals #spot, setting a value less than the umbra causes a "property not found" error.	90.0
attenuation	Get and set	A three-value vector controlling the constant, linear, and quadratic attenuation factors for spotlights.	vector (1.0,0.0,0.0)
specular	Get and set	TRUE (1)/FALSE (0) value that controls whether or not the light produces specular effects on surfaces. The property is ignored for ambient lights. Although TRUE (1) is the default, switching to FALSE (0) may improve performance.	TRUE (1)
spotDecay	Get and set	TRUE (1)/FALSE (0) value that controls whether or not spotlight intensity falls off with camera distance.	FALSE (0)
pointAt Orientation	Get and set	Two orthogonal vectors (objectRelativeDirection and objectRelativeUp) controlling how the light's pointAt method works.	None
boundingSphere	Get	A list containing a vector and a floating-point value, with the vector representing the position and the value the radius of a bounding sphere surrounding the light and all its children.	[vector (0,0,0), 0.0]
worldPosition	Get and set	Position of the light in world coordinates. Shortcut for the method node.getWorldTransform ().position.	Vector object

Light methods

Use these methods to work with lights:

Method	Description	Returns
addChild (*aNode, preserveWorld*)	Adds the node *aNode* to this light's list of children. An equivalent operation is to set *aNode.parent = this light*. The *preserveWorld* argument is optional. It can have two values: #preserveWorld or #preserveParent. If the value is #preserveWorld, the world transform of the child being added remains intact. If #preserveParent, the child's transform is interpreted as remaining parent-relative.	Nothing
child[*index*]	Returns the child at the specified position in the index.	Light object
child(*name*)	Returns a reference to the named child.	Light object

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns
`clone(name)`	Clones a light named `name`, adds it to light's parent's child list, and adds it to the world. All children of the light are automatically cloned.This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the light palette, has no parent, and has no children. This option lets you quickly create temporary light instances.	Light object
`cloneDeep (name)`	Clones both the light and all resources used by the light's children.	Light object
`addtoWorld()`	Adds light to currently active 3D world, setting its parent as `"world"`. All newly created lights are added to the world by default, without it being necessary to use this method.	Nothing
`removeFrom World()`	For lights whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing
`isInWorld()`	Returns a Boolean value indicating if the light is currently in the world (TRUE) or not (FALSE). This is useful for detecting if a given node's parent-heirarchy tree terminates with the world group object or not.	`TRUE (1)` or `FALSE (0)`
`registerScript (eventName, handlerName, scriptInstance)`	Registers a handler named `handlerName` that is called in the `scriptInstance` when the member method `sendEvent()` is called with `eventName` as an argument. If `scriptInstance` is 0, a movie script handler is called. The user defines what `eventName` is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (xIncrement, yIncrement, zIncrement, relativeTo)`	Moves the light forward by `xIncrement` along the *x*-axis, `yIncrement` along the *y*-axis, and `zIncrement` along the *z*-axis. The `relativeTo` parameter is optional. It determines how arguments are interpreted. The possible values are as follows: `#self`: the default. Increments are applied relative to the light's local coordinate system. `#parent`: increments are relative to the light's parent's coordinate system. `#world`: increments are relative to the world's coordinate system. Equivalent to `#parent` if the parent is the world. `node` (model, light, camera, or group): increments are relative to the argument's coordinate system.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns
`translate(x,y,z, relativeTo)`	Moves the light distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *relativeTo* argument is optional and defaults to #self. This method can also be written as translate (vector(*x*,*y*,*z*) *relativeTo*)	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the light by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The *relativeTo* argument is optional and defaults to #self. If included, it defines the coordinate space of the axes. This method can also be written as rotate (vector(*x*,*y*,*z*) *relativeTo*)	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the light around the axis vector in the specified position the specified number of degrees. The *relativeTo* argument is optional and defaults to #self.	Nothing
`pointAt(world Position, worldUp)`	Points the node's "front" at the world position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *pointAtOrientation* property. Both the object-relative axes are defined by the *pointAtOrientation* property. Default values are an object-relative forward direction of vector (0, 0, -1) and an object-relative up direction of vector (0, 1, 0).	Nothing
`getWorld Transform()`	Calculates and returns a transform that converts object-relative positions for this light into world-relative positions.	A transform object

Cameras

Cameras act as view ports into the 3D world. By default, a newly added camera's view is positioned at the world's origin, the vector (0,0,0), and looks down the negative *z*-axis. Changing a camera's `transform` property affects the camera's position and orientation. When a 3D sprite is created from a 3D cast member, the sprite uses one of the cast member's cameras. Changing the camera that the sprite is using changes what's seen in the sprite.

Cameras can also have overlays and backdrops. Overlays are 2D images drawn in front of the camera's lens. Backdrops are 2D images drawn behind the 3D scene. Backdrops provide a background image for the scene regardless of which way the camera is pointing.

Downloaded from www.Manualslib.com manuals search engine

Camera properties

Use these properties to work with cameras:

Property Name	Access	Description	Default
`name`	Get and set	Unique name of this camera. If the camera was exported from a 3D modeling program, the name is the name assigned there.	None
`parent`	Get and set	The model, light, camera, or group that is this light's parent. If the camera has no parent, it cannot contribute light.	`group ("world")`
`child.count`	Get	Number of immediate children (no grandchildren) the camera has.	`0`
`transform`	Get and set	Transform object representing camera's position relative to its parent's transform. The `transform.position` property gives the relative position; `transform.rotation` gives the relative rotation.	Identity transform
`userData`	Get and set	A property list associated with this camera. The list defaults to the properties assigned in the 3D modeling tool, but users can add or delete properties at any time.	Properties assigned in 3D modeling tool
`hither`	Get and set	A specified distance from the camera that defines the near *z*-axis clipping of the view frustum. Objects closer than hither are not drawn.	`5.0`
`yon`	Get and set	A specified distance from the camera that defines the far *z*-axis clipping of the view frustum. Objects farther than yon are not drawn.	`3.403e38`
`rect`	Get and set	The rectangle controlling the screen size and position of the camera, with the coordinates given relative to the upper left corner of the sprite.	`rect(0,0,320,200)`
`projectionAngle`	Get and set	The vertical projection angle of the view frustum.	`30.0`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
`colorBuffer.clearAtRender`	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether color buffer is or isn't cleared out after each frame. If value is set to `FALSE (2)`, the effect is similar to the trails ink effect, although it requires a semi-transparent model behind the whole scene.	`TRUE (1)`
`colorBuffer.clearValue`	Get and set	Color object defining color used to clear out buffer if `colorBuffer.clearAtRender` is `TRUE (1)`.	`color(0,0,0)`
`fog.enabled`	Get and set	`TRUE (1)` or `FALSE (0)` value indicating whether camera adds fog to the scene.	`FALSE (0)`
`fog.near`	Get and set	Distance to start of fog.	`0.0`
`fog.far`	Get and set	Distance to maximum fog intensity.	`1000.0`
`fog.color`	Get and set	Color object describing fog color.	`color(0,0,0)`
`fog.decayMode`	Get and set	How fog varies between near and far, with the following possible values: `#linear`: density is linearly interpolated between `fog.near` and `fog.far`. `#exponential`: `fog.far` is saturation point; fog.near is ignored. `#exponential2`: `fog.near` is saturation point; `fog.far` is ignored.	`#exponential`
`projection`	Get and set	Method of determining the vertical field of view, which must be of type `#perspective` or `#orthographic`.	`#perspective`
`fieldOfView`	Get and set	A floating-point value specifying the vertical projection angle in degrees.	`30.0`
`orthoheight`	Get and set	The number of perpendicular world units that fit vertically into the sprite.	`200.0`
`rootNode`	Get and set	Property controlling which objects are visible in a particular camera's view. Its default value is the world, so all cameras you create show all nodes within the world. If, however, you change `rootNode` to be a particular node within the world, a sprite of the cast member will show only the root node and its children.	`group ("world")`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
`overlay[index].loc`	Get and set	Location, in pixels, of the overlay, as measured from the upper left corner of the sprite's `rect` to the `overlay[index].source's regPoint`.	`point(0,0)`
`overlay[index].source`	Get and set	Texture object used as the source for this overlay.	None
`overlay[index].scale`	Get and set	Scale value used by a specific overlay in the camera's list of overlays.	`1.0`
`overlay[index].regPoint`	Get and set	Texture-relative rotation point, similar to a sprite's `regPoint`.	`point(0.0)`
`overlay[index].rotation`	Get and set	Rotation value used by a specific overlay in the camera's list of overlays.	`0, 0`
`overlay[index].blend`	Get and set	Blend value used by a specific overlay in the camera's list of overlays. `100` is fully opaque; `0` is fully transparent.	`100.0`
`overlay.count`	Get and set	Number of overlays in use on this sprite.	`0`
`backdrop[index].loc`	Get and set	Location, in pixels, of the backdrop, as measured from the upper left corner of the sprite's rect to the `backdrop[index].source's regpoint`.	`point(0,0)`
`backdrop[index].source`	Get and set	Texture object used as the source for this backdrop.	None
`backdrop[index].scale`	Get and set	Scale value used by a specific backdrop in the camera's list of backdrops.	`1.0`
`backdrop[index].rotation`	Get and set	Rotation value used by a specific backdrop in the camera's list of backdrops.	`0.0`
`backdrop[index].regPoint`	Get and set	Texture-relative rotation point, similar to a sprite's `regPoint`.	`point(0,0)`
`backdrop[index].blend`	Get and set	Blend value used by a specific backdrop in the camera's list of backdrops.	`100.0`
`backdrop.count`	Get	Number of backdrops in use on this sprite.	`0`

Downloaded from www.Manualslib.com manuals search engine

Property Name	Access	Description	Default
`boundingSphere`	Get	A list containing a vector and a floating-point value, with the vector representing the position and the value the radius of a bounding sphere surrounding the camera and all its children.	`[vector (0,0,0), 0.0]`
`worldPosition`	Get and set	Position of the camera in world coordinates. Shortcut for the method `node.getWorldTransform ().position`.	Vector object

Camera methods

Use these methods to work with cameras:

Method	Description	Returns
`addChild(`*`aNode`*`, `*`preserveWorld`*`)`	Adds *`aNode`* to this camera's list of children. An equivalent operation is to set `aNode.parent` to equal *`thisCamera`*. The *`preserveWorld`* argument is optional.I t can have two values: `#preserveWorld` or `#preserveParent`. If the value is `#preserveWorld`, the world transform of the child being added remains intact. If `#preserveParent`, the child's transform is interpreted as remaining parent-relative.	Nothing
`child[`*`index`*`]`	Returns the child at the specified position in the index.	Camera object
`child(`*`name`*`)`	Returns the child named *`name`*.	Camera object
`clone(`*`name`*`)`	Clones a camera named *`name`*, adds it to the cameras's parent's child list, and adds it to the world. All children of the camera are automatically cloned.This can be avoided by removing the children, performing the cloning operation, and then adding the children back. If the name is omitted or is `""`, the clone isn't added to the camera palette, has no parent, and has no children. This option lets you quickly create temporary camera instances.	Camera object
`cloneDeep(`*`name`*`)`	Clones both the camera and all resources used by the camera's children.	Camera object
`addtoWorld()`	Adds a camera to the currently active 3D world, setting its parent as `"world"` Equivalent to setting the camera's parent to the world group. All newly created cameras are added to the world by default, without it being necessary to use this method.	Nothing
`removeFrom World()`	For cameras whose parent hierarchy terminates in the world, this sets their parent to `void` and removes them from the world. Otherwise it does nothing.	Nothing

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns
`isInWorld()`	Returns a Boolean value indicating if the camera is currently in the world (TRUE) or not (FALSE). This is useful for detecting if a given node's parent-heirarchy tree terminates with the world group object or not.	`TRUE (1)` or `FALSE (0)`
`registerScript (eventName, handlerName, scriptInstance)`	Registers a handler named *handlerName* that is called in the *scriptInstance* when the member method *sendEvent()* is called with *eventName* as an argument. If *scriptInstance* is 0, a movie script handler is called. The user defines what *eventName* is. The *eventName* specified can be one of a default set of events or a user defined custom event. The default events are #collideAny, #collideWith, #animationStarted, #animationEnded, #timeMS.	Nothing
`translate (directionVector, relativeTo)`	Moves the camera *directionVector.length()* in the direction of the directionVector. The relativeTo argument is optional and defaults to `#self`.	Nothing
`translate(x,y,z, relativeTo)`	Moves the camera distance *x* along the *x*-axis, distance *y* along the *y*-axis, and distance *z* along the *z*-axis. The *relativeTo* argument is optional and defaults to `#self`. This method can also be written as `translate(vector(x,y,z) relativeTo)`.	Nothing
`rotate(x,y,z, relativeTo)`	Rotates the camera by *x* degrees around the *x*-axis, *y* degrees around the *y*-axis, and *z* degrees around the *z*-axis. The *relativeTo* argument is optional and defaults to `#self`. If included, it defines the coordinate space of the axes. This method can also be written as `rotate(vector(x,y,z) relativeTo)`	Nothing
`rotate (position, axis, angle, relativeTo)`	Rotates the camera around the axis vector in the specified position the specified number of degrees. The *relativeTo* argument is optional and defaults to `#self`.	Nothing
`pointAt(worldPosition, worldUp)`	Points the camera's "front" at the world Position and then tries to align the node's "up" with the worldUp specified, and that the node's "front" and "up" are determined by the node's *pointAtOrientation* property. Both the object-relative axes are defined by the *pointAtOrientation* property. Default values are an object-relative forward direction of vector (`0, 0, -1`) and an object-relative up direction of vector (`0, 1, 0`).	Nothing
`getWorldTransform()`	Calculates and returns a transform that converts object-relative positions for this camera into world-relative positions.	A transform object

Downloaded from www.Manualslib.com manuals search engine

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 17
Controlling the 3D World

Macromedia Director MX 2004 provides powerful methods for overall control of the three-dimensional (3D) world, including Lingo and JavaScript syntax for handling new 3D-generated events, selecting models (picking), vector math operations, and transforms. In addition, the properties and methods of the Director global renderer services object supply common rendering properties for all 3D sprites and cast members. Finally, 3D cast member and sprite properties and methods allow additional control of their content during playback.

The methods and properties that you see here in tabular form are also available with accompanying syntax, definitions, and examples in the Scripting Reference topics in the Director Help Panel.

3D Lingo or JavaScript syntax events

Event handling lets you use the `registerForEvent` method to specify a handler to run when a particular event occurs within a specific cast member. With the `registerForEvent` method, you specify the type of event that will trigger the handler, the handler name, and the script object that contains the handler. The Lingo or JavaScript syntax object can be a child script, an instance of a behavior attached to a sprite, or any other script. If the object isn't specified, the handler is called in the first movie script in which it is found.

Downloaded from www.Manualslib.com manuals search engine

Use these methods to set up event handling:

Method	Description	Returns
`registerForEvent (eventName, handlerName, scriptInstance, model)`	The event *`eventName`* is one of the following: `#collideAny`: Called when any collision occurs. `#collideWith`: Called when a collision with a specific model occurs and implicitly registered when `setCollisionCallback(...)` is called. Equivalent to calling `model.collision.setCollision Callback`. `#timeMS`: Sets up a time-based simulation callback using the format `registerForEvent (timeMS, handlerName, scriptInstance) begin, period, repetitions`. The *`begin`* and *`period`* arguments are in milliseconds. If *`repetitions`* is set to `0`, the simulation continues indefinitely. `#animationStarted`: Called whenever a keyFrame or bones motion begins. `#animationEnded`: Called whenever a keyFrame or bones motion ends. Any user-defined symbol: Registers to receive any user-defined event sent from a SendEvent call.	`TRUE (1)` if the operation succeeds. `FALSE (0)` if the operation fails.
`unregister AllEvents()_`	Unregisters all events.	Nothing. A script error is generated if the operation fails.
`sendEvent (eventName, arg1,arg2...)`	Sends an event named *`eventName`* to all scripts registered to receive it. Similar to `sendAllSprites()` except that the event is delivered only to scripts that are registered to receive it.	Nothing. A script error is generated if the operation fails.

Collisions

By attaching the collision modifier (`#collision`) to a model, you can enable that model to automatically respond to collisions with other models. By using the properties of the collision modifier, you can control the details of how the model responds to collisions. For more information about collisions, see "Modifiers" on page 366.

Collision properties

When a collision occurs, it generates a `collideAny` event or a `collideWith` event.The `collideWith` event passes an argument to the handler that is declared with the `registerForEvent`, `registerScript`, or `setCollisionCallBack` method. This argument is called a `collisionData` object and contains a property list with detailed information about the collision.

Downloaded from www.Manualslib.com manuals search engine

These properties are included in the `collisionData` object passed to the handler:

Property	Access	Description
`modelA`	Get	One model in the collision. If the script includes registration for collision with a particular model, `modelA` is that model.
`modelB`	Get	The second model in the collision.
`pointOf Contact`	Get	Vector describing world-space location of collision. Available only if the collision has been resolved. Occurs if the model's collision modifier `resolve` property is `TRUE (1)` or either the `collisionData resolveA()` or `collisionData resolveB()` method is called.
`collision Normal`	Get	Vector indicating direction of collision. Available only if the collision has been resolved. Occurs if the model's collision modifier `resolve` property is `TRUE (1)` or either the `collisionData resolveA()` or `collisionData resolveB()` method is called.

Collision methods

Collision methods let you override certain aspects of the default behavior set for models during collisions. If neither of the models involved in the collision has `resolve` set to `TRUE`, you can manually resolve the collision using `resolveA(true)` for model A or `resolveB(true)` for model B.

Method	Description	Returns
`collisionData.resolveA (`*`trueOrFalse`*`)`	Resolves collision for model A.	Nothing
`collisionData.resolveB (`*`trueOrFalse`*`)`	Resolves collision for model B. If the argument is `FALSE (0)`, the collision won't be resolved. This overrides the `collision.resolve` property, if any, set for model B.	Nothing

Selecting models

Selecting models (picking) refers to clicking on models in a 3D cast member. Because models are objects that exist within a 3D cast member and a 3D sprite, they are not normally sensitive to mouse clicks. Normally it is only the entire sprite that is sensitive to mouse clicks.

You can use scripts to determine specifically which models have been clicked when a user clicks within a 3D sprite. In practice, this allows for changing model positions to make it appear that an action such as a button being pushed or a door being opened has taken place, or to allow the user to select or drag an object. Picking can be accomplished by using cast member or camera methods.

Downloaded from www.Manualslib.com manuals search engine

Camera methods

These camera methods let you determine which models have been clicked when a user clicks the mouse within a 3D sprite. You can also translate coordinates in 3D space to coordinates in 2D sprite space and vice versa.

Method	Description	Returns
`worldSpaceToSpriteSpace(vector)`	Returns the 2D sprite-space coordinates of a point from a 3D world vector.	A point.
`spriteSpaceToWorldSpace(point)`	The opposite of the `worldSpaceToSpriteSpace(vector)`, this method returns a world-space vector on the camera's projection plane from a sprite-space point. Multiple world-space positions can map to the same sprite-space point. A round-trip `y=worldSpaceToSpriteSpace(x)` `z=worldSpaceToSpriteSpace(y)` won't necessarily result in `x=z`.	A vector.
`modelUnderLoc(point)`	Returns the first model intersected by a ray from a location *`point`* within the `rect` of the sprite using this camera. The location *`point`* is relative to the upper left corner of the sprite, in pixels. The ray is cast forward in the direction the camera is looking. This method is useful for picking in conjunction with an `onMouseDown` handler. For accuracy, be sure to subtract the upper left corner of the sprite's `loc` from the `mouseLoc`.	The first model intersected by the ray. A value of `void` means there is no model under the ray.
`modelsUnderLoc(point, optionalMaxNumberOfModels)`	Returns a list of all models intersected by a ray from a location *`point`* within the `rect` of the sprite using this camera. The location *`point`* is relative to the upper left corner of the sprite, in pixels. The ray is cast forward in the direction the camera is looking. This method is useful for picking in conjunction with an `onMouseDown` handler. For accuracy, be sure to subtract the upper left corner of the sprite's `loc` from the `mouseLoc`.	The first model intersected by the ray or a list of up to the specified maximum. If no maximum is specified, the method returns all models under the ray. `A value of void` means there is no model under the ray.
`modelsUnderRay(locationVector, directionVector, optionalMaxNumberOfModels)`	Returns a list of models under the ray starting at the vector *`locationVector`* and pointing down the vector *`directionVector`*, with both vectors specified in world-relative coordinates.	The first model intersected by the ray plus a list of up to the specified maximum number of models. If the maximum number of models isn't specified, all models the ray intersects are returned.

Downloaded from www.Manualslib.com manuals search engine

Vector math

A 3D vector describes both direction and location in 3D space. Vector objects include floating-point values for position along each of the *x*-, *y*-, and *z*-axes. Vectors can be node-relative or world-relative. If they are node-relative, their *x*, *y*, and *z* values are relative to the position of the node. If they are world-relative, their *x*, *y*, and *z* directions are relative to the world.

Vector math operations perform calculations using each of the *x*, *y*, and *z* values. These calculations are useful for performing intelligent movement and rotation of models.

Vector creation methods

Use these methods to create vectors:

Method	Description	Returns
`vector(x,y,z)`	Creates a vector from arguments representing all axes.	A vector object
`randomVector()`	Creates a vector describing a randomly chosen point on the surface of a unit sphere. Differs from `vector(random(10)/10.0, random(10)/10.0, random(10)/10.0)` because the `randomVector()` method always results in a unit vector.	A unit vector

Vector properties

Use these properties to work with vectors:

Property	Access	Description
`magnitude`	Get	The magnitude of the vector. Equivalent to the length of the vector.
`length`	Get	The length of the vector. Equivalent to the magnitude of the vector.
`[index]`	Get and set	Returns the value of a vector at a specified point in an index between 1 and 3.
`x`	Get and set	The *x* component of a vector.
`y`	Get and set	The *y* component of a vector.
`z`	Get and set	The *z* component of a vector.

Downloaded from www.Manualslib.com manuals search engine

Vector methods

Use these methods to work with vectors:

Method	Description	Returns
`normalize()`	Normalizes the vector by modifying it into a unit vector of length 1. This is done by dividing each component of the vector by the vector's original length. That original length is the square root of the sum of the squares of each component.	Nothing. Vector is modified.
`get Normalized()`	Returns a normalized version of the vector.	A new vector object.
`dot(`*`vector2`*`)`	Returns the dot (inner) product of the first vector and the second vector (*`vector2`*). If both vectors are of unit length, the result is the cosine of the angle between the two vectors.	Dot product of the two vectors.
`angleBetween (`*`vector2`*`)`	Returns the angle between *`vector`* and *`vector2`*, in degrees.	Value of the angle in degrees.
`cross(`*`vector2`*`)` or `crossProduct (`*`vector2`*`)` or `perpendicular To(`*`vector2`*`)`	Returns a vector perpendicular to the original vector and to *`vector2`*.	A new vector object.
`distanceTo (`*`vector2`*`)`	Returns the distance between *`vector`* and *`vector2`*. If these vectors represent positions in the 3D world, this is the distance between them.	Floating-point value of distance.
`duplicate()`	A copy of the vector.	A new vector object.

Vector binary operations

Use these syntaxes to perform additional vector math calculations:

Note: JavaScript does not support these operations for vector objects. In JavaScript, you must write the code to perform the vector math calculations using the vectors' *x*, *y*, and *z* coordinates.

Operator	Description	Returns
`vector1 +vector2`	Returns a new vector equaling `vector1 +vector2` for *x* equaling 1 through 3.	A new vector object
`vector1 -vector2`	Returns a new vector equaling `vector1 -vector2` for *x* equaling 1 through 3.	A new vector object
`vector1* vector2`	Returns the product of the two vectors.	A floating-point value
`vector1/vector2`	Not supported.	`0`
`vector2*scalar`	Returns a new vector equaling `vector2 * scalar` for *x* equaling 1 through 3.	A new vector object

Downloaded from www.Manualslib.com manuals search engine

Operator	Description	Returns
`vector2/scalar`	Returns a new vector equaling `vector2/scalar`.	A new vector object
`transform* vector`	Returns a new vector resulting from applying the positional and transformation changes defined by *`transform`* to *`vector`*. Note that vector*transform is an invalid operation.	A new vector object
`scalar-vector1`	Returns a new vector equaling `scalar-vector1`.	A new vector object
`vector1-scalar`	Returns a new vector equaling `vector1-scalar`.	A new vector object
`scalar + vector1`	Returns a new vector equaling `scalar + vector1`.	A new vector object
`vector1 + scalar`	Returns a new vector equaling `vector1 + scalar`.	A new vector object

Transforms

A transform is a data object describing a model's position, orientation, and scale in the 3D world. Transform methods can be used to move a given vector, light, camera, or model from its current location to a new position and/or orientation.

Transform creation method

Use the `transform()` method to create a new `transform` data object:

Method	Description	Returns
`transform()`	Creates a new transform initialized as the identity transform. The identity transform has no rotation and a vector position of `(0,0,0)`	A new transform object

Transform properties

Use these properties to work with transforms:

Property	Access	Description	Default
`position`	Get and set	Script vector object describing the position of a transform with the value `vector(`*`xOffset, yOffset, zOffset`*`)`. A model.transform position represents the model's position in relation to its parent.	`vector (0,0,0)`
`scale`	Get and set	Script vector object describing the x, y, and z scale of the transform with the vector value `vector(`*`xScale, yScale, zScale`*`)`. Scaling is always applied model-relative.	`vector (1,1,1)`

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
`rotation`	Get and set	Script vector object describing the *`xRotation`*, *`yRotation`*, and *`zRotation`* components of the transform with the value `vector(`*`xRotation`*`, `*`yRotation`*`, `*`zRotation`*`)`, with the rotation values defined in degrees. This value can vary because of the permissible types of transform operation. For example, `translate` followed by `rotate` gives a different value than `rotate` followed by `translate`, and the results can't be differentiated after the fact from the rotational information alone. The `rotate()` and `preRotate()` methods are the preferred way to modify a transform's orientation. Rotation is generally relative to the object's original orientation at the start of the movie.	`vector (0,0,0)`
`axisAngle`	Get and set	A list including a vector and a floating-point value that describes this transform's rotation as an axis/angle pair. The vector represents the direction, and the angle represents the rotation around the vector.	`[vector (1.0000, 0.0000, 0.0000) A]`

`x axis`	Get and set	A vector representing the transform's canonical x axis in transform space.
		Example:
		`transform.identity()`
		`transform.rotate(0,90,0))`
		`put transform.xaxis`
		`--vector(0,0,-1)`
		Canonical means reduced to the simplest possible mathematical expression.
			`vector (1,0,0)`

`y axis`	Get and set	A vector representing the transform's canonical y axis in transform space.
		Example:
		`transform.identity()`
		`transform.rotate(90,0,0)`
		`put transform.yaxis`
		`--vector(0,0,1)`
			`vector (0,1,0)`

`z axis`	Get and set	A vector representing the transform's canonical z axis in transform space.
		Example:
		`transform.identity()`
		`transform.rotate(0,90,0)`
		`put transform.zaxis`
		`--vector(1,0,0)`
			`vector (0,0,1)`

Downloaded from www.Manualslib.com manuals search engine

Transform methods

Use these methods to work with transforms:

Method	Description	Returns

`rotate (xAngle, yAngle, zAngle)`	Applies a rotation transformation after the current transformation:
	`model.transform.identity()`
	`model.transform.translate(100,0,0)`
	`model.transform.rotate(0,0,90)`
	After this series of transformations, performed in this order, the model's local origin will be at `(0,100,0)`, assuming the model's parent is the world.
		Nothing

`preRotate (xAngle, yAngle, zAngle)`	Applies a rotation transformation before the current transformation:
	`model.transform.identity()`
	`model.transform.translate(100,0,0)`
	`model.transform.preRotate(0,0,90)`
	After this series of transformations, performed in this order, the model's local origin will be at `(100,0,0)`, assuming the model's parent is the world.
		Nothing

`rotate (point, vector, angle)`	Similar to `transform.rotate(xAngle, yAngle, zAngle)`, except that the arguments are two vectors specifying an axis of rotation as a point and a vector, plus an angle specifying the clockwise rotation around that axis:
	`model.transform.identity()`
	`model.transform.translate(-50,0,0)`
	`model.transform.rotate(vector(100,0,0) vector(0,1,0))`
	After this series of transformations, performed in this order, the model's local origin will be at `(250,0,0)`, assuming the model's parent is the world.
		Nothing

`preRotate (point, vector, angle)`	Similar to `transform.preRotate(xAngle, yAngle, zAngle)`, except that the arguments are two vectors specifying an axis of rotation as a point and a vector, plus an angle specifying the clockwise rotation around that axis.
	`model.transform.identity()`
	`model.transform.translate(-50,0,0)`
	`model.transform.preRotate(vector(100,0,0) vector(0,1,0))`
	After this series of transformations, performed in this order, the model's local origin will be at `(150,0,0)`, assuming the model's parent is the world.
		Nothing

`translate (xIncrement,yIncrement,zIncrement)`	Translates the position of the transform relative to the transform's current orientation:
	`model.transform.identity()`
	`model.transform.rotate(0,90,0)`
	`model.transform.translate(100,0,0)`
	After this series of transformations, performed in this order, the model's local origin will be at `(100,0,0)`, assuming the model's parent is the world.
		Nothing

Downloaded from www.Manualslib.com manuals search engine

Method	Description	Returns

`preTranslate (xIncrement, yIncrement, zIncrement)`	Translates the position of the transform before the current transformation:
	`model.transform.identity()`
	`model.transform.rotate(0,90,0)`
	`model.transform.translate(100,0,0)`
	After this series of transformations, performed in this order, the model's local origin will be at `(0,0,100)`, assuming the model's parent is the world.
		Nothing

`multiply (transform2)`	Alters the original transform by applying the positional/rotational/scaling effects of *`transform2`* to the original transform. If `transform2` describes a rotation of 90° around the x axis and this transform describes a translation of 100 units in the y axis, `transform.multiply(transform2)` alters this transform so that it describes a translation followed by a rotation.	Nothing
`preMultiply (transform2)`	Alters the original transform by preapplying the positional/rotational/scaling effects of `transform2` to the original transform. If *`transform2`* describes a rotation of 90° around the x axis and this transform describes a translation of 100 units in the y axis, `transform.preMultiply(transform2)` alters this transform so that it describes a rotation followed by a translation.	Nothing
`interpolate (oTransform2, fPercentage)`	Returns a new transform by interpolating from the original transform to *`transform2`* by *`fPercentage`*. The value of `fPercentage` should be between 0 and 100.	A new transform object
`interpolateTo (oTransform2, fPercentage)`	Modifies the existing transform by *`fPercentage`*. The value of *`fPercentage`* should be between 0 and 100.	Nothing
`duplicate()`	Returns a new transform that is a copy of the original transform.	A new transform object
`identity`	Resets the transform to an identity transform: position: `–0,0,0` rotation: `0,00` scale: `1,1,1`	Nothing
`invert()`	Turns the transform into the inverse of its previous position and rotation. If you multiply a vector by a transform, the rotational and positional changes described by the transform are applied to the vector. Inverting the transform and multiplying the vector again restores the vector to its original.	Nothing
`inverse()`	Same as `invert()` except that the original transform is unaffected.	A new transform object

Downloaded from www.Manualslib.com manuals search engine

Transform operator

Use the asterisk (*) to multiply two transforms:

Operator	Description	Returns
*transform1 * transform2*	Returns a new transform that is the product of the two original transforms. Useful for combining the effects of two transforms.	A new transform object

Rendering functionality

The Director global `rendererServices` object encapsulates information about the functionality common to all 3D cast members and sprites in a movie. It provides a single place to query for the 3D mesh generators and modifiers available to all cast members.

Renderer services object properties

The global `getRendererServices()` object contains a property list with the following properties. For example, use the syntax `getRendererServices().renderer` to determine the currently active renderer.

Property	Access	Description	Default
`renderer`	Get and set	The rasterizer library all 3D sprites use to draw themselves. This property must be set before any 3D sprite comes into existence. Its default value is determined by the `preferredRenderer` property of the first cast member loaded from file. This is a run-time property that is not saved. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer	None
`renderer DeviceList`	Get	A list of available rasterizer libraries. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer	None

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default

`modifiers`	Get	A list of modifiers available for 3D cast members. Possible values are as follows:
		`#lod`
		`#toon`
		`#sds`
		`#bonesPlayer`
		`#keyframePlayer`
		`#inker`
		`#collision`
		`#meshDeform`
			None

`primitives`	Get	A list of basic 3D shapes available for all 3D cast members. Possible values are as follows:
		`#box`
		`#sphere`
		`#plane`
		`#particle`
		`#cylinder`
			None

`textureRender Format`	Get and set	A four-digit integer identifying the pixel format used for textures on the 3D hardware accelerator card. Adjust this to improved color fidelity or to fit more textures on the card. You can fit twice as many 16-bit textures as 32-bit textures in the same space. If a movie tries to use more textures than will fit on a card at a single time, Director switches to software rendering. Possible values are as follows: `#rgba8888`: one byte for red, green, blue, and alpha `#rgba8880`: same as above, without alpha opacity `#rgba5650`: 16-bit color with no alpha; 5 bits for red, 6 for green, 5 for blue `#rgba5550`: 16-bit color with no alpha; 5 bits each for red, green, and blue `#rgba5551`: 5 bits each for red, green, and blue; 1 bit for alpha `#rgba4444`: 4 bits each for red, green, blue, and alpha	`#rgba5551`
`depthBuffer Depth`	Get and set	Either 16 or 24, depending on the hardware card. Controls the precision of the hardware depth buffer.	None

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
`colorBuffer Depth`	Get	Either 16 or 32, depending on the hardware card. Controls the precision of the hardware output buffer.	None
`getHardware Info()`	Returns a property list of the specifics of the hardware card (if any) on the client's machine	A property list with the following entries: `#present`: `TRUE (1)` if the card is present; `FALSE (0)` if the card is absent `#vendor`: the vendor name as a string, with a value of Unknown if the name can't be determined `#model`: the name of the model of the hardware card, as a string `#maxTextureSize[`*`maxWidth, maxHeight`*`]`: maximum height and width of textures. Textures are reduced in size if they exceed these maximums. `#supportedTexturePixelFormat`: texture pixel formats supported by card. `#textureUnits`: number of texture units the card has `#depthBufferRange`: list of bit-depth resolutions available `#colorBufferRange`: list of bit-depth resolutions	

Movie properties

Use these properties to control which renderer the movie uses:

Property	Access	Description
`preferred3d Renderer`	Get and set	The renderer a particular movie prefers. The default value is `#auto`, which allows the movie to pick the best available renderer. This property is not the same as the `currentRenderer` property. The `preferred3dRenderer` property stipulates which renderer the movie prefers, whereas the `currentRenderer` property gives the renderer currently in use. The possible values for the `preferred3dRenderer` property are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer
`active3d Renderer`	Get	The renderer the movie is actually using. Equivalent to the `RendererServices` object `currentRenderer property`. Possible values are as follows: `#openGL`: openGL drivers for a hardware accelerator `#directX7_0`: DirectX7_0 drivers for a hardware accelerator `#directX5_2`: DirectX5_2 drivers for a hardware accelerator `#software`: built-in Director software renderer

Downloaded from www.Manualslib.com manuals search engine

Cast member properties

You can control most cast member properties using the Property inspector. For more information, see "Using the Property inspector for 3D" on page 308.

Use the following properties to work with 3D cast members in script:

Property	Access	Description	Default
`preload`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether the member is preloaded before display and playback or streamed in during playback. This property is only available for linked cast members.	None
`animation Enabled`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether animation, if any, will play.	`TRUE (1)`
`loop`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether animation loops or not.	`TRUE (1)`
`directTo Stage`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether rendering occurs directly to the Stage or to the Director offscreen buffer. If `TRUE (1)`, other sprites that intersect with this sprite may flicker. If `FALSE (0)`, rendering layers well, but speed declines.	`TRUE (1)`: rendering occurs directly to the Stage
`camera Position`	Get and set	Independent *x*,*y*,*z* translation for the default camera with values ranging from *Float_Min* to *Float_Max*.	`vector (0.0, 0.0, 250.0)`
`camera Rotation`	Get and set	Independent *x*,*y*,*z* rotation transforms for the default camera with values ranging from *Float_Min* to *Float_Max*.	`vector (0.0, 0.0, 0.0)`
`ambient Color`	Get and set	Light applied to entire scene.	`rgb(0,0,0)`
`bgColor`	Get and set	Background color in all views.	`rgb(0,0,0)`
`directional Color`	Get and set	Color of single "default" directional light.	`rgb(255, 255,255)`

`directional Preset`	Get and set	Absolute position of the single "default" directional light:
		`#None`
		`#TopLeft`
		`#TopCenter`
		`#TopRight`
		`#MiddleLeft`
		`#MiddleCenter`
		`#MiddleRight`
		`#BottomLeft`
		`#BottomCenter`
		`#BottomRight`
			#TopCenter

`specularColor`	Get and set	Specular color of first shader: the color of reflections from the shader.	`rgb(255, 255,255)`
`reflectivity`	Get and set	Reflectivity of first shader, with values from `0.0` to `100.0`.	`0.0`

Downloaded from www.Manualslib.com manuals search engine

Property	Access	Description	Default
diffuseColor	Get and set	Diffuse color of first shader: the shader's overall color.	rgb(255, 255,255)
textureType	Get and set	Default texture type for world. Values are as follows: #none: no texture #default: use original texture from Shader #member: use image from specified cast member	#default
texture Member	Get and set	Name of cast member to use as the source for the default texture when textureType is set to #member.	No texture
percent Streamed	Get	Percentage of file that has been downloaded, with values from 0 to 100.	None
bytes Streamed	Get	Integer number of bytes that have been downloaded, with values from 0 to the size of the file in bytes.	None
streamSize	Get	Total size of stream to be downloaded, with values from 0 to the size of the file in bytes.	None
state	Get	Current state of streaming. Values are as follows: 0: unloaded 1: headerLoading 2: headerLoaded 3: mediaLoading 4: =mediaLoaded -1: error After state 3 or 4 has been reached, it's safe to execute script that manipulates the 3D world. Before then, access to particular models may fail because those model definitions may not have been downloaded. Also, the loadFile() cast member method fails except at states 0 or 4.	None

Cast member methods

The following methods let you reset cast member properties to original values they had at the time the cast member was imported into Director:

Method	Description	Returns
revertToWorldDefaults()	Resets all cast member properties to the values stored in the original 3D world	Nothing
resetWorld()	Resets the 3D cast member to the state it was in when the movie first loaded	Nothing

Downloaded from www.Manualslib.com manuals search engine

Sprite properties

You can control the `directToStage` and `bgColor` properties using the Property inspector. For more information, see "Using the Property inspector for 3D" on page 308. Use the following properties to work with 3D sprites in scripts.:

Property	Access	Description	Default
`directToStage`	Get and set	`TRUE (1)` or `FALSE (0)` specification of whether rendering occurs directly to the Stage or to the Director offscreen buffer. If `TRUE (1)`, other sprites that intersect with this sprite may flicker. If `FALSE (0)`, rendering layers well but speed declines.	`TRUE (1)`: rendering occurs directly to the Stage
`bgColor`	Get and set	Background color in all views.	`rgb(0,0,0)`
`camera`	Get and set	Determines which camera this sprite is using.	None
`targetFrameRate`	Get and set	Controls the desired playback speed. If the `useTargetFramerate` property is `TRUE (1)`, then the `lod.bias` property of all model resources is dynamically altered until the target frame rate is met.	`30`
`useTargetFrameRate`	Get and set	If a target frame rate has been set and you want to use it, set this property to `TRUE (1)`.	`TRUE (1)`

Sprite methods

Use these methods to work with 3D sprites:

Method	Description	Returns
`camera(`*`index`*`)`	Accesses a specific camera in the sprite's list of views.	The camera requested
`addCamera(`*`camera, index`*`)`	Adds a camera named *`camera`* at the specified index number. If the index number is greater than the number of cameras in the sprite's camera count, or if there is no index, the camera is added to the end of the list.	An error if a camera of that name can't be found
`deleteCamera (`*`cameraOrIndex`*`)`	If *`cameraOrIndex`* is a camera, the camera by that name is deleted. If *`cameraOrIndex`* is an index number, the camera at that index number is deleted. In either case, the cameras after *`cameraOrIndex`* move forward in the list and the camera count is decreased by 1.	An error if a camera of that name or index number can't be found
`cameraCount()`	Returns the number of cameras in the sprite's `cameraList`.	An integer

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 18
Movies in a Window

Macromedia Director MX 2004 can play several movies simultaneously by creating windows in which additional movies can play. A movie in a window (MIAW) is a distinct Director movie that retains all its interactivity.

You can use a MIAW to play another movie in a separate window while the main movie plays on the Stage. In addition, movies in windows and the main movie can communicate and interact with each other. This lets you create a variety of interactive features, such as an interactive portfolio, a control panel for a second movie or digital video, or a status display window.

The following list shows the typical workflow for creating and using a movie in a window:

- Set up the movie's Display Template with default window properties.
- Create and set up the window.
- Assign a movie to the window.
- Open the window.
- Delete the window when the reason for playing the movie no longer applies.

When you create a MIAW, decide how you want it to function. For example, decide how you want to display it, how users should be able to move the window around the screen and dismiss it, and how the window should appear. You can specify the window's size and whether the window is visible, has a frame and title, or is in front of or behind other windows on the screen.

You can create and control movies in windows using behaviors from the Behavior Library or by writing your own scripts.

Macromedia Shockwave does not support MIAWs. Use MIAWs only with movies that you intend to distribute as projectors (see "About distribution formats" on page 451 and "About projectors" on page 460). However, you can use script to have Shockwave content target a URL in a browser window (see "Jumping to a URL" on page 293).

Note: You can also open MIAWs in authoring. Drop a .dir or .dcr file in the Xtras folder and the MIAW will appear in the Xtras menu.

Downloaded from www.Manualslib.com manuals search engine

About MIAWs

A MIAW is the combination of a window object and a movie object. In fact, there is no such thing as a MIAW object, but it is the name used to describe the concept of this unique combination of window and movie objects. Most of the properties described in this section are really window properties, but apply to a MIAW. MIAWs and windows are not identical, and it is important to keep that in mind.

Creating or declaring a MIAW

You must first explicitly create or declare the MIAW before you can use it or set its properties.

To explicitly declare a new MIAW in Lingo:

- Use the following script:

```
window().new ("window name")
```

To explicitly declare a new MIAW in JavaScript syntax:

- Use the following script:

```
new window("window name");
```

These statements create a window with the name and title "window name" and assign it the movie "window name" to play in that window.

Note: When you declare a MIAW, it automatically is added to the player's window list.

Assigning a MIAW a filename and title

Although the simple MIAW declaration line works well when the movie and the window title and name are all the same, you can also explicitly set the filename and title of the MIAW after creating the MIAW.

You can use a movie's filename as the argument for the `open window` method. This approach assigns that movie to a window and instructs Director to use the filename as the window title.

To assign a filename and title to a MIAW:

- Use the following script to associate the movie `"my_Movie"` with the window:

```
window("window name").filename = "my_Movie"
```

- Use the following script to assign the window's title to be `"window title"`.

```
window("window name").title = "window title"
```

The default behavior of a MIAW is to size itself using the `rect` property of the movie that is being opened in the window.

Opening and closing a MIAW

Use the methods in this section to open and close movies in windows. For more information, see the Scripting Reference topics in the Director Help Panel.

Opening a MIAW

Before you can play a movie in a separate window, you must first open the window explicitly. The window must have already been declared.

Downloaded from www.Manualslib.com manuals search engine

To open a MIAW:

- Use the following script:

```
-- Lingo
window().new("movieName")
window("movieName").open()
```

-or-

```
window().new("movieName").open()
```

```
// JavaScript syntax
new window("movieName");
window("movieName").open();
```

-or-

```
new window("movieName").open();
```

Unless you explicitly preload the movie through scripting, Director doesn't load the movie into memory until the window is opened, which can cause a noticeable pause. To load the first frame of the movie, use the `preLoadMovie` method.

You can specify other window characteristics before or after you open the window and by using the Display Template at authoring time. For more information, see "Window appearance properties" on page 413.

Closing a MIAW

You can close the window for a MIAW but leave the movie in memory, or you can close the MIAW and remove the movie from memory when it's no longer in use.

- If you leave a MIAW in memory, you get better performance if the window is reopened; however, the movie still consumes memory. You might want to use this option if you expect a MIAW to be reopened after it initially runs, or if other windows or global variables refer to the MIAW.
- If you remove a MIAW from memory, performance slows down if the window is reopened because the movie has to reload; however, it doesn't consume memory until the movie is reloaded. You might want to use this option if you don't expect a MIAW to be reopened after it initially runs or if you want to optimize memory on the computer running the MIAW.

To close a MIAW but keep it in memory:

- Use the `close()` method. After the window is closed, the window becomes invisible, but the movie continues playing. (You can also still access it in the `windowList`.) For example, `window("window name").close()`.

To close a MIAW and remove it from memory:

- Use the `forget()` method. The window is closed, and the movie is removed from memory. Use this method only if no other window or global variables still refer to the MIAW. When you remove a MIAW from memory, it is no longer available in the windowList. For example, `window("window name").forget()`. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Downloaded from www.Manualslib.com manuals search engine

Setting the window size and location for a MIAW

Setting the screen coordinates for a MIAW lets you control the size of the window and where it appears. Setting the coordinates before the movie appears controls the initial position of the window; setting them after the window appears moves the window.

About rect, drawRect, and stageRect

There are three rect values of concern: rect, drawRect and sourceRect. All three properties are available off of the stage or the window:

```
_movie.stage.rect
_movie.stage.drawRect
_movie.stage.sourceRect
window("stage").rect
window("stage").drawRect
window("stage").sourceRect
```

sourceRect: This is a property of the movie's stage, it indicates the "template" rect position at which the stage should appear; therefore, when a window is opened that is to display a particular movie, that movie's sourceRect is used to determine the window's initial position on the user's monitor.

Note: The window's sourceRect property is inherited from the movie that it contains

rect: This is a property of the window playing a particular movie and it indicates the current rect coordinates of the window on the user's monitor; this property gets its initial value (when the window is created) from the window's/movie's sourceRect property

Note: The movie's stage.rect property is inherited from the window that it is playing in.

drawRect - this is a window property that indicates the rect area within the window into which the playing movie is to be drawn; this property is initially set to rect(0,0,w,h) where w and h are the width and height of the window (by default we fill the window with the movie)

Note: The movie's stage.drawRect property is inherited.

To specify the screen coordinates for a MIAW:

Set the `rect` property to the coordinates of the location where you want the window to appear.

- Define the coordinates as a rectangle in this order: left, top, right, and bottom, as shown in the following statement:

  ```
  window("Sample").rect = rect(0, 0, 200, 300)
  ```

  For your convenience, assign the coordinates to a variable and use the variable in your script statements.

For more information, see the Scripting Reference topics in the Director Help Panel.

Downloaded from www.Manualslib.com manuals search engine

Controlling the appearance of a MIAW

You can control the appearance of a MIAW by setting different appearance properties. You can do this by using Lingo or JavaScript syntax. Some default appearance properties can also be set by using the Display Template tab found in the Property inspector. You can set the type of window you want—for example, a tool window—and how it will behave and act. For example, a document window can have a titlebar, be resizable, and have a minimize and maximize box. Because the Windows and Macintosh operating systems handle windows differently, some window properties are only available on the Windows platform. For more information about Lingo and JavaScript syntax, see the Scripting Reference topics in the Director Help Panel.

Window appearance properties

Window appearance properties let you customize how your MIAWs look and behave. Depending on what you purpose the MIAW serves, different properties can help you achieve the effects you need. You can set these properties using Lingo or JavaScript syntax. The Display Template, available from the Property Inspector, also allows you to define some default values for a movie's window appearance properties.

General

General properties let you set what type of MIAW you want, its title, whether it is resizable and more. Some properties are for Windows only.

Title lets you set the window's title. It appears in the titlebar of the window.

Type determines the type of window. It can be a tool, document, or dialog window type. If Tool is selected, the window can appear with a short titlebar and a small close box. Tool windows do not receive activate or deactivate events, because tool windows are always active. They appear on top of document windows but layer with each other. If Document is selected, the window can appear with a standard titlebar, a close box, and a minimize and maximize box. If Dialog is selected, the window has a standard titlebar and a close box. Dialog windows are always on top and are modal.

Location determines the distance of the window from the left side and top of the screen. These values specify pixels, and they apply only if the Stage is smaller than the current monitor's screen size.

Centered places the window in the center of the monitor. This option is useful if users might play a movie that was created for a 13-inch screen on a larger screen or if you are creating a movie on a large screen that will be seen on smaller screens.

Resizable determines whether the MIAW is resizable or not.

Docking determines whether a MIAW is a dockable window when opened for authoring. For more information, see "About dockable MIAWs" on page 414.

Titlebar options

Title bar properties determine what icons appear in the title bar of a window. These are accessed as part of the `titleBarOptions` list.

Maximize Box determines whether a maximize box appears in the title bar of a window.

Close Box determines whether a close box appears in the window.

Downloaded from www.Manualslib.com manuals search engine

Minimize Box determines if a minimize box appears in the title bar. For example:

window("window_name").titlebarOptions.maximizebox = true

Visible determines whether the title bar of a window is visible. When you set this property to false, the other options (Maximize Box, Close Box, and so forth) remain untouched. The next time the window's title bar is visible, all other title bar properties are maintained. In other words, it lets you override or keep changes made to these properties.

Icon lets you determine the icon that appears in the title bar of a MIAW. You must select a cast member to act as the icon.

Side Titlebar (Macintosh only property) determines whether or not the title bar appears on the top or the left side of the window.

Appearance options

Appearance properties let you determine how a window looks. For example, if you are designing a window that appears in a Macintosh application, you can set the appearance to mimic a typical Macintosh window. These properties are accessed as part of the `appearanceOptions` list.

Drop Shadow (Mac only) lets you determine if there is a shadow around the window. Most Mac windows have it turned on.

Metal (Mac only) lets you specify whether a window will have a metal or ice appearance.

Mask lets you assign a bitmap cast member as a mask for the window. For example:

window("window_name").appearanceOptions.mask = member("my mask")

Drag Mask lets you use a bitmap cast member as a mask to determine which areas of the window a user can click on to move the window. You can use this to create your own custom nonrectangular title bars or to make the entire window draggable if you wish.

Border lets you set the type of border a window should use. None indicates a borderless window, and line specifies a 1-pixel black border. However, to create a borderless window, you will also need to set both "resizable" and "titlebarOptions.visible" to false, as follows:

```
window("win").resizable = false
window("win").titlebarOptions.visible = false
window("win").appearanceOptions.border = false
```

Live Resize (Macintosh only property) determines whether or not the window contents and the size of the window will update while the user is dragging to resize the window.

About dockable MIAWs

Tool and document MIAWs can now be dockable in authoring. To make a MIAW dockable, set the `dockingEnabled` property to `TRUE`. Setting the type determines what the MIAW can dock with. For example, dockable tool MIAWs can dock to each other, to other tool windows such as the Property inspector, and can be docked to the side channels. They can also be grouped with other tool windows. Dockable document MIAWs can dock to each other and to other document windows such as the Stage, Score, and Cast windows, and all Media Editors.

- To dock a MIAW, click and drag it to a compatible window by the header bar.
- To group a MIAW, click on the Panel menu on the right side of the header bar and select a Panel group.

Downloaded from www.Manualslib.com manuals search engine

Window appearance methods

There are several new window appearance methods in Director MX 2004 that apply to MIAWs. For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

The following are three of these methods:

- `minimize()` lets you minimize your windows. You can call this function when they make custom title bars. Here is an example:

  ```
  window ("miaw").minimize()
  ```

- `maximize()` lets you maximize windows. You can call this function when you make custom title bars. Here is an example:

  ```
  window ("miaw").maximize()
  ```

- `restore()` lets you restore a window after it has been maximized. You can call this function when you set custom title bars. Here is an example:

  ```
  window ("miaw").restore()
  ```

- `notifyUser()` allows developers to notify the user that the application needs the user's attention. On Windows, the window's task bar entry will blink. On Macintosh, the projector's icon will bounce in the dock. The method arguments are flash count and rate. Setting the flash count to -1 causes the flash or bounce to continue indefinitely until the user responds. Here is an example:

  ```
  window("miaw").notifyUser(-1, 10000)
  ```

- `displaySystemTrayMessage()` (Windows only) This method allows users to pop up a message over their system tray icon. The message will appear in a balloon tooltip. This is enabled only if the Window's `systemTrayTooltip` property is set to true. The method arguments are Message Title, Message Content, and display duration. Here is an example:

  ```
  window("miaw").displaySystemTrayMessage("Message Title", "Message....",
    1000);
  ```

Setting default MIAW properties using the Display Template

A movie can define the default properties of its window by using the Display Template. When a MIAW is created, it will use its movie's Display Template properties to determine the initial state and appearance of its window. You can set the Display Template properties through the Property Inspector when authoring a movie. For more information about the different properties, see "Controlling the appearance of a MIAW" on page 413.

To set the Display Template:

1 With a movie open in the Stage, select Window > Property inspector.

2 Click the Display Template tab. A list of MIAW properties appears.

3 Set the General properties:

4 Set the Titlebar options.

5 Set the Appearance options.

Downloaded from www.Manualslib.com manuals search engine

Controlling MIAW layering

To control whether a movie appears in front of or behind other windows:

- Use the `moveToFront()` and `moveToBack()` methods. For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

This code moves a MIAW to the front:

```
window("miaw").moveToFront()
```

This code moves a MIAW to the back:

```
window("miaw").moveToBack()
```

MIAW events

Lingo or JavaScript syntax provides event handlers for typical events that can occur while a MIAW is playing, such as the movement of a window by the user. Such a handler is a good place for instructions that you want to run in response to an event that involves a window.

For example, to cause a sound to play whenever the user closes a MIAW, use the `queue()` and `play()` methods in an `on closeWindow` handler in a movie script within the movie that plays in the window. The `on closeWindow` handler will run whenever the MIAW that contains the handler closes.

Other handlers include:

- `on moveWindow` (called when a user moves the movie's window)
- `on resizeWindow` (called when a user resizes the movie's window)
- `on zoomWindow` (called when the minimize or maximize box of a movie's window is pressed)
- `on openWindow` (called when the movie's window is opened)
- `on closeWindow` (called when the movie's window is closed)
- `on activateWindow` (when a movie's window is activated)
- `on deactivateWindow` (when a movie's window is deactivated)
- `on revealWindow` (when a dockable MIAW in authoring is revealed)
- `on collapseWindow` (when a dockable MIAW in authoring is collapsed)
- `on activateApplication` (when the entire application has been activated)
- `on deactivate Application` (when the entire application has been deactivated)
- `on trayIconMouseDown` (Windows only) This event handler is triggered when a user mouses down on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true.
- `on trayIconDoubleClick` (Windows only) This event handler is triggered when a user double-clicks on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true.
- `on trayIconRightMouseDown` (Windows only) This event handler is triggered when a user right mouses down on the Window's system tray icon. This event is only meaningful if the `systemTrayIcon` property is set to true. For example:

```
on trayIconRightMouseDown
  alert ("right mouse down")
end
```


Downloaded from www.Manualslib.com manuals search engine

For more information, see the Scripting Reference topics in the Director Help Panel.

Listing the current MIAWs

The `windowList` property displays a list of all known MIAWs in the main movie. For example, the following statement displays a list of current MIAW names in the Message window:

```
put_player.windowList
```

For more information about this property, see the Scripting Reference topics in the Director Help Panel.

Controlling interaction between MIAWs

MIAWs can interact with other MIAWs by accessing a window's movie property. With a window's movie, a user can access the movie's handlers, variables, members, and so on.

Here are a couple of examples:

```
window("other MIAW").movie.member(1).name = "changed name"
```

or

```
window("other MIAW").movie.someHandler()
```

Global variables can be declared in the main movie (the Stage) or in a MIAW. No matter where they are declared, they are available to the main movie and to all MIAWs. For more information about global variables, the Scripting Reference topics in the Director Help Panel.

A MIAW can also interact with the main movie via `window("stage")`.

For example:

```
window("stage").movie.member(1).name = "changed name"
```

or

```
window("stage").movie.someHandler( )
```


Downloaded from www.Manualslib.com manuals search engine

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 19
Using the XML Parser Xtra

The XML Parser Xtra lets Macromedia Director MX 2004 movies read, parse, and use the contents of Extensible Markup Language (XML) documents. Using the XML Parser Xtra requires that you understand the structure and content of the documents you are parsing. You can then access the XML document's contents through Lingo or JavaScript syntax or convert the contents to a script list that is meaningful to you and your movie. After your movie has read an XML document, it can perform actions that you define based on the contents of the document.

For example, an XML document might describe the structure of a molecule for an educational chemistry application. After the movie has parsed the XML that describes the molecule, it can use that information to draw an accurate visual representation of the molecule on the screen. In this case, the movie must be programmed to know ahead of time that the XML document describes a molecule and not a grocery list, in order to make logical use of it.

About XML

XML is similar to HTML in that it uses markup tags to define content. However, HTML has predefined tags that you can use to format any data. Any application that reads HTML must understand the meaning of tags such as `TITLE`, `P`, and `BODY`. HTML tags also describe how information appears on the screen. XML, on the other hand, consists of a set of rules that let you define custom tags and the type of data they can contain, and it has no visual component. With XML, there is no predefined way to display any given type of data such as molecular structures or grocery lists. An XML document is merely a container for the data.The Director developer, by knowing what kind of data the XML document contains, can make intelligent decisions about what the Director movie should do with the information.

One key advantage of XML over a regular text document is that XML is not order-dependent. If an application refers to the third item in a line of data, inserting new data or making subsequent changes to the way the data is produced could cause the application to fail. With XML, you can refer to the individual data components by name. If you insert a new chunk of data before the one in use, the name is still valid. Existing code continues to work, and the newly inserted data is ignored.

Downloaded from www.Manualslib.com manuals search engine

There are many sources of information for understanding, creating, and editing XML on the Internet. The following websites offer useful information about XML:

- www.xml.com
- www.ucc.ie/xml/
- www.w3.org/TR/REC-xml
- www.w3.org/DOM/

Using XML parser objects

The XML Parser Xtra lets Director developers access the nodes of an XML document. A node can be a tag (similar to an HTML tag, also called an element), character data (text that does not appear inside the angle brackets of a tag), or a processing instruction (a special type of tag that passes data to the parsing application for special processing). You can extract information from the XML document by looking at its nodes with Lingo or JavaScript syntax. This access to XML data lets users incorporate XML documents into their movies and selectively extract data from the documents.

An XML document is well formed if it has a coherent nesting structure and no misplaced angle brackets. Some XML documents can be associated with a document type declaration (DTD) file that describes specific rules for the XML tags the document uses. The XML parser of the Xtra checks that the XML follows the general rules of creating and using XML tags to ensure that the document is well formed. However, the Xtra does not check the DTD file to ensure that the tags follow the specific rules for XML document tags, which is described by the DTD. For this reason, the Xtra is called *nonvalidating*. The creator of the original XML document must follow the rules described in the DTD. Your Director movie should also include script that checks for errors in the use of the XML document's tags.

To use the XML Parser Xtra, create a parser object by using Lingo or JavaScript syntax to assign a new instance of the Xtra to a variable. This variable now contains the parser object. Use a global variable if you need to access the XML data from anywhere in the Director movie.

```
global gParserObject
gParserObject = new(xtra "xmlparser")
```

The next step is to parse the XML data using the `parseString()` method. The data can come from the text of a cast member or a string variable. To parse XML from a URL, use `parseURL()` instead. `ParseString()` and `parseURL()` return either `VOID`, which indicates that the method is successful, or an error code that indicates a problem with the XML data.

The following script statement sets the variable `errCode` to the return value of the `parseString()` method:

```
errCode = gParserObject.parseString(member("XMLtext").text)
```

After the XML Parser Xtra parses the data, the parser object contains all the data from the XML document.

The XML data can be considered a tree structure because most documents have tags nested within other tags, with each tag being like a branch of the tree.

Downloaded from www.Manualslib.com manuals search engine

The following example shows a short XML document:

```
<?xml version="1.0"?>
<e1><tagName attr1="val1" attr2="val2"/><e2>element 2</e2><e3>element 3</e3>
  </e1>
```

The following example is the same XML with its tree structure shown more clearly:

```
<?xml version="1.0"?>
  <e1>
    <tagName attr1="val1" attr2="val2"/>
    <e2>element 2</e2>
    <e3>element 3</e3>
  </e1>
```

There are two ways to access a parsed XML document. You can use the `makeList()` method to convert the document into a nested property list and use the list-access methods of Lingo or JavaScript syntax, or you can use the special script methods of XML Parser Xtra to access the parsed data directly.

The following script statement uses the `makeList()` method to convert the parsed data to a property list:

```
theList = gParserObject.makeList()
```

If you choose to make a property list with the `makeList()` method, the result is a nested property list, reflecting the tree structure of the XML.

Each element in the document is represented by its own property list, with another property list for each child element that it contains. The name of the element is the property name, and the content of the element is the property value. Attributes of an element are stored in a child list with the name !ATTRIBUTES. The property list of attributes contains the name of each attribute and its value. Character data has the property name !CHARDATA, and the value is the string representation of the character data. A processing instruction is a property with the name !PROCINST; its value is another two-element property list. The first property of this sublist is NAME, and the value is the string that represents the name of the processing instruction. The second property of the sublist has the name TEXT and contains the rest of the text in the processing instruction.

The property list resulting from the previous XML example would look like the following code:

```
["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "e1": ["!ATTRIBUTES": [:],
  "tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]], "e2":
  ["!ATTRIBUTES": [:], "!CHARDATA": "element 2"], "e3": ["!ATTRIBUTES": [:],
  "!CHARDATA": "element 3"]]]]
```

The following example is the same property list with its nested structure shown more clearly:

```
["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:],
  "e1": ["!ATTRIBUTES": [:],
    "tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]],
    "e2": ["!ATTRIBUTES": [:], "!CHARDATA": "element 2"],
    "e3": ["!ATTRIBUTES": [:], "!CHARDATA": "element 3"]
  ]
]]
```

Together, the Lingo or JavaScript syntax statements that create a property list from a string of XML data would look like the following examples:

```
global gParserObject
gParserObject = new(xtra "xmlparser")
```


Downloaded from www.Manualslib.com manuals search engine

```
errCode = gParserObject.parseString(member("XMLtext").text)
theList = gParserObject.makeList()
```

After this code has been executed, the variable `gParserObject` contains the parsed node structure of the XML document, and the variable `theList` is a property list that contains all the information in the document broken into property name and value pairs. All the regular script methods for sorting and accessing lists can work normally with `theList`.

Using XML document nodes

The XML document can contain different types of nodes. Each node can contain different kinds of data, depending on the node type. You should check the node type before accessing its data so you know what type of data to expect. Nodes are read-only, so you can retrieve the type, but you cannot set it.

You use Lingo or JavaScript syntax to access the nodes of an XML document. The following table shows the script terms that refer to nodes and their properties:

Node Script	Return value if an element	Return value if text	Return value if Processing Instruction
type	`#element`	`#text`	`#procInst`
`name`	String representing the name of the element	VOID	String representing the name of the processing instruction
`child[N]` (N is an integer)	The Nth child node of the node; VOID is returned if no Nth child exists or there is a script error	VOID	VOID
`attributeName[N]` (N is an integer)	String representing the name of the Nth attribute; VOID is returned if no Nth attribute exists or there is a script error	VOID	VOID
`attributeValue[N]` (N is an integer)	String representing the value of the Nth attribute; VOID is returned if no Nth attribute exists or there is a script error	VOID	VOID
`attributeValue[N]` (N is a string)	String representing the value of the attribute with the name N; VOID is returned if the node does not have an attribute named N or there is a script error	VOID	VOID
`text`	VOID	String representing the character data contained in this node	String representing the data section of the processing instruction

Note: The subfield `count` exists for any field that is accessible with bracket access. You can specify `whichNode.child.count` to find how many children are in the specified node.

Downloaded from www.Manualslib.com manuals search engine

Using this XML document as a starting point, the following examples demonstrate how to use these script terms to access the data within various node levels of the XML structure.

The XML looks like the following example:

```
<?xml version="1.0"?>
  <e1>
    <tagName attr1="val1" attr2="val2"/>
    <e2>element 2</e2>
    <e3>element 3</e3>
    Here is some text
  </e1>
```

The following script returns the name of the first XML tag:

```
put gParserObject.child[1].name
-- "e1"
```

The `gParserObject` variable contains the parsed XML. When used in the preceding script, it refers to the root node of the XML document. The script term `child[1]` refers to the first level of nested tag, which is the `e1` tag.

To find out what kind of node the first tag is, use the `type` method, as shown in the following example:

```
put gParserObject.child[1].type
-- #element
```

To refer to nodes that are nested more than one level deep, use more than one level of child reference. The following script returns the name of the first tag that is nested within the `e1` tag:

```
put gParserObject.child[1].child[1].name
-- "tagName"
```

The following script returns the name of the second tag that is nested within the `e1` tag:

```
put gParserObject.child[1].child[2].name
-- "e2"
```

To refer to text data that occurs within a particular tag, use the `text` property. The text is a child node of the tag that contains it, so you need an additional level of child reference. This script returns the following string, which appears inside the `e2` tag from the previous XML example:

```
put gParserObject.child[1].child[2].child[1].text
-- "element 2"
```

In this example, the `gParserObject` variable refers to the root node of the XML. The `child[1]` refers to the `e1` tag, which occupies the first level down in the XML's nested structure. The `child[2]` refers to the second tag within the `e1` tag, which is the `e2` tag. The last `child[1]` refers to the text within the `e2` tag, which is `element 2`. Finally, the `text` property is specified, so script returns the text of the node rather than any other property of the node.

The fourth child of the `e1` tag is a line of text that reads `here is some text`. This text is a child the same as the XML tags that precede it. You can get the type of this child the same way you get other children.

The following script returns the type of the fourth child of the `e1` tag:

```
put gParserObject.child[1].child[4].type
-- #text
```


Downloaded from www.Manualslib.com manuals search engine

This script returns the text of the fourth child of the `e1` tag, as shown in the following example:

```
put gParserObject.child[1].child[4].text
-- "
     here is some text
   "
```

The text element includes the white space for Return, Space, and Tab characters as well as the string `"here is some text"`.

You can use the script `count` method to determine the number of children that exist at a particular level of the XML structure. The following script returns the number of children at the 2nd level in the previous XML example:

```
put gparser.child[1].child.count
-- 4
```

Accessing attributes

Some XML tags contain attributes with values. Use the `attributeName` and `attributeValue` properties to access the attributes of tags that have values. In the previous XML example, the first tag nested inside the `e1` tag is called `tagName` and has two attributes called `attr1` and `attr2`.

The following script uses the `attributeName` property to return the name of the first attribute of the tag called `tagName`, which is the first child of the `e1` tag:

```
put gParserObject.child[1].child[1].attributeName[1]
-- "attr1"
```

The following script uses the `attributeValue` property with an integer to return the value of the first attribute of the `tagName` tag:

```
put gParserObject.child[1].child[1].attributeValue[1]
-- "val1"
```

The following script uses the `attributeValue` property with a string to return the value of the `attr1` attribute:

```
put gParserObject.child[1].child[1].attributeValue["attr1"]
-- "val1"
```

The following script uses the `count` method with the `attributeName` property to return the number of attributes in the first child of the `e1` tag:

```
put gParserObject.child[1].child[1].attributeName.count
-- 2
```

Parser objects and XML nodes

As described in earlier sections, the parser object in the `gParserObject` variable stores the root of the parsed tree of the XML document. An XML node is a node within the tree. The root node is like an XML node because almost all the operations on XML nodes can be applied to the root node.

In the previous XML example, the root of the tree is an XML element node named "ROOT OF XML DOCUMENT" that has no attributes and one child (the `e1` tag). You can get the same information about the root node as for any of the child nodes.

Downloaded from www.Manualslib.com manuals search engine

The following script returns the root node's type, name, and number of children:

```
put gParser.type, gParser.name, gParser.count(#child)
-- #element "ROOT OF XML DOCUMENT" 1
```

The main difference between the root node and its child nodes is that there are several script methods that apply to the entire XML document and operate on the root node only. These methods include `doneParsing()`, `getError()`, `ignoreWhiteSpace()`, `makeList()`, `parseString()`, and `parseURL()`.

Treating white space

The default behavior of the XML Parser Xtra is to ignore character data between XML tags when all the characters are white space. This type of white space is usually due to Return characters and superfluous space characters, but sometimes it can have meaning to the XML document.

You can use the `ignoreWhiteSpace()` method to change the way the Xtra treats white space. By setting the `ignoreWhiteSpace()` to `FALSE` instead of its default value of `TRUE`, you can tell the Xtra to treat instances of white space as literal data nodes. This way, white space between elements is treated as actual data.

The following script statements leave `ignoreWhiteSpace()` set to the default `TRUE` value, and parse the given XML into a list. The `sample` element has no children in the list.

```
XMLtext = "<sample> </sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES":
  [:]]]]
```

The following script statements set `ignoreWhiteSpace()` to `FALSE`, and parse the given XML into a list. The `sample` element now has a child that contains one space character.

```
XMLtext = "<sample> </sample>"
parserObj.ignoreWhiteSpace(FALSE)
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES":
  [:], "!CHARDATA": " "]]]
```

If there are non-white space characters in a !CHARDATA node, all the characters of the node, including leading and trailing white space characters, are retained.

XML and character sets

When you use XML, remember that different computer systems use different binary encoding to represent text characters.

The XML Parser Xtra adheres strictly to the XML specification, which states that XML documents are, by default, encoded using the UTF-8 character set. If the document is not encoded in UTF-8, it must include a declaration of its character set in the first line of the document.

The following XML declares the IOS-8859-1 character set, also known as Latin1:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```


Downloaded from www.Manualslib.com manuals search engine

This requirement applies to documents parsed by `parseString()` as well as files that are parsed by `parseURL()`.

The XML Parser Xtra supports the following character sets:

- **ISO-8859-1** Also called Latin 1. This is the most common Western encoding used on the web. It matches the default character set used on Windows in most Western countries. It does not match the character set used in most Western versions of the Mac OS (MacRoman) and does not match character sets commonly used in most non-Western countries. The first 127 characters (binary codes 1-127) are the same in most countries.
- **UTF-8** An 8-bit encoding of the Unicode character set. This is the XML default character set.
- **US-ASCII** Supports only 7-bit characters.
- **EUC-JP** The EUC character set, used widely in Japan.
- **SHIFT_JIS** Also used widely in Japan. Shift-JIS is the character set used by default in Japanese versions of Windows and the Mac OS.
- **UTF-16** A 16-bit encoding of the Unicode character set.

For many developers, Latin 1 is the most convenient encoding to use.

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 20
Making Director Movies Accessible

As a Macromedia Director MX 2004 author, you might want to provide ways for computer users with disabilities to experience the movies that you create. Director includes accessibility features that let you do this. By using these features, you can make existing and new movies accessible to users who have hearing, visual, or mobility impairment.

With scripts and behaviors, you can add text-to-speech capability to your movies. This lets visually impaired users hear the text in your movie read aloud by the computer. You can provide captioning to help users with hearing impairment experience the audio portions of your movies. Finally, you can enable your movies to be navigated with the keyboard instead of a mouse, which benefits users with certain kinds of mobility impairment.

If you work on projects sponsored by the U.S. government, you might be required to follow government guidelines for providing accessibility to disabled users.

You add accessibility to your Director movies by using the included accessibility behaviors or special script methods. The text-to-speech behaviors and scripting require the Speech Xtra. If you use text-to-speech in your movie, you must add the Speech Xtra to your movie's Xtra extensions list. This is discussed in detail in this chapter.

About government requirements

The U.S. government has stated that multimedia created for the purpose of fulfilling a government contract must be made accessible to computer users with disabilities. The government requirements include text-to-speech for the visually impaired, captioning for the hearing impaired, and keyboard navigation for the mobility impaired.

The full text of the government requirements is available at www.section508.gov/.

Making Director movies accessible

You can make your Director movies accessible by adding behaviors or writing custom scripts. Director includes several behaviors that let you easily add text-to-speech, captioning, and keyboard navigation to your movies with simple drag-and-drop procedures. If you want to have more control over how you implement accessibility in your movies, you can write custom scripts that use the text-to-speech methods. For more information about text-to-speech script, see "Accessibility scripting with Lingo or JavaScript syntax" on page 434.

Downloaded from www.Manualslib.com manuals search engine

Using the Speech Xtra

The Speech Xtra adds special methods to scripting that enable the Director text-to-speech capability. The text-to-speech behaviors require this Xtra extension because they use these script methods. If you write custom text-to-speech scripts, you must include the Speech Xtra in your movie's Xtra extensions list. Keyboard navigation does not require the Speech Xtra.

The Speech Xtra supports Microsoft SAPI 4 and 5 on Windows. The Speech Xtra supports all versions of text-to-speech on the Macintosh.

Because the Speech Xtra has no specific cast member type associated with it, it is not added automatically to a movie's Xtra extensions list when a certain type of cast member is added to the movie. Therefore, when you use text-to-speech script or behaviors in a projector, you must remember to add the Speech Xtra to the Xtra extensions list manually. The Speech Xtra is available automatically in the Director application and in Macromedia Shockwave.

To add the Speech Xtra to a movie's Xtra extensions list:

1. Select Modify > Movie > Xtras.
2. Click Add.
3. In the alphabetical list of Xtra extensions, scroll to Speech, and select the Speech Xtra. (In Windows, the file name is Speech.x32.)
4. Click OK to close the Add Xtras dialog box.
5. Click OK to close the Movie Xtras dialog box.
6. Save your movie.

Testing the Speech Xtra

To verify that your computer is configured correctly to let the Speech Xtra work, you can perform a simple test.

- To test the Speech Xtra, type the following script into the Message window:

  ```
  put voiceInitialize()
  ```

  If the result is 1, then the Speech Xtra is working. If the result is 0, then you might not have text-to-speech software installed on your computer. For more information about text-to-speech software, see "System requirements" on page 436.

Using the Accessibility behavior library

You can make movies accessible in three ways. You can add keyboard navigation, text-to-speech, and captioning. The Director Library palette includes an Accessibility section that contains behaviors for enabling each capability.

For keyboard navigation, you use the Accessibility Target, Accessibility Item or Accessibility Text Edit Item, Accessibility Keyboard Controller, and Accessibility Group Order behaviors.

For text-to-speech, you use the keyboard navigation behaviors and then add either the Accessibility Speak or Accessibility Speak Member Text behavior. You can also use the Accessibility Speak Enable/Disable behavior to let users turn the text-to-speech feature on and off.

For captioning, you use the Keyboard navigation behaviors and Accessibility Speak behaviors and then add the Accessibility Captioning and Accessibility Synch Caption behaviors.

Downloaded from www.Manualslib.com manuals search engine

Enabling keyboard navigation

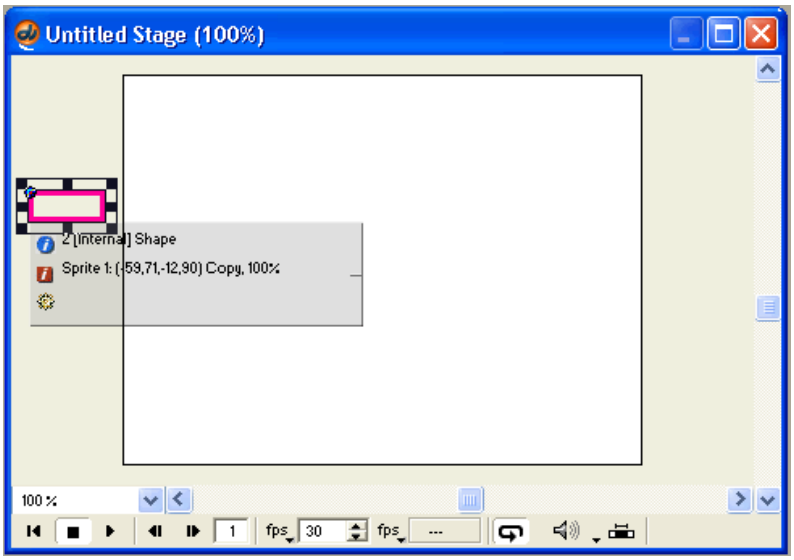
With the Accessibility Behavior library, you can easily make sprites on the Stage navigable with the keyboard. This lets mobility-impaired users select sprites and simulate mouse clicks without using a mouse. For example, if you have four button sprites on the Stage, you can apply accessibility behaviors that let them be selected by using the Tab key. As each sprite is selected, it is highlighted with a colored rectangle, called a focus ring, around its bounding rectangle. After a sprite is selected, the user can press Enter (Windows) or Return (Macintosh) to initiate the same action that a mouse click would initiate on the sprite.

The accessibility behaviors work with each other. Most of them cannot be used alone and must be used with other accessibility behaviors. To enable keyboard navigation, you must use the Accessibility Target, Accessibility Item, Accessibility Text Edit Item, Accessibility Group Order, and Accessibility Keyboard Controller behaviors together.

To apply the Accessibility Target behavior:

1 Create a shape sprite on the Stage using the Rectangle tool in the Tools panel. The accessibility behaviors use this sprite to create a focus ring around other sprites you define as navigable with the keyboard. The focus ring shows which sprite is the current selection during keyboard navigation.

2 Move the shape sprite off the Stage into the area immediately next to the Stage. You can expand the Stage window to display more of this area if necessary. The other accessibility behaviors move the sprite onto the Stage when keyboard navigation is used.



A shape sprite off the visible part of the Stage.

3 Open the Library palette by selecting Window > Library palette.

4 Select Accessibility from the Library List menu in the upper-left corner of the Library palette.

5 Drag the Accessibility Target behavior on to the shape sprite that is located next to the visible part of the Stage.

Downloaded from www.Manualslib.com manuals search engine

6 In the dialog box that appears, enter a name for the behavior group.

Because the accessibility behaviors work together and are dependent on one another, a group name associates all the accessibility behaviors used in a given scene of your movie with one another. When you apply behaviors to a different scene of your movie, you might want to use a different group name. For example, you might use Accessibility_Scene_1. In addition, there should only be one instance of the Accessibility Target behavior in any given scene of your movie.

7 Select an initial state for text-to-speech, either enabled or disabled. If you select disabled, you must use the Accessibility Speak Enable/Disable behavior to let users turn on text-to-speech later. For more information, see "Enabling text-to-speech" on page 431.

8 Click OK.

The next step in the process is to attach the Accessibility Keyboard Controller behavior to a sprite. This behavior intercepts keystrokes from the keyboard and enables those keyboard events to be used for navigation of the sprites on the Stage. You attach the Accessibility Keyboard Controller behavior to an editable text sprite that you place outside the visible part of the Stage.

To apply the Accessibility Keyboard Controller behavior to an off-Stage editable text sprite:

1 Place an editable text sprite on the Stage. Do not use a field sprite.

2 Move the text sprite off the Stage into the area immediately below the Stage. You can expand the Stage window to display more of this area if necessary.

3 Drag the Accessibility Keyboard Controller behavior from the Library palette to the sprite.

4 In the dialog box that appears, select the accessibility group name for the scene, such as Accessibility_Scene_1.

5 Click OK.

6 Select the editable text sprite if it is not still selected.

7 Using the arrow keys, move the sprite off the edge of the Stage until it is no longer on the visible part of the Stage.

Note: If the editable text sprite receives a Return character, its bounding rectangle can expand vertically. To prevent this from happening and unexpectedly causing the sprite to become visible on the Stage, set the sprite's Framing property to Fixed in the Property inspector's Sprite tab.

Now that the Accessibility Target behavior is attached to the shape sprite and the Accessibility Keyboard Controller behavior is attached to a text sprite, you are ready to apply the Accessibility Item or Accessibility Text Edit Item behaviors. To enable navigation to a sprite with the Tab key, attach the Accessibility Item behavior to the sprite. To enable navigation to an editable text sprite with the Tab key, attach the Accessibility Text Edit Item behavior to the sprite.

To apply the Accessibility Item or Accessibility Text Edit Item behaviors:

1 Drag the Accessibility Item or Accessibility Text Edit Item behavior from the Library palette to the designated sprite.

2 In the dialog box that appears, select the same group name that you used when applying the Accessibility Target behavior—for example, Accessibility_Scene_1.

3 You can specify a script string to be executed when the user presses Enter (Windows) or Return (Macintosh) while the sprite is selected. You can use any valid script string, such as a `go to frame 20` method or a call to a separate handler that you have written, such as `startAnimation`.

4 Click OK.

Downloaded from www.Manualslib.com manuals search engine

5 Repeat this process for each sprite in the scene that you want to be navigable.

Note: The Accessibility Item and Accessibility Text Edit Item behaviors cannot be applied to sprites that have the Accessibility Target or Accessibility Keyboard Controller behaviors attached. If you are testing your movie while applying Accessibility Item or Accessibility Text Edit Item behaviors, you might need to rewind your movie to restore the shape sprite with the Accessibility Target behavior to its original location.

After all the sprites have had an Accessibility Item or Accessibility Text Edit Item behavior attached to them, you can add the Accessibility Group Order behavior to them. This behavior lets you specify the order in which each sprite is selected on the Stage when the user presses the Tab key. Apply this behavior to each sprite you want to be navigable with the Tab key.

To apply the Accessibility Group Order behavior:

1 Drag the Accessibility Group Order behavior from the Library palette to one of the sprites that already has an Accessibility Item or Accessibility Text Edit Item behavior attached.

2 In the dialog box that appears, select the group name that is used by the other accessibility behaviors in the scene, such as Accessibility_Scene_1.

3 Enter the tab order for the sprite. This is the order in which the sprites are selected when the user presses the Tab key. Be sure to number each sprite consecutively and to use each number only once. When the movie begins playing, the focus ring automatically goes to the sprite whose group order is 1.

4 Click OK.

5 Repeat this process for each sprite in the scene that is navigable with the Tab key.

All the sprites can now be navigated to and activated with the keyboard. You can repeat this process for each scene in your movie.

Enabling text-to-speech

To help visually impaired users experience your Director movies, you can add the ability for your text cast members to be spoken aloud by the computer. You can also specify text to be spoken when sprites are selected. Because the text-to-speech feature in Director assumes that users are visually impaired, the feature is designed to be used with the keyboard navigation feature. Each text-to-speech behavior requires that the Accessibility Target, Accessibility Item or Accessibility Text Edit Item, Accessibility Group Order, and Accessibility Keyboard Controller also be applied to the appropriate sprites.

The following list describes the three text-to-speech behaviors:

- The Accessibility Speak behavior lets you specify a text string to be spoken when the user navigates to a sprite with the Tab key.
- The Accessibility Speak Member Text behavior lets you specify a text cast member to be spoken when a user navigates to a sprite with the Tab key.
- The Accessibility Speak Enable/Disable behavior lets you specify a user event that can turn the text-to-speech feature on or off.

Downloaded from www.Manualslib.com manuals search engine

To apply the Accessibility Speak behavior:

1 Begin by applying the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Drag the Accessibility Speak behavior from the Library palette to the sprite you want to trigger the spoken text when the user navigates to it with the Tab key. This sprite must already be attached to an Accessibility Item or Accessibility Text Edit Item and an Accessibility Group Order behavior.

3 In the dialog box that appears, select the behavior group name for the scene, such as Accessibility_Scene_1. This group name enables the behaviors in a scene of your movie to communicate with each other so they can operate properly.

4 Enter the text to be spoken when the sprite is selected with the Tab key.

5 Click OK.

If you wish to have a large amount of text spoken, you can specify an entire text cast member to be spoken by using the Accessibility Speak Member Text behavior.

To apply the Accessibility Speak Member Text behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

Drag the Accessibility Speak Member Text behavior from the Library palette to the sprite that you designate to trigger the spoken text when the user navigates to it with the Tab key. This sprite must already be attached to an Accessibility Item or Accessibility Text Edit Item and an Accessibility Group Order behavior.

2 In the dialog box that appears, select the behavior group name for the scene, such as Accessibility_Scene_1. This group name enables the behaviors in a scene of your movie to communicate with each other so they can operate properly.

3 Enter the name of the text cast member to be spoken when the sprite is selected with the Tab key.

4 Click OK.

You can let users toggle the text-to-speech feature on and off by using the Accessibility Speak Enable/Disable behavior. You add this behavior to a sprite that already has the keyboard navigation behaviors attached to it.

To apply the Accessibility Speak Enable/Disable behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Drag the Accessibility Speak Enable/Disable behavior from Library palette to the sprite you want to use to toggle the text-to-speech behaviors.

3 In the dialog box that appears, enter the behavior group name for the scene so that this behavior is associated with the other behaviors in the scene.

4 Select an event to toggle the on/off state of the text-to-speech behaviors. This can be a mouse click or the beginning or ending of the sprite in the Score.

5 Select whether to turn speech on or off when the event you selected in the previous step occurs.

6 Enter the words to be spoken when text-to-speech is turned on.

7 Enter the words to be spoken when text-to-speech is turned off.

8 Click OK.

Downloaded from www.Manualslib.com manuals search engine

Using captioning

For users who are hearing impaired, you can add text captioning to your movies. Captioning is the practice of displaying text that corresponds to spoken narration or other sounds being played. Using the captioning behaviors in addition to the text-to-speech behaviors lets you make your movies accessible to users with all types of disabilities.

There are two captioning behaviors that are designed to be used together. Each is designed to be used with the keyboard navigation and text-to-speech behaviors. For information about applying text-to-speech behaviors, see "Enabling text-to-speech" on page 431.

To enable captioning, you attach the Accessibility Captioning behavior to an empty text sprite that displays the captions. Next, you attach the Accessibility Sync Caption behavior to a sprite that has already had speech enabled with the text-to-speech behaviors.

To apply the Accessibility Captioning behavior:

1 Place the text sprite that you want to display caption text on the Stage.

2 Drag the Accessibility Captioning behavior from the Library palette to the text sprite.

3 In the dialog box that appears, select the behavior group name for the scene. This step associates the accessibility behaviors in the scene with one another so they work properly.

4 Click OK.

Next, you attach the Accessibility Sync Caption behavior to a sprite that has already had speech enabled.

To apply the Accessibility Sync Caption behavior:

1 Apply the keyboard navigation behaviors to sprites in your scene. For more information, see "Enabling keyboard navigation" on page 429.

2 Apply the text-to-speech behaviors to sprites in your scene. For more information, see "Enabling text-to-speech" on page 431.

3 Drag the Accessibility Sync Caption behavior from the Library palette to the sprite that triggers the text-to-speech feature. This could be a text sprite that has the Accessibility Speak behavior attached or another sprite that triggers speech with the Accessibility Speak Member Text behavior attached.

4 In the dialog box that appears, select the behavior group name for the scene. This associates the accessibility behaviors in the scene with one another so they work properly.

5 In the first At Word text box, enter the number of the word in the text that is to be spoken where you want the captioning to begin. For example, if you want to begin captioning at the first word of the spoken text, enter the number **1**.

6 In the Initial Words text box, enter the number of words to display after the starting word number that you selected in the previous step. This is the number of words that can appear in the captioning sprite when the captioning begins. For example if you want the first section of text to begin at the first word and end at word 15, enter **1** in the previous step and **15** in the Initial Words text box. These words are replaced in the captioning display sprite by later sections of the text being spoken. You select the number of words in each section by using the remaining text boxes in the dialog box.

7 In the next At Word text box, enter the number of the word where the second section of text you want to display begins. For example, if you want the second section of displayed text to begin at word 16 of the spoken text, enter **16** in the At Word text box.

Downloaded from www.Manualslib.com manuals search engine

8 In the Display How Many of the Words That Follow text box, enter the number of words to display after the word number you selected in the previous step. For example if you want to display 22 words of text after word 16, enter **22** in this text box.

9 In the remaining text boxes, enter the numbers of the first word of each section of text followed by the number of words in each section. To use more than five sections, drop this behavior on the sprite again. This procedure can be repeated as often as necessary.

10 When you have finished entering values in the dialog box, click OK.

Accessibility scripting with Lingo or JavaScript syntax

If you have a basic understanding of Lingo or JavaScript syntax, you can write custom scripts to add text-to-speech functionality to your movies. For more information about Lingo and JavaScript syntax, see the Scripting Reference topics in the Director Help Panel.

First, you must initialize the speech software.

To initialize the text-to-speech software:

- Use the `voiceInitialize()` method.

The following frame script tests whether text-to-speech software is installed. If no software is installed, the script displays an alert dialog box.

```
on exitFrame
  if voiceInitialize() then
    _movie.go("Start")
  else
    alert "Text-to-speech is not available"
  end if
end
```

To determine the number of available voices:

- Use the `voiceCount()` method.

To return a property list that describes the name, gender, age and index number of the current voice:

- Use the `voiceGet()` method.

To return a list of property lists that describes all the available voices:

- Use the `voiceGetAll()` method.

To set a particular voice as the current voice:

- Use the `voiceSet()` method.

After you select a voice for speech synthesis, you can control the progress of the speech.

To begin speech synthesis:

- Use the `voiceSpeak()` method.

To temporarily pause the speech:

- Use the `voicePause()` method.

  Some speech engines might continue to speak for several seconds after the pause method is used.

Downloaded from www.Manualslib.com manuals search engine

To resume the speech:

- Use the `voiceResume()` method.

To stop speech synthesis:

- Use the `voiceStop()` method.

To check whether the speech is currently speaking, paused, or stopped:

- Use the `voiceState()` method.

To set the volume of the voice:

- Use the `voiceSetVolume()` method.

To set the pitch of the voice:

- Use the `voiceSetPitch()` method.

To determine the chronological number of the current word within the string being spoken:

- Use the `voiceWordPos()` method.

The following frame script tests whether the current voice is female and starts speech if it is. In this case, the `voiceSpeak()` method specifies the text of the cast member named "TextCommentary".

```
on exitFrame
  voiceProps = voiceGet()
  if voiceProps.gender = "female" then
    voiceSpeak(member("TextCommentary").text)
  end if
end
```

For a complete list of script terms that control text-to-speech and keyboard navigation, see the Scripting Reference topics in the Director Help Panel.

Deploying accessible movies

To successfully deploy an accessible movie, you must become familiar with the Speech Xtra extension's system requirements and download procedure as well as the experience users have with your accessible movie in real-world situations.

Adding the Speech Xtra

If you use the text-to-speech feature in a projector, you need to include the Speech Xtra in your movie's Xtra extensions list. Normally, Xtra extensions that are not included in a movie's default Xtra extensions list are added to the list when a cast member that requires one is added to the cast. You can view the default Xtra extensions list by opening a new movie and selecting Modify > Movie > Xtras.

The Speech Xtra is a script-only Xtra, which means that it adds methods and properties to Lingo or JavaScript syntax, but does not add support for any new cast member types. Because the Speech Xtra is not associated with any kind of cast member, you must add it to the movie's Xtra extensions list manually.

Downloaded from www.Manualslib.com manuals search engine

To add the Speech Xtra to the movie's Xtra extensions list:

1 Open the movie.

2 Select Modify > Movie > Xtras.

3 In the dialog box that appears, click Add.

A second dialog box appears.

4 In the alphabetical list of Xtra extensions, scroll to Speech, and select the Speech Xtra. (In Windows, the file name is Speech.x32.)

5 Click OK.

The Add Xtras dialog box closes.

6 Click OK again.

The Movie Xtras dialog box closes.

7 Save your movie.

System requirements

Windows computers must have Microsoft Speech Application Programming Interface (SAPI) 4.0 or later installed. SAPI 5 is recommended. Windows XP includes SAPI 5. Earlier versions of Windows do not include SAPI by default. It can be downloaded from the Microsoft website at www.microsoft.com/speech/. A separate screen reader application is not necessary.

Macintosh OS 8.6 and later include text-to-speech software. No additions are necessary.

Understanding the Xtra download process

Because the Speech Xtra is from Macromedia, it is considered a trusted download. The user does not have to interact with any dialog boxes for the download to occur. When a user encounters accessible Shockwave content for the first time, the Xtra downloads automatically. The Xtra is approximately 45K on Windows and 35K on the Macintosh. After the download is complete, the movie begins to play. When the user encounters accessible movies in the future, no download occurs because the Xtra is already present on their computer.

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 21
Managing Memory

Macromedia Director MX 2004 has effective built-in memory management that is sufficient for most projects. To make memory available for new sprites, Director simply unloads the cast members used for sprites that are no longer on the Stage. However, sometimes large cast members, such as high resolution images, large sounds, or digital video, can take longer to load or unload than typical, smaller cast members. When a cast member takes a lot of time to load or unload, it can cause slight delays in movie playback.

If you test your movie on the lowest performance computers that you want it to be able to play back on, you can determine whether any of these delays occur and make changes to correct them. This chapter describes how Director's memory management works and the steps you can take to ensure smooth playback of movies with large amounts of media.

How Director unloads items from memory

To effectively manage memory while a Director movie is running, it is helpful to understand how automatic unloading of cast members works in Director. By becoming familiar with this process, you can make intelligent choices about when and how to perform memory management tasks yourself, if necessary.

A cast member is automatically loaded into memory when Director needs to draw a sprite of it on the Stage. Immediately after being drawn, each cast member is dealt with according to the value of its `purgePriority` script property. The default value of this property is 3. You can set this property in script or by selecting a number from the Unload menu in the Member panel in the Property inspector while the cast member is selected in the Cast.

The following are the possible values for `purgePriority`:

- When cast members with a purgePriority of 3 (normal) are no longer on the Stage, they can be unloaded from memory whenever Director needs memory for other tasks.
- Cast members with a purgePriority of 2 or 1 are only unloaded if memory is very low. They are added to the top of a list of recently used cast members that Director stores internally. This list is used to further prioritize cast members. Director assumes that the most recently used cast members are most likely to be used again and unloads them from memory only after all cast members with a `purgePriority` value of 3 are unloaded first.
- Cast members with a purgePriority of 0 are left in memory, and are not added to the recently used cast member list.

Downloaded from www.Manualslib.com manuals search engine

Monitoring memory use

The Memory inspector displays information about how much memory is available to Director for your movie and indicates how much memory different parts of the current movie use and the total disk space the movie occupies. It also can purge all removable items from RAM if you are about to perform a memory-intensive operation.

Note: The Memory inspector is not available on Macintosh.

To use the Memory inspector:

1 Select Window > Memory Inspector.

2 Observe the following memory use indicators:

Total Memory displays the total system memory available, including the amount of RAM installed on your computer and any available virtual memory.

Physical Memory shows the amount of actual RAM installed in the system.

Total Used indicates how much RAM is being used for a movie.

Free Memory indicates how much more memory is currently available in your system.

Other Memory indicates the amount of memory used by other applications.

Used by Program indicates the amount of memory used by Director (excluding the amount of memory used by the Director application file).

Cast & Score indicates the amount of memory used by the cast members in the Cast window and the notation in the Score window. Cast members include all the artwork in the Paint window, all the text in the Text windows, cast members that use the Matte ink in the Score, thumbnail images in the Cast window, and any sounds, palettes, buttons, digital video movies, or linked files that are imported into the cast and are currently loaded into memory.

Screen Buffer shows how much memory Director reserves for a working area while executing animation on the Stage.

3 To remove all items that can be purged from RAM, including all thumbnail images in the Cast window, click Purge.

All cast members that have Unload (purge priority) set to a priority other than 0–Never (as specified in the Property inspector's Member tab) are removed from memory. This procedure is useful for gaining as much memory as possible before importing a large file. Edited cast members are not purged.

Whenever Director runs low on memory, such as when trying to load many large cast members, other cast members are automatically unloaded from memory, according to the following rules:

- The first cast members to be unloaded are those that have a `purgePriority` value of 3. This type of discarding is fast but is essentially random, which means you cannot predict which of these cast members unload at any given time. The worst-case scenario is that in order to load the cast member for channel 1 of a particular frame, the cast member that is needed for channel 2 of that same frame gets unloaded.
- When there are no more cast members with a `purgePriority` value of 3 left to unload, Director starts unloading the cast members that have a `purgePriorty` value of 2 and were least recently used. This type of unloading is slower but can be more predictable.
- If more memory is still needed after the cast members with a `purgePriority` value of 2 are unloaded, then the least recently used cast members with a `purgePriority` value of 1 are unloaded.

Downloaded from www.Manualslib.com manuals search engine

Using script to change a cast member's `purgePriority` property is a good technique for controlling memory management. However, changes to the `purgePriority` property do not take effect until the next time a cast member is used. For example, if you change the property in an `exitFrame` or `enterFrame` script in the same frame as a cast member is used, the cast member is treated as if it had its old priority because all drawing on the screen is done before any `enterFrame` or `exitFrame` scripts are performed.

Loading and unloading individual cast members

There are several script methods you can use to force specific cast members to load into memory or unload from memory. By using these methods, you can unload cast members that you know are no longer in use to make room for new ones. You can also force large cast members to load before they are actually needed. This prevents a pause that can occur when large cast members load normally, just before being drawn on the Stage.

Use the following methods to control cast member loading and unloading:

- To load a specific cast member or set of cast members, use the `preLoadMember()` method.
- To load all the cast members used in a specific frame or range of frames, use the `preLoad` method.
- To unload a specific cast member or set of cast members from memory, use the `unLoadMember()` method.
- To unload all the cast members used in a specific frame of your movie, use the `unload` method.
- To determine the number of bytes of memory required to display a range of frames, use the `ramNeeded()` method. To convert the number of bytes to kilobytes, divide the result by 1024.
- To determine the current amount of available memory, use the `freeBytes()` or `freeBlock()` method.

There are several more script terms related to memory management. For a complete list, see the Scripting Reference topics in the Director Help Panel.

Preloading digital video

It is recommended that you do not preload digital video cast members. Digital video is played by streaming the video file from a disk. As the file is streamed, it is decompressed into memory one section at a time. Preloading a digital video file causes the entire file to be decompressed into memory at once, which can cause low memory situations on most computers.

You can cause a digital video to preload only its first segment without consuming unnecessary memory by placing it in the Score a frame or two before it is actually needed.

To preload a digital video safely:

1. Add a sprite of the digital video to the Score.
2. Begin the sprite one or two frames before the frame where you want to display the video.
3. Locate this sprite off the Stage except for at least one pixel of one corner of the sprite. This is so the user won't notice it.
4. Set the sprite's `movieRate` property to 0, which prevents the movie from playing when first loaded.

Downloaded from www.Manualslib.com manuals search engine

5 In the frame where you want the video to appear, use script to set the sprite's `loc` property to place the sprite on the Stage in the location you select.

6 Set the `movieRate` property to 1 to start playing the video.

This way, the video loads its initial segment into memory and is ready to play immediately when your movie reaches the frame where you want it to appear. Experiment with preloading the video a few frames earlier if you find that preloading has not yet finished when your movie reaches the frame where the video appears.

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 22
Managing and Testing Director Projects

As you work with Macromedia Director MX 2004 and develop projects with it, you might encounter some situations where a seemingly small change in the design of a project has a significant effect on the organization of the Director files and media files associated with it. A little bit of planning before you start a Director project can help avoid difficult changes during the course of development. Good testing practices can help you discover problems early in the project while they are still small and easily remedied.

This chapter provides a few simple guidelines that can streamline your development process.

Managing Director projects

By carefully managing the resources that go into your Director movies, you can avoid problems that might arise when you make changes to the movie or change the location of the movie file or its linked media.

The following guidelines can help make your project go smoothly:

- Before you begin a project, plan where media should be located when you deploy your movie and replicate that organization at the beginning of your project. This prevents the links between your Director movies and external media files from being broken when you move the project to a different location on a disk or to a different disk volume.

  The following structure is a good example of a simple file organization:

  ```
  Project Directory
    contains:
    MyDirectorMovie.dir
    Linked_Media subdirectory
      contains:
      sound files
      graphic files
      digital video files
      other linked assets
  ```

  By establishing the final organization of your files before you import them into Director, you prevent the need to update links later in the project.
- When you work on a large project, plan your basic approach to all aspects of the movie before you start construction. This way, you can find problems with your strategies before you have invested a lot of time building on them. Finding problems early in a project makes them much easier to solve.

Downloaded from www.Manualslib.com manuals search engine

- Organize the cast members in your cast in a logical way. You might choose to group all the cast members of a particular type together, or you might choose to group the cast members from each scene together. Choose a system that works for you and that will make it easy to find cast members when your cast becomes large. You can also choose to keep groups of cast members in separate casts.
- When referring to cast members, sprites, and frames in scripts, use the name of the cast member or sprite or the name of a frame marker. This avoids the need to change your code if you need to rearrange cast members, sprites, or frames during the project.

  For example, the following script refers to a cast member by its cast member number:

  ```
  member(16).text = "Good planning makes Director projects easy."
  ```

  If the text cast member 16 has to be moved in the cast, the script becomes invalid.

  Instead, use the following script:

  ```
  member("Output_text").text = "Good planning makes Director projects easy."
  ```

  Naming sprites makes changes to your script even easier. For example, the following script refers to a sprite by its number:

  ```
  sprite (1).text
  ```

  Instead, use the following script:

  ```
  sprite("input").member.text
  ```

  When you add a marker to a frame and use the marker name to refer to the frame, you can move the marker without breaking the script.

  The following script refers to a frame by its number:

  ```
  _movie.go(27)
  ```

  It is better to add a marker to the frame and use the marker name, as in the following script:

  ```
  _movie.go("Main_menu")
  ```

- During work sessions, save your movie file often. Save a copy of the movie after each milestone, such as a day of work or after adding a significant new feature or section. This way, if problems arise, you can easily compare the current version of the file to a slightly older version to locate the source of the problem. Keep several copies of your file at different stages of development in case you need to go back several steps.

Downloaded from www.Manualslib.com manuals search engine

About testing movies to avoid problems

While you develop movies, you might encounter some difficulties because creating interesting movies and trying out new ideas always involves some experimentation. By testing your movies according to the simple guidelines described in this section, you can prevent problems from becoming obstacles.

Testing early in development

When you begin a Director project, it is a good practice to test the functionality of your movie early in the development process to help ensure that you discover any problems while they are still minor. Waiting to test lets small problems become larger ones as you add features to your movie that depend on problematic functionality that was implemented earlier. By incorporating testing into your authoring process early, you will find these problems and have the opportunity to fix them before adding features to your movie.

Testing often during development

Testing should be an integral part of your Director development process. You should test the functionality of each small part of your movie as you add it rather than waiting until the movie's whole feature set is implemented.

When you build features that are interdependent, you should test each one before adding the next. If you test this way, you know that the most recently implemented feature is the most likely source of the problem. If you wait to test one feature until after the next feature is implemented and one of those features exhibits a problem, you have a more complex set of possibilities to evaluate.

Save multiple versions of your movie as you progress. When difficulties arise, compare the current version with the last saved version to help locate the source of the problem.

Testing on all target platforms

When you develop a Director movie, you should spend some time defining its audience. Part of this process is deciding what the minimum system requirements should be for the computers used by that audience.

You should determine the slowest processor speed you want your movie to play on and verify that the performance of your movie is acceptable on a processor of that speed. You should also determine if there is a range of configurations you have to support (such as Macintosh, Windows 2000, and Windows XP) and test enough of them to ensure success. Be sure to include parameters such as browser software, screen resolution, and available memory in your testing.

This approach can help you find problems that are specific to an operating system or configuration, which are distinct from authoring errors.

Downloaded from www.Manualslib.com manuals search engine

Testing strategies

You can use the following strategies to test your movies effectively:

- Testing in the Director authoring environment is different than testing in the Director Projector and Macromedia Shockwave environments for online content. The Preview in Browser command demonstrates the true behavior of the movie in the Shockwave environment.
- If you encounter a problem, try to isolate the problem in a new Director file that incorporates only the problem feature or item. Make a list of the minimum steps that are required to reproduce the problem in a new file. This process usually reveals the source of the problem in your movie. It also reveals whether the problem is limited to one feature or if it is caused by the interaction of two or more features in your movie.
- Try to re-create the problem with different media. Sometimes the source of a problem is within a specific media item used in your movie.
- Try to re-create the problem on a different computer. This helps isolate problems with hardware configuration or with the Director installation on a specific computer. If the problem exists only when the movie is posted to a server, determine whether the problem exists on only one server or all servers. Occasionally the server's MIME types might need editing to include the MIME types for Director. For more information, see Director TechNote 16509, Configuring your server for Macromedia Shockwave Player, at www.macromedia.com/support/director/ts/documents/shockwave_config.htm.
- When you use script, look for typing errors, missing punctuation, or inconsistent naming.
- JavaScript syntax is case-sensitive, while classic Lingo is not.

Printing movies

You can print movie content to review it and mark changes, to distribute edits to a team, to make handouts from a presentation, or to see your work on paper. You can print a movie while in authoring mode in several ways. You can print an image of the Stage in standard or storyboard format, the Score, the cast member number and contents of text cast members in the Cast window, all scripts or a range of scripts (movie, cast, the Score, and sprite scripts), the comments in the Markers window, the Cast window artwork, or the entire Cast window.

You can also use script to control printing. For more information, see the Scripting Reference topics in the Director Help Panel.

To print part of a movie:

1 Select File > Print.

2 To specify what part of the movie to print, select an option from the Print pop-up menu.

You can print an image of the Stage, the Score, all scripts or a range of scripts (movie, cast, the Score, and sprite scripts), cast text, cast art, cast thumbnails, and the comments in the Markers window.

The Scripts, Cast Text, Cast Art, and Cast Thumbnails print options specify a range of casts and cast members—internal or external. Information that appears in the Print dialog box depends on the selection to be printed.

Note: Selecting Cast Text on the Print pop-up menu, prints a table of text cast members your printer's set resolution.

Downloaded from www.Manualslib.com manuals search engine

3 To specify which frames of your movie are printed, select one of the following Frames options:

Current Frame prints the frame that is currently on the Stage.

Selected Frames prints the frames that are selected in the Score.

All prints all of the frames in your movie.

Range prints the range of frames specified in the Begin and End text boxes.

4 To specify which frames in the defined range to print, select one of the following Include options:

Every Frame is the default setting and prints every frame that is specified in Range.

One in Every [number] Frames prints frames at the interval you specify in the text box. For example, if you enter 10, Director prints every tenth frame.

Frames with Markers prints only the frames that have markers in the Score window.

Frames with Artwork Changes in Channel [number] prints the frames in which cast members move or in which new cast members are introduced in the Score. Specify the channel number in the text box.

5 To determine the layout of the items to print, click Options and select from the following:

Scale provides options to print at 100%, 50%, or 25% of the original size.

Frame Borders creates a border around each frame.

Frame Numbers prints the frame number with each frame.

Registration Marks places marks on every page to align the page for reproduction.

Storyboard Format is available only when you select 50% or 25% images to print. This option places marker comments next to the frame image.

Date and Filename in Header prints a header on each page. The header consists of the name of the Director movie and the current date.

Custom Footer prints a footer on each page. Type the footer in the text box.

The image at the left of the dialog box previews the layout options.

Resources

The following resource is available to help you with testing and troubleshooting your Director movies: index of Director testing and troubleshooting TechNotes at www.macromedia.com/go/director_troubleshooting

Downloaded from www.Manualslib.com manuals search engine

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 23
Packaging Movies for Distribution

When you finish authoring your Macromedia Director MX 2004 movie, you have a choice of several ways to prepare it for distribution. You can distribute a movie in the Macromedia Shockwave format that plays in a browser, or you can distribute it as a stand-alone projector. Stand-alone projectors can contain the software necessary to play the movie, or they can use an installed Macromedia Shockwave Player to play the movie independent of a browser. You can also export a movie as a digital video.

You can use several Director features to prepare movies for distribution. These features include determining Publish settings and deciding which Xtra extensions to include, exclude, or download. You can also preview your movie in a browser and batch-process movie files to compress them and protect them from being edited.

About distributing movies

When you finish creating a movie, you have several choices about how to distribute it to users. You can distribute the movie as Shockwave content that plays within a web page or as a projector that downloads to the user's computer or is distributed on a disk.

- Shockwave content is a compressed version of the movie data only.
- A projector is a stand-alone version of a movie. You can include one movie in a projector that links to other external movies or include several movies in a single projector. Projectors are handled by the system as executable application files.

Movies that are distributed from the Internet can begin playing as soon as the content for the first frame is downloaded. This process is called streaming. You can control streaming with behaviors that make the movie wait for media at certain frames, or you can specify that a movie download completely before it begins playing. For more information, see "Setting movie playback options" on page 471.

- To create Shockwave content that can play in a web page, you use the File > Publish command. Director leaves your original movie in its DIR format. Director also creates Shockwave content in the DCR format.

If you use the default Publish settings, Director creates an HTML page that is completely configured with `EMBED` tags and everything else you need to run your movie in a browser. By default, Director saves all these new files in the same folder as your original Director movie. For more information about putting your Director movie on the web, see "Creating Shockwave content" on page 452.

Downloaded from www.Manualslib.com manuals search engine

For more information about how to distribute Xtra extensions with projectors, see TechNote 13965 in the Director Support Center at www.macromedia.com/go/director_support. Although the note might refer to Director 7, the information is the same for more recent versions of Director.

Shockwave browser compatibility

Shockwave can play Director movies in the following browsers on the listed platforms:

- For Microsoft Windows: An Intel Pentium II with 64 MB of available RAM running Windows 98, or an Intel Pentium III with 128 MB of available RAM running Windows 2000 or Windows XP; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.01 Service Pack 2, Microsoft Internet Explorer 5.5 Service Pack 2, Microsoft Internet Explorer 6 Service Pack 1, and a color monitor
- For Macintosh OS X: A Power Macintosh G3 with 128 MB of available RAM running Mac OS X 10.2.6 or 10.3; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.2 or later, or Safari; and a color monitor
- For Macintosh Classic: A Power Macintosh G3 with 64 MB of available RAM running System 9.2; Microsoft Internet Explorer 5.1; and a color monitor

When it first encounters an HTML page that references Shockwave content, Internet Explorer for Windows asks the user for permission to download the Shockwave ActiveX control if it is not already installed. If the user approves, it downloads and installs the control.

Previewing a movie in a browser

You can preview a movie in a browser on your local computer to view JPEG-compressed bitmaps, and to check the movie design, script, and any other performance issues related to playing a movie in a browser. Previewing a movie creates temporary Shockwave (DCR) and HTML files that open in a browser.

Note: When you use the Publish command rather than the Preview in Browser command (with the Preview after Publishing option selected in the Publish Settings Format tab), you can create permanent DCR and HTML files. These permanent files (as opposed to temporary ones created simply by using the Preview in Browser command) can then be placed on a web server for viewing with a browser via http.

You may notice that linked media do not work as expected when you preview a movie in a browser. Because of security restrictions, movies playing in browsers may not be able to read files from a local disk unless they are in the dswmedia folder (also called the support folder), which is a subfolder of the folder containing the Shockwave Player. Therefore, to preview a movie that uses linked media, you may need to put the movie and all of its linked media in the dswmedia folder. The movie can open any file in a subfolder of dswmedia, provided that the relative paths have not changed. If you move the movie and its media to another server, the linked media will continue to work if you preserve the same folder structure. For more information, see "Using dswmedia folders and the support folder to publish Shockwave content".

To specify the browser to use for previewing:

1 Select Edit > Preferences > Network.

Note: If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.

2 In the Preferred Browser box, enter the path to the browser application file.

Downloaded from www.Manualslib.com manuals search engine

To preview a movie in a browser:

- Select File > Preview in Browser or press F12.

About Xtra extensions

All Xtra extensions a movie requires must be installed on your user's system when the movie runs. When you distribute a movie, you must either include these Xtra extensions or provide the user with the means to download them. Using the Movie Xtras dialog box (select Modify > Movies > Xtras), you can specify the Xtra extensions to include in a projector and whether Xtra extensions should download for use with Shockwave content. The Movie Xtras dialog box contains a list of the most commonly used Xtra extensions. Including all these Xtra extensions ensures that your movie will work in most cases but makes the projector much larger. You may want to remove Xtra extensions you know you aren't using to help manage memory.

Each time you create a sprite that requires an Xtra, Director adds the Xtra to the list of required Xtra extensions in the Movie Xtras dialog box. If you remove the sprite, Director does not remove the Xtra from the list, in case you later re-create the sprite. Director cannot detect Xtra extensions required in Lingo or JavaScript syntax. You must manually add any Xtra extensions required by your scripts to the list in the Movie Xtras dialog box. For more information, see "Managing Xtra extensions for distributed movies" on page 450.

Managing Xtra extensions controls the size and capabilities of the movie you distribute. Many important features in Director, such as text and vector shapes, are controlled by Xtra extensions, as is the ability to import all types of linked media. If you don't use a feature or import a media type that is controlled by an Xtra, you should not distribute the related Xtra with your movie. This is especially true for movies distributed on the Internet.

Including Xtra extensions

The Shockwave Player includes the Xtra extensions that support the most common features and media types. These include text; vector shapes; Macromedia Flash; BMP, PICT, JPEG, and GIF file importing; sound management; and Shockwave Audio.

Xtra extensions not included with the Shockwave Player must be installed in a user's system before the movie plays. Use the Download If Needed option in the Movie Xtras dialog box to make the movie prompt the user to download the Xtra. Director downloads Xtra extensions from the URL specified in the Xtrainfo.txt file in the Director application folder.

Xtra extensions downloading from projectors requires use of Lingo or JavaScript syntax. For more information, see the Scripting Reference topics in the Director Help Panel.

Xtrainfo.txt includes URLs for all Macromedia Xtra extensions included with Director, but you may need to manually edit Xtrainfo.txt to add the URL for third-party Xtra extensions or Macromedia Xtra extensions not included with Director. Xtrainfo.txt includes a description of how to enter this information. Xtra developers may also provide installation programs or other means of modifying Xtrainfo.txt automatically.

If a user chooses to download an Xtra extension, Director retrieves the Xtra extension from the URL specified in Xtrainfo.txt using the Verisign download security system. Verisign is a standard means of downloading software from secure sources.

You can also include Xtra extensions in projector files. Select the Include in Projector option in the Movie Xtras dialog box for any Xtra extension you want to include.

Downloaded from www.Manualslib.com manuals search engine

You can also exclude Xtra extensions in projectors. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Xtra extensions usually required for the movie to play back correctly include the following:

- Xtra extensions that create cast members (text, Flash, vector shapes, Windows Media, QuickTime, and so on)
- Shockwave Audio Xtra extensions (if the movie uses files in the SWA format)
- Transition Xtra extensions (if the movie uses third-party transitions)
- Import Xtra extensions, if the movie uses nonstandard types of linked external cast members
- Network Xtra extensions required for a movie to access the Internet
- Scripting Xtra extensions (if the movie uses any special script that requires Xtra extensions).

Managing Xtra extensions for distributed movies

You can manage Xtra extensions for your movie using the Modify menu.

To manage Xtra extensions for the current movie:

1 Select Modify > Movie > Xtras.

2 To add or remove Xtra extensions, do any of the following:

- To add the Xtra extensions required to connect a projector to the Internet, click Add Network.
- To restore the list of default Xtra extensions, click Add Defaults.
- To manually add an Xtra extension to the list (which you would do, for example, if you've used Lingo or JavaScript syntax that requires Xtra extensions), click Add and select from the list of Xtra extensions installed in your system.
- To delete an Xtra extension from the list, highlight the Xtra extension and click Remove.
- To get information about an Xtra extension (when available), highlight the extension and click Info.

3 To change settings for Xtra extensions in the list, select an Xtra extension and select either of the following options:

Include in Projector makes Director include the selected Xtra extension in any projector that includes the current movie.

Download If Needed makes the movie prompt the user to download a required Xtra extension if it is not installed in the user's system. The Xtra extension is downloaded from the location specified in the Xtrainfo.txt file and permanently installed in the user's system.

4 To get information about a selected Xtra extension, click Info.

The information comes from an Internet source. Not all Xtra extensions include information. Third-party Xtra extensions often include some explanations and information about the developers.

Note: Another way to include Xtra extensions with a movie is to create an Xtras folder containing all required Xtra extensions in the same folder as a projector file. This allows you to see which Xtra extensions are included without opening the movie. If you use this method, you cannot include Xtra extensions in the projector file because the movie will fail to initialize.

Downloaded from www.Manualslib.com manuals search engine

About distribution formats

Before deciding how to distribute a movie, it helps to understand how Director plays movies. Director movies play either with the Shockwave Player or through a projector player. The Shockwave Player is a system component that plays movies in web browsers and also outside browsers as stand-alone applications. A projector player can only play movies independently of a web browser.

You can distribute movies as Shockwave content (with the DCR extension), projectors, Shockwave projectors, or protected movies (DXR extension). You should not distribute source movies (DIR extension) unless you want your users to be able to change the movie in the Director authoring environment.

- Shockwave content is a compressed version of a movie's data and does not include a player. Shockwave content is created primarily to distribute over the Internet for playback in a web browser. Another reason to create Shockwave content is to compress it for distribution on a disk when the movie is contained in a projector. In addition to compressing the data, saving a movie in the Shockwave format removes all information necessary to edit the movie.
- A projector is a movie intended for play outside of a web browser. A projector can include a player (called the Standard player), Xtra extensions, multiple casts, and linked media in a single file. A projector can also include several different movie files. Configured in this way, a projector can be a completely stand-alone application.
- A Shockwave projector makes a much smaller projector. A Shockwave projector uses an installed Shockwave Player on the user's system to play a movie instead of including the player code in the projector itself. If no Shockwave Player is installed on the user's system, the user must download a copy. A Shockwave projector is excellent for distributing movies on the Internet that you don't intend to play in a web browser.

  You can also reduce the file size of a projector by turning on projector options that compress the movie data, the player code, or both. In Windows, compressing the player code reduces the minimum projector size from approximately 2.1 MB to 1.1 MB for a projector, and to about 60K for a Shockwave projector.

  On the Macintosh, compressing the player code reduces the minimum projector size from approximately 2.5 MB to 1.2 MB for a projector, and to approximately 12K for a Shockwave projector.
- Protected movies (.DXR extension) are uncompressed movies that users can't open for editing. These can be useful when you want to distribute uncompressed movies on a disk, but you don't want users to edit the source file. Protected movies may play faster than Shockwave content from a disk because they do not need to be decompressed. These movies are preferable if disk space isn't limited. Like Shockwave content, protected movies do not include the information necessary to edit the movie or the software that plays the movie. They can be played only by a projector, a movie in a window, or the Shockwave Player.

Note: To edit a movie packaged for distribution, you must edit the source file (DIR) and create a new movie in one of the distribution formats. Always save your source files.

Using linked media on the Internet

When you distribute a movie on the Internet for playback in a web browser, the linked media must be at the specified URL when the movie plays. Otherwise, the user receives an error message.

Downloaded from www.Manualslib.com manuals search engine

Distributing movies on a disk

Whenever a movie plays from a disk, it accesses all external linked files the same way that it did in the authoring environment. All linked media—bitmaps, sounds, digital videos, and so on—must be in the same relative location as they were when you created the movie. To make sure you don't forget any linked media when you distribute a movie on a disk, place linked files in the same folder as the projector or in a folder inside the Projector folder.

If your movie includes Xtra extensions, you must include the Xtra extensions in the projector. If a movie distributed on a disk connects to the Internet in any way, be sure to click the Add Network button in the Movie Xtras dialog box.

Distributing movies on a local network

If you plan to place a movie on a local area network (LAN), all files must be set to read-only, and users must have read/write access to their system folders. Otherwise, the requirements are the same as for normal disk-based distribution.

Creating Shockwave content

You save your work as a Shockwave (DCR) movie to prepare it for playback in a Shockwave-enabled web browser, or to make disk-based movies smaller. Using Shockwave content also prevents your users from editing the movie if they own Director.

If the Shockwave content you're creating will be distributed on the Internet and requires any Xtra extensions, make sure the Xtra extensions are listed in the Movie Xtras dialog box and that Download If Needed is selected for each required Xtra extension. For more information, see "Managing Xtra extensions for distributed movies" on page 450.

Note: Use Update Movies to convert several movies at once to the Shockwave format. For more information, see "About cross-platform projectors" on page 463.

For more information about Shockwave content, see Chapter 24, "Using Shockwave Player," on page 469.

Using default Publish settings

To create Shockwave content, you use the File > Publish command. The default setting creates a Projector file with Preview enabled.

For more information about changing Publish settings, see the next section.

To create a Projector movie using default settings:

1 Select File > Publish.

2 Save your movie if prompted to do so.

Director creates and automatically starts a Projector version of your movie.

Director creates a Shockwave version of your movie, and an HTML file. Your default browser opens with the HTML page you just created.

Note: Your default browser is specified in your Network Preferences dialog box. To change your default browser, select Edit › Preferences › Network. (If you are using a Macintosh OS X operating system, select the Director menu, instead of the Edit menu, to access Preferences.)

Downloaded from www.Manualslib.com manuals search engine

If your movie needs Xtra extensions that fall beyond the range of the default publish settings, (for example, Windows Media) you will be prompted to add them. In that case, you can set up your movie with the correct settings by changing them.

Changing Publish Settings

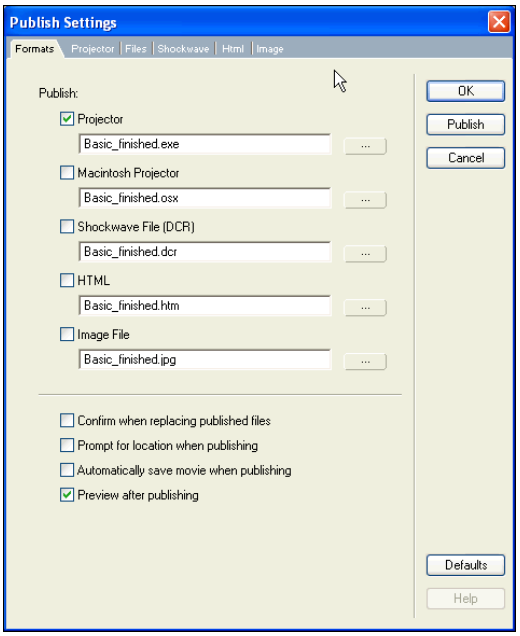
You can change Publish Settings by using the Publish Settings dialog box. Publish Settings lets you determine what type of Director movie you want to create, and what properties that movie shall have.

To access Publish settings:

- Select File > Publish Settings.

  The Publish Settings dialog box appears with the following tabs: Formats, Projector, Files, Shockwave, Html, and Image.

  Each Publish Settings tab contains the following buttons: OK, Publish, Cancel, Save As Defaults, Defaults, and Help.



- To save any changes you have made, click OK.
- To Publish your movie with the selected settings, click Publish.
- To Cancel any changes you have made, click Cancel.
- To save the current settings of the selected tab as default settings, click Save As Defaults.

  Note: Clicking the Save As Defaults button on one tab has no effect on any of the other tabs.

- To return settings of the currently selected tab to their saved default state, click Defaults. To return settings of the currently selected tab to their original default state, hold down Alt (Windows) or Option (Macintosh) and click Defaults. If you have not saved default settings, clicking Defaults without holding down the key returns the settings to their original default state.

  Note: Clicking the Defaults button on one tab has no effect on any of the other tabs.

- To access help, click Help.

Downloaded from www.Manualslib.com manuals search engine

Selecting a publishing format

Use the Formats tab to select the movie format you want and determine the types of files you wish to publish. It includes support for creating cross-platform projectors.

To set options using the Formats tab:

- To publish a Projector movie, click Projector. You can set more Projector options by using the Projector tab. For more information, see "Setting Projector options" on page 455.
- To publish a projector that will run on a different platform than the one in which you are authoring, select Macintosh projector (Windows) or Windows projector (Macintosh). For more information about creating cross-platform projectors, see "About cross-platform projectors" on page 463.
- To publish a Shockwave file (DCR) without an HTML file, select Shockwave File. You can set additional Shockwave options using the Shockwave tab. For more information, see "Setting Shockwave options" on page 456.
- To publish Shockwave content with an HTML file, select HTML; you can set additional Shockwave HTML properties using this tab. For more information, see "Setting HTML options" on page 456.
- To publish an Image file, select Image file. You can set additional Shockwave image options using this tab. For more information, see "Setting Image options" on page 458.
- To be prompted if you want to replace existing files, select Confirm when replacing files.
- To be prompted for the location you want to publish to, select Prompt for location when publishing.
- To automatically save your movie when publishing (instead of being prompted to save as is the case with default settings), select Automatically save when publishing.
- To change the default setting of automatically previewing your movie in a browser after publishing, deselect Preview after publishing.

Using the DCR and HTML file format

If you create a DCR file and an HTML file with all of the tags necessary to display your DCR movie, Director does the following:

- Creates a DCR and HTML file in the same directory as your Director (DIR) movie.

  Note: Director creates a CCT file for each external cast and, by default, saves the CCT file in the same folder as the DCR file. To specify a different file location, hold Alt (Windows) or Option (Macintosh) when you select File › Publish. Continue to hold the key for access to dialog boxes that let you specify new paths for both your DCR and CCT files.

- Gives both your DCR and HTML files the same name as your DIR file, with the appropriate extensions (for example, MyMovie.dcr and MyMovie.html).
- Sets the DCR movie's width and height to match the dimensions of the DIR movie.
- Configures the DCR movie and HTML file so that if your users resize their browsers, the DCR movie remains the same size as the original DIR movie.
- Compresses bitmap images and sound using JPEG compression. Note that if you've compressed images for individual cast members, those settings will override compression Publish settings are movie specific; that is, if you change the default Publish settings, Director saves those changes when you save your movie. A new movie uses the Default Publish settings, not those set for a specific movie.

Downloaded from www.Manualslib.com manuals search engine

Setting Projector options

The Projector tab lets you set options that include how the projector movie appears on stage or in a browser, and what type of player it is shown on.

To set options using the Projector tab:

1 Select File > Publish Settings and click the Projector tab.

2 Set the options you want and click OK to keep your settings and return to the movie or Publish to begin the projector publishing process.

For more information about setting Projector options, see "Creating projectors" on page 460.

Setting Files options

The Files tab lets you set file options for projectors. For example, by default a projector consists of just the current movie, but you can add external casts to the movie using this tab. You can also exclude Xtra extensions from your projector.

To set options using the Files tab:

1 Select File > Publish Settings and click the Files tab.

2 To set the Primary movie components for your projector, select from the following options:

Include linked cast files lets you include automatically all linked cast files to the primary movie.

Exclude all xtras lets you exclude some or all Xtra extensions from your projector. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Compress files (Shockwave format) lets you compress files for a Shockwave projector as you would for standard Shockwave content.

Add Files opens a dialog box that lets you select and add Director Movie or Cast files to the current projector.

Remove All deletes all added files.

Play every movie in list ensures all movies in your movie list play.

Loop sets the movie to keep playing.

3 Click OK to save changes and return to the movie.

4 Click Cancel to reject changes and return to the movie.

5 Click Publish to publish the projector with these settings.

Downloaded from www.Manualslib.com manuals search engine

Setting Shockwave options

The Shockwave tab lets you set options that determine the Shockwave Player version, how images are compressed, how users interact with the movie once it's published, and more.

To set options using the Shockwave tab:

1 Select File > Publish Settings and select the Shockwave tab.

2 Select the options you want to provide for your users, as described in the following list:

Version:

- Select Shockwave Player 10 or Shockwave Player 8.5.

Image compression:

- **Standard:** Select Standard to apply compression techniques used by Director in versions 4 through 7, select Standard. This setting is suitable for graphics with few colors.
- **JPEG:** Select JPEG and specify the image quality setting by moving the slider to a value between 0 and 100 percent. The higher the percentage, the less the image is compressed.

Compression enabled lets you compress the sound in your movie. Select this option and select the level of compression from the kBits/second pop-up menu. For more information about sound compression, see "Compressing internal sounds with Shockwave Audio" on page 237.

Convert Stereo to Mono lets you convert stereo audio to monaural. This option is only available if Shockwave Audio Compression is enabled.

Enabled context menu items:

Include Cast Member Comments lets you include comments you might have entered in the Comments field of the Property inspector for your cast members. You can then use Lingo or JavaScript syntax to access the comments in the DCR file.

- **Volume Control** lets users adjust the volume of the movie's soundtrack.
- **Transport Control** provides controls for rewinding, stopping, and stepping through the movie.

Setting HTML options

If users view your movie in browsers, chances are they will resize their browsers. How your movie behaves when the browser size changes depends on how you set HTML options in the Publish Settings dialog box.

To set HTML options for browser resizing:

1 Select File > Publish Settings.

2 On the Html tab of the Publish Settings dialog box, select from the Dimensions pop-up menu. When you make a selection, the width and height values default to the movie size.

- To change the background color of your HTML file, either click the Page Background color box and select a color, or enter a value in the hexadecimal field.
- To make the DCR movie match the dimensions of your DIR movie, select Match Movie in the Dimensions field.

Downloaded from www.Manualslib.com manuals search engine

3 If you use the default Match Movie setting in the Dimensions field, values in the `OBJECT` and `EMBED` tags in the HTML file are set to the exact dimension of your movie. To change the dimensions, select either Pixels or Percent of Browser Window, and type the new dimensions in the Width and Height fields. Your movie will resize to fit the new rectangle only if you have not selected No Stretching in the Stretch Style pop-up menu on the Shockwave tab.

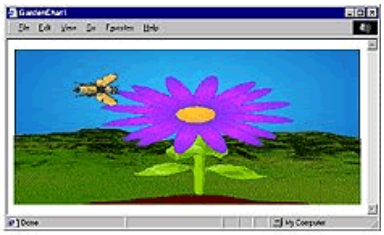
- To create an HTML file with parameters that match the height and width of the movie, select Match Movie.
- To specify height and width values in the HTML file in pixels, select Pixels.
- You can select Percentage of Browser Window, and specify a percentage in the Width and Height fields. (To make browser resizing affect the size of the DCR movie, you must specify percentages and select either Preserve Proportions, Stretch to Fill, or Expand Stage Size on the Shockwave tab of the Publish Settings dialog box.)

4 On the Html tab, select an option from the Stretch Style pop-up menu.

- To specify that your movie not resize at all, select No Stretching.
- To maintain the same aspect ratio of your original Director movie no matter what size the user makes the browser, select Preserve Proportions. The movie will fit within the width and height parameters, as much as possible, while preserving the movie's aspect ratio. The movie aligns within the window based on the align tags that you specify in step 4.




- To change the size of the movie to fit the size of the browser, select Stretch to Fill. Any browser resizing stretches the movie to fill the width and height parameters. Note, however, that if the aspect ratio of the movie changes, sprites on the Stage could appear distorted. If you select Stretch to Fill, Director ignores the align tags that you specify in step 4.

Downloaded from www.Manualslib.com manuals search engine

To let users resize the Stage without resizing the sprites, select Expand Stage Size. The movie is aligned within the browser based on the align tags that you specify in step 4.





5 To specify align tags for your movie, use the Horizontal Align and Vertical Align pop-up menus. You can select left, center, or right horizontal alignment, and top, center, or bottom vertical alignment.

Setting Image options

You can specify the images that should appear if the user doesn't have Shockwave or the ActiveX control.

To set options using the Image tab:

1 Select File > Publish Settings and select the Image tab.

2 Select from following Alternate Shockwave image options

- In the Poster Frame field, enter the frame number from your movie's Score that you want to appear as a JPEG image for users who are unable to view your movie.
- To specify compression for the image, move the Quality slider to the desired compression setting. The higher the percentage, the less the image is compressed.
- To specify that the image download as a progressive JPEG, select Progressive. The JPEG will then display at low resolution and increase in quality as it continues to download. Making a JPEG progressive also reduces its file size.

3 Select Publish to publish the movie with the chosen settings or click Cancel to cancel any changes made.

4 Select Default to return the image options to their default setting.

5 Click OK to save the settings and return to the movie.

Downloaded from www.Manualslib.com manuals search engine

Using dswmedia folders and the support folder to publish Shockwave content

Use a dswmedia folder to obtain data from a local source while a movie is playing in safe player mode. Director treats folders named dswmedia as exceptions to the safe player mode feature that normally stops movies from reading data from any local source. Any linked media or cast files in a dswmedia folder, or a subfolder of any depth, may be read by a movie running in safe player mode. You can use relative links or full file paths to files in dswmedia folders.

To test movies in a browser locally before uploading them to your web server, place the movie, linked casts, and linked media in a directory tree within a dswmedia folder and use relative links. When you move the movie and its media to a web server for testing, keep the directory structure intact. You must use file and folder names which do not have spaces or capital letters, and which have recognized file extensions like .dcr and .gif, in order for them to be accessible from your server.

To access local media while a movie is playing in safe player mode, install the media in a dswmedia folder in a known location on the user's computer. This location can be a CD that you provide.

Using linked media when testing Shockwave content

Linked media do not show up when testing Shockwave content from a local machine. However, when the movie is uploaded to a server, the media show up fine.

Shockwave content is played in what is called 'safe player mode,' which is a feature specific to Director movies. When a movie is played in safe player mode, it is restricted. The movie cannot read or write files on the user's local disk, except to get or set a movie's preferences. Therefore, linked files do not show up when played on a local machine.

There is one exception to the aforementioned restriction. A movie can read files that are stored in a folder named 'dswmedia' (or a sub folder).

To enable linked files to show on a local machine, follow the steps below:

1. Create a folder named 'dswmedia'.
2. Place all linked media in the 'dswmedia' folder or Place all files including the Director file in a dswmedia folder in order to maintain file and folder structure.

Note: Be sure to maintain the original file and folder structure of your project, so as not to break any links.

Converting movies created in previous versions of Director

Director MX 2004 can convert movies from Director 5 and later. It's not necessary to update movies created in Director 8 or Director 8.5 for use in Director MX 2004.

You can also update movies to Director MX by simply opening and saving them, but the Update Movies command is faster for converting large projects. It's also more effective for preserving links to external media. For more information, see "Processing movies with Update Movies" on page 464.

Note: Macromedia Shockwave Player 8.5 can play Shockwave content created with Director 5, 6, 7, 8, and Director MX.

Downloaded from www.Manualslib.com manuals search engine

To be included as a file in a Projector, a movie must be created with Director 8.5 or newer. However, using Lingo, movies created with earlier versions of Director can be played (e.g. go to movie, play movie) back to and including movies created with Director 5.

When you open a Director 7 movie in Director MX or convert it to the new format with Update Movies, the following changes occur to the movie:

- The data structure is changed to the latest file format.
- Shapes are not converted to the new Bézier shapes.
- Ink functionality is not updated unless you turn off Maintain Outdated Ink Mode Limitations in the Movie Properties dialog box.
- Old Score data, from versions of Director before Director 5, is converted to the new Score, combining adjacent frames in the old Score containing the same cast members into single sprites in the new Score. You might want to split or join sprites to make working in the Score more convenient.

About projectors

A projector is a stand-alone version of a movie. Projectors require certain Xtra extensions to use text, Flash content, connect to the Internet, and use certain other features. Director includes the most common required Xtra extensions by default. You can include or exclude Xtra extensions for each movie using the Include in Projector option in the Movie Xtras dialog box. You can also add Xtra extensions to a projector manually the same way you select movie files. You can create Macintosh projectors using a Windows application and create a Windows projector using a Macintosh application. For more information, see "About cross-platform projectors" on page 463.

In addition to the standard projector, you can create a fast-start projector, which typically starts faster. A fast-start projector doesn't include Xtra extensions inside the projector itself, so there's nothing to unpack.

Creating projectors

You can include only Director MX 2004 movies in projectors. You can use the Update Movies command to convert older movies to the latest version of Director. For more information, see "Converting movies created in previous versions of Director" on page 459.

To create a projector:

1 Select File > Publish Settings.

The Publish Settings dialog appears.

2 From the Formats tab, select Projector in the Publish section of the dialog box.

3 From the Player type pop-up menu on the Projector tab, select Standard to create a standard Projector or select Shockwave to create a Shockwave projector.

Standard includes the uncompressed player code in the projector file. This option starts the movie faster than other options but creates the largest projector file.

Shockwave makes the projector use the Shockwave Player installed in a user's system instead of including the player code in the projector file. If the Shockwave Player is not available when the movie runs, the movie prompts the user to download it.

Downloaded from www.Manualslib.com manuals search engine

4 Select from the following options:

Animate in Background lets the movie continue playing if a user switches to another application. This is useful if you want the movie to continue running in the background when its window is not active. If this option is not selected, the movie pauses when the user switches to another application and resumes when the user switches back.

Full screen sets the movie to fill the entire screen when playing.

Center stage in monitor insures that the movie stays in the center of the screen when playing.

Escape key terminates lets the user press the ESC key to exit the movie.

Reset Monitor to Match Movie's Color Depth automatically changes the color depth of your monitor to the color depth of each movie in the projector play list. For example, if you are working on a color monitor set to 256 colors and a movie in the play list was created in thousands of colors, the monitor automatically switches to thousands of colors.

Extra main memory

System temporary memory (Macintosh only) makes Director use available system memory when its own partition is full.

5 Once all Projector options are set, click OK or click Default to return the settings to their defaults.

Note: To avoid problems with linked media, create the new projector in its final folder location and do not move it to a different folder. Director turns the movies, casts, and included Xtra extensions into a single projector.

You can also set options for the files that are associated with the Projector movie. By default, a projector consists of just the current movie. However, you can add external cast files and exclude Xtra extensions, among other options.

To set Files tab options:

1 Select File > Publish Settings.

2 On the Formats tab, click the Projector option.

3 Click the Files tab.

4 To set the Primary movie components, select from the following options:

Include linked cast files includes any linked cast files in the projector.

Exclude all xtras lets you exclude some or all Xtras from your projector. For more information, see "Excluding Xtra extensions from projectors" on page 462.

Compress files (Shockwave format)

Add Files opens a dialog box that lets you select additional Director Movie or Cast files to the current projector.

Remove All deletes all added files.

Play every movie in list ensures all movies in your movie list play.

Loop sets the movie to keep playing.

5 Click OK to save changes and return to the movie.

6 Click Cancel to reject changes and return to the movie.

7 Click Publish to publish the movie with these settings.

Downloaded from www.Manualslib.com manuals search engine

To create a fast-start projector:

1 Create a new folder on your computer desktop.

It does not matter what you name the folder.

2 In Director, select Modify > Movie > Xtras.

The Movies Xtras dialog box appears.

3 Select the name of each Xtra and deselect Include in Projector for each, then click OK.

4 Select File > Save and Compact.

If you are adding multiple movies to the package, repeat steps 2 through 4 for each of the movies.

5 Select File > Create Projector.

6 In the Create Projector dialog box, select the movies to include in the projector and click Add.

7 Click Options and select one of the following:

Shockwave (Windows) and click OK.

Standard (Macintosh) and click OK.

8 In the Create Projector dialog box, click Create.

9 In the dialog box that appears, type a name for the projector. If necessary, use the pop-up menu to browse to the desktop folder you created in step 1, and then click Save.

10 Exit Director and return to your computer desktop.

11 Open the folder you created in step 1. Create a subfolder within this folder and name it Xtras.

12 In your Director application folder, copy the Xtra extensions required to play your movie into the Xtras folder you just created.

You must also include external movies, external casts, and linked media with your projector. If the external files are in the folder that contains the projector, the projector can automatically link to the files.

13 (Windows only) In your Director application folder, copy the files dirapi.dll, iml32.dll, proj.dll, and msvcrt.dll into the Xtras folder.

14 Start your projector to see it open quickly and play your movies.

Excluding Xtra extensions from projectors

Sometimes it is useful to place any Xtra extensions needed by a projector in an external Xtra extensions directory. Some reasons for doing this include:

- Xtra extensions stored in a projector must be uncompressed to a separate file each time the projector starts up, which makes startup slower. If the Xtra extensions are stored in an external directory, the projector starts up more quickly.
- Developers are able to easily track exactly what Xtra extensions versions are provided for the projector.
- Developers can easily update an Xtra extension without having to create a new projector.
- Multiple projectors in the same folder can use the same external Xtra extensions folder.
- Sometimes you need to include other external files with a projector, for example an animated GIF. These files can just be put in the external Xtra extensions folder instead of having a separate location.

Downloaded from www.Manualslib.com manuals search engine

- Shockwave projectors can use the Xtra extensions that have been already installed with Shockwave instead of requiring their own copy.
- Because of the way the Xtra extensions list for the movie is maintained, it is possible to get Xtra extensions included in a projector that you don't need. For example, let's say that you can use an animated GIF member in a movie. In this case, the AnimGif Xtra is added to the Xtra list and marked for downloading. Later you delete the animated GIF member. The AnimGif Xtra is not removed from the Xtra list even though the movie is no longer using it. The reason the Xtra is not removed from the list is because the movie has no way of knowing whether an animated GIF member is loaded using Lingo or JavaScript syntax (either directly, or by branching to another movie that uses it). So developers have to remember to check the movie's Xtra extensions list to find out exactly what is included in a projector. Having the Xtra extensions always external eliminates this issue.

About cross-platform projectors

Cross-platform projectors are projectors created on one platform or operating system which can run on another: in Director's case, you can create projectors that will play on the Macintosh OS X system, but are created or authored on a Windows system and vice versa.

Note: You cannot create Macintosh Classic projectors on a Windows system.

To create a cross-platform projector:

1 Select File > Publish Settings.

2 From the Formats tab, select Macintosh Projector if you are working on a Windows system, or select Windows Projector if you are working on a Macintosh.

3 Make any other selections you want from the Publish Settings dialog box.

Note: You cannot create custom application icons for a cross platform projector

4 Click Publish.

Custom Xtra extensions and cross-platform projectors

To support custom Xtra extensions in cross-platform projectors, consider the following issues:

- Ensure that there is an entry for your Xtra extension in the xtrainfo.txt file that gives the proper filename for the Xtra extension on each platform. If this information is not provided, the code assumes your Xtra extension only exists for the current platform.
- To use your custom Windows Xtra extension in a Macintosh-created projector, put the Xtra extension in the /Cross Platform Resources/Windows/Xtras/ folder of the Macintosh Director installation.
- Macintosh Xtras must have both a data fork and a resource fork. These two fork files aren't supported on Windows. For each Macintosh Xtra extension, two files are created: one for the data fork and one for the resource fork. The names for these files must be the same as the original Xtra extension name with a suffix: ".data" for the data fork and ".rsrc" for the resource fork. Both of these files are placed in the /Cross Platform Resources/Macintosh/Xtras/ folder of the Windows Director installation.

  For example, the Macintosh Xtra called "MyXtra" would have two files on Windows, "MyXtra.data" and "MyXtra.rsrc."

Downloaded from www.Manualslib.com manuals search engine

Processing movies with Update Movies

You can use the Update Movies command on the Xtras menu to do the following:

- Update movies and casts from older version of Director to the latest file format.
- Compress movies for faster downloading from the Internet.
- Remove redundant and fragmented data in movie and cast files. The Save and Compact and Save As commands do this as well.
- Prevent users from opening movie and cast files.
- Batch-process movie and cast files in large projects.

When beginning a project, use Update Movies to convert Director 5 or later version files to the latest file format.

At the end of a project, use Update Movies to compress all your movies and casts at once.

To update and compress movies and casts:

1 Select Xtras > Update Movies.

The Update Movies dialog box appears.

2 Select one of the Action options:

Update converts movies from Director 5 or later versions to the latest file format. As it updates movies, Director consolidates and removes fragmented data, just as when you use Save As. (To update movies from older versions, you must first convert them to the Director 5 file format.)

Protect removes all the data required to edit the movie, but it does not compress the movies further. It adds the DXR extension to movies, and CXT to casts. Protect also flags the movie so it can't be opened in the authoring environment.

Convert to Shockwave Movie(s) rewrites movies and casts in the compressed Shockwave file format and adds the DCR extension to movies and CCT to casts. This options also prevents users from opening the movie or cast and making changes. Once a movie is compressed, there is no way to it to recover an editable file, so be sure to keep the original movie.

3 Select one of the Original Files options:

Back Up into Folder specifies that the original files go in a selected folder. Click Browse to select the folder for the original files. To avoid overwriting old backups, you should select a new folder each time you run Update Movies.

Delete specifies that the newly updated files overwrite the original files. Be very careful when using this option. Once a file is protected or compressed, you cannot open it again in Director.

4 Click OK.

A dialog box appears from which you select the files to change.

5 Select the movies and casts you want to change and click one of the following:

Add to add the selected files.

Add All to add all the movies in the current folder. The items you select appear in the file list at the bottom of the dialog box. You can update movies in different folders at the same time.

Add All Includes Folders before you click the Add All button to include any movies or casts inside folders appearing in the upper list. This option is useful for updating large projects with several levels of folders.

Downloaded from www.Manualslib.com manuals search engine

6 Click Update.

Director saves new versions of the selected movies with the same names and locations as the original movies. This ensures that all links and references to other files continue to work properly. Director copies the original movies to the folder you specified, re-creating their original folder structure. If you didn't specify a folder for the original movies, Director prompts you to select one.

Director adds the DCR extension to Shockwave content and the CCT extension to external casts in the Shockwave format. Protected movies have the DXR extension, and protected casts have the CXT extension.

Exporting digital video and frame-by-frame bitmaps

You can export all or part of a movie as a digital video. You can use this digital video in other applications or import it back into Director. Any interactivity in the movie is lost when it is exported as a digital video. You can also export a movie or a part of a movie as a series of bitmaps: BMP in Windows, and PICT on the Macintosh.

You can export QuickTime digital video from either the Windows or the Macintosh version of Director. QuickTime must be installed on the system to export as QuickTime (version 4 or later is required for Windows; version 3 or later is required for Macintosh). You can export the AVI (Audio-Video Interleaved) format only using the Windows version of Director. When you export to AVI, all sounds are lost.

When Director exports animation as a video or bitmaps, it takes snapshots of the Stage moment by moment and turns each snapshot into a frame in the video or a bitmap file. Sprites animated solely by Lingo or JavaScript syntax are not exported.

When Director exports video or bitmaps, it always uses the entire Stage.

To export to digital video or bitmaps:

1 Select File > Export.

The Export dialog box appears.

2 Select the range of frames you want from the Export options at the top of the dialog box:

Current Frame exports the current frame on the Stage. This is the default.

Selected Frames exports the selected frames in the Score.

All Frames exports all frames.

Frame Range exports only the range of frames that begin and end with the frame numbers you enter in the Begin and End boxes.

3 If you select Selected Frames, All Frames, or Frame Range as the Export option, select one of the following options. These options do not work with digital video.

Every Frame exports all frames in the selected range.

One in Every _ Frames exports only the frames at the interval you specify in the box.

Frames with Markers exports frames with markers set in the Score window.

Frames with Artwork Changes in Channel exports frames only when a cast member changes in the channel you specify in the box.

Downloaded from www.Manualslib.com manuals search engine

4 From the Format pop-up menu at the bottom of the dialog box, select a format.

- Windows: Video for Windows (.AVI), DIB File Sequence (.BMP), or QuickTime Movie (.MOV)
- Macintosh: PICT or QuickTime Movie

BMP is the standard format for a Windows bitmap series. PICT is the standard format for a Macintosh bitmap file format.

5 If you are exporting video, click the Options button.

The Video for Windows or QuickTime Options dialog box appears.

6 Select the options you want to use and then click OK.

For AVI movies, enter a number of frames per second for Frame Rate.

For information about the QuickTime options, see "Setting QuickTime export options" on page 466.

The Export dialog box reappears when you click OK.

7 Click Export.

A dialog box appears, prompting you to save the movie.

8 Name the file and then click Save.

When you click Export, a dialog box appears allowing you to name the file. If you are saving in video format, only one file will be created. If you are saving in BMP or PICT format, Director automatically creates one file for each frame, attaching the corresponding frame number to each file. For example, if the name of the exported file is Myfile, frame 1 will be exported to a file named Myfile0001.

Setting QuickTime export options

You use the QuickTime Options dialog box to specify options for exporting a movie as a QuickTime digital video. This dialog box appears when you click the Options button in the Export dialog box and QuickTime is the specified format.

To set QuickTime export options:

1 Select File > Export.

2 Select QuickTime Movie from the Format pop-up menu.

3 Click Options.

4 To set the speed the video will play, select a Frame Rate option:

Tempo Settings exports the settings in the tempo channel to the QuickTime movie. This setting lets you create a QuickTime movie at any tempo, even if Director is not capable of playing the movie at that tempo in real time.

The size of an exported QuickTime movie is influenced by the tempo settings, transitions, and palette transitions in the Director movie. Fast tempos, certain transitions, and palette transitions all increase the size of the QuickTime movie. The tempo settings determine the number of QuickTime frames per second and the number of frames per transition. The faster the tempo, the more frames per second.

Downloaded from www.Manualslib.com manuals search engine

A movie that would work well with Tempo Settings as the Frame Rate option is one in which the tempos have been carefully timed. For instance, some frames could be set to a tempo of 10 frames per second, and their QuickTime frame durations would be exactly one-tenth of a second. Other frames later in the movie could be set to a tempo of 1 frame per second; when the movie is exported, these slower frames would each last precisely 1 second in the QuickTime movie.

Real Time lets you export a QuickTime movie that matches the performance of the Director movie as it plays on your system. (You should always play the entire movie with script disabled before using this feature.)

When you export a movie with Real Time selected, each Director frame becomes a QuickTime frame. Each frame in the QuickTime movie will match the duration of the same frame in the Director movie.

Director will generate as many frames as required to duplicate each transition, up to 30 frames per second. To increase the number of frames created for any transition, reduce the smoothness of the transition.

This option causes Director to use the actual durations that were stored the last time you played the entire movie, regardless of the actual tempo settings of the movie.

5 To reduce the file size of a QuickTime movie at the expense of quality, select an option from the Compressor pop-up menu. Different options appear on the Compressor pop-up menu depending on the video hardware and software available in your system. Consult your QuickTime documentation.

Animation compression is for simple animations.

Cinepak compresses 16-bit and 24-bit video for playback from CD-ROMs.

Component Video is usually used when capturing from a live video feed.

Graphics compression is for exporting single frames of computer graphics.

None exports with no compression.

Photo-JPEG compression is good for scanned or digitized continuous-tone still images.

Video compression is for exporting video clips.

6 To determine the compression quality and resulting file size when using the chosen compressor, use the Quality slider. A higher-quality setting preserves the appearance of the images and motion but increases the size of the file. A lower-quality setting results in poorer image quality but decreases the size of the file.

7 To determine the color depth (the number of colors) of your artwork, select a setting from the Color Depth pop-up menu. The compression method you select determines the color depth options available to you in this pop-up menu.

8 To determine the method by which the exported QuickTime movie is resized, select values for Scale. You can select a percentage from the Scale pop-up menu, or you can type pixel dimensions in the fields. By entering the number of pixels, you can stretch a movie so that it plays in a rectangle that does not adhere to the original aspect ratio.

9 To choose which soundtracks are exported with your movie, select Channel 1 or Channel 2. A checked box indicates that the associated sound channel in the score is exported with your QuickTime file.

External sounds (sounds you imported as linked cast members) are not exported when you export a digital video. To include sound when you export a digital video movie, you must import the sounds as cast members instead of linking to them.

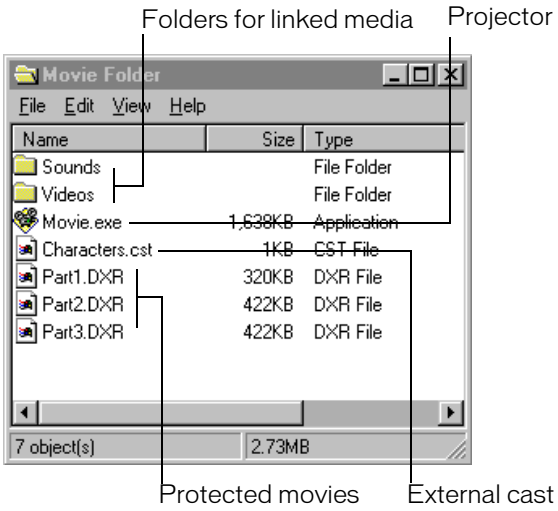
Downloaded from www.Manualslib.com manuals search engine

Looped sounds don't loop in a movie that you have exported as a digital video. To loop a sound in a movie that you plan to export as a digital video, you must trigger the sound by alternating it between the two sound channels.

About organizing movie files

In most cases, you should divide a large production into a series of smaller movies. You can combine as many movies as you want in a projector, but larger files take longer to save and are cumbersome to work with. Also, movies are easier to edit if they are organized in discrete sections.

The best way to organize a large production is to create a small projector file that starts the movie and then branches to Shockwave content or protected content. This saves you the trouble of re-creating the projector every time you change one part of a movie.



A typical file organization for a distributed movie

This approach also makes sense for movies on the Internet, but for different reasons. If the first movie is small, users don't have to wait as long for something to happen. Branching to a series of smaller movies also enables users to avoid downloading time for parts of the movie they do not use.

The size of your movie may be less of an issue if you use streaming Shockwave content. For more information, see "Setting movie playback options" on page 471.

Downloaded from www.Manualslib.com manuals search engine

CHAPTER 24
Using Shockwave Player

Macromedia Director MX 2004 movies can use the Internet in various ways: hosting multiuser sessions such as chats and games, streaming movies and sounds, retrieving data from the network, and interacting with a browser. Whether it is distributed on disk or downloaded from the Internet, a movie can use an active network connection to retrieve linked files, send information, open web pages, and perform many other network activities.

To make a movie appear in a user's browser, you can save it as Macromedia Shockwave content and embed it in an HTML document. The movie can play from a local disk or an Internet server. When the user opens the HTML document stored on an Internet server, the movie often begins streaming to the user's system, and begins playing when either a specified number of frames has been downloaded, or waits until all of the movie is downloaded to the local disk. (The movie begins streaming if you have set the Shockwave streaming playback option to ON. If the default streaming option is left at OFF, the movie begins playing after the entire movie has been downloaded to the local drive. For more information, see "Setting movie playback options" on page 471.)

You can also distribute a movie over the Internet as a projector—a packaged movie that the user downloads and executes. A projector plays as a stand-alone application, not in a browser. For more information, see "About distribution formats" on page 451.

When you author a movie, consider how the movie is to be distributed and played on users' systems. If the movie streams from an Internet source, you might need to modify the movie for the best streaming performance and to use the behaviors that are built in to Director to make the movie wait while certain cast members download. Controls and script methods offer ways to send and retrieve media and other information, interacting with a browser, and monitoring downloading.

About streaming movies

When you distribute a movie on the Internet, streaming provides an immediate and satisfying experience for your users. If you don't enable streaming, your user must wait for the entire movie to download before it begins to play. A streaming movie begins playing as soon as a specified amount of content reaches the user's system. As the movie plays, the remaining content downloads in the background and appears when it is needed. Streaming can dramatically decrease the perceived downloading time.

Downloaded from www.Manualslib.com manuals search engine

When Director streams a movie over the Internet, it first downloads the Score data and other nonmedia information such as scripts and the size of each cast member's bounding rectangle. This data is usually quite small compared with the size of the movie's media—usually only a few kilobytes. Before starting the movie, Director then downloads the internal and linked cast members that are required for the first frame of the movie (or more frames if you have increased the number in the Movie Playback dialog box). After the movie starts, Director continues to download cast members (along with any associated linked media) in the background, in the order the cast members appear in the Score.

If the movie jumps ahead in the Score or uses cast members that are referenced only by scripts, the required cast member might not be available when necessary. If cast members are not available, the movie either ignores them or displays a placeholder, depending on how you set the streaming options in the Movie Playback Properties dialog box. In addition, if cast members are not fully downloaded and present in memory, you might get script errors, especially if you are working with 3D.

A challenge of authoring for Internet streaming is ensuring that all cast members have been downloaded by the time the movie needs them. To avoid missing cast members, make sure that all the cast members required for a particular scene have been downloaded before beginning the scene. You can use the Director behaviors to wait for media in certain frames or for particular cast members. For more information, see "About streaming with the Score and behaviors" on page 473. You can also write custom scripts to do this. See "Checking whether media elements are loaded with Lingo or JavaScript syntax" on page 474.

Director movies stream automatically if the Streaming playback option is set to ON (its default setting is OFF). In addition to turning streaming off and on, you can specify that the media elements for a certain number of frames must finish downloading before the movie starts playing.

You control streaming movies by arranging sprites in the Score and controlling the movement of the playhead either with the Director behaviors or with Lingo or JavaScript syntax. You can also use these scripting methods to specify when externally linked files are downloaded.

About network operations

Director lets a network operation begin even if a previous network operation is not complete. This capability, often referred to as background loading, lets Director perform multiple operations while loading files. Because something else is happening while files are loading, the user does not perceive the wait.

Note: Loading data from a network is different from loading cast members in Director. Loading from a network loads data to the local disk. Loading cast members in Director means loading cast members into memory.

It is a good idea to author Shockwave content so that it performs other tasks while data is loading in the background. Because Internet operations require background loading, Lingo or JavaScript intended to execute on the Internet behaves differently than script methods that run within one movie. For more information, see "Using Lingo or JavaScript syntax in different Internet environments" on page 477.

Downloaded from www.Manualslib.com manuals search engine

Setting movie playback options

To change basic streaming settings for a movie, you use the Movie Playback Properties dialog box. You can turn streaming on and off, specify a number of frames to download before playing the movie, and make Director display placeholders if cast members are not downloaded yet. The Movie Playback Properties dialog box also includes options for locking the current tempo.

Leaving streaming off (the default setting) makes sense for some types of movies. For example, a game that requires all cast members to be available at once might not be suitable for streaming. Other movies work best if the media for a certain number of frames downloads before the movie begins playing. This option is especially useful for streaming movies that were not originally designed for streaming.

Placeholders are rectangles that appear in place of media elements for cast members that have not yet been downloaded. Placeholders are useful when testing to indicate places where media is missing.

You can specify streaming options any time before saving a movie as Shockwave content.

Note: If you want to test how a movie streams from a server before you save the movie as a Shockwave movie, use File › Save and Compact to make sure the data in the movie is properly ordered and that redundant data is removed.

To set movie playback options:

1 Select Modify > Movie > Playback to define streaming options.

2 To let the movie stream automatically, select Play While Downloading Movie.

3 To make the movie wait for all media elements (internal and linked) for a specified range of frames, enter the number of frames in Download __ Frames Before Playing.

By default, movies download the first frame only. Adjust this setting to the number of frames that is best for your movie.

4 To make the movie display placeholders for media elements that have not been downloaded, select Show Placeholders.

The placeholders appear as rectangles when the movie plays.

5 To lock the movie to its current tempo settings, select Lock Frame Durations. For more information, see "Locking frame durations" on page 159.

To set Shockwave playback options, see the next section.

Setting Shockwave playback options

To view Shockwave content, your users must have the Macromedia Shockwave Player, which comes preinstalled on many computer systems. The player is also available for free downloading from the Macromedia website at www.macromedia.com/shockwave/download/.

The Shockwave Player includes a volume control and a standard context menu that appears when a user right-clicks (Windows) or Control-clicks (Macintosh) a movie. You can select specific playback options to include for your users when you save your movie as Shockwave content.

Shockwave content loops by default. To cause Shockwave content to play only once, add the `Hold on Current Frame` behavior to the last frame of the movie.

Downloaded from www.Manualslib.com manuals search engine

To set Shockwave playback options:

1 Select File > Publish Settings.

The Publish Settings dialog box appears.

2 If not already selected, select Shockwave File (DCR) on the Formats tab.

The Shockwave tab appears in the Publish Settings dialog box.

3 Select the Shockwave tab and select the options you want to provide for your users, as described in the following list:

- Image Compression optimizes your content for download over the Internet by letting you manage the file size vs. the image quality a file. The default setting for all bitmap member types when creating a dcr. The default is set to JPEG, with a quality setting of 80. To further optimize the size of your images, you can adjust the Image Compression setting of each individual bitmap.

  Standard: Select Standard to apply compression techniques used by Director in versions 4 through 7, select Standard. This setting is suitable for simple graphics that contain few colors or non-photographic type images.

  JPEG: Select JPEG and specify the image quality setting by moving the slider to a value between 0 and 100 percent. A lower percentage results in increased compression, but produces a lower quality image.

- Audio Compression:

  Compression Enabled lets you compress the sound in your movie. Select this option and select the level of compression from the kBits/second pop-up menu. For more information about sound compression, see "Compressing internal sounds with Shockwave Audio" on page 237.

  Convert Stereo to Mono lets you convert stereo audio to monaural. This option is only available if Shockwave Audio Compression is enabled.

- **Include Cast Member Comments** lets you include comments you might have entered in the Comments text box of the Property inspector for your cast members. You can then use Lingo or JavaScript syntax to access the comments in the DCR file.

- Enabled context menu items:

  Volume Control lets users adjust the volume of the movie's soundtrack.

  Transport Control provides controls for rewinding, stopping, and stepping through the movie.

About creating multiuser applications

If you want to create multiuser movies or applications with Director, you need to use Macromedia Flash Communication Server MX in your Director movie. This can be a Flash movie that has already been authored to communicate with the Flash Communication server or a simple Flash movie that can serve as a container for Flash script objects that you create with Lingo or JavaScript syntax. You can create applications that use the Flash Communications Server entirely in Director by using Lingo or JavaScript to create Flash script objects.

For more information about using Flash Communication Server MX in Director, see Chapter 9, "Using Flash, Flash Components, and Other Interactive Media Types," on page 181.

Downloaded from www.Manualslib.com manuals search engine

About streaming with the Score and behaviors

The easiest way to create a movie that streams well is to arrange the Score properly and use behaviors to control the playhead. Director downloads cast members in the order in which they appear in the Score. Try to arrange the Score so that events don't make the playhead jump far ahead in the Score, where cast members have not yet been downloaded. For example, if you place a menu in the first frame of a movie and a user selects an option that sends the playhead to frame 400, the cast members for frame 400 may not be available right away.

To avoid this problem, begin a movie with a simple introductory scene that contains a few small cast members, preferably vector shapes. You can use a streaming behavior from the Library palette to make the introduction loop until the cast members that are required for the next scene have been downloaded in the background.

Several behaviors that are included with Director control the playhead or a progress bar while media elements are downloading. These behaviors make it easy to allow enough time for downloading to catch up with action in the Score.

Looping behaviors

Looping behaviors make the playhead return (loop) to a frame or stay on the current frame until specified media elements have been downloaded and then continue to the next frame. Attach a looping behavior to a frame in the script channel, not to a sprite. The following looping behaviors are accessible by selecting Internet > Streaming from the Library List menu in the Library palette:

Loop Until Next Frame is Available loops the playhead to a specified frame until all the media elements that are required for the next frame have been downloaded.

Loop Until Member is Available loops the playhead to a specified frame until a certain cast member has been downloaded.

Loop Until Media in Marker is Available loops the playhead to a specified frame until all the media elements for the frame at the specified marker have been downloaded.

Loop Until Media in Frame is Available loops the playhead to a specified frame until all the media elements that are required for a certain frame have been downloaded.

Jumping behaviors

Jumping behaviors make the playhead skip to a specified frame or marker after certain media elements have been downloaded. Attach a jumping behavior to a frame in the script channel, not to a sprite.

Jump When Member is Available moves the playhead to the specified frame after a certain cast member has been downloaded.

Jump When Media in Frame is Available moves the playhead to the specified frame after the media elements for a particular frame have been downloaded.

Jump When Media in Marker is Available moves the playhead to the specified frame after the media elements for the frame at a particular marker have been downloaded.

Downloaded from www.Manualslib.com manuals search engine

Checking whether media elements are loaded with Lingo or JavaScript syntax

Director has several options that let an initial portion of a movie start playing as soon as the required data and cast members are available. You can use Lingo or JavaScript syntax to check whether media elements have been downloaded from a network by testing the following:

- Whether a specific cast member is loaded before the movie proceeds
- Whether the cast members used in a specific frame are loaded before the frame plays

Checking whether a cast member or sprite is loaded

To determine whether a specified cast member is available locally, you use the `mediaReady()` cast member or sprite property. You can check for a specific cast member or the cast member that is assigned to a specific sprite. When `mediaReady()` returns `TRUE`, the cast member is present on the local drive and ready for loading into memory. For more information, see the Scripting Reference topics in the Director Help Panel.

This property always returns `TRUE` for local files. It is useful only for movies that stream from a remote server. Because playback can begin before the entire movie has been downloaded, you must make sure that necessary media elements have been downloaded as the movie plays.

Checking whether a frame's contents are loaded

Use the `frameReady()` method to determine whether all the media elements that the specified frame requires are available locally. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

Downloading files from the Internet with Lingo or JavaScript syntax

Lingo or JavaScript syntax uses the Internet's resources by obtaining files from the Internet. The data is copied to the local disk or cache. After data is available on the local computer, use these scripts to retrieve the data for the movie. For more information, see "Retrieving network operation results with Lingo or JavaScript syntax" on page 476.

For a movie or projector that plays outside a browser, background loading is not required. However, preloading is a good idea because it improves playback performance.

All network Lingo or JavaScript syntax operations that obtain data from the network begin downloading the data and return a network ID. The data is not immediately available.

An unlimited number of network Lingo or JavaScript syntax operations can take place at once. However, depending on the browser chosen and the preferences set, there may be limitations set on the number of network operations taking place. When multiple network Lingo or JavaScript syntax operations run simultaneously, rely on the network ID that the method returns to distinguish which operation is complete. Be aware that running more than four operations at once usually adversely impacts performance.

When using network Lingo or JavaScript syntax, the current handler must finish before an operation's result can return. For best results, place script that initiates a network operation and script that uses the operation's result in different handlers. An `on exitFrame` handler is a good location for checking whether an operation is complete.

Note: You should typically use frame events rather than repeat loops for this process.

Downloaded from www.Manualslib.com manuals search engine

To execute a network script operation:

1 Start the operation.

For example, the following statement initiates a text downloading operation and assigns the network ID returned by the `getNetText()` operation to the variable `theNetID`:

```
theNetID = getNetText("http://www.thenews.com")
```

2 Make sure that the operation finishes.

To check an operation's status regularly until the method indicates that the operation is complete, use the `netDone()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

For example, the following statement loops in the current frame until the download operation is complete:

```
if not(netDone(theNetID)) then _movie.go(_movie.frame)
```

3 Check whether the operation was successful by using the `netError()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

4 Obtain the results if the operation is complete.

Note: You can also use getStreamStatus() to determine the status of a network operation. You can use `netDone()` to check whether an operation is "done" or "not done". By using getStreamStatus(), you can check on an operation's progress, as it displays how many total bytes there are and how many have been downloaded so far.

To cancel a network operation in progress:

- Use the `netAbort()` method to cancel a network operation without waiting for a result. This frees up capacity for Internet access, which lets other network operations finish faster. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve a file as text:

1 Use the `getNetText()` method or the `postNetText` method to start retrieving text. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

2 Use the `netTextResult()` method to return the text you retrieved with `getNetText` or `postNetText`. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve and play a new Shockwave movie from the network:

- Use the `gotoNetMovie()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The current movie continues to run until the new movie is ready to play. After the new movie is ready, the player quits the current movie and plays the new movie in the same display area as the calling movie.

To open a URL in the user's browser:

- Use the `gotoNetPage()` method. This method works whether the URL refers to Shockwave content, HTML, or another MIME type. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

You can specify that this method replaces a page's content or opens a new page. If the browser is not open, the method launches the browser. If the `gotoNetPage` method replaces the page in which the movie is playing, the movie keeps playing until the browser replaces the page.

Downloaded from www.Manualslib.com manuals search engine

The `gotoNetPage` method is similar to the Director `open` method. It does not return a value.

To preload a file from the server into the browser's cache:

- Use the `preloadNetThing()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The `preloadNetThing()` method initiates downloading a linked movie asset into the cache, where it is available for later use. Director can later preload the asset into memory without a download delay.

The current movie continues playing while preloading occurs.

To test whether getNetText(), preloadNetThing, or gotoNetMovie operations are complete:

- Use the `netDone()` method.
- Use the `getStreamStatus()` method.

For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

To post information using HTTP post/get abilities to a server and retrieve a response:

- Use the `getNetText()` or `postNetText()` methods. You can use either method to post data to a server, but HTTP `"get"` commands can only post a limited amount of information compared with HTTP `"post"` operations. If you are posting a great deal of information, the `postNetText()` method is recommended.

For more information about these methods, see the Scripting Reference topics in the Director Help Panel.

Retrieving network operation results with Lingo or JavaScript syntax

Lingo or JavaScript syntax can retrieve network operation results, such as a text result, a unique identifier for a network operation, a file's MIME type, and the date an HTTP item was last modified.

To retrieve the text result of a network operation:

- Use the `netTextResult()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To retrieve the "date last modified" string from the HTTP header for a specific item:

- Use the `netLastModDate()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To obtain the MIME type of the HTTP item:

- Use the `netMIME()` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

The results of the `netTextResult()`, `netDone()`, `netError()`, `netMIME()`, and `netLastModDate()` methods are retained until normal memory clean-up is needed.

Downloaded from www.Manualslib.com manuals search engine

To determine the state of a network operation that retrieves data:

- Use the `Lingo on streamStatus` event handler or the `JavaScript streamStatus` function. For more information, see the Scripting Reference topics in the Director Help Panel.

Using Lingo or JavaScript syntax in different Internet environments

Some Lingo or JavaScript syntax features behave differently, depending on whether the movie is playing back in a browser, as a projector, or within the authoring environment.

Using Lingo or JavaScript syntax with Internet security restrictions

Because of security issues for movies that play back in browsers, the following Lingo or JavaScript syntax features are unsupported for Shockwave movies playing in a browser.

In general, the following Lingo or JavaScript syntax features are unsupported because of Internet security concerns:

- Setting `colorDepth()` for the user's monitor
- Saving a movie by using the `saveMovie()` method
- Printing by using the `printFrom()` method
- Opening an application by using the `open()` method
- Stopping an application or the user's computer by using the `quit()`, `restart()`, or `shutDown()` method
- Opening a local file that is not in the dswmedia folder or a subfolder of the dswmedia folder
- Pasting content from the Clipboard by using the `pasteClipBoardInto()` or the `copyToClipboard` method prompts a security dialog asking for permission to perform the operation.
- Searching for files on a user's system with `getNthFileNameInFolder()`, `searchCurrentFolder()`, or `searchPath()`.
- The permanent downloading of files from a URL to the user's local drive using `downloadNetThing()` is disabled in Shockwave due to security concerns. This function works in authoring and projectors.

Using URLs with Lingo or JavaScript syntax

In addition to the Lingo or JavaScript syntax that is explicitly intended for use with network operations, some script elements can use URLs as references to external files.

The following Lingo or JavaScript syntax elements can use URLs as file references in all circumstances:

- `moviePath`
- `pathName`
- `unloadMovie`

Note: You can also set the fileName of cast members to URLs and link to external media assets.

Downloaded from www.Manualslib.com manuals search engine

The following Lingo or JavaScript syntax supports URLs as references to external files. If you use this script in projectors or during authoring, you can avoid pauses while the file is being downloaded by first using `preloadNetThing()` to download the file. After the file has been downloaded, you can use these Lingo or JavaScript syntax elements with the file's URL without a delay.

When the following Lingo or JavaScript syntax is used in a browser, however, you must first download the file by using the `preloadNetThing` method. If you don't, the script fails.

- Using a `gotoMovie` statement
- Using an `importFileInto()` method
- Using a `preLoadMovie()` method
- Using a `playMovie()` method
- Using an `openWindow` method`()` (disabled in browsers)

The following Lingo or JavaScript syntax elements can use URLs to Shockwave Audio (SWA) sound files as file references:

- `streamName`
- `URL` cast member property

The following elements don't work in Shockwave content because Shockwave does not support movies in windows (MIAWs):

- `open window`
- `forget window`
- `close window`

Differences in scripting for browsers

The following list discusses some general differences in the way to script for a movie that plays over the Internet, depending on whether the movie is in a browser.

- For a movie playing in a browser, it's best to use `preloadNetThing` to load media elements, external casts, or other movies you might navigate to into the browser's cache first. If the media elements are not preloaded using `preloadNetThing`, linked media elements might not be present when they are needed.
- Avoid using long repeat loops in browsers; such repeat loops can make the computer appear unresponsive. As an alternative, you can split long operations into sections and execute them over a series of frames or check for user actions in an `on exitFrame` handler. Also remember that the status of any active network operation does not update while long repeat loops are running.
- Do not use a `repeat while` loop to check whether a network operation is complete.

Lingo or JavaScript syntax that is unsupported in browsers

The following Lingo or JavaScript syntax features are unsupported for movies that play back in browsers:

- Creating and managing a MIAW
- Installing and managing custom menus

Downloaded from www.Manualslib.com manuals search engine

Interacting with browsers

You can use `setPref()` or `getPref()` to read or write simple text *preference* files to the Prefs folder found in the Shockwave Player folder.

To write to a preferences file on a local disk:

- Use the `setPref` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

  After the method runs, a folder named Prefs is created inside the Shockwave Player folder (in the same location as the Xtras folder). The `setPref` method can write only to that folder. The default folder locations for Windows and Macintosh are described in the following list:

  Windows The \Macromed\Shockwave 8 subfolder of the system folder; the system folder is typically c:\winnt\system32 or c:\windows\system

  Macintosh The OS 9 path is The System Folder:Extensions:Macromedia:Shockwave 8 folder. The OS X path is /my Volume/Users/<me>/Library/Application Support/Macromedia/Shockwave/prefs.

  The `setPref` method cannot write to a file that is on a CD.

To return the content of a file that was written by a previous setPref method:

- Use the `getPref()` method. If no `setPref` method has already written such a file, the `getPref()` method returns `VOID` if called from Lingo or `null` if called from JavaScript syntax. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

To specify text in a browser's status area:

- Use the `netStatus` method. For more information about this method, see the Scripting Reference topics in the Director Help Panel.

  Note: Some browsers do not support this method.

Testing your movie

However you decide to create your movie, test it thoroughly before releasing it to the public. Make sure you test on systems with all common types of Internet connections, especially on slow modems and at busy times of day. The following list describes things you might want to check before distributing your movie over the Internet; remember, however, that each movie has its own special needs:

- Compare a streaming version of the movie to a nonstreaming version to see if the performance is different. Some smaller movies might work better without streaming playback.
- Verify that all linked media elements appear correctly. To see if the movie correctly handles an error, try forcing the linked media elements to fail.
- Run the movie on all systems your users are likely to have. For the general public, these include Windows 98, 2000 or XP. For Macintosh OS X users, these include Power Mac G3 running OS X 10.2.6 or 10.3 or later. For Macintosh Classic users, a Power Macintosh G3 running System 9.2. Browsers include Netscape 7.1 or later; Microsoft Internet Explorer 5.2 or later and a color monitor.
- Test your movie on a machine that was not used to develop the content: this is critical.

Downloaded from www.Manualslib.com manuals search engine

- Run the movie on slow and fast connections; problems can arise from fast as well as slow connections.
- Check for display problems on systems set to 8-, 16-, 24-, and 32-bit color. Also test as many types of monitors and display adapters as you can.
- Check for font mapping problems in your movie. If your movie uses nonstandard fonts, use embedded fonts. For more information, see "Embedding fonts in movies" on page 164.
- Check for sound problems, particularly if you stream sounds with SWA.

About downloading speed

Developers distributing multimedia over the Internet usually limit file size, primarily because most users connect at relatively slow speeds. At 28,800 bps, it takes 30 seconds to 1 minute to download a 60K file. Using streaming playback can help you avoid some of the delays caused by downloading large files.

Movies and streaming SWA sounds always compete for control of the network, which can cause a noticeable problem on slower connections.

The following chart shows theoretical throughput times for various types of network access technology. The rates calculated for this chart are based on the average throughput rate shown—actual download times vary, and are highly dependent on the network conditions that exist at the time of download.

Content	28.8 Kbps	56K modem	DSL	T1
Small graphics and animation, 30K	15 sec	7 sec	1 sec or less	1 sec or less
Small complete movie, 100K	45 sec	21 sec	3 sec	1 sec or less
500K movie	220 sec	107 sec	15 sec	4 sec
1 MB movie	--	213 sec	30 sec	8 sec

Downloaded from www.Manualslib.com manuals search engine

INDEX

Numerics

3D behaviors
  about 327
  action 330
  applying 332
  groups 333
  library 328
  triggers 329
  types 328
3D Extruder tab (Property inspector) 324
3D Model tab, Property inspector 308
3D text 324
3D vectors 397
3D window, Shockwave 306
3D world
  boxes 330
  collision properties and commands 394
  defined 313
  event handling 393
  objects 315
3D Xtra 305

A

accelerating sprites 85
accessibility
  captioning 433
  features 427, 428
  government requirements 427
  keyboard navigation 429
  scripting 434
  Speech Xtra extension 428
  text-to-speech 431
action behaviors
  defined 328
  independent 331
  local 330
  public 330
  viewing 329
ActionScript instructions 190
ActiveX 204
Add inks command 79
Airbrush tool, Paint window 103, 107
Align window 66, 67
alternate image options 458
ancestor scripts 288
Animate in Background projector option 461
animated GIFs 101
animation
  Auto Filter bitmap effects 124
  Cast to Time command 92
  complex model 314
  cursor colors 298
  defined 83
  film loops 94
  frame-by-frame 90
  modifiers 372
  motions 315, 372
  onion skinning 125
  Paste Relative command 97
  playing 330
  real-time recording 96
  Root Lock button 307
  scripting with Lingo or JavaScript 97
  Space to Time command 93
  step recording 96
  transitions 160
anti-aliasing
  3D cast members 310
  Flash movies 201
  setting with Lingo or JavaScript 178
  text 170
  toon modifier 331

Downloaded from www.Manualslib.com manuals search engine

appearance properties, MIAWs (movie in a window) 413
arranging sprites 58
ASCII values, keys 298
attaching behaviors 275
attributes, XML 424
audio. *See* sound
Auto Distort command 111
Auto Filter command 124
automatically adjusting line spacing 168
AVI files 465
axis
  Camera Rotate tool 306, 307
  colors 331
  model movement 351

B

background color 75
background operations 470
Background Transparent ink 79
backward compatibility 459
Beep action 281
behaviors
  accessibility 428
  actions 280
  attaching 275
  creating or modifying 279
  defined 275
  descriptions, creating 285
  example 285
  handlers 281
  inheritance 288, 289
  jumping 473
  library 304
  looping 473
  navigation controls 291
  order, changing 278
  parameters, changing 277
  parameters, customizing 282
  properties, setting 282
  sending messages to sprites 287
  streaming 473
  viewing descriptions 278
  writing with Lingo or JavaScript 281
behaviors, 3D
  about 327
  action 330
  applying 332
  groups 333
  library 328
  triggers 329
  types 328
bevel, 3D text 325
binary operations, vectors 398
bitmaps
  animated GIFs 101
  Auto Filter command 124
  cast member properties 128
  color depth 113
  color effects 111
  compression 129
  controlling with script 114
  creating 100
  cursor colors, animating 298
  defined 99
  distorting 111
  exporting frames as 465
  filters 123
  flipping 110
  fonts 164
  gradients 117
  image options 45
  importing 100
  inks 121
  moving selected areas 109
  onion skinning 125
  Paint window 102
  PICT files, importing 43
  registration points 112
  resizing 99, 113
  resolution 100
  rotating 110
  text, converting 173, 176
  texture 325
  tiling 120
  Transform command 113
  troubleshooting colors 155
  tweenable filters 124
Blend ink 79
blending sprites 76, 186
bold text 167
Bones player
  events 376
  methods 373
  modifiers 372
  motions 381
  properties 374
boxes
  creating 330
  properties 338

Downloaded from www.Manualslib.com manuals search engine

boxes, text 178
brightness, changing 154
browsers
  cache, preloading 476
  color palettes 155
  HTML file options 456
  Lingo and JavaScript support 478
  preferences 479
  previewing movies 448
  Shockwave support 447, 448
  status area 479
  tips for 478
  URLs, opening 475
Brush tool, Paint window 104, 107
Bucket tool, Paint window 103
Button component, Flash movies 207
buttons 292

C

cache, preloading files 476
callbacks, Flash objects 194, 196
cameras
  commands 322
  defined 314, 316, 335
  group properties 363
  manipulating 306, 307, 330
  methods 390, 396
  model interactions 316
  models, moving 351
  overview 382
  properties 386
  resetting 331
captioning 433
Cast List view 31
cast members
  3D 315
  3D worlds 313
  ActiveX 204
  anti-aliasing 310
  bitmap properties 128
  button properties 292
  copying 30
  creating 24
  cursors, animated 299
  defined 22
  digital video 248
  downloading, for streaming movies 470
  editing in Macromedia Fireworks 133
  editing text 166
  editors, launching 40, 46
  exchanging 89
  fields 172, 174
  film loop properties 95
  finding 40
  Flash 182, 184
  formatting text 169, 178
  frame-by-frame animation 90
  importing 42
  importing text 165
  libraries 48
  loading, checking status 474
  memory, removing from 47, 437, 439
  model resource commands 345
  moving 35
  naming 29, 31
  optimizing 133
  Paint window 102
  palette properties 155
  PICT properties 128
  properties, setting with Lingo or JavaScript 49
  properties, setting with Property inspector 38, 39
  properties, viewing 29
  renderer services object 403
  rendering commands 407
  rendering properties 406
  scripts, editing 29
  scripts, showing in Cast window 37
  selecting 30
  shapes 142, 143
  Show Data Tips 20
  sorting 35
  sounds, properties 232
  sprites, assigning 80
  sprites, reusing 71
  text, creating 165
  thumbnail view 33
  thumbnails, custom 34
  transition properties 162
  Xtra properties 49
Cast panels and panel groups
  Options Menu 27
  tabs 28
Cast Thumbnail view 33
Cast to Time command 92
Cast window
  controls 29
  list view 31
  moving members 35
  old movies, opening 28
  organizing 35

Downloaded from www.Manualslib.com manuals search engine

preferences 36
viewing 26
casts
creating 23
defined 22
external 23, 47
internal, defined 23
opening in new window 29
preferences 34
properties 38
centimeters, specifying as unit of measure 168
Change Cast Member action 281
Change Cursor action 281
Change Ink action 281
Change Location action 281
Change Palette action 281
Change Tempo action 281
Changes Only sprite label option 62
channels
behavior 277
behaviors 473
film loops 94
layering sprites 58
naming 12
selecting all sprites 54
sound 233, 236
Space to Time command 93
sprite, naming 57
tempo 157
transitions 160
character formatting 167
character sets, XML 425
check boxes 292
child models 346
child nodes, 3D cast members 316
circles, drawing 137
Classic Tool palette 17
clicks, detecting 294
closing MIAWs (movie in a window) 411
ColdFusion MX 13
collision modifier 370, 394
color
axis 331
brightness 154
cast member properties 155
changing palette 153
controls 145
cursor, animating 298
Darken and Lighten inks 78
effects, bitmaps 111
Favorites, editing 148
Flash movies 186
guides 66
hue 154
importing images 45
inks, Paint window 121
inks, sprite 77, 79
Lingo and JavaScript controls 154
mask effects 77
menu 147
modes 146
Paint window inks 104
palettes, changing during movies 149
RGB and index 146
saturation 154
selecting in Paint window 103
shader properties 352
sprites 75
text 325
troubleshooting 154
color depth
bitmaps 113
changing for movies 147
importing bitmaps 100
movie settings 147
Paint window tools 105
Reset Monitor option 461
troubleshooting 155
Color menu 147
Color Palettes window 149, 151
comments, cast members 22
components, Flash 207
components, reducing scripting time with 12
compression
bitmaps 129
JPEG 100
projector players 451
Shockwave 456
Shockwave Audio 237
Shockwave movies 472
context menu, Shockwave Player 200
Continuous at Endpoints tweening option 88
Convert to Bitmap command 173
coordinates, sprites 65
Copy ink 79
copying
cast members 30
frames 21
keyframes 87

Downloaded from www.Manualslib.com manuals search engine

Paste Relative command 97
sprites 69
corner points in vector shapes 137
creating
3D text 324
behaviors 279
bitmaps 100
cast members 24
casts 23
custom cast member thumbnails 34
custom tiles 120
editable text 172
external casts 47
fields 172
film loops 94
guides 66
hypertext links 171
libraries 48
MIAWs (movie in a window) 410
particle systems 330
projectors 460
sprites 52
text cast members 165
vector shapes 135
vectors 397
cropping videos 257
cross-platform features 12
cross-platform font mapping 173
cross-platform keyboards 298
cross-platform projectors 463
cue points, synchronizing 240
cursors
animated color cursors 298
behaviors 280
finding location 297
Flash movies 186
rollovers 331
trigger behaviors 329
Curvature tweening control 88
curves
creating 137
defined 135
joining and splitting 140
reshaping 139
smoothness 331
custom cast member thumbnails, creating 34
custom tiles, creating 120
customer support 15
customizing your workspace 12
cylinders 337

D

Darken button, Paint window 111
Darken ink 78
Darkest ink 79
data, obtaining from a network 474
DCR files
creating 452
distributing 451
format options 454
previewing in browsers 448
settings 452
decelerating sprites 85
Default Tool palette 17
deleting
Cast panels from groups 27
frames 21
guides 66
deploying movies. *See* distributing movies
detecting mouse clicks 294
digital video
AVI files, importing 43
cast member properties 248
controlling in the Score 246
controlling QuickTime with Lingo 250
controlling with Lingo 247
cropping 257
Direct to Stage 245
Direct to Stage option 308
exporting 465
exporting QuickTime 466
importing 244
preloading 439
synchronizing with cue points 240, 259
turning tracks on and off with Lingo 248
using on the Internet 258
Video window 245
Direct to Stage option 245, 308
Director MX 2004, features 304
disks, distributing movies 452
Display Template tab on Property inspector 18
Display Template, MIAWs (movie in a window) 415
dissolve transition 160
distorting images 111
distributing movies
accessibility 435
disks 452
formats 451
local networks 452
methods 447
project management tips 441

Downloaded from www.Manualslib.com manuals search engine

protected formats 451
Publish command 452
testing 479
URLs 451
Xtra extensions 449, 450
dithering 114
dockable MIAWs (movie in a window) 414
docking Tool palette 18
dolly camera 330
downloading
alternate image options 458
cast members 470
looping behaviors 473
speed 480
status, checking 474
streaming movies, options 471
using Lingo or JavaScript 474
Xtra extensions 449
dragging camera 330
drawing tools 17
drawRect property 412
Dropper tool, Paint window 103
dswmedia folder 244, 459, 477
duration of sprites 70
DVD-Video features 12
DXR files 451

E

Ease-In and Ease-Out tweening option 85, 88
editable text 172
editing
3D text 324
behaviors 279
bitmaps 100
cast member scripts 29
cast members in Macromedia Fireworks 133
Favorite color 148
fields 172
film loops 95
Flash cast members 184
frames 21
gradients, bitmap 117
image objects 115
palettes 151, 153
patterns 120
protected movies 451
sprite frames 89
text 166
vector shapes 139
editors, cast members 40, 46
effects
filters 123
Paint window buttons 109
Ellipse tool 137
ellipses, drawing 137
embedded fonts 164
end points
adding 139
creating 137
engraver shader 360
Enter Frame event 280
Eraser tool, Paint window 103
event handlers 394
events
ActiveX 206
behaviors 280
Flash components 209
MIAWs (movie in a window) 416
Every frame sprite label option 62
examples of movies
website 12
Exchange Cast Members command 89
Exit Frame event 280
exporting
movies as digital video 465
QuickTime options 466
Extended display option 20
extending sprites 70
extensions. *See* Xtra extensions
external casts
creating 47
defined 23
external sounds 231
Extreme tweening option 88
extruder model resource 344
Eyedropper tool, Paint window 103

F

Fade to Black/White 150
fading sprites 76
Favorite colors, editing 148
field text 163
fields
creating 172
properties 174
string checking 179
user click detection 296
files
compression 100
downloading with Lingo or JavaScript 474

Downloaded from www.Manualslib.com manuals search engine

downloading, speed 480
importing 42
linking to 44
preloading 476
project management 441
projectors, options 455
types supported 44
URL references 478
Fill color, Paint window 111
fill settings for vector shapes 138
film loops 94, 95
filters, bitmaps 123
finding
cast members 40
sprites 57
text 170
Fireworks
bitmap compression settings 129
editing cast members 133
importing files into Director 131
First frame sprite label option 62
Flash asset commonPlayer 182
Flash Communication Server MX 13, 198
Flash component Tool palette 17
Flash movies
ActionScript instructions 190
Button component 207
components 207
context menu 200
controlling with Lingo or JavaScript syntax 185
creating cast members 182
editing 184
events 209
importing 181
local connection object 195
performance tips 201
playback 188
playback settings 183
Settings panel 200
Stage, placing on 187
streaming 187
tips for using 184
variables, testing 189
XML data, sending to Director 193
Flash MX 2004 12
Flash objects 193
Flip button, Paint window 110
flipping
Flash movies with Lingo 186
sprites 75
fonts
3D text 324
character sets, XML 425
embedding in a movie 164
mapping across platforms 173
Foreground and Background color 75
formatting
fields 172
paragraphs 167
text 167
text inspector 169
frame-by-frame animation 83, 90
frames
adding 22
behaviors, attaching 277
duration, setting 70
editing in Score 21
events 280
exporting as bitmaps 465
jumping to 293
loading status, checking 474
Lock Frame Durations 159
looping 293, 473
moving sprites 69
moving sprites in 69
Paste Relative command 97
Preload options 38
printing 444
returning to 294
selecting within a sprite 54
sound controls 233
sprite preferences 52
tempo 157
transitions 160
functions, Flash movies 189

G

getURL 190
Ghost ink 79
GIFs, animated 101
Go to Frame action 280
Go to Marker action 281
Go to Movie action 280
Go to Net Page action 281
gotoNetMovie command 293
gotoNetPage command 293
government requirements, accessibility 427
gradients
bitmaps 117
vector shapes 141

Downloaded from www.Manualslib.com manuals search engine

grid
configuring 67
positioning sprites 66
properties, setting 18
grouping Cast panels 27
groups
commands 323
defined 314, 316
methods 364
parameters 333
properties 363
guides
color 66
positioning sprites 66
properties, setting 18
Guides tab on Property inspector 18

H

Hand tool, Paint window 103
handlers
anti-aliasing, turning on and off 311
behaviors 281
callback 196
events 394
Lingo or JavaScript events 393
MIAWs (movie in a window) 416
on getPropertyDescriptionList 284
scripts 393
handles 135
handwriting, simulating 97
help
new features 13
online help 13
where to get more information 14
HTML files
browser options 456
format options 454
previewing in browsers 448
settings 452
HTML, compared to XML 419
HTTP servers 476
hue, changing 154
hypertext links, creating 171

I

icons, Cast Thumbnail view 33
image objects
creating with script 115
editing 115
Image Options dialog box 45
image options, Shockwave 458
Import Pict File as PICT option 43
importing media
ActiveX 204
animated GIFs 101
bitmaps 100
cast members 42
color palette requirements 155
digital video 244
Director movies 202
DVD-Video 12
file types supported 44
Fireworks files 131
Flash movies 181
Flash MX 2004 12
image options 45
JPEG files 100
linking to files 44
managing projects 441
sound 231
text 165
inches, specifying as unit of measure 168
Include Original Data for Editing option 43
independent actions 331
independent behaviors 328
index color
mode settings 146
inheritance, behaviors 288
inker modifier 368
inks, Paint window 105, 121
inks, sprite
Darken and Lighten 78
Mask 77
settings 77
types of 79
Insert Blank Frames to Make Room option 21
integration with Macromedia Studio MX 2004 12
interactive features 291
internal casts, defined 23
internal sounds
compressing 237
importing 231
Internet
color palettes 155
digital video and 258
distributing movies 451
downloading files with Lingo or JavaScript 474
linked media 45
sound, streaming 232

Downloaded from www.Manualslib.com manuals search engine

speed 480
streaming movies 470
introduction to Director 11
Invert colors, Paint window 111
italic text 167

J

JavaScript syntax
3D text methods and properties 325
animating sprites 97
assigning cast members to sprites 80
behaviors, writing 281
bitmap controls 114
browser support 478
cast member properties, changing 49
cursor location detection 297
downloading files 474
editable sprites 295
Flash movies 185
handlers 393
importing files 44
jumping 293
keyboard detection 297
MIAWs (movie in a window) properties 413
mouse click detection 294
naming cast members 31
naming sprites 56
network operation results 476
positioning sprites 68
rollovers 296
scripting in 12
security limitations 477
sound controls 235
sprite properties 63
streaming Flash movies 187
streaming media controls 474
streaming sound 239
strings, checking 179
synchronizing sound 241
tempo controls 159
text, formatting 176
text, importing 166
text, user click detection 296
text-to-speech scripting 434
transitions 161
URL references 477
vector shape attributes 138
vector shape controls 141
joining
curves, vector shapes 140
sprites 70
JPEG files, importing 100
jumping 293, 473

K

kerning 167, 170, 176
Key Down event 280
Key Up event 280
keyboards
accessibility navigation 429
cross-platform equivalents 298
press detection 297
trigger behaviors 329
keyframe animation
events 380
methods 377
modifiers 372
motions 381
properties 378
keyframes
duplicating 87
properties 84
selecting 54
showing path 84

L

labels, sprite 62
LANs, distributing movies on 452
Lasso tool, Paint window 102, 106
launching cast member editors 40, 46
layering
MIAWs (movie in a window) 416
sprites 58
level of detail (LOD) modifiers 331, 366
libraries, creating new 48
Library palette, behaviors
3D 328
attaching behaviors 275
viewing descriptions 278
Lighten button, Paint window 111
Lighten ink 78
Lightest ink 79
lights
commands 321
defined 314, 315, 335
group properties 363
methods 384

Downloaded from www.Manualslib.com manuals search engine

model interactions 316
overview 382
properties 383
Property inspector 309
text 325
line settings for vector shapes 138
line spacing, adjusting automatically 168
Linear tweening option 88
lines, Paint window tools 105
Lingo syntax
3D text methods and properties 325
animating sprites 97
assigning cast members to sprites 80
behaviors, writing 281
bitmap controls 114
browser support 478
cast member properties, changing 49
cursor location detection 297
downloading files 474
editable sprites 295
Flash movies 185
handlers 393
importing files 44
jumping 293
keyboard detection 297
MIAWs (movie in a window) properties 413
mouse click detection 294
naming cast members 31
naming sprites 56
network operation results 476
positioning sprites 68
rollovers 296
security limitations 477
sound controls 235
sprite properties 63
streaming Flash movies 187
streaming media controls 474
streaming sound 239
strings, checking 179
synchronizing sound 241
tempo controls 159
text, formatting 176
text, importing 166
text, user click detection 296
text-to-speech scripts 434
transitions 161
URL references 477
vector shape attributes 138
vector shape controls 141
Link to External File option 43
linking
Director movies 202
external casts 47
file structure 441
media files 44
sounds 231
testing 459
links, hypertext 171
loading cast members 47, 437, 439
loading, checking status 474
local actions 330
local behaviors 328
local connection object 195
local folders 459
local networks, distributing movies on 452
Lock Frame Durations option 159
locking sprites 63
looping
behaviors 473
browser support 478
frames 293
sounds 234
loops, film 94, 95

M

Macintosh
browsers supported 447, 448
font mapping 173
keyboards 298
projectors 463
Macromedia Studio MX 2004, integration with 12
Magnifying Glass tool, Paint window) 103
Maintain Outdated Ink Mode Limitations option 459
managing projects 441, 468
manuals available with Director 14
mapping fonts 173
markers, playhead 330
Marquee tool, Paint window 102, 106
Mask ink 77
masks, for QuickTime 250
math, vector 397
Matte ink 79
media
importing 42
linking 44
project management 441
memory
MIAWs (movie in a window) 411
preloading
digital video 439

Downloaded from www.Manualslib.com manuals search engine

sounds, importing 231
textures 360
unloading cast members 47, 437, 439
Xtra extensions 449
Memory inspector 438
mesh deform modifier 380
mesh generator 339, 342
messages, sending to sprites 287
methods
ActiveX 205
Bones player 373
cameras 390, 396
collision 395
downloading files 475
event handling 394
keyframe animation 377
lights 384
models 349
rendering 407
sprites 408
textures 362
transforms 399, 401
vectors 398
MIAWs (movie in a window)
appearance properties 413
browsers, not supported 478
closing 411
creating 410
dockable 414
events 416
interaction, controlling 417
layering 416
listing current 417
naming 410
objects 410
opening 410
setting size and location 412
model resources
about 335
boxes 338
cast members 345
commands 316
cylinders 337
defined 313, 317, 336
extruder 344
mesh generator 339
particle systems 342
planes 339
primitives 337
properties 336
spheres 337
models
3D Xtra 305
automatic rotation 331
cast members 345
commands 318
defined 313, 315
group properties 363
interaction with cameras and lights 316
methods 349
moving 330, 351
parent-child hierarchy 346
properties 347
selecting 395
transforms 399
modes, color 146
modes, Tool palette 17
modifier keys 329
modifiers
animation 372
collision 370, 394
defined 314, 366
inker 368
level of detail (LOD) 331, 366
mesh deform 380
renderer services object 403
subdivision surfaces (SDS) 331, 369
toon 331, 368
motions
commands 321
defined 315
interaction with models 316
keyframe and bones 372
methods 382
properties 381
mouse
clicks, detecting 294
cursor colors, animating 298
Flash movies 186
pointer location detection 297
rollovers 296, 331
trigger behaviors 329
Mouse Down event 280
Mouse Enter event 280
Mouse Leave event 280
Mouse Up event 280
Mouse Within event 280
movie in a window. *See* MIAWs (movie in a window)
Movie tab, Property inspector 146

Downloaded from www.Manualslib.com manuals search engine

movies
accessibility 427, 428
browsers, playing in 478
browsers, previewing in 448
captioning 433
changing color depth of 147
color modes 146
color palettes, changing 149
cursors, animated color 301
disks 452
distribution formats 451
examples of 12
exporting as digital video 465
exporting QuickTime 466
external casts 47
Flash 181
fonts, embedding 164
importing Director within Director 202
linked media 45
linked, properties 203
local connection object 195
local network distribution 452
old versions, opening 28
older versions, converting 459
organizing 468
pausing 293
printing 444
protected 451
Publish command 452
rendering properties 405
safe mode 459
saving as Shockwave 452
streaming 469
tempo 157
testing 443, 459, 479
Update command 464
windows. *See* MIAWs (movie in a window)
Xtra extensions 449, 450
moving
accelerating and decelerating sprites 85
cast members 29
cast members in window 35
frames 21
guides 66
keyframes 87
models 330, 351
selected bitmap areas 109
sprites 66, 69
MP3 audio
controlling with Lingo or JavaScript 239
streaming 238

N

naming
cast members 29, 31
Cast panel groups 27
MIAWs (movie in a window) 410
sprite channels 57
sprites 55
naming sprites and channels 12
navigation
accessibility 429
behaviors 291
interactive features 291
jumping to different locations 293
returning to original frame 294
NetConnection object 198
NetStream object 198
network operations
background loading 470
downloading files 474
results, retrieving 476
security 477
speed 480
networks, local 452
new
behaviors 279
cast members 24
casts 23
film loops 94
guides 66
sprites 52
tiles, custom 120
New Action 281
new features in Director 12
newsprint shader 359
nodes
3D cast members 316
models 347
XML 422, 424
Not Copy ink 79
Not Ghost ink 79
Not Reverse ink 79
Not Transparent ink 79

Downloaded from www.Manualslib.com manuals search engine

O

object-relative model movement 351
objects
  3D 315
  Flash 193
  local connection 195
  MIAWs (movie in a window) 410
  renderer services 403
  scripts 393
  vector 397
offscreen graphics buffer 308
old versions
  converting movies 459
  opening movies 28
  Update Movies command 464
on getPropertyDescriptionList handler 284
onion skinning 125
online help 13
opening
  casts in new windows 29
  MIAWs (movie in a window) 410
  old movies 28
optimizing cast members 133
Options Menu, Cast panel 27
organizing
  files 441
  movies 468
outlining edges of bitmaps 110
Overwrite Existing Sprites option 21

P

Paint Brush tool 104, 107
Paint window
  Effects toolbar 109
  inks 121
  opening 102
  preferences 129
  registration points 112
  rulers 108
  tools 102
  zooming 108
painter shader properties 359
palettes
  transitions 150
palettes, color
  changing colors in 153
  changing during movies 149
  editing 151
  importing 155
  Lingo and JavaScript controls 154
  modes 146
  properties 155
  remapping 114
  troubleshooting 154
  window 151
pan camera 331
panning QuickTime VR 251
paragraph formatting 167, 177
parameters, behaviors 275, 282, 332
parent models 346
parent nodes, 3D cast members 316
parent scripts 288
parent-relative model movement 351
parser objects, XML 420, 424
Parser, XML
  about 419
  character sets 425
  objects 420
  white space, ignoring 425
particle systems
  creating 330
  primitives 335
  properties 342
Paste As Pict command 127
Paste Relative command 97
pasting
  frames 21
  keyframes 87
  sprites 97
paths, tweening sprites 84
patterns, Paint window tools 105, 120
pausing a movie 293
Pen tool
  adding points 139
  creating shapes 137
Pencil tool, Paint window 103
Perform Transition action 281
Perspective button, Paint window 110
picking models 395
PICT files
  importing 43
  Paste as command 127
  properties 128
pixels, specifying as unit of measure 168
planes 339
platforms
  font mapping 173
  keyboards 298
  projectors 463

Downloaded from www.Manualslib.com manuals search engine

Play Cast Member action 281
play done method 294
Play Every Frame digital video property 246
Play External File action 281
play method 294
Playback options, Shockwave 471
playhead
  behaviors 473
  jumping behaviors 473
  markers 330
  sprites, extending 70
playing DVD-Video 12
playing Flash movies 188
PNG images, importing 131
pointers, mouse
  behaviors 280
  click detection 294
  cursor colors, animating 298
  Flash movies 186
  location detection 297
  rollovers 296, 331
  trigger behaviors 329
points
  adding 139
  creating 137
positioning sprites 64
pre-rendered text, Save Bitmap command 176
prebuilt components 12
preferences
  browsers 479
  Cast window 36
  Paint window 129
  Score 20
  sprites 52
preloading
  cast members 38
  files into browser cache 476
preloading digital video 439
Prepare Frame event 280
Preview in Browser command 448
previous versions
  opening movies 28, 459
  Update Movies command 464
primitives 335, 337
printing movies 444
projectors
  about 460
  creating 460
  cross-platform 463
  defined 447, 451
  options 455
  organizing 468
  Xtra extensions 449, 450
  Xtra extensions, excluding 462
projects, managing 441, 468
properties
  ActiveX 205
  animated GIFs 101
  behaviors 282
  bitmaps 128
  Bones player 374
  boxes 338
  buttons 292
  cameras 386
  cast member 39
  collision modifier 370, 394
  Director movies 203
  fields 174
  film loops 95
  groups 363
  inker modifier 369
  keyframe animation 378
  keyframes 84
  lights 383
  mesh generator 339
  MIAWs (movie in a window) 413
  model resources 336
  models 347
  modifiers 366
  motions 381
  painter shader 359
  palette 155
  planes 339
  renderer services object 403
  rendering 405
  shaders 352
  sound 232
  spheres 337
  sprites 59
  textures 360
  toon modifier 368
  transforms 399
  transitions 162
  tweening 84, 86, 88
  vector shapes 142
  vectors 397
  Xtra cast members 49
Property inspector
  3D Extruder tab 324
  3D Model tab 308

Downloaded from www.Manualslib.com manuals search engine

Guides tab 66
Guides tab, Stage 18
Member tab 22, 39
MIAWs (movie in a window) 415
Movie tab 146
new features 304
sprites 59
Stage 18
Text tab 324
protected movies 451
public actions 330
public behaviors 328
Publish settings
changing 453
default 452
Files tab 455
Formats tab 454
HTML 456
Image tab 458
Projector tab 455
Shockwave tab 456
publishing 12
purgePriority 437
push buttons 292

Q

QuickTime
controlling with Lingo 250
exporting options 466
importing files 43
masks 250
QuickTime VR
inserting 251
panning 251

R

radio buttons 292
radius properties 337
real-time recording 96
rect property 412
Rectangle tool 137
rectangles, drawing 137
redraw mode 331
Registration Point tool, Paint window 102
registration points 112
registration points, sprites 65
remapping colors in bitmaps 114
renderer services object 403
rendering
methods 309, 407
properties 405
Reset Monitor option 461
resizing
bitmaps 99, 113
Flash movies 186
HTML pages 456
rotated or skewed sprites 74
sprites 71
vector shapes 99, 140
resolution for screen images 100
Restore Cursor action 281
returning to original frames 294
Reverse ink 79
RGB color
Darken and Lighten inks 78
mode settings 146
sprites, changing 75
Right Mouse Down event 280
Right Mouse Up event 280
rollovers 296, 331
Rotate button, Paint window 110
rotating
cameras 306, 307
models 331, 351
sprites 72, 186
RTF files, importing 43
rulers
Paint window 108
unit of measure 168

S

safe mode 459
saturation, color 154
Save All command 48
Save as Shockwave Movie command 452
Save Bitmap command 176
scaling
bitmaps 99, 113
Flash movies 186
HTML pages 456
rotated or skewed sprites 74
sprites 71
vector shapes 99, 140
Score
Cast to Time command 92
digital video in 246
editing frames 21
film loops 94

Downloaded from www.Manualslib.com manuals search engine

preferences 20
searching for cast members in 41
sound controls 233
Space to Time command 93
sprite display settings 20
sprite labels 62
sprites, viewing by name 56
streaming movies, behavior 473
tempo 157
windows, using multiple 19
screen resolution for graphics 100
Script window, creating cast members in 26
scripting
ActionScript instructions 190
editing 29
handlers 393
JavaScript syntax 12
preview option 20
showing in Cast window 37
scripts, behavior inheritance 288
scroll bar, displaying on Stage 175
scrolling text 179
searching for
cast members 40
sprites 57
text 170
security
Lingo and JavaScript limitations 477
Shockwave linked media 244
selecting
cast members 30
Flash components 207
keyframes 54
models 395
Paint window tools 102, 106
sprites 53
text 166
vertex 137
sending messages to sprites 287
sendSprite command 287
server technologies, integration with 13
Set Volume action 281
Settings panel, Flash movies 200
shaders
commands 319
defined 315
engraver 360
interactions with models 316
newsprint 359
painter 359
properties 352
Property inspector 309
text 325
shape cast members 142
shape drawing tools, Paint window 104
Sharp Changes tweening option 85, 88
shocked fonts 164
Shockwave
alternate image options 458
backward compatibility 459
browsers supported 447, 448
context menu, disabling 200
distribution formats 447, 451
linking to local media 244
MIAWs, not supported 409
options 456
playback options 471
Preview in Browser command 448
Publish command 452
safe mode 459
saving movies 452
testing movies 479
Xtra extensions 449, 450
Shockwave 3D window 306
Shockwave Audio
compressing internal sounds 237
controlling with Lingo or JavaScript 239
description of 237
Show Data Tips 20
Show Paths command 84
silhouettes 368
Skew button, Paint window 110
skewing sprites 72, 186
Smooth button, Paint window 111
Smooth Changes tweening option 85, 88
smoothness
anti-aliasing 310
curves 331
text 325
Snap to Grid 18, 66
Snap to Guides 18, 66
sorting
Cast List 32
cast members 35
sound
AVI files 465
cast member properties 232
channel 233
channels 236
compressing internal 237

Downloaded from www.Manualslib.com manuals search engine

controlling with Lingo or JavaScript 239
importing 231
Lingo and JavaScript controls 235
looping 234
playing external 239
Shockwave Audio 237
streaming Shockwave Audio 238
streaming, formats supported 232
synchronizing with cue points 240, 259
transitions, playing during 161
Windows issues 234
WMA files 236
sourceRect property 412
Space to Time command 93
spacing, lines 168
Speech Xtra 428, 435
speech, converting text to 428, 431, 434
speed
downloading 480
Lock Frame Durations 159
network operations 470
particles 344
tempo 157
tweening 85
spheres 330, 337
splash screens 182
splitting
sprites 70
vector curves 140
Sprite Overlay 61
Sprite toolbar 60
sprites
3D behaviors, attaching 332
Align window 66, 67
anti-aliasing 310
assigning cast members 80
behaviors, attaching 276
blending 76
cast members, exchanging 89
cast members, reusing 71
Cast to Time command 92
changing frames 69
channels, naming 57
color settings 75, 147
coordinates 65
copying 69
creating 52
Darken and Lighten inks 78
defined 51
Direct to Stage option 308
draggable 295
duration, changing 70
editable 295
editing frames 21, 89
finding 57
flipping 75
frame-by-frame animation 90
inks 77
joining 70
labels 62
layering 58
locking and unlocking 63
Mask ink 77
moving to different frames 69
naming 12, 55
Overlay 61
Paste Relative command 97
paths, tweening 84
positioning 64
preferences 52
properties 59, 63
rendering commands 408
rendering properties 408
resizing 71
resizing rotated or skewed 74
rollovers 296
rotating and skewing 72
Score display, settings 20
selecting 53
sending messages to 287
Snap features 66
Snap to Grid 18
Snap to Guides 18
Space to Time command 93
speed, settings 85
splitting 70
step recording 96
toolbar 60
Trails option 60, 97
transitions 160
Tweak window 66
tweening 83
tweening properties 86
tweening settings 88
Width and Height settings 60
Stage
cast members, creating in 26
color 147
Direct to Stage option 308
Flash movies, placing 187

Downloaded from www.Manualslib.com manuals search engine

guides and grid 18
moving sprites 69
positioning sprites 65
Property inspector 18
rendering methods 309
Sprite Overlay 61
sprite preference 52
sprites, viewing by name 56
Standard Import option 43
step recording 96
Storyboard Format 445
streaming
controlling with Lingo or JavaScript 474
Flash movies 187
loading status, checking 474
movies 469
playback options 471
score-based 473
Shockwave Audio 237, 238
sound, formats supported 232
WMA files 236
strings, checking 179
stroke settings for vector shapes 138
subdivision surfaces (SDS) 331, 369
Subtract ink 80
Support Center 15
Switch Colors button, Paint window 111
Sync to Soundtrack digital video property 246
synchronizing
cue points with Lingo 241
events 157
sound 240

T

tabbing, text 178
tabs, Cast panel
rearranging 28
saving 28
tempo
channel 157
comparing actual 159
controlling from Lingo or JavaScript 159
Lock Frame Durations 159
settings 157
testing movies 443, 459, 479
text
3D 324
anti-aliased 170
character formatting 167
converting to bitmap 173, 176
creating 165
editable 172
editing 166
embedding in a movie 164
features 163
field 163
field properties 174
finding and replacing 170
font mapping, cross-platform 173
formatting cast members 169
formatting from Lingo or JavaScript 176
hypertext links 171
importing 165
inspector, formatting with 169
kerning 170
line spacing 168
Paint window tool 103
paragraph formatting 167
printing 444
scrolling 179
selecting 166
user click detection 296
wrapping and tabbing 178
text boxes, formatting with Lingo or JavaScript 178
text files, importing 43
text-to-speech scripting 428, 431, 434
textures
commands 320
defined 315, 360
interactions with models 316
mesh generator 340
methods 362
properties 360
text 325
thumbnails, cast members
custom, creating 34
preferences 37
viewing icons 33
tiles, custom 120
Tool palette 17
tooltips 15
toon modifier 331, 368
Trace Edges button, Paint window 110
Trails option
redraw mode 331
sprites 60, 97
Transform Bitmap command 113
transforms 399
transitions 150, 160
transparency effects, sprites 77

Downloaded from www.Manualslib.com manuals search engine

Transparent ink 79
trigger behaviors
  defined 328
  groups 333
  parameters 333
  viewing 328
troubleshooting
  color palettes 154
  resources 445
  testing movies 443
Truncate Sprites Being Pasted option 21
tunnel depth 325
Tweak window 66
tweening
  bitmap filters 124
  defined 83
  paths 84
  properties 84
  settings, changing 88
  speed 85
  sprite properties 86
  sprites 83
  tips 87

U

unlinking external casts 47
unloading cast members 47, 437, 439
unlocking sprites 63
Update Movies command 464
updating movies from older versions 459
URLs
  distributing movies 451
  getURL action 190
  jumping to 293
  maximum length 44
  opening in browsers 475
  references in Lingo or JavaScript 477
  third-party extensions 449
user interaction 291

V

variables, Flash movies 189
vector objects 397
Vector Shape window
  defined 135
  editing shapes 139
  zooming 136
vector shapes
  cast members 142, 143
  color settings 147
  controlling with Lingo or JavaScript 142
  corner points 137
  creating 135
  curve points 137
  defined 99
  editing 139
  fill and line settings 138
  gradients 141
  Pen tool 137
  properties 142
  resizing 99
  scaling 140
  setting attributes with Lingo or JavaScript 138
Verisign Xtra downloading 449
versions, previous
  opening movies 28, 459
  Update Movies command 464
vertex
  colors 135
  defined 135
  selecting 137
Video window 245
video. *See* digital video
Volume Control 472

W

Wait for Cue Point tempo option 158
Wait for Mouse Click tempo option 158
Wait for Time Duration action 281
Wait on Current Frame action 281
Wait Until Click action 281
Wait Until Key Press action 281
Warp button, Paint window 110
Web216 color palette 149, 155
Windows
  ActiveX 204
  browsers supported 447, 448
  font mapping 173
  keyboards 298
  projectors 463
  sound issues 234
Windows Media Audio (WMA) 236
windows, movies in. *See* MIAWs (movie in a window)
windows, using multiple Score 19
WMA (Windows Media Audio) 236

Downloaded from www.Manualslib.com manuals search engine

workspace
customizing 12
world-relative model movement 351
wrapping, text 178

X

XML
about 419
attributes 424
character sets 425
data, sending from Flash to Director 193
nodes 422, 424
parser objects 420, 424
websites, resources 420
white space, ignoring 425
Xtra extensions
3D 305
Auto Filter command 124
bitmap filters 123
cast member properties 49
cursor animation support 299
distributing movies 449, 450
excluding from projectors 462
included 449
linked media 45
properties 39
Speech 428, 435
transitions 161
XML Parser 419, 420

Z

zero point, moving for paint rulers 108
zooming
Paint window 103, 108
Vector Shape window 136

Downloaded from www.Manualslib.com manuals search engine