JavaScript™ for Acrobat® 3D Annotations API Reference

Adobe® Acrobat® SDK

April 2007

Version 8.1

© 2007 Adobe Systems Incorporated. All rights reserved.

Adobe® Acrobat® SDK 8.1 JavaScript for Acrobat 3D Annotations API Reference for Microsoft® Windows® and Mac OS®

Edition 2.0, April 2007

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

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

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

Any references to company names, company logos and user names in sample material or sample forms included in this documentation and/or software are for demonstration purposes only and are not intended to refer to any actual organization or persons.

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

JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

Mac OS is a trademark of Apple Computer, Inc., registered in the United States and other countries.

Microsoft and Windows are either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

All other trademarks are the property of their respective owners.

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

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

Contents

Preface **8**
What's in this guide? 8
Who should read this guide? 8
Related documentation 8

1 Introduction **9**
Object overview 10
Basic objects 10
Scene object 10
Canvas object 10
Runtime object 11
Console object 11
Resource objects 11
Event handlers 11
CamaraEvent 11
KeyEvent 12
MenuEvent 12
MouseEvent 12
RenderEvent 12
ScrollWheelEvent 13
SelectionEvent 13
TimeEvent 13
ToolEvent 13

2 JavaScript Objects for Acrobat 3D **14**
Animation 15
Background 16
getColor 16
getImage 16
setColor 16
setImage 17
Bone 18
BoundingBox 19
Camera 20
getScreenFromPosition 21
getDirectionFromScreen 22
CameraEvent 23
CameraEventHandler 24
CameraEventHandler 24
onEvent 24
Canvas 25
getCamera 25
setCamera 25
ClippingPlane 26
remove 26

2 JavaScript Objects for Acrobat 3D (Continued)

Color 27
Color 27
Color 27
set 27
set 28
set3 28
Console 29
print 29
println 29
Dummy 30
HitInfo 31
Host 32
Image 33
Image 33
KeyEvent 34
KeyEventHandler 36
KeyEventHandler 36
onEvent 36
Light 37
Material 39
Matrix4x4 40
Matrix4x4 40
Matrix4x4 40
invertInPlace 40
isEqual 41
multiply 41
multiplyInPlace 41
rotateWithQuaternion 42
rotateWithQuaternionInPlace 42
rotateAboutLine 42
rotateAboutLineInPlace 43
rotateAboutX 43
rotateAboutXInPlace 44
rotateAboutVector 44
rotateAboutVectorInPlace 44
rotateAboutY 45
rotateAboutYInPlace 45
rotateAboutZ 45
rotateAboutZInPlace 46
scale 46
scaleInPlace 46
set 47
setIdentity 47
setView 47
transformDirection 48
transformPosition 48
translate 49
translateInPlace 49
transposeInPlace 49

2 JavaScript Objects for Acrobat 3D (Continued)

MenuEvent....50
MenuEventHandler....51
MenuEventHandler....51
onEvent....51
Mesh....52
MouseEvent....53
MouseEventHandler....55
MouseEventHandler....56
onEvent....56
Node....57
computeBoundingBox....58
detachFromCurrentAnimation....58
Procedural....59
Quaternion....60
Quaternion....60
Quaternion....60
Quaternion....60
interpolate....61
interpolateInPlace....61
normalize....61
RenderEvent....62
RenderEventHandler....63
RenderEventHandler....63
onEvent....63
RenderOptions....64
Resource....66
Resource....66
Runtime....67
addCustomMenuItem....70
addCustomToolButton....70
addEventHandler....71
disableTool....71
enableTool....71
getEventHandler....72
getRendererName....72
refresh....72
removeEventHandler....72
removeCustomMenuItem....73
removeCustomToolButton....73
setCurrentTool....73
Scene....75
activateAnimation....81
addModel....81
createClippingPlane....81
createLight....81
createSquareMesh....82
computeBoundingBox....82
update....82
SceneObject....83

2 JavaScript Objects for Acrobat 3D (Continued)

SceneObjectList 84
getByGUID 84
getByID 84
getByIndex 84
getByName 85
removeAll 85
removeByIndex 85
removeItem 86
ScrollWheelEvent 87
ScrollWheelEventHandler 88
ScrollWheelEventHandler 88
onEvent 88
SelectionEvent 89
SelectionEventHandler 90
SelectionEventHandler 90
onEvent 90
Texture 91
getImage 91
setImage 92
TimeEvent 93
TimeEventHandler 94
TimeEventHandler 94
onEvent 94
ToolEvent 95
ToolEventHandler 96
ToolEventHandler 96
onEvent 96
Vector3 97
Vector3 97
Vector3 97
add 98
addInPlace 98
addScaled 98
addScaledInPlace 99
blend 99
blendInPlace 99
cross 100
dot 100
normalize 100
scale 101
scaleInPlace 101
set 101
set 102
set3 102
subtract 102
subtractInPlace 103

3 New Features and Changes 104
Acrobat 8.1 changes 104
New objects 104
Additional properties in existing objects 104
Deprecated objects or properties 104
Acrobat 8.0 changes 104
Additional properties in existing objects 105
Index 106

Preface

The JavaScript™ API allows you to manipulate 3D annotations within Adobe® PDF documents.

What's in this guide?

This document provides a brief overview of the API followed by a description of the objects.

Who should read this guide?

This guide is for developers who want to enhance the 3D experience of the user beyond the default behaviors. Using the JavaScript API for 3D annotations, you can specify the render modes and 3D matrix transformations of any of the individual meshes; set camera position, target, and field of view; detect mouse and keyboard events; control animations; and many more behaviors.

Related documentation

This document refers to the following sources for additional information about 3D annotations, JavaScript and related technologies. The Adobe Acrobat® documentation is available through the Acrobat Family Developer Center, http://www.adobe.com/go/acrobat_developer.

Document	Description
Developing Acrobat Applications Using JavaScript	Using JavaScript to develop and enhance standard workflows in Acrobat and Adobe Reader®.
JavaScript for Acrobat API Reference	Detailed descriptions of JavaScript APIs for developing and enhancing workflows in Acrobat and Adobe Reader.
PDF Reference	A detailed description of the PDF file format.

1 Introduction

To create 3D annotations and to attach scripts to them using this API, you will need Adobe® Acrobat® Professional and Acrobat 3D. Scripts attached to 3D annotations can run on Acrobat Professional, Acrobat Standard, and Adobe Reader® for Windows® and Mac OS® platforms. Unless otherwise noted, all JavaScript objects, properties, and methods have support starting in version 7.0.

The 3D JavaScript engine, which is distinct from the JavaScript engine for Acrobat, can be accessed in one of two ways. The primary way is by attaching a default script to the 3D annotation. This can be accomplished while placing a 3D annotation using the 3D Tool or on an existing 3D annotation by accessing its properties dialog box using the Select Object tool. This script will be run directly by the 3D JavaScript engine.

In addition, Acrobat provides a mechanism to directly access the entire 3D JavaScript engine API from within the Acrobat scripting engine by means of the JavaScript `Annot3D.context3D` property. For more details about JavaScript for Acrobat and its `Annot3D` object, see the *JavaScript for Acrobat API Reference* and *Developing Acrobat Applications Using JavaScript*.

The following example illustrates how to access the 3D JavaScript engine. In this example, a button (or link) contains JavaScript code that rotates the U3D object named "Axes".

```
// Get index of page containing the Annot3D object (count starts at 0).
pageIndex = this.pageNum;

// Index of the Annot3D (count starts at 0).
annotIndex = 0;

// Get a reference to the Annot3D script context.
c3d = this.getAnnots3D( pageIndex )[ annotIndex ].context3D;

// Get a reference to the node in the scene named "Axes".
axes = c3d.scene.nodes.getByName( "Axes" );

// Rotate the object about the X-Axis PI/6 radians (30 degrees).
axes.transform.rotateAboutXInPlace( Math.PI / 6 );
```

More extensive actions can be executed by having a button or link get the `SceneContext3d` object and call a function defined in the default script of the 3D annotation, as in the following example.

```
// Get the Annot3D script context of the targeted annot.
context3D = getAnnots3D(0)[0].context3D;

// Call the JavaScript function setRenderMode() defined in the default
// script of the referenced 3D annotation.
context3D.setRenderMode("transparent");
```

The default script of the 3D annotation makes the definition.

```
function setRenderMode( renderModeName ) {
   for (var i=0; i < scene.meshes.count; i++) {
      scene.meshes.getByIndex(i).renderMode = renderModeName;
   }
}
```

Object overview

This section provides an overview of the objects in the 3D JavaScript API.

Basic objects

There are several basic objects, such as `Color`, `Matrix4x4`, and `Vector3`, that are used to create general-purpose objects. The basic objects are used throughout the API and are only meaningful when attached to objects such as `Scene` or `Runtime`. For example, you could create a `Color` object and use it to set the `Background` color of a `Canvas`.

Vector3 Examples

```
v1 = new Vector3( 1.2, 3, 4.5 );
v2 = new Vector3( 5, 8, 13 );
v3 = new Vector3();
```

Matrix4x4 Examples

```
m1 = new Matrix4x4().rotateAboutX(Math.PI/1.5).rotateAboutY(Math.PI/3);
m2 = new Matrix4x4().rotateAboutZ(Math.PI/4).translate(new Vector3(0,5,0));
m3 = new Matrix4x4(m1);
```

Color Examples

```
c1 = new Color( 0.6, 0.8, 1.0 ); // light blue
c2 = new Color( 0.5, 0.5, 0.5 ); // middle grey
c3 = new Color(); //black

// A function to blend two Colors
Color.prototype.blend = function( color, amount )
{
   red     = ( this.r * ( 1 - amount ) ) + ( color.r * amount );
   green   = ( this.g * ( 1 - amount ) ) + ( color.g * amount );
   blue    = ( this.b * ( 1 - amount ) ) + ( color.b * amount );
   return( new Color( red, green, blue ) );
}
c4 = c1.blend( c2, 0.25 );
```

Scene object

The `Scene` is an object that contains all of the 3D-related content. It can be accessed using the global variable `scene`, which is a reference to the main `Scene` object. Most of the contents of the `Scene` are structured into a hierarchy of `Node` objects, and maintains lists of all these objects in the form of a `SceneObjectList`.

For more information, see `Scene` on page 75.

Canvas object

Represents a rectangular region into which a `Scene` is rendered from a particular viewpoint.

For more information, see `Canvas` on page 25.

Runtime object

The `Runtime` object is used to represent the instance of the playback engine. It manages all event processing and places where the graphic and textual content is rendered. It is accessed via the global variable `runtime`, which is a reference to the main `Runtime` object.

For more information, see "Runtime" on page 67.

Console object

The Console is the Acrobat text output area. It is helpful in debugging scripts.

Resource objects

Some objects, such as `Image`, are driven by content that is streamed from a file or over a network. To create an `Image`, load a .png, .jpg, or .gif file as a `Resource`, which you may subsequently use to create a new `Image` object, as shown in the following example:

```
faceRes = new Resource("pdf://picture.jpg");
faceImage = new Image( faceRes );
aMaterial = scene.meshes.getByIndex(0).material;
aMaterial.diffuseTexture.setImage( faceImage );
```

The `Resource` and `Image` objects are covered on page 66 and page 33, respectively.

Event handlers

There are several types of event handlers:

- `CameraEventHandler`
- `KeyEventHandler`
- `MouseEventHandler`
- `MenuEventHandler`
- `RenderEventHandler`
- `ScrollWheelEventHandler`
- `SelectionEventHandler`
- `TimeEventHandler`
- `ToolEventHandler`

Each one responds to a different type of event during simulation. They use a callback mechanism to run a function when an event occurs. The event is passed as an argument to the event handler's `onEvent` function so that it can be queried when the function runs. Event handlers are registered via the `addEventHandler` method, page 71, of the `Runtime` object.

CamaraEvent

A `CamaraEvent` is created when a View is selected.

For information, see `CameraEvent` on page 23.

KeyEvent

A `KeyEvent` is created when a key is pressed or released while the 3D `Canvas` is in focus. The following example illustrates how to handle a key event:

```
myKeyHandler = new KeyEventHandler();
myKeyHandler.onEvent = function( event )
{
   console.print( "Key pressed with code: " + event.characterCode );
}
runtime.addEventHandler( myKeyHandler );
```

For information, see `KeyEvent` on page 34.

MenuEvent

A `MenuEvent` is created when a custom menu item is selected. To create a custom menu item on the context menu, invoke the `Runtime` object's `addCustomMenuItem` method, which allows a script to be attached to the item selection event.

For more information, see `MenuEvent` on page 50.

MouseEvent

A `MouseEvent` is created when the mouse is clicked on an active 3D `Canvas` or the cursor moves over an active 3D `Canvas`. The following syntax could be used to handle a mouse event:

```
myMouseHandler                = new MouseEventHandler();
myMouseHandler.onMouseDown    = true;
myMouseHandler.target         = scene.meshes.getByIndex(0);
myMouseHandler.onEvent        = function( event )
{
   console.print( "Mouse down at pixel " + event.mouseX );
   console.print( ", " + event.mouseY );
}
runtime.addEventHandler( myMouseHandler );
```

For more information, see `MouseEvent` on page 53.

RenderEvent

A `RenderEvent` is created immediately before an instance of the `Canvas` is drawn. If there is a split view in Acrobat resulting in two visible 3D rendered areas, a unique `RenderEvent` will be called for each of them. This is necessary in the case of a camera-aligned image (sprite) in the 3D content that needs to be pixel-aligned. Since the pixel dimensions of the two areas are possibly different, there are two callbacks that pass the different dimensions. This makes it possible to modify the `Scene` in the appropriate manner before it is drawn.

For more information, see `RenderEvent` on page 62.

ScrollWheelEvent

A `ScrollWheelEvent` object is created when the mouse scroll wheel is activated over an active 3D `Canvas` object.

For more information, see `ScrollWheelEvent` on page 87.

SelectionEvent

A `SelectionEvent` object is created when an object is selected from an active 3D `Canvas` object or from a model tree. If the selection is made from a `Canvas` object, a `MouseEvent` is also created.

For more information, see `SelectionEvent` on page 89.

TimeEvent

A `TimeEvent` is created when the 3D annotation is enabled and simulation is active. The time and deltaTime properties are measured in terms of simulation time, not real time. `TimeEvent` objects are used to drive animation. If you need an accurate, real-time measurement, use the JavaScript `Date` object. The following syntax is used to handle a time event:

```
myTimeHandler            = new TimeEventHandler();
myTimeHandler.onEvent    = function( event )
{
   console.print( "Current simulation time is:" + event.time );
   console.print( " second(s)" );
}
runtime.addEventHandler( myTimeHandler );
```

For more information on the TimeEvent, see page 93.

ToolEvent

A `ToolEvent` is created when a tool is clicked in the Acrobat 3D toolbar. The `Runtime` object's `addCustomToolButton` method allows you to add a custom tool to the toolbar which will also be generated, and allows a script to be attached to the tool selection event.

For more information, see "ToolEvent" on page 95.

2 JavaScript Objects for Acrobat 3D

This chapter describes the following 3D JavaScript objects:

Animation

Background

Bone

BoundingBox

Camera

CameraEvent

CameraEventHandler

Canvas

ClippingPlane

Color

Console

Dummy

HitInfo

Host

Image

KeyEvent

KeyEventHandler

Light

Material

Matrix4x4

MenuEvent

MenuEventHandler

Mesh

MouseEvent

MouseEventHandler

Node

Procedural

Quaternion

RenderEvent

RenderEventHandler

RenderOptions

Resource

Runtime

Scene

SceneObject

SceneObjectList

ScrollWheelEvent

ScrollWheelEventHandler

SelectionEvent

SelectionEventHandler

Texture

TimeEvent

TimeEventHandler

ToolEvent

ToolEventHandler

Vector3

Note: A property labeled as read-only is one whose value cannot be set. An object labeled as read-only is one whose reference cannot be modified, though the object itself can be set and its properties may be modified. Unless otherwise indicated, all properties and objects are assumed to have read/write access.

Animation

A type of `SceneObject`, page 83, used to store keyframe animation sequences of `Node` objects in the Scene. In addition to the methods and properties below, it also contains the same methods and properties as a `SceneObject`.

Properties

Property	Type	Access	Description
`currentTime`	number		The current time measured in seconds.
`endTime`	number	read-only	The end time of the sequence, measured in seconds.
`framesPerSecond`	number	read-only	The number of frames per second used to author the sequence.
`length`	number	read-only	The length of the `Animation`, measured in seconds.
`startTime`	number	read-only	The start time of the sequence, measured in seconds.

Background

Represents the background of a `Canvas`. It can be used as a target of a `MouseEventHandler`.

For information on the `Canvas` and `MouseEventHandler`, see page 25 and page 55, respectively.

Properties

Property	Type	Access	Description
`image`	Image		Acrobat 7.0.7 The `Image` to be used by the `Background`.

getColor

Obtains the background `Color`.

Syntax

```
getColor()
```

Returns

A `Color` object representing the background color of the `Canvas`.

getImage

Deprecated

Obtains the background `Image`.

Syntax

```
getImage()
```

Returns

An `Image` object representing the background image of the `Canvas`.

setColor

Sets the background `Color`. If only one color is passed to this method, the background is a constant color. If two colors are passed to this method, the background will be a linear gradient from top to bottom, with the first color argument representing the top color and the second representing the bottom color.

Syntax

```
setColor(topColor, bottomColor)
```

Parameters

`topColor`	A `Color` object representing the desired background color. If `bottomColor` is used, `topColor` represents the top background color used in a linear gradient.
`bottomColor`	(Optional) A `Color` object representing the bottom background color used in a linear gradient.

Returns

undefined

setImage

Deprecated

Sets the background Image.

Syntax

```
setImage(image)
```

Parameters

`image`	An `Image` object representing the desired background image.

Returns

undefined

Bone

A type of `Node` used to modify the shape of a `Mesh`, and is usually moved over time to create animated characters. It contains the same methods and properties as a `Node`.

Related objects are `Node` on page 57 and `Mesh` on page 52.

BoundingBox

Represents an axis-aligned bounding box.

Properties

Property	Type	Access	Description
`center`	Vector3	read-only	Acrobat 7.0.7 The coordinates of the `BoundingBox` center.
`max`	Vector3	read-only	The coordinates of the `BoundingBox` corner with the greatest x, y, and z values.
`min`	Vector3	read-only	The coordinates of the `BoundingBox` corner with the smallest x, y, and z values.

Camera

A `Node` that controls the projection from world space to screen space. In addition to the methods and properties below, it also contains the same methods and properties as a `Node`. (See `Node` on page 57.)

Properties

Property	Type	Access	Description
`binding`	string		The view plane calculation type, which can take one of the following values: • "min" • "max" • "horizontal" • "vertical"
`BINDING_HORIZONTAL`	string	read-only	Acrobat 7.0.7 A string constant for the binding value of "horizontal".
`BINDING_MAX`	string	read-only	Acrobat 7.0.7 A string constant for the binding value of "max".
`BINDING_MIN`	string	read-only	Acrobat 7.0.7 A string constant for the binding value of "min".
`BINDING_VERTICAL`	string	read-only	Acrobat 7.0.7 A string constant for the binding value of "vertical".
`far`	number		The distance from the `Camera` to the far clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`fov`	number		The size of the field of view for perspective `Camera` objects, measured in radians.
`near`	number		The distance from the `Camera` to the near clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`position`	Vector3	read-only	The position of the origin of the `Camera` in world space.
`positionLocal`	Vector3	read-only	The position of the origin of the `Camera` in local space.
`projectionType`	string		The type of projection, which can take one of the following values: • "perspective" • "orthographic"

Property	Type	Access	Description
`roll`	number		The roll angle of the `Camera`, measured in radians.
`target`	Node	read-only	The current `Node` used as the `Camera` object's target.
`targetPosition`	Vector3	read-only	The position of the `Camera` object's target in world space.
`targetPositionLocal`	Vector3		The position of the `Camera` object's target in local space.
`TYPE_ORTHOGRAPHIC`	string	read-only	Acrobat 7.0.7 A string constant for the camera projection type of "orthographic".
`TYPE_PERSPECTIVE`	string	read-only	Acrobat 7.0.7 A string constant for the camera projection type of "perspective".
`up`	Vector3	read-only	The up direction in world space.
`upLocal`	Vector3	read-only	The up direction in local space.
`viewPlaneSize`	number		The size of the view plane for orthographic `Camera` objects, measured in scene units.

getScreenFromPosition

Obtains the screen coordinates of the provided 3D position.

Syntax

```
getScreenFromPosition(position, canvasWidth, canvasHeight)
```

Parameters

`position`	A `Vector3` object representing the 3D position.
`canvasWidth`	The width of the `Canvas`, measured in pixels.
`canvasHeight`	The height of the `Canvas`, measured in pixels.

Returns

A `Vector3` object representing the screen coordinates, with `x` and `y` as pixel positions and `z` equal to zero

See "Vector3" on page 97 for more information on the return object.

getDirectionFromScreen

Obtains the direction from the normalized coordinates

Syntax

```
getDirectionFromScreen(x, y, canvasWidth, canvasHeight)
```

Parameters

`x`	The x-coordinate, measured in pixels.
`y`	The y-coordinate, measured in pixels.
`canvasWidth`	The width of the `Canvas`, measured in pixels.
`canvasHeight`	The height of the `Canvas`, measured in pixels.

Returns

A `Vector3` object representing the direction

See "Vector3" on page 97 for more information on the return object.

CameraEvent

Describes the format of the object that is passed as an argument to the `onEvent` method of the `CameraEventHandler` object.

Properties

Property	Type	Access	Description
`binding`	string	read-only	The view plane calculation type, which can take one of the following values: • "min" • "max" • "horizontal" • "vertical"
`canvas`	Canvas	read-only	The `Canvas` in which the event took place.
`currentTool`	string	read-only	The name of the current tool.
`far`	number	read-only	The distance from the `Camera` to the far clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`fov`	number	read-only	The size of the field of view for perspective `Camera` objects, measured in radians.
`isNewCanvas`	Boolean	read-only	Deprecated Determines whether this is the first event for this `Canvas`.
`near`	number	read-only	The distance from the `Camera` to the near clipping plane. A value of -1 for both `near` and `far` signifies to use auto clipping plane calculations.
`projectionType`	string	read-only	The type of projection, which can take one of the following values: • "perspective" • "orthographic"
`targetDistance`	Vector3	read-only	The distance from the `Camera` to its target.
`transform`	Matrix4x4	read-only	The Camera object's transformation matrix.
`viewPlaneSize`	number	read-only	The size of the view plane, measured in scene units.

CameraEventHandler

Exposes a callback mechanism that allows a function to be evaluated when an camera event occurs. Event handlers are registered with the Runtime `addEventHandler` method, page 71.

CameraEventHandler

Constructor

Syntax

```
new CameraEventHandler()
```

Returns

A `CameraEventHandler` object

onEvent

A method that is called when a view is selected from the list of views on the 3D toolbar or in the context menu for an active 3D annotation.

syntax

```
onEvent(event)
```

Parameters

`event`	A `CameraEvent` object representing the event.

Returns

undefined

Canvas

Represents a rectangular region into which the `Scene` is rendered from the viewpoint of the attached `Camera`.

See related objects, `Scene` on page 75 and `Camera` on page 20.

Properties

Property	Type	Access	Description
`background`	Background	read-only	The `Background` object associated with the `Canvas`.

getCamera

Obtains the `Camera` object attached to the `Canvas`.

Syntax

```
getCamera()
```

Returns

A `Camera` object.

setCamera

Sets the `Camera` object attached to the `Canvas`.

Syntax

```
setCamera(camera)
```

Parameters

`camera`	The `Camera` object used to set the object's value.

Returns

undefined

ClippingPlane

An object representing a plane, within the `Scene`, that clips all geometry on one side of it. It is created by invoking the `createClippingPlane` method of the `Scene` object, described on page 81.

remove

Removes the `ClippingPlane` object from the `Scene`.

Syntax

```
remove()
```

Returns

undefined

Color

An object that represents a RGB encoded color.

Properties

Property	Type	Description
`b`	number	The blue component, which may contain a value from `0.0` to `1.0`.
`g`	number	The green component, which may contain a value from `0.0` to `1.0`.
`r`	number	The red component, which may contain a value from `0.0` to `1.0`.

Color

Constructor

Syntax

```
new Color()
```

Returns

A `Color` object, initialized to black

Color

Constructor

Syntax

```
new Color(r, g, b)
```

Parameters

`r`	The red component, which may contain a value from `0.0` to `1.0`.
`g`	The green component, which may contain a value from `0.0` to `1.0`.
`b`	The blue component, which may contain a value from `0.0` to `1.0`.

Returns

A `Color` object, initialized to the supplied RGB values

set

Sets the `Color` object's value using an existing `Color` object

Syntax

```
set(color)
```

Parameters

`color`	The `Color` object used to set the object's value.

Returns

undefined

set

Acrobat 7.0.7

Sets the `Color` object's value using the given RGB components.

Syntax

```
set(r, g, b)
```

Parameters

`r`	The red component, which may contain a value from `0.0` to `1.0`.
`g`	The green component, which may contain a value from `0.0` to `1.0`.
`b`	The blue component, which may contain a value from `0.0` to `1.0`.

Returns

undefined

set3

Deprecated

Sets the `Color` object's value using the given RGB components.

Syntax

```
set3(r, g, b)
```

Parameters

`r`	The red component, which may contain a value from `0.0` to `1.0`.
`g`	The green component, which may contain a value from `0.0` to `1.0`.
`b`	The blue component, which may contain a value from `0.0` to `1.0`.

Returns

undefined

Console

This object can direct output to the Acrobat console for debugging purposes. The variable `console` is a global reference to this object.

print

Prints a string to the console.

Syntax

```
print(string)
```

Parameters

`string`	The text to be printed to the console.

Returns

undefined

println

Prints a string with an accompanying newline to the console.

Syntax

```
println(string)
```

Parameters

`string`	The text to be printed to the console.

Returns

undefined

Dummy

Deprecated

A `Node` object used as an empty placeholder or a group within a `Scene`.

HitInfo

The object returned when a hit test occurs during a `MouseEvent`, page 50.

Properties

Property	Type	Access	Description
`distance`	number	read-only	The distance from the `Camera` to the `HitInfo` object's `position`.
`material`	Material	read-only	Acrobat 8.1 The material of the node that was hit.
`position`	Vector3	read-only	The position of the point where the hit occurred.
`surfaceNormal`	Vector3	read-only	Acrobat 8.1 World-space surface normal direction at hit location.
`target`	Node	read-only	The target of the hit test.
`textureCoordinate`	Vector3	read-only	Acrobat 8.1 The texture coordinate of the material that was hit.

Host

Acrobat 7.0.7

An object that provides access to the JavaScript engine context and to pertinent objects within it. The variable `host` is a global reference to this object. It is a reference to the JavaScript `Document` object in which the 3D annotation is contained.

Image

An object that represents an image.

Properties

Property	Type	Access	Description
`height`	number	read-only	The image's height, measured in pixels.
`width`	number	read-only	The image's width, measured in pixels.

Image

Constructor

Syntax

```
new Image(resource)
```

Parameters

`resource`	An `Image` object used to create the new object.

Returns

An `Image` object

See "Image" on page 33 for more information on the return object.

KeyEvent

An object that is passed as an argument to the `onEvent` method, page 24, of the `KeyEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	read-only	The `Canvas` in which the `KeyEvent` took place
`canvasPixelHeight`	integer	read-only	The height, measured in pixels, of the `Canvas`
`canvasPixelWidth`	integer	read-only	The width, measured in pixels, of the `Canvas`
`characterCode`	integer	read-only	The value of the character pressed according to Acrobat's character mapping, as per this listing of Acrobat character codes:

#	Keys	#	Keys	#	Keys
		65	A	97	a
		66	B	98	b
		67	C	99	c
		68	D	100	d
28	Left	69	E	101	e
29	Right	70	F	102	f
30	Down	71	G	103	g
31	Up	72	H	104	h
		73	I	105	i
		74	J	106	j
32	Space	75	K	107	k
		76	L	108	l
		77	M	109	m
48	0	78	N	110	n
49	1	79	O	111	o
50	2	80	P	112	p
51	3	81	Q	113	q
52	4	82	R	114	r
53	5	83	S	115	s
54	6	84	T	116	t
55	7	85	U	117	u
56	8	86	V	118	v
57	9	87	W	119	w
		88	X	120	x
		89	Y	121	y
		90	Z	122	z

Property	Type	Access	Description
`ctrlKeyDown`	Boolean	read-only	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed. **Note:** Acrobat will intercept many of the Ctrl + key events because they are used for accelerators in the main application.
`currentTool`	string	read-only	The name of the current tool.
`shiftKeyDown`	Boolean	read-only	Determines whether the Shift key was pressed. **Note:** Holding the shift key down changes the value of the `KeyEvent.characterCode` property.

KeyEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when a key event occurs. Event handlers are registered with the Runtime `addEventHandler` method, described on page 71.

KeyEventHandler

Constructor

Syntax

```
new KeyEventHandler()
```

Returns

A `KeyEventHandler` object

onEvent

A method that is called when a key is pressed.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `KeyEvent` object representing the event.

Returns

undefined

Light

A `Node` object that illuminates meshes in the `Scene`. There are different types of `Light` objects, each with their own distinct behavior. Infinite `Light` objects behave much like sunlight in that they cast parallel light in a given direction. Spot `Light` objects have a fixed cone angle that limits their beam to a conical projection. Point `Light` objects act similarly to a light bulb, where the light comes from a specific location in 3D space. Currently, none of the `Light` objects cast shadows. In addition to the methods and properties below, it also contains the same methods and properties as a `Node`.

Properties

Property	Type	Access	Description
`attenuationA`	number		The `a` coefficient for `attenuationType` "abc".
`attenuationB`	number		The `b` coefficient for `attenuationType` "abc".
`attenuationC`	number		The `c` coefficient for `attenuationType` "abc".
`attenuationType`	string		The style of attenuation for the `Light` object being represented. Attenuation determines how fast the light intensity decreases with distance. The attenuation type of “abc” uses the equation 1 / max((a + bd + cdd), 1) to determine the intensity where d is the distance from the light. One of the following values may be assigned: • "abc" • "none"
`ATTENUATION_ABC`	string	read-only	Acrobat 7.0.7 A string constant for the `attenuationType` of "abc".
`ATTENUATION_NONE`	string	read-only	Acrobat 7.0.7 A string constant for the `attenuationType` of "none".
`brightness`	number		Specifies the brightness of the emission from the `Light`. A value of `1` represents a brightness of 100%, though the property may be assigned higher values.
`color`	Color	read-only	Specifies the color of the light.
`direction`	Vector3	read-only	The direction toward which the light is pointing.
`directionLocal`	Vector3	read-only	Acrobat 7, but not documented until Acrobat 8.1 The direction toward which the light is pointing relative to its parent Node.

Property	Type	Access	Description
`innerConeAngle`	number		The angle, measured in radians, about the `direction` in which the light is of uniform full density.
`innerRadius`	number		The distance within which the light is of uniform full density.
`outerConeAngle`	number		The angle, measured in radians, about the `direction` outside of which the light's intensity is zero.
`outerRadius`	number		The distance beyond which the light's intensity is zero.
`position`	Vector3	read-only	The position of the `Light` object.
`positionLocal`	Vector3	read-only	The position of the `Light` object relative to its parent `Node`.
`type`	string		The type of `Light` object being represented. One of the following values may be assigned: • "point" • "spot" • "infinite"
`TYPE_INFINITE`	string	read-only	Acrobat 7.0.7 A string constant for the `Light` type of "infinite".
`TYPE_POINT`	string	read-only	Acrobat 7.0.7 A string constant for the `Light` type of "point".
`TYPE_SPOT`	string	read-only	Acrobat 7.0.7 A string constant for the `Light` type of "spot".

Material

A `SceneObject` that controls the appearance of materials using the fixed function shader. In addition to the properties below, it also contains the same methods and properties as a `SceneObject`, documented on page 83.

Properties

Property	Type	Access	Description
`ambientColor`	Color	read-only	The ambient color.
`ambientTexture`	Texture	read-only	The ambient texture.
`bumpTexture`	Texture	read-only	A texture map whose value is used to describe the roughness of the object.
`diffuseColor`	Color	read-only	The matte color of an object.
`diffuseTexture`	Texture	read-only	A texture map that is used for the matte color of the object.
`emissiveColor`	Color	read-only	The emissive color.
`emissiveTexture`	Texture	read-only	The emissive texture.
`opacity`	number		The total opacity of the material.
`opacityTexture`	Texture	read-only	A texture map whose brightness is used for the level of opacity of the object. White signifies completely opaque while black signifies completely transparent.
`phongExponent`	number		The phong exponent.
`reflectionStrength`	Number		The reflection level, which may contain a value from `0.0` to `1.0`.
`reflectionTexture`	Texture	read-only	The reflection texture.
`specularColor`	Color	read-only	The specular color.
`specularStrength`	number		The specular strength, which is a measure of how shiny the material is.

Matrix4x4

A four-by-four matrix commonly used for transformations.

Properties

Property	Type	Access	Description
`determinant`	number		The determinant of the matrix.
`inverse`	Matrix4x4	read-only	The inverse of the matrix.
`scaleComponent`	Vector3	read-only	The scale component of the transformation.
`translation`	Vector3	read-only	The translation component of the transformation.
`transpose`	Matrix4x4	read-only	The transpose of the matrix.

Matrix4x4

Constructor

Syntax

```
new Matrix4x4()
```

Returns

A `Matrix4x4` object initialized to the identity matrix

Matrix4x4

Constructor

Syntax

```
new Matrix4x4(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used to initialize the new matrix.

Returns

A `Matrix4x4` object initialized to the specified matrix

invertInPlace

Inverts the matrix

Returns

undefined

isEqual

Determines whether the current matrix is equal to the specified matrix

Syntax

```
isEqual(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the comparison.

Returns

`True` if the matrices are equal, `false` otherwise.

multiply

Multiplies the current matrix by the specified matrix.

Syntax

```
multiply(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the multiplication.

Returns

A `Matrix4x4` object

multiplyInPlace

Multiplies the current matrix by the specified matrix, and updates the current matrix with the resulting value.

Syntax

```
multiplyInPlace(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object used for the multiplication.

Returns

undefined

rotateWithQuaternion

Rotates the current matrix using the specified `Quaternion`

Syntax

```
rotateWithQuaternion(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used for the rotation.

Returns

A `Matrix4x4` object

rotateWithQuaternionInPlace

Rotates the current matrix using the specified quaternion, and updates the current matrix with the resulting value.

Syntax

```
rotateWithQuaternionInPlace(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used for the rotation.

Returns

undefined

rotateAboutLine

Rotates the current matrix about the specified line.

Syntax

```
rotateAboutLine(angle, start, end)
```

Parameters

`angle`	The angle of rotation, in radians
`start`	A point described by a `Vector3` object used to specify the beginning of the line of rotation (which is represented by `start - end`).
`end`	A point described by a `Vector3` object used to specify the end of the line of rotation (which is represented by `start - end`).

Returns

A `Matrix4x4` object

rotateAboutLineInPlace

Rotates the current matrix about the specified line, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutLineInPlace(angle, start, end)
```

Parameters

`angle`	The angle of rotation, in radians
`start`	A `Vector3` object used to specify the line of rotation (which is represented by `start - end`).
`end`	A `Vector3` object used to specify the line of rotation (which is represented by `start - end`).

Returns

undefined

rotateAboutX

Rotates the current matrix about the x-axis.

Syntax

```
rotateAboutX(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object

rotateAboutXInPlace

Rotates the current matrix about the x-axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutXInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

rotateAboutVector

Rotates the current matrix about the specified vector.

Syntax

```
rotateAboutVector(angle, axis)
```

Parameters

`angle`	The angle of rotation, in radians.
`axis`	A `Vector3` object about which the matrix is rotated.

Returns

A `Matrix4x4` object

rotateAboutVectorInPlace

Rotates the current matrix about the specified vector, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutVectorInPlace(angle, axis)
```

Parameters

`angle`	The angle of rotation, in radians.
`axis`	A `Vector3` object about which the matrix is rotated.

Returns

undefined

rotateAboutY

Rotates the current matrix about the y-axis.

Syntax

```
rotateAboutY(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object

rotateAboutYInPlace

Rotates the current matrix about the y-axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutYInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

rotateAboutZ

Rotates the current matrix about the z-axis.

Syntax

```
rotateAboutZ(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

A `Matrix4x4` object

rotateAboutZInPlace

Rotates the current matrix about the z-axis, and updates the current matrix with the resulting value.

Syntax

```
rotateAboutZInPlace(angle)
```

Parameters

`angle`	The angle of rotation, in radians.

Returns

undefined

scale

Scales the current matrix using the specified scaling components.

Syntax

```
scale(x, y, z)
```

Parameters

`x`	The scaling component in the x-direction.
`y`	The scaling component in the y-direction.
`z`	The scaling component in the z-direction.

Returns

A `Matrix4x4` object

scaleInPlace

Scales the current matrix using the specified scaling components, and updates the current matrix with the resulting value.

Syntax

```
scaleInPlace(x, y, z)
```

Parameters

`x`	The scaling component in the x-direction.
`y`	The scaling component in the y-direction.
`z`	The scaling component in the z-direction.

Returns

undefined

set

Sets the value of the current matrix using the specified matrix.

Syntax

```
set(matrix)
```

Parameters

`matrix`	The matrix whose value is copied into the current matrix.

Returns

undefined

setIdentity

Sets the value of the current matrix to the identity matrix.

Syntax

```
setIdentity()
```

Returns

undefined

setView

Sets the current matrix according to the specified component vectors.

Syntax

```
setView(position, direction, up)
```

Parameters

`position`	A `Vector3` object used to specify the position component.
`direction`	A `Vector3` object used to specify the direction component.
`up`	A `Vector3` object used to specify the upward component.

Returns

undefined

transformDirection

Transforms the specified vector by the current matrix.

Syntax

```
transformDirection(vector)
```

Parameters

`vector`	The `Vector3` object to be transformed.

Returns

A `Vector3` object

transformPosition

Transforms the specified position by the current matrix.

Syntax

```
transformPosition(position)
```

Parameters

`position`	A `Vector3` object representing the position to be transformed.

Returns

A `Vector3` object

translate

Translates the current matrix by the components of the specified vector.

Syntax

```
translate(translation)
```

Parameters

`translation`	The `Vector3` object whose components are used to perform the matrix translation.

Returns

A `Matrix4x4` object

translateInPlace

Translates the current matrix by the components of the specified vector, and updates the current matrix with the resulting value.

Syntax

```
translateInPlace(translation)
```

Parameters

`translation`	The `Vector3` object whose components are used to perform the matrix translation.

Returns

undefined

transposeInPlace

Sets the value of the current matrix to its transpose.

Syntax

```
transposeInPlace()
```

Returns

undefined

MenuEvent

An object that is passed as an argument to the `onEvent` method of the `MenuEventHandler` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	read-only	The `Canvas` in which the `MenuEvent` took place.
`currentTool`	string	read-only	The name of the current tool.
`menuItemChecked`	Boolean	read-only	Determines whether the menu item was selected.
`menuItemName`	string	read-only	The name of the selected menu item.

MenuEventHandler

A `MenuEventHandler` object exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` `addEventHandler` method.

MenuEventHandler

Constructor

Syntax

```
new MenuEventHandler()
```

Returns

A `MenuEventHandler` object

onEvent

A method that is called when a custom menu item is selected on the context menu for an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `MenuEvent` object representing the event.

Returns

undefined

Mesh

A `Node` object that contains geometry. A `Mesh` object with no geometry will have children `Node` objects that may be transformed as a group. In addition to the methods and properties below, it also contains the same methods and properties as a `Node`, see page 57.

Properties

Property	Type	Description
`material`	Material	The `Mesh` object's default `Material`.
`renderMode`	string	The `Mesh` object's rendering style, which can be one of the following values: • "default" • "bounding box" • "transparent bounding box" • "transparent bounding box outline" • "vertices" • "shaded vertices" • "wireframe" • "shaded wireframe" • "solid" • "transparent" • "solid wireframe" • "transparent wireframe" • "illustration" • "solid outline" • "shaded illustration" • "hidden wireframe"

MouseEvent

An object that is passed as an argument to the `onEvent` method of the `MouseEventHandler` object, page 55.

Properties

Property	Type	Access	Description
`canvas`	Canvas	read-only	The `Canvas` in which the `MouseEvent` took place.
`canvasPixelHeight`	integer	read-only	The height, measured in pixels, of the `Canvas` in which the `MouseEvent` took place.
`canvasPixelWidth`	integer	read-only	The width, measured in pixels, of the `Canvas` in which the `MouseEvent` took place.
`ctrlKeyDown`	Boolean	read-only	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`currentTool`	string	read-only	The name of the current tool.
`hits`	Array	read-only	A set of `HitInfo` objects ordered by distance from nearest to furthest.
`isDoubleClick`	Boolean	read-only	Determines whether the a double-click event occurred
`isMouseDown`	Boolean	read-only	Determines whether the mouse button has been pressed
`isMouseHit`	Boolean	read-only	Determines whether the target is under the mouse cursor.
`isMouseMove`	Boolean	read-only	Determines whether the mouse position has changed.
`isMouseOut`	Boolean	read-only	Determines whether the mouse position has moved off the target.
`isMouseOver`	Boolean	read-only	Determines whether the mouse position has moved onto the target.
`isMouseUp`	Boolean	read-only	Determines whether the mouse button has been released.
`leftButtonDown`	Boolean	read-only	Determines whether the left mouse button has been pressed.
`mouseX`	integer	read-only	The x position of the mouse cursor in the `Canvas`.
`mouseY`	integer	read-only	The y position of the mouse cursor in the `Canvas`.

Property	Type	Access	Description
`rightButtonDown`	Boolean	read-only	Version 7.0.1 Determines whether the right mouse button has been pressed.
`shiftKeyDown`	Boolean	read-only	Determines whether the Shift key has been pressed.

MouseEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when mouse event occurs. The handler may be customized to filter out certain event types. Event handlers are registered with the Runtime `addEventHandler` method, see page 71.

Properties

Property	Type	Access	Description
`onMouseDoubleClick`	Boolean		When set to true the handler will be called back when a mouse button is clicked twice on the target object in rapid succession. If no target is specified the handler will call back on any double-click.
`onMouseDown`	Boolean		When set to true the handler will be called back when a mouse button is initially pressed while the cursor is over the target object. If no target is specified the handler will call back on any button press.
`onMouseHit`	Boolean		When set to true the handler will be called back continuously when the cursor is over the target object. In the case of onMouseHit it doesn't matter if the target object is behind another object in the scene. The list of resultant hit objects are provided in the `MouseEvent hits` property.
`onMouseMove`	Boolean		When set to true the handler will be called back when the cursor moves over the target object. If no target is specified the handler will call back on any mouse movement over the 3D annotation.
`onMouseOut`	Boolean		When set to true the handler will be called back once when the cursor moves off of the target object. In order to be called back the target must be the frontmost object. To exclude objects use `Node hitEnabled` property.
`onMouseOver`	Boolean		When set to true the handler will be called back once when the cursor moves over the target object.
`onMouseUp`	Boolean		When set to true the handler will be called back when a mouse button is initially released. If a target is specified it will only call back when the cursor is over the handler's target.

Property	Type	Access	Description
`reportAllTargets`	Boolean		Determines whether a hit test will be performed. When set to `false`, a hit test will not be performed except on a mouse-down or mouse-up event. This is an optimization feature because the current hit test is extremely expensive on complex models. When set to `false`, the following events will not be reported because they depend on hit testing: • `mouse-hit` • `mouse-move` • `mouse-out` • `mouse-over`
`target`	Object		The `Mesh` or `Background` object on which the mouse event occurs.

MouseEventHandler

Constructor

Syntax

```
new MouseEventHandler()
```

Returns

A `MouseEventHandler` object

onEvent

A method that is called when a mouse event occurs.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `MouseEvent` object representing the event.

Returns

undefined

Node

An object within the `Scene` hierarchy (a `SceneObject`) that has a 3D representation. The following objects are considered `Node` objects:

- `Bone`
- `Camera`
- `ClippingPlane`
- `Dummy`
- `Light`
- `Mesh`
- `Procedural`

To obtain a `Node` object's type, use the standard JavaScript `constructor` property. For example, the following syntax would print the `Node` object's type to the console:

```
console.println(myNode.constructor.name);
```

In addition to the methods and properties below, it also contains the same methods and properties as a `SceneObject`, see page 75.

Properties

Property	Type	Access	Description
`firstChild`	Node (if the first child exists), None otherwise	read-only	The `Node` object's first child.
`hitEnabled`	Boolean		Determines whether the `Node` is included in hit tests. The default value is `true`.
`info`	String	read-only	Acrobat 7.0.7 Information associated with the `Node` .
`metadataString`	String	read-only	Acrobat 8.1 A string containing Node-specific metadata.
`nextSibling`	Node (if the next sibling exists), None otherwise	read-only	The next sibling.
`parent`	Object	read-only	The `Node` object's parent `Node` or `Scene` .
`transform`	Matrix4x4	read-only	The local to world transformation matrix for the `Node` .
`wireframeColor`	Color	read-only	The `Color` object used to determine the wireframe appearance.
`visible`	Boolean		Determines whether the `Node` object should be shown.

computeBoundingBox

Acrobat 7.0.7

Computes the bounds of the `Node` object.

Syntax

```
computeBoundingBox()
```

Returns

A `BoundingBox` object.

detachFromCurrentAnimation

Removes the ability of the `Node` object's currently active `Animation` to transform the `Node`.

Syntax

```
detachFromCurrentAnimation()
```

Returns

undefined

Procedural

Deprecated

A `Node` object used to represent procedurally created geometry such as constructive solid geometry (CSG) solids, procedural spheres, or NURB objects (a 3D curve or surface). A `Procedural` object contains the same methods and properties as a `Node`, see page 57.

Quaternion

Represents a rotation in 3D space, and allows for smooth interpolation (blending) between orientations of subjects. A `Quaternion` is typically used for animating a `Camera` or `Mesh` over time, and can be converted to and from angles of rotation about the axes.

Quaternion

Constructor that initializes the object with the identity matrix.

Syntax

```
new Quaternion()
```

Returns

A `Quaternion` object

Quaternion

Constructor that initializes the object with the specified rotation matrix

Syntax

```
new Quaternion(matrix)
```

Parameters

`matrix`	A `Matrix4x4` object representing the rotation matrix.

Returns

A `Quaternion` object

Quaternion

Constructor that initializes the object with the specified `Quaternion`

Syntax

```
new Quaternion(quaternion)
```

Parameters

`quaternion`	A `Quaternion` object used to initialize the new object.

Returns

A `Quaternion` object

interpolate

Creates a `Quaternion` object interpolated from the current and specified `Quaternion` objects and `a`.

Syntax

```
interpolate(quaternion, a)
```

Parameters

`quaternion`	A `Quaternion` object used to interpolate the new object.
`a`	A number value, from `0.0` to `1.0`, that specifies the degree (percentage) of interpolation. A value of `0.5` represents an interpolation halfway between the current and specified `Quaternion` objects.

Returns

A `Quaternion` object

interpolateInPlace

Creates a `Quaternion` object interpolated from the current and specified `Quaternion` objects and `a`, and updates the current `Quaternion` object with the new value.

Syntax

```
interpolateInPlace(quaternion, a)
```

Parameters

`quaternion`	A `Quaternion` object used to interpolate the new object.
`a`	A number value, from `0.0` to `1.0`, that specifies the degree (percentage) of interpolation. A value of `0.5` represents an interpolation halfway between the current and specified `Quaternion` objects.

Returns

A `Quaternion` object

normalize

Normalizes the `Quaternion` object

Syntax

```
normalize()
```

Returns

undefined

RenderEvent

An object that is passed as an argument to the `RenderEventHandler` object's `onEvent` method.

Properties

Property	Type	Access	Description
`canvas`	Canvas	read-only	The `Canvas` that is the target of the `RenderEvent`.
`canvasPixelHeight`	integer	read-only	The height, measured in pixels, of the `Canvas` for which the `RenderEvent` is intended.
`canvasPixelWidth`	integer	read-only	The width, measured in pixels, of the `Canvas` for which the `RenderEvent` is intended.
`currentTool`	string	read-only	The name of the current tool.

RenderEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the Runtime `addEventHandler` method, page 71. It issues a callback just before each `Canvas` is rendered.

RenderEventHandler

Constructor

Syntax

```
new RenderEventHandler()
```

Returns

A `RenderEventHandler` object

onEvent

A method that is called immediately before the `Canvas` is rendered.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `RenderEvent` object representing the event

Returns

undefined

RenderOptions

An object that describes the style with which to render `Node` objects in the `Scene`.

Properties

Property	Type	Access	Description
`boundingBoxColor`	Color	read-only	A `Color` object to be applied to the bounding box.
`clippingPlaneColor`	Color	read-only	A `Color` object to be applied to the clipping plane.
`clippingPlaneIntersectionColor`	Color	read-only	A `Color` object to be applied to the clipping plane intersection.
`defaultAmbientColor`	Color	read-only	A `Color` object to be applied to the default ambient `Material`.
`defaultDiffuseColor`	Color	read-only	A `Color` object to be applied to the default diffuse `Material`.
`defaultEmissiveColor`	Color	read-only	A `Color` object to be applied to the default emissive `Material`.
`defaultSpecularColor`	Color	read-only	A `Color` object to be applied to the default specular `Material`.
`illustrationRenderModeFaceColor`	Color	read-only	Acrobat 7.0.7 The color of the faces when the render mode is Illustration.
`illustrationRenderModeLineColor`	Color	read-only	A `Color` object to be applied to the illustration lines.
`pointsRenderModeColor`	Color	read-only	A `Color` object to be applied to the vertices in point render mode.
`shadedIllustrationRenderModeLineColor`	Color	read-only	A `Color` object to be applied to the shaded illustration lines.

Property	Type	Access	Description
`solidGridColorEven`	Color	read-only	Acrobat 7.0.7 The color of the even squares of the checkered grid when drawn in "solid" mode.
`solidGridColorOdd`	Color	read-only	Acrobat 7.0.7 The color of the odd squares of the checkered grid when drawn in "solid" mode.
`solidRenderModeLineColor`	Color	read-only	A `Color` object to be applied to the solid or transparent lines in render mode.
`transparentBoundsRenderModeColor`	Color	read-only	A `Color` object to be applied to the transparent bounding box.
`transparentGridColorEven`	Color	read-only	Acrobat 7.0.7 The color of the even squares of the checkered grid when drawn in "transparent" mode.
`transparentGridColorOdd`	Color	read-only	Acrobat 7.0.7 The color of the odd squares of the checkered grid when drawn in "transparent" mode.
`wireframeRenderModeColor`	Color	read-only	Acrobat 7.0.7 The color of the wires when the render mode is Wireframe.
`xAxisColor`	Color	read-only	Acrobat 7.0.7 The color of the x-axis.
`yAxisColor`	Color	read-only	Acrobat 7.0.7 The color of the y-axis.
`zAxisColor`	Color	read-only	Acrobat 7.0.7 The color of the z-axis.

Resource

An object that creates an abstraction for loading behavior in files and streams.

Properties

Property	Type	Access	Description
`type`	string	read-only	The type of `Resource` object, which can be one of the following values: • "image" • "model" • "unknown"
`TYPE_IMAGE`	string	read-only	Acrobat 7.0.7 A string constant for the `Resource` type of "image".
`TYPE_MODEL`	string	read-only	Acrobat 7.0.7 A string constant for the `Resource` type of "model".
`TYPE_UNKNOWN`	string	read-only	Acrobat 7.0.7 A string constant for the `Resource` type of "unknown".

Resource

Constructor

Syntax

```
new Resource(pathname)
```

Parameters

`pathname`	A string representing the path of the file or stream. Can only load embedded resources from within the PDF file. The pathname string must start with pdf://

Returns

A `Resource` object.

Runtime

An object that represents the runtime instance of the player. Each `Runtime` object can have its own unique script engine and set of annotations. The variable `runtime` is a global reference to this object.

Properties

Property	Type	Access	Description
`BUTTON_TYPE_PUSH`	string	read-only	Acrobat 7.0.7 A string constant for the custom tool button type of "push button". It is used with the `addCustomToolButton` method.
`BUTTON_TYPE_TOOL`	string	read-only	Acrobat 7.0.7 A string constant for the custom button type of "tool button". It is used with the `addCustomToolButton` method.
`canvasCount`	Number	read-only	Acrobat 8.1 The number of Canvases that are attached to the active 3D annotation.
`ctrlKeyDown`	Boolean	read-only	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`eventHandlerCount`	integer	read-only	The number of registered event handlers.
`instances`	Array	read-only	Acrobat 7.0.7 An array of JavaScript `Annot3D` objects that are attached to the 3D script context.
`MENU_ITEM_TYPE_CHECKED`	string	read-only	Acrobat 7.0.7 A string constant for the custom menu item type of "checked". It is used with the `addCustomMenuItem` method.
`MENU_ITEM_TYPE_DEFAULT`	string	read-only	Acrobat 7.0.7 A string constant for the custom menu item type of "default". It is used with the `addCustomMenuItem` method.
`overrideNavTools`	Boolean		Determines whether to disable all default navigation behavior. **Note:** Setting this property does not prevent view changes.

Property	Type	Access	Description
`overridePanTool`	Boolean		Determines whether to override the built-in Pan tool behavior. **Note:** Setting this property does not effect the pan behavior of other navigation tools.
`overrideRotateTool`	Boolean		Determines whether to override the built-in Rotate tool behavior.
`overrideSelection`	Boolean		Acrobat 7.0.7 Determines whether to override the built-in Selection tool behavior.
`overrideSpinTool`	Boolean		Acrobat 8.0 Determines whether to override the built-in Spin tool behavior.
`overrideViewChange`	Boolean		Determines whether to override the setting of Views from Acrobat.
`overrideWalkTool`	Boolean		Determines whether to override the built-in Walk tool behavior.
`overrideWheelSpeed`	Number		Acrobat 8.1 A speed multiplier for the value of the scroll-wheel motion.
`overrideZoomTool`	Boolean		Determines whether to override the built-in Zoom tool behavior. **Note:** Setting this property does not effect the zoom behavior of other navigation tools.
`scrollWheelSpeed`	Number		Acrobat 8.1 A speed multiplier for the value of the scroll-wheel motion.
`shiftKeyDown`	Boolean	read-only	Determines whether the Shift key was pressed.

Property	Type	Access	Description
`speedThreshold`	Number		Acrobat 8.1 A length (based upon the diagonal of the scene's bounding box) under which the Walk tool's motion is scaled relative to the size of the model. The Walk tool's motion is constant based upon the scene's scale factor, such that it emulates a natural pace relative to the model's size. This works well for architectural models that are created with a defined scale; however, the walk motion will be too quick for very small models.
`strafeSpeed`	Number		Acrobat 8.1 A speed multipier for the lateral motion while using the Walk tool.
`TOOL_NAME_MEASURE`	string	read-only	Acrobat 7.0.7 A string constant for the name of the measure tool. Its value is "`Measure`".
`TOOL_NAME_PAN`	string	read-only	Acrobat 7.0.7 A string constant for the name of the pan tool. Its value is "`Pan`".
`TOOL_NAME_ROTATE`	string	read-only	Acrobat 7.0.7 A string constant for the name of the rotate tool. Its value is "`Rotate`".
`TOOL_NAME_SPIN`	string	read-only	Acrobat 8.0 A string constant for the name of the Spin tool. Its value is "`Spin`".
`TOOL_NAME_WALK`	string	read-only	Acrobat 7.0.7 A string constant for the name of the walk tool. Its value is "`Walk`".
`TOOL_NAME_ZOOM`	string	read-only	Acrobat 7.0.7 A string constant for the name of the zoom tool. Its value is "`Zoom`".
`version`	Number	read-only	The number corresponding to the version of the `Runtime` system.
`walkSpeed`	Number		Acrobat 8.1 A speed multipier for the forward/backward motion while using the Walk tool.

addCustomMenuItem

Creates a custom menu item in the 3D annotation context menu.

Syntax

```
addCustomMenuItem(name, label, type, checkedState)
```

Parameters

`name`	A string identifying the menu item.
`label`	A string appearing on the menu item.
`type`	A string indicating whether it is a checked menu item. A checked menu item has a check mark toggle next to it. Its possible values are: • "default" • "checked"
`checkedState`	A Boolean value indicating the state of a checked menu item.

Returns

undefined

addCustomToolButton

Creates a custom tool button in the 3D toolbar.

Syntax

```
addCustomToolButton(name, label, type)
```

Parameters

`name`	A string identifying the tool button.
`label`	A string appearing on the tool button.
`type`	A string indicating whether it is a tool button or a push button. Its possible values are: • "tool button" • "push button"

Returns

undefined

addEventHandler

Registers the provided event handler.

Syntax

```
addEventHandler(eventHandler)
```

Parameters

`eventHandler`	The event handler object to be registered.

Returns

undefined

disableTool

Disables the tool with the specified ID.

Syntax

```
disableTool(toolName)
```

Parameters

`toolName`	A string identifying the tool.

Returns

undefined

enableTool

Enables the tool with the specified ID.

Syntax

```
enableTool(toolName)
```

Parameters

`toolName`	A string identifying the tool.

Returns

undefined

getEventHandler

Obtains the event handler corresponding to the specified index.

Syntax

```
getEventHandler(index)
```

Parameters

`index`	An integer identifying the event handler.

Returns

An event handler object

getRendererName

Obtains the name of the current renderer.

Syntax

```
getRendererName()
```

Returns

A string containing the current renderer's name

refresh

Version 7.0.1

Marks the render area dirty so that it will be redrawn. This is useful when something has changed in the scene but the annotation is in a "Loaded" and not "Live" state.

Syntax

```
refresh()
```

Returns

undefined

removeEventHandler

Unregisters the specified event handler.

Syntax

```
removeEventHandler(handler)
```

Parameters

`handler`	An event handler object representing the event handler.

Returns

undefined

removeCustomMenuItem

Removes the custom menu item with the specified ID.

Syntax

```
removeCustomMenuItem(menuName)
```

Parameters

`menuName`	A string identifying the custom menu item.

Returns

undefined

removeCustomToolButton

Removes the custom tool button with the specified ID.

Syntax

```
removeCustomToolButton(toolName)
```

Parameters

`toolName`	A string identifying the custom tool button.

Returns

undefined

setCurrentTool

Sets the current tool to the one with the specified ID.

Syntax

```
setCurrentTool(toolName)
```

Parameters

`toolName`	A string identifying the tool

Returns

undefined

Scene

An object that represents the hierarchy of the 3D related content, including `Animation`, `Light`, `Material`, and `Mesh` objects. The variable `scene` is a global reference to this object.

Related objects are `Animation` on page 15, `Light` on page 37, `Material` on page 39 and `Mesh` on page 52.

Properties

Property	Type	Access	Description
`ambientIlluminationColor`	Color	read-only	Modulates the ambient `Color` of all materials.
`animations`	SceneObjectList	read-only	A list of all `Animation` objects.
`cameras`	SceneObjectList	read-only	A list of all `Camera` objects in the `Scene`.
`defaultRenderOptions`	RenderOptions	read-only	A set of all default rendering options for the `Scene`.
`gridMode`	string		Acrobat 7.0.7 The display style of the grid that represents a portion of the ground plane in the `Scene`. It can have one of the following values: • "off" — no grid • "wire" — a wireframe grid • "solid" — a solid checkerboard grid • "transparent" — a semi-transparent checkerboard grid
`GRID_MODE_OFF`	string	read-only	Acrobat 7.0.7 A string constant for the grid mode of "off".
`GRID_MODE_SOLID`	string	read-only	Acrobat 7.0.7 A string constant for the grid mode of "solid".
`GRID_MODE_TRANSPARENT`	string	read-only	Acrobat 7.0.7 A string constant for the grid mode of "transparent".

Property	Type	Access	Description
GRID_MODE_WIRE	string	read-only	Acrobat 7.0.7 A string constant for the grid mode of "wire".
gridSize	number	read-only	Acrobat 7.0.7 The number of squares on the ground plane grid.
lengthUnits	number	read-only	The scale of a unit of length, specified in meters.
LIGHT_MODE_FILE	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "file".
LIGHT_MODE_NONE	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "none".
LIGHT_MODE_WHITE	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "white".
LIGHT_MODE_DAY	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "day".
LIGHT_MODE_BRIGHT	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "bright".
LIGHT_MODE_RGB	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "rgb".
LIGHT_MODE_NIGHT	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "night".
LIGHT_MODE_BLUE	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "blue".
LIGHT_MODE_RED	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "red".
LIGHT_MODE_CUBE	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "cube".

Property	Type	Access	Description
`LIGHT_MODE_CAD`	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "cad".
`LIGHT_MODE_HEADLAMP`	string	read-only	Acrobat 7.0.7 A string constant for the light mode of "headlamp".
`lights`	SceneObjectList	read-only	A list of all `Light` objects in the `Scene`.
`lightScaleFactor`	number		A uniform scale factor for all `Light` objects in the `Scene`.
`lightScheme`	string		Acrobat 7.0.7 The current, preconfigured lighting scheme for the `Scene`. It can take one of the following values: • "file" • "none" • "white" • "day" • "bright" • "rgb" • "night" • "blue" • "red" • "cube" • "cad" • "headlamp"
`materials`	SceneObjectList	read-only	A list of all `Material` objects.
`meshes`	SceneObjectList	read-only	A list of all `Mesh` objects in the `Scene`.
`nodes`	SceneObjectList	read-only	A list of all `Node` objects except the default `Camera` and default `Light` objects.

Property	Type	Access	Description
`outlineAngle`	number		Acrobat 7.0.7 The crease angle (in degrees) for the appearance of lines in Illustration render modes.
`showGrid`	Boolean		Acrobat 7.0.7 Determines whether the ground plane grid is displayed.
`renderDoubleSided`	Boolean		Acrobat 8.1 Toggles if backfacing polygons are rendered.
`renderMode`	string		Acrobat 7.0.7 The default rendering type for all objects in the `Scene`, which can be one of the following values: • "default" • "bounding box" • "transparent bounding box" • "transparent bounding box outline" • "vertices" • "shaded vertices" • "wireframe" • "shaded wireframe" • "solid" • "transparent" • "solid wireframe" • "transparent wireframe" • "illustration" • "solid outline" • "shaded illustration" • "hidden wireframe"
`RENDER_MODE_DEFAULT`	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "default".

Property	Type	Access	Description
RENDER_MODE_BOUNDING_BOX	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "bounding box".
RENDER_MODE_TRANSPARENT_BOUNDING_BOX	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "transparent bounding box".
RENDER_MODE_TRANSPARENT_BOUNDING_BOX_OUTLINE	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "transparent bounding box outline".
RENDER_MODE_VERTICES	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "vertices".
RENDER_MODE_SHADED_VERTICES	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "shaded vertices".
RENDER_MODE_WIREFRAME	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "wireframe".
RENDER_MODE_SHADED_WIREFRAME	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "shaded wireframe".
RENDER_MODE_SOLID	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "solid".
RENDER_MODE_TRANSPARENT	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "transparent".
RENDER_MODE_SHADED_SOLID_WIREFRAME	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "solid wireframe".

Property	Type	Access	Description
RENDER_MODE_TRANSPARENT_WIREFRAME	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "transparent wireframe".
RENDER_MODE_ILLUSTRATION	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "illustration".
RENDER_MODE_SOLID_OUTLINE	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "solid outline".
RENDER_MODE_SHADED_ILLUSTRATION	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "shaded illustration".
RENDER_MODE_HIDDEN_WIREFRAME	string	read-only	Acrobat 7.0.7 A string constant for the render mode of "hidden wireframe".
selectedNode	Node		Acrobat 8.1 The currently selected Node.
showAxes	Boolean		Acrobat 7.0.7 Determines whether the world axes are displayed.
smoothing	Boolean		Acrobat 7.0.7 When `true`, smoothing is enabled for meshes in the scene.
smoothingAngle	number		Acrobat 7.0.7 The default smoothing angle (in degrees) for meshes in the scene.
smoothingOverride	Boolean		Acrobat 7.0.7 When set to `true`, overrides the smoothing values from the loaded model file.

activateAnimation

Sets the given `Animation` to be active on its `Node` objects.

Syntax

```
activateAnimation(animation)
```

Parameters

`animation`	The `Animation` object to be activated.

Returns

undefined

addModel

Adds a model `Resource` to the top level of the `Scene`.

Syntax

```
addModel(modelRes)
```

Parameters

`modelRes`	The `Resource` object to be added.

Returns

A `Node` object representing the top-level `Mesh` of the loaded model

createClippingPlane

Creates a new clipping plane

Syntax

```
createClippingPlane()
```

Returns

A `ClippingPlane` object

createLight

Creates a new `Light` and attaches it to the `Scene`

Syntax

```
createLight()
```

Returns

A `Light` object

createSquareMesh

Creates a `Mesh` that is a unit square. The default UV parameterization fits once in U and V.

Syntax

```
createSquareMesh(sizeX, sizeY, name)
```

Parameters

`sizeX`	Model units in the x-direction used to size the `Mesh`.
`sizeY`	Model units in the y-direction used to size the `Mesh`.
`name`	(Optional) The name that will be assigned to the `Mesh`.

Returns

A `Mesh` object

computeBoundingBox

Computes the `BoundingBox` of the `Scene`

Syntax

```
computeBoundingBox()
```

Returns

A `BoundingBox` object

update

Applies all changes to the `Scene`

Syntax

```
update()
```

Returns

undefined

SceneObject

The base type for objects within the `Scene`, including `Animation`, `Material`, and `Node` objects.

Related objects are `Scene` on page 75, `Animation` on page 15, `Light` on page 37, `Material` on page 39 and `Mesh` on page 52.

Properties

Property	Type	Description
`name`	string	The `SceneObject` object's name.
`objectGUID`	string	Deprecated A value that uniquely identifies the `SceneObject` in custom data. This property has a default value.
`objectID`	integer	An unsigned 32-bit value that uniquely identifies the `SceneObject`. This property can be assigned, but does not have a default value (it always returns `0`).

SceneObjectList

A structure that contains references to several `SceneObject` objects.

Properties

Property	Type	Access	Description
`count`	integer	read-only	The number of elements in the `SceneObjectList`.

getByGUID

Deprecated

Obtains the specified `SceneObject` object from the list

Syntax

```
getByGUID(guid)
```

Parameters

`guid`	A string representing the GUID for the specified element.

Returns

A `SceneObject` object

getByID

Obtains the specified `SceneObject` object from the list

Syntax

```
getByID(id)
```

Parameters

`id`	An integer representing the object identifier for the specified `SceneObject` object.

Returns

A `SceneObject` object

getByIndex

Obtains the specified `SceneObject` object from the list

Syntax

```
getByIndex(index)
```

Parameters

index	An integer representing the index of the specified SceneObject object.

Returns

A SceneObject object

getByName

Obtains the specified SceneObject object from the list

Syntax

```
getByName(name)
```

Parameters

name	A string representing the name of the specified SceneObject object.

Returns

A SceneObject object

removeAll

Removes all the SceneObject objects from the list

Syntax

```
removeAll()
```

Returns

undefined

removeByIndex

Removes the specified SceneObject object from the list

Syntax

```
removeByIndex(index)
```

Parameters

`index`	An index to the specified element.

Returns

undefined

removeItem

Removes a `SceneObject` object from the list

Syntax

```
removeItem(scene_object)
```

Parameters

`scene_object`	A scene object.

Returns

undefined

ScrollWheelEvent

(Acrobat 8.1) An object that is passed as an argument to the `onEvent` method of the `ScrollWheelHandler` object, page 88. A `ScrollWheelEvent` object is created when the mouse scroll wheel is activated over an active 3D `Canvas` object.

Properties

Property	Type	Access	Description
`canvas`	Canvas	read-only	The `Canvas` in which the `ScrollWheelEvent` took place.
`canvasPixelHeight`	integer	read-only	The height, measured in pixels, of the `Canvas` in which the `ScrollWheelEvent` took place.
`canvasPixelWidth`	integer	read-only	The width, measured in pixels, of the `Canvas` in which the `ScrollWheelEvent` took place.
`ctrlKeyDown`	Boolean	read-only	Determines whether the Ctrl key (Windows) or Command key (Mac OS) was pressed.
`currentTool`	string	read-only	The name of the current tool.
`deltaY`	Number	read-only	The amount the scroll-wheel has been moved in the Y direction.
`shiftKeyDown`	Boolean	read-only	Determines whether the Shift key has been pressed.

ScrollWheelEventHandler

(Acrobat 8.1) An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` method `addEventHandler` on page 71.

ScrollWheelEventHandler

Constructor

Syntax

```
new ScrollWheelEventHandler()
```

Returns

A `ScrollWheelEventHandler` object.

onEvent

A method that is called when the scroll wheel is used in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ScrollWheelEvent` object representing the event.

Returns

undefined

SelectionEvent

(Acrobat 8.1) An object that is passed as an argument to the `onEvent` method of the `SelectionEventHandler` object, page 90.

A `SelectionEvent` object is created when an object is selected from an active 3D `Canvas` object or from a model tree. If the selection is made from a `Canvas` object, a `MouseEvent` is also created.

Properties

Property	Type	Access	Description
`node`	Node	read-only	The Node that is the target of the selection change.
`selected`	Boolean	read-only	The selected state of the target Node.

SelectionEventHandler

(Acrobat 8.1) An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime` `addEventHandler` method.

SelectionEventHandler

Constructor

Syntax

```
new SelectionEventHandler()
```

Returns

A `SelectionEventHandler` object.

onEvent

A method that is called when the selection state changes in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ScrollWheelEvent` object representing the event.

Returns

undefined

Texture

A `Texture` object represents the mapping of a texture. All `Texture` properties have read-write permissions.

Properties

Property	Type	Description
`amount`	number	The degree to which the `Texture` is applied, which can be a value from `0.0` to `1.0`.
`angle`	number	The rotation angle, measured in degrees, of the map.
`clampU`	Boolean	Determines whether the map should be clamped in the U direction.
`clampV`	Boolean	Determines whether the map should be clamped in the V direction.
`image`	Image	Acrobat 7.0.7 The `Image` to be used by the `Texture`.
`modulate`	Boolean	Determines whether to set the blend mode of the `Texture` with its corresponding color.
`uOffset`	number	The U-offset of the given map.
`uScale`	number	The U-scale of the given map.
`use3DSStyleMapping`	Boolean	Determines whether to use 3D Studio style map parameterizations.
`vOffset`	number	The V-offset of the given map.
`vScale`	number	The V-scale of the given map.

getImage

Deprecated

Gets the `Image` currently used by the `Texture`.

Syntax

```
getImage()
```

Returns

The Image currently being used.

setImage

Deprecated

Sets the `Image` to be used by the `Texture`.

Syntax

```
setImage(image)
```

Parameters

`image`	The `Image` to be used.

Returns

undefined

TimeEvent

An object that is passed as an argument to the `TimeEventHandler` object's `onEvent` method.

Properties

Property	Type	Access	Description
`deltaTime`	number	read-only	The difference between the current time and the last time.
`time`	number	read-only	The current time.

TimeEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime addEventHandler` method.

TimeEventHandler

Constructor

Syntax

```
new TimeEventHandler()
```

Returns

A `TimeEventHandler` object.

onEvent

A method that is called when simulation time is incremented in an active 3D annotation.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `TimeEvent` object representing the event.

Returns

undefined

ToolEvent

An object that is passed as an argument to the `onEvent` method of the `ToolEventHandler` object, see page 96.

Properties

`Property`	**Type**	**Access**	**Description**
`canvas`	Canvas	read-only	The `Canvas` that is the target of the `ToolEvent`.
`canvasPixelHeight`	integer	read-only	The height, measured in pixels, of the `Canvas` for which the `ToolEvent` is intended.
`canvasPixelWidth`	integer	read-only	The width, measured in pixels, of the `Canvas` for which the `ToolEvent` is intended.
`currentTool`	string	read-only	The name of the current tool.
`toolName`	string	read-only	The name of the tool that was clicked.

ToolEventHandler

An object that exposes a callback mechanism that allows a function to be evaluated when an event occurs. Event handlers are registered with the `Runtime addEventHandler` method, page 71.

ToolEventHandler

Constructor

Syntax

```
new ToolEventHandler()
```

Returns

A `ToolEventHandler` object

onEvent

A method that is called when a tool button is pressed on the 3D toolbar.

Syntax

```
onEvent(event)
```

Parameters

`event`	A `ToolEvent` object representing the event.

Returns

undefined

Vector3

An object comprised of three values that represent a point in space or a direction and magnitude.

Properties

Property	Type	Access	Description
`x`	number		The x-component of the `Vector3` object.
`y`	number		The y-component of the `Vector3` object.
`z`	number		The z-component of the `Vector3` object.
`length`	number	read-only	The length of the `Vector3` object.

Vector3

Constructor that initializes the new object to `(0.0, 0.0, 0.0)`.

Syntax

```
new Vector3()
```

Returns

A `Vector3` object

Vector3

Constructor that initializes the new object to the specified components

Syntax

```
new Vector3(x, y, z)
```

Parameters

`x`	The x-component used to initialize the new object.
`y`	The y-component used to initialize the new object.
`z`	The z-component used to initialize the new object.

Returns

A `Vector3` object

add

Adds the specified `Vector3` to the current one.

Syntax

```
add(offset)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.

Returns

A `Vector3` object

addInPlace

Adds the specified `Vector3` to the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
addInPlace(offset)
```

Parameters

`offset`	The `Vector3` object to be added to the current one

Returns

undefined

addScaled

Adds the specified `Vector3` with the scaled offset to the current one.

Syntax

```
addScaled(offset, scale)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.
`scale`	The scaling factor for the `offset`.

Returns

A `Vector3` object

addScaledInPlace

Adds the specified `Vector3` with the scaled offset to the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
addScaledInPlace(offset, scale)
```

Parameters

`offset`	The `Vector3` object to be added to the current one.
`scale`	The scaling factor for the `offset`.

Returns

undefined

blend

Blends the current and specified `Vector3` by the specified degree.

Syntax

```
blend(vec, blendFactor)
```

Parameters

`vec`	The `Vector3` object to be blended with the current one.
`blendFactor`	The degree of blending to be applied, which may be a value from `0.0` to `1.0`.

Returns

A `Vector3` object.

blendInPlace

Blends the current and specified `Vector3` by the specified degree, and updates the current `Vector3` with the resulting value.

Syntax

```
blendInPlace(vec, blendFactor)
```

Parameters

`vec`	The `Vector3` object to be blended with the current one
`blendFactor`	The degree of blending to be applied, which may be a value from `0.0` to `1.0`

Returns

undefined

cross

Calculates the cross product between the specified `Vector3` and the current one.

Syntax

```
cross(vec)
```

Parameters

`vec`	The `Vector3` object to be used in calculating the cross product.

Returns

A `Vector3` object.

dot

Calculates the dot product between the specified `Vector3` and the current one.

Syntax

```
dot(vec)
```

Parameters

`vec`	The `Vector3` object to be used in calculating the dot product

Returns

A number value representing the scalar dot product.

normalize

Normalizes the current `Vector3`.

Syntax

```
normalize()
```

Returns

undefined

scale

Scales the current `Vector3`.

Syntax

```
scale(scale)
```

Parameters

`scale`	A number value used to scale the current `Vector3`.

Returns

A `Vector3` object

scaleInPlace

Scales the current `Vector3`, and updates the current `Vector3` with the resulting value.

Syntax

```
scaleInPlace(scale)
```

Parameters

`scale`	A number value used to scale the current `Vector3`.

Returns

undefined

set

Sets the current `Vector3` to the value contained in the specified `Vector3`.

Syntax

```
set(vec)
```

Parameters

`vec`	The Vector3 used to set the current `Vector3`.

Returns

undefined

set

Acrobat 7.0.7

Sets the current `Vector3` to the values contained in the specified components.

Syntax

```
set(x, y, z)
```

Parameters

`x`	The x-component used to set the current `Vector3`.
`y`	The y-component used to set the current `Vector3`.
`z`	The z-component used to set the current `Vector3`.

Returns

undefined

set3

Deprecated

Sets the current `Vector3` to the values contained in the specified components.

Syntax

```
set3(x, y, z)
```

Parameters

`x`	The x-component used to set the current `Vector3`.
`y`	The y-component used to set the current `Vector3`.
`z`	The z-component used to set the current `Vector3`.

Returns

undefined

subtract

Subtracts the specified `Vector3` from the current one.

Syntax

```
subtract(offset)
```

Parameters

`offset`	The `Vector3` object to be subtracted from the current one.

Returns

A `Vector3` object

subtractInPlace

Subtracts the specified `Vector3` from the current one, and updates the current `Vector3` with the resulting value.

Syntax

```
subtractInPlace(offset)
```

Parameters

`offset`	The `Vector3` object to be subtracted from the current one.

Returns

undefined

3 New Features and Changes

This chapter summarizes the new features and changes introduced in Acrobat 8.1 and earlier.

Acrobat 8.1 changes

This section describes the changes introduced in Acrobat 8.1.

New objects

The following objects are new: `ScrollWheelEvent`, `ScrollWheelEventHandler`, `SelectionEvent`, and `ScrollWheelEventHandler`.

Additional properties in existing objects

The `HitInfo` object has additional properties: `material`, `surfaceNormal`, and `textureCoordinate`.

The `Node` object has an additional property: `metadataString`.

The `Light` object has an additional property: `directionLocal` (Acrobat 7, but previously undocumented).

The `Runtime` object has the additional properties: `canvasCount`, `overrideSpinTool`, `scrollWheelSpeed`, `speedThreshold`, `strafeSpeed`, and `walkSpeed`.

The `Scene` object has the additional properties: `node` and `selected`.

Deprecated objects or properties

The following APIs have been deprecated:

`CameraEvent.isNewCanvas` (a property)

`Dummy` (an object)

`Procedural` (an object)

`SceneObject.objectGUID` (a property)

`SceneObject.getByGUID` (a method)

Acrobat 8.0 changes

This section describes the changes introduced in Acrobat 8.0.

Additional properties in existing objects

The Runtime object has the additional properties: `overrideSpinTool` and `TOOL_NAME_SPIN`.

Index

3

3D JavaScript engine 9

A

Accessing the 3D JavaScript engine 9
activateAnimation method 81
add method 98
addCustomMenuItem method 70
addCustomToolButton method 70
addEventHandler method 71
addInPlace method 98
addModel method 81
addScaled method 98
addScaledInPlace method 99
ambientColor property 39
ambientIlluminationColor property 75
ambientTexture property 39
angle property 91
Animation object 15
    currentTime property 15
    endTime property 15
    framesPerSecond property 15
    length property 15
    startTime property 15
animations property 75
Annot3D.context3D property 9
ATTENUATION_ABC property 37
ATTENUATION_NONE property 37
attenuationA property 37
attenuationB property 37
attenuationC property 37
attenuationType property 37

B

Background object 16
    getColor method 16
    getImage method 16
    image property 16
    setColor method 16
    setImage method 17
background property 25
binding property 20, 23
BINDING_HORIZONTAL property 20
BINDING_MAX property 20
BINDING_MIN property 20
BINDING_VERTICAL property 20
blend method 99
blendInPlace method 99
Bone object 18
BoundingBox object 19
    center property 19
    max property 19
    min property 19
boundingBoxColor property 64
brightness property 37
bumpTexture property 39
BUTTON_TYPE_PUSH property 67
BUTTON_TYPE_TOOL property 67

C

Camera object
    binding property 20
    BINDING_HORIZONTAL property 20
    BINDING_MAX property 20
    BINDING_MIN property 20
    BINDING_VERTICAL property 20
    far property 20
    fov property 20
    getDirectionFromScreen method 22
    getScreenFromPosition method 21
    near property 20
    position property 20
    positionLocal property 20
    projectionType property 20
    roll property 21
    target property 21
    targetPosition property 21
    targetPositionLocal property 21
    TYPE_ORTHOGRAPHIC property 21
    TYPE_PERSPECTIVE property 21
    up property 21
    upLocal property 21
    viewPlaneSize property 21
CameraEvent object 23
    binding property 23
    canvas property 23
    currentTool property 23
    far property 23
    fov property 23
    isNewCanvas property 23
    near property 23
    projectionType property 23
    targetDistance property 23
    transform property 23
    viewPlaneSize property 23
CameraEventHandler method 24
CameraEventHandler object 24
    CameraEventHandler method 24
cameras property 75

Canvas object 25
    background property 25
    getCamera method 25
    setCamera method 25
canvas property 23, 34, 50, 53, 62, 87, 95
canvasCount property 67
canvasPixelHeight property 34, 53, 62, 87, 95
canvasPixelWidth property 34, 53, 62, 87, 95
center property 19
characterCode property 34
clampU property 91
clampV property 91
ClippingPlane object 26
    remove method 26
clippingPlaneColor property 64
clippingPlaneIntersectionColor property 64
Color method 27
Color object 27
    Color method 27
    r, g, b properties 27
    set method 27
    set3 method 28
color property 37
computeBoundingBox method 58, 82
Console object 29
    print method 29
    println method 29
count property 84
createClippingPlane method 81
createLight method 81
createSquareMesh method 82
cross method 100
ctrlKeyDown property 35, 53, 67, 87
currentTime property 15
currentTool property 23, 35, 50, 53, 62, 87, 95

D

Date object 13
defaultAmbientColor property 64
defaultDiffuseColor property 64
defaultEmissiveColor property 64
defaultRenderOptions property 75
defaultSpecularColor property 64
deltaTime property 93
deltaY property 87
detachFromCurrentAnimation method 58
determinant property 40
diffuseColor property 39
diffuseTexture property 39
direction property 37
directionLocal property 37
disableTool method 71
distance property 31
dot method 100
Dummy object 30

E

emissiveColor property 39
emissiveTexture property 39
enableTool method 71
endTime property 15
eventHandlerCount property 67

F

far property 20, 23
firstChild property 57
fov property 20, 23
framesPerSecond property 15

G

getByGUID method 84
getByID method 84
getByIndex method 84
getByName method 85
getCamera method 25
getColor method 16
getDirectionFromScreen method 22
getEventHandler method 72
getImage method 16, 91
getRendererName method 72
getScreenFromPosition method 21
Getting the SceneContext3d object 9
GRID_MODE_OFF property 75
GRID_MODE_SOLID property 75
GRID_MODE_TRANSPARENT property 75
GRID_MODE_WIRE property 76
gridMode property 75
gridSize property 76

H

height property 33
hitEnabled property 57
HitInfo object 31
    distance property 31
    material property 31
    position property 31
    surfaceNormal property 31
    target property 31
    textureCoordinate property 31
hits property 53
Host object 32

I

illustrationRenderModeFaceColor property 64
illustrationRenderModeLineColor property 64
Image method 33
Image object 33
    height property 33
    Image method 33
    width property 33
image property 16, 91
info property 57
innerConeAngle property 38
innerRadius property 38
instances property 67
interpolate method 61
interpolateInPlace method 61

inverse property 40
invertInPlace method 40
isDoubleClick property 53
isEqual method 41
isMouseDown property 53
isMouseHit property 53
isMouseMove property 53
isMouseOut property 53
isMouseOver property 53
isMouseUp property 53
isNewCanvas property 23

K

KeyEvent object
    canvas property 34
    canvasPixelHeight property 34
    canvasPixelWidth property 34
    characterCode property 34
    ctrlKeyDown property 35
    currentTool property 35
    shiftKeyDown property 35
KeyEventHandler method 36
KeyEventHandler object
    KeyEventHandler method 36
    onEvent method 36

L

leftButtonDown property 53
length property 15, 97
lengthUnits property 76
Light object 37
    ATTENUATION_ABC property 37
    ATTENUATION_NONE property 37
    attenuationA property 37
    attenuationB property 37
    attenuationC property 37
    attenuationType property 37
    brightness property 37
    color property 37
    direction property 37
    innerConeAngle property 38
    innerRadius property 38
    outerConeAngle property 38
    outerRadius property 38
    position property 38
    positionLocal property 38
    type property 38
    TYPE_INFINITE property 38
    TYPE_POINT property 38
    TYPE_SPOT property 38
LIGHT_MODE_BLUE property 76
LIGHT_MODE_BRIGHT property 76
LIGHT_MODE_CAD property 77
LIGHT_MODE_CUBE property 76
LIGHT_MODE_DAY property 76
LIGHT_MODE_FILE property 76
LIGHT_MODE_HEADLAMP property 77
LIGHT_MODE_NIGHT property 76
LIGHT_MODE_NONE property 76
LIGHT_MODE_RED property 76
LIGHT_MODE_RGB property 76
LIGHT_MODE_WHITE property 76
lights property 77
lightScaleFactor property 77
lightScheme property 77

M

Material object 39
    ambientColor property 39
    ambientTexture property 39
    bumpTexture property 39
    diffuseColor property 39
    diffuseTexture property 39
    emissiveColor property 39
    emissiveTexture property 39
    opacity property 39
    opacityTexture property 39
    phongExponent property 39
    reflectionStrength property 39
    reflectionTexture property 39
    specularColor property 39
    specularStrength property 39
material property 31, 52
materials property 77
Matrix4x4 method 40
Matrix4x4 object
    determinant property 40
    inverse property 40
    invertInPlace method 40
    isEqual method 41
    Matrix4x4 method 40
    multiply method 41
    multiplyInPlace method 41
    rotateAboutLine method 42
    rotateAboutLineInPlace method 43
    rotateAboutVector method 44
    rotateAboutVectorInPlace method 44
    rotateAboutX method 43
    rotateAboutXInPlace method 44
    rotateAboutY method 45
    rotateAboutYInPlace method 45
    rotateAboutZ method 45
    rotateAboutZInPlace method 46
    rotateWithQuaternion method 42
    rotateWithQuaternionInPlace method 42
    scale method 46
    scaleComponent property 40
    scaleInPlace method 46
    set method 47
    setIdentity method 47
    setView method 47
    transformDirection method 48
    transformPosition method 48
    translate method 49
    translateInPlace method 49
    translation property 40
    transpose property 40
    transposeInPlace method 49

max property 19
MENU_ITEM_TYPE_CHECKED property 67
MENU_ITEM_TYPE_DEFAULT property 67
MenuEvent object 50
    canvas property 50
    currentTool property 50
    menuItemChecked property 50
    menuItemName property 50
MenuEventHandler method 51
MenuEventHandler object
    MenuEventHandler method 51
    onEvent method 51
menuItemChecked property 50
menuItemName property 50
Mesh object 52
    material property 52
    renderMode property 52
meshes property 77
metadataString property 57
min property 19
modulate property 91
MouseEvent object 53
    canvas property 53, 87
    canvasPixelHeight property 53, 87
    canvasPixelWidth property 53, 87
    ctrlKeyDown property 53, 87
    currentTool property 53, 87
    hits property 53
    isDoubleClick property 53
    isMouseDown property 53
    isMouseHit property 53
    isMouseMove property 53
    isMouseOut property 53
    isMouseOver property 53
    isMouseUp property 53
    leftButtonDown property 53
    mouseX property 53
    mouseY property 53
    rightButtonDown property 54
    shiftKeyDown property 54, 87
MouseEventHandler method 56
MouseEventHandler object 55
    MouseEventHandler method 56
    onEvent method 56
    onMouseDoubleClick property 55
    onMouseDown property 55
    onMouseHit property 55
    onMouseMove property 55
    onMouseOut property 55
    onMouseOver property 55
    onMouseUp property 55
    reportAllTargets property 56
    target property 56
mouseX property 53
mouseY property 53
multiply method 41
multiplyInPlace method 41

N

name property 83
near property 20, 23
nextSibling property 57
Node object 57
    computeBoundingBox method 58
    detachFromCurrentAnimation method 58
    directionLocal property 37
    firstChild property 57
    hitEnabled property 57
    info property 57
    metadataString property 57
    nextSibling property 57
    parent property 57
    transform property 57
    visible property 57
    wireframeColor property 57
node property 89
nodes property 77
normalize method 61, 100

O

objectGUID property 83
objectID property 83
onEvent method 24, 36, 51, 56, 63, 88, 90, 94, 96
onMouseDoubleClick property 55
onMouseDown property 55
onMouseHit property 55
onMouseMove property 55
onMouseOut property 55
onMouseOver property 55
onMouseUp property 55
opacityTexture property 39
outerConeAngle property 38
outerRadius property 38
outlineAngle property 78
overrideNavTools property 67
overridePanTool property 68
overrideRotateTool property 68
overrideSelection property 68
overrideSpinTool property 68
overrideViewChange property 68
overrideWalkTool property 68
overrideWheelSpeed property 68
overrideZoomTool property 68

P

parent property 57
phongExponent property 39
pointsRenderModeColor property 64
position property 20, 31, 38
positionLocal property 20, 38
print method 29
println method 29
Procedural object 59
projectionType property 20, 23

Q

Quaternion method 60
Quaternion object 60
    interpolate method 61
    interpolateInPlace method 61
    normalize method 61
    Quaternion method 60

R

reflectionStrength property 39
reflectionTexture property 39
refresh method 72
remove method 26
removeAll method 85
removeByIndex method 85
removeCustomMenuItem method 73
removeCustomToolButton method 73
removeEventHandler method 72
removeItem method 86
RENDER_MODE_BOUNDING_BOX property 79
RENDER_MODE_DEFAULT property 78
RENDER_MODE_HIDDEN_WIREFRAME property 80
RENDER_MODE_ILLUSTRATION property 80
RENDER_MODE_SHADED_ ILLUSTRATION property 80
RENDER_MODE_SHADED_SOLID_ WIREFRAME property 79
RENDER_MODE_SHADED_VERTICES property 79
RENDER_MODE_SHADED_WIREFRAME property 79
RENDER_MODE_SOLID property 79
RENDER_MODE_SOLID_OUTLINE property 80
RENDER_MODE_TRANSPARENT property 79
RENDER_MODE_TRANSPARENT_ BOUNDING_BOX property 79
RENDER_MODE_TRANSPARENT_ BOUNDING_BOX_OUTLINE property 79
RENDER_MODE_TRANSPARENT_ WIREFRAME property 80
RENDER_MODE_VERTICES property 79
RENDER_MODE_WIREFRAME property 79
renderDoubleSided property 78
RenderEvent object 62
    canvas property 62
    canvasPixelHeight property 62
    canvasPixelWidth property 62
    currentTool property 62
RenderEventHandler method 63
RenderEventHandler object 63
    onEvent method 63
    RenderEventHandler method 63
renderMode property 52, 78
RenderOptions object 64
    boundingBoxColor property 64
    clippingPlaneColor property 64
    clippingPlaneIntersectionColor property 64
    defaultAmbientColor property 64
    defaultDiffuseColor property 64
    defaultEmissiveColor property 64
    defaultSpecularColor property 64
    illustrationRenderModeFaceColor property 64
    illustrationRenderModeLineColor property 64
    pointsRenderModeColor property 64
    shadedIllustrationRenderModeLineColor property 64
RenderOptions object (Continued)
    solidGridColorEven property 65
    solidGridColorOdd property 65
    solidRenderModeLineColor property 65
    transparentBoundsRenderModeColor property 65
    transparentGridColorEven property 65
    transparentGridColorOdd property 65
    wireframeRenderModeColor property 65
    xAxisColor property 65
    yAxisColor property 65
    zAxisColor property 65
reportAllTargets property 56
Resource method 66
Resource object 66
    Resource method 66
    type property 66
    TYPE_IMAGE property 66
    TYPE_MODEL property 66
    TYPE_UNKNOWN property 66
rightButtonDown property 54
roll property 21
rotateAboutLine method 42
rotateAboutLineInPlace method 43
rotateAboutVector method 44
rotateAboutVectorInPlace method 44
rotateAboutX method 43
rotateAboutXInPlace method 44
rotateAboutY method 45
rotateAboutYInPlace method 45
rotateAboutZ method 45
rotateAboutZInPlace method 46
rotateWithQuaternion method 42
rotateWithQuaternionInPlace method 42
Runtime object 67
    addCustomMenuItem method 70
    addCustomToolButton method 70
    addEventHandler method 71
    BUTTON_TYPE_PUSH property 67
    BUTTON_TYPE_TOOL property 67
    canvasCount property 67
    ctrlKeyDown property 67
    disableTool method 71
    enableTool method 71
    eventHandlerCount property 67
    getEventHandler method 72
    getRendererName method 72
    instances property 67
    MENU_ITEM_TYPE_CHECKED property 67
    MENU_ITEM_TYPE_DEFAULT property 67
    overrideNavTools property 67
    overridePanTool property 68
    overrideRotateTool property 68
    overrideSelection property 68
    overrideSpinTool property 68
    overrideViewChange property 68
    overrideWalkTool property 68
    overrideWheelSpeed property 68
    overrideZoomTool property 68
    refresh method 72
    removeCustomMenuItem method 73

Runtime object (Continued)
removeCustomToolButton method 73
removeEventHandler method 72
scrollWheelSpeed property 68
setCurrentTool method 73
shiftKeyDown property 68
speedThreshold property 69
strafeSpeed property 69
TOOL_NAME_MEASURE property 69
TOOL_NAME_PAN property 69
TOOL_NAME_ROTATE property 69
TOOL_NAME_SPIN property 69
TOOL_NAME_WALK property 69
TOOL_NAME_ZOOM property 69
version property 69
walkSpeed property 69

S

scale method 46, 101
scaleComponent property 40
scaleInPlace method 46, 101
Scene object 75
activateAnimation method 81
addModel method 81
ambientIlluminationColor property 75
animations property 75
cameras property 75
computeBoundingBox method 82
createClippingPlane method 81
createLight method 81
createSquareMesh method 82
defaultRenderOptions property 75
GRID_MODE_OFF property 75
GRID_MODE_SOLID property 75
GRID_MODE_TRANSPARENT property 75
GRID_MODE_WIRE property 76
gridMode property 75
gridSize property 76
lengthUnits property 76
LIGHT_MODE_BLUE property 76
LIGHT_MODE_BRIGHT property 76
LIGHT_MODE_CAD property 77
LIGHT_MODE_CUBE property 76
LIGHT_MODE_DAY property 76
LIGHT_MODE_FILE property 76
LIGHT_MODE_HEADLAMP property 77
LIGHT_MODE_NIGHT property 76
LIGHT_MODE_NONE property 76
LIGHT_MODE_RED property 76
LIGHT_MODE_RGB property 76
LIGHT_MODE_WHITE property 76
lights property 77
lightScaleFactor property 77
lightScheme property 77
materials property 77
meshes property 77
nodes property 77
outlineAngle property 78
RENDER_MODE_BOUNDING_BOX property 79
Scene object (Continued)
RENDER_MODE_DEFAULT property 78
RENDER_MODE_HIDDEN_WIREFRAME property 80
RENDER_MODE_ILLUSTRATION property 80
RENDER_MODE_SHADED_ ILLUSTRATION property 80
RENDER_MODE_SHADED_SOLID_ WIREFRAME property 79
RENDER_MODE_SHADED_VERTICES property 79
RENDER_MODE_SHADED_WIREFRAME property 79
RENDER_MODE_SOLID property 79
RENDER_MODE_SOLID_OUTLINE property 80
RENDER_MODE_TRANSPARENT property 79
RENDER_MODE_TRANSPARENT_ BOUNDING_BOX property 79
RENDER_MODE_TRANSPARENT_ BOUNDING_BOX_OUTLINE property 79
RENDER_MODE_TRANSPARENT_ WIREFRAME property 80
RENDER_MODE_VERTICES property 79
RENDER_MODE_WIREFRAME property 79
renderDoubleSided property 78
renderMode property 78
selectedNode property 80
showAxes property 80
showGrid property 78
smoothing property 80
smoothingAngle property 80
smoothingOverride property 80
update method 82
SceneObject object 83
name property 83
objectGUID property 83
objectID property 83
SceneObjectList object 84
count property 84
getByGUID method 84
getByID method 84
getByIndex method 84
getByName method 85
removeAll method 85
removeByIndex method 85
removeItem method 86
ScrollWheelEvent object
canvas property 87
canvasPixeleight property 87
canvasPixelWidth property 87
ctrlKeyDown property 87
currentTool property 87
deltaY property 87
shiftKeyDown property 87
ScrollWheelEventHandler method 88
ScrollWheelEventHandler object
onEvent method 88
ScrollWheelEventHandler method 88
scrollWheelSpeed property 68
selected property 89
selectedNode property 80
SelectionEvent object
node property 89
selected property 89
SelectionEventHandler method 90

SelectionEventHandler object
    onEvent method 90
    SelectionEventHandler method 90
set method 27, 47, 101
set3 method 28, 102
setCamera method 25
setColor method 16
setCurrentTool method 73
setIdentity method 47
setImage method 17, 92
setView method 47
shadedIllustrationRenderModeLineColor property 64
shiftKeyDown property 35, 54, 68, 87
showAxes property 80
showGrid property 78
smoothing property 80
smoothingAngle property 80
smoothingOverride property 80
solidGridColorEven property 65
solidGridColorOdd property 65
solidRenderModeLineColor property 65
specularColor property 39
specularStrength property 39
speedThreshold property 69
startTime property 15
strafeSpeed property 69
subtract method 102
subtractInPlace method 103
surfaceNormal property 31

T

target property 21, 31, 56
targetDistance property 23
targetPosition property 21
targetPositionLocal property 21
Texture object 91
    amount property 91
    angle property 91
    clampU property 91
    clampV property 91
    getImage method 91
    image property 91
    modulate property 91
    setImage method 92
    uOffset property 91
    uScale property 91
    use3DSStyleMapping property 91
    vOffset property 91
    vScale property 91
textureCoordinate property 31
time property 93
TimeEvent object 93
    deltaTime property 93
    time property 93
TimeEventHandler method 94
TimeEventHandler object 94
    onEvent method 94
    TimeEventHandler method 94
TOOL_NAME_MEASURE property 69
TOOL_NAME_PAN property 69
TOOL_NAME_ROTATE property 69
TOOL_NAME_SPIN property 69
TOOL_NAME_WALK property 69
TOOL_NAME_ZOOM property 69
ToolEvent object 95
    canvas property 95
    canvasPixelHeight property 95
    canvasPixelWidth property 95
    currentTool property 95
    toolName property 95
ToolEventHandler method 96
ToolEventHandler object 96
    onEvent method 96
    ToolEventHandler method 96
toolName property 95
transform property 23, 57
transformDirection method 48
transformPosition method 48
translate method 49
translateInPlace method 49
translation property 40
transparentBoundsRenderModeColor property 65
transparentGridColorEven property 65
transparentGridColorOdd property 65
transpose property 40
transposeInPlace method 49
type property 38, 66
TYPE_IMAGE property 66
TYPE_INFINITE property 38
TYPE_MODEL property 66
TYPE_ORTHOGRAPHIC property 21
TYPE_PERSPECTIVE property 21
TYPE_POINT property 38
TYPE_SPOT property 38
TYPE_UNKNOWN property 66

U

uOffset property 91
up property 21
update method 82
upLocal property 21
uScale property 91
use3DSStyleMapping property 91

V

Vector3 method 97
Vector3 object 97
    add method 98
    addInPlace method 98
    addScaled method 98
    addScaledInPlace method 99
    blend method 99
    blendInPlace method 99
    cross method 100
    dot method 100
    length property 97
    normalize method 100
    scale method 101

Vector3 object (Continued)
scaleInPlace method 101
set method 101
set3 method 102
subtract method 102
subtractInPlace method 103
Vector3 method 97
x property 97
y property 97
version property 69
viewPlaneSize property 21, 23
visible property 57
vOffset property 91
vScale property 91

W

walkSpeed property 69
width property 33
wireframeColor property 57
wireframeRenderModeColor property 65

X

x property 97
xAxisColor property 65

Y

y property 97
yAxisColor property 65

Z

z property 97
zAxisColor property 65