

**The  
Google  
O3D Bible**

created by:

Kara Rawson



# **foreword**

The purpose of this document, is to mainly provide an offline resource for referencing the o3d API library. Currently this text is directly copied from o3d website based on the most stable current version that o3d is release under, 0.1.38. I do not know how long or can garentee the validity of all of the text contained within this document, as the o3d project is constantly changing. All link you find within this document point directly to googles website for o3d, so keep that in mind when you click around.

When I first started this I didn't actually realize how enormous this API was until I began copy and pasting. For starters there are over a hundred classes, all of which are fairly indepth, easy to understand, but extremely comprehensive. All in all this document will be a work in progress, and hopefully as o3d becomes more stable and adobted we will be able to start including examples, tutorials and more details explanation written in laymens terms within this doc.

I hope you find this document useful, as do I. If you have any corrections, comments, or feedback, please sent them to katacutey at gmail dot com or post them up on the o3d discussion group and I will get to them there. Thank you, and happy coding.

~Kara Rawson

# Installation Instructions

## Contents

1. [Install the O3D Plug-in](#)
2. [System Requirements](#)
3. [Graphics Card Support](#)
4. [Graphics Hardware with Known Issues](#)
5. [How to Determine Your Graphics Hardware](#)
  - [On Windows XP](#)
  - [On Windows Vista](#)
  - [On Mac OS X](#)

## Install the O3D Plug-in

Click one of the following links to install the O3D Plug-in and run the sample applications. To develop an O3D application, all you need is the O3D plug-in and a text editor for writing JavaScript code (you do not need the O3D SDK). Be sure to check out the graphics card requirements.

- Install the [Windows O3D Plug-in](#).
- Install the [Mac O3D Plug-in](#). (Safari users: you will need to close and restart your browser after installing O3D.)
- Linux users, follow [these instructions](#) for building O3D from the source code.

## System Requirements

### Software Requirements

- *Windows*: XP Service Pack 2, Vista (x86/x64) Service Pack 1; Firefox 2+, Internet Explorer 7.0+ (x86 only), or Google Chrome
- *Mac*: Intel Mac with OS X v10.4 or later; Firefox 2+, Safari 3+, or Camino

### Hardware Requirements

- *Windows*: x86 CPU; DX9-compatible GPU with VS2.0 and PS2.0 support
- *Mac*: any Intel Mac (unsupported GPUs will use SW renderer)

The O3D plug-in implementation may not work with some configurations that meet the requirements specified above. These incompatibilities can be caused by certain hardware combinations, different driver versions, or different OS or browser versions.

[Back to top](#)

## Graphics Card Support

This list contains graphics hardware that we have tested and believe works with O3D. Other configurations may work but have not yet been tested.

<b>Graphics Hardware</b>	<b>Windows</b>	<b>Mac</b>
ASUS ATI EAH 3450	✓	
ATI Radeon HD 2400 Pro SB Edition	✓	
ATI Radeon HD 2600		✓
ATI Radeon HD 2600 Pro	✓	
ATI Radeon HD 3850	✓	
ATI Radeon HD 4850	✓	
ATI Radeon Sapphire HD3650	✓	
ATI Radeon X1550	✓	
Nvidia GeForce 6200 LE	✓	
Nvidia GeForce 7300 SE/7200 GS	✓	
Nvidia GeForce 8400 GS	✓	
Nvidia GeForce 8500 GT	✓	
Nvidia GeForce 8600 GT	✓	
Nvidia GeForce 8600M GT		✓
Nvidia GeForce 8800 GT		✓
Nvidia GeForce 9600 GT	✓	
Nvidia GeFroce 9800 GT	✓	
Nvidia Quadro FX 570	✓	

[Back to top](#)

## Graphics Hardware with Known Issues

This list contains graphics hardware that we have tested that causes known issues (such as browser crashes) with O3D.

<b>Graphics Hardware</b>	<b>Windows</b>	<b>Mac</b>
Intel GMA 950	No support	
Intel X3100		Unstable
ATI Radeon X1650	Some issues	

## How to Determine Your Graphics Hardware

The following sections describe how to determine the type of your graphics hardware on Windows XP, Windows Vista, and Mac OS X systems.

### On Windows XP

To determine your graphics hardware type on Windows XP systems, follow these steps:

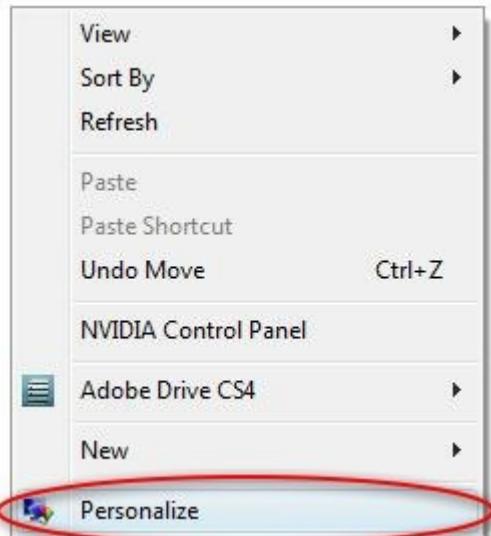
1. Right-click the Windows desktop and choose "Properties" from the menu.
2. Click the "Advanced Settings..." button.
3. Your graphics hardware type is listed under "Adapter Type" in the "Advanced Settings" window.

[Back to top](#)

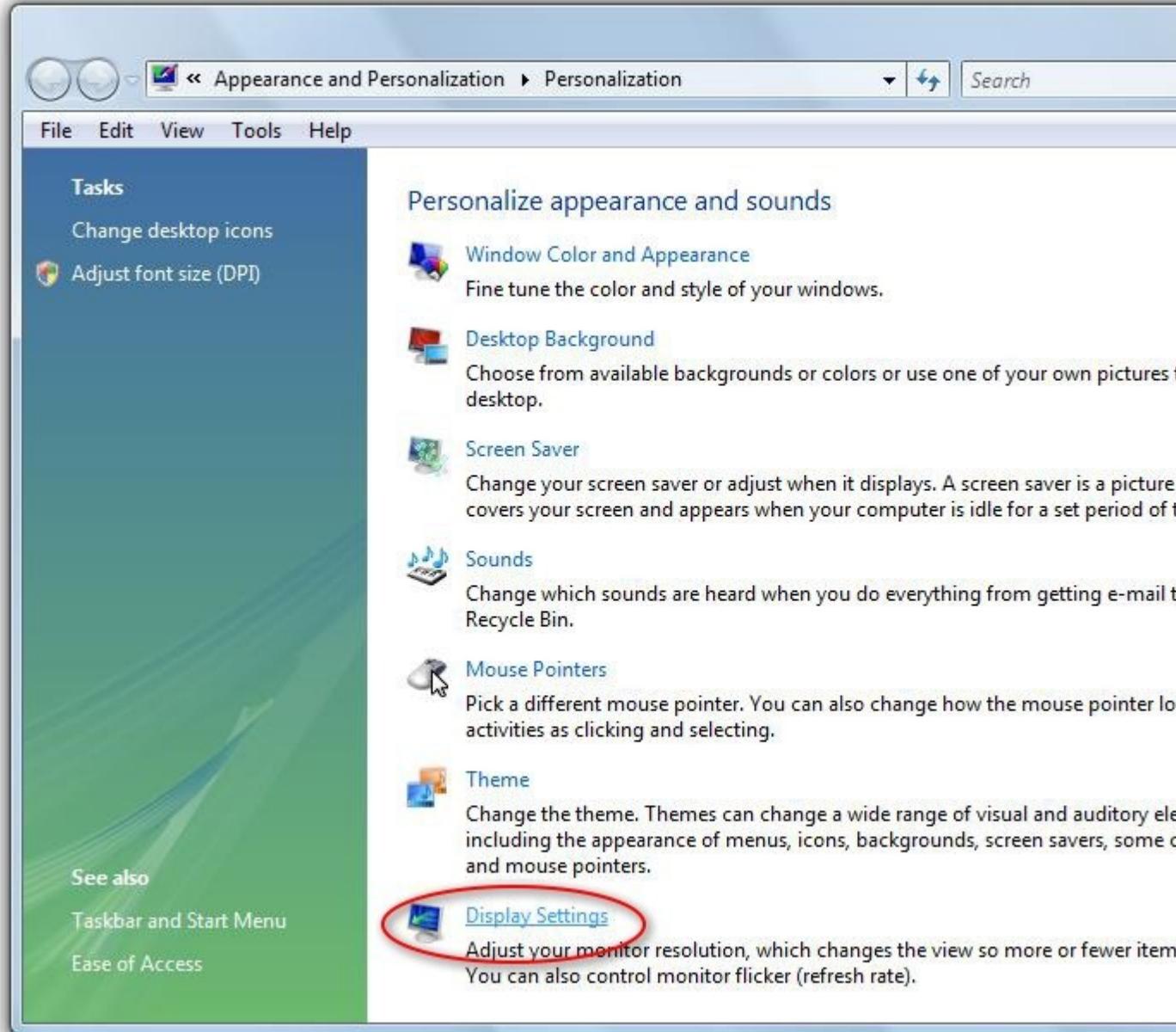
## On Windows Vista

To determine your graphics hardware type on Windows Vista systems, follow these steps:

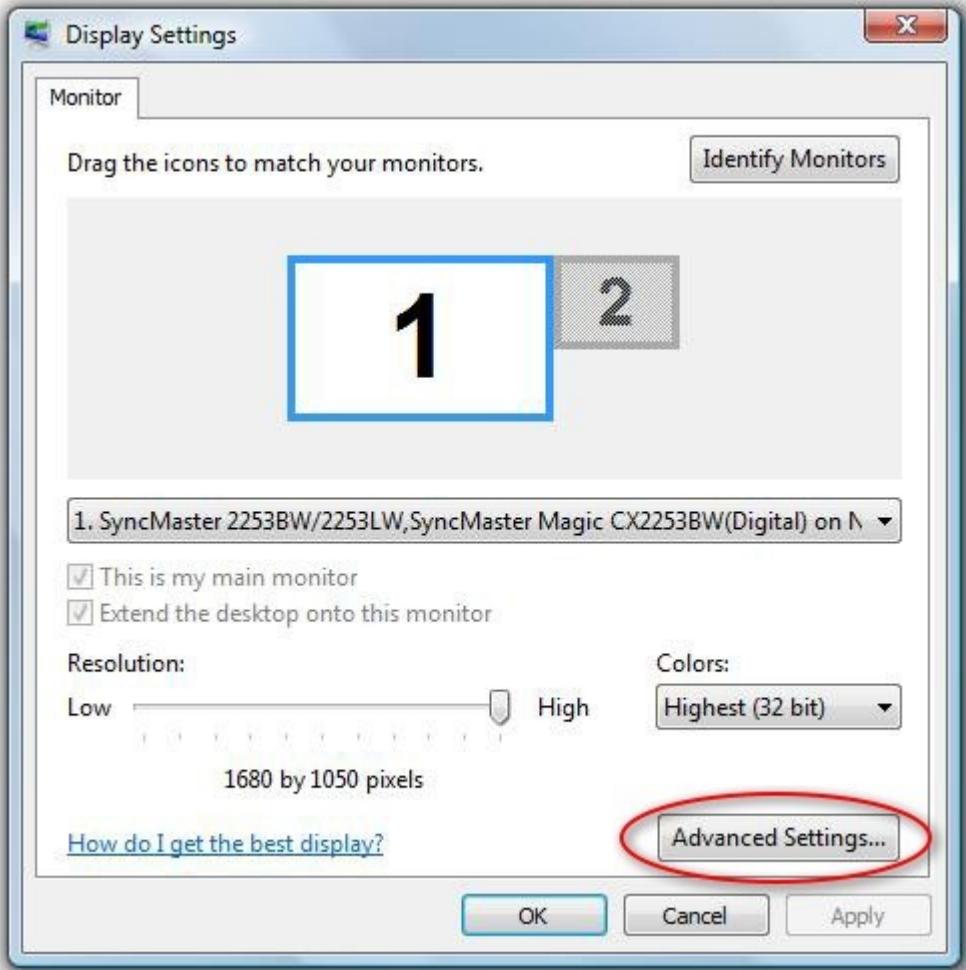
1. Right-click the Windows Desktop and choose "Personalize" from the menu.



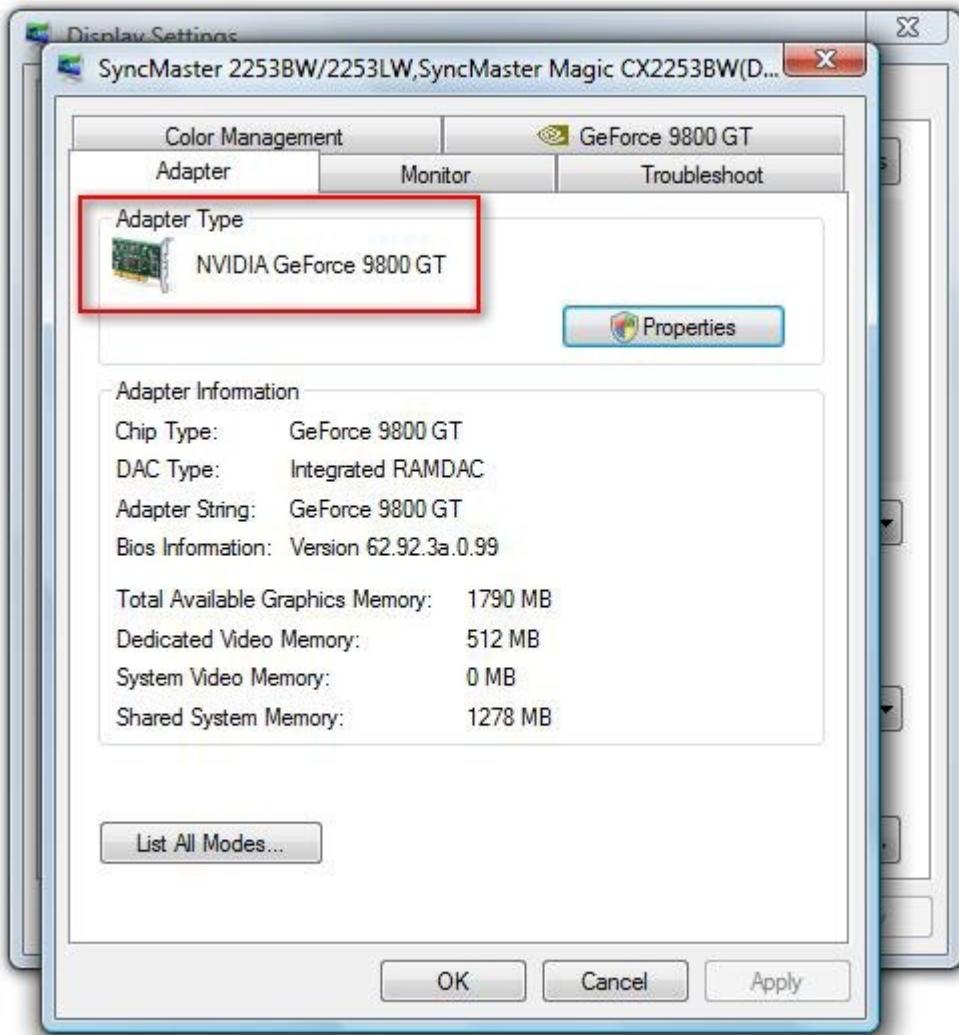
2. Click the "Display Settings" link at the bottom of the Personalize Control Panel.



3. Click the "Advanced Settings..." button in the Display Settings Control Panel.



4. Your graphics hardware type will be listed under "Adapter Type" in the "Advanced Settings" window.

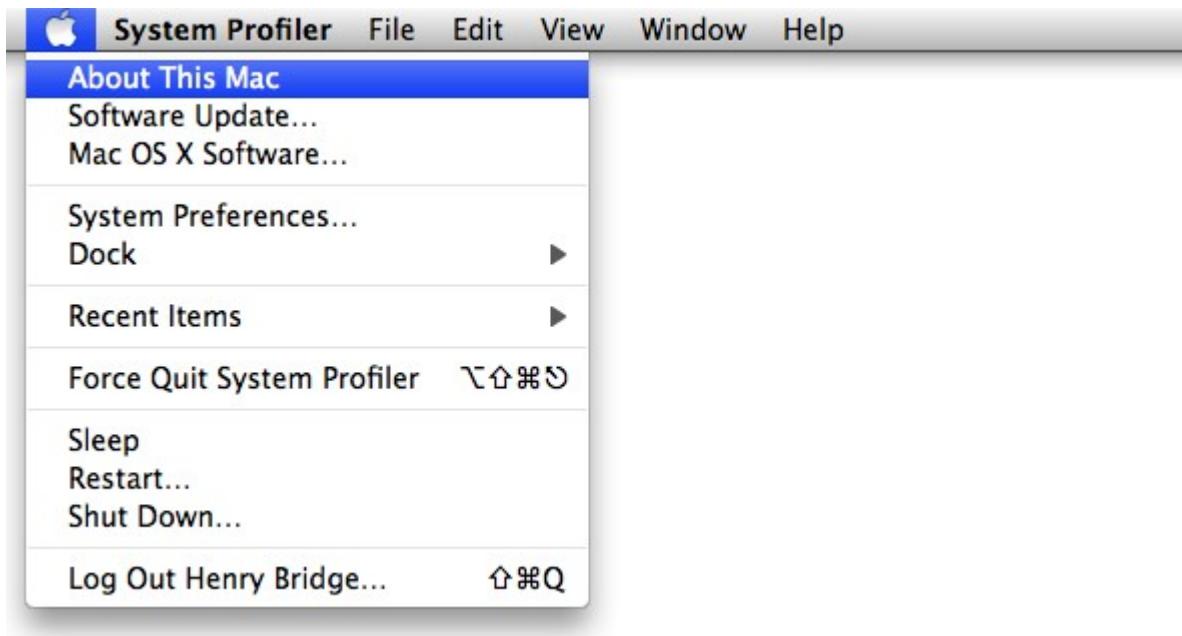


[Back to top](#)

## On Mac OS X

To determine your graphics hardware type on Mac OS X systems, follow these steps:

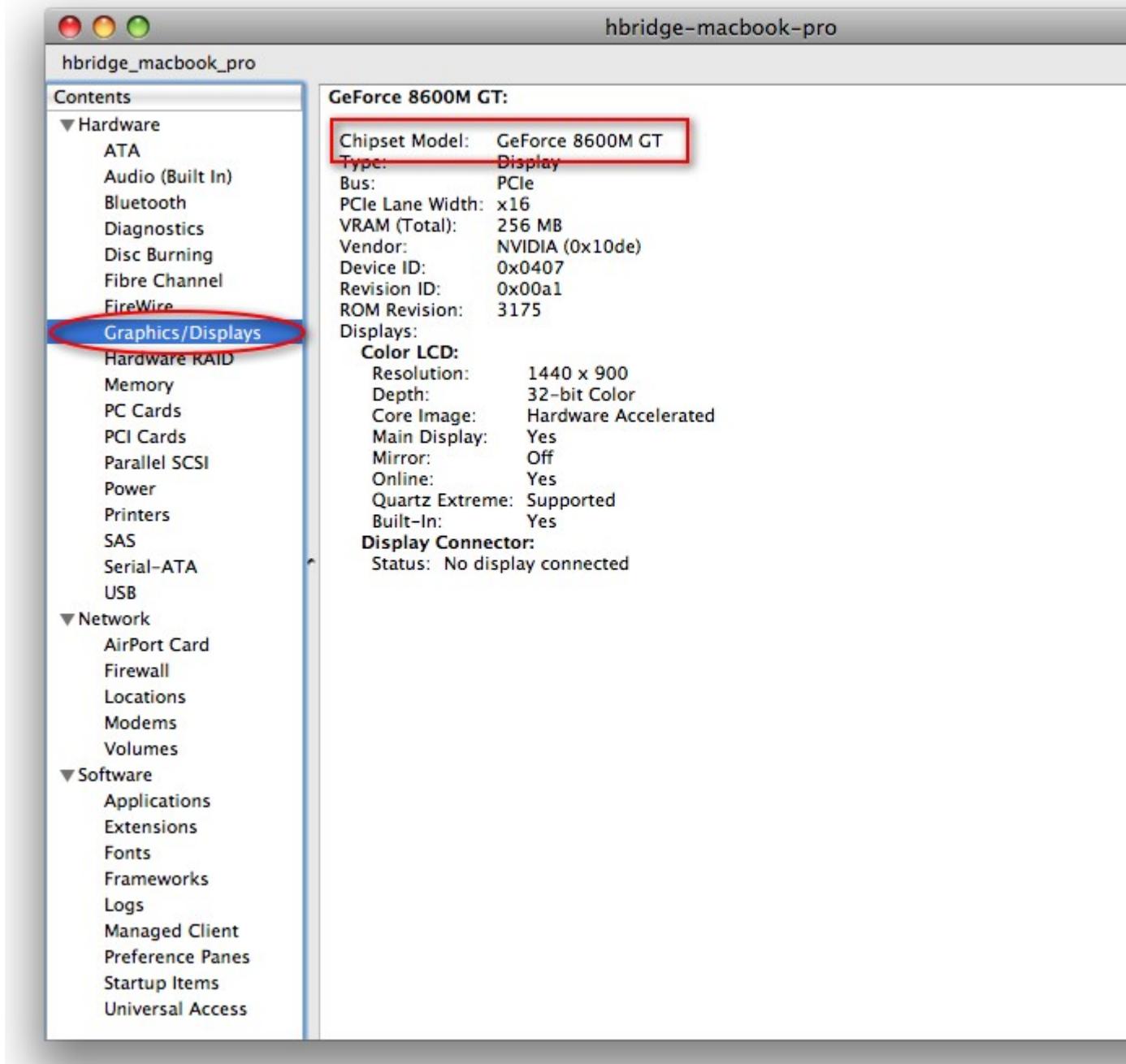
1. Click the Apple menu and choose the "About This Mac" option.



2. Click the "More Info..." button.



3. Choose "Graphics/Displays" from the sidebar on the left. Your graphics hardware type will be listed next to "Chipset Model."





# Programmable Graphics Pipeline

O3D uses a programmable graphics pipeline model instead of a fixed-function pipeline. This programmable pipeline makes use of a shader language, based on HLSL and Cg, that enables you to program the GPU directly through the use of *vertex shaders* and *pixel shaders*. Before this era of programmable GPUs, the graphics programmer was limited to a fixed-function pipeline. Algorithms for calculating transformations, lighting, texture coordinates, and other environmental effects were pre-programmed into software such as early OpenGL or Direct 3D, which controlled the graphics hardware. In systems based on a fixed-function pipeline, global states are set up for lights, materials, and textures, and then the shape information is passed in to this pipeline. In contrast, with a programmable graphics pipeline, the developer has complete control over the algorithms used in the vertex shader and the pixel shader. In addition, rasterizing and frame-buffer operations can be configured using the O3D API.

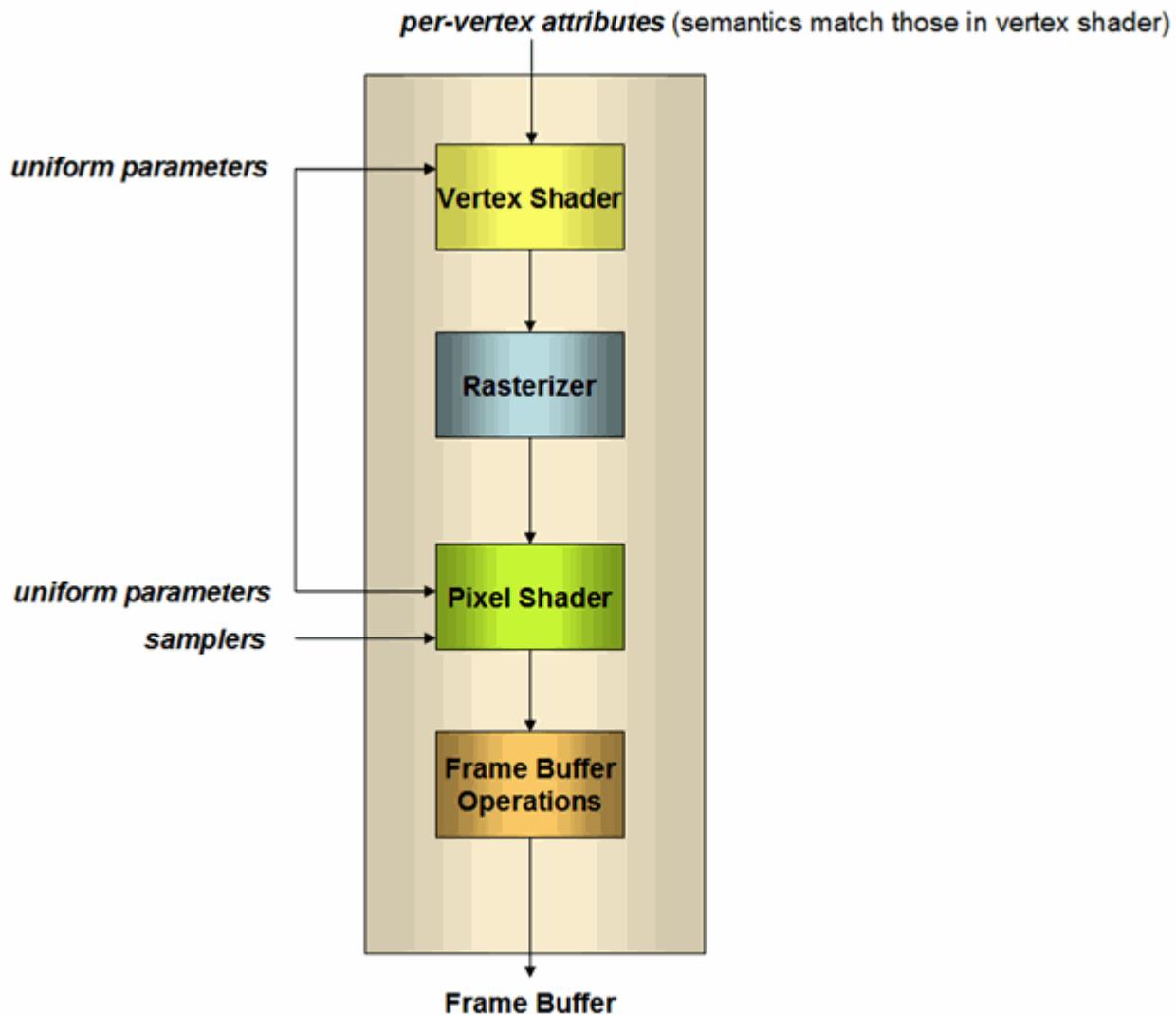
The following sections provide an overview of how to program the O3D graphics pipeline, with brief contrasting descriptions of how the same techniques would be accomplished in a fixed-function system.

## Contents

1. [Components of the O3D Pipeline](#)
2. [Defining the Shader Algorithms](#)
3. [O3D Shading Language](#)
4. [What the Vertex Shader Does](#)
5. [Rasterization](#)
6. [What the Pixel Shader Does](#)
7. [Frame Buffer Operations](#)
8. [What's Next?](#)

## Components of the O3D Pipeline

The O3D programmable pipeline offers complete flexibility with respect to the algorithms used for vertex shading and pixel shading. The following diagram shows the major components of this pipeline:



The **vertex shader** is composed of algorithms that you write, "borrow," or modify. These algorithms calculate the values of the per-vertex attributes as well as the position of each vertex in homogeneous clip space.

The **rasterizer** has a set of configurable states that can be set in the O3D State object. The rasterizer is used for interpolation of the vertex attribute values as well as for calculations related to other operations, such as viewport clipping and backface culling.

The **pixel shader** takes input from the rasterizer and outputs one color for each pixel of the primitive. You write, "borrow," or modify the algorithms that compute these pixel colors.

**Frame buffer operations** are a set of configurable states set in the O3D State object. These operations are used for depth testing, stenciling, and blending among others.

## Defining the Shader Algorithms

The vertex shader and pixel shader each have user-defined input and output. Vertex streams form the input to the vertex shader. The vertex shader uses two types of input values (per-vertex attributes and uniform parameters) to produce a modified position and an arbitrary number of attributes for each vertex in the primitive. These computations are based on custom algorithms you define in your shaders.

# O3D Shading Language

O3D uses a variation of the HLSL and Cg shading languages. Vertex and pixel shaders are specified by supplying a string containing the source code. Because it is just a string, you can store a shader in a variable, which can be contained in a hidden <textarea> element within the HTML document, downloaded from a URL, or stored anywhere else your web page would keep a string. See [O3D Shading Language](#) for more information.

[Back to top](#)

## What the Vertex Shader Does

The vertex shader processes each vertex in the input vertex streams and outputs two kinds of information for each vertex:

- a vertex position in homogeneous clip space
- an arbitrary number of interpolants (that is, attribute values to be interpolated)

### Input: Per-Vertex Attributes

Each vertex can contain an arbitrary number of numerical attributes. Examples of these per-vertex attributes are:

- Position
- Texture coordinate
- Color
- Normal
- Weight
- Magnetic attraction
- Index in an array
- and anything else you can think of!

These numerical attributes are used as input to the vertex shader. They can be of type `float1`, `float2`, `float3`, or `float4`, which are groups of 1 to 4 floating point values, respectively.

### Input: Uniform Parameters

In addition to the per-vertex attributes, a number of parameters that apply to all vertices in the primitive are used as input to the vertex shader. These parameters can be matrices, vectors, or floats. For example:

- Matrices
  - Projection matrix
  - View matrix
  - Model matrix
  - Texture matrix
  - Color matrix
  - Bone matrix
  - etc.
- Vectors
  - Light position

- Light color
- Motion vector
- Floats
  - Time
  - Mass

## Output

The output of the vertex shader is of two types:

- a vertex position in homogeneous clip space
- an arbitrary number of attributes for each vertex

$-w < x < w, -w < y < w$  and  $0 < z < w$

This means that

$-1 < x/w < 1, -1 < y/w < 1, 0 < z/w < 1$

## Fixed-Function Pipeline Implementations

The following sections provide examples of how a fixed-function system such as OpenGL or Direct3D would implement the operations performed by the vertex shader, which include:

- Transformations
- Lighting calculations
- Texture coordinate generation
- Fog

### Calculating Transformations in a Fixed-Function System

Here is how you would calculate the transformations for a vertex from local space to homogeneous clip space in a fixed function system. You could incorporate this calculation into your vertex shader, or you could modify it to suit your needs.

```
OUT.POSITION = projectionTransform * viewTransform * modelTransform * IN.POSITION
```

### Calculating Lighting in a Fixed-Function System

Here is how you would calculate lighting for a vertex in a fixed-function system. You could incorporate this calculation into your vertex shader, or you could modify it to suit your needs.

```
OUT.COLOR = material.emissive
           + light.ambient * material.ambient
           + light.diffuse * material.diffuse * dot(normal, lightvector)
           + light.specular * material.specular *
             dot(normal, halfvector) ^ material.shininess
```

### Calculating Texture Coordinates in a Fixed-Function System

Here is how you would calculate the texture coordinates for a vertex in a fixed function system. You could incorporate this calculation into your vertex shader, or you could modify it to suit your needs.

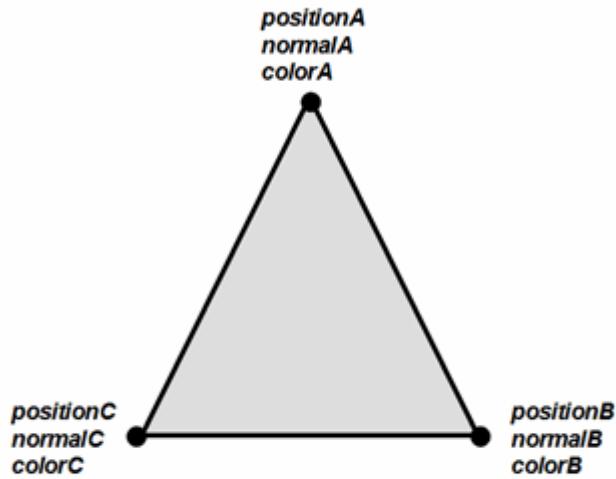
```
OUT.textureCoord = textureMatrix * in.textureCoord
```

[Back to top](#)

## Rasterization

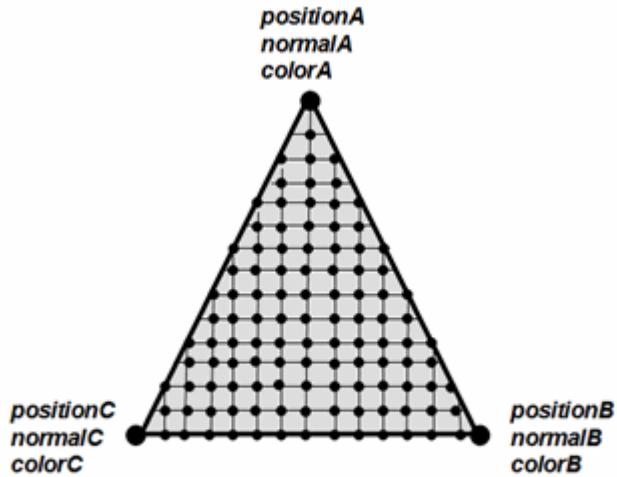
The output of the vertex shader has a one-to-one correspondence between each vertex, its position in homogeneous clip space, and the attribute values that are to be interpolated. The rasterizer prepares this data for the pixel shader by interpolating between the values assigned to each vertex for each attribute and producing a corresponding attribute value for each *pixel*. The input to the pixel shader is an attribute value for each pixel in the primitive.

For example, here is a triangle with three vertex attributes that is used as input to the rasterizer:



Triangle

The rasterizer then creates position, normal, and color attribute values for each of the pixels in this triangle, as shown in the diagram below. The rasterizer computes these per-pixel values by interpolating between the values that have been specified explicitly for the vertices in the triangle. These per-pixel values are then used as input to the pixel shader.



Rasterized Triangle

The O3D rasterizer is similar to the rasterizer in fixed-function systems. O3D has a `State` object that allows you to set certain rendering states that affect how the rasterizer does its job. For example, you can set a custom viewport as well as specify parameters for backface culling, polygon mode, polygon offset, and more.

## What the Pixel Shader Does

The job of the pixel shader is to perform all of its calculations on the per-pixel interpolants supplied as input and, at the end, to output a single `float4` color value for each pixel in the primitive.

A simple shader could return a constant color or the color of a uniform param. For example:

```
float4 pixelShaderFunction(PixelShaderInput input) : COLOR {
    return float4(1,0,0,1);
}
```

or

```
uniform float4 myColor;

float4 pixelShaderFunction(PixelShaderInput input) : COLOR {
    return myColor;
}
```

A shader might do some lighting calculations. For example:

```
float4 pixelShaderFunction(PixelShaderInput input) : COLOR {
    float3 surfaceToLight = normalize(lightWorldPos - input.worldPosition);
    float3 normal = normalize(input.normal);
    float3 surfaceToView = normalize(viewInverse[3].xyz - input.worldPosition);
    float3 halfVector = normalize(surfaceToLight + surfaceToView);
    float4 litResult = lit(dot(normal, surfaceToLight),
                           dot(normal, halfVector), shininess);
    return float4((diffuse * litResult.y)).rgb, diffuse.a;
}
```

A shader can get colors or values from a texture. In your pixel shader code, use the `tex2D()` function to return a `Float4` color from a 2D texture. For example:

```
sampler mySampler;  
  
float4 pixelShaderFunction(PixelShaderInput input): COLOR {  
    return tex2D(mySampler, input.texCoords);  
}
```

For cube-mapped textures, use the `texCUBE()` function.

One advantage of the pixel shader is that it allows you to perform per-pixel computations for any attribute value. This capability is particularly effective, for example, in calculating the specular lighting component. You will need to balance performance considerations against quality and evaluate the tradeoffs of using the pixel shader for fine-grained computation, or the vertex shader for more efficient (and somewhat less explicit) specification of attributes.

Another advantage of the pixel shader is that it has access to textures, whereas the vertex shader does not. Your pixel shader, for example, could have two texture samplers set as uniform parameters. One texture could be a bitmap of colors, while a second texture could be a map of per-pixel normals. (Think of a hooked rug, where the yarns in the rug are the color values, and the directions the threads are pointing in are the normal values.) If you had a texture consisting of per-pixel normals, the formula for computing the diffuse color using per-pixel normals would be as follows:

```
OUT.COLOR = material.diffuse * light.diffuse * dot(tex2D(sampler, textureCoord),  
lightvector)
```

The variables in the shader code correspond to parameters that have the same names and matching types in the O3D JavaScript transform graph. Typically, these parameters are set on the material. However, they could also be set on a transform, draw element, primitive, or effect. When the render graph is traversed, the values for these parameters in the O3D transform graph are fed into the vertex and pixel shaders, which perform their calculations using these input values to produce the final color for each pixel on the screen.

## Frame Buffer Operations

Frame buffer operations are configurable through the `State` object in O3D. These operations include alpha testing, depth testing, blending, and stenciling. This is the final step in the programmable graphics pipeline. After the frame buffer operations are performed, the output is displayed on the screen.

## What's Next?

Learn about the basic structure of a O3D program in the chapter on [Program Structure](#).

# Program Structure

## Overview

O3D is an open-source JavaScript API for creating interactive 3D graphics applications that run in a browser window: games, ads, 3D model viewers, product demos, virtual worlds. This *Developer's Guide* presents a series of examples that illustrate the key concepts and typical programming tasks used in O3D application development.

This chapter introduces you to the general structure of an O3D application. To begin, all you need is the O3D plug-in, a text editor, and a web browser to create and view the HTML document that includes the O3D JavaScript code.

The *Hello, Cube* sample creates a spinning red cube. (To view the code for *Hello, Cube* in a new window, [click here](#)).

## Basic Tasks

The basic tasks performed in an O3D program are the following:

1. Create the O3D object.
2. Assign values to global variables and initialize utility libraries.
3. Create the pack to manage O3D objects.
4. Create the render graph.
5. Set up the draw context (perspective and viewing transformations).
6. Create an effect and load the shader information into it.
7. Create the material and shape, set the material's draw list, and set up other material parameters.
8. Add the transforms and shapes to the transform graph.
9. Create the draw elements for the primitives.
10. Set the optional render callback function, if desired, to perform special tasks each time the 3D scene is rendered.

The following sections discuss how the *Hello, Cube* sample implements these tasks.

## HTML Document

An O3D application is contained in an HTML document. The main code for the O3D JavaScript application is contained in a `<script>` element inside the `<head>` element of the HTML document. In this example, when the HTML page is finished loading, the O3D `init()` function is called. The HTML page also sets up an `onunload` event, which calls the application's `uninit()` function to perform necessary cleanup, such as removing callback functions when the HTML page is unloaded..

```
<script type="text/javascript" src="o3djs/base.js"></script>
<script type="text/javascript">
o3djs.require('o3djs.util');
o3djs.require('o3djs.math');
o3djs.require('o3djs.rendergraph');
window.onload = init;
window.onunload = uninit;
.
```

.

## Utility Libraries

O3D includes a number of [utility libraries](#) that simplify the coding of common tasks. If you use functions from one of these libraries, you also need to include that code in `<script>` tags at the beginning of your program as shown below. (The first `<script>` element defines the `require` function, which is used thereafter to initialize the other utility libraries.):

```
<script type="text/javascript" src="o3djs/base.js"></script>
<script type="text/javascript">
o3djs.require('o3djs.util');
o3djs.require('o3djs.math');
o3djs.require('o3djs.rendergraph');
```

## Creating the O3D Plug-in Object

In the code sample, the `init()` function calls the utility function `o3djs.util.makeClients()` to [make the O3D objects](#). When this function returns, it invokes the callback function `initStep2()`. The user's browser must have scripting enabled in order for O3D HTML elements to be created.

```
function init() {
    o3djs.util.makeClients(initStep2)
}
```

## Setting the Size of the Client Area on the Page

Use the `o3djs.util.makeClients()` function to create O3D objects in HTML that can be used across platforms. This function finds all `<div>` elements with an id that starts with the word "o3d" (for example `o3d`, `o3d-element`, and so on) and inserts a client area inside. The size of the client area is always set to 100 percent, which means the `<div>` must have its size set or managed by the browser. For example:

```
<!-- A div of a specific size -->
<div id="o3d" style="width:800px; height:600px" />

<!-- A div that fills its containing element -->
<div id="o3d" style="width:100%; height:100%" />
```

The `makeClients()` utility takes a callback function as one of its parameters. This callback is triggered once the O3D objects have been created.

## Basic Setup for O3D

This code assigns values to global variables:

```
var o3dElement = clientElements[0];
g_client = o3dElement.client;
g_o3d = o3dElement.o3d;
g_math = o3djs.math;
```

These variables have the following meaning:

- `o3dElement` is the HTML O3D element, which is part of the DOM
- `g_client` is the entry point of the O3D application
- `g_o3d` is the namespace for O3D
- `g_math` is the namespace for the math library

## Creating the Pack

The pack contains all O3D objects and manages their lifetime.

```
g_pack = g_client.createPack();
```

## Creating the Render Graph

This example uses the utility function `renderGraph.createBasicView()` to create a standard render graph, as described in the [Technical Overview](#). This render graph has two draw lists, one for draw elements with opaque materials (the performance draw list) and one for draw elements with transparent materials (the transparency draw list). A separate draw pass is performed for each draw list.

```
var viewInfo = o3djs.renderGraph.createBasicView(  
    g_pack,  
    g_client.root,  
    g_client.renderGraphRoot);
```

## Setting Up the Draw Context

The draw context specifies the view projection and the position of a virtual camera that is viewing the scene (the view transformation). The `drawContext` object is created by the utility function `renderGraph.createBasicView()`. Here is an example of setting its values:

```
// Set up a simple perspective view.  
viewInfo.drawContext.projection = g_math.matrix4.perspective(  
    g_math.degToRad(30), // 30 degree fov.  
    g_client.width / g_client.height,  
    1, // Near plane.  
    5000); // Far plane.  
// Set up our view transformation to look towards the world origin where the  
// cube is located.  
viewInfo.drawContext.view = g_math.matrix4.lookAt([0, 1, 5], // eye  
                                                [0, 0, 0], // target  
                                                [0, 1, 0]); // up
```

## Creating an Effect and Loading the Shaders

The vertex and pixel shaders are defined in the `<textarea>` element of the HTML document. The shaders control the calculations for the color of each pixel in each draw element. This code creates the effect (`redEffect`) and reads in the contents of the shaders:

```
var redEffect = g_pack.createObject('Effect');
```

```

// looks in the HTML document for an element named "effect"
var shaderString = document.getElementById('effect').value;

// loads the entire contents of the <textarea id="effect"> element into the
redEffect object
redEffect.loadFromFXString(shaderString);

```

## Creating the Material and Shape

The red material for the cube is created in the `initStep2()` function and assigned to the performance draw list, which handles opaque materials. The following code also sets the material's effect to `redEffect` so that the graphics hardware can apply the proper shading to the cube. In this example, no shading calculations are performed. The simple color red is returned for all pixels. The `createCube()` function constructs the cube geometry, as described in [Shapes](#), and uses the red material. An alternative to setting up the geometry within O3D, as shown in the *Hello, Cube* examples, is to import geometry constructed using an application such as SketchUp, 3ds Max, or Maya.

```

var redMaterial = g_pack.createObject('Material');
redMaterial.drawList = viewInfo.performanceDrawList;
redMaterial.effect = redEffect;
var cubeShape = createCube(redMaterial);

```

## Setting Up the Transform Graph

The *Hello, Cube* example has a very simple transform graph. The following code creates one transform for the cube shape, adds the shape to the transform, and then adds the transform to the O3D root.

```

g_cubeTransform = g_pack.createObject('Transform');
g_cubeTransform.addShape(cubeShape);
g_cubeTransform.parent(g_client.root);

```

## Creating Draw Elements

Every primitive has a draw element constructed for it that describes what material and effect to use when the primitive is rendered. The `createDrawElements()` function creates draw elements for all primitives in the specified shape and adds the draw elements to the draw list associated with the primitive's material. At render time, one draw pass is performed for each draw list, and all draw elements in the lists are rendered.

```
cubeShape.createDrawElements(g_pack, null);
```

## Setting the Render Callback Function

The scene is automatically rendered each time the hardware refreshes the screen. In this example, `setRenderCallback()` sets a callback that updates the cube's transform values each time the scene is rendered. This update makes the cube spin.

```
g_client.setRenderCallback(renderCallback);
```

## Next

[Shapes](#) describes how to specify the vertex information that describes a primitive—indices, normals, texture coordinates. Vertices are specified in arrays that are then loaded into *buffers*. A *stream* contains information on how to access the contents of the buffer.

# Shapes

This chapter describes how to specify 3D geometry using vertex data. It explains how to set up the objects in the transform graph and how to define arrays, buffers, fields, and streams for the geometric data.

The *Hello, Cube* sample discussed in this chapter creates a spinning red cube. To view the code for *Hello, Cube* in a new window, [click here](#).

## Contents

1. [Shapes and Primitives](#)
2. [Buffers](#)
3. [Streams](#)
4. [Code Example: Hello, Cube Colors](#)
5. [Next: Materials](#)

## Shapes and Primitives

A *shape* is a collection of *primitives*, which are the containers for the geometry and other vertex data for the shape. A given primitive can reference only a single material. The vertex data itself is stored in regular JavaScript arrays. These arrays are added to O3D *buffers*, described in the following section.

## Buffers

A *buffer* is a collection of vertex data. Logically, a buffer is divided into fields, one field for each data type in the buffer. A *field* specifies the type of values it contains (`FloatField`, `UInt32Field`, or `UByteNField`) and the *number of components* in each logical unit. For example, a buffer that contains vertex positions would contain a single field of type `FloatField`, with 3 components (*x*, *y*, and *z*):

```
var positionsField = positionsBuffer.createField('FloatField', 3);
```

In addition to positions, buffers can store colors, normals, texture coordinates, tangents, and other per-vertex information for the primitive.

A buffer can contain multiple, interleaved types of data—for example, positions and texture coordinates for each vertex in the primitive. In this case, you create a field for each type of data in the buffer. Here is how you would create a buffer to hold both position and texture data:

```
// Create buffers containing the vertex data.  
var vertexBuffer = g_pack.createObject('VertexBuffer');  
var positionsField = vertexBuffer.createField('FloatField', 3);  
var texCoordsField = vertexBuffer.createField('FloatField', 2);
```

```
vertexBuffer.set(interleavedPositionsAndTexcoordArray);
```

## Streams

A field is associated with a *stream*, which tells the system how to interpret the buffer data. A stream associates a field with the input to a vertex shader, as follows:

Variable	Description	Values
<i>semantic</i>	Specifies what semantic this stream will be mapped to in the vertex shader.	POSITION, NORMAL, TANGENT, BINORMAL, COLOR, TEXCOORD
<i>semantic_index</i>	Corresponds to the number on the semantic in the shader. In other words, a stream with a semantic of NORMAL and an index of 3 corresponds to the shader semantic NORMAL3.	A number from 0 to $n$
<i>field</i>	The field that supplies the data for this stream.	FloatField, UByteNField, UInt32Field
<i>start_index</i>	The first element to use in the field.	A number from 0 to $n$

Shaders use the *semantic* to find the appropriate vertex stream to process (POSITION, NORMAL, TEXCOORD, and so on).

All the vertex streams for a given primitive are associated with one *stream bank*.

## Index Buffer: A Special Case

The index buffer is associated directly with the primitive. (It is not part of a stream bank.) This buffer describes how to connect the triangle indices to create the geometry of the primitive. The index buffer is treated separately from the other buffers so that the vertex data can be re-indexed in different ways.

The index buffer is optional. If you do not provide one, the primitive is not indexed.

Here is the code that creates the index buffer, sets the indices array, and associates the `indexBuffer` with the primitive:

```
var indexBuffer = g_pack.createObject('IndexBuffer');
indexBuffer.set(indicesArray);
cubePrimitive.indexBuffer = indexBuffer;
```

[Back to top](#)

## Code Example: Hello, Cube Colors

This section describes the general structure of *Hello, Cube Colors*. The major tasks are as follows:

1. Create the shape and primitive(s).
2. Specify the vertex data.
3. Add the data to a buffer, specifying fields.
4. Set vertex streams and index buffer to the primitive.

## Create the Shape, Primitive, and Stream Bank

The `createCube()` function creates the shape, primitive, and stream bank and associates the primitive with the shape and stream bank. It also assigns a material to the primitive, as follows:

```
function createCube(material) {  
    var cubeShape = g_pack.createObject('Shape');  
    var cubePrimitive = g_pack.createObject('Primitive');  
    var streamBank = g_pack.createObject('StreamBank');  
    cubePrimitive.material = material;  
    cubePrimitive.owner(cubeShape);  
    cubePrimitive.streamBank = streamBank;  
    .  
    .  
    .
```

Later, this branch of objects will be attached to the transform graph.

## Specify the Vertex Data

The *Hello, Cube* example specifies a cube primitive using an array of 8 vertices that are referenced by indices in an index buffer that defines triangles. The first 3 points in the indices array are used to create the first triangle, the second 3 points are used to create the second triangle, and so on.

### Specify Primitive Attributes

The cube in this example is constructed from 12 triangles, 2 for each face. First, the attributes for the primitive (`cubePrimitive`) are set as follows:

```
cubePrimitive.primitiveType = g_o3d.Primitive.TRIANGLELIST;  
cubePrimitive.numberPrimitives = 12; // 12 triangles  
cubePrimitive.numberVertices = 8; // 8 vertices in total  
cubePrimitive.createDrawElement(g_pack, null); // Create the draw element for  
this primitive.
```

### Specify the Positions

Here is how you create an array of vertices for the 8 corners of the cube in this example:

```
var positionArray = [  
    -0.5, -0.5, 0.5, // vertex 0  
    0.5, -0.5, 0.5, // vertex 1  
    -0.5, 0.5, 0.5, // vertex 2  
    0.5, 0.5, 0.5, // vertex 3  
    -0.5, 0.5, -0.5, // vertex 4  
    0.5, 0.5, -0.5, // vertex 5  
    -0.5, -0.5, -0.5, // vertex 6  
    0.5, -0.5, -0.5 // vertex 7  
];
```

### Add the Array to a Buffer

The `positionArray` is now ready to add to the `positionsBuffer`. First, though, you need to create a field within the buffer to describe the type of data (`FloatField`) and the number of components in each logical unit (3):

```
// Create buffers containing the vertex data.
var positionsBuffer = g_pack.createObject('VertexBuffer');
var positionsField = positionsBuffer.createField('FloatField', 3);
positionsBuffer.set(positionArray);
```

## Associate a Field with a Vertex Shader Input

Now, you're ready to set a vertex stream on the stream bank to supply data to a vertex shader. The `setVertexStream()` function associates fields with vertex shader inputs:

```
// Associate the positions Buffer with the StreamBank.
streamBank.setVertexStream(
    g_o3d.Stream.POSITION, // semantic: This stream stores vertex positions
    0,                     // semantic index: First (and only) position stream
    positionsField,        // field: the field this stream uses.
    0);                   // start_index: How many elements to skip in the field.
```

## Specify the Index Data

Here is how you create an array of indices that specifies how to connect the vertices in the `positionArray` to form the 6 faces of the cube (each face is composed of 2 triangles):

```
var indicesArray = [
    0, 1, 2, // face 1
    2, 1, 3,
    2, 3, 4, // face 2
    4, 3, 5,
    4, 5, 6, // face 3
    6, 5, 7,
    6, 7, 0, // face 4
    0, 7, 1,
    1, 7, 3, // face 5
    3, 7, 5,
    6, 0, 4, // face 6
    4, 0, 2
];
```

As described earlier in [Index Buffer: A Special Case](#), the indices array is special. It is added directly to the `IndexBuffer`, which is set on the primitive directly.

```
var indexBuffer = g_pack.createObject('IndexBuffer');
indexBuffer.set(indicesArray);
cubePrimitive.indexBuffer = indexBuffer;
```

## Next: Materials

[Materials](#) describes how to create a material, an effect, and a shader. It also introduces you to [\*transform semantics\*](#), which are based on Nvidia's Standard Annotations and Semantics (SAS) and are used by O3D shaders to describe standard transformation matrices.

# Materials

This chapter describes how to assign a material to a shape primitive in the transform graph and how to assign an effect to a material. It also covers how to create parameters on materials that are required by the shader, and it introduces the concept of *transform semantics*, described in detail in [Shading Language](#).

The *Hello, Cube Colors* sample creates a red cube with a button that allows the user to change the color of the cube. (To view the code for *Hello, Cube Colors* in a new window, [click here](#)).

## Contents

1. [Creating the Shape and Primitive](#)
2. [Creating the Material and Setting Parameters](#)
  - [SAS Parameters](#)
  - [Non-SAS Parameters](#)
  - [Custom Semantics](#)
3. [Setting Up the Transform Graph](#)
4. [Render Callback Function](#)
5. [See Also: Instance Override Sample](#)
6. [Next: Textures](#)

## Creating the Shape and Primitive

In the *Hello, Cube Colors* sample, the `createCube()` function creates the shape, primitive, and stream bank objects. It then constructs the cube primitive using arrays, buffers, streams, and stream banks, as discussed in [Shapes](#). Here is the code for creating the objects and setting up the relationships between them:

```
function createCube(material) {  
    var cubeShape = g_pack.createObject('Shape');  
    var cubePrimitive = g_pack.createObject('Primitive');  
    var streamBank = g_pack.createObject('StreamBank');  
  
    cubePrimitive.material = material;  
    cubePrimitive.owner(cubeShape);  
    cubePrimitive.streamBank = streamBank;  
    //followed by code that sets up the arrays, buffers, streams, and stream banks  
(see chapter on "Shapes")  
    .  
    .  
    .  
}
```

## Creating the Material and Setting Parameters

The next step is to create a material, set its draw list, and assign an effect to it. The `cubeMaterial` is set to the performance draw list, which handles draw elements for primitives that have opaque materials.

```

// Create a Material for the mesh.
var cubeMaterial = g_pack.createObject('Material');
// Set the material's drawList.
cubeMaterial.drawList = viewInfo.performanceDrawList;

// Apply our effect to this material. The effect tells the 3D hardware
// which shaders to use.
cubeMaterial.effect = cubeEffect;

// Create an O3D Param on the material for every uniform parameter used by the
// shader.
cubeEffect.createUniformParameters(cubeMaterial);

// Get the color parameter from the material and set its value to red.
g_cubeColorParam = cubeMaterial.getParam('color');
g_cubeColorParam.value = [1, 0, 0, 1];

```

## Parameters

A shader defines a number of parameters whose values are supplied by objects in the transform graph (usually by the Material) or by O3D, depending on the whether the parameter falls into one of the following two general categories:

- Parameters that are associated with *standard semantics* (see the section on [SAS Parameters](#))
- Parameters without standard semantics (see the section on [Non-SAS Parameters](#))

Shader parameters are defined at the start of the shader code, contained in the <textarea> element at the end of the sample program. The *Hello, Cube Colors* example declares two parameters (`worldViewProjection` and `color`):

```

<textarea id="effect">
// World View Projection matrix that will transform the input vertices
// to screen space.
float4x4 worldViewProjection : WorldViewProjection;
// This specifies the color of the triangles. A Param with the same name will
// be created on the material that uses the effect.
float4 color;
...

```

When the parameter name is followed by a colon (:) and another string, the string is a *semantic* that explains how to use this parameter value. If the string is contained in the list defined in [Standard Annotations and Semantics \(SAS\)](#), then it is treated specially by the system.

## SAS Parameters

If the shader specifies an SAS *semantic* value for the parameter, O3D automatically computes the value for this parameter and supplies it to the shader behind the scenes. This is the case for the `worldViewProjection` parameter:

```
float4x4 worldViewProjection : WorldViewProjection;
```

In this piece of code, the colon (:) is followed by this parameter's *semantic*, which is **WorldViewProjection**. The [Shading Language](#) chapter describes the meaning of all standard transform semantics used by O3D. For example, it explains that a `WorldViewProjection` is a concatenation of the world, view, and projection matrices. The transform semantics used by O3D are

based on NVidia's Standard Annotations and Semantics (SAS) which describe standard transformation matrices.

If you explicitly create an SAS parameter, you are also responsible for setting the value's parameter; the system will not supply a value in this case. For example, if you explicitly create a `worldViewMatrix` parameter, O3D will not specify its value:

```
material.createParam('worldViewMatrix', 'ParamMatrix4');
```

## Non-SAS Parameters

The `createUniformParameters()` function creates a parameter on the material for every non-SAS parameter required by the shader. Values for these parameters must be explicitly set within your program.

In this example, the `color` parameter is a non-SAS parameter. The function call `cubeEffect.createUniformParameters(cubeMaterial);` creates this parameter on the material. Parameters such as `color`, `intensity`, `shininess`, and so on are called *uniform parameters* because they are the same for every vertex and pixel that uses this material.

The value for the `color` parameter is set with the call `g_cubeColorParam.value = [1, 0, 0, 1];`. When this program is running, the user can change the value of this parameter by pressing the button on the page. This action invokes the `changeColor()` function.

## Custom Semantics

There may also be cases where you want to supply your own semantic, for use in a custom, high-level framework. For example:

```
float4x4 decoration : flower;
```

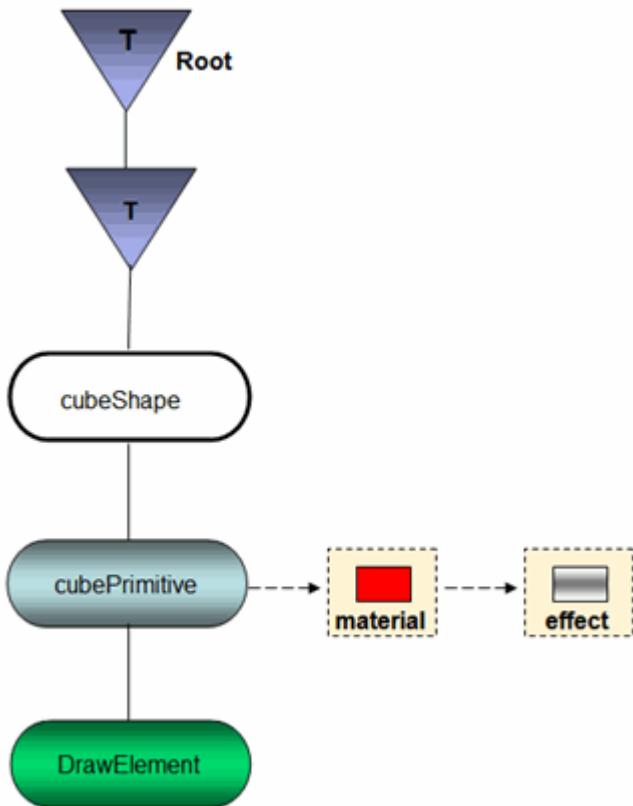
The semantic of `decoration` is now `flower`. In this case, O3D will not supply a value for the parameter since it is not a standard semantic, but you can look up the semantic with `effect.getParameterInfo()`.

## Setting Up the Transform Graph

The `initStep2()` function calls `createCube()` to create the shape object, the cube triangle mesh, and the material. Then `initStep2()` constructs the small transform graph, which contains the client root object, a Transform, and the `cubeShape`.

```
// Create the Shape for the cube mesh and assign its material.  
var cubeShape = createCube(cubeMaterial);  
// Create a new transform and parent the Shape under it.  
g_cubeTransform = g_pack.createObject('Transform');  
g_cubeTransform.addShape(cubeShape);  
// Parent the cube's transform to the client root.  
g_cubeTransform.parent(g_client.root);
```

Here is the Transform graph for this sample:



## Render Callback Function

The scene is automatically rendered each time the hardware refreshes the screen. In this example, the `setRenderCallback()` function calls `renderCallback()`, which causes the cube to spin by updating the transform values.

```
// Set our render callback for animation.
// This sets a function to be executed every time a frame is rendered.
g_client.setRenderCallback(renderCallback);
```

## See Also: Instance Override Sample

The [`instance-override.html`](#) sample creates one sphere object that is instanced multiple times with a different Transform for each instance. A `colorMult` parameter is set on each Transform. This is an efficient technique to give each sphere a different color without creating a thousand different spheres.

## Next: Textures

[Textures](#) describes how to set up a texture sampler. It explains texture coordinates, address modes, magnification and minification filters and mip filtering using mipmaps.

# Textures

## Introduction

This chapter describes how to create a texture sampler and apply it to a shape. In its simplest form, a *texture* is an image (a 2D array of pixel information) that is applied to the surface of a 3D shape. The image can be in TGA, JPEG, PNG, or DDS format. Using a *texture sampler*, you set a number of states to describe how to apply the texture to the shape.

The *Hello, Cube Textures* sample creates a cube with a texture applied to each face. The user can specify the URL of the texture to use. (To view the code for *Hello, Cube Textures* in a new window, [click here](#)).

## About Textures

O3D uses the letter  $u$  for the horizontal texture coordinate and  $v$  for the vertical texture coordinate. Texture coordinates  $(u, v)$  range from  $(0.0, 0.0)$  in the lower left of the image to  $(1.0, 1.0)$  in the upper right, as shown here:



In O3D, a texture can also be a 3D cube. A *cube texture* stores images for the six faces of a cube and is addressed by three texture coordinates ( $u$ ,  $v$ , and  $w$ ). The effect of applying a cube texture to a shape is roughly analogous to shrink-wrapping the texture onto a shape that is enclosed by the cube.

## Texture Samplers

A texture sampler encapsulates a reference to a `texture` object and a set of states that define how the texture bitmap is applied to a surface. Sampler states can be set using parameters defined on a `Sampler` object or specified directly using one of the convenience methods supplied by the `Sampler`

class. The *Hello, Cube Textures* example sets these values using convenience methods. The `Sampler` class defines the following attributes (default values are in parentheses):

- `addressModeU` (WRAP)
- `addressModeV` (WRAP)
- `minFilter` (LINEAR)
- `magFilter` (LINEAR)
- `mipFilter` (POINT)
- `borderColor` (Float4(0,0,0,0))
- `maxAnisotropy` (1)

## Address Mode

The `addressModeU` and `addressModeV` attributes specify what happens when the texture coordinates fall outside of the 0.0 to 1.0 range defined for the texture. For example, in Texture Samplers, shown below on this page, the texture coordinates range from 0.0 to 2.0 in both *u* and *v*. A different address mode can be specified for the *u* and *v* directions. The choices are as follows:

Value	Meaning
WRAP	Repeat the texture to fill in the area (default)
MIRROR	Repeat the texture, inverting it when it crosses a <i>uv</i> boundary (0.0 or 1.0 in either direction)
CLAMP	The last row of pixels in the texture is repeated to cover the rest of the face
BORDER	A border color is used for the pixels that fall outside the range of the texture coordinates specified

## Minification Filter

The minification filter (`minFilter`) specifies how to apply the texture if the area to be textured has fewer pixels than the texture (that is, the texture needs to be shrunk to fill the area). Possible filtering types for `minFilter` are as follows:

Value	Meaning
POINT	Use the closest pixel.
LINEAR	Perform a linear interpolation between neighboring pixels and use the result.
ANISOTROPIC	Stretch the texture according to its orientation in screen space (may be more in one direction than in the other). See <a href="#">Anisotropy</a> , below.

If the `minFilter` is set to POINT or LINEAR, a mipmap filter (`mipFilter`) can be used to control the interpolation of texture values. If the `mipFilter` is set to NONE, it is ignored. See [Mipmap Filter](#).

## Mipmap Filter

Creating minified versions of textures is a more difficult process than creating magnified versions of textures. More care needs to be taken about which pixels are cut from the image to prevent aliasing (jagged diagonal boundaries between color areas). *Mipmapping* refers to the process of creating a set of reduced textures to use in place of the original full-size texture. The usual technique is to start with the original texture, then create another texture half its size, and continue this process until the reduced texture is only one pixel in size. When a minification filter is required for the texture, the mipmap level is selected according to the value for the `mipFilter` as follows:

<b>Value of mipFilter</b>	<b>Which Mipmap Is Used</b>
NONE	Don't use a mipmap at all.
POINT	Use the mipmap level that is closest to the size of the triangle on the screen. Within that level, filter the pixels based on the minification filter (minFilter).
LINEAR	Start with the two mipmap levels that are closest in size to the triangle on the screen. Filter each level with the minFilter. Then perform a linear interpolation between these two levels to produce the final color values.

## Magnification Filter

The magnification filter (magFilter) specifies how to apply the texture if the area to be textured contains more pixels than the texture (that is, the texture needs to be stretched to fill the area). This magnification can result in a blurred image. Possible filtering types for magFilter are as follows:

<b>Value</b>	<b>Meaning</b>
POINT	Use the closest pixel.
LINEAR	Perform a linear interpolation between neighboring pixels and use the result.

## Border Color

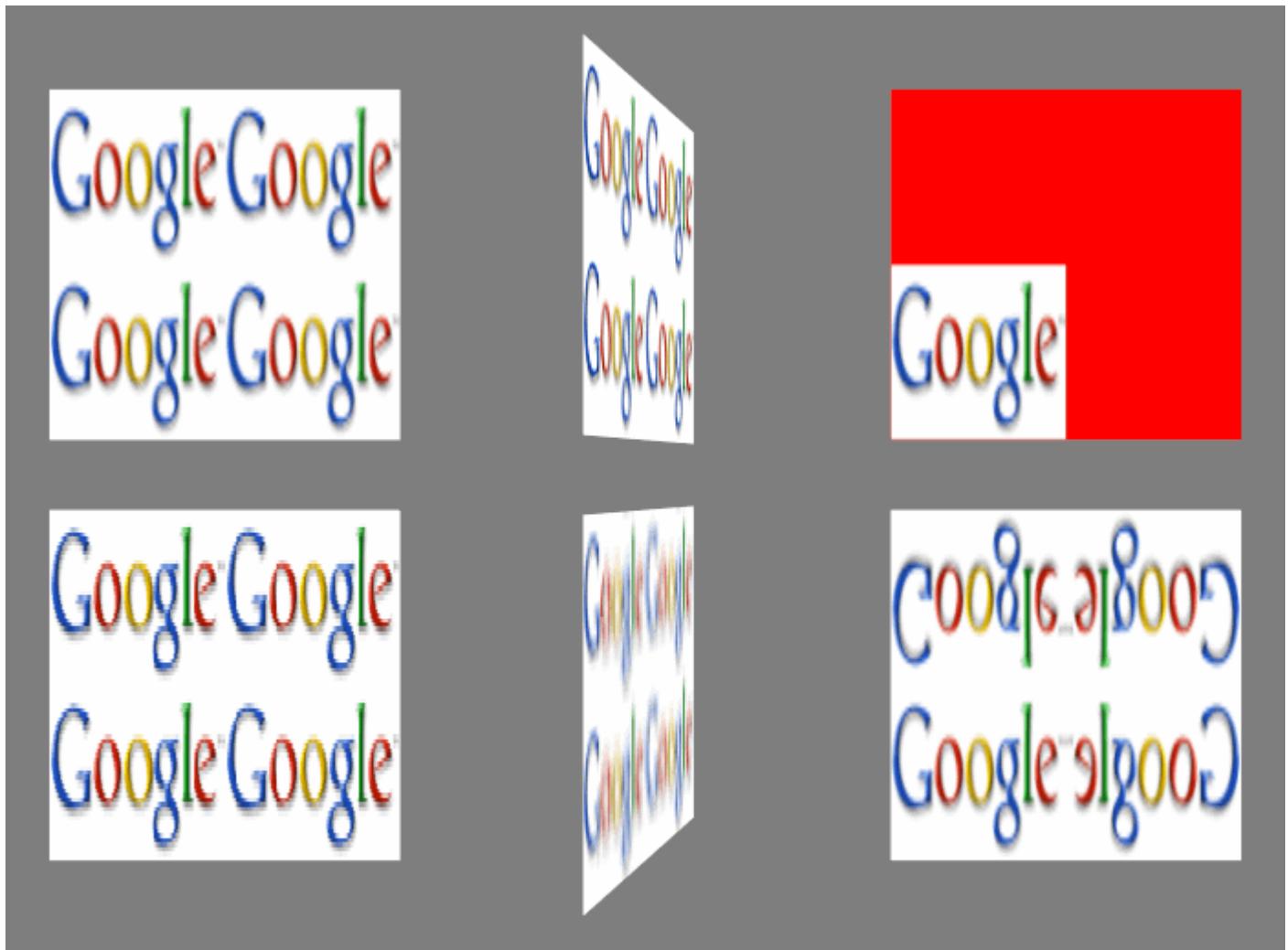
When the address mode is set to BORDER, the borderColor is used for any pixels that fall outside of the 0.0 to 1.0 range defined for the texture coordinates. The top right texture in the [Texture Samplers](#) example, shown below, illustrates using a border color of red.

## Anisotropy

When the value of minFilter is ANISOTROPIC, a maxAnisotropy value specifies the degree of anisotropy. Anisotropic filtering takes into account the distance of the texture pixels from the viewer as well as the angle at which the texture is being viewed.

## Examples

The following figure shows the output of the [Texture Samplers](#) example. The figure is followed by a table that lists the corresponding values set for each nondefault texture sampler variable.



### Top Left

- addressModeU = Sampler.WRAP
- addressModeV = Sampler.WRAP
- minFilter = Sampler.LINEAR
- magFilter = Sampler.LINEAR
- mipFilter = Sampler.POINT
- borderColor = Float4(0, 0, 0, 0)
- maxAnisotropy = 1

### Top Middle

- minFilter = Sampler.ANISOTROPIC
- maxAnisotropy = 4

### Top Right

- addressModeU = Sampler.BORDER
- addressModeV = Sampler.WRAP
- borderColor = Float4(1, 0, 0, 1)

### Bottom Left

- minFilter = Sampler.POINT

### Bottom Middle

Uses default (LINEAR) filtering;  
compare to texture above

### Bottom Right

- addressModeU = Sampler.MIRROR

- addressModeV = Sampler.MIRROR

## A Detailed Look at "Hello, Cube Textures"

The [Hello, Cube Textures](#) program builds on the *Hello, Cube Materials* program by specifying a texture sampler for the cube's material. The new tasks related to textures and texture samplers are as follows:

1. Define the texture coordinates.
2. Create the texture sampler and set its states.
3. Obtain the texture from the specified URL.
4. Create the shaders to access the texture.

The following sections describe these tasks in more detail.

### Step 1: Defining the Texture Coordinates

Here are the steps and code for specifying the texture coordinates, adding them to an array, and adding the array to a buffer. The `setVertexStream()` function specifies how to read the buffer and sets the buffer to a vertex stream.

- Specify the (u,v) texture coordinates for each vertex and add them to an array (`texCoordsArray`):

```
var texCoordsArray = [
  0, 0,
  1, 0,
  1, 1,
  0, 1,
  0, 0,
  1, 0,
  1, 1,
  0, 1,
  1, 1,
  0, 1,
  0, 0,
  1, 0,
  0, 0,
  1, 0,
  1, 1,
  0, 1,
  0, 0,
  1, 0,
  1, 1,
  0, 1,
  0, 0,
  1, 0,
  1, 1,
  0, 1
];
```

- Create a vertex buffer to hold the texture coordinates (`texCoordsBuffer`). Set the texture coordinates array to this buffer:

```
var texCoordsBuffer = g_pack.createObject('VertexBuffer');
```

```
var texCoordsField = texCoordsBuffer.createField('FloatField', 2);
texCoordsBuffer.set(texCoordsArray);
```

- Call `setVertexStream()` to associate the texture coordinates buffer with the primitive's stream bank. This function call sets the semantic for the texture coordinates vertex stream (`TEXCOORD`) as well as the other information about how to read the data in the stream:

```
streamBank.setVertexStream(
    g_o3d.Stream.TEXCOORD, // semantic
    0,                    // semantic index
    texCoordsField,       // field
    0);                  // start_index
```

Note that the way this program specifies the vertices for the cube differs from how they are specified in the *Hello, Cube Materials* program. Two values are specified for each vertex in the triangles that make up the cube: a position and a texture coordinate. The vertices between adjacent cube faces cannot be shared because, although their positions are the same, their texture coordinates are different. This example therefore creates 24 vertices in the `positionArray`, 4 for each of the cube's 6 faces. Then the coordinates in the `texCoordsArray` are matched, one for one, to the vertices in the `positionArray`.

## Step 2: Creating the Texture Sampler

Here are the steps and code for creating the texture sampler and setting its state.

- Call `Effect.createUniformParams()` to create a material parameter for each of the effect's uniform parameters (that is, parameters that apply to the primitive as a whole). In this example, there is one uniform parameter, the texture sampler:

```
cubeEffect.createUniformParameters(cubeMaterial);
```

Alternatively, you could create the uniform parameters required by the shader using explicit function calls.

- Get the material's sampler parameter:

```
var samplerParam = cubeMaterial.getParam('texSampler0');
```

- Create the texture sampler and set its states. The sampler's minification filter (`minFilter`) is set to `ANISOTROPIC`. This value improves the quality of the texture when it is viewed at an angle:

```
g_sampler = g_pack.createObject('Sampler');
g_sampler.minFilter = g_o3d.Sampler.ANISOTROPIC;
g_sampler.maxAnisotropy = 4;
```

- Assign this texture sampler to the material's sampler parameter:

```
samplerParam.value = g_sampler;
```

## Step 3: Obtaining the Texture

Here are the steps and code for initializing the value of the texture's URL and changing this value when the user specifies a new texture:

- Initialize the texture URL (at the beginning of the `initStep2()` function).

```
var path = window.location.href;
var index = path.lastIndexOf('/');
path = path.substring(0, index+1) + 'assets/texture_b3.jpg';
var url = document.getElementById("url").value = path;
```

- Whenever the user specifies a new texture, the `changeTexture()` function fetches a new texture file.

```
function changeTexture() {
    var textureUrl = document.getElementById('url').value;
    try {
        o3djs.io.loadTexture(g_pack, textureUrl, function(texture) {
            // Remove the currently used texture from the pack so that when it's
            not
            // referenced anymore, it can get destroyed.
            if (g_sampler.texture)
                g_pack removeObject(g_sampler.texture);

            // Set the texture on the sampler object to the newly created texture
            // object returned by the request.
            g_sampler.texture = texture;
            // We can now safely add the cube transform to the root of the
            // transform graph since it now has a valid texture. If the transform
            // is already parented under the root, the call will have no effect.
            g_cubeTransform.parent = g_client.root;
            .
            .
            //Error handling calls here.
    }
}
```

This function uses the utility `util.loadTexture()` to load the texture from the specified URL. The utility, in turn, calls the `createFileRequest()` function. By default, `createFileRequest()` creates a set of mipmaps when it loads a texture. If you know that you will not use mipmapping, you can make explicit calls to load the texture file and specify FALSE for the `generateMipmaps` attribute of the `FileRequest`.

- In the `<body>` of the HTML page, add the user input box and button that allows the user to specify the URL for a new texture:

```
<body>
<h1>Hello Cube: Textures</h1>
This example shows how to texture map a cube using an image fetched from a
URL.
<br/>
<!-- Start of O3D plugin -->
<div id="o3d" style="width: 600px; height: 600px;"></div>
<!-- End of O3D plugin -->
<br />
Image URL: <input type="text" id="url" size="100">
<input type="button" onclick="changeTexture();" value="Update Texture"><BR>
```

## Step 4: Creating the Shaders

In this example, the `<textarea>` element contains the code for the vertex and pixel shaders. The

input attributes for the vertex shader are `position`, which has a semantic of `POSITION`, and `tex`, which has a semantic of `TEXCOORD0`. The vertex shader transforms the `POSITION` vertices into homogeneous clip space. Here, the positions are contained in the `positionsBuffer`, and the texture coordinates are contained in the `texCoordsBuffer`. The vertex shader passes the texture coordinate values (`TEXCOORD0`) through unchanged (`output.tex = input.tex;`). The pixel shader then looks up each pixel in the primitive and returns the color of the corresponding bit in the texture map. Here is the code for the vertex and pixel shaders:

```
<textarea id="effect">
    // World View Projection matrix that will transform the input vertices
    // to screen space.
    float4x4 worldViewProjection : WorldViewProjection;
    // The texture sampler is used to access the texture bitmap in the fragment
    // shader.
    sampler texSampler0;
    // input for our vertex shader
    struct VertexShaderInput {
        float4 position : POSITION;
        float2 tex : TEXCOORD0; // Texture coordinates
    };
    // input for our pixel shader
    struct PixelShaderInput {
        float4 position : POSITION;
        float2 tex : TEXCOORD0; // Texture coordinates
    };
    /**
     * The vertex shader simply transforms the input vertices to screen space.
     */
    PixelShaderInput vertexShaderFunction(VertexShaderInput input) {
        PixelShaderInput output;
        // Multiply the vertex positions by the worldViewProjection matrix to
        // transform them to screen space.
        output.position = mul(input.position, worldViewProjection);
        output.tex = input.tex;
        return output;
    }
    /**
     * Given the texture coordinates, our pixel shader grabs the corresponding
     * color from the texture.
     */
    float4 pixelShaderFunction(PixelShaderInput input): COLOR {
        return tex2D(texSampler0, input.tex);
    }
    // Here we tell our effect file *which* functions are
    // our vertex and pixel shaders.
    // #o3d VertexShaderEntryPoint vertexShaderFunction
    // #o3d PixelShaderEntryPoint pixelShaderFunction
    // #o3d MatrixLoadOrder RowMajor
</textarea>
<!-- End of effect -->
```

## Next: Shaders

The [O3D Shading Language](#) section describes how to create vertex and pixel shaders.

# O3D Shading Language

## Introduction

O3D is a system designed around fully programmable GPUs. At its lowest level, all functionality is expressed using programmable shaders, with no fixed-function pipeline. The O3D shading language is supported on all targeted platforms: Windows (with DirectX), Mac OSX (with OpenGL) and Linux (with OpenGL). The language uses features supported by both the **HLSL Shader Model 2** and **Cg** (with the arbvp1/arbsp1 compiler targets).

For O3D programming, both the vertex shader and the pixel shader (the "vertex program" and "fragment program," in OpenGL terminology) are contained in the same file. In this way, the two shaders can share data structure declarations, uniform parameters, and so on.

This document covers the following material:

- Standard annotations and semantics (SAS), which allow O3D programs to pass values for standard transform parameters in to programmable shaders
- Differences between the O3D shading language and the HLSL and Cg shading languages.

## Standard Annotations and Semantics (SAS)

In order to make programmable shaders easier to use, the O3D plug-in supports NVidia's Standard Annotations and Semantics (SAS), specifically with respect to the Transform semantics. When setting up an effect and a material, it is common to call

`effect.createUniformParameters(material)`. This call will create parameters on the material for all uniform variables located in the `.fx` source. However, if any matrix parameters contained in the shader are tagged with one of the 24 predefined standard transform semantics (see table below), they will be treated specially. In particular, no explicit parameter will be created on the material. This task will be handled internally by O3D. The value of the parameter will be some combination of the world, view, and projection matrices, as described below.

The world matrix in O3D is computed to be the concatenation of the current Transform path above the `DrawElement` being rendered (due to instancing, there may be more than one path to a `DrawElement`). The view and projection matrices are retrieved from the `DrawContext` associated with the material's `DrawList` in the current `TreeTraversal`. The `DrawContext` node holds view and projection parameters of `Matrix4` type, which represent the viewing and projection transformations respectively.

In the following table, the Expression column describes the expression used to evaluate the given semantic. It's not a real expression syntax, but rather uses a loose `foo.bar` syntax to mean "the parameter named `bar` on the node named `foo`." `transform` is the `Transform` node immediately above the `DrawElement` being rendered, and `draw_context` is the `DrawContext` associated with the material's `DrawList` through the current `TreeTraversal`. \* means matrix multiplication, `inverse()` is the matrix inverse of its argument, and `transpose()` is the matrix transpose of its argument.

Semantic	Value	Expression
<b>World</b>	The concatenation of the current Transform path above the DrawElement being rendered.	<code>transform.world</code>
<b>WorldInverse</b>	The inverse of the World matrix.	<code>inverse(transform.world)</code>
<b>WorldTranspose</b>	The transpose of the World matrix.	<code>transpose(transform.world)</code>
<b>WorldInverseTranspose</b>	The inverse transpose of the World matrix.	<code>transpose(inverse(transform.world))</code>
<b>View</b>	The Context's View matrix (usually a camera transform).	<code>draw_context.view</code>
<b>ViewInverse</b>	The inverse of the Context's View matrix.	<code>inverse(draw_context.view)</code>
<b>ViewTranspose</b>	The transpose of the Context's View matrix.	<code>transpose(draw_context.view)</code>
<b>ViewInverseTranspose</b>	The inverse transpose of the Context's View matrix.	<code>transpose(inverse(draw_context.view))</code>
<b>Projection</b>	The Context's Projection matrix (usually a perspective or orthographic projection).	<code>draw_context.projection</code>
<b>ProjectionInverse</b>	The inverse of the Projection matrix.	<code>inverse(draw_context.projection)</code>
<b>ProjectionTranspose</b>	The transpose of the Projection matrix.	<code>transpose(draw_context.projection)</code>
<b>ProjectionInverseTranspose</b>	The inverse transpose of the Projection matrix.	<code>transpose(inverse(draw_context.projection))</code>
<b>WorldView</b>	Concatenation of the World and View matrices.	<code>draw_context.view*transform.world</code>
<b>WorldViewInverse</b>	Inverse of the WorldView matrix.	<code>inverse(draw_context.view*transform.world)</code>

<b>WorldViewTranspose</b>	Transpose of the WorldView matrix.	<code>transpose(draw_context.view*transform.world)</code>
<b>WorldViewInverseTranspose</b>	Inverse transpose of the WorldView matrix.	<code>transpose(inverse(draw_context.view*transform.world))</code>
<b>ViewProjection</b>	Concatenation of the View and Projection matrices.	<code>draw_context.projection*draw_context.view</code>
<b>ViewProjectionInverse</b>	Inverse of the ViewProjection matrix.	<code>inverse(draw_context.projection*draw_context.view)</code>
<b>ViewProjectionTranspose</b>	Transpose of the ViewProjection matrix.	<code>transpose(draw_context.projection*draw_context.view)</code>
<b>ViewProjectionInverseTranspose</b>	Inverse transpose of the ViewProjection matrix.	<code>transpose(inverse(draw_context.projection*draw_context.view))</code>
<b>WorldViewProjection</b>	Concatenation of the World, View and Projection matrices.	<code>draw_context.projection*draw_context.view*transform.world</code>
<b>WorldViewProjectionInverse</b>	Inverse of the WorldViewProjection matrix.	<code>inverse(draw_context.projection*draw_context.view*transform.world)</code>
<b>WorldViewProjectionTranspose</b>	Transpose of the WorldViewProjection matrix.	<code>transpose(draw_context.projection*draw_context.view*transform.world)</code>
<b>WorldViewProjectionInverseTranspose</b>	Inverse transpose of the WorldViewProjection matrix.	<code>transpose(inverse(draw_context.projection*draw_context.view*transform.world))</code>

## O3D Shading Language

### Parameter Datatypes

The following table lists the datatypes supported as uniform parameters in O3D shaders and the corresponding types in the O3D JavaScript API:

#### Shading Language Datatype O3D Datatype

float	ParamFloat
-------	------------

float2	ParamFloat2
float3	ParamFloat3
float4	ParamFloat4
float4x4	ParamMatrix4
int	ParamInteger
bool	ParamBoolean
sampler	ParamSampler

## Unsupported Parameter Datatypes

The following Cg/HLSL datatypes are unsupported as uniform parameters. Note that they can still be used as temporaries, varyings, and function parameters inside shaders.

```
int2
int3
int4
bool2
bool3
bool4
float2x2
float2x3
float2x4
float3x2
float3x3
float3x4
float4x2
float4x3
```

Arrays of any type other than `float` are also unsupported.

## Mandatory Comments in O3D Shaders

O3D requires three pieces of information in a special comment format. They should appear as follows:

```
#o3d VertexShaderEntryPoint FunctionName
#o3d PixelShaderEntryPoint FunctionName
#o3d MatrixLoadOrder ( RowMajor | ColumnMajor )
```

where

`VertexShaderEntryPoint`

Indicates the function entry point for the vertex shader. This is the function that is called for each vertex in the current primitive when this shader is active. It can be used for computations that are performed per vertex, such as transformation from world space into clip space, or for per-vertex lighting.

`PixelShaderEntryPoint`

Indicates the entry point for the pixel shader. This is the function that is called for each rasterized pixel in the current primitive while this shader is active. It is used for per-pixel computations, such as texture mapping or per-pixel lighting.

`MatrixLoadOrder`

Indicates how matrix uniform parameters are loaded into the GPU. `RowMajor` indicates that all matrix parameters are loaded in row-major order (DX-style). Use this setting if your shaders specify

matrix/vector multiplications as `mul(vector, matrix)`. `ColumnMajor` indicates that all matrix parameters are loaded in column-major order (OpenGL-style). Use this setting if your shaders specify matrix/multiplications as `mul(matrix, vector)`.

## Unsupported Features

There are several features of FX and CgFX files that are not supported in O3D.

### Technique Blocks

No techniques should be specified in the shader file. Instead, the mandatory comment fields described above should indicate the vertex and pixel shader entry points. The [sample COLLADA Converter](#) can be used to automatically convert CgFX and FX shader files containing techniques to the O3D format.

### Multipass Techniques

FX and CgFX files may contain techniques with more than one pass. The O3D system supports only single-pass shaders and will reject shaders containing more than one pass. Multipass techniques can still be created programmatically in JavaScript using the O3D API.

### Techniques with Compile-time Parameters

Although the O3D system will not accept a shader file containing a technique block, the [sample COLLADA Converter](#) will accept such technique blocks and automatically convert them to the O3D format. However, the Converter will not accept techniques containing compile statements with arguments. For example:

```
VertexShader = compile vs_2_0 std_dp_VS(foo, bar);
```

All arguments to the vertex shader entry point function should be vertex attributes, and all arguments to the pixel shader entry point function should be varying parameters.

## Unsupported Cg features

### Unsized Arrays

Cg allows the user to specify arrays as having no size:

```
uniform float kernel[];
```

This technique allows the programmer to change the size of the array on the fly, and the Cg runtime will automatically recompile the shader when necessary to accommodate the new array size. This technique is not supported in HLSL and hence is prohibited in O3D shaders. To achieve similar functionality in O3D, create the shader string with the size of the array as a replaceable constant string and perform a JavaScript substitution on the shader string at runtime before compiling the shader. For example, in the shader, use this code:

```
uniform float kernel[KERNEL_SIZE];
```

In JavaScript, call this code each time `KERNEL_SIZE` changes:

```
fxString = fxString.replace(/KERNEL_SIZE/, kernelSize);
```

```
var effect = g_pack.createObject('Effect');
effect.loadFromFXString(fxString);
```

## Interfaces

Cg allows the user to specify interfaces or variables whose type can be redefined on the fly. This functionality is not supported in O3D.

## Unsupported HLSL Features

### Implicit Casts in Intrinsic Functions

The HLSL compiler is a bit more forgiving about datatypes than Cg. For example, the HLSL compiler will happily perform a `mul()` of a `float3` vector and a `float4x4` matrix:

```
uniform float4x4 worldViewProjection;
...
float3 v1 = float3(1.0, 2.0, 3.0);
float4 v2 = mul(v1, worldViewProjection);
```

Cg will not perform this operations and requires a conversion—in this case:

```
uniform float4x4 worldViewProjection;
...
float3 v1 = float3(1.0, 2.0, 3.0);
float4 v2 = mul(float4(v1, 1.0), worldViewProjection);
```

All parameters to intrinsic functions in O3D should match the width of their datatypes (`float4x4` to `float4`, `float3x3` to `float3`, etc).

## Scripts

The Script feature of HLSL, used to implement complex rendering effects, is not supported in O3D.

### Functions

The `tex2Dbias()` function is not supported in O3D pixel shaders.

## COLLADAFX

There are actually two paths for using shaders in the sample COLLADA converter: separate shaders, embedded in the COLLADA file itself, or combined vertex and fragment shaders in an FX or CgFX file. The O3D sample converter currently supports a modification of the second option, where both shaders are combined in a separate file, referenced by the COLLADA file, in the special shader format described above.

## Sample COLLADA Converter

As part of the O3D SDK, an executable sample COLLADA Converter can be used to convert an existing COLLADA file into a format supported by O3D. The result is a gzipped tar file containing all assets (texture files, shader files, vertices, skinning data, and animation) as well as a sample format JSON file that contains a representation of the structure of the original COLLADA file.

In addition, the sample Converter will convert shader files in either supported format (FX shaders with a `ps_2_0/vs_2_0` technique, as supported in Max, or CgFX shaders having an `arbfp1/arbvp1` technique, as supported in Maya) to our internal shading language format. It will remove the technique blocks from the shader and insert the entry points into the `VertexShaderEntryPoint` and `PixelShaderEntryPoint` comment formats. It will also set the `MatrixLoadOrder` to `RowMajor` (for DX) or `ColumnMajor` (for OpenGL). Finally, it will remove all sampler state and render state from the shader files and place those states into the sample format JSON file as `O3D State` objects and settings.

## O3D Shading Language Grammar

The following is the ANTLR grammar used by the Converter to recognize O3D-compatible shaders.

```

translation_unit
: ( noise_global_declaration )* EOF
;

noise
: ( COMMENT | WHITESPACE | MULTILINE_COMMENT )*
| LINE_DIRECTIVE
;

global_declaration
: function_declaration
| sampler_declaration
| texture_declaration
| struct_definition
| typedef_definition
| var_declaration
| technique_definition
;

var_declaration
: var_storage_class* var_type_modifier? var_datatype id_declaration semantic?
annotation_list? ('=' initializer)? var_packoffset? var_register_bind? SEMI
;

var_storage_class
: EXTERN
| NOINTERPOLATION
| SHARED
| STATIC
| UNIFORM
| VOLATILE
;

var_type_modifier
: T_CONST
| ROW_MAJOR
| COLUMN_MAJOR;
;

var_datatype
: buffer_type_specifier
| scalar_type_or_string_specifier
| vector_type_specifier
| matrix_type_specifier
;
```

```

| struct_type_specifier
;

var_packoffset
: 'packoffset' LPAREN register_name (DOT ID)? RPAREN
;

var_register_bind
: COLON register_name
;

id_declaration
: ID ( LBRACKET constant_expression RBRACKET )?
;

function_declaration
: function_storage_class?
  function_typeSpecifier ID LPAREN argument_list RPAREN semantic?
  function_body (SEMI)?
;
;

function_storage_class
: INLINE
;

function_typeSpecifier
: scalar_typeSpecifier
| vector_typeSpecifier
| matrix_typeSpecifier
| struct_typeSpecifier
| T_VOID
;
;

semantic
: COLON semanticSpecifier ;
;

param_typeSpecifier
: scalar_typeSpecifier
| vector_typeSpecifier
| matrix_typeSpecifier
| struct_typeSpecifier
| string_typeSpecifier
| sampler_typeSpecifier
;
;

basic_typeSpecifier
: scalar_typeSpecifier
| vector_typeSpecifier
| matrix_typeSpecifier
| string_typeSpecifier
;
;

typedef_definition
: TYPEDEF
;
;

buffer_typeSpecifier
: (BUFFER '<' var_datatype '>' ID)
;
;
```

```

technique_definition
: TECHNIQUE ID annotation_list? '{' ( pass )+ '}' SEMI?
| TECHNIQUE annotation_list? '{' ( pass )+ '}' SEMI?
;

pass
: PASS ID? annotation_list? '{' ( state_assignment )* '}' SEMI?
;

state_assignment [o3d
::PassDeclaration& pass]
: VERTEXSHADER '=' 'compile' ID variable_or_call_expression SEMI
| FRAGMENTSHADER '=' 'compile' ID variable_or_call_expression SEMI
| ID '=' primary_expression SEMI
;

scalar_typeSpecifier
: 'bool'
| 'int'
| 'uint'
| 'half'
| FLOAT
| 'double'
;

scalar_typeOrStringSpecifier
: scalar_typeSpecifier
| string_typeSpecifier
;

vector_typeSpecifier
: 'bool1'
| 'bool2'
| 'bool3'
| 'bool4'
| 'int1'
| 'int2'
| 'int3'
| 'int4'
| 'uint1'
| 'uint2'
| 'uint3'
| 'uint4'
| 'half1'
| 'half2'
| 'half3'
| 'half4'
| 'float1'
| 'float2'
| 'float3'
| 'float4'
| 'double1'
| 'double2'
| 'double3'
| 'double4'
| VECTOR '<' scalar_typeSpecifier ',' DECIMAL_LITERAL '>'
;

```

```

matrix_typeSpecifier
: 'float1x1'
| 'float1x2'
| 'float1x3'
| 'float1x4'
| 'float2x1'
| 'float2x2'
| 'float2x3'
| 'float2x4'
| 'float3x1'
| 'float3x2'
| 'float3x3'
| 'float3x4'
| 'float4x1'
| 'float4x2'
| 'float4x3'
| 'float4x4'
| MATRIX '<' scalarTypeSpecifier ',' DECIMAL_LITERAL ',' DECIMAL_LITERAL '>'
;

stringTypeSpecifier
: STRING
;

samplerDeclaration
: samplerTypeSpecifier idDeclaration
( '=' 'samplerState' '{' samplerStateDeclaration+ '}' )? SEMI
;

samplerStateDeclaration
: TEXTURE '=' '<' ID '>' SEMI
| TEXTURE '=' '(' ID ')' SEMI
| ID '=' initializer SEMI
;

samplerTypeSpecifier
: 'sampler'
| 'sampler1D'
| 'sampler2D'
| 'sampler3D'
| 'samplerCUBE'
| 'sampler_state'
| 'SamplerComparisonState' | 'samplercomparisonstate'
;

textureDeclaration
: textureTypeSpecifier ID semantic? annotationList? SEMI
;

textureTypeSpecifier
: TEXTURE
| TEXTURE1D
| TEXTURE2D
| TEXTURE3D
| TEXTURECUBE
| TEXTURERECT
;

```

```

struct_typeSpecifier
: ID
| struct_definition
| STRUCT ID
;

annotation_list
: '<' annotation* '>'
;

annotation
: basic_typeSpecifier ID '=' initializer SEMI
;

initializer
: expression
| '{' expression ( ',' expression )* '}'
;

register_name
: REGISTER '(' input_register_name | output_register_name ')';
;

input_register_name
: ID DECIMAL_LITERAL
| ID
;

output_register_name
: ID
;

pack_offset
: .+
;

argument_list
: ( param_declaration ( COMMA param_declaration )* )?
;

param_declaration
: param_direction? param_variability? param_typeSpecifier idDeclaration semantic?
;

param_variability
: T_CONST
| UNIFORM
;

param_direction
: T_IN
| T_OUT
| T_INOUT
;

function_body
: LCURLY ( decl_or_statement )* RCURLY
;

decl_or_statement
;

```

```

: assignment_statement
| ( T_CONST )? vector_typeSpecifier init_declarator_list SEMI
| ( T_CONST )? scalar_typeSpecifier init_declarator_list SEMI
| ( T_CONST )? matrix_typeSpecifier init_declarator_list SEMI
| struct_definition( init_declarator_list )? SEMI
| STRUCT ID init_declarator_list SEMI
| init_declarator_list SEMI
| statement
;

init_declarator_list
: init_declarator ( COMMA init_declarator )* ;

init_declarator
: declarator ( ASSIGN expression )? ;

declarator
: ID ( LBRACKET ( constant_expression )? RBRACKET )* ;
;

struct_definition
: STRUCT ( ID )? LCURLY struct_declaration_list RCURLY ID? SEMI
;

struct_declaration_list
: ( struct_interpolation_modifier?
  ( scalar_typeSpecifier | vector_typeSpecifier ) ID
  ( COLON semanticSpecifier )? SEMI )+
;

struct_interpolation_modifier
: T_LINEAR
| CENTROID
| NOINTERPOLATION
| NOPERSPECTIVE
;

semanticSpecifier
: ID
;

statement
: assignment_statement
| post_modify_statement
| pre_modify_statement
| expression_statement
| compound_statement
| selection_statement
| iteration_statement
| jump_statement
| SEMI
;

assignment_statement
: lvalue_expression assignment_operator expression SEMI
;

pre_modify_statement
: pre_modify_expression SEMI
;
```

```

;

pre_modify_expression
: self_modify_operator lvalue_expression
;

post_modify_statement
: post_modify_expression SEMI
;

post_modify_expression
: lvalue_expression self_modify_operator
;

self_modify_operator
: PLUS_PLUS | MINUS_MINUS
;

expression_statement
: expression SEMI
;

compound_statement
: LCURLY (
    init_declarator_list SEMI
    | ( T_CONST )? vector_typeSpecifier init_declarator_list SEMI
    | ( T_CONST )? scalar_typeSpecifier init_declarator_list SEMI
    | struct_definition ( init_declarator_list )? SEMI
    | STRUCT ID init_declarator_list SEMI
    | statement
) *
RCURLY
;

selection_statement
: IF LPAREN expression RPAREN statement ( ELSE statement )?
;

iteration_statement
: WHILE LPAREN expression RPAREN statement
| FOR LPAREN assignment_statement equality_expression SEMI modify_expression
RPAREN statement
| DO statement WHILE LPAREN expression RPAREN SEMI
;

modify_expression
: lvalue_expression assignment_operator expression
| pre_modify_expression
| post_modify_expression
;

jump_statement
: BREAK SEMI
| CONTINUE
| RETURN ( expression )? SEMI
| DISCARD
;

expression

```

```

: conditional_expression
;

assignment_operator
: ASSIGN
| MUL_ASSIGN
| DIV_ASSIGN
| ADD_ASSIGN
| SUB_ASSIGN
| BITWISE_AND_ASSIGN
| BITWISE_OR_ASSIGN
| BITWISE_XOR_ASSIGN
| BITWISE_SHIFTL_ASSIGN
| BITWISE_SHIFTR_ASSIGN
;

constant_expression
: (ID) => variable_expression
| literal_value
;

conditional_expression
: logical_or_expression ( QUESTION expression COLON conditional_expression )?
;

logical_or_expression
: exclusive_or_expression ( OR exclusive_or_expression )*
;

logical_and_expression
: ( NOT )? inclusive_or_expression ( AND ( NOT )? inclusive_or_expression )*
;

inclusive_or_expression
: exclusive_or_expression (BITWISE_OR exclusive_or_expression )*
;

exclusive_or_expression
: and_expression ( BITWISE_XOR and_expression )*
;

and_expression
: equality_expression ( BITWISE_AND equality_expression )*
;

equality_expression
: relational_expression ( (EQUAL|NOT_EQUAL) relational_expression )*
;

relational_expression
: shift_expression ( (LESS|GREATER|LESSEQUAL|GREATEREQUAL) shift_expression )*
;

shift_expression
: additive_expression ( (BITWISE_SHIFTL|BITWISE_SHIFTR) additive_expression )*
;

additive_expression
: multiplicative_expression ( (PLUS|MINUS) multiplicative_expression )*
;

```

```

;

multiplicative_expression
: cast_expression ( (MUL|DIV|MOD) cast_expression )*
;

cast_expression
: LPAREN ID RPAREN postfix_expression
| LPAREN basic_type_specifier RPAREN postfix_expression
| unary_expression
;

unary_expression
: (PLUS|MINUS) unary_expression
| postfix_expression
;
;

postfix_expression
: primary_expression ( postfix_suffix )?
;
;

lvalue_expression
: variable_expression ( postfix_suffix )?
;
;

postfix_suffix
: ( DOT primary_expression )+
;
;

primary_expression
: constructor
| variable_or_call_expression
| literal_value
| LPAREN expression RPAREN
;
;

variable_expression
: ID ( LBRACKET expression RBRACKET )?
;
;

variable_or_call_expression
returns [o3d
::String identifier, o3d
::String arglist]
: ID
(
  ( ( LBRACKET expression RBRACKET )? )
|
  ( LPAREN argument_expression_list RPAREN )
)
;
;

constructor
: ( vector_type_specifier | matrix_type_specifier )
LPAREN expression ( COMMA expression )* RPAREN
;
;

argument_expression_list
: ( expression ( COMMA expression )* )?
;
```

```

;

int_literal
: DECIMAL_LITERAL
;

literal_value
: DECIMAL_LITERAL
| FLOAT_LITERAL
| STRING_LITERAL+
| ( T_FALSE | T_TRUE )
;

float_literal
: FLOAT_LITERAL
;

NOT          : '!' ;
NOT_EQUAL    : '!=' ;
AND          : '&&' ;
LPAREN       : '(' ;
RPAREN       : ')' ;
MUL          : '*' ;
MUL_ASSIGN   : '*=' ;
PLUS         : '+' ;
PLUS_PLUS    : '++' ;
ADD_ASSIGN   : '+=' ;
COMMA        : ',' ;
MINUS        : '-' ;
MINUS_MINUS  : '--' ;
SUB_ASSIGN   : '--=' ;
DIV          : '/' ;
DIV_ASSIGN   : '/=' ;
MOD_ASSIGN   : '%=' ;
COLON        : ':' ;
SEMI         : ';' ;
LESS          : '<' ;
LESSEQUAL    : '<=' ;
ASSIGN        : '=' ;
EQUAL         : '==' ;
GREATER       : '>' ;
GREATEREQUAL : '>=' ;
QUESTION      : '?' ;
LBRACKET     : '[' ;
RBRACKET     : ']' ;
LCURLY        : '{' ;
OR            : '||' ;
RCURLY        : '}' ;
DOT           : '.' ;
BITWISE_NOT  : '~' ;
BITWISE_SHIFTL : '<<' ;
BITWISE_SHIFTR : '>>' ;
BITWISE_AND   : '&' ;
BITWISE_OR    : '|';
BITWISE_XOR   : '^' ;
BITWISE_SHIFTL_ASSIGN : '<<=' ;
BITWISE_SHIFTR_ASSIGN : '>>=' ;
BITWISE_AND_ASSIGN   : '&=' ;
BITWISE_OR_ASSIGN    : '|=' ;

```

```

BITWISE_XOR_ASSIGN      : '^=';

BREAK                  : 'break';
BUFFER                 : 'buffer';
COLUMN_MAJOR           : 'column_major';
CBUFFER                : 'cbuffer';
CENTROID               : 'centroid';
T_CONST                : 'const';
CONTINUE               : 'continue';
DISCARD                : 'discard';
DO                     : 'do';
ELSE                   : 'else';
EXTERN                 : 'extern';
T_FALSE                : 'false';
FLOAT                  : ('f' | 'F') ('l' | 'L') ('o' | 'O') ('a' | 'A') ('t' | 'T'));
FOR                     : 'for';
IF                      : 'if';
T_IN                   : 'in';
INLINE                 : 'inline';
T_INOUT                : 'inout';
T_LINEAR               : 'linear';
MATRIX                 : ('m' | 'M') ('a' | 'A') ('t' | 'T') ('r' | 'R') ('i' | 'I') ('x' | 'X');
NAMESPACE              : 'namespace';
NOINTERPOLATION        : 'nointerpolation';
NOPERSPECTIVE          : 'noperspective';
T_OUT                  : 'out';
RETURN                 : 'return';
REGISTER               : 'register';
ROW_MAJOR              : 'row_major';
SHARED                 : 'shared';
STATEBLOCK              : 'stateblock';
STATEBLOCK_STATE        : 'stateblock_state';
STATIC                 : 'static';
STRING                 : ('s' | 'S') ('t' | 'T') ('r' | 'R') ('i' | 'I') ('n' | 'N') ('g' | 'G');
STRUCT                 : 'struct';
SWITCH                 : 'switch';
TBUFFER                : 'tbuffer';
TEXTURE                :
    ('t' | 'T') ('e' | 'E') ('x' | 'X') ('t' | 'T') ('u' | 'U') ('r' | 'R') ('e' | 'E');
TEXTURE1D               : 'Texture1D';
TEXTURE1DARRAY          : 'Texture1DArray';
TEXTURE2D               : 'Texture2D';
TEXTURE2DARRAY          : 'Texture2DArray';
TEXTURE2DMS              : 'Texture2DMS';
TEXTURE2DMSARRAY        : 'Texture2DMSArray';
TEXTURE3D               : 'Texture3D';
TEXTURECUBE              : 'TextureCUBE';
TEXTURECUBEARRAY         : 'TextureCUBEArray';
TEXTURERECT              : 'TextureRECT';
T_TRUE                  : 'true';
TYPEDEF                 : 'typedef';
UNIFORM                 : 'uniform';
VECTOR                  : ('v' | 'V') ('e' | 'E') ('c' | 'C') ('t' | 'T') ('o' | 'O') ('r' | 'R');
T_VOID                  : 'void';
VOLATILE               : 'volatile';
WHILE                   : 'while';

PASS                   : ('p' | 'P') ('a' | 'A') ('s' | 'S') ('s' | 'S');

```

```

TECHNIQUE      : ('t'||'T') ('e'||'E') ('c'||'C') ('h'||'H')
                  ('n'||'N') ('i'||'I') ('q'||'Q') ('u'||'U') ('e'||'E')
;
;

VERTEXSHADER
: ((('v'||'V') ('e'||'E') ('r'||'R') ('t'||'T') ('e'||'E') ('x'||'X'))
  (((('s'||'S') ('h'||'H') ('a'||'A') ('d'||'D') ('e'||'E') ('r'||'R')) |
    (('p'||'P') ('r'||'R') ('o'||'O') ('g'||'G') ('r'||'R') ('a'||'A') ('m'||'M'))))
;
;

FRAGMENTSHADER
: (((('f'||'F') ('r'||'R') ('a'||'A') ('g'||'G') ('m'||'M') ('e'||'E') ('n'||'N') ('t'||'T')
  ('p'||'P') ('r'||'R') ('o'||'O') ('g'||'G') ('r'||'R') ('a'||'A') ('m'||'M')) |
  (((('p'||'P') ('i'||'I') ('x'||'X') ('e'||'E') ('l'||'L')
    ('s'||'S') ('h'||'H') ('a'||'A') ('d'||'D') ('e'||'E') ('r'||'R'))))
;
;

RESERVED_WORDS
: (((('a'||'A') ('S'||'s') ('m'||'M')) |
  'asm_fragment' |
  'auto' |
  'case' |
  'catch' |
  'char' |
  'class' |
  'const_cast' |
  (((('d'||'D') ('e'||'E') ('c'||'C') ('l'||'L')) |
  'default' |
  'delete' |
  (((('d'||'D') ('w'||'W') ('o'||'O') ('r'||'R') ('d'||'D')) |
  'dynamic_cast' |
  'emit' |
  'enum' |
  'explicit' |
  'fixed' |
  'friend' |
  'get' |
  'goto' |
  'interface' |
  'long' |
  'mutable' |
  'new' |
  'operator' |
  'packed' |
  (((('p'||'P') ('i'||'I') ('x'||'X') ('e'||'E') ('l'||'L')
    ('f'||'F') ('r'||'R') ('a'||'A') ('g'||'G') ('m'||'M') ('e'||'E') ('n'||'N') ('t'||'T')) |
  'private' |
  'protected' |
  'public' |
  'reinterpret_cast' |
  'short' |
  'signed' |
  'sizeof' |
  'snorm' |
  'static_cast' |
  'template' |
  'this' |
  'throw' |
  'try'
;
```

```

| 'typeid'
| 'typename'
| 'union'
| 'unorm'
| 'unsigned'
| 'using'
| (('v'||'V') ('e'||'E') ('r'||'R') ('t'||'T') ('e'||'E') ('x'||'X')
  ('f'||'F') ('r'||'R') ('a'||'A')
  ('g'||'G') ('m'||'M') ('e'||'E') ('n'||'N') ('t'||'T'))
| 'virtual'
;

ID
: ('a'..'z'||'A'..'Z'||'_') ('a'..'z'||'A'..'Z'||'_'||'0'..'9')*
;

DECIMAL_LITERAL
: ('0'..'9')+ ;
;

fragment ESCAPE_SEQUENCE
: '\\' ('b'||'t'||'n'||'f'||'r'||'"'||'\''||'\\')
;

CHARACTER_LITERAL
: '\'' (ESCAPE_SEQUENCE | ~('\\'||'\\')) ) '\''
;

STRING_LITERAL
: '"' (ESCAPE_SEQUENCE | ~('\\'||'"')) )* '"'
;

fragment EXPONENT : ('e'||'E') (PLUS | MINUS)? ('0'..'9')+ ;
;

fragment FLOATSUFFIX : ('f'||'F'||'h'||'H') ;
;

FLOAT_LITERAL
: ('0'..'9')+ '.' ('0'..'9')* (EXPONENT)? (FLOATSUFFIX)?
| '.' ('0'..'9')+ (EXPONENT)? (FLOATSUFFIX)?
;

LINE_DIRECTIVE
: '#line' (' '|'\t')* DECIMAL_LITERAL '\r'? '\n'
| '#line' (' '|'\t')* DECIMAL_LITERAL (' '|'\t')* STRING_LITERAL '\r'? '\n'
;

WHITESPACE
: (' '|'\r'||'\t'||'\u000C'||'\n') { $channel = HIDDEN; }
;

COMMENT
: '//' ~('\n'||'\r')* '\r'? '\n' { $channel = HIDDEN; }
;

MULTILINE_COMMENT
: '/*' (options {greedy=false;} : .)* '*/' { $channel = HIDDEN; }
;
;
```

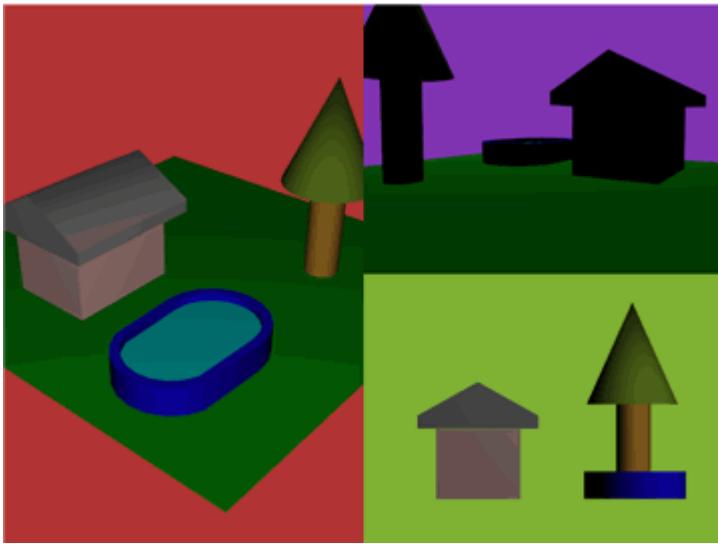
# Rendering

## Overview

The O3D [Technical Overview](#) describes the basic components of the transform graph, the render graph, and the shaders. This chapter goes into more detail on how a typical render graph is created, what happens when a view is rendered, and some of the varied ways you could use this flexible system.

## Multiple Views of the Same 3D Scene

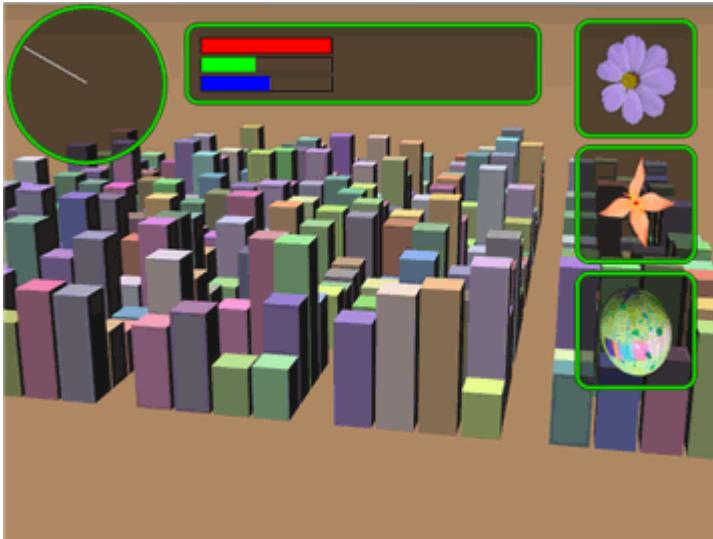
One of the many ways you can use the O3D transform graph and render graph is to specify one transform graph, which contains the 3D scene, and then specify multiple sub-trees within the render graph that display this same scene in different ways using different viewport settings. The [Viewport](#) object specifies which portion of the screen to affect. The [Creating Multiple Views](#) sample sets up three sub-trees within the render graph that display the same transform graph from different viewpoints and with a different background color. Each part of the render graph draws the scene with a different viewport setting using a different [DrawContext](#):



See [Example: Multiple Views](#) for more details on creating multiple views of the same scene.

## One View with Multiple Scenes

You can also build multiple transform graphs that are rendered into the same viewport. The [Creating a Heads-Up Display](#) example demonstrates using a render graph with two sub-trees, each specifying a different view, to display different branches of the transform graph. One [ViewInfo](#) object is used to display the branch of the transform graph that contains the 3D view, and a second [ViewInfo](#) object is used to display the branch of the transform graph with the heads-up display (HUD). The view for the HUD has a higher priority than the view for the 3D objects, so the HUD view is drawn after (that is, on top of) the 3D view. The HUD view also has a [ClearBuffer](#) node that specifies to clear only the depth and stencil buffers—but not the color buffer—before rendering the HUD.



See [Example: Multiple Scenes](#) for more details.

## Utilities for Creating a Render Graph

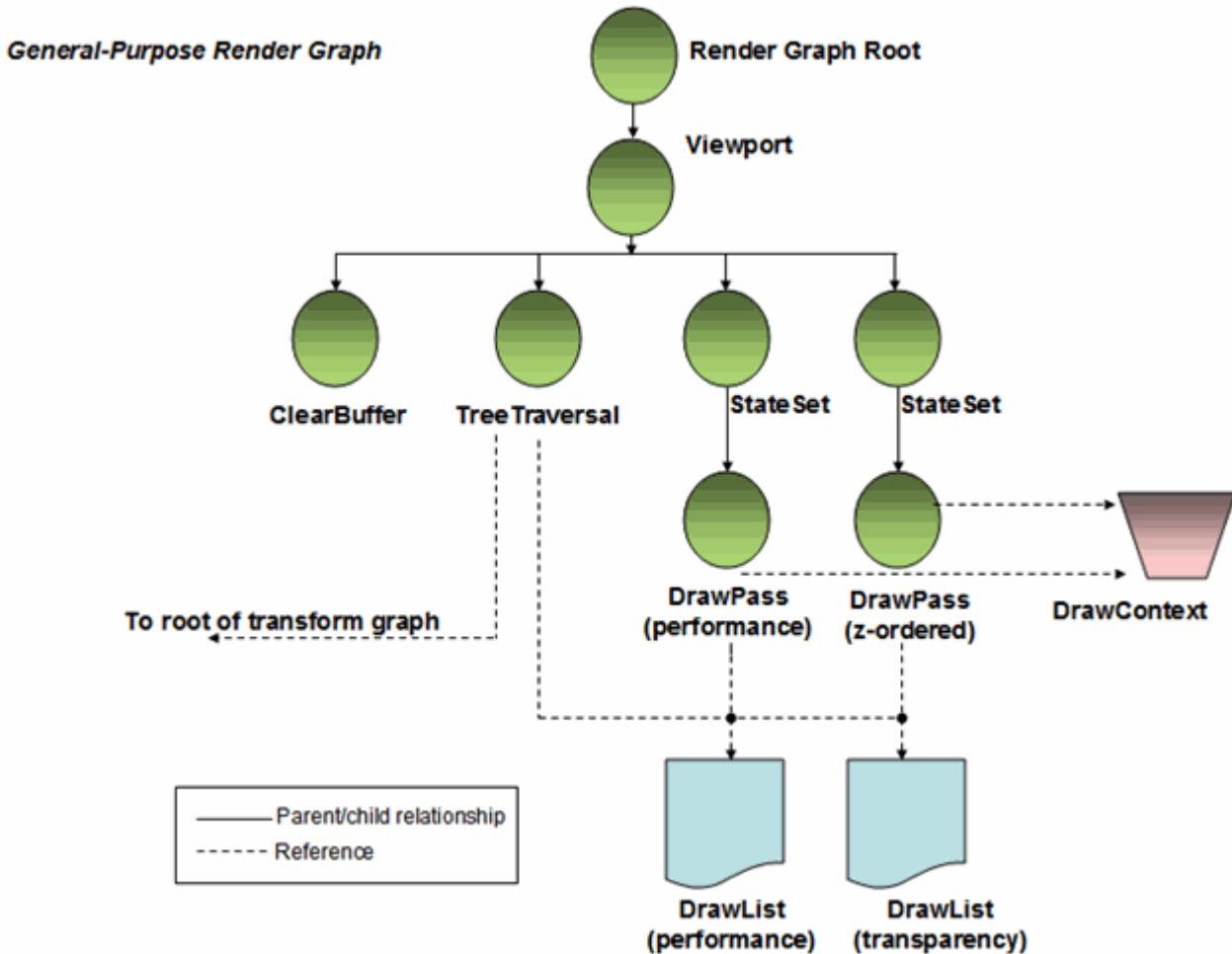
The `o3djs.renderGraph` utility contains code for the `createBasicView()` function, which creates a general-purpose render graph sub-tree that contains the nodes commonly required to render a scene into a viewport. This section explains the arguments to this utility function, which is used in many of the sample programs. Once you understand how it works, you can customize it to suit your needs.

The following table describes the arguments to `createBasicView()`:

Function Arguments	Type	Description
<code>pack</code>	<a href="#">Pack</a>	Pack to manage the objects that are created
<code>treeRoot</code>	<a href="#">Transform</a>	Root transform of the transform graph to be rendered
<code>opt_parent</code>	<a href="#">RenderNode</a>	Parent node for the render graph that is being created
<code>opt_clearColor</code>	Float4 or Array of 4 numbers	Color to clear the view
<code>opt_priority</code>	number	Optional base priority for drawing this set of objects
<code>opt_viewport</code>	Float4 or Array of 4 numbers	Area of the viewport where this scene is to be rendered (see <a href="#">Viewport</a> )

## General-Purpose Render Graph

The `createBasicView()` function returns a [ViewInfo](#) object that contains references to a group of O3D objects used to render a single 3D view. Here is what this render graph looks like:



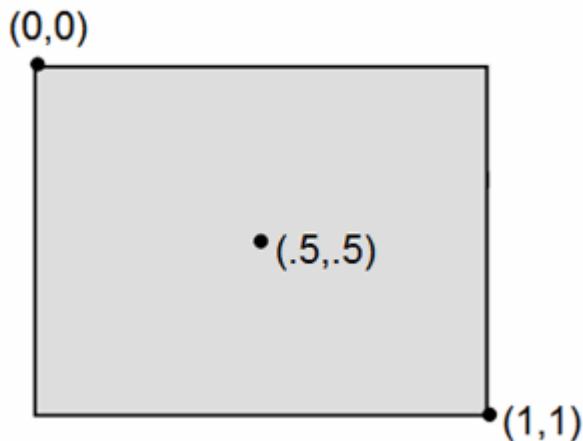
The following paragraphs describe the nodes in the render graph in more detail. The render graph, like the transform graph, is traversed from top to bottom, and then by priority of the render nodes. The system walks the nodes in the render graph each time the screen is refreshed.

Many of the examples have an `onRender()` function that performs certain operations on the transform graph and render graph. In this case, the `onRender()` function is called each time before the render graph is traversed.

## Viewport

A [Viewport](#) sets the rectangular area on the screen where the scene is rendered. The viewport coordinates range from (0,0) in the top left corner of the screen to (1,1) in the bottom right corner. To render a scene into a smaller portion of the screen, pass in the coordinates to the `opt_viewport` argument of `createBasicView()`. If you're creating the render graph manually, create a `Viewport` object with the desired coordinates.

## Viewport coordinates (origin is top left)



## Clear Buffer

The [ClearBuffer](#) node is a [RenderNode](#) that clears the color buffer, z buffer and/or stencil buffer of the current render target. The default specifies to clear all three buffers before rendering.

## Tree Traversal

The [TreeTraversal](#) node causes a complete traversal of the transform graph, beginning with the specified root transform. This node has multiple [DrawList](#) objects registered with it. Each [DrawList](#), in turn, has a [DrawContext](#) registered with it (see diagram of render graph and next section). As the [TreeTraversal](#) node walks the transform tree, whenever it finds a draw element with a material that matches one of its registered [DrawLists](#), the [TreeTraversal](#) node adds that draw element to the [DrawList](#). For example, the render graph created by `createBasicView()` has two draw lists: a performance draw list (draws opaque objects) and a z-ordered draw list (draws transparent objects). Draw elements with materials that are assigned to the *performance* draw list are added to the performance draw list at traversal time. Similarly, draw elements with materials that are assigned to the *z-ordered* draw list are added to the z-ordered draw list at traversal time.

While the [TreeTraversal](#) walks the transform graph, it can optionally cull transforms, primitives, draw elements, and even whole sub-trees of the transform graph. In order to be culled (that is, trimmed), an object must have its `culling` attribute set to TRUE (default is FALSE) and its `boundBox` must be set to something that matches how you want it culled.

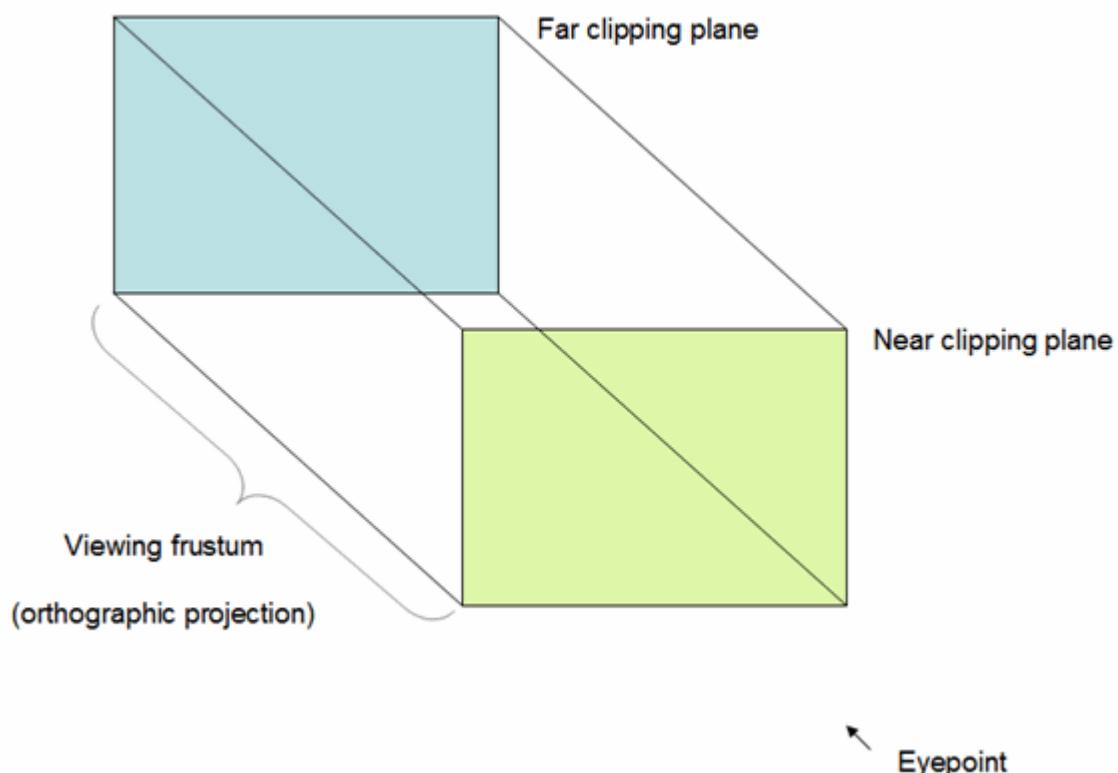
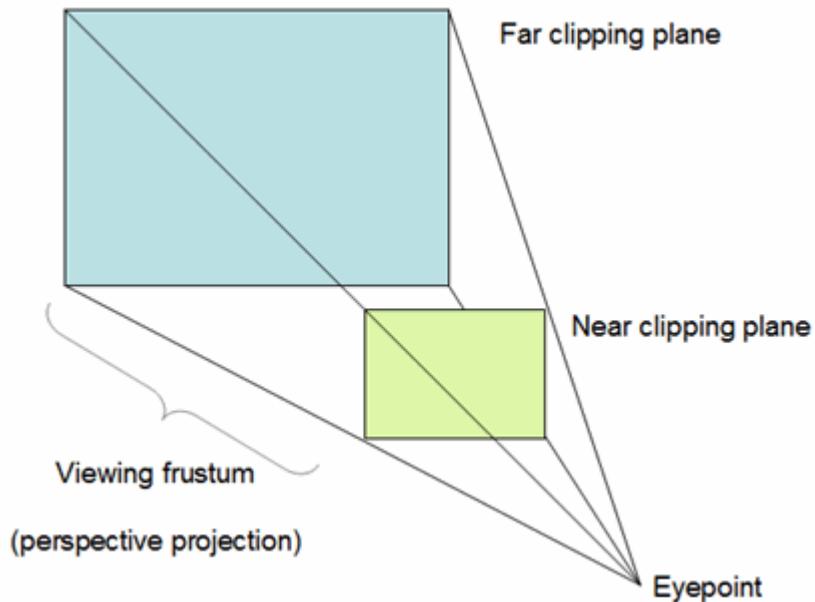
## DrawContext

Each view references a [DrawContext](#) object. The `createBasicView()` function creates a [DrawContext](#) object automatically for its view. Each [DrawPass](#) object contains a pointer to a [DrawContext](#). The [DrawContext](#) uses two matrices to specify how the objects in the scene are viewed:

- **View matrix:** specifies the transformation that converts vertices from world coordinates to view

coordinates.

- **Projection matrix:** specifies the type of "lens" that is viewing the scene, which defines a view volume (also called a *viewing frustum*). Common projections are *perspective*, which has the effect of making objects further from the eyepoint appear smaller (which is how the human eye sees things) and *orthographic*, which projects all objects in their original size, regardless of distance from the eyepoint. A perspective projection is useful when you're trying to simulate how humans view objects. An orthographic projection is useful for making very accurate renderings, where precise measurements are more important than realistic images.



## Performance Draw Pass

After walking the transform graph, O3D returns to the next node in the render graph, the [StateSet](#) element for the performance DrawPass. The StateSet node for the performance draw pass sets no states by default but could be used to set some nondefault behavior—for example to make objects render double-sided. The next node in the traversal is the DrawPass node, which references the DrawContext that sets up the viewing parameters. After that, O3D draws each of the draw elements in the performance draw list.

## Z-Ordered Draw Pass

After rendering the draw elements in the performance draw list, the system traverses the second StateSet element, which specifies state for the z-ordered (transparency) DrawPass. The 2D view in the [Creating a Heads-Up Display](#) sample uses the StateSet node to turn off culling, so that objects flipped horizontally or vertically will still be drawn. It also turns off z-buffering, so that objects further back will be drawn on top of objects in front, even if the objects in front are drawn first. After traversing the StateSet node, O3D traverses the DrawPass node, which references the DrawContext that sets up the viewing parameters. Finally, O3D draws each of the draw elements in the z-ordered draw list.

## Example: Multiple Views

In the [Creating Multiple Views](#) sample, the left viewport takes up the left half of the screen and is specified with the array [0, 0, 0.5, 1], where 0,0 are the screen coordinates for the *top left* corner of this viewport, .5 is the portion of the screen *width* used by this viewport and 1 is the portion of the screen *height* used by this viewport. The draw context in this render graph specifies a perspective projection.

This example uses the [o3djs.renderGraph.createExtraView\(\)](#) utility to specify the size, background color, and priority for the two views on the right side of the screen:

```
for (var yy = 0; yy < 2; yy++) {  
    // Set the view extents to the right half, top or bottom.  
    var viewExtents = [0.5, yy * 0.5, 0.5, 0.5];  
    // Give each view a different background color.  
    var viewBackgroundColor = [0.5, 0.2 + 0.5 * yy, 0.7 - 0.5 * yy, 1.0];  
    // Make this view get processes after the first view.  
    var viewPriority = yy + 1;  
    var viewInfo = o3djs.renderGraph.createExtraView(  
        g_viewInfos[0],  
        viewExtents,  
        viewBackgroundColor,  
        viewPriority);
```

The second and third views are considered "extra" because they re-use the draw lists created for the first view. The code iterates through the second and third views and creates a perspective projection for the top right draw context and an orthographic projection for the bottom right draw context. The view matrices are also different for the two views.

## Example: Multiple Scenes

The [Creating a Heads-Up Display](#) sample keeps things clear and simple by creating two transform

graphs, one for the 3D objects (its root is `g_3dRoot`) and one for the 2D objects (its root is `g_hudRoot`). It calls `createBasicView()` twice, once to create the sub-tree of the render graph for the 3D view (`g_viewInfo`) and a second time to create the sub-tree of the render graph for the 2D view (`g_hudViewInfo`).

```

g_3dRoot = g_pack.createObject('Transform');
g_hudRoot = g_pack.createObject('Transform');
g_viewInfo = o3djs.renderGraph.createBasicView(
    g_pack,
    g_3dRoot,
    g_client.renderGraphRoot);
g_hudViewInfo = o3djs.renderGraph.createBasicView(
    g_pack,
    g_hudRoot,
    g_client.renderGraphRoot);
// Make sure the HUD view gets drawn after the 3D view
g_hudViewInfo.root.priority = g_viewInfo.root.priority + 1;

```

This example uses an orthographic projection for the 2D view and a perspective projection for the 3D view:

```

g_hudViewInfo.drawContext.projection = g_math.matrix4.orthographic(
    0 + 0.5,
    800 + 0.5,
    600 + 0.5,
    0 + 0.5,
    0.001,
    1000);
g_hudViewInfo.drawContext.view = g_math.matrix4.lookAt(
    [0, 0, 1], // eye
    [0, 0, 0], // target
    [0, 1, 0]); // up
g_viewInfo.drawContext.projection = g_math.matrix4.perspective(
    g_math.degToRad(30), // 30 degree fov.
    g_client.width / g_client.height,
    0.1, // Near plane.
    5000); // Far plane.

```

The view matrix for the 2D view (`g_hudViewInfo.drawContext.view`) has the virtual camera (the eye point) positioned at  $(0, 0, 1)$ , aimed at the origin  $(0, 0, 0)$  with the  $y$  axis as the up vector (see previous code snippet). The view matrix for the 3D view (`g_viewInfo.drawContext.view`) has the virtual camera rotating around the "city," but is still aimed at the origin  $(0, 0, 0)$  and with  $y$  as the up vector.

```

// Fly the camera around the city.
var eye = [
    Math.sin(g_clock * g_cameraSpeed) * g_cameraRadius,
    10,
    Math.cos(g_clock * g_cameraSpeed) * g_cameraRadius];
g_viewInfo.drawContext.view = g_math.matrix4.lookAt(
    eye,
    [0, 0, 0], // target
    [0, 1, 0]); // up

```

## Render Targets

The examples discussed so far in this chapter build the render graph under a `RenderNode`. The [Render Targets](#) example creates two views, one for the teapot scene and one for the 3D scene (the cube). The teapot scene is parented by a `RenderSurfaceSet` node. The cube scene is parented by a `RenderNode`. After the teapot is rendered into the `RenderSurfaceSet` node, this *render target* is drawn into a texture. The texture is then applied to the surface of the cube in the same manner as any other texture.

## What's Next?

The *Creating a Heads-Up Display* example discussed in this chapter modifies the position of the virtual camera and the transformation matrices of the 2D objects to animate the view and the 2D objects. The next chapter (*to be provided*), Animation and Skinning, shows you how to animate objects using the O3D animation and skinning system.

# Handling Events

## Introduction

The O3D plug-in supports several ways of receiving events from the host operating system. This support is modeled after standard DOM3 mechanisms as much as possible and includes support for the following:

- Event callbacks
- Event listeners for multiple handlers
- Keyboard events

## Event Callbacks

The lowest-level interface to O3D events, for those who want things as fast and unfiltered as possible, is through the event callback API:

```
pluginObject.client.setEventCallback(type, handler);  
pluginObject.client.clearEventCallback(type);
```

Both functions are idempotent and return void.

The `pluginObject` is the plug-in's DOM element. Most commonly you'll get that element from the array passed in to the callback you supplied to `o3djs.util.makeClients()`.

The currently supported values for `type` are

```
'click'      'mousedown'  
'dblclick'   'mousemove'  
'keydown'    'mouseup'  
'keypress'   'wheel'  
'keyup'
```

The handler gets called with an O3D [Event](#) object as its only argument. This interface is as consistent as possible across browsers and operating systems. It is modeled after the [DOM3 Event](#) interface and its subinterfaces but has some differences.

Important differences are that since an O3D event just comes to a single handler via a direct call, there's no bubbling, capturing, cancellation, default action, phase, or target. In addition, there is currently no `timeStamp` field.

Mouse events contain only two coordinate pairs:

<code>(screenX, screenY)</code>	Coordinates of the mouse, in pixels, relative to the top-left corner of the screen. This matches DOM3.
<code>(x, y)</code>	Coordinates of the mouse, in pixels, relative to the top-left corner of the plug-in drawing region. This is completely independent of scrolling, page layout, and window placement.

## Event Listeners for Multiple Handlers

In cases where you have more than one handler for a given event type, you can use an optional utility function that manages multiple handlers. Its interface is very similar to the standard `addEventListener` API:

```
o3djs.event.addEventListener(pluginObject, type, eventListener);  
o3djs.event.removeEventListener(pluginObject, type, eventListener);
```

As in the API for a single handler, both of these functions are idempotent and return `void`.

The `pluginObject` is the plug-in's DOM element; most commonly you'll get that from the array passed into the callback you supplied to `o3djs.util.makeClients()`.

The currently supported values for `type` are the same as for the event callback methods, namely:

```
'click'      'mousedown'  
'dblclick'   'mousemove'  
'keydown'    'mouseup'  
'keypress'   'wheel'  
'keyup'
```

The `eventListener` is either a function or something that implements the [EventListener](#) interface. Either way, it will be invoked with an O3D [Event](#) object as its only argument.

If you use this API, you should not also use the lower-level API for the same event types. You can, however, use it for other event types.

## Keyboard Events

Since web developers commonly handle keystrokes by placing a top-level event handler on the document, O3D includes a special helper API to make this technique work.

`o3djs.util.makeClients()` calls

`o3djs.event.startKeyboardEventSynthesis()`. This sets up event listeners (via the above `o3djs.event.addEventListener`) for the keyboard event types. These handlers attempt to simulate standard DOM keyboard events for those actions. On Internet Explorer, `startKeyboardEventSynthesis()` currently does nothing, since keyboard events will come

both to the document (via Internet Explorer directly) and through the plug-in's low-level calls. This is a known issue.

These synthesized events, dispatched via [dispatchEvent](#), will bubble, capture, and cancel just like normal events that happen when the plug-in region doesn't have keyboard focus. However, since the O3D code can't precisely duplicate the events that your browser creates itself, some of the fields may not be quite right. Exactly which differ depends on which browser and host operating system you're using, so O3D supplies a helper function `o3djs.event.getEventKeyChar()` to help hide those differences.

Since this API uses `o3djs.event.addEventListener()`, you shouldn't use the event callback functions `setEventCallback()` and `clearEventCallback()` (as described earlier in the [Event Callbacks](#) section) for keyboard events if you use this one. If you'd rather use the low-level callback interface, remove the call to `startKeyboardEventSynthesis()`.

## KeyCodes for Mac (Known Issue)

On the Mac, keypress events work as expected. Each keypress event has a `keyChar` value equal to the Unicode character for that key. However, keyup and keydown events on the Mac have `keyCode` values that do not always match the `keyCode` seen in the native browser events. There are two cases:

- If the key is a letter key, an arrow key, or Return/Enter, the O3D `keyCodes` match the standard browser `keyCodes`.
- If the key is a punctuation mark, the `keyCode` returned by O3D is the Unicode value for that key, not the standard browser `keyCode`. For example the "[" character would return a `keyCode` of 91, which is the Unicode value for "[". It would not return 219, which is the standard `keyCode` for browsers.

This discrepancy is a known issue on the Mac.

## Importing Scene Files

### Introduction

This chapter describes the sample COLLADA Converter, which is used to export COLLADA files for O3D from digital content creation tools such as 3ds Max, Maya, and SketchUp. This sample converter can be used, for example, to import COLLADA files created using one of the following applications:

- Google SketchUp 6 or later
- Autodesk 3ds Max 2008
- Autodesk Maya 2008

The sample COLLADA Converter can also be used as a model as you develop your own converter and/or loader for O3D. Please note that O3D is still in the development phase, and this material is subject to change.

Download the [sample COLLADA Converter](#).

## Contents

1. [How to Use the Sample COLLADA Converter](#)
  - [Google SketchUp](#)
  - [3ds Max](#)
  - [Autodesk Maya](#)
  - [Converting Your Assets into a TGZ Archive](#)
  - [Texture Sampler Address Modes](#)
2. [Sample Converter: Under the Hood](#)

## How to Use the Sample COLLADA Converter

This section describes how to use the sample COLLADA Converter with content created using Google SketchUp, Autodesk 3ds Max, and Autodesk Maya.

### Google SketchUp

To export from Google SketchUp, select "Collada (.DAE)" as the export type.

### Caveat

Since SketchUp uses two-sided materials and COLLADA supports only single-sided materials, meshes are exported twice: once with the front material, and once with the back material (they share vertex data). In some cases, the inner material is not exported correctly from SketchUp, which causes the model to appear incorrectly in O3D.

### 3ds Max

In order to export COLLADA files from 3ds Max, download the [ColladaMax plug-in](#) for 3ds Max from the [colladamaya project](#) on SourceForge. After installation, under the "File/Export" menu, select "COLLADA (DAE)" as the export type. Note that you may see "Autodesk FBX/DAE" listed as an export type: this is the FBX plug-in from Autodesk, which exports only a limited subset of the COLLADA file format, does not support shaders, and is not recommended for use with O3D.

### Supported 3ds Max Material Types

The following material types are currently supported by O3D: Standard/Phong and DirectX (FX). All other material types are currently unsupported. Animation is not supported.

The shaders accepted by O3D should be written in the .FX file format, containing HLSL shaders. This format is a subset of Cg, containing no interfaces, unsized arrays, or scripting. There must be a single ShaderModel2 technique, containing a single pass, comprised of a vs 2.0 vertex shader, and a ps 2.0 pixel shader. See [O3D Shading Language](#) for further details.

### Using DirectX Shaders

To shade existing objects with programmable shaders in 3ds Max:

1. Open Max with the objects you wish to shade.
2. Open the Material editor ('m'), and select a material.
3. Click the material type selector button (it says "Standard Material" by default).

4. In the dialog that pops up, select "DirectX Shader." Back in the material editor, you should have a DirectX shader property page up.
5. In the material editor, click the filename of the shader (beside the button which now says "DirectX Shader"). This will open up a file dialog.
6. Navigate to the .FX file you wish to use, and select it. You should now see the properties for the shader come up.
7. Click and drag the material from the sample sphere at the top left to the object in the viewport.
8. The object will now be textured with the shader material, and will respond to property changes in the material editor.
9. Choose "File/Export" and select "COLLADA (DAE)" as the export type (\*not\* Autodesk FBX/DAE).
10. Export the file.

## Autodesk Maya

In order to export COLLADA files from Maya, download the [ColladaMaya plug-in](#) from the [colladamaya project](#) on SourceForge. After installing the plug-in, choose either "File/Export All..." or "File/Export Selected...", and select "COLLADA exporter (\*.dae, \*.xml)" in the file type selector.

## Caveats

- Only polygonal meshes are supported (NURBS surfaces are not supported).

## Converting Your Assets into a TGZ Archive

After exporting a scene from your content creation tool, you will have a single `.dae` file containing the geometry. This `.dae` file may reference one or more texture files, and one or more shader (`.fx`) files, often using local path names. In order to prepare your scene for delivery to the O3D plug-in, you will have to package all the files up as a single `.tgz` archive. The sample COLLADA Converter can perform this task for you. It resolves all absolute file references into relative URLs and converts the `.dae` file into a file called `scene.json` in the archive.

## Texture Sampler Address Modes

For imported files, the O3D Converter uses one of two default modes for wrapping textures when the  $u$ ,  $v$  texture coordinates fall outside of the 0 to 1 range. For 2D textures, the default for `Sampler.AddressMode` is WRAP. For cube map textures (environment maps), the default is CLAMP.

[Back to top](#)

## Sample Converter: Under the Hood

O3D provides a sample implementation of a converter and an archive reader for COLLADA files. You can implement your own reader (in JavaScript) for your own file format. This section explains some of the considerations addressed in the sample code that may help you develop your own converter and reader.

The O3D plug-in has support for reading compressed (gzipped) `.tar` archives to help you with delivery of your file format in a compact form.

To be clear, you don't need to use this format, or even use the sample reader code to read your assets. In the end, this format is simply converted to calls to the O3D API, which you could generate on your own from any input you choose. If you want to read your own format, you will need to write your own import code in JavaScript that calls the O3D API calls to construct a scene graph with your data.

The files that the sample JavaScript scene loader (`o3djs.scene.loadScene()`) reads are gzip-compressed `tar` archives (`.tgz` files) which contain:

- A sentinel file (must be called `aaaaaaaa.o3d`). This file is required.

This file is simply to prevent O3D from opening arbitrary `.tgz` files. If this file isn't the first file in the archive, O3D won't open the file. The file is three characters long, and contains the letters "o3d".

- Binary files in the formats that buffers, skins, and animation need (for more information on the O3D binary file formats, see this [ReadMe file](#)).
- Texture files in the formats O3D supports.
- UTF-8 text files. Note that the `scene.json` file is an example of this kind of file.

The **JSON file** is just a JSON representation of the scene graph in O3D, and it is what the sample JavaScript scene loader code reads to construct a scene graph from the archive file. It is what references the other assets in the archive.

The **binary blobs** are indexed in the JSON file by file name, offset and length, and the rest of the archive contents are referenced by name. The binary blobs are there because they are more efficient to load than the same data encoded as JSON strings. You could also define the data as text in the JSON file (but it would be slower to load and larger).

**Shader files** are text files containing the shader code referenced by the JSON file.

**Image files** can be JPG files, DDS (DirectX SDK) files, or RGB PNG files (greyscale PNG files aren't yet supported).

## Using Google SketchUp Models

### Introduction

[Google SketchUp](#) is a software package for creating 3D models, from simple to complex. These models can be subsequently used in other 3D application frameworks such as O3D. Using a 3D drawing package simplifies application development by eliminating various boilerplate tasks and allowing the developer to focus on higher-level implementation of the application itself.

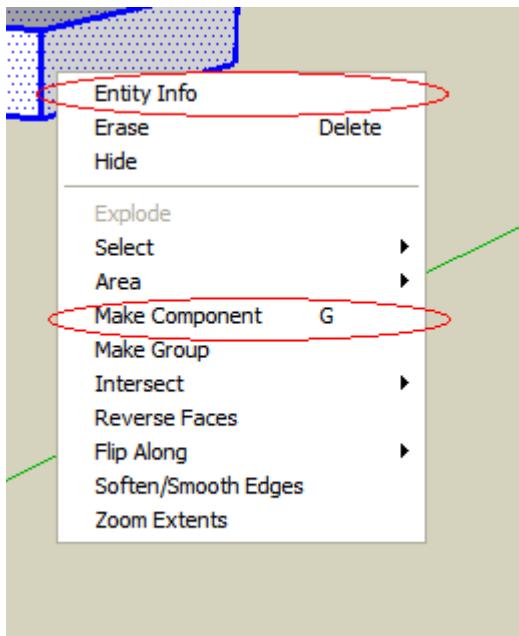
### Creating the Models in SketchUp

First, decide on the complexity of the 3D models needed for your application. You can use SketchUp to create the models yourself or use models from [3D Warehouse](#). Your models can be in one file or multiple files.

# Creating and Naming Components

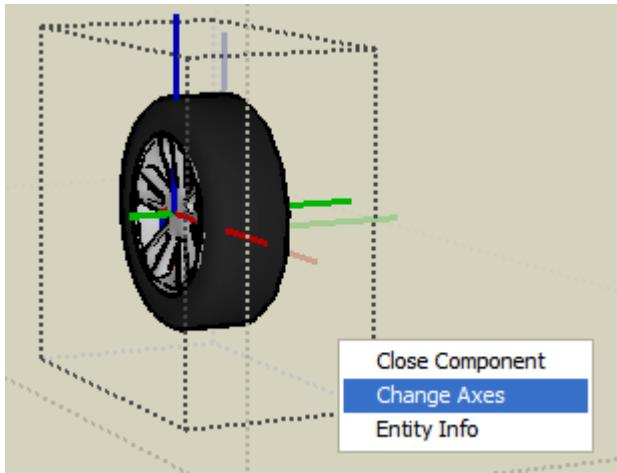
Before and during the model building stage, consider how you will use the models in the O3D application. In particular, the application will need to correctly position your model and possibly translate and rotate it. This means that the model, or any parts of it, will need to be uniquely identifiable by O3D.

In SketchUp, this goal can be accomplished by creating *components*. To create a component, select the model elements that you want to be part of the component and then use the right-click menu to select "Make Component." Once the component is created, assign it a unique name. As shown below, use the right-click menu to select "Entity Info" and fill in the "Name" field. O3D will use this unique name to find and control the component.



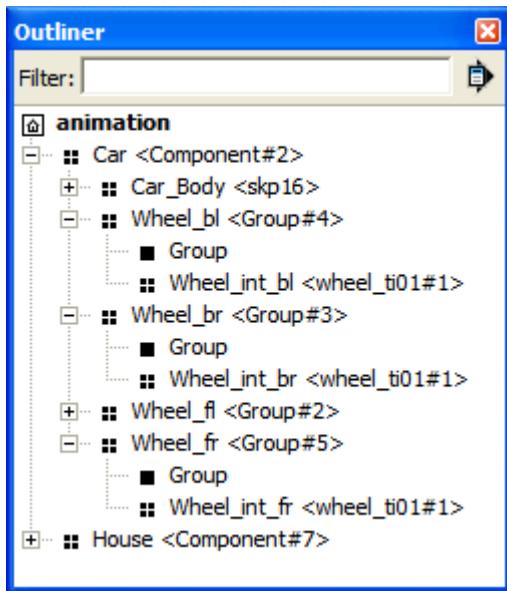
## Component Axes

Every component created in SketchUp has a set of 3D axes associated with it. To correctly control the component in O3D, you will need to know the placement and orientation of these axes. You can change a component's axes using the right-click Change Axes menu. For example, to animate a rotating wheel, create a wheel component and place the axes in the center of the wheel. This will simplify the work needed to rotate the wheel in the O3D application.



## Component Hierarchy

To implement an interesting animation application, you will need to independently control different parts of your model. For example, if your model is a car and you want to spin and turn the wheels while the car is moving, you will need a component hierarchy. To create a component hierarchy, use the Outliner tool under **Windows->Outliner** and parent components as appropriate.



## Exporting the Models from SketchUp

Once you're satisfied with the first version of your SketchUp model, you can export it from SketchUp and then import and test it in the O3D application. This process involves several steps. The first step is to export the model into a generic graphics format such as COLLADA. Use **File->Export->3D Model....** The next step is to convert the COLLADA file into a format accessible to O3D. Use the O3D Converter tool to create a zipped archive (*.tgz*) containing all the graphics assets from the SketchUp model.

# Importing the Models into an O3D Application

The resulting `.tgz` archive can be imported into your O3D JavaScript application using one of the provided utilities methods such as `o3djs.scene.loadScene()`. The importing of the model is asynchronous, so all processing should be done in a callback method that is called after the model has been loaded:

```
// Create a pack to manage our resources/assets.  
g_pack = g_client.createPack();  
  
// Create a transform to parent the loaded scene.  
g_root = g_pack.createObject('Transform');  
  
// Load the tgz file given the full path and call the callback function  
// when it's done loading.  
o3djs.scene.loadScene(g_pack, g_root, path, callback);  
  
// Callback function to control the scene after loading.  
function callback(pack, parent, success) {  
}
```

## Controlling the Models

After the model has been loaded, your application can access its components and component hierarchy. Each component is accessed through its transform. For example, if your model is a car and you named the car component "Car", then the following code accesses the transform:

```
var carTransArray = g_root.getTransformsByNameInTree('Car');  
carTransform = carTransArray[0];
```

If your car's *x* axis points forward, you can animate the car by moving it along the *x* axis:

```
for (var i = 0; i < 100; i++) {  
    carTransform.translate(10, 0, 0);  
}
```

While the car is moving forward, you can simulate the spinning wheels by rotating the wheel transform:

```
var wheelTransArray = g_root.getTransformsByNameInTree('Wheel');  
wheelTransform = wheelTransArray[0];
```

If you set up your wheel axes such that the *y* axis is orthogonal to the wheel plane, then you can rotate the wheel along the *y* axis:

```
for (var i = 0; i < 100; i++) {  
    wheelTransform.rotateY(rotationAngle);  
}
```

By using a hierarchy of transforms, you can implement more advanced animations such as turning and spinning the wheels at the same time. This is done by using a different transform for each animation.

## Updating the Models

If you decide to enhance your application by adding additional models or animating more parts, you will need to go back to your original SketchUp model and make the desired changes (for example, add new graphics, create new components or component hierarchies, change component axes, and so on). Then follow the exporting and importing steps and implement the JavaScript code to handle the new updates.

# Switching to Full-Screen Mode

## Introduction

In *full-screen mode*, all browser buttons and toolbars are removed, and only the O3D window is visible. In O3D applications, this mode is activated by an explicit user gesture—clicking on a special region displayed on the screen for this purpose. In full-screen mode, the screen resolution is set by the O3D application.

## User Interaction

When the user clicks on a specified rectangular region, O3D switches to full-screen mode, and the following message appears on the screen for a few seconds:

Press ESC to exit fullscreen.

Pressing **Escape** or pressing any other key sequence that causes the foreground window to lose focus (for example, pressing **Alt+Tab** in Windows or **Command+Tab** in OSX) also takes O3D back to plug-in mode.

## Related API Calls

The following API calls play an important role in switching to full-screen mode:

### [`client.getDisplayModes\(\)`](#)

This function returns an array that enumerates the available screen modes on this system. The array contains the resolution and the refresh rate for each possible mode. (The refresh rate reported may be 0 if the O3D plug-in is running on an LCD monitor.) The display mode is passed in to the `setFullScreenClickRegion()` method. Specifying a mode of `Renderer.DISPLAY_MODE_DEFAULT` indicates to use the refresh rate and resolution of the current screen mode.

### [`client.setFullScreenClickRegion\(Number x, Number y, Number width, Number height, Number mode\_id\)`](#)

This function sets a region of the plug-in that, if clicked by the user, activates full-screen mode. This region is not visible. You are responsible for setting up the user interface that corresponds to this region, as well as resizing the region if the plug-in window is resized. The x, y coordinates of

this region are relative to the top-left corner of the plug-in region. The x and y coordinates, as well as the width and height, are measured in pixels.

```
o3djs.event.addEventListener(Element pluginObject, string event_type, Object callback_function)
```

This is a handy utility function that manages an event listener on the O3D plug-in object. By listening for the resize event, an application can be alerted when O3D switches to full-screen mode.

client.cancelFullScreenDisplay()

If the application is in full-screen mode, this function returns the display to plug-in mode, restricting content to the plug-in region; if the application is already in plug-in mode, this function call has no effect. Note that this call does not clear the clickable region. The region remains active until `clearFullscreenClickRegion()` is called, or until it is replaced by `setFullscreenClickRegion()`.

client.clearFullscreenClickRegion()

This function deactivates the rectangle specified as the clickable region.

## Code Sample

The [fullscreen.html](#) sample modifies the [helloworld.html](#) sample to toggle between plug-in and full-screen modes. This section highlights key tasks performed in this sample.

### Querying the Display Mode

In this example, the `getFullscreenModeId()` function queries the system and selects the mode to use. This return value is then passed in to `setFullscreenClickRegion()`.

```
function getFullscreenModeId() {
    var displayModes = g_client.getDisplayModes();
    var bestMode;
    for (var index in displayModes) {
        var mode = displayModes[index];
        if (!bestMode ||
            (mode.width > bestMode.width && mode.height > bestMode.height)) {
            bestMode = mode;
        }
    }
    if (bestMode) {
        return bestMode.id;
    } else {
        return g_o3d.Renderer.DISPLAY_MODE_DEFAULT; // getDisplayModes isn't
implemented on all platforms yet
    }
}
```

## Displaying the User Click Region

The `setFullscreenClickRegion()` function sets up the area on the screen that activates full-screen mode, but you are responsible for creating the accompanying graphical user interface (GUI) that corresponds to this rectangle. In the [`fullscreen.html`](#) example, the `createBannerSurfaceAndClickableRegion()` function performs these related tasks:

- Creates a material.
- Loads a texture into the material. (This texture contains the text for the clickable region.)
- Creates a polygon for the texture.
- Sets up an event handler to listen for the resize event.
- Set up the clickable region on the screen.

Here is the code for adding the event listener and setting up the clickable region, which occurs at the end of the `createBannerSurfaceAndClickableRegion()` function):

```
o3djs.event.addEventListener(g_o3dElement, 'resize', handleResizeEvent);
g_client.setFullscreenClickRegion(10, clientHeight - texture.height - 10,
    texture.width, texture.height, getFullscreenModeId());
```

The call for a resize event receives an event object that contains the dimensions of the window you're drawing into and a flag that indicates whether you're in full-screen mode.

## Setting Up Two Views

The [`fullscreen.html`](#) sample sets up two views, a perspective view (`g_viewInfo`) for the 3D teapot scene, and an orthographic view (`orthoViewInfo`) that is drawn after (that is, on top of) the 3D scene. The orthographic view contains the textured rectangle that corresponds to the clickable region for full-screen mode.

The `setUpBanner()` function creates the orthographic view for the 2D user interface "layer." This view turns off the clear color flag so that the 3D view remains on the screen. It sets the priority for the orthographic view to one greater than that of the perspective view, so that it is drawn after the perspective view (`g_viewInfo`):

```
function setupBanner(width, height, viewInfo) {
    g_orthoRoot = g_pack.createObject('Transform');
    // Create a view for the orthographic display of the fullscreen banner.
    var orthoViewInfo = o3djs.rendergraph.createBasicView(
        g_pack,
        g_orthoRoot,
        g_client.renderGraphRoot);
    // Make sure the orthographic view gets drawn after the 3d stuff
    orthoViewInfo.root.priority = g_viewInfo.root.priority + 1;
    // Turn off clearing the color for the orthographic view, since that would
    // erase the 3d parts. Leave clearing the depth and stencil, so it's
    // unaffected by anything done by the 3d parts.
    orthoViewInfo.clearBuffer.clearColorFlag = false;
    .
    .
    .
}
```

See the [`Creating a heads-up display`](#) sample for another example of creating a 2D overlay and an accompanying 3D view.

# O3D Best Practices and Tips

## Introduction

This chapter presents a few useful tips for writing O3D applications.

Always initialize the perspective matrix from the width and height of the client area, not from magic numbers.

This method ensures that if the client area changes size, the perspective matrix changes accordingly.

If you are using a render target, make constants for the width and height of the render target such that it's clear how any perspective matrix used by the render target is set.

Don't use the DOM values `clientWidth` and `clientHeight` to initialize the perspective matrix; these values will give you the wrong answer in full-screen mode. Use `client.width` and `client.height`, which are correct in both full-screen and plug-in modes.

If your client area is resizeable, set the perspective matrix every frame in your render callback.

If you do not reset the perspective matrix for every frame, the browser will resize the client area with the wrong aspect ratio when the user resizes the window.

Always clear a render callback.

If you do not clear the callback in `window.unload`, the browser will attempt to render during cleanup after the JavaScript has been freed, which causes JavaScript errors in the callback.

Minimize the number of effects used in your application.

Since it's expensive to create and render effects, create objects that share effects when possible. For example, the [Picking](#) example creates seven different objects that all use the same effect.

Reuse geometry when possible.

It is much less expensive to create one cube and instance it 256 times than to create 256 identical cubes.

Adding transforms to the transform graph can simplify some tasks.

Often, adding a separate transform above a shape can help to isolate behavior to that shape.

Use the [`o3djs.debug`](#) library to debug your O3D application.

This library provides a debug helper object that can add axes, spheres, and boxes to your

transforms as well as draw lines between two points in 3D space.

Use the camera utilities, [o3djs.camera](#) to create a simple camera that views your scene.

This library contains functions that provide an easy way to get your content in front of the camera.

A shader either handles textures or it doesn't. A trick that enables you to write only one shader, which handles textures and simulates color shading, is to create an error texture (with [Client.setErrorTexture\(\)](#)) that is white.

With this solution, textures are applied to the objects as specified. The texture, when supplied, is multiplied by the COLOR value, as shown in the code sample below. If no texture is supplied, the white error texture is multiplied by the COLOR value, which produces an acceptable effect.

```
float4 color;
sampler texSampler;
float4 pixelShaderFunction(PixelShaderInput input): COLOR {
    float4 diffuseColor = tex2D(texSampler, input.texcoord) * color;
    return diffuseColor;
}
```

# Performance Tuning

This chapter provides information on how to achieve maximum performance for your O3D application.

## Contents

1. [Using GPU Hardware Acceleration](#)
2. [Using the V8 Engine](#)

## Using GPU Hardware Acceleration

O3D is designed to use hardware acceleration on a wide variety of GPU chipsets. The result is that your application can present high-end, real-time 3D graphics on most systems. GPU support for O3D features can be divided into three general categories:

- **The GPU supports all O3D features.** In this case, your system will almost always use GPU hardware acceleration. The exception is when the system runs out of GPU resources (for example, when an application that requires a large amount of resources is run multiple times simultaneously).
- **The GPU supports most O3D features, except for the following:**
  - Floating point textures (includes the formats R32F, ABGR16F and ABGR32F)
  - Geometry with more than 65,534 vertices

In this case, your system will use GPU acceleration if you don't require either of those features. If you do require them, your system will use the software renderer.

- **The GPU does not support O3D features.** In this case, the system will always use the

software renderer.

If your application requires either floating point textures or large geometry (or both), follow the instructions below for creating the O3D client. An alternative is to consider whether there are workarounds—for example, splitting geometry that exceeds the maximum vertex count into smaller groups of polygons that can be rendered by all GPUs.

## The `makeClients()` Function

The `o3djs.util.makeClients()` utility function provides an example of how to create an O3D object that has one of the special hardware requirements. The second argument to the `makeClients()` function is `opt_features`, which takes a comma-separated string of values:

- `FloatingPointTextures` - includes the formats R32F, ABGR16F and ABGR32F
- `LargeGeometry` - allows buffers to have more than 65534 elements

You can also [create the O3D object explicitly](#) and pass in these values when the O3D element is created on the HTML page.

## Using the V8 Engine

JavaScript performance across browsers can vary greatly. An O3D application may perform well in one browser while not performing adequately in another. There are a number of causes for this difference. Some JavaScript engines interpret JavaScript code, while others compile it to native code. Some browsers host plug-ins in the same process as the browser, while others host them in a separate process, leading to a high overhead for calls between JavaScript and the O3D API.

To mitigate these issues, the O3D plug-in embeds the V8 JavaScript engine. This is the same high-performance JavaScript engine found in the Chrome browser. An O3D application using the embedded JavaScript engine instead of or in addition to the browser's own JavaScript engine may see improved performance, more consistent performance across browsers, and sometimes significantly lower overhead when calling between JavaScript and the O3D API. A disadvantage of this approach is it is no longer possible to debug such JavaScript code in the browser's JavaScript debugger.

## High-Level Interface

There are two interfaces to the V8 engine, one higher level and one lower level. The high-level interface is provided by the `o3djs` utility code and is available when you require `o3djs.util`. Then, you select the engine by calling `o3djs.util.setMainEngine()` and initialize the client by calling `o3djs.util.makeClients()`. If you do not call `setMainEngine()`, JavaScript will run in the browser engine by default. Here is an example of running code in V8:

```
function init() {
    o3djs.util.setMainEngine(o3djs.util.Engine.V8);
    o3djs.util.makeClients(initStep2);
    // This function and all functions it calls will run in V8 rather than the browser.
    function initStep2(clientElements) {
        // ...
    }
}
```

The contents of all the script tags will be evaluated in V8 in addition to the browser. All the the modules requested through `o3djs.require` will also be available in V8.

To run in V8, there are a few other steps to take. First of all, make sure none of the code in your script tags does anything other than define variables and functions. The problem is, if the code outside of function definitions actually does anything, it will happen twice—once when the browser first evaluates it and a second time when it gets uploaded to V8. Here is an example:

```
// At top level
window.onload = init;
window.onunload = uninit;
```

This code will actually be executed twice. The first time, `onload` and `onunload` will be made to reference the browser's copies of the `init` and `uninit` functions. The second time, they will be made references to V8's. It would be better to set up the callbacks in the body tag, so that they are always set up to point to the browser's copies. It is best to always run these two functions in the browser:

```
<body onload="init()" onunload="uninit()">
...
</body>
```

Another thing to consider is that separate global variables with the same name can coexist, one in V8 and one in the browser. V8 can read all the browser's global variables until it tries to modify them. At that point, a new global variable, only visible to V8 is created, which hides the browser's variable from V8. It is sometimes necessary to synchronize their values. This can be done from V8 like this:

```
// Running in V8
function initStep2() {
  g_client = clientElements[0].client; // Set V8's copy of the variable.
  window.g_client = g_client;           // Set the browser's copy of the variable to
be the same as V8's.
}
```

This shows how V8 can be sure it is accessing the browser's copy of a global variable by accessing it through `window`. There is no straightforward way to synchronize from the browser. Do it from V8.

When V8 is set as the main engine, one and only one of the `o3d` divs must have the id "o3d". This is how `o3djs` knows which plug-in instance to invoke the `makeClients()` callback in:

```
<div id="o3d" style="width: 800px; height: 600px;"></div>
```

## Low-Level Interface

With the low-level interface, JavaScript code is not automatically pulled out of script tags and evaluated in V8. Instead, you pass a string of JavaScript code to the `eval` method on the plug-in object. If the variable `g_plugin` holds a reference to the plug-in, then this code:

```
var result = g_plugin.eval("2+2");
```

will pass the expression "2+2" from the browser's JavaScript engine to the embedded V8 JavaScript engine. The embedded engine will evaluate the result to be 4. The number 4 is then returned to the browser's JavaScript engine and stored in the `result` variable.

Although this simple example demonstrates how to use the embedded V8 engine, it will probably not improve performance. In fact, it will most likely be much worse than simply writing:

```
var result = 2+2;
```

The reason is that there is some overhead for transitioning control from the browser JavaScript engine, evaluating the string as JavaScript in the V8 engine, and then transitioning back to the browser. It is only worthwhile running JavaScript in the V8 engine when the time savings for doing so balances the cost of the overhead. For example, a loop iterating many hundreds of times would be worth the transition time.

## Evaluating a Function as a String

It is rather inconvenient to have to construct a string containing a complicated sequence of JavaScript code. O3D provides a means of evaluating JavaScript that is more convenient in these cases. It is a two-step process:

1. Create a V8 function object that optionally takes arguments and assign it to a browser JavaScript variable.
2. Whenever you want to invoke that V8 function, call the returned variable with any applicable arguments.

For example, here is how you would define the variable for the function:

```
var func = g_plugin.eval("function(a, b) { return a + b; }");
```

Usually you only need to call this first step once. From now on, it is possible to evaluate an addition operation in the V8 engine by calling the variable `func` as a function. For example:

```
var result = func(2, 2);
```

This call has the same effect as the earlier example, but it can be easier in some cases. In addition to standalone functions, the same trick works with object methods. If the V8 engine creates a new object and returns it to the browser, whenever the browser invokes a method on that object, control automatically transitions from the browser to the V8 engine, where the method will be evaluated. For example:

```
var mathHelper = g_plugin.eval("{ add: function(a, b) { return a + b; } }");
var result = mathHelper.add(2, 2);
```

## Callback Functions

In some cases, you will want to evaluate a function to do complex computation in the V8 engine and then return control to the browser to perform additional operations, such as calling an Alert or logging data to the console. Here is an example of how the browser could pass a callback function in to the V8 engine, and the V8 engine could invoke the callback to the browser. For example:

```
var greatestCommonDivisor = g_plugin.eval(
  "function(a, b, logCallback) {" +
  "  if (a === 0)" +
  "    return b;" +
  "  while (b !== 0) {" +
  "    if (a > b) {" +
  "      a -= b;" +
  "      logCallback('a = ' + a);" +
  "    } else {" +
  "      b -= a;" +
  "      logCallback('b = ' + b);" +
  "    }" +
  "  }" +
```

```

    "  return a" +
  "}");
function logResult(r) {
  var textNode = document.createTextNode(String(r));
  var paragraph = document.createElement('p');
  paragraph.appendChild(textNode);
  document.body.appendChild(paragraph);
}
var result = greatestCommonDivisor(151698344, 205238936, logResult);
alert(result);

```

**Note:** In JavaScript, a string cannot span more than one line, so the + signs are necessary to make the entire eval expression one string. Alternatives to this notation are to put the text in a <text> element or in a separate file that is referenced by the function.

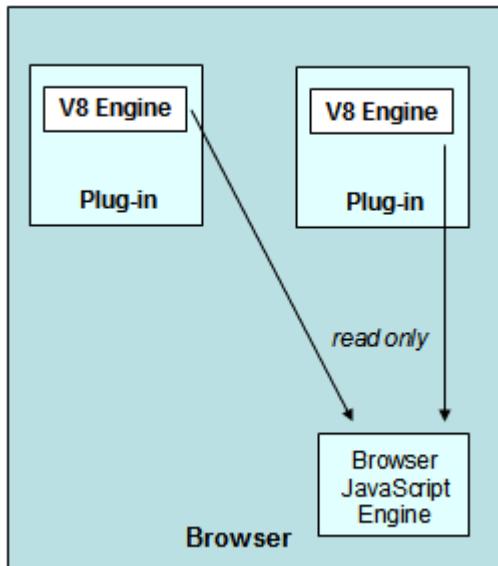
## The *plugin* Variable

Within V8, there is a special variable called `plugin`. This variable always references the `plugin` object and can be used to locate other useful objects. For example:

```
var o3d = plugin.o3d;
```

## Accessing Browser Variables from V8

A V8 engine can read all the global variables in the browser's JavaScript engine, but it cannot write to them. Any changes the V8 engine makes are visible only to the plug-in instance for a given V8 engine (see the following diagram). The browser cannot directly read or write to the V8 engine's global variables.



## Accessing the DOM from V8

In most browsers, it is possible to access and manipulate the DOM from V8. For example, the following code works when run from V8:

```
var element = document.getElementById("myElement");
element.value = "foo";
```

Unfortunately, at the present time, this code does not work in Internet Explorer. From V8, it is not possible to invoke a DOM method in Internet Explorer. However, it is possible to read and write DOM properties in Internet Explorer.

As a partial workaround, since it is one of the most common DOM methods, we provide a utility function for `getElementById()` that works in Internet Explorer:

```
var element = o3djs.util.getElementById("myElement");
element.value = "foo";
```

## How to Obfuscate Your Code

### What is JavaScript obfuscation?

JavaScript obfuscation is the process of transforming more-readable code into less-readable code. For example, take this simple function:

```
function sumAllArrayElements(inputArray) {
    var sum = 0;
    for (var index = 0; index < inputArray.length; i++) {
        sum += inputArray[index];
    }
    return sum;
}
```

It's pretty clear that this code adds up the elements in an array and returns the sum. But how about if we write it like this?

```
function input(ret){var
square=0;for(var
product=0;product<ret.length;i++){square+=ret[product];}return
square;}
```

It does just the same thing, but it takes a bit longer to figure out what that is. All we've done is change variable names and remove some whitespace. However, there are a lot of other tricks one can use, such as unicode-escaping strings, writing code to generate JavaScript from a packed data format and then execute it (self-loading or self-decrypting code, in effect), and so on.

Here's another way to obfuscate the function:

```
var foo = [
  102,117,110,99,116,105,111,110,32,115,117,109,65,108,108,65,114,
  114,97,121,69,108,101,109,101,110,116,115,40,105,110,112,117,116,
  65,114,114,97,121,41,32,123,10,32,32,118,97,114,32,115,117,109,32,
  61,32,48,59,10,32,32,102,111,114,32,40,118,97,114,32,105,110,100,
  101,120,32,61,32,48,59,32,105,110,100,101,120,32,60,32,105,110,112,
  117,116,65,114,114,97,121,46,108,101,110,103,116,104,59,32,105,43,
  43,41,32,123,10,32,32,32,115,117,109,32,43,61,32,105,110,112,
  117,116,65,114,114,97,121,91,105,110,100,101,120,93,59,10,32,32,
  125,10,32,32,114,101,116,117,114,110,32,115,117,109,59,10,125];
```

```
var bar = "";
for (var i in foo) {
    bar[i] = String.fromCharCode(foo[i]);
}
document.write(bar.join(""));
```

And of course, you can use multiple techniques on top of each other. Now imagine trying to figure out a 10000-line JavaScript application that's been thoroughly obfuscated: while it's always possible, it can be extremely tedious and time-consuming.

## What is JavaScript compilation/optimization/compression?

In this context, JavaScript "compilation" means automatically transforming handwritten code into more compact and/or higher performance code. This term can also mean just-in-time (JIT) compiling, which is something that interpreters do to make code run faster without user intervention. Or it can mean compiling JavaScript into another language, or another language into JavaScript. But that's not what we're talking about here.

For example, we can speed up the original example by using the "in" operator (generally saving both run time, by using more efficient code paths, and download time, by saving bytes from our download), by shortening all symbols down to the bare minimum, by hoisting the declaration of the index variable into the previous declaration of the sum, and by removing all nonessential whitespace:

```
function f(l){var i,s=0;for(i in l){s+=l[i];}return s;}
```

Note that in both the obfuscated and the optimized versions, we've changed the name of the function. That means that whatever code elsewhere references that function by name will also have to be aware of that change. In practice, it also means that you won't get to use single-character names for functions, because you'll run out of them too quickly, but you'll probably still shrink them substantially. It's also worth noticing that the optimized code is somewhat obfuscated, although not as much as it could be if we had prioritized obfuscation over performance.

## Why would I want to obfuscate my code?

If you want to write applications in JavaScript but don't want users to be able to read, reuse, or reverse-engineer your code, obfuscation can help. It's not a guarantee of privacy—any obfuscation can, with sufficient effort, be undone—but it can make reverse-engineering tedious enough that it's generally not worth the bother. Don't make it worth the bother by trying to hide secret keys or account passwords in the code, though!

## Why would I want to compile my code?

If you want your code to download and/or run faster, compilation can help. Virtually any large-scale JavaScript project can benefit from compilation. Also, compilation will invariably obfuscate your code somewhat, so if you want both optimization and obfuscation, you may want to prioritize optimization, and find that that you get sufficient obfuscation for free.

## How do I compile/obfuscate/compress my code?

There are a number of free and commercial JavaScript [obfuscators](#) and [compilers / optimizers](#) available

on the web. They vary in cost and ease of use. In choosing between them, you also need to decide what you're prioritizing (download size, speed, obfuscation). Some will be able to optimize for each of these, while some will specialize in just obfuscation or just speed. Examples of well-respected optimizers are the [Dojo Toolkit's ShrinkSafe](#) compressor and [Yahoo's YUI Compressor](#).

## I tried to obfuscate/compile my code, and now it doesn't work. What happened?

Chances are the tool you used changed some symbol that it shouldn't have. For example, if it changes the name of a function when it is declared, but not all the calls to the function, that'll break your code. Also, obfuscators generally aren't smart enough to figure out when you're using symbols inside strings. For example, here's something that may confuse them:

```
var foo = {};
var bar = "fun";
foo.fun = function() { return 7; };
alert(foo[bar]);
```

If your obfuscator changes the name of the field `fun`, it probably won't understand that it also has to change the string held in variable `bar` to match. In fact, in many situations, it will be completely impossible for it to do that kind of replacement. So you'll have to help it along by telling it not to rename `fun`, since it's referenced in a string. Any compiler or obfuscator will have a way for you to tell it what symbols not to change (or, more conservatively, which symbols it's OK to change). It will already understand that it can't change JavaScript keywords such as `function`, but if you're using any libraries, you'll have to tell it about their special symbols. We supply lists of the O3D symbols that can and cannot be changed.

Another issue to be aware of is that JavaScript obfuscators do not work with shader code. Be sure to protect your shader code from the obfuscator—for example, by putting the shader code into a string literal or a separate file.

## How to Add O3D to a Gadget

### Overview

This document outlines how to use O3D inside an iGoogle or OpenSocial gadget. This process is very similar to creating a standalone page, except for a few restrictions imposed on the gadgets. iGoogle itself provides a very brief guide on [how to migrate a standalone page](#).

### Converting an O3D Application into a Gadget

Gadgets are essentially the same as pages, except that you only have control over a small part of the page—you cannot change the `<head>` section, and a lot of other things may be put in the body for you.

1. Since both OpenSocial and iGoogle will add a lot of JavaScript to the generated page for you, you cannot overwrite `window.onload`. Instead, you must use the provided API for registering such events. For instance, a call like this:

```
window.onload = myInitFunction;  
  
should become something like this on both iGoogle v2 and OpenSocial:  
gadgets.util.registerOnLoadHandler(myInitFunction);
```

or, if using the legacy iGoogle API:

```
_IG_RegisterOnloadHandler(myInitFunction);
```

2. If you wish to set the title of the gadget, you cannot access <head><title>. Instead, add the title to your module preferences, like this:

```
<ModulePrefs  
    title="My cool O3D app"  
    title_url="http://www.mydomain.com/"  
    height="200"  
    author="Jane Smith"  
    author_email="name@mydomain.com"  
/>
```

or set it dynamically with:

```
gadgets.window.setTitle("My cool O3D app");
```

3. If you are using utility libraries (e.g., o3djs/base.js), host the utilities on a server so that they are available via http:

```
<script type="text/javascript"  
src="http://www.mydomain.com/o3djs/base.js</script>
```

If listing your utility libraries using o3djs.require, also specify the base path of the 'o3djs' utilities directory location. For example:

```
o3djs.basePath=http://www.mydomain.com/;  
o3djs.require('o3djs.util');  
o3djs.require('o3djs.event');
```

## Testing the Gadget

1. To test your gadget, save the gadget's XML file on a publicly accessible web server and then use that URL to embed it in a page that supports gadgets. For example, in iGoogle navigate to your iGoogle page and click "Add Stuff." The page will display an "Add feed or gadget" link. Enter the URL of your gadget's XML file.
2. By default, gadget XML specs are cached by the gadgets server. When developing gadgets and making frequent changes to the XML spec, you may want to disable gadget caching, so that changes are immediately reflected every time the page is refreshed.

To disable caching, append the ?nocache suffix to the URL of your gadget XML specification. For example: <http://www.mydomain.com/mygadget.xml?nocache>.

# Using Gears to Run Your App Offline

## Benefits of Using Gears

Running your O3D application with Gears has a number of benefits:

- Your application works offline.
- Performance improves because resources are cached locally.
- Server load is lighter.

## How to Use Gears

Here are the steps required to run your O3D application offline using Gears:

1. Download `gears_init.js` and install it with your application.
2. Include this script, which does the following:
  - creates a local server
  - creates a managed store for the resources used by the application
  - gives the store the list of required files
  - checks for updates to the resource files

```
<script src="../assets/gears_init.js" type="text/javascript"> </script>
<script type="text/javascript">
try {
    var localServer = google.gears.factory.create('beta.localserver');
    var store = localServer.createManagedStore('test-store');
    store.manifestUrl = 'site-manifest.txt';
    store.checkForUpdate();
} catch (ex) {
    dump("No offline access.\n");
}
</script>
```

This script should be included in the main html file in the `<head>` element before anything else is loaded. The only time the script requires a server connection is when it runs the `store.checkForUpdate()` function.

3. Create the `site-manifest.txt` file, which lists all the resources used by the application. This file uses the JSON format, with no wildcards. The file specifications need to be local references. For example:

```
{
  "betaManifestVersion": 1,
  "version": "0.1",
  "entries": [
    { "url": "../assets/gears_init.js" },
    { "url": "../assets/texture_b3.jpg" },
    { "url": "game.css" },
    { "url": "game.html" },
    { "url": "client.js" },
    { "url": "third_party/soundmanagerv25b-20080505/script/soundmanager2.js" },
    { "url": "third_party/soundmanagerv25b-20080505/soundmanager2.swf" },
    { "url": "third_party/soundsnap/11115901.mp3" },
```

```
{ "url": "third_party/soundsnap/11118801-modified.mp3" },  
]  
}
```

## For More Information

Go to <http://code.google.com/apis/gears> for more information on how to use the open-source Gears API.

# Package o3d

## Classes

- class [ArchiveRequest](#)
- class [BoundingBox](#)
- class [ParamBoundingBox](#)
- class [Buffer](#)
- class [VertexBufferBase](#)
- class [VertexBuffer](#)
- class [SourceBuffer](#)
- class [IndexBuffer](#)
- class [Canvas](#)
- class [CanvasFontMetrics](#)
- class [CanvasPaint](#)
- class [CanvasShader](#)
- class [CanvasLinearGradient](#)
- class [ClearBuffer](#)
- class [Renderer](#)
- class [Client](#)
- class [Counter](#)
- class [SecondCounter](#)
- class [RenderFrameCounter](#)
- class [TickCounter](#)
- class [CurveKey](#)
- class [StepCurveKey](#)
- class [LinearCurveKey](#)
- class [BezierCurveKey](#)
- class [Curve](#)
- class [DisplayMode](#)
- class [DrawContext](#)

- class [DrawElement](#)
- class [DrawList](#)
- class [DrawPass](#)
- class [EffectParameterInfo](#)
- class [EffectStreamInfo](#)
- class [Effect](#)
- class [Element](#)
- class [Event](#)
- class [Field](#)
- class [FloatField](#)
- class [UInt32Field](#)
- class [UByteNField](#)
- class [FileRequest](#)
- class [Function](#)
- class [ParamFunction](#)
- class [FunctionEval](#)
- class [Material](#)
- class [Matrix4AxisRotation](#)
- class [Matrix4Composition](#)
- class [Matrix4Scale](#)
- class [Matrix4Translation](#)
- class [ObjectBase](#)
- class [NamedObjectBase](#)
- class [NamedObject](#)
- class [Pack](#)
- class [Param](#)
- class [ParamFloat](#)
- class [ParamFloat2](#)
- class [ParamFloat3](#)
- class [ParamFloat4](#)
- class [ParamMatrix4](#)
- class [ParamInteger](#)
- class [ParamBoolean](#)

- class [ParamString](#)
- class [ParamSampler](#)
- class [ParamMaterial](#)
- class [ParamState](#)
- class [ParamEffect](#)
- class [ParamTransform](#)
- class [ParamDrawList](#)
- class [ParamDrawContext](#)
- class [ParamArray](#)
- class [ParamParamArray](#)
- class [ParamObject](#)
- class [ParamOp2FloatsToFloat2](#)
- class [ParamOp3FloatsToFloat3](#)
- class [ParamOp4FloatsToFloat4](#)
- class [ParamOp16FloatsToMatrix4](#)
- class [TRSToMatrix4](#)
- class [Primitive](#)
- class [RawData](#)
- class [RayIntersectionInfo](#)
- class [RenderEvent](#)
- class [RenderNode](#)
- class [RenderSurfaceBase](#)
- class [RenderSurface](#)
- class [RenderDepthStencilSurface](#)
- class [RenderSurfaceSet](#)
- class [Sampler](#)
- class [Shape](#)
- class [Skin](#)
- class [SkinEval](#)
- class [ParamSkin](#)
- class [WorldParamMatrix4](#)
- class [WorldInverseParamMatrix4](#)
- class [WorldTransposeParamMatrix4](#)

- class [WorldInverseTransposeParamMatrix4](#)
- class [ViewParamMatrix4](#)
- class [ViewInverseParamMatrix4](#)
- class [ViewTransposeParamMatrix4](#)
- class [ViewInverseTransposeParamMatrix4](#)
- class [ProjectionParamMatrix4](#)
- class [ProjectionInverseParamMatrix4](#)
- class [ProjectionTransposeParamMatrix4](#)
- class [ProjectionInverseTransposeParamMatrix4](#)
- class [WorldViewParamMatrix4](#)
- class [WorldViewInverseParamMatrix4](#)
- class [WorldViewTransposeParamMatrix4](#)
- class [WorldViewInverseTransposeParamMatrix4](#)
- class [ViewProjectionParamMatrix4](#)
- class [ViewProjectionInverseParamMatrix4](#)
- class [ViewProjectionTransposeParamMatrix4](#)
- class [ViewProjectionInverseTransposeParamMatrix4](#)
- class [WorldViewProjectionParamMatrix4](#)
- class [WorldViewProjectionInverseParamMatrix4](#)
- class [WorldViewProjectionTransposeParamMatrix4](#)
- class [WorldViewProjectionInverseTransposeParamMatrix4](#)
- class [State](#)
- class [StateSet](#)
- class [Stream](#)
- class [ParamVertexBufferStream](#)
- class [StreamBank](#)
- class [ParamStreamBank](#)
- class [Texture](#)
- class [Texture2D](#)
- class [TextureCUBE](#)
- class [TickEvent](#)
- class [Transform](#)
- class [TreeTraversal](#)

- class [Float2](#)
- class [Float3](#)
- class [Float4](#)
- class [VertexSource](#)
- class [Viewport](#)

# **o3d.ArchiveRequest Class Reference**

Inherits [o3d.ObjectBase](#).

[List of all members.](#)

---

## Detailed Description

An [ArchiveRequest](#) object is used to carry out an asynchronous request for a compressed archive (containing multiple files).

Note: The archive must have as its first file a file named 'aaaaaaaa.o3d' who's contents is 'o3d'. This is to prevent O3D being used to open archive files that were not meant for it.

```
var request = pack.createArchiveRequest();
request.open("GET", url);

request.onfileavailable = myFileAvailableCallback;
request.onreadystatechange = myReadyStateChangeCallback;
request.send();

function myFileAvailableCallback() {
    dump("uri: " + request.data.uri + "\n");
    dump("content: " + request.data.stringValue + "\n");

    // You can pass a RawData to various creation functions. Note: request.data
    // is only valid during an onfileavailable callback.

    // Examples:
```

```

if (request.data.uri == 'mytexture.jpg')
pack.createTexture2d(request.data, makeMips);
if (request.data.uri == 'myvertices.bin')
vertexBuffer.set(request.data);
if (request.data.uri == 'myAudio.mp3')
audioSystem.createSound(request.data);
}

function myReadyStateChangeCallback() {
if (request.done) {
if (request.success) {
// there were no errors trying to read the archive
} else {
dump(request.error);
}
}
}

// When you are done with the RawDatas loaded by the request, remove
// the request from the pack to free them.
pack removeObject(request);

```

## Public Member Functions

- [open](#) (String method, String [uri](#))
- [send](#) ()
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- ArchiveRequestCallback [onreadystatechange](#)
- ArchiveRequestCallback [onfileavailable](#)
- String [uri](#)
- [RawData data](#)
- Number [streamLength](#)
- Number [bytesReceived](#)
- Number [readyState](#)

- bool [done](#)
  - bool [success](#)
  - String [error](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.ArchiveRequest.open ( String *method*,  
 String *uri*  
 )

Sets up several of the request fields.

### Parameters:

*method* "GET" is the only supported method at this time  
*uri* the location of the file to fetch

o3d.ArchiveRequest.send ( )

Send the request. Unlike XMLHttpRequest the onreadystatechange callback will be called no matter what, with success or failure.

---

# Member Data Documentation

Number [o3d.ArchiveRequest.bytesReceived](#)

The number of bytes downloaded so far.

You can use this value along with streamLength to figure out the download progress.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[RawData o3d.ArchiveRequest.data](#)

A [RawData](#) object representing the file that is currently available. Note: This value is only valid inside the onfileavailable callback.

This property is read-only.

bool [o3d.ArchiveRequest.done](#)

Indicates whether processing for this [FileRequest](#) has finished.

This property is read-only.

String [o3d.ArchiveRequest.error](#)

An error message. If done is true and success is false this will be an error message describing what went wrong.

This property is read-only.

ArchiveRequestCallback [o3d.ArchiveRequest.onfileavailable](#)

A callback that gets called each time a file fully downloads and becomes available.

This property is write-only.

ArchiveRequestCallback [o3d.ArchiveRequest.onreadystatechange](#)

A callback that gets called each time readyState changes.

This property is write-only.

Number [o3d.ArchiveRequest.readyState](#)

Holds the same values as in XMLHttpRequest:

- 0 = uninitialized
- 1 = opened
- 2 = sent
- 3 = receiving
- 4 = loaded (the file has been downloaded, but may or may not have been parsed yet)

This property is read-only.

Number [o3d.ArchiveRequest.streamLength](#)

The length of the entire archive in bytes.

Use this value along with bytesReceived to figure out the download progress.

This property is read-only.

bool [o3d.ArchiveRequest.success](#)

This field is only valid if done is true. It indicates whether or not the request succeeded. If false see error for an error message.

This property is read-only.

String [o3d.ArchiveRequest.uri](#)

The uri of the archive being downloaded.

This property is read-only.

# o3d.BoundingBox Class Reference

[List of all members.](#)

---

## Detailed Description

A [BoundingBox](#) represents an Axis Aligned Bounding Box.

## Public Member Functions

- [BoundingBox \(Point3 min\\_extent, Point3 max\\_extent\)](#)
- [BoundingBox mul \(Matrix4 matrix\)](#)
- [BoundingBox add \(BoundingBox box\)](#)
- [RayIntersectionInfo intersectRay \(Point3 start, Point3 end\)](#)
- [RayIntersectionInfo intersectRay \(Number startX, Number startY, Number startZ, Number endX, Number endY, Number endZ\)](#)
- bool [inFrustum \(Matrix4 matrix\)](#)

## Public Attributes

- Array [marshaled](#)
  - bool [valid](#)
  - Point3 [minExtent](#)
  - Point3 [maxExtent](#)
- 

## Constructor & Destructor Documentation

```
o3d.BoundingBox.BoundingBox ( Point3 min\_extent,
                             Point3 max\_extent
                           )
```

Creates [BoundingBox](#) from min\_extent and max\_extent

**Parameters:**

*min\_extent* minimum extent of the box.  
*max\_extent* maximum extent of the box.

---

## Member Function Documentation

[BoundingBox](#) o3d.BoundingBox.add ( [BoundingBox](#) *box* )

Adds a bounding box to this bounding box returning a bounding box that encompasses both.

**Parameters:**

*box* [BoundingBox](#) to add to this [BoundingBox](#).

**Returns:**

The new bounding box.

bool o3d.BoundingBox.inFrustum ( [Matrix4](#) *matrix* )

Returns true if the bounding box is inside the frustum.

**Parameters:**

*matrix* Matrix to transform the box from its local space to view frustum space.

**Returns:**

True if the box is in the frustum.

[RayIntersectionInfo](#) o3d.BoundingBox.intersectRay ( Number *startX*,  
Number *startY*,  
Number *startZ*,  
Number *endX*,  
Number *endY*,  
Number *endZ*  
)

Checks if a ray defined in same coordinate system as this box intersects this bounding box.

**Parameters:**

*startX* The x coordinate of start of ray in local space.  
*startY* The y coordinate of start of ray in local space.  
*startZ* The z coordinate of start of ray in local space.  
*endX* The x coordinate of end of ray in local space.  
*endY* The y coordinate of end of ray in local space.  
*endZ* The z coordinate of end of ray in local space.

**Returns:**

[RayIntersectionInfo](#). If result.value is false then something was wrong like using this function with an uninitialized bounding box. If result.intersected is true then the ray intersected the box and result.position is the exact point of intersection.

[RayIntersectionInfo](#) o3d.BoundingBox.intersectRay ( [Point3](#) start,  
[Point3](#) end  
)

Checks if a ray defined in same coordinate system as this box intersects this bounding box.

**Parameters:**

*start* position of start of ray in local space.  
*end* position of end of ray in local space.

**Returns:**

[RayIntersectionInfo](#). If result.value is false then something was wrong like using this function with an uninitialized bounding box. If result.intersected is true then the ray intersected the box and result.position is the exact point of intersection.

[BoundingBox](#) o3d.BoundingBox.mul ( [Matrix4](#) matrix )

Multiplies the bounding box by the given matrix returning a new bounding box.

**Parameters:**

*matrix* The matrix to multiply by.

**Returns:**

The new bounding box.

---

## Member Data Documentation

### Array [o3d.BoundingBox.marshaled](#)

Assigns a JavaScript array of arrays of numbers into a [BoundingBox](#). This property is implicitly invoked whenever such an array is assigned to a [BoundingBox](#) property or such an array is passed as a [BoundingBox](#) method parameter.

This property is write-only.

### Point3 [o3d.BoundingBox.maxExtent](#)

The max extent of the box.

This property is read-only.

#### Point3 [o3d.BoundingBox.minExtent](#)

The min extent of the box.

This property is read-only.

#### bool [o3d.BoundingBox.valid](#)

True if this boundingbox has been initialized.

This property is read-only.

# **o3d.ParamBoundingBox Class Reference**

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) which stores a [BoundingBox](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [BoundingBox value](#)
- bool [updateInput](#)

- bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`  
Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`  
Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`  
The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`  
The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections [inherited]`  
The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[BoundingBox](#) [o3d.ParamBoundingBox.value](#)

The [BoundingBox](#) stored by the [Param](#).

## o3d.Buffer Class Reference

Inherits [o3d.NamedObject](#).

Inherited by [o3d.IndexBuffer](#), and [o3d.VertexBufferBase](#).

[List of all members.](#)

---

# Detailed Description

The [Buffer](#) object is a low level container for a flat list of floating point or integer values. These are currently used to define geometry.

## Public Member Functions

- bool [allocateElements](#) ( Number num\_elements)
- [Field createField](#) (String field\_type, Number num\_components)
- [removeField](#) ([Field](#) field)
- bool [set](#) ([RawData](#) raw\_data)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numElements](#)
  - Number [totalComponents](#)
  - Array [fields](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Buffer.allocateElements ( Number *num\_elements* )

Allocates memory for the data to be stored in the buffer based on the types of fields set on the buffer.

### Parameters:

*num\_elements* Number of elements to allocate..

### Returns:

True if operation was successful.

[Field](#) o3d.Buffer.createField ( String *field\_type*,

```
        Number num_components
    )
```

Defines a field on this buffer.

Note: Creating a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to make room for the new field.

**Parameters:**

*field\_type* type of data in the field. Valid types are "FloatField", "UInt32Field",  
"UByteNField".

*num\_components* number of components in the field.

**Returns:**

The created field.

```
bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]
```

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

```
o3d.Buffer.removeField ( Field field )
```

Removes a field from this buffer.

Note: Removing a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to remove the old field.

**Parameters:**

*field* field to remove.

```
bool o3d.Buffer.set ( RawData raw_data,
                      size_t     offset,
                      size_t     length
                  )
```

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.  
*offset* is a byte offset from the start of *raw\_data*  
*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

bool o3d.Buffer.set ( [RawData](#) *raw\_data* )

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Buffer.fields](#)

The fields currently set on the buffer.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Buffer.numElements](#)

Number of elements in the buffer.

This property is read-only.

Number [o3d.Buffer.totalComponents](#)

The total components in all fields in this buffer.

This property is read-only.

## o3d.VertexBufferBase Class Reference

Inherits [o3d.Buffer](#).

Inherited by [o3d.SourceBuffer](#), and [o3d.VertexBuffer](#).

[List of all members](#).

---

### Detailed Description

[VertexBufferBase](#) is a the base class for both [VertexBuffer](#) and [SourceBuffer](#)

**See also:**

[VertexBuffer](#)

[SourceBuffer](#)

### Public Member Functions

- Array [get\(\)](#)

- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [set](#) (Array values)
- [setAt](#) ( Number start\_index, Array values)
- bool [allocateElements](#) ( Number num\_elements)
- [Field createField](#) (String field\_type, Number num\_components)
- [removeField](#) ([Field](#) field)
- bool [set](#) ([RawData](#) raw\_data)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numElements](#)
  - Number [totalComponents](#)
  - Array [fields](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`bool o3d.Buffer.allocateElements ( Number num_elements ) [inherited]`

Allocates memory for the data to be stored in the buffer based on the types of fields set on the buffer.

### Parameters:

*num\_elements* Number of elements to allocate..

### Returns:

True if operation was successful.

[Field](#) `o3d.Buffer.createField ( String field_type,`  
`Number num_components`  
`) [inherited]`

Defines a field on this buffer.

Note: Creating a field after having allocated the buffer is an expensive operation as the data currently in

the buffer has to be shuffled around to make room for the new field.

**Parameters:**

*field\_type* type of data in the field. Valid types are "FloatField", "UInt32Field",  
"UByteNField".

*num\_components* number of components in the field.

**Returns:**

The created field.

Array o3d.VertexBufferBase.get ( )

Gets a copy of the values of the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Returns:**

An array of values.

Array o3d.VertexBufferBase.getValueAt ( Number *start\_index*,  
Number *num\_elements*  
)

Gets a copy of a sub range of the values in the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Parameters:**

*start\_index* index of the element value to get.  
*num\_elements* the number of elements to get.

**Returns:**

An array of values.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.Buffer.removeField ( [Field](#) *field* ) [inherited]

Removes a field from this buffer.

Note: Removing a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to remove the old field.

#### Parameters:

*field* field to remove.

bool o3d.Buffer.set ( [RawData](#) *raw\_data*,  
                      size\_t *offset*,  
                      size\_t *length*  
                      )

[inherited]

Sets the values in the buffer given a [RawData](#) object.

#### Parameters:

*raw\_data* contains data to assign to the [Buffer](#) data itself.

*offset* is a byte offset from the start of *raw\_data*

*length* is the byte length of the data to set

#### Returns:

True if operation was successful.

bool o3d.Buffer.set ( [RawData](#) *raw\_data* ) [inherited]

Sets the values in the buffer given a [RawData](#) object.

#### Parameters:

*raw\_data* contains data to assign to the [Buffer](#) data itself.

bool o3d.VertexBufferBase.set ( [Array](#) *values* )

Sets the values of the data stored in the buffer. The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

#### Parameters:

*values* Values to be stored in the buffer.

#### Returns:

True if operation was successful.

o3d.VertexBufferBase.setAt ( Number *start\_index*,

[Array](#) *values*

)

Sets the values of the data stored in the buffer. The buffer must have already been created either through buffer.set or buffer.allocateElements

The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

**Parameters:**

*start\_index* index of first value to set.  
*values* Values to be stored in the buffer starting at index.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Buffer.fields](#) [inherited]

The fields currently set on the buffer.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Buffer.numElements](#) [inherited]

Number of elements in the buffer.

This property is read-only.

Number [o3d.Buffer.totalComponents](#) [inherited]

The total components in all fields in this buffer.

This property is read-only.

## o3d.VertexBuffer Class Reference

Inherits [o3d.VertexBufferBase](#).

[List of all members.](#)

---

### Detailed Description

[VertexBuffer](#) is a [Buffer](#) object used for storing vertex data for geometry. (e.g. vertex positions, normals, colors, etc). A [VertexBuffer](#) can be rendered directly by the GPU.

See also:

[SourceBuffer](#)

### Public Member Functions

- Array [get\(\)](#)
- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [set](#) (Array values)
- bool [set](#) ([RawData](#) raw\_data)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- [setAt](#) ( Number start\_index, Array values)
- bool [allocateElements](#) ( Number num\_elements)
- [Field](#) [createField](#) (String field\_type, Number num\_components)

- [removeField](#) ([Field](#) field)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numElements](#)
  - Number [totalComponents](#)
  - Array [fields](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`bool o3d.Buffer.allocateElements ( Number num_elements ) [inherited]`

Allocates memory for the data to be stored in the buffer based on the types of fields set on the buffer.

**Parameters:**

*num\_elements* Number of elements to allocate..

**Returns:**

True if operation was successful.

[Field](#) `o3d.Buffer.createField ( String field_type,`  
`Number num_components`  
`) [inherited]`

Defines a field on this buffer.

Note: Creating a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to make room for the new field.

**Parameters:**

*field\_type* type of data in the field. Valid types are "FloatField", "UInt32Field",  
 "UByteNField".

*num\_components* number of components in the field.

**Returns:**

The created field.

Array o3d.VertexBufferBase.get ( ) [inherited]

Gets a copy of the values of the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Returns:**

An array of values.

Array o3d.VertexBufferBase.getValueAt ( Number *start\_index*,  
Number *num\_elements*  
) [inherited]

Gets a copy of a sub range of the values in the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Parameters:**

*start\_index* index of the element value to get.  
*num\_elements* the number of elements to get.

**Returns:**

An array of values.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.Buffer.removeField ( [Field](#) *field* ) [inherited]

Removes a field from this buffer.

Note: Removing a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to remove the old field.

**Parameters:**

*field* field to remove.

```
bool o3d.Buffer.set ( RawData raw_data,
                      size_t      offset,
                      size_t      length
                  ) [inherited]
```

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

*offset* is a byte offset from the start of *raw\_data*

*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

```
bool o3d.Buffer.set ( RawData raw_data ) [inherited]
```

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

```
bool o3d.VertexBufferBase.set ( Array values ) [inherited]
```

Sets the values of the data stored in the buffer. The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

**Parameters:**

*values* Values to be stored in the buffer.

**Returns:**

True if operation was successful.

```
o3d.VertexBufferBase.setAt ( Number start_index,
```

Array values

```
                  ) [inherited]
```

Sets the values of the data stored in the buffer. The buffer must have already been created either through `buffer.set` or `buffer.allocateElements`

The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

**Parameters:**

*start\_index* index of first value to set.

*values* Values to be stored in the buffer starting at index.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Buffer.fields](#) [inherited]

The fields currently set on the buffer.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Buffer.numElements](#) [inherited]

Number of elements in the buffer.

This property is read-only.

Number [o3d.Buffer.totalComponents](#) [inherited]

The total components in all fields in this buffer.

This property is read-only.

# o3d.SourceBuffer Class Reference

Inherits [o3d.VertexBufferBase](#).

[List of all members.](#)

---

## Detailed Description

[SourceBuffer](#) is a [Buffer](#) object used for storing vertex data for geometry. (e.g. vertex positions, normals, colors, etc).

A [SourceBuffer](#) is the source for operations like skinning and morph targets. It can not be directly rendered by the GPU.

**See also:**

[VertexBuffer](#)

## Public Member Functions

- Array [get\(\)](#)
- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [set](#) (Array values)
- bool [set](#) ([RawData](#) raw\_data)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- [setAt](#) ( Number start\_index, Array values)
- bool [allocateElements](#) ( Number num\_elements)
- [Field](#) [createField](#) (String field\_type, Number num\_components)
- [removeField](#) ([Field](#) field)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numElements](#)
- Number [totalComponents](#)

- Array [fields](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Buffer.allocateElements ( Number *num\_elements* ) [inherited]

Allocates memory for the data to be stored in the buffer based on the types of fields set on the buffer.

**Parameters:**

*num\_elements* Number of elements to allocate..

**Returns:**

True if operation was successful.

[Field](#) o3d.Buffer.createField ( String *field\_type*,  
Number *num\_components*  
)

[inherited]

Defines a field on this buffer.

Note: Creating a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to make room for the new field.

**Parameters:**

*field\_type* type of data in the field. Valid types are "FloatField", "UInt32Field",  
"UByteNField".

*num\_components* number of components in the field.

**Returns:**

The created field.

Array o3d.VertexBufferBase.get ( ) [inherited]

Gets a copy of the values of the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Returns:**

An array of values.

```
Array o3d.VertexBufferBase.getValueAt ( Number start_index,
                                         Number num_elements
                                         )
                                         [inherited]
```

Gets a copy of a sub range of the values in the data stored in the buffer. Modifying this copy has no effect on the buffer.

**Parameters:**

*start\_index* index of the element value to get.  
*num\_elements* the number of elements to get.

**Returns:**

An array of values.

```
bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]
```

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

```
o3d.Buffer.removeField ( Field field ) [inherited]
```

Removes a field from this buffer.

Note: Removing a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to remove the old field.

**Parameters:**

*field* field to remove.

```
bool o3d.Buffer.set ( RawData raw_data,
                      size_t offset,
                      size_t length
                    )
                    [inherited]
```

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.  
*offset* is a byte offset from the start of *raw\_data*  
*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

bool o3d.Buffer.set ( [RawData](#) *raw\_data* ) [inherited]

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

bool o3d.VertexBufferBase.set ( Array *values* ) [inherited]

Sets the values of the data stored in the buffer. The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

**Parameters:**

*values* Values to be stored in the buffer.

**Returns:**

True if operation was successful.

o3d.VertexBufferBase.setAt ( Number *start\_index*,  
                            Array *values*  
                            ) [inherited]

Sets the values of the data stored in the buffer. The buffer must have already been created either through buffer.set or buffer.allocateElements

The number of values passed in must be a multiple of the number of components needed for the fields defined on this buffer.

**Parameters:**

*start\_index* index of first value to set.

*values* Values to be stored in the buffer starting at index.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#)** [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Array [o3d.Buffer.fields](#)** [inherited]

The fields currently set on the buffer.

This property is read-only.

**String [o3d.NamedObject.name](#)** [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Number [o3d.Buffer.numElements](#)** [inherited]

Number of elements in the buffer.

This property is read-only.

**Number [o3d.Buffer.totalComponents](#)** [inherited]

The total components in all fields in this buffer.

This property is read-only.

# o3d.IndexBuffer Class Reference

Inherits [o3d.Buffer](#).

[List of all members.](#)

---

## Detailed Description

[IndexBuffer](#) is a buffer object used for storing geometry index data (e.g. triangle indices).

## Public Member Functions

- bool [set](#) (Array values)
- [setAt](#) ( Number start\_index, Array values)
- bool [allocateElements](#) ( Number num\_elements)
- [Field createField](#) (String field\_type, Number num\_components)
- [removeField](#) ([Field](#) field)
- bool [set](#) ([RawData](#) raw\_data)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numElements](#)
  - Number [totalComponents](#)
  - Array [fields](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Buffer.allocateElements ( Number *num\_elements* ) [inherited]

Allocates memory for the data to be stored in the buffer based on the types of fields set on the buffer.

## Parameters:

*num\_elements* Number of elements to allocate..

## Returns:

True if operation was successful.

Field o3d.Buffer.createField ( String *field\_type*,  
Number *num\_components*  
)

[inherited]

Defines a field on this buffer.

Note: Creating a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to make room for the new field.

## Parameters:

*field\_type* type of data in the field. Valid types are "FloatField", "UInt32Field",  
"UByteNField".

*num\_components* number of components in the field.

## Returns:

The created field.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

`o3d.Buffer.removeField ( Field field ) [inherited]`

Removes a field from this buffer.

Note: Removing a field after having allocated the buffer is an expensive operation as the data currently in the buffer has to be shuffled around to remove the old field.

**Parameters:**

*field* field to remove.

`bool o3d.Buffer.set ( RawData raw_data,  
                      size_t     offset,  
                      size_t     length  
                     ) [inherited]`

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

*offset* is a byte offset from the start of *raw\_data*

*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

`bool o3d.Buffer.set ( RawData raw_data ) [inherited]`

Sets the values in the buffer given a [RawData](#) object.

**Parameters:**

*raw\_data* contains data to assign to the [Buffer](#) data itself.

`bool o3d.IndexBuffer.set ( Array values )`

Sets the values of the data stored in the buffer.

**Parameters:**

*values* Values to be stored in the buffer.

**Returns:**

True if operation was successful.

`o3d.IndexBuffer.setAt ( Number start_index,  
                      Array    values  
                     )`

Sets the values of the data stored in the buffer. The buffer must have already been created either through `buffer.set` or `buffer.allocateElements`.

**Parameters:**

---

*start\_index* index of first value to set.  
*values* Values to be stored in the buffer starting at index.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Buffer.fields](#) [inherited]

The fields currently set on the buffer.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Buffer.numElements](#) [inherited]

Number of elements in the buffer.

This property is read-only.

Number [o3d.Buffer.totalComponents](#) [inherited]

The total components in all fields in this buffer.

This property is read-only.

# o3d.Canvas Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

[Canvas](#) provides an interface for drawing text and 2D primitives on a 2D surface. The contents of the canvas surface can be transferred to a compatible [Texture2D](#) object via the [copyToTexture\(\)](#) method. Each [Canvas](#) object maintains a stack of 2D transformation matrices which allow fine control over the placement of drawable elements. Both geometry and drawing coordinates provided to every draw call are transformed by the concatenation of all matrices in the stack.

## Public Member Functions

- bool [setSize](#) (Number [width](#), Number [height](#))
- [clear](#) ([Float4](#) color)
- [drawRect](#) (Number left, Number top, Number right, Number bottom, [CanvasPaint](#) paint)
- [drawText](#) (String text, Number x, Number y, [CanvasPaint](#) paint)
- [drawTextOnPath](#) (String text, Array positions, Number horizontalOffset, Number verticalOffset, [CanvasPaint](#) paint)
- [drawBitmap](#) ([Texture2D](#) texture, Number left, Number bottom)
- [saveMatrix](#) ()
- [restoreMatrix](#) ()
- [rotate](#) (Number degrees)
- [scale](#) (Number sx, Number sy)
- [translate](#) (Number dx, Number dy)
- [copyToTexture](#) ([Texture2D](#) texture)

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [width](#)
  - Number [height](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.Canvas.clear ( Float4 color )`

Clears the bitmap's pixels with the specified color.

### Parameters:

*color* The color to clear the bitmap with, provided as an array of four values [red, green, blue, alpha]. All values should be between 0.0 and 1.0. For alpha values 0.0 is transparent and 1.0 opaque.

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

`o3d.Canvas.copyToTexture ( Texture2D texture )`

Copies the contents of the [Canvas](#) to a 2D texture object. The size of the texture must match exactly the size of the [Canvas](#) set by the [setSize\(\)](#) method. The format of the texture must be set to either ARGB8 or XRGB8.

## Parameters:

*texture* [Texture](#) to copy to.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
)

[inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',

- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

```
o3d.Canvas.drawBitmap ( Texture2D texture,
                      Number    left,
                      Number    bottom
                    )
```

Draws the contents of the specified texture onto the canvas surface. The bottom left corner of the bitmap will be at (x, y) and transformed by the current matrix. Only ARGB8 and XRGB8 textures are supported. For XRG8 textures, Alpha is assumed to be 1 (opaque).

## Parameters:

*texture* The [Texture2D](#) object where the bitmap is extracted from.  
*left* The position of the left side of the bitmap.  
*bottom* The position of the bottom side of the bitmap.

```
o3d.Canvas.drawRect ( Number    left,  
                      Number    top,  
                      Number    right,  
                      Number    bottom,  
                      CanvasPaint paint  
                    )
```

Draws the specified rectangle using the specified paint. The rectangle will be filled based on the color and shader in the paint.

## Parameters:

*left* The left side of the rectangle to be drawn  
*top* The top side of the rectangle to be drawn  
*right* The right side of the rectangle to be drawn  
*bottom* The bottom side of the rectangle to be drawn  
*paint* The paint used to draw the rectangle

```
o3d.Canvas.drawText ( String      text,  
                      Number     x,  
                      Number     y,  
                      CanvasPaint paint  
                    )
```

Draws the text, with origin at (x,y), using the specified paint. The origin is interpreted based on the textAlign property in the paint.

## Parameters:

*text* String of text to be drawn  
*x* The x coordinate for the text origin  
*y* The y coordinate for the text origin  
*paint* The [CanvasPaint](#) object that specifies the text style, size, etc

```
o3d.Canvas.drawTextOnPath ( String      text,  
                           Array       positions,  
                           Number     horizontalOffset,  
                           Number     verticalOffset,  
                           CanvasPaint paint  
                         )
```

Draws the text with its baseline along the specified path. The paint's textAlign property determines where along the path to start the text. The path must contain at least two positions.

## Parameters:

*text* String of text to be drawn  
*positions* An array of x,y positions making up the path.  
*horizontalOffset* The distance along the path to add to the text starting position.  
*verticalOffset* The distance above(-) or below(+) the path to position the text.  
*paint* The [CanvasPaint](#) object that specifies the text style, size, etc.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

o3d.Canvas.restoreMatrix ( )

Balances a call to [saveMatrix\(\)](#), and removes modifications to matrix since the last save call. It is an error to call this more than previous calls to [saveMatrix\(\)](#).

`o3d.Canvas.rotate ( Number degrees )`  
Adds a rotation to the current canvas matrix.

**Parameters:**

*degrees* The amount to rotate, in degrees

`o3d.Canvas.saveMatrix ( )`

This call saves the current matrix and pushes a copy onto a private stack. Subsequent calls to translate, scale, rotate all operate on this copy. When the balancing call to [restoreMatrix\(\)](#) is made, this copy is deleted and the previous matrix is restored.

`o3d.Canvas.scale ( Number sx,  
                  Number sy  
                  )`

Adds a scale to the current canvas matrix.

**Parameters:**

*sx* The amount to scale in x  
*sy* The amount to scale in y

`bool o3d.Canvas.setSize ( Number width,  
                      Number height  
                      )`

Sets the size of the bitmap area that the [Canvas](#) uses.

**Parameters:**

*width* The width of the bitmap.  
*height* The height of the bitmap.

**Returns:**

true if the [Canvas](#) surface was allocated successfully.

`o3d.Canvas.translate ( Number dx,  
                      Number dy  
                      )`

Adds a translation to the current canvas matrix.

**Parameters:**

*dx* The amount to translate in x  
*dy* The amount to translate in y

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.Canvas.height](#)

The height of the bitmap used by the [Canvas](#) (read only).

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;  
for (var i = 0; i < params.length; i++) {  
    var param = params[i];  
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while

modifications to the array's members **will** affect them.

This property is read-only.

Number [o3d.Canvas.width](#)

The width of the bitmap used by the [Canvas](#) (read only).

This property is read-only.

## o3d.CanvasFontMetrics Class Reference

[List of all members.](#)

---

### Detailed Description

[CanvasFontMetrics](#) is used to return values, measured in pixels, describing the properties of a font used by the [CanvasPaint](#) objects. All the properties of [CanvasFontMetrics](#) are read-only.

### Public Attributes

- Number [top](#)
  - Number [ascent](#)
  - Number [descent](#)
  - Number [bottom](#)
  - Number [leading](#)
- 

### Member Data Documentation

Number [o3d.CanvasFontMetrics.ascent](#)

The recommended distance above the baseline (will be  $\leq 0$ )

This property is read-only.

Number [o3d.CanvasFontMetrics.bottom](#)

The greatest distance below the baseline for any glyph (will be  $\geq 0$ )

This property is read-only.

Number [o3d.CanvasFontMetrics.descent](#)

The recommended distance below the baseline (will be  $\geq 0$ )

This property is read-only.

Number [o3d.CanvasFontMetrics.leading](#)

The recommended distance to add between lines of text (will be  $\geq 0$ )

This property is read-only.

Number [o3d.CanvasFontMetrics.top](#)

The greatest distance above the baseline for any glyph (will be  $\leq 0$ )

This property is read-only.

## **o3d.CanvasPaint Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

### **Detailed Description**

The [CanvasPaint](#) class is used for specifying how to draw objects and text to a canvas.

### **Public Types**

- enum [Style](#)
- enum [TextAlign](#)

## Public Member Functions

- [setOutline](#) (Number radius, [Float4 color](#))
- [setShadow](#) (Number radius, Number offset\_x, Number offset\_y, [Float4 color](#))
- [CanvasFontMetrics getFontMetrics](#) ()
- [Float4 measureText](#) (String text)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float4 color](#)
  - Number [textSize](#)
  - String [textTypeface](#)
  - [Style textStyle](#)
  - [TextAlign textAlign](#)
  - [CanvasShader shader](#)
  - [CanvasFontMetrics fontMetrics](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.CanvasPaint.Style](#)

- NORMAL,
- BOLD,
- ITALIC,

- **BOLD\_ITALIC** Text styles

enum [o3d.CanvasPaint.TextAlign](#)

- LEFT,
  - CENTER,
  - RIGHT, Text alignment options
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',

- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',

- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[CanvasFontMetrics](#) o3d.CanvasPaint.getFontMetrics ( )

Returns metrics describing the font currently set on this paint object.

**Returns:**

The font metrics.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

[Float4](#) o3d.CanvasPaint.measureText ( String *text* )

Returns the bounds of the given text string when rendered with this paint. The bounds are returned as an array containing [left, top, right, bottom] values relative to (0, 0).

**Parameters:**

*text* The string of text to be measured.

**Returns:**

The bounds of text.

```
bool o3d.ParamObject.removeParam ( Param param ) [inherited]  
Removes a Param from a ParamObject.
```

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

```
o3d.CanvasPaint.setOutline ( Number radius,  
                            Float4 color  
                           )
```

Sets the color and radius of an outline around the text. Setting the radius to 0 cancels the outline effect.

The outline and shadow effects are mutually exclusive.

**Parameters:**

*radius* Distance outward from object to draw the background

*color* Color of the outline

```
o3d.CanvasPaint.setShadow ( Number radius,  
                           Number offset_x,  
                           Number offset_y,  
                           Float4 color  
                          )
```

Create a blur shadow effect on this paint. Setting the radius to 0 cancels the shadow effect.

**Parameters:**

*radius* radius to blur the paint

*offset\_y* offset of the blur in X

*offset\_x* offset of the blur in Y

*color* color for the blur

---

## Member Data Documentation

**String [o3d.ObjectBase.className](#)** [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#)** [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**[Float4 o3d.CanvasPaint.color](#)**

The color used for all the draw operations using this paint.

**[CanvasFontMetrics o3d.CanvasPaint.fontMetrics](#)**

Metrics of the current font used by the paint object.

This property is read-only.

**String [o3d.NamedObject.name](#)** [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), `Pack.getObject`, [RenderNode.getRenderNodesByNameInTree](#) and

`RenderNode.getTransformsByNameInTree` which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array [o3d.ParamObject.params](#)** [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;  
for (var i = 0; i < params.length; i++) {  
    var param = params[i];  
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### [CanvasShader o3d.CanvasPaint.shader](#)

The 2D shader used by this paint. Set to null to stop using a shader.

#### [TextAlign o3d.CanvasPaint.textAlign](#)

The alignment mode used for drawing text.

#### [Number o3d.CanvasPaint.fontSize](#)

The size of the font used for drawing text.

#### [Style o3d.CanvasPaint.textStyle](#)

The style applied to the text (e.g. italic, bold, etc)

#### [String o3d.CanvasPaint.typeface](#)

The font typeface used for drawing text. Passing an empty string will revert to the default font.

# **o3d.CanvasShader Class Reference**

Inherits [o3d.ParamObject](#).

Inherited by [o3d.CanvasLinearGradient](#).

[List of all members.](#)

---

## **Detailed Description**

[CanvasShader](#) is the base class of 2D gradient shaders that can be applied to a [CanvasPaint](#). When a shader is assigned to a [CanvasPaint](#) object, all subsequent drawing (text and objects) will use the shader pattern as a fill material.

## Public Types

- enum [TileMode](#)

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.CanvasShader.TileMode](#)

- CLAMP copy the edge color if the shader draws outside of its bounds
  - REPEAT repeat horizontally and vertically outside its bounds
  - MIRROR same as above, alternating mirror images
- 

## Member Function Documentation

[o3d.ParamObject.copyParams](#) ( [ParamObject](#) source\_param\_object ) [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

Param o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
)

[inherited]

Creates a Param with the given name and type on the ParamObject. Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the Param to be created.

*param\_type\_name* The type of Param to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',

- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
 Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## **o3d.CanvasLinearGradient Class Reference**

Inherits [o3d.CanvasShader](#).

[List of all members.](#)

---

### **Detailed Description**

A shader that generates a linear gradient between two specified points. Two or more colors need to be specified for the gradient.

## Public Types

- enum [TileMode](#)

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float2 startPoint](#)
  - [Float2 endPoint](#)
  - Array [colors](#)
  - Array [positions](#)
  - [CanvasShader](#) [TileMode tileMode](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.CanvasShader.TileMode](#) [inherited]

- CLAMP copy the edge color if the shader draws outside of its bounds
  - REPEAT repeat horizontally and vertically outside its bounds
  - MIRROR same as above, alternating mirror images
-

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',

- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Array** [o3d.CanvasLinearGradient.colors](#)

The array of colors to be distributed between the two points in RBGA format stored as an array of [r, g, b, a] colors.

**Float2** [o3d.CanvasLinearGradient.endPoint](#)

The end point of this gradient.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

**Array** [o3d.CanvasLinearGradient.positions](#)

The relative position of each corresponding color in the color array. Values must begin with 0 and end with 1.0 and there should be exactly as many numbers as there are colors. If positions is set to an empty

array then the colors are distributed evenly between between the start and end point.

#### [Float2 o3d.CanvasLinearGradient.startPoint](#)

The start point of this gradient.

#### [CanvasShader TileMode o3d.CanvasLinearGradient.tileMode](#)

The TileMode of this gradient which controls how the gradient pattern repeats.

# **o3d.ClearBuffer Class Reference**

Inherits [o3d.RenderNode](#).

[List of all members.](#)

---

## Detailed Description

A [ClearBuffer](#) is a render node that clears the color buffer, zbuffer and/or stencil buffer of the current render target.

## Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)
- Array [getRenderNodesByClassNameInTree \(String class\\_name\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- bool [removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Float4 clearColor](#)

- bool [clearColorFlag](#)
  - Number [clearDepth](#)
  - bool [clearDepthFlag](#)
  - Number [clearStencil](#)
  - bool [clearStencilFlag](#)
  - Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',

- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',

- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array  
o3d.RenderNode.getRenderNodesByClassNameInTree ( String *class\_name* ) [inherited]  
Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* ) [inherited]  
Searches for render nodes that match the given name in the hierarchy under and including this render

node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]

Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

[Float4 o3d.ClearBuffer.clearColor](#)

The color to clear the buffer in RGBA [Float4](#) format.

bool [o3d.ClearBuffer.clearColorFlag](#)

true clears the current render target's color buffer to the clear color. false does not clear the color buffer.

### Number [o3d.ClearBuffer.clearDepth](#)

The value to clear the depth buffer (0.0 to 1.0)

### bool [o3d.ClearBuffer.clearDepthFlag](#)

true clears the current render target's depth buffer to the clear depth value. false does not clear the depth buffer.

### Number [o3d.ClearBuffer.clearStencil](#)

The value to clear the stencil buffer to (0 - 255).

### bool [o3d.ClearBuffer.clearStencilFlag](#)

true clears the current render target's stencil buffer to the clear stencil value. false does not clear the stencil buffer

### Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

### String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

### Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode](#) [o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under parent\_node. Setting parent\_node to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

Number [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

# o3d.Renderer Class Reference

[List of all members.](#)

---

## Detailed Description

The [Renderer](#) class provides the abstract interface to each platform's rendering library.

## Public Types

- enum [InitStatus](#)
  - enum [DisplayModes](#)
- 

## Member Enumeration Documentation

### enum [o3d.Renderer.DisplayModes](#)

This is used in SetFullscreenClickRegion to request the current display mode, such that the change to full-screen mode won't change the screen resolution or refresh rate.

- DISPLAY\_MODE\_DEFAULT

### enum [o3d.Renderer.InitStatus](#)

The initialization status of the renderer.

- UNINITIALIZED,
- SUCCESS, The renderer is initialized.
- GPU\_NOT\_UP\_TO\_SPEC, The renderer determined the user's machine cannot run O3D.
- OUT\_OF\_RESOURCES, The user's machine does not have enough graphic resources available to start another instance of the O3D renderer.
- INITIALIZATION\_ERROR, Some unknown error such as e.g. drivers not being installed correctly.

# **o3d.Client Class Reference**

[List of all members.](#)

---

## **Detailed Description**

The [Client](#) class is the main point of entry to O3D. It defines methods for creating and deleting packs. Each new object created by the [Client](#) is assigned a unique ID.

The [Client](#) has a root transform for the transform graph and a root render node for the render graph.

## **Public Types**

- enum [RenderMode](#)

## **Public Member Functions**

- [cleanup \(\)](#)
- [Pack createPack \(\)](#)
- [ObjectBase getObjectById \(Id id\)](#)
- Array [getObjects \(String name, String class\\_name\)](#)
- Array [getObjectsByClassName \(String class\\_name\)](#)
- [render \(\)](#)
- [renderTree \(RenderNode render\\_node\)](#)
- Array [getDisplayModes \(\)](#)

- [setFullscreenClickRegion](#) (Number x, Number y, Number [width](#), Number [height](#), Number mode\_id)
- [clearFullscreenClickRegion](#) ()
- [cancelFullscreenDisplay](#) ()
- [setRenderCallback](#) (RenderCallback render\_callback)
- [clearRenderCallback](#) ()
- [setPostRenderCallback](#) (RenderCallback post\_render\_callback)
- [clearPostRenderCallback](#) ()
- [setLostResourcesCallback](#) (LostResourcesCallback lost\_resources\_callback)
- [clearLostResourcesCallback](#) ()
- [setEventCallback](#) (String type, EventCallback handler)
- [clearEventCallback](#) (String type)
- [setErrorTexture](#) ([Texture](#) texture)
- [setTickCallback](#) (TickCallback tick\_callback)
- [clearTickCallback](#) ()
- [setErrorCallback](#) (ErrorCallback error\_callback)
- [clearErrorCallback](#) ()
- [invalidateAllParameters](#) ()
- bool [saveScreen](#) (String file\_name)
- String [getMessageQueueAddress](#) ()
- [clearLastError](#) ()
- [profileReset](#) ()
- String [profileToString](#) ()

## Public Attributes

- [Transform](#) root
- [RenderMode](#) renderMode
- bool [fullscreen](#)
- Number [width](#)
- Number [height](#)
- [RenderNode](#) renderGraphRoot
- [Renderer](#) InitStatus rendererInitStatus

- Cursor CursorType [cursor](#)
  - String [lastError](#)
  - Array [objects](#)
  - Id [clientId](#)
- 

## Member Enumeration Documentation

enum [o3d.Client.RenderMode](#)

- RENDERMODE\_CONTINUOUS, Draw as often as possible up to refresh rate.
  - RENDERMODE\_ON\_DEMAND, Draw once then only when the OS requests it (like uncovering part of a window.)
- 

## Member Function Documentation

`o3d.Client.cancelFullscreenDisplay ( )`

Cancels full-screen display, reverting to displaying content only in the plugin region. If the plugin is already not in full-screen mode, this has no effect. This does not deactivate the plugin click region--if the user clicks there again, we'll go back to full-screen display.

`o3d.Client.cleanup ( )`

Call this function from `window.onunload` to ensure the browser does not continue to call callbacks (like the render callback) after the page is unloaded. It is possible that during unload the browser unloads all the javascript code, but then, after that, still asks the plugin to render. The browser then calls javascript functions that no longer exist which causes an error. To prevent that situation you need to clear all your callbacks on unload. `cleanup` handles that for you so you don't have to dispose each and every callback by hand.

`o3d.Client.clearErrorCallback ( )`

Clears the Error callback

NOTE: The client takes ownership of the `ErrorCallback` you pass in. It will be deleted if you call `SetErrorCallback` a second time or if you call `ClearErrorCallback`.

`o3d.Client.clearEventCallback ( String type )`

Removes the previously-registered callback for an event of the given type.

**Parameters:**

*type* Type of event to clear callback for.

`o3d.Client.clearFullscreenClickRegion ( )`

Deactivates the plugin click region that was previously created with SetFullscreenClickRegion().

`o3d.Client.clearLastError ( )`

Clears the error returned by GetLastError.

`o3d.Client.clearLostResourcesCallback ( )`

Clears the lost resources callback.

`o3d.Client.clearPostRenderCallback ( )`

Clears the post render callback.

`o3d.Client.clearRenderCallback ( )`

Clears the per frame render callback.

`o3d.Client.clearTickCallback ( )`

Clears the tick callback

NOTE: The client takes ownership of the TickCallback you pass in. It will be deleted if you call

SetTickCallback a second time or if you call ClearTickCallback

Pack `o3d.Client.createPack ( )`

Creates a pack object.

**Returns:**

A pack object.

Array `o3d.Client.getDisplayModes ( )`

Returns an array of DisplayModes which are available for use in full-screen mode.

**Returns:**

An array of DisplayModes.

`String o3d.Client.getMessageQueueAddress ( )`

Returns the socket address of the IMC message queue associated with the [Client](#).

**Returns:**

The socket address.

[ObjectBase](#) o3d.Client.getObjectById ( Id *id* )

Searches the [Client](#) for an object matching the given id.

**Parameters:**

*id* The id of the object to look for.

**Returns:**

The object or null if a object with the given id is not found.

Array o3d.Client.getObjects ( String *name*,  
String *class\_name*  
)

Searches the [Client](#) for objects of a particular name and type.

**Parameters:**

*name* name of object to look for.

*class\_name* name of class to look for.

**Returns:**

Array of objects found.

Array o3d.Client.getObjectsByClassName ( String *class\_name* )

Searches the [Client](#) for objects of a particular type.

**Parameters:**

*class\_name* name of class to look for.

**Returns:**

Array of objects found.

o3d.Client.invalidateAllParameters ( )

Makes all parameters get re-evaluated.

o3d.Client.profileReset ( )

Resets the profiling information.

String o3d.Client.profileToString ( )

Returns the profiling information as a string.

**Returns:**

The profiling info.

`o3d.Client.render ( )`

Forces a render of the current scene if the current render mode is RENDERMODE\_ON\_DEMAND.

`o3d.Client.renderTree ( RenderNode render_node )`

Renders a render graph.

Normally the client calls this function automatically for you effectively doing a `client.renderTree(client.renderGraphRoot)` but there are cases where it is beneficial to be able to call this yourself and pass it different roots when you need to manipulate something between calls.

This function can only be called from inside a render callback. If you call it the client will not do its default call as mentioned above.

#### Parameters:

*render\_node* root [RenderNode](#) to start rendering from.

`bool o3d.Client.saveScreen ( String file_name )`

This function is only available in the test version of the plugin.

#### Parameters:

*file\_name* Name to save screenshot to.

#### Returns:

True on success.

`o3d.Client.setErrorCallback ( ErrorCallback error_callback )`

Sets a callback for when the client gets an error. For example when a shader is compiled and there is an error or if you attempt to bind a param to a param of an incompatible type.

NOTE: The client takes ownership of the ErrorCallback you pass in. It will be deleted if you call SetErrorCallback a second time or if you call ClearErrorCallback.

NOTE: The callback will not be called recursively. If you are in a callback, and do something that causes another error before you have returned from the callback, your callback will not be called a second time.

NOTE: If you put up an alert in response to an error it is best if you clear the error callback before you put up the alert. Otherwise you'll get an alert everytime the client tries to render which is every time you close the current alert which means you'll be in an infinite loop of alerts.

#### Parameters:

*error\_callback* ErrorCallback to call when the [Client](#) gets an error.

`o3d.Client.setErrorTexture ( Texture texture )`

Sets the texture to use when a [Texture](#) or [Sampler](#) is missing while rendering. The default is a red texture with a yellow no symbol. If you set it to null you'll get an error if you try to render something that is missing a needed [Texture](#), [Sampler](#) or [ParamSampler](#).

For example if you don't care about missing textures, setting it to a black texture would be one option. Another example is if you want to write all your shaders to expect a texture then set this to a white texture. If you want to make sure you are not missing any textures set it null and see if you get any errors using [Client.setErrorCallback](#) or [Client.lastError](#).

```
// Set the error texture to black.  
var t = g_pack.createTexture2D('', 1, 1, g_o3d.Texture.XRGB8, 1);  
t.set(0, [0, 0, 0]);  
g_client.setErrorTexture(t);
```

#### Parameters:

*texture* texture to use for missing textures or null.

`o3d.Client.setEventCallback ( String type,  
 EventCallback handler  
 )`

Sets a callback for a given event type. types. There can be only one callback for a given event type at a time; setting a new one deletes the old one.

#### Parameters:

*type* Type of event to set callback for.  
*handler* [Function](#) to call on event.

#### See also:

[Event](#) for a list of event

`o3d.Client.setFullscreenClickRegion ( Number x,  
 Number y,  
 Number width,  
 Number height,  
 Number mode_id  
 )`

Makes a region of the plugin area that will invoke full-screen mode if clicked. The developer is responsible for communicating this to the user, as this region has no visible marker. The developer is also responsible for updating this region if the plugin gets resized, as we don't know whether or how to scale it. There can be only one full-screen click region at a time; calling this again will override any

previous call.

**Parameters:**

*x*      x position in pixels.  
*y*      y position in pixels.  
*width*    width in pixels.  
*height*   height in pixels.  
*mode\_id* Id of mode to use.

`o3d.Client.setLostResourcesCallback ( LostResourcesCallback lost_resources_callback )`  
Sets the lost resources callback.

The contents of certain resources, RenderSurfaces, can get discarded by the system under certain circumstances. If your application needs that contents to be in a certain state then you can set a callback giving your program the opportunity to restore that state if and when it is lost.

**Parameters:**

*lost\_resources\_callback* The callback when resources are lost.

`o3d.Client.setPostRenderCallback ( RenderCallback post_render_callback )`  
Sets a render callback to be called at the end of the rendering cycle of each frame.

Note: The callback will not be called recursively. When your callback is called if you somehow manage to cause the client to render more frames before you've returned from the callback you will not be called for those frames.

```
g_client.setPostRenderCallback(onpostrender);

function onpostrender(render_event) {
    var elapsedTime = render_event.elapsedTime;

    // elapsedTime is the time elapsed since the last callback.
    // You can use this value to make your application frame rate independent.
    // For example:
    //    position = position + velocity_in_units_per_second * elapsedTime;
}
```

**Parameters:**

*post\_render\_callback* The callback to call each frame.

`o3d.Client.setRenderCallback ( RenderCallback render_callback )`  
Sets the per frame render callback.

Note: The callback will not be called recursively. When your callback is called if you somehow manage

to cause the client to render more frames before you've returned from the callback you will not be called for those frames.

```
g_client.setRenderCallback(onrender);

function onrender(render_event) {
    var elapsedTime = render_event.elapsedTime;

    // elapsedTime is the time elasped since the last callback.
    // You can use this value to make your application frame rate independent.
    // For example:
    //     position = position + velocity_in_units_per_second * elapsedTime;
}
```

#### Parameters:

*render\_callback* The callback to call each frame.

`o3d.Client.setTickCallback ( TickCallback tick_callback )`

Sets a callback for when the client ticks. The client processes some things like animation timers at up to 100hz. This callback will get called before each of those process ticks.

NOTE: The client takes ownership of the TickCallback you pass in. It will be deleted if you call SetTickCallback a second time or if you call ClearTickCallback.

Note: The callback will not be called recursively.

#### Parameters:

*tick\_callback* TickCallback to call when the [Client](#) ticks.

---

## Member Data Documentation

Id [`o3d.Client.clientId`](#)

A unique id for this client.

This property is read-only.

Cursor CursorType [`o3d.Client.cursor`](#)

Gets / Sets the cursor's shape.

bool [`o3d.Client.fullscreen`](#)

Whether content is displayed in full-screen mode or in a plugin window. The default is false [not full-

screen].

This property is read-only.

#### Number [o3d.Client.height](#)

Returns the height of the current drawing area [plugin or full-screen] in pixels.

This property is read-only.

#### String [o3d.Client.lastError](#)

The last error reported by the plugin.

This property is read-only.

#### Array [o3d.Client.objects](#)

All the objects managed by this client.

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var objects = client.objects;
for (var i = 0; i < objects.length; i++) {
    var object = objects[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [Client](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### [Renderer](#) InitStatus [o3d.Client.rendererInitStatus](#)

Returns the status of initializing the renderer so we can display the appropriate message. We require a certain minimum set of graphics capabilities. If the user's computer does not have his minimum set this will be GPU\_NOT\_UP\_TO\_SPEC. If the user is out of graphics resources this will be OUT\_OF\_RESOURCES. If some other error happened this will be INITIALIZATION\_ERROR. Otherwise it will be SUCCESS.

This property is read-only.

#### [RenderNode](#) [o3d.Client.renderGraphRoot](#)

The root of the render graph.

This property is read-only.

## [RenderMode o3d.Client.renderMode](#)

The current render mode. The default mode is RENDERMODE\_CONTINUOUS.

Valid values are:

- RENDERMODE\_CONTINUOUS, Draw as often as possible up to refresh rate.
- RENDERMODE\_ON\_DEMAND, Draw when the OS requests it (like uncovering part of a window.)

## [Transform o3d.Client.root](#)

The transform graph root [Transform](#)

This property is read-only.

## [Number o3d.Client.width](#)

Returns the width of the current drawing area [plugin or full-screen] in pixels.

This property is read-only.

# **o3d.Counter Class Reference**

Inherits [o3d.ParamObject](#).

Inherited by [o3d.RenderFrameCounter](#), [o3d.SecondCounter](#), and [o3d.TickCounter](#).

[List of all members.](#)

---

## **Detailed Description**

A [Counter](#) counts seconds, ticks or render frames depending on the type of counter. You can set where it starts counting from and where it stops counting at, whether or not it is running or paused and how it loops or does not loop. You can also give it callbacks to call at specific count values.

## **Public Types**

- enum [CountMode](#)

## Public Member Functions

- [setCount](#) (Number `count`)
- [reset](#) ()
- [advance](#) (Number `advance_amount`)
- [addCallback](#) (Number `count`, CounterCallback `counter_callback`)
- bool [removeCallback](#) (Number `count`)
- [removeAllCallbacks](#) ()
- Array [getCallbackCounts](#) ()
- [Param createParam](#) (String `param_name`, String `param_type_name`)
- [Param getParam](#) (String `param_name`)
- bool [removeParam](#) ([Param](#) `param`)
- [copyParams](#) ([ParamObject](#) `source_param_object`)
- bool [isAClassName](#) (String `class_name`)

## Public Attributes

- bool [running](#)
  - bool [forward](#)
  - Number [start](#)
  - Number [end](#)
  - Number [count](#)
  - [CountMode countMode](#)
  - Number [multiplier](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Counter.CountMode](#)

- CONTINUOUS, Keep running the counter forever.
  - ONCE, Stop at start or end depending on the direction.
  - CYCLE, When at end, jump back to start or visa versa.
  - OSCILLATE, Go from start to end back to start. };
- 

## Member Function Documentation

```
o3d.Counter.addCallback ( Number      count,
                         CounterCallback counter_callback
                       )
```

Adds a callback for a given count value. Only one callback can be added to a specific count value. If another callback is added with the same count value the previous callback for that value will be replaced.

Note: A callback at start will only get called when counting backward, a callback at end will only get called counting forward.

### Parameters:

*count* Count at which to call callback.  
*counter\_callback* Callback to call at given count.

```
o3d.Counter.advance ( Number advance_amount )
```

Advances the counter the given amount. The actual amount advanced depends on the forward and multiplier settings. The formula is

```
new_count = count + advance_amount * multiplier * (forward ? 1.0 :
-1.0);
```

Any callbacks that fall in the range between the counter's current count and the amount advanced will be called.

This function is normally called automatically by the client if the counter is set to running = true. but you can call it manually.

### Parameters:

*advance\_amount* Amount to advance count.

```
o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]
```

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any

currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

Param o3d.ParamObject.createParam ( String *param\_name*,  
   String *param\_type\_name*  
   )

[inherited]

Creates a Param with the given name and type on the ParamObject. Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the Param to be created.

*param\_type\_name* The type of Param to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',

- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

Array o3d.Counter.getCallbackCounts ( )

Returns all the counts for which all callback has been added.

**Returns:**

Array of counts.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.Counter.removeAllCallbacks ( )  
Removes all the callbacks on this counter.

bool o3d.Counter.removeCallback ( Number *count* )  
Removes a callback for a given count value.

**Parameters:**

*count* Count to remove callback for,

**Returns:**

true if there was a callback for that count, false if there was not a callback for that count.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

## Returns:

True if the param was removed.

`o3d.Counter.reset ( )`

Resets the counter back to the start or end time depending on the forward setting and also resets the Callback state. Note: Reset does not change the running state of the counter.

`o3d.Counter.setCount ( Number count )`

Sets the current count value for this counter as well as the resetting the state of the callbacks.

In other words. Assume start = 1, end = 5, count = 1, and you have a callback at 3.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();
```

```
myCounter.advance(2); // count is now 3, myCallback is called.
```

```
myCounter.advance(2); // count is now 5
```

vs.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();
```

```
myCounter.advance(2); // count is now 3, myCallback is called.
```

```
myCounter.setCount(3); // count is now 3, callback state has been  
reset. myCounter.advance(2); // count is now 5, myCallback is called.
```

In the second case myCallback was called twice.

## Parameters:

*count* Value to set the count to.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Number** [o3d.Counter.count](#)

The current count value for this counter. Default = 0.

This property is read-only.

[CountMode](#) [o3d.Counter.countMode](#)

The current count mode for this counter. Default = CONTINUOUS.

**Number** [o3d.Counter.end](#)

The end count for this counter. Default = 0.

**bool** [o3d.Counter.forward](#)

Which direction this counter is counting. Default = true.

**Number** [o3d.Counter.multiplier](#)

A multiplier used when advancing the count. The default value is 1.0. For example you could set this to 0.5 to run the counter at half speed or 2.0 to run it at double speed. Default = 1.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
```

```
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

bool [o3d.Counter.running](#)

Whether or not this counter is running. Default = true.

Number [o3d.Counter.start](#)

The start count for this counter. Default = 0.

## o3d.SecondCounter Class Reference

Inherits [o3d.Counter](#).

[List of all members.](#)

---

### Detailed Description

A [Counter](#) that counts seconds.

### Public Types

- enum [CountMode](#)

### Public Member Functions

- [setCount](#) (Number [count](#))
- [reset](#) ()
- [advance](#) (Number [advance\\_amount](#))
- [addCallback](#) (Number [count](#), CounterCallback [counter\\_callback](#))
- bool [removeCallback](#) (Number [count](#))
- [removeAllCallbacks](#) ()
- Array [getCallbackCounts](#) ()

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- bool [running](#)
  - bool [forward](#)
  - Number [start](#)
  - Number [end](#)
  - Number [count](#)
  - [CountMode countMode](#)
  - Number [multiplier](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

```
enum o3d.Counter.CountMode [inherited]
    • CONTINUOUS, Keep running the counter forever.
    • ONCE, Stop at start or end depending on the direction.
    • CYCLE, When at end, jump back to start or visa versa.
    • OSCILLATE, Go from start to end back to start. };
```

---

## Member Function Documentation

`o3d.Counter.addCallback ( Number count,`

CounterCallback *counter\_callback*

) [inherited]

Adds a callback for a given count value. Only one callback can be added to a specific count value. If another callback is added with the same count value the previous callback for that value will be replaced.

Note: A callback at start will only get called when counting backward, a callback at end will only get called counting forward.

#### Parameters:

*count* Count at which to call callback.

*counter\_callback* Callback to call at given count.

o3d.Counter.advance ( Number *advance\_amount* ) [inherited]

Advances the counter the given amount. The actual amount advanced depends on the forward and multiplier settings. The formula is

```
new_count = count + advance_amount * multiplier * (forward ? 1.0 : -1.0);
```

Any callbacks that fall in the range between the counter's current count and the amount advanced will be called.

This function is normally called automatically by the client if the counter is set to running = true. but you can call it manually.

#### Parameters:

*advance\_amount* Amount to advance count.

o3d.ParamObject.copyParams ( [ParamObject](#) *source\_param\_object* ) [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',

- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

Array o3d.Counter.getCallbackCounts ( ) [inherited]

Returns all the counts for which all callback has been added.

**Returns:**

Array of counts.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Counter.removeAllCallbacks ( ) [inherited]`

Removes all the callbacks on this counter.

`bool o3d.Counter.removeCallback ( Number count ) [inherited]`

Removes a callback for a given count value.

**Parameters:**

*count* Count to remove callback for,

**Returns:**

true if there was a callback for that count, false if there was not a callback for that count.

`bool o3d.ParamObject.removeParam ( Param param ) [inherited]`

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

`o3d.Counter.reset ( ) [inherited]`

Resets the counter back to the start or end time depending on the forward setting and also resets the Callback state. Note: Reset does not change the running state of the counter.

`o3d.Counter.setCount ( Number count ) [inherited]`

Sets the current count value for this counter as well as the resetting the state of the callbacks.

In other words. Assume start = 1, end = 5, count = 1, and you have a callback at 3.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,
```

```
myCallback); myCounter.reset();

myCounter.advance(2); // count is now 3, myCallback is called.

myCounter.advance(2); // count is now 5
```

vs.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,
myCallback); myCounter.reset();

myCounter.advance(2); // count is now 3, myCallback is called.

myCounter.setCount(3); // count is now 3, callback state has been
reset. myCounter.advance(2); // count is now 5, myCallback is called.
```

In the second case myCallback was called twice.

#### Parameters:

*count* Value to set the count to.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.Counter.count](#) [inherited]

The current count value for this counter. Default = 0.

This property is read-only.

[CountMode](#) [o3d.Counter.countMode](#) [inherited]

The current count mode for this counter. Default = CONTINUOUS.

Number [o3d.Counter.end](#) [inherited]

The end count for this counter. Default = 0.

bool [o3d.Counter.forward](#) [inherited]

Which direction this counter is counting. Default = true.

Number [o3d.Counter.multiplier](#) [inherited]

A multiplier used when advancing the count. The default value is 1.0. For example you could set this to 0.5 to run the counter at half speed or 2.0 to run it at double speed. Default = 1.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

bool [o3d.Counter.running](#) [inherited]

Whether or not this counter is running. Default = true.

Number [o3d.Counter.start](#) [inherited]

The start count for this counter. Default = 0.

# o3d.RenderFrameCounter Class Reference

Inherits [o3d.Counter](#).

[List of all members.](#)

---

## Detailed Description

A [Counter](#) that counts render frames.

## Public Types

- enum [CountMode](#)

## Public Member Functions

- [setCount](#) (Number [count](#))
- [reset](#) ()
- [advance](#) (Number [advance\\_amount](#))
- [addCallback](#) (Number [count](#), CounterCallback [counter\\_callback](#))
- bool [removeCallback](#) (Number [count](#))
- [removeAllCallbacks](#) ()
- Array [getCallbackCounts](#) ()
- [Param createParam](#) (String [param\\_name](#), String [param\\_type\\_name](#))
- [Param getParam](#) (String [param\\_name](#))
- bool [removeParam](#) ([Param](#) [param](#))
- [copyParams](#) ([ParamObject](#) [source\\_param\\_object](#))
- bool [isAClassName](#) (String [class\\_name](#))

## Public Attributes

- bool [running](#)

- bool [forward](#)
  - Number [start](#)
  - Number [end](#)
  - Number [count](#)
  - [CountMode countMode](#)
  - Number [multiplier](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

```
enum o3d.Counter.CountMode [inherited]
    • CONTINUOUS, Keep running the counter forever.
    • ONCE, Stop at start or end depending on the direction.
    • CYCLE, When at end, jump back to start or visa versa.
    • OSCILLATE, Go from start to end back to start. };
```

---

## Member Function Documentation

```
o3d.Counter.addCallback ( Number count,
                           CounterCallback counter_callback
                         ) [inherited]
```

Adds a callback for a given count value. Only one callback can be added to a specific count value. If another callback is added with the same count value the previous callback for that value will be replaced.

Note: A callback at start will only get called when counting backward, a callback at end will only get called counting forward.

### Parameters:

<i>count</i>	Count at which to call callback.
--------------	----------------------------------

*counter\_callback* Callback to call at given count.

`o3d.Counter.advance ( Number advance_amount ) [inherited]`

Advances the counter the given amount. The actual amount advanced depends on the forward and multiplier settings. The formula is

```
new_count = count + advance_amount * multiplier * (forward ? 1.0 : -1.0);
```

Any callbacks that fall in the range between the counter's current count and the amount advanced will be called.

This function is normally called automatically by the client if the counter is set to running = true, but you can call it manually.

#### Parameters:

*advance\_amount* Amount to advance count.

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',

- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',

- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

Array o3d.Counter.getCallbackCounts ( ) [inherited]

Returns all the counts for which all callback has been added.

**Returns:**

Array of counts.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Counter.removeAllCallbacks ( )` [inherited]

Removes all the callbacks on this counter.

`bool o3d.Counter.removeCallback ( Number count )` [inherited]

Removes a callback for a given count value.

**Parameters:**

*count* Count to remove callback for,

**Returns:**

true if there was a callback for that count, false if there was not a callback for that count.

`bool o3d.ParamObject.removeParam ( Param param )` [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

`o3d.Counter.reset ( )` [inherited]

Resets the counter back to the start or end time depending on the forward setting and also resets the Callback state. Note: Reset does not change the running state of the counter.

`o3d.Counter.setCount ( Number count )` [inherited]

Sets the current count value for this counter as well as the resetting the state of the callbacks.

In other words. Assume start = 1, end = 5, count = 1, and you have a callback at 3.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();
```

```
myCounter.advance(2); // count is now 3, myCallback is called.
```

```
myCounter.advance(2); // count is now 5
```

vs.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();
```

```
myCounter.advance(2); // count is now 3, myCallback is called.
```

```
myCounter.setCount(3); // count is now 3, callback state has been  
reset. myCounter.advance(2); // count is now 5, myCallback is called.
```

In the second case myCallback was called twice.

#### Parameters:

*count* Value to set the count to.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.Counter.count](#) [inherited]

The current count value for this counter. Default = 0.

This property is read-only.

[CountMode o3d.Counter.countMode](#) [inherited]

The current count mode for this counter. Default = CONTINUOUS.

Number [o3d.Counter.end](#) [inherited]

The end count for this counter. Default = 0.

bool [o3d.Counter.forward](#) [inherited]

Which direction this counter is counting. Default = true.

Number [o3d.Counter.multiplier](#) [inherited]

A multiplier used when advancing the count. The default value is 1.0. For example you could set this to 0.5 to run the counter at half speed or 2.0 to run it at double speed. Default = 1.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

bool [o3d.Counter.running](#) [inherited]

Whether or not this counter is running. Default = true.

Number [o3d.Counter.start](#) [inherited]

The start count for this counter. Default = 0.

## o3d.TickCounter Class Reference

Inherits [o3d.Counter](#).

[List of all members](#).

---

# Detailed Description

A [Counter](#) that counts ticks.

## Public Types

- enum [CountMode](#)

## Public Member Functions

- [setCount](#) (Number [count](#))
- [reset](#) ()
- [advance](#) (Number [advance\\_amount](#))
- [addCallback](#) (Number [count](#), CounterCallback [counter\\_callback](#))
- bool [removeCallback](#) (Number [count](#))
- [removeAllCallbacks](#) ()
- Array [getCallbackCounts](#) ()
- [Param createParam](#) (String [param\\_name](#), String [param\\_type\\_name](#))
- [Param getParam](#) (String [param\\_name](#))
- bool [removeParam](#) ([Param](#) [param](#))
- [copyParams](#) ([ParamObject](#) [source\\_param\\_object](#))
- bool [isAClassName](#) (String [class\\_name](#))

## Public Attributes

- bool [running](#)
- bool [forward](#)
- Number [start](#)
- Number [end](#)
- Number [count](#)
- [CountMode countMode](#)
- Number [multiplier](#)
- Array [params](#)
- String [name](#)

- Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Counter.CountMode](#) [inherited]

- CONTINUOUS, Keep running the counter forever.
  - ONCE, Stop at start or end depending on the direction.
  - CYCLE, When at end, jump back to start or visa versa.
  - OSCILLATE, Go from start to end back to start. };
- 

## Member Function Documentation

`o3d.Counter.addCallback ( Number count,  
                          CounterCallback counter_callback  
                          )` [inherited]

Adds a callback for a given count value. Only one callback can be added to a specific count value. If another callback is added with the same count value the previous callback for that value will be replaced.

Note: A callback at start will only get called when counting backward, a callback at end will only get called counting forward.

### Parameters:

*count* Count at which to call callback.  
*counter\_callback* Callback to call at given count.

`o3d.Counter.advance ( Number advance_amount )` [inherited]

Advances the counter the given amount. The actual amount advanced depends on the forward and multiplier settings. The formula is

```
new_count = count + advance_amount * multiplier * (forward ? 1.0 :  
-1.0);
```

Any callbacks that fall in the range between the counter's current count and the amount advanced will be called.

This function is normally called automatically by the client if the counter is set to running = true, but you can call it manually.

#### Parameters:

*advance\_amount* Amount to advance count.

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',

- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

Array o3d.Counter.getCallbackCounts ( ) [inherited]  
Returns all the counts for which all callback has been added.

**Returns:**

Array of counts.

Param o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a Param defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The Param with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.Counter.removeAllCallbacks ( ) [inherited]  
Removes all the callbacks on this counter.

bool o3d.Counter.removeCallback ( Number *count* ) [inherited]  
Removes a callback for a given count value.

**Parameters:**

*count* Count to remove callback for,

**Returns:**

true if there was a callback for that count, false if there was not a callback for that count.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

o3d.Counter.reset ( ) [inherited]

Resets the counter back to the start or end time depending on the forward setting and also resets the Callback state. Note: Reset does not change the running state of the counter.

o3d.Counter.setCount ( Number *count* ) [inherited]

Sets the current count value for this counter as well as the resetting the state of the callbacks.

In other words. Assume start = 1, end = 5, count = 1, and you have a callback at 3.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();  
  
myCounter.advance(2); // count is now 3, myCallback is called.  
myCounter.advance(2); // count is now 5
```

vs.

```
myCounter.start = 1; myCounter.end = 5; myCounter.addCallback(3,  
myCallback); myCounter.reset();  
  
myCounter.advance(2); // count is now 3, myCallback is called.  
myCounter.setCount(3); // count is now 3, callback state has been  
reset. myCounter.advance(2); // count is now 5, myCallback is called.
```

In the second case myCallback was called twice.

**Parameters:**

*count* Value to set the count to.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.Counter.count](#) [inherited]

The current count value for this counter. Default = 0.

This property is read-only.

CountMode [o3d.Counter.countMode](#) [inherited]

The current count mode for this counter. Default = CONTINUOUS.

Number [o3d.Counter.end](#) [inherited]

The end count for this counter. Default = 0.

bool [o3d.Counter.forward](#) [inherited]

Which direction this counter is counting. Default = true.

Number [o3d.Counter.multiplier](#) [inherited]

A multiplier used when advancing the count. The default value is 1.0. For example you could set this to 0.5 to run the counter at half speed or 2.0 to run it at double speed. Default = 1.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

bool [o3d.Counter.running](#) [inherited]

Whether or not this counter is running. Default = true.

Number [o3d.Counter.start](#) [inherited]

The start count for this counter. Default = 0.

## o3d.CurveKey Class Reference

Inherits [o3d.ObjectBase](#).

Inherited by [o3d.BezierCurveKey](#), [o3d.LinearCurveKey](#), and [o3d.StepCurveKey](#).

[List of all members.](#)

---

## Detailed Description

A [CurveKey](#) represents a key on an [Curve](#).

## Public Member Functions

- [destroy\(\)](#)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [input](#)
  - Number [output](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.CurveKey.destroy ( )`

Destroys this key, removing it from its owner.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.CurveKey.input](#)

The input of this key.

Number [o3d.CurveKey.output](#)

The output of this key.

## o3d.StepCurveKey Class Reference

Inherits [o3d.CurveKey](#).

[List of all members.](#)

---

### Detailed Description

An [CurveKey](#) that holds its output (is not interpolated between this key and the next.)

### Public Member Functions

- [destroy\(\)](#)
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- Number [input](#)
  - Number [output](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

`o3d.CurveKey.destroy ( )` [inherited]

Destroys this key, removing it from its owner.

`bool o3d.ObjectBase.isAClassName ( String class_name )` [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

---

# Member Data Documentation

`String o3d.ObjectBase.className` [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId` [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Number o3d.CurveKey.input` [inherited]

The input of this key.

Number [o3d.CurveKey.output](#) [inherited]

The output of this key.

# o3d.LinearCurveKey Class Reference

Inherits [o3d.CurveKey](#).

[List of all members.](#)

---

## Detailed Description

An [CurveKey](#) that linearly interpolates between this key and the next key.

## Public Member Functions

- [destroy \(\)](#)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [input](#)
  - Number [output](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.CurveKey.destroy ( ) [inherited]`

Destroys this key, removing it from its owner.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.CurveKey.input](#) [inherited]

The input of this key.

Number [o3d.CurveKey.output](#) [inherited]

The output of this key.

# o3d.BezierCurveKey Class Reference

Inherits [o3d.CurveKey](#).

[List of all members.](#)

---

## Detailed Description

An [CurveKey](#) that uses a bezier curve for interpolation between this key and the next.

## Public Member Functions

- [destroy \(\)](#)
- bool [isAClassName](#) (String *class\_name*)

## Public Attributes

- [Float2 inTangent](#)
  - [Float2 outTangent](#)
  - Number [input](#)
  - Number [output](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.CurveKey.destroy ( ) [inherited]`

Destroys this key, removing it from its owner.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
```

```
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.CurveKey.input](#) [inherited]

The input of this key.

[Float2 o3d.BezierCurveKey.inTangent](#)

The in tangent for this key.

Number [o3d.CurveKey.output](#) [inherited]

The output of this key.

[Float2 o3d.BezierCurveKey.outTangent](#)

The out tangent for this key.

# o3d.Curve Class Reference

Inherits [o3d.Function](#).

[List of all members.](#)

---

## Detailed Description

A [Curve](#) stores a bunch of CurveKeys and given a value representing an input point on a curve returns the output of the curve for that input. [Curve](#) is data only. It is used by 1 or more [FunctionEval](#) objects or by direct use from javascript.

## Public Types

- enum [Infinity](#)

## Public Member Functions

- bool [isDiscontinuous](#) ()
- [addStepKeys](#) (Array [keys](#))
- [addLinearKeys](#) (Array [keys](#))
- [addBezierKeys](#) (Array [keys](#))
- [CurveKey createKey](#) (String key\_type)
- bool [set](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- bool [set](#) ([RawData](#) raw\_data)
- Number [evaluate](#) (Number input)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Infinity preInfinity](#)
- [Infinity postInfinity](#)
- bool [useCache](#)

- Number [sampleRate](#)
  - Array [keys](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Curve.Infinity](#)

enum Infinity {

- CONSTANT, Uses the output value of the first or last animation key.
  - LINEAR, Takes the distance between the closest animation key input value and the evaluation time. Multiplies this distance against the instant slope at the closest animation key and offsets the result with the closest animation key output value.
  - CYCLE, Cycles over the first and last keys using  $\text{input} = (\text{input} - \text{first}) \% (\text{last} - \text{first}) + \text{first}$ ; Note that in CYCLE mode you can never get the end output because a cycle goes from start to end exclusive of end.
  - CYCLE\_RELATIVE, Same as cycle except the offset of the entire cycle is added to each consecutive cycle.
  - OSCILLATE, Ping Pongs between the first and last keys. };
- 

## Member Function Documentation

`o3d.Curve.addBezierKeys ( Array keys )`

Adds 1 or more BezierKeys to this [Curve](#).

Example:

```
// Creates 2 keys.  
// 1 key at 0 with an output of 10, in tangent of 1,9, out tangent 9,0.5  
// 1 key at 20 with an output of 30, in tangent of 30, 3, out tangent 4, 28  
curve.addBezierKeys([0,10,1,9,9,0.5,2,30,3,4,28]);
```

**Parameters:**

*keys* Array of input, output pairs.

`o3d.Curve.addLinearKeys ( Array keys )`

Adds 1 or more LinearKeys to this [Curve](#).

Example:

```
// Creates 2 keys.  
// 1 key at 0 with an output of 10  
// 1 key at 20 with an output of 30  
curve.addLinearKeys([0,10,20,30]);
```

**Parameters:**

*keys* Array of input, output pairs.

`o3d.Curve.addStepKeys ( Array keys )`

Adds 1 or more StepKeys to this [Curve](#).

Example:

```
// Creates 2 keys.  
// 1 key at 0 with an output of 10  
// 1 key at 20 with an output of 30  
curve.addStepKeys([0,10,20,30]);
```

**Parameters:**

*keys* Array of input, output pairs.

[CurveKey](#) `o3d.Curve.createKey ( String key_type )`

Creates a new key for this curve.

**Parameters:**

*key\_type* name of key class to create. Valid type names are:

- 'o3d.StepCurveKey',
- 'o3d.LinearCurveKey',
- 'o3d.BezierCurveKey',

**Returns:**

The created key.

Number o3d.Function.evaluate ( Number *input* ) [inherited]  
Gets an output for this function for the given input.

**Parameters:**

*input* Input to get output at.

**Returns:**

The output for the given input.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.Curve.isDiscontinuous ( )  
Returns whether or not the curve is discontinuous. A discontinuous curve takes more time to evaluate.

**Returns:**

True if the curve is discontinuous.

bool o3d.Curve.set ( [RawData](#) *raw\_data* )  
Deserializes from the curve data given a [RawData](#) object.

**Parameters:**

*raw\_data* entire contents contains curve data

**Returns:**

True if operation was successful.

```
bool o3d.Curve.set ( RawData raw_data,
                     size_t      offset,
                     size_t      length
                   )
```

Deserializes from the curve data given a [RawData](#) object.

**Parameters:**

*raw\_data* contains curve data  
*offset* is a byte offset from the start of *raw\_data*  
*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Curve.keys](#)

The keys for this curve.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### [Infinity o3d.Curve.postInfinity](#)

The behavior of the curve after the last key.

#### [Infinity o3d.Curve.preInfinity](#)

The behavior of the curve before the first key.

#### Number [o3d.Curve.sampleRate](#)

Gets the sample rate for the cache. By default Animation data is cached so that using the animation is fast. To do this the keys that represent the animation are sampled. The higher the frequency of the samples the closer the cache will match the actual keys. The default is 1/30 (30hz). You can set it anywhere from 1/240th (240hz) to any larger value. Note: Setting the sample rate has the side effect of invalidating the cache thereby causing it to get rebuilt. Must be 1/240 or greater. Default = 1/30

#### bool [o3d.Curve.useCache](#)

Whether or not a cache is used to speed up evaluation of this [Curve](#)

**See also:**

[SetSampleRate](#)

## **o3d.DisplayMode Class Reference**

[List of all members.](#)

---

### **Detailed Description**

A [DisplayMode](#) describes the size and refresh rate of a display mode; it's usually used in [or when transitioning into] fullscreen mode.

### **Public Attributes**

- Number [width](#)

- Number [height](#)
  - Number [refreshRate](#)
  - Number [id](#)
- 

## Member Data Documentation

Number [o3d.DisplayMode.height](#)

The height in pixels of the screen when in this mode.

This property is read-only.

Number [o3d.DisplayMode.id](#)

An opaque identifier used to identify the mode when requesting a fullscreen transition.

This property is read-only.

Number [o3d.DisplayMode.refreshRate](#)

The refresh rate in Hz.

This property is read-only.

Number [o3d.DisplayMode.width](#)

The width in pixels of the screen when in this mode.

This property is read-only.

## o3d.DrawContext Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

The [DrawContext](#) defines the parameters used for a particular drawing pass. It contains two 4-by-4 matrix params, view and projection. These correspond to the viewing and projection transformation

matrices.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Matrix4 [view](#)
  - Matrix4 [projection](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`) [inherited]`

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',

- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

Matrix4 [o3d.DrawContext.projection](#)

The projection matrix represents the projection transformation, used to take vertices from view space to screen space. This matrix is usually an orthographic or perspective transformation.

Matrix4 [o3d.DrawContext.view](#)

The view matrix represents the viewing transformation, used to take vertices from world space to view space.

## o3d.DrawElement Class Reference

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

### Detailed Description

A [DrawElement](#) causes an [Element](#) to be Drawn with a particular material. You can override other [Effect](#) parameters by adding corresponding params to the [DrawElement](#).

**See also:**

[Element](#)

[Material](#)

[Effect](#)

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Material material](#)
  - [Element owner](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`) [inherited]`

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',

- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

## Returns:

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

## Parameters:

*param\_name* Name to search for.

## Returns:

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
```

```
t.isAClassName('o3d.Shape');           // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Material](#) [o3d.DrawElement.material](#)

The [Material](#) for this Draw [Element](#). If it is null the material of owner will be used.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### [Element o3d.DrawElement.owner](#)

The current owner of this Draw [Element](#). Set to null to stop being owned.

Note: DrawElements are referenced by the [Pack](#) they are created in and their owner. If the [DrawElement](#) is removed from its [Pack](#) then setting the owner to null will free the [DrawElement](#). Or, visa versa, if you set the DrawElement's owner to null then removing it from its [Pack](#) will free the [DrawElement](#).

#### Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

# **o3d.DrawList Class Reference**

Inherits [o3d.NamedObject](#).

[List of all members](#).

---

# Detailed Description

A [DrawList](#) gets used during rendering to collect DrawElements to render. Each [Material](#) references a [DrawList](#). Depending on the material, as DrawElements get collected they will be put on different DrawLists.

## Public Types

- enum [SortMethod](#)

## Public Member Functions

- bool [isAClassName](#) (String class\_name)

## Public Attributes

- String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.DrawList.SortMethod](#)

- BY\_PERFORMANCE
- BY\_Z\_ORDER
- BY\_PRIORITY

Method to sort [DrawList](#) by.

---

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

# o3d.DrawPass Class Reference

Inherits [o3d.RenderNode](#).

[List of all members.](#)

---

## Detailed Description

A [DrawPass](#) renders a [DrawList](#).

## Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)
- Array [getRenderNodesByClassNameInTree \(String class\\_name\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- bool [removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [DrawList drawList](#)
- [DrawList SortMethod sortMethod](#)
- Number [priority](#)
- bool [active](#)
- [RenderNode parent](#)
- Array [children](#)
- Array [params](#)
- String [name](#)
- Id [clientId](#)
- String [className](#)

---

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',

- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array  
o3d.RenderNode.getRenderNodesByClassNameInTree ( String  $\underset{e}{\overset{\text{class\_nam}}{}}$  ) [inherited]  
Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* ) [inherited]  
Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]  
Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[DrawList o3d.DrawPass.drawList](#)

The [DrawList](#) for this [DrawPass](#).

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under parent\_node. Setting parent\_node to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

Number [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

[DrawList SortMethod o3d.DrawPass.sortMethod](#)

The sort method for this [DrawPass](#) to draw the [DrawList](#) by.

## o3d.EffectParameterInfo Class Reference

[List of all members.](#)

---

### Detailed Description

[EffectParameterInfo](#) holds information about the Parameters an [Effect](#) needs.

See also:

[o3d.Effect.getParameterInfo](#)

### Public Attributes

- String [name](#)
- String [className](#)

- Number [numElements](#)
  - String [semantic](#)
  - String [sasClassName](#)
- 

## Member Data Documentation

String [o3d.EffectParameterInfo.className](#)

The type of the parameter.

This property is read-only.

String [o3d.EffectParameterInfo.name](#)

The name of the parameter.

This property is read-only.

Number [o3d.EffectParameterInfo.numElements](#)

The number of elements. Non-zero for array types, zero for non-array types.

This property is read-only.

String [o3d.EffectParameterInfo.sasClassName](#)

If this is a standard parameter (SAS) this will be the name of the type of [Param](#) needed. Otherwise it will be the empty string.

Standard Parameters are generally handled automatically by [o3d](#) but you can supply your own if you have a unique situation.

This property is read-only.

String [o3d.EffectParameterInfo.semantic](#)

The semantic of the parameter. This is always in UPPERCASE.

This property is read-only.

## **o3d.EffectStreamInfo Class Reference**

[List of all members.](#)

---

## Detailed Description

[EffectStreamInfo](#) holds information about the Streams an [Effect](#) needs.

See also:

[o3d.Effect.getStreamInfo](#)

---

## Public Attributes

- [Stream](#) Semantic [semantic](#)
  - Number [semanticIndex](#)
- 

## Member Data Documentation

[Stream](#) Semantic [o3d.EffectStreamInfo.semantic](#)

The semantic of the stream.

This property is read-only.

Number [o3d.EffectStreamInfo.semanticIndex](#)

The semantic index of the stream.

This property is read-only.

## o3d.Effect Class Reference

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

## Detailed Description

An [Effect](#) contains a vertex and pixel shader.

## Public Types

- enum [MatrixLoadOrder](#)

## Public Member Functions

- bool [loadFromFXString](#) (String effect)
- [createUniformParameters](#) ([ParamObject](#) param\_object)
- [createSASParameters](#) ([ParamObject](#) param\_object)
- Array [getParameterInfo](#) ()
- Array [getStreamInfo](#) ()
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [MatrixLoadOrder matrixLoadOrder](#)
  - String [source](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

### enum [o3d.Effect.MatrixLoadOrder](#)

- ROW\_MAJOR, Matrix parameters are loaded in row-major order (DX-style).
  - COLUMN\_MAJOR, Matrix parameters are loaded in column-major order (OpenGL-style).
-

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',

- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

`o3d.Effect.createSASParameters ( ParamObject param_object )`

For each of the effect's uniform parameters, if it is a SAS parameter creates corresponding

StandardParamMatrix4 parameters on the given [ParamObject](#). Note that SAS parameters are handled automatically by the rendering system. so except in some rare cases there is no reason to call this function. Also be aware that the StandardParamMatrix4 Paramters like

[WorldViewProjectionParamMatrix4](#), etc.. are only valid during rendering. At all other times they will not return valid values.

If a [Param](#) with the same name but the wrong type already exists on the given [ParamObject](#) CreateSASParameters will attempt to replace it with one of the correct type.

#### Parameters:

*param\_object* The param object on which the new paramters will be created.

#### See also:

[o3d.Effect.createUniformParameters](#)

`o3d.Effect.createUniformParameters ( ParamObject param_object )`

For each of the effect's uniform parameters, creates corresponding parameters on the given [ParamObject](#). Skips SAS Parameters.

If a [Param](#) with the same name but the wrong type already exists on the given [ParamObject](#) CreateUniformParameters will attempt to replace it with one of the correct type.

Note: The most common thing to pass to this function is a [Material](#) but depending on your application it may be more appropriate to pass in a [Transform](#), [Effect](#), [Element](#) or [DrawElement](#).

#### Parameters:

*param\_object* The param object on which the new paramters will be created.

#### See also:

[o3d.Effect.createSASParameters](#)

[Param](#) `o3d.ParamObject.getParam ( String param_name ) [inherited]`

Searches by name for a [Param](#) defined in the object.

#### Parameters:

*param\_name* Name to search for.

#### Returns:

The [Param](#) with the given name, or null otherwise.

Array `o3d.Effect.getParameterInfo ( )`

Gets info about the parameters this effect needs.

**Returns:**

an array of EffectParameterInfos.

Array o3d.Effect.getStreamInfo ( )

Gets info about the streams this effect needs.

**Returns:**

an array of EffectParameterInfos.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.Effect.loadFromFXString ( String *effect* )

Loads the vertex and pixel shader programs from an string containing an O3D FX description.

The string is subset of CG and HLSL. No techinques are allowed.

To define the entry points add 2 lines in the following format.

```
// #o3d VertexShaderEntryPoint myVertexShader\n
// #o3d PixelShaderEntryPoint myPixelShader\n
```

where "myVertexShader" and "myPixelShader" are the names of your vertex and pixel shaders. At this time the format of those 2 lines is extremely strict. You must have 1 and exactly 1 space between // and o3d, between o3d and VertexShaderEntryPoint/PixelShaderEntryPoint and between those and your entry points.

You must also specify a matrix load order like this.

```
// #o3d MatrixLoadOrder RowMajor
```

Valid orders are RowMajor and ColumnMajor

Note: Currently it is possible to create effects strings that work on only one platform (GL or D3D). You should test your shaders on both platforms.

By version 1.0 this function will enforce shaders that only work on both platforms. That format is mostly CG.

**Parameters:**

*effect* the code of the effect.

**Returns:**

True if successful.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

#### [MatrixLoadOrder](#) `o3d.Effect.matrixLoadOrder`

The order in which matrix data is loaded to the GPU.

This property is read-only.

#### `String` [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### `Array` [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### `String` [o3d.Effect.source](#)

The source for the shaders on this [Effect](#).

This property is read-only.

## **o3d.Element Class Reference**

Inherits [o3d.ParamObject](#).

Inherited by [o3d.Primitive](#).

[List of all members](#).

---

## Detailed Description

An [Element](#) manages DrawElements for classes inherited from [Element](#).

## Public Member Functions

- [DrawElement createDrawElement \(Pack pack, Material material\)](#)
- [RayIntersectionInfo intersectRay \(Number position\\_stream\\_index, State.Cull cull, Point3 start, Point3 end\)](#)
- [BoundingBox getBoundingBox \(Number position\\_stream\\_index\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- [bool removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- [bool isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Material material](#)
  - [BoundingBox boundingBox](#)
  - [Float3 zSortPoint](#)
  - [Number priority](#)
  - [bool cull](#)
  - [Shape owner](#)
  - [Array drawElements](#)
  - [Array params](#)
  - [String name](#)
  - [Id clientId](#)
  - [String className](#)
-

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[DrawElement](#) `o3d.Element.createDrawElement ( Pack pack,  
                         Material material  
                         )`

Creates a [DrawElement](#) for this [Element](#). Note that unlike [Shape.createDrawElements](#) and [Transform.createDrawElements](#) this one will create more than one element for the same material.

## Parameters:

*pack* pack used to manage [DrawElement](#).

*material* material to use for [DrawElement](#). If you pass null it will use the material on this [Element](#). This allows you to easily setup the default (just draw as is) by passing null or setup a shadow pass by passing in a shadow material.

## Returns:

The created draw element.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
                         String param_type_name  
                         )` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',

- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',

- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[BoundingBox](#) o3d.Element.getBoundingBox ( Number *position\_stream\_index* )

Computes the bounding box in same coordinate system as the specified POSITION stream.

**Parameters:**

*position\_stream\_index* Index of POSITION stream.

**Returns:**

The boundingbox for this element in local space.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[RayIntersectionInfo](#) o3d.Element.intersectRay ( Number *position\_stream\_index*,  
*State.Cull* *cull*,  
*Point3* *start*,  
*Point3* *end*  
)

Computes the intersection of a ray in the same coordinate system as the specified POSITION stream.

**Parameters:**

<i>position_stream_index</i>	Index of POSITION stream.
<i>cull</i>	which side of the triangles to ignore.
<i>start</i>	position of start of ray in local space.
<i>end</i>	position of end of ray. in local space.

**Returns:**

[RayIntersectionInfo](#) class. If valid() is false then something was wrong, Check client.GetLastError(). If intersected() is true then the ray intersected a something. position() is the exact point of intersection.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

### [BoundingBox](#) [o3d.Element.boundingBox](#)

The [BoundingBox](#) for this element. If culling is on this bounding box will be tested against the view frustum of any draw context used to render this [Element](#).

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#) [inherited]**

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**bool [o3d.Element.cull](#)**

The cull settings for this element. If true this [Element](#) will be culled by the bounding box above compared to the view frustum it is being rendered with.

**Array [o3d.Element.drawElements](#)**

Gets all the DrawElements under this [Element](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var drawElements = element.drawElements;  
for (var i = 0; i < drawElements.length; i++) {  
    var drawElement = drawElements[i];  
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [Element](#), while modifications to the members of the array **will** affect them.

This property is read-only.

**[Material o3d.Element.material](#)**

The [Material](#) for this element.

**String [o3d.NamedObject.name](#) [inherited]**

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), `Pack.getObject`, [RenderNode.getRenderNodesByNameInTree](#) and `RenderNode.getTransformsByNameInTree` which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

### [Shape](#) [o3d.Element.owner](#)

The current owner of this [Draw Element](#). Pass in null to stop being owned.

Note: Elements are referenced by the [Pack](#) they are created in and their owner. If the [Element](#) is removed from its [Pack](#) then setting the owner to null will free the [Element](#). Or, visa versa, if you set the Element's owner to null then removing it from its [Pack](#) will free the [Element](#).

### [Array](#) [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

### [Number](#) [o3d.Element.priority](#)

The priority for this element. Used to sort if this [Element](#) is drawn by a [DrawPass](#) that is set to sort by priority.

### [Float3](#) [o3d.Element.zSortPoint](#)

The z sort point for this element. If this [Element](#) is drawn by a [DrawPass](#) that is set to sort by z order this value will be multiplied by the worldViewProjection matrix to compute a z value to sort by.

## **o3d.Event Class Reference**

[List of all members](#).

---

### **Detailed Description**

An [Event](#) object contains information describing a JavaScript event; it's used as an argument to event handlers triggered by the plugin.

## Public Types

- enum [Button](#)

## Public Attributes

- String [type](#)
  - Number [button](#)
  - bool [ctrlKey](#)
  - bool [altKey](#)
  - bool [shiftKey](#)
  - bool [metaKey](#)
  - Number [keyCode](#)
  - Number [charCode](#)
  - Number [x](#)
  - Number [y](#)
  - Number [screenX](#)
  - Number [screenY](#)
  - Number [deltaX](#)
  - Number [deltaY](#)
  - Number [width](#)
  - Number [height](#)
  - bool [fullscreen](#)
- 

## Member Enumeration Documentation

enum [o3d.Event.Button](#)

- BUTTON\_LEFT
- BUTTON\_RIGHT
- BUTTON\_MIDDLE
- BUTTON\_4
- BUTTON\_5 Constants used to identify mouse buttons.

---

## Member Data Documentation

bool [o3d.Event.altKey](#)

Whether the alt [option, on OSX] key was pressed at the time of the event.

This property is read-only.

Number [o3d.Event.button](#)

Which mouse button caused the event, in the case of mousedown, mouseup, click, and dblclick events.

This uses the values in enum Button.

This property is read-only.

Number [o3d.Event.charCodeAt](#)

The character created by a keypress event.

This property is read-only.

bool [o3d.Event.ctrlKey](#)

Whether the ctrl key was pressed at the time of the event.

This property is read-only.

Number [o3d.Event.deltaX](#)

The horizontal scroll offset for wheel events, in arbitrary units. Positive values mean right; negative mean left.

This property is read-only.

Number [o3d.Event.deltaY](#)

The vertical scroll offset for wheel events, in arbitrary units. Positive values mean up or away from the user; negative mean down or toward the user.

This property is read-only.

bool [o3d.Event.fullscreen](#)

Whether we're currently displaying in fullscreen mode.

This property is read-only.

Number [o3d.Event.height](#)

The height in pixels of the plugin or fullscreen display region as a result of this event.

This property is read-only.

#### Number [o3d.Event.keyCode](#)

The key code of the key pressed or released.

This property is read-only.

#### bool [o3d.Event.metaKey](#)

Whether the meta [command, on OSX] key was pressed at the time of the event.

This property is read-only.

#### Number [o3d.Event.screenX](#)

The x-coordinate in pixels from the left side of the screen.

This property is read-only.

#### Number [o3d.Event.screenY](#)

The y-coordinate in pixels from the top of the screen.

This property is read-only.

#### bool [o3d.Event.shiftKey](#)

Whether the shift key was pressed at the time of the event.

This property is read-only.

#### [o3d.Event.type](#)

String identifiers for JavaScript events.

- invalid
- click
- dblclick
- mousedown
- mousemove
- mouseup
- wheel
- keydown
- keypress
- keyup

- `resize`

The type of event this object represents.

- `invalid`
- `click`
- `dblclick`
- `keydown`
- `keypress`
- `keyup`
- `mousedown`
- `mousemove`
- `mouseup`
- `wheel`
- `resize`

This property is read-only.

#### Number [o3d.Event.width](#)

The width in pixels of the plugin or fullscreen display region as a result of this event.

This property is read-only.

#### Number [o3d.Event.x](#)

The x-coordinate in pixels from the left side of the plugin or fullscreen display region.

This property is read-only.

#### Number [o3d.Event.y](#)

The y-coordinate in pixels from the top of the plugin or fullscreen display region.

This property is read-only.

# **o3d.Field Class Reference**

Inherits [o3d.NamedObject](#).

Inherited by [o3d.FloatField](#), [o3d.UByteNField](#), and [o3d.UInt32Field](#).

[List of all members.](#)

---

## Detailed Description

A [Field](#) is a base class that manages a set of components in a [Buffer](#) of a specific type. Fields are managed by Buffers and can not be directly created. When a [Buffer](#) is destroyed or if a [Field](#) is removed from a [Buffer](#) the Field's buffer property will be set to null.

## Public Member Functions

- bool [isAClassName](#) (String *class\_name*)

## Public Attributes

- Number [numComponents](#)
  - [Buffer buffer](#)
  - Number [offset](#)
  - Number [size](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

---

# Member Data Documentation

## [Buffer o3d.Field.buffer](#)

The [Buffer](#) the field belongs to.

This property is read-only.

## [String o3d.ObjectBase.className \[inherited\]](#)

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

## [Id o3d.ObjectBase.clientId \[inherited\]](#)

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

## [String o3d.NamedObject.name \[inherited\]](#)

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

## [Number o3d.Field.numComponents](#)

The number of components in this field.

This property is read-only.

## Number [o3d.Field.offset](#)

The offset of this field in the [Buffer](#) in bytes.

This property is read-only.

## Number [o3d.Field.size](#)

The size of one element of this field.

This property is read-only.

# **o3d.FloatField Class Reference**

Inherits [o3d.Field](#).

[List of all members.](#)

---

## Detailed Description

A field that contains floating point numbers.

## Public Member Functions

- [setAt](#) ( Number start\_index, Array values)
- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numComponents](#)
- [Buffer buffer](#)
- Number [offset](#)
- Number [size](#)
- String [name](#)
- Id [clientId](#)
- String [className](#)

---

## Member Function Documentation

```
Array o3d.FloatField.getValueAt ( Number start_index,
                                 Number num_elements
                               )
```

Gets the values stored in the field.

**Parameters:**

*start\_index* index of the first value to get.  
*num\_elements* number of elements to read from field.

**Returns:**

The values of the field.

```
bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]
```

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

```
o3d.FloatField.setAt ( Number start_index,
                      Array values
                    )
```

Sets the values of the data stored in the field.

The buffer for the field must have already been created either through buffer.set or through buffer.allocateElements.

The number of values passed in must be a multiple of the number of components needed for the field.

**Parameters:**

---

*start\_index* index of first value to set.  
*values* Values to be stored in the buffer starting at index.

---

## Member Data Documentation

[Buffer](#) [o3d.Field.buffer](#) [inherited]

The [Buffer](#) the field belongs to.

This property is read-only.

[String](#) [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

[Id](#) [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[String](#) [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

[Number](#) [o3d.Field.numComponents](#) [inherited]

The number of components in this field.

This property is read-only.

[Number](#) [o3d.Field.offset](#) [inherited]

The offset of this field in the [Buffer](#) in bytes.

This property is read-only.

Number [o3d.Field.size](#) [inherited]

The size of one element of this field.

This property is read-only.

## o3d.UInt32Field Class Reference

Inherits [o3d.Field](#).

[List of all members.](#)

---

### Detailed Description

A field that contains integers.

### Public Member Functions

- [setAt](#) ( Number start\_index, Array values)
- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- Number [numComponents](#)
  - [Buffer buffer](#)
  - Number [offset](#)
  - Number [size](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

```
Array o3d.UInt32Field.getValueAt ( Number start_index,  
                                  Number num_elements  
                                )
```

Gets the values stored in the field.

## Parameters:

*start\_index* index of the first value to get.  
*num\_elements* number of elements to read from field.

## Returns:

The values of the field.

```
bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]
```

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

```
o3d.UInt32Field.setAt ( Number start_index,  
                      Array values  
                    )
```

Sets the values of the data stored in the field.

The buffer for the field must have already been created either through buffer.set or through buffer.allocateElements.

The number of values passed in must be a multiple of the number of components needed for the field.

## Parameters:

*start\_index* index of first value to set.  
*values* Values to be stored in the buffer starting at index.

---

# Member Data Documentation

Buffer [o3d.Field.buffer](#) [inherited]

The [Buffer](#) the field belongs to.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Field.numComponents](#) [inherited]

The number of components in this field.

This property is read-only.

Number [o3d.Field.offset](#) [inherited]

The offset of this field in the [Buffer](#) in bytes.

This property is read-only.

Number [o3d.Field.size](#) [inherited]

The size of one element of this field.

This property is read-only.

# o3d.UByteNField Class Reference

Inherits [o3d.Field](#).

[List of all members.](#)

---

## Detailed Description

A field that contains bytes that each represent a 0 to 1 value. A typical use is for vertex colors since 4 bytes per vertex color are much smaller than 4 floats per vertex color.

## Public Member Functions

- [setAt](#) ( Number start\_index, Array values)
- Array [getAt](#) ( Number start\_index, Number num\_elements)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [numComponents](#)
  - [Buffer buffer](#)
  - Number [offset](#)
  - Number [size](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

```
Array o3d.UByteNField.getValueAt ( Number start_index,  
                                  Number num_elements  
                                )
```

Gets the values stored in the field.

## Parameters:

*start\_index* index of the first value to get.  
*num\_elements* number of elements to read from field.

## Returns:

The values of the field.

```
bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]
```

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

```
o3d.UByteNField.setAt ( Number start_index,  
                        Array values  
                      )
```

Sets the values of the data stored in the field.

The buffer for the field must have already been created either through buffer.set or through buffer.allocateElements.

The number of values passed in must be a multiple of the number of components needed for the field.

## Parameters:

*start\_index* index of first value to set.  
*values* Values to be stored in the buffer starting at index.

---

# Member Data Documentation

Buffer [o3d.Field.buffer](#) [inherited]

The [Buffer](#) the field belongs to.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Number [o3d.Field.numComponents](#) [inherited]

The number of components in this field.

This property is read-only.

Number [o3d.Field.offset](#) [inherited]

The offset of this field in the [Buffer](#) in bytes.

This property is read-only.

Number [o3d.Field.size](#) [inherited]

The size of one element of this field.

This property is read-only.

# o3d.FileRequest Class Reference

Inherits [o3d.ObjectBase](#).

[List of all members.](#)

---

## Detailed Description

A [FileRequest](#) object is used to carry out an asynchronous request for a file to be loaded. Its use parallels that of XMLHttpRequest; you create one, call open, set the onreadystatechange callback, and call send. Note that unlike XMLHttpRequests, FileRequests cannot be reused.

For texture loads, on success the texture will be stored in a field of the same name on the [FileRequest](#) itself.

```
var request = pack.createFileRequest("TEXTURE");
var texture;
request.open("GET", url, true);
request.onreadystatechange = function() {
    if (request.done) {
        if (request.success) {
            texture = request.texture;
        } else {
            dump('Load of texture file returned failure.');
        }
    }
};
request.send();
```

## Public Member Functions

- [open](#) (String method, String uri, bool async)
- [send](#) ()

- bool [isAClassName](#) (String *class\_name*)

## Public Attributes

- [Texture texture](#)
  - bool [generateMipmaps](#)
  - Number [readyState](#)
  - bool [done](#)
  - bool [success](#)
  - String [error](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

```
o3d.FileRequest.open ( String method,
                      String uri,
                      bool   async
                    )
```

Set up several of the request fields.

### Parameters:

*method* "GET" is the only supported method at this time  
*uri* the location of the file to fetch  
*async* true is the only legal value at this time

`o3d.FileRequest.send( )`

Send the request. Unlike XMLHttpRequest the onreadystatechange callback will be called no matter what, with success or failure.

---

## Member Data Documentation

**String [o3d.ObjectBase.className](#)** [inherited]  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#)** [inherited]  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**bool [o3d.FileRequest.done](#)**  
This indicates whether any further processing will be done on this [FileRequest](#).

This property is read-only.

**String [o3d.FileRequest.error](#)**  
An error message. If done is true and success is false this will be an error message describing what went wrong.

This property is read-only.

**bool [o3d.FileRequest.generateMipmaps](#)**  
Whether or not to generate mip-maps on textures that are loaded (default: true). Mip-maps are not generated for DXTC textures. DDS files can contain pre-computed mip-maps for DXTC textures though.

Number [o3d.FileRequest.readyState](#)

This holds the same values as in XMLHttpRequest:

- 0 = uninitialized
- 1 = opened
- 2 = sent
- 3 = receiving
- 4 = loaded (the file has been downloaded, but may or may not have been parsed yet)

This property is read-only.

bool [o3d.FileRequest.success](#)

This field is only valid if done is true. It indicates whether or not the request succeeded. If it failed error holds an error message.

This property is read-only.

[Texture o3d.FileRequest.texture](#)

On completion of successful texture file loads, this holds the loaded texture.

This property is read-only.

## **o3d.Function Class Reference**

Inherits [o3d.NamedObject](#).

Inherited by [o3d.Curve](#).

[List of all members.](#)

---

## **Detailed Description**

A [Function](#) is a class that has an Evaluate method. Evaluate takes 1 input and returns 1 output.

## **Public Member Functions**

- Number [evaluate](#) (Number input)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

Number o3d.Function.evaluate ( Number *input* )

Gets an output for this function for the given input.

### Parameters:

*input* Input to get output at.

### Returns:

The output for the given input.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

## o3d.ParamFunction Class Reference

Inherits [o3d.Param](#).

[List of all members](#).

---

### Detailed Description

A [Param](#) which stores a [Function](#).

### Public Member Functions

- `bool bind (Param source_param)`
- `unbindInput ()`
- `unbindOutput (Param destination_param)`

- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Function value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Function o3d.ParamFunction.value](#)

The [Function](#) stored by the [Param](#).

## o3d.FunctionEval Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

# Detailed Description

A [FunctionEval](#) evaluates a [Function](#) through parameters.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [input](#)
  - Number [output](#)
  - [Function functionObject](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same

name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',

- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
```

```
t.isAClassName('o3d.ParamObject'); // true  
t.isAClassName('o3d.Shape'); // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Function o3d.FunctionEval.functionObject](#)

The function to evaluate.

## Number [o3d.FunctionEval.input](#)

The input to the function

## String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

## Number [o3d.FunctionEval.output](#)

The output of the function for the given input.

This property is read-only.

## Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

# o3d.Material Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

# Detailed Description

A [Material](#) holds the various uniform parameters an [Effect](#) needs to render. For example a Lambert effect might need "diffuse", "ambient", "emissive". The parameters needed on a [Material](#) will vary depending its [Effect](#). Note that a material MUST have its drawList set in order for objects using it to render.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Effect effect](#)
  - [State state](#)
  - [DrawList drawList](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
                                 String *param\_type\_name*  
                                 )

[inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name*      The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',

- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

### [DrawList o3d.Material.drawList](#)

The [DrawList](#) this material will render on.

### [Effect o3d.Material.effect](#)

The [Effect](#) for this material.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

### [State o3d.Material.state](#)

The [State](#) for this material.

# **o3d.Matrix4AxisRotation Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

# Detailed Description

This param operation applies a rotation to its input matrix. The rotation is specified in terms of an angle and an axis of rotation. It can be used, for example, to supply the local matrix of a transform.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float3 axis](#)
  - Number [angle](#)
  - Matrix4 [inputMatrix](#)
  - Matrix4 [outputMatrix](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`

) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',

- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

Number [o3d.Matrix4AxisRotation.angle](#)

The angle of rotation in radians.

[Float3 o3d.Matrix4AxisRotation.axis](#)

The local rotation axis. This must be a unit vector.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

#### Matrix4 [o3d.Matrix4AxisRotation.inputMatrix](#)

The input matrix. Identity by default.

#### String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### Matrix4 [o3d.Matrix4AxisRotation.outputMatrix](#)

The output is equal to the rotation of the input matrix.

This property is read-only.

#### Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

# **o3d.Matrix4Composition Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

## Detailed Description

This param operation composes (multiplies) a local matrix with an input matrix. It can be used, for example, to supply the local matrix of a transform.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Matrix4 [localMatrix](#)
  - Matrix4 [inputMatrix](#)
  - Matrix4 [outputMatrix](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`  
Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`

```
String param_type_name  
)
```

[inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',

- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

#### Matrix4 [o3d.Matrix4Composition.inputMatrix](#)

The input matrix. Identity by default.

#### Matrix4 [o3d.Matrix4Composition.localMatrix](#)

The local matrix.

#### String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### Matrix4 [o3d.Matrix4Composition.outputMatrix](#)

The output is equal to the composition (multiplication) of the input matrix with the local matrix.

This property is read-only.

#### Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## **o3d.Matrix4Scale Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

# Detailed Description

This param operation applies a scaling to its input matrix. It can be used, for example, to supply the local matrix of a transform.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float3 scale](#)
  - Matrix4 [inputMatrix](#)
  - Matrix4 [outputMatrix](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

**Parameters:**

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',

- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
```

```
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Matrix4 [o3d.Matrix4Scale.inputMatrix](#)

The parent matrix. Identity by default.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Matrix4 [o3d.Matrix4Scale.outputMatrix](#)

The output is equal to the scale of the input matrix.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[Float3 o3d.Matrix4Scale.scale](#)

The local scaling.

## o3d.Matrix4Translation Class Reference

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

# Detailed Description

This param operation applies a translation to its input matrix. It can be used, for example, to supply the local matrix of a transform.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float3 translation](#)
  - Matrix4 [inputMatrix](#)
  - Matrix4 [outputMatrix](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

**Parameters:**

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',

- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
```

```
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Matrix4 [o3d.Matrix4Translation.inputMatrix](#)

The input matrix. Identity by default.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Matrix4 [o3d.Matrix4Translation.outputMatrix](#)

The output is equal to the translation of the input matrix.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

Float3 [o3d.Matrix4Translation.translation](#)

The local translation.

## o3d.ObjectBase Class Reference

Inherited by [o3d.ArchiveRequest](#), [o3d.CurveKey](#), [o3d.FileRequest](#), [o3d.NamedObjectBase](#), and

[o3d.Stream](#).

[List of all members](#).

---

# Detailed Description

The base class of most O3D run-time objects.

## Public Member Functions

- bool [isAClassName](#) (String *class\_name*)

## Public Attributes

- Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* )

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#)

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#)

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

## **o3d.NamedObjectBase Class Reference**

Inherits [o3d.ObjectBase](#).

Inherited by [o3d.NamedObject](#), and [o3d.Param](#).

[List of all members.](#)

---

### **Detailed Description**

Base class for all objects that are identifiable by a name.

### **Public Member Functions**

- bool [isAClassName](#) (String class\_name)

### **Public Attributes**

- String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObjectBase.name](#)

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

# o3d.NamedObject Class Reference

Inherits [o3d.NamedObjectBase](#).

Inherited by [o3d.Buffer](#), [o3d.DrawLine](#), [o3d.Field](#), [o3d.Function](#), [o3d.Pack](#), [o3d.ParamArray](#), [o3d.ParamObject](#), [o3d.Skin](#), and [o3d.StreamBank](#).

[List of all members.](#)

---

## Detailed Description

Base class for all objects that can have their name set.

## Public Member Functions

- bool [isAClassName](#) (String *class\_name*)

## Public Attributes

- String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
```

```
t.isAClassName('o3d.Shape');           // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#)

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

## o3d.Pack Class Reference

Inherits [o3d.NamedObject](#).

[List of all members.](#)

---

## Detailed Description

A [Pack](#) object functions as a container for O3D objects. The [Pack](#) is used to control the lifetime scope of a collection of objects in bulk. The [Pack](#) object achieves this by simply storing a set of references to its contained objects, which ensures that the ref-counts for those objects never reach zero while the pack is alive.

See also:

[o3d.Pack.removeObject](#)

[o3d.Pack.destroy](#)

## Public Member Functions

- [destroy \(\)](#)
- bool [removeObject \(ObjectBase object\)](#)
- [ObjectBase createObject \(String type\\_name\)](#)
- [Texture2D createTexture2D \(Number width, Number height, Texture.Format format, Number levels, bool enable\\_render\\_surfaces\)](#)
- [TextureCUBE createTextureCUBE \(Number edge\\_length, Texture.Format format, Number levels, bool enable\\_render\\_surfaces\)](#)
- [RenderDepthStencilSurface createDepthStencilSurface \(Number width, Number height\)](#)
- Array [getObjects \(String name, String class\\_type\\_name\)](#)
- Array [getObjectsByClassName \(String class\\_type\\_name\)](#)
- [FileRequest createFileRequest \(String type\)](#)
- [ArchiveRequest createArchiveRequest \(\)](#)
- [Texture createTextureFromRawData \(RawData raw\\_data, bool generate\\_mips\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Array [objects](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

[ArchiveRequest](#) o3d.Pack.createArchiveRequest ( )

Creates an [ArchiveRequest](#) so we can stream in assets from an archive

### Returns:

an [ArchiveRequest](#)

[RenderDepthStencilSurface](#) o3d.Pack.createDepthStencilSurface ( Number *width*,  
Number *height*  
)

Creates a new [RenderDepthStencilSurface](#) object of a format suitable for use as a depth-stencil render target. Note: The dimensions of the [RenderDepthStencilSurface](#) must be a power of two.

### Parameters:

*width* The width of the [RenderSurface](#) in pixels  
*height* The height of the [RenderSurface](#) in pixels

### Returns:

The [RenderSurface](#) object.

[FileRequest](#) o3d.Pack.createFileRequest ( String *type* )

Creates a [FileRequest](#) to be used to asynchronously load a [Texture](#).

### Parameters:

*type* Must be "TEXTURE"

### Returns:

a [FileRequest](#)

[ObjectBase](#) o3d.Pack.createObject ( String *type\_name* )

Creates an Object by Class name.

Note: You may omit the 'o3d.'

### Parameters:

*type\_name* name of Class to create. Valid type names are:

- [o3d.Canvas](#)
- [o3d.CanvasLinearGradient](#)
- [o3d.CanvasPaint](#)
- [o3d.ClearBuffer](#)
- [o3d.Counter](#)
- [o3d.Curve](#)
- [o3d.DrawContext](#)
- [o3d.DrawElement](#)
- [o3d.DrawLine](#)
- [o3d.DrawPass](#)
- [o3d.Effect](#)
- [o3d.FunctionEval](#)
- [o3d.IndexBuffer](#)
- [o3d.Material](#)
- [o3d.ParamArray](#)
- [o3d.ParamObject](#)
- [o3d.Primitive](#)
- [o3d.RenderFrameCounter](#)
- [o3d.RenderNode](#)
- [o3d.RenderSurfaceSet](#)
- [o3d.Sampler](#)
- [o3d.SecondCounter](#)
- [o3d.Shape](#)
- [o3d.Skin](#)
- [o3d.SkinEval](#)
- [o3d.SourceBuffer](#)
- [o3d.State](#)
- [o3d.StateSet](#)
- [o3d.StreamBank](#)

- [o3d.Texture2D](#)
- [o3d.TextureCUBE](#)
- [o3d.TickCounter](#)
- [o3d.Transform](#)
- [o3d.TreeTraversal](#)
- [o3d.VertexBuffer](#)
- [o3d.Viewport](#)
- [o3d.Matrix4AxisRotation](#)
- [o3d.Matrix4Composition](#)
- [o3d.Matrix4Scale](#)
- [o3d.Matrix4Translation](#)
- [o3d.ParamOp2FloatsToFloat2](#)
- [o3d.ParamOp3FloatsToFloat3](#)
- [o3d.ParamOp4FloatsToFloat4](#)
- [o3d.ParamOp16FloatsToMatrix4](#)
- [o3d.TRSToMatrix4](#)

### Returns:

The created object.

```
Texture2D o3d.Pack.createTexture2D ( Number      width,
                                         Number      height,
                                         Texture.Format format,
                                         Number      levels,
                                         bool        enable_render_surfaces
                                       )
```

Creates a new [Texture2D](#) object of the specified size and format and reserves the necessary resources for it.

Note: If *enable\_render\_surfaces* is true, then the dimensions must be a power of two.

### Parameters:

<i>width</i>	The width of the texture area in texels (max = 2048)
<i>height</i>	The height of the texture area in texels (max = 2048)
<i>format</i>	The memory format of each texel
<i>levels</i>	The number of mipmap levels. Use zero to create the complete mipmap chain.

*enable\_render\_surfaces* If true, the texture object will expose [RenderSurface](#) objects through [GetRenderSurface\(...\)](#).

**Returns:**

The [Texture2D](#) object.

[TextureCUBE](#) o3d.Pack.createTextureCUBE ( Number *edge\_length*,  
[Texture.Format](#) *format*,  
Number *levels*,  
bool *enable\_render\_surfaces*  
)

Creates a new [TextureCUBE](#) object of the specified size and format and reserves the necessary resources for it. Note: If *enable\_render\_surfaces* is true, then the dimensions must be a power of two.

**Parameters:**

*edge\_length* The edge of the texture area in texels (max = 2048)  
*format* The memory format of each texel  
*levels* The number of mipmap levels. Use zero to create the complete mipmap chain.  
*enable\_render\_surfaces* If true, the texture object will expose [RenderSurface](#) objects through [GetRenderSurface\(...\)](#).

**Returns:**

The [TextureCUBE](#) object.

[Texture](#) o3d.Pack.createTextureFromRawData ( [RawData](#) *raw\_data*,  
bool *generate\_mips*  
)

Creates a [Texture](#) given a [RawData](#) object

**Parameters:**

*raw\_data* The [RawData](#) to create the texture from.  
*generate\_mips* True if you want O3D to generate mip maps for the texture.

**Returns:**

the [Texture](#)

o3d.Pack.destroy ( )

Removes all internal references to the [Pack](#) from the client. The pack, and all objects contained in it are permitted to be destroyed after the pack's destruction. Objects will only be destroyed after all references to them have been removed.

NOTE: Calling pack.destroy does NOT free your resources. It just releases the pack's reference to those resources. An example should hopefully make it clearer.

pack.destroy() is effectively almost the same as this.

```
var objectsInPack = pack.getObjectsByClassName('o3d.ObjectBase');
for (var ii = 0; ii < objectsInPack.length; ++ii) {
    pack removeObject(objectsInPack[ii]);
}
```

The only difference is that after all the objects are removed the pack itself will be released from the client. See documentation on pack.removeObject for why this is important.

It's important to note that many objects are only referenced by the pack. Textures, Effects, Materials, for example. That means the moment you call pack.destroy() those objects will be freed. If the client then tries to render and some objects are missing you'll immediately get an error.

Array o3d.Pack.getObjects ( String *name*,  
 String *class\_type\_name*  
 )

Search the pack for all objects of a certain class with a certain name.

Note that modifications to this array [e.g. push()] will not affect the underlying [Pack](#), while modifications to the array's members **will** affect them.

#### Parameters:

*name* Name to look for  
*class\_type\_n* the Class of the object. It is okay to pass base types for example "o3d.RenderNode"  
*ame* will return ClearBuffers, DrawPasses, etc...

#### Returns:

Array of Objects.

Array o3d.Pack.getObjectsByClassName ( String *class\_type\_name* )

Search the pack for all objects of a certain class

Note that modifications to this array [e.g. push()] will not affect the underlying [Pack](#), while modifications to the array's members **will** affect them.

#### Parameters:

*class\_type\_n* the Class of the object. It is okay to pass base types for example "o3d.RenderNode"  
*ame* will return ClearBuffers, DrawPasses, etc...

**Returns:**

Array of Objects.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.Pack.removeObject ( ObjectBase *object* )

Removes a pack's reference to an object. Any object created from pack.create\_\_\_ function can be removed. This releases the pack's reference to that object so if nothing else is referencing that object it will be deleted.

NOTE: Calling pack.removeObject does NOT automatically free your resource. It just releases the pack's reference to that resource. An example should hopefully make it clearer.

Suppose you make a transform like this:

```
var myTransform = pack.createObject('Transform');
myTransform.parent = g_client.root;
pack removeObject(myTransform);
```

In the code above, myTransform is referenced twice. Once by the pack, and again by g\_client.root So in this case, calling pack.removeObject() only releases the pack's reference leaving the reference by g\_client.root.

```
myTransform.parent = null;
```

Now the last reference has been removed and myTransform will be freed.

**Parameters:**

*object* Object to remove.

**Returns:**

True if the object was successfully removed. False if the object is not part of this pack.

**See also:**

[o3d.Pack.destroy](#)

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.Pack.objects](#)

All the objects managed by this pack.

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var objects = pack.objects;
```

```
for (var i = 0; i < objects.length; i++) {  
    var object = objects[i];  
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [Pack](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.Param Class Reference

Inherits [o3d.NamedObjectBase](#).

Inherited by [o3d.ParamBoolean](#), [o3d.ParamBoundingBox](#), [o3d.ParamDrawContext](#), [o3d.ParamDrawList](#), [o3d.ParamEffect](#), [o3d.ParamFloat](#), [o3d.ParamFloat2](#), [o3d.ParamFloat3](#), [o3d.ParamFloat4](#), [o3d.ParamFunction](#), [o3d.ParamInteger](#), [o3d.ParamMaterial](#), [o3d.ParamMatrix4](#), [o3d.ParamParamArray](#), [o3d.ParamSampler](#), [o3d.ParamSkin](#), [o3d.ParamState](#), [o3d.ParamStreamBank](#), [o3d.ParamString](#), o3d.ParamTexture, [o3d.ParamTransform](#), and [o3d.ParamVertexBufferStream](#).

[List of all members](#).

---

## Detailed Description

Params store data defined name/value pairs on ParamObjects. Each [Param](#) has a name, a type and a value that can be set and queried. One of their uses is to hold "uniform constants" used to parameterize shaders. Params can be connected in a source/destination fashion such that a target [Param](#) gets its value from the source param.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) )

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( )`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param )`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( )`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) `o3d.Param.inputConnection`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

### Array [o3d.Param.outputConnections](#)

The output connections for this param.

This property is read-only.

### bool [o3d.Param.readOnly](#)

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

### bool [o3d.Param.updateInput](#)

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

## **o3d.ParamFloat Class Reference**

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) class which stores a single Number.

### Public Member Functions

- bool [bind](#) ([Param](#) source\_param)

- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Number [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
```

```
t.isAClassName('o3d.Shape');           // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Number [o3d.ParamFloat.value](#)

The Number stored by the [Param](#).

## o3d.ParamFloat2 Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) class which stores a [Float2](#).

## Public Member Functions

- [set](#) (Number v0, Number v1)
- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)
- [unbindOutputs](#) ()
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float2 value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param](#) source\_param ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any

*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.ParamFloat2.set ( Number *v0*,  
 Number *v1*  
 )

Sets the value of [ParamFloat2](#) by 2 numbers.

**Parameters:**

*v0* first value.

*v1* second value.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

### [Float2](#) [o3d.ParamFloat2.value](#)

Sets the [Float2](#) stored by the [Param](#) by an array of 2 numbers.

## **o3d.ParamFloat3 Class Reference**

Inherits [o3d.Param](#).

[List of all members.](#)

---

### **Detailed Description**

A [Param](#) class which stores a [Float3](#).

### **Public Member Functions**

- [set](#) (Number v0, Number v1, Number v2)
- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)
- [unbindOutputs](#) ()
- bool [isAClassName](#) (String class\_name)

# Public Attributes

- [Float3 value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

# Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

```
o3d.ParamFloat3.set ( Number v0,  
                      Number v1,  
                      Number v2  
                    )
```

Sets the entries of the value of [ParamFloat3](#) to the 3 given numbers.

**Parameters:**

v0 first value.  
v1 second value.  
v2 third value.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) destination\_param ) [inherited]

Breaks a specific param-bind output connection on this param.

**Parameters:**

destination\_param param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Float3](#) [o3d.ParamFloat3.value](#)

Sets the [Float3](#) stored by the [Param](#) by an array of 3 numbers.

# o3d.ParamFloat4 Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) class which stores a [Float4](#).

## Public Member Functions

- [set](#) (Number v0, Number v1, Number v2, Number v3)
- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)
- [unbindOutputs](#) ()
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Float4 value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.ParamFloat4.set ( Number *v0*,  
 Number *v1*,  
 Number *v2*,  
 Number *v3*  
 )

Sets the value of [ParamFloat4](#) by 4 numbers.

## Parameters:

*v0* first value.  
*v1* second value.  
*v2* third value.  
*v3* fourth value.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param )` [inherited]  
Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( )` [inherited]  
Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className` [inherited]  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId` [inherited]  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection` [inherited]  
The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name` [inherited]  
The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections` [inherited]  
The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Float4 o3d.ParamFloat4.value](#)

Sets the [Float4](#) stored by the [Param](#) by an array of 4 numbers.

## o3d.ParamMatrix4 Class Reference

Inherits [o3d.Param](#).

Inherited by [o3d.ProjectionInverseParamMatrix4](#), [o3d.ProjectionInverseTransposeParamMatrix4](#),  
[o3d.ProjectionParamMatrix4](#), [o3d.ProjectionTransposeParamMatrix4](#), [o3d.ViewInverseParamMatrix4](#),  
[o3d.ViewInverseTransposeParamMatrix4](#), [o3d.ViewParamMatrix4](#),  
[o3d.ViewProjectionInverseParamMatrix4](#), [o3d.ViewProjectionInverseTransposeParamMatrix4](#),  
[o3d.ViewProjectionParamMatrix4](#), [o3d.ViewProjectionTransposeParamMatrix4](#),  
[o3d.ViewTransposeParamMatrix4](#), [o3d.WorldInverseParamMatrix4](#),  
[o3d.WorldInverseTransposeParamMatrix4](#), [o3d.WorldParamMatrix4](#),  
[o3d.WorldTransposeParamMatrix4](#), [o3d.WorldViewInverseParamMatrix4](#),  
[o3d.WorldViewInverseTransposeParamMatrix4](#), [o3d.WorldViewParamMatrix4](#),

[o3d.WorldViewProjectionInverseParamMatrix4](#),  
[o3d.WorldViewProjectionInverseTransposeParamMatrix4](#), [o3d.WorldViewProjectionParamMatrix4](#),  
[o3d.WorldViewProjectionTransposeParamMatrix4](#), and [o3d.WorldViewTransposeParamMatrix4](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) class which stores a 4-by-4 matrix.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.  
The source must be compatible with this parameter.

**Parameters:**

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Param** [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

**String** [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

**Array** [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

**bool** [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

**bool** [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#)

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## o3d.ParamInteger Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) class which stores an integer.

### Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

### Public Attributes

- Number [value](#)
- bool [updateInput](#)
- bool [readOnly](#)
- [Param inputConnection](#)

- Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param destination\\_param](#) ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections [inherited]`

The output connections for this param.

This property is read-only.

`bool o3d.Param.readOnly [inherited]`

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Number [o3d.ParamInteger.value](#)

The integer stored by the [Param](#).

## o3d.ParamBoolean Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) class which stores a boolean.

### Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)

- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- bool [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source\_param* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

bool [o3d.ParamBoolean.value](#)

The boolean stored by the [Param](#).

## **o3d.String Class Reference**

Inherits [o3d.Param](#).

[List of all members](#).

---

# Detailed Description

A [Param](#) which stores a string.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- String [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[String](#) [o3d.ParamString.value](#)

The string stored by the [Param](#).

# o3d.ParamSampler Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) which stores a [Texture](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Sampler value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.  
The source must be compatible with this parameter.

**Parameters:**

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#) [inherited]**

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Param [o3d.Param.inputConnection](#) [inherited]**

The input connection for this param.

This property is read-only.

**String [o3d.NamedObjectBase.name](#) [inherited]**

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

**Array [o3d.Param.outputConnections](#) [inherited]**

The output connections for this param.

This property is read-only.

**bool [o3d.Param.readOnly](#) [inherited]**

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

**bool [o3d.Param.updateInput](#) [inherited]**

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

#### [Sampler](#) [o3d.ParamSampler.value](#)

The [Sampler](#) stored by the [Param](#).

## o3d.ParamMaterial Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [Material](#).

### Public Member Functions

- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)
- [unbindOutputs](#) ()
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- [Material value](#)
- bool [updateInput](#)
- bool [readOnly](#)
- [Param inputConnection](#)

- Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param destination\\_param](#) ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections [inherited]`

The output connections for this param.

This property is read-only.

`bool o3d.Param.readOnly [inherited]`

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Material](#) [o3d.ParamMaterial.value](#)

The [Material](#) stored by the [Param](#).

## o3d.ParamState Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [State](#).

### Public Member Functions

- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)

- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [State value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[State](#) [o3d.ParamState.value](#)

The [State](#) stored by the [Param](#).

## o3d.ParamEffect Class Reference

Inherits [o3d.Param](#).

[List of all members](#).

---

# Detailed Description

A [Param](#) which stores a [Effect](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Effect value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Effect o3d.ParamEffect.value](#)

The [Effect](#) stored by the [Param](#).

# o3d.ParamTransform Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) which stores a [Transform](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Transform value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.  
The source must be compatible with this parameter.

**Parameters:**

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Param** [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

**String** [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

**Array** [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

**bool** [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

**bool** [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

#### [Transform](#) [o3d.ParamTransform.value](#)

The [Transform](#) stored by the [Param](#).

---

## **o3d.ParamDrawList Class Reference**

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [DrawList](#).

### Public Member Functions

- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)
- [unbindOutputs](#) ()
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- [DrawList value](#)
- bool [updateInput](#)
- bool [readOnly](#)
- [Param inputConnection](#)

- Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param destination\\_param](#) ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections [inherited]`

The output connections for this param.

This property is read-only.

`bool o3d.Param.readOnly [inherited]`

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[DrawList o3d.ParamDrawList.value](#)

The [DrawList](#) stored by the [Param](#).

## o3d.ParamDrawContext Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [DrawContext](#).

### Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)

- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [DrawContext value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[DrawContext](#) [o3d.ParamDrawContext.value](#)

The [DrawContext](#) stored by the [Param](#).

## o3d.ParamArray Class Reference

Inherits [o3d.NamedObject](#).

[List of all members.](#)

---

## Detailed Description

A [ParamArray](#) is an object that holds an array of Params.

## Public Member Functions

- Param createParam ( Number index, String param\_type\_name)
  - Param getParam ( Number index)
  - removeParams ( Number start\_index, Number num\_to\_remove)
  - resize ( Number num\_params, String param\_type\_name)
  - bool isAClassName (String class\_name)

## Public Attributes

- Array params
  - Number length
  - String name
  - Id clientId
  - String className

# Member Function Documentation

Param o3d.ParamArray.createParam ( Number *index*,  
                                 String    *param\_type\_name*  
                                 )

Creates a [Param](#) of the given type at the index requested. If a [Param](#) already exists at that index the new param will be replace it. If the index is past the end of the current array params of the requested type will be created to fill out the array to the requested index.

### Parameters:

*index* index at which to create new param.  
*param\_type\_name* The type of [Param](#) to create. For a list of valid types see  
[ParamObject.createParam](#)

## Returns:

[Param](#) created at index or null if failure.

[Param](#) o3d.ParamArray.getParam ( Number *index* )

Gets a [Param](#) by index.

**Parameters:**

*index* Index of [Param](#) to get.

**Returns:**

The [Param](#) at index, or null if out of range.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.ParamArray.removeParams ( Number *start\_index*,  
 Number *num\_to\_remove*  
 )

Removes a range of params.

**Parameters:**

*start\_index* Index of first param to remove.

*num\_to\_remove* The number of params to remove starting at *start\_index*.

o3d.ParamArray.resize ( Number *num\_params*,  
 String *param\_type\_name*  
 )

Resizes the array.

**Parameters:**

*num\_params* The number of params to make the array.

*param\_type\_n* The type of [Param](#) to create if resizing requires params to be created. For a list of

*ame*

valid types see [ParamObject.createParam](#)

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.ParamArray.length](#)

Returns the number of parameters in this [ParamArray](#).

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamArray.params](#)

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = ParamArray.params;  
for (var i = 0; i < params.length; i++) {  
    var param = params[i];
```

}

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.ParamParamArray Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [ParamArray](#).

### Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

### Public Attributes

- [ParamArray value](#)
- bool [updateInput](#)
- bool [readOnly](#)
- [Param inputConnection](#)
- Array [outputConnections](#)
- String [name](#)
- Id [clientId](#)

- String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param destination\\_param](#) ) [inherited]

Breaks a specific param-bind output connection on this param.

### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[ParamArray o3d.ParamParamArray.value](#)

The [ParamArray](#) stored by the [Param](#).

## o3d.ParamObject Class Reference

Inherits [o3d.NamedObject](#).

Inherited by [o3d.Canvas](#), [o3d.CanvasPaint](#), [o3d.CanvasShader](#), [o3d.Counter](#), [o3d.DrawContext](#), [o3d.DrawElement](#), [o3d.Effect](#), [o3d.Element](#), [o3d.FunctionEval](#), [o3d.Material](#), [o3d.Matrix4AxisRotation](#), [o3d.Matrix4Composition](#), [o3d.Matrix4Scale](#), [o3d.Matrix4Translation](#), [o3d.ParamOp16FloatsToMatrix4](#), [o3d.ParamOp2FloatsToFloat2](#), [o3d.ParamOp3FloatsToFloat3](#), [o3d.ParamOp4FloatsToFloat4](#), [o3d.RawData](#), [o3d.RenderNode](#), [o3d.RenderSurfaceBase](#), [o3d.Sampler](#), [o3d.Shape](#), [o3d.State](#), [o3d.Texture](#), [o3d.Transform](#), [o3d.TRSToMatrix4](#), and [o3d.VertexSource](#).

[List of all members](#).

---

## Detailed Description

A [ParamObject](#) is the base class for all objects that can have Params. Defines methods to add and remove params, search for params, etc.

## Public Member Functions

- Param createParam (String param\_name, String param\_type\_name)
  - Param getParam (String param\_name)
  - bool removeParam (Param param)
  - copyParams (ParamObject source\_param\_object)
  - bool isAClassName (String class\_name)

## Public Attributes

- Array params
  - String name
  - Id clientId
  - String className

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source param object* param object to copy params from.

```
Param o3d.ParamObject.createParam ( String param_name,  
                                    String param_type_name  
                                )
```

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the Param to be created.

*param type name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
  - 'o3d.ParamBoundingBox',

- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',

- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* )

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* )

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#)

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.ParamOp2FloatsToFloat2 Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) operation that takes 2 floats to produce a [Float2](#).

### Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- Number [input0](#)
- Number [input1](#)
- [Float2 output](#)

- Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',

- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',

- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.ParamOp2FloatsToFloat2.input0](#)

The first value for the [Float2](#).

Number [o3d.ParamOp2FloatsToFloat2.input1](#)

The second value for the [Float2](#).

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

[Float2 o3d.ParamOp2FloatsToFloat2.output](#)

The [Float2](#) that results from the inputs.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.ParamOp3FloatsToFloat3 Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) operation that takes 3 floats to produce a [Float3](#).

### Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

### Public Attributes

- Number [input0](#)
- Number [input1](#)
- Number [input2](#)
- [Float3 output](#)
- Array [params](#)

- String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',

- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',

- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.ParamOp3FloatsToFloat3.input0](#)

The first value for the [Float3](#).

Number [o3d.ParamOp3FloatsToFloat3.input1](#)

The second value for the [Float3](#).

Number [o3d.ParamOp3FloatsToFloat3.input2](#)

The third value for the [Float3](#).

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

[Float3 o3d.ParamOp3FloatsToFloat3.output](#)

The [Float3](#) that results from the inputs.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.ParamOp4FloatsToFloat4 Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) operation that takes 4 floats to produce a [Float4](#).

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [input0](#)
- Number [input1](#)

- Number [input2](#)
  - Number [input3](#)
  - [Float4 output](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',

- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',

- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.ParamOp4FloatsToFloat4.input0](#)

The first value for the [Float4](#).

Number [o3d.ParamOp4FloatsToFloat4.input1](#)

The second value for the [Float4](#).

Number [o3d.ParamOp4FloatsToFloat4.input2](#)

The third value for the [Float4](#).

Number [o3d.ParamOp4FloatsToFloat4.input3](#)

The fourth value for the [Float4](#).

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Float4 [o3d.ParamOp4FloatsToFloat4.output](#)

The [Float4](#) that results from the inputs.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## **o3d.ParamOp16FloatsToMatrix4 Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

### **Detailed Description**

A [Param](#) operation that takes 16 floats to produce a Matrix4.

### **Public Member Functions**

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)

- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [input0](#)
  - Number [input1](#)
  - Number [input2](#)
  - Number [input3](#)
  - Number [input4](#)
  - Number [input5](#)
  - Number [input6](#)
  - Number [input7](#)
  - Number [input8](#)
  - Number [input9](#)
  - Number [input10](#)
  - Number [input11](#)
  - Number [input12](#)
  - Number [input13](#)
  - Number [input14](#)
  - Number [input15](#)
  - Matrix4 [output](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

**Parameters:**

*source\_param\_object* param object to copy params from.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',

- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.ParamOp16FloatsToMatrix4.input0](#)

The first value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input1](#)

The second value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input10](#)

The eleventh value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input11](#)

The twelfth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input12](#)

The thirteenth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input13](#)

The fourteenth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input14](#)

The fifteenth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input15](#)

The sixteenth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input2](#)

The third value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input3](#)

The fourth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input4](#)

The fifth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input5](#)

The sixth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input6](#)

The seventh value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input7](#)

The eighth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input8](#)

The ninth value for the [Float4](#).

Number [o3d.ParamOp16FloatsToMatrix4.input9](#)

The tenth value for the [Float4](#).

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Matrix4 [o3d.ParamOp16FloatsToMatrix4.output](#)

The Matrix4 that results from the inputs.

This property is read-only.

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.TRSToMatrix4 Class Reference

Inherits [o3d.ParamObject](#).

[List of all members](#).

---

# Detailed Description

A [Param](#) operation that takes 9 floats to produce a 4-by-4 matrix. The 9 floats encode a translation vector, angles of rotation around the x, y, and z axes, and three scaling factors. The resulting transformation scales first, then then rotates around the z-axis, then the y-axis, then the x-axis, then translates.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [translateX](#)
  - Number [translateY](#)
  - Number [translateZ](#)
  - Number [rotateX](#)
  - Number [rotateY](#)
  - Number [rotateZ](#)
  - Number [scaleX](#)
  - Number [scaleY](#)
  - Number [scaleZ](#)
  - Matrix4 [output](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',

- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
 Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Matrix4** [o3d.TRSToMatrix4.output](#)

The Matrix4 that results from the inputs.

This property is read-only.

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

**Number** [o3d.TRSToMatrix4.rotateX](#)

The x rotation for the Matrix4.

**Number** [o3d.TRSToMatrix4.rotateY](#)

The y rotation for the Matrix4.

Number [o3d.TRSToMatrix4.rotateZ](#)

The z rotation for the Matrix4.

Number [o3d.TRSToMatrix4.scaleX](#)

The x scale for the Matrix4.

Number [o3d.TRSToMatrix4.scaleY](#)

The y scale for the Matrix4.

Number [o3d.TRSToMatrix4.scaleZ](#)

The z scale for the Matrix4.

Number [o3d.TRSToMatrix4.translateX](#)

The x translation for the Matrix4.

Number [o3d.TRSToMatrix4.translateY](#)

The y translation for the Matrix4.

Number [o3d.TRSToMatrix4.translateZ](#)

The z translation for the Matrix4.

## **o3d.Primitive Class Reference**

Inherits [o3d.Element](#).

[List of all members.](#)

---

### **Detailed Description**

A [Primitive](#) is a type of [Element](#) that is made from a list of points, lines or triangles that use a single material.

### **Public Types**

- enum [PrimitiveType](#)

## Public Member Functions

- [DrawElement createDrawElement \(Pack pack, Material material\)](#)
- [RayIntersectionInfo intersectRay \(Number position\\_stream\\_index, State.Cull cull, Point3 start, Point3 end\)](#)
- [BoundingBox getBoundingBox \(Number position\\_stream\\_index\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- [bool removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- [bool isAClassName \(String class\\_name\)](#)

## Public Attributes

- [PrimitiveType primitiveType](#)
  - Number [numberVertices](#)
  - Number [numberPrimitives](#)
  - Number [startIndex](#)
  - [StreamBank streamBank](#)
  - [IndexBuffer indexBuffer](#)
  - [Material material](#)
  - [BoundingBox boundingBox](#)
  - [Float3 zSortPoint](#)
  - Number [priority](#)
  - bool [cull](#)
  - [Shape owner](#)
  - Array [drawElements](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Enumeration Documentation

enum [o3d.Primitive.PrimitiveType](#)

- POINTLIST,
- LINELIST,
- LINESTRIP,
- TRIANGLELIST,
- TRIANGLESTRIP,
- TRIANGLEFAN

Type of geometric primitives used by the [Primitive](#).

---

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[DrawElement](#) `o3d.Element.createDrawElement ( Pack pack,  
                         Material material  
                         ) [inherited]`

Creates a [DrawElement](#) for this [Element](#). Note that unlike [Shape.createDrawElements](#) and [Transform.createDrawElements](#) this one will create more than one element for the same material.

## Parameters:

*pack* pack used to manage created [DrawElement](#).  
*material* material to use for [DrawElement](#). If you pass null it will use the material on this [Element](#). This allows you to easily setup the default (just draw as is) by passing null or setup a shadow pass by passing in a shadow material.

## Returns:

The created draw element.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
                         String param_type_name`

) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',

- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[BoundingBox](#) o3d.Element.getBoundingBox ( Number *position\_stream\_index* ) [inherited]  
Computes the bounding box in same coordinate system as the specified POSITION stream.

**Parameters:**

*position\_stream\_index* Index of POSITION stream.

**Returns:**

The boundingbox for this element in local space.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[RayIntersectionInfo](#) o3d.Element.intersectRay ( Number *position\_stream\_index*,  
[State.Cull](#) *cull*,  
[Point3](#) *start*,  
[Point3](#) *end*  
 ) [inherited]

Computes the intersection of a ray in the same coordinate system as the specified POSITION stream.

**Parameters:**

*position\_stream\_index* Index of POSITION stream.  
*cull* which side of the triangles to ignore.  
*start* position of start of ray in local space.  
*end* position of end of ray. in local space.

**Returns:**

[RayIntersectionInfo](#) class. If valid() is false then something was wrong, Check client.GetLastError(). If intersected() is true then the ray intersected a something. position() is the exact point of intersection.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

[BoundingBox](#) `o3d.Element.boundingBox` [inherited]

The [BoundingBox](#) for this element. If culling is on this bounding box will be tested against the view frustum of any draw context used to render this [Element](#).

[String](#) `o3d.ObjectBase.className` [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

[Id](#) `o3d.ObjectBase.clientId` [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[bool](#) `o3d.Element.cull` [inherited]

The cull settings for this element. If true this [Element](#) will be culled by the bounding box above compared to the view frustum it is being rendered with.

[Array](#) `o3d.Element.drawElements` [inherited]

Gets all the DrawElements under this [Element](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var drawElements = element.drawElements;  
for (var i = 0; i < drawElements.length; i++) {  
    var drawElement = drawElements[i];
```

}

Note that modifications to this array [e.g. push()] will not affect the underlying [Element](#), while modifications to the members of the array **will** affect them.

This property is read-only.

#### [IndexBuffer o3d.Primitive.indexBuffer](#)

The index buffer for the primitive. If null the primitive is non-indexed.

#### [Material o3d.Element.material \[inherited\]](#)

The [Material](#) for this element.

#### [String o3d.NamedObject.name \[inherited\]](#)

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### [Number o3d.Primitive.numberPrimitives](#)

The number of rendering primitives (i.e., triangles, points, lines) the primitive has.

#### [Number o3d.Primitive.numberVertices](#)

The number of vertices the primitive has.

#### [Shape o3d.Element.owner \[inherited\]](#)

The current owner of this Draw [Element](#). Pass in null to stop being owned.

Note: Elements are referenced by the [Pack](#) they are created in and their owner. If the [Element](#) is removed from its [Pack](#) then setting the owner to null will free the [Element](#). Or, visa versa, if you set the Element's owner to null then removing it from its [Pack](#) will free the [Element](#).

#### [Array o3d.ParamObject.params \[inherited\]](#)

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
```

}

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### [PrimitiveType o3d.Primitive.primitiveType](#)

The type of primitive the primitive is (i.e., Primitive.POINTLIST, Primitive.LINELIST, Primitive.TRIANGLELIST, etc.)

#### Number [o3d.Element.priority](#) [inherited]

The priority for this element. Used to sort if this [Element](#) is drawn by a [DrawPass](#) that is set to sort by priority.

#### [StreamBank o3d.Primitive.streamBank](#)

The stream bank this primitive uses for vertices.

#### [Float3 o3d.Element.zSortPoint](#) [inherited]

The z sort point for this element. If this [Element](#) is drawn by a [DrawPass](#) that is set to sort by z order this value will be multiplied by the worldViewProjection matrix to compute a z value to sort by.

# **o3d.RawData Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

A [RawData](#) object contains raw binary data which could contain image, audio, text, or other information.

```
var request = g_pack.createArchiveRequest();
```

```

request.onfileavailable = function() {
    var texture = g_pack.createTextureFromRawData(request.data, true);
    ...
};

request.send();

```

## Public Member Functions

- [discard \(\)](#)
- [flush \(\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- [bool removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- [bool isAClassName \(String class\\_name\)](#)

## Public Attributes

- String [stringValue](#)
  - String [uri](#)
  - size\_t [length](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

**Parameters:**

*source\_param\_object* param object to copy params from.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',

- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

`o3d.RawData.discard ( )`

Discards all the resources associated with this data object.

`o3d.RawData.flush ( )`

Flushes the memory resources associated with this data object, but keeps a cache in case the data object is used later.

[Param](#) `o3d.ParamObject.getParam ( String param_name ) [inherited]`

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**size\_t** [o3d.RawData.length](#)

The length in bytes of the [RawData](#).

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

**String** [o3d.RawData.stringValue](#)

Returns the raw data as a string. The data must be a valid utf-8 string and the uri must end in .json, .txt, .xml, .ini or .csv

This property is read-only.

String [o3d.RawData.uri](#)

The uri of the [RawData](#).

This property is read-only.

## **o3d.RayIntersectionInfo Class Reference**

[List of all members.](#)

---

### **Detailed Description**

A [RayIntersectionInfo](#) is used to return the results of ray intersection tests.

### **Public Attributes**

- bool [valid](#)
  - bool [intersected](#)
  - Point3 [position](#)
  - Number [primitiveIndex](#)
- 

### **Member Data Documentation**

bool [o3d.RayIntersectionInfo.intersected](#)

True if this ray intersection intersected something.

This property is read-only.

Point3 [o3d.RayIntersectionInfo.position](#)

The position the ray intersected something.

This property is read-only.

Number [o3d.RayIntersectionInfo.primitiveIndex](#)

The index of the primitive that was intersected.

This property is read-only.

bool [o3d.RayIntersectionInfo.valid](#)

True if this ray intersection info is valid. For example if you call element.intersectRay on an element that has no vertex buffers the result will be invalid.

This property is read-only.

## o3d.RenderEvent Class Reference

[List of all members.](#)

---

### Detailed Description

This class is used to pass information to a registered onrender callback.

### Public Attributes

- Number [elapsedTime](#)
  - Number [renderTime](#)
  - Number [activeTime](#)
  - Number [transformsProcessed](#)
  - Number [transformsCulled](#)
  - Number [drawElementsProcessed](#)
  - Number [drawElementsCulled](#)
  - Number [drawElementsRendered](#)
  - Number [primitivesRendered](#)
- 

### Member Data Documentation

Number [o3d.RenderEvent.activeTime](#)

The time it took to render and process tick callbacks, render callbacks, counter callbacks.

This property is read-only.

### Number [o3d.RenderEvent.drawElementsCulled](#)

The number of draw elements that were culled last frame directly. (vs culled hierarchically)

This property is read-only.

### Number [o3d.RenderEvent.drawElementsProcessed](#)

The number of draw elements processed last frame.

This is the number of draw elements the renderer considered for rendering. If a transform is not traversed either because it is not in one of the subtrees the TreeTraversals are setup to traverse or because it is marked as visible = false then any draw elements in that part of the transform graph are not processed.

This property is read-only.

### Number [o3d.RenderEvent.drawElementsRendered](#)

The number of draw elements rendered last frame. Note: a draw element can be rendered more than once per frame based on how many transforms it is under and how many DrawPasses use the DrawLists it is put on.

This property is read-only.

### Number [o3d.RenderEvent.elapsedTime](#)

The elapsed time in seconds since the last frame was rendered.

This property is read-only.

### Number [o3d.RenderEvent.primitivesRendered](#)

The number of low-level primitives rendered last frame. This is the sum of the number primitives (triangles, lines) submitted to the renderer.

This property is read-only.

### Number [o3d.RenderEvent.renderTime](#)

The time it took to render.

This property is read-only.

### Number [o3d.RenderEvent.transformsCulled](#)

The number of transforms that were culled last frame directly.

Of the transforms processed, how many were culled.

This property is read-only.

Number [o3d.RenderEvent.transformsProcessed](#)

The number of transforms processed last frame.

This is the number of transforms the renderer considered for traversing. A [Transform](#) may not be considered for traversing either because it is not in one of the subtrees the TreeTraversals are setup to traverse or because it one of its parents was culled or set to visible = false.

This property is read-only.

## o3d.RenderNode Class Reference

Inherits [o3d.ParamObject](#).

Inherited by [o3d.ClearBuffer](#), [o3d.DrawPass](#), [o3d.RenderSurfaceSet](#), [o3d.StateSet](#), [o3d.TreeTraversal](#), and [o3d.Viewport](#).

[List of all members.](#)

---

### Detailed Description

[RenderNode](#) is the base of all RenderNodes in the render graph. All RenderNodes have o3d.priority, o3d.active parameters. RenderNodes are rendered in order of priority.

### Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)
- Array [getRenderNodesByClassNameInTree \(String class\\_name\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- bool [removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name )` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',

- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',

- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array o3d.RenderNode.getRenderNodesByClassNameInTree ( String *class\_name* )  
Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* )  
Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( )

Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

# Member Data Documentation

bool [o3d.RenderNode.active](#)

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#)

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

`RenderNode.getTransformsByNameInTree` which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode o3d.RenderNode.parent](#)

Sets the parent of the node by re-parenting the node under `parent_node`. Setting `parent_node` to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

Number [o3d.RenderNode.priority](#)

Sets the priority of this render node. lower priorities are rendered first.

## o3d.RenderSurfaceBase Class Reference

Inherits [o3d.ParamObject](#).

Inherited by [o3d.RenderDepthStencilSurface](#), and [o3d.RenderSurface](#).

[List of all members.](#)

---

### Detailed Description

A [RenderSurfaceBase](#) is the base for [RenderSurface](#) and [RenderDepthStencilSurface](#).

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [width](#)
  - Number [height](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',

- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.RenderSurfaceBase.height](#)

The height of the surface, in pixels.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

Number [o3d.RenderSurfaceBase.width](#)

The width of the surface, in pixels.

This property is read-only.

## o3d.RenderSurface Class Reference

Inherits [o3d.RenderSurfaceBase](#).

[List of all members.](#)

---

### Detailed Description

A [RenderSurface](#) encapsulates the notion of a renderable surface. When used in conjunction with a [RenderSurfaceSet](#) node in the render graph, the API allows for rendering of primitives to the given surface. [RenderSurface](#) objects are not constructable through the [Pack](#) API, they may only be accessed through the texture `getRenderSurface(...)` interfaces.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Texture texture](#)
  - Number [width](#)
  - Number [height](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

`param_name` The name of the [Param](#) to be created.

`param_type_name` The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',

- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.RenderSurfaceBase.height](#) [inherited]

The height of the surface, in pixels.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[Texture o3d.RenderSurface.texture](#)

The texture in which this surface is contained.

This property is read-only.

Number [o3d.RenderSurfaceBase.width](#) [inherited]

The width of the surface, in pixels.

This property is read-only.

## o3d.RenderDepthStencilSurface Class

### Reference

Inherits [o3d.RenderSurfaceBase](#).

[List of all members](#).

---

# Detailed Description

A [RenderDepthStencilSurface](#) represents a depth stencil render surface.

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [width](#)
  - Number [height](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',

- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

## Returns:

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

## Parameters:

*param\_name* Name to search for.

## Returns:

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
```

```
t.isAClassName('o3d.Shape');           // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.RenderSurfaceBase.height](#) [inherited]

The height of the surface, in pixels.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

Number [o3d.RenderSurfaceBase.width](#) [inherited]

The width of the surface, in pixels.

This property is read-only.

## o3d.RenderSurfaceSet Class Reference

Inherits [o3d.RenderNode](#).

[List of all members](#).

---

### Detailed Description

A [RenderSurfaceSet](#) node will bind depth and color [RenderSurface](#) nodes to the current rendering context. All RenderNodes descending from the given [RenderSurfaceSet](#) node will operate on the

contents of the bound depth and color buffers. Note the following usage constraints of this node: 1) If both a color and depth surface is bound, then they must be of matching dimensions. 2) At least one of render\_surface and render\_depth\_surface must non-null.

## Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)
- Array [getRenderNodesByClassNameInTree \(String class\\_name\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- bool [removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [RenderSurface renderSurface](#)
  - [RenderDepthStencilSurface renderDepthStencilSurface](#)
  - Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`  
Copies all the params from a the given source\_param\_object to this param object. Does not replace any

currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

Param o3d.ParamObject.createParam ( String *param\_name*,  
   String *param\_type\_name*  
   )

[inherited]

Creates a Param with the given name and type on the ParamObject. Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the Param to be created.

*param\_type\_name* The type of Param to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',

- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
 Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array

o3d.RenderNode.getRenderNodesByClassNameInTree

( String  $\frac{class\_nam}{e}$  ) [inherited]

Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

$class\_name$  class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is  $class\_name$ .

Array o3d.RenderNode.getRenderNodesByNameInTree ( String  $name$  ) [inherited]

Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

$name$  Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]

Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String  $class\_name$  ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while

modifications to the array's members **will** affect them.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

### [RenderNode](#) [o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under parent\_node. Setting parent\_node to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

### [Number](#) [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

### [RenderDepthStencilSurface](#) [o3d.RenderSurfaceSet.renderDepthStencilSurface](#)

The render depth stencil surface to which the depth contents of all [RenderNode](#) children should be drawn.

### [RenderSurface](#) [o3d.RenderSurfaceSet.renderSurface](#)

The render surface to which the color contents of all [RenderNode](#) children should be drawn.

## **o3d.Sampler Class Reference**

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

[Sampler](#) is the base of all texture samplers. [Texture](#) samplers encapsulate a texture reference with a set of states that define how the texture gets applied to a surface. [Sampler](#) states are set either via Params defined on the [Sampler](#) object or directly via one the convenience methods defined on the [Sampler](#). The following states are supported (default values are in parenthesis):

- 'addressModeU' (WRAP)
- 'addressModeV' (WRAP)
- 'addressModeW' (WRAP)
- 'magFilter' (LINEAR)
- 'minFilter' (LINEAR)
- 'mipFilter' (POINT)
- 'borderColor' ([0,0,0,0])

- 'maxAnisotropy' (1)

## Public Types

- enum [AddressMode](#)
- enum [FilterType](#)

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [AddressMode addressModeU](#)
  - [AddressMode addressModeV](#)
  - [AddressMode addressModeW](#)
  - [FilterType magFilter](#)
  - [FilterType minFilter](#)
  - [FilterType mipFilter](#)
  - [Float4 borderColor](#)
  - Number [maxAnisotropy](#)
  - [Texture texture](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Enumeration Documentation

## enum [o3d.Sampler.AddressMode](#)

Controls what happens with texture coordinates outside the [0..1] range.

- WRAP
- MIRROR
- CLAMP
- BORDER

## enum [o3d.Sampler.FilterType](#)

[Texture](#) filtering types.

- NONE
  - POINT
  - LINEAR
  - ANISOTROPIC
- 

# Member Function Documentation

## [o3d.ParamObject](#).copyParams ( [ParamObject](#) *source\_param\_object* ) [inherited]

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

## [Param](#) [o3d.ParamObject](#).createParam ( String *param\_name*, String *param\_type\_name* ) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',

- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',

- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

```
bool o3d.ParamObject.removeParam ( Param param ) [inherited]  
Removes a Param from a ParamObject.
```

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

### [AddressMode](#) [o3d.Sampler.addressModeU](#)

The texture address mode for the u coordinate.

### [AddressMode](#) [o3d.Sampler.addressModeV](#)

The texture address mode for the v coordinate.

### [AddressMode](#) [o3d.Sampler.addressModeW](#)

The texture address mode for the w coordinate.

### [Float4](#) [o3d.Sampler.borderColor](#)

Color returned for texture coordinates outside the [0,1] range when the address mode is set to BORDER.

### [String](#) [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

### [Id](#) [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

#### [FilterType o3d.Sampler.magFilter](#)

The magnification filter. Valid values for the mag filter are: POINT and LINEAR.

#### [Number o3d.Sampler.maxAnisotropy](#)

Degree of anisotropy used when the ANISOTROPIC filter type is used.

#### [FilterType o3d.Sampler.minFilter](#)

The minification filter. Valid values for the min filter are: POINT, LINEAR and ANISOTROPIC.

#### [FilterType o3d.Sampler.mipFilter](#)

The mipmap filter used during minification. Valid values for the mip filter are: NONE, POINT and LINEAR.

#### [String o3d.NamedObject.name \[inherited\]](#)

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### [Array o3d.ParamObject.params \[inherited\]](#)

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### [Texture o3d.Sampler.texture](#)

The [Texture](#) object used by this [Sampler](#).

# o3d.Shape Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

The [Shape](#) represents a collection of Elements. The typical example is a cube with 6 faces where each face uses a different material would be represented as 1 [Shape](#) with 6 Elements, one for each material.

## Public Member Functions

- [createDrawElements \(Pack pack, Material material\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- [bool removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- [bool isAClassName \(String class\\_name\)](#)

## Public Attributes

- Array [elements](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given source\_param\_object to this param object. Does not replace any

currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

```
o3d.Shape.createDrawElements ( Pack      pack,  
                           Material material  
                           )
```

Creates a DrawElement for each Element owned by this Shape. If an Element already has a DrawElement that uses *material* a new DrawElement will not be created.

#### Parameters:

*pack* pack used to manage created DrawElements.

*material* material to use for each DrawElement. If you pass null it will use the material on the corresponding Element. This allows you to easily setup the default (just draw as is) by passing null or setup a shadow pass by passing in a shadow material.

```
Param o3d.ParamObject.createParam ( String param_name,  
                                    String param_type_name  
                                    )
```

[inherited]

Creates a Param with the given name and type on the ParamObject. Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the Param to be created.

*param\_type\_name* The type of Param to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',

- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',

- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Shape.elements](#)

The elements owned by this shape.

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var elements = renderNode.elements;  
for (var i = 0; i < elements.length; i++) {  
    var element = elements[i];  
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [Shape](#), while modifications to the array's members **will** affect them.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.Skin Class Reference

Inherits [o3d.NamedObject](#).

[List of all members.](#)

---

### Detailed Description

A [Skin](#) holds an array of matrix indices and influences for vertices in a skin. A [Skin](#) is data only and can be used by one or more SkinEvals to implement skinning.

### Public Member Functions

- [`setVertexInfluences`](#) ( Number vertex\_index, Array [influences](#))
- Array [`getVertexInfluences`](#) ( Number vertex\_index)
- [`setInverseBindPoseMatrix`](#) ( Number index, [Matrix4](#) matrix)
- bool [`set`](#) ([RawData](#) raw\_data, size\_t offset, size\_t length)
- bool [`set`](#) ([RawData](#) raw\_data)
- bool [`isAClassName`](#) (String class\_name)

### Public Attributes

- Array [influences](#)

- Array [inverseBindPoseMatrices](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

Array o3d.Skin.getVertexInfluences ( Number *vertex\_index* )

Gets the influences for an individual vertex.

### Parameters:

*vertex\_index* the index of the vertex to get influences from.

### Returns:

Returns an Array of number pairs where the first number of each pair is the index of a matrix that influences this vertex and the second number is the amount of influence where 0 = no influence and 1 = 100% influence.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

bool o3d.Skin.set ( [RawData](#) *raw\_data* )

Deserializes from the skin data given a [RawData](#) object.

### Parameters:

*raw\_data* entire contents contains skin data

### Returns:

True if operation was successful.

```
bool o3d.Skin.set ( RawData raw_data,  
                    size_t      offset,  
                    size_t      length  
                )
```

Deserializes from the skin data given a [RawData](#) object.

**Parameters:**

*raw\_data* contains skin data  
*offset* is a byte offset from the start of *raw\_data*  
*length* is the byte length of the data to set

**Returns:**

True if operation was successful.

```
o3d.Skin.setInverseBindPoseMatrix ( Number index,  
                                    Matrix4 matrix  
                                )
```

Sets the inverse bind pose matrix for a particular joint/bone/transform.

**Parameters:**

*index* index of bone/joint/transform.  
*matrix* Inverse bind pose matrix for that joint.

```
o3d.Skin.setVertexInfluences ( Number vertex_index,  
                             Array    influences  
                           )
```

Sets the influences for an individual vertex.

**Parameters:**

*vertex\_index* The index of the vertex to set influences for  
*x*  
*influences* An array of pairs of numbers where the first number is the index of the matrix to influence the vertex and the second number is the amount of influence where 0 = no influence and 1 = 100% influence.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Array [o3d.Skin.influences](#)

An array with one element per vertex, where each element is an array of influences consisting of a number pair, where the first number is a matrix index and the second is the amount of influence.

Array [o3d.Skin.inverseBindPoseMatrices](#)

The array of inverse bone matrices

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

## o3d.SkinEval Class Reference

Inherits [o3d.VertexSource](#).

[List of all members.](#)

---

### Detailed Description

A [SkinEval](#) is a [VertexSource](#) that takes its Streams, a [ParamArray](#) of 4-by-4 matrices and a [Skin](#) and

skins the vertices in its streams storing the results in bound output streams.

## Public Member Functions

- bool [setVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index, [Field](#) field, Number start\_index)
- bool [removeVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- [Stream getVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- bool [bindStream](#) ([VertexSource](#) source, [Stream.Semantic](#) semantic, Number semantic\_index)
- bool [unbindStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Skin skin](#)
  - [ParamArray matrices](#)
  - [Matrix4 base](#)
  - [Array vertexStreams](#)
  - [Array params](#)
  - [String name](#)
  - [Id clientId](#)
  - [String className](#)
- 

## Member Function Documentation

```
bool o3d.VertexSource.bindStream ( VertexSource      source,
                                  Stream.Semantic semantic,
                                  Number          semantic_index
)                                     [inherited]
```

Bind the source stream to the corresponding stream in this [VertexSource](#).

**Parameters:**

*source* Source to get vertices from.  
*semantic* The semantic of the vertices to get  
*semantic\_index* The semantic index of the vertices to get.

**Returns:**

True if success. False if failure. If the requested semantic or semantic index do not exist on the source or this source the bind will fail.

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

**Parameters:**

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

**Parameters:**

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',

- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',

- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[Stream](#) o3d.SkinEval.getVertexStream ( [Stream.Semantic](#) *semantic*,

Number

*semantic\_index*

)

Searches the vertex streams bound to the [SkinEval](#) for one with the given stream semantic.

**Parameters:**

*semantic* The particular use of this stream.

*semantic\_index* Which index of a particular semantic to use.

**Returns:**

The found stream or null if it does not exist.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that

class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

```
bool o3d.SkinEval.removeVertexStream ( Stream.Semantic semantic,  
                                     Number      semantic_index  
                                     )
```

Removes a vertex stream from this [SkinEval](#).

**Parameters:**

*semantic* The particular use of this stream.

*semantic\_index* Which index of a particular semantic to use.

**Returns:**

true if the specified stream existed.

```
bool o3d.SkinEval.setVertexStream ( Stream.Semantic semantic,  
                                    Number      semantic_index,  
                                    Field       field,  
                                    Number      start_index  
                                    )
```

Binds a [SourceBuffer](#) field to the [SkinEval](#) and defines how the data in the field should be interpreted.

The field's buffer must be of a compatible type otherwise the binding fails and the function returns false.

**Parameters:**

*semantic* The particular use of this stream.

*semantic\_index* Which index of a particular semantic to use.

*field* The field containing information for this stream.

*start\_index* The first element to use.

**Returns:**

True if successful.

```
bool o3d.VertexSource.unbindStream ( Stream.Semantic semantic,  
                                    Number      semantic_index  
                                    )                                     [inherited]
```

Unbinds the requested stream.

**Parameters:**

*semantic* The semantic of the vertices to unbind  
*semantic\_index* The semantic index of the vertices to unbind.

**Returns:**

True if unbound. False those vertices do not exist or were not bound.

---

## Member Data Documentation

### Matrix4 [o3d.SkinEval.base](#)

The base matrix to subtract from the matrices before skinning.

### String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

### Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

### ParamArray [o3d.SkinEval.matrices](#)

The array of matrices to use for skinning.

### String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[Skin](#) [o3d.SkinEval.skin](#)

The [Skin](#) to use for skinning.

Array [o3d.SkinEval.vertexStreams](#)

A vector of the vertex streams on this [SkinEval](#).

This property is read-only.

## o3d.ParamSkin Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

### Detailed Description

A [Param](#) which stores a [Skin](#).

### Public Member Functions

- bool [bind](#) ([Param](#) source\_param)
- [unbindInput](#) ()
- [unbindOutput](#) ([Param](#) destination\_param)

- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Skin value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Skin](#) [o3d.ParamSkin.value](#)

The [Skin](#) stored by the [Param](#).

## o3d.WorldParamMatrix4 Class Reference

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

# Detailed Description

A type of [ParamMatrix4](#) that supplies the current world matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Matrix4](#) [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldInverseParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse world matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current world transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.WorldInverseTransposeParamMatrix4**

### **Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

# Detailed Description

A type of [ParamMatrix4](#) that supplies the current inverse world transpose matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Matrix4](#) [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# o3d.ViewParamMatrix4 Class Reference

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## Detailed Description

A type of [ParamMatrix4](#) that supplies the current view matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.  
The source must be compatible with this parameter.

**Parameters:**

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#) [inherited]**

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Param [o3d.Param.inputConnection](#) [inherited]**

The input connection for this param.

This property is read-only.

**String [o3d.NamedObjectBase.name](#) [inherited]**

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

**Array [o3d.Param.outputConnections](#) [inherited]**

The output connections for this param.

This property is read-only.

**bool [o3d.Param.readOnly](#) [inherited]**

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

**bool [o3d.Param.updateInput](#) [inherited]**

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## o3d.ViewInverseParamMatrix4 Class Reference

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

### Detailed Description

A type of [ParamMatrix4](#) that supplies the current inverse view matrix at render time.

### Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

### Public Attributes

- Matrix4 [value](#)
- bool [updateInput](#)
- bool [readOnly](#)
- [Param inputConnection](#)

- Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param destination\\_param](#) ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections [inherited]`

The output connections for this param.

This property is read-only.

`bool o3d.Param.readOnly [inherited]`

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.ViewTransposeParamMatrix4 Class**

### **Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

### **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current view transpose matrix at render time.

### **Public Member Functions**

- bool [bind](#) ([Param](#) source\_param)

- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - Param [inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
```

```
t.isAClassName('o3d.Shape');           // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.ViewInverseTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse view transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# o3d.ProjectionParamMatrix4 Class Reference

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## Detailed Description

A type of [ParamMatrix4](#) that supplies the current projecton matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)

- bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param )` [inherited]  
Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( )` [inherited]  
Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className` [inherited]  
The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId` [inherited]  
Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection` [inherited]  
The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name` [inherited]  
The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

`Array o3d.Param.outputConnections` [inherited]  
The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.ProjectionInverseParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

### **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse projection matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.ProjectionTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current projection transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.ProjectionInverseTransposeParamMatrix4**

### **Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

### **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse projection transpose matrix at render time.

### **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

# Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

# Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## o3d.WorldViewParamMatrix4 Class Reference

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

### Detailed Description

A type of [ParamMatrix4](#) that supplies the current world view matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldViewInverseParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse world view matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldViewTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current transpose world view matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

# Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

# Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.WorldViewInverseTransposeParamMatrix4**

## **Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

# Detailed Description

A type of [ParamMatrix4](#) that supplies the current inverse world view transpose matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Matrix4](#) [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.ViewProjectionParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current view projection matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.ViewProjectionInverseParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse view projection matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.ViewProjectionTransposeParamMatrix4**

### **Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

# Detailed Description

A type of [ParamMatrix4](#) that supplies the current view projection transpose matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Matrix4](#) [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.ViewProjectionInverseTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse view projection transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldViewProjectionParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current world view projection matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

# Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

# Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## **o3d.WorldViewProjectionInverseParamMatrix4**

## **Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

# Detailed Description

A type of [ParamMatrix4](#) that supplies the current inverse world view projection matrix at render time.

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

[Matrix4](#) [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldViewProjectionTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current world view projection transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## **Public Attributes**

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

bool o3d.Param.bind ( [Param](#) *source\_param* ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

o3d.Param.unbindInput ( ) [inherited]

Breaks any input connection coming into the [Param](#).

o3d.Param.unbindOutput ( [Param](#) *destination\_param* ) [inherited]

Breaks a specific param-bind output connection on this param.

## Parameters:

*destination\_param* param to unbind.

o3d.Param.unbindOutputs ( ) [inherited]

Breaks all param-bind output connections on this param.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Param [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

String [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

# **o3d.WorldViewProjectionInverseTransposeParamMatrix4 Class Reference**

Inherits [o3d.ParamMatrix4](#).

[List of all members.](#)

---

## **Detailed Description**

A type of [ParamMatrix4](#) that supplies the current inverse world view projection transpose matrix at render time.

## **Public Member Functions**

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

# Public Attributes

- Matrix4 [value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

# Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

## Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

## Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

## Parameters:

*class\_name* Name of class to check for.

## Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

`Param o3d.Param.inputConnection [inherited]`

The input connection for this param.

This property is read-only.

`String o3d.NamedObjectBase.name [inherited]`

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

Array [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

bool [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

bool [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

Matrix4 [o3d.ParamMatrix4.value](#) [inherited]

Sets the 4-by-4 matrix stored by the [Param](#) by a length-4 array of arrays of 4 numbers.

## o3d.State Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

### Detailed Description

A [State](#) object sets the RenderStates for a particular material or [StateSet](#).

## Public Types

- enum [Comparison](#)
- enum [Cull](#)
- enum [Fill](#)
- enum [BlendingFunction](#)
- enum [BlendingEquation](#)
- enum [StencilOperation](#)

## Public Member Functions

- [Param getStateParam](#) (String state\_name)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.State.BlendingEquation](#)

- BLEND\_ADD
- BLEND\_SUBTRACT
- BLEND\_REVERSE\_SUBTRACT
- BLEND\_MIN
- BLEND\_MAX

enum [o3d.State.BlendingFunction](#)

- BLENDFUNC\_ZERO
- BLENDFUNC\_ONE
- BLENDFUNC\_SOURCE\_COLOR
- BLENDFUNC\_INVERSE\_SOURCE\_COLOR
- BLENDFUNC\_SOURCE\_ALPHA
- BLENDFUNC\_INVERSE\_SOURCE\_ALPHA
- BLENDFUNC\_DESTINATION\_ALPHA
- BLENDFUNC\_INVERSE\_DESTINATION\_ALPHA
- BLENDFUNC\_DESTINATION\_COLOR
- BLENDFUNC\_INVERSE\_DESTINATION\_COLOR
- BLENDFUNC\_SOURCE\_ALPHA\_SATURATE

enum [o3d.State.Comparison](#)

- CMP\_NEVER (Never)
- CMP\_LESS (Less Than)
- CMP\_EQUAL (Equal To)
- CMP\_LEQUAL (Less Than or Equal To)
- CMP\_GREATER (Greater Than)
- CMP\_NOTEQUAL (Not Equal To)
- CMP\_GEQUAL (Greater Than or Equal To)
- CMP\_ALWAYS (Always)

enum [o3d.State.Cull](#)

- CULL\_NONE Don't Cull by face
- CULL\_CW Cull clock-wise faces
- CULL\_CCW Cull counter clock-wise faces

enum [o3d.State.Fill](#)

- POINT
- WIREFRAME
- SOLID

enum [o3d.State.StencilOperation](#)

- STENCIL\_KEEP
- STENCIL\_ZERO

- STENCIL\_REPLACE
  - STENCIL\_INCREMENT\_SATURATE
  - STENCIL\_DECREMENT\_SATURATE
  - STENCIL\_INVERT
  - STENCIL\_INCREMENT
  - STENCIL\_DECREMENT
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',

- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',

- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[Param](#) o3d.State.getStateParam ( String *state\_name* )

Returns a [Param](#) for a given state. If the param does not already exist it will be created. If the *state\_name* is invalid it will return null.

**Parameters:**

*state\_name* name of the state

**Returns:**

param or null if no matching state.

**Example:**

```
// Gets the client.
g_o3d = document.o3d.o3d;
...
// Creates a state object.
var state = my_pack.createState("my_state");

// Sets some states.
state.getStateParam('o3d.StencilEnable').value = true;
state.getStateParam('o3d.StencilReference').value = 25;
state.getStateParam('o3d.StencilPassOperation').value =
g_o3d.State.STENCIL_REPLACE;
state.getStateParam('o3d.StencilComparisonFunction').value =
g_o3d.State.CMP_ALWAYS;
state.getStateParam('o3d.ZEnable').value = false;
```

```

state.getStateParam('o3d.ZWriteEnable').value = false;
state.getStateParam('o3d.ColorWriteEnable').value = 0;

```

Valid states:

State Name	Type	Default Value
o3d.AlphaTestEnable	<a href="#">ParamBoolean</a>	default = false
o3d.AlphaReference	<a href="#">ParamFloat</a> 0-1	default = 0
o3d.AlphaComparisonFunction	<a href="#">ParamInteger</a> , <a href="#">State.Comparison</a>	default = State.CMP_ALWAYS
o3d.CullMode	<a href="#">ParamInteger</a> , <a href="#">State.Cull</a>	default = State.CULL_CW
o3d.DitherEnable	<a href="#">ParamBoolean</a>	default = false
o3d.LineSmoothEnable	<a href="#">ParamBoolean</a>	default = false
o3d.PointSpriteEnable	<a href="#">ParamBoolean</a>	default = false
o3d.PointSize	<a href="#">ParamFloat</a>	TBD
o3d.PolygonOffset1	<a href="#">ParamFloat</a>	TBD
o3d.PolygonOffset2	<a href="#">ParamFloat</a>	TBD
o3d.FillMode	<a href="#">ParamInteger</a> , <a href="#">State.Fill</a>	default = State.SOLID
o3d.ZEnable	<a href="#">ParamBoolean</a>	default = true
o3d.ZWriteEnable	<a href="#">ParamBoolean</a>	default = true
o3d.ZComparisonFunction	<a href="#">ParamInteger</a> , <a href="#">State.Comparison</a>	default = State.CMP_ALWAYS
o3d.AlphaBlendEnable	<a href="#">ParamBoolean</a>	default = false
o3d.SourceBlendFunction	<a href="#">ParamInteger</a> , <a href="#">State.BlendingFunction</a>	default = State.BLENDFUNC_ONE
o3d.DestinationBlendFunction	<a href="#">ParamInteger</a> , <a href="#">State.BlendingFunction</a>	default = State.BLENDFUNC_ZERO
o3d.StencilEnable	<a href="#">ParamBoolean</a>	default = false
o3d.StencilFailOperation	<a href="#">ParamInteger</a> , <a href="#">State.StencilOperation</a>	default = State.STENCIL_KEEP
o3d.StencilZFailOperation	<a href="#">ParamInteger</a> , <a href="#">State.StencilOperation</a>	default = State.STENCIL_KEEP
o3d.StencilPassOperation	<a href="#">ParamInteger</a> ,	default =

	<a href="#">State.StencilOperation</a>	State.STENCIL_KEEP
o3d.StencilComparisonFunction	<a href="#">ParamInteger</a> , <a href="#">State.Comparison</a>	default = State.CMP_ALWAYS
o3d.StencilReference	<a href="#">ParamInteger</a> 0-255	default = 0
o3d.StencilMask	<a href="#">ParamInteger</a> 0-255	default = 255
o3d.StencilWriteMask	<a href="#">ParamInteger</a> 0-255	default = 255
o3d.ColorWriteEnable	<a href="#">ParamInteger</a> 0-15 bit 0 = red, bit 1 = green, bit 2 = blue, bit 3 = alpha	default = 15
o3d.BlendEquation	<a href="#">ParamInteger</a> , <a href="#">State.BlendingEquation</a>	default = State.BLEND_ADD
o3d.TwoSidedStencilEnable	<a href="#">ParamBoolean</a>	default = false
o3d.CCWStencilFailOperation	<a href="#">ParamInteger</a> , <a href="#">State.StencilOperation</a>	default = State.STENCIL_KEEP
o3d.CCWStencilZFailOperation	<a href="#">ParamInteger</a> , <a href="#">State.StencilOperation</a>	default = State.STENCIL_KEEP
o3d.CCWStencilPassOperation	<a href="#">ParamInteger</a> , <a href="#">State.StencilOperation</a>	default = State.STENCIL_KEEP
o3d.CCWStencilComparisonFunction	<a href="#">ParamInteger</a> , <a href="#">State.Comparison</a>	default = State.CMP_ALWAYS
o3d.SeparateAlphaBlendEnable	<a href="#">ParamBoolean</a>	default = false;
o3d.SourceBlendAlphaFunction	<a href="#">ParamInteger</a> , <a href="#">State.BlendingFunction</a>	default = State.BLENDFUNC_ONE
o3d.DestinationBlendAlphaFunction	<a href="#">ParamInteger</a> , <a href="#">State.BlendingFunction</a>	default = State.BLENDFUNC_ZERO
o3d.BlendAlphaEquation	<a href="#">ParamInteger</a> , <a href="#">State.BlendingEquation</a>	default = State.BLEND_ADD

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
```

```
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.StateSet Class Reference

Inherits [o3d.RenderNode](#).

[List of all members](#).

---

### Detailed Description

A [StateSet](#) is a render node that sets render states of all of its children. You can make this a parent of a part of the render graph to set render states in a more global way.

### Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)

- Array [getRenderNodesByClassNameInTree](#) (String class\_name)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [State state](#)
  - Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

`source_param_object` param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',

- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array

( String  $\frac{class\_nam}{e}$  ) [inherited]

o3d.RenderNode.getRenderNodesByClassNameInTree

Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* ) [inherited]  
Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]  
Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode](#) [o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under `parent_node`. Setting `parent_node` to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

**Number** [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

[State](#) [o3d.StateSet.state](#)

The [State](#) for this [StateSet](#).

# o3d.Stream Class Reference

Inherits [o3d.ObjectBase](#).

[List of all members.](#)

---

## Detailed Description

A stream object defines how [Buffer](#) data is interpreted by an [Effect](#) when rendering a [Primitive](#). It contains a pointer to a [Field](#), a semantic and semantic index that determine how the data is accessed.

## Public Types

- enum [Semantic](#)

## Public Member Functions

- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Field field](#)
  - [Semantic semantic](#)
  - Number [semanticIndex](#)
  - Number [startIndex](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Stream.Semantic](#)

- UNKNOWN\_SEMANTIC = 0,

- POSITION,
- NORMAL,
- TANGENT,
- BINORMAL,
- COLOR,
- TEXCOORD

Semantics used when binding buffers to the streambank. They determine how the [Stream](#) links up to the shader inputs.

---

## Member Function Documentation

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

### Parameters:

*class\_name* Name of class to check for.

### Returns:

true if this object is a or is derived from the given class name.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Field o3d.Stream.field](#)

The associated [Field](#).

This property is read-only.

[Semantic o3d.Stream.semantic](#)

The semantic specified for the [Stream](#).

This property is read-only.

Number [o3d.Stream.semanticIndex](#)

The semantic index specified for the [Stream](#) (eg., TEXCOORD1 = 1, BINORMAL7 = 7, etc).

This property is read-only.

Number [o3d.Stream.startIndex](#)

The start index for the [Stream](#).

This property is read-only.

## **o3d.ParamVertexBufferStream Class Reference**

Inherits [o3d.Param](#).

[List of all members.](#)

---

### **Detailed Description**

A [Param](#) which stores a [Stream](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Stream stream](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.

The source must be compatible with this parameter.

### Parameters:

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

### Returns:

True if the bind succeeded.

bool o3d.ObjectBase.isAClassName ( String [class\\_name](#) ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

#### Parameters:

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

`String o3d.ObjectBase.className [inherited]`

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

`Id o3d.ObjectBase.clientId [inherited]`

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Param](#) [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

[String](#) [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

[Array](#) [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

[bool](#) [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

[Stream](#) [o3d.ParamVertexBufferStream.stream](#)

The [Stream](#) stored by the [Param](#).

This property is read-only.

[bool](#) [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

# o3d.StreamBank Class Reference

Inherits [o3d.NamedObject](#).

[List of all members.](#)

---

## Detailed Description

The [StreamBank](#) a collection of streams that hold vertices.

## Public Member Functions

- bool [setVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index, [Field](#) field, Number start\_index)
- [Stream getVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- bool [removeVertexStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- bool [bindStream](#) ([VertexSource](#) source, [Stream.Semantic](#) semantic, Number semantic\_index)
- bool [unbindStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Array [vertexStreams](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

```
bool o3d.StreamBank.bindStream ( VertexSource source,
                                 Stream.Semantic semantic,
                                 Number semantic_index
```

)

Binds the source stream to the corresponding stream in this [VertexSource](#).

**Parameters:**

*source* Source to get vertices from.  
*semantic* The semantic of the vertices to get  
*semantic\_index* The semantic index of the vertices to get.

**Returns:**

True if success. False if failure. If the requested semantic or semantic index do not exist on the source or this source the bind will fail.

[Stream](#) o3d.StreamBank.getVertexStream ( [Stream.Semantic](#) *semantic*,  
Number *semantic\_index*  
)

Searches the vertex streams bound to the [StreamBank](#) for one with the given stream semantic. If a stream is not found then it returns null.

**Parameters:**

*semantic* The particular use of this stream.  
*semantic\_index* Which index of a particular semantic to use.

**Returns:**

The found stream or null if it does not exist.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.StreamBank.removeVertexStream ( [Stream.Semantic](#) *semantic*,  
Number *semantic\_index*

)

Removes a vertex stream from this [StreamBank](#).

**Parameters:**

*semantic* The particular use of this stream.  
*semantic\_index* Which index of a particular semantic to use.

**Returns:**

true if the specified stream existed.

```
bool o3d.StreamBank.setVertexStream ( Stream.Semantic semantic,
                                      Number      semantic_index,
                                      Field       field,
                                      Number      start_index
                                    )
```

Binds a [VertexBuffer](#) field to the [StreamBank](#) and defines how the data in the buffer should be interpreted. The field's buffer must be of a compatible type otherwise the binding fails and the function returns false.

**Parameters:**

*semantic* The particular use of this stream.  
*semantic\_index* Which index of a particular semantic to use.  
*field* The field containing information for this stream.  
*start\_index* The first element to use.

**Returns:**

True if successful.

```
bool o3d.StreamBank.unbindStream ( Stream.Semantic semantic,
                                   Number      semantic_index
                                 )
```

Unbinds the requested stream.

**Parameters:**

*semantic* The semantic of the vertices to unbind  
*semantic\_index* The semantic index of the vertices to unbind.

**Returns:**

True if unbound. False those vertices do not exist or were not bound.

---

# Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), `Pack.getObject`, [RenderNode.getRenderNodesByNameInTree](#) and `RenderNode.getTransformsByNameInTree` which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.StreamBank.vertexStreams](#)

An array of the vertex streams on this [StreamBank](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var streams = streamBank.vertexStreams;  
for (var i = 0; i < streams.length; i++) {  
    var stream = streams[i];  
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [StreamBank](#), while modifications to the members of the array **will** affect them.

This property is read-only.

# o3d.ParamStreamBank Class Reference

Inherits [o3d.Param](#).

[List of all members.](#)

---

## Detailed Description

A [Param](#) which stores a [StreamBank](#).

## Public Member Functions

- bool [bind \(Param source\\_param\)](#)
- [unbindInput \(\)](#)
- [unbindOutput \(Param destination\\_param\)](#)
- [unbindOutputs \(\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [StreamBank value](#)
  - bool [updateInput](#)
  - bool [readOnly](#)
  - [Param inputConnection](#)
  - Array [outputConnections](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

bool o3d.Param.bind ( [Param source\\_param](#) ) [inherited]

Directly binds two [Param](#) elements such that this parameter gets its value from the source parameter.  
The source must be compatible with this parameter.

**Parameters:**

*source para* The parameter that the value originates from. Passing in null will unbind any  
*m* parameter currently bound.

**Returns:**

True if the bind succeeded.

`bool o3d.ObjectBase.isAClassName ( String class_name ) [inherited]`

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

`o3d.Param.unbindInput ( ) [inherited]`

Breaks any input connection coming into the [Param](#).

`o3d.Param.unbindOutput ( Param destination_param ) [inherited]`

Breaks a specific param-bind output connection on this param.

**Parameters:**

*destination\_param* param to unbind.

`o3d.Param.unbindOutputs ( ) [inherited]`

Breaks all param-bind output connections on this param.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Param** [o3d.Param.inputConnection](#) [inherited]

The input connection for this param.

This property is read-only.

**String** [o3d.NamedObjectBase.name](#) [inherited]

The object's name.

This property is read-only.

Reimplemented in [o3d.NamedObject](#).

**Array** [o3d.Param.outputConnections](#) [inherited]

The output connections for this param.

This property is read-only.

**bool** [o3d.Param.readOnly](#) [inherited]

If true the param is read only. Its value can not be set nor can it be used as the destination in a ParamBind

This property is read-only.

**bool** [o3d.Param.updateInput](#) [inherited]

If true, this param will make sure its input param is up to date when using it as a source. Default = true.

This is for helping with [Param](#) cycles.

If paramA gets its value from paramB and paramB gets its value from paramA:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramA will evaluate then copy to paramB.

If you set paramB.updateInput = false, then:

If you go paramA.value, paramB will evaluate then copy to paramA.

If you go paramB.value, paramB just copy paramA. paramA will NOT evaluate when paramB asks for its value.

#### [StreamBank](#) [o3d.ParamStreamBank.value](#)

The [StreamBank](#) stored by the [Param](#).

## o3d.Texture Class Reference

Inherits [o3d.ParamObject](#).

Inherited by [o3d.Texture2D](#), and [o3d.TextureCUBE](#).

[List of all members.](#)

---

## Detailed Description

The [Texture](#) class is a base class for image data used in texture mapping.

## Public Types

- enum [Format](#)

## Public Member Functions

- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Format format](#)
  - Number [levels](#)
  - bool [alphaIsOne](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Texture.Format](#)

- UNKNOWN\_FORMAT
- XRGB8
- ARGB8
- ABGR16F
- R32F
- ABGR32F
- DXT1
- DXT3
- DXT5

The in-memory format of the texture bitmap.

NOTE: The R32F format is different on GL vs D3D. If you use it in a shader you must only use the red channel. The green, blue and alpha channels are undefined.

For example:

...

```
// The texture sampler is used to access the texture bitmap in the fragment  
// shader.  
sampler texSampler0;
```

```

...
// input parameters for our vertex shader
struct PixelShaderInput {
    float4 position : POSITION;
    float2 texcoord : TEXCOORD0; // Texture coordinates
};

float4 pixelShaderFunction(PixelShaderInput input): COLOR {
    // ** Use only valid channels. ** -----
    // |  

    // V
    return tex2D(texSampler0, input.texcoord).rrrr;
}

```

---

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name ) [inherited]`

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',

- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',

- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]  
Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

bool [o3d.Texture.alphaIsOne](#)

True of all the alpha values in the texture are 1.0

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Format o3d.Texture.format](#)

The memory format used for storing the bitmap associated with the texture object.

This property is read-only.

Number [o3d.Texture.levels](#)

The number of mipmap levels used by the texture.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.Texture2D Class Reference

Inherits [o3d.Texture](#).

[List of all members](#).

---

### Detailed Description

A class for 2D textures that defines the interface for getting the dimensions of the texture, its memory format and number of mipmap levels.

### Public Types

- enum [Format](#)

## Public Member Functions

- [RenderSurface getRenderSurface](#) (Number mip\_level, [Pack](#) pack)
- [set](#) (Number level, Array values)
- [setRect](#) (Number level, Number destination\_x, Number destination\_y, Number source\_width, Array values)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [width](#)
  - Number [height](#)
  - [Format format](#)
  - Number [levels](#)
  - bool [alphaIsOne](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.Texture.Format](#) [inherited]

- UNKNOWN\_FORMAT
- XRGB8
- ARGB8
- ABGR16F
- R32F

- ABGR32F
- DXT1
- DXT3
- DXT5

The in-memory format of the texture bitmap.

NOTE: The R32F format is different on GL vs D3D. If you use it in a shader you must only use the red channel. The green, blue and alpha channels are undefined.

For example:

```
...
// The texture sampler is used to access the texture bitmap in the fragment
// shader.
sampler texSampler0;

...
// input parameters for our vertex shader
struct PixelShaderInput {
    float4 position : POSITION;
    float2 texcoord : TEXCOORD0; // Texture coordinates
};

float4 pixelShaderFunction(PixelShaderInput input): COLOR {
    // ** Use only valid channels. ** -----+
    //                                         |
    //                                         v
    return tex2D(texSampler0, input.texcoord).rrrr;
}
```

---

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given *source\_param\_object* to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

Param o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
)

[inherited]

Creates a Param with the given name and type on the ParamObject. Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the Param to be created.

*param\_type\_name* The type of Param to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',

- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
 Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[RenderSurface](#) o3d.Texture2D.getRenderSurface ( Number *mip\_level*,  
                                    Pack      *pack*  
                                    )

Returns a [RenderSurface](#) object associated with a *mip\_level* of a texture.

**Parameters:**

*mip\_level* The mip-level of the surface to be returned.  
*pack*       The pack in which the surface will reside.

**Returns:**

The [RenderSurface](#) object.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');  // true  
t.isAClassName('o3d.Shape');        // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

o3d.Texture2D.set ( Number *level*,  
                            Array      *values*  
                            )

Sets the values of the data stored in the texture.

It is not recommend that you call this for large textures but it is useful for making simple ramps or

noise textures for shaders.

NOTE: the number of values must equal the size of the texture \* the number of elements. In other words, for a XRGB8 texture there must be width \* height \* 3 values. For an ARGB8, ABGR16F or ABGR32F texture there must be width \* height \* 4 values. For an R32F texture there must be width \* height values.

NOTE: the order of channels is R G B for XRGB8 textures and R G B A for ARGB8, ABGR16F and ABGR32F textures so for example for XRGB8 textures

[1, 0, 0] = a red pixel

[0, 0, 1] = a blue pixel

For ARGB8, ABGR16F, ABGR32F textures

[1, 0, 0, 0] = a red pixel with zero alpha

[1, 0, 0, 1] = a red pixel with one alpha

[0, 0, 1, 1] = a blue pixel with one alpha

#### Parameters:

*level* the mip level to update.

*values* Values to be stored in the buffer.

```
o3d.Texture2D.setRect ( Number level,  
                        Number destination_x,  
                        Number destination_y,  
                        Number source_width,  
                        Array values  
                    )
```

Sets a rectangular area of values in a texture.

Does clipping. In other words if you pass in a 10x10 pixel array and give it destination of (-5, -5) it will only use the bottom 5x5 pixels of the array you passed in to set the top 5x5 pixels of the texture.

See [o3d.Texture2D.set](#) for details on formats.

#### Parameters:

*level* the mip level to update.

*destination\_x* The x coordinate of the area in the texture to affect.

*destination\_y* The y coordinate of the area in the texture to affect.

*source\_width* The width of the area to effect. The height is determined by the size of the array

passed in.  
*values* Values to be stored in the buffer.

See also:

[o3d.Texture2D.set](#)

---

## Member Data Documentation

bool [o3d.Texture.alphaIsOne](#) [inherited]

True of all the alpha values in the texture are 1.0

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

[Format o3d.Texture.format](#) [inherited]

The memory format used for storing the bitmap associated with the texture object.

This property is read-only.

Number [o3d.Texture2D.height](#)

The height of the texture, in texels.

This property is read-only.

Number [o3d.Texture.levels](#) [inherited]

The number of mipmap levels used by the texture.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

Number [o3d.Texture2D.width](#)

The width of the texture, in texels.

This property is read-only.

## o3d.TextureCUBE Class Reference

Inherits [o3d.Texture](#).

[List of all members](#).

---

### Detailed Description

[TextureCUBE](#) is a class for textures used for cube mapping. A cube texture stores bitmaps for the 6 faces of a cube and is addressed via three texture coordinates.

## Public Types

- enum [CubeFace](#)
- enum [Format](#)

## Public Member Functions

- [RenderSurface getRenderSurface](#) ([CubeFace](#) face, Number mip\_level, [Pack](#) pack)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Number [edgeLength](#)
  - [Format format](#)
  - Number [levels](#)
  - bool [alphaIsOne](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Enumeration Documentation

enum [o3d.TextureCUBE.CubeFace](#)

- FACE\_POSITIVE\_X
- FACE\_NEGATIVE\_X
- FACE\_POSITIVE\_Y
- FACE\_NEGATIVE\_Y
- FACE\_POSITIVE\_Z

- FACE\_NEGATIVE\_Z

The names of each of the six faces of a cube map texture.

```
enum o3d.Texture.Format [inherited]
```

- UNKNOWN\_FORMAT
- XRGB8
- ARGB8
- ABGR16F
- R32F
- ABGR32F
- DXT1
- DXT3
- DXT5

The in-memory format of the texture bitmap.

NOTE: The R32F format is different on GL vs D3D. If you use it in a shader you must only use the red channel. The green, blue and alpha channels are undefined.

For example:

```
...
// The texture sampler is used to access the texture bitmap in the fragment
// shader.
sampler texSampler0;

...
// input parameters for our vertex shader
struct PixelShaderInput {
    float4 position : POSITION;
    float2 texcoord : TEXCOORD0; // Texture coordinates
};

float4 pixelShaderFunction(PixelShaderInput input): COLOR {
    // ** Use only valid channels. ** -----
    // | 
    // V
```

```
    return tex2D(texSampler0, input.texcoord).rrrr;  
}
```

---

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
 String param_type_name  
 ) [inherited]`

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',

- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

[RenderSurface](#) o3d.TextureCUBE.getRenderSurface ( [CubeFace](#) *face*,

Number *mip\_level*,

[Pack](#) *pack*

)

Returns a [RenderSurface](#) object associated with a given cube face and *mip\_level* of a texture.

**Parameters:**

*face* The cube face from which to extract the surface.

*mip\_level* The mip-level of the surface to be returned.

*pack* The pack in which the surface will reside.

**Returns:**

The [RenderSurface](#) object.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

---

## Member Data Documentation

bool [o3d.Texture.alphaIsOne](#) [inherited]

True of all the alpha values in the texture are 1.0

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

Id [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

Number [o3d.TextureCUBE.edgeLength](#)

The length of each edge of the cube, in texels.

This property is read-only.

[Format o3d.Texture.format](#) [inherited]

The memory format used for storing the bitmap associated with the texture object.

This property is read-only.

Number [o3d.Texture.levels](#) [inherited]

The number of mipmap levels used by the texture.

This property is read-only.

String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.TickEvent Class Reference

[List of all members](#).

---

### Detailed Description

This class is used to pass information to a registered ontick callback.

## Public Attributes

- Number [elapsedTime](#)
- 

## Member Data Documentation

Number [o3d.TickEvent.elapsedTime](#)

Use this property to get the elapsed time since the last tick event in seconds.

This property is read-only.

## o3d.Transform Class Reference

Inherits [o3d.ParamObject](#).

[List of all members.](#)

---

## Detailed Description

The [Transform](#) defines parent child relationship and a localMatrix.. A [Transform](#) can have one or no parents and an arbitrary number of children.

## Public Member Functions

- Array [getTransformsInTree \(\)](#)
- Array [getTransformsByNameInTree \(String name\)](#)
- Matrix4 [getUpdatedWorldMatrix \(\)](#)
- [addShape \(Shape shape\)](#)
- bool [removeShape \(Shape shape\)](#)
- [createDrawElements \(Pack pack, Material material\)](#)
- [identity \(\)](#)
- [translate \(Vector3 translation\)](#)
- [translate \(Number x, Number y, Number z\)](#)

- [rotateX](#) (Number radians)
- [rotateY](#) (Number radians)
- [rotateZ](#) (Number radians)
- [rotateZYX](#) ([Vector3](#) radiansXYZ)
- [axisRotate](#) ([Vector3](#) axis, Number radians)
- [quaternionRotate](#) ([Quat](#) unit\_quat)
- [scale](#) ([Vector3](#) scale)
- [scale](#) (Number x, Number y, Number z)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- bool [visible](#)
  - [Transform parent](#)
  - Array [children](#)
  - Array [shapes](#)
  - Matrix4 [worldMatrix](#)
  - Matrix4 [localMatrix](#)
  - bool [cull](#)
  - [BoundingBox boundingBox](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.Transform.addShape ( Shape shape )`

Adds a shape do this transform.

#### Parameters:

*shape* [Shape](#) to add.

```
o3d.Transform.axisRotate ( Vector3 axis,  
                           Number radians  
                         )
```

Pre-composes the local matrix of this [Transform](#) with a rotation around the given axis. For example, if the local matrix is a translation, the new local matrix will rotate around the given axis and then translate.

#### Parameters:

*radians* The number of radians to rotate.

*axis* a non-zero vector representing the axis around which to rotate.

```
o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]
```

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

#### Parameters:

*source\_param\_object* param object to copy params from.

```
o3d.Transform.createDrawElements ( Pack pack,  
                                 Material material  
                               )
```

Walks the tree of transforms starting with this transform and creates draw elements. If an [Element](#) already has a [DrawElement](#) that uses material a new [DrawElement](#) will not be created.

#### Parameters:

*pack* [Pack](#) used to manage created elements.

*material* [Material](#) to use for each element. If you pass null, it will use the material on the element to which a draw element is being added. In other words, a [DrawElement](#) will use the material of the corresponding [Element](#) if material is null. This allows you to easily setup the default (just draw as is) by passing null or setup a shadow pass by passing in a shadow material.

```
Param o3d.ParamObject.createParam ( String param_name,  
                                         String param_type_name  
                                       ) [inherited]
```

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',

- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

## Returns:

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

## Parameters:

*param\_name* Name to search for.

## Returns:

The [Param](#) with the given name, or null otherwise.

Array o3d.Transform.getTransformsByNameInTree ( String *name* )

Searches for transforms that match the given name in the hierarchy under and including this transform.

Since there can be more than one transform with a given name, results are returned in an array.

Note that modifications to this array [e.g. additions to it] will not affect the underlying [Transform](#), while modifications to the members of the array **will** affect them.

**Parameters:**

*name* [Transform](#) name to look for.

**Returns:**

An array containing the transforms of the under and including this transform matching the given name.

Array o3d.Transform.getTransformsInTree ( )

Returns all the transforms under this transform including this one.

Note that modifications to this array [e.g. additions to it] will not affect the underlying [Transform](#), while modifications to the members of the array **will** affect them.

**Returns:**

An array containing the transforms of the subtree.

Matrix4 o3d.Transform.getUpdatedWorldMatrix ( )

Evaluates and returns the current world matrix.

**Returns:**

The updated world matrix.

o3d.Transform.identity ( )

Sets the local matrix of this transform to the identity matrix.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.Transform.quaternionRotate ( [Quat](#) *unit\_quat* )

Pre-composes the local matrix of this [Transform](#) with a rotation defined by the given quaternion.

**Parameters:**

*unit\_quat* A non-zero quaternion to be interpreted as a rotation.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

bool o3d.Transform.removeShape ( [Shape](#) *shape* )

Removes a shape from this transform.

**Parameters:**

*shape* [Shape](#) to remove.

**Returns:**

true if successful, false if shape was not in this transform.

o3d.Transform.rotateX ( Number *radians* )

Pre-composes the local matrix of this [Transform](#) with a rotation about the x-axis. For example, if the local matrix is a translation, the new local matrix will rotate around the x-axis and then translate.

**Parameters:**

*radians* The number of radians to rotate around x-axis.

o3d.Transform.rotateY ( Number *radians* )

Pre-composes the local matrix of this [Transform](#) with a rotation about the y-axis. For example, if the local matrix is a translation, the new local matrix will rotate around the y-axis and then translate.

**Parameters:**

*radians* The number of radians to rotate around y-axis.

o3d.Transform.rotateZ ( Number *radians* )

Pre-composes the local matrix of this [Transform](#) with a rotation about the z-axis. For example, if the local matrix is a translation, the new local matrix will rotate around the z-axis and then translate.

**Parameters:**

*radians* The number of radians to rotate around z-axis.

`o3d.Transform.rotateZYX ( Vector3 radiansXYZ )`

Pre-composes the local matrix of this [Transform](#) with a rotation. Interprets the three entries of the given vector as angles by which to rotate around the x, y and z axes. Rotates around the x-axis first, then the y-axis, then the z-axis.

**Parameters:**

*radiansXYZ* A vector of angles (in radians) by which to rotate around the x, y and z axes.

`o3d.Transform.scale ( Number x,  
Number y,  
Number z  
)`

Pre-composes the local matrix of this transform by a scaling transformation. For example, if the local matrix is a rotation, the new local matrix will scale and then rotate.

**Parameters:**

*x* amount to scale in the x dimension.  
*y* amount to scale in the y dimension.  
*z* amount to scale in the z dimension.

`o3d.Transform.scale ( Vector3 scale )`

Pre-composes the local matrix of this transform by a scaling transformation. For example, if the local matrix is a rotation, the new local matrix will scale and then rotate.

**Parameters:**

*scale* amount to scale.

`o3d.Transform.translate ( Number x,  
Number y,  
Number z  
)`

Pre-composes the local matrix of this [Transform](#) with a translation. For example, if the local matrix is a rotation then the new local matrix will translate by the given amounts and then rotate.

**Parameters:**

*x* amount to translate in x.  
*y* amount to translate in y.  
*z* amount to translate in z.

`o3d.Transform.translate ( Vector3 translation )`

Pre-composes the local matrix of this [Transform](#) with a translation. For example, if the local matrix is a rotation then new local matrix will translate by the given vector and then rotated.

## Parameters:

*translation* vector of 3 entries by which to translate.

---

# Member Data Documentation

## [BoundingBox](#) [o3d.Transform.boundingBox](#)

The [BoundingBox](#) for this [Transform](#). If culling is on this bounding box will be tested against the view frustum of any draw context used to with this [Transform](#).

## [Array](#) [o3d.Transform.children](#)

The immediate children of this [Transform](#).

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var children = transform.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. additions to it] will not affect the underlying [Transform](#), while modifications to the members of the array **will** affect them.

This property is read-only.

## [String](#) [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

## [Id](#) [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

## [bool](#) [o3d.Transform.cull](#)

The cull setting for this transform. If true this [Transform](#) will be culled by having its bounding box compared to the view frustum of any draw context it is used with.

#### Matrix4 [o3d.Transform.localMatrix](#)

Local transformation matrix.

#### String [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

#### Array [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

#### [Transform](#) [o3d.Transform.parent](#)

Sets the parent of the transform by re-parenting the transform under parent. Setting parent to null removes the transform and the entire subtree below it from the transform graph. If the operation would create a cycle it fails.

This property is write-only.

#### Array [o3d.Transform.shapes](#)

Gets the shapes owned by this transform.

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var shapes = transform.shapes;
```

```
for (var i = 0; i < shapes.length; i++) {  
    var shape = shapes[i];  
}
```

Note that modifications to this array [e.g. additions to it] will not affect the underlying [Transform](#), while modifications to the members of the array **will** affect them.

bool [o3d.Transform.visible](#)

The Visibility for this transform.

Matrix4 [o3d.Transform.worldMatrix](#)

World (model) matrix as it was last computed.

This property is read-only.

## o3d.TreeTraversal Class Reference

Inherits [o3d.RenderNode](#).

[List of all members.](#)

---

### Detailed Description

A [TreeTraversal](#) has multiple DrawLists registered with it. Each [DrawList](#) has a [DrawContext](#) registered with it. At render time the [TreeTraversal](#) walks the transform graph from the transform it's pointing at and for each [DrawElement](#) it finds who's material matches one of its registered DrawLists it adds that [DrawElement](#) to that [DrawList](#).

### Public Member Functions

- [registerDrawList](#) ([DrawList](#) draw\_list, [DrawContext](#) draw\_context, bool reset)
- bool [unregisterDrawList](#) ([DrawList](#) draw\_list)
- Array [getRenderNodesInTree](#) ()
- Array [getRenderNodesByNameInTree](#) (String [name](#))
- Array [getRenderNodesByClassNameInTree](#) (String class\_name)
- [Param](#) [createParam](#) (String param\_name, String param\_type\_name)

- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- [Transform transform](#)
  - Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object )` [inherited]  
Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,`  
`String param_type_name`  
`)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

### Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',

- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]  
Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array  
o3d.RenderNode.getRenderNodesByClassNameInTree ( String  $\underset{e}{\text{class\_nam}}$  ) [inherited]  
Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* ) [inherited]  
Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]  
Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]  
Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');  
t.isAClassName('o3d.Transform');      // true  
t.isAClassName('o3d.ParamObject');   // true  
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

o3d.TreeTraversal.registerDrawList ( [DrawList](#) *draw\_list*,  
[DrawContext](#) *draw\_context*,  
bool *reset*

)

Registers a [DrawList](#) with this [TreeTraversal](#) so that when this [TreeTraversal](#) traverses its tree, DrawElements using materials that use this [DrawList](#) will be added though possibly culled by the view frustum of the [DrawContext](#). Note: passing in the same [DrawList](#) more than once will override the previous settings for that [DrawList](#).

**Parameters:**

*draw\_list* [DrawList](#) to register.

*draw\_context* [DrawContext](#) to use with the [DrawList](#).

*reset* true if you want the [DrawList](#) reset before traversing.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

bool o3d.TreeTraversal.unregisterDrawList ( [DrawList](#) *draw\_list* )

Unregisters a [DrawList](#) with this [TreeTraversal](#).

**Parameters:**

*draw\_list* [DrawList](#) to unregister.

**Returns:**

true if unregistered. false if this draw\_list was not registered.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

**String** [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions [Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and [RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
```

```
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under parent\_node. Setting parent\_node to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

Number [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

[Transform o3d.TreeTraversal.transform](#)

The root [Transform](#) this [TreeTraversal](#) will start traversing from.

## o3d.Float2 Class Reference

[List of all members.](#)

---

### Detailed Description

A data type consisting of 2 numbers. A [Float2](#) is represented in JavaScript by an array containing 2 numbers: [x, y].

## o3d.Float3 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type consisting of 3 numbers. A [Float3](#) is represented in JavaScript by an array containing 3 numbers: [x, y, z].

## o3d.Float4 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type consisting of 4 numbers. A [Float4](#) is represented in JavaScript by an array containing 4 numbers: [x, y, z, w].

## o3d.VertexSource Class Reference

Inherits [o3d.ParamObject](#).

Inherited by [o3d.SkinEval](#).

[List of all members.](#)

---

## Detailed Description

A [VertexSource](#) is an object that allows binding Streams such that the [VertexSource](#) updates the Buffers of the Streams that have been bound to it. An example of a [VertexSource](#) object is a [SkinEval](#)

## Public Member Functions

- bool [bindStream](#) ([VertexSource](#) source, [Stream.Semantic](#) semantic, Number semantic\_index)

- bool [unbindStream](#) ([Stream.Semantic](#) semantic, Number semantic\_index)
- [Param createParam](#) (String param\_name, String param\_type\_name)
- [Param getParam](#) (String param\_name)
- bool [removeParam](#) ([Param](#) param)
- [copyParams](#) ([ParamObject](#) source\_param\_object)
- bool [isAClassName](#) (String class\_name)

## Public Attributes

- Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
- 

## Member Function Documentation

```
bool o3d.VertexSource.bindStream ( VertexSource source,
                                  Stream.Semantic semantic,
                                  Number semantic_index
                                )
```

Bind the source stream to the corresponding stream in this [VertexSource](#).

### Parameters:

*source* Source to get vertices from.  
*semantic* The semantic of the vertices to get  
*semantic\_index* The semantic index of the vertices to get.

### Returns:

True if success. False if failure. If the requested semantic or semantic index do not exist on the source or this source the bind will fail.

[o3d.ParamObject.copyParams](#) ( [ParamObject](#) source\_param\_object ) [inherited]

Copies all the params from a the given source\_param\_object to this param object. Does not replace any currently existing params with the same name.

### Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) o3d.ParamObject.createParam ( String *param\_name*,  
String *param\_type\_name*  
) [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

#### Parameters:

*param\_name* The name of the [Param](#) to be created.  
*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',
- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',

- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

**Parameters:**

*class\_name* Name of class to check for.

**Returns:**

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param param](#) ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

**Parameters:**

*param* param to remove.

**Returns:**

True if the param was removed.

bool o3d.VertexSource.unbindStream ( [Stream.Semantic semantic](#),  
 Number *semantic\_index*  
 )

Unbinds the requested stream.

**Parameters:**

*semantic* The semantic of the vertices to unbind

*semantic\_index* The semantic index of the vertices to unbind.

**Returns:**

True if unbound. False those vertices do not exist or were not bound.

---

## Member Data Documentation

String [o3d.ObjectBase.className](#) [inherited]

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use `objectBase.isAClassName`

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id** [o3d.ObjectBase.clientId](#) [inherited]

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**String** [o3d.NamedObject.name](#) [inherited]

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array** [o3d.ParamObject.params](#) [inherited]

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;  
for (var i = 0; i < params.length; i++) {  
    var param = params[i];  
}
```

Note that modifications to this array [e.g. `push()`] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

## o3d.Viewport Class Reference

Inherits [o3d.RenderNode](#).

[List of all members](#).

---

## Detailed Description

A [Viewport](#) is a render node that sets the render viewport and depth range for its children. It uses an array in the format [left, top, width, height] where left, top, width and height are in a 0.0 to 1.0 range that represent positions and dimensions relative to the size of the client's rendering area. The depth range is represented by an array in the format [min Z, max Z]. The depth range provides the mapping of the clip space coordinates into normalized z buffer coordinates.

## Public Member Functions

- Array [getRenderNodesInTree \(\)](#)
- Array [getRenderNodesByNameInTree \(String name\)](#)
- Array [getRenderNodesByClassNameInTree \(String class\\_name\)](#)
- [Param createParam \(String param\\_name, String param\\_type\\_name\)](#)
- [Param getParam \(String param\\_name\)](#)
- bool [removeParam \(Param param\)](#)
- [copyParams \(ParamObject source\\_param\\_object\)](#)
- bool [isAClassName \(String class\\_name\)](#)

## Public Attributes

- [Float4 viewport](#)
  - [Float2 depthRange](#)
  - Number [priority](#)
  - bool [active](#)
  - [RenderNode parent](#)
  - Array [children](#)
  - Array [params](#)
  - String [name](#)
  - Id [clientId](#)
  - String [className](#)
-

# Member Function Documentation

`o3d.ParamObject.copyParams ( ParamObject source_param_object ) [inherited]`

Copies all the params from a the given `source_param_object` to this param object. Does not replace any currently existing params with the same name.

## Parameters:

*source\_param\_object* param object to copy params from.

[Param](#) `o3d.ParamObject.createParam ( String param_name,  
String param_type_name  
)` [inherited]

Creates a [Param](#) with the given name and type on the [ParamObject](#). Will fail if a param with the same name already exists.

## Parameters:

*param\_name* The name of the [Param](#) to be created.

*param\_type\_name* The type of [Param](#) to create. Valid types are

- 'o3d.ParamBoolean',
- 'o3d.ParamBoundingBox',
- 'o3d.ParamDrawContext',
- 'o3d.ParamDrawList',
- 'o3d.ParamEffect',
- 'o3d.ParamFloat',
- 'o3d.ParamFloat2',
- 'o3d.ParamFloat3',
- 'o3d.ParamFloat4',
- 'o3d.ParamInteger',
- 'o3d.ParamMaterial',
- 'o3d.ParamMatrix4',
- 'o3d.ParamParamArray',
- 'o3d.ParamRenderSurface',
- 'o3d.ParamRenderDepthStencilSurface',
- 'o3d.ParamSampler',
- 'o3d.ParamSkin',
- 'o3d.ParamSteamBank',

- 'o3d.ParamState',
- 'o3d.ParamString',
- 'o3d.ParamTexture',
- 'o3d.ParamTransform',
- 'o3d.ProjectionParamMatrix4',
- 'o3d.ProjectionInverseParamMatrix4',
- 'o3d.ProjectionTransposeParamMatrix4',
- 'o3d.ProjectionInverseTransposeParamMatrix4',
- 'o3d.ViewParamMatrix4',
- 'o3d.ViewInverseParamMatrix4',
- 'o3d.ViewTransposeParamMatrix4',
- 'o3d.ViewInverseTransposeParamMatrix4',
- 'o3d.ViewProjectionParamMatrix4',
- 'o3d.ViewProjectionInverseParamMatrix4',
- 'o3d.ViewProjectionTransposeParamMatrix4',
- 'o3d.ViewProjectionInverseTransposeParamMatrix4',
- 'o3d.WorldParamMatrix4',
- 'o3d.WorldInverseParamMatrix4',
- 'o3d.WorldTransposeParamMatrix4',
- 'o3d.WorldInverseTransposeParamMatrix4',
- 'o3d.WorldViewParamMatrix4',
- 'o3d.WorldViewInverseParamMatrix4',
- 'o3d.WorldViewTransposeParamMatrix4',
- 'o3d.WorldViewInverseTransposeParamMatrix4',
- 'o3d.WorldViewProjectionParamMatrix4',
- 'o3d.WorldViewProjectionInverseParamMatrix4',
- 'o3d.WorldViewProjectionTransposeParamMatrix4',
- 'o3d.WorldViewProjectionInverseTransposeParamMatrix4'

**Returns:**

The newly created [Param](#) or null on failure.

[Param](#) o3d.ParamObject.getParam ( String *param\_name* ) [inherited]

Searches by name for a [Param](#) defined in the object.

**Parameters:**

*param\_name* Name to search for.

**Returns:**

The [Param](#) with the given name, or null otherwise.

Array

( String  $\frac{class\_nam}{e}$  ) [inherited]

o3d.RenderNode.getRenderNodesByClassNameInTree

Searches for render nodes that match the given class name in the hierarchy under and including this render node.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*class\_name* class name to look for.

**Returns:**

An array containing all nodes among this node and its descendants whose type is *class\_name*.

Array o3d.RenderNode.getRenderNodesByNameInTree ( String *name* ) [inherited]

Searches for render nodes that match the given name in the hierarchy under and including this render node. Since there can be several render nodes with a given name the results are returned in an array.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Parameters:**

*name* Rendernode name to look for.

**Returns:**

An array containing all nodes among this node and its descendants that have the given name.

Array o3d.RenderNode.getRenderNodesInTree ( ) [inherited]

Returns this render node and all its descendants. Note that this render node might not be in the render graph.

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

**Returns:**

An array containing all render nodes of the subtree.

bool o3d.ObjectBase.isAClassName ( String *class\_name* ) [inherited]

Takes the name of a class as an argument, and returns true if this object is either an instance of that class or derives from that class.

```
var t = pack.createObject('o3d.Transform');
t.isAClassName('o3d.Transform');      // true
t.isAClassName('o3d.ParamObject');   // true
t.isAClassName('o3d.Shape');         // false
```

#### Parameters:

*class\_name* Name of class to check for.

#### Returns:

true if this object is a or is derived from the given class name.

bool o3d.ParamObject.removeParam ( [Param](#) *param* ) [inherited]

Removes a [Param](#) from a [ParamObject](#).

This function will fail if the param does not exist on this [ParamObject](#) or if the param is unremovable.

#### Parameters:

*param* param to remove.

#### Returns:

True if the param was removed.

---

## Member Data Documentation

bool [o3d.RenderNode.active](#) [inherited]

Setting false skips this render node. Setting true processes this render node. (ie, renders whatever it's supposed to render)

Array [o3d.RenderNode.children](#) [inherited]

The immediate children of this [RenderNode](#).

Each access to this field gets the entire list so it is best to get it just once. For example:

```
var children = renderNode.children;
for (var i = 0; i < children.length; i++) {
    var child = children[i];
```

```
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [RenderNode](#), while modifications to the array's members **will** affect them.

This property is read-only.

**String [o3d.ObjectBase.className](#) [inherited]**

The concrete class name for an object derived from [ObjectBase](#).

If you want to know if an object is of a certain type you should use objectBase.isAClassName

```
var t = pack.createObject('o3d.Transform');  
t.className == 'o3d.Transform'; // true
```

This property is read-only.

**Id [o3d.ObjectBase.clientId](#) [inherited]**

Unique id of the object.

This id will be unique, even across multiple O3D clients in the same page.

This property is read-only.

**Float2 [o3d.Viewport.depthRange](#)**

The min Z and max Z depth range in [min Z, max Z] format. The default values are [0.0, 1.0].

**See also:**

[Viewport](#)

**String [o3d.NamedObject.name](#) [inherited]**

The object's name.

Setting this has no meaning to O3D, but is useful for debugging and for the functions

[Client.getObjects](#), [Pack.getObject](#), [RenderNode.getRenderNodesByNameInTree](#) and

[RenderNode.getTransformsByNameInTree](#) which search for objects by name.

Reimplemented from [o3d.NamedObjectBase](#).

**Array [o3d.ParamObject.params](#) [inherited]**

Gets all the param on this param object.

Each access to this field gets the entire list, so it is best to get it just once. For example:

```
var params = paramObject.params;
for (var i = 0; i < params.length; i++) {
    var param = params[i];
}
```

Note that modifications to this array [e.g. push()] will not affect the underlying [ParamObject](#), while modifications to the array's members **will** affect them.

This property is read-only.

[RenderNode o3d.RenderNode.parent](#) [inherited]

Sets the parent of the node by re-parenting the node under parent\_node. Setting parent\_node to null removes the node and the entire subtree below it from the render graph.

This property is write-only.

Number [o3d.RenderNode.priority](#) [inherited]

Sets the priority of this render node. lower priorities are rendered first.

[Float4 o3d.Viewport.viewport](#)

The position and size to set the viewport in [left, top, width, height] format. The default values are (0.0, 0.0, 1.0, 1.0). In other words, the full area.

Note: These values must describe a rectangle that is 100% inside the client area. In other words, (0.5, 0.0, 1.0, 1.0) would describe an area that is 1/2 off right side of the screen. That is an invalid value and will be clipped to (0.5, 0.0, 0.5, 1.0).

**See also:**

[Viewport](#)

## Vectormath.Aos.Matrix3 Class Reference

[List of all members.](#)

---

### Detailed Description

A data type representing a 3x3 Matrix. A [Matrix3](#) is represented in JavaScript by an array containing 3 arrays of 3 numbers each:

`[[x0, y0, z0],`

```
[x1, y1, z1],  
[x2, y2, z2]]
```

# Matrix4 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type representing a 4x4 Matrix. A [Matrix4](#) is represented in JavaScript by an array containing 4 arrays of 4 numbers each:

```
[[x0, y0, z0, w0],  
 [x1, y1, z1, w1],  
 [x2, y2, z2, w2],  
 [x3, y3, z3, w3]]
```

# Point3 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type representing a 3d point. A [Point3](#) is represented in JavaScript by an array containing 3 numbers: [x, y, z].

# Quat Class Reference

[List of all members.](#)

---

## Detailed Description

A data type representing a Quaterion. A [Quat](#) is represented in JavaScript by an array containing 4 numbers: [x, y, z, w]

# Vector3 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type representing a 3d direction. A [Vector3](#) is represented in JavaScript by an array containing 3 numbers: [x, y, z].

## Vector4 Class Reference

[List of all members.](#)

---

## Detailed Description

A data type representing a 4 value Vector. A [Vector4](#) is represented in JavaScript by an array containing 4 numbers: [x, y, z, w].

# O3DJS Module List

Here are the modules with brief descriptions:

[global](#)

[o3djs](#)

[o3djs.arcball](#)

[o3djs.base](#)

[o3djs.camera](#)

[o3djs.canvas](#)

[o3djs.debug](#)

[o3djs.dump](#)

[o3djs.effect](#)

[o3djs.element](#)

[o3djs.error](#)

[o3djs.event](#)

[o3djs.fps](#)

[o3djs.io](#)

[o3djs.loader](#)

[o3djs.material](#)

[o3djs.math](#)

[o3djs.pack](#)

[o3djs.particles](#)

[o3djs.picking](#)

[o3djs.primitives](#)

[o3djs.quaternions](#)

[o3djs.rendergraph](#)

[o3djs.scene](#)

[o3djs.serialization](#)

[o3djs.shape](#)

[o3djs.simple](#)

[o3djs.util](#)

# global Module Reference

[List of all members.](#)

---

## Detailed Description

### Public Properties

- undefined [goog](#)
- 

## Member Property Documentation

undefined \_global\_.goog

Define this because the Google internal JSCompiler needs goog.typedef below.

# o3djs Module Reference

[List of all members.](#)

---

## Detailed Description

A namespace for all the o3djs utility libraries.

### Source

[o3djs/base.js](#)

### Public Member Functions

- <static> [exportSymbol](#)(publicPath, object, opt\_objectToExportTo)
- <static> Object [getObjectName](#)(name, opt\_obj)
- <static> boolean [isDef](#)(val)
- <static> [provide](#)(name)
- <static> [require](#)(rule)

### Public Properties

- <static> undefined [arcball](#)

- <static> undefined [base](#)
  - <static> string [basePath](#)
  - <static> boolean [BROWSER\\_ONLY](#)
  - <static> undefined [camera](#)
  - <static> undefined [canvas](#)
  - <static> undefined [debug](#)
  - <static> undefined [dump](#)
  - <static> undefined [effect](#)
  - <static> undefined [element](#)
  - <static> undefined [error](#)
  - <static> undefined [event](#)
  - <static> undefined [fps](#)
  - <static> undefined [global](#)
  - <static> undefined [io](#)
  - <static> undefined [loader](#)
  - <static> undefined [material](#)
  - <static> undefined [math](#)
  - <static> undefined [pack](#)
  - <static> undefined [particles](#)
  - <static> undefined [picking](#)
  - <static> undefined [primitives](#)
  - <static> undefined [quaternions](#)
  - <static> undefined [rendergraph](#)
  - <static> undefined [scene](#)
  - <static> undefined [serialization](#)
  - <static> undefined [shape](#)
  - <static> undefined [simple](#)
  - <static> undefined [test](#)
  - <static> undefined [util](#)
- 

## Member Function Documentation

```
o3djs.exportSymbol ( string publicPath
                    Object object
                    Object opt_objectToExportTo )
```

Exposes an unobfuscated global namespace path for the given object. Note that fields of the exported object \*will\* be obfuscated, unless they are exported in turn via this function or o3djs.exportProperty.

Also handy for making public items that are defined in anonymous closures.

ex. o3djs.exportSymbol('Foo', Foo);

ex. o3djs.exportSymbol('public.path.Foo.staticFunction', Foo.staticFunction);  
public.path.Foo.staticFunction();

ex. o3djs.exportSymbol('public.path.Foo.prototype.myMethod', Foo.prototype.myMethod); new  
public.path.Foo().myMethod();

**Parameters:**

*publicPath* Unobfuscated name to export.  
*object* Object the name should point to.  
*opt\_objectToExportTo* The object to add the path to; default is |  
o3djs.global|.

Object o3djs.getObjectName ( string *name*  
                                  Object *opt\_obj* )

Returns an object based on its fully qualified external name. If you are using a compilation pass that renames property names beware that using this function will not find renamed properties.

**Parameters:**

*name* The fully qualified name.  
*opt\_obj* The object within which to look; default is |  
o3djs.global|.

**Returns:**

Object.The object or, if not found, null.

boolean o3djs.isDef ( \* *val* )

Returns true if the specified value is not |undefined|. WARNING: Do not use this to test if an object has a property. Use the in operator instead.

**Parameters:**

*val* Variable to test.

**Returns:**

boolean.Whether variable is defined.

o3djs.provide ( string *name* )

Creates object stubs for a namespace. When present in a file, o3djs.provide also indicates that the file defines the indicated object.

**Parameters:**

*name* name of the object that this file defines.

o3djs.require ( string *rule* )

Implements a system for the dynamic resolution of dependencies.

**Parameters:**

*rule* Rule to include, in the form o3djs.package.part.

---

## Member Property Documentation

undefined o3djs.arcball

A Module for arcball manipulation.

This is useful for rotating a model with the mouse.

undefined o3djs.base

The base module for o3djs.

string o3djs.basePath

Path for included scripts.

boolean o3djs.BROWSER\_ONLY

Flag used to force a function to run in the browser when it is called from V8.

undefined o3djs.camera

A Module for camera utilites.

undefined o3djs.canvas

A Module for using a 2d canvas.

undefined o3djs.debug

Defines a namespace for o3djs.debug.

undefined o3djs.dump

A Module for dumping information about o3d objects.

undefined o3djs.effect

A Module for dealing with effects.

undefined o3djs.element

A Module for element functions.

undefined o3djs.error

A Module with various error handing functions.

This module is for helping to manage the client's error callback. Because you can not read the current callback on the client we wrap it with these utilities which track the last callback added so as long as you use o3djs.error.setErrorHandler(client, callback) instead of client.setErrorCallback you'll be able to get and restore the error callback when you need to.

undefined o3djs.event

A Module for handling events related to o3d and various browsers.

undefined o3djs.fps

A Module with a fps class for helping to easily display frames per second.

undefined o3djs.global

Reference to the global context. In most cases this will be 'window'.

undefined o3djs.io

A Module with various io functions and classes.

undefined o3djs.loader

A Module with a loader class for helping to load muliple assets in an asynchronous manner.

undefined o3djs.material

A Module for materials.

undefined o3djs.math

A module for math for o3djs.math.

undefined o3djs.pack

A Module with utilities for dealing with packs..

undefined o3djs.particles

A Module with various GPU particle functions and classes. Note: GPU particles have the issue that they are not sorted per particle but rather per emitter.

undefined o3djs.picking

A Module for picking.

undefined o3djs.primitives

A Module for creating primitives.

undefined o3djs.quaternions

A Module for quaternion math.

undefined o3djs.rendergraph

A Module for creating render graphs.

undefined o3djs.scene

A Module with various scene functions and classes.

undefined o3djs.serialization

A Module for handling events related to o3d and various browsers.

undefined o3djs.shape

A Module for shapes.

undefined o3djs.simple

A Module for using o3d in a very simple way.

undefined o3djs.test

A unit testing library

undefined o3djs.util

A Module with various utilities.

## **o3djs.arcball Module Reference**

[List of all members.](#)

---

### **Detailed Description**

A Module for arcball manipulation.

This is useful for rotating a model with the mouse.

## Source

[o3djs/arcball.js](#)

## Public Member Functions

- <static> ![o3djs.arcball.ArcBall](#) [create](#)(areaWidth, areaHeight)
- 

## Member Function Documentation

[!o3djs.arcball.ArcBall](#) o3djs.arcball.create ( number *areaWidth*  
number *areaHeight* )

Creates a new arcball.

### Parameters:

*areaWidth* width of area arcball should cover.  
*areaHeight* height of area arcball should cover.

### Returns:

[!o3djs.arcball.ArcBall](#).The created arcball.

### See Also:

- [o3djs.arcball](#)

## o3djs.base Module Reference

[List of all members.](#)

---

## Detailed Description

The base module for o3djs.

## Source

[o3djs/base.js](#)

## Public Member Functions

- <static> string [formatErrorStack](#)(stack)
- <static> string [getFunctionName](#)(aFunction)
- <static> string [getStackTrace](#)(stripCount)
- <static> [init](#)(clientObject)
- <static> [initV8](#)(clientObject)
- <static> boolean [isArray](#)(value)

- <static> boolean [IsChrome10\(\)](#)
- <static> boolean [IsMSIE\(\)](#)
- <static> !Array.<string> [parseErrorStack\(excp\)](#)
- <static> boolean [ready\(\)](#)
- <static> [setErrorHandler\(client\)](#)
- <static> [snapshotProvidedNamespaces\(\)](#)

## Public Properties

- <static> [o3d.o3d o3d](#)
- 

## Member Function Documentation

string o3djs.base.formatErrorStack ( Array.<string> *stack* )

Pretty prints an exception's stack, if it has one.

**Parameters:**

*stack* An array of errors.

**Returns:**

string.The pretty stack.

string o3djs.base.getFunctionName ( !function(...): \* *aFunction* )

Gets a function name from a function object.

**Parameters:**

*aFunction* The function object to try to get a name from.

**Returns:**

string.function name or 'anonymous' if not found.

string o3djs.base.getStackTrace ( number *stripCount* )

Gets a stack trace as a string.

**Parameters:**

*stripCount* The number of entries to strip from the top of the stack. Example: Pass in 1 to remove yourself from the stack trace.

**Returns:**

string.The stack trace.

o3djs.base.init ( !Element *clientObject* )

Initializes the o3djs.sample library. Basically all it does is record the o3djs.namespace object which is used by other functions to look up o3d constants.

**Parameters:**

*clientObject* O3D.Plugin Object.

o3djs.base.initV8 ( ![o3d.plugin](#) *clientObject* )

Initializes the o3djs.sample library in a v8 instance. This should be called for every V8 instance that uses the sample library. It is automatically called by o3djs.util.makeClients.

**Parameters:**

*clientObject* O3D.Plugin Object.

boolean o3djs.base.isArray ( \* *value* )

Determine whether a value is an array. Do not use instanceof because that will not work for V8 arrays (the browser thinks they are Objects).

**Parameters:**

*value* A value.

**Returns:**

boolean.Whether the value is an array.

o3djs.base.IsChrome10 ( )

Returns true if the user's browser is Chrome 1.0, that requires a workaround to create the plugin.

**Returns:**

boolean.true if the user's browser is Chrome 1.0.

o3djs.base.IsMSIE ( )

Returns true if the user's browser is Microsoft IE.

**Returns:**

boolean.true if the user's browser is Microsoft IE.

!Array.<string> o3djs.base.parseErrorStack ( !Exception *excp* )

Parses an error stack from an exception

**Parameters:**

*excp* The exception to get a stack trace from.

**Returns:**

!Array.<string>.An array of strings of the stack trace.

o3djs.base.ready ( )

Check if the o3djs library has been initialized.

**Returns:**

boolean.true if ready, false if not.

o3djs.base.setErrorHandler ( ![o3d.Client](#) *client* )

Sets the error handler on a client to a handler that displays an alert on the first error.

**Parameters:**

*client* The client object of the plugin.

o3djs.base.snapshotProvidedNamespaces ( )

Snapshots the current state of all provided namespaces. This state will be used to initialize future V8 instances. It is automatically called by o3djs.util.makeClients.

---

# Member Property Documentation

## [o3d.o3d](#) o3djs.base.o3d

The a Javascript copy of the o3d namespace object. (holds constants, enums, etc...)

# o3djs.camera Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for camera utilites.

## Source

[o3djs/camera.js](#)

## Public Member Functions

- <static> !Array.<!o3d.Transform> [findCameras](#)(treeRoot)
  - <static> [fitContextToScene](#)(treeRoot, clientWidth, clientHeight, drawContext)
  - <static> ![o3djs.camera.CameraInfo](#) [getCameraFitToScene](#)(treeRoot, clientWidth, clientHeight)
  - <static> !Array.<![o3djs.camera.CameraInfo](#)> [getCameraInfos](#)(treeRoot, areaWidth, areaHeight)
  - <static> ![o3djs.camera.CameraInfo](#) [getViewAndProjectionFromCamera](#)(camera, areaWidth, areaHeight)
  - <static> ![o3djs.camera.CameraInfo](#) [getViewAndProjectionFromCameras](#)(treeRoot, areaWidth, areaHeight)
- 

## Member Function Documentation

`!Array.<!o3d.Transform> o3djs.camera.findCameras ( !o3d.Transform treeRoot )`

Searches for all nodes with a "o3d.tags" ParamString that contains the word "camera" assuming comma separated words.

### Parameters:

*treeRoot* Root of tree to search for cameras.

### Returns:

`!Array.<!o3d.Transform>`.Array of camera transforms.

`o3djs.camera.fitContextToScene ( !o3d.Transform treeRoot`

number

*treeRoot*

*clientWidth*

number

*clientHeight*

[!o3d.DrawContext](#) *drawContext* )

Sets the view and projection of a DrawContext to view the bounding box that encompasses the tree of transforms passed.

This function is here to help debug a program by providing an easy way to attempt to get your content in front of the camera.

### Parameters:

*treeRoot* Root of sub tree to get extents from.

*clientWidth* width of client area.

*clientHeight* height of client area.

*drawContext* DrawContext to set view and projection on.

Get CameraInfo that represents a view of the bounding box that encompasses a tree of transforms.

### Parameters:

*treeRoot* Root of sub tree to get extents from.

*clientWidth* width of client area.

*clientHeight* height of client area.

## Returns:

`!o3djs.camera.CameraInfo`.A CameraInfo object.

Calls `findCameras` and creates an array of `CameraInfos` for each camera found.

### Parameters:

*treeRoot* Root of tree to search for cameras.

*areaWidth* Width of client area.

*areaHeight* Height of client area.

## Returns:

`!Array.<!o3djs.camera.CameraInfo>.A CameraInfo object.`

**!o3djs.camera.CameraInfo**  
o3djs.camera.getViewAndProjectionFromCamera  
( **!o3d.Transform** *camera*  
          number       *areaWidth*  
          number       *areaHeight* )

Creates a object with view and projection matrices using parameters found on the camera  
'o3d.projection\_near\_z', 'o3d.projection\_far\_z', and 'o3d.perspective\_fov\_y' as well as the areaWidth  
and areaHeight passed in.

### Parameters:

*camera* Transform with camera information on it.

*areaWidth* width of client area.

*areaHeight* height of client area.

**Returns:**

[!o3djs.camera.CameraInfo](#).A CameraInfo object.

[!o3djs.camera.CameraInfo](#) ([!o3d.Transform](#) *treeRoot*  
*o3djs.camera.getViewAndProjectionFromCameras* number *areaWidth*  
number *areaHeight*)

Calls `findCameras` and takes the first camera. Then calls `o3djs.camera.getViewAndProjectionFromCamera`. If no camera is found it sets up some defaults.

**Parameters:**

*treeRoot* Root of tree to search for cameras.

*areaWidth* Width of client area.

*areaHeight* Height of client area.

**Returns:**

[!o3djs.camera.CameraInfo](#).A CameraInfo object.

# o3djs.canvas Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for using a 2d canvas.

## Source

[o3djs/canvas.js](#)

## Public Member Functions

- <static> [!o3djs.canvas.CanvasInfo](#) `create(pack, root, viewInfo)`

## Public Properties

- <static> string [FX\\_STRING](#)
- 

## Member Function Documentation

[!o3djs.canvas.CanvasInfo](#) `o3djs.canvas.create ( !o3d.Pack pack  
                                  !o3d.Transform root  
                                  !o3djs.rendergraph.ViewInfo viewInfo )`

Creates an o3djs.canvas library object through which CanvasQuad objects can be created.

**Parameters:**

*pack* to manage objects created by this library.

*root* Default root for visual objects.

*viewInfo* A ViewInfo object as created by o3djs.createView which contains draw lists that the created quads will be placed into.

**Returns:**

[!o3djs.canvas.CanvasInfo](#).A CanvasInfo object containing references to all the common O3D elements used by this instance of the library.

---

## Member Property Documentation

string o3djs.canvas.FX\_STRING

The shader code used by the canvas quads. It only does two things: 1. Transforms the shape to screen space via the worldViewProjection matrix. 2. Performs a texture lookup to display the contents of the texture bound to texSampler0.

## o3djs.debug Module Reference

[List of all members.](#)

---

### Detailed Description

Defines a namespace for o3djs.debug.

### Source

[o3djs/debug.js](#)

### Public Member Functions

- <static> [!o3djs.debug.DebugHelper](#) [createDebugHelper](#)(pack, viewInfo)
  - <static> [!o3d.Shape](#) [createLineCube](#)(pack, material, size, opt\_matrix)
  - <static> [!o3djs.debug.VertexInfo](#) [createLineCubeVertices](#)(size, opt\_matrix)
  - <static> [!o3d.Shape](#) [createLineShape](#)(pack, material, vertices, indices)
  - <static> [!o3d.Shape](#) [createLineSphere](#)(pack, material, radius, subdivisionsAxis, subdivisionsHeight, opt\_matrix)
  - <static> [!o3djs.debug.VertexInfo](#) [createLineSphereVertices](#)(radius, subdivisionsAxis, subdivisionsHeight, opt\_matrix)
  - <static> [!o3djs.debug.VertexInfo](#) [createVertexInfo](#)(opt\_vertices, opt\_indices)
  - <static> boolean [isDebugTransform](#)(transform)
-

# Member Function Documentation

[!o3djs.debug.DebugHelper](#)  
o3djs.debug.createDebugHelper

( [!o3d.Pack](#) *pack*  
[!o3djs.rendergraph.viewInfo](#) *viewInfo* )

Creates a debug helper object.

A debug helper object provides functions to help debug your o3d application and manages the resources needed to do that for you. For example it can add axes, spheres and boxes to your transforms as well as draw lines in 3d space given 2 points.

## Parameters:

*pack* Pack for DebugHelper to manage its resources with.  
*viewInfo* ViewInfo for debug visuals.

## Returns:

[!o3djs.debug.DebugHelper](#).the DebugHelper object.

[!o3d.Shape](#) o3djs.debug.createLineCube ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material*  
*number* *size*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates a cube of lines.

## Parameters:

*pack* Pack to create sphere elements in.  
*material* to use.  
*size* Width, height and depth of the cube.  
*opt\_matrix* A matrix by which to multiply all the vertices.

## Returns:

[!o3d.Shape](#).The created cube.

[!o3djs.debug.VertexInfo](#) o3djs.debug.createLineCubeVertices ( *number* *size*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates the vertices and indices for a cube of lines. The cube will be created around the origin. (-size / 2, size / 2) The created cube has a position stream only and can therefore only be used with shaders that support those a position stream.

## Parameters:

*size* Width, height and depth of the cube.  
*opt\_matrix* A matrix by which to multiply all the vertices.

## Returns:

[!o3djs.debug.VertexInfo](#).The created cube vertices.

[!o3d.Shape](#) o3djs.debug.createLineShape ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material* )

```

!Array.<!o3djs.math.Vector3> vertices
!Array.<number> indices
)
```

Creates a line list shape and primitive given a material, vertex array and index array.

**Parameters:**

- pack* Pack to create objects in.
- material* to use.
- vertices* array of numbers in the format positionX, positionY, positionZ.
- indices* array of vertex indices, 2 per line.

**Returns:**

[!o3d.Shape](#).The created shape.

```

!o3d.Shape o3djs.debug.createLineSphere ( !o3d.Pack pack
                                         !o3d.Material material
                                         number radius
                                         number subdivisionsAxis
                                         number subdivisionsHeight
                                         !o3djs.math.Matrix4 opt_matrix
)
```

Creates a sphere. The created sphere has position, normal, uv and vertex color streams only and can therefore only be used with shaders that support those 4 streams.

**Parameters:**

- pack* Pack to create sphere elements in.
- material* to use.
- radius* radius of the sphere.
- subdivisionsAxis* number of steps around the sphere.
- subdivisionsHeight* number of vertically on the sphere.
- opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created sphere.

**See Also:**

- [o3d.Pack](#)
- [o3d.Shape](#)

```

!o3djs.debug.VertexInfo
o3djs.debug.createLineSphereVertices ( number radius
                                         number subdivisionsAxis
                                         number subdivisionsHeight
                                         !o3djs.math.Matrix4 opt_matrix
)

```

Creates sphere vertices. The created sphere has a position stream only and can therefore only be used with shaders that support those a position stream.

**Parameters:**

- radius* radius of the sphere.

*subdivisionsAxis* number of steps around the sphere.  
*subdivisionsHeight* number of vertically on the sphere.  
*opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.debug.VertexInfo](#).The created sphere vertices.

[!o3djs.debug.VertexInfo](#)  
o3djs.debug.createVertexInfo  
( !Array.<!  
    [o3djs.math.Vector3](#)>  
    !Array.<number>  
    *opt\_vertices*  
    *opt\_indices* )

Creates a new VertexInfo.

**Parameters:**

*opt\_vertices* array of numbers in the format positionX, positionY, positionZ.  
*opt\_indices* array of indices, 2 per line.

**Returns:**

[!o3djs.debug.VertexInfo](#).The new VertexInfo.

boolean o3djs.debug.isDebugEnabledTransform ( [!o3d.Transform](#) *transform* )  
Checks whether or not a transform is a debug transform.

**Parameters:**

*transform* Transform to check.

**Returns:**

boolean.true if this transform is a debug transform.

## o3djs.dump Module Reference

[List of all members.](#)

---

### Detailed Description

A Module for dumping information about o3d objects.

### Source

[o3djs/dump.js](#)

### Public Member Functions

- <static> [dump\(string\)](#)
- <static> [dumpBoundingBox\(label, boundingBox, opt\\_prefix\)](#)
- <static> [dumpElement\(element, opt\\_prefix\)](#)
- <static> [dumpFloat3\(label, float3, opt\\_prefix\)](#)

- <static> dumpFloat4(label, float4, opt\_prefix)
  - <static> dumpMatrix(label, matrix, opt\_prefix)
  - <static> dumpNamedObjectName(namedObject, opt\_prefix)
  - <static> dumpParam(param, opt\_prefix)
  - <static> dumpParamObject(param\_object, opt\_prefix)
  - <static> dumpParams(param\_object, opt\_prefix)
  - <static> dumpPoint3(label, point3, opt\_prefix)
  - <static> dumpRenderNode(render\_node, opt\_prefix)
  - <static> dumpRenderNodeTree(render\_node, opt\_prefix)
  - <static> dumpShape(shape, opt\_prefix)
  - <static> dumpStackTrace()
  - <static> dumpStream(stream, opt\_prefix)
  - <static> dumpTexture(texture, opt\_prefix)
  - <static> dumpTransform(transform, opt\_prefix)
  - <static> dumpTransformList(transform\_list)
  - <static> dumpTransformTree(transform, opt\_prefix)
  - <static> dumpVector3(label, vector3, opt\_prefix)
  - <static> dumpVector4(label, vector4, opt\_prefix)
  - <static> string getMatrixAsString(matrix, opt\_prefix)
  - <static> string getParamValueAsString(param, opt\_prefix)

# Member Function Documentation

**o3djs.dump.dump ( string *string* )**

Prints a value the console or log or wherever it thinks is appropriate for debugging.

### Parameters:

*string* String to print.

Dump a bounding box.

### Parameters:

*label* Label to put in front of dump.

*boundingBox* BoundingBox to dump.

*opt\_prefix* optional prefix for indenting.

```
o3djs.dump.dumpElement ( !o3d.Element element  
                        string          opt prefix )
```

Given a element dumps its name, all the Params and DrawElements on it.

## Parameters:

*element* Element to dump.

*opt\_prefix* Optional prefix for indenting.

`o3djs.dump.dumpFloat3 ( string label  
!o3d.Float3 float3 )`

```
        string      opt_prefix )
```

Dumps a float3

**Parameters:**

*label* Label to put in front of dump.  
*float3* Float3 to dump.  
*opt\_prefix* optional prefix for indenting.

```
o3djs.dump.dumpFloat3 ( string      label
                        !o3d.Float3 float3
                        string      opt_prefix )
```

Dumps a float4

**Parameters:**

*label* Label to put in front of dump.  
*float4* Float4 to dump.  
*opt\_prefix* optional prefix for indenting.

```
o3djs.dump.dumpMatrix ( string      label
                        !o3djs.math.Matrix4 matrix
                        string      opt_prefix )
```

Dumps a matrix

**Parameters:**

*label* Label to put in front of dump.  
*matrix* Matrix to dump.  
*opt\_prefix* optional prefix for indenting.

```
o3djs.dump.dumpNamedObjectName ( !o3d.NamedObject namedObject
                                    string      opt_prefix )
```

Dumps the name and class of a NamedObject.

**Parameters:**

*namedObject* to use.  
*opt\_prefix* Optional prefix for indenting.

```
o3djs.dump.dumpParam ( !o3d.Param param
                        string      opt_prefix )
```

Dumps an single parameter

**Parameters:**

*param* Param to dump.  
*opt\_prefix* Optional prefix for indenting.

```
o3djs.dump.dumpParamObject ( !o3d.ParamObject param_object
                            string      opt_prefix )
```

Given a ParamObject dumps it and all the Params on it.

**Parameters:**

*param\_object* ParamObject to dump.  
*opt\_prefix* Optional prefix for indenting.

```
o3djs.dump.dumpParams ( !o3d.ParamObject param_object
                        string      opt_prefix )
```

Given a ParamObject dumps all the Params on it.

**Parameters:**

*param\_object* ParamObject to dump Params of.  
*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpPoint3 ( *string*      *label*  
                          *!o3d.Point3* *point3*  
                          *string*      *opt\_prefix* )

Dumps a point3

**Parameters:**

*label*      Label to put in front of dump.  
*point3*     Point3 to dump.  
*opt\_prefix* optional prefix for indenting.

o3djs.dump.dumpRenderNode ( *!o3d.RenderNode* *render\_node*  
                              *string*              *opt\_prefix* )

Dumps a RenderNode and all its paramaters.

**Parameters:**

*render\_node* RenderNode to use.  
*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpRenderNodeTree ( *!o3d.RenderNode* *render\_node*  
                              *string*              *opt\_prefix* )

Dumps an entire RenderGraph tree.

**Parameters:**

*render\_node* RenderNode to start dumping from.  
*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpShape ( *!o3d.Shape* *shape*  
                              *string*              *opt\_prefix* )

Given a shape dumps its name, all the Params and Primitves on it.

**Parameters:**

*shape*      Shape to dump.  
*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpStackTrace (    )

Dumps a javascript stack track.

o3djs.dump.dumpStream ( *!o3d.Stream* *stream*  
                              *string*              *opt\_prefix* )

Given a Stream dumps it and all the Params on it.

**Parameters:**

*stream*      Stream to dump.  
*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpTexture ( *!o3d.Texture* *texture*  
                              *string*              *opt\_prefix* )

Given a texture dumps its name and other info. it.

**Parameters:**

*texture* Texture to dump.

*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpTransform ( ![o3d.Transform](#) *transform*  
                          *string*              *opt\_prefix* )

Given a transform dumps its name and all the Params and Shapes on it.

**Parameters:**

*transform* Transform to dump.

*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpTransformList ( !Array.<![o3d.Transform](#)> *transform\_list* )

Dumps a list of Transforms.

**Parameters:**

*transform\_list* Array of Transforms to dump.

o3djs.dump.dumpTransformTree ( ![o3d.Transform](#) *transform*  
                          *string*              *opt\_prefix* )

Dumps an entire transform graph tree.

**Parameters:**

*transform* Transform to start dumping from.

*opt\_prefix* Optional prefix for indenting.

o3djs.dump.dumpVector3 ( *string*            *label*  
                          [!o3d.Vector3](#) *vector3*  
                          *string*            *opt\_prefix* )

Dumps a vector3

**Parameters:**

*label* Label to put in front of dump.

*vector3* Vector3 to dump.

*opt\_prefix* optional prefix for indenting.

o3djs.dump.dumpVector4 ( *string*            *label*  
                          !Array.<number> *vector4*  
                          *string*            *opt\_prefix* )

Dumps a vector4

**Parameters:**

*label* Label to put in front of dump.

*vector4* vector to dump.

*opt\_prefix* optional prefix for indenting.

*string* o3djs.dump.getMatrixAsString ( ![o3djs.math.Matrix4](#) *matrix*  
                          *string*              *opt\_prefix* )

Gets the value of a matrix as a string.

**Parameters:**

*matrix* Matrix4 to get value of.

*opt\_prefix* Optional prefix for indenting.

**Returns:**

string.Value of param.

```
string o3djs.dump.getParamValueAsString ( !o3d.Param param
                                         string      opt_prefix )
```

Gets the value of a parameter as a string.

**Parameters:**

*param* Parameter to get value of.

*opt\_prefix* Optional prefix for indenting.

**Returns:**

string.Value of param.

# o3djs.effect Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for dealing with effects.

## Source

[o3djs/effect.js](#)

## Public Member Functions

- <static> boolean [attachStandardShader](#)(pack, material, lightPos, effectType)
- <static> { {description: string, shader: string} } [buildStandardShaderString](#)(material, effectType)
- <static> ![o3d.Effect createEffectFromFile](#)(pack, url)
- <static> string [getColladaLightingType](#)(material)
- <static> number [getNumTexCoordStreamsNeeded](#)(material)
- <static> [o3d.Effect getStandardShader](#)(pack, material, effectType)
- <static> boolean [isColladaLightingType](#)(lightingType)
- <static> [loadEffect](#)(effect, url)

## Public Properties

- <static> string [COLLADA\\_LIGHTING\\_TYPE\\_PARAM\\_NAME](#)
  - <static> !Object [COLLADA\\_LIGHTING\\_TYPES](#)
  - <static> !Array.<string> [COLLADA\\_SAMPLER\\_PARAMETER\\_PREFIXES](#)
- 

## Member Function Documentation

```
boolean o3djs.effect.attachStandardShader ( !o3d.Pack      pack
                                            !o3d.Material   material
                                            !o3djs.math.Vector3 lightPos
                                            string          effectType )
```

Attaches a shader for a given standard COLLADA material type to the material.

Looks at the material passed in and assigns it an Effect that matches its Params. If a suitable Effect already exists in pack it will use that Effect.

**Parameters:**

*pack* Pack in which to create the new Effect.  
*material* Material for which to build the shader.  
*lightPos* Position of the default light.  
*effectType* Type of effect to create ('phong', 'lambert', 'constant').

**Returns:**

boolean. True on success.

```
{ {description: string, shader: string} }                                ( !o3d.Material material
o3djs.effect.buildStandardShaderString                                         string          effectType )
```

Builds a shader string for a given standard COLLADA material type.

**Parameters:**

*material* Material for which to build the shader.  
*effectType* Type of effect to create ('phong', 'lambert', 'constant').

**Returns:**

{ {description: string, shader: string} }. A description and the shader string.

```
!o3d.Effect o3djs.effect.createEffectFromFile ( !o3d.Pack pack
                                                string      url  )
```

Creates an effect from a file. If the effect already exists in the pack that effect will be returned.

**Parameters:**

*pack* Pack to create effect in.  
*url* Url for effect file.

**Returns:**

[!o3d.Effect](#).The effect.

```
string o3djs.effect.getColladaLightingType ( !o3d.Material material )
```

Returns the collada lighting type of a collada standard material.

**Parameters:**

*material* Material to get lighting type from.

**Returns:**

string. The lighting type or "" if it's not a collada standard material.

```
number o3djs.effect.getNumTexCoordStreamsNeeded ( !o3d.Material material )
```

Get the number of TEXCOORD streams needed by this material.

**Parameters:**

*material* The material MUST be a standard collada material.

**Returns:**

number.The number of TEXCOORD streams needed.

```
o3d.Effect o3djs.effect.getStandardShader ( !o3d.Pack    pack  
                                         !o3d.Material material  
                                         string      effectType )
```

Gets or builds a shader for given standard COLLADA material type.

Looks at the material passed in and assigns it an Effect that matches its Params. If a suitable Effect already exists in pack it will use that Effect.

**Parameters:**

*pack* Pack in which to create the new Effect.

*material* Material for which to build the shader.

*effectType* Type of effect to create ('phong', 'lambert', 'constant').

**Returns:**

[o3d.Effect](#).The created effect.

```
boolean o3djs.effect.isColladaLightingType ( string lightingType )
```

Check if lighting type is a collada lighting type.

**Parameters:**

*lightingType* Lighting type to check.

**Returns:**

boolean.true if it's a collada lighting type.

```
o3djs.effect.loadEffect ( !o3d.Effect effect  
                           string      url    )
```

Loads shader source from an external file and creates shaders for an effect.

**Parameters:**

*effect* The effect to create the shaders in.

*url* The url of the shader source.

---

## Member Property Documentation

string o3djs.effect.COLLADA\_LIGHTING\_TYPE\_PARAM\_NAME

The name of the parameter on a material if it's a collada standard material.

NOTE: This parameter is just a string attached to a material. It has no meaning to the plugin, it is passed from the conditioner to the javascript libraries so that they can build collada like effects.

!Object o3djs.effect.COLLADA\_LIGHTING\_TYPES

The collada standard lighting types.

**!Array.<string> o3djs.effect.COLLADA\_SAMPLER\_PARAMETER\_PREFIXES**  
The FCollada standard materials sampler parameter name prefixes.

# **o3djs.element Module Reference**

## List of all members.

# Detailed Description

## A Module for element functions.

## Source

## o3djs/element.js

## Public Member Functions

- <static> [addMissingTexCoordStreams\(element\)](#)
  - <static> [!o3d.Element duplicateElement\(pack, sourceElement\)](#)
  - <static> [setBoundingBoxAndZSortPoint\(element\)](#)

# Member Function Documentation

```
o3djs.element.addMissingTexCoordStreams ( !o3d.Element element )
```

Adds missing texture coordinate streams to a primitive.

This is very application specific but if it's a primitive and if it uses a collada material the material builder assumes 1 TEXCOORD stream per texture. In other words if you have both a specular texture AND a diffuse texture the builder assumes you have 2 TEXCOORD streams. This assumption is often false.

To work around this we check how many streams the material expects and if there are not enough UVs streams we duplicate the last TEXCOORD stream until there are, making a BIG assumption that that will work.

The problem is maybe you have 4 textures and each of them share texture coordinates. There is information in the collada file about what stream to connect each texture to.

### Parameters:

*element* Element to add streams to.

Copies an element and streambank or buffers so the two will share streambanks, vertex and index

buffers.

**Parameters:**

*pack* Pack to manage created objects.  
*sourceElement* The element to copy.

**Returns:**

[!o3d.Element](#).the new copy of sourceElement.

`o3djs.element.setBoundingBoxAndZSortPoint ( !o3d.Element element )`

Sets the bounding box and z sort point of an element.

**Parameters:**

*element* Element to set bounding box and z sort point  
on.

# o3djs.error Module Reference

[List of all members.](#)

---

## Detailed Description

A Module with various error handing functions.

This module is for helping to manage the client's error callback. Because you can not read the current callback on the client we wrap it with these utilities which track the last callback added so as long as you use `o3djs.error.setErrorHandler(client, callback)` instead of `client.setErrorCallback` you'll be able to get and restore the error callback when you need to.

## Source

[o3djs/error.js](#)

## Public Member Functions

- <static> [!o3djs.error.ErrorCollector](#) `createErrorCollector(client)`
  - <static> [setErrorHandler](#)(client)
  - <static> (function(string): void|null) [setErrorHandler](#)(client, callback)
- 

## Member Function Documentation

[!o3djs.error.ErrorCollector](#) `o3djs.error.createErrorCollector ( !o3d.Client client )`

Creates an ErrorCollector.

**Parameters:**

*client* The client object of the plugin.

**Returns:**

[!o3djs.error.ErrorCollector](#).The created error collector.

`o3djs.error.setDefaultErrorHandler ( !o3d.Client client )`

Sets a default error handler on the client. The default error handler displays an alert on the first error encountered.

**Parameters:**

*client* The client object of the plugin.

`(function(string): void|null)`

`o3djs.error.setErrorHandler`

`( !o3d.Client`

*client*

`(function(string): void|  
null)`

*callback* )

Sets the error handler on a client to a handler that manages the client's error callback. displays an alert on the first error.

**Parameters:**

*client* The client object of the plugin.

*callback* The callack to use, null to clear.

**Returns:**

`(function(string): void|null)`.the previous error callback for this client.

# o3djs.event Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for handling events related to o3d and various browsers.

## Source

[o3djs/event.js](#)

## Public Member Functions

- <static> [addEventListener](#)(pluginObject, type, handler)
- <static> string [appendWithSpace](#)(inStr, extraStr)
- <static> string [appendWithSpaceIf](#)(state, inStr, extraStr)
- <static> [cancel](#)(event)
- <static> [createKeyEvent](#)(eventName, charCode, keyCode, control, alt, shift, meta)
- <static> number [getEventKeyChar](#)(event)
- <static> string [getKeyIdentifier](#)(charCode, keyCode)
- <static> string [getModifierString](#)(control, alt, shift, meta)
- <static> number [keyIdentifierToChar](#)(keyIdent)

- <static> `onKey`(event, pluginObject)
  - <static> string `padWithLeadingZeroes`(str, to\_length)
  - <static> `removeEventListener`(pluginObject, type, handler)
  - <static> `startKeyboardEventSynthesis`(pluginObject)

# Member Function Documentation

```
o3djs.event.addEventListener ( !Element pluginObject  
                           string   type  
                           !Object  handler
```

Convenience function to manage event listeners on the o3d plugin object, intended as a drop-in replacement for the DOM addEventListener [with slightly different arguments, but the same effect].

## Parameters:

*pluginObject* the html object where the o3d plugin lives, which the caller probably obtained by calling getElementById or makeClients.

*type* the event type on which to trigger, e.g. 'mousedown', 'mousemove', etc.  
*handler* either a function or an EventListener object.

## Parameters:

*inStr* base string.

*extraStr* string to append.

## Returns:

string.inStr + '' + extraStr, or just inStr if extraStr is ''.

### Parameters:

*state* whether to append or not.

*inStr* base string.

*extraStr* string to append.

### Returns:

`string.inStr + '' + extraStr`, or just `inStr` if `state` is false.

`o3djs.event.cancel ( !Event event )`

Cancel an event we've handled so it stops propagating upwards. The cancelBubble is for IE, stopPropagation is for all other browsers. preventDefault ensures that the default action is also canceled.

### Parameters:

*event* - the event to cancel

```
    number keyCode
    boolean control
    boolean alt
    boolean shift
    boolean meta )
```

Creates a DOM-level 3 KeyboardEvent. see <http://www.w3.org/TR/DOM-Level-3-Events/events.html#Events-KeyboardEvents-Interfaces>. see <http://developer.mozilla.org/en/DOM/event.initKeyEvent>

**Parameters:**

*eventName* one of 'keypress', 'keydown' or 'keyup'.  
*charCode* the character code for the key.  
*keyCode* the key code for the key.  
*control* whether the control key is down.  
*alt* whether the alt/option key is down.  
*shift* whether the shift key is down.  
*meta* whether the meta/command key is down.

number o3djs.event.getEventKeyChar ( !Event *event* )

Extracts the key char in number form from the event, in a cross-browser manner.

**Parameters:**

*event*.

**Returns:**

number.unicode code point for the key.

string o3djs.event.getKeyIdentifier ( number *charCode*
 number *keyCode* )

Creates a keyIdentifier string for a given keystroke as specified in the w3c spec on <http://www.w3.org/TR/DOM-Level-3-Events/events.html>.

**Parameters:**

*charCode* numeric unicode code point as reported by the OS.  
*keyCode* numeric keyCode as reported by the OS, currently unused but will probably be necessary in the future.

**Returns:**

string.eg 'Left' or 'U+0040'.

string o3djs.event.getModifierString ( boolean *control*
 boolean *alt*
 boolean *shift*
 boolean *meta* )

Builds a DOM-level 3 modifier string for a KeyboardEvent - see <http://www.w3.org/TR/DOM-Level-3-Events/events.html#Events-KeyboardEvents-Interfaces>.

**Parameters:**

*control* whether the control key is down.  
*alt* whether the alt/option key is down.  
*shift* whether the shift key is down.  
*meta* whether the meta/command key is down.

## Returns:

string.space delimited list of keys that are down.

number o3djs.event.keyIdentifierToChar ( string *keyIdent* )

Takes a keyIdentifier string and remaps it to an ASCII/Unicode value suitable for javascript event handling.

## Parameters:

*keyIdent* a keyIdentifier string as generated above.

## Returns:

number.the numeric Unicode code point represented.

```
o3djs.event.onKey ( !Element event  
                    !o3d.Plugin pluginObject )
```

Dispatches a DOM-level 3 KeyboardEvent when called back by the plugin. see <http://www.w3.org/TR/DOM-Level-3-Events/events.html#Events-KeyboardEvents-Interfaces> see <http://developer.mozilla.org/en/DOM/event.initKeyEvent>

## Parameters:

*event* an O3D event object.

*pluginObject* the plugin object on the page.

string o3djs.event.padWithLeadingZeroes ( string str  
number to

Pad a string with leading zeroes if needed until it is the length desired.

### Parameters:

*str* The input string, probably representing a number.

*to\_length* The desired minimum length of string with padding.

## Returns:

`string`.A string padded with leading zeroes as needed to be the length desired.

Convenience function to manage event listeners on the o3d plugin object, intended as a drop-in replacement for the DOM removeEventListener [with slightly different arguments, but the same effect].

### Parameters:

*pluginObject* the where the o3d plugin lives, which the caller probably obtained by calling `getElementById`.

*type* the event type on which the handler to be removed was to trigger, e.g. 'mousedown', 'mousemove', etc.

*handler* either a function or an EventListener object.

`o3djs.event.startKeyboardEventSynthesis ( !Element pluginObject )`

Convenience function to setup synthesizing and dispatching of keyboard events whenever the focussed plug-in calls Javascript to report a keyboard action.

## Parameters:

*pluginObject* the where the o3d plugin lives, which the caller probably obtained by calling `getElementById`.

# o3djs.fps Module Reference

[List of all members.](#)

---

## Detailed Description

A Module with a fps class for helping to easily display frames per second.

## Source

[o3djs/fps.js](#)

## Public Member Functions

- <static> [createFPSManager](#)(*pack*, *clientWidth*, *clientHeight*, *opt\_parent*)

## Public Properties

- <static> string [CONST\\_COLOR\\_EFFECT](#)
  - <static> number [NUM\\_FRAMES\\_TO\\_AVERAGE](#)
  - <static> !Array<!o3djs.math.Vector4> [PERF\\_BAR\\_COLORS](#)
- 

## Member Function Documentation

`o3djs.fps.createFPSManager ( !o3d.Pack                   pack  
  number                           clientWidth  
  number                           clientHeight  
  !o3d.RenderNode opt_parent  )`

Creates an object for displaying frames per second.

You can use it like this.

```
<html><body>
<script type="text/javascript" src="o3djs/base.js">
</script>
<script type="text/javascript">
o3djs.require('o3djs.util');
o3djs.require('o3djs.rendergraph');
o3djs.require('o3djs.fps');
window.onload = init;
window.onunload = uninit;
var g_client;
var g_fpsManager;
```

```

function init() {
    o3djs.base.makeClients(initStep2);
}
function initStep2(clientElements) {
    var clientElement = clientElements[0];
    var g_client = clientElement.client;
    var pack = g_client.createPack();
    var viewInfo = o3djs.rendergraph.createBasicView(
        pack,
        g_client.root,
        g_client.renderGraphRoot);
    g_fpsManager = o3djs.fps.createFPSManager(pack,
                                                g_client.width,
                                                g_client.height,
                                                g_client.renderGraphRoot);
    g_client.setRenderCallback(onRender);
}
function onrender(renderEvent) {
    g_fpsManager.update(renderEvent);
}
function uninit() {
    if (g_client) {
        g_client.cleanup();
    }
}
</script>
<div id="o3d" style="width: 600px; height: 600px"></div>
</body></html>

```

#### Parameters:

*pack* Pack to create objects in.

*clientWidth* width of client area.

*clientHeight* Height of client area.

*opt\_parent* RenderNode to use as parent for ViewInfo that will be used to render the FPS with.

---

## Member Property Documentation

string o3djs.fps.CONST\_COLOR\_EFFECT

The shader code used by the pref quads.

number o3djs.fps.NUM\_FRAMES\_TO\_AVERAGE

Number of frames to average over for computing FPS.

!Array.<!o3djs.math.Vector4> o3djs.fps.PERF\_BAR\_COLORS

Colors used for each second of the performance bar.

## o3djs.io Module Reference

[List of all members.](#)

---

# Detailed Description

A Module with various io functions and classes.

## Source

## o3djs/io.js

## Public Member Functions

- <static> [!o3djs.io.LoadInfo](#) [createLoadInfo](#)(opt\_request, opt\_hasStatus)
  - <static> [!o3djs.io.LoadInfo](#) [loadArchive](#)(pack, url, onFinished)
  - <static> [!o3djs.io.LoadInfo](#) [loadArchiveAdvanced](#)(pack, url, onFileAvailable, onFinished)
  - <static> [!o3djs.io.LoadInfo](#) [loadTextFile](#)(url, callback)
  - <static> string [loadTextFileSyncronous](#)(url)
  - <static> [!o3djs.io.LoadInfo](#) [loadTexture](#)(pack, url, callback)

## Member Function Documentation

```
!o3djs.io.LoadInfo  
o3djs.io.createLoadInfo  
    ( (!o3d.ArchiveRequest|!o3d.FileRequest|!  
        XMLHttpRequest)  
        boolean  
    )  
        opt_request  
        opt_hasStat  
    )  
        us
```

Creates a LoadInfo object.

### Parameters:

*opt\_request* The request to watch.

*opt\_hasStatus* true if *opt\_request* is a `o3d.ArchiveRequest` vs for example an `o3d.FileRequest` or  
is an XMLHttpRequest.

## Returns:

[!o3djs.io.LoadInfo](http://o3djs.io.LoadInfo), The new LoadInfo.

#### See Also:

- o3djs.io.LoadInfo

```
!o3djs.io.LoadInfo o3djs.io.loadArchive ( !o3d.Pack pack  
                                         string url  
                                         !function(!o3djs.io.ArchiveInfo, *): void onFinished )
```

Loads an archive file. When the entire archive is ready the `onFinished` callback will be called with an `ArchiveInfo` for accessing the archive.

### Parameters:

*pack*      Pack to create request in.

*url* The url of the archive file.

*onFinished* A callback that is called when the archive is successfully loaded and an Exception object which is null on success.

**Returns:**

[!o3djs.io.LoadInfo](#).The a LoadInfo for tracking progress.

**See Also:**

- [o3djs.io.ArchiveInfo](#)

[!o3djs.io.LoadInfo](#)

o3djs.io.loadArchiveAdvanced

( [!o3d.Pack](#)

*pack*

*string*

*url*

*!function(!o3d.RawData): void*

*onFileAvailable*

*!function(!o3d.ArchiveRequest,*

*onFinished*

*\*) : void*

)

Loads an archive file. This function is asynchronous. This is a low level version of o3djs.io.loadArchive which can be used for things like progressive loading.

**Parameters:**

*pack* Pack to create request in.

*url* The url of the archive file.

*onFileAvailable* A callback, taking a single argument 'data'. As each file is loaded from the archive, this function is called with the file's data.

*onFinished* A callback that is called when the archive is successfully loaded. It is passed the ArchiveRquest and null on success or a javascript exception on failure.

**Returns:**

[!o3djs.io.LoadInfo](#).A LoadInfo for tracking progress.

[!o3djs.io.LoadInfo](#) o3djs.io.loadTextFile ( *string*

*url*

*function(string, \*): void callback* )

Loads text from an external file. This function is asynchronous.

**Parameters:**

*url* The url of the external file.

*callback* A callback passed the loaded string and an exception which will be null on success.

**Returns:**

[!o3djs.io.LoadInfo](#).A LoadInfo to track progress.

*string* o3djs.io.loadTextFileSyncous ( *string url* )

Loads text from an external file. This function is synchronous.

**Parameters:**

*url* The url of the external file.

**Returns:**

*string*.the loaded text if the request is synchronous.

[!o3djs.io.LoadInfo](#) o3djs.io.loadTexture ( [!o3d.Pack](#)

*pack*

*string*

*url*

*!function(o3d.Texture, \*): void callback* )

Loads a texture.

Textures are loaded asynchronously.

Example:

```
var loadInfo = o3djs.io.loadTexture(pack,
                                      'http://google.com/someimage.jpg',
                                      callback);
function callback(texture, exception) {
  if (!exception) {
    g_mySampler.texture = texture;
  } else {
    alert(exception);
  }
}
```

**Parameters:**

*pack* Pack to load texture into.

*url* URL of texture to load.

*callback* Callback when texture is loaded. It will be passed the texture and an exception on error or null on success.

**Returns:**

[!o3djs.io.LoadInfo](#).A LoadInfo to track progress.

**See Also:**

- [o3djs.io.createLoader](#)

## o3djs.loader Module Reference

[List of all members.](#)

---

### Detailed Description

A Module with a loader class for helping to load multiple assets in an asynchronous manner.

### Source

[o3djs/loader.js](#)

### Public Member Functions

- <static> [!o3djs.loader.Loader createLoader](#)(onFinished)
-

# Member Function Documentation

[!o3djs.loader.Loader](#) o3djs.loader.createLoader ( !function(): void *onFinished* )

Creates a Loader for helping to load a bunch of items asynchronously.

The way you use this is as follows.

```
var loader = o3djs.loader.createLoader(myFinishedCallback);
loader.loadTexture(pack, texture1Url, callbackForTexture);
loader.loadTexture(pack, texture2Url, callbackForTexture);
loader.loadTexture(pack, texture3Url, callbackForTexture);
loader.finish();
```

The loader guarantees that myFinishedCallback will be called after all the items have been loaded.

**Parameters:**

*onFinished* Function to call when final item has loaded.

**Returns:**

[!o3djs.loader.Loader](#).A Loader Object.

# o3djs.material Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for materials.

## Source

[o3djs/material.js](#)

## Public Member Functions

- <static> [attachStandardEffect](#)(pack, material, viewInfo, effectType)
- <static> [bindParams](#)(pack, params)
- <static> [bindParamsOnMaterial](#)(material, params)
- <static> !Object [createAndBindStandardParams](#)(pack)
- <static> ![o3d.Material](#) [createBasicMaterial](#)(pack, viewInfo, colorOrTexture, opt\_transparent)
- <static> ![o3d.Material](#) [createMaterialFromFile](#)(pack, url, drawList)
- <static> !Object [createParams](#)(pack, paramSpec)
- <static> !Object [createStandardParams](#)(pack)
- <static> [prepareMaterial](#)(pack, viewInfo, material, opt\_effectType)
- <static> [prepareMaterials](#)(pack, viewInfo, opt\_effectPack)
- <static> [setDrawListOnMaterials](#)(pack, drawList)

---

# Member Function Documentation

```
o3djs.material.attachStandardEffect ( !o3d.Pack pack
                                      !o3d.Material material
                                      !o3djs.rendergraph.ViewInfo viewInfo
                                      string effectType )
```

Builds a standard effect for a given material. The position of the default light is set to the view position. If the material already has an effect, none is created.

**Parameters:**

*pack* Pack to manage created objects.  
*material* The material for which to create an effect.  
*viewInfo* as returned from o3djs.rendergraph.createView.  
*effectType* Type of effect to create ('phong', 'lambert', 'constant').

**See Also:**

- [o3djs.effect.attachStandardShader](#)

```
o3djs.material.bindParams ( !o3d.Pack pack
                           !Object params )
```

Binds params to all materials in a pack by name.

**Parameters:**

*pack* Pack with materials to bind.  
*params* A object where each property is the name of a param and its value is that param.

```
o3djs.material.bindParamsOnMaterial ( !o3d.Material material
                                       !Object params )
```

Binds params to all materials in a pack by name.

**Parameters:**

*material* Material to bind params on.  
*params* A object where each property is the name of a param and its value is that param.

```
!Object o3djs.material.createAndBindStandardParams ( !o3d.Pack pack )
```

Creates the global params need by the shaders built in effect.js then binds all the matching params on materials in pack to these global params.

The params currently created are 'lightColor' which is a ParamFloat4 and 'lightWorldPos' which is a ParamFloat3. You can set their values like this

```
var params = o3djs.material.createAndBindStandardParams (pack) ;
params.lightColor.value = [1, 0, 0, 1]; // red
params.lightWorldPos.value = [1000, 2000, 3000]; // set light position.
```

**Parameters:**

*pack* Pack to create params in.

## Returns:

`!Object.params` A object where each property is the name of a param and its value is that param.

## See Also:

- [o3djs.material.createParams](#)
- [o3djs.material.bindParams](#)

## [!o3d.Material](#)

`o3djs.material.createBasicMaterial`

( [!o3d.Pack](#) *pack*  
[!o3djs.rendergraph.ViewInfo](#) *viewInfo*  
([!o3djs.math.Vector4](#)!  
[o3d.Texture](#)) *colorOrTexture*  
boolean *opt\_transparent* )

This function creates a basic material for when you just want to get something on the screen quickly without having to manually setup shaders. You can call this function something like.

```
<html><body>
<script type="text/javascript" src="o3djs/all.js">
</script>
<script>
window.onload = init;
function init() {
    o3djs.base.makeClients(initStep2);
}
function initStep2(clientElements) {
    var clientElement = clientElements[0];
    var client = clientElement.client;
    var pack = client.createPack();
    var viewInfo = o3djs.rendergraph.createBasicView(
        pack,
        client.root,
        client.renderGraphRoot);
    var material = o3djs.material.createBasicMaterial(
        pack,
        viewInfo,
        [1, 0, 0, 1]); // red
    var shape = o3djs.primitives.createCube(pack, material, 10);
    var transform = pack.createObject('Transform');
    transform.parent = client.root;
    transform.addShape(shape);
    o3djs.camera.fitContextToScene(client.root,
                                    client.width,
                                    client.height,
                                    viewInfo.drawContext);
}
</script>
<div id="o3d" style="width: 600px; height: 600px"></div>
</body></html>
```

## Parameters:

- pack* Pack to manage created objects.  
*viewInfo* as returned from `o3djs.rendergraph.createBasicView`.  
*colorOrTexture* Either a color in the format [r, g, b, a] or an O3D texture.

*opt\_transparent* Whether or not the material is transparent. Defaults to false.

**Returns:**

[!o3d.Material](#).The created material.

[!o3d.Material](#) o3djs.material.createMaterialFromFile ( [!o3d.Pack](#) *pack*  
*string* *url*  
[!o3d.DrawList](#) *drawList* )

Creates a material for an effect loaded from a file. If the effect has already been loaded in the pack it will be reused.

**Parameters:**

*pack* Pack to create effect in.

*url* Url for effect file.

*drawList* DrawList to assign effect to.

**Returns:**

[!o3d.Material](#).The material.

[!Object](#) o3djs.material.createParams ( [!o3d.Pack](#) *pack*  
[!Object](#) *paramSpec* )

Creates params from a param spec.

**Parameters:**

*pack* Pack to create params in.

*paramSpec* An object where each property is the name of a param and its value is the type of param.

**Returns:**

[!Object.params](#) A object where each property is the name of a param and its value is that param.

[!Object](#) o3djs.material.createStandardParams ( [!o3d.Pack](#) *pack* )

Creates the global params need by the shaders built in effect.js

The params currently created are 'lightColor' which is a ParamFloat4 and 'lightWorldPos' which is a ParamFloat3. You can set their values like this

```
var params = o3djs.material.createStandardParams(pack);
param.lightColor.value = [1, 0, 0, 1]; // red
param.lightWorldPos.value = [1000, 2000, 3000]; // set light position.
```

Note: This function just creates the params. It does not connect them to anything. See o3djs.material.createAndBindStandardParams, o3djs.material.createParams and o3djs.material.bindParams

**Parameters:**

*pack* Pack to create params in.

**Returns:**

[!Object.params](#) A object where each property is the name of a param and its value is that param.

**See Also:**

- [o3djs.material.createAndBindStandardParams](#)
- [o3djs.material.createParams](#)
- [o3djs.material.bindParams](#)

```
o3djs.material.prepareMaterial ( !o3d.Pack           pack
                                !o3djs.rendergraph.ViewInfo viewInfo
                                !o3d.Material          material
                                string                 opt_effectType )
```

Prepares a material by setting their drawList and possibly creating an standard effect if one does not already exist.

This function is very specific to our sample importer. It expects that if no Effect exists on a material that certain extra Params have been created on the Material to give us instructions on what to Effects to create.

#### Parameters:

*pack* Pack to manage created objects.  
*viewInfo* as returned from o3djs.rendergraph.createView.  
*material* to prepare.  
*opt\_effectType* type of effect to create ('phong', 'lambert', 'constant').

#### See Also:

- [o3djs.material.attachStandardEffect](#)

```
o3djs.material.prepareMaterials ( !o3d.Pack           pack
                                 !o3djs.rendergraph.ViewInfo viewInfo
                                 !o3d.Pack            opt_effectPack )
```

Prepares all the materials in the given pack by setting their drawList and if they don't have an Effect, creating one for them.

This function is very specific to our sample importer. It expects that if no Effect exists on a material that certain extra Params have been created on the Material to give us instructions on what to Effects to create.

#### Parameters:

*pack* Pack to prepare.  
*viewInfo* as returned from o3djs.rendergraph.createView.  
*opt\_effectPack* Pack to create effects in. If this is not specified the pack to prepare above will be used.

#### See Also:

- [o3djs.material.prepareMaterial](#)

```
o3djs.material.setDrawListOnMaterials ( !o3d.Pack      pack
                                         !o3d.DrawList drawList )
```

Prepares all the materials in the given pack by setting their drawList.

#### Parameters:

*pack* Pack to manage created objects.

*drawList* DrawList to assign to materials.

# o3djs.math Module Reference

[List of all members.](#)

---

## Detailed Description

A module for math for o3djs.math.

## Source

[o3djs/math.js](#)

## Public Member Functions

- <static> ![o3djs.math.Matrix addMatrix](#)(a, b)
- <static> ![o3djs.math.Vector addVector](#)(a, b)
- <static> number [codet](#)(a, x, y)
- <static> ![o3djs.math.Matrix copyMatrix](#)(m)
- <static> number [copyScalar](#)(a)
- <static> ![o3djs.math.Vector copyVector](#)(v)
- <static> ![o3djs.math.Vector cross](#)(a, b)
- <static> number [degToRad](#)(degrees)
- <static> number [det](#)(m)
- <static> number [det1](#)(m)
- <static> number [det2](#)(m)
- <static> number [det3](#)(m)
- <static> number [det4](#)(m)
- <static> number [distance](#)(a, b)
- <static> number [distanceSquared](#)(a, b)
- <static> ![o3djs.math.Matrix divMatrixScalar](#)(m, k)
- <static> ![o3djs.math.Vector divVectorScalar](#)(v, k)
- <static> ![o3djs.math.Vector divVectorVector](#)(a, b)
- <static> number [dot](#)(a, b)
- <static> ![o3djs.math.Matrix identity](#)(n)
- <static> [installColumnMajorFunctions](#)()
- <static> [installErrorCheckFreeFunctions](#)()
- <static> [installErrorCheckFunctions](#)()
- <static> [installRowMajorFunctions](#)()
- <static> ![o3djs.math.Matrix inverse](#)(m)
- <static> ![o3djs.math.Matrix1 inverse1](#)(m)
- <static> ![o3djs.math.Matrix2 inverse2](#)(m)
- <static> ![o3djs.math.Matrix3 inverse3](#)(m)
- <static> ![o3djs.math.Matrix4 inverse4](#)(m)
- <static> number [length](#)(a)

- <static> number [lengthSquared\(a\)](#)
- <static> number [lerpCircular\(a, b, t, range\)](#)
- <static> ![o3djs.math.Matrix](#) [lerpMatrix\(a, b, t\)](#)
- <static> number [lerpRadian\(a, b, t\)](#)
- <static> number [lerpScalar\(a, b, t\)](#)
- <static> ![o3djs.math.Vector](#) [lerpVector\(a, b, t\)](#)
- <static> number [modClamp\(v, range, opt\\_rangeStart\)](#)
- <static> ![o3djs.math.Matrix](#) [mulMatrixScalar\(m, k\)](#)
- <static> ![o3djs.math.Matrix](#) [mulScalarMatrix\(k, m\)](#)
- <static> number [mulScalarScalar\(a, b\)](#)
- <static> ![o3djs.math.Vector](#) [mulScalarVector\(k, v\)](#)
- <static> ![o3djs.math.Vector](#) [mulVectorScalar\(v, k\)](#)
- <static> ![o3djs.math.Vector](#) [mulVectorVector\(a, b\)](#)
- <static> ![o3djs.math.Matrix](#) [negativeMatrix\(m\)](#)
- <static> number [negativeScalar\(a\)](#)
- <static> ![o3djs.math.Vector](#) [negativeVector\(v\)](#)
- <static> ![o3djs.math.Vector](#) [normalize\(a\)](#)
- <static> ![o3djs.math.Matrix](#) [orthonormalize\(m\)](#)
- <static> number [radToDeg\(radians\)](#)
- <static> ![o3djs.math.Matrix](#) [subMatrix\(a, b\)](#)
- <static> ![o3djs.math.Vector](#) [subVector\(a, b\)](#)
- <static> number [trace\(m\)](#)
- <static> ![o3djs.math.Matrix](#) [transpose\(m\)](#)

## Public Properties

- <static> ![o3djs.math.Vector](#) [column](#)
- <static> undefined [columnMajor](#)
- <static> undefined [errorCheck](#)
- <static> undefined [errorCheckFree](#)
- <static> (!Array.<!Array.<number>>|![o3d.Matrix4](#)) [Matrix](#)
- <static> !Array.<!Array.<number>> [Matrix1](#)
- <static> !Array.<!Array.<number>> [Matrix2](#)
- <static> !Array.<!Array.<number>> [Matrix3](#)
- <static> (!Array.<!Array.<number>>|![o3d.Matrix4](#)) [Matrix4](#)
- <static> undefined [matrix4](#)
- <static> ![o3djs.math.Matrix](#) [mulMatrixMatrix](#)
- <static> ![o3djs.math.Matrix2](#) [mulMatrixMatrix2](#)
- <static> ![o3djs.math.Matrix3](#) [mulMatrixMatrix3](#)
- <static> ![o3djs.math.Matrix4](#) [mulMatrixMatrix4](#)
- <static> ![o3djs.math.Vector](#) [mulMatrixVector](#)
- <static> ![o3djs.math.Vector](#) [mulVectorMatrix](#)
- <static> ![o3djs.math.Vector](#) [row](#)
- <static> undefined [rowMajor](#)
- <static> !Array.<number> [Vector](#)
- <static> (!Array.<number>|![o3d.Float2](#)) [Vector2](#)
- <static> (!Array.<number>|![o3d.Float3](#)) [Vector3](#)
- <static> (!Array.<number>|![o3d.Float4](#)) [Vector4](#)

---

## Member Function Documentation

[!o3djs.math.Matrix](#) o3djs.math.addMatrix ( [!o3djs.math.Matrix](#) *a*  
                                 [!o3djs.math.Matrix](#) *b* )

Adds two matrices; assumes a and b are the same size.

**Parameters:**

*a* Operand matrix.  
  *b* Operand matrix.

**Returns:**

[!o3djs.math.Matrix](#).The sum of a and b.

[!o3djs.math.Vector](#) o3djs.math.addVector ( [!o3djs.math.Vector](#) *a*  
                                 [!o3djs.math.Vector](#) *b* )

Adds two vectors; assumes a and b have the same dimension.

**Parameters:**

*a* Operand vector.  
  *b* Operand vector.

**Returns:**

[!o3djs.math.Vector](#).The sum of a and b.

number o3djs.math.codet ( [!o3djs.math.Matrix](#) *a*  
                                 number                   *x*  
                                 number                   *y* )

Computes the determinant of the cofactor matrix obtained by removal of a specified row and column.  
This is a helper function for the general determinant and matrix inversion functions.

**Parameters:**

*a* The original matrix.  
  *x* The row to be removed.  
  *y* The column to be removed.

**Returns:**

  number.The determinant of the matrix obtained by removing row *x* and column *y* from *a*.

[!o3djs.math.Matrix](#) o3djs.math.copyMatrix ( [!o3djs.math.Matrix](#) *m* )

Copies a matrix.

**Parameters:**

*m* The matrix.

**Returns:**

[!o3djs.math.Matrix](#).A copy of *m*.

number o3djs.math.copyScalar ( number *a* )

Copies a scalar.

**Parameters:**

*a* The scalar.

**Returns:**

number.a.

[!o3djs.math.Vector](#) o3djs.math.copyVector ( [!o3djs.math.Vector](#) *v* )

Copies a vector.

**Parameters:**

*v* The vector.

**Returns:**

[!o3djs.math.Vector](#).A copy of v.

[!o3djs.math.Vector](#) o3djs.math.cross ( [!o3djs.math.Vector](#) *a*

[!o3djs.math.Vector](#) *b* )

Computes the cross product of two vectors; assumes both vectors have three entries.

**Parameters:**

*a* Operand vector.

*b* Operand vector.

**Returns:**

[!o3djs.math.Vector](#).The vector a cross b.

number o3djs.math.degToRad ( number *degrees* )

Converts degrees to radians.

**Parameters:**

*degrees* A value in degrees.

**Returns:**

number.the value in radians.

number o3djs.math.det ( [!o3djs.math.Matrix](#) *m* )

Computes the determinant of an arbitrary square matrix.

**Parameters:**

*m* The matrix.

**Returns:**

number.the determinant of m.

number o3djs.math.det1 ( [!o3djs.math.Matrix1](#) *m* )

Computes the determinant of a 1-by-1 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

number.The determinant of m.

number o3djs.math.det2 ( [!o3djs.math.Matrix2](#) *m* )

Computes the determinant of a 2-by-2 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

number.The determinant of m.

number o3djs.math.det3 ( [!o3djs.math.Matrix3](#) *m* )

Computes the determinant of a 3-by-3 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

number.The determinant of m.

number o3djs.math.det4 ( [!o3djs.math.Matrix4](#) *m* )

Computes the determinant of a 4-by-4 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

number.The determinant of m.

number o3djs.math.distance ( [!o3djs.math.Vector](#) *a*

[!o3djs.math.Vector](#) *b* )

Computes the Euclidean distance between two vectors.

**Parameters:**

*a* A vector.

*b* A vector.

**Returns:**

number.The distance between a and b.

number o3djs.math.distanceSquared ( [!o3djs.math.Vector](#) *a*

[!o3djs.math.Vector](#) *b* )

Computes the square of the Euclidean distance between two vectors.

**Parameters:**

*a* A vector.

*b* A vector.

**Returns:**

number.The distance between a and b.

[!o3djs.math.Matrix](#) o3djs.math.divMatrixScalar ( [!o3djs.math.Matrix](#) *m*  
number *k* )

Divides a matrix by a scalar.

**Parameters:**

*m* The matrix.

*k* The scalar.

**Returns:**

[!o3djs.math.Matrix](#).The matrix m divided by k.

[!o3djs.math.Vector](#) o3djs.math.divVectorScalar ( [!o3djs.math.Vector](#) v  
number k )

Divides a vector by a scalar.

**Parameters:**

v The vector.

k The scalar.

**Returns:**

[!o3djs.math.Vector](#).v The vector v divided by k.

[!o3djs.math.Vector](#) o3djs.math.divVectorVector ( [!o3djs.math.Vector](#) a  
[!o3djs.math.Vector](#) b )

Divides a vector by another vector (component-wise); assumes a and b have the same length.

**Parameters:**

a Operand vector.

b Operand vector.

**Returns:**

[!o3djs.math.Vector](#).The vector of quotients of entries of a and b.

number o3djs.math.dot ( [!o3djs.math.Vector](#) a  
[!o3djs.math.Vector](#) b )

Computes the dot product of two vectors; assumes that a and b have the same dimension.

**Parameters:**

a Operand vector.

b Operand vector.

**Returns:**

number.The dot product of a and b.

[!o3djs.math.Matrix](#) o3djs.math.identity ( number n )

Creates an n-by-n identity matrix.

**Parameters:**

n The dimension of the identity matrix required.

**Returns:**

[!o3djs.math.Matrix](#).An n-by-n identity matrix.

o3djs.math.installColumnMajorFunctions ( )

Sets each function in the namespace o3djs.math to the column major version in o3djs.math.columnMajor (provided such a function exists in o3djs.math.columnMajor). Call this function to establish the column major convention.

o3djs.math.installErrorCheckFreeFunctions ( )

Sets each function in the namespace o3djs.math to the error checking free version in o3djs.math.errorCheckFree (provided such a function exists in o3djs.math.errorCheckFree).

`o3djs.math.installErrorCheckFunctions ( )`

Sets each function in the namespace o3djs.math to the error checking version in o3djs.math.errorCheck (provided such a function exists in o3djs.math.errorCheck).

`o3djs.math.installRowMajorFunctions ( )`

Sets each function in the namespace o3djs.math to the row major version in o3djs.math.rowMajor (provided such a function exists in o3djs.math.rowMajor). Call this function to establish the row major convention.

**`!o3djs.math.Matrix`** `o3djs.math.inverse ( !o3djs.math.Matrix m )`

Computes the inverse of an arbitrary square matrix.

**Parameters:**

*m* The matrix.

**Returns:**

**`!o3djs.math.Matrix`**. The inverse of *m*.

**`!o3djs.math.Matrix1`** `o3djs.math.inverse1 ( !o3djs.math.Matrix1 m )`

Computes the inverse of a 1-by-1 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

**`!o3djs.math.Matrix1`**. The inverse of *m*.

**`!o3djs.math.Matrix2`** `o3djs.math.inverse2 ( !o3djs.math.Matrix2 m )`

Computes the inverse of a 2-by-2 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

**`!o3djs.math.Matrix2`**. The inverse of *m*.

**`!o3djs.math.Matrix3`** `o3djs.math.inverse3 ( !o3djs.math.Matrix3 m )`

Computes the inverse of a 3-by-3 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

**`!o3djs.math.Matrix3`**. The inverse of *m*.

**`!o3djs.math.Matrix4`** `o3djs.math.inverse4 ( !o3djs.math.Matrix4 m )`

Computes the inverse of a 4-by-4 matrix.

**Parameters:**

*m* The matrix.

**Returns:**

**`!o3djs.math.Matrix4`**. The inverse of *m*.

number `o3djs.math.length ( !o3djs.math.Vector a )`

Computes the Euclidean length of a vector, i.e. the square root of the sum of the squares of the entries.

**Parameters:**

*a* The vector.

**Returns:**

number.The length of a.

number o3djs.math.lengthSquared ( [!o3djs.math.Vector](#) *a* )

Computes the square of the Euclidean length of a vector, i.e. the sum of the squares of the entries.

**Parameters:**

*a* The vector.

**Returns:**

number.The square of the length of a.

number o3djs.math.lerpCircular ( number *a*

                  number *b*

                  number *t*

                  number *range* )

Lerps in a circle. Does a lerp between a and b but inside range so for example if range is 100, a is 95 and b is 5 lerping will go in the positive direction.

**Parameters:**

*a* Start value.

*b* Target value.

*t* Amount to lerp (0 to 1).

*range* Range of circle.

**Returns:**

number.lerped result.

[!o3djs.math.Matrix](#) o3djs.math.lerpMatrix ( [!o3djs.math.Matrix](#) *a*

[!o3djs.math.Matrix](#) *b*

                  number                   *t* )

Performs linear interpolation on two matrices. Given matrices a and b and interpolation coefficient t, returns  $(1 - t) * a + t * b$ .

**Parameters:**

*a* Operand matrix.

*b* Operand matrix.

*t* Interpolation coefficient.

**Returns:**

[!o3djs.math.Matrix](#).The weighted of a and b.

number o3djs.math.lerpRadian ( number *a*

                  number *b*

                  number *t* )

Lerps radians.

## Parameters:

- $a$  Start value.
- $b$  Target value.
- $t$  Amount to lerp (0 to 1).

## Returns:

number.lerped result.

```
number o3djs.math.lerpScalar ( number a  
                           number b  
                           number t )
```

Performs linear interpolation on two scalars. Given scalars a and b and interpolation coefficient t, returns  $(1 - t) * a + t * b$ .

## Parameters:

- a* Operand scalar.
- b* Operand scalar.
- t* Interpolation coefficient.

## Returns:

number. The weighted sum of a and b.

[!o3djs.math.Vector](#) o3djs.math.lerpVector ( [!o3djs.math.Vector](#) *a*  
[!o3djs.math.Vector](#) *b*  
number *t* )

Performs linear interpolation on two vectors. Given vectors  $a$  and  $b$  and interpolation coefficient  $t$ , returns  $(1 - t) * a + t * b$ .

### Parameters:

- a* Operand vector.
- b* Operand vector.
- t* Interpolation coefficient.

### Returns:

`!o3djs.math.Vector`. The weighted sum of a and b.

Clamps a value between 0 and range using a modulo.

### Parameters:

*v* Value to clamp mod.  
*range* Range to clamp to.  
*opt rangeStart* start of range. Default = 0.

## Returns:

number.Clamp modded value.

Multiplies a matrix by a scalar.

## Parameters:

$m$  The matrix.  
 $k$  The scalar.

## Returns:

!o3djs.math.Matrix. The product of m and k.

Multiplies a scalar by a matrix.

## Parameters:

- $k$  The scalar.
- $m$  The matrix.

## Returns:

`!o3djs.math.Matrix`. The product of m and k.

Multiplies two scalars.

### Parameters:

*a* Operand scalar.  
*b* Operand scalar.

## Returns:

number. The product of a and b,

Multiplies a scalar by a vector.

## Parameters:

- $k$  The scalar.
- $v$  The vector.

## Returns:

[!o3djs.math.Vector](#). The product of k and v.

Multiplies a vector by a scalar.

## Parameters:

$v$  The vector.  
 $k$  The scalar.

### Returns:

`!o3djs.math.Vector`. The product of k and v.

`!o3djs.math.Vector o3djs.math.mulVectorVector ( !o3djs.math.Vector a`

[!o3djs.math.Vector](#) *b* )

Multiplies a vector by another vector (component-wise); assumes a and b have the same length.

**Parameters:**

*a* Operand vector.

*b* Operand vector.

**Returns:**

[!o3djs.math.Vector](#).The vector of products of entries of a and b.

[!o3djs.math.Matrix](#) o3djs.math.negativeMatrix ( [!o3djs.math.Matrix](#) *m* )

Negates a matrix.

**Parameters:**

*m* The matrix.

**Returns:**

[!o3djs.math.Matrix](#).-*m*.

number o3djs.math.negativeScalar ( number *a* )

Negates a scalar.

**Parameters:**

*a* The scalar.

**Returns:**

number.-*a*.

[!o3djs.math.Vector](#) o3djs.math.negativeVector ( [!o3djs.math.Vector](#) *v* )

Negates a vector.

**Parameters:**

*v* The vector.

**Returns:**

[!o3djs.math.Vector](#).-*v*.

[!o3djs.math.Vector](#) o3djs.math.normalize ( [!o3djs.math.Vector](#) *a* )

Divides a vector by its Euclidean length and returns the quotient.

**Parameters:**

*a* The vector.

**Returns:**

[!o3djs.math.Vector](#).The normalized vector.

[!o3djs.math.Matrix](#) o3djs.math.orthonormalize ( [!o3djs.math.Matrix](#) *m* )

Performs Graham-Schmidt orthogonalization on the vectors which make up the given matrix and returns the result in the rows of a new matrix. When multiplying many orthogonal matrices together, errors can accumulate causing the product to fail to be orthogonal. This function can be used to correct that.

**Parameters:**

*m* The matrix.

**Returns:**

[!o3djs.math.Matrix](#).A matrix whose rows are obtained from the rows of m by the Graham-Schmidt process.

number o3djs.math.radToDeg ( number *radians* )

Converts radians to degrees.

**Parameters:**

*radians* A value in radians.

**Returns:**

number.the value in degrees.

[!o3djs.math.Matrix](#) o3djs.math.subMatrix ( [!o3djs.math.Matrix](#) *a*  
[!o3djs.math.Matrix](#) *b* )

Subtracts two matrices; assumes a and b are the same size.

**Parameters:**

*a* Operand matrix.

*b* Operand matrix.

**Returns:**

[!o3djs.math.Matrix](#).The sum of a and b.

[!o3djs.math.Vector](#) o3djs.math.subVector ( [!o3djs.math.Vector](#) *a*  
[!o3djs.math.Vector](#) *b* )

Subtracts two vectors.

**Parameters:**

*a* Operand vector.

*b* Operand vector.

**Returns:**

[!o3djs.math.Vector](#).The difference of a and b.

number o3djs.math.trace ( [!o3djs.math.Matrix](#) *m* )

Computes the trace (sum of the diagonal entries) of a square matrix; assumes m is square.

**Parameters:**

*m* The matrix.

**Returns:**

number.The trace of m.

[!o3djs.math.Matrix](#) o3djs.math.transpose ( [!o3djs.math.Matrix](#) *m* )

Takes the transpose of a matrix.

**Parameters:**

*m* The matrix.

**Returns:**

[!o3djs.math.Matrix](#).The transpose of m.

---

# Member Property Documentation

[!o3djs.math.Vector](#) o3djs.math.column

Gets the jth column of the given matrix m.

undefined o3djs.math.columnMajor

Functions that are specifically column major are kept in their own namespace.

undefined o3djs.math.errorCheck

Functions that do error checking are stored in their own namespace.

undefined o3djs.math.errorCheckFree

Functions that do no error checking and have a separate version that does in o3djs.math.errorCheck are stored in their own namespace.

(!Array.<!Array.<number>>|[!o3d.Matrix4](#)) o3djs.math.Matrix

A arbitrary size Matrix of floats

!Array.<!Array.<number>> o3djs.math.Matrix1

A 1x1 Matrix of floats

!Array.<!Array.<number>> o3djs.math.Matrix2

A 2x2 Matrix of floats

!Array.<!Array.<number>> o3djs.math.Matrix3

A 3x3 Matrix of floats

(!Array.<!Array.<number>>|[!o3d.Matrix4](#)) o3djs.math.Matrix4

A 4x4 Matrix of floats

undefined o3djs.math.matrix4

Functions which deal with 4-by-4 transformation matrices are kept in their own namespaces.

[!o3djs.math.Matrix](#) o3djs.math.mulMatrixMatrix

Multiplies two matrices; assumes that the sizes of the matrices are appropriately compatible.

[!o3djs.math.Matrix2](#) o3djs.math.mulMatrixMatrix2

Multiplies two 2-by-2 matrices.

[!o3djs.math.Matrix3](#) o3djs.math.mulMatrixMatrix3

Multiplies two 3-by-3 matrices; assumes that the given matrices are 3-by-3.

[!o3djs.math.Matrix4](#) o3djs.math.mulMatrixMatrix4

Multiplies two 4-by-4 matrices; assumes that the given matrices are 4-by-4.

[!o3djs.math.Vector](#) o3djs.math.mulMatrixVector

Multiplies a matrix by a vector; treats the vector as a column vector.

[!o3djs.math.Vector](#) o3djs.math.mulVectorMatrix

Multiplies a vector by a matrix; treats the vector as a row vector.

[!o3djs.math.Vector](#) o3djs.math.row

Gets the ith row of the given matrix m.

undefined o3djs.math.rowMajor

Functions that are specifically row major are kept in their own namespace.

[!Array.<number>](#) o3djs.math.Vector

An Array of floats.

[\(!Array.<number>|!o3d.Float2\)](#) o3djs.math.Vector2

An Array of 2 floats

[\(!Array.<number>|!o3d.Float3\)](#) o3djs.math.Vector3

An Array of 3 floats

[\(!Array.<number>|!o3d.Float4\)](#) o3djs.math.Vector4

An Array of 4 floats

## o3djs.pack Module Reference

[List of all members.](#)

---

### Detailed Description

A Module with utilities for dealing with packs..

### Source

[o3djs/pack.js](#)

### Public Member Functions

- <static> [preparePack](#)(pack, viewInfo, opt\_effectPack)
- 

### Member Function Documentation

o3djs.pack.preparePack ( [!o3d.Pack](#) *pack*  
[!o3djs.rendergraph.ViewInfo](#) *viewInfo*  
[!o3d.Pack](#) *opt\_effectPack* )

Prepares a pack for rendering.

#### Parameters:

*pack* Pack to prepare.  
*viewInfo* as returned from o3djs.rendergraph.createView.

*opt\_effectPack* Pack to create effects in. If this is not specified the pack to prepare above will be used.

#### See Also:

- [o3djs.material.prepareMaterials](#)
  - [o3djs.shape.prepareShapes](#)

# **o3djs.particles Module Reference**

## List of all members.

## Detailed Description

A Module with various GPU particle functions and classes. Note: GPU particles have the issue that they are not sorted per particle but rather per emitter.

## Source

## o3djs/particles.js

## Public Member Functions

- <static> [!o3djs.particles.ParticleSystem](#) [createParticleSystem](#)(pack, viewInfo, opt\_clockParam, opt\_randomFunction)

## Public Properties

- <static> !Array.<{ {name: string, fxString: string} }> [FX\\_STRINGS](#)
  - <static> undefined [ParticleStateIds](#)

# Member Function Documentation

[o3djs.particles.ParticleSystem](#)  
o3djs.particles.createParticleSystem

( <a href="#">!o3d.Pack</a>	<i>pack</i>
!	
<a href="#">o3djs.rendergraph.viewInfo</a>	<i>viewInfo</i>
<a href="#">!o3d.ParamFloat</a>	<i>opt_clockParam</i>
<a href="#">!function(): number</a>	<i>opt_randomFunction</i>
	<i>on</i>

Creates a particle system. You only need one of these to run multiple emitters of different types of particles.

### Parameters:

<i>pack</i>	The pack for the particle system to manage resources.
<i>viewInfo</i>	A viewInfo so the particle system can do the default setup. The only thing used from viewInfo is the zOrderedDrawList. If that is not where you want your particles, after you create the particleEmitter use particleEmitter.material.drawList = myDrawList to set it to something else.
<i>opt_clockParam</i>	A ParamFloat to be the default clock for emitters of this particle system.
<i>opt_randomFunc</i>	A function that returns a random number between 0.0 and 1.0. This allows you to pass in a pseudo random function if you need particles that are reproducible.

#### Returns:

[!o3djs.particles.ParticleSystem](#).The created particle system.

---

## Member Property Documentation

`!Array.<{name: string, fxString: string}>` o3djs.particles.FX\_STRINGS

Particle Effect strings

`undefined` o3djs.particles.ParticleStateIds

Enum for pre-made particle states.

## o3djs.picking Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for picking.

## Source

[o3djs/picking.js](#)

## Public Member Functions

- <static> [!o3djs.picking.Ray clientPositionToWorldRay](#)(clientXPosition, clientYPosition, drawContext, clientWidth, clientHeight)
- <static> [!o3djs.picking.Ray clientPositionToWorldRayEx](#)(clientXPosition, clientYPosition, view, projection, clientWidth, clientHeight)
- <static> [!o3djs.picking.PickInfo createPickInfo](#)(shapeInfo, rayIntersectionInfo, worldIntersectionPosition)
- <static> [!o3djs.picking.ShapeInfo createShapeInfo](#)(shape, parent)
- <static> [!o3djs.picking.TransformInfo createTransformInfo](#)(transform, parent)
- <static> [dprint](#)(msg)

- <static> [dprintBoundingBox](#)(label, boundingBox, opt\_prefix)
- <static> [dprintPoint3](#)(label, float3, prefix)
- <static> [dumpRayIntersectionInfo](#)(label, rayIntersectionInfo)

## Public Properties

- <static> {{near: [!o3djs.math.Vector3](#), far: [!o3djs.math.Vector3](#)}} [Ray](#)
- 

## Member Function Documentation

[!o3djs.picking.Ray](#) o3djs.picking.clientPositionToWorldRay ( number *clientXPosition*  
                           number *clientYPosition*  
                           [!o3d.DrawContext](#) *drawContext*  
                           number *clientWidth*  
                           number *clientHeight* )

Convert a pixel position relative to the top left corner of the client area into the corresponding ray through the frustum in world space.

### Parameters:

*clientXPosition* x position relative to client area.  
*clientYPosition* y position relative to client area.  
*drawContext* DrawContext to get view and projection matrices from.  
*clientWidth* width of client area.  
*clientHeight* height of client area.

### Returns:

[!o3djs.picking.Ray](#).ray in world space.

[!o3djs.picking.Ray](#)  
 o3djs.picking.clientPositionToWorldRayEx ( number *clientXPosition*  
                           number *clientYPosition*  
                           [!o3djs.math.Matrix4](#) *view*  
                           [!o3djs.math.Matrix4](#) *projection*  
                           number *clientWidth*  
                           number *clientHeight* )

Convert a pixel position relative to the top left corner of the client area into the corresponding ray through the frustum in world space.

### Parameters:

*clientXPosition* x position relative to client area.  
*clientYPosition* y position relative to client area.  
*view* View matrix to transform with.  
*projection* Projection matrix to transform with.  
*clientWidth* width of client area.  
*clientHeight* height of client area.

### Returns:

[!o3djs.picking.Ray](#).ray in world space.

[!o3djs.picking.PickInfo](#)  
o3djs.picking.createPickInfo  
( [!o3djs.picking.ShapeInfo](#) *shapeInfo*  
[!o3d.RayIntersectionInfo](#) *rayIntersectionInfo*  
[!o3djs.math.Vector3](#) *worldIntersectionPosition* )

Creates a new PickInfo.

**Parameters:**

*shapeInfo* The ShapeInfo that was picked.  
*rayIntersectionInfo* Information about the pick.  
*worldIntersectionPosition* world position of intersection.

**Returns:**

[!o3djs.picking.PickInfo](#).The new PickInfo.

[!o3djs.picking.ShapeInfo](#) o3djs.picking.createShapeInfo ( [!o3d.Shape](#) *shape*  
[!o3djs.picking.TransformInfo](#) *parent* )

Creates a new ShapeInfo.

**Parameters:**

*shape* Shape to keep info about.  
*parent* Parent transform of the shape.

**Returns:**

[!o3djs.picking.ShapeInfo](#).The new ShapeInfo.

[!o3djs.picking.TransformInfo](#)  
o3djs.picking.createTransformInfo  
( [!o3d.Transform](#) *transform*  
[!o3djs.picking.TransformInfo](#) *parent* )

Creates a new TransformInfo.

**Parameters:**

*transform* Transform to keep info about.  
*parent* Parent transform of the transform. Can be null.

**Returns:**

[!o3djs.picking.TransformInfo](#).The new TransformInfo.

o3djs.picking.dprint ( string *msg* )

A local dump function so we can easily comment it out.

**Parameters:**

*msg* Message to dump.

o3djs.picking.dprintBoundingBox ( string *label*  
[!o3d.BoundingBox](#) *boundingBox*  
string *opt\_prefix* )

A local dump function so we can easily comment it out.

**Parameters:**

*label* Label to put in front of dump.

A local dump function so we can easily comment it out.

## Parameters:

*label* Label to print before value.

*float3* Value to print.

*prefix* optional prefix for indenting.

```
o3djs.picking.dumpRayIntersectionInfo ( string label  
                                         !o3d.RayIntersectionInfo rayIntersectionInfo )
```

A local dump function so we can easily comment it out.

## Parameters:

*label* Label to print before value.

*rayIntersectionInfo* Value to print.

# Member Property Documentation

```
{near: !o3djs.math.Vector3, far: !o3djs.math.Vector3} o3djs.picking.Ray
```

A ray.

# o3djs.primitives Module Reference

## List of all members.

## Detailed Description

## A Module for creating primitives.

## Source

## o3djs/primitives.js

# Public Member Functions

- <static> [!o3d.Shape](#) [createBox](#)(pack, material, width, height, depth, opt\_matrix)
  - <static> [!o3d.Shape](#) [createCube](#)(pack, material, size, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createCubeVertices](#)(size, opt\_matrix)
  - <static> [!o3d.Shape](#) [createCylinder](#)(pack, material, radius, depth, radialSubdivisions, verticalSubdivisions, opt\_matrix)
  - <static> [o3djs.primitives.VertexInfo](#) [createCylinderVertices](#)(radius, height, radialSubdivisions,

- verticalSubdivisions, opt\_matrix)
  - <static> [!o3d.Shape](#) [createDisc](#)(pack, material, radius, divisions, stacks, startStack, stackPower, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createDiscVertices](#)(radius, divisions, opt\_stacks, opt\_startStack, opt\_stackPower, opt\_matrix)
  - <static> [!o3d.Shape](#) [createFadePlane](#)(pack, material, width, depth, subdivisionsWidth, subdivisionsDepth, opt\_matrix)
  - <static> [!o3d.Shape](#) [createPlane](#)(pack, material, width, depth, subdivisionsWidth, subdivisionsDepth, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createPlaneVertices](#)(width, depth, subdivisionsWidth, subdivisionsDepth, opt\_matrix)
  - <static> [!o3d.Shape](#) [createPrism](#)(pack, material, points, depth, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createPrismVertices](#)(points, depth, opt\_matrix)
  - <static> [!o3d.Shape](#) [createRainbowCube](#)(pack, material, size, opt\_matrix)
  - <static> [!o3d.Shape](#) [createSphere](#)(pack, material, radius, subdivisionsAxis, subdivisionsHeight, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createSphereVertices](#)(radius, subdivisionsAxis, subdivisionsHeight, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createVertexInfo\(\)](#)
  - <static> [!o3djs.primitives.VertexStreamInfo](#) [createVertexStreamInfo](#)(numComponents, semantic, opt\_semanticIndex)
  - <static> [!o3d.Shape](#) [createWedge](#)(pack, material, points, depth, opt\_matrix)
  - <static> [!o3djs.primitives.VertexInfo](#) [createWedgeVertices](#)(inPoints, depth, opt\_matrix)
  - <static> [setCullingInfo](#)(primitive)
- 

## Member Function Documentation

[!o3d.Shape](#) o3djs.primitives.createBox ( [!o3d.Pack](#) *pack*  
                   [!o3d.Material](#)      *material*  
                   number           *width*  
                   number           *height*  
                   number           *depth*  
                   [!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates a box. The box will be created around the origin. The created box has position, normal and uv streams.

### Parameters:

<i>pack</i>	Pack to create Box elements in.
<i>material</i>	to use.
<i>width</i>	Width of the box.
<i>height</i>	Height of the box.
<i>depth</i>	Depth of the box.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

### Returns:

[!o3d.Shape](#).The created Box.

## See Also:

- [o3d.Pack](#)
- [o3d.Shape](#)

[!o3d.Shape](#) o3djs.primitives.createCube ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material*  
*number* *size*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates a cube. The cube will be created around the origin. (-size / 2, size / 2) The created cube has position, normal and uv streams.

## Parameters:

*pack* Pack to create cube elements in.  
*material* to use.  
*size* Width, height and depth of the cube.  
*opt\_matrix* A matrix by which to multiply all the vertices.

## Returns:

[!o3d.Shape](#).The created cube.

## See Also:

- [o3d.Pack](#)
- [o3d.Shape](#)

[!o3djs.primitives.VertexInfo](#)  
o3djs.primitives.createCubeVertices ( *number* *size*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates the vertices and indices for a cube. The cube will be created around the origin. (-size / 2, size / 2) The created cube has position, normal and uv streams.

## Parameters:

*size* Width, height and depth of the cube.  
*opt\_matrix* A matrix by which to multiply all the vertices.

## Returns:

[!o3djs.primitives.VertexInfo](#).The created cube vertices.

[!o3d.Shape](#) o3djs.primitives.createCylinder ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material*  
*number* *radius*  
*number* *depth*  
*number* *radialSubdivisions*  
*number* *verticalSubdivisions*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates cylinder a cylinder shape. The cylinder will be created around the origin along the y-axis. The created cylinder has position, normal and uv streams.

**Parameters:**

<i>pack</i>	Pack to create cylinder elements in.
<i>material</i>	to use.
<i>radius</i>	Radius of cylinder.
<i>depth</i>	Depth of cylinder.
<i>radialSubdivisions</i>	The number of subdivisions around the cylinder.
<i>verticalSubdivisions</i>	The number of subdivisions down the cylinder.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created cylinder.

[o3djs.primitives.VertexInfo](#)

`o3djs.primitives.createCylinderVertices`

(	number	<i>radius</i>
	number	<i>height</i>
	number	<i>radialSubdivisions</i>
	number	<i>verticalSubdivisions</i>
	<a href="#">!o3djs.math.Matrix4</a> <i>opt_matrix</i>	)

Creates cylinder vertices. The cylinder will be created around the origin along the y-axis. The created cylinder has position, normal and uv streams.

**Parameters:**

<i>radius</i>	Radius of cylinder.
<i>height</i>	Height of cylinder.
<i>radialSubdivisions</i>	The number of subdivisions around the cylinder.
<i>verticalSubdivisions</i>	The number of subdivisions down the cylinder.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[o3djs.primitives.VertexInfo](#).The created cylinder vertices.

[!o3d.Shape](#) `o3djs.primitives.createDisc`

(	<a href="#">!o3d.Pack</a>	<i>pack</i>
	<a href="#">!o3d.Material</a>	<i>material</i>
	number	<i>radius</i>
	number	<i>divisions</i>
	number	<i>stacks</i>
	number	<i>startStack</i>
	number	<i>stackPower</i>
	<a href="#">!o3djs.math.Matrix4</a> <i>opt_matrix</i>	)

Creates a disc shape. The disc will be in the xz plane, centered at the origin. When creating, at least 3 divisions, or pie pieces, need to be specified, otherwise the triangles making up the disc will be degenerate. You can also specify the number of radial pieces (*opt\_stacks*). A value of 1 for *opt\_stacks* will give you a simple disc of pie pieces. If you want to create an annulus by omitting some of the center stacks, you can specify the stack at which to start creating triangles. Finally, *stackPower* allows you to have the widths increase or decrease as you move away from the center. This is particularly useful when using the disc as a ground plane with a fixed camera such that you don't need the resolution of small triangles near the perimeter. For example, a value of 2 will produce stacks whose outside radius increases with the square of the stack index. A value of 1 will give uniform stacks.

**Parameters:**

*pack* Pack to create disc elements in.  
*material* to use.  
*radius* Radius of the disc.  
*divisions* Number of triangles in the disc (at least 3).  
*stacks* Number of radial divisions.  
*startStack* Which radial division to start dividing at.  
*stackPower* Power to raise stack size to for decreasing width.  
*opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created disc.

**See Also:**

- [o3d.Pack](#)
- [o3d.Shape](#)

[!o3djs.primitives.VertexInfo](#)

`o3djs.primitives.createDiscVertices`

(	number	<i>radius</i>
number		<i>divisions</i>
number		<i>opt_stacks</i>
number		<i>opt_startStack</i>
number		<i>opt_stackPower</i>
		<i>r</i>

[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates disc vertices. The disc will be in the xz plane, centered at the origin. When creating, at least 3 divisions, or pie pieces, need to be specified, otherwise the triangles making up the disc will be degenerate. You can also specify the number of radial pieces (opt\_stacks). A value of 1 for opt\_stacks will give you a simple disc of pie pieces. If you want to create an annulus by omitting some of the center stacks, you can specify the stack at which to start creating triangles. Finally, stackPower allows you to have the widths increase or decrease as you move away from the center. This is particularly useful when using the disc as a ground plane with a fixed camera such that you don't need the resolution of small triangles near the perimeter. For example, a value of 2 will produce stacks whose outside radius increases with the square of the stack index. A value of 1 will give uniform stacks.

**Parameters:**

*radius* Radius of the ground plane.  
*divisions* Number of triangles in the ground plane (at least 3).  
*opt\_stacks* Number of radial divisions (default=1).  
*opt\_startStack* Which radial division to start dividing at.  
*opt\_stackPower* Power to raise stack size to for decreasing width.  
*opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The created plane vertices.

[!o3d.Shape](#) `o3djs.primitives.createFadePlane` (

<a href="#">!o3d.Pack</a>	<i>pack</i>
<a href="#">!o3d.Material</a>	<i>material</i>
number	<i>width</i>

<i>number</i>	<i>depth</i>
<i>number</i>	<i>subdivisionsWidth</i>
<i>number</i>	<i>subdivisionsDepth</i>
<b><u>!o3djs.math.Matrix4</u></b> <i>opt_matrix</i>	

Creates an XZ fade plane, where the alpha channel of the color stream fades from 1 to 0. The created plane has position, normal, uv and vertex color streams.

#### Parameters:

<i>pack</i>	Pack to create plane elements in.
<i>material</i>	to use.
<i>width</i>	Width of the plane.
<i>depth</i>	Depth of the plane.
<i>subdivisionsWidth</i>	Number of steps across the plane.
<i>subdivisionsDepth</i>	Number of steps down the plane.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

#### Returns:

**!o3d.Shape**.The created plane.

#### See Also:

- **o3d.Pack**
- **o3d.Shape**

**!o3d.Shape** o3djs.primitives.createPlane ( **!o3d.Pack** *pack*  
**!o3d.Material** *material*  
*number* *width*  
*number* *depth*  
*number* *subdivisionsWidth*  
*number* *subdivisionsDepth*  
**!o3djs.math.Matrix4** *opt\_matrix* )

Creates an XZ plane. The created plane has position, normal and uv streams.

#### Parameters:

<i>pack</i>	Pack to create plane elements in.
<i>material</i>	to use.
<i>width</i>	Width of the plane.
<i>depth</i>	Depth of the plane.
<i>subdivisionsWidth</i>	Number of steps across the plane.
<i>subdivisionsDepth</i>	Number of steps down the plane.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

#### Returns:

**!o3d.Shape**.The created plane.

#### See Also:

- **o3d.Pack**

- [o3d.Shape](#)

```
!o3djs.primitives.VertexInfo
o3djs.primitives.createPlaneVertices ( number width
                                         number depth
                                         number subdivisionsWidth
                                         number subdivisionsDepth
                                         !o3djs.math.Matrix4 opt_matrix )
```

Creates XZ plane vertices. The created plane has position, normal and uv streams.

**Parameters:**

<i>width</i>	Width of the plane.
<i>depth</i>	Depth of the plane.
<i>subdivisionsWidth</i>	Number of steps across the plane.
<i>subdivisionsDepth</i>	Number of steps down the plane.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The created plane vertices.

```
!o3d.Shape o3djs.primitives.createPrism ( !o3d.Pack pack
                                         !o3d.Material material
                                         !Array.<!Array.<number>> points
                                         number depth
                                         !o3djs.math.Matrix4 opt_matrix )
```

Creates a prism shape by extruding a polygon. The prism will be created around the 2d points passed in an array and extruded along the z-axis. The end caps of the prism are constructed using a triangle fan originating at the first point, so a non-convex polygon might not get the desired shape, but it will if it is convex with respect to the first point. Texture coordinates map each face of the wall exactly to the unit square. Texture coordinates on the front and back faces are scaled such that the bounding rectangle of the polygon is mapped to the unit square. The created prism has position, normal and uv streams.

**Parameters:**

<i>pack</i>	Pack to create wedge elements in.
<i>material</i>	to use.
<i>points</i>	Array of 2d points in the format: [[x1, y1], [x2, y2], [x3, y3],...] that describe a 2d polygon.
<i>depth</i>	The depth to extrude the triangle.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created wedge.

```
!o3djs.primitives.VertexInfo
o3djs.primitives.createPrismVertices ( !Array.<!Array.<number>> points
                                         number depth
                                         !o3djs.math.Matrix4 opt_matrix )
```

Creates prism vertices by extruding a polygon. The prism will be created around the 2d points passed in and extruded along the z axis. The end caps of the prism are constructed using a triangle fan originating at point 0, so a non-convex polygon might not get the desired shape, but it will if it is convex with respect to point 0. Texture coordinates map each face of the wall exactly to the unit square. Texture coordinates on the front and back faces are scaled such that the bounding rectangle of the polygon is mapped to the unit square. The created prism has position, normal, uv streams.

**Parameters:**

- points* Array of 2d points in the format [[x1, y1], [x2, y2], [x3, y3],...] that describe a 2d polygon.
- depth* The depth to extrude the triangle.
- opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The created cylinder vertices.

[!o3d.Shape](#) o3djs.primitives.createRainbowCube ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material*  
number *size*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates a cube with varying vertex colors. The cube will be created around the origin. (-size / 2, size / 2)

**Parameters:**

- pack* Pack to create cube elements in.
- material* to use.
- size* Width, height and depth of the cube.
- opt\_matrix* A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created cube.

**See Also:**

- [o3d.Pack](#)
- [o3d.Shape](#)

[!o3d.Shape](#) o3djs.primitives.createSphere ( [!o3d.Pack](#) *pack*  
[!o3d.Material](#) *material*  
number *radius*  
number *subdivisionsAxis*  
number *subdivisionsHeight*  
[!o3djs.math.Matrix4](#) *opt\_matrix* )

Creates a sphere. The created sphere has position, normal and uv streams.

**Parameters:**

- pack* Pack to create sphere elements in.
- material* to use.
- radius* radius of the sphere.

`subdivisionsAxis` number of steps around the sphere.  
`subdivisionsHeight` number of vertically on the sphere.  
`opt_matrix` A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created sphere.

**See Also:**

- [o3d.Pack](#)
- [o3d.Shape](#)

<a href="#">!o3djs.primitives.VertexInfo</a>	(	number	<i>radius</i>
<code>o3djs.primitives.createSphereVertices</code>		number	<i>subdivisionsAxis</i>
		number	<i>subdivisionsHeight</i>
		<a href="#">!o3djs.math.Matrix4</a>	<i>opt_matrix</i>
	)		

Creates sphere vertices. The created sphere has position, normal and uv streams.

**Parameters:**

`radius` radius of the sphere.  
`subdivisionsAxis` number of steps around the sphere.  
`subdivisionsHeight` number of vertically on the sphere.  
`opt_matrix` A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The created sphere vertices.

`o3djs.primitives.createVertexInfo ( )`

Creates a new VertexInfo.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The new VertexInfo.

<a href="#">!o3djs.primitives.VertexStreamInfo</a>	(	number	<i>numComponents</i>
<code>o3djs.primitives.createVertexStreamInfo</code>		!	
		<a href="#">!o3d.Stream.Semantic</a>	<i>semantic</i>
		number	<i>opt_semanticIndex</i>
	x		)

Create a VertexStreamInfo.

**Parameters:**

`numComponents` The number of numerical components per element.  
`semantic` The semantic of the stream.  
`opt_semanticIndex` The semantic index of the stream. Defaults to zero.

**Returns:**

[!o3djs.primitives.VertexStreamInfo](#).The new stream.

```
!o3d.Shape o3djs.primitives.createWedge ( !o3d.Pack pack
                                         !o3d.Material material
                                         !Array.<!Array.<number>> points
                                         number depth
                                         !o3djs.math.Matrix4 opt_matrix )
```

Creates a wedge shape. A wedge being an extruded triangle. The wedge will be created around the 3 2d points passed in and extruded along the z-axis. The created wedge has position, normal and uv streams.

**Parameters:**

<i>pack</i>	Pack to create wedge elements in.
<i>material</i>	to use.
<i>points</i>	Array of 2d points in the format [[x1, y1], [x2, y2], [x3, y3]] that describe a 2d triangle.
<i>depth</i>	The depth to extrude the triangle.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[!o3d.Shape](#).The created wedge.

```
!o3djs.primitives.VertexInfo  
o3djs.primitives.createWedgeVertices ( !Array.<!Array.<number>> inPoints
                                         number depth
                                         !o3djs.math.Matrix4 x opt_matrix )
```

Creates wedge vertices, wedge being an extruded triangle. The wedge will be created around the 3 2d points passed in and extruded along the z axis. The created wedge has position, normal and uv streams.

**Parameters:**

<i>inPoints</i>	Array of 2d points in the format [[x1, y1], [x2, y2], [x3, y3]] that describe a 2d triangle.
<i>depth</i>	The depth to extrude the triangle.
<i>opt_matrix</i>	A matrix by which to multiply all the vertices.

**Returns:**

[!o3djs.primitives.VertexInfo](#).The created cylinder vertices.

[o3djs.primitives.setCullingInfo](#) ( [!o3d.Primitive](#) *primitive* )

Sets the bounding box and zSortPoint for a primitive based on its vertices

**Parameters:**

*primitive* Primitive to set culling info for.

# o3djs.quaternions Module Reference

[List of all members.](#)

---

# Detailed Description

A Module for quaternion math.

## Source

[o3djs/quaternions.js](#)

## Public Member Functions

- <static> (![o3djs.quaternions.Quaternion](#)|number) [add](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [addQuaternionQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [addQuaternionScalar](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [addScalarQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [axisRotation](#)(axis, angle)
- <static> ![o3djs.quaternions.Quaternion](#) [conjugate](#)(q)
- <static> ![o3djs.quaternions.Quaternion](#) [copy](#)(q)
- <static> (![o3djs.quaternions.Quaternion](#)|number) [div](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [divQuaternionQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [divQuaternionScalar](#)(q, k)
- <static> ![o3djs.quaternions.Quaternion](#) [divScalarQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [inverse](#)(q)
- <static> number [length](#)(a)
- <static> number [lengthSquared](#)(a)
- <static> string [mathType](#)(a)
- <static> (![o3djs.quaternions.Quaternion](#)|number) [mul](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [mulQuaternionQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [mulQuaternionScalar](#)(q, k)
- <static> Array [mulScalarQuaternion](#)(k, q)
- <static> ![o3djs.quaternions.Quaternion](#) [negative](#)(q)
- <static> ![o3djs.quaternions.Quaternion](#) [normalize](#)(a)
- <static> ![o3djs.math.Matrix4](#) [quaternionToRotation](#)(q)
- <static> ![o3djs.quaternions.Quaternion](#) [rotationToQuaternion](#)(m)
- <static> ![o3djs.quaternions.Quaternion](#) [rotationX](#)(angle)
- <static> ![o3djs.quaternions.Quaternion](#) [rotationY](#)(angle)
- <static> ![o3djs.quaternions.Quaternion](#) [rotationZ](#)(angle)
- <static> (![o3djs.quaternions.Quaternion](#)|number) [sub](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [subQuaternionQuaternion](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [subQuaternionScalar](#)(a, b)
- <static> ![o3djs.quaternions.Quaternion](#) [subScalarQuaternion](#)(a, b)

## Public Properties

- <static> !Array.<number> [Quaterion](#)
-

# Member Function Documentation

([!o3djs.quaternions.Quaternion](#)|number) o3djs.quaternions.add ( ([!o3djs.quaternions.Quaternion](#)|number) *a* ([!o3djs.quaternions.Quaternion](#)|number) *b* )

Adds two objects which are either scalars or quaternions.

**Parameters:**

*a* Operand.  
*b* Operand.

**Returns:**

([!o3djs.quaternions.Quaternion](#)|number).The sum of a and b.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.addQuaternionQuaternion ( [!o3djs.quaternions.Quaternion](#) *a* [!o3djs.quaternions.Quaternion](#) *b* )

Adds two Quaternions.

**Parameters:**

*a* Operand Quaternion.  
*b* Operand Quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The sum of a and b.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.addQuaternionScalar ( [!o3djs.quaternions.Quaternion](#) *a* number *b* )

Adds a quaternion to a scalar.

**Parameters:**

*a* Operand Quaternion.  
*b* Operand Scalar.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The sum of a and b.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.addScalarQuaternion ( number *a* [!o3djs.quaternions.Quaternion](#) *b* )

Adds a scalar to a quaternion.

**Parameters:**

*a* Operand scalar.  
*b* Operand quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The sum of a and b.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.axisRotation ( [!o3djs.math.Vector3](#) *axis*

number                          *angle* )

Creates a quaternion which rotates around the given axis by the given angle.

**Parameters:**

*axis* The axis about which to rotate.

*angle* The angle by which to rotate (in radians).

**Returns:**

[!o3djs.quaternions.Quaternion](#).A quaternion which rotates angle radians around the axis.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.conjugate ( [!o3djs.quaternions.Quaternion](#) *q* )

Computes the conjugate of the given quaternion.

**Parameters:**

*q* The quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The conjugate of q.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.copy ( [!o3djs.quaternions.Quaternion](#) *q* )

Copies a quaternion.

**Parameters:**

*q* The quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).A new quaternion identical to q.

([!o3djs.quaternions.Quaternion](#)|number)

([!o3djs.quaternions.Quaternion](#)|

*a*

o3djs.quaternions.div

number)

([!o3djs.quaternions.Quaternion](#)|

*b* )

number)

Divides two objects which are either scalars or quaternions.

**Parameters:**

*a* Operand.

*b* Operand.

**Returns:**

[!o3djs.quaternions.Quaternion](#)|number).The quotient of a and b.

[!o3djs.quaternions.Quaternion](#)

( [!o3djs.quaternions.Quaternion](#) *a*

o3djs.quaternions.divQuaternionQuaternion

[!o3djs.quaternions.Quaternion](#) *b* )

Divides two quaternions; assumes the convention that a/b = a\*(1/b).

**Parameters:**

*a* Operand quaternion.

*b* Operand quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The quaternion quotient a / b.

[!o3djs.quaternions.Quaternion](#)

( [!o3djs.quaternions.Quaternion](#) *q*

o3djs.quaternions.divQuaternionScalar  
number  $k$  )

Divides a Quaternion by a scalar.

**Parameters:**

$q$  The quaternion.

$k$  The scalar.

**Returns:**

[!o3djs.quaternions.Quaternion](#).q The quaternion q divided by k.

[!o3djs.quaternions.Quaternion](#)  
o3djs.quaternions.divScalarQuaternion  
( number  $a$   
[!o3djs.quaternions.Quaternion](#)  $b$  )

Divides a scalar by a quaternion.

**Parameters:**

$a$  Operand scalar.

$b$  Operand quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The quaternion product.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.inverse ( [!o3djs.quaternions.Quaternion](#)  $q$  )

Computes the multiplicative inverse of a quaternion.

**Parameters:**

$q$  The quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The multiplicative inverse of q.

number o3djs.quaternions.length ( [!o3djs.quaternions.Quaternion](#)  $a$  )

Computes the length of a Quaternion, i.e. the square root of the sum of the squares of the coefficients.

**Parameters:**

$a$  The Quaternion.

**Returns:**

number.The length of a.

number o3djs.quaternions.lengthSquared ( [!o3djs.quaternions.Quaternion](#)  $a$  )

Computes the square of the length of a quaternion, i.e. the sum of the squares of the coefficients.

**Parameters:**

$a$  The quaternion.

**Returns:**

number.The square of the length of a.

string o3djs.quaternions.mathType ( (number|[!o3djs.quaternions.Quaternion](#))  $a$  )

Quickly determines if the object a is a scalar or a quaternion; assumes that the argument is either a number (scalar), or an array of numbers.

## Parameters:

*a* A number or array the type of which is in question.

## Returns:

string. Either the string 'Scalar' or 'Quaternion'.

```
(!o3djs.quaternions.Quaternion|number)
o3djs.quaternions.mul
( (!o3djs.quaternions.Quaternion|number)
  (!o3djs.quaternions.Quaternion|number) )
```

Multiplies two objects which are either scalars or quaternions.

## Parameters:

*a* Operand.  
*b* Operand.

## Returns:

(!o3djs.math.Quaternion|number). The product of a and b.

[!o3djs.quaternions.Quaternion](#)  
o3djs.quaternions.mulQuaternionQuaternion  
( [!o3djs.quaternions.Quaternion](#) a  
[!o3djs.quaternions.Quaternion](#) b )

Multiplies two quaternions.

### Parameters:

*a* Operand quaternion.  
*b* Operand quaternion.

## Returns:

`!o3djs.math.Quaternion`. The quaternion product  $a * b$ .

[!o3djs.quaternions.Quaternion](#) ( [!o3djs.quaternions.Quaternion](#) *q*   
 o3djs.quaternions.mulQuaternionScalar *number* *k* )

**Multiplies a quaternion by a scalar.**

### Parameters:

*q* The Quaternion.  
*k* The scalar.

## Returns:

[!o3djs.quaternions.Quaternion](#). The product of k and v.

Array o3djs.quaternions.mulScalarQuaternion ( Array  $k$   
Array  $q$  )

**Multiples a scalar by a quaternion.**

## Parameters:

- $k$  The scalar.
- $q$  The quaternion.

## Returns:

Array. The product of k and q.

[!o3djs.math.Quaternion](#) o3djs.math.Quaternion ( [!o3djs.math.Quaternion](#) *q* )  
Negates a quaternion.

**Parameters:**

*q* The quaternion.

**Returns:**

[!o3djs.math.Quaternion](#).-*q*.

[!o3djs.math.Quaternion](#) o3djs.math.Quaternion ( [!o3djs.math.Quaternion](#) *a* )  
Divides a Quaternion by its length and returns the quotient.

**Parameters:**

*a* The Quaternion.

**Returns:**

[!o3djs.math.Quaternion](#).A unit length quaternion pointing in the same direction as *a*.

[!o3djs.math.Matrix4](#) o3djs.math.Matrix4 ( [!o3djs.math.Quaternion](#) *q* )  
Computes a 4-by-4 rotation matrix (with trivial translation component) given a quaternion. We assume the convention that to rotate a vector *v* by a quaternion *r* means to express that vector as a quaternion *q* by letting *q* = [v[0], v[1], v[2], 0] and then obtain the rotated vector by evaluating the expression (*r* \* *q*) / *r*.

**Parameters:**

*q* The quaternion.

**Returns:**

[!o3djs.math.Matrix4](#).A 4-by-4 rotation matrix.

[!o3djs.math.Quaternion](#) ( [!o3djs.math.Matrix4](#)!  
o3djs.math.Matrix4.rotationToQuaternion ( [!o3djs.math.Matrix3](#) ) *m* )

Computes a quaternion whose rotation is equivalent to the given matrix.

**Parameters:**

*m* A 3-by-3 or 4-by-4 rotation matrix.

**Returns:**

[!o3djs.math.Quaternion](#).A quaternion *q* such that *quaternions.quaternionToRotation(q)* is *m*.

[!o3djs.math.Quaternion](#) o3djs.math.Quaternion ( number *angle* )

Creates a quaternion which rotates around the x-axis by the given angle.

**Parameters:**

*angle* The angle by which to rotate (in radians).

**Returns:**

[!o3djs.math.Quaternion](#).The quaternion.

[!o3djs.math.Quaternion](#) o3djs.math.Quaternion ( number *angle* )

Creates a quaternion which rotates around the y-axis by the given angle.

**Parameters:**

*angle* The angle by which to rotate (in radians).

**Returns:**

[!o3djs.quaternions.Quaternion](#).The quaternion.

[!o3djs.quaternions.Quaternion](#) o3djs.quaternions.rotationZ ( number *angle* )

Creates a quaternion which rotates around the z-axis by the given angle.

**Parameters:**

*angle* The angle by which to rotate (in radians).

**Returns:**

[!o3djs.quaternions.Quaternion](#).The quaternion.

([!o3djs.quaternions.Quaternion](#)|number)

o3djs.quaternions.sub

([!o3djs.quaternions.Quaternion](#)|

number)

*a*

([!o3djs.quaternions.Quaternion](#)|

number)

*b* )

Subtracts two objects which are either scalars or quaternions.

**Parameters:**

*a* Operand.

*b* Operand.

**Returns:**

([!o3djs.quaternions.Quaternion](#)|number).The difference of a and b.

[!o3djs.quaternions.Quaternion](#)

o3djs.quaternions.subQuaternionQuaternion

( [!o3djs.quaternions.Quaternion](#) *a*

[!o3djs.quaternions.Quaternion](#) *b* )

Subtracts two quaternions.

**Parameters:**

*a* Operand quaternion.

*b* Operand quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The difference a - b.

[!o3djs.quaternions.Quaternion](#)

o3djs.quaternions.subQuaternionScalar

( [!o3djs.quaternions.Quaternion](#) *a*

number

*b* )

Subtracts a scalar from a quaternion.

**Parameters:**

*a* Operand quaternion.

*b* Operand scalar.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The difference a - b.

[!o3djs.quaternions.Quaternion](#)

o3djs.quaternions.subScalarQuaternion

( number *a*

[!o3djs.quaternions.Quaternion](#) *b* )

Subtracts a quaternion from a scalar.

**Parameters:**

*a* Operand scalar.  
*b* Operand quaternion.

**Returns:**

[!o3djs.quaternions.Quaternion](#).The difference a - b.

---

## Member Property Documentation

`!Array.<number> o3djs.quaternions.Quaternion`

A Quaternion.

## o3djs.rendergraph Module Reference

[List of all members.](#)

---

### Detailed Description

A Module for creating render graphs.

### Source

[o3djs/rendergraph.js](#)

### Public Member Functions

- <static> [!o3djs.rendergraph.ViewInfo createBasicView](#)(pack, treeRoot, opt\_parent, opt\_clearColor, opt\_priority, opt\_viewport)
  - <static> [!o3djs.rendergraph.ViewInfo createExtraView](#)(viewInfo, opt\_viewport, opt\_clearColor, opt\_priority)
  - <static> [!o3djs.rendergraph.ViewInfo createView](#)(pack, treeRoot, opt\_parent, opt\_clearColor, opt\_priority, opt\_viewport, opt\_performanceDrawList, opt\_zOrderedDrawList)
- 

## Member Function Documentation

[!o3djs.rendergraph.ViewInfo](#)  
o3djs.rendergraph.createBasicView

( [!o3d.Pack](#) *pack*  
[!o3d.Transform](#) *treeRoot*  
[!o3d.RenderNode](#) *opt\_parent*  
[!o3djs.math.Vector4](#) *opt\_clearColor*

number *opt\_priority*  
*!o3djs.math.Vector4 opt\_viewport* )

Creates a basic render graph setup to draw opaque and transparent 3d objects.

**Parameters:**

*pack* Pack to manage created objects.  
*treeRoot* root Transform of tree to render.  
*opt\_parent* RenderNode to build this view under.  
*opt\_clearColor* color to clear view.  
*opt\_priority* Optional base priority for created objects.  
*opt\_viewport* viewport settings for view.

**Returns:**

[!o3djs.rendergraph.ViewInfo](#).A ViewInfo object with info about everything created.

[!o3djs.rendergraph.ViewInfo](#)

o3djs.rendergraph.createExtraView

( [!o3djs.rendergraph.ViewInfo](#) *viewInfo*  
*!o3djs.math.Vector4* *opt\_viewport*  
*!o3djs.math.Vector4* *opt\_clearColor*  
or  
number *opt\_priority* )

Creates an extra view render graph setup to draw opaque and transparent 3d objects based on a previously created view. It uses the previous view to share draw lists and to set the priority.

**Parameters:**

*viewInfo* ViewInfo returned from createBasicView.  
*opt\_viewport* viewport settings for view.  
*opt\_clearColor* color to clear view.  
*opt\_priority* base priority for created objects.

**Returns:**

[!o3djs.rendergraph.ViewInfo](#).A ViewInfo object with info about everything created.

[!o3djs.rendergraph.ViewInfo](#)

o3djs.rendergraph.createView

( [!o3d.Pack](#) *pack*  
[!o3d.Transform](#) *treeRoot*  
[!o3d.RenderNode](#) *opt\_parent*  
[!o3djs.math.Vector4](#) *opt\_clearColor*  
number *opt\_priority*  
[!o3djs.math.Vector4](#) *opt\_viewport*  
[!o3d.DrawList](#) *opt\_performanceDrawList*  
[!o3d.DrawList](#) *opt\_zOrderedDrawList* )

Creates a basic render graph setup to draw opaque and transparent 3d objects.

**Parameters:**

*pack* Pack to manage created objects.  
*treeRoot* root Transform of tree to render.  
*opt\_parent* RenderNode to build this view under.

<i>optClearColor</i>	color to clear view.
<i>optPriority</i>	Optional base priority for created objects.
<i>optViewport</i>	viewport settings for view.
<i>optPerformanceDrawList</i>	Optional DrawList to use for performanceDrawPass.
<i>optZOrderedDrawList</i>	Optional DrawList to use for zOrderedDrawPass.

**Returns:**

[!o3djs.rendergraph.ViewInfo](#).A ViewInfo object with info about everything created.

# o3djs.scene Module Reference

[List of all members.](#)

---

## Detailed Description

A Module with various scene functions and classes.

## Source

[o3djs/scene.js](#)

## Public Member Functions

- <static> [!o3djs.io.LoadInfo loadScene](#)(client, pack, parent, url, callback, opt\_options)
- 

## Member Function Documentation

<a href="#">!o3djs.io.LoadInfo</a> o3djs.scene.loadScene	(	<a href="#">!o3d.Client</a>	<i>client</i>
		<a href="#">!o3d.Pack</a>	<i>pack</i>
		<a href="#">!o3d.Transform</a>	<i>parent</i>
		string	<i>url</i>
		!function(! <a href="#">o3d.Pack</a> , ! <a href="#">o3d.Transform</a> , *): void	<i>callback</i>
		<a href="#">!o3djs.serialization.Options</a>	<i>opt_option</i> <i>s</i>

Loads a scene.

**Parameters:**

<i>client</i>	An O3D client object.
<i>pack</i>	Pack to load scene into.
<i>parent</i>	Transform to parent scene under.
<i>url</i>	URL of scene to load.

*callback* Callback when scene is loaded. It will be passed the pack, the parent and an exception which is null on success.

$opt\_option_s$  Options passed into the loader.

## Returns:

[!o3djs.io.LoadInfo](https://o3djs.io/LoadInfo) A LoadInfo for tracking progress.

#### See Also:

- [o3djs.loader.createLoader](#)

# **o3djs.serialization Module Reference**

## List of all members.

## Detailed Description

A Module for handling events related to o3d and various browsers.

## Source

## o3djs/serialization.js

## Public Member Functions

- <static> [!o3djs.serialization.Deserializer](#) [createDeserializer](#)(pack, json)
  - <static> [deserialize](#)(pack, json)
  - <static> [deserializeArchive](#)(archiveInfo, sceneJsonUri, client, pack, parent, callback, opt options)

## Public Properties

- <static> {{opt\_animSource: ![o3d.ParamFloat](#), opt\_async: boolean}} [Options](#)
  - <static> number [supportedVersion](#)

# Member Function Documentation

Creates a deserializer that will incrementally deserialize a transform graph. The deserializer object has a method called run that does a fixed amount of work and returns. It returns true until the transform graph is fully serialized. It returns false from then on.

**Parameters:**

*pack* The pack to create the deserialized objects in.  
*json* An object tree conforming to the JSON rules.

**Returns:**

[!o3djs.serialization.Deserializer](#).A deserializer object.

```
o3djs.serialization.deserialize ( !o3d.Pack pack
                                !Object json )
```

Deserializes a transform graph.

**Parameters:**

*pack* The pack to create the deserialized objects in.  
*json* An object tree conforming to the JSON rules.

```
o3djs.serialization.deserializeArchive ( !o3djs.io.archiveInfo
                                         string
                                         !o3d.Client
                                         !o3d.Pack
                                         !o3d.Transform
                                         !function(!o3d.Pack, !o3d.Transform, *):
                                         void
                                         !o3djs.serialization.Options
                                         opt_options )
```

*archiveInfo*  
*sceneJsonUri*  
*client*  
*pack*  
*parent*  
*callback*  
*opt\_options*

Deserializes a single json object named 'scene.json' from a loaded o3djs.io.ArchiveInfo.

**Parameters:**

*archiveInfo* Archive to load from.  
*sceneJsonUri* The relative URI of the scene JSON file within the archive.  
*client* An O3D client object.  
*pack* The pack to create the deserialized objects in.  
*parent* Transform to parent loaded stuff from.  
*callback* A function that will be called when deserialization is finished. It will be passed the pack, the parent transform, and an exception which will be null on success.  
*opt\_options* Options.

---

## Member Property Documentation

{ {opt\_animSource: !o3d.ParamFloat, opt\_async: boolean} } o3djs.serialization.Options  
Options for deserialization.

opt\_animSource is an optional ParamFloat that will be bound as the source param for all animation time params in the scene. opt\_async is a bool that will make the deserialization process async.

number o3djs.serialization.supportedVersion

The oldest supported version of the serializer. It isn't necessary to increment this version whenever the format changes. Only change it when the deserializer becomes incapable of deserializing an older version.

# o3djs.shape Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for shapes.

## Source

[o3djs/shape.js](#)

## Public Member Functions

- <static> [addMissingTexCoordStreams](#)(shape)
  - <static> [deleteDuplicateShape](#)(shape, pack)
  - <static> [!o3d.Shape duplicateShape](#)(pack, source)
  - <static> [prepareShape](#)(pack, shape)
  - <static> [prepareShapes](#)(pack)
  - <static> [setBoundingBoxesAndZSortPoints](#)(shape)
- 

## Member Function Documentation

`o3djs.shape.addMissingTexCoordStreams ( !o3d.Shape shape )`

Adds missing tex coord streams to a shape's elements.

### Parameters:

*shape* Shape to add missing streams to.

### See Also:

- [o3djs.element.addMissingTexCoordStreams](#)

`o3djs.shape.deleteDuplicateShape ( !o3d.Shape shape )`

*!o3d.Pack pack* )

Attempts to delete the parts of a shape that were created by `duplicateShape` as well as any `drawElements` attached to it.

### Parameters:

*shape* shape to delete.

*pack* Pack to release objects from.

`!o3d.Shape o3djs.shape.duplicateShape ( !o3d.Pack pack  
!o3d.Shape source )`

Copies a shape's elements and streambank or buffers so the two will share streambanks, vertex and index buffers.

**Parameters:**

*pack* Pack to manage created objects.  
*source* The Shape to copy.

**Returns:**

[!o3d.Shape](#).the new copy of the shape.

`o3djs.shape.prepareShape ( !o3d.Pack pack  
                          !o3d.Shape shape )`

Prepares a shape by setting its boundingBox, zSortPoint and creating DrawElements.

**Parameters:**

*pack* Pack to manage created objects.  
*shape* Shape to prepare.

`o3djs.shape.prepareShapes ( !o3d.Pack pack )`

Prepares all the shapes in the given pack by setting their boundingBox, zSortPoint and creating DrawElements.

**Parameters:**

*pack* Pack to manage created objects.

`o3djs.shape.setBoundingBoxesAndZSortPoints ( !o3d.Shape shape )`

Sets the bounding box and z sort points of a shape's elements.

**Parameters:**

*shape* Shape to set info on.

# o3djs.simple Module Reference

[List of all members.](#)

---

## Detailed Description

A Module for using o3d in a very simple way.

## Source

[o3djs/simple.js](#)

## Public Member Functions

- <static> [!o3djs.simple.SimpleInfo](#) [create](#)(clientObject)
- 

## Member Function Documentation

[!o3djs.simple.SimpleInfo](#) `o3djs.simple.create ( !Element clientObject )`

Creates an o3djs.simple library object that helps manage o3d for the extremely simple cases.

```
<html><body>
<script type="text/javascript" src="o3djs/all.js">
</script>
<script type="text/javascript">
windows.onload = init;
function init() {
    o3djs.base.makeClients(initStep2);
}
function initStep2(clientElements) {
    var clientElement = clientElements[0];
    // Create an o3djs.simple object to manage things in a simple way.
    g_simple = o3djs.simple.create(clientElement);
    // Create a cube.
    g_cube = g_simple.createCube(50);
    // DONE!
}
</script>
<div id="o3d" style="width: 600px; height: 600px"></div>
</body></html>
```

**Parameters:**

*clientObject* O3D.Plugin Object.

**Returns:**

[!o3djs.simple.SimpleInfo](#).Javascript object that hold info for the simple library.

## o3djs.util Module Reference

[List of all members.](#)

---

### Detailed Description

A Module with various utilities.

### Source

[o3djs/util.js](#)

### Public Member Functions

- <static> [addScriptUri](#)(uri)
- <static> boolean [arrayContains](#)(array, value)
- <static> \* [callV8](#)(clientElement, callback, thisArg, args)
- <static> Element [createClient](#)(element, opt\_features, opt\_requestVersion)
- <static> !function(...): \* [curry](#)(func)
- <static> string [getAbsoluteURI](#)(uri)
- <static> [!o3d.BoundingBox](#) [getBoundingBoxOfTree](#)(treeRoot)
- <static> string [getCurrentURI](#)()

- <static> Node [getElementById](#)(id)
  - <static> string [getElementContentById](#)(id)
  - <static> ?string [getPluginVersion](#)()
  - <static> number [getPowerOfTwoSize](#)(size)
  - <static> string [getScriptTagText](#)()
  - <static> !Array.<!o3d.Transform> [getTransformsInTreeByPrefix](#)(treeRoot, prefix)
  - <static> !Array.<!o3d.Transform> [getTransformsInTreeByTags](#)(treeRoot, searchTags)
  - <static> [informNoGraphics](#)(initStatus, error, opt\_id, opt\_tag)
  - <static> [informPluginFailure](#)(initStatus, error, opt\_id, opt\_tag)
  - <static> boolean [isScriptUri](#)(uri)
  - <static> [makeClients](#)(callback, opt\_features, opt\_requiredVersion, opt\_failureCallback, opt\_id, opt\_tag)
  - <static> [offerPlugin](#)(opt\_id, opt\_tag)
  - <static> boolean [requiredVersionAvailable](#)(requiredVersion)
  - <static> [setMainEngine](#)(engine)
  - <static> string [toAbsoluteUri](#)(uri)

# Public Properties

- <static> undefined [Engine](#)
  - <static> string [PLUGIN\\_DOWNLOAD\\_URL](#)
  - <static> string [PLUGIN\\_NAME](#)
  - <static> undefined [rendererInitStatus](#)
  - <static> string [REQUIRED\\_VERSION](#)
  - <static> !Array.<string> [scriptUris\\_](#)

## Member Function Documentation

`o3djs.util.addScriptUri ( string uri )`

Add a script URI. Scripts that are referenced from script tags that are within this URI are automatically loaded into the alternative JavaScript main JavaScript engine. Do not include directories of scripts that are included with o3djs.require. These are always available. This mechanism is not able to load scripts in a different domain from the document.

### Parameters:

*uri* The URI.

```
boolean o3djs.util.arrayContains ( !Array.<*> array  
                                *           value )
```

Searches an array for a specific value.

## Parameters:

*array* Array to search.

*value* Value to search for.

## Returns:

`boolean`. True if value is in array.

```
* o3djs.util.callV8 ( !Object      clientElement
                      !function(...): * callback
                      !Object      thisArg
                      !Array.<*>    args      )
```

Evaluate a callback function in the V8 engine.

**Parameters:**

*clientElement* The plugin containing the V8 engine.  
*callback* A function to call.  
*thisArg* The value to be bound to "this".  
*args* The arguments to pass to the callback.

**Returns:**

\*.The result of calling the callback.

```
Element o3djs.util.createClient ( !Element element
                                    string    opt_features
                                    string    opt_requestVersion )
```

Creates a client element. In other words it creates an tag for o3d. Note that the browser may not have initialized the plugin before returning.

**Parameters:**

*element* The DOM element under which the client element will be appended.  
*opt\_features* A comma separated list of the features you need for your application. The current list of features:

- FloatingPointTextures: Includes the formats R32F, ABGR16F and ABGR32F The features are case sensitive.

*opt\_requestVersion* version string in "major.minor.revision.build" format. You can leave out any non-important numbers for example "3" = request major version 3, "2.4" = request major version 2, minor version 4. If no string is passed in the newest version of the plugin will be created.

**Returns:**

Element.O3D element or null if requested version is not available.

```
!function(...): * o3djs.util.curry ( !function(...): * func )
```

This implements a JavaScript version of currying. Currying allows you to take a function and fix its initial arguments, resulting in a function expecting only the remaining arguments when it is invoked. For example:

```
function add(a, b) {
  return a + b;
}
var increment = o3djs.util.curry(add, 1);
var result = increment(10);
```

Now result equals 11.

**Parameters:**

*func* The function to curry.

**Returns:**

!function(...): \*.The curried function.

string o3djs.util.getAbsoluteURI ( string *uri* )

Given a URI that is relative to the current page, returns the absolute URI.

**Parameters:**

*uri* URI relative to the current page.

**Returns:**

string.Absolute uri. If the page is "http://some.com/folder/sompage.html" and you pass in "images/someimage.jpg" will return "http://some.com/folder/images/someimage.jpg".

[!o3d.BoundingBox](#) o3djs.util.getBoundingBoxOfTree ( [!o3d.Transform](#) *treeRoot* )

Finds the bounding box of all primitives in the tree, in the local space of the tree root. This will use existing bounding boxes on transforms and elements, but not create new ones.

**Parameters:**

*treeRoot* Root of tree to search.

**Returns:**

[!o3d.BoundingBox](#).The boundinding box of the tree.

o3djs.util.getCurrentURI ( )

Gets the URI in which the current page is located, omitting the file name.

**Returns:**

string.The base URI of the page. If the page is "http://some.com/folder/sompage.html" returns "http://some.com/folder/".

Node o3djs.util.getElementById ( string *id* )

Utility to get an element from the DOM by ID. This must be used from V8 in preference to document.getElementById because we do not currently support invoking methods on DOM objects in IE.

**Parameters:**

*id* The Node id.

**Returns:**

Node.The node or null if not found.

string o3djs.util.getElementContentById ( string *id* )

Utility to get the text contents of a DOM element with a particular ID. Currently only supports textarea and script nodes.

**Parameters:**

*id* The Node id.

**Returns:**

string.The text content.

o3djs.util.getPluginVersion ( )

Gets the version of the installed plugin.

**Returns:**

?string.version string in 'major.minor.revision.build' format. If the plugin does not exist returns null.

`number o3djs.util.getPowerOfTwoSize ( number size )`  
Returns the smallest power of 2 that is larger than or equal to size.

### Parameters:

*size* Size to get power of 2 for.

## Returns:

number.smallest power of 2 that is larger than or equal to size.

`o3djs.util.getScriptTagText_()`

Concatenate the text of all the script tags in the document and invokes the callback when complete. This function is asynchronous if any of the script tags reference JavaScript through a URI.

## Returns:

string. The script tag text.

Finds transforms in the tree by prefix.

## Parameters:

*treeRoot* Root of tree to search.

*prefix*    Prefix to look for.

### Returns:

`!Array.<!o3d.Transform>.`Array of transforms matching prefix.

Searches for all transforms with a "o3d.tags" ParamString that contains specific tag keywords assuming comma separated words.

## Parameters:

*treeRoot* Root of tree to search for tags.

*searchTags* Tags to look for. eg "camera",  
"ogre,dragon".

## Returns:

**!Array.<!o3d.Transform>.**Array of transforms.

```
o3djs.util.informNoGraphics ( !o3d.Renderer.InitStatus initStatus
                            string error
                            string opt_id
                            string opt_tag )
```

Tells the user their graphics card is not able to run the plugin or is out of resources etc.

Finds all divs with the id "`^o3d`" and inserts a message. If no areas exist OR if none of them are large enough for the message then displays an alert.

## Parameters:

*initStatus* The initialization status of the renderer.

*error* An error message. Will be " if there is no message.

*opt\_id* The id to look for. This can be a regular expression. The default is "`^o3d`".

*opt\_tag* The type of tag to look for. The default is "div".

```
o3djs.util.informPluginFailure ( !o3d.Renderer.InitStatus initStatus
                                string error
                                string opt_id
                                string opt_tag )
```

Handles failure to create the plugin.

## Parameters:

*initStatus* The initialization status of the renderer.

*error* An error message. Will be " if there is no message.

*opt\_id* The id to look for. This can be a regular expression. The default is "`^o3d`".

*opt tag* The type of tag to look for. The default is "div".

boolean o3djs.util.isScriptUri ( string *uri* )

Determine whether a URI is a script URI that should be loaded into the alternative main JavaScript engine.

### Parameters:

*uri* The URI.

## Returns:

boolean. Whether it is a script URI.

Finds all divs with the an id that starts with "o3d" and inserts a client area inside.

NOTE: the size of the client area is always set to 100% which means the div must have its size set or managed by the browser. Examples:

-- A div of a specific size -- <div id="o3d" style="width:800px; height:600px"></div>

-- A div that fills its containing element -- <div id="o3d" style="width:100%; height:100%"></div>

In both cases, a DOCTYPE is probably required.

You can also request certain features by adding the attribute 'o3d\_features' as in

```
<div id="o3d" o3d features="FloatingPointTextures"></div>
```

This allows you to specify different features per area. Otherwise you can request features as an argument to this function.

**Parameters:**

*callback*

Function to call when client objects have been created.

*opt\_features*

A comma separated list of the features you need for your application. The current list of features:

- FloatingPointTextures: Includes the formats R32F, ABGR16F and ABGR32F
- LargeGeometry: Allows buffers to have more than 65534 elements.
- NotAntiAliased: Turns off anti-aliasing
- InitStatus=X: Where X is a number. Allows simulation of the plugin failing

The features are case sensitive.

*opt\_requiredVersion*

version string in "major.minor.revision.build" format. You can leave out any non-important numbers for example "3" = require major version 3, "2.4" = require major version 2, minor version 4. If no string is passed in the version of the needed by this version of the javascript libraries will be created.

*opt\_failureCallback*

Function to call if the plugin does not exist, if the required version is not installed, or if for some other reason the plugin can not start. If this function is not specified or is null the default behavior of leading the user to the download page will be provided.

*opt\_id*

The id to look for. This can be a regular expression. The default is "^o3d".

*opt\_tag*

The type of tag to look for. The default is "div".

`o3djs.util.offerPlugin ( string opt_id`

`string opt_tag )`

Offers the user the option to download the plugin.

Finds all divs with the id "^o3d" and inserts a message and link inside to download the plugin. If no areas exist OR if none of them are large enough for the message then displays an alert.

**Parameters:**

*opt\_id* The id to look for. This can be a regular expression. The default is "^o3d".

*opt\_tag* The type of tag to look for. The default is "div".

`boolean o3djs.util.requiredVersionAvailable ( string requiredVersion )`

Checks if the required version of the plugin is available.

**Parameters:**

*requiredVersion* version string in "major.minor.revision.build" format. You can leave out any non-important numbers for example "3" = require major version 3, "2.4" = require major version 2, minor version 4.

**Returns:**

`boolean`.True if the required version is available.

`o3djs.util.setMainEngine ( o3djs.util.Engine engine )`

Select an engine to use as the main engine (the one the makeClients callback will be invoked on). If an

embedded engine is requested, one element must be identified with the id 'o3d'. The callback will be invoked in this element.

**Parameters:**

*engine* The engine.

string o3djs.util.toAbsoluteUri ( string *uri* )

Turn a URI into an absolute URI.

**Parameters:**

*uri* The URI.

**Returns:**

string.The absolute URI.

---

## Member Property Documentation

undefined o3djs.util.Engine

Identifies a JavaScript engine.

string o3djs.util.PLUGIN\_DOWNLOAD\_URL

A URL at which to download the client.

string o3djs.util.PLUGIN\_NAME

The name of the o3d plugin. Used to find the plugin when checking for its version.

undefined o3djs.util.rendererInitStatus

The Renderer InitStatus constants so we don't need an o3d object to look them up.

string o3djs.util.REQUIRED\_VERSION

The version of the plugin needed to use this version of the javascript utility libraries.

!Array.<string> o3djs.util.scriptUris\_

The script URIs.

# O3DJS Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[o3djs.arcball.ArcBall](#)  
[o3djs.camera.CameraInfo](#)  
[o3djs.canvas.CanvasInfo](#)  
[o3djs.canvas.CanvasQuad](#)  
[o3djs.debug.DebugHelper](#)  
[o3djs.debug.DebugLine](#)  
[o3djs.debug.DebugLineGroup](#)  
[o3djs.debug.VertexInfo](#)  
[o3djs.error.ErrorCollector](#)  
[o3djs.fps.ColorRect](#)  
[o3djs.fps.FPSManager](#)  
[o3djs.io.ArchiveInfo](#)  
[o3djs.io.LoadInfo](#)  
[o3djs.loader.Loader](#)  
[o3djs.particles.OneShot](#)  
[o3djs.particles.ParticleEmitter](#)  
[o3djs.particles.ParticleSpec](#)  
[o3djs.particles.ParticleSystem](#)  
[o3djs.picking.PickInfo](#)  
[o3djs.picking.ShapeInfo](#)  
[o3djs.picking.TransformInfo](#)  
[o3djs.primitives.VertexInfo](#)  
[o3djs.primitives.VertexStreamInfo](#)  
[o3djs.rendergraph.ViewInfo](#)  
[o3djs.serialization.Deserializer](#)  
[o3djs.simple.SimpleInfo](#)  
[o3djs.simple.SimpleShape](#)

# o3djs.arcball.ArcBall Class Reference

[List of all members.](#)

---

## Detailed Description

A class that implements an arcball.

## Source

[o3djs/arcball.js](#)

## See Also

- [o3djs.arcball](#)

## Constructor

- [o3djs.arcball.ArcBall](#)(areaWidth, areaHeight)

## Public Member Functions

- [click](#)(newPoint)
  - [!o3djs.quaternions.Quaternion drag](#)(newPoint)
  - [!o3djs.math.Vector3 mapToSphere](#)(newPoint)
  - [setAreaSize](#)(areaWidth, areaHeight)
- 

## Constructor

`o3djs.arcball.ArcBall ( number areaWidth  
                          number areaHeight )`

A class that implements an arcball.

### Parameters:

*areaWidth* width of area arcball should cover.  
*areaHeight* height of area arcball should cover.

---

## Member Function Documentation

`o3djs.arcball.ArcBall.click ( !o3djs.math.Vector2 newPoint )`

Records the starting point on the sphere.

**Parameters:**

*newPoint* point in 2d.

[!o3djs.quaternions.Quaternion](#) o3djs.arcball.ArcBall.drag ( [!o3djs.math.Vector2](#) *newPoint* )

Computes the rotation of the sphere based on the initial point clicked as set through Arcball.click and the current point passed in as newPoint

**Parameters:**

*newPoint* point in 2d.

**Returns:**

[!o3djs.quaternions.Quaternion](#).A quaternion representing the new orientation.

[!o3djs.math.Vector3](#) o3djs.arcball.ArcBall.mapToSphere ( [!o3djs.math.Vector2](#) *newPoint* )

Converts a 2d point to a point on the sphere of radius 1 sphere.

**Parameters:**

*newPoint* A point in 2d.

**Returns:**

[!o3djs.math.Vector3](#).A point on the sphere of radius 1.

o3djs.arcball.ArcBall.setAreaSize ( number *areaWidth*  
number *areaHeight* )

Sets the size of the arcball.

**Parameters:**

*areaWidth* width of area arcball should cover.

*areaHeight* height of area arcball should cover.

# o3djs.camera.CameraInfo Class Reference

[List of all members.](#)

---

## Detailed Description

Class to hold Camera information.

## Source

[o3djs/camera.js](#)

## Constructor

- [o3djs.camera.CameraInfo](#)(view, zNear, zFar, opt\_eye, opt\_target, opt\_up)

## Public Member Functions

- [!o3djs.math.Matrix4](#) [computeProjection](#)(areaWidth, areaHeight)

- [setAsOrthographic](#)(magX, magY)
- [setAsPerspective](#)(fieldOfView)

## Public Properties

- (![o3djs.math.Vector3](#)|undefined) [eye](#)
  - number [fieldOfViewRadians](#)
  - (number|undefined) [magX](#)
  - (number|undefined) [magY](#)
  - boolean [orthographic](#)
  - ![o3djs.math.Matrix4](#) [projection](#)
  - (![o3djs.math.Vector3](#)|undefined) [target](#)
  - (![o3djs.math.Vector3](#)|undefined) [up](#)
  - ![o3djs.math.Matrix4](#) [view](#)
  - number [zFar](#)
  - number [zNear](#)
- 

## Constructor

```
o3djs.camera.CameraInfo ( !o3djs.math.Matrix4 view
                          number           zNear
                          number           zFar
                          !o3djs.math.Vector3 opt_eye
                          !o3djs.math.Vector3 opt_target
                          !o3djs.math.Vector3 opt_up )
```

Class to hold Camera information.

### Parameters:

<i>view</i>	The 4-by-4 view matrix.
<i>zNear</i>	near z plane.
<i>zFar</i>	far z plane.
<i>opt_eye</i>	The eye position.
<i>opt_target</i>	The target position.
<i>opt_up</i>	The up vector.

---

## Member Function Documentation

```
!o3djs.math.Matrix4 o3djs.camera.CameraInfo.computeProjection ( number areaWidth
                                                               number areaHeight )
```

Computes a projection matrix for this CameraInfo using the areaWidth and areaHeight passed in.

### Parameters:

<i>areaWidth</i>	width of client area.
<i>areaHeight</i>	height of client area.

**Returns:**

[!o3djs.math.Matrix4](#).The computed projection matrix.

o3djs.camera.CameraInfo.setAsOrthographic ( number *magX*  
number *magY* )

Sets the CameraInfo to an orthographic camera.

**Parameters:**

*magX* horizontal magnification.

*magY* vertical magnification.

o3djs.camera.CameraInfo.setAsPerspective ( number *fieldOfView* )

Sets the CameraInfo to an orthographic camera.

**Parameters:**

*fieldOfView* Field of view in radians.

---

## Member Property Documentation

([!o3djs.math.Vector3](#)|undefined) o3djs.camera.CameraInfo.eye

Eye position.

number o3djs.camera.CameraInfo.fieldOfViewRadians

Field of view in radians.

(number|undefined) o3djs.camera.CameraInfo.magX

horizontal magnification for an orthographic view.

(number|undefined) o3djs.camera.CameraInfo.magY

vertical magnification for an orthographic view.

boolean o3djs.camera.CameraInfo.orthographic

Projection is orthographic.

[!o3djs.math.Matrix4](#) o3djs.camera.CameraInfo.projection

Projection Matrix.

([!o3djs.math.Vector3](#)|undefined) o3djs.camera.CameraInfo.target

Target position.

([!o3djs.math.Vector3](#)|undefined) o3djs.camera.CameraInfo.up

Up Vector.

[!o3djs.math.Matrix4](#) o3djs.camera.CameraInfo.view

View Matrix.

number o3djs.camera.CameraInfo.zFar

Far z plane.

number o3djs.camera.CameraInfo.zNear

Near z plane.

# o3djs.canvas.CanvasInfo Class Reference

[List of all members.](#)

---

## Detailed Description

The CanvasInfo object creates and keeps references to the O3D objects that are shared between all CanvasQuad objects created through it.

## Source

[o3djs/canvas.js](#)

## Constructor

- [o3djs.canvas.CanvasInfo\(pack, root, viewInfo\)](#)

## Public Member Functions

- [!o3djs.canvas.CanvasQuad createQuad\(width, height, transparent, opt\\_parent\)](#)
- [!o3djs.canvas.CanvasQuad createXYQuad\(topX, topY, z, width, height, transparent, opt\\_parent\)](#)

## Public Properties

- [!o3d.Shape opaqueQuadShape](#)
  - [!o3d.Pack pack](#)
  - [!o3d.Transform root](#)
  - [!o3d.Shape transparentQuadShape](#)
  - [!o3djs.rendergraph.ViewInfo viewInfo](#)
- 

## Constructor

o3djs.canvas.CanvasInfo ( [!o3d.Pack](#) *pack*  
[!o3d.Transform](#) *root*  
[!o3djs.rendergraph.ViewInfo](#) *viewInfo* )

The CanvasInfo object creates and keeps references to the O3D objects that are shared between all CanvasQuad objects created through it.

### Parameters:

*pack* Pack to manage CanvasInfo objects.

*root* Default root for visual objects.

*viewInfo* A ViewInfo object as created by o3djs.createView which contains draw lists that the created quads will be placed into.

---

## Member Function Documentation

[!o3djs.canvas.CanvasQuad](#) o3djs.canvas.CanvasInfo.createQuad ( *number width*  
*number height*  
*boolean transparent*  
*!o3d.Transform opt\_parent* )

Creates a CanvasQuad object of the given size. The resulting rectangle Shape is centered at the origin. It can be moved around by setting the localMatrix on the Transform object referenced to by the canvasQuad.transform property.

### Parameters:

*width* The width of the quad.  
*height* The height of the quad.  
*transparent* Set to true if the canvas bitmap uses transparency so that the appropriate blending modes are set.  
*opt\_parent* parent transform to parent the newly created quad under. If no parent transform is provided then the quad gets parented under the CanvasInfo's root.

### Returns:

[!o3djs.canvas.CanvasQuad](#).The newly created CanvasQuad object.

[!o3djs.canvas.CanvasQuad](#) o3djs.canvas.CanvasInfo.createXYQuad ( *number topX*  
*number topY*  
*number z*  
*number width*  
*number height*  
*boolean transparent*  
*!o3d.Transform opt\_parent* )

Creates a CanvasQuad object on the XY plane at the specified position.

### Parameters:

*topX* The x coordinate of the top left corner of the quad.  
*topY* The y coordinate of the top left corner of the quad.  
*z* The z coordinate of the quad. z values are negative numbers, the smaller the number the further back the quad will be.  
*width* The width of the quad.  
*height* The height of the quad.  
*transparent* Set to true if the canvas bitmap uses transparency so that the appropriate blending modes are set.  
*opt\_parent* parent transform to parent the newly created quad under. If no parent transform is provided then the quad gets parented under the CanvasInfo's root.

### Returns:

[!o3djs.canvas.CanvasQuad](#).The newly created CanvasQuad object.

---

## Member Property Documentation

[!o3d.Shape](#) o3djs.canvas.CanvasInfo.opaqueQuadShape

A shape for opaque quads.

[!o3d.Pack](#) o3djs.canvas.CanvasInfo.pack

The pack being used to manage objects created by this CanvasInfo.

[!o3d.Transform](#) o3djs.canvas.CanvasInfo.root

The default root for objects created by this CanvasInfo.

[!o3d.Shape](#) o3djs.canvas.CanvasInfo.transparentQuadShape

A shape for transparent quads.

[!o3djs.rendergraph.ViewInfo](#) o3djs.canvas.CanvasInfo.viewInfo

The ViewInfo this CanvasInfo uses for rendering.

## o3djs.canvas.CanvasQuad Class Reference

[List of all members.](#)

---

### Detailed Description

The CanvasQuad object encapsulates a Transform, a rectangle Shape, an effect that applies a texture to render the quad, and a matching Canvas object that can render into the texture. The dimensions of the texture and the canvas object match those of the quad in order to get pixel-accurate results with the appropriate orthographic projection. The resulting rectangle Shape is positioned at the origin. It can be moved around by setting the localMatrix on the Transform object referenced to by the canvasQuad.transform property. The Canvas associated with the returned CanvasQuad object can be retrieved from the object's 'canvas' property. After issuing any draw commands on the Canvas, you need to call the updateTexture() method on the CanvasQuad to update the contents of the quad surface.

### Source

[o3djs/canvas.js](#)

### Constructor

- [o3djs.canvas.CanvasQuad\(canvasInfo, width, height, transparent, opt\\_parent\)](#)

### Public Member Functions

- [updateTexture\(\)](#)

## Public Properties

- [!o3d.Canvas canvas](#)
  - [!o3d.Transform scaleTransform](#)
  - [!o3d.Transform transform](#)
- 

## Constructor

```
o3djs.canvas.CanvasQuad ( !o3djs.canvas.CanvasInfo canvasInfo
                           number           width
                           number           height
                           boolean          transparent
                           !o3d.Transform   opt_parent )
```

The CanvasQuad object encapsulates a Transform, a rectangle Shape, an effect that applies a texture to render the quad, and a matching Canvas object that can render into the texture. The dimensions of the texture and the canvas object match those of the quad in order to get pixel-accurate results with the appropriate orthographic projection. The resulting rectangle Shape is positioned at the origin. It can be moved around by setting the localMatrix on the Transform object referenced to by the canvasQuad.transform property. The Canvas associated with the returned CanvasQuad object can be retrieved from the object's 'canvas' property. After issuing any draw commands on the Canvas, you need to call the updateTexture() method on the CanvasQuad to update the contents of the quad surface.

### Parameters:

*canvasInfo* The CanvasInfo object instance creating this CanvasQuad.  
*width* The width of the quad.  
*height* The height of the quad.  
*transparent* Set to true if the canvas will be transparent so that the appropriate blending modes are set.  
*opt\_parent* parent transform to parent the newly created quad under. If no parent transform is provided then the quad gets parented under the CanvasInfo's root.

---

## Member Function Documentation

o3djs.canvas.CanvasQuad.updateTexture ( )

Copies the current contents of the Canvas object to the texture associated with the quad. This method should be called after any new draw calls have been issued to the CanvasQuad's Canvas object.

---

## Member Property Documentation

[!o3d.Canvas](#) o3djs.canvas.CanvasQuad.canvas

The Canvas object used to draw on this quad.

## [!o3d.Transform](#) o3djs.canvas.CanvasQuad.scaleTransform

A scale transform for this quad. You can change the scale the quad without effecting its positon using this transform.

## [!o3d.Transform](#) o3djs.canvas.CanvasQuad.transform

A transform for this quad.

# **o3djs.debug.DebugHelper Class Reference**

[List of all members.](#)

---

## Detailed Description

A Debug object to help with debugging o3d apps.

A debug helper object provides functions to help debug your o3d application and manages the resources needed to do that for you. For example it can add axes, spheres and boxes to your transforms as well as draw lines in 3d space given 2 points.

## Source

[o3djs/debug.js](#)

## Constructor

- [o3djs.debug.DebugHelper\(pack, viewInfo\)](#)

## Public Member Functions

- [addAxes\(treeRoot\)](#)
- [addAxis\(transform\)](#)
- [addCube\(transform, opt\\_color, opt\\_scale\)](#)
- [addCubes\(treeRoot\)](#)
- [addSphere\(transform, opt\\_color, opt\\_scale\)](#)
- [addSpheres\(treeRoot\)](#)
- [clearAxisColor\(transform\)](#)
- [!o3d.Transform createCube\(position, opt\\_color, opt\\_scale\)](#)
- [!o3djs.debug.DebugLineGroup createDebugLineGroup\(root\)](#)
- [!o3d.Transform createSphere\(position, opt\\_color, opt\\_scale\)](#)
- [!o3d.Shape getLineShape\(\)](#)
- [!o3d.Pack getPack\(\)](#)
- [removeAxes\(treeRoot\)](#)
- [removeAxis\(transform\)](#)
- [removeCube\(transform\)](#)
- [removeCubes\(treeRoot\)](#)

- [removeSphere](#)(transform)
  - [removeSpheres](#)(treeRoot)
  - [setAxisColor](#)(transform, color)
  - [setAxisScale](#)(length, width)
  - [setCubeColor](#)(transform, color)
  - [setCubeScale](#)(transform, scale)
  - [setSphereColor](#)(transform, color)
  - [setSphereScale](#)(transform, scale)
- 

## Constructor

```
o3djs.debug.DebugHelper ( !o3d.Pack pack  
                          !o3djs.rendergraph.viewInfo viewInfo )
```

A Debug object to help with debugging o3d apps.

A debug helper object provides functions to help debug your o3d application and manages the resources needed to do that for you. For example it can add axes, spheres and boxes to your transforms as well as draw lines in 3d space given 2 points.

### Parameters:

*pack* Pack for this debug object to use to manage its resources.  
*viewInfo* ViewInfo for debug visuals.

---

## Member Function Documentation

```
o3djs.debug.DebugHelper.addAxes ( !o3d.Transform treeRoot )
```

Adds axes to all transform in a tree.

### Parameters:

*treeRoot* root of tree to add axes to.

```
o3djs.debug.DebugHelper.addAxis ( !o3d.Transform transform )
```

Adds an axis to a transform.

### Parameters:

*transform* Transform to add axis to.

```
o3djs.debug.DebugHelper.addCube ( !o3d.Transform transform  
                              !o3djs.math.Vector4 opt_color  
                              number                 opt_scale )
```

Adds a cube to a transform.

### Parameters:

*transform* Transform to add cube to.

*opt\_color* RGBA color for cube.

*opt\_scale* of cube.

```
o3djs.debug.DebugHelper.addCubes ( !o3d.Transform treeRoot )
```

Adds cubes to all transform in a tree.

## Parameters:

*treeRoot* root of tree to add cubes to.

Adds a sphere to a transform.

### Parameters:

*transform* Transform to add sphere to.

*opt\_color* RGBA color for sphere.

*opt scale* of sphere.

```
o3djs.debug.DebugHelper.addSpheres ( !o3d.Transform treeRoot )
```

Adds spheres to all transform a tree.

### Parameters:

*treeRoot* root of tree to add spheres to.

```
o3djs.debug.DebugHelper.clearAxisColor ( !o3d.Transform transform )
```

**RemoveColor** Removes the color from an axis.

### Parameters:

*transform* Transform on which to remove color.

Creates a cube at a world position..

### Parameters:

*position* Position at which to create sphere.

*opt\_color* RGBA color for cube.

*opt scale* of cube.

### Returns:

`!o3d.Transform.transform` for cube.

`!o3djs.debug.DebugLineGroup`

o3dis.debug.DebugHelper.createDebugLineGroup

Creates a debug line group. A Debug line group's purpose is the let you quickly delete a set of lines.

### Parameters:

*root* Root transform to use for lines

788

## Returns:

!o3dis.debug.DebugLineGroup. The debug line group.

```
!o3d.Transform o3djs.debug.DebugHelper.createSphere ( !o3djs.math.Vector3 position  
                         !o3djs.math.Vector4 opt_color  
                         number                         opt scale )
```

Creates a sphere in world space.

**Parameters:**

*position* Position at which to create sphere.  
*opt\_color* RGBA color for sphere.  
*opt\_scale* of sphere.

**Returns:**

[!o3d.Transform](#).transform for sphere.

`o3djs.debug.DebugHelper.getLineShape ( )`

Gets the line shape.

**Returns:**

[!o3d.Shape](#).The shape for debug lines.

`o3djs.debug.DebugHelper.getPack ( )`

Gets the pack for this DebugHelper.

**Returns:**

[!o3d.Pack](#).The pack for this DebugHelper.

`o3djs.debug.DebugHelper.removeAxes ( !o3d.Transform treeRoot )`

Removes axes from all transforms in a tree.

**Parameters:**

*treeRoot* root of tree to remove axes from.

`o3djs.debug.DebugHelper.removeAxis ( !o3d.Transform transform )`

Removes an axis from a transform

**Parameters:**

*transform* Transform to remove axis from.

`o3djs.debug.DebugHelper.removeCube ( !o3d.Transform transform )`

Removes a cube from a transform

**Parameters:**

*transform* Transform to remove cube from.

`o3djs.debug.DebugHelper.removeCubes ( !o3d.Transform treeRoot )`

Removes cubes from all transforms in a tree.

**Parameters:**

*treeRoot* root of tree to remove cubes from.

`o3djs.debug.DebugHelper.removeSphere ( !o3d.Transform transform )`

Removes a sphere from a transform

**Parameters:**

*transform* Transform to remove sphere from.

`o3djs.debug.DebugHelper.removeSpheres ( !o3d.Transform treeRoot )`

Removes spheres from all transforms in a tree.

**Parameters:**

*treeRoot* root of tree to remove spheres from.

`o3djs.debug.DebugHelper.setAxisColor ( !o3d.Transform transform )`

[!o3djs.math.Vector4](#) *color* )

Set axis color.

**Parameters:**

*transform* Transform on which to change axis color.  
*color* 4 number array in RGBA format.

o3djs.debug.DebugHelper.setAxisScale ( number *length*  
                                  number *width* )

Sets the length and width of the axis lines.

**Parameters:**

*length* Length of an axis in the direction of the  
                                 axis.

*width* Width of the axis or its thickness.

o3djs.debug.DebugHelper.setCubeColor ( [!o3d.Transform](#) *transform*  
                                  [!o3djs.math.Vector3](#) *color* )

Set cube color.

**Parameters:**

*transform* Transform on which to change cube color.  
*color* 4 number array in RGBA format.

o3djs.debug.DebugHelper.setCubeScale ( [!o3d.Transform](#) *transform*  
                                  number         *scale* )

Sets the scale of a cubes.

**Parameters:**

*transform* Transform on which to change cube scale.  
*scale* Scale to make the cube.

o3djs.debug.DebugHelper.setSphereColor ( [!o3d.Transform](#) *transform*  
                                  [!o3djs.math.Vector4](#) *color* )

Set sphere color.

**Parameters:**

*transform* Transform on which to change sphere color.  
*color* 4 number array in RGBA format.

o3djs.debug.DebugHelper.setSphereScale ( [!o3d.Transform](#) *transform*  
                                  number         *scale* )

Sets the scale of a debug sphere.

**Parameters:**

*transform* Transform on which to change sphere scale.  
*scale* Scale to make the sphere.

## **o3djs.debug.DebugLine Class Reference**

[List of all members.](#)

---

# Detailed Description

An object to manage a single debug line.

## Source

[o3djs/debug.js](#)

## Constructor

- [o3djs.debug.DebugLine\(debugLineGroup\)](#)

## Public Member Functions

- [destroy\(\)](#)
  - number [getId\(\)](#)
  - [remove\(\)](#)
  - [setColor\(color\)](#)
  - [setEnd\(end\)](#)
  - [setEndPoints\(start, end\)](#)
  - [setStart\(start\)](#)
  - [setVisible\(visible\)](#)
- 

## Constructor

`o3djs.debug.DebugLine ( !o3djs.debug.DebugLineGroup debugLineGroup )`

An object to manage a single debug line.

### Parameters:

`debugLineGroup` DebugLineGroup this line belongs too.

---

## Member Function Documentation

`o3djs.debug.DebugLine.destroy ( )`

Destroys this line object cleaning up the resources used by it.

`o3djs.debug.DebugLine.getId ( )`

Returns a unique Id for the line.

### Returns:

number.the Id of the line.

`o3djs.debug.DebugLine.remove ( )`

Removes this line.

`o3djs.debug.DebugLine.setColor ( !o3djs.math.Vector4 color )`

Sets the color of the DebugLine.

**Parameters:**

*color* The color of the debug line.

`o3djs.debug.DebugLine.setEnd ( !o3djs.math.Vector3 end )`

Sets the end point of the DebugLine.

**Parameters:**

*end* End point for line.

`o3djs.debug.DebugLine.setEndPoints ( !o3djs.math.Vector3 start  
!o3djs.math.Vector3 end )`

Sets the end points of the DebugLine.

**Parameters:**

*start* Start point for line.

*end* End point for line.

`o3djs.debug.DebugLine.setStart ( !o3djs.math.Vector3 start )`

Sets the start point of the DebugLine.

**Parameters:**

*start* Start point for line.

`o3djs.debug.DebugLine.setVisible ( boolean visible )`

Sets the visibility of the DebugLine.

**Parameters:**

*visible* True = visible.

# **o3djs.debug.DebugLineGroup Class Reference**

[List of all members.](#)

---

## Detailed Description

An object to manage debug lines.

## Source

[o3djs/debug.js](#)

## Constructor

- [o3djs.debug.DebugLineGroup](#)(debugHelper, root)

## Public Member Functions

- [!o3djs.debug.debugLine addLine\(opt\\_start, opt\\_end, opt\\_color\)](#)

- [clear\(\)](#)
  - [destroy\(\)](#)
  - [!o3djs.math.Vector4 getColor\(\)](#)
  - [!o3d.Shape getLineShape\(\)](#)
  - [!o3d.Pack getPack\(\)](#)
  - [!o3d.Transform getRoot\(\)](#)
  - [remove\(line\)](#)
  - [setColor\(color\)](#)
- 

## Constructor

`o3djs.debug.DebugLineGroup ( !o3djs.debug.DebugHelper debugHelper  
  root  )`

An object to manage debug lines.

### Parameters:

*debugHelper* The DebugHelper associated with this object.  
*root*               Transform to put debug lines under.

---

## Member Function Documentation

[!o3djs.debug.debugLine](#) `o3djs.debug.DebugLineGroup.addLine ( !o3djs.math.Vector3 opt_start  
  opt_end  
  opt_color )`

Adds a debug line.

### Parameters:

*opt\_start* Start position for line.  
*opt\_end* End position for line.  
*opt\_color* Color for line.

### Returns:

[!o3djs.debug.debugLine](#).The DebugLine.

`o3djs.debug.DebugLineGroup.clear ( )`

Clears all the lines in this group.

`o3djs.debug.DebugLineGroup.destroy ( )`

Destroys a DeubgLineGroup, freeing all its lines and resources.

`o3djs.debug.DebugLineGroup.getColor ( )`

Gets the current color for this line group.

### Returns:

[!o3djs.math.Vector4](#).The current color.

`o3djs.debug.DebugLineGroup.getLineShape ( )`

Gets the shape for lines.

**Returns:**

[!o3d.Shape](#).The shape for lines.

`o3djs.debug.DebugLineGroup.getPack ( )`

Gets the pack for this line group.

**Returns:**

[!o3d.Pack](#).The pack for this line group.

`o3djs.debug.DebugLineGroup.getRoot ( )`

Gets the root transform for this line group.

**Returns:**

[!o3d.Transform](#).The root transform for this line group.

`o3djs.debug.DebugLineGroup.remove ( !o3djs.debug.DebugLine line )`

Removes a line.

**Parameters:**

*line* Line to remove.

`o3djs.debug.DebugLineGroup.setColor ( !o3djs.math.Vector4 color )`

Sets the current color for this line group. All lines added after setting this will be this color by default.

**Parameters:**

*color* The color for this line group.

## **o3djs.debug.VertexInfo Class Reference**

[List of all members.](#)

---

### **Detailed Description**

`VertexInfo`. Used to store vertices and indices.

### **Source**

[o3djs/debug.js](#)

### **Constructor**

- [o3djs.debug.VertexInfo\(opt\\_vertices, opt\\_indices\)](#)

## Public Member Functions

- [addLine\(index1, index2\)](#)
- [addVertex\(positionX, positionY, positionZ\)](#)
- [!o3d.Shape createShape\(pack, material\)](#)
- [reorient\(matrix\)](#)

## Public Properties

- undefined [Offset](#)
- 

## Constructor

```
o3djs.debug.VertexInfo ( !Array.<!o3djs.math.Vector3> opt_vertices  
                        !Array.<number>                  opt_indices )
```

VertexInfo. Used to store vertices and indices.

### Parameters:

*opt\_vertices* array of numbers in the format positionX, positionY,  
positionZ.  
*opt\_indices* array of indices in pairs.

---

## Member Function Documentation

```
o3djs.debug.VertexInfo.addLine ( number index1  
                                number index2 )
```

Adds a line.

### Parameters:

*index1* The index of the first vertex of the triangle.  
*index2* The index of the second vertex of the triangle.

```
o3djs.debug.VertexInfo.addVertex ( number positionX  
                                    number positionY  
                                    number positionZ )
```

Adds a vertex.

### Parameters:

*positionX* The x position of the vertex.  
*positionY* The y position of the vertex.  
*positionZ* The z position of the vertex.

```
!o3d.Shape o3djs.debug.VertexInfo.createShape ( !o3d.Pack      pack  
                                                !o3d.Material material )
```

Creates a shape from a VertexInfo

### Parameters:

*pack* Pack to create objects in.  
*material* to use.

**Returns:**

[!o3d.Shape](#).The created shape.

`o3djs.debug.VertexInfo.reorient ( !o3djs.math.Matrix4 matrix )`

Reorients the vertex positions of this vertexInfo by the given matrix. In other words it multiplies each vertex by the given matrix.

**Parameters:**

*matrix* Matrix to multiply by.

---

## Member Property Documentation

`undefined o3djs.debug.VertexInfo.Offset`

Enum for vertex component offsets.

To get/set a vertex component you can do something like this.

```
vertInfo.vertices[vertInfo.vertexIndex(index) + vertInfo.Offset.X] *= 2;
```

## o3djs.error.ErrorCollector Class Reference

[List of all members.](#)

---

### Detailed Description

An ErrorCollector takes over the client error callback and continues to collect errors until ErrorCollector.finish() is called.

### Source

[o3djs/error.js](#)

### Constructor

- [o3djs.error.ErrorCollector\(client\)](#)

### Public Member Functions

- [finish\(\)](#)

## Public Properties

- !Array.<string> [errors](#)
- 

## Constructor

`o3djs.error.ErrorCollector ( !o3d.Client client )`

An ErrorCollector takes over the client error callback and continues to collect errors until `ErrorCollector.finish()` is called.

### Parameters:

*client* The client object of the plugin.

---

## Member Function Documentation

`o3djs.error.ErrorCollector.finish ( )`

Stops the ErrorCollector from collecting errors and restores the previous error callback.

---

## Member Property Documentation

`!Array.<string> o3djs.error.ErrorCollector.errors`

The collected errors.

## **o3djs.fps.ColorRect Class Reference**

[List of all members.](#)

---

## Detailed Description

A Class the manages a color rect.

## Source

[o3djs/fps.js](#)

## Constructor

- [o3djs.fps.ColorRect](#)(pack, shape, parent, x, y, z, width, height, color)

## Public Member Functions

- [setColor\(color\)](#)
  - [setPosition\(x, y\)](#)
  - [setSize\(width, height\)](#)
- 

## Constructor

```
o3djs.fps.ColorRect ( !o3d.Pack      pack
                      !o3d.Shape      shape
                      !o3d.Transform  parent
                      number          x
                      number          y
                      number          z
                      number          width
                      number          height
                      !o3djs.math.Vector4 color )
```

A Class the manages a color rect.

### Parameters:

*pack* Pack to create things in.  
*shape* Shape to use for rectangle.  
*parent* Transform to parent rect under.  
*x* initial x position.  
*y* initial y position.  
*z* initial z position.  
*width* initial width.  
*height* initial height.  
*color* initial color.

---

## Member Function Documentation

`o3djs.fps.ColorRect.setColor ( !o3djs.math.Vector4 color )`

Sets the color of this ColorRect.

### Parameters:

*color* initial color.

`o3djs.fps.ColorRect.setPosition ( number x
 number y )`

Sets the position of this ColorRect.

### Parameters:

*x* x position.  
*y* y position.

```
o3djs.fps.ColorRect.setSize ( number width  
                           number height )
```

Sets the size of this ColorRect

**Parameters:**

*width* width.  
*height* height.

## o3djs.fps.FPSManager Class Reference

[List of all members.](#)

---

### Detailed Description

A class for displaying frames per second.

### Source

[o3djs/fps.js](#)

### Constructor

- [o3djs.fps.FPSManager](#)(pack, clientWidth, clientHeight, opt\_parent)

### Public Member Functions

- [resize](#)(clientWidth, clientHeight)
- [setPerfVisible](#)(visible)
- [setPosition](#)(x, y)
- [setVisible](#)(visible)
- [update](#)(renderEvent)

### Public Properties

- undefined [fpsQuad](#)
  - ![o3djs.rendergraph.ViewInfo](#) [viewInfo](#)
- 

### Constructor

```
o3djs.fps.FPSManager ( !o3d.Pack           pack  
                      number            clientWidth  
                      number            clientHeight  
                      !o3d.RenderNode opt_parent  )
```

A class for displaying frames per second.

**Parameters:**

*pack* Pack to create objects in.  
*clientWidth* width of client area.  
*clientHeight* Height of client area.  
*opt\_parent* RenderNode to use as parent for ViewInfo that will be used to render the FPS with.

---

## Member Function Documentation

`o3djs.fps.FPSManager.resize ( number clientWidth  
                              number clientHeight )`

Resizes the area for the FPS display. Note: you must call this if your client area changes size.

**Parameters:**

*clientWidth* width of client area.  
*clientHeight* height of client area.

`o3djs.fps.FPSManager.setPerfVisible ( boolean visible )`

Sets the visibility of the performance bar.

**Parameters:**

*visible* true = visible.

`o3djs.fps.FPSManager.setPosition ( number x  
                              number y )`

Sets the position of the FPS display

The position is in pixels assuming the size of the client matches the size last set either on creation or with `FPSManager.resize`.

**Parameters:**

*x* The x position.  
*y* The y position.

`o3djs.fps.FPSManager.setVisible ( boolean visible )`

Sets the visibility of the fps display.

**Parameters:**

*visible* true = visible.

`o3djs.fps.FPSManager.update ( !o3d.RenderEvent renderEvent )`

Updates the fps display. You must call this every frame to update the FPS display.

```
...  
client.setRenderCallback(onRender);  
...  
function onRender(renderEvent) {  
    myFpsManager.update(renderEvent);  
}
```

**Parameters:**

*renderEvent* The object passed into the render callback.

---

## Member Property Documentation

undefined o3djs.fps.FPSManager.fpsQuad

The quad used to display the FPS.

[!o3djs.rendergraph.ViewInfo](#) o3djs.fps.FPSManager.viewInfo

The ViewInfo to display FPS.

## o3djs.io.ArchiveInfo Class Reference

[List of all members.](#)

---

### Detailed Description

A ArchiveInfo object loads and manages an archive of files. There are methods for locating a file by uri and for freeing the archive.

You can only read archives that have as their first file a file named 'aaaaaaaa.o3d' the contents of the which is 'o3d'. This is to prevent O3D from being used to read arbitrary tar gz files.

Example:

```
var loadInfo = o3djs.io.loadArchive(pack,
                                      'http://google.com/files.o3dtgz',
                                      callback);
function callback(archiveInfo, exception) {
    if (!exception) {
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('logo.jpg'), true);
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('wood/oak.png'), true);
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('wood/maple.dds'), true);
        archiveInfo.destroy();
    } else {
        alert(exception);
    }
}
```

## Source

[o3djs/io.js](#)

## Constructor

- [o3djs.io.ArchiveInfo](#)(pack, url, onFinished)

## Public Member Functions

- [destroy\(\)](#)
- ([o3d.RawData](#)|undefined) [getFileByURI](#)(uri, opt\_caseInsensitive)
- !Array.<!a3d.RawData> [getFiles](#)(uri, opt\_caseInsensitive)

## Public Properties

- boolean [destroyed](#)
  - !Object [files](#)
  - [!o3djs.io.LoadInfo loadInfo](#)
  - [!o3d.Pack pack](#)
- 

## Constructor

```
o3djs.io.ArchiveInfo ( !o3d.Pack           pack
                      string            url
                      !function(!o3djs.io.ArchiveInfo, *): void onFinished )
```

A ArchiveInfo object loads and manages an archive of files. There are methods for locating a file by uri and for freeing the archive.

You can only read archives that have as their first file a file named 'aaaaaaaa.o3d' the contents of the which is 'o3d'. This is to prevent O3D from being used to read arbitrary tar gz files.

Example:

```
var loadInfo = o3djs.io.loadArchive(pack,
                                      'http://google.com/files.o3dtgz',
                                      callback);
function callback(archiveInfo, exception) {
    if (!exception) {
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('logo.jpg'), true);
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('wood/oak.png'), true);
        pack.createTextureFromRawData(
            archiveInfo.getFileByURI('wood/maple.dds'), true);
        archiveInfo.destroy();
    } else {
        alert(exception);
    }
}
```

Parameters:

*pack*      Pack to create request in.

*url* The url of the archive file.

*onFinished* A callback that is called when the archive is finished loading and passed the ArchiveInfo and an exception which is null on success.

---

## Member Function Documentation

`o3djs.ioArchiveInfo.destroy()`

Releases all the RAW data associated with this archive. It does not release any objects created from that RAW data.

`(o3d.RawData|undefined) o3djs.ioArchiveInfo.getFileByURI ( string uri  
boolean opt_caseInsensitive )`

Gets a file by URI from the archive.

**Parameters:**

*uri* of file to get.

*opt\_caseInsensitive* True to be case insensitive. Default false.

**Returns:**

`(o3d.RawData|undefined)`.The RawData for the file or undefined if it doesn't exist.

`!Array.<!o3d.RawData> o3djs.ioArchiveInfo.GetFiles ( (string|!RegExp) uri  
boolean opt_caseInsensitive )`

Gets files by regular expression or wildcards from the archive.

**Parameters:**

*uri* of file to get. Can have wildcards '\*' and '?'.

*opt\_caseInsensitive* Only valid if it's a wildcard string.

**Returns:**

`!Array.<!o3d.RawData>`.An array of the matching RawDatas for the files matching or undefined if it doesn't exist.

---

## Member Property Documentation

`boolean o3djs.ioArchiveInfo.destroyed`

True if this archive has not been destroyed.

`!Object o3djs.ioArchiveInfo.files`

The list of files in the archive by uri.

`!o3djs.io.LoadInfo o3djs.ioArchiveInfo.loadInfo`

The LoadInfo to track loading progress.

`!o3d.Pack o3djs.ioArchiveInfo.pack`

The pack used to create the archive request.

# o3djs.io.LoadInfo Class Reference

[List of all members.](#)

---

## Detailed Description

A class to help with progress reporting for most loading utilities.

Example:

```
var g_loadInfo = null;
g_id = window.setInterval(statusUpdate, 500);
g_loadInfo = o3djs.scene.loadScene(client, pack, parent,
                                    'http://google.com/somescene.o3dtgz',
                                    callback);
function callback(pack, parent, exception) {
    g_loadInfo = null;
    window.clearInterval(g_id);
    if (!exception) {
        // do something with scene just loaded
    }
}
function statusUpdate() {
    if (g_loadInfo) {
        var progress = g_loadInfo.getKnownProgressInfoSoFar();
        document.getElementById('loadstatus').innerHTML = progress.percent;
    }
}
```

## Source

[o3djs/io.js](#)

## See Also

- [o3djs.scene.loadScene](#)
- [o3djs.io.loadArchive](#)
- [o3djs.io.loadTexture](#)
- [o3djs.loader.Loader](#)

## Constructor

- [o3djs.io.LoadInfo\(opt\\_request, opt\\_hasStatus\)](#)

## Public Member Functions

- [addChild\(loadInfo\)](#)
- [finish\(\)](#)
- [{{percent: number, downloaded: string, totalBytes: string, base: number, suffix: string}}](#)

- number `getKnownProgressInfoSoFar()`
- number `getTotalBytesDownloaded()`
- number `getTotalKnownBytesToStreamSoFar()`
- number `getTotalKnownRequestsToStreamSoFar()`
- number `getTotalRequestsDownloaded()`

# Constructor

```
o3djs.io.LoadInfo ( (!o3d.ArchiveRequest!o3d.FileRequest!XMLHttpRequest) opt_request  
                      boolean                                opt_hasStatus )
```

A class to help with progress reporting for most loading utilities.

## Example:

```
var g_loadInfo = null;
g_id = window.setInterval(statusUpdate, 500);
g_loadInfo = o3djs.scene.loadScene(client, pack, parent,
                                    'http://google.com/somescene.o3dtgz',
                                    callback);
function callback(pack, parent, exception) {
    g_loadInfo = null;
    window.clearInterval(g_id);
    if (!exception) {
        // do something with scene just loaded
    }
}
function statusUpdate() {
    if (g_loadInfo) {
        var progress = g_loadInfo.getKnownProgressInfoSoFar();
        document.getElementById('loadstatus').innerHTML = progress.percent;
    }
}
```

## Parameters:

*opt\_request* The request to watch.

`opt_hasStatus` true if `opt_request` is a `o3d.ArchiveRequest` vs for example an `o3d.FileRequest` or  
or  
`s` an XMLHttpRequest.

# Member Function Documentation

`o3djs.io.LoadInfo.addChild ( !o3djs.io.LoadInfo loadInfo )`

Adds another LoadInfo as a child of this LoadInfo so they can be managed as a group.

## Parameters:

*loadInfo* The child LoadInfo.

o3djs.io.LoadInfo.finish ( )

Marks this LoadInfo as finished.

```
o3djs.io.LoadInfo.getKnownProgressInfoSoFar( )
```

Gets progress info. This is commonly formatted version of the information available from a LoadInfo.

See LoadInfo.getTotalKnownBytesToStreamSoFar for details.

**Returns:**

`{ {percent: number, downloaded: string, totalBytes: string, base: number, suffix: string} } .progress info.`

**See Also:**

- [o3djs.io.LoadInfo.getTotalKnownBytesToStreamSoFar](#)

`o3djs.io.LoadInfo.getTotalBytesDownloaded ( )`

Gets the total bytes downloaded so far.

**Returns:**

`number`.The total number of currently known bytes to be streamed.

`o3djs.io.LoadInfo.getTotalKnownBytesToStreamSoFar ( )`

Gets the total bytes that will be streamed known so far. If you are only streaming 1 file then this will be the info for that file but if you have queued up many files using an o3djs.loader.Loader only a couple of files are streamed at a time meaning that the size is not known for files that have yet started to download.

If you are downloading many files for your application and you want to provide a progress status you have about 4 options

1) Use LoadInfo.getTotalBytesDownloaded() / LoadInfo.getTotalKnownBytesToStreamSoFar() and just be aware the bar will grow and then shrink as new files start to download and their lengths become known.

2) Use LoadInfo.getTotalRequestsDownloaded() / LoadInfo.getTotalKnownRequestsToStreamSoFar() and be aware the granularity is not all that great since it only reports fully downloaded files. If you are downloading a bunch of small files this might be ok.

3) Put all your files in one archive. Then there will be only one file and method 1 will work well.

4) Figure out the total size in bytes of the files you will download and put that number in your application, then use LoadInfo.getTotalBytesDownloaded() /  
`MY_APPS_TOTAL_BYTES_TO_DOWNLOAD`.

**Returns:**

`number`.The total number of currently known bytes to be streamed.

`o3djs.io.LoadInfo.getTotalKnownRequestsToStreamSoFar ( )`

Gets the total streams that will be download known so far. We can't know all the streams since you could use an o3djs.loader.Loader object, request some streams, then call this function, then request some more.

See LoadInfo.getTotalKnownBytesToStreamSoFar for details.

**Returns:**

number.The total number of requests currently known to be streamed.

**See Also:**

- [o3djs.io.LoadInfo.getTotalKnownBytesToStreamSoFar](#)

`o3djs.io.LoadInfo.getTotalRequestsDownloaded ( )`

Gets the total requests downloaded so far.

**Returns:**

number.The total requests downloaded so far.

# o3djs.loader.Loader Class Reference

[List of all members.](#)

---

## Detailed Description

A simple Loader class to call some callback when everything has loaded.

## Source

[o3djs/loader.js](#)

## Constructor

- [o3djs.loader.Loader\(onFinished\)](#)

## Public Member Functions

- [!o3djs.loader.Loader createLoader\(onFinished\)](#)
- [finish\(\)](#)
- [loadScene\(client, pack, parent, url, opt\\_options, opt\\_onSceneLoaded\)](#)
- [loadTextFile\(url, onTextLoaded\)](#)
- [loadTexture\(pack, url, opt\\_onTextureLoaded\)](#)

## Public Properties

- [!o3djs.io.LoadInfo loadInfo](#)
- 

## Constructor

`o3djs.loader.Loader ( !function(): void onFinished )`  
A simple Loader class to call some callback when everything has loaded.

## Parameters:

*onFinished* Function to call when final item has loaded.

# Member Function Documentation

**!o3djs.loader.Loader** o3djs.loader.Loader.createLoader ( !function(): void *onFinished* )

Creates a loader that is tracked by this loader so that when the new loader is finished it will be reported to this loader.

## Parameters:

*onFinished* Function to be called when everything loaded with this loader has finished.

## Returns:

# **!o3djs.loader.Loader**.The new Loader.

```
o3djs.loader.Loader.finish( )
```

Finishes the loading process. Actually this just calls countDown to account for the count starting at 1.

```
o3djs.loader.Loader.loadScene ( !o3d.Client client  
                                !o3d.Pack pack  
                                !o3d.Transform parent  
                                string url  
                                !o3djs.serialization.Options opt_options  
                                !function(!o3d.Pack, !o3d.Transform, *): opt_onSceneLoaded )  
                                void
```

Loads a 3d scene.

### Parameters:

*client* An O3D client object.

*pack* Pack to load texture into.

*parent* Transform to parent scene under.

*url* URL of scene to load.

*opt options* Options passed into the loader.

*opt\_onSceneLoad* optional callback when scene is loaded. It will be passed the pack and parented and an exception which is null on success.

`o3djs.loader.Loader.loadTextFile ( string url )`

```
!function(string, *); void onTextLoaded )
```

Loads a text file

### Parameters:

*url* URL of scene to load.

*onTextLoaded* Function to call when the file is loaded. It will be passed the contents of the file as a string and an exception which is null on success.

o3dis loader Loader loadTexture ( '03d Pack pack

```
string url  
!function(o3d.Texture, *): void opt_onTextureLoaded )
```

Loads a texture.

#### Parameters:

<i>pack</i>	Pack to load texture into.
<i>url</i>	URL of texture to load.
<i>opt_onTextureLoaded</i>	optional callback when texture is loaded. It will be passed the texture and an exception which is null on success.

---

## Member Property Documentation

[!o3djs.io.LoadInfo](#) o3djs.loader.Loader.loadInfo

The LoadInfo for this loader you can use to track progress.

# o3djs.particles.OneShot Class Reference

[List of all members.](#)

---

## Detailed Description

An object to manage a particle emitter instance as a one shot. Examples of one shot effects are things like an explosion, some fireworks.

## Source

[o3djs/particles.js](#)

## Constructor

- [o3djs.particles.OneShot\(emitter, opt\\_parent\)](#)

## Public Member Functions

- [setParent\(parent\)](#)
- [trigger\(opt\\_position, opt\\_parent\)](#)

## Public Properties

- [!o3d.Transform transform](#)
-

## Constructor

```
o3djs.particles.OneShot ( !o3djs.particles.ParticleEmitter emitter
                           !o3d.Transform           opt_parent )
```

An object to manage a particle emitter instance as a one shot. Examples of one shot effects are things like an explosion, some fireworks.

### Parameters:

*emitter* The emitter to use for the one shot.  
*opt\_parent* The parent for this one shot.

---

## Member Function Documentation

```
o3djs.particles.OneShot.setParent ( !o3d.Transform parent )
```

Sets the parent transform for this OneShot.

### Parameters:

*parent* The parent for this one shot.

```
o3djs.particles.OneShot.trigger ( !o3djs.math.Vector3 opt_position
                                    !o3d.Transform       opt_parent )
```

Triggers the oneshot.

Note: You must have set the parent either at creation, with setParent, or by passing in a parent here.

### Parameters:

*opt\_position* The position of the one shot relative to its parent.  
*opt\_parent* The parent for this one shot.

---

## Member Property Documentation

[!o3d.Transform](#) o3djs.particles.OneShot.transform

Transform for OneShot.

## o3djs.particles.ParticleEmitter Class Reference

[List of all members.](#)

---

## Detailed Description

A ParticleEmitter

## Source

## o3djs/particles.js

# Constructor

- [o3djs.particles.ParticleEmitter](#)(particleSystem, opt\_texture, opt\_clockParam)

## Public Member Functions

- `!o3djs.particles.OneShot createOneShot(opt_parent)`
  - `setColorRamp(colorRamp)`
  - `setParameters(parameters, opt_perParticleParamSetter)`
  - `setState(stateId)`

## Public Properties

- [!o3d.ParamFloat](#) [clockParam](#)
  - [!o3d.Matрerial](#) [material](#)
  - [!o3djs.particles.ParticleSystem](#) [particleSystem](#)
  - [!o3d.Shape](#) [shape](#)

# Constructor

```
o3djs.particles.ParticleEmitter ( !o3djs.particles.ParticleSystem particleSystem  
                                !o3d.Texture                opt_texture  
                                !o3d.ParamFloat             opt_clockParam )
```

## A ParticleEmitter

### Parameters:

*particleSystem* The particle system to manage this emitter.  
*opt\_texture* The texture to use for the particles. If you don't supply a texture a default is provided.  
*opt\_clockParam* A ParamFloat to be the clock for the emitter.

# Member Function Documentation

!o3djs.particles.OneShot

( !o3d.Transform *opt parent* )

`o3djs.particles.ParticleEmitter.createOneShot`

Creates a OneShot particle emitter instance. You can use this for dust puffs, explosions, fireworks, etc...

## Parameters:

*opt\_parent* The parent for the oneshot.

## Returns:

**!o3djs.particles.OneShot**.A OneShot object.

```
o3djs.particles.ParticleEmitter.setColorRamp ( !Array.<number> colorRamp )
```

Sets the colorRamp for the particles. The colorRamp is used as a multiplier for the texture. When a particle starts it is multiplied by the first color, as it ages to progressed through the colors in the ramp.

```
particleEmitter.setColorRamp ( [  
    1, 0, 0, 1, // red  
    0, 1, 0, 1, // green  
    1, 0, 1, 0] ); // purple but with zero alpha
```

The code above sets the particle to start red, change to green then fade out while changing to purple.

#### Parameters:

*colorRamp* An array of color values in the form  
RGBA.

```
o3djs.particles.ParticleEmitter.setParameters ( !o3djs.particles.ParticleSpec parameters  
                                            !function(number, !  
                                                    o3djs.particles.ParticleSpec) opt_perParticleParamS )  
                                            : void
```

Sets the parameters of the particle emitter.

Each of these parameters are in pairs. They are used to create a table of particle parameters. For each particle a specific value is set like this

```
particle.field = value + Math.random() - 0.5 * valueRange * 2;
```

or in English

particle.field = value plus or minus valueRange.

So for example, if you wanted a value from 10 to 20 you'd pass 15 for value and 5 for valueRange because

15 + or - 5 = (10 to 20)

#### Parameters:

*parameters* The parameters for the emitters.  
*opt\_perParticleParamSe* A function that is called for each particle to allow its parameters to be  
adjusted per particle. The number is the index of the particle being  
created, in other words, if numParticles is 20 this value will be 0 to 19.  
The ParticleSpec is a spec for this particular particle. You can set any  
per particle value before returning.

```
o3djs.particles.ParticleEmitter.setState ( o3djs.particles.ParticleStateIds stateId )
```

Sets the blend state for the particles. You can use this to set the emitter to draw with BLEND, ADD,  
SUBTRACT, etc.

#### Parameters:

*stateId* The state you want.

---

# Member Property Documentation

[!o3d.ParamFloat](#) o3djs.particles.ParticleEmitter.clockParam

The param that is the source for the time for this emitter.

[!o3d.Materrial](#) o3djs.particles.ParticleEmitter.material

The material used by this emitter.

[!o3djs.particles.ParticleSystem](#) o3djs.particles.ParticleEmitter.particleSystem

The particle system managing this emitter.

[!o3d.Shape](#) o3djs.particles.ParticleEmitter.shape

The Shape used to render these particles.

# o3djs.particles.ParticleSpec Class Reference

[List of all members.](#)

---

## Detailed Description

A ParticleSpec specifies how to emit particles.

NOTE: For all particle functions you can specific a ParticleSpec as a Javascript object, only specifying the fields that you care about.

```
emitter.setParameters({
    numParticles: 40,
    lifeTime: 2,
    timeRange: 2,
    startSize: 50,
    endSize: 90,
    positionRange: [10, 10, 10],
    velocity: [0, 0, 60], velocityRange: [15, 15, 15],
    acceleration: [0, 0, -20],
    spinSpeedRange: 4
});
```

Many of these parameters are in pairs. For paired paramters each particle specific value is set like this

```
particle.field = value + Math.random() - 0.5 * valueRange * 2;
```

or in English

```
particle.field = value plus or minus valueRange.
```

So for example, if you wanted a value from 10 to 20 you'd pass 15 for value and 5 for valueRange because

15 + or - 5 = (10 to 20)

## Source

[o3djs/particles.js](#)

## Constructor

- [o3djs.particles.ParticleSpec\(\)](#)

## Public Properties

- [!o3djs.math.Vector3 acceleration](#)
- [!o3djs.math.Vector3 accelerationRange](#)
- boolean [billboard](#)
- [!o3djs.math.Vector4 colorMult](#)
- [colorMultRange colorMultRange](#)
- number [endSize](#)
- number [endSizeRange](#)
- number [frameDuration](#)
- number [frameStart](#)
- number [frameStartRange](#)
- number [lifeTime](#)
- number [lifeTimeRange](#)
- number [numFrames](#)
- number [numParticles](#)
- [!o3djs.quaternions.Quaternion orientation](#)
- [!o3djs.math.Vector3 position](#)
- [!o3djs.math.Vector3 positionRange](#)
- number [spinSpeed](#)
- number [spinSpeedRange](#)
- number [spinStart](#)
- number [spinStartRange](#)
- number [startSize](#)
- number [startSizeRange](#)
- number? [startTime](#)
- number [timeRange](#)
- [!o3djs.math.Vector3 velocity](#)
- [!o3djs.math.Vector3 velocityRange](#)
- [!o3djs.math.Vector3 worldAcceleration](#)
- [!o3djs.math.Vector3 worldVelocity](#)

---

## Constructor

`o3djs.particles.ParticleSpec ( )`

A ParticleSpec specifies how to emit particles.

NOTE: For all particle functions you can specific a ParticleSpec as a Javascript object, only specifying the fields that you care about.

```
emitter.setParameters({
  numParticles: 40,
  lifeTime: 2,
  timeRange: 2,
  startSize: 50,
  endSize: 90,
  positionRange: [10, 10, 10],
  velocity:[0, 0, 60], velocityRange: [15, 15, 15],
  acceleration: [0, 0, -20],
  spinSpeedRange: 4}
);
```

Many of these parameters are in pairs. For paired paramters each particle specific value is set like this

```
particle.field = value + Math.random() - 0.5 * valueRange * 2;
```

or in English

particle.field = value plus or minus valueRange.

So for example, if you wanted a value from 10 to 20 you'd pass 15 for value and 5 for valueRange because

15 + or - 5 = (10 to 20)

---

## Member Property Documentation

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.acceleration

The acceleration of a particle in local space.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.accelerationRange

The acceleration range.

boolean o3djs.particles.ParticleSpec.billboard

Whether these particles are oriented in 2d or 3d. true = 2d, false = 3d.

[!o3djs.math.Vector4](#) o3djs.particles.ParticleSpec.colorMult

The color multiplier of a particle.

colorMultRange o3djs.particles.ParticleSpec.colorMultRange

The color multiplier range.

number o3djs.particles.ParticleSpec.endSize

The ending size of a particle.

number o3djs.particles.ParticleSpec.endSizeRange

The ending size range.

number o3djs.particles.ParticleSpec.frameDuration

The frame duration at which to animate the particle texture in seconds per frame.

number o3djs.particles.ParticleSpec.frameStart

The initial frame to display for a particular particle.

number o3djs.particles.ParticleSpec.frameStartRange

The frame start range.

number o3djs.particles.ParticleSpec.lifeTime

The lifeTime of a particle.

number o3djs.particles.ParticleSpec.lifeTimeRange

The lifeTime range.

number o3djs.particles.ParticleSpec.numFrames

The number of frames in the particle texture.

number o3djs.particles.ParticleSpec.numParticles

The number of particles to emit.

[!o3djs.quaternions.Quaternion](#) o3djs.particles.ParticleSpec.orientation

The orientation of a particle. This is only used if billboard is false.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.position

The starting position of a particle in local space.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.positionRange

The starting position range.

number o3djs.particles.ParticleSpec.spinSpeed

The spin speed of a particle in radians.

number o3djs.particles.ParticleSpec.spinSpeedRange

The spin speed range.

number o3djs.particles.ParticleSpec.spinStart

The starting spin value for a particle in radians.

number o3djs.particles.ParticleSpec.spinStartRange

The spin start range.

number o3djs.particles.ParticleSpec.startSize

The starting size of a particle.

number o3djs.particles.ParticleSpec.startSizeRange

The starting size range.

number? o3djs.particles.ParticleSpec.startTime

The startTime of a particle.

number o3djs.particles.ParticleSpec.timeRange

The life time of the entire particle system. To make a particle system be continuous set this to match the lifeTime.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.velocity

The velocity of a paritcle in local space.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.velocityRange

The velocity range.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.worldAcceleration

The acceleration of all paritcles in world space.

[!o3djs.math.Vector3](#) o3djs.particles.ParticleSpec.worldVelocity

The velocity of all paritcles in world space.

# o3djs.particles.ParticleSystem Class Reference

[List of all members.](#)

---

## Detailed Description

An Object to manage Particles.

## Source

[o3djs/particles.js](#)

## Constructor

- [o3djs.particles.ParticleSystem](#)(pack, viewInfo, opt\_clockParam, opt\_randomFunction)

## Public Member Functions

- [!o3djs.particles.ParticleEmitter](#) [createParticleEmitter](#)(opt\_texture, opt\_clockParam)

## Public Properties

- [!o3d.ParamFloat](#) [clockParam](#)
- [!o3d.Texture2D](#) [defaultColorTexture](#)
- [!o3d.Texture2D](#) [defaultRampTexture](#)
- [!Array.<!o3d.Effect>](#) [effects](#)
- [!o3d.Pack](#) [pack](#)

- !Array.<!o3d.State> [particleStates](#)
  - ![o3djs.rendergraph.ViewInfo](#) [viewInfo](#)
- 

## Constructor

```
o3djs.particles.ParticleSystem ( !o3d.Pack pack
                                !o3djs.rendergraph.ViewInfo viewInfo
                                !o3d.ParamFloat opt_clockParam
                                opt_randomFunction )
```

An Object to manage Particles.

### Parameters:

<i>pack</i>	The pack for the particle system to manage resources.
<i>viewInfo</i>	A viewInfo so the particle system can do the default setup. The only thing used from viewInfo is the zOrderedDrawList. If that is not where you want your particles, after you create the particleEmitter use particleEmitter.material.drawList = myDrawList to set it to something else.
<i>opt_clockParam</i>	A ParamFloat to be the default clock for emitters of this particle system.
<i>opt_randomFunc tion</i>	

---

## Member Function Documentation

[!o3djs.particles.ParticleEmitter](#)  
o3djs.particles.ParticleSystem.createParticleEmitter ( ![o3d.Texture](#) *opt\_texture*  
! [o3d.ParamFloat](#) *opt\_clockPar  
am* )

Creates a particle emitter.

### Parameters:

<i>opt_texture</i>	The texture to use for the particles. If you don't supply a texture a default is provided.
<i>opt_clockParam</i>	A ParamFloat to be the clock for the emitter.

### Returns:

[!o3djs.particles.ParticleEmitter](#).The new emitter.

---

## Member Property Documentation

[!o3d.ParamFloat](#) o3djs.particles.ParticleSystem.clockParam  
The default ParamFloat to use as the clock for emitters created by this system.

[!o3d.Texture2D](#) o3djs.particles.ParticleSystem.defaultColorTexture

The default color texture for particles.

[!o3d.Texture2D](#) o3djs.particles.ParticleSystem.defaultRampTexture

The default ramp texture for particles.

[!Array.<!o3d.Effect>](#) o3djs.particles.ParticleSystem.effects

The effects for particles.

[!o3d.Pack](#) o3djs.particles.ParticleSystem.pack

The pack used to manage particle system resources.

[!Array.<!o3d.State>](#) o3djs.particles.ParticleSystem.particleStates

The states for the various blend modes.

[!o3djs.rendergraph.ViewInfo](#) o3djs.particles.ParticleSystem.viewInfo

The viewInfo that is used to get drawLists.

## o3djs.picking.PickInfo Class Reference

[List of all members.](#)

---

### Detailed Description

Creates a new PickInfo. Used to return picking information.

### Source

[o3djs/picking.js](#)

### Constructor

- [o3djs.picking.PickInfo](#)(shapeInfo, rayIntersectionInfo, worldIntersectionPosition)

### Public Properties

- [!o3d.RayInterstionInfo rayIntersectionInfo](#)
  - [!o3djs.picking.ShapeInfo shapeInfo](#)
  - [!o3djs.math.Vector3 worldIntersectionPosition](#)
- 

### Constructor

o3djs.picking.PickInfo ( [!o3djs.picking.ShapeInfo](#) *shapeInfo*  
[!o3d.RayIntersectionInfo](#) *rayIntersectionInfo*  
[!o3djs.math.Vector3](#) *worldIntersectionPosition* )

Creates a new PickInfo. Used to return picking information.

**Parameters:**

<i>shapeInfo</i>	The ShapeInfo that was picked.
<i>rayIntersectionInfo</i>	Information about the pick.
<i>worldIntersectionPosition</i>	world position of intersection.

---

## Member Property Documentation

[!o3d.RayInterstionInfo](#) o3djs.picking.PickInfo.rayIntersectionInfo

Information about the pick.

[!o3djs.picking.ShapeInfo](#) o3djs.picking.PickInfo.shapeInfo

The ShapeInfo that was picked.

[!o3djs.math.Vector3](#) o3djs.picking.PickInfo.worldIntersectionPosition

The worldIntersectionPosition world position of intersection.

## o3djs.picking.ShapeInfo Class Reference

[List of all members.](#)

---

### Detailed Description

Creates a new ShapeInfo. Used to store information about Shapes.

### Source

[o3djs/picking.js](#)

### Constructor

- [o3djs.picking.ShapeInfo\(shape, parent\)](#)

### Public Member Functions

- [dump\(prefix\)](#)
- [o3d.BoundingBox getBoundingBox\(\)](#)
- [o3djs.picking.PickInfo pick\(worldRay\)](#)
- [update\(\)](#)

### Public Properties

- [o3d.BoundingBox boundingBox](#)

- [!o3djs.picking.TransformInfo parent](#)
  - [!o3d.Shape shape](#)
- 

## Constructor

`o3djs.picking.ShapeInfo ( !o3d.Shape shape  
                                  !o3djs.picking.TransformInfo parent )`

Creates a new ShapeInfo. Used to store information about Shapes.

**Parameters:**

*shape* Shape to keep info about.  
*parent* Parent transform of the shape.

---

## Member Function Documentation

`o3djs.picking.ShapeInfo.dump ( string prefix )`

Dumps info about a ShapeInfo

**Parameters:**

*prefix* optional prefix for indenting.

`o3djs.picking.ShapeInfo.getBoundingBox ( )`

Gets the BoundingBox of the Shape in this ShapeInfo.

**Returns:**

[o3d.BoundingBox](#).The Shape's BoundingBox.

[o3djs.picking.PickInfo](#) `o3djs.picking.ShapeInfo.pick ( !o3djs.picking.Ray worldRay )`

Attempts to "pick" this Shape by checking for the intersection of a ray in world space to the triangles this shape uses.

**Parameters:**

*worldRay* A ray in world space to pick against.

**Returns:**

[o3djs.picking.PickInfo](#).Information about the picking. null if the ray did not intersect any triangles.

`o3djs.picking.ShapeInfo.update ( )`

Updates the BoundingBox of the Shape in this ShapeInfo.

---

## Member Property Documentation

[o3d.BoundingBox](#) `o3djs.picking.ShapeInfo.boundingBox`

The bounding box for this Shape

[!o3djs.picking.TransformInfo](#) o3djs.picking.ShapeInfo.parent

The parent TransformInfo of this Shape.

[!o3d.Shape](#) o3djs.picking.ShapeInfo.shape

The Shape for this ShapeInfo

# o3djs.picking.TransformInfo Class Reference

[List of all members.](#)

---

## Detailed Description

Creates a new TransformInfo. Used to store information about Transforms.

## Source

[o3djs/picking.js](#)

## Constructor

- [o3djs.picking.TransformInfo\(transform, parent\)](#)

## Public Member Functions

- [dump\(prefix\)](#)
- [o3d.BoundingBox getBoundingBox\(\)](#)
- [o3djs.picking.PickInfo pick\(worldRay\)](#)
- [update\(\)](#)

## Public Properties

- [o3djs.picking.TransformInfo parent](#)
  - [!o3d.Transform transform](#)
- 

## Constructor

o3djs.picking.TransformInfo ( [!o3d.Transform](#) *transform*  
[o3djs.picking.TransformInfo](#) *parent* )

Creates a new TransformInfo. Used to store information about Transforms.

### Parameters:

*transform* Transform to keep info about.

*parent* Parent transformInfo of the transform. Can be null.

---

## Member Function Documentation

`o3djs.picking.TransformInfo.dump ( string prefix )`

Dumps info about a TransformInfo

### Parameters:

*prefix* optional prefix for indenting.

`o3djs.picking.TransformInfo.getBoundingBox ( )`

Gets the BoundingBox of the Transform in this TransformInfo.

### Returns:

`o3d.BoundingBox`.The Transform's BoundingBox.

`o3djs.picking.PickInfo o3djs.picking.TransformInfo.pick ( !o3djs.picking.Ray worldRay )`

Attempts to "pick" this TransformInfo by checking for the intersection of a ray in world space to the boundingbox of the TransformInfo. If intesection is succesful recursively calls its children and shapes to try to find a single Shape that is hit by the ray.

### Parameters:

*worldRay* A ray in world space to pick against.

### Returns:

`o3djs.picking.PickInfo`.Information about the picking. null if the ray did not intersect any triangles.

`o3djs.picking.TransformInfo.update ( )`

Updates the shape and child lists for this TransformInfo and recomputes its BoundingBox.

---

## Member Property Documentation

`o3djs.picking.TransformInfo o3djs.picking.TransformInfo.parent`

The transform of this transform info.

`!o3d.Transform o3djs.picking.TransformInfo.transform`

The transform of this transform info.

## **`o3djs.primitives.VertexInfo` Class Reference**

[List of all members.](#)

---

## Detailed Description

`VertexInfo`. Used to store vertices and indices.

# Source

[o3djs/primitives.js](#)

## Constructor

- [o3djs.primitives.VertexInfo\(\)](#)

## Public Member Functions

- [!o3djs.primitives.VertexStreamInfo addStream](#)(numComponents, semantic, opt\_semanticIndex)
  - [addTangentStreams](#)(opt\_semanticIndex)
  - [addTriangle](#)(index1, index2, index3)
  - [!o3d.Shape createShape](#)(pack, material)
  - [o3djs.primitives.VertexStreamInfo findStream](#)(semantic, opt\_semanticIndex)
  - !Array.<number> [getTriangle](#)(triangleIndex)
  - number [numTriangles](#)()
  - [removeStream](#)(semantic, opt\_semanticIndex)
  - [reorient](#)(matrix)
  - [setTriangle](#)(triangleIndex, index1, index2, index3)
  - [validate](#)()
- 

## Constructor

`o3djs.primitives.VertexInfo ( )`

VertexInfo. Used to store vertices and indices.

---

## Member Function Documentation

[!o3djs.primitives.VertexStreamInfo](#)

`o3djs.primitives.VertexInfo.addStream`

( number

*numComponents*

!

[o3d.Stream.Semantic](#)

*semantic*

number

*opt\_semanticIndex*

*x*

Add a new stream to the VertexInfo, replacing it with a new empty one if it already exists.

### Parameters:

*numComponents* The number of components per vector.

*semantic* The semantic of the stream.

*opt\_semanticIndex* The semantic index of the stream. Defaults to zero.

### Returns:

[!o3djs.primitives.VertexStreamInfo](#).The new stream.

`o3djs.primitives.VertexInfo.addTangentStreams ( number opt_semanticIndex )`  
Calculate tangents and binormals based on the positions, normals and texture coordinates found in existing streams.

## Parameters:

*opt\_semanticInd* The semantic index of the texture coordinate to use and the tangent and binormal streams to add. Defaults to zero.  
*ex*

Adds a triangle.

### Parameters:

*index1* The index of the first vertex of the triangle.  
*index2* The index of the second vertex of the triangle.  
*index3* The index of the third vertex of the triangle.

Creates a shape from a VertexInfo

### Parameters:

*pack* Pack to create objects in.  
*material* to use

### Returns:

**!o3d.Shape**: The created shape.

[o3djs.primitives.VertexStreamInfo](#) ( !  
o3djs.primitives.VertexInfo.findStream [o3d.Stream.Semantic](#) semantic  
number *opt\_semanticInd\_ex* )

Find a stream in the VertexInfo

### Parameters:

*semantic* The semantic of the stream.  
*opt semanticIndex* The semantic index of the stream. Defaults to zero.

### Returns:

`o3djs.primitives.VertexStreamInfo`. The stream or null if it is not present.

**!Array.<number> o3djs.primitives.VertexInfo.getTriangle ( number *triangleIndex* )**  
Gets the vertex indices of the triangle at the given triangle index.

### Parameters:

*triangleIndex* The index of the triangle.

### Returns:

**!Array.<number>.**An array of three triangle indices.

o3djs.primitives.VertexInfo.numTriangles ( )

Returns the number of triangles represented by the VertexInfo.

## Returns:

number. The number of triangles represented by `VertexInfo`.

Remove a stream from the VertexInfo. Does nothing if a matching stream does not exist.

## Parameters:

*semantic* The semantic of the stream.

*opt\_semanticIndex* The semantic index of the stream. Defaults to zero.

`o3djs.primitives.VertexInfo.reorient ( !o3djs.math.Matrix4 matrix )`

Reorients the vertices, positions and normals, of this vertexInfo by the given matrix. In other words, it multiplies each vertex by the given matrix and each normal by the inverse-transpose of the given matrix.

### Parameters:

*matrix* Matrix by which to multiply.

Sets the vertex indices of the triangle at the given triangle index.

### Parameters:

*triangleIndex* The index of the triangle.

*index1* The index of the first vertex of the triangle.

*index2* The index of the second vertex of the triangle.

*index3* The index of the third vertex of the triangle.

```
o3djs.primitives.VertexInfo.validate( )
```

Validates that all the streams contain the same number of elements, that all the indices are within range and that a position stream is present.

# **o3djs.primitives.VertexStreamInfo Class Reference**

## List of all members.

## Detailed Description

Used to store the elements of a stream.

## Source

## o3djs/primitives.js

## Constructor

- [o3djs.primitives.VertexStreamInfo](#)(numComponents, semantic, opt\_semanticIndex)

## Public Member Functions

- [addElement](#)(value1, opt\_value2, opt\_value3, opt\_value4)
- [addElementVector](#)(vector)
- !Array.<number> [getElementVector](#)(index)
- number [numElements](#)()
- [setElement](#)(index, value1, opt\_value2, opt\_value3, opt\_value4)
- [setElementVector](#)(index, vector)

## Public Properties

- !Array.<number> [elements](#)
  - number [numComponents](#)
  - ![o3d.Stream.Semantic semantic](#)
  - number [semanticIndex](#)
- 

## Constructor

```
o3djs.primitives.VertexStreamInfo ( number numComponents
                                     !o3d.Stream.Semantic semantic
                                     number opt_semanticIndex )
```

Used to store the elements of a stream.

### Parameters:

*numComponents* The number of numerical components per element.  
*semantic* The semantic of the stream.  
*opt\_semanticIndex* The semantic index of the stream. Defaults to zero.

---

## Member Function Documentation

```
o3djs.primitives.VertexStreamInfo.addElement ( number value1
                                                number opt_value2
                                                number opt_value3
                                                number opt_value4 )
```

Adds an element to this VertexStreamInfo. The number of values passed must match the number of components for this VertexStreamInfo.

### Parameters:

*value1* First value.  
*opt\_value2* Second value.

*opt\_value3* Third value.  
*opt\_value4* Fourth value.

`o3djs.primitives.VertexStreamInfo.addElementVector ( !Array.<number> vector )`

Adds an element to this VertexStreamInfo. The number of values in the vector must match the number of components for this VertexStreamInfo.

**Parameters:**

*vector* Array of values for element.

`!Array.<number> o3djs.primitives.VertexStreamInfo.getElementVector ( number index )`

Sets an element on this VertexStreamInfo. The number of values in the vector will match the number of components for this VertexStreamInfo.

**Parameters:**

*index* Index of element to set.

**Returns:**

`!Array.<number>.Array of values for element.`

`o3djs.primitives.VertexStreamInfo.numElements ( )`

Get the number of elements in the stream.

**Returns:**

`number`.The number of elements in the stream.

`o3djs.primitives.VertexStreamInfo.setElement ( number index`

`number value1`

`number opt_value2`

`number opt_value3`

`number opt_value4 )`

Sets an element on this VertexStreamInfo. The number of values passed must match the number of components for this VertexStreamInfo.

**Parameters:**

*index* Index of element to set.

*value1* First value.

*opt\_value2* Second value.

*opt\_value3* Third value.

*opt\_value4* Fourth value.

`o3djs.primitives.VertexStreamInfo.setElementVector ( number index`

`!Array.<number> vector )`

Sets an element on this VertexStreamInfo. The number of values in the vector must match the number of components for this VertexStreamInfo.

**Parameters:**

*index* Index of element to set.

*vector* Array of values for element.

---

# Member Property Documentation

`!Array.<number> o3djs.primitives.VertexStreamInfo.elements`  
The elements of the stream.

`number o3djs.primitives.VertexStreamInfo.numComponents`  
The number of numerical components per element.

`!o3d.Stream.Semantic o3djs.primitives.VertexStreamInfo.semantic`  
The semantic of the stream.

`number o3djs.primitives.VertexStreamInfo.semanticIndex`  
The semantic index of the stream.

# o3djs.rendergraph.ViewInfo Class Reference

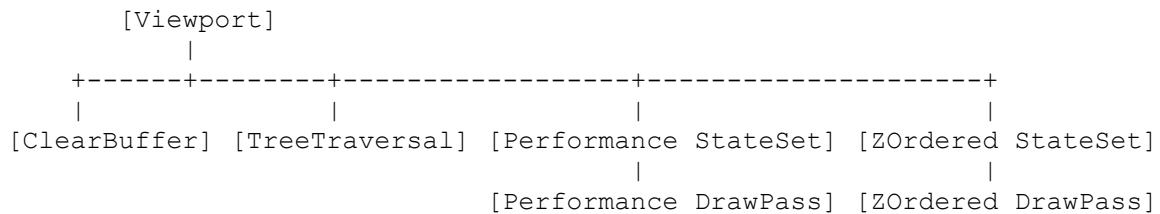
[List of all members.](#)

---

## Detailed Description

A ViewInfo object creates the standard o3d objects needed for a single 3d view. Those include a ClearBuffer followed by a TreeTraveral followed by 2 DrawPasses all of which are children of a Viewport. On top of those a DrawContext and optionally 2 DrawLists although you can pass in your own DrawLists if there is a reason to reuse the same DrawLists such was with mulitple views of the same scene.

The render graph created is something like:



## Source

[o3djs/rendergraph.js](#)

## Constructor

- `o3djs.rendergraph.ViewInfo(pack, treeRoot, opt_parent, opt_clearColor, opt_priority, opt_viewport, opt_performanceDrawList, opt_zOrderedDrawList)`

## Public Member Functions

- [destroy\(opt\\_destroyDrawContext, opt\\_destroyDrawList\)](#)

## Public Properties

- [!o3d.ClearBuffer clearBuffer](#)
  - [!o3d.DrawContext drawContext](#)
  - [!o3d.Pack pack](#)
  - [!o3d.DrawLine performanceDrawList](#)
  - [!o3d.DrawPass performanceDrawPass](#)
  - [!o3d.State performanceState](#)
  - [!o3d.StateSet performanceStateSet](#)
  - number [priority](#)
  - ([!o3d.RenderNode](#)|[undefined](#)) [renderGraphRoot](#)
  - [!o3d.RenderGraph root](#)
  - [!o3d.Transform treeRoot](#)
  - [!o3d.TreeTraversal treeTraversal](#)
  - [!o3d.Viewport viewport](#)
  - [!o3d.DrawLine zOrderedDrawList](#)
  - [!o3d.DrawPass zOrderedDrawPass](#)
  - [!o3d.State zOrderedState](#)
  - [!o3d.StateSet zOrderedStateSet](#)
- 

## Constructor

```
o3djs.rendergraph.ViewInfo ( !o3d.Pack      pack
                            !o3d.Transform   treeRoot
                            !o3d.RenderNode  opt_parent
                            !o3djs.math.Vector4 opt_clearColor
                            number          opt_priority
                            !o3djs.math.Vector4 opt_viewport
                            !o3d.DrawLine    opt_performanceDrawList
                            !o3d.DrawLine    opt_zOrderedDrawList )
```

A ViewInfo object creates the standard o3d objects needed for a single 3d view. Those include a ClearBuffer followed by a TreeTraveral followed by 2 DrawPasses all of which are children of a Viewport. On top of those a DrawContext and optionally 2 DrawLists although you can pass in your own DrawLists if there is a reason to reuse the same DrawLists such was with mulitple views of the same scene.

The render graph created is something like:



```
[ClearBuffer] [TreeTraversal] [Performance StateSet] [ZOrdered StateSet]
          |           |
          [Performance DrawPass] [ZOrdered DrawPass]
```

#### Parameters:

<i>pack</i>	Pack to manage created objects.
<i>treeRoot</i>	root Transform of tree to render.
<i>opt_parent</i>	RenderNode to build this view under.
<i>opt_clearColor</i>	color to clear view.
<i>opt_priority</i>	Optional base priority for created objects.
<i>opt_viewport</i>	viewport settings for view.
<i>opt_performanceDrawList</i>	DrawList to use for performanceDrawPass.
<i>opt_zOrderedDrawList</i>	DrawList to use for zOrderedDrawPass.

## Member Function Documentation

`o3djs.rendergraph.ViewInfo.destroy ( Boolean opt_destroyDrawContext  
                                  Boolean opt_destroyDrawList )`

Destroys the various objects created for the view.

#### Parameters:

<i>opt_destroyDrawContext</i>	True if you want view's DrawContext destroyed. Default = true.
<i>opt_destroyDrawList</i>	True if you want view's DrawLists destroyed. Default = true.

## Member Property Documentation

[!o3d.ClearBuffer](#) `o3djs.rendergraph.ViewInfo.clearBuffer`

The ClearBuffer RenderNode created for this ViewInfo.

[!o3d.DrawContext](#) `o3djs.rendergraph.ViewInfo.drawContext`

The DrawContext used by this ViewInfo.

[!o3d.Pack](#) `o3djs.rendergraph.ViewInfo.pack`

Pack that manages the objects created for this ViewInfo.

[!o3d.DrawLine](#) `o3djs.rendergraph.ViewInfo.performanceDrawList`

The DrawList used for the performance draw pass. Generally for opaque materials.

[!o3d.DrawPass](#) `o3djs.rendergraph.ViewInfo.performanceDrawPass`

The DrawPass used with the performance DrawList created by this ViewInfo.

[!o3d.State](#) `o3djs.rendergraph.ViewInfo.performanceState`

The State object used by the performanceStateSet object in this ViewInfo. By default, no states are set here.

[!o3d.StateSet](#) `o3djs.rendergraph.ViewInfo.performanceStateSet`

The StateSet RenderNode above the performance DrawPass in this ViewInfo

number o3djs.rendergraph.ViewInfo.priority

The highest priority used for objects under the Viewport RenderNode created by this ViewInfo.

(!o3d.RenderNode|undefined) o3djs.rendergraph.ViewInfo.renderGraphRoot

The RenderNode this ViewInfo render graph subtree is parented under.

[!o3d.RenderGraph](#) o3djs.rendergraph.ViewInfo.root

The root of the subtree of the render graph this ViewInfo is managing. If you want to set the priority of a ViewInfo's rendergraph subtree use

```
viewInfo.root.priority = desiredPriority;
```

[!o3d.Transform](#) o3djs.rendergraph.ViewInfo.treeRoot

The root node of the transform graph this ViewInfo renders.

[!o3d.TreeTraversal](#) o3djs.rendergraph.ViewInfo.treeTraversal

The TreeTraversal used by this ViewInfo.

[!o3d.Viewport](#) o3djs.rendergraph.ViewInfo.viewport

The Viewport RenderNode created for this ViewInfo.

[!o3d.DrawList](#) o3djs.rendergraph.ViewInfo.zOrderedDrawList

The DrawList used for the zOrdered draw pass. Generally for transparent materials.

[!o3d.DrawPass](#) o3djs.rendergraph.ViewInfo.zOrderedDrawPass

The DrawPass used with the zOrdered DrawList created by this ViewInfo.

[!o3d.State](#) o3djs.rendergraph.ViewInfo.zOrderedState

The State object used by the zOrderedStateSet object in this ViewInfo. By default AlphaBlendEnable is set to true, SourceBlendFunction is set to State.BLENDFUNC\_SOURCE\_ALPHA and DestinationBlendFunction is set to State.BLENDFUNC\_INVERSE\_SOURCE\_ALPHA

[!o3d.StateSet](#) o3djs.rendergraph.ViewInfo.zOrderedStateSet

The StateSet RenderNode above the ZOrdered DrawPass in this ViewInfo

## o3djs.serialization.Deserializer Class Reference

[List of all members.](#)

---

### Detailed Description

A Deserializer incrementally deserializes a transform graph.

### Source

[o3djs/serialization.js](#)

## Constructor

- [o3djs.serialization.Deserializer\(pack, json\)](#)

## Public Member Functions

- [addObject\(id, object\)](#)
- \* [deserializeValue\(valueJson\)](#)
- (Object|undefined) [getObjectById\(id\)](#)
- boolean [run\(opt\\_amountOfWork\)](#)
- [runBackground\(client, pack, time, callback\)](#)

## Public Properties

- [o3djs.io.ArchiveInfo archiveInfo](#)
  - !Object [createCallbacks](#)
  - !Object [initCallbacks](#)
  - !Object [json](#)
  - ![o3d.Pack pack](#)
- 

## Constructor

`o3djs.serialization.Deserializer ( !o3d.Pack pack  
                                  !Object json )`

A Deserializer incrementally deserializes a transform graph.

### Parameters:

*pack* The pack to deserialize into.  
*json* An object tree conforming to the JSON rules.

---

## Member Function Documentation

`o3djs.serialization.Deserializer.addObject ( number id  
                                  !Object object )`

When a creation or init callback creates an object that the Deserializer is not aware of, it can associate it with an id using this function, so that references to the object can be resolved.

### Parameters:

*id* The is of the object.  
*object* The object to register.

\* `o3djs.serialization.Deserializer.deserializeValue ( * valueJson )`

Deserialize a value. Identifies reference values and converts their object id into an object reference. Otherwise returns the value unchanged.

### Parameters:

*valueJson* The JSON representation of the value.

**Returns:**

\*.The JavaScript representation of the value.

(Object|undefined) o3djs.serialization.Deserializer.getObjectById ( number *id* )

Get the object with the given id.

**Parameters:**

*id* The id to lookup.

**Returns:**

(Object|undefined).The object with the given id.

boolean o3djs.serialization.Deserializer.run ( number *opt\_amountOfWork* )

Perform a certain number of iterations of the deserializer. Keep calling this function until it returns false.

**Parameters:**

*opt\_amountOfWork* The number of loop iterations to run. If not specified, runs the deserialization to completion.

**Returns:**

boolean.Whether work remains to be done.

o3djs.serialization.Deserializer.runBackground ( ![o3d.Client](#) *client*

![o3d.Pack](#) *pack*

*number* *time*

!function([o3d.Pack](#), \*): void *callback* )

Deserializes (loads) a transform graph in the background. Invokes a callback function on completion passing the pack and the thrown exception on failure or the pack and a null exception on success.

**Parameters:**

*client* An O3D client object.

*pack* The pack to create the deserialized objects in.

*time* The amount of the time (in seconds) the deserializer should aim to complete in.

*callback* The function that is called on completion. The second parameter is null on success or the thrown exception on failure.

---

## Member Property Documentation

[o3djs.io.ArchiveInfo](#) o3djs.serialization.Deserializer.archiveInfo

The archive from which assets referenced from JSON are retrieved.

!Object o3djs.serialization.Deserializer.createCallbacks

A map from classname to a function that will create instances of objects. Add entries to support additional classes.

!Object o3djs.serialization.Deserializer.initCallbacks

A map from classname to a function that will initialize instances of the given class from JSON data.

Add entries to support additional classes.

`!Object o3djs.serialization.Deserializer.json`

An object tree conforming to the JSON rules.

`!o3d.Pack o3djs.serialization.Deserializer.pack`

The pack to deserialize into.

# o3djs.simple.SimpleInfo Class Reference

[List of all members.](#)

---

## Detailed Description

A SimpleInfo contains information for the simple library.

## Source

[o3djs/simple.js](#)

## Constructor

- [o3djs.simple.SimpleInfo\(clientObject\)](#)

## Public Member Functions

- [!o3djs.simple.SimpleShape createBox\(width, height, depth\)](#)
  - [!o3djs.simple.SimpleShape createCube\(size\)](#)
  - [!o3d.Material createMaterialFromEffect\(effect\)](#)
  - [!o3d.Material createNonTexturedMaterial\(type\)](#)
  - [!o3djs.simple.SimpleShape createSimpleShape\(shape, material\)](#)
  - [!o3djs.simple.SimpleShape createSphere\(radius, smoothness\)](#)
  - [!o3d.Material createTexturedMaterial\(type\)](#)
  - [!o3djs.simple.Scene loadScene\(url\)](#)
  - [setCameraPosition\(x, y, z\)](#)
  - [setCameraTarget\(x, y, z\)](#)
  - [setCameraUp\(x, y, z\)](#)
  - [setFieldOfView\(fieldOfView\)](#)
  - [setLightColor\(r, g, b, a\)](#)
  - [setLightPosition\(x, y, z\)](#)
  - [setZClip\(zNear, zFar\)](#)
  - [viewAll\(\)](#)
-

# Constructor

`o3djs.simple.SimpleInfo ( !Element clientObject )`  
A SimpleInfo contains information for the simple library.

**Parameters:**

*clientObject* O3D.Plugin Object.

---

## Member Function Documentation

[!o3djs.simple.SimpleShape](#) `o3djs.simple.SimpleInfo.createBox ( number width  
                                  number height  
                                  number depth )`

Creates a box and adds it to the root of this SimpleInfo's transform graph.

**Parameters:**

*width* Width of the box.  
*height* Height of the box.  
*depth* Depth of the box.

**Returns:**

[!o3djs.simple.SimpleShape](#).A Javascript object to manage the shape.

[!o3djs.simple.SimpleShape](#) `o3djs.simple.SimpleInfo.createCube ( number size )`

Creates a cube and adds it to the root of this SimpleInfo's transform graph.

**Parameters:**

*size* Width, height and depth of the cube.

**Returns:**

[!o3djs.simple.SimpleShape](#).A Javascript object to manage the shape.

[!o3d.Material](#) `o3djs.simple.SimpleInfo.createMaterialFromEffect ( !o3d.Effect effect )`

Create meterial from effect.

**Parameters:**

*effect* Effect to use for material.

**Returns:**

[!o3d.Material](#).The created material.

[!o3d.Material](#) `o3djs.simple.SimpleInfo.createNonTexturedMaterial ( string type )`

Create a new non-textured material.

**Parameters:**

*type* Type of material 'phong', 'lambert', 'constant'.

**Returns:**

[!o3d.Material](#).The created material.

[!o3djs.simple.SimpleShape](#) `o3djs.simple.SimpleInfo.createSimpleShape ( !o3d.Shape    shape`

[!o3d.Material material](#) )

Creates a SimpleShape. A SimpleShape manages a transform with 1 shape that holds 1 primitive and 1 unique material.

**Parameters:**

*shape* that holds 1 primitive and 1 unique material.  
*material* assigned to shape.

**Returns:**

[!o3djs.simple.SimpleShape](#).the created SimpleShape.

[!o3djs.simple.SimpleShape](#) o3djs.simple.SimpleInfo.createSphere ( number *radius*  
number *smoothness* )

Creates a sphere and adds it to the root of this SimpleInfo's transform graph.

**Parameters:**

*radius* radius of sphere.  
*smoothness* determines the number of subdivisions.

**Returns:**

[!o3djs.simple.SimpleShape](#).A Javascript object to manage the shape.

[!o3d.Material](#) o3djs.simple.SimpleInfo.createTexturedMaterial ( string *type* )

**Parameters:**

*type* Type of material 'phong', 'lambert', 'constant'.

**Returns:**

[!o3d.Material](#).The created material.

[!o3djs.simple.Scene](#) o3djs.simple.SimpleInfo.loadScene ( string *url* )

Loads a scene from a URL. TODO: Implement

**Parameters:**

*url* Url of scene to load.

**Returns:**

[!o3djs.simple.Scene](#).A Javascript object to manage the scene.

o3djs.simple.SimpleInfo.setCameraPosition ( number *x*  
number *y*  
number *z* )

Sets the camera position

**Parameters:**

*x* x position.  
*y* y position.  
*z* z position.

o3djs.simple.SimpleInfo.setCameraTarget ( number *x*  
number *y*  
number *z* )

Sets the camera target

**Parameters:**

*x* x position.  
*y* y position.  
*z* z position.

`o3djs.simple.SimpleInfo.setCameraUp ( number x  
  number y  
  number z )`

Sets the camera up

**Parameters:**

*x* x position.  
*y* y position.  
*z* z position.

`o3djs.simple.SimpleInfo.setFieldOfView ( number fieldOfView )`

Sets the field of view

**Parameters:**

*fieldOfView* in Radians.

For degrees use `setFieldOfView(o3djs.math.degToRad(degrees))`.

`o3djs.simple.SimpleInfo.setLightColor ( number r  
  number g  
  number b  
  number a )`

Sets the light color

**Parameters:**

*r* red (0-1).  
*g* green (0-1).  
*b* blue (0-1).  
*a* alpha (0-1).

`o3djs.simple.SimpleInfo.setLightPosition ( number x  
  number y  
  number z )`

Sets the light position

**Parameters:**

*x* x position.  
*y* y position.  
*z* z position.

`o3djs.simple.SimpleInfo.setZClip ( number zNear  
  number zFar )`

Sets the z clip range.

**Parameters:**

*zNear* near z value.  
*zFar* far z value.

`o3djs.simple.SimpleInfo.viewAll ( )`

Moves the camera so everything in the current scene is visible.

# o3djs.simple.SimpleShape Class Reference

[List of all members.](#)

---

## Detailed Description

A SimpleShape manages a transform with 1 shape that holds 1 primitive and 1 unique material.

## Source

[o3djs/simple.js](#)

## Constructor

- [o3djs.simple.SimpleShape\(simpleInfo, transform, material\)](#)

## Public Member Functions

- [o3d.Texture getTexture\(\)](#)
- [loadTexture\(url\)](#)
- [setDiffuseColor\(r, g, b, a\)](#)
- [setMaterial\(material\)](#)

## Public Properties

- [!o3d.Material material](#)
  - [!o3djs.simple.SimpleInfo simpleInfo](#)
  - [!o3d.Transform transform](#)
- 

## Constructor

o3djs.simple.SimpleShape ( [!o3djs.simple.SimpleInfo simpleInfo](#)  
[!o3d.Transform transform](#)  
[!o3d.Material material](#) )

A SimpleShape manages a transform with 1 shape that holds 1 primitive and 1 unique material.

### Parameters:

*simpleInfo* SimpleInfo to manage this shape.

*transform* Transform with 1 shape that holds 1 primitive and 1 unique material.

*material* assigned to shape.

---

# Member Function Documentation

`o3djs.simple.SimpleShape.getTexture ( )`

Gets the texture on this shape.

## Returns:

[o3d.Texture](#).The texture on this shape. May be null.

`o3djs.simple.SimpleShape.loadTexture ( string url )`

Loads a texture onto the given shape. It will replace the material if it needs to with one that supports a texture. Note that the texture is loaded asynchronously and so the result of this call may appear several seconds after it is called depending on how long it takes to download the texture.

## Parameters:

*url* Url of texture.

`o3djs.simple.SimpleShape.setDiffuseColor ( number r  
number g  
number b  
number a )`

Sets the diffuse color of this shape.

## Parameters:

*r* Red (0-1).

*g* Green (0-1).

*b* Blue (0-1).

*a* Alpha (0-1).

`o3djs.simple.SimpleShape.setMaterial ( !o3d.Material material )`

Sets the material for this SimpleShape, deleting any old one.

## Parameters:

*material* new material.

---

# Member Property Documentation

[!o3d.Material](#) `o3djs.simple.SimpleShape.material`

The material for this SimpleShape.

[!o3djs.simple.SimpleInfo](#) `o3djs.simple.SimpleShape.simpleInfo`

The SimpleInfo managing this SimpleShape.

[!o3d.Transform](#) `o3djs.simple.SimpleShape.transform`

The transform for this SimpleShape.

# O3D Release Notes

## Release 0.1.38 (29 June 2009)

### Added

- [`Texture2D.setRect`](#)(level, destinationX, destinationY, sourceWidth, arrayOfNumbers): Lets you set a rectangular area of a texture from JavaScript.
- [`o3djs.math.modClamp`](#): Clamps something using modulo; handles negative values.  
`o3djs.math.modClamp(7, 10) = 7`  
`o3djs.math.modClamp(-3, 10) = 7`
- [`o3djs.math.lerpCircular`](#): Lerps in a circle—for example, if the range is 0 to 100 and you are going from 98 to 2, it will go 98, 99, 0, 1, 2 (around the circle).
- [`o3djs.math.lerpRadian`](#): Calls `lerpCircular()` with `Math.PI * 2` for the range.
- [Requirements \(`o3d\_features`\)](#)
  - LargeGeometry
  - FloatingPointTextures

You can also specify `NotAntiAliased` for `o3d_features`, which forces anti-aliasing off.

- Support for Intel 950 and other lower-end GPUs.
- Software renderer support.
- `oncontextmenu` support.
- [UByteNField](#) support.

Also now, by default, vertex color fields in a COLLADA file are converted to `UByteNFields`, which saves space in our sample converter.

- [Full-screen mode](#).
- The sample converter and sample shader builder ([`o3djs/effect.js`](#)) now support a `specularFactor` setting from 3ds Max.
- A OneShot particle emitter (good for puffs, explosions, fireworks, etc.): See [`samples/particles.html`](#).
- [`o3djs.effect.createEffectFromFile\(\)`](#): Loads an fx string *and* creates the effect.
- [`o3djs.material.createMaterialFromFile\(\)`](#): Loads an fx string, creates an effect, and creates a material using that effect.
- [`o3djs.material.bindParamsOnMaterial\(\)`](#): Given a JavaScript object of O3D params, connects them to the corresponding params on a material.
- [`o3djs.material.bindParams\(\)`](#): Given a JavaScript object of O3D params, connects them to the the corresponding params on all materials in a pack.
- [`o3djs.material.createParams\(\)`](#): Creates a params from a JavaScript object, where the name of each property of the object is the name of the param to create and the property's value is the type of param to create.
- [`o3djs.material.createStandardParams\(\)`](#): Creates the params the shader builder - effect.js- expects for globals, (for example, `lightColor`, `lightWorldPos`).
- [`o3djs.material.createAndBindStandardParams\(\)`](#): Creates the standard params and binds them to all the materials in a pack. Effectively, this calls

`o3djs.material.createStandardParams()` and then calls  
`o3djs.material.bindParams()`.

Added aliases for most matrix related math functions. For example:

- `o3djs.math.matrix4.mul`
- `o3djs.math.matrix4.copy`
- `o3djs.math.matrix4.det`
- `o3djs.math.matrix4.inverse`

## Changed

- Moved to chromium.org and Subversion.
- `o3djs.util.makeClients`: The second argument is now the application's requirements, as in

```
o3djs.util.makeClients(initStep2, 'LargeGeometry,FloatingPointTextures')
```

See [Performance Tuning](#).

## Removed

- GR16F and GR32F texture formats. There is no easy way to make these work across GL/D3D, so we removed them. Sorry if you were using them.

## Fixed

- Render target and textures get cleared so they never have garbage.
- Fixed `o3djs.picking` bug that occurred when tree traversal encounters a transform with no shapes below it.
- Fixed bug in beach demo that occurred when the user attempted to use both mouse and keys at the same time.
- Fixed viewport settings for rendertargets. The viewport is correctly set and restored across render targets.
- Fixed bug on Windows where mouseup event was sometimes not received.
- Fixed `o3djs.loader.loadScene()` to have `opt_options`.
- Shader builder now builds a shader that does not need normals for constant materials.
- Fixed crash bug with bad gzipped tar files.

# Release 0.1.35.2 (6 May 2009)

## Bug Fixes

- Fixed touchpad scrolling for Mac Safari 4.
- Fixed Mac bug that would crash the plug-in when a tab is dragged out of the browser in Safari.
- Mouse wheel events now return client-area-relative mouse coordinates.
- Fixed a problem in the Mac where the O3D window would hover over other HTML elements in certain instances.
- Fixed a bug that prevented the O3D plug-in from unloading in Internet Explorer.
- Assorted crash bug fixes.
- Installer: Display an actual visible error message when O3D fails to install for lack of DirectX 9.0c.
- Installer: Mac installer now explicitly checks for OSX version 10.5 (Leopard) and refuses to install on earlier versions.

## Other Plug-in Changes

- Added missing sampler addressModeW setting.
- Upgraded the embedded V8 to the latest version from Chromium. This should provide even better performance and will probably fix some crash bugs too.

## Samples/Utilities Changes

- Samples: Improved performance of Prince IO. The sample was previously hard-coded to run at 20fps.
- Samples: Several improvements to the Beach demo including:
  - The ocean and the island now display the correct textures and have shaders that mix between two textures based on vertex alpha.
  - There is an animated camera.
  - There is a proxy island that loads quickly before the detailed island loads.
  - 3D-oriented particles are used at the bottom of the waterfalls.
- Utilities: o3djs/shape.js is now split into two files, shape.js and element.js.
- Utilities: o3djs/quaternions.js: Changed quaternion representation from [w, x, y, z] to [x, y, z, w]. The scalar part, w, is now the fourth component.
- Utilities: Added 3D particles to particle.js and the sample particles.html.
- Utilities: Fixed bugs in effects.js related to making shaders for normal mapped assets.
- Utilities: o3djs/debug.js: Added createSphere and createCube utilities to let you create those in arbitrary space.
- Utilities: o3djs/math.js: Added method to create a 4-by-4 perspective transformation given a set of clipping planes.
- Utilities: o3djs/math.js: Added aliases for inverse, multiply, and the determinant of 4-by-4 matrices.

## Tools Changes

- Added o3dVerifier, a tool that test-compiles the contents of a shader (.fx) file and converts it into the shader format accepted by O3D.
- Removed performance timer output from the O3D Sample Converter (o3dConverter).
- Fixed o3dConverter to return all the mesh streams found in the COLLADA file.
- Fixed od3Converter to better handle local paths to shader files in the archive.
- Changed the default extension to asset files generated by the o3dConverter from .tgz to .o3dtgz to avoid issues on certain server configurations (blog post with details to follow).

# Release 0.1.34 (21 April 2009)

## Known Issues

- Fonts are different on Mac and Windows. There is a handful of fonts that will work across platforms (look at [samples/canvas-fonts.html](#)), but even those aren't guaranteed to be pixel equivalent.
- Text support for the Canvas object isn't available on Linux.
- Slow performance on some graphics cards, especially with Chrome.
- Samples Issues
  - Prince IO may sometimes start in an inconsistent state. Reloading and turning introduction pages slowly fixes this issue.
  - Prince IO may cause Firefox on Windows to become unresponsive.

- The Home Designer sample does not work properly in Google Chrome or Internet Explorer on Windows.
- The Julia and Juggler examples may display intermittent vertical line artifacts on the Mac.
- Some samples may be slow using certain combinations of graphics hardware and Google Chrome.
- Display Issues
  - Navigating to pages with O3D content via the browser history in Firefox for Mac may not initialize the plug-in properly. As a workaround, refresh the page.
  - The O3D content area may not move properly when the user is scrolling in Firefox for the Mac.
  - The O3D content area flickers when the user is scrolling in Internet Explorer or Firefox on Windows Vista.
  - The O3D content area will sometimes get stuck above other content in Firefox on Windows Vista.
  - The O3D content area will flicker when the user is scrolling the page or changing focus on the O3D content area in Firefox on the Mac.
  - O3D doesn't render correctly after a screen resolution change (Windows).
  - O3D rendering can slow the entire system down when running on multiple displays.
  - The O3D content area flashes when receiving focus on the Mac.
- Crashes and Hanging
  - Chrome 2.0+ (on the Chrome Beta and Developer channels) sometimes hangs when loading O3D content.
  - The O3D plug-in may become unresponsive when running the Animated Scene demo on Google Chrome.
  - IE 8 on Vista may crash running samples on ATI Radeon X1550 and Radeon X1650 cards.
  - Firebug breakpoints in O3D-sensitive code may freeze Firefox on the Mac.
  - Safari 4 for Mac crashes when the user pulls off a tab that is displaying an O3D content area.
  - Resizing a browser window that contains O3D content and spans two monitors may crash the browser on Windows.
  - Debugging an O3D application on Firebug in Firefox on the Mac tends to hang the browser.
- Asset loading
  - The plug-in will refuse to load JSON .tgz archives from servers that automatically decompress the zip file.
- Internet Explorer for Windows Vista Troubleshooting

If you have already installed the O3D plug-in but are still seeing a message that the page requires the O3D plug-in to be installed as you try to load O3D content in Internet Explorer on Windows Vista, we recommend you try the following steps:

- Turn User Access Control on from the Windows Vista control panel, restart your machine, and load the O3D content again in Internet Explorer. Due to new security features in Windows Vista, some ActiveX controls won't run with UAC off.
- If UAC is already on, test the O3D content in another web browser, such as Mozilla Firefox or Google Chrome. If the O3D content functions normally in these browsers, this may indicate a problem with your install. Try reinstalling the plug-in from an Administrator account.

# Glossary

## draw context

A draw context contains a view matrix and a projection matrix. Together, these matrices define the parameters for a virtual camera that views the 3D scene.

## effect

An effect contains two shaders: a vertex shader and a pixel shader (also referred to as a "fragment shader"). The *vertex shader* processes the vertex positions, which are defined in local object space, and converts them to screen space. It also performs lighting and other per-vertex calculations. The *pixel shader* generates the color of the individual pixels based on interpolated vertex data coming from the vertex shader.

## fragment shader

See **pixel shader**.

## material

A material defines the surface parameters of a 3D object, such as its diffuse, ambient, and specular colors. The material's parameters are used by the effect to determine the rendered appearance of the object's surface.

## pack

A pack holds references to other data. Its only function is to manage the lifetime of data. For example, if you create a new pack, then load a COLLADA file into that pack, all the textures, states, buffers, shapes, and transforms are associated with that pack.

## pixel shader

The pixel shader uses the output of the rasterizer and calculates the color of each pixel in the O3D window. Also called **fragment shader**.

## primitive

A shape is composed of one or more primitives. A given primitive can have only one material assigned to it.

## render graph

A tree of nodes and associated parameters that describe how to display the transform graph's draw elements.

## rendering

Rendering is the process of traversing the render graph, executing the operations of each node in the render graph to draw objects on the screen.

## sampler

A sampler is an object that describes how to apply a texture bitmap to a shape's surfaces.

## shape

Shapes contain references to the data to render a shape: vertex buffers, index buffers, effects, state, as well as the material parameters an effect needs to be rendered.

## texture

A texture is a 2D array of pixel data.

## transform

A transform contains a local matrix parameter. When transforms are arranged in a parent/child relationship, their local matrices will be concatenated automatically to provide a world matrix.

## transform graph

A transform graph is a hierarchy of transforms, with associated shapes, materials, and effects.

## vertex shader

The vertex shader calculates the position of each vertex, in homogeneous clip space, as well as the value of each per-vertex attribute.