OpenSceneGraph Reference Manual

v2.2

Editors

Bob Kuehne *Paul Martz*

OpenSceneGraph Reference Manual v2.2

Blue Newt Software, LLC
445 Second Street
Ann Arbor, MI 48103, USA
http://www.blue-newt.com

Skew Matrix Software, LLC
284 W. Elm St.
Louisville, CO 80027, USA
http://www.skew-matrix.com

First Printing, December 2007

Contents

Preface

This book is a reference manual for OpenSceneGraph (OSG) - the cross-platform open source scene graph application programmer interface (API). Specifically, this book documents the core OSG libraries in OSG v2.2. OSG plays a key role in the 3D application software stack. It's the middleware above the lower-level OpenGL hardware abstraction layer (HAL), providing extensive higher-level rendering, I/O, and spatial organization functionality to the 3D application.

For many years, OSG developers could generate their own reference manual using Doxygen directly from the OSG source code. Such documentation, while useful, suffered from formatting issues and has never been published.

This reference manual uses customized Doxygen files, a variety of scripts, enhanced OSG documentation, and improved indexing to address the formatting issues and organize the OSG reference material into a format suitable for printing. However, much of text content comes directly from comments within the source code, written by Robert Osfield and other OSG contributors. The editors have been gradually updating text content and contributing that back to the OSG project. However, all text presented here is directly excerpted from stock OSG, edited only for purposes of presentation and layout. The authors have, however, contributed extensive additional documentation on OSG and various features in the early chapters of each reference manual produced.

The *OpenSceneGraph Reference Manual v2.2* is the fourth book in the OpenSceneGraph Programming Series, a series of books to document OSG. The *OpenSceneGraph Reference Manual v2.2* has the following goals.

- Document OSG's most stable classes and member functions - those found in the three core libraries in the OSG v2.2 release.
- Increase developer productivity by providing OSG developers with immediate access to a softbound reference manual.
- Solicit feedback and suggestions from readers on the content of future versions of this and other OSG reference manuals.

In the spirit of open source, you can still generate your own reference material using Doxygen and the OSG source code. However, you can contribute to the OSG community by purchasing this book. To place an online order, please visit the book website:

`http://www.osgbooks.com/shop.html`

Proceeds from the sale of this manual fund ongoing documentation revisions to ensure that the manual is always up-to-date.

Your feedback on the book is essential in ensuring this documentation remains current and useful. Please post your comments to the `osg-users` email list.

For information about new revisions to the book, visit the *OpenSceneGraph Reference Manual v2.2* Web site:

`http://www.osgbooks.com`

This URL contains the most up-to-date information on obtaining the latest revision and information on related publications.

Audience

This book is not an introduction to OSG. This book is intended for software developers who already have some experience with OSG and need a convenient reference manual to look up class and method names, parameters, enumerants, derivations, and other OSG syntax minutiae.

OSG is a C++ API, and this reference manual assumes a strong familiarity with C++. In particular, you should be familiar with C++ features, such as public and private access, virtual functions, memory allocation, class derivation, operator overloading, and constructors and destructors. OSG makes extensive use of the standard template library (STL), so you should be familiar with STL constructs, especially list, vector, and map. Some familiarity with design patterns as implemented in C++ is useful, but is not required.

OSG is based on OpenGL, and there is a strong mapping between OSG and OpenGL state. Familiarity with OpenGL helps with grasping many of the OSG state-related classes described in this book.

As a requirement for developing OSG applications, you should be familiar with 3D graphics and linear algebra.

Recommended Reading

Many in the OSG community recommend the following material for anyone new to OSG.

- *OpenGL® Programming Guide, Sixth Edition*, by OpenGL ARB, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis (Addison-Wesley)
- *Geometric Tools for Computer Graphics*, by Philip Schneider and David H. Eberly (Morgan Kaufmann).
- *Real-Time Rendering, Second Edition*, by Tomas Akenine-Moller and Eric Haines (AK Peters).
- *Computer Graphics, Principles and Practice, Second Edition*, by James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes (Addison-Wesley).
- *The C++ Programming Language, Third Edition*, by Bjarne Stroustrup (Addison-Wesley).

Organization of the Book

The *OpenSceneGraph Reference Manual v2.2* is composed of six chapters.

Chapter 1, Overview of OpenSceneGraph, presents a high-level glimpse of OSG's design, organization, and key components. This is essentially review material for the experienced OSG developer. Chapter 2 contains an exhaustive summary of the changes in OSG from v1.2 to v2.2, useful for anyone porting their OSG-based applications. New in the v2.2 edition, Chapter 3 includes a reference of all environment variables that OSG uses to control its default behavior. Together, these chapters provide context for the reference material that follows in Chapters 4, 5, and 6. These chapters document the `osg`, `osgUtil`, and `osgDB` libraries, respectively. These chapters present a full reference for all public classes, methods, and variables, derived directly from the OSG code and its code comments.

About the Editors

Bob Kuehne

Bob is a graphics consultant, entrepreneur, engineer, and head of the graphics consultancy, Blue Newt Software. Blue Newt works with clients around the world to enhance their 3D graphics applications. Blue Newt helps clients developing shader-based applications, from single to multi-pipe, multi-display systems, and from baseline OpenGL to scenegraphs as their needs change. He spends most of his time in modern OpenGL working with these clients to design new applications, develop new features, and to do performance engineering.

Before Blue Newt, Bob worked for nearly eight years at SGI, working in a variety of roles from leading the OpenGL Shader project, creating demos for high-end multi-pipe graphics systems, to helping SGI developers create high-performance, high-quality applications.

Bob has worked in the graphics industry and with graphics systems for more than two decades and has been developing in OpenGL since it first existed. Bob has been a Mac developer since the early 1990s when he was finally able to afford one. He has presented at numerous conferences over his career, among these, SIGGRAPH, Graphic Hardware, SGI Developer Forum Worldwide, and the ex-conference, MacHack. He is the author of *OpenGL Programming on Mac OS X* and teaches OpenGL and scenegraph training courses around the world as part of his work for Blue Newt Software.

When Bob is able to be pulled away from his Mac and graphics work, you'll find him perfecting his espresso pulls, playing volleyball, sailing with his wife. Please don't hesitate to email him here: *rpk@blue-newt.com* or visit his website, *http://www.blue-newt.com*.

Paul Martz

Paul is a 3D graphics software engineer, consultant, published author, and small business owner. He's currently the president of Skew Matrix Software LLC, which specializes in OpenSceneGraph and OpenGL software development, training, and technical writing services.

Paul has been involved in 3D graphics software development since 1987. In the past, he has worked for

both Evans & Sutherland and Hewlett Packard. In the course of his career, he has designed and developed a variety of graphics products, including 3D visualization applications, high-performance graphics hardware device drivers, test suites, demos and tutorials, and desktop window systems.

Paul is the author of *OpenGL Distilled* and the *OpenSceneGraph Quick Start Guide*, and has written book reviews for publications such as the C/C++ Users Journal and Dr Dobb's Journal. Prior to print publication, Paul developed several online tutorials and FAQs to educate the software developer community.

When not writing code and running his business, Paul plays drums and gives music instruction, plays a little poker, and chases total solar eclipses around the world. He can be reached at *pmartz@skew-matrix.com*.

Acknowledgements

Obviously, there would be no *OpenSceneGraph Reference Manual v2.2* if there were not an OpenSceneGraph. Robert Osfield is responsible for developing the vast majority of the OSG code and its comments that were used as source material for this manual. Many others have also contributed to OSG's development. The editors wish to sincerely thank Robert and the entire OSG community, over 1700 strong, for their support and contribution to this important open source standard.

Chapter 1

Overview of OpenSceneGraph

OSG is a set of open source libraries that primarily provide scene management and graphics rendering optimization functionality to applications. It's written in portable ANSI C++ and uses the industry standard OpenGL low-level graphics API. As a result, OSG is cross platform and runs on Windows, Mac OS X, and most UNIX and Linux operating systems. Most of OSG operates independently of the native windowing system. However, OSG includes code to support some windowing system specific functionality, such as input devices, window creation, and PBuffers.

OSG is open source and is available under a modified GNU Lesser General Public License, or Library GPL (LGPL) software license. OSG's open source nature results in improved quality (both for OSG and OSG-based applications) and reduced cost to developers.

1.1 Design and Architecture

OSG is designed up front for portability and scalability. As a result, it is useful on a wide variety of platforms, and renders efficiently on a large number and variety of graphics hardware.

OSG is designed and built with the following concepts and tools:

- ANSI standard C++
- C++ Standard Template Library (STL)
- Design patterns

These tools allow developers using OSG to develop on the platform of their choice and deploy on any platform the customer requires.

1.1.1 Naming Conventions

The following list enumerates the OSG source code naming conventions. These conventions are not always strictly enforced. [1]

- Namespaces - OSG namespace names start with a lower-case letter, but can include upper case for clarity.
  Examples: `osg`, `osgSim`, `osgFX`.
- Classes - OSG class names start with an upper-case letter. If the class name is composed of multiple words, each word starts with an upper-case letter.
  Examples: `MatrixTransform`, `NodeVisitor`, `Optimizer`.
- Class methods - Names of methods within an OSG class start with a lower-case letter. If the method name is composed of multiple words, each additional word starts with an upper-case letter.
  Examples: `addDrawable`, `getNumChildren`, `setAttributeAndModes`.
- Class member variables - Names of member variables within a class use the same convention as method names.
- Templates - OSG template names are lower case with multiple words separated by underscores.
  Examples: `ref_ptr<>`, `graph_array<>`, `observer_ptr<>`.
- Statics - Static variables and functions begin with `s_` and otherwise use the same naming conventions as class member variables and methods.
  Examples: `s_applicationUsage`, `s_ArrayNames()`.
- Globals - Global class instances begin with `g_`.
  Examples: `g_NotifyLevel`, `g_readerWriter_BMP_Proxy`.

1.1.2 Components

The OSG runtime exists as a set of dynamically loaded libraries (or shared objects) and executables. These libraries fall into five conceptual categories:

- The Core OSG libraries provide essential scene graph and rendering functionality, as well as additional functionality that 3D graphics applications typically require.
- NodeKits extend the functionality of core OSG scene graph node classes to provide higher-level node types and special effects.
- OSG plugins are libraries that read and write 2D image and 3D model files.
- The interoperability libraries allow OSG to easily integrate into other environments, including scripting languages such as Python and Lua.
- An extensive collection of applications and examples provide useful functionality and demonstrate correct OSG usage.

This manual provides reference material for the three core OSG libraries, `osg`, `osgUtil`, and `osgDB`.

[1] The OSG plugins, for example, contain many convention violations.

1.2 The osg Library

The osg library is the heart of OpenSceneGraph. It defines the core nodes that make up the scene graph, as well as several additional classes that aid in scene graph management and application development. Chapter 4 covers this library in greater detail.

1.2.1 Scene Graph Classes

Scene graph classes aid in scene graph construction. All scene graph classes in OSG are derived from osg::Node. The list below identifies some of the more commonly used node types in OSG.

- Group-The Group class is the base class for any node that can have children. It is a key class in the spatial organization of scene graphs.
- Geode-The Geode (or Geometry Node) class corresponds to the leaf node in OSG. It has no children, but contains osg::Drawable objects (see below) that contain geometry for rendering.
- LOD-The LOD class displays its children based on their distance to the view point. This is commonly used to create a varying levels of detail for objects in a scene.
- MatrixTransform-The MatrixTransform class contains a matrix that transforms the geometry of its children, allowing scene objects to be rotated, translated, scaled, skewed, projected, etc.
- Switch-The Switch class contains a Boolean mask to enable or disable processing of its children.

1.2.2 Geometry Classes

The Geode class is the OSG leaf node, and it contains geometric data for rendering. The osg library contains the following classes for geometric data storage.

- Drawable - The Drawable class is the base class for storing geometric data, and Geode stores them in a std::list<osg::Drawable>. Drawable is a pure virtual class and can't be instantiated directly. You must use a derived class, such as Geometry or ShapeDrawable (which allows your application to draw predefined shapes such as spheres, cones, and boxes).
- Geometry-The Geometry class, in conjunction with the PrimitiveSet class, act as high-level wrappers around the OpenGL vertex array functionality. Geometry stores the vertex arrays vertex, texture coordinate, color, and normal arrays.
- PrimitiveSet-The PrimitiveSet class provides high-level support for the OpenGL vertex array drawing commands. Use this class to specify the types of primitives to draw from the data stored in the associated Geometry class.
- Vector classes (Vec2, Vec3, etc.)-OSG provides a set of predefined 2-, 3-, and 4-element vectors of type float or double. Use these vectors to specify vertices, colors, normals, and texture coordinates.
- Array classes (Vec2Array, Vec3Array, etc.)-OSG defines several commonly used array types, such as Vec2Array for texture coordinates. When specifying vertex array data, your application stores geometric data in these arrays before passing them to Geometry objects.

1.2.3 State Management Classes

OSG provides a mechanism for storing the desired OpenGL rendering state in the scene graph. During the cull traversal, geometry with identical states is grouped together to minimize state changes. During the draw traversal, the state management code keeps track of the current state to eliminate redundant state changes.

Unlike other scene graphs, OSG allows state to be associated with any scene graph node, and state is inherited hierarchically during a traversal.

- `StateSet` - OSG stores a collection of state values (called modes and attributes) in the `StateSet` class. Any osg::Node in the scene graph can have a `StateSet` associated with it.
- Modes - Analogous to the OpenGL calls `glEnable()` and `glDisable()`, modes allow you to turn on and off features in the OpenGL fixed-function rendering pipeline such as lighting, blending, and fog. Use the method `osg::StateSet::setMode()` to set an OpenGL mode in a `StateSet`.
- `Attributes` - Attributes store state parameters, such as the blending function, material properties, and fog color. Use the method `osg::StateSet::setAttribute()` to store an attribute in a `StateSet`.
- Texture attributes and modes - These attributes and modes apply to a specific texture unit in OpenGL multi-texturing. Unlike OpenGL, there is no default texture unit; your application must supply the texture unit when setting texture attributes and modes. To set these state values and specify their texture unit, use the `StateSet` methods `setTextureMode()` and `setTextureAttribute()`.
- Inheritance flags - OSG provides flags for controlling how state is inherited during a scene graph traversal. By default, state set in a child node overrides the same state set in a parent node. However, you can force parent state to override child node state, and you can specify that child state be protected from parent overriding.

1.2.4 Utilities and Other Classes

Finally, the osg library contains several useful classes and utilities. Some of these deal with the OSG reference-counted memory scheme, which helps avoid memory leaks by deleting unreferenced memory.

1.3 The `osgUtil` Library

The `osgUtil` library is a broad collection of utilities for processing a scene graph and modifying the geometry within it. Chapter 5 covers this library in greater detail.

The `osgUtil` library is probably best known for the set of classes that support the update, cull, and draw traversals. In a typical OSG application, these traversals are handled by higher-level support classes, such as `osgViewer::Viewer`, and you don't have to interact with them directly.

1.3.1 Intersection

Typically, 3D applications support user interaction or selection, such as picking. The osgUtil library efficiently supports picking with a variety of classes that test the scene graph for intersection.

- Intersector - This is a pure virtual class that defines an interface for intersection testing. The osgUtil library derives several classes from Intersector, one for each type of geometry (line segment, plane, etc.). To perform an intersection test, your application instantiates one of the classes derived from Intersector, passes it to an instance of IntersectionVisitor, and queries it for intersection results.
- IntersectionVisitor - The IntersectionVisitor class searches a scene graph for nodes that intersect a specified piece of geometry. Classes derived from Intersector perform the actual intersection tests.
- LineSegmentIntersector - Derived from Intersector, the LineSegmentIntersector class tests for intersections between a specified line segment and a scene graph, and provides methods for the application to query the results.
- PolytopeIntersector - Like LineSegmentIntersector, the PolytopeIntersector class tests for intersections against a polytope defined by a list of planes. This class is especially useful for picking, in which the polytope defines a bounded area in world space around the mouse-click point.
- PlaneIntersector - Like LineSegmentIntersector, the PlaneIntersector class tests for intersections against a plane that is bounded by a list of planes.

1.3.2 Optimization

The scene graph data structure is ideally suited for optimization and easy statistics gathering. The osgUtil library contains classes that traverse the scene graph to modify it for optimal rendering and gather statistical data about its contents. You can use the statistical data to guide tuning efforts for increased performance.

1.3.3 Geometry Manipulation

Many 3D applications require modification of loaded geometry to achieve desired performance or rendering results. The osgUtil library contains classes to provide several types of common geometrical operations, such as simplification, tessellation, triangulation, and stripification.

1.4 The osgDB Library

The osgDB library allows applications to load, use, and write 3D databases. The osgDB plugin architecture provides support for a wide variety of common 2D image and 3D model file formats. The osgDB maintains a registry of and oversees access to the loaded OSG plugins.

OSG supports its own file formats. `.osg` is a plain ASCII text description of a scene graph, and `.osga` is an archive (or group) of .osg files. The `osgDB` library contains support code for these file formats. (OSG also supports a binary `.ive` format.)

Large 3D terrain databases are often created in sections that tile together. In this case, applications require that portions of the database load from file in a background thread without interrupting rendering. The `osgDB::DatabasePager` provides this functionality.

Chapter 6 covers this library in greater detail.

Chapter 2

OpenSceneGraph Changes: v1.2 to v2.2

2.1 Version Number

The OSG version number number changed from `1.2.0.0` to `2.2.0` . The version component format changed as well: `OSG_VERSION_RELEASE` and `OSG_VERSION_REVISION` have been replaced by `OSG_VERSION_PATCH` (`OSG_VERSION_MAJOR` and `OSG_VERSION_MINOR` are unchanged).

This change is potentially incompatible if your code references the removed version components.

2.2 osg: New Additions

2.2.1 RenderInfo Class

This class associates a `View` object with a `State` object. Its introduction impacts OSG in several ways.

- The first parameter to `Drawable::drawImplementation` changed. In v1.2, the first parameter is `osg::State&`. In v2.2, it is `osg::RenderInfo&`. This is an incompatible change if your code contains classes that derive from `Drawable` and override the `drawImplementation` method.
- The parameter to `Drawable::draw` changed. In v1.2, it is `osg::State&`. In v2.2 it is `osg::RenderInfo&`. This is an incompatible change if your code contains classes that derive from `Drawable` and override the `draw` method.
- The `Drawable::CullCallback::cull` method in v1.2 is now deprecated. A new overloaded `cull` method has been added in v2.2. It takes an `osg::RenderInfo&` as a parameter. Code that uses the deprecated `Drawable::CullCallback::cull` method should port to the new `cull` method.
- To support the new `RenderInfo` class, `CullVisitor`, `GLObjectsVisitor`, and `SceneView` all feature a new `getRenderInfo` method.

2.2.2 View Class

This class owns a master `Camera`, and possibly several slave `Cameras` controlled by the master camera to create tiled rendering. It also manages consistent viewer lighting and statistics collection across all `Camera` objects.

2.2.3 Stats Class

This class is used by the new `osgViewer` library to gather rendering statistics.

2.2.4 TemplatePrimitiveFunctor Class

This class is used by `osgUtil::PolytopeIntersector` to support `osgUtil`'s new intersection framework.

2.2.5 StencilTwoSided Class

This class supports the two-sided stencil feature. New stencil operations were also added: `DECR_WRAP` and `INCR_WRAP`.

2.2.6 ScreenIdentifier Class

This class contains screen and display identification information. `GraphicsContext::Traits` now inherits from `GraphicsContext::ScreenIdentifier`.

2.2.7 DataVariance Enumerant

v2.2 introduces a new `Object::DataVariance` enumerant, `UNSPECIFIED`. All `Objects` now default to `UNSPECIFIED` instead of `DYNAMIC`.

This change could impact code that relies on the v1.2 default value.

v2.2 defines a new method, `Object::computeDataVariance` as a no-op. The method is declared virtual. Both `StateSet` and `Drawable` override it to support the new `STATIC_OBJECT_DETECTION` `osgUtil::Optimizer` flag.

2.2.8 TransferFunction Class

`TransferFunction` provides a 1D,2D or 3D color look up table that can be used on the GPU as a 1D, 2D or 3D texture. Typical uses include mapping heights to colors when contouring terrain or mapping intensities to colors when volume rendering.

2.2.9 Plane

The Plane object's default internal precision changed from single precision floats in v1.2 to double precision floats in v2.2. The default is configurable at build time using CMake. Many interface methods changed to support the new default precision.

Several new Plane methods are available in v2.2. dotProductNormal, intersect, isNaN, and num_components.

2.2.10 StateSet: Access To Dynamic Object Counts

v2.2 introduces two new methods to the State class. setDynamicObjectCount and getDynamicObjectCount are used by osgUtil::RenderStage and the new osgViewer library to synchronize cull and draw thread access.

2.2.11 Support For OpenGL Hints

A new StateAttribute class, Hint, was added to allow code to specify OpenGL hints.

2.2.12 VariablesMask Enumerants

v2.2 introduces new enumerants to the CullSettings::VariablesMask to support slave Camera objects in the View class. The new enumerants are CLEAR_COLOR, LIGHTING_MODE, and LIGHT.

2.2.13 ReferenceFrame Enumerant

v2.2 introduces a new enumerant to Transform::ReferenceFrame. The ABSOLUTE_RF_INHERIT_VIEWPOINT reference frame is useful for render-to-texture (RTT) operations that must use the actual viewpoint for LOD computations rather than the RTT Camera viewpoint.

2.2.14 AnimationPath

AnimationPath now supports a clear operation.

2.2.15 Matrix

v2.2 introduces a new Matrix method, isIdentity, used by the osgUtil::Optimizer to identify no-op transformations.

2.2.16 HeightField

Several changes to the `HeightField` class to support a new functionality in the `osgTerrain` library.

2.2.17 Timer

Several new convenience methods were added to `Timer` to obtain delta time values in different formats.

2.2.18 Billboard

The `Billboard` object was modified to resolve an issue related to degenerate transformation matrices.

2.2.19 Buffer Object Support

OSG v2.2 inproves support for OpenGL buffer objects, which can be used for both vertex and element data. `PrimitiveSet` classes (such as `DrawElements`) contain several changes related to these improvements.

2.2.20 Changes to GraphicsContext

The `GraphicsContext` class contains significant changes in v2.2.

2.2.21 Enhancements to ImageStream

v2.2 introduces new methods to ImageStream to support length inquiries and looping.

2.2.22 `buffered_value` And `buffered_object`

The `buffered_value` and `buffered_object` classes, used internally to store per-context information, now support a `resize` method to change the size of their internal arrays.

2.2.23 Math

A new function, `signOrZero` was added, which returns -1, 0, or 1 for negative, zero, or positive input.

2.2.24 TexMat

`TexMat` can now be scaled so that normalized (0..1) texture coordinates can be used with `TextureRectangle`.

2.2.25 Access To GPU Time

v2.2 adds support for OpenGL's query feature. The new `osgViewer` library queries the GPU for elapsed time to display hardware usage in the statistics display.

2.2.26 Polytope

`Polytope` now has a new convenience function, `setToBoundingBox`, which sets six plane equations corresponding to the `BoundingBox` limits.

2.3 osg: Bugfixes, Deprecations, and API Changes

2.3.1 Camera and CameraNode

The v1.2 `CameraNode` object has been renamed to `Camera` in v2.2. Correspondingly, the signature of several methods that take `CameraNode` as a parameter or return it as a value were modified to reflect the name change.

This impacts any code that uses `CameraNode`, includes the `CameraNode` header file, or uses any of the methods that take `CameraNode` as a parameter or return it as a value.

2.3.2 Enhancements to PointSprite and Image

The `PointSprite` and `Image` objects were enhanced to provide support for OpenGL's coordinate origin modes.

2.3.3 Memory Leak Fixed

`ArgumentParser` now keeps a `ref_ptr<>` to its `ApplicationUsage` member variable. `ApplicationUsage` now derives from `Referenced`.

2.3.4 AutoTransform

`AutoTransform` was modified to recompute its position after changing the auto rotate mode.

2.3.5 UnitTestFramework Removed

The `UnitTestFramework` source and header files have been removed and their classes deleted.

2.3.6 ClipNode Method Name Corrected

In v1.2, the `ClipNode` method to set a `ClipPlaneList` was incorrectly named `getClipPlaneList`. v2.2 renames this method as `setClipPlaneList`.

2.3.7 CullStack Method Return Value

The `CullStack` methods `getMVPW`, `getModelViewMatrix`, and `getProjectionMatrix` formerly returned a `RefMatrix&` in v1.2. In v2.2, they all return a `RefMatrix*`.

This is an incompatible change for any code that uses these functions.

2.3.8 CullSettings

The `inheritCullSettings` method is now virtual.

2.3.9 BlendColor

In v1.2, the `BlendColor::getConstantColor` method returned a copy of the color as a `Vec4`. v2.2 has two `getConstantColor` methods, and both return a `Vec4&`. One returns a const reference, the other returns a non-const reference.

2.3.10 Math

The `equivalent` functions now return bool instead of double/float.

2.3.11 ArrayList

In `Geometry`, The typedef `ArrayList` in v1.2 was renamed `ArrayDataList` in v2.2. The `Geometry` methods `getTexCoordArrayList` and `getVertexAttribArrayList` now return `ArrayDataList` instead of `ArrayList`.

OSG v2.2 has a typedef called `ArrayList`, but it differs from the v1.2 `ArrayList`. It is used by `Geometry::getArrayList`, which returns all `Geometry` array data as a `std::vector< osg::Array* >`.

This is an incompatible change.

2.3.12 State

The `State` member class `StateSetStack` moved from protected to public. v2.2 also adds a new public method, `getStateSetStack`.

State has a new member class, DynamicObjectRenderingCompletedCallback, used in the osgViewer library's thread models to notify code when DYNAMIC nodes have been rendered. v2.2 also adds some related support methods, such as decrementDynamicObjectCount.

2.3.13 Introduction Of Simulation Time

The FrameStamp class now supports a new time concept, simulation time. Many aspects of OSG which operated using reference time in v1.2 now use simulation time in v2.2, such as AnimationPath, Sequence, osgUtil::TransformCallback, several classes in the osgSim library (commonly used to support OpenFlight models), several classes in the osgParticle library, and the GEO plugin. Reference time continues to be used by the osgDB DatabasePager.

This is an incompatible change. If your application uses SceneView, you must modify your code to explicitly set the simulation time each frame in the same way that you set the reference time in v1.2.

2.3.14 DeleteHandler

DeleteHandler was used internally in v1.2. v2.2 moves this class into its own header and implementation files and exports it for use by other code. The class was also modified to allow non-immediate deletion of controlled memory.

2.3.15 Viewport

The Viewport class now stores x, y, width, and height internally as doubles instead of ints. Many of its methods' parameters changed accordingly.

v2.2 changes the behavior of the Viewport::valid method. It now returns true if width and height are positive. (In v1.2, it returned true if they were non-zero.)

2.3.16 GraphicsThread

v2.2 introduced several changes to GraphicsThread and related classes.

2.3.17 StateAttribute

StateAttribute::apply(State&) is pure virtual in v1.2. In v2.2 it is defined as a no-op method. Classes derived from StateAttribute no longer must override this method.

2.3.18 Enhancements to Sequence

The Sequence class has been enhances to provide better support for OpenFlight animation sequences.

2.3.19 Change to PrimitiveSet

`PrimitiveSet::dirty` is now virtual in v2.2 so that it can be overridden in `DrawElements`, which also dirties the element buffer object.

2.4 osgUtil: New Additions

2.4.1 ComputeBoundsVisitor

A new class, `ComputeBoundsVisitor`, was added to facilitate bounding box computation.

2.4.2 Statistics and StatsVisitor Exported

The `Statistics` and `StatsVisitor` classes are now exported from `osgUtil`, primarily so that they can be used by the new `osgViewer` library.

2.4.3 osgUtil::SceneView and osg::View

`SceneView` was modified to support the new `osg::View` class. Accessor methods `getView` and `setView` were added.

2.4.4 Optimizer Flag Added

A new `osgUtil::Optimizer` flag is available, `STATIC_OBJECT_DETECTION`, which is included in the `DEFAULT_OPTIMIZATIONS` flag set. This flag causes `StateSets` (and related attributes and uniforms) with `UNSPECIFIED` data variance to be made either `DYNAMIC` or `STATIC` depending on whether they have any update/event callbacks associated with them or not. This behavior is the same for `Drawables`, but based upon consideration of cull/update/event callbacks.

This change could impact code that relies on the v1.2 default `DataVariance` value because the `Optimizer` could change it to either `STATIC` or `DYNAMIC`.

2.4.5 osgUtil::RenderLeaf

`osgUtil::RenderLeaf` now lets you access its `Drawable` with the `getDrawable` method.

2.4.6 Dynamic Object Counts

v2.2 introduces two new methods to the `SceneView` class. `setDynamicObjectCount` and `getDynamicObjectCount` are used by `RenderStage` and the new `osgViewer` library to synchronize cull and draw thread access to objects of `DYNAMIC` variance.

2.4.7 Intersection Support

v2.2 introduces a new framework for intersecting scene graph geometry. The `IntersectionVisitor` class, and `Intersector` (and derived classes), work together to provide software with intersection information. The `IntersectVisitor` class is now deprecated.

Code that uses `IntersectVisitor` should port to the new framework.

2.5 osgUtil: Bugfixes, Deprecations, and API Changes

2.5.1 LeafDepthSortFunctor

The v1.2 `StateGraph::LeafDepthSortFunctor` object has been renamed to `LessDepthSortFunctor` in v2.2. This class is used internally and isn't exported. It appears in the `StateGraph` header only to facilitate inlined `StateGraph` methods.

2.5.2 Tesselator

The `Tesselator` class name, and all occurrences of this spelling throughout the code, was corrected to `Tessellator`.

This is an incompatible change. When porting code that uses the `Tesselator` class from v1.2 to v2.2, you must modify your code to account for this spelling change.

2.5.3 SceneView

`SceneView` now derives from `osg::Object` instead of directly from `osg::Referenced`.

2.6 `osgDB`: New Additions

2.6.1 Improved Plugin Access for Static Linking

In v2.2, plugins export a symbol to facilitate plugin linking and registration by statically-linked applications. The `osgDB Registry` header defines a compiler preprocessor macro, `USE_OSGPLUGIN`, that allows applications to conveniently reference this symbol.

2.6.2 Improved Extension Alias Registration

v2.2 introduces a new `Registry` method, `readPluginAliasConfigurationFile`, which loads a text file containing parameters to `addFileExtensionAlias`. This allows code to register several extension aliases for a plugin with a single function call.

2.6.3 Changes to DatabasePager

The `DatabasePager` was modified to allow code to specify an `Options` object with a load request.

The `DatabasePager` was also modified to work with osgViewer.

2.7 `osgDB`: Bugfixes, Deprecations, and API Changes

2.7.1 ReentrantMutex

The `ReentrantMutex` class has moved out of the `osgDB` library and is now part of `OpenThreads`. As a result, its namespace has changed from `osgDB` to `OpenThreads`.

Code that references `ReentrantMutex` should be modified to reflect the namespace change.

2.7.2 Copy Constructors

Copy constructors for three classes were modified to take an `osg::CopyOp&` instead of an `osg::CopyOp`. The affected classes are `ReaderWriter`, `ReaderWriter::Options`, and `ImageOptions`. These classes are now consistent with copy constructors throughout OSG.

2.8 Other Changes within the OpenSceneGraph Suite

2.8.1 New osgViewer Library

`osgViewer` provides native windowing support and viewer functionality that scales from a single view to multiple display systems.

2.8.2 New osgManipulator Library

`osgManipulator` provides a set of interactive manipulators for scaling, rotating and moving objects in the scene.

2.8.3 New osgShadow Library

`osgShadow` provides an extensible framework for adding dynamic shadows to the scene.

2.8.4 OpenThreads Library Name Change

v2.2 changes the name of the OpenThreads library on Windows platforms. Called OpenThreadsWin32.{lib,dll} in v1.2, the libraries are called OpenThreads.{lib,dll} in v2.2.

This is an incompatible change. When porting Windows-based code that links with OpenThreads from v1.2 to v2.2, the project files must be updated to reflect the library name change.

2.8.5 osgIntrospection

Added full qualification to three classes, osg::Drawable::AttributeFunctor, osg::Drawable::ConstAttributeFunctor, and osg::StateAttribute::ModeUsage, where they were referenced in subclasses. This allows them to be properly reflected in osgIntrospection.

2.8.6 Enhanced Overlay Support

The osgSim Library now supports overlaying text, country boundaries, and other geometry and imagery directly on terrain, including paged databases.

2.8.7 Plugins

- v2.2 includes many improvements to the OpenFlight, COLLADA, AC3D, and DDS plugins.
- v2.2 includes a new TXF texture font reader.
- Finally, v2.2 introduces a new VRML2 plugin, based on the OpenVRML library.

Chapter 3

OpenSceneGraph Environment Variables

3.1 Introduction

OpenSceneGraph's run-time behavior can be changed by setting environment variables. This chapter contains an alphabetical reference of all environment variables used by OpenSceneGraph v2.2 and their valid values or range of values. Default values are listed in ***this font*** within the documented valid values in this chapter.

3.2 Environment Variables

3.2.1 DISPLAY

OSG supports the DISPLAY environment variable on X11-based windowing systems to specify the client display and screen information.

3.2.2 DYLD_LIBRARY_PATH

Specifies the Apple-specific contribution to the dynamic library search paths. See `OSG_FILE_PATH` for more info.

3.2.3 LD_LIBRARY_PATH

Specifies the platform-specific contribution to the dynamic library search paths for platforms other than Windows and Apple. See `OSG_FILE_PATH` for more info.

3.2.4 LD_LIBRARY64_PATH, LD_LIBRARYN32_PATH

3.2.5 LD_LIBRARY_PATH

These environment variables specify additional dynamic library directories. If both are set, `LD_LIBRARY_PATH` is ignored.

3.2.6 OSG_COMPILE_CONTEXTS

When set to `ON`, OSG creates a second context for each existing rendering context. This second context is configured to share display lists and objects with the rendering context. OSG uses this second context to create display lists and texture objects in an offline thread. Enabling this feature can prevent frame stalls in applications that need to load new data while simultaneously rendering.

Valid values:

- ***OFF***
- `ON`

3.2.7 OSG_COMPUTE_NEAR_FAR_MODE

This environment variable specifies how or if OSG computes the near and far clip planes. By default, OSG attempts to maximize depth buffer precision by computing the largest possible near plane location and smallest possible far plane location based on the scene graph bounding volume. OSG can also compute the near plane even more precisely by considering each primitive in the scene graph, though this can be more expensive.

Some applications must disable this feature and use the near and far planes implicit in the projection matrix, for example, when mixing OSG rendering with direct OpenGL commands.

Valid values:

- `DO_NOT_COMPUTE_NEAR_FAR`
- ***COMPUTE_NEAR_FAR_USING_BOUNDING_VOLUMES***
- `COMPUTE_NEAR_FAR_USING_PRIMITIVES`

3.2.8 OSG_CONFIG_FILE

Causes `osgViewer::Viewer` to read a config file containing an `osgViewer::View` object. The `Viewer` configures its view parameters to match the retrieved `View`.

If this variable is set, the `OSG_SCREEN` and `OSG_WINDOW` environment variables are ignored.

3.2.9 OSG_DATABASE_PAGER_DRAWABLE

Specifies the OpenGL rendering strategy of `Drawables` created by the `DatabasePager`. This variable can affect paging speed on systems with slow display list compilation performance.

Valid values:

- ***`DoNotModify`*** : The `DatabasePager` doesn't modify the rendering strategy specified in the model file. *Default: All platforms but Mac OS X*
- `DisplayList` (or `DL`): The `DatabasePager` calls `setUseDisplayLists( true )` and `setUseVertexBufferObjects( false )` on all `Drawables`.
- ***`VertexArrays`*** (or `VA`): The `DatabasePager` calls `setUseDisplayLists( false )` and `setUseVertexBufferObjects( false )` on all `Drawables`. *Default: Mac OS X*
- `VBO`: The `DatabasePager` calls `setUseDisplayLists( true )` and `setUseVertexBufferObjects( true )` on all `Drawables`.

3.2.10 OSG_DATABASE_PAGER_GEOMETRY

This is a deprecated synonym for `OSG_DATABASE_PAGER_DRAWABLE`.

3.2.11 OSG_DATABASE_PAGER_PRIORITY

Controls the priority of the `DatabasePager` thread by calling the `OpenThreads` `setSchedulePriority()` method. If this value is not set, the `DatabasePager` does not change its default priority (the default is determined by `OpenThreads`).

Valid values:

- ***`DEFAULT`***
- `MIN`
- `LOW`
- `NOMINAL`
- `HIGH`
- `MAX`

3.2.12 OSG_DEFAULT_BIN_SORT_MODE

OSG places `Drawables` in render bins during the cull traversal and sorts them before drawing them. This environment variable controls the sort algorithm that the default render bin uses.

Valid values:

- ***SORT_BY_STATE*** : OSG groups Drawables together if they share the same StateSet.
- SORT_BY_STATE_THEN_FRONT_TO_BACK: This is like SORT_BY_STATE, except groups of Drawables with the same StateSet are further sorted based on the distance from the eye to the center of the groups' bounding volumes.
- SORT_FRONT_TO_BACK: OSG sorts Drawables to render in front to back order.
- SORT_BACK_TO_FRONT: OSG sorts Drawables to render in back to front order. This is useful for alpha blending algorithms.

3.2.13 OSG_DISABLE_FAST_PATH_IN_DISPLAY_LISTS

When this variable is set, Geometry that uses display lists is forced to use the OpenGL slow rendering path (glBegin/glEnd) instead of a fast path (vertex arrays or buffer objects).

Default value: By default, OSG determines the use of the fast or slow path on a per Drawable basis, considering many factors.

Note: To avoid the OpenGL slow path, avoid using BIND_PER_PRIMITIVE, and avoid using indices in conjunction with BIND_PER_VERTEX.

3.2.14 OSG_DISPLAY_TYPE

Specifies the type of display.

Valid values:

- ***MONITOR*** : Ignored.
- POWERWALL: Ignored.
- REALITY_CENTER: Ignored.
- HEAD_MOUNTED_DISPLAY: Causes OSG's internal SceneView object to scale the left and right projection matrices for proper stereo viewing.

3.2.15 OSG_DO_PRE_COMPILE

By default, the DatabasePager precompiles display lists and creates other OpenGL objects, such as display lists. If this environment variable is set to any invalid value, precompile is disabled.

Valid values:

- YES (case insensitive)
- ON (case insensitive)

Default behavior: DatabasePager precompile is enabled.

3.2.16 OSG_DRIVE_MANIPULATOR_HEIGHT

Specifies the view offset from ground level that the `DriveManipulator` uses. The value is in world coordinate units.

Valid values: Any floating-point number.

Default value: ***1.5***

3.2.17 OSG_EYE_SEPARATION

Specifies the eye separation for stereo viewing. The value is in world coordinate units.

Valid values: Any floating point number.

Default value: ***0.05***

3.2.18 OSGFILEPATH

This is a deprecated synonym for `OSG_FILE_PATH`.

3.2.19 OSG_FILE_PATH

This environment variable specifies directory locations for data files, such as 3D model and texture files. The value is a list of directories, delimited by a semicolon (";") on Windows and a colon (":") on all other platforms.

By default, OSG looks for data files in the current working directory and any directories specified to the `osgDB` in the `ReaderWriter::Options` data file path list.

3.2.20 OSG_GL_EXTENSION_DISABLE

Specifies a list of OpenGL extensions to disable on a per-renderer basis.

The format of the value is: `GLRendererString : ExtensionName [, ExtensionName][...][; GLRendererString : ExtensionName [, ExtensionName][...]][...]`

For example: `SUN_XVR1000:GL_EXT_texture_filter_anisotropic` This disables the extension named `GL_EXT_texture_filter_anisotropic` if the OpenGL renderer string matches `SUN_XVR1000`.

By default, all extensions are enabled on all renders.

3.2.21 OSG_LD_LIBRARY_PATH

This is a deprecated synonym for `OSG_LIBRARY_PATH`.

3.2.22 OSG_LIBRARY_PATH

OSG uses the directories in this environment variable when searching for dynamically loaded libraries (such as plugins). If OSG doesn't find the dynamic library, it also searches platform-specific locations (see `DYLD_LIBRARY_PATH`, `LD_LIBRARY_PATH`, and `PATH`).

3.2.23 OSG_MAX_NUMBER_OF_GRAPHICS_CONTEXTS

Specifies the initial number of graphics contexts. If OSG requires more than this initial value, OSG adds more as needed. (This is not an upper limit.)

Valid values: Any integer.

Default value: ***32***

3.2.24 OSG_MAX_TEXTURE_SIZE

Sets the maximum dimension (width, height, and depth) of an OpenGL texture object. This is typically used as a diagnostic tool to determine whether excessive texture size causes poor performance.

Default value: OSG obtains the maximum dimension by querying OpenGL's `GL_MAX_TEXTURE_SIZE` state variable.

3.2.25 OSG_MAXIMUM_OBJECTS_TO_COMPILE_PER_FRAME

The `DatabasePager` compiles display lists, buffer objects, and texture objects prior to rendering with those objects. This environment variable controls the number of OpenGL objects that the `DatabasePager` will attempt to compile per frame.

Default value: ***4***

3.2.26 OSG_MINIMUM_COMPILE_TIME_PER_FRAME

This environment variable is reserved for future use.

3.2.27 OSG_NEAR_FAR_RATIO

OSG uses the value of this variable to avoid loss of depth buffer precision when automatically computing the near and far clip planes. OSG multiplies this value by the computed far plane to compute a minimum

value for the near plane.

Default value: ***0.0005***

3.2.28 OSGNOTIFYLEVEL

This is a deprecated synonym for `OSG_NOTIFY_LEVEL`.

3.2.29 OSG_NOTIFY_LEVEL

This variable controls the verbosity of OSG messages written to `std::cout`. Increasing the verbosity can be a useful debugging aid, because OSG displays a large amount of additional information at the `DEBUG` and `INFO` levels.

Valid values, in order of increasing verbosity:

- `ALWAYS`
- `FATAL`
- `WARN`
- ***`NOTICE`***
- `INFO`
- `DEBUG`
- `DEBUG_INFO`
- `DEBUG_FP`

Note that values are not case sensitive.

3.2.30 OSG_OPEN_FLIGHT_PLUGIN

OSG v2.0 introduces a new OpenFlight import plugin, replacing the previous one. If this environment variable is set to any invalid value, OSG disables use of the new OpenFlight import plugin. This might be useful if your application requires the old (deprecated) OpenFlight import plugin.

Valid value: `new`

Default value: ***new***

3.2.31 OSG_OPTIMIZER

This environment variable specifies the scene graph modifications that are performed if your application uses the `osgUtil::Optimizer` class. If your application doesn't use the `osgUtil::Optimizer`, this environment variable has no effect. [1]

Valid values: `osgUtil::Optimizer` searches the environment variable string for the following substrings. All substrings (except `OFF`) can be preceded by a tilde ( ) to remove the operation. For example, `DEFAULT FLATTEN_STATIC_TRANSFORMS` causes the `Optimizer` to use all default operations except the `FLATTEN_STATIC_TRANSFORM` operation.

- `OFF`: No operations are performed.
- `DEFAULT`: All default operations are performed. (See below for the definition of `DEFAULT`.)
- `FLATTEN_STATIC_TRANSFORMS`: Tranform `Geometry` and `Billboard` vertices and normals by static Tranform nodes, then replace the Transform with an identity transformation. This operation reduces vertex-matrix multiplications performed at render time.
- `REMOVE_REDUNDANT_NODES`: Removes empty `Geometry`, `Geode`, and `Group` objects with no children. Removes `Group` objects and identity `Transform` nodes that have only one `Group` child.
- `REMOVE_LOADED_PROXY_NODES`: Removes `ProxyNode` objects that have already loaded their data file.
- `COMBINE_ADJACENT_LODS`: Combines sibling `LOD` nodes into a single `LOD` node. This reduces the number of times OSG has to compute the distance from the eye to the `LOD` node at render time.
- `SHARE_DUPLICATE_STATE`: Replaces copies of identical state such as `StateSet`, `StateAttribute`, and `Uniform`, with references. This reduces the number of overall state items and improves draw time by reducing the state management workload.
- `MERGE_GEODES`: Merges sibling `Geode` objects that share the same `StateSet` and `NodeMask` into a single `Geode` containing all `Drawable` objects.
- `MERGE_GEOMETRY`: Merges multiple `Geometry` objects into a single `Geometry` object.
- `SPATIALIZE_GROUPS`: Adds additional `Group` nodes to a scene graph to organize the scene graph spatially. This makes culling more effective and can therefore increase cull- and render-time performance.
- `COPY_SHARED_NODES`: Makes copies of any shared nodes. This allows operations that require unique children, such as `FLATTEN_STATIC_TRANSFORMS`, to operate more efficiently.
- `TESSELLATE_GEOMETRY`: Executes the `osgUtil::Tessellator` on polygons in `Geometry` objects, which eliminates all polygons.
- `TRISTRIP_GEOMETRY`: Executes the `osgUtil::TriStripVisitor`, which creates `GL_TRIANGLE_STRIP` primitives from non-strip primitives in `Geometry` objects.
- `OPTIMIZE_TEXTURE_SETTINGS`: Configures textures to unreference their `Image` objects after creating OpenGL texture objects from the `Image` data. This reduces host memory usage by freeing image data already stored in graphics memory.

[1] If the `OSG_NOTIFY_LEVEL` is set to `INFO`, the `osgUtil::Optimizer` writes several informative messages to `std::cout`.

- CHECK_GEOMETRY: Sets the correct data binding for all Geometry data arrays.
- FLATTEN_BILLBOARDS: Transforms Billboard positions by MatrixTransform parents, then removes the transformation.
- TEXTURE_ATLAS_BUILDER: Combines separate textures into texture atlases to reduce the overall number of texture objects, and therefore the number of glBindTexture() commands required to render the scene graph.
- STATIC_OBJECT_DETECTION: Looks for objects with DataVariance
- UNSPECIFIED and changes it to STATIC if the objects have no callbacks associated with them. This can improve rendering performance for multithreaded osgViewer rendering.
- Default value: By default, the Optimizer uses the DEFAULT setting, which is equivalent to setting this environment variable to the following value: FLATTEN_STATIC_TRANSFORMS REMOVE_REDUNDANT_NODES REMOVE_LOADED_PROXY_NODES COMBINE_ADJACENT_LODS SHARE_DUPLICATE_STATE MERGE_GEOMETRY CHECK_GEOMETRY OPTIMIZE_TEXTURE_SETTINGS STATIC_OBJECT_DETECTION

3.2.32 OSG_PROXY_HOST

Specifies the proxy host name used by the .NET plugin.

3.2.33 OSG_PROXY_PORT

If OSG_PROXY_HOST is specified, this environment variable specifies the host port number. If unspecified, the .NET plugin uses port ***8080***.

3.2.34 OSG_RECORD_CAMERA_PATH_FPS

Specifies the desired number of samples per second used by osgViewer::RecordCameraPathHandler when recording a Camera animation path.

Valid values: Any floating-point number.

Default value: ***25.0***

3.2.35 OSG_RUN_FRAME_COUNT

Specifies the number of frames rendered by the render loop in Viewer::run() and CompositeViewer::run(). By default, run() renders until interrupted.

Valid values: Any integer.

3.2.36 OSG_SCREEN

If set to a non-negative number, this variable controls the single screen used by `Viewer` objects. It's equivalent to calling `Viewer::setUpViewOnSingleScreen( n )`, where n is the screen number. This variable is only used if `OSG_WINDOW` is not set (or is set but specifies a zero width and height), and if `OSG_CONFIG_FILE` is not set.

See also `OSG_WINDOW`, `OSG_CONFIG_FILE`.

Valid values: Any non-negative integer.

3.2.37 OSG_SCREEN_DISTANCE

Specifies the physical distance between the eyes and the screen. Used to compute the default `osg::View` field of view, and used for correct stereo viewing by `osgUtil::SceneView`.

Valid values: Any floating point number.

Default value: ***0.5***

3.2.38 OSG_SCREEN_HEIGHT

Specifies the physical height of the screen. Used to compute the default `osg::View` field of view.

Valid values: Any floating point number.

Default value: ***0.26***

3.2.39 OSG_SCREEN_WIDTH

Specifies the physical width of the screen. Used to compute the default `osg::View` field of view.

Valid values: Any floating point number.

Default value: ***0.325***

3.2.40 OSG_SERIALIZE_DRAW_DISPATCH

When set to `ON`, `osgViewer` uses a mutex to ensure that all draw traversals are performed sequentially.

Valid values:

- ***OFF***
- `ON`

3.2.41 OSG_SPLIT_STEREO_AUTO_ADJUST_ASPECT_RATIO

Specifies whether or not OSG attempts to compensate for the compression of the aspect ratio when using split screen stereo.

Valid values:

- OFF
- ***ON***

3.2.42 OSG_SPLIT_STEREO_HORIZONTAL_EYE_MAPPING

Speficies the viewport for display of the left eye image in a horizontal split stereo display. Only used when OSG_STEREO_MODE is HORIZONTAL_SPLIT.

Valid values:

- ***LEFT_EYE_LEFT_VIEWPORT***
- LEFT_EYE_RIGHT_VIEWPORT

3.2.43 OSG_SPLIT_STEREO_HORIZONTAL_SEPARATION

Specifies the number of pixels between viewports. Only used when OSG_STEREO_MODE is HORIZONTAL_SPLIT.

Valid values: Any integer.

Default value: ***0***

3.2.44 OSG_SPLIT_STEREO_VERTICAL_EYE_MAPPING

Speficies the viewport for display of the left eye image in a vertical split stereo display. Only used when OSG_STEREO_MODE is VERTICAL_SPLIT.

Valid values:

- ***LEFT_EYE_TOP_VIEWPORT***
- LEFT_EYE_BOTTOM_VIEWPORT

3.2.45 OSG_SPLIT_STEREO_VERTICAL_SEPARATION

Specifies the number of pixels between viewports. Only used when OSG_STEREO_MODE is VERTICAL_SPLIT.

Valid values: Any integer.

Default value: ***0***

3.2.46 OSG_STEREO

Specifies whether or not OSG should use stereo rendering.

Valid values:

- ***OFF***
- ON

3.2.47 OSG_STEREO_MODE

When set to a valid value, enables stereo and displays stereo images using the specified stereo mode.

Valid values:

- QUAD_BUFFER
- ***ANAGLYPHIC***
- HORIZONTAL_SPLIT
- VERTICAL_SPLIT
- LEFT_EYE
- RIGHT_EYE
- VERTICAL_INTERLACE
- HORIZONTAL_INTERLACE

3.2.48 OSG_TEXT_INCREMENTAL_SUBLOADING

When set to a valid value of `OFF` or `Off`, forces the `osgText::Font` object to subload all font glyphs at once. If set to any invalid value, glyphs are subloaded incrementally.

Valid values:

- OFF
- Off

Default value: The default is `OFF` (disable incremental subloading), except if the OpenGL `GL_RENDERER` string contains any of the substrings “IMPACT”, “Radeon”, or “RADEON”.

3.2.49 OSG_THREAD_SAFE_REF_UNREF

If this variable is set in the environment (`getenv()` returns non-NULL), `osg::Referenced` uses a mutex to ensure that `ref()` and `unref()` calls are thread-safe.

3.2.50 OSG_THREADING

Specifies the threaded rendering mechanism used by `osgViewer`.

Valid values:

- `SingleThreaded`: All traversals are performed sequentially from a single thread.
- `CullDrawThreadPerContext`: `osgViewer` uses a thread for each `GraphicsContext` to perform the cull and draw traversals.
- `DrawThreadPerContext`: `osgViewer` uses a thread for each `GraphicsContext` to perform the draw traversal.
- `CullThreadPerCameraDrawThreadPerContext`: `osgViewer` uses a thread for each `Camera` to execute the cull traversal, and a thread for each `GraphicsContext` to execute the draw traversal.

Default value: `osgViewer` determines an appropriate threading mechanism at run-time, based on your hardware configuration (number of CPUs and number of GPUs).

3.2.51 OSG_TXP_DEFAULT_MAX_ANISOTROPY

Specifies the maximum anisotropy value on `Texture2D` objects created by the TXP plugin. (Passes the specified value as a parameter to `Texture2D::setMaxAnisotropy()`).

This variable is used only when loading `.txp` files.

Valid values: Any floating point number.

Default value: ***1.0***

3.2.52 OSG_WINDOW

Specifies a string containing the space-separated x, y, width, and height of `Viewer` object windows, and causes `Viewer` to create such a window in the `realize()` call. It's equivalent to calling `Viewer::setUpViewInWindow( x, y, width, height )`. When set with positive width or height, `OSG_SCREEN` is ignored.

See also `OSG_SCREEN`, `OSG_CONFIG_FILE`.

Valid values: Four space-separated integers representing x, y, width, and height. Width or height must be positive.

3.2.53 OSG_WRITE_OUT_DEFAULT_VALUES

This environment variable is reserved for future use.

3.2.54 OSGHANGGLIDE_REVERSE_CONTROLS

If this variable is set in the environment (`getenv()` returns non-NULL), the `osghangglide` example program inverts its steering controls.

3.2.55 OUTPUT_THREADLIB_SCHEDULING_INFO

If this variable is set in the environment (`getenv()` returns non-NULL), each `Thread` in the `OpenThreads` library writes scheduling-related information to `std::out`.

3.2.56 PATH

Specifies additional directories for the dynamic library search path on WIndows/CygWin platforms. See `OSG_FILE_PATH` for more info.

3.2.57 ProgramFiles

Used by the ZIP plugin on Windows/CygWin platforms to attempt to locate the `winrar` executable. If not found, the plugin falls back to the `unzip` third party dependency.

3.2.58 TEMP

Used by the TGZ and ZIP plugins on Windows/CygWin platforms for temporary file storage.

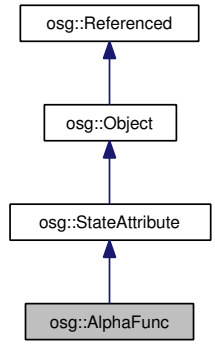
3.2.59 windir

Used by the `osgText::Font` object on Windows platforms as the parent of the `fonts` directory. `osgText` searches this directory for font files on Windows platforms.

Chapter 4

osg Documentation

4.1 osg::AlphaFunc Class Reference



Public Types

- enum **ComparisonFunction** {
  NEVER,
  LESS,
  EQUAL,
  LEQUAL,
  GREATER,
  NOTEQUAL,
  GEQUAL,
  ALWAYS }

Public Member Functions

- **AlphaFunc** (ComparisonFunction func, float ref)
- AlphaFunc (const AlphaFunc &af, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, AlphaFunc, ALPHAFUNC)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFunction** (ComparisonFunction func, float ref)
- void **setFunction** (ComparisonFunction func)
- ComparisonFunction **getFunction** () const
- void **setReferenceValue** (float value)
- float **getReferenceValue** () const
- virtual void apply (State &state) const

4.2 Detailed Description

Encapsulates OpenGL glAlphaFunc.

4.3 Constructor & Destructor Documentation

osg::AlphaFunc::AlphaFunc (const AlphaFunc & *af*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.4 Member Function Documentation

virtual int osg::AlphaFunc::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::AlphaFunc::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

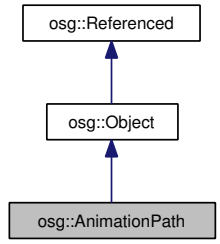
Reimplemented from osg::StateAttribute.

virtual void osg::AlphaFunc::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.5 osg::AnimationPath Class Reference



Public Types

- enum **LoopMode** {
  SWING,
  LOOP,
  NO_LOOPING }
- typedef std::map< double, ControlPoint > **TimeControlPointMap**

Public Member Functions

- **AnimationPath** (const AnimationPath &ap, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_Object** (osg, AnimationPath)
- bool getMatrix (double time, Matrixf &matrix) const
- bool getMatrix (double time, Matrixd &matrix) const
- bool getInverse (double time, Matrixf &matrix) const
- bool **getInverse** (double time, Matrixd &matrix) const
- virtual bool getInterpolatedControlPoint (double time, ControlPoint &controlPoint) const
- void **insert** (double time, const ControlPoint &controlPoint)
- double **getFirstTime** () const
- double **getLastTime** () const
- double **getPeriod** () const
- void **setLoopMode** (LoopMode lm)

- LoopMode **getLoopMode** () const
- void **setTimeControlPointMap** (TimeControlPointMap &tcpm)
- TimeControlPointMap & **getTimeControlPointMap** ()
- const TimeControlPointMap & **getTimeControlPointMap** () const
- bool **empty** () const
- void **clear** ()
- void read (std::istream &in)
- void write (std::ostream &out) const

Classes

- class **ControlPoint**

4.6 Detailed Description

AnimationPath encapsulates a time varying transformation pathway. Can be used for updating camera position and model object position. AnimationPathCallback can be attached directly to Transform nodes to move subgraphs around the scene.

4.7 Member Function Documentation

bool osg::AnimationPath::getMatrix (double *time*, Matrixf & *matrix*) const `[inline]`

Given a specific time, return the transformation matrix for a point.

bool osg::AnimationPath::getMatrix (double *time*, Matrixd & *matrix*) const `[inline]`

Given a specific time, return the transformation matrix for a point..

bool osg::AnimationPath::getInverse (double *time*, Matrixf & *matrix*) const `[inline]`

Given a specific time, return the inverse transformation matrix for a point.

virtual bool osg::AnimationPath::getInterpolatedControlPoint (double *time*, ControlPoint & *controlPoint*) const `[virtual]`

Given a specific time, return the local ControlPoint frame for a point.

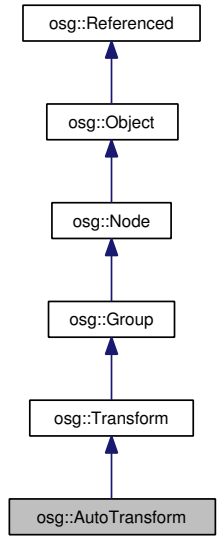
void osg::AnimationPath::read (std::istream & *in*)

Read the animation path from a flat ASCII file stream.

void osg::AnimationPath::write (std::ostream & *out*) const

Write the animation path to a flat ASCII file stream.

4.8 osg::AutoTransform Class Reference



Public Types

- enum **AutoRotateMode** {
  NO_ROTATION,
  ROTATE_TO_SCREEN,
  ROTATE_TO_CAMERA }

Public Member Functions

- **AutoTransform** (const AutoTransform &pat, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &copyop) const
- virtual bool isSameKindAs (const osg::Object *obj) const
- virtual const char * className () const
- virtual const char * libraryName () const
- virtual void accept (NodeVisitor &nv)

- virtual AutoTransform * **asAutoTransform** ()
- virtual const AutoTransform * **asAutoTransform** () const
- void **setPosition** (const Vec3 &pos)
- const Vec3 & **getPosition** () const
- void **setRotation** (const Quat &quat)
- const Quat & **getRotation** () const
- void **setScale** (float scale)
- void **setScale** (const Vec3 &scale)
- const Vec3 & **getScale** () const
- void **setPivotPoint** (const Vec3 &pivot)
- const Vec3 & **getPivotPoint** () const
- void **setAutoUpdateEyeMovementTolerance** (float tolerance)
- float **getAutoUpdateEyeMovementTolerance** () const
- void **setAutoRotateMode** (AutoRotateMode mode)
- AutoRotateMode **getAutoRotateMode** () const
- void **setAutoScaleToScreen** (bool autoScaleToScreen)
- bool **getAutoScaleToScreen** () const
- virtual bool **computeLocalToWorldMatrix** (Matrix &matrix, NodeVisitor *nv) const
- virtual bool **computeWorldToLocalMatrix** (Matrix &matrix, NodeVisitor *nv) const
- virtual BoundingSphere computeBound () const

4.9 Detailed Description

AutoTransform is a derived form of Transform that automatically scales or rotates to keep its children aligned with screen coordinates.

4.10 Member Function Documentation

virtual osg::Object* osg::AutoTransform::cloneType () const `[inline, virtual]`

clone an object of the same type as the node.

Reimplemented from osg::Node.

virtual osg::Object* osg::AutoTransform::clone (const osg::CopyOp & *copyop*) const `[inline, virtual]`

return a clone of a node, with Object* return type.

Reimplemented from osg::Node.

virtual bool osg::AutoTransform::isSameKindAs (const osg::Object * *obj*) const `[inline, virtual]`

return true if this and obj are of the same kind of object.

Reimplemented from osg::Node.

virtual const char* osg::AutoTransform::className () const `[inline, virtual]`

return the name of the node's class type.

Reimplemented from osg::Node.

virtual const char* osg::AutoTransform::libraryName () const `[inline, virtual]`

return the name of the node's library.

Reimplemented from osg::Node.

virtual void osg::AutoTransform::accept (NodeVisitor & *nv*) `[virtual]`

Visitor Pattern : calls the apply method of a NodeVisitor with this node's type.

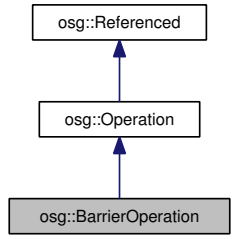
Reimplemented from osg::Node.

virtual BoundingSphere osg::AutoTransform::computeBound () const `[virtual]`

Overrides Group's computeBound. There is no need to override in subclasses from osg::Transform since this computeBound() uses the underlying matrix (calling computeMatrix if required).

Reimplemented from osg::Transform.

4.11 osg::BarrierOperation Struct Reference



Public Types

- enum **PreBlockOp** {

  NO_OPERATION,

  GL_FLUSH,

  GL_FINISH }

Public Member Functions

- **BarrierOperation** (int numThreads, PreBlockOp op=NO_OPERATION)
- virtual void release ()
- virtual void operator() (Object *object)

Public Attributes

- PreBlockOp **_preBlockOp**

4.12 Detailed Description

BarrierOperation allows one to syncronize multiple GraphicsThreads with each other.

4.13 Member Function Documentation

virtual void osg::BarrierOperation::release () `[virtual]`

if this operation is a barrier then release it.

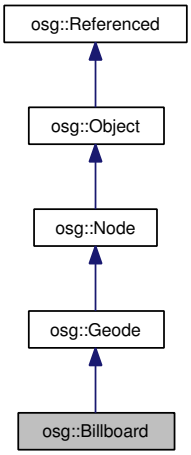
Reimplemented from osg::Operation.

virtual void osg::BarrierOperation::operator() (Object *) `[virtual]`

Do the actual task of this operation.

Implements osg::Operation.

4.14 osg::Billboard Class Reference



Public Types

- enum **Mode** {
  POINT_ROT_EYE,
  POINT_ROT_WORLD,
  AXIAL_ROT }
- typedef std::vector< Vec3 > PositionList

Public Member Functions

- Billboard (const Billboard &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Billboard)
- void setMode (Mode mode)
- Mode getMode () const
- void setAxis (const Vec3 &axis)
- const Vec3 & getAxis () const
- void setNormal (const Vec3 &normal)
- const Vec3 & getNormal () const
- void setPosition (unsigned int i, const Vec3 &pos)
- const Vec3 & getPosition (unsigned int i) const
- void setPositionList (PositionList &pl)
- PositionList & getPositionList ()
- const PositionList & getPositionList () const
- virtual bool addDrawable (Drawable *gset)

- virtual bool addDrawable (Drawable *gset, const Vec3 &pos)
- virtual bool removeDrawable (Drawable *gset)
- bool **computeMatrix** (Matrix &modelview, const Vec3 &eye_local, const Vec3 &pos_local) const
- virtual BoundingSphere computeBound () const

4.15 Detailed Description

Billboard is a derived form of Geode that orients its osg::Drawable children to face the eye point. Typical uses incllude trees and particle explosions,

4.16 Member Typedef Documentation

typedef std::vector<Vec3> osg::Billboard::PositionList

Type definition for pivot point position list.

4.17 Constructor & Destructor Documentation

osg::Billboard::Billboard (const Billboard &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)**

Copy constructor using CopyOp to manage deep vs shallow copy.

4.18 Member Function Documentation

void osg::Billboard::setMode (Mode *mode*)

Set the billboard rotation mode.

Mode osg::Billboard::getMode () const `[inline]`

Get the billboard rotation mode.

void osg::Billboard::setAxis (const Vec3 & *axis*)

Set the rotation axis for the billboard's child Drawables. Only utlized when mode==AXIAL_ROT.

const Vec3& osg::Billboard::getAxis () const `[inline]`

Get the rotation axis.

void osg::Billboard::setNormal (const Vec3 & *normal*)

This normal defines child Drawables' front face direction when unrotated.

const Vec3& osg::Billboard::getNormal () const `[inline]`

Get the front face direction normal.

void osg::Billboard::setPosition (unsigned int *i*, const Vec3 & *pos*) `[inline]`

Set the specified child Drawable's position.

const Vec3& osg::Billboard::getPosition (unsigned int *i*) const `[inline]`

Get the specified child Drawable's position.

void osg::Billboard::setPositionList (PositionList & *pl*) `[inline]`

Set the list of pivot point positions.

PositionList& osg::Billboard::getPositionList () `[inline]`

Get the list of pivot point positions.

const PositionList& osg::Billboard::getPositionList () const `[inline]`

Get a const list of pivot point positions.

virtual bool osg::Billboard::addDrawable (Drawable * *gset*) `[virtual]`

Add a Drawable with a default position of Vec3(0,0,0). Call the base-class Geode::addDrawble() to add the given Drawable gset as a child. If Geode::addDrawable() returns true, add the default position to the pivot point position list and return true. Otherwise, return false.

Reimplemented from osg::Geode.

virtual bool osg::Billboard::addDrawable (Drawable * *gset*, const Vec3 & *pos*) `[virtual]`

Add a Drawable with a specified position. Call the base-class Geode::addDrawble() to add the given Drawable gset as a child. If Geode::addDrawable() returns true, add the given position pos to the pivot point position list and return true. Otherwise, return false.

virtual bool osg::Billboard::removeDrawable (Drawable * *gset*) `[virtual]`

Remove a Drawable and its associated position. If gset is a child, remove it, decrement its reference count, remove its pivot point position. and return true. Otherwise, return false.

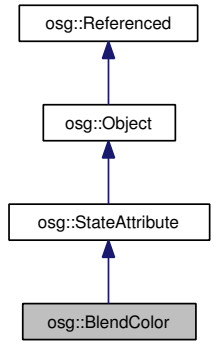
Reimplemented from osg::Geode.

virtual BoundingSphere osg::Billboard::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Geode.

4.19 osg::BlendColor Class Reference



Public Member Functions

- **BlendColor** (const osg::Vec4 &constantColor)
- BlendColor (const BlendColor &trans, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, BlendColor, BLENDCOLOR)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setConstantColor** (const osg::Vec4 &color)
- osg::Vec4 & **getConstantColor** ()
- const osg::Vec4 & **getConstantColor** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.20 Detailed Description

Encapsulates OpenGL blend/transparency state.

4.21 Constructor & Destructor Documentation

osg::BlendColor::BlendColor (const BlendColor & *trans*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.22 Member Function Documentation

virtual int osg::BlendColor::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::BlendColor::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::BlendColor::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

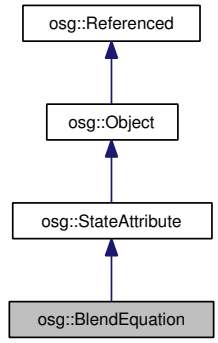
static Extensions* osg::BlendColor::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Returns the Extensions object for the given context. If createIfNotInitalized is true and the Exentsions object doesn't exist, getExtensions() creates it on the given context. Returns NULL if createIfNotInitalized is false and the Extensions object doesn't exist.

static void osg::BlendColor::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions() allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes, but need to ensure that they all use the same low common denominator extensions.

4.23 osg::BlendEquation Class Reference



Public Types

- enum **Equation** {
  RGBA_MIN,
  RGBA_MAX,
  ALPHA_MIN,
  ALPHA_MAX,
  LOGIC_OP,
  FUNC_ADD,
  FUNC_SUBTRACT,
  FUNC_REVERSE_SUBTRACT }

Public Member Functions

- **BlendEquation** (Equation equation)
- BlendEquation (const BlendEquation &trans, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, BlendEquation, BLENDEQUATION)

- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setEquation** (Equation equation)
- Equation **getEquation** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.24 Detailed Description

Encapsulates OpenGL BlendEquation state.

4.25 Constructor & Destructor Documentation

osg::BlendEquation::BlendEquation (const BlendEquation & *trans*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.26 Member Function Documentation

virtual int osg::BlendEquation::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::BlendEquation::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::BlendEquation::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

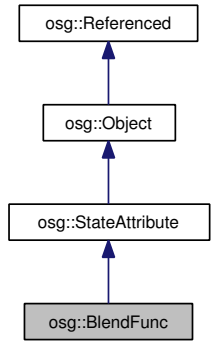
static Extensions* osg::BlendEquation::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Returns the Extensions object for the given context. If createIfNotInitalized is true and the Exentsions object doesn't exist, getExtensions() creates it on the given context. Returns NULL if createIfNotInitalized is false and the Extensions object doesn't exist.

static void osg::BlendEquation::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions() allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes, but need to ensure that they all use the same low common denominator extensions.

4.27 osg::BlendFunc Class Reference



Public Types

- enum **BlendFuncMode** {
  DST_ALPHA,
  DST_COLOR,

ONE,

ONE_MINUS_DST_ALPHA,

ONE_MINUS_DST_COLOR,

ONE_MINUS_SRC_ALPHA,

ONE_MINUS_SRC_COLOR,

SRC_ALPHA,

SRC_ALPHA_SATURATE,

SRC_COLOR,

CONSTANT_COLOR,

ONE_MINUS_CONSTANT_COLOR,

CONSTANT_ALPHA,

ONE_MINUS_CONSTANT_ALPHA,

ZERO }

Public Member Functions

- **BlendFunc** (GLenum source, GLenum destination)
- **BlendFunc** (GLenum source, GLenum destination, GLenum source_alpha, GLenum destination_alpha)
- BlendFunc (const BlendFunc &trans, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, BlendFunc, BLENDFUNC)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFunction** (GLenum source, GLenum destination)
- void **setFunction** (GLenum source_rgb, GLenum destination_rgb, GLenum source_alpha, GLenum destination_alpha)
- void **setSource** (GLenum source)
- GLenum **getSource** () const
- void **setSourceRGB** (GLenum source)
- GLenum **getSourceRGB** () const
- void **setSourceAlpha** (GLenum source)
- GLenum **getSourceAlpha** () const
- void **setDestination** (GLenum destination)
- GLenum **getDestination** () const
- void **setDestinationRGB** (GLenum destination)
- GLenum **getDestinationRGB** () const
- void **setDestinationAlpha** (GLenum destination)
- GLenum **getDestinationAlpha** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.28 Detailed Description

Encapsulates OpenGL blend/transparency state.

4.29 Constructor & Destructor Documentation

osg::BlendFunc::BlendFunc (const BlendFunc & *trans*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.30 Member Function Documentation

virtual int osg::BlendFunc::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::BlendFunc::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::BlendFunc::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

static Extensions* osg::BlendFunc::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Returns the Extensions object for the given context. If createIfNotInitalized is true and the Exentsions object doesn't exist, getExtensions() creates it on the given context. Returns NULL if createIfNotInitalized is false and the Extensions object doesn't exist.

static void osg::BlendFunc::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions() allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes, but need to ensure that they all use the same low common denominator extensions.

4.31 osg::BoundingBox Class Reference

Public Member Functions

- BoundingBox ()
- BoundingBox (float xmin, float ymin, float zmin, float xmax, float ymax, float zmax)
- BoundingBox (const Vec3 &min, const Vec3 &max)
- void init ()
- bool valid () const
- void set (float xmin, float ymin, float zmin, float xmax, float ymax, float zmax)
- void set (const Vec3 &min, const Vec3 &max)
- float & **xMin** ()
- float **xMin** () const
- float & **yMin** ()
- float **yMin** () const
- float & **zMin** ()
- float **zMin** () const
- float & **xMax** ()
- float **xMax** () const
- float & **yMax** ()
- float **yMax** () const
- float & **zMax** ()
- float **zMax** () const
- const Vec3 center () const
- float radius () const
- float radius2 () const
- const Vec3 corner (unsigned int pos) const
- void expandBy (const Vec3 &v)
- void expandBy (float x, float y, float z)
- void expandBy (const BoundingBox &bb)

- void expandBy (const BoundingSphere &sh)
- BoundingBox intersect (const BoundingBox &bb) const
- bool intersects (const BoundingBox &bb) const
- bool contains (const Vec3 &v) const

Public Attributes

- Vec3 _min
- Vec3 _max

4.32 Detailed Description

General purpose axis-aligned bounding box class for enclosing objects/vertices. Bounds leaf objects in a scene such as osg::Drawable objects. Used for frustum culling etc.

4.33 Constructor & Destructor Documentation

osg::BoundingBox::BoundingBox () `[inline]`

Creates an uninitialized bounding box.

osg::BoundingBox::BoundingBox (float *xmin*, float *ymin*, float *zmin*, float *xmax*, float *ymax*, float *zmax*) `[inline]`

Creates a bounding box initialized to the given extents.

osg::BoundingBox::BoundingBox (const Vec3 & *min*, const Vec3 & *max*) `[inline]`

Creates a bounding box initialized to the given extents.

4.34 Member Function Documentation

void osg::BoundingBox::init () `[inline]`

Clear the bounding box. Erases existing minimum and maximum extents.

bool osg::BoundingBox::valid () const `[inline]`

Returns true if the bounding box extents are valid, false otherwise.

void osg::BoundingBox::set (float *xmin*, float *ymin*, float *zmin*, float *xmax*, float *ymax*, float *zmax*) `[inline]`

Sets the bounding box extents.

void osg::BoundingBox::set (const Vec3 & *min*, const Vec3 & *max*) `[inline]`

Sets the bounding box extents.

const Vec3 osg::BoundingBox::center () const `[inline]`

Calculates and returns the bounding box center.

float osg::BoundingBox::radius () const `[inline]`

Calculates and returns the bounding box radius.

float osg::BoundingBox::radius2 () const `[inline]`

Calculates and returns the squared length of the bounding box radius. Note, radius2() is faster to calculate than radius().

const Vec3 osg::BoundingBox::corner (unsigned int *pos*) const `[inline]`

Returns a specific corner of the bounding box. pos specifies the corner as a number between 0 and 7. Each bit selects an axis, X, Y, or Z from least- to most-significant. Unset bits select the minimum value for that axis, and set bits select the maximum.

void osg::BoundingBox::expandBy (const Vec3 & *v*) `[inline]`

Expands the bounding box to include the given coordinate. If the box is uninitialized, set its min and max extents to v.

void osg::BoundingBox::expandBy (float *x*, float *y*, float *z*) `[inline]`

Expands the bounding box to include the given coordinate. If the box is uninitialized, set its min and max extents to Vec3(x,y,z).

void osg::BoundingBox::expandBy (const BoundingBox & *bb*)

Expands this bounding box to include the given bounding box. If this box is uninitialized, set it equal to bb.

void osg::BoundingBox::expandBy (const BoundingSphere & *sh*)

Expands this bounding box to include the given sphere. If this box is uninitialized, set it to include sh.

BoundingBox osg::BoundingBox::intersect (const BoundingBox & *bb*) const `[inline]`

Returns the intesection of this bounding box and the specified bounding box.

bool osg::BoundingBox::intersects (const BoundingBox & *bb*) const `[inline]`

Return true if this bounding box intersects the specified bounding box.

bool osg::BoundingBox::contains (const Vec3 & *v*) const `[inline]`

Returns true if this bounding box contains the specified coordinate.

4.35 Member Data Documentation

Vec3 osg::BoundingBox::_min

Minimum extent. (Smallest X, Y, and Z values of all coordinates.)

Vec3 osg::BoundingBox::_max

Maximum extent. (Greatest X, Y, and Z values of all coordinates.)

4.36 osg::BoundingSphere Class Reference

Public Member Functions

- BoundingSphere ()
- BoundingSphere (const Vec3 &center, float radius)
- BoundingSphere (const BoundingSphere &bs)
- BoundingSphere (const BoundingBox &bb)
- void init ()
- bool valid () const
- void set (const Vec3 &center, float radius)
- Vec3 & center ()
- const Vec3 & center () const
- float & radius ()
- float radius () const
- float radius2 () const

- void expandBy (const Vec3 &v)
- void expandRadiusBy (const Vec3 &v)
- void expandBy (const BoundingSphere &sh)
- void expandRadiusBy (const BoundingSphere &sh)
- void expandBy (const BoundingBox &bb)
- void expandRadiusBy (const BoundingBox &bb)
- bool contains (const Vec3 &v) const
- bool intersects (const BoundingSphere &bs) const

Public Attributes

- Vec3 **_center**
- float **_radius**

4.37 Detailed Description

General purpose bounding sphere class for enclosing nodes/objects/vertices. Bounds internal osg::Nodes in the scene, assists in view frustum culling, etc. Similar in function to BoundingBox, it's quicker for evaluating culling but generally will not cull as aggressively because it encloses a greater volume.

4.38 Constructor & Destructor Documentation

osg::BoundingSphere::BoundingSphere () `[inline]`

Construct a default bounding sphere with radius to -1.0f, representing an invalid/unset bounding sphere.

osg::BoundingSphere::BoundingSphere (const Vec3 & *center*, float *radius*) `[inline]`

Creates a bounding sphere initialized to the given extents.

osg::BoundingSphere::BoundingSphere (const BoundingSphere & *bs*) `[inline]`

Creates a bounding sphere initialized to the given extents.

osg::BoundingSphere::BoundingSphere (const BoundingBox & *bb*) `[inline]`

Creates a bounding sphere initialized to the given extents.

4.39 Member Function Documentation

void osg::BoundingSphere::init () `[inline]`

Clear the bounding sphere. Reset to default values.

bool osg::BoundingSphere::valid () const `[inline]`

Returns true of the bounding sphere extents are valid, false otherwise.

void osg::BoundingSphere::set (const Vec3 & *center*, float *radius*) `[inline]`

Set the bounding sphere to the given center/radius.

Vec3& osg::BoundingSphere::center () `[inline]`

Returns the center of the bounding sphere.

const Vec3& osg::BoundingSphere::center () const `[inline]`

Returns the const center of the bounding sphere.

float& osg::BoundingSphere::radius () `[inline]`

Returns the radius of the bounding sphere.

float osg::BoundingSphere::radius () const `[inline]`

Returns the const radius of the bounding sphere.

float osg::BoundingSphere::radius2 () const `[inline]`

Returns the squared length of the radius. Note, For performance reasons, the calling method is responsible for checking to make sure the sphere is valid.

void osg::BoundingSphere::expandBy (const Vec3 & *v*)

Expands the sphere to encompass the given point. Repositions the sphere center to minimize the radius increase. If the sphere is uninitialized, set its center to v and radius to zero.

void osg::BoundingSphere::expandRadiusBy (const Vec3 & *v*)

Expands the sphere to encompass the given point. Does not reposition the sphere center. If the sphere is uninitialized, set its center to v and radius to zero.

void osg::BoundingSphere::expandBy (const BoundingSphere & *sh*)

Expands the sphere to encompass the given sphere. Repositions the sphere center to minimize the radius increase. If the sphere is uninitialized, set its center and radius to match sh.

void osg::BoundingSphere::expandRadiusBy (const BoundingSphere & *sh*)

Expands the sphere to encompass the given sphere. Does not repositions the sphere center. If the sphere is uninitialized, set its center and radius to match sh.

void osg::BoundingSphere::expandBy (const BoundingBox & *bb*)

Expands the sphere to encompass the given box. Repositions the sphere center to minimize the radius increase.

void osg::BoundingSphere::expandRadiusBy (const BoundingBox & *bb*)

Expands the sphere to encompass the given box. Does not repositions the sphere center.

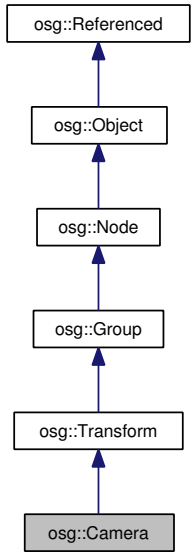
bool osg::BoundingSphere::contains (const Vec3 & *v*) const `[inline]`

Returns true if v is within the sphere.

bool osg::BoundingSphere::intersects (const BoundingSphere & *bs*) const `[inline]`

Returns true if there is a non-empty intersection with the given bounding sphere.

4.40 osg::Camera Class Reference



Public Types

- enum **TransformOrder** {
  PRE_MULTIPLY,
  POST_MULTIPLY }
- enum ProjectionResizePolicy {
  FIXED,
  HORIZONTAL,
  VERTICAL }
- enum **RenderOrder** {
  PRE_RENDER,
  NESTED_RENDER,
  POST_RENDER }
- enum **RenderTargetImplementation** {
  FRAME_BUFFER_OBJECT,
  PIXEL_BUFFER_RTT,
  PIXEL_BUFFER,
  FRAME_BUFFER,
  SEPERATE_WINDOW }

- enum **BufferComponent** {
  DEPTH_BUFFER,
  STENCIL_BUFFER,
  COLOR_BUFFER,
  COLOR_BUFFER0,
  COLOR_BUFFER1,
  COLOR_BUFFER2,
  COLOR_BUFFER3,
  COLOR_BUFFER4,
  COLOR_BUFFER5,
  COLOR_BUFFER6,
  COLOR_BUFFER7 }
- typedef std::map< BufferComponent, Attachment > **BufferAttachmentMap**

Public Member Functions

- Camera (const Camera &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Camera)
- void setView (View *view)
- View * getView ()
- const View * getView () const
- void setStats (osg::Stats *stats)
- osg::Stats * getStats ()
- const osg::Stats * getStats () const
- void setAllowEventFocus (bool focus)
- bool getAllowEventFocus () const
- void setDisplaySettings (osg::DisplaySettings *ds)
- osg::DisplaySettings * getDisplaySettings ()
- const osg::DisplaySettings * getDisplaySettings () const
- void setClearColor (const Vec4 &color)
- const Vec4 & getClearColor () const
- void setClearMask (GLbitfield mask)
- GLbitfield getClearMask () const
- void setColorMask (osg::ColorMask *colorMask)
- void setColorMask (bool red, bool green, bool blue, bool alpha)
- const ColorMask * getColorMask () const
- ColorMask * getColorMask ()
- void setViewport (osg::Viewport *viewport)
- void setViewport (int x, int y, int width, int height)
- const Viewport * getViewport () const
- Viewport * getViewport ()
- void setTransformOrder (TransformOrder order)

- TransformOrder getTransformOrder () const
- void setProjectionResizePolicy (ProjectionResizePolicy policy)
- ProjectionResizePolicy getProjectionResizePolicy () const
- void setProjectionMatrix (const osg::Matrixf &matrix)
- void setProjectionMatrix (const osg::Matrixd &matrix)
- void setProjectionMatrixAsOrtho (double left, double right, double bottom, double top, double zNear, double zFar)
- void setProjectionMatrixAsOrtho2D (double left, double right, double bottom, double top)
- void setProjectionMatrixAsFrustum (double left, double right, double bottom, double top, double zNear, double zFar)
- void setProjectionMatrixAsPerspective (double fovy, double aspectRatio, double zNear, double zFar)
- osg::Matrixd & getProjectionMatrix ()
- const osg::Matrixd & getProjectionMatrix () const
- bool getProjectionMatrixAsOrtho (double &left, double &right, double &bottom, double &top, double &zNear, double &zFar)
- bool getProjectionMatrixAsFrustum (double &left, double &right, double &bottom, double &top, double &zNear, double &zFar)
- bool getProjectionMatrixAsPerspective (double &fovy, double &aspectRatio, double &zNear, double &zFar)
- void setViewMatrix (const osg::Matrixf &matrix)
- void setViewMatrix (const osg::Matrixd &matrix)
- void setViewMatrixAsLookAt (const osg::Vec3 &eye, const osg::Vec3 &center, const osg::Vec3 &up)
- osg::Matrixd & getViewMatrix ()
- const osg::Matrixd & getViewMatrix () const
- void getViewMatrixAsLookAt (osg::Vec3 &eye, osg::Vec3 &center, osg::Vec3 &up, float lookDistance=1.0f)
- Matrixd getInverseViewMatrix () const
- void setRenderOrder (RenderOrder order, int orderNum=0)
- RenderOrder getRenderOrder () const
- int getRenderOrderNum () const
- bool isRenderToTextureCamera () const
- void setRenderTargetImplementation (RenderTargetImplementation impl)
- void setRenderTargetImplementation (RenderTargetImplementation impl, RenderTargetImplementation fallback)
- RenderTargetImplementation getRenderTargetImplementation () const
- RenderTargetImplementation getRenderTargetFallback () const
- void setDrawBuffer (GLenum buffer)
- GLenum getDrawBuffer () const
- void setReadBuffer (GLenum buffer)
- GLenum getReadBuffer () const
- void **attach** (BufferComponent buffer, GLenum internalFormat)
- void **attach** (BufferComponent buffer, osg::Texture *texture, unsigned int level=0, unsigned int face=0, bool mipMapGeneration=false)
- void **attach** (BufferComponent buffer, osg::Image *image)
- void **detach** (BufferComponent buffer)
- BufferAttachmentMap & getBufferAttachmentMap ()

- const BufferAttachmentMap & getBufferAttachmentMap () const
- void createCameraThread ()
- void setCameraThread (OperationThread *gt)
- OperationThread * getCameraThread ()
- const OperationThread * getCameraThread () const
- void setGraphicsContext (GraphicsContext *context)
- GraphicsContext * getGraphicsContext ()
- const GraphicsContext * getGraphicsContext () const
- void setRenderer (osg::GraphicsOperation *rc)
- osg::GraphicsOperation * getRenderer ()
- const osg::GraphicsOperation * getRenderer () const
- void setRenderingCache (osg::Object *rc)
- osg::Object * getRenderingCache ()
- const osg::Object * getRenderingCache () const
- void setPreDrawCallback (DrawCallback *cb)
- DrawCallback * getPreDrawCallback ()
- const DrawCallback * getPreDrawCallback () const
- void setPostDrawCallback (DrawCallback *cb)
- DrawCallback * getPostDrawCallback ()
- const DrawCallback * getPostDrawCallback () const
- OpenThreads::Mutex * **getDataChangeMutex** () const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (osg::State *=0) const
- virtual bool computeLocalToWorldMatrix (Matrix &matrix, NodeVisitor *) const
- virtual bool computeWorldToLocalMatrix (Matrix &matrix, NodeVisitor *) const

Classes

- struct **Attachment**
- struct DrawCallback

4.41 Detailed Description

Camera - is a subclass of Transform which represents encapsulates the settings of a Camera.

4.42 Member Enumeration Documentation

enum osg::Camera::ProjectionResizePolicy

Enumerator:

HORIZONTAL Keep the projection matrix fixed, despite window resizes.

VERTICAL Adjust the HORIZOTNAL field of view on window resizes. Adjust the VERTICAL field of view on window resizes.

4.43 Constructor & Destructor Documentation

osg::Camera::Camera (const Camera &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.44 Member Function Documentation

void osg::Camera::setView (View * *view*) `[inline]`

Set the View that this Camera is part of.

View* osg::Camera::getView () `[inline]`

Get the View that this Camera is part of.

const View* osg::Camera::getView () const `[inline]`

Get the const View that this Camera is part of.

void osg::Camera::setStats (osg::Stats * *stats*) `[inline]`

Set the Stats object used for collect various frame related timing and scene graph stats.

osg::Stats* osg::Camera::getStats () `[inline]`

Get the Stats object.

const osg::Stats* osg::Camera::getStats () const `[inline]`

Get the const Stats object.

void osg::Camera::setAllowEventFocus (bool *focus*) `[inline]`

Set whether this camera allows events to be generated by the associated graphics window to be associated with this camera.

bool osg::Camera::getAllowEventFocus () const `[inline]`

Get whether this camera allows events to be generated by the associated graphics window to be associated with this camera.

void osg::Camera::setDisplaySettings (osg::DisplaySettings * *ds*) `[inline]`

Set the DsplaySettings object associated with this view.

osg::DisplaySettings* osg::Camera::getDisplaySettings () `[inline]`

Set the DsplaySettings object associated with this view.

const osg::DisplaySettings* osg::Camera::getDisplaySettings () const `[inline]`

Set the DsplaySettings object associated with this view.

void osg::Camera::setClearColor (const Vec4 & *color*) `[inline]`

Sets the clear color.

const Vec4& osg::Camera::getClearColor () const `[inline]`

Returns the clear color.

void osg::Camera::setClearMask (GLbitfield *mask*) `[inline]`

Set the clear mask used in glClear(..). Defaults to GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_-BIT.

GLbitfield osg::Camera::getClearMask () const `[inline]`

Get the clear mask.

void osg::Camera::setColorMask (osg::ColorMask * *colorMask*)

Set the color mask of the camera to use specified osg::ColorMask.

void osg::Camera::setColorMask (bool *red*, bool *green*, bool *blue*, bool *alpha*)

Set the color mask of the camera to specified values.

const ColorMask* osg::Camera::getColorMask () const `[inline]`

Get the const ColorMask.

ColorMask* osg::Camera::getColorMask () `[inline]`

Get the ColorMask.

void osg::Camera::setViewport (osg::Viewport * *viewport*)

Set the viewport of the camera to use specified osg::Viewport.

void osg::Camera::setViewport (int *x*, int *y*, int *width*, int *height*)

Set the viewport of the camera to specified dimensions.

const Viewport* osg::Camera::getViewport () const `[inline]`

Get the const viewport.

Viewport* osg::Camera::getViewport () `[inline]`

Get the viewport.

void osg::Camera::setTransformOrder (TransformOrder *order*) `[inline]`

Set the transformation order for world-to-local and local-to-world transformation.

TransformOrder osg::Camera::getTransformOrder () const `[inline]`

Get the transformation order.

void osg::Camera::setProjectionResizePolicy (ProjectionResizePolicy *policy*) `[inline]`

Set the policy used to determin if and how the projection matrix should be adjusted on window resizes.

ProjectionResizePolicy osg::Camera::getProjectionResizePolicy () const `[inline]`

Get the policy used to determin if and how the projection matrix should be adjusted on window resizes.

void osg::Camera::setProjectionMatrix (const osg::Matrixf & *matrix*) `[inline]`

Set the projection matrix. Can be thought of as setting the lens of a camera.

void osg::Camera::setProjectionMatrix (const osg::Matrixd & *matrix*) `[inline]`

Set the projection matrix. Can be thought of as setting the lens of a camera.

void osg::Camera::setProjectionMatrixAsOrtho (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*)

Set to an orthographic projection. See OpenGL glOrtho for documentation further details.

void osg::Camera::setProjectionMatrixAsOrtho2D (double *left*, double *right*, double *bottom*, double *top*)

Set to a 2D orthographic projection. See OpenGL glOrtho2D documentation for further details.

void osg::Camera::setProjectionMatrixAsFrustum (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*)

Set to a perspective projection. See OpenGL glFrustum documentation for further details.

void osg::Camera::setProjectionMatrixAsPerspective (double *fovy*, double *aspectRatio*, double *zNear*, double *zFar*)

Create a symmetrical perspective projection, See OpenGL gluPerspective documentation for further details. Aspect ratio is defined as width/height.

osg::Matrixd& osg::Camera::getProjectionMatrix () `[inline]`

Get the projection matrix.

const osg::Matrixd& osg::Camera::getProjectionMatrix () const `[inline]`

Get the const projection matrix.

bool osg::Camera::getProjectionMatrixAsOrtho (double & *left*, double & *right*, double & *bottom*, double & *top*, double & *zNear*, double & *zFar*)

Get the othographic settings of the orthographic projection matrix. Returns false if matrix is not an orthographic matrix, where parameter values are undefined.

bool osg::Camera::getProjectionMatrixAsFrustum (double & *left*, double & *right*, double & *bottom*, double & *top*, double & *zNear*, double & *zFar*)

Get the frustum setting of a perspective projection matrix. Returns false if matrix is not a perspective matrix, where parameter values are undefined.

bool osg::Camera::getProjectionMatrixAsPerspective (double & *fovy*, double & *aspectRatio*, double & *zNear*, double & *zFar*)

Get the frustum setting of a symmetric perspective projection matrix. Returns false if matrix is not a perspective matrix, where parameter values are undefined. Note, if matrix is not a symmetric perspective matrix then the shear will be lost. Asymmetric matrices occur when stereo, power walls, caves and reality center display are used. In these configurations one should use the 'getProjectionMatrixAsFrustum' method instead.

void osg::Camera::setViewMatrix (const osg::Matrixf & *matrix*) `[inline]`

Set the view matrix. Can be thought of as setting the position of the world relative to the camera in camera coordinates.

void osg::Camera::setViewMatrix (const osg::Matrixd & *matrix*) `[inline]`

Set the view matrix. Can be thought of as setting the position of the world relative to the camera in camera coordinates.

void osg::Camera::setViewMatrixAsLookAt (const osg::Vec3 & *eye*, const osg::Vec3 & *center*, const osg::Vec3 & *up*)

Set to the position and orientation of view matrix, using the same convention as gluLookAt.

osg::Matrixd& osg::Camera::getViewMatrix () `[inline]`

Get the view matrix.

const osg::Matrixd& osg::Camera::getViewMatrix () const `[inline]`

Get the const view matrix.

void osg::Camera::getViewMatrixAsLookAt (osg::Vec3 & *eye*, osg::Vec3 & *center*, osg::Vec3 & *up*, float *lookDistance* = `1.0f`)

Get to the position and orientation of a modelview matrix, using the same convention as gluLookAt.

Matrixd osg::Camera::getInverseViewMatrix () const

Get the inverse view matrix.

void osg::Camera::setRenderOrder (RenderOrder *order*, int *orderNum* = `0`) `[inline]`

Set the rendering order of this camera's subgraph relative to any camera that this subgraph is nested within. For rendering to a texture, one typically uses PRE_RENDER. For Head Up Displays, one would typically use POST_RENDER.

RenderOrder osg::Camera::getRenderOrder () const `[inline]`

Get the rendering order of this camera's subgraph relative to any camera that this subgraph is nested within.

int osg::Camera::getRenderOrderNum () const `[inline]`

Get the rendering order number of this camera relative to any sibling cameras in this subgraph.

bool osg::Camera::isRenderToTextureCamera () const

Return true if this Camera is set up as a render to texture camera, i.e. it has textures assigned to it.

void osg::Camera::setRenderTargetImplementation (RenderTargetImplementation *impl*)

Set the render target.

void osg::Camera::setRenderTargetImplementation (RenderTargetImplementation *impl*, RenderTargetImplementation *fallback*)

Set the render target and fall-back that's used if the former isn't available.

RenderTargetImplementation osg::Camera::getRenderTargetImplementation () const `[inline]`

Get the render target.

RenderTargetImplementation osg::Camera::getRenderTargetFallback () const `[inline]`

Get the render target fallback.

void osg::Camera::setDrawBuffer (GLenum *buffer*) `[inline]`

Set the draw buffer used at the start of each frame draw. Note, a buffer value of GL_NONE is used to sepecify that the rendering back-end should choose the most appropriate buffer.

GLenum osg::Camera::getDrawBuffer () const `[inline]`

Get the draw buffer used at the start of each frame draw.

void osg::Camera::setReadBuffer (GLenum *buffer*) `[inline]`

Set the read buffer for any required copy operations to use. Note, a buffer value of GL_NONE is used to sepecify that the rendering back-end should choose the most appropriate buffer.

GLenum osg::Camera::getReadBuffer () const `[inline]`

Get the read buffer for any required copy operations to use.

BufferAttachmentMap& osg::Camera::getBufferAttachmentMap () `[inline]`

Get the BufferAttachmentMap, used to configure frame buffer objects, pbuffers and texture reads.

const BufferAttachmentMap& osg::Camera::getBufferAttachmentMap () const `[inline]`

Get the const BufferAttachmentMap, used to configure frame buffer objects, pbuffers and texture reads.

void osg::Camera::createCameraThread ()

Create a operation thread for this camera.

void osg::Camera::setCameraThread (OperationThread * *gt*)

Assign a operation thread to the camera.

OperationThread* osg::Camera::getCameraThread () `[inline]`

Get the operation thread assigned to this camera.

const OperationThread* osg::Camera::getCameraThread () const `[inline]`

Get the const operation thread assigned to this camera.

void osg::Camera::setGraphicsContext (GraphicsContext * *context*)

Set the GraphicsContext that provides the mechansim for managing the OpenGL graphics context associated with this camera.

GraphicsContext* osg::Camera::getGraphicsContext () `[inline]`

Get the GraphicsContext.

const GraphicsContext* osg::Camera::getGraphicsContext () const `[inline]`

Get the const GraphicsContext.

void osg::Camera::setRenderer (osg::GraphicsOperation * *rc*) `[inline]`

Set the Rendering object that is used to implement rendering of the subgraph.

osg::GraphicsOperation* osg::Camera::getRenderer () `[inline]`

Get the Rendering object that is used to implement rendering of the subgraph.

const osg::GraphicsOperation* osg::Camera::getRenderer () const `[inline]`

Get the const Rendering object that is used to implement rendering of the subgraph.

void osg::Camera::setRenderingCache (osg::Object * *rc*) `[inline]`

Set the Rendering cache that is used for cached objects associated with rendering of subgraphs.

osg::Object* osg::Camera::getRenderingCache () `[inline]`

Get the Rendering cache that is used for cached objects associated with rendering of subgraphs.

const osg::Object* osg::Camera::getRenderingCache () const `[inline]`

Get the const Rendering cache that is used for cached objects associated with rendering of subgraphs.

void osg::Camera::setPreDrawCallback (DrawCallback * *cb*) `[inline]`

Set the pre draw callback for custom operations to be done before the drawing of the camera's subgraph has been completed.

DrawCallback* osg::Camera::getPreDrawCallback () `[inline]`

Get the pre draw callback.

const DrawCallback* osg::Camera::getPreDrawCallback () const `[inline]`

Get the const pre draw callback.

void osg::Camera::setPostDrawCallback (DrawCallback * *cb*) `[inline]`

Set the post draw callback for custom operations to be done after the drawing of the camera's subgraph has been completed.

DrawCallback* osg::Camera::getPostDrawCallback () `[inline]`

Get the post draw callback.

const DrawCallback* osg::Camera::getPostDrawCallback () const `[inline]`

Get the const post draw callback.

virtual void osg::Camera::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Group.

virtual void osg::Camera::releaseGLObjects (osg::State * = 0) const `[virtual]`

If State is non-zero, this function releases any associated OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::Group.

virtual bool osg::Camera::computeLocalToWorldMatrix (Matrix & *matrix*, NodeVisitor *) const `[virtual]`

Transform method that must be defined to provide generic interface for scene graph traversals.

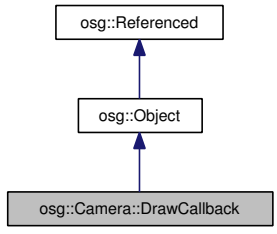
Reimplemented from osg::Transform.

virtual bool osg::Camera::computeWorldToLocalMatrix (Matrix & *matrix*, NodeVisitor *) const `[virtual]`

Transform method that must be defined to provide generic interface for scene graph traversals.

Reimplemented from osg::Transform.

4.45 osg::Camera::DrawCallback Struct Reference



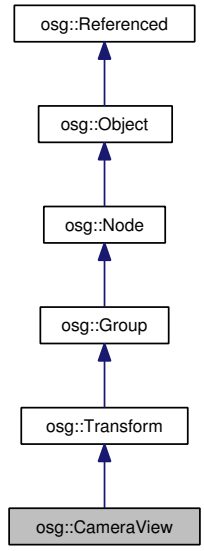
Public Member Functions

- **DrawCallback** (const DrawCallback &, const CopyOp &)
- **META_Object** (osg, DrawCallback) virtual void operator()(const osg

4.46 Detailed Description

Draw callback for custom operations.

4.47 osg::CameraView Class Reference



Public Types

- enum **FieldOfViewMode** {
  UNCONSTRAINED,
  HORIZONTAL,
  VERTICAL }

Public Member Functions

- **CameraView** (const CameraView &pat, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, CameraView)
- void setPosition (const Vec3d &pos)
- const Vec3d & getPosition () const
- void setAttitude (const Quat &quat)
- const Quat & getAttitude () const

- void setFieldOfView (double fieldOfView)
- double getFieldOfView () const
- void setFieldOfViewMode (FieldOfViewMode mode)
- FieldOfViewMode getFieldOfViewMode () const
- void setFocalLength (double focalLength)
- double getFocalLength () const
- virtual bool **computeLocalToWorldMatrix** (Matrix &matrix, NodeVisitor *nv) const
- virtual bool **computeWorldToLocalMatrix** (Matrix &matrix, NodeVisitor *nv) const

4.48 Detailed Description

CameraView - is a Transform that is used to specify camera views from within the scene graph. The application must attach a camera to a CameraView via the NodePath from the top of the scene graph to the CameraView node itself, and accimulate the view matrix from this NodePath.

4.49 Member Function Documentation

void osg::CameraView::setPosition (const Vec3d & *pos*) `[inline]`

Set the position of the camera view.

const Vec3d& osg::CameraView::getPosition () const `[inline]`

Get the position of the camera view.

void osg::CameraView::setAttitude (const Quat & *quat*) `[inline]`

Set the attitide of the camera view.

const Quat& osg::CameraView::getAttitude () const `[inline]`

Get the attitide of the camera view.

void osg::CameraView::setFieldOfView (double *fieldOfView*) `[inline]`

Set the field of view. The cameras field of view can be contrained to either the horizontal or vertex axis of the camera, or unconstrained in which case the camera/application are left to choose an appropriate field of view. The default value if 60 degrres.

double osg::CameraView::getFieldOfView () const `[inline]`

Get the field of view.

void osg::CameraView::setFieldOfViewMode (FieldOfViewMode *mode*) `[inline]`

Set the field of view mode - controlling how the field of view of the camera is contrained by the CameaView settings.

FieldOfViewMode osg::CameraView::getFieldOfViewMode () const `[inline]`

Get the field of view mode.

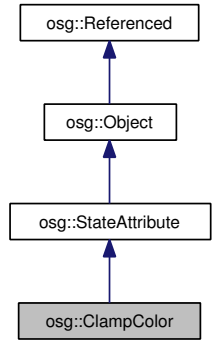
void osg::CameraView::setFocalLength (double *focalLength*) `[inline]`

Set the focal length of the camera. A focal length of 0.0 indicates that the camera/application should determine the focal length. The default value is 0.0.

double osg::CameraView::getFocalLength () const `[inline]`

Get the focal length of the camera.

4.50 osg::ClampColor Class Reference



Public Member Functions

- **ClampColor** (GLenum vertexMode, GLenum fragmentMode, GLenum readMode)
- ClampColor (const ClampColor &rhs, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, ClampColor, CLAMPCOLOR)
- virtual int compare (const StateAttribute &sa) const
- void **setClampVertexColor** (GLenum mode)
- GLenum **getClampVertexColor** () const
- void **setClampFragmentColor** (GLenum mode)
- GLenum **getClampFragmentColor** () const

- void **setClampReadColor** (GLenum mode)
- GLenum **getClampReadColor** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.51 Detailed Description

Encapsulates OpenGL ClampColor state.

4.52 Constructor & Destructor Documentation

osg::ClampColor::ClampColor (const ClampColor & *rhs*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.53 Member Function Documentation

virtual int osg::ClampColor::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual void osg::ClampColor::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

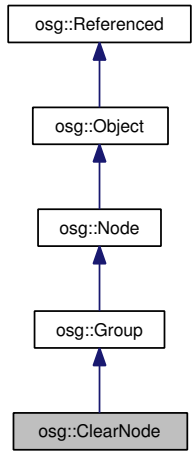
static Extensions* osg::ClampColor::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Returns the Extensions object for the given context. If createIfNotInitalized is true and the Exentsions object doesn't exist, getExtensions() creates it on the given context. Returns NULL if createIfNotInitalized is false and the Extensions object doesn't exist.

static void osg::ClampColor::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions() allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes, but need to ensure that they all use the same low common denominator extensions.

4.54 osg::ClearNode Class Reference



Public Member Functions

- **ClearNode** (const ClearNode &cs, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, ClearNode)
- void setRequiresClear (bool requiresClear)
- bool getRequiresClear () const
- void setClearColor (const Vec4 &color)
- const Vec4 & getClearColor () const
- void setClearMask (GLbitfield mask)
- GLbitfield getClearMask () const

4.55 Detailed Description

A Group node for clearing the color and depth buffers. Use setClearColor to change the clear color, and setRequiresClear to disable/enable the call clearing. You might want to disable clearing if you perform your clear by drawing fullscreen geometry. If you do this, add child nodes to perform such drawing. The default StateSet associated with this node places children in render bin -1 to ensure that children are rendered prior to the rest of the scene graph.

4.56 Member Function Documentation

void osg::ClearNode::setRequiresClear (bool *requiresClear*) `[inline]`

Enable/disable clearing via glClear.

bool osg::ClearNode::getRequiresClear () const `[inline]`

Gets whether clearing is enabled or disabled.

void osg::ClearNode::setClearColor (const Vec4 & *color*) `[inline]`

Sets the clear color.

const Vec4& osg::ClearNode::getClearColor () const `[inline]`

Returns the clear color.

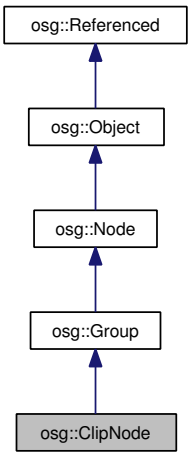
void osg::ClearNode::setClearMask (GLbitfield *mask*) `[inline]`

Set the clear mask used in glClear(..). Defaults to GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT.

GLbitfield osg::ClearNode::getClearMask () const `[inline]`

Get the clear mask.

4.57 osg::ClipNode Class Reference



Public Types

- typedef std::vector< ref_ptr< ClipPlane > > **ClipPlaneList**

Public Member Functions

- **ClipNode** (const ClipNode &es, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, ClipNode)
- void createClipBox (const BoundingBox &bb, unsigned int clipPlaneNumberBase=0)
- bool addClipPlane (ClipPlane *clipplane)
- bool removeClipPlane (ClipPlane *clipplane)
- bool removeClipPlane (unsigned int pos)
- unsigned int getNumClipPlanes () const
- ClipPlane * getClipPlane (unsigned int pos)
- const ClipPlane * getClipPlane (unsigned int pos) const
- void setClipPlaneList (const ClipPlaneList &cpl)
- ClipPlaneList & getClipPlaneList ()
- const ClipPlaneList & getClipPlaneList () const
- void setStateSetModes (StateSet &, StateAttribute::GLModeValue) const
- void setLocalStateSetModes (StateAttribute::GLModeValue=StateAttribute::ON)
- virtual BoundingSphere computeBound () const

4.58 Detailed Description

Node for defining the position of ClipPlanes in the scene.

4.59 Member Function Documentation

void osg::ClipNode::createClipBox (const BoundingBox & *bb*, unsigned int *clipPlaneNumberBase* = 0)

Creates six clip planes corresponding to the given BoundingBox.

bool osg::ClipNode::addClipPlane (ClipPlane * *clipplane*)

Adds the clipplane. Returns true on success, and false if the plane has already been added, or if clipplane is NULL.

bool osg::ClipNode::removeClipPlane (ClipPlane * *clipplane*)

Removes the clipplane. Returns true on success, false if clipplane isn't in this ClipNode.

bool osg::ClipNode::removeClipPlane (unsigned int *pos*)

Remove the ClipPlane with the given index. Returns true on success, false if pos is not a valid plane index.

unsigned int osg::ClipNode::getNumClipPlanes () const `[inline]`

Returns the number of ClipPlanes.

ClipPlane* osg::ClipNode::getClipPlane (unsigned int *pos*) `[inline]`

Get ClipPlane at the given index position.

const ClipPlane* osg::ClipNode::getClipPlane (unsigned int *pos*) const `[inline]`

Get const ClipPlane at the given index position.

void osg::ClipNode::setClipPlaneList (const ClipPlaneList & *cpl*) `[inline]`

Set the ClipPlaneList.

ClipPlaneList& osg::ClipNode::getClipPlaneList () `[inline]`

Get the ClipPlaneList.

const ClipPlaneList& osg::ClipNode::getClipPlaneList () const `[inline]`

Get the const ClipPlaneList.

void osg::ClipNode::setStateSetModes (StateSet &, StateAttribute::GLModeValue) const

Set the GLModes for all ClipPlanes, on the StateSet.

void osg::ClipNode::setLocalStateSetModes (StateAttribute::GLModeValue = `StateAttribute::ON`**)**

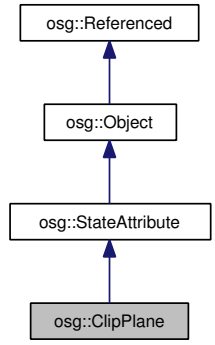
Set up the local StateSet.

virtual BoundingSphere osg::ClipNode::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Group.

4.60 osg::ClipPlane Class Reference



Public Member Functions

- **ClipPlane** (unsigned int no)
- **ClipPlane** (unsigned int no, const Vec4d &plane)
- **ClipPlane** (unsigned int no, const Plane &plane)
- **ClipPlane** (unsigned int no, double a, double b, double c, double d)
- ClipPlane (const ClipPlane &cp, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &copyop) const

- virtual bool isSameKindAs (const osg::Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Type getType () const
- virtual int compare (const StateAttribute &sa) const
- virtual unsigned int getMember () const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void setClipPlane (const Plane &plane)
- void setClipPlane (double a, double b, double c, double d)
- void setClipPlane (const Vec4d &plane)
- const Vec4d & getClipPlane () const
- void setClipPlaneNum (unsigned int num)
- unsigned int getClipPlaneNum () const
- virtual void apply (State &state) const

4.61 Detailed Description

Encapsulates OpenGL glClipPlane().

4.62 Constructor & Destructor Documentation

osg::ClipPlane::ClipPlane (const ClipPlane & *cp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.63 Member Function Documentation

virtual osg::Object* osg::ClipPlane::cloneType () const `[inline, virtual]`

Clone the type of an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual osg::Object* osg::ClipPlane::clone (const osg::CopyOp &) const `[inline, virtual]`

Clone an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual bool osg::ClipPlane::isSameKindAs (const osg::Object * *obj*) const `[inline, virtual]`

Return true if this and obj are of the same kind of object.

Reimplemented from osg::StateAttribute.

virtual const char* osg::ClipPlane::libraryName () const `[inline, virtual]`

Return the name of the attribute's library.

Reimplemented from osg::StateAttribute.

virtual const char* osg::ClipPlane::className () const `[inline, virtual]`

Return the name of the attribute's class type.

Reimplemented from osg::StateAttribute.

virtual Type osg::ClipPlane::getType () const `[inline, virtual]`

Return the Type identifier of the attribute's class type.

Implements osg::StateAttribute.

virtual int osg::ClipPlane::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual unsigned int osg::ClipPlane::getMember () const `[inline, virtual]`

Return the member identifier within the attribute's class type. Used for light number/clip plane number etc.

Reimplemented from osg::StateAttribute.

virtual bool osg::ClipPlane::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

void osg::ClipPlane::setClipPlane (const Plane & *plane*) `[inline]`

Set the clip plane with the given Plane.

void osg::ClipPlane::setClipPlane (double *a*, double *b*, double *c*, double *d*) `[inline]`

Defines the plane as [a b c d].

void osg::ClipPlane::setClipPlane (const Vec4d & *plane*) `[inline]`

Set the clip plane with the given Vec4.

const Vec4d& osg::ClipPlane::getClipPlane () const `[inline]`

Gets the clip plane as a Vec4d.

void osg::ClipPlane::setClipPlaneNum (unsigned int *num*)

Sets the clip plane number.

unsigned int osg::ClipPlane::getClipPlaneNum () const

Gets the clip plane number.

virtual void osg::ClipPlane::apply (State & *state*) const `[virtual]`

Applies the clip plane's state to the OpenGL state machine.

Reimplemented from osg::StateAttribute.

4.64 osg::ClusterCullingCallback Class Reference

Public Member Functions

- **ClusterCullingCallback** (const ClusterCullingCallback &ccc, const CopyOp &copyop)
- **ClusterCullingCallback** (const osg::Vec3 &controlPoint, const osg::Vec3 &normal, float deviation)
- **ClusterCullingCallback** (const osg::Drawable *drawable)
- **META_Object** (osg, ClusterCullingCallback)
- void computeFrom (const osg::Drawable *drawable)
- void transform (const osg::Matrixd &matrix)
- void **set** (const osg::Vec3 &controlPoint, const osg::Vec3 &normal, float deviation, float radius)
- void **setControlPoint** (const osg::Vec3 &controlPoint)
- const osg::Vec3 & **getControlPoint** () const
- void **setNormal** (const osg::Vec3 &normal)
- const osg::Vec3 & **getNormal** () const
- void **setRadius** (float radius)
- float **getRadius** () const

- void **setDeviation** (float deviation)
- float **getDeviation** () const
- virtual bool **cull** (osg::NodeVisitor *, osg::Drawable *, osg::State *) const
- virtual void operator() (Node *node, NodeVisitor *nv)

4.65 Detailed Description

Implements cluster culling to cull back facing drawables. Derived from Drawable::CullCallback.

4.66 Member Function Documentation

void osg::ClusterCullingCallback::computeFrom (const osg::Drawable * *drawable*)

Computes the control point, normal, and deviation from the given drawable contents.

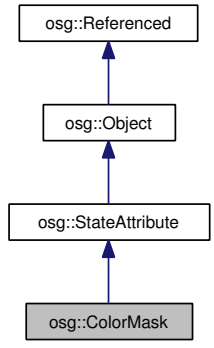
void osg::ClusterCullingCallback::transform (const osg::Matrixd & *matrix*)

Transform the ClusterCullingCallback's positional members to a new coordinate frame.

virtual void osg::ClusterCullingCallback::operator() (Node * *node*, NodeVisitor * *nv*) `[virtual]`

Callback method called by the NodeVisitor when visiting a node.

4.67 osg::ColorMask Class Reference



Public Member Functions

- **ColorMask** (bool red, bool green, bool blue, bool alpha)
- ColorMask (const ColorMask &cm, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, ColorMask, COLORMASK)
- virtual int compare (const StateAttribute &sa) const
- void **setMask** (bool red, bool green, bool blue, bool alpha)
- void **setRedMask** (bool mask)
- bool **getRedMask** () const
- void **setGreenMask** (bool mask)
- bool **getGreenMask** () const
- void **setBlueMask** (bool mask)
- bool **getBlueMask** () const
- void **setAlphaMask** (bool mask)
- bool **getAlphaMask** () const
- virtual void apply (State &state) const

4.68 Detailed Description

Encapsulates OpenGL glColorMaskFunc/Op/Mask functions.

4.69 Constructor & Destructor Documentation

osg::ColorMask::ColorMask (const ColorMask & *cm*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.70 Member Function Documentation

virtual int osg::ColorMask::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

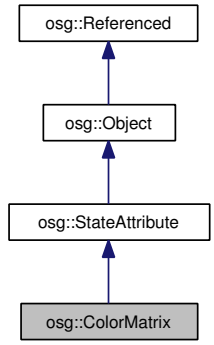
Implements osg::StateAttribute.

virtual void osg::ColorMask::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.71 osg::ColorMatrix Class Reference



Public Member Functions

- ColorMatrix (const ColorMatrix &cm, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, ColorMatrix, COLORMATRIX)
- virtual int compare (const StateAttribute &sa) const
- void setMatrix (const Matrix &matrix)
- Matrix & getMatrix ()
- const Matrix & getMatrix () const
- virtual void apply (State &state) const

4.72 Detailed Description

Encapsulates OpenGL color matrix functionality.

4.73 Constructor & Destructor Documentation

osg::ColorMatrix::ColorMatrix (const ColorMatrix & *cm*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.74 Member Function Documentation

virtual int osg::ColorMatrix::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::ColorMatrix::setMatrix (const Matrix & *matrix*) `[inline]`

Sets the color matrix.

Matrix& osg::ColorMatrix::getMatrix () `[inline]`

Gets the color matrix.

const Matrix& osg::ColorMatrix::getMatrix () const `[inline]`

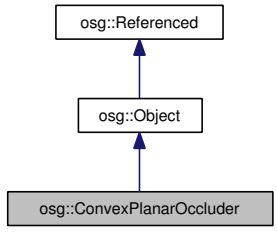
Gets the const color matrix.

virtual void osg::ColorMatrix::apply (State & *state*) const `[virtual]`

Applies as OpenGL texture matrix.

Reimplemented from osg::StateAttribute.

4.75 osg::ConvexPlanarOccluder Class Reference



Public Types

- typedef std::vector< ConvexPlanarPolygon > **HoleList**

Public Member Functions

- **ConvexPlanarOccluder** (const ConvexPlanarOccluder &cpo, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_Object** (osg, ConvexPlanarOccluder)
- void **setOccluder** (const ConvexPlanarPolygon &cpp)
- ConvexPlanarPolygon & **getOccluder** ()
- const ConvexPlanarPolygon & **getOccluder** () const
- void **addHole** (const ConvexPlanarPolygon &cpp)
- void **setHoleList** (const HoleList &holeList)
- HoleList & **getHoleList** ()
- const HoleList & **getHoleList** () const

4.76 Detailed Description

A class for representing convex clipping volumes made up of several ConvexPlanarPolygon.

4.77 osg::ConvexPlanarPolygon Class Reference

Public Types

- typedef std::vector< osg::Vec3 > **VertexList**

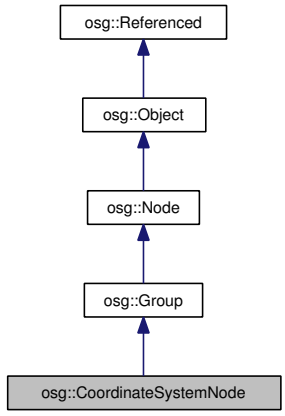
Public Member Functions

- void **add** (const Vec3 &v)
- void **setVertexList** (const VertexList &vertexList)
- VertexList & **getVertexList** ()
- const VertexList & **getVertexList** () const

4.78 Detailed Description

A class for representing components of convex clipping volumes.

4.79 osg::CoordinateSystemNode Class Reference



Public Member Functions

- **CoordinateSystemNode** (const std::string &format, const std::string &cs)
- CoordinateSystemNode (const CoordinateSystemNode &, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- **META_Node** (osg, CoordinateSystemNode)
- void set (const CoordinateSystemNode &csn)
- void setFormat (const std::string &format)
- const std::string & getFormat () const
- void setCoordinateSystem (const std::string &cs)
- const std::string & getCoordinateSystem () const
- void setEllipsoidModel (EllipsoidModel *ellipsode)
- EllipsoidModel * getEllipsoidModel ()
- const EllipsoidModel * getEllipsoidModel () const
- CoordinateFrame computeLocalCoordinateFrame (const Vec3d &position) const
- osg::Vec3d computeLocalUpVector (const Vec3d &position) const

4.80 Detailed Description

CoordinateSystem encapsulate the coordinate system that is associated with objects in a scene. For an overview of common earth bases coordinate systems see http://www.colorado.edu/geography/gcraft/notes/coordsys/coordsys_f.html

4.81 Constructor & Destructor Documentation

osg::CoordinateSystemNode::CoordinateSystemNode (const CoordinateSystemNode &, const osg::CopyOp & *copyop* = `osg::CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.82 Member Function Documentation

void osg::CoordinateSystemNode::set (const CoordinateSystemNode & *csn*)

Set the coordinate system node up by copy the format, coordinate system string, and ellipsoid model of another coordinate system node.

void osg::CoordinateSystemNode::setFormat (const std::string & *format*) `[inline]`

Set the coordinate system format string. Typical values would be WKT, PROJ4, USGS etc.

const std::string& osg::CoordinateSystemNode::getFormat () const `[inline]`

Get the coordinate system format string.

void osg::CoordinateSystemNode::setCoordinateSystem (const std::string & *cs*) `[inline]`

Set the CoordinateSystem reference string, should be stored in a form consistent with the Format.

const std::string& osg::CoordinateSystemNode::getCoordinateSystem () const `[inline]`

Get the CoordinateSystem reference string.

void osg::CoordinateSystemNode::setEllipsoidModel (EllipsoidModel * *ellipsode*) `[inline]`

Set EllipsoidModel to describe the model used to map lat, long and height into geocentric XYZ and back.

EllipsoidModel* osg::CoordinateSystemNode::getEllipsoidModel () `[inline]`

Get the EllipsoidModel.

const EllipsoidModel* osg::CoordinateSystemNode::getEllipsoidModel () const `[inline]`

Get the const EllipsoidModel.

CoordinateFrame osg::CoordinateSystemNode::computeLocalCoordinateFrame (const Vec3d & *position*) const

Compute the local coordinate frame for specified point.

osg::Vec3d osg::CoordinateSystemNode::computeLocalUpVector (const Vec3d & *position*) const

Compute the local coordinate frame for specified point.

4.83 osg::CopyOp Class Reference

Public Types

- enum **Options** {
  SHALLOW_COPY,
  DEEP_COPY_OBJECTS,
  DEEP_COPY_NODES,
  DEEP_COPY_DRAWABLES,
  DEEP_COPY_STATESETS,
  DEEP_COPY_STATEATTRIBUTES,
  DEEP_COPY_TEXTURES,
  DEEP_COPY_IMAGES,
  DEEP_COPY_ARRAYS,
  DEEP_COPY_PRIMITIVES,
  DEEP_COPY_SHAPES,
  DEEP_COPY_UNIFORMS,
  DEEP_COPY_ALL }
- typedef unsigned int **CopyFlags**

Public Member Functions

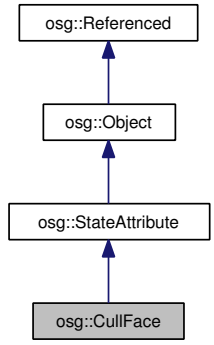
- **CopyOp** (CopyFlags flags=SHALLOW_COPY)
- virtual Referenced * **operator()** (const Referenced *ref) const
- virtual Object * **operator()** (const Object *obj) const
- virtual Node * **operator()** (const Node *node) const
- virtual Drawable * **operator()** (const Drawable *drawable) const
- virtual StateSet * **operator()** (const StateSet *stateset) const
- virtual StateAttribute * **operator()** (const StateAttribute *attr) const
- virtual Texture * **operator()** (const Texture *text) const

- virtual Image * **operator()** (const Image *image) const
- virtual Array * **operator()** (const Array *array) const
- virtual PrimitiveSet * **operator()** (const PrimitiveSet *primitives) const
- virtual Shape * **operator()** (const Shape *shape) const
- virtual Uniform * **operator()** (const Uniform *shape) const

4.84 Detailed Description

Copy Op(erator) used to control whether shallow or deep copy is used during copy construction and clone operation.

4.85 osg::CullFace Class Reference



Public Types

- enum **Mode** {
  FRONT,
  BACK,
  FRONT_AND_BACK }

Public Member Functions

- **CullFace** (Mode mode=BACK)
- CullFace (const CullFace &cf, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, CullFace, CULLFACE)
- virtual int compare (const StateAttribute &sa) const

- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- virtual void apply (State &state) const

4.86 Detailed Description

Class to globally enable/disable OpenGL's polygon culling mode.

4.87 Constructor & Destructor Documentation

osg::CullFace::CullFace (const CullFace & *cf*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.88 Member Function Documentation

virtual int osg::CullFace::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::CullFace::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

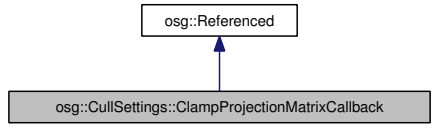
Reimplemented from osg::StateAttribute.

virtual void osg::CullFace::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.89 osg::CullSettings::ClampProjectionMatrixCallback Struct Reference



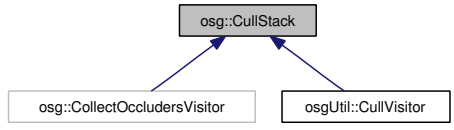
Public Member Functions

- virtual bool **clampProjectionMatrixImplementation** (osg::Matrixf &projection, double &znear, double &zfar) const =0
- virtual bool **clampProjectionMatrixImplementation** (osg::Matrixd &projection, double &znear, double &zfar) const =0

4.90 Detailed Description

Callback for overriding the CullVisitor's default clamping of the projection matrix to computed near and far values. Note, both Matrixf and Matrixd versions of clampProjectionMatrixImplementation must be implemented as the CullVisitor can target either Matrix data type, configured at compile time.

4.91 osg::CullStack Class Reference



Public Types

- typedef std::vector< ShadowVolumeOccluder > **OccluderList**
- typedef std::vector< CullingSet > **CullingStack**

Public Member Functions

- **CullStack** (const CullStack &cs)

- void **reset** ()
- void **setOccluderList** (const ShadowVolumeOccluderList &svol)
- ShadowVolumeOccluderList & **getOccluderList** ()
- const ShadowVolumeOccluderList & **getOccluderList** () const
- void **pushViewport** (osg::Viewport *viewport)
- void **popViewport** ()
- void **pushProjectionMatrix** (osg::RefMatrix *matrix)
- void **popProjectionMatrix** ()
- void **pushModelViewMatrix** (osg::RefMatrix *matrix, Transform::ReferenceFrame referenceFrame)

- void **popModelViewMatrix** ()
- float **getFrustumVolume** ()
- float pixelSize (const Vec3 &v, float radius) const
- float pixelSize (const BoundingSphere &bs) const
- float clampedPixelSize (const Vec3 &v, float radius) const
- float clampedPixelSize (const BoundingSphere &bs) const
- void **disableAndPushOccludersCurrentMask** (NodePath &nodePath)
- void **popOccludersCurrentMask** (NodePath &nodePath)
- bool **isCulled** (const std::vector< Vec3 > &vertices)
- bool **isCulled** (const BoundingBox &bb)
- bool **isCulled** (const BoundingSphere &bs)
- bool **isCulled** (const osg::Node &node)
- void **pushCurrentMask** ()
- void **popCurrentMask** ()
- CullingStack & **getClipSpaceCullingStack** ()
- CullingStack & **getProjectionCullingStack** ()
- CullingStack & **getModelViewCullingStack** ()
- CullingSet & **getCurrentCullingSet** ()
- const CullingSet & **getCurrentCullingSet** () const
- osg::Viewport * **getViewport** ()
- osg::RefMatrix * **getModelViewMatrix** ()
- osg::RefMatrix * **getProjectionMatrix** ()
- osg::Matrix **getWindowMatrix** ()
- const osg::RefMatrix * **getMVPW** ()
- const osg::Vec3 & **getReferenceViewPoint** () const
- void **pushReferenceViewPoint** (const osg::Vec3 &viewPoint)
- void **popReferenceViewPoint** ()
- const osg::Vec3 & **getEyeLocal** () const
- const osg::Vec3 & **getViewPointLocal** () const
- const osg::Vec3 **getUpLocal** () const
- const osg::Vec3 **getLookVectorLocal** () const

4.92 Detailed Description

A CullStack class which accumulates the current project, modelview matrices and the CullingSet.

4.93 Member Function Documentation

float osg::CullStack::pixelSize (const Vec3 & *v*, float *radius*) const `[inline]`

Compute the pixel size of an object at position v, with specified radius.

float osg::CullStack::pixelSize (const BoundingSphere & *bs*) const `[inline]`

Compute the pixel size of the bounding sphere.

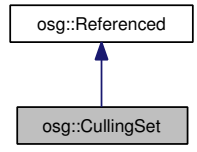
float osg::CullStack::clampedPixelSize (const Vec3 & *v*, float *radius*) const `[inline]`

Compute the pixel size of an object at position v, with specified radius. fabs()ed to always be positive.

float osg::CullStack::clampedPixelSize (const BoundingSphere & *bs*) const `[inline]`

Compute the pixel size of the bounding sphere. fabs()ed to always be positive.

4.94 osg::CullingSet Class Reference



Public Types

- enum **MaskValues** {
  NO_CULLING,
  VIEW_FRUSTUM_SIDES_CULLING,
  NEAR_PLANE_CULLING,
  FAR_PLANE_CULLING,
  VIEW_FRUSTUM_CULLING,
  SMALL_FEATURE_CULLING,
  SHADOW_OCCLUSION_CULLING,
  DEFAULT_CULLING,
  ENABLE_ALL_CULLING }
- typedef std::pair< osg::ref_ptr< osg::StateSet >, osg::Polytope > **StateFrustumPair**

- typedef std::vector< StateFrustumPair > **StateFrustumList**
- typedef std::vector< ShadowVolumeOccluder > **OccluderList**
- typedef unsigned int **Mask**

Public Member Functions

- **CullingSet** (const CullingSet &cs)
- **CullingSet** (const CullingSet &cs, const Matrix &matrix, const Vec4 &pixelSizeVector)
- CullingSet & **operator=** (const CullingSet &cs)
- void **set** (const CullingSet &cs)
- void **set** (const CullingSet &cs, const Matrix &matrix, const Vec4 &pixelSizeVector)
- void **setCullingMask** (Mask mask)
- Mask **getCullingMask** () const
- void **setFrustum** (Polytope &cv)
- Polytope & **getFrustum** ()
- const Polytope & **getFrustum** () const
- void **addStateFrustum** (StateSet *stateset, Polytope &polytope)
- void **getStateFrustumList** (StateFrustumList &sfl)
- StateFrustumList & **getStateFrustumList** ()
- void **addOccluder** (ShadowVolumeOccluder &cv)
- void **setPixelSizeVector** (const Vec4 &v)
- Vec4 & **getPixelSizeVector** ()
- const Vec4 & **getPixelSizeVector** () const
- void **setSmallFeatureCullingPixelSize** (float value)
- float & **getSmallFeatureCullingPixelSize** ()
- float **getSmallFeatureCullingPixelSize** () const
- float pixelSize (const Vec3 &v, float radius) const
- float pixelSize (const BoundingSphere &bs) const
- float clampedPixelSize (const Vec3 &v, float radius) const
- float clampedPixelSize (const BoundingSphere &bs) const
- bool **isCulled** (const std::vector< Vec3 > &vertices)
- bool **isCulled** (const BoundingBox &bb)
- bool **isCulled** (const BoundingSphere &bs)
- void **pushCurrentMask** ()
- void **popCurrentMask** ()
- void **disableAndPushOccludersCurrentMask** (NodePath &nodePath)
- void **popOccludersCurrentMask** (NodePath &nodePath)

Static Public Member Functions

- static osg::Vec4 **computePixelSizeVector** (const Viewport &W, const Matrix &P, const Matrix &M)

4.95 Detailed Description

A CullingSet class which contains a frustum and a list of occluders.

4.96 Member Function Documentation

float osg::CullingSet::pixelSize (const Vec3 & *v*, float *radius*) const `[inline]`

Compute the pixel of an object at position v, with specified radius.

float osg::CullingSet::pixelSize (const BoundingSphere & *bs*) const `[inline]`

Compute the pixel of a bounding sphere.

float osg::CullingSet::clampedPixelSize (const Vec3 & *v*, float *radius*) const `[inline]`

Compute the pixel of an object at position v, with specified radius. fabs()ed to always be positive.

float osg::CullingSet::clampedPixelSize (const BoundingSphere & *bs*) const `[inline]`

Compute the pixel of a bounding sphere. fabs()ed to always be positive.

4.97 osg::DeleteHandler Class Reference

Public Types

- typedef std::pair< int, const osg::Referenced * > **FrameNumberObjectPair**
- typedef std::list< FrameNumberObjectPair > **ObjectsToDeleteList**

Public Member Functions

- **DeleteHandler** (int numberOfFramesToRetainObjects=0)
- void setNumFramesToRetainObjects (int numberOfFramesToRetainObjects)
- int **getNumFramesToRetainObjects** () const
- void setFrameNumber (int frameNumber)
- int getFrameNumber () const
- void **doDelete** (const Referenced *object)
- virtual void flush ()
- virtual void flushAll ()
- virtual void requestDelete (const osg::Referenced *object)

4.98 Detailed Description

Class for override the default delete behavior so that users can implement their own object deletion schemes. This might be done to help implement protection of multiple threads from deleting objects unintentionally. Note, the DeleteHandler cannot itself be reference counted, otherwise it would be responsible for deleting itself! An static auto_ptr<> is used internally in Referenced.cpp to manage the DeleteHandler's memory.

4.99 Member Function Documentation

void osg::DeleteHandler::setNumFramesToRetainObjects (int *numberOfFramesToRetainObjects*) `[inline]`

Set the number of frames to retain objects that are have been requested for deletion. When set to zero objects are deleted immediately, by set to 1 there are kept around for an extra frame etc. The ability to retain obejcts for several frames is useful to prevent premature deletion when objects are stil be used the graphics threads that are using double buffering of rendering data structures with non ref_ptr<> pointers to scene graph elements.

void osg::DeleteHandler::setFrameNumber (int *frameNumber*) `[inline]`

Set the current frame numberso that subsequent deletes get tagged as associated with this frame.

int osg::DeleteHandler::getFrameNumber () const `[inline]`

Get the current frame number.

virtual void osg::DeleteHandler::flush () `[virtual]`

Flush objects that ready to be fully deleted.

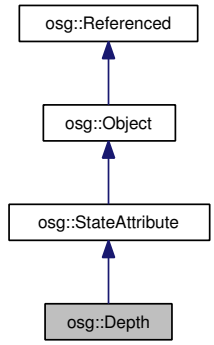
virtual void osg::DeleteHandler::flushAll () `[virtual]`

Flush all objects that the DeleteHandler holds. Note, this should only be called if there are no threads running with non ref_ptr<> pointers, such as graphics threads.

virtual void osg::DeleteHandler::requestDelete (const osg::Referenced * *object*) `[virtual]`

Request the deletion of an object. Depending on users implementation of DeleteHandler, the delete of the object may occur straight away or be delayed until doDelete is called. The default implementation does a delete straight away.

4.100 osg::Depth Class Reference



Public Types

- enum **Function** {
  NEVER,
  LESS,
  EQUAL,
  LEQUAL,
  GREATER,
  NOTEQUAL,
  GEQUAL,
  ALWAYS }

Public Member Functions

- **Depth** (Function func=LESS, double zNear=0.0, double zFar=1.0, bool writeMask=true)
- Depth (const Depth &dp, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Depth, DEPTH)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFunction** (Function func)
- Function **getFunction** () const
- void **setRange** (double zNear, double zFar)
- void **setZNear** (double zNear)
- double **getZNear** () const
- void **setZFar** (double zFar)
- double **getZFar** () const

- void **setWriteMask** (bool mask)
- bool **getWriteMask** () const
- virtual void apply (State &state) const

4.101 Detailed Description

Encapsulate OpenGL glDepthFunc/Mask/Range functions.

4.102 Constructor & Destructor Documentation

osg::Depth::Depth (const Depth & *dp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.103 Member Function Documentation

virtual int osg::Depth::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Depth::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

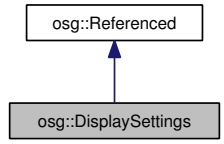
Reimplemented from osg::StateAttribute.

virtual void osg::Depth::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.104 osg::DisplaySettings Class Reference



Public Types

- enum **DisplayType** {
  MONITOR,
  POWERWALL,
  REALITY_CENTER,
  HEAD_MOUNTED_DISPLAY }
- enum **StereoMode** {
  QUAD_BUFFER,
  ANAGLYPHIC,
  HORIZONTAL_SPLIT,
  VERTICAL_SPLIT,
  LEFT_EYE,
  RIGHT_EYE,
  HORIZONTAL_INTERLACE,
  VERTICAL_INTERLACE,
  CHECKERBOARD }
- enum **SplitStereoHorizontalEyeMapping** {
  LEFT_EYE_LEFT_VIEWPORT,
  LEFT_EYE_RIGHT_VIEWPORT }
- enum **SplitStereoVerticalEyeMapping** {
  LEFT_EYE_TOP_VIEWPORT,
  LEFT_EYE_BOTTOM_VIEWPORT }

Public Member Functions

- **DisplaySettings** (ArgumentParser &arguments)
- **DisplaySettings** (const DisplaySettings &vs)
- DisplaySettings & **operator=** (const DisplaySettings &vs)

- void **setDisplaySettings** (const DisplaySettings &vs)
- void **merge** (const DisplaySettings &vs)
- void **setDefaults** ()
- void readEnvironmentalVariables ()
- void readCommandLine (ArgumentParser &arguments)
- void **setDisplayType** (DisplayType type)
- DisplayType **getDisplayType** () const
- void **setStereo** (bool on)
- bool **getStereo** () const
- void **setStereoMode** (StereoMode mode)
- StereoMode **getStereoMode** () const
- void **setEyeSeparation** (float eyeSeparation)
- float **getEyeSeparation** () const
- void **setSplitStereoHorizontalEyeMapping** (SplitStereoHorizontalEyeMapping m)
- SplitStereoHorizontalEyeMapping **getSplitStereoHorizontalEyeMapping** () const
- void **setSplitStereoHorizontalSeparation** (int s)
- int **getSplitStereoHorizontalSeparation** () const
- void **setSplitStereoVerticalEyeMapping** (SplitStereoVerticalEyeMapping m)
- SplitStereoVerticalEyeMapping **getSplitStereoVerticalEyeMapping** () const
- void **setSplitStereoVerticalSeparation** (int s)
- int **getSplitStereoVerticalSeparation** () const
- void **setSplitStereoAutoAdjustAspectRatio** (bool flag)
- bool **getSplitStereoAutoAdjustAspectRatio** () const
- void **setScreenWidth** (float width)
- float **getScreenWidth** () const
- void **setScreenHeight** (float height)
- float **getScreenHeight** () const
- void **setScreenDistance** (float distance)
- float **getScreenDistance** () const
- void **setDoubleBuffer** (bool flag)
- bool **getDoubleBuffer** () const
- void **setRGB** (bool flag)
- bool **getRGB** () const
- void **setDepthBuffer** (bool flag)
- bool **getDepthBuffer** () const
- void **setMinimumNumAlphaBits** (unsigned int bits)
- unsigned int **getMinimumNumAlphaBits** () const
- bool **getAlphaBuffer** () const
- void **setMinimumNumStencilBits** (unsigned int bits)
- unsigned int **getMinimumNumStencilBits** () const
- bool **getStencilBuffer** () const
- void **setMinimumNumAccumBits** (unsigned int red, unsigned int green, unsigned int blue, unsigned int alpha)
- unsigned int **getMinimumNumAccumRedBits** () const
- unsigned int **getMinimumNumAccumGreenBits** () const
- unsigned int **getMinimumNumAccumBlueBits** () const

- unsigned int **getMinimumNumAccumAlphaBits** () const
- bool **getAccumBuffer** () const
- void **setMaxNumberOfGraphicsContexts** (unsigned int num)
- unsigned int **getMaxNumberOfGraphicsContexts** () const
- void **setNumMultiSamples** (unsigned int samples)
- unsigned int **getNumMultiSamples** () const
- bool **getMultiSamples** () const
- void **setCompileContextsHint** (bool useCompileContexts)
- bool **getCompileContextsHint** () const
- void **setSerializeDrawDispatch** (bool serializeDrawDispatch)
- bool **getSerializeDrawDispatch** () const

Static Public Member Functions

- static DisplaySettings * instance ()

4.105 Detailed Description

DisplaySettings class for encapsulating what visuals are required and have been set up, and the status of stereo viewing.

4.106 Member Function Documentation

static DisplaySettings* osg::DisplaySettings::instance () `[static]`

Maintain a DisplaySettings singleton for objects to query at runtime.

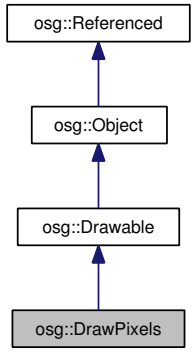
void osg::DisplaySettings::readEnvironmentalVariables ()

read the environmental variables.

void osg::DisplaySettings::readCommandLine (ArgumentParser & *arguments*)

read the commandline arguments.

4.107 osg::DrawPixels Class Reference



Public Member Functions

- DrawPixels (const DrawPixels &drawimage, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- void **setPosition** (const osg::Vec3 &position)
- osg::Vec3 & **getPosition** ()
- const osg::Vec3 & **getPosition** () const
- void **setImage** (osg::Image *image)
- osg::Image * **getImage** ()
- const osg::Image * **getImage** () const
- void **setUseSubImage** (bool useSubImage)
- bool **getUseSubImage** () const
- void **setSubImageDimensions** (unsigned int offsetX, unsigned int offsetY, unsigned int width, unsigned int height)
- void **getSubImageDimensions** (unsigned int &offsetX, unsigned int &offsetY, unsigned int &width, unsigned int &height) const
- virtual void drawImplementation (RenderInfo &renderInfo) const
- virtual BoundingBox computeBound () const

4.108 Detailed Description

DrawPixels is an osg::Drawable subclass which encapsulates the drawing of images using glDrawPixels.

4.109 Constructor & Destructor Documentation

osg::DrawPixels::DrawPixels (const DrawPixels & *drawimage*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.110 Member Function Documentation

virtual Object* osg::DrawPixels::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::DrawPixels::clone (const CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual const char* osg::DrawPixels::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Reimplemented from osg::Drawable.

virtual const char* osg::DrawPixels::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Reimplemented from osg::Drawable.

virtual void osg::DrawPixels::drawImplementation (RenderInfo & *renderInfo*) const `[virtual]`

drawImplementation(RenderInfo&) is a pure virtual method for the actual implementation of OpenGL drawing calls, such as vertex arrays and primitives, that must be implemented in concrete subclasses of the Drawable base class, examples include osg::Geometry and osg::ShapeDrawable. drawImplementation(RenderInfo&) is called from the draw(RenderInfo&) method, with the draw method handling management of OpenGL display lists, and drawImplementation(RenderInfo&) handling the actuall drawing itself.

Parameters:

renderInfo The osg::RenderInfo object that encapulates the current rendering information including the osg::State OpenGL state for the current graphics context.

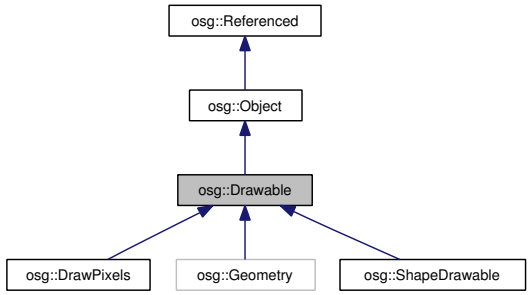
Implements osg::Drawable.

virtual BoundingBox osg::DrawPixels::computeBound () const `[virtual]`

Compute the bounding box around Drawables's geometry.

Reimplemented from osg::Drawable.

4.111 osg::Drawable Class Reference



Public Types

- enum **AttributeTypes** {
  VERTICES,
  WEIGHTS,
  NORMALS,
  COLORS,
  SECONDARY_COLORS,
  FOG_COORDS,
  ATTRIBUTE_6,
  ATTRIBUTE_7,
  TEXTURE_COORDS,
  TEXTURE_COORDS_0,
  TEXTURE_COORDS_1,
  TEXTURE_COORDS_2,
  TEXTURE_COORDS_3,
  TEXTURE_COORDS_4,
  TEXTURE_COORDS_5,

TEXTURE_COORDS_6,
TEXTURE_COORDS_7 }

- typedef std::vector< Node * > ParentList
- typedef unsigned int **AttributeType**

Public Member Functions

- Drawable (const Drawable &drawable, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Geometry * asGeometry ()
- virtual const Geometry * asGeometry () const
- virtual void computeDataVariance ()
- const ParentList & getParents () const
- ParentList getParents ()
- Node * getParent (unsigned int i)
- const Node * getParent (unsigned int i) const
- unsigned int getNumParents () const
- void setStateSet (StateSet *stateset)
- StateSet * getStateSet ()
- const StateSet * getStateSet () const
- StateSet * getOrCreateStateSet ()
- void setInitialBound (const osg::BoundingBox &bbox)
- const BoundingBox & getInitialBound () const
- void dirtyBound ()
- const BoundingBox & getBound () const
- virtual BoundingBox computeBound () const
- void setComputeBoundingBoxCallback (ComputeBoundingBoxCallback *callback)
- ComputeBoundingBoxCallback * getComputeBoundingBoxCallback ()
- const ComputeBoundingBoxCallback * getComputeBoundingBoxCallback () const
- void setShape (Shape *shape)
- Shape * getShape ()
- const Shape * getShape () const
- void setSupportsDisplayList (bool flag)
- bool getSupportsDisplayList () const
- void setUseDisplayList (bool flag)
- bool getUseDisplayList () const
- GLuint & getDisplayList (unsigned int contextID) const
- virtual void setUseVertexBufferObjects (bool flag)
- bool getUseVertexBufferObjects () const
- virtual void dirtyDisplayList ()
- virtual unsigned int getGLObjectSizeHint () const
- void draw (RenderInfo &renderInfo) const

- virtual void compileGLObjects (RenderInfo &renderInfo) const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (State *state=0) const
- virtual void setUpdateCallback (UpdateCallback *ac)
- UpdateCallback * getUpdateCallback ()
- const UpdateCallback * getUpdateCallback () const
- bool requiresUpdateTraversal () const
- virtual void setEventCallback (EventCallback *ac)
- EventCallback * getEventCallback ()
- const EventCallback * getEventCallback () const
- bool requiresEventTraversal () const
- virtual void setCullCallback (CullCallback *cc)
- CullCallback * getCullCallback ()
- const CullCallback * getCullCallback () const
- virtual void setDrawCallback (DrawCallback *dc)
- DrawCallback * getDrawCallback ()
- const DrawCallback * getDrawCallback () const
- virtual void drawImplementation (RenderInfo &renderInfo) const =0
- virtual bool supports (const AttributeFunctor &) const
- virtual void accept (AttributeFunctor &)
- virtual bool supports (const ConstAttributeFunctor &) const
- virtual void accept (ConstAttributeFunctor &) const
- virtual bool supports (const PrimitiveFunctor &) const
- virtual void accept (PrimitiveFunctor &) const
- virtual bool supports (const PrimitiveIndexFunctor &) const
- virtual void accept (PrimitiveIndexFunctor &) const

Static Public Member Functions

- static GLuint generateDisplayList (unsigned int contextID, unsigned int sizeHint=0)
- static void setMinimumNumberOfDisplayListsToRetainInCache (unsigned int minimum)
- static unsigned int getMinimumNumberOfDisplayListsToRetainInCache ()
- static void deleteDisplayList (unsigned int contextID, GLuint globj, unsigned int sizeHint=0)
- static void flushAllDeletedDisplayLists (unsigned int contextID)
- static void flushDeletedDisplayLists (unsigned int contextID, double &availableTime)
- static void deleteVertexBufferObject (unsigned int contextID, GLuint globj)
- static void flushDeletedVertexBufferObjects (unsigned int contextID, double currentTime, double &availableTime)
- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Static Public Attributes

- static unsigned int **s_numberDrawablesReusedLastInLastFrame**
- static unsigned int **s_numberNewDrawablesInLastFrame**
- static unsigned int **s_numberDeletedDrawablesInLastFrame**

Friends

- class **Node**
- class **Geode**
- class **StateSet**

Classes

- class **AttributeFunctor**
- struct ComputeBoundingBoxCallback
- class **ConstAttributeFunctor**
- struct **CullCallback**
- struct DrawCallback
- struct **EventCallback**
- class Extensions
- struct **UpdateCallback**

4.112 Detailed Description

Pure virtual base class for drawable geometry. In OSG, everything that can be rendered is implemented as a class derived from `Drawable`. The `Drawable` class contains no drawing primitives, since these are provided by subclasses such as `osg::Geometry`.

Notice that a `Drawable` is not a `Node`, and therefore it cannot be directly added to a scene graph. Instead, `Drawable`s are attached to `Geode`s, which are scene graph nodes.

The OpenGL state that must be used when rendering a `Drawable` is represented by a `StateSet`. Since a `Drawable` has a reference (`osg::ref_ptr`) to a `StateSet`, `StateSet`s can be shared between different `Drawable`s. In fact, sharing `StateSet`s is a good way to improve performance, since this allows OSG to reduce the number of expensive changes in the OpenGL state.

Finally, `Drawable`s can also be shared between different `Geode`s, so that the same geometry (loaded to memory just once) can be used in different parts of the scene graph.

4.113 Member Typedef Documentation

typedef std::vector<Node*> osg::Drawable::ParentList

A vector of osg::Node pointers which is used to store the parent(s) of drawable.

4.114 Constructor & Destructor Documentation

osg::Drawable::Drawable (const Drawable & *drawable*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.115 Member Function Documentation

virtual const char* osg::Drawable::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

Reimplemented in osg::DrawPixels, and osg::ShapeDrawable.

virtual const char* osg::Drawable::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osg::DrawPixels, and osg::ShapeDrawable.

virtual Geometry* osg::Drawable::asGeometry () `[inline, virtual]`

Convert 'this' into a Geometry pointer if Drawable is a Geometry, otherwise return 0. Equivalent to dynamic_cast<Geometry*>(this).

virtual const Geometry* osg::Drawable::asGeometry () const `[inline, virtual]`

Convert 'const this' into a const Geometry pointer if Drawable is a Geometry, otherwise return 0. Equivalent to dynamic_cast<const Geometry*>(this).

virtual void osg::Drawable::computeDataVariance () `[virtual]`

Compute the DataVariance based on an assestment of callback etc.

Reimplemented from osg::Object.

const ParentList& osg::Drawable::getParents () const `[inline]`

Get the parent list of drawable.

ParentList osg::Drawable::getParents () `[inline]`

Get the a copy of parent list of node. A copy is returned to prevent modification of the parent list.

Node* osg::Drawable::getParent (unsigned int *i*) `[inline]`

Get a single parent of Drawable.

Parameters:

i index of the parent to get.

Returns:

the parent i.

const Node* osg::Drawable::getParent (unsigned int *i*) const `[inline]`

Get a single const parent of Drawable.

Parameters:

i index of the parent to get.

Returns:

the parent i.

unsigned int osg::Drawable::getNumParents () const `[inline]`

Get the number of parents of node.

Returns:

the number of parents of this node.

void osg::Drawable::setStateSet (StateSet * *stateset*)

Set the StateSet attached to the Drawable. Previously attached StateSet are automatically unreferenced on assignment of a new drawstate.

StateSet* osg::Drawable::getStateSet () `[inline]`

Get the attached StateSet.

const StateSet* osg::Drawable::getStateSet () const `[inline]`

Get the attached const StateSet.

StateSet* osg::Drawable::getOrCreateStateSet ()

Get the attached const StateSet, if one is not already attached create one, attach it to the drawable and return a pointer to it.

void osg::Drawable::setInitialBound (const osg::BoundingBox & *bbox*) `[inline]`

Set the initial bounding volume to use when computing the overall bounding volume.

const BoundingBox& osg::Drawable::getInitialBound () const `[inline]`

Set the initial bounding volume to use when computing the overall bounding volume.

void osg::Drawable::dirtyBound ()

Dirty the bounding box, forcing a computeBound() on the next call to getBound(). Should be called in the internal geometry of the Drawable is modified.

const BoundingBox& osg::Drawable::getBound () const `[inline]`

Get BoundingBox of Drawable. If the BoundingBox is not up to date then its updated via an internal call to computeBond().

virtual BoundingBox osg::Drawable::computeBound () const `[virtual]`

Compute the bounding box around Drawables's geometry.

Reimplemented in osg::DrawPixels, and osg::ShapeDrawable.

void osg::Drawable::setComputeBoundingBoxCallback (ComputeBoundingBoxCallback * *callback*) `[inline]`

Set the compute bound callback to override the default computeBound.

ComputeBoundingBoxCallback* osg::Drawable::getComputeBoundingBoxCallback () `[inline]`

Get the compute bound callback.

const ComputeBoundingBoxCallback* osg::Drawable::getComputeBoundingBoxCallback () const `[inline]`

Get the const compute bound callback.

void osg::Drawable::setShape (Shape * *shape*) `[inline]`

Set the Shape of the `Drawable`. The shape can be used to speed up collision detection or as a guide for procedural geometry generation.

See also:
osg::Shape.

Shape* osg::Drawable::getShape () `[inline]`

Get the Shape of the Drawable.

const Shape* osg::Drawable::getShape () const `[inline]`

Get the const Shape of the const Drawable.

void osg::Drawable::setSupportsDisplayList (bool *flag*)

Set the drawable so that it can or cannot be used in conjunction with OpenGL display lists. When set to true, calls to Drawable::setUseDisplayList, whereas when set to false, no display lists can be created and calls to setUseDisplayList are ignored, and a warning is produced. The latter is typically used to guard against the switching on of display lists on objects with dynamic internal data such as continuous Level of Detail algorithms.

bool osg::Drawable::getSupportsDisplayList () const `[inline]`

Get whether display lists are supported for this drawable instance.

void osg::Drawable::setUseDisplayList (bool *flag*)

When set to true, force the draw method to use OpenGL Display List for rendering. If false, rendering directly. If the display list has not been compiled already, the next call to draw will automatically create the display list.

bool osg::Drawable::getUseDisplayList () const `[inline]`

Return whether OpenGL display lists are being used for rendering.

GLuint& osg::Drawable::getDisplayList (unsigned int *contextID*) const `[inline]`

Return OpenGL display list for specified contextID.

virtual void osg::Drawable::setUseVertexBufferObjects (bool *flag*) `[virtual]`

When set to true, ignore the setUseDisplayList() settings, and hints to the drawImplementation method to use OpenGL vertex buffer objects for rendering.

bool osg::Drawable::getUseVertexBufferObjects () const `[inline]`

Return whether OpenGL vertex buffer objects should be used when supported by OpenGL driver.

virtual void osg::Drawable::dirtyDisplayList () `[virtual]`

Force a recompile on next draw() of any OpenGL display list associated with this geoset.

virtual unsigned int osg::Drawable::getGLObjectSizeHint () const `[inline, virtual]`

Return the estimated size of GLObjects (display lists/vertex buffer objects) that are associated with this drawable. This size is used a hint for reuse of deleted display lists/vertex buffer objects.

void osg::Drawable::draw (RenderInfo & *renderInfo*) const `[inline]`

Draw OpenGL primitives. If the `Drawable` has `_useDisplayList` set to `true`, then use an OpenGL display list, automatically compiling one if required. Otherwise, call `drawImplementation()`.

Note:

This method should *not* be overridden in subclasses, as it manages the optional display list (notice this is not even `virtual`). Subclasses should override `drawImplementation()` instead.

virtual void osg::Drawable::compileGLObjects (RenderInfo & *renderInfo*) const `[virtual]`

Immediately compile this `Drawable` into an OpenGL Display List.

Note:

Operation is ignored if `_useDisplayList` is `false`.

virtual void osg::Drawable::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Referenced.

virtual void osg::Drawable::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Object.

virtual void osg::Drawable::releaseGLObjects (State * *state* = 0) const `[virtual]`

If State is non-zero, this function releases OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::Object.

virtual void osg::Drawable::setUpdateCallback (UpdateCallback * *ac*) `[virtual]`

Set the UpdateCallback which allows users to attach customize the updating of an object during the update traversal.

UpdateCallback* osg::Drawable::getUpdateCallback () `[inline]`

Get the non const UpdateCallback.

const UpdateCallback* osg::Drawable::getUpdateCallback () const `[inline]`

Get the const UpdateCallback.

bool osg::Drawable::requiresUpdateTraversal () const `[inline]`

Return whether this Drawable has update callbacks associated with it, and therefore must be traversed.

virtual void osg::Drawable::setEventCallback (EventCallback * *ac*) `[virtual]`

Set the EventCallback which allows users to attach customize the updating of an object during the Event traversal.

EventCallback* osg::Drawable::getEventCallback () `[inline]`

Get the non const EventCallback.

const EventCallback* osg::Drawable::getEventCallback () const `[inline]`

Get the const EventCallback.

bool osg::Drawable::requiresEventTraversal () const `[inline]`

Return whether this Drawable has event callbacks associated with it, and therefore must be traversed.

virtual void osg::Drawable::setCullCallback (CullCallback * *cc*) `[inline, virtual]`

Set the CullCallback which allows users to customize the culling of Drawable during the cull traversal.

CullCallback* osg::Drawable::getCullCallback () `[inline]`

Get the non const CullCallback.

const CullCallback* osg::Drawable::getCullCallback () const `[inline]`

Get the const CullCallback.

virtual void osg::Drawable::setDrawCallback (DrawCallback * *dc*) `[inline, virtual]`

Set the DrawCallback which allows users to attach customize the drawing of existing Drawable object.

DrawCallback* osg::Drawable::getDrawCallback () `[inline]`

Get the non const DrawCallback.

const DrawCallback* osg::Drawable::getDrawCallback () const `[inline]`

Get the const DrawCallback.

virtual void osg::Drawable::drawImplementation (RenderInfo & *renderInfo*) const `[pure virtual]`

drawImplementation(RenderInfo&) is a pure virtual method for the actual implementation of OpenGL drawing calls, such as vertex arrays and primitives, that must be implemented in concrete subclasses of the Drawable base class, examples include osg::Geometry and osg::ShapeDrawable. drawImplementation(RenderInfo&) is called from the draw(RenderInfo&) method, with the draw method handling management of OpenGL display lists, and drawImplementation(RenderInfo&) handling the actuall drawing itself.

Parameters:

renderInfo The osg::RenderInfo object that encapulates the current rendering information including the osg::State OpenGL state for the current graphics context.

Implemented in osg::DrawPixels, and osg::ShapeDrawable.

static GLuint osg::Drawable::generateDisplayList (unsigned int *contextID*, unsigned int *sizeHint* = 0) `[static]`

Return a OpenGL display list handle a newly generated or reused from display list cache.

static void osg::Drawable::setMinimumNumberOfDisplayListsToRetainInCache (unsigned int *minimum*) `[static]`

Set the minimum number of display lists to retain in the deleted display list cache.

static unsigned int osg::Drawable::getMinimumNumberOfDisplayListsToRetainInCache () `[static]`

Get the minimum number of display lists to retain in the deleted display list cache.

static void osg::Drawable::deleteDisplayList (unsigned int *contextID*, GLuint *globj*, unsigned int *sizeHint* = 0) `[static]`

Use deleteDisplayList instead of glDeleteList to allow OpenGL display list to be cached until they can be deleted by the OpenGL context in which they were created, specified by contextID.

static void osg::Drawable::flushAllDeletedDisplayLists (unsigned int *contextID*) `[static]`

Flush all the cached display list which need to be deleted in the OpenGL context related to contextID.

static void osg::Drawable::flushDeletedDisplayLists (unsigned int *contextID*, double & *availableTime*) `[static]`

Flush the cached display list which need to be deleted in the OpenGL context related to contextID.

static void osg::Drawable::deleteVertexBufferObject (unsigned int *contextID*, GLuint *globj*) `[static]`

Use deleteVertexBufferObject instead of glDeleteBuffers to allow OpenGL buffer objects to be cached until they can be deleted by the OpenGL context in which they were created, specified by contextID.

static void osg::Drawable::flushDeletedVertexBufferObjects (unsigned int *contextID*, double *currentTime*, double & *availableTime*) `[static]`

Flush all the cached vertex buffer objects which need to be deleted in the OpenGL context related to contextID.

virtual bool osg::Drawable::supports (const AttributeFunctor &) const `[inline, virtual]`

Return true if the Drawable subclass supports accept(AttributeFunctor&).

virtual void osg::Drawable::accept (AttributeFunctor &) `[inline, virtual]`

accept an AttributeFunctor and call its methods to tell it about the internal attributes that this Drawable has. return true if functor handled by drawable, return false on failure of drawable to generate functor calls.

virtual bool osg::Drawable::supports (const ConstAttributeFunctor &) const `[inline, virtual]`

Return true if the Drawable subclass supports accept(ConstAttributeFunctor&).

Reimplemented in osg::ShapeDrawable.

virtual void osg::Drawable::accept (ConstAttributeFunctor &) const `[inline, virtual]`

Accept an AttributeFunctor and call its methods to tell it about the internal attributes that this Drawable has. return true if functor handled by drawable, return false on failure of drawable to generate functor calls.

Reimplemented in osg::ShapeDrawable.

virtual bool osg::Drawable::supports (const PrimitiveFunctor &) const `[inline, virtual]`

Return true if the Drawable subclass supports accept(PrimitiveFunctor&).

Reimplemented in osg::ShapeDrawable.

virtual void osg::Drawable::accept (PrimitiveFunctor &) const `[inline, virtual]`

Accept a PrimitiveFunctor and call its methods to tell it about the internal primitives that this Drawable has. return true if functor handled by drawable, return false on failure of drawable to generate functor calls. Note, PrimtiveFunctor only provides const access of the primitives, as primitives may be procedurally generated so one cannot modify it.

Reimplemented in osg::ShapeDrawable.

virtual bool osg::Drawable::supports (const PrimitiveIndexFunctor &) const `[inline, virtual]`

Return true if the Drawable subclass supports accept(PrimitiveIndexFunctor&).

virtual void osg::Drawable::accept (PrimitiveIndexFunctor &) const `[inline, virtual]`

Accept a PrimitiveIndexFunctor and call its methods to tell it about the internal primitives that this Drawable has. return true if functor handled by drawable, return false on failure of drawable to generate functor calls. Note, PrimtiveIndexFunctor only provide const access of the primitives, as primitives may be procedurally generated so one cannot modify it.

static Extensions* osg::Drawable::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Extension object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object is only created with the graphics context associated with ContextID..

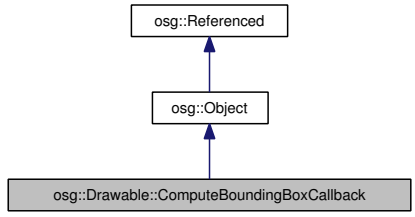
static void osg::Drawable::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions allows users to override the extensions across graphics contexts. typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

void osg::Drawable::setBound (const BoundingBox & *bb*) const `[protected]`

set the bounding box .

4.116 osg::Drawable::ComputeBoundingBoxCallback Struct Reference



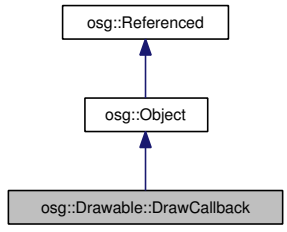
Public Member Functions

- **ComputeBoundingBoxCallback** (const ComputeBoundingBoxCallback &, const CopyOp &)
- **META_Object** (osg, ComputeBoundingBoxCallback)
- virtual BoundingBox **computeBound** (const osg::Drawable &) const

4.117 Detailed Description

Callback to allow users to override the default computation of bounding volume.

4.118 osg::Drawable::DrawCallback Struct Reference



Public Member Functions

- **DrawCallback** (const DrawCallback &, const CopyOp &)
- **META_Object** (osg, DrawCallback)
- virtual void drawImplementation (osg::RenderInfo &, const osg::Drawable *) const

4.119 Detailed Description

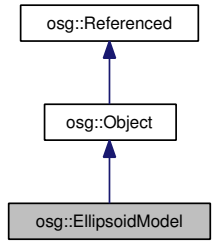
Callback attached to an Drawable which allows the users to customize the drawing of an exist Drawable object. The draw callback is implement as a replacement to the Drawable's own drawImplementation() method, if the the user intends to decorate the existing draw code then simple call the drawable->drawImplementation() from with the callbacks drawImplementation() method. This allows the users to do both pre and post callbacks without fuss and can even disable the inner draw if required.

4.120 Member Function Documentation

virtual void osg::Drawable::DrawCallback::drawImplementation (osg::RenderInfo &, const osg::Drawable *) const `[inline, virtual]`

do customized draw code.

4.121 osg::EllipsoidModel Class Reference



Public Member Functions

- EllipsoidModel (double radiusEquator=WGS_84_RADIUS_EQUATOR, double radiusPolar=WGS_84_RADIUS_POLAR)
- **EllipsoidModel** (const EllipsoidModel &et, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Object** (osg, EllipsoidModel)

- void **setRadiusEquator** (double radius)
- double **getRadiusEquator** () const
- void **setRadiusPolar** (double radius)
- double **getRadiusPolar** () const
- void **convertLatLongHeightToXYZ** (double latitude, double longitude, double height, double &X, double &Y, double &Z) const
- void **convertXYZToLatLongHeight** (double X, double Y, double Z, double &latitude, double &longitude, double &height) const
- void **computeLocalToWorldTransformFromLatLongHeight** (double latitude, double longitude, double height, osg::Matrixd &localToWorld) const
- void **computeLocalToWorldTransformFromXYZ** (double X, double Y, double Z, osg::Matrixd &localToWorld) const
- osg::Vec3d **computeLocalUpVector** (double X, double Y, double Z) const

4.122 Detailed Description

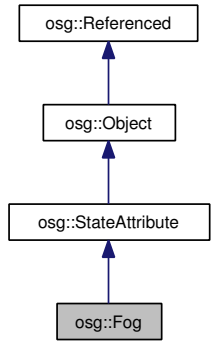
EllipsoidModel encapsulates the ellipsoid used to model astronomical bodies, such as sun, planets, moon etc.

4.123 Constructor & Destructor Documentation

osg::EllipsoidModel::EllipsoidModel (double *radiusEquator* = `WGS_84_RADIUS_EQUATOR`**, double *radiusPolar* =** `WGS_84_RADIUS_POLAR`**)** `[inline]`

WGS_84 is a common representation of the earth's spheroid

4.124 osg::Fog Class Reference



Public Types

- enum **Mode** {
  LINEAR,
  EXP,
  EXP2 }
- enum **FogCoordinateSource** {
  FOG_COORDINATE,
  FRAGMENT_DEPTH }

Public Member Functions

- Fog (const Fog &fog, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Fog, FOG)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- void **setDensity** (float density)
- float **getDensity** () const
- void **setStart** (float start)
- float **getStart** () const
- void **setEnd** (float end)
- float **getEnd** () const
- void **setColor** (const Vec4 &color)
- const Vec4 & **getColor** () const
- void **setFogCoordinateSource** (GLint source)
- GLint **getFogCoordinateSource** () const
- virtual void apply (State &state) const

4.125 Detailed Description

Fog - encapsulates OpenGL fog state.

4.126 Constructor & Destructor Documentation

osg::Fog::Fog (const Fog & *fog*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.127 Member Function Documentation

virtual int osg::Fog::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Fog::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

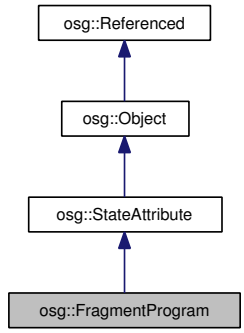
Reimplemented from osg::StateAttribute.

virtual void osg::Fog::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.128 osg::FragmentProgram Class Reference



Public Types

- typedef std::map< GLuint, Vec4 > **LocalParamList**
- typedef std::map< GLenum, Matrix > **MatrixList**

Public Member Functions

- FragmentProgram (const FragmentProgram &vp, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, FragmentProgram, FRAGMENTPROGRAM)
- virtual int compare (const osg::StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- GLuint & getFragmentProgramID (unsigned int contextID) const
- void setFragmentProgram (const char *program)
- void setFragmentProgram (const std::string &program)
- const std::string & getFragmentProgram () const
- void setProgramLocalParameter (const GLuint index, const Vec4 &p)
- void setLocalParameters (const LocalParamList &lpl)
- LocalParamList & getLocalParameters ()
- const LocalParamList & getLocalParameters () const
- void setMatrix (const GLenum mode, const Matrix &matrix)
- void setMatrices (const MatrixList &matrices)
- MatrixList & getMatrices ()
- const MatrixList & getMatrices () const
- void dirtyFragmentProgramObject ()
- virtual void apply (State &state) const
- virtual void compileGLObjects (State &state) const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (State *state=0) const

Static Public Member Functions

- static void deleteFragmentProgramObject (unsigned int contextID, GLuint handle)
- static void flushDeletedFragmentProgramObjects (unsigned int contextID, double currentTime, double &availableTime)
- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.129 Detailed Description

FragmentProgram - encapsulates the OpenGL ARB fragment program state.

4.130 Constructor & Destructor Documentation

osg::FragmentProgram::FragmentProgram (const FragmentProgram & *vp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.131 Member Function Documentation

virtual int osg::FragmentProgram::compare (const osg::StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::FragmentProgram::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

GLuint& osg::FragmentProgram::getFragmentProgramID (unsigned int *contextID*) const `[inline]`

Get the handle to the fragment program id for the current context.

void osg::FragmentProgram::setFragmentProgram (const char * *program*) `[inline]`

Set the fragment program using a C style string.

void osg::FragmentProgram::setFragmentProgram (const std::string & *program*) `[inline]`

Set the fragment program using C++ style string.

const std::string& osg::FragmentProgram::getFragmentProgram () const `[inline]`

Get the fragment program.

void osg::FragmentProgram::setProgramLocalParameter (const GLuint *index*, const Vec4 & *p*) `[inline]`

Set Program Parameters

void osg::FragmentProgram::setLocalParameters (const LocalParamList & *lpl*) `[inline]`

Set list of Program Parameters

LocalParamList& osg::FragmentProgram::getLocalParameters () `[inline]`

Get list of Program Parameters

const LocalParamList& osg::FragmentProgram::getLocalParameters () const `[inline]`

Get const list of Program Parameters

void osg::FragmentProgram::setMatrix (const GLenum *mode*, const Matrix & *matrix*) `[inline]`

Matrix

void osg::FragmentProgram::setMatrices (const MatrixList & *matrices*) `[inline]`

Set list of Matrices

MatrixList& osg::FragmentProgram::getMatrices () `[inline]`

Get list of Matrices

const MatrixList& osg::FragmentProgram::getMatrices () const `[inline]`

Get list of Matrices

void osg::FragmentProgram::dirtyFragmentProgramObject ()

Force a recompile on next apply() of associated OpenGL vertex program objects.

static void osg::FragmentProgram::deleteFragmentProgramObject (unsigned int *contextID*, GLuint *handle*) `[static]`

use deleteFragmentProgramObject instead of glDeletePrograms to allow OpenGL Fragment Program objects to be cached until they can be deleted by the OpenGL context in which they were created, specified by contextID.

static void osg::FragmentProgram::flushDeletedFragmentProgramObjects (unsigned int *contextID*, double *currentTime*, double & *availableTime*) `[static]`

flush all the cached fragment programs which need to be deleted in the OpenGL context related to contextID.

virtual void osg::FragmentProgram::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

virtual void osg::FragmentProgram::compileGLObjects (State &) const `[inline, virtual]`

Default to nothing to compile - all state is applied immediately.

Reimplemented from osg::StateAttribute.

virtual void osg::FragmentProgram::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::StateAttribute.

virtual void osg::FragmentProgram::releaseGLObjects (State * *state* = 0) const `[virtual]`

release an OpenGL objects in specified graphics context if State object is passed, otherwise release OpenGL objexts for all graphics context if State object pointer == NULL.

Reimplemented from osg::StateAttribute.

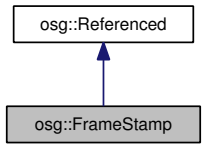
static Extensions* osg::FragmentProgram::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Extension object for that context has not yet been created and the 'createIfNotInitalized' flag has been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID..

static void osg::FragmentProgram::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions allows users to override the extensions across graphics contexts. typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

4.132 osg::FrameStamp Class Reference



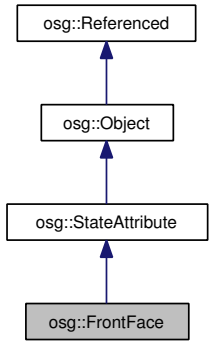
Public Member Functions

- **FrameStamp** (const FrameStamp &fs)
- FrameStamp & **operator=** (const FrameStamp &fs)
- void **setFrameNumber** (int fnum)
- int **getFrameNumber** () const
- void **setReferenceTime** (double refTime)
- double **getReferenceTime** () const
- void **setSimulationTime** (double refTime)
- double **getSimulationTime** () const
- void **setCalendarTime** (const tm &calendarTime)
- void **getCalendarTime** (tm &calendarTime) const

4.133 Detailed Description

Class which encapsulates the frame number, reference time and calendar time of specific frame, used to synchonize operations on the scene graph and other machines when using a graphics cluster. Note the calendar time can be an artificial simulation time or capture the real time of day etc.

4.134 osg::FrontFace Class Reference



Public Types

- enum **Mode** {
  CLOCKWISE,
  COUNTER_CLOCKWISE }

Public Member Functions

- **FrontFace** (Mode face=COUNTER_CLOCKWISE)
- FrontFace (const FrontFace &ff, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, FrontFace, FRONTFACE)
- virtual int compare (const StateAttribute &sa) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- virtual void apply (State &state) const

4.135 Detailed Description

Class to specify the orientation of front-facing polygons.

4.136 Constructor & Destructor Documentation

osg::FrontFace::FrontFace (const FrontFace & *ff*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.137 Member Function Documentation

virtual int osg::FrontFace::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

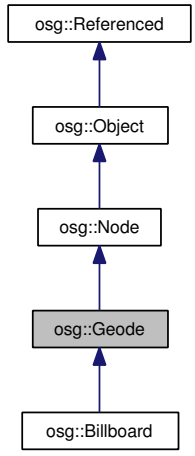
Implements osg::StateAttribute.

virtual void osg::FrontFace::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.138 osg::Geode Class Reference



Public Types

- typedef std::vector< ref_ptr< Drawable > > **DrawableList**

Public Member Functions

- Geode (const Geode &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Geode)
- virtual bool addDrawable (Drawable *drawable)
- virtual bool removeDrawable (Drawable *drawable)
- virtual bool removeDrawables (unsigned int i, unsigned int numDrawablesToRemove=1)
- virtual bool replaceDrawable (Drawable *origDraw, Drawable *newDraw)
- virtual bool setDrawable (unsigned int i, Drawable *drawable)
- unsigned int getNumDrawables () const
- Drawable * getDrawable (unsigned int i)
- const Drawable * getDrawable (unsigned int i) const
- bool containsDrawable (const Drawable *gset) const
- unsigned int getDrawableIndex (const Drawable *drawable) const
- const DrawableList & getDrawableList () const
- void compileDrawables (RenderInfo &renderInfo)
- const BoundingBox & getBoundingBox () const
- virtual BoundingSphere computeBound () const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (osg::State *=0) const

4.139 Detailed Description

A Geode is a "geometry node", that is, a leaf node on the scene graph that can have "renderable things" attached to it. In OSG, renderable things are represented by objects from the Drawable class, so a Geode is a Node whose purpose is grouping Drawables.

4.140 Constructor & Destructor Documentation

osg::Geode::Geode (const Geode &, const CopyOp & *copyop* = CopyOp::SHALLOW_COPY)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.141 Member Function Documentation

virtual bool osg::Geode::addDrawable (Drawable * *drawable*) [virtual]

Add a Drawable to the Geode. If drawable is not NULL and is not contained in the Geode then increment its reference count, add it to the drawables list and dirty the bounding sphere to force it to be recomputed on the next call to getBound().

Parameters:

drawable The Drawable to be added to the Geode.

Returns:

true for success; false otherwise.

Reimplemented in osg::Billboard.

virtual bool osg::Geode::removeDrawable (Drawable * *drawable*) [virtual]

Remove a Drawable from the Geode. Equivalent to removeDrawable(getDrawableIndex(drawable)).

Parameters:

drawable The drawable to be removed.

Returns:

true if at least one Drawable was removed. false otherwise.

Reimplemented in osg::Billboard.

virtual bool osg::Geode::removeDrawables (unsigned int *i*, unsigned int *numDrawablesToRemove* = 1) [virtual]

Remove Drawable(s) from the specified position in Geode's drawable list.

Parameters:

i The index of the first Drawable to remove.

numDrawablesToRemove The number of Drawable to remove.

Returns:

true if at least one Drawable was removed. false otherwise.

virtual bool osg::Geode::replaceDrawable (Drawable * *origDraw*, Drawable * *newDraw*) [virtual]

Replace specified Drawable with another Drawable. Equivalent to setDrawable(getDrawableIndex(origDraw),newDraw), see docs for setDrawable() for further details on implementation.

virtual bool osg::Geode::setDrawable (unsigned int *i*, Drawable * *drawable*) [virtual]

Set Drawable at position i. Decrement the reference count origGSet and increments the reference count of newGset, and dirty the bounding sphere to force it to recompute on next getBound() and returns true. If origDrawable is not found then return false and do not add newGset. If newGset is NULL then return false and do not remove origGset.

Returns:

true if set correctly, false on failure (if node==NULL || i is out of range).

unsigned int osg::Geode::getNumDrawables () const [inline]

Return the number of Drawables currently attached to the Geode.

Drawable* osg::Geode::getDrawable (unsigned int *i*) [inline]

Return the Drawable at position i.

const Drawable* osg::Geode::getDrawable (unsigned int *i*) const [inline]

Return the Drawable at position i.

bool osg::Geode::containsDrawable (const Drawable * *gset*) const [inline]

Return true if a given Drawable is contained within Geode.

unsigned int osg::Geode::getDrawableIndex (const Drawable * *drawable*) const `[inline]`

Get the index number of `drawable`.

Returns:

A value between 0 and `getNumDrawables()-1` if `drawable` is found; if not found, then `getNumDrawables()` is returned.

const DrawableList& osg::Geode::getDrawableList () const `[inline]`

Get the list of drawables.

void osg::Geode::compileDrawables (RenderInfo & *renderInfo*)

Compile OpenGL Display List for each drawable.

const BoundingBox& osg::Geode::getBoundingBox () const `[inline]`

Return the Geode's bounding box, which is the union of all the bounding boxes of the geode's drawables.

virtual BoundingSphere osg::Geode::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Node.

Reimplemented in osg::Billboard.

virtual void osg::Geode::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Node.

virtual void osg::Geode::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

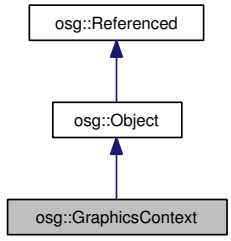
Reimplemented from osg::Node.

virtual void osg::Geode::releaseGLObjects (osg::State * = 0) const `[virtual]`

If State is non-zero, this function releases any associated OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::Node.

4.142 osg::GraphicsContext Class Reference



Public Types

- typedef std::vector< GraphicsContext * > **GraphicsContexts**
- typedef std::list< ref_ptr< Operation > > **OperationQueue**
- typedef std::list< osg::Camera * > **Cameras**

Public Member Functions

- void add (Operation *operation)
- void remove (Operation *operation)
- void remove (const std::string &name)
- void removeAllOperations ()
- void runOperations ()
- OperationQueue & getOperationsQueue ()
- OpenThreads::Mutex * getOperationsMutex ()
- osg::RefBlock * getOperationsBlock ()
- Operation * getCurrentOperation ()
- const Traits * getTraits () const
- virtual bool valid () const =0
- void setState (State *state)
- State * getState ()
- const State * getState () const
- void setClearColor (const Vec4 &color)
- const Vec4 & getClearColor () const
- void setClearMask (GLbitfield mask)
- GLbitfield getClearMask () const
- virtual void clear ()
- bool realize ()
- void close (bool callCloseImplementation=true)
- void swapBuffers ()

- bool isRealized () const
- bool makeCurrent ()
- bool makeContextCurrent (GraphicsContext *readContext)
- bool releaseContext ()
- bool isCurrent () const
- void bindPBufferToTexture (GLenum buffer)
- void createGraphicsThread ()
- void setGraphicsThread (GraphicsThread *gt)
- GraphicsThread * getGraphicsThread ()
- const GraphicsThread * getGraphicsThread () const
- virtual bool realizeImplementation ()=0
- virtual bool isRealizedImplementation () const =0
- virtual void closeImplementation ()=0
- virtual bool makeCurrentImplementation ()=0
- virtual bool makeContextCurrentImplementation (GraphicsContext *readContext)=0
- virtual bool releaseContextImplementation ()=0
- virtual void bindPBufferToTextureImplementation (GLenum buffer)=0
- virtual void swapBuffersImplementation ()=0
- void resized (int x, int y, int width, int height)
- void setResizedCallback (ResizedCallback *rc)
- ResizedCallback * getResizedCallback ()
- const ResizedCallback * getResizedCallback () const
- virtual void resizedImplementation (int x, int y, int width, int height)
- Cameras & getCameras ()
- const Cameras & getCameras () const
- virtual bool **isSameKindAs** (const Object *object) const
- virtual const char * libraryName () const
- virtual const char * className () const

Static Public Member Functions

- static void setWindowingSystemInterface (WindowingSystemInterface *wsInterface)
- static WindowingSystemInterface * getWindowingSystemInterface ()
- static GraphicsContext * createGraphicsContext (Traits *traits)
- static unsigned int createNewContextID ()
- static unsigned int getMaxContextID ()
- static void incrementContextIDUsageCount (unsigned int contextID)
- static void decrementContextIDUsageCount (unsigned int contextID)
- static GraphicsContexts getAllRegisteredGraphicsContexts ()
- static GraphicsContexts getRegisteredGraphicsContexts (unsigned int contextID)
- static void setCompileContext (unsigned int contextID, GraphicsContext *gc)
- static GraphicsContext * getOrCreateCompileContext (unsigned int contextID)
- static GraphicsContext * getCompileContext (unsigned int contextID)

Friends

- class osg::Camera

Classes

- struct **ResizedCallback**
- struct **ScreenIdentifier**
- struct Traits
- struct WindowingSystemInterface

4.143 Detailed Description

Base class for providing Windowing API agnostic access to creating and managing graphics context.

4.144 Member Function Documentation

static void osg::GraphicsContext::setWindowingSystemInterface (WindowingSystemInterface * *wsInterface*) `[static]`

Set the querry the windowing system for screens and create graphics context - this functor should be supplied by the windows toolkit.

static WindowingSystemInterface* osg::GraphicsContext::getWindowingSystemInterface () `[static]`

Get the WindowingSystemInterface

static GraphicsContext* osg::GraphicsContext::createGraphicsContext (Traits * *traits*) `[static]`

Create a graphics context for a specified set of traits.

static unsigned int osg::GraphicsContext::createNewContextID () `[static]`

Create a contextID for a new graphics context, this contextID is used to set up the osg::State associate with context. Automatically increments the usage count of the contextID to 1.

static unsigned int osg::GraphicsContext::getMaxContextID () `[static]`

Get the current max ContextID.

static void osg::GraphicsContext::incrementContextIDUsageCount (unsigned int *contextID*) `[static]`

Increment the usage count associate with a contextID. The usage count speficies how many graphics contexts a specific contextID is shared between.

static void osg::GraphicsContext::decrementContextIDUsageCount (unsigned int *contextID*) `[static]`

Decrement the usage count associate with a contextID. Once the contextID goes to 0 the contextID is then free to be reused.

static GraphicsContexts osg::GraphicsContext::getAllRegisteredGraphicsContexts () `[static]`

Get all the registered graphics contexts.

static GraphicsContexts osg::GraphicsContext::getRegisteredGraphicsContexts (unsigned int *contextID*) `[static]`

Get all the registered graphics contexts associated with a specific contextID.

static void osg::GraphicsContext::setCompileContext (unsigned int *contextID*, GraphicsContext * *gc*) `[static]`

Get the GraphicsContext for doing background compilation for GraphicsContexts associated with specified contextID.

static GraphicsContext* osg::GraphicsContext::getOrCreateCompileContext (unsigned int *contextID*) `[static]`

Get existing or create a new GraphicsContext to do background compilation for GraphicsContexts associated with specified contextID.

static GraphicsContext* osg::GraphicsContext::getCompileContext (unsigned int *contextID*) `[static]`

Get the GraphicsContext for doing background compilation for GraphicsContexts associated with specified contextID.

void osg::GraphicsContext::add (Operation * *operation*)

Add operation to end of OperationQueue.

void osg::GraphicsContext::remove (Operation * *operation*)

Remove operation from OperationQueue.

void osg::GraphicsContext::remove (const std::string & *name*)

Remove named operation from OperationQueue.

void osg::GraphicsContext::removeAllOperations ()

Remove all operations from OperationQueue.

void osg::GraphicsContext::runOperations ()

Run the operations.

OperationQueue& osg::GraphicsContext::getOperationsQueue () `[inline]`

Get the operations queue, not you must use the OperationsMutex when accessing the queue.

OpenThreads::Mutex* osg::GraphicsContext::getOperationsMutex () `[inline]`

Get the operations queue mutex.

osg::RefBlock* osg::GraphicsContext::getOperationsBlock () `[inline]`

Get the operations queue block used to mark an empty queue, if you end items into the empty queu you must release this block.

Operation* osg::GraphicsContext::getCurrentOperation () `[inline]`

Get the current operations that is being run.

const Traits* osg::GraphicsContext::getTraits () const `[inline]`

Get the traits of the GraphicsContext.

virtual bool osg::GraphicsContext::valid () const `[pure virtual]`

Return whether a valid and usable GraphicsContext has been created.

void osg::GraphicsContext::setState (State * *state*) `[inline]`

Set the State object which tracks the current OpenGL state for this graphics context.

State* osg::GraphicsContext::getState () [inline]

Get the State object which tracks the current OpenGL state for this graphics context.

const State* osg::GraphicsContext::getState () const [inline]

Get the const State object which tracks the current OpenGL state for this graphics context.

void osg::GraphicsContext::setClearColor (const Vec4 & *color*) [inline]

Sets the clear color.

const Vec4& osg::GraphicsContext::getClearColor () const [inline]

Returns the clear color.

void osg::GraphicsContext::setClearMask (GLbitfield *mask*) [inline]

Set the clear mask used in glClear(..). Defaults to GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT.

GLbitfield osg::GraphicsContext::getClearMask () const [inline]

Get the clear mask.

virtual void osg::GraphicsContext::clear () [virtual]

Do an OpenGL clear of the full graphics context/window. Note, must only be called from a thread with this context current.

bool osg::GraphicsContext::realize ()

Realise the GraphicsContext.

void osg::GraphicsContext::close (bool *callCloseImplementation* = true)

close the graphics context. close(bool) stops any associated graphics threads, releases the contextID for the GraphicsContext then optional calls closeImplementation() to do the actual deletion of the graphics. This call is made optional as there are times when the graphics context has already been deleted externally and only the OSG side of the its data need to be closed down.

void osg::GraphicsContext::swapBuffers ()

swap the front and back buffers.

bool osg::GraphicsContext::isRealized () const `[inline]`

Return true if the graphics context has been realised and is ready to use.

bool osg::GraphicsContext::makeCurrent ()

Make this graphics context current. Implementated by calling makeCurrentImplementation(). Returns true on success.

bool osg::GraphicsContext::makeContextCurrent (GraphicsContext * *readContext*)

Make this graphics context current with specified read context. Implementated by calling makeContextCurrentImplementation(). Returns true on success.

bool osg::GraphicsContext::releaseContext ()

Release the graphics context. Returns true on success.

bool osg::GraphicsContext::isCurrent () const `[inline]`

Return true if the current thread has this OpenGL graphics context.

void osg::GraphicsContext::bindPBufferToTexture (GLenum *buffer*) `[inline]`

Bind the graphics context to associated texture.

void osg::GraphicsContext::createGraphicsThread ()

Create a graphics thread to the graphics context, so that the thread handles all OpenGL operations.

void osg::GraphicsContext::setGraphicsThread (GraphicsThread * *gt*)

Assign a graphics thread to the graphics context, so that the thread handles all OpenGL operations.

GraphicsThread* osg::GraphicsContext::getGraphicsThread () `[inline]`

Get the graphics thread assigned the graphics context.

const GraphicsThread* osg::GraphicsContext::getGraphicsThread () const `[inline]`

Get the const graphics thread assigned the graphics context.

virtual bool osg::GraphicsContext::realizeImplementation () `[pure virtual]`

Realise the GraphicsContext implementation, Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual bool osg::GraphicsContext::isRealizedImplementation () const `[pure virtual]`

Return true if the graphics context has been realised, and is ready to use, implementation. Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual void osg::GraphicsContext::closeImplementation () `[pure virtual]`

Close the graphics context implementation. Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual bool osg::GraphicsContext::makeCurrentImplementation () `[pure virtual]`

Make this graphics context current implementation. Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual bool osg::GraphicsContext::makeContextCurrentImplementation (GraphicsContext * *readContext*) `[pure virtual]`

Make this graphics context current with specified read context implementation. Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual bool osg::GraphicsContext::releaseContextImplementation () `[pure virtual]`

Release the graphics context implementation.

virtual void osg::GraphicsContext::bindPBufferToTextureImplementation (GLenum *buffer*) `[pure virtual]`

Pure virtual, Bind the graphics context to associated texture implementation. Pure virtual - must be implemented by concrate implementations of GraphicsContext.

virtual void osg::GraphicsContext::swapBuffersImplementation () `[pure virtual]`

Swap the front and back buffers implementation. Pure virtual - must be implemented by Concrate implementations of GraphicsContext.

void osg::GraphicsContext::resized (int *x*, int *y*, int *width*, int *height*) `[inline]`

resized method should be called when the underlying window has been resized and the GraphicsWindow and associated Cameras must be updated to keep in sync with the new size.

void osg::GraphicsContext::setResizedCallback (ResizedCallback * *rc*) `[inline]`

Set the resized callback which overrides the GraphicsConext::realizedImplementation(), allow developers to provide custom behavior in response to a window being resized.

ResizedCallback* osg::GraphicsContext::getResizedCallback () `[inline]`

Get the resized callback which overrides the GraphicsConext::realizedImplementation().

const ResizedCallback* osg::GraphicsContext::getResizedCallback () const `[inline]`

Get the const resized callback which overrides the GraphicsConext::realizedImplementation().

virtual void osg::GraphicsContext::resizedImplementation (int *x*, int *y*, int *width*, int *height*) `[virtual]`

resized implementation, by default resizes the viewports and aspect ratios the cameras associated with the graphics Window.

Cameras& osg::GraphicsContext::getCameras () `[inline]`

Get the the list of cameras associated with this graphics context.

const Cameras& osg::GraphicsContext::getCameras () const `[inline]`

Get the the const list of cameras associated with this graphics context.

virtual const char* osg::GraphicsContext::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

virtual const char* osg::GraphicsContext::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::GraphicsContext::cloneType () const `[inline, protected, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::GraphicsContext::clone (const CopyOp &) const `[inline, protected, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

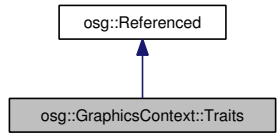
static void osg::GraphicsContext::registerGraphicsContext (GraphicsContext * *gc*) `[static, protected]`

Register a GraphicsContext.

static void osg::GraphicsContext::unregisterGraphicsContext (GraphicsContext * *gc*) `[static, protected]`

Unregister a GraphicsContext.

4.145 osg::GraphicsContext::Traits Struct Reference



Public Attributes

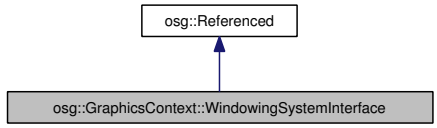
- int **x**
- int **y**
- int **width**
- int **height**
- std::string **windowName**
- bool **windowDecoration**
- bool **supportsResize**
- unsigned int **red**
- unsigned int **blue**
- unsigned int **green**
- unsigned int **alpha**
- unsigned int **depth**
- unsigned int **stencil**
- unsigned int **sampleBuffers**
- unsigned int **samples**
- bool **pbuffer**

- bool **quadBufferStereo**
- bool **doubleBuffer**
- GLenum **target**
- GLenum **format**
- unsigned int **level**
- unsigned int **face**
- unsigned int **mipMapGeneration**
- bool **vsync**
- bool **useMultiThreadedOpenGLEngine**
- bool **useCursor**
- GraphicsContext * **sharedContext**
- osg::ref_ptr< osg::Referenced > **inheritedWindowData**
- bool **setInheritedWindowPixelFormat**

4.146 Detailed Description

GraphicsContext Traits object provides the specification of what type of graphics context is required.

4.147 osg::GraphicsContext::WindowingSystemInterface Struct Reference



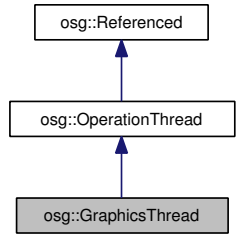
Public Member Functions

- virtual unsigned int **getNumScreens** (const ScreenIdentifier &screenIdentifier=ScreenIdentifier())=0
- virtual void **getScreenResolution** (const ScreenIdentifier &screenIdentifier, unsigned int &width, unsigned int &height)=0
- virtual bool **setScreenResolution** (const ScreenIdentifier &, unsigned int, unsigned int)
- virtual bool **setScreenRefreshRate** (const ScreenIdentifier &, double)
- virtual GraphicsContext * **createGraphicsContext** (Traits *traits)=0

4.148 Detailed Description

Callback to be implemented to provide access to Windowing API's ability to create Windows/pbuffers.

4.149 osg::GraphicsThread Class Reference



Public Member Functions

- virtual void run ()

4.150 Detailed Description

GraphicsThread is a helper class for running OpenGL GraphicsOperation within a single thread assigned to a specific GraphicsContext.

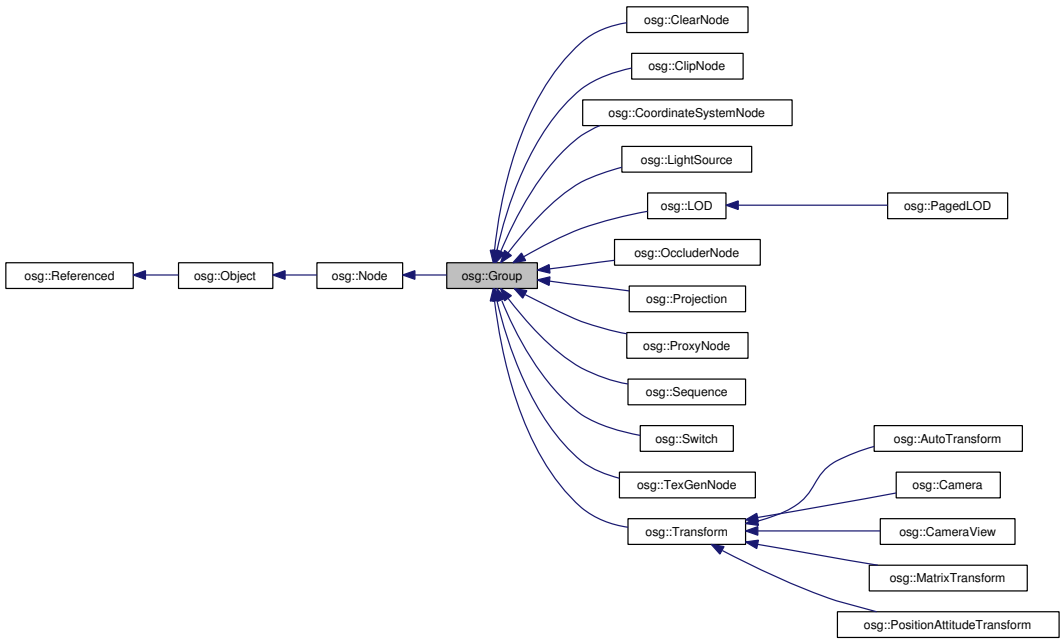
4.151 Member Function Documentation

virtual void osg::GraphicsThread::run () `[virtual]`

Run does the graphics thread run loop.

Reimplemented from osg::OperationThread.

4.152 osg::Group Class Reference



Public Member Functions

- Group (const Group &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Group)
- virtual Group * asGroup ()
- virtual const Group * asGroup () const
- virtual void traverse (NodeVisitor &nv)
- virtual bool addChild (Node *child)
- virtual bool insertChild (unsigned int index, Node *child)
- bool removeChild (Node *child)
- bool removeChild (unsigned int pos, unsigned int numChildrenToRemove=1)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove)
- virtual bool replaceChild (Node *origChild, Node *newChild)
- unsigned int getNumChildren () const
- virtual bool setChild (unsigned int i, Node *node)
- Node * getChild (unsigned int i)
- const Node * getChild (unsigned int i) const
- bool containsNode (const Node *node) const
- unsigned int getChildIndex (const Node *node) const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual void resizeGLObjectBuffers (unsigned int maxSize)

- virtual void releaseGLObjects (osg::State *=0) const
- virtual BoundingSphere computeBound () const

4.153 Detailed Description

General group node which maintains a list of children. Children are reference counted. This allows children to be shared with memory management handled automatically via osg::Referenced.

4.154 Constructor & Destructor Documentation

osg::Group::Group (const Group &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.155 Member Function Documentation

virtual Group* osg::Group::asGroup () `[inline, virtual]`

convert 'this' into a Group pointer if Node is a Group, otherwise return 0. Equivalent to dynamic_cast<Group*>(this).

Reimplemented from osg::Node.

virtual const Group* osg::Group::asGroup () const `[inline, virtual]`

convert 'const this' into a const Group pointer if Node is a Group, otherwise return 0. Equivalent to dynamic_cast<const Group*>(this).

Reimplemented from osg::Node.

virtual void osg::Group::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::Node.

Reimplemented in osg::LOD, osg::PagedLOD, osg::ProxyNode, osg::Sequence, and osg::Switch.

virtual bool osg::Group::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented in osg::LOD, osg::PagedLOD, osg::ProxyNode, osg::Sequence, and osg::Switch.

virtual bool osg::Group::insertChild (unsigned int *index*, Node * *child*) `[virtual]`

Insert Node to Group at specific location. The new child node is inserted into the child list before the node at the specified index. No nodes are removed from the group with this operation.

Reimplemented in osg::Sequence, and osg::Switch.

bool osg::Group::removeChild (Node * *child*) `[inline]`

Remove Node from Group. If Node is contained in Group then remove it from the child list, decrement its reference count, and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. If Node is not found then return false and do not change the reference count of the Node. Note, do not override, only override removeChildren(,) is required.

Reimplemented in osg::Sequence.

bool osg::Group::removeChild (unsigned int *pos*, unsigned int *numChildrenToRemove* = 1) `[inline]`

Remove Node from Group. If Node is contained in Group then remove it from the child list, decrement its reference count, and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. If Node is not found then return false and do not change the reference count of the Node. Note, do not override, only override removeChildren(,) is required.

virtual bool osg::Group::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove*) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented in osg::LOD, osg::PagedLOD, osg::ProxyNode, osg::Sequence, and osg::Switch.

virtual bool osg::Group::replaceChild (Node * *origChild*, Node * *newChild*) `[virtual]`

Replace specified Node with another Node. Equivalent to setChild(getChildIndex(orignChild),node) See docs for setChild for futher details on implementation.

unsigned int osg::Group::getNumChildren () const `[inline]`

Return the number of chilren nodes.

virtual bool osg::Group::setChild (unsigned int *i*, Node * *node*) `[virtual]`

Set child node at position i. Return true if set correctly, false on failure (if node==NULL || i is out of range). When Set can be successful applied, the algorithm is : decrement the reference count origNode and increment the reference count of newNode, and dirty the bounding sphere to force it to recompute on next

getBound() and return true. If origNode is not found then return false and do not add newNode. If newNode is NULL then return false and do not remove origNode. Also returns false if newChild is a Scene node.

Node* osg::Group::getChild (unsigned int *i*) `[inline]`

Return child node at position i.

const Node* osg::Group::getChild (unsigned int *i*) const `[inline]`

Return child node at position i.

bool osg::Group::containsNode (const Node * *node*) const `[inline]`

Return true if node is contained within Group.

unsigned int osg::Group::getChildIndex (const Node * *node*) const `[inline]`

Get the index number of child, return a value between 0 and _children.size()-1 if found, if not found then return _children.size().

virtual void osg::Group::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Node.

Reimplemented in osg::LightSource, and osg::TexGenNode.

virtual void osg::Group::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Node.

Reimplemented in osg::Camera.

virtual void osg::Group::releaseGLObjects (osg::State * = 0) const `[virtual]`

If State is non-zero, this function releases any associated OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::Node.

Reimplemented in osg::Camera.

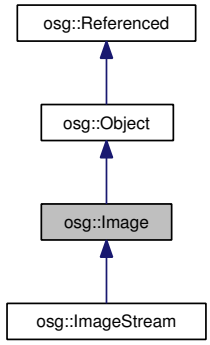
virtual BoundingSphere osg::Group::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Node.

Reimplemented in osg::AutoTransform, osg::ClipNode, osg::LightSource, osg::LOD, osg::OccluderNode, osg::ProxyNode, osg::Switch, and osg::Transform.

4.156 osg::Image Class Reference



Public Types

- enum **AllocationMode** {
  NO_DELETE,
  USE_NEW_DELETE,
  USE_MALLOC_FREE }
- enum **Origin** {
  BOTTOM_LEFT,
  TOP_LEFT }
- typedef std::vector< unsigned int > MipmapDataType

Public Member Functions

- Image (const Image &image, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const

- virtual const char * className () const
- virtual int compare (const Image &rhs) const
- void **setFileName** (const std::string &fileName)
- const std::string & **getFileName** () const
- void setAllocationMode (AllocationMode mode)
- AllocationMode getAllocationMode () const
- void allocateImage (int s, int t, int r, GLenum pixelFormat, GLenum type, int packing=1)
- void setImage (int s, int t, int r, GLint internalTextureformat, GLenum pixelFormat, GLenum type, unsigned char *data, AllocationMode mode, int packing=1)
- void readPixels (int x, int y, int width, int height, GLenum pixelFormat, GLenum type)
- void readImageFromCurrentTexture (unsigned int contextID, bool copyMipMapsIfAvailable, GLenum type=GL_UNSIGNED_BYTE)
- void scaleImage (int s, int t, int r)
- void scaleImage (int s, int t, int r, GLenum newDataType)
- void copySubImage (int s_offset, int t_offset, int r_offset, osg::Image *source)
- void setOrigin (Origin origin)
- Origin getOrigin () const
- int s () const
- int t () const
- int r () const
- void **setInternalTextureFormat** (GLint internalFormat)
- GLint **getInternalTextureFormat** () const
- void **setPixelFormat** (GLenum pixelFormat)
- GLenum **getPixelFormat** () const
- void **setDataType** (GLenum dataType)
- GLenum **getDataType** () const
- void **setPacking** (unsigned int packing)
- unsigned int **getPacking** () const
- unsigned int getPixelSizeInBits () const
- unsigned int getRowSizeInBytes () const
- unsigned int getImageSizeInBytes () const
- unsigned int getTotalSizeInBytes () const
- unsigned int getTotalSizeInBytesIncludingMipmaps () const
- bool valid () const
- unsigned char * data ()
- const unsigned char * data () const
- unsigned char * **data** (int column, int row=0, int image=0)
- const unsigned char * **data** (int column, int row=0, int image=0) const
- Vec4 getColor (unsigned int s, unsigned t=0, unsigned r=0) const
- Vec4 getColor (const Vec2 &texcoord) const
- Vec4 getColor (const Vec3 &texcoord) const
- void flipHorizontal ()
- void flipVertical ()
- void ensureValidSizeForTexturing (GLint maxTextureSize)
- void dirty ()
- void setModifiedCount (unsigned int value)

- unsigned int getModifiedCount () const
- bool **isMipmap** () const
- unsigned int **getNumMipmapLevels** () const
- void setMipmapLevels (const MipmapDataType &mipmapDataVector)
- const MipmapDataType & **getMipmapLevels** () const
- unsigned int **getMipmapOffset** (unsigned int mipmapLevel) const
- unsigned char * **getMipmapData** (unsigned int mipmapLevel)
- const unsigned char * **getMipmapData** (unsigned int mipmapLevel) const
- bool isImageTranslucent () const
- void setPixelBufferObject (PixelBufferObject *buffer)
- PixelBufferObject * getPixelBufferObject ()
- const PixelBufferObject * getPixelBufferObject () const

Static Public Member Functions

- static bool **isPackedType** (GLenum type)
- static GLenum **computePixelFormat** (GLenum pixelFormat)
- static unsigned int **computeNumComponents** (GLenum pixelFormat)
- static unsigned int **computePixelSizeInBits** (GLenum pixelFormat, GLenum type)
- static unsigned int **computeRowWidthInBytes** (int width, GLenum pixelFormat, GLenum type, int packing)
- static int **computeNearestPowerOfTwo** (int s, float bias=0.5f)
- static int **computeNumberOfMipmapLevels** (int s, int t=1, int r=1)

4.157 Detailed Description

Image class for encapsulating the storage texture image data.

4.158 Member Typedef Documentation

typedef std::vector< unsigned int > osg::Image::MipmapDataType

Precomputed mipmaps stuff.

4.159 Constructor & Destructor Documentation

osg::Image::Image (const Image & *image*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.160 Member Function Documentation

virtual Object* osg::Image::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osg::ImageStream.

virtual Object* osg::Image::clone (const CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osg::ImageStream.

virtual const char* osg::Image::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

Reimplemented in osg::ImageStream.

virtual const char* osg::Image::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osg::ImageStream.

virtual int osg::Image::compare (const Image & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Reimplemented in osg::ImageStream.

void osg::Image::setAllocationMode (AllocationMode *mode*) `[inline]`

Set the method used for deleting data once it goes out of scope.

AllocationMode osg::Image::getAllocationMode () const `[inline]`

Get the method used for deleting data once it goes out of scope.

void osg::Image::allocateImage (int *s*, int *t*, int *r*, GLenum *pixelFormat*, GLenum *type*, int *packing* = 1)

Allocate a pixel block of specified size and type.

void osg::Image::setImage (int *s*, int *t*, int *r*, GLint *internalTextureformat*, GLenum *pixelFormat*, GLenum *type*, unsigned char * *data*, AllocationMode *mode*, int *packing* = 1)

Set the image dimensions, format and data.

void osg::Image::readPixels (int *x*, int *y*, int *width*, int *height*, GLenum *pixelFormat*, GLenum *type*)

Read pixels from current frame buffer at specified position and size, using glReadPixels. Create memory for storage if required, reuse existing pixel coords if possible.

void osg::Image::readImageFromCurrentTexture (unsigned int *contextID*, bool *copyMipMapsIfAvailable*, GLenum *type* = GL_UNSIGNED_BYTE)

Read the contents of the current bound texture, handling compressed pixelFormats if present. Create memory for storage if required, reuse existing pixel coords if possible.

void osg::Image::scaleImage (int *s*, int *t*, int *r*) [inline]

Scale image to specified size.

void osg::Image::scaleImage (int *s*, int *t*, int *r*, GLenum *newDataType*)

Scale image to specified size and with specified data type.

void osg::Image::copySubImage (int *s_offset*, int *t_offset*, int *r_offset*, osg::Image * *source*)

Copy a source Image into a subpart of this Image at specified position. Typically used to copy to an already allocated image, such as creating a 3D image from a stack 2D images. If this Image is empty then image data is created to accomodate the source image in its offset position. If source is NULL then no operation happens, this Image is left unchanged.

void osg::Image::setOrigin (Origin *origin*) [inline]

Set the origin of the image. The default value is BOTTOM_LEFT and is consistent with OpenGL. TOP_-LEFT is used for imagery that follows standard Imagery convention, such as movies, and hasn't been flipped yet. For such images one much flip the t axis of the tex coords. to handle this origin position.

Origin osg::Image::getOrigin () const [inline]

Get the origin of the image.

int osg::Image::s () const `[inline]`

Width of image.

int osg::Image::t () const `[inline]`

Height of image.

int osg::Image::r () const `[inline]`

Depth of image.

unsigned int osg::Image::getPixelSizeInBits () const `[inline]`

Return the number of bits required for each pixel.

unsigned int osg::Image::getRowSizeInBytes () const `[inline]`

Return the number of bytes each row of pixels occupies once it has been packed.

unsigned int osg::Image::getImageSizeInBytes () const `[inline]`

Return the number of bytes each image (_s*_t) of pixels occupies.

unsigned int osg::Image::getTotalSizeInBytes () const `[inline]`

Return the number of bytes the whole row/image/volume of pixels occupies.

unsigned int osg::Image::getTotalSizeInBytesIncludingMipmaps () const

Return the number of bytes the whole row/image/volume of pixels occupies, including all mip maps if included.

bool osg::Image::valid () const `[inline]`

Return true if the Image represent a valid and usable imagery.

unsigned char* osg::Image::data () `[inline]`

Raw image data.

const unsigned char* osg::Image::data () const `[inline]`

Raw const image data.

Vec4 osg::Image::getColor (unsigned int *s*, unsigned *t* = 0, unsigned *r* = 0) const

Get the color value for specified texcoord.

Vec4 osg::Image::getColor (const Vec2 & *texcoord*) const [inline]

Get the color value for specified texcoord.

Vec4 osg::Image::getColor (const Vec3 & *texcoord*) const

Get the color value for specified texcoord.

void osg::Image::flipHorizontal ()

Flip the image horizontally.

void osg::Image::flipVertical ()

Flip the image vertically.

void osg::Image::ensureValidSizeForTexturing (GLint *maxTextureSize*)

Ensure image dimensions are a power of two. Mipmapped textures require the image dimensions to be power of two and are within the maxiumum texture size for the host machine.

void osg::Image::dirty () [inline]

Dirty the image, which increments the modified count, to force osg::Texture to reload the image.

void osg::Image::setModifiedCount (unsigned int *value*) [inline]

Set the modified count value. Used by osg::Texture when using texture subloading.

unsigned int osg::Image::getModifiedCount () const [inline]

Get modified count value. Used by osg::Texture when using texture subloading.

void osg::Image::setMipmapLevels (const MipmapDataType & *mipmapDataVector*) [inline]

Send offsets into data. It is assumed that first mipmap offset (index 0) is 0.

bool osg::Image::isImageTranslucent () const

Return true if this image is translucent - i.e. it has alpha values that are less 1.0 (when normalized).

void osg::Image::setPixelBufferObject (PixelBufferObject * *buffer*) `[inline]`

Set the optional PixelBufferObject used to map the image memory efficiently to graphics memory.

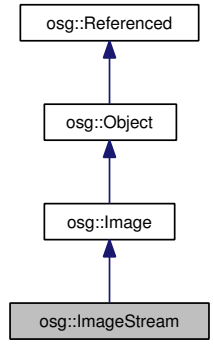
PixelBufferObject* osg::Image::getPixelBufferObject () `[inline]`

Get the PixelBufferObject.

const PixelBufferObject* osg::Image::getPixelBufferObject () const `[inline]`

Get the const PixelBufferObject.

4.161 osg::ImageStream Class Reference



Public Types

- enum **StreamStatus** {
  INVALID,
  PLAYING,
  PAUSED,
  REWINDING }
- enum **LoopingMode** {
  NO_LOOPING,
  LOOPING }

Public Member Functions

- ImageStream (const ImageStream &image, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual int compare (const Image &rhs) const
- virtual void **play** ()
- virtual void **pause** ()
- virtual void **rewind** ()
- virtual void **quit** (bool=true)
- StreamStatus **getStatus** ()
- void **setLoopingMode** (LoopingMode mode)
- LoopingMode **getLoopingMode** () const
- virtual double **getLength** () const
- virtual void **setReferenceTime** (double)
- virtual double **getReferenceTime** () const
- virtual void **setTimeMultiplier** (double)
- virtual double **getTimeMultiplier** () const
- virtual void **setVolume** (float)
- virtual float **getVolume** () const
- virtual void **update** ()

4.162 Detailed Description

Image Stream class.

4.163 Constructor & Destructor Documentation

osg::ImageStream::ImageStream (const ImageStream & *image*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.164 Member Function Documentation

virtual Object* osg::ImageStream::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Reimplemented from osg::Image.

virtual Object∗ osg::ImageStream::clone (const CopyOp &) const `[inline, virtual]`

Clone an object, with Object∗ return type. Must be defined by derived classes.

Reimplemented from osg::Image.

virtual const char∗ osg::ImageStream::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Reimplemented from osg::Image.

virtual const char∗ osg::ImageStream::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

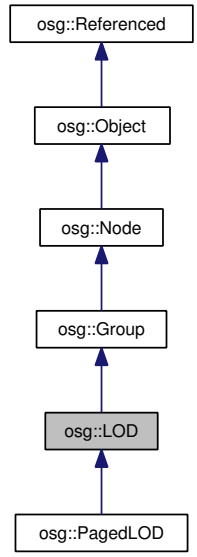
Reimplemented from osg::Image.

virtual int osg::ImageStream::compare (const Image & *rhs*) const `[virtual]`

Return -1 if ∗this < ∗rhs, 0 if ∗this==∗rhs, 1 if ∗this>∗rhs.

Reimplemented from osg::Image.

4.165 osg::LOD Class Reference



Public Types

- enum CenterMode {
  USE_BOUNDING_SPHERE_CENTER,
  USER_DEFINED_CENTER }
- enum RangeMode {
  DISTANCE_FROM_EYE_POINT,
  PIXEL_SIZE_ON_SCREEN }
- typedef std::pair< float, float > **MinMaxPair**
- typedef std::vector< MinMaxPair > **RangeList**

Public Member Functions

- LOD (const LOD &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, LOD)
- virtual void traverse (NodeVisitor &nv)
- virtual bool addChild (Node *child)
- virtual bool **addChild** (Node *child, float min, float max)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove=1)
- void setCenterMode (CenterMode mode)
- CenterMode getCenterMode () const
- void setCenter (const Vec3 &center)
- const Vec3 & getCenter () const
- void setRadius (float radius)
- float getRadius () const
- void setRangeMode (RangeMode mode)
- RangeMode getRangeMode () const
- void setRange (unsigned int childNo, float min, float max)
- float getMinRange (unsigned int childNo) const
- float getMaxRange (unsigned int childNo) const
- unsigned int getNumRanges () const
- void setRangeList (const RangeList &rangeList)
- const RangeList & getRangeList () const
- virtual BoundingSphere computeBound () const

4.166 Detailed Description

LOD - Level Of Detail group node which allows switching between children depending on distance from eye point. Typical uses are for load balancing - objects further away from the eye point are rendered at a lower level of detail, and at times of high stress on the graphics pipeline lower levels of detail can also be chosen by adjusting the viewers's Camera/CullSettings LODScale value. Each child has a corresponding valid range

consisting of a minimum and maximum distance. Given a distance to the viewer (d), LOD displays a child if min <= d < max. LOD may display multiple children simultaneously if their corresponding ranges overlap. Children can be in any order, and don't need to be sorted by range or amount of detail. If the number of ranges (m) is less than the number of children (n), then children m+1 through n are ignored.

4.167 Member Enumeration Documentation

enum osg::LOD::CenterMode

Modes which control how the center of object should be determined when computing which child is active.

enum osg::LOD::RangeMode

Modes that control how the range values should be intepreted when computing which child is active.

4.168 Constructor & Destructor Documentation

osg::LOD::LOD (const LOD &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.169 Member Function Documentation

virtual void osg::LOD::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::Group.

Reimplemented in osg::PagedLOD.

virtual bool osg::LOD::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented from osg::Group.

Reimplemented in osg::PagedLOD.

virtual bool osg::LOD::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove* = `1`) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented from osg::Group.

Reimplemented in osg::PagedLOD.

void osg::LOD::setCenterMode (CenterMode *mode*) `[inline]`

Set how the center of object should be determined when computing which child is active.

CenterMode osg::LOD::getCenterMode () const `[inline]`

Get how the center of object should be determined when computing which child is active.

void osg::LOD::setCenter (const Vec3 & *center*) `[inline]`

Sets the object-space point which defines the center of the osg::LOD. center is affected by any transforms in the hierarchy above the osg::LOD.

const Vec3& osg::LOD::getCenter () const `[inline]`

return the LOD center point.

void osg::LOD::setRadius (float *radius*) `[inline]`

Set the object-space reference radius of the volume enclosed by the LOD. Used to detmine the bounding sphere of the LOD in the absense of any children.

float osg::LOD::getRadius () const `[inline]`

Get the object-space radius of the volume enclosed by the LOD.

void osg::LOD::setRangeMode (RangeMode *mode*) `[inline]`

Set how the range values should be intepreted when computing which child is active.

RangeMode osg::LOD::getRangeMode () const `[inline]`

Get how the range values should be intepreted when computing which child is active.

void osg::LOD::setRange (unsigned int *childNo*, float *min*, float *max*)

Sets the min and max visible ranges of range of specifiec child. Values are floating point distance specified in local objects coordinates.

float osg::LOD::getMinRange (unsigned int *childNo*) const `[inline]`

returns the min visible range for specified child.

float osg::LOD::getMaxRange (unsigned int *childNo*) const `[inline]`

returns the max visible range for specified child.

unsigned int osg::LOD::getNumRanges () const `[inline]`

returns the number of ranges currently set. An LOD which has been fully set up will have getNumChildren()==getNumRanges().

void osg::LOD::setRangeList (const RangeList & *rangeList*) `[inline]`

set the list of MinMax ranges for each child.

const RangeList& osg::LOD::getRangeList () const `[inline]`

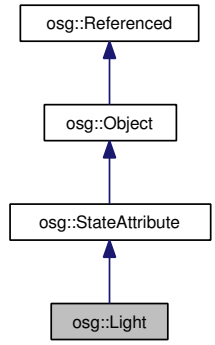
return the list of MinMax ranges for each child.

virtual BoundingSphere osg::LOD::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Group.

4.170 osg::Light Class Reference



Public Member Functions

- **Light** (unsigned int lightnum)
- Light (const Light &light, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &copyop) const
- virtual bool isSameKindAs (const osg::Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Type getType () const
- virtual int compare (const StateAttribute &sa) const
- virtual unsigned int getMember () const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void setLightNum (int num)
- int getLightNum () const
- void setAmbient (const Vec4 &ambient)
- const Vec4 & getAmbient () const
- void setDiffuse (const Vec4 &diffuse)
- const Vec4 & getDiffuse () const
- void setSpecular (const Vec4 &specular)
- const Vec4 & getSpecular () const
- void setPosition (const Vec4 &position)
- const Vec4 & getPosition () const
- void setDirection (const Vec3 &direction)
- const Vec3 & getDirection () const
- void setConstantAttenuation (float constant_attenuation)
- float getConstantAttenuation () const
- void setLinearAttenuation (float linear_attenuation)
- float getLinearAttenuation () const
- void setQuadraticAttenuation (float quadratic_attenuation)
- float getQuadraticAttenuation () const
- void setSpotExponent (float spot_exponent)
- float getSpotExponent () const
- void setSpotCutoff (float spot_cutoff)
- float getSpotCutoff () const
- void captureLightState ()
- virtual void apply (State &state) const

4.171 Detailed Description

Light state class which encapsulates OpenGL glLight() functionality.

4.172 Constructor & Destructor Documentation

osg::Light::Light (const Light & *light*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.173 Member Function Documentation

virtual osg::Object* osg::Light::cloneType () const `[inline, virtual]`

Clone the type of an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual osg::Object* osg::Light::clone (const osg::CopyOp &) const `[inline, virtual]`

Clone an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual bool osg::Light::isSameKindAs (const osg::Object * *obj*) const `[inline, virtual]`

Return true if this and obj are of the same kind of object.

Reimplemented from osg::StateAttribute.

virtual const char* osg::Light::libraryName () const `[inline, virtual]`

Return the name of the attribute's library.

Reimplemented from osg::StateAttribute.

virtual const char* osg::Light::className () const `[inline, virtual]`

Return the name of the attribute's class type.

Reimplemented from osg::StateAttribute.

virtual Type osg::Light::getType () const `[inline, virtual]`

Return the Type identifier of the attribute's class type.

Implements osg::StateAttribute.

virtual int osg::Light::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual unsigned int osg::Light::getMember () const `[inline, virtual]`

Return the member identifier within the attribute's class type. Used for light number/clip plane number etc.

Reimplemented from osg::StateAttribute.

virtual bool osg::Light::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

void osg::Light::setLightNum (int *num*)

Set which OpenGL light to operate on.

int osg::Light::getLightNum () const `[inline]`

Get which OpenGL light this osg::Light operates on.

void osg::Light::setAmbient (const Vec4 & *ambient*) `[inline]`

Set the ambient component of the light.

const Vec4& osg::Light::getAmbient () const `[inline]`

Get the ambient component of the light.

void osg::Light::setDiffuse (const Vec4 & *diffuse*) `[inline]`

Set the diffuse component of the light.

const Vec4& osg::Light::getDiffuse () const `[inline]`

Get the diffuse component of the light.

void osg::Light::setSpecular (const Vec4 & *specular*) `[inline]`

Set the specular component of the light.

const Vec4& osg::Light::getSpecular () const `[inline]`

Get the specular component of the light.

void osg::Light::setPosition (const Vec4 & ***position*****)** `[inline]`

Set the position of the light.

const Vec4& osg::Light::getPosition () const `[inline]`

Get the position of the light.

void osg::Light::setDirection (const Vec3 & ***direction*****)** `[inline]`

Set the direction of the light.

const Vec3& osg::Light::getDirection () const `[inline]`

Get the direction of the light.

void osg::Light::setConstantAttenuation (float ***constant_attenuation*****)** `[inline]`

Set the constant attenuation of the light.

float osg::Light::getConstantAttenuation () const `[inline]`

Get the constant attenuation of the light.

void osg::Light::setLinearAttenuation (float ***linear_attenuation*****)** `[inline]`

Set the linear attenuation of the light.

float osg::Light::getLinearAttenuation () const `[inline]`

Get the linear attenuation of the light.

void osg::Light::setQuadraticAttenuation (float ***quadratic_attenuation*****)** `[inline]`

Set the quadratic attenuation of the light.

float osg::Light::getQuadraticAttenuation () const `[inline]`

Get the quadratic attenuation of the light.

void osg::Light::setSpotExponent (float *spot_exponent*) `[inline]`

Set the spot exponent of the light.

float osg::Light::getSpotExponent () const `[inline]`

Get the spot exponent of the light.

void osg::Light::setSpotCutoff (float *spot_cutoff*) `[inline]`

Set the spot cutoff of the light.

float osg::Light::getSpotCutoff () const `[inline]`

Get the spot cutoff of the light.

void osg::Light::captureLightState ()

Capture the lighting settings of the current OpenGL state and store them in this object.

virtual void osg::Light::apply (State & *state*) const `[virtual]`

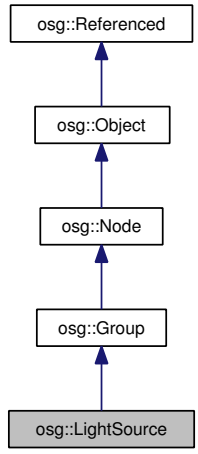
Apply the light's state to the OpenGL state machine.

Reimplemented from osg::StateAttribute.

void osg::Light::init () `[protected]`

Initialize the light's settings with some decent defaults.

4.174 osg::LightSource Class Reference



Public Types

- enum **ReferenceFrame** {
  RELATIVE_RF,
  ABSOLUTE_RF }

Public Member Functions

- LightSource (const LightSource &ls, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, LightSource)
- void setReferenceFrame (ReferenceFrame rf)
- ReferenceFrame **getReferenceFrame** () const
- void setLight (Light *light)
- Light * getLight ()
- const Light * getLight () const
- void setStateSetModes (StateSet &, StateAttribute::GLModeValue) const
- void setLocalStateSetModes (StateAttribute::GLModeValue value=StateAttribute::ON)
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual BoundingSphere computeBound () const

4.175 Detailed Description

Leaf Node for defining a light in the scene.

4.176 Constructor & Destructor Documentation

osg::LightSource::LightSource (const LightSource & *ls*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.177 Member Function Documentation

void osg::LightSource::setReferenceFrame (ReferenceFrame *rf*)

Set the light sources's ReferenceFrame, either to be relative to its parent reference frame, or relative to an absolute coordinate frame. RELATIVE_RF is the default. Note: setting the ReferenceFrame to be ABSOLUTE_RF will also set the CullingActive flag on the light source, and hence all of its parents, to false, thereby disabling culling of it and all its parents. This is necessary to prevent inappropriate culling, but may impact cull times if the absolute light source is deep in the scene graph. It is therefore recommended to only use absolute light source at the top of the scene.

void osg::LightSource::setLight (Light * *light*)

Set the attached light.

Light* osg::LightSource::getLight () `[inline]`

Get the attached light.

const Light* osg::LightSource::getLight () const `[inline]`

Get the const attached light.

void osg::LightSource::setStateSetModes (StateSet &, StateAttribute::GLModeValue) const

Set the GLModes on StateSet associated with the LightSource.

void osg::LightSource::setLocalStateSetModes (StateAttribute::GLModeValue *value* = `StateAttribute::ON`)

Set up the local StateSet.

virtual void osg::LightSource::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

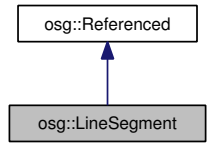
Reimplemented from osg::Group.

virtual BoundingSphere osg::LightSource::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Group.

4.178 osg::LineSegment Class Reference



Public Member Functions

- **LineSegment** (const LineSegment &seg)
- **LineSegment** (const Vec3 &s, const Vec3 &e)
- LineSegment & **operator=** (const LineSegment &seg)
- void **set** (const Vec3 &s, const Vec3 &e)
- Vec3 & **start** ()
- const Vec3 & **start** () const
- Vec3 & **end** ()
- const Vec3 & **end** () const
- bool **valid** () const
- bool intersect (const BoundingBox &bb) const
- bool intersect (const BoundingBox &bb, float &r1, float &r2) const
- bool intersect (const BoundingSphere &bs) const
- bool intersect (const BoundingSphere &bs, float &r1, float &r2) const
- bool intersect (const Vec3 &v1, const Vec3 &v2, const Vec3 &v3, float &r)
- void mult (const LineSegment &seg, const Matrix &m)
- void mult (const Matrix &m, const LineSegment &seg)

4.179 Detailed Description

LineSegment class for representing a line segment.

4.180 Member Function Documentation

bool osg::LineSegment::intersect (const BoundingBox & *bb*) const

return true if segment intersects BoundingBox.

bool osg::LineSegment::intersect (const BoundingBox & *bb*, float & *r1*, float & *r2*) const

return true if segment intersects BoundingBox and return the intersection ratios.

bool osg::LineSegment::intersect (const BoundingSphere & *bs*) const

return true if segment intersects BoundingSphere.

bool osg::LineSegment::intersect (const BoundingSphere & *bs*, float & *r1*, float & *r2*) const

return true if segment intersects BoundingSphere and return the intersection ratio.

bool osg::LineSegment::intersect (const Vec3 & *v1*, const Vec3 & *v2*, const Vec3 & *v3*, float & *r*)

return true if segment intersects triangle and set ratio long segment.

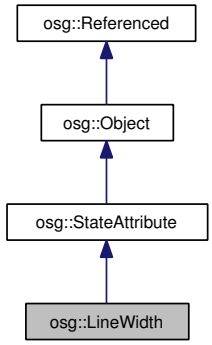
void osg::LineSegment::mult (const LineSegment & *seg*, const Matrix & *m*) `[inline]`

post multiply a segment by matrix.

void osg::LineSegment::mult (const Matrix & *m*, const LineSegment & *seg*) `[inline]`

pre multiply a segment by matrix.

4.181 osg::LineWidth Class Reference



Public Member Functions

- **LineWidth** (float width=1.0f)
- LineWidth (const LineWidth &lw, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, LineWidth, LINEWIDTH)
- virtual int compare (const StateAttribute &sa) const
- void **setWidth** (float width)
- float **getWidth** () const
- virtual void apply (State &state) const

4.182 Detailed Description

LineWidth - encapsulates the OpenGL glLineWidth for setting the width of lines in pixels.

4.183 Constructor & Destructor Documentation

osg::LineWidth::LineWidth (const LineWidth & *lw*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.184 Member Function Documentation

virtual int osg::LineWidth::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

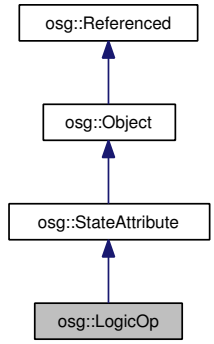
Implements osg::StateAttribute.

virtual void osg::LineWidth::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.185 osg::LogicOp Class Reference



Public Types

- enum **Opcode** {
  CLEAR,
  SET,
  COPY,
  COPY_INVERTED,
  NOOP,
  INVERT,
  AND,
  NAND,
  OR,
  NOR,
  XOR,
  EQUIV,
  AND_REVERSE,
  AND_INVERTED,
  OR_REVERSE,
  OR_INVERTED }

Public Member Functions

- **LogicOp** (Opcode opcode)
- LogicOp (const LogicOp &trans, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_StateAttribute** (osg, LogicOp, LOGICOP)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setOpcode** (Opcode opcode)
- Opcode **getOpcode** () const
- virtual void apply (State &state) const

4.186 Detailed Description

Encapsulates OpenGL LogicOp state.

4.187 Constructor & Destructor Documentation

osg::LogicOp::LogicOp (const LogicOp & *trans*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.188 Member Function Documentation

virtual int osg::LogicOp::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::LogicOp::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

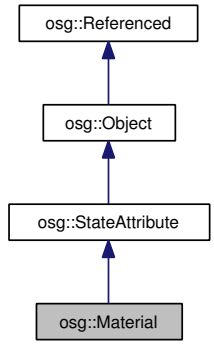
Reimplemented from osg::StateAttribute.

virtual void osg::LogicOp::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.189 osg::Material Class Reference



Public Types

- enum **Face** {
  FRONT,
  BACK,
  FRONT_AND_BACK }
- enum **ColorMode** {
  AMBIENT,
  DIFFUSE,
  SPECULAR,
  EMISSION,
  AMBIENT_AND_DIFFUSE,
  OFF }

Public Member Functions

- Material (const Material &mat, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Material, MATERIAL)
- virtual int compare (const StateAttribute &sa) const
- Material & **operator=** (const Material &rhs)
- virtual bool getModeUsage (StateAttribute::ModeUsage &) const
- virtual void apply (State &state) const
- void **setColorMode** (ColorMode mode)
- ColorMode **getColorMode** () const
- void **setAmbient** (Face face, const Vec4 &ambient)

- const Vec4 & **getAmbient** (Face face) const
- bool **getAmbientFrontAndBack** () const
- void **setDiffuse** (Face face, const Vec4 &diffuse)
- const Vec4 & **getDiffuse** (Face face) const
- bool **getDiffuseFrontAndBack** () const
- void setSpecular (Face face, const Vec4 &specular)
- const Vec4 & getSpecular (Face face) const
- bool getSpecularFrontAndBack () const
- void setEmission (Face face, const Vec4 &emission)
- const Vec4 & getEmission (Face face) const
- bool getEmissionFrontAndBack () const
- void setShininess (Face face, float shininess)
- float getShininess (Face face) const
- bool getShininessFrontAndBack () const
- void setTransparency (Face face, float trans)
- void setAlpha (Face face, float alpha)

4.190 Detailed Description

Material - encapsulates OpenGL glMaterial state.

4.191 Constructor & Destructor Documentation

osg::Material::Material (const Material & *mat*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.192 Member Function Documentation

virtual int osg::Material::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Material::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::Material::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

void osg::Material::setSpecular (Face *face*, const Vec4 & *specular*)

Set specular value of specified face(s) of the material, valid specular[0..3] range is 0.0 to 1.0.

const Vec4& osg::Material::getSpecular (Face *face*) const

Get the specular value for specified face.

bool osg::Material::getSpecularFrontAndBack () const `[inline]`

Return whether specular values are equal for front and back faces or not.

void osg::Material::setEmission (Face *face*, const Vec4 & *emission*)

Set emission value of specified face(s) of the material, valid emission[0..3] range is 0.0 to 1.0.

const Vec4& osg::Material::getEmission (Face *face*) const

Get the emission value for specified face.

bool osg::Material::getEmissionFrontAndBack () const `[inline]`

Return whether emission values are equal for front and back faces or not.

void osg::Material::setShininess (Face *face*, float *shininess*)

Set shininess of specified face(s) of the material. valid shininess range is 0.0 to 128.0.

float osg::Material::getShininess (Face *face*) const

Get the shininess value for specified face.

bool osg::Material::getShininessFrontAndBack () const `[inline]`

Return whether shininess values are equal for front and back faces or not.

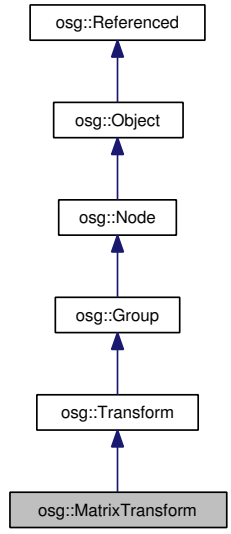
void osg::Material::setTransparency (Face *face*, float *trans*)

Set the alpha value of ambient, diffuse, specular and emission colors of specified face, to 1-transparency. Valid transparency range is 0.0 to 1.0.

void osg::Material::setAlpha (Face *face*, float *alpha*)

Set the alpha value of ambient, diffuse, specular and emission colors. Valid transparency range is 0.0 to 1.0.

4.193 osg::MatrixTransform Class Reference



Public Member Functions

- MatrixTransform (const MatrixTransform &, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **MatrixTransform** (const Matrix &matix)
- **META_Node** (osg, MatrixTransform)
- virtual MatrixTransform * **asMatrixTransform** ()
- virtual const MatrixTransform * **asMatrixTransform** () const
- void setMatrix (const Matrix &mat)
- const Matrix & getMatrix () const
- void preMult (const Matrix &mat)
- void postMult (const Matrix &mat)
- const Matrix & getInverseMatrix () const

- virtual bool **computeLocalToWorldMatrix** (Matrix &matrix, NodeVisitor *) const
- virtual bool **computeWorldToLocalMatrix** (Matrix &matrix, NodeVisitor *) const

4.194 Detailed Description

MatrixTransform - is a subclass of Transform which has an osg::Matrix which represents a 4x4 transformation of its children from local coordinates into the Transform's parent coordinates.

4.195 Constructor & Destructor Documentation

osg::MatrixTransform::MatrixTransform (const MatrixTransform &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.196 Member Function Documentation

void osg::MatrixTransform::setMatrix (const Matrix & *mat*) `[inline]`

Set the transform's matrix.

const Matrix& osg::MatrixTransform::getMatrix () const `[inline]`

Get the matrix.

void osg::MatrixTransform::preMult (const Matrix & *mat*) `[inline]`

pre multiply the transform's matrix.

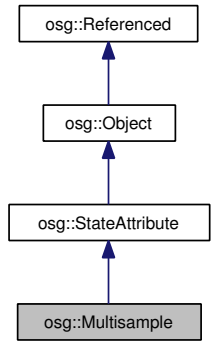
void osg::MatrixTransform::postMult (const Matrix & *mat*) `[inline]`

post multiply the transform's matrix.

const Matrix& osg::MatrixTransform::getInverseMatrix () const `[inline]`

Get the inverse matrix.

4.197 osg::Multisample Class Reference



Public Types

- enum **Mode** {
  FASTEST,
  NICEST,
  DONT_CARE }

Public Member Functions

- Multisample (const Multisample &trans, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Multisample, MULTISAMPLE)
- virtual int compare (const StateAttribute &sa) const
- void **setSampleCoverage** (float coverage, bool invert)
- void **setCoverage** (float coverage)
- float **getCoverage** () const
- void **setInvert** (bool invert)
- bool **getInvert** () const
- void **setHint** (Mode mode)
- Mode **getHint** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.198 Detailed Description

Multisample - encapsulates the OpenGL Multisample state.

4.199 Constructor & Destructor Documentation

osg::Multisample::Multisample (const Multisample & *trans*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.200 Member Function Documentation

virtual int osg::Multisample::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if ∗this < ∗rhs, 0 if ∗this==∗rhs, 1 if ∗this>∗rhs.

Implements osg::StateAttribute.

virtual void osg::Multisample::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

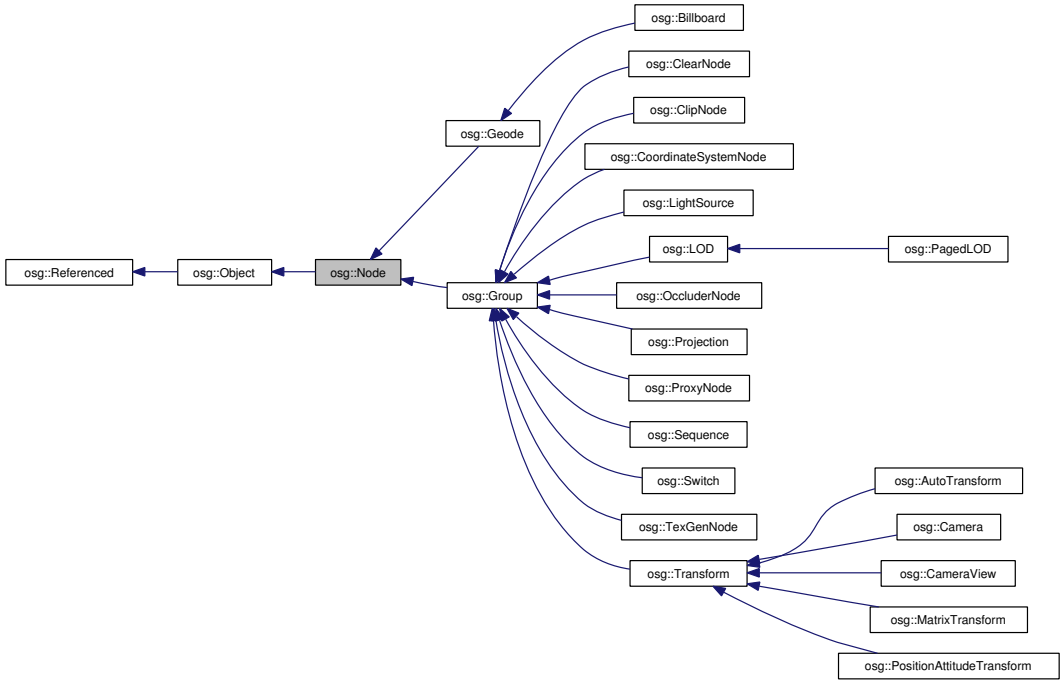
static Extensions∗ osg::Multisample::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Exentsion object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID..

static void osg::Multisample::setExtensions (unsigned int *contextID*, Extensions ∗ *extensions*) `[static]`

setExtensions allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

4.201 osg::Node Class Reference



Public Types

- typedef std::vector< Group * > ParentList
- typedef unsigned int **NodeMask**
- typedef std::vector< std::string > DescriptionList

Public Member Functions

- Node ()
- Node (const Node &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool isSameKindAs (const Object *obj) const

- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Group * asGroup ()
- virtual const Group * asGroup () const
- virtual Transform * asTransform ()
- virtual const Transform * asTransform () const
- virtual void accept (NodeVisitor &nv)
- virtual void ascend (NodeVisitor &nv)
- virtual void traverse (NodeVisitor &)
- const ParentList & getParents () const
- ParentList getParents ()
- Group * **getParent** (unsigned int i)
- const Group * getParent (unsigned int i) const
- unsigned int getNumParents () const
- NodePathList getParentalNodePaths (osg::Node *haltTraversalAtNode=0) const
- MatrixList getWorldMatrices (osg::Node *haltTraversalAtNode=0) const
- void setUpdateCallback (NodeCallback *nc)
- NodeCallback * getUpdateCallback ()
- const NodeCallback * getUpdateCallback () const
- unsigned int getNumChildrenRequiringUpdateTraversal () const
- void setEventCallback (NodeCallback *nc)
- NodeCallback * getEventCallback ()
- const NodeCallback * getEventCallback () const
- unsigned int getNumChildrenRequiringEventTraversal () const
- void setCullCallback (NodeCallback *nc)
- NodeCallback * getCullCallback ()
- const NodeCallback * getCullCallback () const
- void setCullingActive (bool active)
- bool getCullingActive () const
- unsigned int getNumChildrenWithCullingDisabled () const
- bool isCullingActive () const
- unsigned int getNumChildrenWithOccluderNodes () const
- bool containsOccluderNodes () const
- void setNodeMask (NodeMask nm)
- NodeMask getNodeMask () const
- void setDescriptions (const DescriptionList &descriptions)
- DescriptionList & getDescriptions ()
- const DescriptionList & getDescriptions () const
- const std::string & getDescription (unsigned int i) const
- std::string & getDescription (unsigned int i)
- unsigned int getNumDescriptions () const
- void addDescription (const std::string &desc)
- void setStateSet (osg::StateSet *stateset)
- osg::StateSet * getOrCreateStateSet ()
- osg::StateSet * getStateSet ()
- const osg::StateSet * getStateSet () const

- void setInitialBound (const osg::BoundingSphere &bsphere)
- const BoundingSphere & getInitialBound () const
- void dirtyBound ()
- const BoundingSphere & getBound () const
- virtual BoundingSphere computeBound () const
- void setComputeBoundingSphereCallback (ComputeBoundingSphereCallback *callback)
- ComputeBoundingSphereCallback * getComputeBoundingSphereCallback ()
- const ComputeBoundingSphereCallback * getComputeBoundingSphereCallback () const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual void resizeGLObjectBuffers (unsigned int)
- virtual void releaseGLObjects (osg::State *=0) const

Friends

- class osg::Group
- class osg::Drawable
- class osg::StateSet

Classes

- struct ComputeBoundingSphereCallback

4.202 Detailed Description

Base class for all internal nodes in the scene graph. Provides interface for most common node operations (Composite Pattern).

4.203 Member Typedef Documentation

typedef std::vector<Group*> osg::Node::ParentList

A vector of osg::Group pointers which is used to store the parent(s) of node.

typedef std::vector<std::string> osg::Node::DescriptionList

A vector of std::string's which are used to describe the object.

4.204 Constructor & Destructor Documentation

osg::Node::Node ()

Construct a node. Initialize the parent list to empty, node name to "" and bounding sphere dirty flag to true.

osg::Node::Node (const Node &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

virtual osg::Node::~Node () `[protected, virtual]`

Node destructor. Note, is protected so that Nodes cannot be deleted other than by being dereferenced and the reference count being zero (see osg::Referenced), preventing the deletion of nodes which are still in use. This also means that Nodes cannot be created on stack i.e Node node will not compile, forcing all nodes to be created on the heap i.e Node* node = new Node().

4.205 Member Function Documentation

virtual Object* osg::Node::cloneType () const `[inline, virtual]`

clone an object of the same type as the node.

Implements osg::Object.

Reimplemented in osg::AutoTransform.

virtual Object* osg::Node::clone (const CopyOp & *copyop*) const `[inline, virtual]`

return a clone of a node, with Object* return type.

Implements osg::Object.

Reimplemented in osg::AutoTransform.

virtual bool osg::Node::isSameKindAs (const Object * *obj*) const `[inline, virtual]`

return true if this and obj are of the same kind of object.

Reimplemented from osg::Object.

Reimplemented in osg::AutoTransform.

virtual const char* osg::Node::libraryName () const `[inline, virtual]`

return the name of the node's library.

Implements osg::Object.

Reimplemented in osg::AutoTransform.

virtual const char* osg::Node::className () const `[inline, virtual]`

return the name of the node's class type.

Implements osg::Object.

Reimplemented in osg::AutoTransform.

virtual Group* osg::Node::asGroup () `[inline, virtual]`

convert 'this' into a Group pointer if Node is a Group, otherwise return 0. Equivalent to dynamic_cast<Group*>(this).

Reimplemented in osg::Group.

virtual const Group* osg::Node::asGroup () const `[inline, virtual]`

convert 'const this' into a const Group pointer if Node is a Group, otherwise return 0. Equivalent to dynamic_cast<const Group*>(this).

Reimplemented in osg::Group.

virtual Transform* osg::Node::asTransform () `[inline, virtual]`

Convert 'this' into a Transform pointer if Node is a Transform, otherwise return 0. Equivalent to dynamic_cast<Transform*>(this).

Reimplemented in osg::Transform.

virtual const Transform* osg::Node::asTransform () const `[inline, virtual]`

convert 'const this' into a const Transform pointer if Node is a Transform, otherwise return 0. Equivalent to dynamic_cast<const Transform*>(this).

Reimplemented in osg::Transform.

virtual void osg::Node::accept (NodeVisitor & *nv*) `[virtual]`

Visitor Pattern : calls the apply method of a NodeVisitor with this node's type.

Reimplemented in osg::AutoTransform.

virtual void osg::Node::ascend (NodeVisitor & *nv*) `[virtual]`

Traverse upwards : calls parents' accept method with NodeVisitor.

virtual void osg::Node::traverse (NodeVisitor &) `[inline, virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented in osg::Group, osg::LOD, osg::PagedLOD, osg::ProxyNode, osg::Sequence, and osg::Switch.

const ParentList& osg::Node::getParents () const `[inline]`

Get the parent list of node.

ParentList osg::Node::getParents () `[inline]`

Get the a copy of parent list of node. A copy is returned to prevent modification of the parent list.

const Group* osg::Node::getParent (unsigned int *i*) const `[inline]`

Get a single const parent of node.

Parameters:

i index of the parent to get.

Returns:

the parent i.

unsigned int osg::Node::getNumParents () const `[inline]`

Get the number of parents of node.

Returns:

the number of parents of this node.

NodePathList osg::Node::getParentalNodePaths (osg::Node * *haltTraversalAtNode* = `0`) const

Get the list of node paths parent paths. The optional Node* haltTraversalAtNode allows the user to prevent traversal beyond a specifed node.

MatrixList osg::Node::getWorldMatrices (osg::Node * *haltTraversalAtNode* = `0`) const

Get the list of matrices that transform this node from local coordinates to world coordinates. The optional Node* haltTraversalAtNode allows the user to prevent traversal beyond a specifed node.

void osg::Node::setUpdateCallback (NodeCallback * *nc*)

Set update node callback, called during update traversal.

NodeCallback∗ osg::Node::getUpdateCallback () `[inline]`

Get update node callback, called during update traversal.

const NodeCallback∗ osg::Node::getUpdateCallback () const `[inline]`

Get const update node callback, called during update traversal.

unsigned int osg::Node::getNumChildrenRequiringUpdateTraversal () const `[inline]`

Get the number of Children of this node which require Update traversal, since they have an Update Callback attached to them or their children.

void osg::Node::setEventCallback (NodeCallback ∗ *nc*)

Set update node callback, called during update traversal.

NodeCallback∗ osg::Node::getEventCallback () `[inline]`

Get update node callback, called during update traversal.

const NodeCallback∗ osg::Node::getEventCallback () const `[inline]`

Get const update node callback, called during update traversal.

unsigned int osg::Node::getNumChildrenRequiringEventTraversal () const `[inline]`

Get the number of Children of this node which require Event traversal, since they have an Event Callback attached to them or their children.

void osg::Node::setCullCallback (NodeCallback ∗ *nc*) `[inline]`

Set cull node callback, called during cull traversal.

NodeCallback∗ osg::Node::getCullCallback () `[inline]`

Get cull node callback, called during cull traversal.

const NodeCallback∗ osg::Node::getCullCallback () const `[inline]`

Get const cull node callback, called during cull traversal.

void osg::Node::setCullingActive (bool *active*)

Set the view frustum/small feature culling of this node to be active or inactive. The default value is true for _cullingActive. Used as a guide to the cull traversal.

bool osg::Node::getCullingActive () const `[inline]`

Get the view frustum/small feature _cullingActive flag for this node. Used as a guide to the cull traversal.

unsigned int osg::Node::getNumChildrenWithCullingDisabled () const `[inline]`

Get the number of Children of this node which have culling disabled.

bool osg::Node::isCullingActive () const `[inline]`

Return true if this node can be culled by view frustum, occlusion or small feature culling during the cull traversal. Note, returns true only if no children have culling disabled, and the local _cullingActive flag is true.

unsigned int osg::Node::getNumChildrenWithOccluderNodes () const `[inline]`

Get the number of Children of this node which are or have OccluderNode's.

bool osg::Node::containsOccluderNodes () const

return true if this node is an OccluderNode or the subgraph below this node are OccluderNodes.

void osg::Node::setNodeMask (NodeMask *nm*) `[inline]`

Set the node mask.

NodeMask osg::Node::getNodeMask () const `[inline]`

Get the node Mask.

void osg::Node::setDescriptions (const DescriptionList & *descriptions*) `[inline]`

Set the description list of the node.

DescriptionList& osg::Node::getDescriptions () `[inline]`

Get the description list of the node.

const DescriptionList& osg::Node::getDescriptions () const `[inline]`

Get the const description list of the const node.

const std::string& osg::Node::getDescription (unsigned int *i*) const `[inline]`

Get a single const description of the const node.

std::string& osg::Node::getDescription (unsigned int *i*) `[inline]`

Get a single description of the node.

unsigned int osg::Node::getNumDescriptions () const `[inline]`

Get the number of descriptions of the node.

void osg::Node::addDescription (const std::string & *desc*) `[inline]`

Add a description string to the node.

void osg::Node::setStateSet (osg::StateSet * *stateset*)

Set the node's StateSet.

osg::StateSet* osg::Node::getOrCreateStateSet ()

return the node's StateSet, if one does not already exist create it set the node and return the newly created StateSet. This ensures that a valid StateSet is always returned and can be used directly.

osg::StateSet* osg::Node::getStateSet () `[inline]`

Return the node's StateSet. returns NULL if a stateset is not attached.

const osg::StateSet* osg::Node::getStateSet () const `[inline]`

Return the node's const StateSet. Returns NULL if a stateset is not attached.

void osg::Node::setInitialBound (const osg::BoundingSphere & *bsphere*) `[inline]`

Set the intial bounding volume to use when computing the overall bounding volume.

const BoundingSphere& osg::Node::getInitialBound () const `[inline]`

Set the intial bounding volume to use when computing the overall bounding volume.

void osg::Node::dirtyBound ()

Mark this node's bounding sphere dirty. Forcing it to be computed on the next call to getBound().

const BoundingSphere& osg::Node::getBound () const `[inline]`

Get the bounding sphere of node. Using lazy evaluation computes the bounding sphere if it is 'dirty'.

virtual BoundingSphere osg::Node::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented in osg::AutoTransform, osg::Billboard, osg::ClipNode, osg::Geode, osg::Group, osg::LightSource, osg::LOD, osg::OccluderNode, osg::ProxyNode, osg::Switch, and osg::Transform.

void osg::Node::setComputeBoundingSphereCallback (ComputeBoundingSphereCallback * *callback*) `[inline]`

Set the compute bound callback to override the default computeBound.

ComputeBoundingSphereCallback* osg::Node::getComputeBoundingSphereCallback () `[inline]`

Get the compute bound callback.

const ComputeBoundingSphereCallback* osg::Node::getComputeBoundingSphereCallback () const `[inline]`

Get the const compute bound callback.

virtual void osg::Node::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Referenced.

Reimplemented in osg::Geode, osg::Group, osg::LightSource, and osg::TexGenNode.

virtual void osg::Node::resizeGLObjectBuffers (unsigned *int*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Object.

Reimplemented in osg::Camera, osg::Geode, and osg::Group.

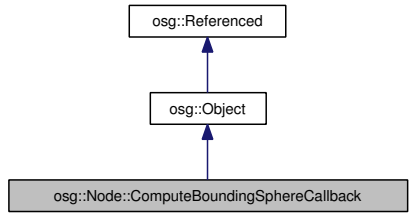
virtual void osg::Node::releaseGLObjects (osg::State * = 0) const `[virtual]`

If State is non-zero, this function releases any associated OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::Object.

Reimplemented in osg::Camera, osg::Geode, and osg::Group.

4.206 osg::Node::ComputeBoundingSphereCallback Struct Reference



Public Member Functions

- **ComputeBoundingSphereCallback** (const ComputeBoundingSphereCallback &, const CopyOp &)
- **META_Object** (osg, ComputeBoundingSphereCallback)
- virtual BoundingSphere **computeBound** (const osg::Node &) const

4.207 Detailed Description

Callback to allow users to override the default computation of bounding volume.

4.208 osg::NodeAcceptOp Struct Reference

Public Member Functions

- **NodeAcceptOp** (NodeVisitor &nv)
- void **operator()** (Node *node)
- void **operator()** (ref_ptr< Node > node)

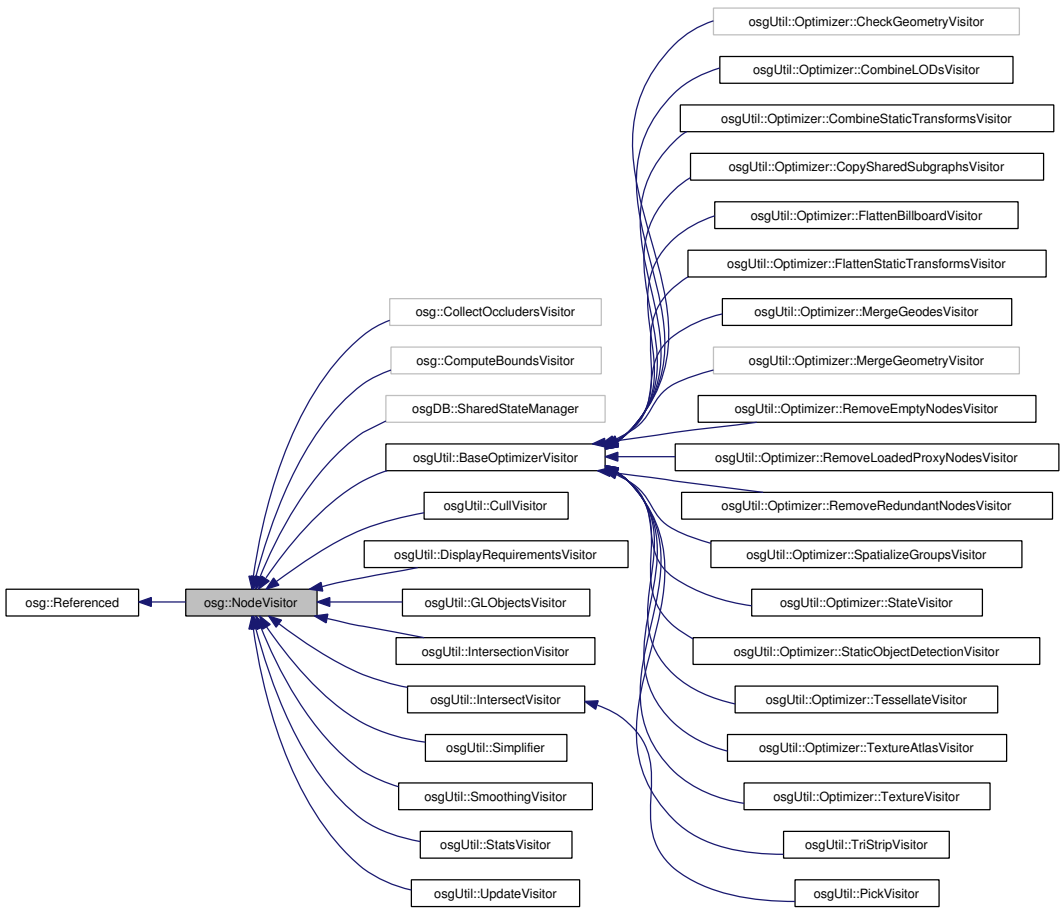
Public Attributes

- NodeVisitor & **_nv**

4.209 Detailed Description

Convenience functor for assisting visiting of arrays of osg::Node's.

4.210 osg::NodeVisitor Class Reference



Public Types

- enum **TraversalMode** {
  TRAVERSE_NONE,
  TRAVERSE_PARENTS,
  TRAVERSE_ALL_CHILDREN,
  TRAVERSE_ACTIVE_CHILDREN }
- enum **VisitorType** {
  NODE_VISITOR,
  UPDATE_VISITOR,
  EVENT_VISITOR,
  COLLECT_OCCLUDER_VISITOR,
  CULL_VISITOR }

Public Member Functions

- **NodeVisitor** (TraversalMode tm=TRAVERSE_NONE)
- **NodeVisitor** (VisitorType type, TraversalMode tm=TRAVERSE_NONE)
- virtual void reset ()
- void setVisitorType (VisitorType type)
- VisitorType getVisitorType () const
- void setTraversalNumber (int fn)
- int getTraversalNumber () const
- void setFrameStamp (FrameStamp *fs)
- const FrameStamp * getFrameStamp () const
- void setTraversalMask (Node::NodeMask mask)
- Node::NodeMask getTraversalMask () const
- void setNodeMaskOverride (Node::NodeMask mask)
- Node::NodeMask getNodeMaskOverride () const
- bool validNodeMask (const osg::Node &node) const
- void setTraversalMode (TraversalMode mode)
- TraversalMode getTraversalMode () const
- void setUserData (Referenced *obj)
- Referenced * getUserData ()
- const Referenced * getUserData () const
- void traverse (Node &node)
- void pushOntoNodePath (Node *node)
- void popFromNodePath ()
- NodePath & getNodePath ()
- const NodePath & getNodePath () const
- virtual osg::Vec3 getEyePoint () const

- virtual osg::Vec3 getViewPoint () const
- virtual float getDistanceToEyePoint (const Vec3 &, bool) const
- virtual float getDistanceFromEyePoint (const Vec3 &, bool) const
- virtual float getDistanceToViewPoint (const Vec3 &, bool) const
- virtual void **apply** (Node &node)
- virtual void **apply** (Geode &node)
- virtual void **apply** (Billboard &node)
- virtual void **apply** (Group &node)
- virtual void **apply** (ProxyNode &node)
- virtual void **apply** (Projection &node)
- virtual void **apply** (CoordinateSystemNode &node)
- virtual void **apply** (ClipNode &node)
- virtual void **apply** (TexGenNode &node)
- virtual void **apply** (LightSource &node)
- virtual void **apply** (Transform &node)
- virtual void **apply** (Camera &node)
- virtual void **apply** (CameraView &node)
- virtual void **apply** (MatrixTransform &node)
- virtual void **apply** (PositionAttitudeTransform &node)
- virtual void **apply** (Switch &node)
- virtual void **apply** (Sequence &node)
- virtual void **apply** (LOD &node)
- virtual void **apply** (PagedLOD &node)
- virtual void **apply** (ClearNode &node)
- virtual void **apply** (OccluderNode &node)
- void setDatabaseRequestHandler (DatabaseRequestHandler *handler)
- DatabaseRequestHandler * getDatabaseRequestHandler ()
- const DatabaseRequestHandler * getDatabaseRequestHandler () const

Classes

- class DatabaseRequestHandler

4.211 Detailed Description

Visitor for type safe operations on osg::Nodes. Based on GOF's Visitor pattern. The NodeVisitor is useful for developing type safe operations to nodes in the scene graph (as per Visitor pattern), and adds to this support for optional scene graph traversal to allow operations to be applied to whole scenes at once. The Visitor pattern uses a technique of double dispatch as a mechanism to call the appropriate apply(..) method of the NodeVisitor. To use this feature one must use the Node::accept(NodeVisitor) which is extended in each Node subclass, rather than the NodeVisitor apply directly. So use root->accept(myVisitor); instead of myVisitor.apply(*root). The later method will bypass the double dispatch and the appropriate NodeVisitor::apply(..) method will not be called.

4.212 Member Function Documentation

virtual void osg::NodeVisitor::reset () `[inline, virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented in osgUtil::GLObjectsVisitor, osgUtil::IntersectionVisitor, osgUtil::IntersectVisitor, osgUtil::Optimizer::StateVisitor, osgUtil::Optimizer::FlattenBillboardVisitor, osgUtil::Optimizer::TextureAtlasVisitor, osgUtil::StatsVisitor, and osgUtil::UpdateVisitor.

void osg::NodeVisitor::setVisitorType (VisitorType *type*) `[inline]`

Set the VisitorType, used to distinguish different visitors during traversal of the scene, typically used in the Node::traverse() method to select which behaviour to use for different types of traversal/visitors.

VisitorType osg::NodeVisitor::getVisitorType () const `[inline]`

Get the VisitorType.

void osg::NodeVisitor::setTraversalNumber (int *fn*) `[inline]`

Set the traversal number. Typically used to denote the frame count.

int osg::NodeVisitor::getTraversalNumber () const `[inline]`

Get the traversal number. Typically used to denote the frame count.

void osg::NodeVisitor::setFrameStamp (FrameStamp * *fs*) `[inline]`

Set the FrameStamp that this traversal is associated with.

const FrameStamp* osg::NodeVisitor::getFrameStamp () const `[inline]`

Get the FrameStamp that this traversal is associated with.

void osg::NodeVisitor::setTraversalMask (Node::NodeMask *mask*) `[inline]`

Set the TraversalMask of this NodeVisitor. The TraversalMask is used by the NodeVisitor::validNodeMask() method to determine whether to operate on a node and its subgraph. validNodeMask() is called automatically in the Node::accept() method before any call to NodeVisitor::apply(), apply() is only ever called if validNodeMask returns true. Note, if NodeVisitor::_traversalMask is 0 then all operations will be switched off for all nodes. Whereas setting both _traversalMask and _nodeMaskOverride to 0xffffffff will allow a visitor to work on all nodes regardless of their own Node::_nodeMask state.

Node::NodeMask osg::NodeVisitor::getTraversalMask () const `[inline]`

Get the TraversalMask.

void osg::NodeVisitor::setNodeMaskOverride (Node::NodeMask *mask*) `[inline]`

Set the NodeMaskOverride mask. Used in validNodeMask() to determine whether to operate on a node or its subgraph, by OR'ing NodeVisitor::_nodeMaskOverride with the Node's own Node::_nodeMask. Typically used to force on nodes which may have been switched off by their own Node::_nodeMask.

Node::NodeMask osg::NodeVisitor::getNodeMaskOverride () const `[inline]`

Get the NodeMaskOverride mask.

bool osg::NodeVisitor::validNodeMask (const osg::Node & *node*) const `[inline]`

Method to called by Node and its subclass' Node::accept() method, if the result is true it is used to cull operations of nodes and their subgraphs. Return true if the result of a bit wise and of the NodeVisitor::_traversalMask with the bit or between NodeVistor::_nodeMaskOverride and the Node::_nodeMask. default values for _traversalMask is 0xffffffff, _nodeMaskOverride is 0x0, and osg::Node::_nodeMask is 0xffffffff.

void osg::NodeVisitor::setTraversalMode (TraversalMode *mode*) `[inline]`

Set the traversal mode for Node::traverse() to use when deciding which children of a node to traverse. If a NodeVisitor has been attached via setTraverseVisitor() and the new mode is not TRAVERSE_VISITOR then the attached visitor is detached. Default mode is TRAVERSE_NONE.

TraversalMode osg::NodeVisitor::getTraversalMode () const `[inline]`

Get the traversal mode.

void osg::NodeVisitor::setUserData (Referenced * *obj*) `[inline]`

Set user data, data must be subclased from Referenced to allow automatic memory handling. If your own data isn't directly subclassed from Referenced then create an adapter object which points to your own objects and handles the memory addressing.

Referenced* osg::NodeVisitor::getUserData () `[inline]`

Get user data.

const Referenced* osg::NodeVisitor::getUserData () const `[inline]`

Get const user data.

void osg::NodeVisitor::traverse (Node & *node*) `[inline]`

Method for handling traversal of a nodes. If you intend to use the visitor for actively traversing the scene graph then make sure the accept() methods call this method unless they handle traversal directly.

void osg::NodeVisitor::pushOntoNodePath (Node * *node*) `[inline]`

Method called by osg::Node::accept() method before a call to the NodeVisitor::apply(..). The back of the list will, therefore, be the current node being visited inside the apply(..), and the rest of the list will be the parental sequence of nodes from the top most node applied down the graph to the current node. Note, the user does not typically call pushNodeOnPath() as it will be called automatically by the Node::accept() method.

void osg::NodeVisitor::popFromNodePath () `[inline]`

Method called by osg::Node::accept() method after a call to NodeVisitor::apply(..). Note, the user does not typically call popFromNodePath() as it will be called automatically by the Node::accept() method.

NodePath& osg::NodeVisitor::getNodePath () `[inline]`

Get the non const NodePath from the top most node applied down to the current Node being visited.

const NodePath& osg::NodeVisitor::getNodePath () const `[inline]`

Get the const NodePath from the top most node applied down to the current Node being visited.

virtual osg::Vec3 osg::NodeVisitor::getEyePoint () const `[inline, virtual]`

Get the eye point in local coordinates. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement.

Reimplemented in osgUtil::CullVisitor, and osgUtil::IntersectVisitor.

virtual osg::Vec3 osg::NodeVisitor::getViewPoint () const `[inline, virtual]`

Get the view point in local coordinates. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement.

Reimplemented in osgUtil::CullVisitor.

virtual float osg::NodeVisitor::getDistanceToEyePoint (const Vec3 &, bool) const `[inline, virtual]`

Get the distance from a point to the eye point, distance value in local coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceFromEyePoint(pos) is not implemented then a default value of 0.0 is returned.

Reimplemented in osgUtil::CullVisitor, and osgUtil::IntersectVisitor.

virtual float osg::NodeVisitor::getDistanceFromEyePoint (const Vec3 &, bool) const `[inline, virtual]`

Get the distance of a point from the eye point, distance value in the eye coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceFromEyePoint(pos) is not implemented than a default value of 0.0 is returned.

Reimplemented in osgUtil::CullVisitor.

virtual float osg::NodeVisitor::getDistanceToViewPoint (const Vec3 &, bool) const `[inline, virtual]`

Get the distance from a point to the view point, distance value in local coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceToViewPoint(pos) is not implemented then a default value of 0.0 is returned.

Reimplemented in osgUtil::CullVisitor.

void osg::NodeVisitor::setDatabaseRequestHandler (DatabaseRequestHandler * *handler*) `[inline]`

Set the handler for database requests.

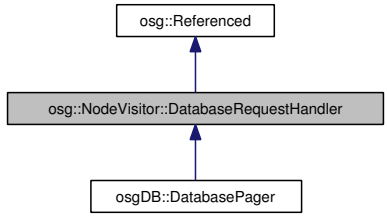
DatabaseRequestHandler* osg::NodeVisitor::getDatabaseRequestHandler () `[inline]`

Get the handler for database requests.

const DatabaseRequestHandler* osg::NodeVisitor::getDatabaseRequestHandler () const `[inline]`

Get the const handler for database requests.

4.213 osg::NodeVisitor::DatabaseRequestHandler Class Reference



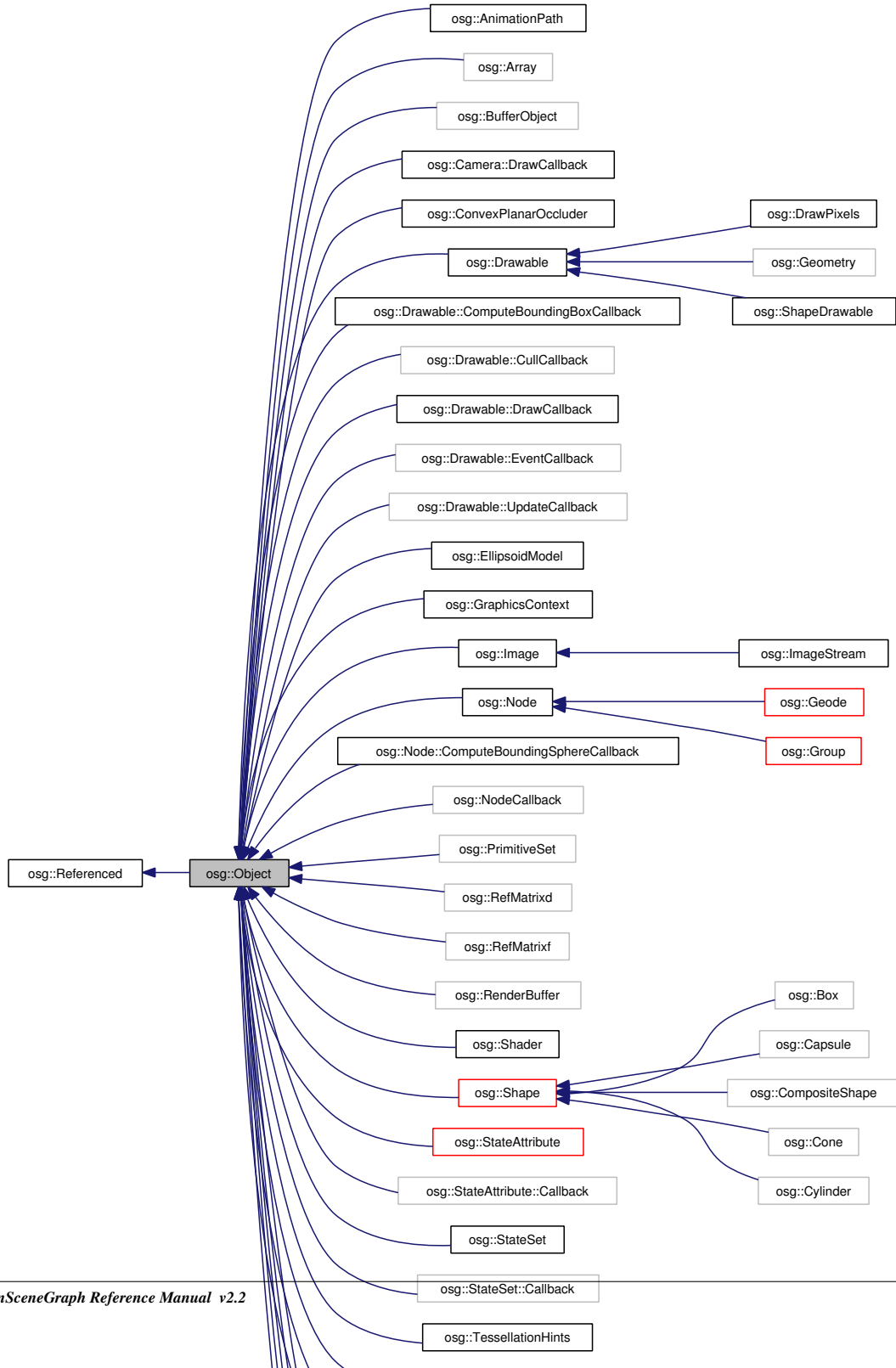
Public Member Functions

- virtual void **requestNodeFile** (const std::string &fileName, osg::Group *group, float priority, const FrameStamp *framestamp)=0

4.214 Detailed Description

Callback for managing database paging, such as generated by PagedLOD nodes.

4.215 osg::Object Class Reference



Public Types

- enum **DataVariance** {
  DYNAMIC,
  STATIC,
  UNSPECIFIED }

Public Member Functions

- Object ()
- **Object** (bool threadSafeRefUnref)
- Object (const Object &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const =0
- virtual Object * clone (const CopyOp &) const =0
- virtual bool **isSameKindAs** (const Object *) const
- virtual const char * libraryName () const =0
- virtual const char * className () const =0
- void setName (const std::string &name)
- void setName (const char *name)
- const std::string & getName () const
- void setDataVariance (DataVariance dv)
- DataVariance getDataVariance () const
- virtual void computeDataVariance ()
- void setUserData (Referenced *obj)
- Referenced * getUserData ()
- const Referenced * getUserData () const
- virtual void resizeGLObjectBuffers (unsigned int)
- virtual void releaseGLObjects (osg::State *=0) const

4.216 Detailed Description

Base class/standard interface for objects which require IO support, cloning and reference counting. Based on GOF Composite, Prototype and Template Method patterns.

4.217 Constructor & Destructor Documentation

osg::Object::Object () `[inline]`

Construct an object. Note Object is a pure virtual base class and therefore cannot be constructed on its own, only derived classes which override the clone and className methods are concrete classes and can be constructed.

osg::Object::Object (const Object &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor, optional CopyOp object can be used to control shallow vs deep copying of dynamic data.

virtual osg::Object::~Object () `[inline, protected, virtual]`

Object destructor. Note, is protected so that Objects cannot be deleted other than by being dereferenced and the reference count being zero (see osg::Referenced), preventing the deletion of nodes which are still in use. This also means that Nodes cannot be created on stack i.e Node node will not compile, forcing all nodes to be created on the heap i.e Node* node = new Node().

4.218 Member Function Documentation

virtual Object* osg::Object::cloneType () const `[pure virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implemented in osg::AutoTransform, osg::ClipPlane, osg::DrawPixels, osg::GraphicsContext, osg::Image, osg::ImageStream, osg::Light, osg::Node, osg::Shape, osg::ShapeDrawable, osg::StateAttribute, osg::StateSet, osg::Texture, osgUtil::PositionalStateContainer, osgUtil::RenderBin, and osgUtil::RenderStage.

virtual Object* osg::Object::clone (const CopyOp &) const `[pure virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implemented in osg::AutoTransform, osg::ClipPlane, osg::DrawPixels, osg::GraphicsContext, osg::Image, osg::ImageStream, osg::Light, osg::Node, osg::Shape, osg::ShapeDrawable, osg::StateAttribute, osg::StateSet, osg::Texture, osgUtil::PositionalStateContainer, osgUtil::RenderBin, and osgUtil::RenderStage.

virtual const char* osg::Object::libraryName () const `[pure virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implemented in osg::AutoTransform, osg::ClipPlane, osg::Drawable, osg::DrawPixels, osg::GraphicsContext, osg::Image, osg::ImageStream, osg::Light, osg::Node, osg::Shape, osg::ShapeDrawable, osg::StateAttribute, osg::StateSet, osg::Texture, osgDB::Archive, osgUtil::PositionalStateContainer, and osgUtil::RenderBin.

virtual const char* osg::Object::className () const `[pure virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implemented in osg::AutoTransform, osg::ClipPlane, osg::Drawable, osg::DrawPixels, osg::GraphicsContext, osg::Image, osg::ImageStream, osg::Light, osg::Node, osg::Shape, osg::ShapeDrawable, osg::StateAttribute, osg::StateSet, osg::Texture, osgDB::Archive, osgUtil::PositionalStateContainer, osgUtil::RenderBin, and osgUtil::RenderStage.

void osg::Object::setName (const std::string & *name*) `[inline]`

Set the name of object using C++ style string.

Reimplemented in osg::Uniform.

void osg::Object::setName (const char * *name*) `[inline]`

Set the name of object using a C style string.

const std::string& osg::Object::getName () const `[inline]`

Get the name of object.

void osg::Object::setDataVariance (DataVariance *dv*) `[inline]`

Set the data variance of this object. Can be set to either STATIC for values that do not change over the lifetime of the object, or DYNAMIC for values that vary over the lifetime of the object. The DataVariance value can be used by routines such as optimzation codes that wish to share static data. UNSPECIFIED is used to sepecify that the DataVariance hasn't been set yet.

DataVariance osg::Object::getDataVariance () const `[inline]`

Get the data variance of this object.

virtual void osg::Object::computeDataVariance () `[inline, virtual]`

Compute the DataVariance based on an assestment of callback etc.

Reimplemented in osg::Drawable, and osg::StateSet.

void osg::Object::setUserData (Referenced * *obj*) `[inline]`

Set user data, data must be subclassed from Referenced to allow automatic memory handling. If your own data isn't directly subclassed from Referenced then create an adapter object which points to your own object and handles the memory addressing.

Referenced* osg::Object::getUserData () `[inline]`

Get user data.

const Referenced* osg::Object::getUserData () const `[inline]`

Get const user data.

virtual void osg::Object::resizeGLObjectBuffers (unsigned int) `[inline, virtual]`

Resize any per context GLObject buffers to specified size.

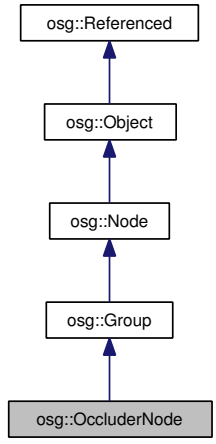
Reimplemented in osg::Camera, osg::Drawable, osg::FragmentProgram, osg::Geode, osg::Group, osg::Node, osg::Program, osg::Shader, osg::StateAttribute, osg::StateSet, osg::Texture, and osg::VertexProgram.

virtual void osg::Object::releaseGLObjects (osg::State * = 0) const `[inline, virtual]`

If State is non-zero, this function releases any associated OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented in osg::Camera, osg::Drawable, osg::FragmentProgram, osg::Geode, osg::Group, osg::Node, osg::Program, osg::Shader, osg::StateAttribute, osg::StateSet, osg::Texture, and osg::VertexProgram.

4.219 osg::OccluderNode Class Reference



Public Member Functions

- OccluderNode (const OccluderNode &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, OccluderNode)
- void setOccluder (ConvexPlanarOccluder *occluder)
- ConvexPlanarOccluder * getOccluder ()

- const ConvexPlanarOccluder * getOccluder () const
- virtual BoundingSphere computeBound () const

4.220 Detailed Description

OccluderNode is a Group node which allows OccluderNodeing between children. Typical uses would be for objects which might need to be rendered differently at different times, for instance a OccluderNode could be used to represent the different states of a traffic light.

4.221 Constructor & Destructor Documentation

osg::OccluderNode::OccluderNode (const OccluderNode &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.222 Member Function Documentation

void osg::OccluderNode::setOccluder (ConvexPlanarOccluder * *occluder*) `[inline]`

Attach a ConvexPlanarOccluder to an OccluderNode.

ConvexPlanarOccluder* osg::OccluderNode::getOccluder () `[inline]`

Get the ConvexPlanarOccluder* attached to a OccluderNode.

const ConvexPlanarOccluder* osg::OccluderNode::getOccluder () const `[inline]`

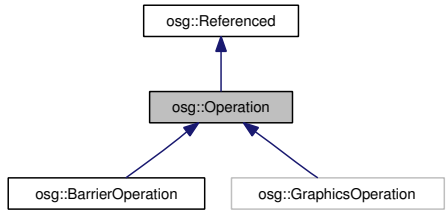
Get the const ConvexPlanarOccluder* attached to a OccluderNode.

virtual BoundingSphere osg::OccluderNode::computeBound () const `[virtual]`

Overrides Group's computeBound.

Reimplemented from osg::Group.

4.223 osg::Operation Class Reference



Public Member Functions

- **Operation** (const std::string &name, bool keep)
- void setName (const std::string &name)
- const std::string & getName () const
- void setKeep (bool keep)
- bool getKeep () const
- virtual void release ()
- virtual void operator() (Object *)=0

4.224 Detailed Description

Base class for implementing graphics operations.

4.225 Member Function Documentation

void osg::Operation::setName (const std::string & *name*) `[inline]`

Set the human readable name of the operation.

const std::string& osg::Operation::getName () const `[inline]`

Get the human readable name of the operation.

void osg::Operation::setKeep (bool *keep*) `[inline]`

Set whether the operation should be kept once its been applied.

bool osg::Operation::getKeep () const `[inline]`

Get whether the operation should be kept once its been applied.

virtual void osg::Operation::release () `[inline, virtual]`

if this operation is a barrier then release it.

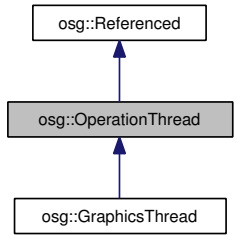
Reimplemented in osg::BarrierOperation.

virtual void osg::Operation::operator() (Object ∗) `[pure virtual]`

Do the actual task of this operation.

Implemented in osg::BarrierOperation.

4.226 osg::OperationThread Class Reference



Public Member Functions

- void **setParent** (Object ∗parent)
- Object ∗ **getParent** ()
- const Object ∗ **getParent** () const
- void setOperationQueue (OperationQueue ∗opq)
- OperationQueue ∗ getOperationQueue ()
- const OperationQueue ∗ getOperationQueue () const
- void add (Operation ∗operation)
- void remove (Operation ∗operation)
- void remove (const std::string &name)
- void removeAllOperations ()
- osg::ref_ptr< Operation > getCurrentOperation ()
- virtual void run ()
- void **setDone** (bool done)
- bool **getDone** () const
- virtual int cancel ()

4.227 Detailed Description

OperationThread is a helper class for running Operation within a single thread.

4.228 Member Function Documentation

void osg::OperationThread::setOperationQueue (OperationQueue * *opq*)

Set the OperationQueue.

OperationQueue* osg::OperationThread::getOperationQueue () [inline]

Get the OperationQueue.

const OperationQueue* osg::OperationThread::getOperationQueue () const [inline]

Get the const OperationQueue.

void osg::OperationThread::add (Operation * *operation*)

Add operation to end of OperationQueue, this will be executed by the graphics thread once this operation gets to the head of the queue.

void osg::OperationThread::remove (Operation * *operation*)

Remove operation from OperationQueue.

void osg::OperationThread::remove (const std::string & *name*)

Remove named operation from OperationQueue.

void osg::OperationThread::removeAllOperations ()

Remove all operations from OperationQueue.

osg::ref_ptr<Operation> osg::OperationThread::getCurrentOperation () [inline]

Get the operation currently being run.

virtual void osg::OperationThread::run () [virtual]

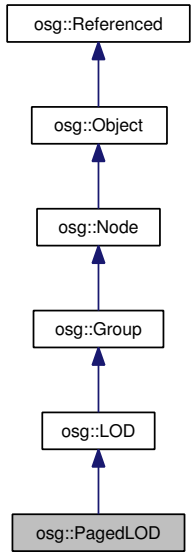
Run does the opertion thread run loop.

Reimplemented in osg::GraphicsThread.

virtual int osg::OperationThread::cancel () `[virtual]`

Cancel this graphics thread.

4.229 osg::PagedLOD Class Reference



Public Types

- typedef std::vector< PerRangeData > **PerRangeDataList**

Public Member Functions

- PagedLOD (const PagedLOD &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, PagedLOD)
- virtual void traverse (NodeVisitor &nv)
- virtual bool addChild (Node *child)
- virtual bool **addChild** (Node *child, float min, float max)
- virtual bool **addChild** (Node *child, float min, float max, const std::string &filename, float priorityOffset=0.0f, float priorityScale=1.0f)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove=1)

- void setDatabasePath (const std::string &path)
- const std::string & getDatabasePath () const
- void **setFileName** (unsigned int childNo, const std::string &filename)
- const std::string & **getFileName** (unsigned int childNo) const
- unsigned int **getNumFileNames** () const
- void **setPriorityOffset** (unsigned int childNo, float priorityOffset)
- float **getPriorityOffset** (unsigned int childNo) const
- unsigned int **getNumPriorityOffsets** () const
- void **setPriorityScale** (unsigned int childNo, float priorityScale)
- float **getPriorityScale** (unsigned int childNo) const
- unsigned int **getNumPriorityScales** () const
- void **setTimeStamp** (unsigned int childNo, double timeStamp)
- double **getTimeStamp** (unsigned int childNo) const
- unsigned int **getNumTimeStamps** () const
- void setFrameNumberOfLastTraversal (int frameNumber)
- int getFrameNumberOfLastTraversal () const
- void setNumChildrenThatCannotBeExpired (unsigned int num)
- unsigned int getNumChildrenThatCannotBeExpired () const
- virtual bool removeExpiredChildren (double expiryTime, NodeList &removedChildren)

Classes

- struct **PerRangeData**

4.230 Detailed Description

PagedLOD.

4.231 Constructor & Destructor Documentation

osg::PagedLOD::PagedLOD (const PagedLOD &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.232 Member Function Documentation

virtual void osg::PagedLOD::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::LOD.

virtual bool osg::PagedLOD::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented from osg::LOD.

virtual bool osg::PagedLOD::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove* = 1) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented from osg::LOD.

void osg::PagedLOD::setDatabasePath (const std::string & *path*)

Set the database path to prepend to children's filenames.

const std::string& osg::PagedLOD::getDatabasePath () const `[inline]`

Get the database path used to prepend to children's filenames.

void osg::PagedLOD::setFrameNumberOfLastTraversal (int *frameNumber*) `[inline]`

Set the frame number of the last time that this PageLOD node was traversed. Note, this frame number is automatically set by the traverse() method for all traversals (update, cull etc.).

int osg::PagedLOD::getFrameNumberOfLastTraversal () const `[inline]`

Get the frame number of the last time that this PageLOD node was traversed.

void osg::PagedLOD::setNumChildrenThatCannotBeExpired (unsigned int *num*) `[inline]`

Set the number of children that the PagedLOD must keep around, even if they are older than their expiry time.

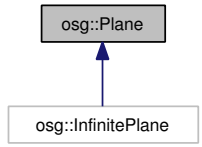
unsigned int osg::PagedLOD::getNumChildrenThatCannotBeExpired () const `[inline]`

Get the number of children that the PagedLOD must keep around, even if they are older than their expiry time.

virtual bool osg::PagedLOD::removeExpiredChildren (double *expiryTime*, NodeList & *removedChildren*) `[virtual]`

Remove the children from the PagedLOD which haven't been visited since specified expiry time. The removed children are added to the removeChildren list passed into the method, this allows the children to be deleted later at the caller's discretion. Return true if children are removed, false otherwise.

4.233 osg::Plane Class Reference



Public Types

- enum { **num_components** }
- typedef double value_type
- typedef Vec3d **Vec3_type**
- typedef Vec4d **Vec4_type**

Public Member Functions

- **Plane** (const Plane &pl)
- **Plane** (value_type a, value_type b, value_type c, value_type d)
- **Plane** (const Vec4f &vec)
- **Plane** (const Vec4d &vec)
- **Plane** (const Vec3_type &norm, value_type d)
- **Plane** (const Vec3_type &v1, const Vec3_type &v2, const Vec3_type &v3)
- **Plane** (const Vec3_type &norm, const Vec3_type &point)
- Plane & **operator=** (const Plane &pl)
- void **set** (const Plane &pl)
- void **set** (value_type a, value_type b, value_type c, value_type d)
- void **set** (const Vec4f &vec)
- void **set** (const Vec4d &vec)
- void **set** (const Vec3_type &norm, double d)
- void **set** (const Vec3_type &v1, const Vec3_type &v2, const Vec3_type &v3)
- void **set** (const Vec3_type &norm, const Vec3_type &point)
- void flip ()
- void **makeUnitLength** ()
- void calculateUpperLowerBBCorners ()
- bool **valid** () const

- bool **isNaN** () const
- bool **operator==** (const Plane &plane) const
- bool **operator!=** (const Plane &plane) const
- bool **operator**< (const Plane &plane) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- Vec4_type **asVec4** () const
- value_type & **operator[ ]** (unsigned int i)
- value_type **operator[ ]** (unsigned int i) const
- Vec3_type **getNormal** () const
- float distance (const osg::Vec3f &v) const
- double **distance** (const osg::Vec3d &v) const
- float dotProductNormal (const osg::Vec3f &v) const
- double dotProductNormal (const osg::Vec3d &v) const
- int intersect (const std::vector< Vec3 > &vertices) const
- int intersect (const std::vector< Vec3d > &vertices) const
- int intersect (const BoundingSphere &bs) const
- int intersect (const BoundingBox &bb) const
- void transform (const osg::Matrix &matrix)
- void transformProvidingInverse (const osg::Matrix &matrix)

4.234 Detailed Description

A plane class. It can be used to represent an infinite plane.

4.235 Member Typedef Documentation

typedef double osg::Plane::value_type

Type of Plane class.

4.236 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.237 Member Function Documentation

void osg::Plane::flip () `[inline]`

flip/reverse the orientation of the plane.

void osg::Plane::calculateUpperLowerBBCorners () `[inline]`

calculate the upper and lower bounding box corners to be used in the intersect(BoundingBox&) method for speeding calculations.

float osg::Plane::distance (const osg::Vec3f & *v*) const `[inline]`

calculate the distance between a point and the plane.

float osg::Plane::dotProductNormal (const osg::Vec3f & *v*) const `[inline]`

calculate the dot product of the plane normal and a point.

double osg::Plane::dotProductNormal (const osg::Vec3d & *v*) const `[inline]`

calculate the dot product of the plane normal and a point.

int osg::Plane::intersect (const std::vector< Vec3 > & *vertices*) const `[inline]`

intersection test between plane and vertex list return 1 if the bs is completely above plane, return 0 if the bs intersects the plane, return -1 if the bs is completely below the plane.

int osg::Plane::intersect (const std::vector< Vec3d > & *vertices*) const `[inline]`

intersection test between plane and vertex list return 1 if the bs is completely above plane, return 0 if the bs intersects the plane, return -1 if the bs is completely below the plane.

int osg::Plane::intersect (const BoundingSphere & *bs*) const `[inline]`

intersection test between plane and bounding sphere. return 1 if the bs is completely above plane, return 0 if the bs intersects the plane, return -1 if the bs is completely below the plane.

int osg::Plane::intersect (const BoundingBox & *bb*) const `[inline]`

intersection test between plane and bounding sphere. return 1 if the bs is completely above plane, return 0 if the bs intersects the plane, return -1 if the bs is completely below the plane.

void osg::Plane::transform (const osg::Matrix & *matrix*) `[inline]`

Transform the plane by matrix. Note, this operation carries out the calculation of the inverse of the matrix since a plane must be multiplied by the inverse transposed to transform it. This make this operation expensive. If the inverse has been already calculated elsewhere then use transformProvidingInverse() instead. See http://www.worldserver.com/turk/computergraphics/NormalTransformations.pdf

void osg::Plane::transformProvidingInverse (const osg::Matrix & *matrix*) `[inline]`

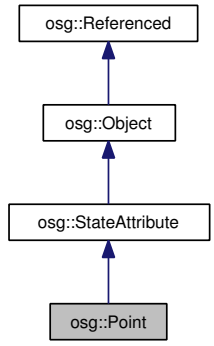
Transform the plane by providing a pre inverted matrix. see transform for details.

4.238 Member Data Documentation

value_type osg::Plane::_fv[4] `[protected]`

Vec member varaible.

4.239 osg::Point Class Reference



Public Member Functions

- Point (const Point &point, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Point, POINT)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setSize** (float size)
- float **getSize** () const
- void **setFadeThresholdSize** (float fadeThresholdSize)
- float **getFadeThresholdSize** () const
- void **setDistanceAttenuation** (const Vec3 &distanceAttenuation)
- const Vec3 & **getDistanceAttenuation** () const
- void **setMinSize** (float minSize)
- float **getMinSize** () const
- void **setMaxSize** (float maxSize)
- float **getMaxSize** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.240 Detailed Description

Point - encapsulates the OpenGL point smoothing and size state.

4.241 Constructor & Destructor Documentation

osg::Point::Point (const Point & *point*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.242 Member Function Documentation

virtual int osg::Point::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Point::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::Point::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

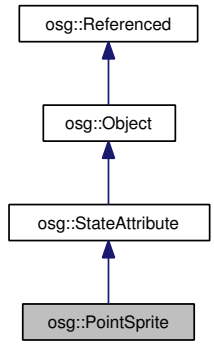
static Extensions* osg::Point::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Returns the Extensions object for the given context. If createIfNotInitalized is true and the Exentsions object doesn't exist, getExtensions() creates it on the given context. Returns NULL if createIfNotInitalized is false and the Extensions object doesn't exist.

static void osg::Point::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

setExtensions() allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes, but need to ensure that they all use the same low common denominator extensions.

4.243 osg::PointSprite Class Reference



Public Types

- enum **CoordOriginMode** {
  UPPER_LEFT,
  LOWER_LEFT }

Public Member Functions

- PointSprite (const PointSprite &ps, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, PointSprite, POINTSPRITE)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- virtual bool checkValidityOfAssociatedModes (osg::State &) const

- virtual bool isTextureAttribute () const
- virtual void apply (osg::State &state) const
- void **setCoordOriginMode** (CoordOriginMode mode)
- CoordOriginMode **getCoordOriginMode** () const

Static Public Member Functions

- static bool **isPointSpriteSupported** (unsigned int context)

4.244 Detailed Description

PointSprite base class which encapsulates enabling of point sprites .

4.245 Constructor & Destructor Documentation

osg::PointSprite::PointSprite (const PointSprite & *ps*, const osg::CopyOp & *copyop* = `osg::CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.246 Member Function Documentation

virtual int osg::PointSprite::compare (const StateAttribute & *sa*) const `[virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::PointSprite::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual bool osg::PointSprite::checkValidityOfAssociatedModes (osg::State &) const `[virtual]`

Check the modes associated with this StateAttribute are supported by current OpenGL drivers, and if not set the associated mode in osg::State to be black listed/invalid. Return true if all associated modes are valid.

Reimplemented from osg::StateAttribute.

virtual bool osg::PointSprite::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

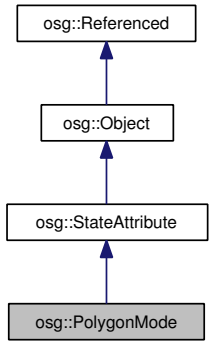
Reimplemented from osg::StateAttribute.

virtual void osg::PointSprite::apply (osg::State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.247 osg::PolygonMode Class Reference



Public Types

- enum **Mode** {

  POINT,

  LINE,

  FILL }
- enum **Face** {

  FRONT_AND_BACK,

  FRONT,

  BACK }

Public Member Functions

- **PolygonMode** (Face face, Mode mode)
- PolygonMode (const PolygonMode &pm, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, PolygonMode, POLYGONMODE)
- virtual int compare (const StateAttribute &sa) const
- void **setMode** (Face face, Mode mode)
- Mode **getMode** (Face face) const
- bool **getFrontAndBack** () const
- virtual void apply (State &state) const

4.248 Detailed Description

State Class for setting OpenGL's polygon culling mode.

4.249 Constructor & Destructor Documentation

osg::PolygonMode::PolygonMode (const PolygonMode & *pm*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.250 Member Function Documentation

virtual int osg::PolygonMode::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

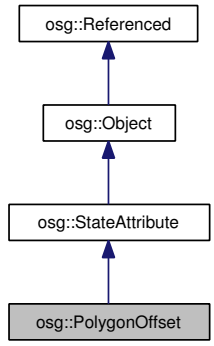
Implements osg::StateAttribute.

virtual void osg::PolygonMode::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.251 osg::PolygonOffset Class Reference



Public Member Functions

- **PolygonOffset** (float factor, float units)
- PolygonOffset (const PolygonOffset &po, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, PolygonOffset, POLYGONOFFSET)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFactor** (float factor)
- float **getFactor** () const
- void **setUnits** (float units)
- float **getUnits** () const
- virtual void apply (State &state) const

Static Public Member Functions

- static void **setFactorMultiplier** (float multiplier)
- static float **getFactorMultiplier** ()
- static void **setUnitsMultiplier** (float multiplier)
- static float **getUnitsMultiplier** ()
- static bool **areFactorAndUnitsMultipliersSet** ()
- static void setFactorAndUnitsMultipliersUsingBestGuessForDriver ()

4.252 Detailed Description

PolygonOffset - encapsulates the OpenGL glPolygonOffset state.

4.253 Constructor & Destructor Documentation

osg::PolygonOffset::PolygonOffset (const PolygonOffset & *po*, const CopyOp & *copyop* = CopyOp::SHALLOW_COPY) [inline]

Copy constructor using CopyOp to manage deep vs shallow copy.

4.254 Member Function Documentation

virtual int osg::PolygonOffset::compare (const StateAttribute & *sa*) const [inline, virtual]

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::PolygonOffset::getModeUsage (StateAttribute::ModeUsage &) const [inline, virtual]

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::PolygonOffset::apply (State &) const [virtual]

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

static void osg::PolygonOffset::setFactorAndUnitsMultipliersUsingBestGuessForDriver () [static]

Checks with the OpenGL driver to try and pick multiplier approrpriate for the hardware. note, requires a valid graphics context to be current.

4.255 osg::Polytope Class Reference

Public Types

- typedef unsigned int **ClippingMask**
- typedef std::vector< Plane > **PlaneList**
- typedef std::vector< Vec3 > **VertexList**

- typedef fast_back_stack< ClippingMask > **MaskStack**

Public Member Functions

- **Polytope** (const Polytope &cv)
- **Polytope** (const PlaneList &pl)
- void **clear** ()
- Polytope & **operator=** (const Polytope &cv)
- void setToUnitFrustum (bool withNear=true, bool withFar=true)
- void setToBoundingBox (const BoundingBox &bb)
- void **setAndTransformProvidingInverse** (const Polytope &pt, const osg::Matrix &matrix)
- void **set** (const PlaneList &pl)
- void **add** (const osg::Plane &pl)
- void flip ()
- PlaneList & **getPlaneList** ()
- const PlaneList & **getPlaneList** () const
- void **setReferenceVertexList** (VertexList &vertices)
- VertexList & **getReferenceVertexList** ()
- const VertexList & **getReferenceVertexList** () const
- void **setupMask** ()
- ClippingMask & **getCurrentMask** ()
- ClippingMask **getCurrentMask** () const
- void **setResultMask** (ClippingMask mask)
- ClippingMask **getResultMask** () const
- MaskStack & **getMaskStack** ()
- const MaskStack & **getMaskStack** () const
- void **pushCurrentMask** ()
- void **popCurrentMask** ()
- bool contains (const osg::Vec3 &v) const
- bool contains (const std::vector< Vec3 > &vertices)
- bool contains (const osg::BoundingSphere &bs)
- bool contains (const osg::BoundingBox &bb)
- bool containsAllOf (const std::vector< Vec3 > &vertices)
- bool containsAllOf (const osg::BoundingSphere &bs)
- bool containsAllOf (const osg::BoundingBox &bb)
- void transform (const osg::Matrix &matrix)
- void transformProvidingInverse (const osg::Matrix &matrix)

4.256 Detailed Description

A Polytope class for representing convex clipping volumes made up of a set of planes. When adding planes, their normals should point inwards (into the volume)

4.257 Member Function Documentation

void osg::Polytope::setToUnitFrustum (bool *withNear* = `true`, bool *withFar* = `true`) `[inline]`

Create a Polytope which is a cube, centered at 0,0,0, with sides of 2 units.

void osg::Polytope::setToBoundingBox (const BoundingBox & *bb*) `[inline]`

Create a Polytope which is a equivilant to BoundingBox.

void osg::Polytope::flip () `[inline]`

flip/reverse the orientation of all the planes.

bool osg::Polytope::contains (const osg::Vec3 & *v*) const `[inline]`

Check whether a vertex is contained within clipping set.

bool osg::Polytope::contains (const std::vector< Vec3 > & *vertices*) `[inline]`

Check whether any part of vertex list is contained within clipping set.

bool osg::Polytope::contains (const osg::BoundingSphere & *bs*) `[inline]`

Check whether any part of a bounding sphere is contained within clipping set. Using a mask to determine which planes should be used for the check, and modifying the mask to turn off planes which wouldn't contribute to clipping of any internal objects. This feature is used in osgUtil::CullVisitor to prevent redundant plane checking.

bool osg::Polytope::contains (const osg::BoundingBox & *bb*) `[inline]`

Check whether any part of a bounding box is contained within clipping set. Using a mask to determine which planes should be used for the check, and modifying the mask to turn off planes which wouldn't contribute to clipping of any internal objects. This feature is used in osgUtil::CullVisitor to prevent redundant plane checking.

bool osg::Polytope::containsAllOf (const std::vector< Vec3 > & *vertices*) `[inline]`

Check whether all of vertex list is contained with clipping set.

bool osg::Polytope::containsAllOf (const osg::BoundingSphere & *bs*) `[inline]`

Check whether the entire bounding sphere is contained within clipping set.

bool osg::Polytope::containsAllOf (const osg::BoundingBox & *bb*) `[inline]`

Check whether the entire bounding box is contained within clipping set.

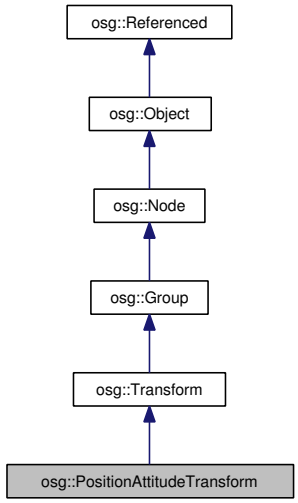
void osg::Polytope::transform (const osg::Matrix & *matrix*) `[inline]`

Transform the clipping set by matrix. Note, this operations carries out the calculation of the inverse of the matrix since a plane must be multiplied by the inverse transposed to transform it. This makes this operation expensive. If the inverse has been already calculated elsewhere then use transformProvidingInverse() instead. See http://www.worldserver.com/turk/computergraphics/NormalTransformations.pdf

void osg::Polytope::transformProvidingInverse (const osg::Matrix & *matrix*) `[inline]`

Transform the clipping set by provide a pre inverted matrix. see transform for details.

4.258 osg::PositionAttitudeTransform Class Reference



Public Member Functions

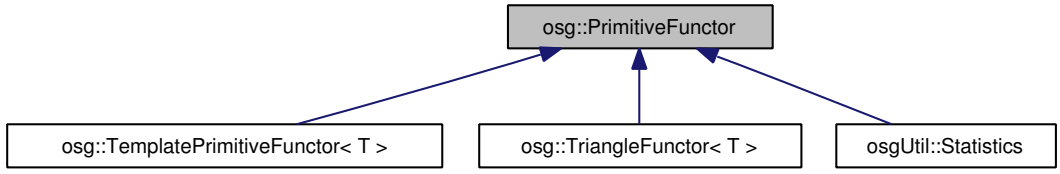
- **PositionAttitudeTransform** (const PositionAttitudeTransform &pat, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, PositionAttitudeTransform)
- virtual PositionAttitudeTransform * **asPositionAttitudeTransform** ()
- virtual const PositionAttitudeTransform * **asPositionAttitudeTransform** () const

- void **setPosition** (const Vec3d &pos)
- const Vec3d & **getPosition** () const
- void **setAttitude** (const Quat &quat)
- const Quat & **getAttitude** () const
- void **setScale** (const Vec3d &scale)
- const Vec3d & **getScale** () const
- void **setPivotPoint** (const Vec3d &pivot)
- const Vec3d & **getPivotPoint** () const
- virtual bool **computeLocalToWorldMatrix** (Matrix &matrix, NodeVisitor *nv) const
- virtual bool **computeWorldToLocalMatrix** (Matrix &matrix, NodeVisitor *nv) const

4.259 Detailed Description

PositionAttitudeTransform - is a Transform. Sets the coordinate transform via a Vec3 position and Quat attitude.

4.260 osg::PrimitiveFunctor Class Reference



Public Member Functions

- virtual void setVertexArray (unsigned int count, const Vec2 *vertices)=0
- virtual void setVertexArray (unsigned int count, const Vec3 *vertices)=0
- virtual void setVertexArray (unsigned int count, const Vec4 *vertices)=0
- virtual void setVertexArray (unsigned int count, const Vec2d *vertices)=0
- virtual void setVertexArray (unsigned int count, const Vec3d *vertices)=0
- virtual void setVertexArray (unsigned int count, const Vec4d *vertices)=0
- virtual void drawArrays (GLenum mode, GLint first, GLsizei count)=0

  Mimics the OpenGL `glDrawArrays()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLubyte *indices)=0

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLushort *indices)=0

Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLuint *indices)=0

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void begin (GLenum mode)=0

  Mimics the OpenGL `glBegin()` *function.*

- virtual void vertex (const Vec2 &vert)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec3 &vert)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec4 &vert)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y, float z)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y, float z, float w)=0

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void end ()=0

  Mimics the OpenGL `glEnd()` *function.*

4.261 Detailed Description

A `PrimitiveFunctor` is used (in conjunction with `osg::Drawable::accept(PrimitiveFunctor&)`) to get access to the primitives that compose the things drawn by OSG.

If `osg::Drawable::accept()` is called with a `PrimitiveFunctor` parameter, the `Drawable` will "pretend" it is drawing itself, but instead of calling real OpenGL functions, it will call `PrimitiveFunctor`'s member functions that "mimic" the OpenGL calls.

Concrete subclasses of `PrimitiveFunctor` must implement these methods so that they performs whatever they want.

4.262 Member Function Documentation

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec2 * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec3 * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec4 * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec2d * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec3d * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

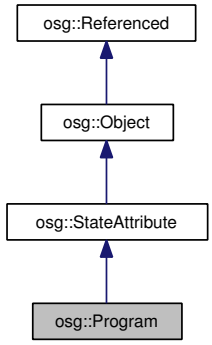
Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

virtual void osg::PrimitiveFunctor::setVertexArray (unsigned int *count*, const Vec4d * *vertices*) `[pure virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implemented in osg::TemplatePrimitiveFunctor< T >, osg::TriangleFunctor< T >, and osgUtil::Statistics.

4.263 osg::Program Class Reference



Public Types

- typedef std::map< std::string, GLuint > **AttribBindingList**
- typedef std::map< std::string, GLuint > **FragDataBindingList**
- typedef std::map< std::string, ActiveVarInfo > **ActiveVarInfoMap**

Public Member Functions

- Program (const Program &rhs, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Program, PROGRAM)
- virtual int compare (const osg::StateAttribute &sa) const
- virtual void apply (osg::State &state) const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- virtual void compileGLObjects (osg::State &state) const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (osg::State *state=0) const
- void dirtyProgram ()
- bool addShader (Shader *shader)
- unsigned int **getNumShaders** () const
- Shader * **getShader** (unsigned int i)
- const Shader * **getShader** (unsigned int i) const
- bool removeShader (Shader *shader)
- void addBindAttribLocation (const std::string &name, GLuint index)
- void removeBindAttribLocation (const std::string &name)

- void addBindFragDataLocation (const std::string &name, GLuint index)
- void removeBindFragDataLocation (const std::string &name)
- const AttribBindingList & **getAttribBindingList** () const
- const FragDataBindingList & **getFragDataBindingList** () const
- bool isFixedFunction () const
- bool getGlProgramInfoLog (unsigned int contextID, std::string &log) const
- const ActiveVarInfoMap & **getActiveUniforms** (unsigned int contextID) const
- const ActiveVarInfoMap & **getActiveAttribs** (unsigned int contextID) const
- PerContextProgram * getPCP (unsigned int contextID) const

Static Public Member Functions

- static void deleteGlProgram (unsigned int contextID, GLuint program)
- static void flushDeletedGlPrograms (unsigned int contextID, double currentTime, double &available-Time)

Friends

- class **PerContextProgram**

Classes

- struct **ActiveVarInfo**
- class PerContextProgram

4.264 Detailed Description

osg::Program is an application-level abstraction of an OpenGL glProgram. It is an osg::StateAttribute that, when applied, will activate a glProgram for subsequent rendering. osg::Shaders containing the actual shader source code are attached to a Program, which will then manage the compilation, linking, and activation of the GLSL program. osg::Program will automatically manage per-context instancing of the OpenGL glPrograms, if that is necessary for a particular display configuration.

4.265 Constructor & Destructor Documentation

osg::Program::Program (const Program & *rhs*, const osg::CopyOp & *copyop* = `osg::CopyOp::SHALLOW_COPY`**)**

Copy constructor using CopyOp to manage deep vs shallow copy.

4.266 Member Function Documentation

virtual int osg::Program::compare (const osg::StateAttribute & *sa*) const `[virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual void osg::Program::apply (osg::State & *state*) const `[virtual]`

If enabled, activate our program in the GL pipeline, performing any rebuild operations that might be pending.

Reimplemented from osg::StateAttribute.

virtual void osg::Program::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Referenced.

virtual void osg::Program::compileGLObjects (osg::State & *state*) const `[virtual]`

Compile program and associated shaders.

Reimplemented from osg::StateAttribute.

virtual void osg::Program::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::StateAttribute.

virtual void osg::Program::releaseGLObjects (osg::State * *state* = 0) const `[virtual]`

release OpenGL objects in specified graphics context if State object is passed, otherwise release OpenGL objects for all graphics context if State object pointer NULL.

Reimplemented from osg::StateAttribute.

void osg::Program::dirtyProgram ()

Mark our PCSOs as needing relink

bool osg::Program::addShader (Shader * *shader*)

Attach an osg::Shader to this osg::Program. Mark Program as needing relink. Return true for success

bool osg::Program::removeShader (Shader * *shader*)

Remove osg::Shader from this osg::Program. Mark Program as needing relink. Return true for success

void osg::Program::addBindAttribLocation (const std::string & *name*, GLuint *index*)

Add an attribute location binding.

void osg::Program::removeBindAttribLocation (const std::string & *name*)

Remove an attribute location binding.

void osg::Program::addBindFragDataLocation (const std::string & *name*, GLuint *index*)

Add an frag data location binding. See EXT_gpu_shader4 for BindFragDataLocationEXT

void osg::Program::removeBindFragDataLocation (const std::string & *name*)

Remove an frag data location binding.

bool osg::Program::isFixedFunction () const

Return true if this Program represents "fixed-functionality" rendering

bool osg::Program::getGlProgramInfoLog (unsigned int *contextID*, std::string & *log*) const

Query InfoLog from a glProgram

static void osg::Program::deleteGlProgram (unsigned int *contextID*, GLuint *program*) `[static]`

Mark internal glProgram for deletion. Deletion requests are queued until they can be executed in the proper GL context.

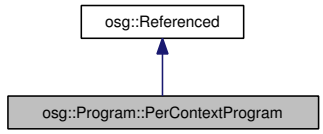
static void osg::Program::flushDeletedGlPrograms (unsigned int *contextID*, double *currentTime*, double & *availableTime*) `[static]`

flush all the cached glPrograms which need to be deleted in the OpenGL context related to contextID.

PerContextProgram* osg::Program::getPCP (unsigned int *contextID*) const

Get the PCP for a particular GL context

4.267 osg::Program::PerContextProgram Class Reference



Public Member Functions

- **PerContextProgram** (const Program *program, unsigned int contextID)
- GLuint **getHandle** () const
- void **requestLink** ()
- void **linkProgram** ()
- bool **validateProgram** ()
- bool **needsLink** () const
- bool **isLinked** () const
- bool **getInfoLog** (std::string &infoLog) const
- void **useProgram** () const
- void **resetAppliedUniforms** () const
- void **apply** (const Uniform &uniform) const
- const ActiveVarInfoMap & **getActiveUniforms** () const
- const ActiveVarInfoMap & **getActiveAttribs** () const
- GLint **getUniformLocation** (const std::string &name) const
- GLint **getAttribLocation** (const std::string &name) const

4.268 Detailed Description

PerContextProgram (PCP) is an OSG-internal encapsulation of glPrograms per-GL context.

4.269 Member Data Documentation

const Program* osg::Program::PerContextProgram::_program `[protected]`

Pointer to our parent Program

osg::ref_ptr<GL2Extensions> osg::Program::PerContextProgram::_extensions `[protected]`

Pointer to this context's extension functions

GLuint osg::Program::PerContextProgram::_glProgramHandle `[protected]`

Handle to the actual OpenGL glProgram

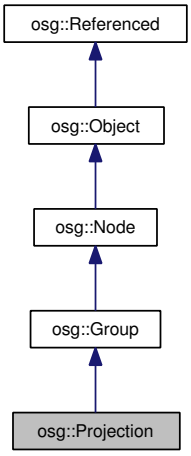
bool osg::Program::PerContextProgram::_needsLink `[protected]`

Does our glProgram need to be linked?

bool osg::Program::PerContextProgram::_isLinked `[protected]`

Is our glProgram successfully linked?

4.270 osg::Projection Class Reference



Public Member Functions

- Projection (const Projection &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **Projection** (const Matrix &matix)
- **META_Node** (osg, Projection)
- void setMatrix (const Matrix &mat)
- const Matrix & getMatrix () const
- void preMult (const Matrix &mat)
- void postMult (const Matrix &mat)

4.271 Detailed Description

Projection nodes set up the frustum/orthographic projection used when rendering the scene.

4.272 Constructor & Destructor Documentation

osg::Projection::Projection (const Projection &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.273 Member Function Documentation

void osg::Projection::setMatrix (const Matrix & *mat*) `[inline]`

Set the transform's matrix.

const Matrix& osg::Projection::getMatrix () const `[inline]`

Get the transform's matrix.

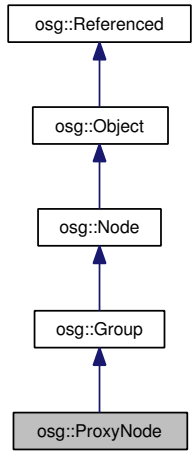
void osg::Projection::preMult (const Matrix & *mat*) `[inline]`

preMult transform.

void osg::Projection::postMult (const Matrix & *mat*) `[inline]`

postMult transform.

4.274 osg::ProxyNode Class Reference



Public Types

- enum CenterMode {
  USE_BOUNDING_SPHERE_CENTER,
  USER_DEFINED_CENTER }
- typedef std::vector< std::string > **FileNameList**

Public Member Functions

- ProxyNode (const ProxyNode &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, ProxyNode)
- virtual void traverse (NodeVisitor &nv)
- virtual bool addChild (Node *child)
- virtual bool **addChild** (Node *child, const std::string &filename)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove)
- void setDatabasePath (const std::string &path)
- const std::string & getDatabasePath () const
- void **setFileName** (unsigned int childNo, const std::string &filename)
- const std::string & **getFileName** (unsigned int childNo) const
- unsigned int **getNumFileNames** () const
- void setCenterMode (CenterMode mode)
- CenterMode getCenterMode () const
- void setCenter (const Vec3 &center)
- const Vec3 & getCenter () const

- void setRadius (float radius)
- float getRadius () const
- virtual BoundingSphere computeBound () const

4.275 Detailed Description

ProxyNode.

4.276 Member Enumeration Documentation

enum osg::ProxyNode::CenterMode

Modes which control how the center of object should be determined when computed which child is active.

4.277 Constructor & Destructor Documentation

osg::ProxyNode::ProxyNode (const ProxyNode &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.278 Member Function Documentation

virtual void osg::ProxyNode::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::Group.

virtual bool osg::ProxyNode::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented from osg::Group.

virtual bool osg::ProxyNode::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove*) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented from osg::Group.

void osg::ProxyNode::setDatabasePath (const std::string & *path*)

Set the database path to prepend to children's filenames.

const std::string& osg::ProxyNode::getDatabasePath () const `[inline]`

Get the database path used to prepend to children's filenames.

void osg::ProxyNode::setCenterMode (CenterMode *mode*) `[inline]`

Set how the center of object should be determined when computed which child is active.

CenterMode osg::ProxyNode::getCenterMode () const `[inline]`

Get how the center of object should be determined when computed which child is active.

void osg::ProxyNode::setCenter (const Vec3 & *center*) `[inline]`

Sets the object-space point which defines the center of the osg::ProxyNode. center is affected by any transforms in the hierarchy above the osg::ProxyNode.

const Vec3& osg::ProxyNode::getCenter () const `[inline]`

return the ProxyNode center point.

void osg::ProxyNode::setRadius (float *radius*) `[inline]`

Set the object-space reference radius of the volume enclosed by the ProxyNode. Used to detmine the bounding sphere of the ProxyNode in the absense of any children.

float osg::ProxyNode::getRadius () const `[inline]`

Get the object-space radius of the volume enclosed by the ProxyNode.

virtual BoundingSphere osg::ProxyNode::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Group.

4.279 osg::Quat Class Reference

Public Types

- typedef double **value_type**

Public Member Functions

- **Quat** (value_type x, value_type y, value_type z, value_type w)
- **Quat** (const Vec4f &v)
- **Quat** (const Vec4d &v)
- **Quat** (value_type angle, const Vec3f &axis)
- **Quat** (value_type angle, const Vec3d &axis)
- **Quat** (value_type angle1, const Vec3f &axis1, value_type angle2, const Vec3f &axis2, value_type angle3, const Vec3f &axis3)
- **Quat** (value_type angle1, const Vec3d &axis1, value_type angle2, const Vec3d &axis2, value_type angle3, const Vec3d &axis3)
- Quat & **operator=** (const Quat &v)
- bool **operator==** (const Quat &v) const
- bool **operator!=** (const Quat &v) const
- bool **operator**< (const Quat &v) const
- Vec4d **asVec4** () const
- Vec3d **asVec3** () const
- void **set** (value_type x, value_type y, value_type z, value_type w)
- void **set** (const osg::Vec4f &v)
- void **set** (const osg::Vec4d &v)
- void **set** (const Matrixf &matrix)
- void **set** (const Matrixd &matrix)
- void **get** (Matrixf &matrix) const
- void **get** (Matrixd &matrix) const
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type & **w** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- value_type **w** () const
- bool zeroRotation () const
- const Quat operator ∗ (value_type rhs) const

  Multiply by scalar.

- Quat & operator *= (value_type rhs)

  Unary multiply by scalar.

- const Quat operator * (const Quat &rhs) const

  Binary multiply.

- Quat & operator *= (const Quat &rhs)

  Unary multiply.

- Quat operator/ (value_type rhs) const

  Divide by scalar.

- Quat & operator/= (value_type rhs)

  Unary divide by scalar.

- const Quat operator/ (const Quat &denom) const

  Binary divide.

- Quat & operator/= (const Quat &denom)

  Unary divide.

- const Quat operator+ (const Quat &rhs) const

  Binary addition.

- Quat & operator+= (const Quat &rhs)

  Unary addition.

- const Quat operator- (const Quat &rhs) const

  Binary subtraction.

- Quat & operator-= (const Quat &rhs)

  Unary subtraction.

- const Quat operator- () const
- value_type length () const

  Length of the quaternion = sqrt(vec . vec).

- value_type length2 () const

  Length of the quaternion = vec . vec.

- Quat conj () const

  Conjugate.

- const Quat inverse () const

Multiplicative inverse method: q^(-1) = q^*/(q.q^*).

- void **makeRotate** (value_type angle, value_type x, value_type y, value_type z)
- void **makeRotate** (value_type angle, const Vec3f &vec)
- void **makeRotate** (value_type angle, const Vec3d &vec)
- void **makeRotate** (value_type angle1, const Vec3f &axis1, value_type angle2, const Vec3f &axis2, value_type angle3, const Vec3f &axis3)
- void **makeRotate** (value_type angle1, const Vec3d &axis1, value_type angle2, const Vec3d &axis2, value_type angle3, const Vec3d &axis3)
- void makeRotate (const Vec3f &vec1, const Vec3f &vec2)
- void makeRotate (const Vec3d &vec1, const Vec3d &vec2)
- void **makeRotate_original** (const Vec3d &vec1, const Vec3d &vec2)
- void getRotate (value_type &angle, value_type &x, value_type &y, value_type &z) const
- void getRotate (value_type &angle, Vec3f &vec) const
- void getRotate (value_type &angle, Vec3d &vec) const
- void slerp (value_type t, const Quat &from, const Quat &to)
- Vec3f operator * (const Vec3f &v) const
- Vec3d operator * (const Vec3d &v) const

Public Attributes

- value_type **_v** [4]

4.280 Detailed Description

A quaternion class. It can be used to represent an orientation in 3D space.

4.281 Member Function Documentation

bool osg::Quat::zeroRotation () const `[inline]`

return true if the Quat represents a zero rotation, and therefore can be ignored in computations.

const Quat osg::Quat::operator- () const `[inline]`

Negation operator - returns the negative of the quaternion. Basically just calls operator - () on the Vec4

void osg::Quat::makeRotate (const Vec3f & *vec1*, const Vec3f & *vec2*)

Make a rotation Quat which will rotate vec1 to vec2. Generally take a dot product to get the angle between these and then use a cross product to get the rotation axis Watch out for the two special cases when the vectors are co-incident or opposite in direction.

void osg::Quat::makeRotate (const Vec3d & *vec1*, const Vec3d & *vec2*)

Make a rotation Quat which will rotate vec1 to vec2. Generally take a dot product to get the angle between these and then use a cross product to get the rotation axis Watch out for the two special cases of when the vectors are co-incident or opposite in direction.

void osg::Quat::getRotate (value_type & *angle*, value_type & *x*, value_type & *y*, value_type & *z*) const

Return the angle and vector components represented by the quaternion.

void osg::Quat::getRotate (value_type & *angle*, Vec3f & *vec*) const

Return the angle and vector represented by the quaternion.

void osg::Quat::getRotate (value_type & *angle*, Vec3d & *vec*) const

Return the angle and vector represented by the quaternion.

void osg::Quat::slerp (value_type *t*, const Quat & *from*, const Quat & *to*)

Spherical Linear Interpolation. As t goes from 0 to 1, the Quat object goes from "from" to "to".

Vec3f osg::Quat::operator * (const Vec3f & *v*) const `[inline]`

Rotate a vector by this quaternion.

Vec3d osg::Quat::operator * (const Vec3d & *v*) const `[inline]`

Rotate a vector by this quaternion.

4.282 osg::Referenced Class Reference

Inherited by osg::ApplicationUsage, osg::BlendColor::Extensions, osg::BlendEquation::Extensions, osg::BlendFunc::Extensions, osg::BufferObject::Extensions, osg::ClampColor::Extensions, osg::CullingSet, osg::CullSettings::ClampProjectionMatrixCallback, osg::DisplaySettings, osg::Drawable::Extensions, osg::FBOExtensions, osg::FragmentProgram::Extensions, osg::FrameStamp, osg::GL2Extensions, osg::GraphicsContext::ResizedCallback, osg::GraphicsContext::Traits, osg::GraphicsContext::WindowingSystemInterface, osg::LineSegment, osg::Multisample::Extensions, osg::NodeVisitor `[virtual]`, osg::NodeVisitor::DatabaseRequestHandler, osg::Object, osg::Operation `[virtual]`, osg::OperationQueue, osg::OperationThread, osg::Point::Extensions, osg::Program::PerContextProgram, osg::RefBlock `[virtual]`, osg::RefBlockCount `[virtual]`, osg::Shader::PerContextShader, osg::State, osg::State::DynamicObjectRenderingCompletedCallback, osg::Stats,

osg::StencilTwoSided::Extensions, osg::Texture1D::SubloadCallback, osg::Texture2D::SubloadCallback, osg::Texture2DArray::Extensions, osg::Texture2DArray::SubloadCallback, osg::Texture3D::Extensions, osg::Texture3D::SubloadCallback, osg::Texture::Extensions, osg::Texture::TextureObject, osg::TextureCubeMap::Extensions, osg::TextureCubeMap::SubloadCallback, osg::TextureRectangle::SubloadCallback, osg::TransferFunction, osg::VertexProgram::Extensions, osgDB::DatabasePager::DatabaseRequest, osgDB::DotOsgWrapper, osgDB::DynamicLibrary, osgDB::ImageOptions::TexCoordRange, osgDB::Registry, osgDB::Registry::ReadFileCallback [virtual], osgDB::Registry::WriteFileCallback [virtual], osgUtil::CubeMapGenerator, osgUtil::DelaunayTriangulator, osgUtil::IntersectionVisitor::ReadCallback, osgUtil::Intersector, osgUtil::IntersectVisitor::IntersectState, osgUtil::Optimizer::IsOperationPermissibleForObjectCallback, osgUtil::Optimizer::TextureAtlasBuilder::Atlas, osgUtil::Optimizer::TextureAtlasBuilder::Source, osgUtil::RenderBin::DrawCallback, osgUtil::RenderBin::SortCallback, osgUtil::RenderLeaf, osgUtil::SceneView::ComputeStereoMatricesCallback, osgUtil::Simplifier::ContinueSimplificationCallback, osgUtil::StateGraph, osgUtil::TangentSpaceGenerator, osgUtil::Tessellator, and osgUtil::Tessellator::Prim.

Public Member Functions

- **Referenced** (bool threadSafeRefUnref)
- **Referenced** (const Referenced &)
- Referenced & **operator=** (const Referenced &)
- virtual void setThreadSafeRefUnref (bool threadSafe)
- bool getThreadSafeRefUnref () const
- OpenThreads::Mutex * getRefMutex () const
- void ref () const
- void unref () const
- void unref_nodelete () const
- int referenceCount () const
- void addObserver (Observer *observer)
- void removeObserver (Observer *observer)

Static Public Member Functions

- static void setThreadSafeReferenceCounting (bool enableThreadSafeReferenceCounting)
- static bool getThreadSafeReferenceCounting ()
- static void setDeleteHandler (DeleteHandler *handler)
- static DeleteHandler * getDeleteHandler ()

Friends

- class **DeleteHandler**

4.283 Detailed Description

Base class from providing referencing counted objects.

4.284 Member Function Documentation

virtual void osg::Referenced::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented in osg::Drawable, osg::Geode, osg::Group, osg::LightSource, osg::Node, osg::Program, osg::StateSet, and osg::TexGenNode.

bool osg::Referenced::getThreadSafeRefUnref () const `[inline]`

Get whether a mutex is used to ensure ref() and unref() are thread safe.

OpenThreads::Mutex* osg::Referenced::getRefMutex () const `[inline]`

Get the mutex used to ensure thread safety of ref()/unref().

void osg::Referenced::ref () const `[inline]`

Increment the reference count by one, indicating that this object has another pointer which is referencing it.

void osg::Referenced::unref () const `[inline]`

Decrement the reference count by one, indicating that a pointer to this object is referencing it. If the reference count goes to zero, it is assumed that this object is no longer referenced and is automatically deleted.

void osg::Referenced::unref_nodelete () const

Decrement the reference count by one, indicating that a pointer to this object is referencing it. However, do not delete it, even if ref count goes to 0. Warning, unref_nodelete() should only be called if the user knows exactly who will be resonsible for, one should prefer unref() over unref_nodelete() as the later can lead to memory leaks.

int osg::Referenced::referenceCount () const `[inline]`

Return the number pointers currently referencing this object.

void osg::Referenced::addObserver (Observer * *observer*)

Add a Observer that is observering this object, notify the Observer when this object gets deleted.

void osg::Referenced::removeObserver (Observer * *observer*)

Add a Observer that is observering this object, notify the Observer when this object gets deleted.

static void osg::Referenced::setThreadSafeReferenceCounting (bool *enableThreadSafeReferenceCounting*) `[static]`

Set whether reference counting should be use a mutex to create thread reference counting.

static bool osg::Referenced::getThreadSafeReferenceCounting () `[static]`

Get whether reference counting is active.

static void osg::Referenced::setDeleteHandler (DeleteHandler * *handler*) `[static]`

Set a DeleteHandler to which deletion of all referenced counted objects will be delegated to.

static DeleteHandler* osg::Referenced::getDeleteHandler () `[static]`

Get a DeleteHandler.

4.285 osg::ReleaseContext_Block_MakeCurrentOperation Struct Reference

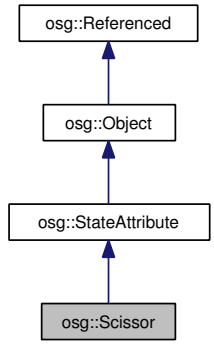
Public Member Functions

- virtual void **release** ()
- virtual void **operator()** (GraphicsContext *context)

4.286 Detailed Description

ReleaseContext_Block_MakeCurrentOperation releases the context for another thread to aquire, then blocks waiting for context to be released, once the block is release the context is re-aqquired.

4.287 osg::Scissor Class Reference



Public Member Functions

- **Scissor** (int x, int y, int width, int height)
- Scissor (const Scissor &vp, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Scissor, SCISSOR)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setScissor** (int x, int y, int width, int height)
- void **getScissor** (int &x, int &y, int &width, int &height) const
- int & **x** ()
- int **x** () const
- int & **y** ()
- int **y** () const
- int & **width** ()
- int **width** () const
- int & **height** ()
- int **height** () const
- virtual void apply (State &state) const

4.288 Detailed Description

Encapsulate OpenGL glScissor.

4.289 Constructor & Destructor Documentation

osg::Scissor::Scissor (const Scissor & *vp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.290 Member Function Documentation

virtual int osg::Scissor::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Scissor::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

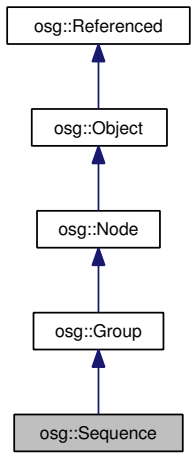
Reimplemented from osg::StateAttribute.

virtual void osg::Scissor::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.291 osg::Sequence Class Reference



Public Types

- enum LoopMode {
  LOOP,
  SWING }
- enum SequenceMode {
  START,
  STOP,
  PAUSE,
  RESUME }

Public Member Functions

- Sequence (const Sequence &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Sequence)
- virtual void traverse (NodeVisitor &nv)
- virtual bool addChild (Node *child)
- virtual bool **addChild** (Node *child, double t)
- virtual bool insertChild (unsigned int index, Node *child)
- virtual bool **insertChild** (unsigned int index, Node *child, double t)
- virtual bool removeChild (Node *child)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove)
- void setValue (int value)
- int **getValue** () const
- void setTime (unsigned int frame, double t)
- double getTime (unsigned int frame) const
- void setDefaultTime (double t)
- double getDefaultTime (void) const
- void setLastFrameTime (double t)
- double getLastFrameTime (void) const
- unsigned int getNumFrames () const
- void setInterval (LoopMode mode, int begin, int end)
- void getInterval (LoopMode &mode, int &begin, int &end) const
- void setDuration (float speed, int nreps=-1)
- void getDuration (float &speed, int &nreps) const
- void setMode (SequenceMode mode)
- SequenceMode getMode () const
- void setSync (bool sync)
- void getSync (bool &sync) const
- void setClearOnStop (bool clearOnStop)
- void getClearOnStop (bool &clearOnStop) const

4.292 Detailed Description

Sequence is a Group node which allows automatic, time based switching between children.

4.293 Member Enumeration Documentation

enum osg::Sequence::LoopMode

Interval modes. 'Loop' repeats frames 1-N; 'swing' repeats 1->N, (N-1)->1.

enum osg::Sequence::SequenceMode

Sequence modes.

4.294 Constructor & Destructor Documentation

osg::Sequence::Sequence (const Sequence &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)**

Copy constructor using CopyOp to manage deep vs shallow copy.

4.295 Member Function Documentation

virtual void osg::Sequence::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::Group.

virtual bool osg::Sequence::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented from osg::Group.

virtual bool osg::Sequence::insertChild (unsigned int *index*, Node * *child*) `[virtual]`

Insert Node to Group at specific location. The new child node is inserted into the child list before the node at the specified index. No nodes are removed from the group with this operation.

Reimplemented from osg::Group.

virtual bool osg::Sequence::removeChild (Node * *child*) `[virtual]`

Remove Node from Group. If Node is contained in Group then remove it from the child list, decrement its reference count, and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. If Node is not found then return false and do not change the reference count of the Node. Note, do not override, only override removeChildren(,) is required.

Reimplemented from osg::Group.

virtual bool osg::Sequence::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove*) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented from osg::Group.

void osg::Sequence::setValue (int *value*) `[inline]`

value is which child node is to be displayed

void osg::Sequence::setTime (unsigned int *frame*, double *t*)

Set time in seconds for child.

double osg::Sequence::getTime (unsigned int *frame*) const

Get time for child.

void osg::Sequence::setDefaultTime (double *t*) `[inline]`

Set default time in seconds for new child. if t<0, t=0

double osg::Sequence::getDefaultTime (void) const `[inline]`

Get default time in seconds for new child.

void osg::Sequence::setLastFrameTime (double *t*) `[inline]`

Set time of last frame of last loop, in seconds. if t<= 0, then ignored

double osg::Sequence::getLastFrameTime (void) const `[inline]`

Get last frame time in seconds

unsigned int osg::Sequence::getNumFrames () const `[inline]`

Get number of frames

void osg::Sequence::setInterval (LoopMode *mode*, int *begin*, int *end*)

Set sequence mode & interval (range of children to be displayed).

void osg::Sequence::getInterval (LoopMode & *mode*, int & *begin*, int & *end*) const `[inline]`

Get sequence mode & interval.

void osg::Sequence::setDuration (float *speed*, int *nreps* = −1)

Set duration: speed-up & number of repeats

void osg::Sequence::getDuration (float & *speed*, int & *nreps*) const `[inline]`

Get duration & number of repeats.

void osg::Sequence::setMode (SequenceMode *mode*)

Set sequence mode. Start/stop & pause/resume.

SequenceMode osg::Sequence::getMode () const `[inline]`

Get sequence mode.

void osg::Sequence::setSync (bool *sync*) `[inline]`

If false (default), frames will not be sync'd to frameTime. If true, frames will be sync'd to frameTime.

void osg::Sequence::getSync (bool & *sync*) const `[inline]`

Get sync value

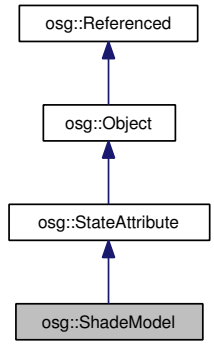
void osg::Sequence::setClearOnStop (bool *clearOnStop*) `[inline]`

If true, show no child nodes after stopping

void osg::Sequence::getClearOnStop (bool & *clearOnStop*) const `[inline]`

If true, show no child nodes after stopping

4.296 osg::ShadeModel Class Reference



Public Types

- enum **Mode** {
  FLAT,
  SMOOTH }

Public Member Functions

- **ShadeModel** (Mode mode=SMOOTH)
- ShadeModel (const ShadeModel &sm, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, ShadeModel, SHADEMODEL)
- virtual int compare (const StateAttribute &sa) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- virtual void apply (State &state) const

4.297 Detailed Description

Class which encapsulates glShadeModel(..).

4.298 Constructor & Destructor Documentation

osg::ShadeModel::ShadeModel (const ShadeModel & *sm*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.299 Member Function Documentation

virtual int osg::ShadeModel::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

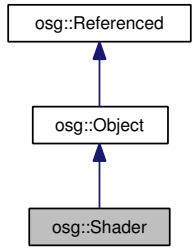
Implements osg::StateAttribute.

virtual void osg::ShadeModel::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.300 osg::Shader Class Reference



Public Types

- enum **Type** {
  VERTEX,
  FRAGMENT,
  UNDEFINED }

Public Member Functions

- **Shader** (Type type=UNDEFINED)
- **Shader** (Type type, const std::string &source)

- Shader (const Shader &rhs, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- **META_Object** (osg, Shader)
- int **compare** (const Shader &rhs) const
- bool **setType** (Type t)
- void setShaderSource (const std::string &sourceText)
- bool loadShaderSourceFromFile (const std::string &fileName)
- const std::string & getShaderSource () const
- Type getType () const
- const char * getTypename () const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- void releaseGLObjects (osg::State *state=0) const
- void dirtyShader ()
- void compileShader (unsigned int contextID) const
- void attachShader (unsigned int contextID, GLuint program) const
- bool getGlShaderInfoLog (unsigned int contextID, std::string &log) const

Static Public Member Functions

- static Shader * readShaderFile (Type type, const std::string &fileName)
- static void deleteGlShader (unsigned int contextID, GLuint shader)
- static void flushDeletedGlShaders (unsigned int contextID, double currentTime, double &available-Time)
- static Shader::Type **getTypeId** (const std::string &tname)

Friends

- class osg::Program

Classes

- class PerContextShader

4.301 Detailed Description

osg::Shader is an application-level abstraction of an OpenGL glShader. It is a container to load the shader source code text and manage its compilation. An osg::Shader may be attached to more than one osg::Program. Shader will automatically manage per-context instancing of the internal objects, if that is necessary for a particular display configuration.

4.302 Member Typedef Documentation

typedef std::set< osg::Program* > osg::Shader::ProgramSet `[protected]`

osg::Programs that this osg::Shader is attached to

4.303 Constructor & Destructor Documentation

osg::Shader::Shader (const Shader & *rhs*, const osg::CopyOp & *copyop* = `osg::CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.304 Member Function Documentation

void osg::Shader::setShaderSource (const std::string & *sourceText*)

Load the Shader's source code text from a string.

static Shader* osg::Shader::readShaderFile (Type *type*, const std::string & *fileName*) `[static]`

Read shader source from file and then constructor shader of specified type. Return the resulting Shader or 0 if no valid shader source code be read.

bool osg::Shader::loadShaderSourceFromFile (const std::string & *fileName*)

Load the Shader's source code text from a file.

const std::string& osg::Shader::getShaderSource () const `[inline]`

Query the shader's source code text

Type osg::Shader::getType () const `[inline]`

Get the Shader type as an enum.

const char* osg::Shader::getTypename () const

Get the Shader type as a descriptive string.

virtual void osg::Shader::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Object.

void osg::Shader::releaseGLObjects (osg::State * *state* = 0) const `[virtual]`

release OpenGL objects in specified graphics context if State object is passed, otherwise release OpenGL objects for all graphics context if State object pointer NULL.

Reimplemented from osg::Object.

void osg::Shader::dirtyShader ()

Mark our PCSs as needing recompilation. Also mark Programs that depend on us as needing relink

void osg::Shader::compileShader (unsigned int *contextID*) const

If needed, compile the PCS's glShader

void osg::Shader::attachShader (unsigned int *contextID*, GLuint *program*) const

For a given GL context, attach a glShader to a glProgram

bool osg::Shader::getGlShaderInfoLog (unsigned int *contextID*, std::string & *log*) const

Query InfoLog from a glShader

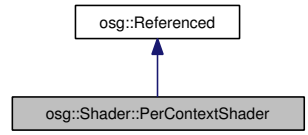
static void osg::Shader::deleteGlShader (unsigned int *contextID*, GLuint *shader*) `[static]`

Mark internal glShader for deletion. Deletion requests are queued tuntil they can be executed in the proper GL context.

static void osg::Shader::flushDeletedGlShaders (unsigned int *contextID*, double *currentTime*, double & *availableTime*) `[static]`

flush all the cached glShaders which need to be deleted in the OpenGL context related to contextID.

4.305 osg::Shader::PerContextShader Class Reference



Public Member Functions

- **PerContextShader** (const Shader *shader, unsigned int contextID)
- GLuint **getHandle** () const
- void **requestCompile** ()
- void **compileShader** ()
- bool **needsCompile** () const
- bool **isCompiled** () const
- bool **getInfoLog** (std::string &infoLog) const
- void attachShader (GLuint program) const
- void detachShader (GLuint program) const

4.306 Detailed Description

PerContextShader (PCS) is an OSG-internal encapsulation of glShader per-GL context.

4.307 Member Function Documentation

void osg::Shader::PerContextShader::attachShader (GLuint *program*) const

Attach our glShader to a glProgram

void osg::Shader::PerContextShader::detachShader (GLuint *program*) const

Detach our glShader from a glProgram

4.308 Member Data Documentation

const Shader* osg::Shader::PerContextShader::_shader `[protected]`

Pointer to our parent osg::Shader

osg::ref_ptr<osg::GL2Extensions> osg::Shader::PerContextShader::_extensions `[protected]`

Pointer to this context's extension functions.

GLuint osg::Shader::PerContextShader::_glShaderHandle `[protected]`

Handle to the actual glShader.

bool osg::Shader::PerContextShader::_needsCompile `[protected]`

Does our glShader need to be recompiled?

bool osg::Shader::PerContextShader::_isCompiled `[protected]`

Is our glShader successfully compiled?

4.309 osg::ShadowVolumeOccluder Class Reference

Public Types

- typedef std::vector< Polytope > **HoleList**

Public Member Functions

- **ShadowVolumeOccluder** (const ShadowVolumeOccluder &svo)
- bool **operator**< (const ShadowVolumeOccluder &svo) const
- bool computeOccluder (const NodePath &nodePath, const ConvexPlanarOccluder &occluder, CullStack &cullStack, bool createDrawables=false)
- void **disableResultMasks** ()
- void **pushCurrentMask** ()
- void **popCurrentMask** ()
- bool matchProjectionMatrix (const osg::Matrix &matrix) const
- void setNodePath (NodePath &nodePath)
- NodePath & **getNodePath** ()
- const NodePath & **getNodePath** () const
- float getVolume () const
- Polytope & getOccluder ()
- const Polytope & getOccluder () const
- HoleList & getHoleList ()
- const HoleList & getHoleList () const
- bool contains (const std::vector< Vec3 > &vertices)
- bool contains (const BoundingSphere &bound)
- bool contains (const BoundingBox &bound)
- void **transformProvidingInverse** (const osg::Matrix &matrix)

4.310 Detailed Description

ShadowVolumeOccluder is a helper class for implementating shadow occlusion culling.

4.311 Member Function Documentation

bool osg::ShadowVolumeOccluder::computeOccluder (const NodePath & *nodePath*, const ConvexPlanarOccluder & *occluder*, CullStack & *cullStack*, bool *createDrawables* = `false`)

compute the shadow volume occluder.

bool osg::ShadowVolumeOccluder::matchProjectionMatrix (const osg::Matrix & *matrix*) const `[inline]`

return true if the matrix passed in matches the projection matrix that this ShadowVolumeOccluder is associated with.

void osg::ShadowVolumeOccluder::setNodePath (NodePath & *nodePath*) `[inline]`

Set the NodePath which describes which node in the scene graph that this occluder is attached to.

float osg::ShadowVolumeOccluder::getVolume () const `[inline]`

get the volume of the occluder minus its holes, in eye coords, the volume is normalized by dividing by the volume of the view frustum in eye coords.

Polytope& osg::ShadowVolumeOccluder::getOccluder () `[inline]`

return the occluder polytope.

const Polytope& osg::ShadowVolumeOccluder::getOccluder () const `[inline]`

return the const occluder polytope.

HoleList& osg::ShadowVolumeOccluder::getHoleList () `[inline]`

return the list of holes.

const HoleList& osg::ShadowVolumeOccluder::getHoleList () const `[inline]`

return the const list of holes.

bool osg::ShadowVolumeOccluder::contains (const std::vector< Vec3 > & *vertices*)

return true if the specified vertex list is contained entirely within this shadow occluder volume.

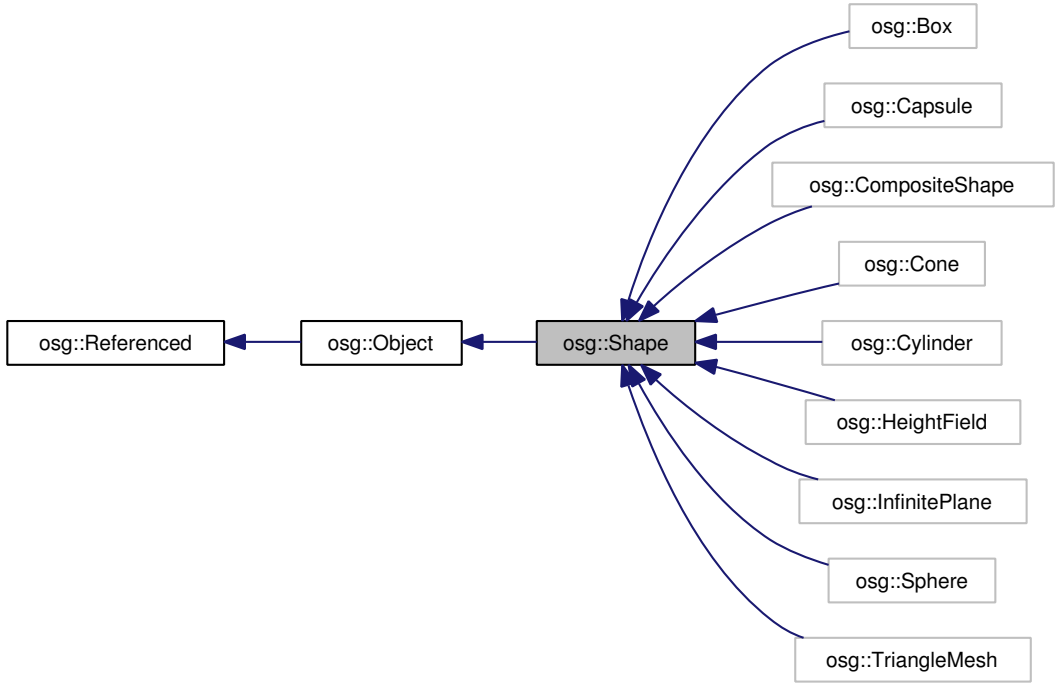
bool osg::ShadowVolumeOccluder::contains (const BoundingSphere & *bound*)

return true if the specified bounding sphere is contained entirely within this shadow occluder volume.

bool osg::ShadowVolumeOccluder::contains (const BoundingBox & *bound*)

return true if the specified bounding box is contained entirely within this shadow occluder volume.

4.312 osg::Shape Class Reference



Public Member Functions

- **Shape** (const Shape &sa, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const =0
- virtual Object * clone (const CopyOp &) const =0
- virtual bool isSameKindAs (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual void accept (ShapeVisitor &)=0
- virtual void accept (ConstShapeVisitor &) const =0

4.313 Detailed Description

Base class for all shape types. Shapes are used to either for culling and collision detection or to define the geometric shape of procedurally generate Geometry.

4.314 Member Function Documentation

virtual Object* osg::Shape::cloneType () const `[pure virtual]`

Clone the type of an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::Shape::clone (const CopyOp &) const `[pure virtual]`

Clone an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual bool osg::Shape::isSameKindAs (const Object * *obj*) const `[inline, virtual]`

return true if this and obj are of the same kind of object.

Reimplemented from osg::Object.

virtual const char* osg::Shape::libraryName () const `[inline, virtual]`

return the name of the attribute's library.

Implements osg::Object.

virtual const char* osg::Shape::className () const `[inline, virtual]`

return the name of the attribute's class type.

Implements osg::Object.

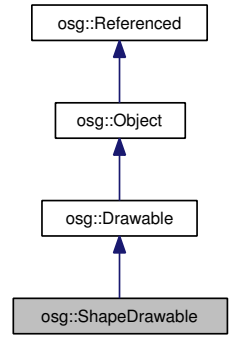
virtual void osg::Shape::accept (ShapeVisitor &) `[pure virtual]`

accept a non const shape visitor which can be used on non const shape objects. Must be defined by derived classes.

virtual void osg::Shape::accept (ConstShapeVisitor &) const `[pure virtual]`

accept a const shape visitor which can be used on const shape objects. Must be defined by derived classes.

4.315 osg::ShapeDrawable Class Reference



Public Member Functions

- **ShapeDrawable** (Shape *shape, TessellationHints *hints=0)
- ShapeDrawable (const ShapeDrawable &pg, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- void setColor (const Vec4 &color)
- const Vec4 & getColor () const
- void **setTessellationHints** (TessellationHints *hints)
- TessellationHints * **getTessellationHints** ()
- const TessellationHints * **getTessellationHints** () const
- virtual void drawImplementation (RenderInfo &renderInfo) const
- virtual bool supports (const AttributeFunctor &) const
- virtual bool supports (const Drawable::ConstAttributeFunctor &) const
- virtual void accept (Drawable::ConstAttributeFunctor &af) const
- virtual bool supports (const PrimitiveFunctor &) const
- virtual void accept (PrimitiveFunctor &pf) const
- virtual BoundingBox computeBound () const

4.316 Detailed Description

Allow the use of `Shape`s as `Drawable`s, so that they can be rendered with reduced effort. The implementation of `ShapeDrawable` is not geared to efficiency; it's better to think of it as a convenience to render `Shape`s easily (perhaps for test or debugging purposes) than as the right way to render basic shapes in some efficiency-critical section of code.

4.317 Constructor & Destructor Documentation

osg::ShapeDrawable::ShapeDrawable (const ShapeDrawable & *pg*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.318 Member Function Documentation

virtual Object* osg::ShapeDrawable::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::ShapeDrawable::clone (const CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual const char* osg::ShapeDrawable::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Reimplemented from osg::Drawable.

virtual const char* osg::ShapeDrawable::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Reimplemented from osg::Drawable.

void osg::ShapeDrawable::setColor (const Vec4 & *color*)

Set the color of the shape.

const Vec4& osg::ShapeDrawable::getColor () const `[inline]`

Get the color of the shape.

virtual void osg::ShapeDrawable::drawImplementation (RenderInfo & *renderInfo*) const `[virtual]`

Draw ShapeDrawable directly ignoring an OpenGL display list which could be attached. This is the internal draw method which does the drawing itself, and is the method to override when deriving from ShapeDrawable for user-drawn objects.

Implements osg::Drawable.

virtual bool osg::ShapeDrawable::supports (const AttributeFunctor &) const `[inline, virtual]`

Return false, osg::ShapeDrawable does not support accept(AttributeFunctor&).

virtual bool osg::ShapeDrawable::supports (const Drawable::ConstAttributeFunctor &) const `[inline, virtual]`

Return true, osg::ShapeDrawable does support accept(Drawable::ConstAttributeFunctor&).

Reimplemented from osg::Drawable.

virtual void osg::ShapeDrawable::accept (Drawable::ConstAttributeFunctor & *af*) const `[virtual]`

Accept a Drawable::ConstAttributeFunctor and call its methods to tell it about the interal attributes that this Drawable has.

Reimplemented from osg::Drawable.

virtual bool osg::ShapeDrawable::supports (const PrimitiveFunctor &) const `[inline, virtual]`

Return true, osg::ShapeDrawable does support accept(PrimitiveFunctor&) .

Reimplemented from osg::Drawable.

virtual void osg::ShapeDrawable::accept (PrimitiveFunctor & *pf*) const `[virtual]`

Accept a PrimitiveFunctor and call its methods to tell it about the internal primitives that this Drawable has.

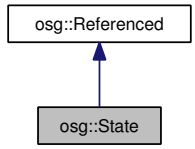
Reimplemented from osg::Drawable.

virtual BoundingBox osg::ShapeDrawable::computeBound () const `[virtual]`

Compute the bounding box around Drawables's geometry.

Reimplemented from osg::Drawable.

4.319 osg::State Class Reference



Public Types

- enum CheckForGLErrors {
  NEVER_CHECK_GL_ERRORS,
  ONCE_PER_FRAME,
  ONCE_PER_ATTRIBUTE }
- typedef std::vector< const StateSet * > **StateSetStack**

Public Member Functions

- void setGraphicsContext (GraphicsContext *context)
- GraphicsContext * getGraphicsContext ()
- const GraphicsContext * getGraphicsContext () const
- void setContextID (unsigned int contextID)
- unsigned int getContextID () const
- void pushStateSet (const StateSet *dstate)
- void popStateSet ()
- void popAllStateSets ()
- void insertStateSet (unsigned int pos, const StateSet *dstate)
- void removeStateSet (unsigned int pos)
- unsigned int getStateSetStackSize ()
- void popStateSetStackToSize (unsigned int size)
- StateSetStack & getStateSetStack ()
- void captureCurrentState (StateSet &stateset) const
- void reset ()
- const Viewport * **getCurrentViewport** () const
- void **setInitialViewMatrix** (const osg::RefMatrix *matrix)
- const osg::Matrix & **getInitialViewMatrix** () const
- const osg::Matrix & **getInitialInverseViewMatrix** () const
- void **applyProjectionMatrix** (const osg::RefMatrix *matrix)
- const osg::Matrix & **getProjectionMatrix** () const
- void **applyModelViewMatrix** (const osg::RefMatrix *matrix)

- const osg::Matrix & **getModelViewMatrix** () const
- Polytope **getViewFrustum** () const
- void apply (const StateSet *dstate)
- void apply ()
- void setModeValidity (StateAttribute::GLMode mode, bool valid)
- bool getModeValidity (StateAttribute::GLMode mode)
- void **setGlobalDefaultModeValue** (StateAttribute::GLMode mode, bool enabled)
- bool **getGlobalDefaultModeValue** (StateAttribute::GLMode mode)
- bool applyMode (StateAttribute::GLMode mode, bool enabled)
- void **setGlobalDefaultTextureModeValue** (unsigned int unit, StateAttribute::GLMode mode, bool enabled)
- bool **getGlobalDefaultTextureModeValue** (unsigned int unit, StateAttribute::GLMode mode)
- bool **applyTextureMode** (unsigned int unit, StateAttribute::GLMode mode, bool enabled)
- void **setGlobalDefaultAttribute** (const StateAttribute *attribute)
- const StateAttribute * **getGlobalDefaultAttribute** (StateAttribute::Type type, unsigned int member=0)
- bool applyAttribute (const StateAttribute *attribute)
- void **setGlobalDefaultTextureAttribute** (unsigned int unit, const StateAttribute *attribute)
- const StateAttribute * **getGlobalDefaultTextureAttribute** (unsigned int unit, StateAttribute::Type type, unsigned int member=0)
- bool **applyTextureAttribute** (unsigned int unit, const StateAttribute *attribute)
- void haveAppliedMode (StateAttribute::GLMode mode, StateAttribute::GLModeValue value)
- void haveAppliedMode (StateAttribute::GLMode mode)
- void haveAppliedAttribute (const StateAttribute *attribute)
- void haveAppliedAttribute (StateAttribute::Type type, unsigned int member=0)
- bool getLastAppliedMode (StateAttribute::GLMode mode) const
- const StateAttribute * getLastAppliedAttribute (StateAttribute::Type type, unsigned int member=0) const
- void haveAppliedTextureMode (unsigned int unit, StateAttribute::GLMode mode, StateAttribute::GLModeValue value)
- void haveAppliedTextureMode (unsigned int unit, StateAttribute::GLMode mode)
- void haveAppliedTextureAttribute (unsigned int unit, const StateAttribute *attribute)
- void haveAppliedTextureAttribute (unsigned int unit, StateAttribute::Type type, unsigned int member=0)
- bool getLastAppliedTextureMode (unsigned int unit, StateAttribute::GLMode mode) const
- const StateAttribute * getLastAppliedTextureAttribute (unsigned int unit, StateAttribute::Type type, unsigned int member=0) const
- void dirtyAllModes ()
- void dirtyAllAttributes ()
- void disableAllVertexArrays ()
- void dirtyAllVertexArrays ()
- void **setCurrentVertexBufferObject** (osg::VertexBufferObject *vbo)
- const VertexBufferObject * **getCurrentVertexBufferObject** ()
- void **bindVertexBufferObject** (const osg::VertexBufferObject *vbo)
- void **unbindVertexBufferObject** ()
- void **setCurrentElementBufferObject** (osg::ElementBufferObject *ebo)

- const ElementBufferObject * **getCurrentElementBufferObject** ()
- void **bindElementBufferObject** (const osg::ElementBufferObject *ebo)
- void **unbindElementBufferObject** ()
- void **setCurrentPixelBufferObject** (osg::PixelBufferObject *pbo)
- const PixelBufferObject * **getCurrentPixelBufferObject** ()
- void **bindPixelBufferObject** (const osg::PixelBufferObject *pbo)
- void **unbindPixelBufferObject** ()
- void setInterleavedArrays (GLenum format, GLsizei stride, const GLvoid *pointer)
- void setVertexPointer (const Array *array)
- void setVertexPointer (GLint size, GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableVertexPointer ()
- void **dirtyVertexPointer** ()
- void setNormalPointer (const Array *array)
- void setNormalPointer (GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableNormalPointer ()
- void **dirtyNormalPointer** ()
- void setColorPointer (const Array *array)
- void setColorPointer (GLint size, GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableColorPointer ()
- void **dirtyColorPointer** ()
- bool **isSecondaryColorSupported** () const
- void setSecondaryColorPointer (const Array *array)
- void setSecondaryColorPointer (GLint size, GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableSecondaryColorPointer ()
- void **dirtySecondaryColorPointer** ()
- void setIndexPointer (GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableIndexPointer ()
- void **dirtyIndexPointer** ()
- bool **isFogCoordSupported** () const
- void setFogCoordPointer (const Array *array)
- void setFogCoordPointer (GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableFogCoordPointer ()
- void **dirtyFogCoordPointer** ()
- void setTexCoordPointer (unsigned int unit, const Array *array)
- void setTexCoordPointer (unsigned int unit, GLint size, GLenum type, GLsizei stride, const GLvoid *ptr)
- void disableTexCoordPointer (unsigned int unit)
- void **dirtyTexCoordPointer** (unsigned int unit)
- void **disableTexCoordPointersAboveAndIncluding** (unsigned int unit)
- void **dirtyTexCoordPointersAboveAndIncluding** (unsigned int unit)
- bool setActiveTextureUnit (unsigned int unit)
- unsigned int getActiveTextureUnit () const
- bool setClientActiveTextureUnit (unsigned int unit)
- unsigned int getClientActiveTextureUnit () const
- void setVertexAttribPointer (unsigned int unit, const Array *array, GLboolean normalized)

- void setVertexAttribPointer (unsigned int index, GLint size, GLenum type, GLboolean normalized, GLsizei stride, const GLvoid *ptr)
- void disableVertexAttribPointer (unsigned int index)
- void **disableVertexAttribPointersAboveAndIncluding** (unsigned int index)
- void **dirtyVertexAttribPointersAboveAndIncluding** (unsigned int index)
- bool **isVertexBufferObjectSupported** () const
- void **setLastAppliedProgramObject** (const Program::PerContextProgram *program)
- const Program::PerContextProgram * **getLastAppliedProgramObject** () const
- GLint **getUniformLocation** (const std::string &name) const
- GLint **getAttribLocation** (const std::string &name) const
- void setFrameStamp (FrameStamp *fs)
- FrameStamp * getFrameStamp ()
- const FrameStamp * getFrameStamp () const
- void setDisplaySettings (DisplaySettings *vs)
- const DisplaySettings * getDisplaySettings () const
- void setAbortRenderingPtr (bool *abortPtr)
- bool getAbortRendering () const
- void setDynamicObjectRenderingCompletedCallback (DynamicObjectRenderingCompletedCallback *cb)
- DynamicObjectRenderingCompletedCallback * getDynamicObjectRenderingCompletedCallback ()
- void setDynamicObjectCount (unsigned int count, bool callCallbackOnZero=false)
- unsigned int getDynamicObjectCount () const
- void decrementDynamicObjectCount ()
- void setCheckForGLErrors (CheckForGLErrors check)
- CheckForGLErrors getCheckForGLErrors () const
- bool **checkGLErrors** (const char *str) const
- bool **checkGLErrors** (StateAttribute::GLMode mode) const
- bool **checkGLErrors** (const StateAttribute *attribute) const
- void initializeExtensionProcs ()

Classes

- struct **AttributeStack**
- struct **DynamicObjectRenderingCompletedCallback**
- struct **EnabledArrayPair**
- struct **ModeStack**
- struct **UniformStack**

4.320 Detailed Description

Encapsulates the current applied OpenGL modes, attributes and vertex arrays settings, implements lazy state updating and provides accessors for querrying the current state. . The venerable Red Book says that "OpenGL is a state machine", and this class represents the OpenGL state in OSG. Furthermore, `State` also has other important features:

- It works as a stack of states (see `pushStateSet()` and `popStateSet()`). Manipulating this stack of OpenGL states manually is seldom needed, since OSG does this in the most common situations.
- It implements lazy state updating. This means that, if one requests a state change and that particular state is already in the requested state, no OpenGL call will be made. This ensures that the OpenGL pipeline is not stalled by unncessary state changes.
- It allows to query the current OpenGL state without calls to `glGet*()`, which typically stall the graphics pipeline (see, for instance, `captureCurrentState()` and `getModelViewMatrix()`).

4.321 Member Enumeration Documentation

enum osg::State::CheckForGLErrors

Enumerator:

NEVER_CHECK_GL_ERRORS NEVER_CHECK_GL_ERRORS hints that OpenGL need not be checked for, this is the fastest option since checking for errors does incurr a small overhead.

ONCE_PER_FRAME ONCE_PER_FRAME means that OpenGl errors will be checked for once per frame, the overhead is still small, but at least OpenGL errors that are occurring will be caught, the reporting isn't fine grained enough for debugging purposes.

ONCE_PER_ATTRIBUTE ONCE_PER_ATTRIBUTE means that OpenGL errors will be checked for after every attribute is applied, allow errors to be directly associated with particular operations which makes debugging much easier.

4.322 Member Function Documentation

void osg::State::setGraphicsContext (GraphicsContext * *context*) `[inline]`

Set the graphics context associated with that owns this State object.

GraphicsContext* osg::State::getGraphicsContext () `[inline]`

Get the graphics context associated with that owns this State object.

const GraphicsContext* osg::State::getGraphicsContext () const `[inline]`

Get the const graphics context associated with that owns this State object.

void osg::State::setContextID (unsigned int *contextID*) `[inline]`

Set the current OpenGL context uniqueID. Note, it is the application developers responsibility to set up unique ID for each OpenGL context. This value is then used by osg::StateAttribute's and osg::Drawable's to help manage OpenGL display list and texture binds appropriate for each context, the contextID simply acts

as an index in local arrays that they maintain for the purpose. Typical settings for contextID are 0,1,2,3... up to the maximum number of graphics contexts you have set up. By default contextID is 0.

unsigned int osg::State::getContextID () const `[inline]`

Get the current OpenGL context unique ID.

void osg::State::pushStateSet (const StateSet * *dstate*)

Push stateset onto state stack.

void osg::State::popStateSet ()

Pop stateset off state stack.

void osg::State::popAllStateSets ()

pop all statesets off state stack, ensuring it is empty ready for the next frame. Note, to return OpenGL to default state, one should do any state.popAllStatSets(); state.apply().

void osg::State::insertStateSet (unsigned int *pos*, const StateSet * *dstate*)

Insert stateset onto state stack.

void osg::State::removeStateSet (unsigned int *pos*)

Pop stateset off state stack.

unsigned int osg::State::getStateSetStackSize () `[inline]`

Get the number of StateSet's on the StateSet stack.

void osg::State::popStateSetStackToSize (unsigned int *size*) `[inline]`

Pop StateSet's for the StateSet stack till its size equals the specified size.

StateSetStack& osg::State::getStateSetStack () `[inline]`

Get the StateSet stack.

void osg::State::captureCurrentState (StateSet & *stateset*) const

Copy the modes and attributes which capture the current state.

void osg::State::reset ()

reset the state object to an empty stack.

void osg::State::apply (const StateSet * *dstate*)

Apply stateset.

void osg::State::apply ()

Updates the OpenGL state so that it matches the `StateSet` at the top of the stack of `StateSet`s maintained internally by a `State`.

void osg::State::setModeValidity (StateAttribute::GLMode *mode*, bool *valid*) `[inline]`

Set whether a particular OpenGL mode is valid in the current graphics context. Use to disable OpenGL modes that are not supported by current graphics drivers/context.

bool osg::State::getModeValidity (StateAttribute::GLMode *mode*) `[inline]`

Get whether a particular OpenGL mode is valid in the current graphics context. Use to disable OpenGL modes that are not supported by current graphics drivers/context.

bool osg::State::applyMode (StateAttribute::GLMode *mode*, bool *enabled*) `[inline]`

Apply an OpenGL mode if required. This is a wrapper around `glEnable()` and `glDisable()`, that just actually calls these functions if the `enabled` flag is different than the current state.

Returns:

`true` if the state was actually changed. `false` otherwise. Notice that a `false` return does not indicate an error, it just means that the mode was already set to the same value as the `enabled` parameter.

bool osg::State::applyAttribute (const StateAttribute * *attribute*) `[inline]`

Apply an attribute if required.

void osg::State::haveAppliedMode (StateAttribute::GLMode *mode*, StateAttribute::GLModeValue *value*)

Mode has been set externally, update state to reflect this setting.

void osg::State::haveAppliedMode (StateAttribute::GLMode *mode*)

Mode has been set externally, therefore dirty the associated mode in osg::State so it is applied on next call to osg::State::apply(..)

void osg::State::haveAppliedAttribute (const StateAttribute * *attribute*)

Attribute has been applied externally, update state to reflect this setting.

void osg::State::haveAppliedAttribute (StateAttribute::Type *type*, unsigned int *member* = 0)

Attribute has been applied externally, and therefore this attribute type has been dirtied and will need to be re-applied on next osg::State.apply(..). note, if you have an osg::StateAttribute which you have applied externally then use the have_applied(attribute) method as this will cause the osg::State to track the current state more accurately and enable lazy state updating such that only changed state will be applied.

bool osg::State::getLastAppliedMode (StateAttribute::GLMode *mode*) const

Get whether the current specified mode is enabled (true) or disabled (false).

const StateAttribute* osg::State::getLastAppliedAttribute (StateAttribute::Type *type*, unsigned int *member* = 0) const

Get the current specified attribute, return NULL if one has not yet been applied.

void osg::State::haveAppliedTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*, StateAttribute::GLModeValue *value*)

texture Mode has been set externally, update state to reflect this setting.

void osg::State::haveAppliedTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*)

texture Mode has been set externally, therefore dirty the associated mode in osg::State so it is applied on next call to osg::State::apply(..)

void osg::State::haveAppliedTextureAttribute (unsigned int *unit*, const StateAttribute * *attribute*)

texture Attribute has been applied externally, update state to reflect this setting.

void osg::State::haveAppliedTextureAttribute (unsigned int *unit*, StateAttribute::Type *type*, unsigned int *member* = 0)

texture Attribute has been applied externally, and therefore this attribute type has been dirtied and will need to be re-appplied on next osg::State.apply(..). note, if you have an osg::StateAttribute which you have applied externally then use the have_applied(attribute) method as this will the osg::State to track the current state more accurately and enable lazy state updating such that only changed state will be applied.

bool osg::State::getLastAppliedTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*) const

Get whether the current specified texture mode is enabled (true) or disabled (false).

const StateAttribute* osg::State::getLastAppliedTextureAttribute (unsigned int *unit*, StateAttribute::Type *type*, unsigned int *member* = 0) const

Get the current specified texture attribute, return NULL if one has not yet been applied.

void osg::State::dirtyAllModes ()

Dirty the modes previously applied in osg::State.

void osg::State::dirtyAllAttributes ()

Dirty the modes attributes previously applied in osg::State.

void osg::State::disableAllVertexArrays ()

disable the vertex, normal, color, tex coords, secondary color, fog coord and index arrays.

void osg::State::dirtyAllVertexArrays ()

dirty the vertex, normal, color, tex coords, secondary color, fog coord and index arrays.

void osg::State::setInterleavedArrays (GLenum *format*, GLsizei *stride*, const GLvoid * *pointer*)

Wrapper around glInterleavedArrays(..). also resets the internal array points and modes within osg::State to keep the other vertex array operations consistent.

void osg::State::setVertexPointer (const Array * *array*) `[inline]`

Set the vertex pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setVertexPointer (GLint *size*, GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*) `[inline]`

wrapper around glEnableClientState(GL_VERTEX_ARRAY);glVertexPointer(..); note, only updates values that change.

void osg::State::disableVertexPointer () `[inline]`

wrapper around glDisableClientState(GL_VERTEX_ARRAY). note, only updates values that change.

void osg::State::setNormalPointer (const Array * *array*) `[inline]`

Set the normal pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setNormalPointer (GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*) `[inline]`

wrapper around glEnableClientState(GL_NORMAL_ARRAY);glNormalPointer(..); note, only updates values that change.

void osg::State::disableNormalPointer () `[inline]`

wrapper around glDisableClientState(GL_NORMAL_ARRAY); note, only updates values that change.

void osg::State::setColorPointer (const Array * *array*) `[inline]`

Set the color pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setColorPointer (GLint *size*, GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*) `[inline]`

wrapper around glEnableClientState(GL_COLOR_ARRAY);glColorPointer(..); note, only updates values that change.

void osg::State::disableColorPointer () `[inline]`

wrapper around glDisableClientState(GL_COLOR_ARRAY); note, only updates values that change.

void osg::State::setSecondaryColorPointer (const Array * *array*) `[inline]`

Set the secondary color pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setSecondaryColorPointer (GLint *size*, GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*)

wrapper around glEnableClientState(GL_SECONDARY_COLOR_ARRAY);glSecondayColorPointer(..); note, only updates values that change.

void osg::State::disableSecondaryColorPointer () `[inline]`

wrapper around glDisableClientState(GL_SECONDARY_COLOR_ARRAY); note, only updates values that change.

void osg::State::setIndexPointer (GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*) `[inline]`

wrapper around glEnableClientState(GL_INDEX_ARRAY);glIndexPointer(..); note, only updates values that change.

void osg::State::disableIndexPointer () `[inline]`

wrapper around glDisableClientState(GL_INDEX_ARRAY); note, only updates values that change.

void osg::State::setFogCoordPointer (const Array * *array*) `[inline]`

Set the fog coord pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setFogCoordPointer (GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*)

wrapper around glEnableClientState(GL_FOG_COORDINATE_ARRAY);glFogCoordPointer(..); note, only updates values that change.

void osg::State::disableFogCoordPointer () `[inline]`

wrapper around glDisableClientState(GL_FOG_COORDINATE_ARRAY); note, only updates values that change.

void osg::State::setTexCoordPointer (unsigned int *unit*, const Array * *array*) `[inline]`

Set the tex coord pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setTexCoordPointer (unsigned int *unit*, GLint *size*, GLenum *type*, GLsizei *stride*, const GLvoid * *ptr*) `[inline]`

wrapper around glEnableClientState(GL_TEXTURE_COORD_ARRAY);glTexCoordPointer(..); note, only updates values that change.

void osg::State::disableTexCoordPointer (unsigned int *unit*) `[inline]`

wrapper around glDisableClientState(GL_TEXTURE_COORD_ARRAY); note, only updates values that change.

bool osg::State::setActiveTextureUnit (unsigned int *unit*)

Set the current texture unit, return true if selected, false if selection failed such as when multitexturing is not supported. note, only updates values that change.

unsigned int osg::State::getActiveTextureUnit () const `[inline]`

Get the current texture unit.

bool osg::State::setClientActiveTextureUnit (unsigned int *unit*)

Set the current tex coord array texture unit, return true if selected, false if selection failed such as when multitexturing is not supported. note, only updates values that change.

unsigned int osg::State::getClientActiveTextureUnit () const `[inline]`

Get the current tex coord array texture unit.

void osg::State::setVertexAttribPointer (unsigned int *unit*, const Array * *array*, GLboolean *normalized*) `[inline]`

Set the vertex attrib pointer using an osg::Array, and manage any VBO that are required.

void osg::State::setVertexAttribPointer (unsigned int *index*, GLint *size*, GLenum *type*, GLboolean *normalized*, GLsizei *stride*, const GLvoid * *ptr*)

wrapper around glEnableVertexAttribArrayARB(index);glVertexAttribPointerARB(..); note, only updates values that change.

void osg::State::disableVertexAttribPointer (unsigned int *index*)

wrapper around DisableVertexAttribArrayARB(index); note, only updates values that change.

void osg::State::setFrameStamp (FrameStamp * *fs*) `[inline]`

Set the frame stamp for the current frame.

FrameStamp* osg::State::getFrameStamp () `[inline]`

Get the frame stamp for the current frame.

const FrameStamp* osg::State::getFrameStamp () const `[inline]`

Get the const frame stamp for the current frame.

void osg::State::setDisplaySettings (DisplaySettings * *vs*) `[inline]`

Set the DisplaySettings. Note, nothing is applied, the visual settings are just used in the State object to pass the current visual settings to Drawables during rendering.

const DisplaySettings* osg::State::getDisplaySettings () const `[inline]`

Get the DisplaySettings

void osg::State::setAbortRenderingPtr (bool * *abortPtr*) `[inline]`

Set flag for early termination of the draw traversal.

bool osg::State::getAbortRendering () const `[inline]`

Get flag for early termination of the draw traversal, if true steps should be taken to complete rendering early.

void osg::State::setDynamicObjectRenderingCompletedCallback (DynamicObjectRenderingCompletedCallback * *cb*) `[inline]`

Set the callback to be called when the dynamic object count hits 0.

DynamicObjectRenderingCompletedCallback* osg::State::getDynamicObjectRenderingCompletedCallback () const `[inline]`

Get the callback to be called when the dynamic object count hits 0.

void osg::State::setDynamicObjectCount (unsigned int *count*, bool *callCallbackOnZero* = `false`) `[inline]`

Set the number of dynamic objects that will be rendered in this graphics context this frame.

unsigned int osg::State::getDynamicObjectCount () const `[inline]`

Get the number of dynamic objects that will be rendered in this graphics context this frame.

void osg::State::decrementDynamicObjectCount () `[inline]`

Decrement the number of dynamic objects left to render this frame, and once the count goes to zero call the DynamicObjectRenderingCompletedCallback to inform of completion.

void osg::State::setCheckForGLErrors (CheckForGLErrors *check*) `[inline]`

Set whether and how often OpenGL errors should be checked for.

CheckForGLErrors osg::State::getCheckForGLErrors () const `[inline]`

Get whether and how often OpenGL errors should be checked for.

void osg::State::initializeExtensionProcs ()

Initialize extension used by osg:State.

bool osg::State::applyMode (StateAttribute::GLMode *mode*, bool *enabled*, ModeStack & *ms*) `[inline, protected]`

Apply an OpenGL mode if required, passing in mode, enable flag and appropriate mode stack. This is a wrapper around `glEnable()` and `glDisable()`, that just actually calls these functions if the `enabled` flag is different than the current state.

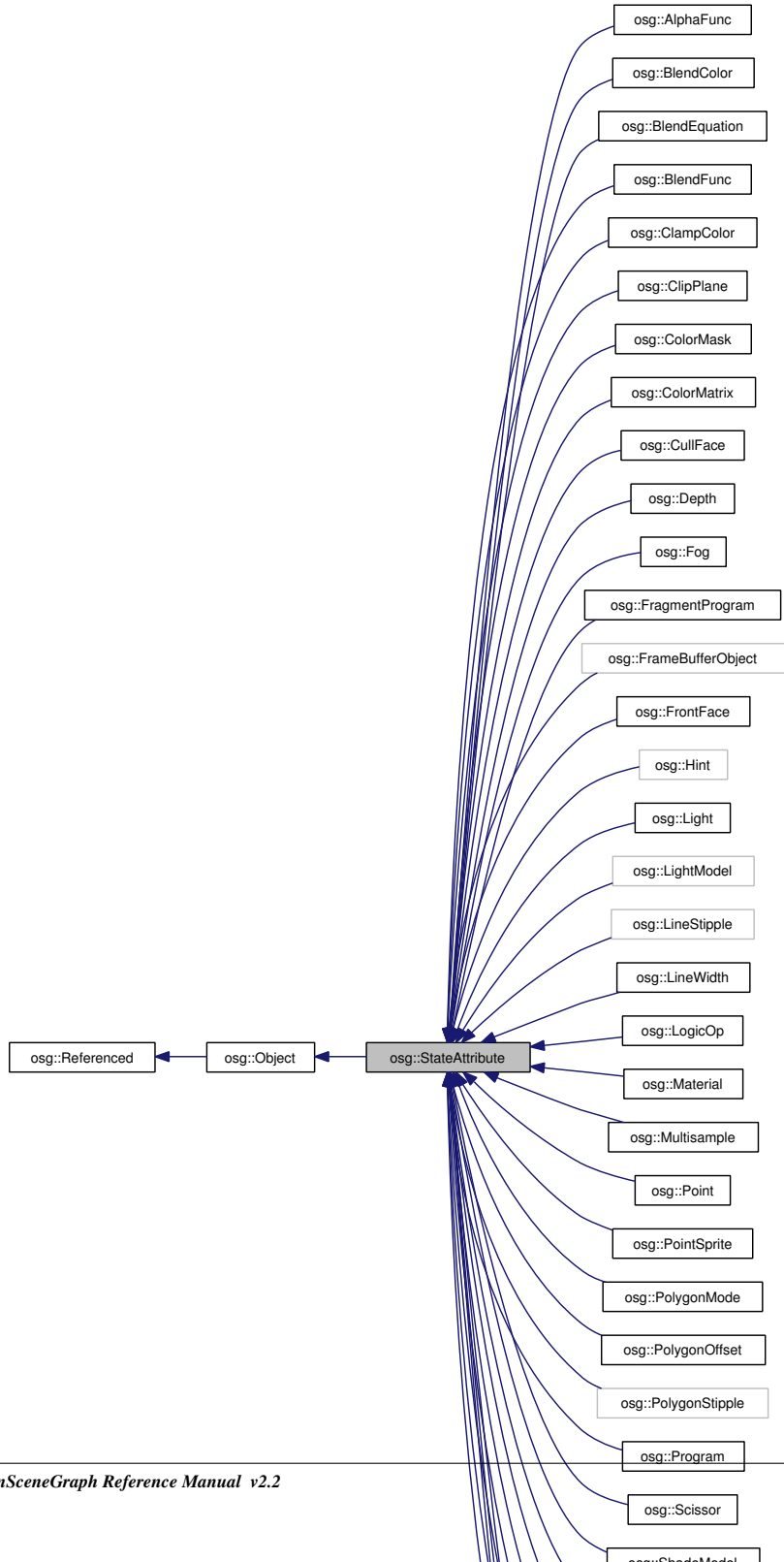
Returns:

`true` if the state was actually changed. `false` otherwise. Notice that a `false` return does not indicate an error, it just means that the mode was already set to the same value as the `enabled` parameter.

bool osg::State::applyAttribute (const StateAttribute * *attribute*, AttributeStack & *as*) `[inline, protected]`

apply an attribute if required, passing in attribute and appropriate attribute stack

4.323 osg::StateAttribute Class Reference



Public Types

- enum Values {
  OFF,
  ON,
  OVERRIDE,
  PROTECTED,
  INHERIT }
- enum Type {
  TEXTURE,
  POLYGONMODE,
  POLYGONOFFSET,
  MATERIAL,
  ALPHAFUNC,
  ANTIALIAS,
  COLORTABLE,
  CULLFACE,
  FOG,
  FRONTFACE,
  LIGHT,
  POINT,
  LINEWIDTH,
  LINESTIPPLE,
  POLYGONSTIPPLE,
  SHADEMODEL,
  TEXENV,
  TEXENVFILTER,
  TEXGEN,
  TEXMAT,
  LIGHTMODEL,
  BLENDFUNC,
  BLENDEQUATION,
  LOGICOP,
  STENCIL,
  COLORMASK,
  DEPTH,
  VIEWPORT,

SCISSOR,
BLENDCOLOR,
MULTISAMPLE,
CLIPPLANE,
COLORMATRIX,
VERTEXPROGRAM,
FRAGMENTPROGRAM,
POINTSPRITE,
PROGRAM,
CLAMPCOLOR,
HINT,
VALIDATOR,
VIEWMATRIXEXTRACTOR,
OSGNV_PARAMETER_BLOCK,
OSGNVEXT_TEXTURE_SHADER,
OSGNVEXT_VERTEX_PROGRAM,
OSGNVEXT_REGISTER_COMBINERS,
OSGNVCG_PROGRAM,
OSGNVSLANG_PROGRAM,
OSGNVPARSE_PROGRAM_PARSER }

- typedef GLenum GLMode
- typedef unsigned int GLModeValue
- typedef unsigned int OverrideValue
- typedef std::pair< Type, unsigned int > TypeMemberPair
- typedef std::vector< StateSet * > ParentList

Public Member Functions

- **StateAttribute** (const StateAttribute &sa, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const =0
- virtual Object * clone (const CopyOp &) const =0
- virtual bool isSameKindAs (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Type getType () const =0
- virtual unsigned int getMember () const
- TypeMemberPair getTypeMemberPair () const
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const =0
- bool **operator**< (const StateAttribute &rhs) const

- bool **operator==** (const StateAttribute &rhs) const
- bool **operator!=** (const StateAttribute &rhs) const
- const ParentList & getParents () const
- StateSet * **getParent** (unsigned int i)
- const StateSet * getParent (unsigned int i) const
- unsigned int getNumParents () const
- virtual bool getModeUsage (ModeUsage &) const
- virtual bool checkValidityOfAssociatedModes (osg::State &) const
- void setUpdateCallback (Callback *uc)
- Callback * getUpdateCallback ()
- const Callback * getUpdateCallback () const
- void setEventCallback (Callback *ec)
- Callback * getEventCallback ()
- const Callback * getEventCallback () const
- virtual void apply (State &) const
- virtual void compileGLObjects (State &) const
- virtual void resizeGLObjectBuffers (unsigned int)
- virtual void releaseGLObjects (State *=0) const

Friends

- class osg::StateSet

Classes

- struct **Callback**
- struct **ModeUsage**

4.324 Detailed Description

Base class for state attributes.

4.325 Member Typedef Documentation

typedef GLenum osg::StateAttribute::GLMode

GLMode is the value used in glEnable/glDisable(mode)

typedef unsigned int osg::StateAttribute::GLModeValue

GLModeValue is used to specify whether a mode is enabled (ON) or disabled (OFF). GLMoveValue is also used to specify the override behavior of modes from parent to children. See enum Value description for more details.

typedef unsigned int osg::StateAttribute::OverrideValue

Override is used to specify the override behavior of StateAttributes from parent to children. See enum Value description for more details.

typedef std::pair<Type,unsigned int> osg::StateAttribute::TypeMemberPair

Simple pairing between an attribute type and the member within that attribute type group.

typedef std::vector<StateSet*> osg::StateAttribute::ParentList

A vector of osg::StateSet pointers which is used to store the parent(s) of this StateAttribute.

4.326 Member Enumeration Documentation

enum osg::StateAttribute::Values

list values which can be used to set either GLModeValues or OverrideValues. When using in conjunction with GLModeValues, all Values have meaning. When using in conjection with StateAttribute OverrideValue only OFF,OVERRIDE and INHERIT are meaningful. However, they are useful when using GLModeValue and OverrideValue in conjunction with each other as when using StateSet::setAttributeAndModes(..).

Enumerator:

OFF means that associated GLMode and Override is disabled.

ON means that associated GLMode is enabled and Override is disabled.

OVERRIDE Overriding of GLMode's or StateAttributes is enabled, so that state below it is overridden.

PROTECTED Protecting of GLMode's or StateAttributes is enabled, so that state from above cannot override this and below state.

INHERIT means that GLMode or StateAttribute should be inherited from above.

enum osg::StateAttribute::Type

Type identifier to differentiate between different state types. Values of StateAttribute::Type used to aid identification of different StateAttribute subclasses. Each subclass defines its own value in the virtual Type getType() method. When extending the osg's StateAttribute's simply define your own Type value which is unique, using the StateAttribute::Type enum as a guide of what values to use. If your new subclass needs to override a standard StateAttriubte then simply use that type's value.

Enumerator:

VALIDATOR osgFX namespace

OSGNV_PARAMETER_BLOCK osgNV namespace

OSGNVCG_PROGRAM osgNVCg namespace

4.327 Member Function Documentation

virtual Object* osg::StateAttribute::cloneType () const `[pure virtual]`

Clone the type of an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

Implemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual Object* osg::StateAttribute::clone (const CopyOp &) const `[pure virtual]`

Clone an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

Implemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual bool osg::StateAttribute::isSameKindAs (const Object * *obj*) const `[inline, virtual]`

Return true if this and obj are of the same kind of object.

Reimplemented from osg::Object.

Reimplemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual const char* osg::StateAttribute::libraryName () const `[inline, virtual]`

Return the name of the attribute's library.

Implements osg::Object.

Reimplemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual const char* osg::StateAttribute::className () const `[inline, virtual]`

Return the name of the attribute's class type.

Implements osg::Object.

Reimplemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual Type osg::StateAttribute::getType () const `[pure virtual]`

Return the Type identifier of the attribute's class type.

Implemented in osg::ClipPlane, osg::Light, and osg::Texture.

virtual unsigned int osg::StateAttribute::getMember () const `[inline, virtual]`

Return the member identifier within the attribute's class type. Used for light number/clip plane number etc.

Reimplemented in osg::ClipPlane, and osg::Light.

TypeMemberPair osg::StateAttribute::getTypeMemberPair () const `[inline]`

Return the TypeMemberPair that uniquely identifies this type member.

virtual bool osg::StateAttribute::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented in osg::PointSprite, osg::TexEnv, osg::TexEnvCombine, osg::TexEnvFilter, osg::TexGen, osg::TexMat, and osg::Texture.

virtual int osg::StateAttribute::compare (const StateAttribute & *sa*) const `[pure virtual]`

Return -1 if ∗this < ∗rhs, 0 if ∗this==∗rhs, 1 if ∗this>∗rhs.

Implemented in osg::AlphaFunc, osg::BlendColor, osg::BlendEquation, osg::BlendFunc, osg::ClampColor, osg::ClipPlane, osg::ColorMask, osg::ColorMatrix, osg::CullFace, osg::Depth, osg::Fog, osg::FragmentProgram, osg::FrontFace, osg::Light, osg::LineWidth, osg::LogicOp, osg::Material, osg::Multisample, osg::Point, osg::PointSprite, osg::PolygonMode, osg::PolygonOffset, osg::Program, osg::Scissor, osg::ShadeModel, osg::Stencil, osg::StencilTwoSided, osg::TexEnv, osg::TexEnvCombine, osg::TexEnvFilter, osg::TexGen, osg::TexMat, osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, osg::TextureRectangle, osg::VertexProgram, and osg::Viewport.

const ParentList& osg::StateAttribute::getParents () const `[inline]`

Get the parent list of this StateAttribute.

const StateSet∗ osg::StateAttribute::getParent (unsigned int *i*) const `[inline]`

Get a single const parent of this StateAttribute.

Parameters:

i index of the parent to get.

Returns:
the parent i.

unsigned int osg::StateAttribute::getNumParents () const `[inline]`

Get the number of parents of this StateAttribute.

Returns:
the number of parents of this StateAttribute.

virtual bool osg::StateAttribute::getModeUsage (ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented in osg::AlphaFunc, osg::BlendColor, osg::BlendEquation, osg::BlendFunc, osg::ClipPlane, osg::CullFace, osg::Depth, osg::Fog, osg::FragmentProgram, osg::Light, osg::LogicOp, osg::Material, osg::Point, osg::PointSprite, osg::PolygonOffset, osg::Scissor, osg::Stencil, osg::StencilTwoSided, osg::TexGen, osg::Texture, and osg::VertexProgram.

virtual bool osg::StateAttribute::checkValidityOfAssociatedModes (osg::State &) const `[inline, virtual]`

Check the modes associated with this StateAttribute are supported by current OpenGL drivers, and if not set the associated mode in osg::State to be black listed/invalid. Return true if all associated modes are valid.

Reimplemented in osg::PointSprite.

void osg::StateAttribute::setUpdateCallback (Callback * *uc*)

Set the UpdateCallback which allows users to attach customize the updating of an object during the update traversal.

Callback* osg::StateAttribute::getUpdateCallback () `[inline]`

Get the non const UpdateCallback.

const Callback* osg::StateAttribute::getUpdateCallback () const `[inline]`

Get the const UpdateCallback.

void osg::StateAttribute::setEventCallback (Callback * *ec*)

Set the EventCallback which allows users to attach customize the updating of an object during the Event traversal.

Callback∗ osg::StateAttribute::getEventCallback () `[inline]`

Get the non const EventCallback.

const Callback∗ osg::StateAttribute::getEventCallback () const `[inline]`

Get the const EventCallback.

virtual void osg::StateAttribute::apply (State &) const `[inline, virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented in osg::AlphaFunc, osg::BlendColor, osg::BlendEquation, osg::BlendFunc, osg::ClampColor, osg::ClipPlane, osg::ColorMask, osg::ColorMatrix, osg::CullFace, osg::Depth, osg::Fog, osg::FragmentProgram, osg::FrontFace, osg::Light, osg::LineWidth, osg::LogicOp, osg::Material, osg::Multisample, osg::Point, osg::PointSprite, osg::PolygonMode, osg::PolygonOffset, osg::Program, osg::Scissor, osg::ShadeModel, osg::Stencil, osg::StencilTwoSided, osg::TexEnv, osg::TexEnvCombine, osg::TexEnvFilter, osg::TexGen, osg::TexMat, osg::Texture, osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, osg::TextureRectangle, osg::VertexProgram, and osg::Viewport.

virtual void osg::StateAttribute::compileGLObjects (State &) const `[inline, virtual]`

Default to nothing to compile - all state is applied immediately.

Reimplemented in osg::FragmentProgram, osg::Program, osg::Texture, and osg::VertexProgram.

virtual void osg::StateAttribute::resizeGLObjectBuffers (unsigned int) `[inline, virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::Object.

Reimplemented in osg::FragmentProgram, osg::Program, osg::Texture, and osg::VertexProgram.

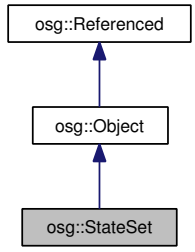
virtual void osg::StateAttribute::releaseGLObjects (State ∗ = 0) const `[inline, virtual]`

Release OpenGL objects in specified graphics context if State object is passed, otherwise release OpenGL objexts for all graphics context if State object pointer NULL.

Reimplemented from osg::Object.

Reimplemented in osg::FragmentProgram, osg::Program, osg::Texture, and osg::VertexProgram.

4.328 osg::StateSet Class Reference



Public Types

- enum **RenderingHint** {
  DEFAULT_BIN,
  OPAQUE_BIN,
  TRANSPARENT_BIN }
- enum **RenderBinMode** {
  INHERIT_RENDERBIN_DETAILS,
  USE_RENDERBIN_DETAILS,
  OVERRIDE_RENDERBIN_DETAILS }
- typedef std::vector< Object * > ParentList
- typedef std::map< StateAttribute::GLMode, StateAttribute::GLModeValue > ModeList
- typedef std::pair< ref_ptr< StateAttribute >, StateAttribute::OverrideValue > RefAttributePair
- typedef std::map< StateAttribute::TypeMemberPair, RefAttributePair > AttributeList
- typedef std::vector< ModeList > **TextureModeList**
- typedef std::vector< AttributeList > **TextureAttributeList**
- typedef std::pair< ref_ptr< Uniform >, StateAttribute::OverrideValue > RefUniformPair
- typedef std::map< std::string, RefUniformPair > UniformList

Public Member Functions

- **StateSet** (const StateSet &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual Object * cloneType () const
- virtual Object * clone (const CopyOp &copyop) const
- virtual bool **isSameKindAs** (const Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- int compare (const StateSet &rhs, bool compareAttributeContents=false) const
- bool **operator<** (const StateSet &rhs) const

- bool **operator==** (const StateSet &rhs) const
- bool **operator!=** (const StateSet &rhs) const
- const ParentList & getParents () const
- ParentList getParents ()
- Object * **getParent** (unsigned int i)
- const Object * getParent (unsigned int i) const
- unsigned int getNumParents () const
- virtual void computeDataVariance ()
- void setGlobalDefaults ()
- void clear ()
- void merge (const StateSet &rhs)
- void setMode (StateAttribute::GLMode mode, StateAttribute::GLModeValue value)
- void removeMode (StateAttribute::GLMode mode)
- StateAttribute::GLModeValue getMode (StateAttribute::GLMode mode) const
- void setModeList (ModeList &ml)
- ModeList & getModeList ()
- const ModeList & getModeList () const
- void setAttribute (StateAttribute *attribute, StateAttribute::OverrideValue value=StateAttribute::OFF)
- void setAttributeAndModes (StateAttribute *attribute, StateAttribute::GLModeValue value=StateAttribute::ON)
- void removeAttribute (StateAttribute::Type type, unsigned int member=0)
- void removeAttribute (StateAttribute *attribute)
- StateAttribute * getAttribute (StateAttribute::Type type, unsigned int member=0)
- const StateAttribute * getAttribute (StateAttribute::Type type, unsigned int member=0) const
- const RefAttributePair * getAttributePair (StateAttribute::Type type, unsigned int member=0) const
- void setAttributeList (AttributeList &al)
- AttributeList & getAttributeList ()
- const AttributeList & getAttributeList () const
- void setTextureMode (unsigned int unit, StateAttribute::GLMode mode, StateAttribute::GLModeValue value)
- void removeTextureMode (unsigned int unit, StateAttribute::GLMode mode)
- StateAttribute::GLModeValue getTextureMode (unsigned int unit, StateAttribute::GLMode mode) const
- void setTextureModeList (TextureModeList &tml)
- TextureModeList & getTextureModeList ()
- const TextureModeList & getTextureModeList () const
- void setTextureAttribute (unsigned int unit, StateAttribute *attribute, StateAttribute::OverrideValue value=StateAttribute::OFF)
- void setTextureAttributeAndModes (unsigned int unit, StateAttribute *attribute, StateAttribute::GLModeValue value=StateAttribute::ON)
- void removeTextureAttribute (unsigned int unit, StateAttribute::Type type)
- void removeTextureAttribute (unsigned int unit, StateAttribute *attribute)
- StateAttribute * getTextureAttribute (unsigned int unit, StateAttribute::Type type)
- const StateAttribute * getTextureAttribute (unsigned int unit, StateAttribute::Type type) const
- const RefAttributePair * getTextureAttributePair (unsigned int unit, StateAttribute::Type type) const
- void setTextureAttributeList (TextureAttributeList &tal)

- TextureAttributeList & getTextureAttributeList ()
- const TextureAttributeList & getTextureAttributeList () const
- void **setAssociatedModes** (const StateAttribute *attribute, StateAttribute::GLModeValue value)
- void **setAssociatedTextureModes** (unsigned int unit, const StateAttribute *attribute, StateAttribute::GLModeValue value)
- void addUniform (Uniform *uniform, StateAttribute::OverrideValue value=StateAttribute::ON)
- void removeUniform (const std::string &name)
- void removeUniform (Uniform *uniform)
- Uniform * getUniform (const std::string &name)
- Uniform * getOrCreateUniform (const std::string &name, Uniform::Type type, unsigned int numElements=1)
- const Uniform * getUniform (const std::string &name) const
- const RefUniformPair * getUniformPair (const std::string &name) const
- void setUniformList (UniformList &al)
- UniformList & getUniformList ()
- const UniformList & getUniformList () const
- void setRenderingHint (int hint)
- int getRenderingHint () const
- void setRenderBinDetails (int binNum, const std::string &binName, RenderBinMode mode=USE_RENDERBIN_DETAILS)
- void setRenderBinToInherit ()
- bool useRenderBinDetails () const
- void setRenderBinMode (RenderBinMode mode)
- RenderBinMode getRenderBinMode () const
- void setBinNumber (int num)
- int getBinNumber () const
- void setBinName (const std::string &name)
- const std::string & getBinName () const
- void setUpdateCallback (Callback *ac)
- Callback * getUpdateCallback ()
- const Callback * getUpdateCallback () const
- bool requiresUpdateTraversal () const
- unsigned int getNumChildrenRequiringUpdateTraversal () const
- void runUpdateCallbacks (osg::NodeVisitor *nv)
- void setEventCallback (Callback *ac)
- Callback * getEventCallback ()
- const Callback * getEventCallback () const
- bool requiresEventTraversal () const
- unsigned int getNumChildrenRequiringEventTraversal () const
- void runEventCallbacks (osg::NodeVisitor *nv)
- bool checkValidityOfAssociatedModes (State &state) const
- virtual void setThreadSafeRefUnref (bool threadSafe)
- void compileGLObjects (State &state) const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (State *state=0) const

Friends

- class osg::Node
- class osg::Drawable
- class osg::Uniform
- class osg::StateAttribute

Classes

- struct **Callback**

4.329 Detailed Description

Stores a set of modes and attributes which respresent a set of OpenGL state. Notice that a `StateSet` contains just a subset of the whole OpenGL state.

In OSG, each `Drawable` and each `Node` has a reference to a `StateSet`. These `StateSet`s can be shared between different `Drawable`s and `Node`s (that is, several `Drawable`s and `Node`s can reference the same `StateSet`). Indeed, this practice is recommended whenever possible, as this minimizes expensive state changes in the graphics pipeline.

4.330 Member Typedef Documentation

typedef std::vector<Object*> osg::StateSet::ParentList

A vector of osg::Object pointers which is used to store the parent(s) of this Stateset, the parents could be osg::Node or osg::Drawable.

typedef std::map<StateAttribute::GLMode,StateAttribute::GLModeValue> osg::StateSet::ModeList

a container to map GLModes to their respective GLModeValues.

typedef std::pair<ref_ptr<StateAttribute>,StateAttribute::OverrideValue> osg::StateSet::RefAttributePair

Simple pairing between an attribute and its override flag.

typedef std::map<StateAttribute::TypeMemberPair,RefAttributePair> osg::StateSet::AttributeList

a container to map <StateAttribyte::Types,Member> to their respective RefAttributePair.

typedef std::pair<ref_ptr<Uniform>,StateAttribute::OverrideValue> osg::StateSet::RefUniformPair

Simple pairing between a Uniform and its override flag.

typedef std::map<std::string,RefUniformPair> osg::StateSet::UniformList

a container to map Uniform name to its respective RefUniformPair.

4.331 Member Function Documentation

virtual Object* osg::StateSet::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual Object* osg::StateSet::clone (const CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual const char* osg::StateSet::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

virtual const char* osg::StateSet::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

int osg::StateSet::compare (const StateSet & *rhs*, bool *compareAttributeContents* = `false`) const

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

const ParentList& osg::StateSet::getParents () const `[inline]`

Get the parent list of this StateSet.

ParentList osg::StateSet::getParents () `[inline]`

Get the a copy of parent list of node. A copy is returned to prevent modification of the parent list.

const Object* osg::StateSet::getParent (unsigned int *i*) const `[inline]`

Get a single const parent of this StateSet.

Parameters:
i index of the parent to get.

Returns:
the parent i.

unsigned int osg::StateSet::getNumParents () const `[inline]`

Get the number of parents of this StateSet.

Returns:
the number of parents of this StateSet.

virtual void osg::StateSet::computeDataVariance () `[virtual]`

Compute the DataVariance based on an assestment of callback etc.

Reimplemented from osg::Object.

void osg::StateSet::setGlobalDefaults ()

Set all the modes to on or off so that it defines a complete state, typically used for a default global state.

void osg::StateSet::clear ()

Clear the StateSet of all modes and attributes.

void osg::StateSet::merge (const StateSet & *rhs*)

Merge this `StateSet` with the `StateSet` passed as parameter. Every mode and attribute in this `StateSet` that is marked with `StateAttribute::OVERRIDE` is replaced with the equivalent mode or attribute from `rhs`.

void osg::StateSet::setMode (StateAttribute::GLMode *mode*, StateAttribute::GLModeValue *value*)

Set this `StateSet` to contain the specified `GLMode` with a given value.

Note:

Don't use this method to set modes related to textures. For this purpose, use `setTextureMode()`, that accepts an extra parameter specifying which texture unit shall be affected by the call.

void osg::StateSet::removeMode (StateAttribute::GLMode *mode*)

Remove `mode` from this `StateSet`.

Note:

Don't use this method to remove modes related to textures. For this purpose, use `removeTextureMode()`, that accepts an extra parameter specifying which texture unit shall be affected by the call.

StateAttribute::GLModeValue osg::StateSet::getMode (StateAttribute::GLMode *mode*) const

Get the value for a given `GLMode`.

Parameters:

mode The `GLMode` whose value is desired.

Returns:

If `mode` is contained within this `StateSet`, returns the value associated with it. Otherwise, returns `StateAttribute::INHERIT`.

Note:

Don't use this method to get the value of modes related to textures. For this purpose, use `removeTextureMode()`, that accepts an extra parameter specifying which texture unit shall be affected by the call.

void osg::StateSet::setModeList (ModeList & *ml*) `[inline]`

Set the list of all `GLModes` contained in this `StateSet`.

ModeList& osg::StateSet::getModeList () `[inline]`

Return the list of all `GLModes` contained in this `StateSet`.

const ModeList& osg::StateSet::getModeList () const `[inline]`

Return the `const` list of all `GLModes` contained in this `const StateSet`.

void osg::StateSet::setAttribute (StateAttribute * *attribute*, StateAttribute::OverrideValue *value* = StateAttribute::OFF)

Set this StateSet to contain specified attribute and override flag.

void osg::StateSet::setAttributeAndModes (StateAttribute * *attribute*, StateAttribute::GLModeValue *value* = `StateAttribute::ON`)

Set this StateSet to contain specified attribute and set the associated GLMode's to specified value.

void osg::StateSet::removeAttribute (StateAttribute::Type *type*, unsigned int *member* = `0`)

remove attribute of specified type from StateSet.

void osg::StateSet::removeAttribute (StateAttribute * *attribute*)

remove attribute from StateSet.

StateAttribute* osg::StateSet::getAttribute (StateAttribute::Type *type*, unsigned int *member* = `0`)

Get specified StateAttribute for specified type. Returns NULL if no type is contained within StateSet.

const StateAttribute* osg::StateSet::getAttribute (StateAttribute::Type *type*, unsigned int *member* = `0`) const

Get specified const StateAttribute for specified type. Returns NULL if no type is contained within const StateSet.

const RefAttributePair* osg::StateSet::getAttributePair (StateAttribute::Type *type*, unsigned int *member* = `0`) const

Get specified RefAttributePair for specified type. Returns NULL if no type is contained within StateSet.

void osg::StateSet::setAttributeList (AttributeList & *al*) `[inline]`

set the list of all StateAttributes contained in this StateSet.

AttributeList& osg::StateSet::getAttributeList () `[inline]`

return the list of all StateAttributes contained in this StateSet.

const AttributeList& osg::StateSet::getAttributeList () const `[inline]`

return the const list of all StateAttributes contained in this const StateSet.

void osg::StateSet::setTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*, StateAttribute::GLModeValue *value*)

Set this `StateSet` to contain specified `GLMode` with a given value.

Parameters:

unit The texture unit to be affected (used with multi-texturing).

mode The OpenGL mode to be added to the `StateSet`.

value The value to be assigned to `mode`.

void osg::StateSet::removeTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*)

Remove texture mode from StateSet.

StateAttribute::GLModeValue osg::StateSet::getTextureMode (unsigned int *unit*, StateAttribute::GLMode *mode*) const

Get specified GLModeValue for specified GLMode. returns INHERIT if no GLModeValue is contained within StateSet.

void osg::StateSet::setTextureModeList (TextureModeList & *tml*) `[inline]`

set the list of all Texture related GLModes contained in this StateSet.

TextureModeList& osg::StateSet::getTextureModeList () `[inline]`

return the list of all Texture related GLModes contained in this StateSet.

const TextureModeList& osg::StateSet::getTextureModeList () const `[inline]`

return the const list of all Texture related GLModes contained in this const StateSet.

void osg::StateSet::setTextureAttribute (unsigned int *unit*, StateAttribute * *attribute*, StateAttribute::OverrideValue *value* = `StateAttribute::OFF`)

Set this StateSet to contain specified attribute and override flag.

void osg::StateSet::setTextureAttributeAndModes (unsigned int *unit*, StateAttribute * *attribute*, StateAttribute::GLModeValue *value* = `StateAttribute::ON`)

Set this StateSet to contain specified attribute and set the associated GLMode's to specified value.

void osg::StateSet::removeTextureAttribute (unsigned int *unit*, StateAttribute::Type *type*)

remove texture attribute of specified type from StateSet.

void osg::StateSet::removeTextureAttribute (unsigned int *unit*, StateAttribute * *attribute*)

remove texture attribute from StateSet.

StateAttribute* osg::StateSet::getTextureAttribute (unsigned int *unit*, StateAttribute::Type *type*)

Get specified Texture related StateAttribute for specified type. Returns NULL if no type is contained within StateSet.

const StateAttribute* osg::StateSet::getTextureAttribute (unsigned int *unit*, StateAttribute::Type *type*) const

Get specified Texture related const StateAttribute for specified type. Returns NULL if no type is contained within const StateSet.

const RefAttributePair* osg::StateSet::getTextureAttributePair (unsigned int *unit*, StateAttribute::Type *type*) const

Get specified Texture related RefAttributePair for specified type. Returns NULL if no type is contained within StateSet.

void osg::StateSet::setTextureAttributeList (TextureAttributeList & *tal*) `[inline]`

Set the list of all Texture related StateAttributes contained in this StateSet.

TextureAttributeList& osg::StateSet::getTextureAttributeList () `[inline]`

Return the list of all Texture related StateAttributes contained in this StateSet.

const TextureAttributeList& osg::StateSet::getTextureAttributeList () const `[inline]`

Return the const list of all Texture related StateAttributes contained in this const StateSet.

void osg::StateSet::addUniform (Uniform * *uniform*, StateAttribute::OverrideValue *value* = `StateAttribute::ON`)

Set this StateSet to contain specified uniform and override flag.

void osg::StateSet::removeUniform (const std::string & *name*)

remove uniform of specified name from StateSet.

void osg::StateSet::removeUniform (Uniform * *uniform*)

remove Uniform from StateSet.

Uniform* osg::StateSet::getUniform (const std::string & *name*)

Get Uniform for specified name. Returns NULL if no matching Uniform is contained within StateSet.

Uniform* osg::StateSet::getOrCreateUniform (const std::string & *name*, Uniform::Type *type*, unsigned int *numElements* = 1)

Get Uniform for specified name, if one is not available create it, add it to this StateSet and return a pointer to it.

const Uniform* osg::StateSet::getUniform (const std::string & *name*) const

Get const Uniform for specified name. Returns NULL if no matching Uniform is contained within StateSet.

const RefUniformPair* osg::StateSet::getUniformPair (const std::string & *name*) const

Get specified RefUniformPair for specified Uniform name. Returns NULL if no Uniform is contained within StateSet.

void osg::StateSet::setUniformList (UniformList & *al*) `[inline]`

set the list of all Uniforms contained in this StateSet.

UniformList& osg::StateSet::getUniformList () `[inline]`

return the list of all Uniforms contained in this StateSet.

const UniformList& osg::StateSet::getUniformList () const `[inline]`

return the const list of all Uniforms contained in this const StateSet.

void osg::StateSet::setRenderingHint (int *hint*)

Set the `RenderingHint` of this `StateSet`. `RenderingHint` is used by the renderer to determine which draw bin to drop associated `osg::Drawable`s in. Typically, users will set this to either `StateSet::OPAQUE_BIN` or `StateSet::TRANSPARENT_BIN`. `Drawable`s in the opaque bin are sorted by their `StateSet`, so that the number of expensive changes in the OpenGL state is minimized. `Drawable`s in the transparent bin are sorted by depth, so that objects farther from the viewer are rendered first (and hence alpha blending works nicely for translucent objects).

int osg::StateSet::getRenderingHint () const `[inline]`

Get the `RenderingHint` of this `StateSet`.

void osg::StateSet::setRenderBinDetails (int *binNum*, const std::string & *binName*, RenderBinMode *mode* = USE_RENDERBIN_DETAILS)

Set the render bin details.

void osg::StateSet::setRenderBinToInherit ()

Set the render bin details to inherit.

bool osg::StateSet::useRenderBinDetails () const `[inline]`

Get whether the render bin details are set and should be used.

void osg::StateSet::setRenderBinMode (RenderBinMode *mode*) `[inline]`

Set the render bin mode.

RenderBinMode osg::StateSet::getRenderBinMode () const `[inline]`

Get the render bin mode.

void osg::StateSet::setBinNumber (int *num*) `[inline]`

Set the render bin number.

int osg::StateSet::getBinNumber () const `[inline]`

Get the render bin number.

void osg::StateSet::setBinName (const std::string & *name*) `[inline]`

Set the render bin name.

const std::string& osg::StateSet::getBinName () const `[inline]`

Get the render bin name.

void osg::StateSet::setUpdateCallback (Callback * *ac*)

Set the Update Callback which allows users to attach customize the updating of an object during the update traversal.

Callback* osg::StateSet::getUpdateCallback () `[inline]`

Get the non const Update Callback.

const Callback* osg::StateSet::getUpdateCallback () const `[inline]`

Get the const Update Callback.

bool osg::StateSet::requiresUpdateTraversal () const `[inline]`

Return whether this StateSet has update callbacks associated with it, and therefore must be traversed.

unsigned int osg::StateSet::getNumChildrenRequiringUpdateTraversal () const `[inline]`

Get the number of Objects of this StateSet which require Update traversal, since they have an Update Callback attached to them or their children.

void osg::StateSet::runUpdateCallbacks (osg::NodeVisitor * *nv*)

Run the update callbacks attached directly to this StateSet or to its children.

void osg::StateSet::setEventCallback (Callback * *ac*)

Set the Event Callback which allows users to attach customize the updating of an object during the event traversal.

Callback* osg::StateSet::getEventCallback () `[inline]`

Get the non const Event Callback.

const Callback* osg::StateSet::getEventCallback () const `[inline]`

Get the const Event Callback.

bool osg::StateSet::requiresEventTraversal () const `[inline]`

Return whether this StateSet has event callbacks associated with it, and therefore must be traversed.

unsigned int osg::StateSet::getNumChildrenRequiringEventTraversal () const `[inline]`

Get the number of Objects of this StateSet which require Event traversal, since they have an Eevnt Callback attached to them or their children.

void osg::StateSet::runEventCallbacks (osg::NodeVisitor * *nv*)

Run the event callbacks attached directly to this StateSet or to its children.

bool osg::StateSet::checkValidityOfAssociatedModes (State & *state*) const

Check the modes associated with this StateSet are supported by current OpenGL drivers, and if not set the associated mode in osg::State to be black listed/invalid. Return true if all associated modes are valid.

virtual void osg::StateSet::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Referenced.

void osg::StateSet::compileGLObjects (State & *state*) const

call compile on all StateAttributes contained within this StateSet.

virtual void osg::StateSet::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

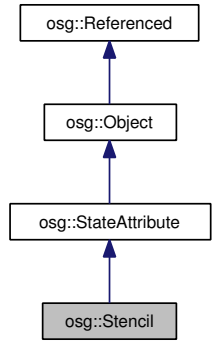
Reimplemented from osg::Object.

virtual void osg::StateSet::releaseGLObjects (State * *state* = 0) const `[virtual]`

call release on all StateAttributes contained within this StateSet.

Reimplemented from osg::Object.

4.332 osg::Stencil Class Reference



Public Types

- enum **Function** {
  NEVER,
  LESS,
  EQUAL,

LEQUAL,

GREATER,

NOTEQUAL,

GEQUAL,

ALWAYS }

- enum **Operation** {

KEEP,

ZERO,

REPLACE,

INCR,

DECR,

INVERT,

INCR_WRAP,

DECR_WRAP }

Public Member Functions

- Stencil (const Stencil &stencil, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Stencil, STENCIL)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFunction** (Function func, int ref, unsigned int mask)
- void **setFunction** (Function func)
- Function **getFunction** () const
- void **setFunctionRef** (int ref)
- int **getFunctionRef** () const
- void **setFunctionMask** (unsigned int mask)
- unsigned int **getFunctionMask** () const
- void setOperation (Operation sfail, Operation zfail, Operation zpass)
- void setStencilFailOperation (Operation sfail)
- Operation getStencilFailOperation () const
- void setStencilPassAndDepthFailOperation (Operation zfail)
- Operation getStencilPassAndDepthFailOperation () const
- void setStencilPassAndDepthPassOperation (Operation zpass)
- Operation getStencilPassAndDepthPassOperation () const
- void **setWriteMask** (unsigned int mask)
- unsigned int **getWriteMask** () const
- virtual void apply (State &state) const

4.333 Detailed Description

Encapsulate OpenGL glStencilFunc/Op/Mask functions.

4.334 Constructor & Destructor Documentation

osg::Stencil::Stencil (const Stencil & *stencil*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.335 Member Function Documentation

virtual int osg::Stencil::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::Stencil::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

void osg::Stencil::setOperation (Operation *sfail*, Operation *zfail*, Operation *zpass*) `[inline]`

set the operations to apply when the various stencil and depth tests fail or pass. First parameter is to control the operation when the stencil test fails. The second parameter is to control the operation when the stencil test passes, but depth test fails. The third parameter controls the operation when both the stencil test and depth pass. Ordering of parameter is the same as if using glStencilOp(,,).

void osg::Stencil::setStencilFailOperation (Operation *sfail*) `[inline]`

set the operation when the stencil test fails.

Operation osg::Stencil::getStencilFailOperation () const `[inline]`

get the operation when the stencil test fails.

void osg::Stencil::setStencilPassAndDepthFailOperation (Operation *zfail*) `[inline]`

set the operation when the stencil test passes but the depth test fails.

Operation osg::Stencil::getStencilPassAndDepthFailOperation () const `[inline]`

get the operation when the stencil test passes but the depth test fails.

void osg::Stencil::setStencilPassAndDepthPassOperation (Operation *zpass*) `[inline]`

set the operation when both the stencil test and the depth test pass.

Operation osg::Stencil::getStencilPassAndDepthPassOperation () const `[inline]`

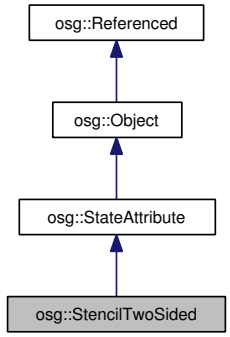
get the operation when both the stencil test and the depth test pass.

virtual void osg::Stencil::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.336 osg::StencilTwoSided Class Reference



Public Types

- enum **Face** {
  FRONT,
  BACK }

- enum **Function** {
  NEVER,
  LESS,
  EQUAL,
  LEQUAL,
  GREATER,
  NOTEQUAL,
  GEQUAL,
  ALWAYS }
- enum **Operation** {
  KEEP,
  ZERO,
  REPLACE,
  INCR,
  DECR,
  INVERT,
  INCR_WRAP,
  DECR_WRAP }

Public Member Functions

- StencilTwoSided (const StencilTwoSided &stencil, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, StencilTwoSided, STENCIL)
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- void **setFunction** (Face face, Function func, int ref, unsigned int mask)
- void **setFunction** (Face face, Function func)
- Function **getFunction** (Face face) const
- void **setFunctionRef** (Face face, int ref)
- int **getFunctionRef** (Face face) const
- void **setFunctionMask** (Face face, unsigned int mask)
- unsigned int **getFunctionMask** (Face face) const
- void setOperation (Face face, Operation sfail, Operation zfail, Operation zpass)
- void setStencilFailOperation (Face face, Operation sfail)
- Operation getStencilFailOperation (Face face) const
- void setStencilPassAndDepthFailOperation (Face face, Operation zfail)
- Operation getStencilPassAndDepthFailOperation (Face face) const
- void setStencilPassAndDepthPassOperation (Face face, Operation zpass)
- Operation getStencilPassAndDepthPassOperation (Face face) const
- void **setWriteMask** (Face face, unsigned int mask)
- unsigned int **getWriteMask** (Face face) const
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.337 Detailed Description

Encapsulate OpenGL two sided glStencilFunc/Op/Mask functions.

4.338 Constructor & Destructor Documentation

osg::StencilTwoSided::StencilTwoSided (const StencilTwoSided & *stencil*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.339 Member Function Documentation

virtual int osg::StencilTwoSided::compare (const StateAttribute & *sa*) const `[virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::StencilTwoSided::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

void osg::StencilTwoSided::setOperation (Face *face*, Operation *sfail*, Operation *zfail*, Operation *zpass*) `[inline]`

set the operations to apply when the various stencil and depth tests fail or pass. First parameter is to control the operation when the stencil test fails. The second parameter is to control the operation when the stencil

test passes, but depth test fails. The third parameter controls the operation when both the stencil test and depth pass. Ordering of parameter is the same as if using glStencilOp(,,).

void osg::StencilTwoSided::setStencilFailOperation (Face *face*, Operation *sfail*) `[inline]`

set the operation when the stencil test fails.

Operation osg::StencilTwoSided::getStencilFailOperation (Face *face*) const `[inline]`

get the operation when the stencil test fails.

void osg::StencilTwoSided::setStencilPassAndDepthFailOperation (Face *face*, Operation *zfail*) `[inline]`

set the operation when the stencil test passes but the depth test fails.

Operation osg::StencilTwoSided::getStencilPassAndDepthFailOperation (Face *face*) const `[inline]`

get the operation when the stencil test passes but the depth test fails.

void osg::StencilTwoSided::setStencilPassAndDepthPassOperation (Face *face*, Operation *zpass*) `[inline]`

set the operation when both the stencil test and the depth test pass.

Operation osg::StencilTwoSided::getStencilPassAndDepthPassOperation (Face *face*) const `[inline]`

get the operation when both the stencil test and the depth test pass.

virtual void osg::StencilTwoSided::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

static Extensions* osg::StencilTwoSided::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Exentsion object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID.

static void osg::StencilTwoSided::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

The setExtensions method allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

4.340 osg::SwapBuffersOperation Struct Reference

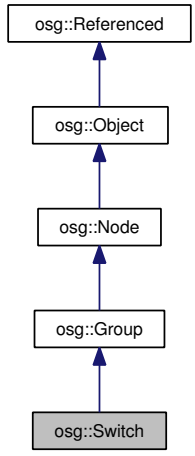
Public Member Functions

- virtual void **operator()** (GraphicsContext *context)

4.341 Detailed Description

SwapBufferOperation calls swap buffers on the GraphicsContext.

4.342 osg::Switch Class Reference



Public Types

- typedef std::vector< bool > **ValueList**

Public Member Functions

- Switch (const Switch &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, Switch)
- virtual void traverse (NodeVisitor &nv)
- void **setNewChildDefaultValue** (bool value)
- bool **getNewChildDefaultValue** () const
- virtual bool addChild (Node *child)
- virtual bool **addChild** (Node *child, bool value)
- virtual bool insertChild (unsigned int index, Node *child)
- virtual bool **insertChild** (unsigned int index, Node *child, bool value)
- virtual bool removeChildren (unsigned int pos, unsigned int numChildrenToRemove)
- void **setValue** (unsigned int pos, bool value)
- bool **getValue** (unsigned int pos) const
- void **setChildValue** (const Node *child, bool value)
- bool **getChildValue** (const Node *child) const
- bool setAllChildrenOff ()
- bool setAllChildrenOn ()
- bool setSingleChildOn (unsigned int pos)
- void **setValueList** (const ValueList &values)
- const ValueList & **getValueList** () const
- virtual BoundingSphere computeBound () const

4.343 Detailed Description

Switch is a Group node that allows switching between children. Typical uses would be for objects which might need to be rendered differently at different times, for instance a switch could be used to represent the different states of a traffic light.

4.344 Constructor & Destructor Documentation

osg::Switch::Switch (const Switch &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.345 Member Function Documentation

virtual void osg::Switch::traverse (NodeVisitor &) `[virtual]`

Traverse downwards : calls children's accept method with NodeVisitor.

Reimplemented from osg::Group.

virtual bool osg::Switch::addChild (Node * *child*) `[virtual]`

Add Node to Group. If node is not NULL and is not contained in Group then increment its reference count, add it to the child list and dirty the bounding sphere to force it to recompute on next getBound() and return true for success. Otherwise return false. Scene nodes can't be added as child nodes.

Reimplemented from osg::Group.

virtual bool osg::Switch::insertChild (unsigned int *index*, Node * *child*) `[virtual]`

Insert Node to Group at specific location. The new child node is inserted into the child list before the node at the specified index. No nodes are removed from the group with this operation.

Reimplemented from osg::Group.

virtual bool osg::Switch::removeChildren (unsigned int *pos*, unsigned int *numChildrenToRemove*) `[virtual]`

Remove children from Group. Note, must be override by subclasses of Group which add per child attributes.

Reimplemented from osg::Group.

bool osg::Switch::setAllChildrenOff ()

Set all the children off (false), and set the new default child value to off (false).

bool osg::Switch::setAllChildrenOn ()

Set all the children on (true), and set the new default child value to on (true).

bool osg::Switch::setSingleChildOn (unsigned int *pos*)

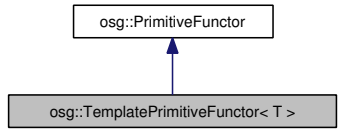
Set a single child on, switch off all other children.

virtual BoundingSphere osg::Switch::computeBound () const `[virtual]`

Compute the bounding sphere around Node's geometry or children. This method is automatically called by getBound() when the bounding sphere has been marked dirty via dirtyBound().

Reimplemented from osg::Group.

4.346 osg::TemplatePrimitiveFunctor< T > Class Template Reference



Public Member Functions

- void **setTreatVertexDataAsTemporary** (bool treatVertexDataAsTemporary)
- bool **getTreatVertexDataAsTemporary** () const
- virtual void setVertexArray (unsigned int, const Vec2 *)
- virtual void setVertexArray (unsigned int count, const Vec3 *vertices)
- virtual void setVertexArray (unsigned int, const Vec4 *)
- virtual void setVertexArray (unsigned int, const Vec2d *)
- virtual void setVertexArray (unsigned int count, const Vec3d *vertices)
- virtual void setVertexArray (unsigned int, const Vec4d *)
- virtual void drawArrays (GLenum mode, GLint first, GLsizei count)

  Mimics the OpenGL `glDrawArrays()` *function.*

- template<class IndexType>
  void **drawElementsTemplate** (GLenum mode, GLsizei count, const IndexType *indices)
- virtual void drawElements (GLenum mode, GLsizei count, const GLubyte *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLushort *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLuint *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void begin (GLenum mode)
- virtual void vertex (const Vec2 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec3 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec4 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y)
  Mimics the OpenGL `glVertex()` *"family of functions".*
- virtual void vertex (float x, float y, float z)
  Mimics the OpenGL `glVertex()` *"family of functions".*
- virtual void vertex (float x, float y, float z, float w)
  Mimics the OpenGL `glVertex()` *"family of functions".*
- virtual void end ()
  Mimics the OpenGL `glEnd()` *function.*

4.347 Detailed Description

template<class T> class osg::TemplatePrimitiveFunctor< T >

Provides access to the primitives that compose an `osg::Drawable`.

Notice that `TemplatePrimitiveFunctor` is a class template, and that it inherits from its template parameter `T`. This template parameter must implement `operator()(const osg::Vec3 v1, const osg::Vec3 v2, const osg::Vec3 v3, bool treatVertexDataAsTemporary)`, `operator()(const osg::Vec3 v1, const osg::Vec3 v2, bool treatVertexDataAsTemporary)`, `operator()(const osg::Vec3 v1, const osg::Vec3 v2, const osg::Vec3 v3, bool treatVertexDataAsTemporary)`, and `operator()(const osg::Vec3 v1, const osg::Vec3 v2, const osg::Vec3 v3, const osg::Vec3 v4, bool treatVertexDataAsTemporary)` which will be called for the matching primitive when the functor is applied to a `Drawable`. Parameters `v1`, `v2`, `v3`, and `v4` are the vertices of the primitive. The last parameter, `treatVertexDataAsTemporary`, indicates whether these vertices are coming from a "real" vertex array, or from a temporary vertex array, created by the `TemplatePrimitiveFunctor` from some other geometry representation.

See also:
`PrimitiveFunctor` for general usage hints.

4.348 Member Function Documentation

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned *count*, const Vec2 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned int *count*, const Vec3 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned *count*, const Vec4 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned *count*, const Vec2d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned int *count*, const Vec3d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::setVertexArray (unsigned *count*, const Vec4d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

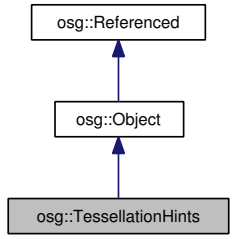
Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TemplatePrimitiveFunctor< T >::begin (GLenum *mode*) `[inline, virtual]`

Note: begin(..),vertex(..) & end() are convenience methods for adapting non vertex array primitives to vertex array based primitives. This is done to simplify the implementation of primtive functor subclasses - users only need override drawArray and drawElements.

Implements osg::PrimitiveFunctor.

4.349 osg::TessellationHints Class Reference



Public Types

- enum **TessellationMode** {
  USE_SHAPE_DEFAULTS,
  USE_TARGET_NUM_FACES }

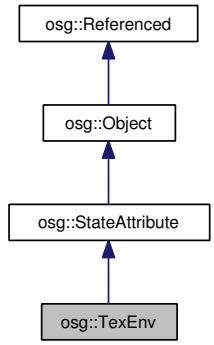
Public Member Functions

- **TessellationHints** (const TessellationHints &tess, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_Object** (osg, TessellationHints)
- void **setTessellationMode** (TessellationMode mode)
- TessellationMode **getTessellationMode** () const
- void **setDetailRatio** (float ratio)
- float **getDetailRatio** () const
- void **setTargetNumFaces** (unsigned int target)
- unsigned int **getTargetNumFaces** () const
- void **setCreateFrontFace** (bool on)
- bool **getCreateFrontFace** () const
- void **setCreateBackFace** (bool on)
- bool **getCreateBackFace** () const
- void **setCreateNormals** (bool on)
- bool **getCreateNormals** () const
- void **setCreateTextureCoords** (bool on)
- bool **getCreateTextureCoords** () const
- void **setCreateTop** (bool on)
- bool **getCreateTop** () const
- void **setCreateBody** (bool on)
- bool **getCreateBody** () const
- void **setCreateBottom** (bool on)
- bool **getCreateBottom** () const

4.350 Detailed Description

Describe several hints that can be passed to a Tessellator (like the one used by ShapeDrawable) as a mean to try to influence the way it works.

4.351 osg::TexEnv Class Reference



Public Types

- enum **Mode** {
  DECAL,
  MODULATE,
  BLEND,
  REPLACE,
  ADD }

Public Member Functions

- **TexEnv** (Mode mode=MODULATE)
- TexEnv (const TexEnv &texenv, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TexEnv, TEXENV)
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- void **setColor** (const Vec4 &color)
- Vec4 & **getColor** ()

- const Vec4 & **getColor** () const
- virtual void apply (State &state) const

4.352 Detailed Description

TexEnv encapsulates the OpenGL glTexEnv (texture environment) state.

4.353 Constructor & Destructor Documentation

osg::TexEnv::TexEnv (const TexEnv & *texenv*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.354 Member Function Documentation

virtual bool osg::TexEnv::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual int osg::TexEnv::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

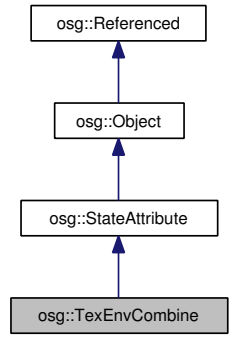
Implements osg::StateAttribute.

virtual void osg::TexEnv::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.355 osg::TexEnvCombine Class Reference



Public Types

- enum **CombineParam** {
  REPLACE,
  MODULATE,
  ADD,
  ADD_SIGNED,
  INTERPOLATE,
  SUBTRACT,
  DOT3_RGB,
  DOT3_RGBA }
- enum **SourceParam** {
  CONSTANT,
  PRIMARY_COLOR,
  PREVIOUS,
  TEXTURE,
  TEXTURE0,
  TEXTURE1,
  TEXTURE2,
  TEXTURE3,
  TEXTURE4,
  TEXTURE5,
  TEXTURE6,
  TEXTURE7 }

- enum **OperandParam** {
  SRC_COLOR,
  ONE_MINUS_SRC_COLOR,
  SRC_ALPHA,
  ONE_MINUS_SRC_ALPHA }

Public Member Functions

- TexEnvCombine (const TexEnvCombine &texenv, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TexEnvCombine, TEXENV)
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const
- void **setCombine_RGB** (GLint cm)
- void **setCombine_Alpha** (GLint cm)
- GLint **getCombine_RGB** () const
- GLint **getCombine_Alpha** () const
- void **setSource0_RGB** (GLint sp)
- void **setSource1_RGB** (GLint sp)
- void **setSource2_RGB** (GLint sp)
- void **setSource0_Alpha** (GLint sp)
- void **setSource1_Alpha** (GLint sp)
- void **setSource2_Alpha** (GLint sp)
- GLint **getSource0_RGB** () const
- GLint **getSource1_RGB** () const
- GLint **getSource2_RGB** () const
- GLint **getSource0_Alpha** () const
- GLint **getSource1_Alpha** () const
- GLint **getSource2_Alpha** () const
- void **setOperand0_RGB** (GLint op)
- void **setOperand1_RGB** (GLint op)
- void **setOperand2_RGB** (GLint op)
- void **setOperand0_Alpha** (GLint op)
- void **setOperand1_Alpha** (GLint op)
- void **setOperand2_Alpha** (GLint op)
- GLint **getOperand0_RGB** () const
- GLint **getOperand1_RGB** () const
- GLint **getOperand2_RGB** () const
- GLint **getOperand0_Alpha** () const
- GLint **getOperand1_Alpha** () const
- GLint **getOperand2_Alpha** () const
- void **setScale_RGB** (float scale)
- void **setScale_Alpha** (float scale)

- float **getScale_RGB** () const
- float **getScale_Alpha** () const
- void **setConstantColor** (const Vec4 &color)
- const Vec4 & **getConstantColor** () const
- void setConstantColorAsLightDirection (const Vec3 &direction)
- Vec3 **getConstantColorAsLightDirection** () const
- virtual void apply (State &state) const

4.356 Detailed Description

TexEnvCombine encapsulates the OpenGL glTexEnvCombine (texture environment) state.

4.357 Constructor & Destructor Documentation

osg::TexEnvCombine::TexEnvCombine (const TexEnvCombine & *texenv*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.358 Member Function Documentation

virtual bool osg::TexEnvCombine::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual int osg::TexEnvCombine::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::TexEnvCombine::setConstantColorAsLightDirection (const Vec3 & *direction*) `[inline]`

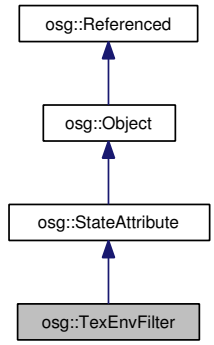
Set the constant color attribute to the given light direction for use with DOT3 combine operation.

virtual void osg::TexEnvCombine::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.359 osg::TexEnvFilter Class Reference



Public Member Functions

- **TexEnvFilter** (float lodBias=0.0f)
- TexEnvFilter (const TexEnvFilter &texenv, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TexEnvFilter, TEXENVFILTER)
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const
- void **setLodBias** (float lodBias)
- float **getLodBias** () const
- virtual void apply (State &state) const

4.360 Detailed Description

TexEnvFilter - encapsulates the OpenGL glTexEnv (GL_TEXTURE_FILTER_CONTROL) state.

4.361 Constructor & Destructor Documentation

osg::TexEnvFilter::TexEnvFilter (const TexEnvFilter & *texenv*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.362 Member Function Documentation

virtual bool osg::TexEnvFilter::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual int osg::TexEnvFilter::compare (const StateAttribute & *sa*) const `[inline, virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

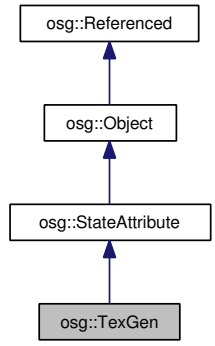
Implements osg::StateAttribute.

virtual void osg::TexEnvFilter::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

4.363 osg::TexGen Class Reference



Public Types

- enum **Mode** {
  OBJECT_LINEAR,

EYE_LINEAR,

SPHERE_MAP,

NORMAL_MAP,

REFLECTION_MAP }

- enum **Coord** {

S,

T,

R,

Q }

Public Member Functions

- TexGen (const TexGen &texgen, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TexGen, TEXGEN)
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- virtual void apply (State &state) const
- void **setMode** (Mode mode)
- Mode **getMode** () const
- void **setPlane** (Coord which, const Plane &plane)
- Plane & **getPlane** (Coord which)
- const Plane & **getPlane** (Coord which) const
- void setPlanesFromMatrix (const Matrixd &matrix)

4.364 Detailed Description

TexGen encapsulates the OpenGL glTexGen (texture coordinate generation) state.

4.365 Constructor & Destructor Documentation

osg::TexGen::TexGen (const TexGen & *texgen*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)** `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.366 Member Function Documentation

virtual bool osg::TexGen::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual int osg::TexGen::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::TexGen::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

virtual void osg::TexGen::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

void osg::TexGen::setPlanesFromMatrix (const Matrixd & *matrix*)

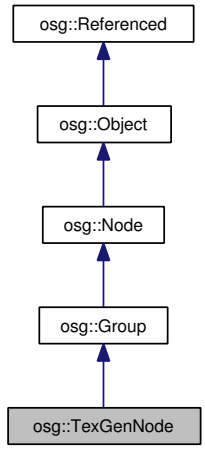
Set the tex gen planes from specified matrix. Typical usage would be to pass in a projection matrix to set up projective texturing.

4.367 Member Data Documentation

Plane osg::TexGen::_plane_s `[protected]`

Additional texgen coefficents for GL_OBJECT_PLANE or GL_EYE_PLANE,

4.368 osg::TexGenNode Class Reference



Public Types

- enum **ReferenceFrame** {
  RELATIVE_RF,
  ABSOLUTE_RF }

Public Member Functions

- **TexGenNode** (TexGen ∗texgen)
- **TexGenNode** (const TexGenNode &tgb, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Node** (osg, TexGenNode)
- void setReferenceFrame (ReferenceFrame rf)
- ReferenceFrame getReferenceFrame () const
- void setTextureUnit (unsigned int textureUnit)
- unsigned int **getTextureUnit** () const
- void setTexGen (TexGen ∗texgen)
- TexGen ∗ getTexGen ()
- const TexGen ∗ getTexGen () const
- virtual void setThreadSafeRefUnref (bool threadSafe)

4.369 Detailed Description

Node for defining the position of TexGen in the scene.

4.370 Member Function Documentation

void osg::TexGenNode::setReferenceFrame (ReferenceFrame *rf*)

Set the TexGenNode's ReferenceFrame, either to be relative to its parent reference frame.

ReferenceFrame osg::TexGenNode::getReferenceFrame () const `[inline]`

Ge thte TexGenNode's ReferenceFrame.

void osg::TexGenNode::setTextureUnit (unsigned int *textureUnit*) `[inline]`

Set the texture unit that this TexGenNode is associated with.

void osg::TexGenNode::setTexGen (TexGen * *texgen*)

Set the TexGen.

TexGen* osg::TexGenNode::getTexGen () `[inline]`

Get the TexGen.

const TexGen* osg::TexGenNode::getTexGen () const `[inline]`

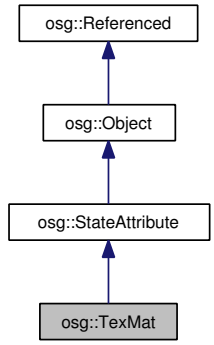
Get the const TexGen.

virtual void osg::TexGenNode::setThreadSafeRefUnref (bool *threadSafe*) `[virtual]`

Set whether to use a mutex to ensure ref() and unref() are thread safe.

Reimplemented from osg::Group.

4.371 osg::TexMat Class Reference



Public Member Functions

- **TexMat** (const Matrix &matrix)
- TexMat (const TexMat &texmat, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TexMat, TEXMAT)
- virtual bool isTextureAttribute () const
- virtual int compare (const StateAttribute &sa) const
- void setMatrix (const Matrix &matrix)
- Matrix & getMatrix ()
- const Matrix & getMatrix () const
- void setScaleByTextureRectangleSize (bool flag)
- bool getScaleByTextureRectangleSize () const
- virtual void apply (State &state) const

4.372 Detailed Description

A texture matrix state class that encapsulates OpenGL texture matrix functionality.

4.373 Constructor & Destructor Documentation

osg::TexMat::TexMat (const TexMat & *texmat*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.374 Member Function Documentation

virtual bool osg::TexMat::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual int osg::TexMat::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::TexMat::setMatrix (const Matrix & *matrix*) `[inline]`

Set the texture matrix

Matrix& osg::TexMat::getMatrix () `[inline]`

Get the texture matrix

const Matrix& osg::TexMat::getMatrix () const `[inline]`

Get the const texture matrix

void osg::TexMat::setScaleByTextureRectangleSize (bool *flag*) `[inline]`

Switch on/off the post scaling of the TexMat matrix by the size of the last applied texture rectangle. Use a TexMat alongside a TextureRectangle with this scaling applied allows one to treat a TextureRectnagles texture coordinate range as if it were the usual non dimensional 0.0 to 1.0 range. Note, the TexMat matrix itself is not modified by the post scaling, its purely an operation passed to OpenGL to do the post scaling once the the TexMat matrix has been loaded.

bool osg::TexMat::getScaleByTextureRectangleSize () const `[inline]`

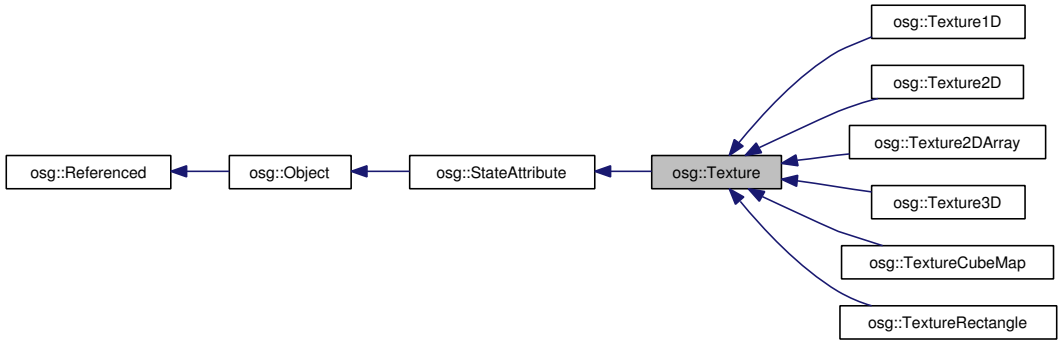
Get whether the post scaling of the TexMat matrix, by the size of the last applied texture rectangle, is switched on/off.

virtual void osg::TexMat::apply (State & *state*) const `[virtual]`

Apply texture matrix to OpenGL state.

Reimplemented from osg::StateAttribute.

4.375 osg::Texture Class Reference



Public Types

- enum **WrapParameter** {
  WRAP_S,
  WRAP_T,
  WRAP_R }
- enum **WrapMode** {
  CLAMP,
  CLAMP_TO_EDGE,
  CLAMP_TO_BORDER,
  REPEAT,
  MIRROR }
- enum **FilterParameter** {
  MIN_FILTER,
  MAG_FILTER }
- enum **FilterMode** {
  LINEAR,
  LINEAR_MIPMAP_LINEAR,
  LINEAR_MIPMAP_NEAREST,
  NEAREST,
  NEAREST_MIPMAP_LINEAR,
  NEAREST_MIPMAP_NEAREST }
- enum **InternalFormatMode** {
  USE_IMAGE_DATA_FORMAT,
  USE_USER_DEFINED_FORMAT,

USE_ARB_COMPRESSION,
USE_S3TC_DXT1_COMPRESSION,
USE_S3TC_DXT3_COMPRESSION,
USE_S3TC_DXT5_COMPRESSION }

- enum InternalFormatType {
  NORMALIZED,
  FLOAT,
  SIGNED_INTEGER,
  UNSIGNED_INTEGER }
- enum **ShadowCompareFunc** {
  LEQUAL,
  GEQUAL }
- enum **ShadowTextureMode** {
  LUMINANCE,
  INTENSITY,
  ALPHA }
- typedef std::list< ref_ptr< TextureObject > > **TextureObjectList**
- typedef osg::buffered_object< TextureObjectList > **TextureObjectListMap**

Public Member Functions

- Texture (const Texture &text, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- virtual osg::Object * cloneType () const =0
- virtual osg::Object * clone (const CopyOp &copyop) const =0
- virtual bool isSameKindAs (const osg::Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual Type getType () const
- virtual bool isTextureAttribute () const
- virtual GLenum **getTextureTarget** () const =0
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void setWrap (WrapParameter which, WrapMode wrap)
- WrapMode getWrap (WrapParameter which) const
- void setBorderColor (const Vec4d &color)
- const Vec4d & getBorderColor () const
- void setBorderWidth (GLint width)
- GLint **getBorderWidth** () const
- void setFilter (FilterParameter which, FilterMode filter)

- FilterMode getFilter (FilterParameter which) const
- void setMaxAnisotropy (float anis)
- float getMaxAnisotropy () const
- void setUseHardwareMipMapGeneration (bool useHardwareMipMapGeneration)
- bool getUseHardwareMipMapGeneration () const
- void setUnRefImageDataAfterApply (bool flag)
- bool getUnRefImageDataAfterApply () const
- void setClientStorageHint (bool flag)
- bool getClientStorageHint () const
- void setResizeNonPowerOfTwoHint (bool flag)
- bool getResizeNonPowerOfTwoHint () const
- void setInternalFormatMode (InternalFormatMode mode)
- InternalFormatMode getInternalFormatMode () const
- void setInternalFormat (GLint internalFormat)
- GLint getInternalFormat () const
- bool isCompressedInternalFormat () const
- void setSourceFormat (GLenum sourceFormat)
- GLenum getSourceFormat () const
- void setSourceType (GLenum sourceType)
- GLenum getSourceType () const
- InternalFormatType getInternalFormatType () const
- TextureObject * getTextureObject (unsigned int contextID) const
- void dirtyTextureObject ()
- bool areAllTextureObjectsLoaded () const
- unsigned int & getTextureParameterDirty (unsigned int contextID) const
- void dirtyTextureParameters ()
- void allocateMipmapLevels ()
- void setShadowComparison (bool flag)
- void setShadowCompareFunc (ShadowCompareFunc func)
- ShadowCompareFunc **getShadowCompareFunc** () const
- void setShadowTextureMode (ShadowTextureMode mode)
- ShadowTextureMode **getShadowTextureMode** () const
- void setShadowAmbient (float shadow_ambient)
- float **getShadowAmbient** () const
- virtual void setImage (unsigned int face, Image *image)=0
- virtual Image * getImage (unsigned int face)=0
- virtual const Image * getImage (unsigned int face) const =0
- virtual unsigned int getNumImages () const =0
- void setReadPBuffer (GraphicsContext *context)
- GraphicsContext * getReadPBuffer ()
- const GraphicsContext * getReadPBuffer () const
- virtual void apply (State &state) const =0
- virtual void compileGLObjects (State &state) const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (State *state=0) const

- void applyTexImage2D_load (State &state, GLenum target, const Image *image, GLsizei width, GLsizei height, GLsizei numMipmapLevels) const
- void applyTexImage2D_subload (State &state, GLenum target, const Image *image, GLsizei width, GLsizei height, GLint inInternalFormat, GLsizei numMipmapLevels) const
- void takeTextureObjects (TextureObjectListMap &toblm)

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)
- static bool isCompressedInternalFormat (GLint internalFormat)
- static void getCompressedSize (GLenum internalFormat, GLint width, GLint height, GLint depth, GLint &blockSize, GLint &size)
- static TextureObject * **generateTextureObject** (unsigned int contextID, GLenum target)
- static TextureObject * **generateTextureObject** (unsigned int contextID, GLenum target, GLint numMipmapLevels, GLenum internalFormat, GLsizei width, GLsizei height, GLsizei depth, GLint border)
- static void setMinimumNumberOfTextureObjectsToRetainInCache (unsigned int minimum)
- static unsigned int getMinimumNumberOfTextureObjectsToRetainInCache ()
- static void **flushAllDeletedTextureObjects** (unsigned int contextID)
- static void **flushDeletedTextureObjects** (unsigned int contextID, double currentTime, double &availableTime)

Static Public Attributes

- static unsigned int **s_numberTextureReusedLastInLastFrame**
- static unsigned int **s_numberNewTextureInLastFrame**
- static unsigned int **s_numberDeletedTextureInLastFrame**

Classes

- class Extensions
- class **TextureObject**

4.376 Detailed Description

Texture pure virtual base class that encapsulates OpenGl texture functionality common to the various types of OSG textures.

4.377 Member Enumeration Documentation

enum osg::Texture::InternalFormatType

Texture type determined by the internal texture format

Enumerator:

NORMALIZED default OpenGL format (clamped values to [0,1) or [0,255])

FLOAT float values, Shader Model 3.0 (see ARB_texture_float)

SIGNED_INTEGER Signed integer values (see EXT_texture_integer).

UNSIGNED_INTEGER Unsigned integer value (see EXT_texture_integer).

4.378 Constructor & Destructor Documentation

osg::Texture::Texture (const Texture & *text*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.379 Member Function Documentation

virtual osg::Object* osg::Texture::cloneType () const `[pure virtual]`

Clone the type of an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual osg::Object* osg::Texture::clone (const CopyOp &) const `[pure virtual]`

Clone an attribute, with Object* return type. Must be defined by derived classes.

Implements osg::StateAttribute.

virtual bool osg::Texture::isSameKindAs (const osg::Object * *obj*) const `[inline, virtual]`

Return true if this and obj are of the same kind of object.

Reimplemented from osg::StateAttribute.

virtual const char* osg::Texture::libraryName () const `[inline, virtual]`

Return the name of the attribute's library.

Reimplemented from osg::StateAttribute.

virtual const char* osg::Texture::className () const `[inline, virtual]`

Return the name of the attribute's class type.

Reimplemented from osg::StateAttribute.

virtual Type osg::Texture::getType () const `[inline, virtual]`

Return the Type identifier of the attribute's class type.

Implements osg::StateAttribute.

virtual bool osg::Texture::isTextureAttribute () const `[inline, virtual]`

Return true if StateAttribute is a type which controls texturing and needs to be issued w.r.t to specific texture unit.

Reimplemented from osg::StateAttribute.

virtual bool osg::Texture::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

void osg::Texture::setWrap (WrapParameter *which*, WrapMode *wrap*)

Sets the texture wrap mode.

WrapMode osg::Texture::getWrap (WrapParameter *which*) const

Gets the texture wrap mode.

void osg::Texture::setBorderColor (const Vec4d & *color*) `[inline]`

Sets the border color. Only used when wrap mode is CLAMP_TO_BORDER. The border color will be casted to the appropriate type to match the internal pixel format of the texture.

const Vec4d& osg::Texture::getBorderColor () const `[inline]`

Gets the border color.

void osg::Texture::setBorderWidth (GLint *width*) `[inline]`

Sets the border width.

void osg::Texture::setFilter (FilterParameter *which*, FilterMode *filter*)

Sets the texture filter mode.

FilterMode osg::Texture::getFilter (FilterParameter *which*) const

Gets the texture filter mode.

void osg::Texture::setMaxAnisotropy (float *anis*)

Sets the maximum anisotropy value, default value is 1.0 for no anisotropic filtering. If hardware does not support anisotropic filtering, use normal filtering (equivilant to a max anisotropy value of 1.0. Valid range is 1.0f upwards. The maximum value depends on the graphics system.

float osg::Texture::getMaxAnisotropy () const `[inline]`

Gets the maximum anisotropy value.

void osg::Texture::setUseHardwareMipMapGeneration (bool *useHardwareMipMapGeneration*) `[inline]`

Sets the hardware mipmap generation hint. If enabled, it will only be used if supported in the graphics system.

bool osg::Texture::getUseHardwareMipMapGeneration () const `[inline]`

Gets the hardware mipmap generation hint.

void osg::Texture::setUnRefImageDataAfterApply (bool *flag*) `[inline]`

Sets whether or not the apply() function will unreference the image data. If enabled, and the image data is only referened by this Texture, apply() will delete the image data.

bool osg::Texture::getUnRefImageDataAfterApply () const `[inline]`

Gets whether or not apply() unreferences image data.

void osg::Texture::setClientStorageHint (bool *flag*) `[inline]`

Sets whether to use client storage for the texture, if supported by the graphics system. Note: If enabled, and the graphics system supports it, the osg::Image(s) associated with this texture cannot be deleted, so the UnRefImageDataAfterApply flag would be ignored.

bool osg::Texture::getClientStorageHint () const `[inline]`

Gets whether to use client storage for the texture.

void osg::Texture::setResizeNonPowerOfTwoHint (bool *flag*) `[inline]`

Sets whether to force the texture to resize images that have dimensions that are not a power of two. If enabled, NPOT images will be resized, whether or not NPOT textures are supported by the hardware. If disabled, NPOT images will not be resized if supported by hardware.

bool osg::Texture::getResizeNonPowerOfTwoHint () const `[inline]`

Gets whether texture will force non power to two images to be resized.

void osg::Texture::setInternalFormatMode (InternalFormatMode *mode*) `[inline]`

Sets the internal texture format mode. Note: If the texture format is USE_IMAGE_DATA_FORMAT, USE_-ARB_COMPRESSION, or USE_S3TC_COMPRESSION, the internal format mode is set automatically and will overwrite the previous _internalFormat.

InternalFormatMode osg::Texture::getInternalFormatMode () const `[inline]`

Gets the internal texture format mode.

void osg::Texture::setInternalFormat (GLint *internalFormat*) `[inline]`

Sets the internal texture format. Implicitly sets the internalFormatMode to USE_USER_DEFINED_-FORMAT. The corresponding internal format type will be computed.

GLint osg::Texture::getInternalFormat () const `[inline]`

Gets the internal texture format.

bool osg::Texture::isCompressedInternalFormat () const

Return true if the internal format is one of the compressed formats.

void osg::Texture::setSourceFormat (GLenum *sourceFormat*) `[inline]`

Sets the external source image format, used as a fallback when no osg::Image is attached to provide the source image format.

GLenum osg::Texture::getSourceFormat () const `[inline]`

Gets the external source image format.

void osg::Texture::setSourceType (GLenum *sourceType*) `[inline]`

Sets the external source data type, used as a fallback when no osg::Image is attached to provide the source image format.

GLenum osg::Texture::getSourceType () const `[inline]`

Gets the external source data type.

InternalFormatType osg::Texture::getInternalFormatType () const `[inline]`

Get the internal texture format type.

TextureObject* osg::Texture::getTextureObject (unsigned int *contextID*) const `[inline]`

Returns a pointer to the texture object for the current context.

void osg::Texture::dirtyTextureObject ()

Forces a recompile on next apply() of associated OpenGL texture objects.

bool osg::Texture::areAllTextureObjectsLoaded () const

Returns true if the texture objects for all the required graphics contexts are loaded.

unsigned int& osg::Texture::getTextureParameterDirty (unsigned int *contextID*) const `[inline]`

Gets the dirty flag for the current contextID.

void osg::Texture::dirtyTextureParameters ()

Force a reset on next apply() of associated OpenGL texture parameters.

void osg::Texture::allocateMipmapLevels ()

Force a manual allocation of the mipmap levels on the next apply() call. User is responsible for filling the mipmap levels with valid data. The OpenGL's glGenerateMipmapEXT function is used to generate the mipmap levels. If glGenerateMipmapEXT is not supported or texture's internal format is not supported by the glGenerateMipmapEXT, then empty mipmap levels will be allocated manualy. The mipmap levels are also allocated if a non-mipmapped min filter is used.

void osg::Texture::setShadowComparison (bool *flag*) `[inline]`

Sets GL_TEXTURE_COMPARE_MODE_ARB to GL_COMPARE_R_TO_TEXTURE_ARB See http://oss.sgi.com/projects/ogl-sample/registry/ARB/shadow.txt.

void osg::Texture::setShadowCompareFunc (ShadowCompareFunc *func*) `[inline]`

Sets shadow texture comparison function.

void osg::Texture::setShadowTextureMode (ShadowTextureMode *mode*) `[inline]`

Sets shadow texture mode after comparison.

void osg::Texture::setShadowAmbient (float *shadow_ambient*) `[inline]`

Sets the TEXTURE_COMPARE_FAIL_VALUE_ARB texture parameter. See http://oss.sgi.com/projects/ogl-sample/registry/ARB/shadow_ambient.txt.

virtual void osg::Texture::setImage (unsigned int *face*, Image * *image*) `[pure virtual]`

Sets the texture image for the specified face.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

virtual Image* osg::Texture::getImage (unsigned int *face*) `[pure virtual]`

Gets the texture image for the specified face.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

virtual const Image* osg::Texture::getImage (unsigned int *face*) const `[pure virtual]`

Gets the const texture image for specified face.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

virtual unsigned int osg::Texture::getNumImages () const `[pure virtual]`

Gets the number of images that can be assigned to this Texture.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

void osg::Texture::setReadPBuffer (GraphicsContext * *context*) `[inline]`

Set the PBuffer graphis context to read from when using PBuffers for RenderToTexture.

GraphicsContext* osg::Texture::getReadPBuffer () `[inline]`

Get the PBuffer graphis context to read from when using PBuffers for RenderToTexture.

const GraphicsContext* osg::Texture::getReadPBuffer () const `[inline]`

Get the const PBuffer graphis context to read from when using PBuffers for RenderToTexture.

virtual void osg::Texture::apply (State & *state*) const `[pure virtual]`

Texture is a pure virtual base class, apply must be overriden.

Reimplemented from osg::StateAttribute.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

virtual void osg::Texture::compileGLObjects (State & *state*) const `[virtual]`

Calls apply(state) to compile the texture.

Reimplemented from osg::StateAttribute.

virtual void osg::Texture::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::StateAttribute.

virtual void osg::Texture::releaseGLObjects (State * *state* = 0) const `[virtual]`

If State is non-zero, this function releases OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

Reimplemented from osg::StateAttribute.

static Extensions* osg::Texture::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Gets the extension for the specified context. Creates the Extensions object for that context if it doesn't exist. Returns NULL if the Extensions object for the context doesn't exist and the createIfNotInitalized flag is false.

Reimplemented in osg::Texture2DArray, osg::Texture3D, and osg::TextureCubeMap.

static void osg::Texture::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

Overrides Extensions objects across graphics contexts. Typically used to ensure the same lowest common denominator of extensions on sustems with different graphics pipes.

static bool osg::Texture::isCompressedInternalFormat (GLint *internalFormat*) `[static]`

Determine whether the given internalFormat is a compressed image format.

static void osg::Texture::getCompressedSize (GLenum *internalFormat*, GLint *width*, GLint *height*, GLint *depth*, GLint & *blockSize*, GLint & *size*) `[static]`

Determine the size of a compressed image, given the internalFormat, the width, the height, and the depth of the image. The block size and the size are output parameters.

void osg::Texture::applyTexImage2D_load (State & *state*, GLenum *target*, const Image * *image*, GLsizei *width*, GLsizei *height*, GLsizei *numMipmapLevels*) const

Helper method. Creates the texture, but doesn't set or use a texture binding. Note: Don't call this method directly unless you're implementing a subload callback.

void osg::Texture::applyTexImage2D_subload (State & *state*, GLenum *target*, const Image * *image*, GLsizei *width*, GLsizei *height*, GLint *inInternalFormat*, GLsizei *numMipmapLevels*) const

Helper method. Subloads images into the texture, but doesn't set or use a texture binding. Note: Don't call this method directly unless you're implementing a subload callback.

void osg::Texture::applyTexParameters (GLenum *target*, State & *state*) const `[protected]`

Helper method. Sets texture paramters.

Reimplemented in osg::TextureRectangle.

void osg::Texture::generateMipmap (State & *state*) const `[protected]`

Helper method to generate empty mipmap levels by calling of glGenerateMipmapEXT. If it is not supported, then call the virtual allocateMipmap() method

virtual void osg::Texture::allocateMipmap (State & *state*) const `[protected, pure virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage* function.

Implemented in osg::Texture1D, osg::Texture2D, osg::Texture2DArray, osg::Texture3D, osg::TextureCubeMap, and osg::TextureRectangle.

int osg::Texture::compareTexture (const Texture & *rhs*) const `[protected]`

Returns -1 if ∗this < ∗rhs, 0 if ∗this==∗rhs, 1 if ∗this>∗rhs.

int osg::Texture::compareTextureObjects (const Texture & *rhs*) const `[protected]`

Returns -1 if ∗this < ∗rhs, 0 if ∗this==∗rhs, 1 if ∗this>∗rhs.

void osg::Texture::takeTextureObjects (TextureObjectListMap & *toblm*)

Takes the active texture objects from the Texture and places them in the specified TextureObjectListMap.

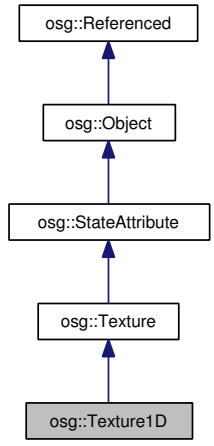
static void osg::Texture::setMinimumNumberOfTextureObjectsToRetainInCache (unsigned int *minimum*) `[static]`

Set the minimum number of texture objects to retain in the deleted display list cache.

static unsigned int osg::Texture::getMinimumNumberOfTextureObjectsToRetainInCache () `[static]`

Get the minimum number of display lists to retain in the deleted display list cache.

4.380 osg::Texture1D Class Reference



Public Member Functions

- Texture1D (const Texture1D &text, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_StateAttribute** (osg, Texture1D, TEXTURE)
- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- void setImage (Image *image)
- Image * getImage ()
- const Image * getImage () const
- unsigned int & **getModifiedCount** (unsigned int contextID) const
- virtual void setImage (unsigned int, Image *image)
- virtual Image * getImage (unsigned int)
- virtual const Image * getImage (unsigned int) const
- virtual unsigned int getNumImages () const
- void setTextureWidth (int width) const
- virtual int getTextureWidth () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void setNumMipmapLevels (unsigned int num) const
- unsigned int getNumMipmapLevels () const
- void copyTexImage1D (State &state, int x, int y, int width)
- void copyTexSubImage1D (State &state, int xoffset, int x, int y, int width)
- virtual void apply (State &state) const

Classes

- class **SubloadCallback**

4.381 Detailed Description

Encapsulates OpenGl 1D texture functionality. Doesn't support cube maps, so ignore *face* parameters.

4.382 Constructor & Destructor Documentation

osg::Texture1D::Texture1D (const Texture1D & *text*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.383 Member Function Documentation

virtual int osg::Texture1D::compare (const StateAttribute & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::Texture1D::setImage (Image * *image*)

Sets the texture image.

Image* osg::Texture1D::getImage () `[inline]`

Gets the texture image.

const Image* osg::Texture1D::getImage () const `[inline]`

Gets the const texture image.

virtual void osg::Texture1D::setImage (unsigned int, Image * *image*) `[inline, virtual]`

Sets the texture image, ignoring face.

Implements osg::Texture.

virtual Image* osg::Texture1D::getImage (unsigned int) `[inline, virtual]`

Gets the texture image, ignoring face.

Implements osg::Texture.

virtual const Image* osg::Texture1D::getImage (unsigned int) const `[inline, virtual]`

Gets the const texture image, ignoring face.

Implements osg::Texture.

virtual unsigned int osg::Texture1D::getNumImages () const `[inline, virtual]`

Gets the number of images that can be assigned to the Texture.

Implements osg::Texture.

void osg::Texture1D::setTextureWidth (int *width*) const `[inline]`

Sets the texture width. If width is zero, calculate the value from the source image width.

virtual int osg::Texture1D::getTextureWidth () const `[inline, virtual]`

Gets the texture width.

Reimplemented from osg::Texture.

void osg::Texture1D::setNumMipmapLevels (unsigned int *num*) const `[inline]`

Helper function. Sets the number of mipmap levels created for this texture. Should only be called within an osg::Texuture::apply(), or during a custom OpenGL texture load.

unsigned int osg::Texture1D::getNumMipmapLevels () const `[inline]`

Gets the number of mipmap levels created.

void osg::Texture1D::copyTexImage1D (State & *state*, int *x*, int *y*, int *width*)

Copies pixels into a 1D texture image, as per glCopyTexImage1D. Creates an OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width*. *width* must be a power of two.

void osg::Texture1D::copyTexSubImage1D (State & *state*, int *xoffset*, int *x*, int *y*, int *width*)

Copies a one-dimensional texture subimage, as per glCopyTexSubImage1D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width*.

virtual void osg::Texture1D::apply (State & *state*) const `[virtual]`

Bind the texture object. If the texture object hasn't already been compiled, create the texture mipmap levels.

Implements osg::Texture.

void osg::Texture1D::allocateMipmap (State & *state*) const `[protected, virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage∗ function.

Implements osg::Texture.

void osg::Texture1D::applyTexImage1D (GLenum *target*, Image ∗ *image*, State & *state*, GLsizei & *width*, GLsizei & *numMipmapLevels*) const `[protected]`

Helper method. Createa the texture without setting or using a texture binding.

4.384 Member Data Documentation

ref_ptr<Image> osg::Texture1D::_image `[mutable, protected]`

It's not ideal that _image is mutable, but it's required since Image::ensureDimensionsArePowerOfTwo() can only be called in a valid OpenGL context, and therefore within Texture::apply, which is const.

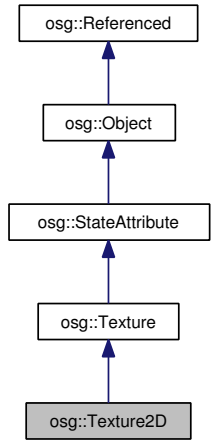
GLsizei osg::Texture1D::_textureWidth `[mutable, protected]`

Subloaded images can have different texture and image sizes.

GLsizei osg::Texture1D::_numMipmapLevels `[mutable, protected]`

Number of mipmap levels created.

4.385 osg::Texture2D Class Reference



Public Member Functions

- **Texture2D** (Image *image)
- Texture2D (const Texture2D &text, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Texture2D, TEXTURE)
- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- void setImage (Image *image)
- Image * getImage ()
- const Image * getImage () const

- unsigned int & **getModifiedCount** (unsigned int contextID) const
- virtual void setImage (unsigned int, Image *image)
- virtual Image * getImage (unsigned int)
- virtual const Image * getImage (unsigned int) const
- virtual unsigned int getNumImages () const
- void setTextureSize (int width, int height) const
- void **setTextureWidth** (int width)
- void **setTextureHeight** (int height)
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void setNumMipmapLevels (unsigned int num) const
- unsigned int getNumMipmapLevels () const
- void copyTexImage2D (State &state, int x, int y, int width, int height)
- void copyTexSubImage2D (State &state, int xoffset, int yoffset, int x, int y, int width, int height)
- virtual void apply (State &state) const

Classes

- class **SubloadCallback**

4.386 Detailed Description

Encapsulates OpenGl 2D texture functionality. Doesn't support cube maps, so ignore *face* parameters.

4.387 Constructor & Destructor Documentation

osg::Texture2D::Texture2D (const Texture2D & *text*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.388 Member Function Documentation

virtual int osg::Texture2D::compare (const StateAttribute & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::Texture2D::setImage (Image * *image*)

Sets the texture image.

Image* osg::Texture2D::getImage () `[inline]`

Gets the texture image.

const Image* osg::Texture2D::getImage () const `[inline]`

Gets the const texture image.

virtual void osg::Texture2D::setImage (unsigned int, Image * *image*) `[inline, virtual]`

Sets the texture image, ignoring face.

Implements osg::Texture.

virtual Image* osg::Texture2D::getImage (unsigned int) `[inline, virtual]`

Gets the texture image, ignoring face.

Implements osg::Texture.

virtual const Image* osg::Texture2D::getImage (unsigned int) const `[inline, virtual]`

Gets the const texture image, ignoring face.

Implements osg::Texture.

virtual unsigned int osg::Texture2D::getNumImages () const `[inline, virtual]`

Gets the number of images that can be assigned to the Texture.

Implements osg::Texture.

void osg::Texture2D::setTextureSize (int *width*, int *height*) const `[inline]`

Sets the texture width and height. If width or height are zero, calculate the respective value from the source image size.

void osg::Texture2D::setNumMipmapLevels (unsigned int *num*) const `[inline]`

Helper function. Sets the number of mipmap levels created for this texture. Should only be called within an osg::Texuture::apply(), or during a custom OpenGL texture load.

unsigned int osg::Texture2D::getNumMipmapLevels () const `[inline]`

Gets the number of mipmap levels created.

void osg::Texture2D::copyTexImage2D (State & *state*, int *x*, int *y*, int *width*, int *height*)

Copies pixels into a 2D texture image, as per glCopyTexImage2D. Creates an OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. *width* and *height* must be a power of two.

void osg::Texture2D::copyTexSubImage2D (State & *state*, int *xoffset*, int *yoffset*, int *x*, int *y*, int *width*, int *height*)

Copies a two-dimensional texture subimage, as per glCopyTexSubImage2D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. Loads framebuffer data into the texture using offsets *xoffset* and *yoffset*. *width* and *height* must be powers of two.

virtual void osg::Texture2D::apply (State & *state*) const `[virtual]`

Bind the texture object. If the texture object hasn't already been compiled, create the texture mipmap levels.

Implements osg::Texture.

void osg::Texture2D::allocateMipmap (State & *state*) const `[protected, virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage∗ function.

Implements osg::Texture.

4.389 Member Data Documentation

ref_ptr<Image> osg::Texture2D::_image `[protected]`

It's not ideal that _image is mutable, but it's required since Image::ensureDimensionsArePowerOfTwo() can only be called in a valid OpenGL context, and therefore within Texture::apply, which is const.

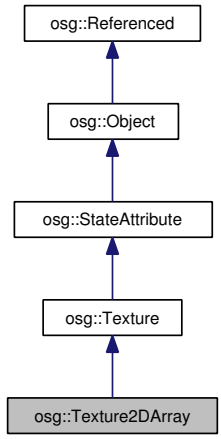
GLsizei osg::Texture2D::_textureWidth `[mutable, protected]`

Subloaded images can have different texture and image sizes.

GLsizei osg::Texture2D::_numMipmapLevels `[mutable, protected]`

Number of mipmap levels created.

4.390 osg::Texture2DArray Class Reference



Public Member Functions

- Texture2DArray (const Texture2DArray &cm, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, Texture2DArray, TEXTURE)
- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- virtual void setImage (unsigned int layer, Image *image)
- virtual Image * getImage (unsigned int layer)
- virtual const Image * getImage (unsigned int layer) const
- virtual unsigned int getNumImages () const
- unsigned int & getModifiedCount (unsigned int layer, unsigned int contextID) const
- void setTextureSize (int width, int height, int depth)
- void **setTextureWidth** (int width)
- void **setTextureHeight** (int height)
- void **setTextureDepth** (int depth)
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void setNumMipmapLevels (unsigned int num) const
- unsigned int getNumMipmapLevels () const
- void copyTexSubImage2DArray (State &state, int xoffset, int yoffset, int zoffset, int x, int y, int width, int height)
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions
- class **SubloadCallback**

4.391 Detailed Description

Texture2DArray state class which encapsulates OpenGL 2D array texture functionality. Texture arrays were introduced with Shader Model 4.0 hardware.

A 2D texture array does contain textures sharing the same properties (e.g. size, bitdepth,...) in a layered structure. See http://www.opengl.org/registry/specs/EXT/texture_array.txt for more info.

4.392 Constructor & Destructor Documentation

osg::Texture2DArray::Texture2DArray (const Texture2DArray & *cm*, const CopyOp & *copyop* = CopyOp::SHALLOW_COPY)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.393 Member Function Documentation

virtual int osg::Texture2DArray::compare (const StateAttribute & *rhs*) const [virtual]

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual void osg::Texture2DArray::setImage (unsigned int *layer*, Image * *image*) [virtual]

Set the texture image for specified layer.

Implements osg::Texture.

virtual Image* osg::Texture2DArray::getImage (unsigned int *layer*) `[virtual]`

Get the texture image for specified layer.

Implements osg::Texture.

virtual const Image* osg::Texture2DArray::getImage (unsigned int *layer*) const `[virtual]`

Get the const texture image for specified layer.

Implements osg::Texture.

virtual unsigned int osg::Texture2DArray::getNumImages () const `[inline, virtual]`

Get the number of images that are assigned to the Texture. The number is equal to the texture depth. To get the maximum possible image/layer count, you have to use the extension subclass, since it provides graphic context dependent information.

Implements osg::Texture.

unsigned int& osg::Texture2DArray::getModifiedCount (unsigned int *layer*, unsigned int *contextID*) const `[inline]`

Check how often was a certain layer in the given context modified

void osg::Texture2DArray::setTextureSize (int *width*, int *height*, int *depth*)

Set the texture width and height. If width or height are zero then the repsective size value is calculated from the source image sizes. Depth parameter specifies the number of layers to be used.

void osg::Texture2DArray::setNumMipmapLevels (unsigned int *num*) const `[inline]`

Set the number of mip map levels the the texture has been created with. Should only be called within an osg::Texuture::apply() and custom OpenGL texture load.

unsigned int osg::Texture2DArray::getNumMipmapLevels () const `[inline]`

Get the number of mip map levels the the texture has been created with.

void osg::Texture2DArray::copyTexSubImage2DArray (State & *state*, int *xoffset*, int *yoffset*, int *zoffset*, int *x*, int *y*, int *width*, int *height*)

Copies a two-dimensional texture subimage, as per glCopyTexSubImage3D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. Loads framebuffer data into the texture using offsets *xoffset* and *yoffset*. *zoffset* specifies the layer of the texture array to which the result is copied.

virtual void osg::Texture2DArray::apply (State & *state*) const `[virtual]`

Bind the texture if already compiled. Otherwise recompile.

Implements osg::Texture.

static Extensions* osg::Texture2DArray::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Exentsion object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID.

Reimplemented from osg::Texture.

static void osg::Texture2DArray::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

The setExtensions method allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

void osg::Texture2DArray::allocateMipmap (State & *state*) const `[protected, virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage* function.

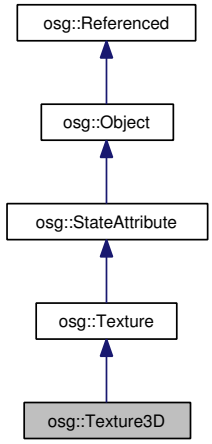
Implements osg::Texture.

4.394 Member Data Documentation

std::vector<ref_ptr<Image> > osg::Texture2DArray::_images `[protected]`

Use std::vector to encapsulate referenced pointers to images of different layers. Vectors gives us a random access iterator. The overhead of non-used elements is negligeable

4.395 osg::Texture3D Class Reference



Public Member Functions

- Texture3D (const Texture3D &text, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Texture3D, TEXTURE)
- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- void setImage (Image *image)
- Image * getImage ()
- const Image * getImage () const
- unsigned int & **getModifiedCount** (unsigned int contextID) const
- virtual void setImage (unsigned int, Image *image)
- virtual Image * getImage (unsigned int)
- virtual const Image * getImage (unsigned int) const
- virtual unsigned int getNumImages () const
- void setTextureSize (int width, int height, int depth) const
- void getTextureSize (int &width, int &height, int &depth) const
- void **setTextureWidth** (int width)
- void **setTextureHeight** (int height)
- void **setTextureDepth** (int depth)
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void setNumMipmapLevels (unsigned int num) const

- unsigned int getNumMipmapLevels () const
- void copyTexSubImage3D (State &state, int xoffset, int yoffset, int zoffset, int x, int y, int width, int height)
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions
- class **SubloadCallback**

4.396 Detailed Description

Encapsulates OpenGl 2D texture functionality. Doesn't support cube maps, so ignore *face* parameters.

4.397 Constructor & Destructor Documentation

osg::Texture3D::Texture3D (const Texture3D & *text*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.398 Member Function Documentation

virtual int osg::Texture3D::compare (const StateAttribute & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::Texture3D::setImage (Image * *image*)

Sets the texture image.

Image* osg::Texture3D::getImage () `[inline]`

Gets the texture image.

const Image* osg::Texture3D::getImage () const `[inline]`

Gets the const texture image.

virtual void osg::Texture3D::setImage (unsigned int, Image * *image*) `[inline, virtual]`

Sets the texture image, ignoring face.

Implements osg::Texture.

virtual Image* osg::Texture3D::getImage (unsigned int) `[inline, virtual]`

Gets the texture image, ignoring face.

Implements osg::Texture.

virtual const Image* osg::Texture3D::getImage (unsigned int) const `[inline, virtual]`

Gets the const texture image, ignoring face.

Implements osg::Texture.

virtual unsigned int osg::Texture3D::getNumImages () const `[inline, virtual]`

Gets the number of images that can be assigned to the Texture.

Implements osg::Texture.

void osg::Texture3D::setTextureSize (int *width*, int *height*, int *depth*) const `[inline]`

Sets the texture width, height, and depth. If width, height, or depth are zero, calculate the respective value from the source image size.

void osg::Texture3D::getTextureSize (int & *width*, int & *height*, int & *depth*) const `[inline]`

Gets the texture subload width.

void osg::Texture3D::setNumMipmapLevels (unsigned int *num*) const `[inline]`

Helper function. Sets the number of mipmap levels created for this texture. Should only be called within an osg::Texuture::apply(), or during a custom OpenGL texture load.

unsigned int osg::Texture3D::getNumMipmapLevels () const `[inline]`

Gets the number of mipmap levels created.

void osg::Texture3D::copyTexSubImage3D (State & *state*, int *xoffset*, int *yoffset*, int *zoffset*, int *x*, int *y*, int *width*, int *height*)

Copies a two-dimensional texture subimage, as per glCopyTexSubImage3D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. Loads framebuffer data into the texture using offsets *xoffset*, *yoffset*, and *zoffset*. *width* and *height* must be powers of two.

virtual void osg::Texture3D::apply (State & *state*) const `[virtual]`

Bind the texture object. If the texture object hasn't already been compiled, create the texture mipmap levels.

Implements osg::Texture.

static Extensions* osg::Texture3D::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Encapsulates queries of extension availability, obtains extension function pointers, and provides convinience wrappers for calling extension functions.

Reimplemented from osg::Texture.

static void osg::Texture3D::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

Overrides Extensions objects across graphics contexts. Typically used to ensure the same lowest common denominator of extensions on sustems with different graphics pipes.

void osg::Texture3D::allocateMipmap (State & *state*) const `[protected, virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage* function.

Implements osg::Texture.

4.399 Member Data Documentation

ref_ptr<Image> osg::Texture3D::_image `[mutable, protected]`

It's not ideal that _image is mutable, but it's required since Image::ensureDimensionsArePowerOfTwo() can only be called in a valid OpenGL context, and therefore within Texture::apply, which is const.

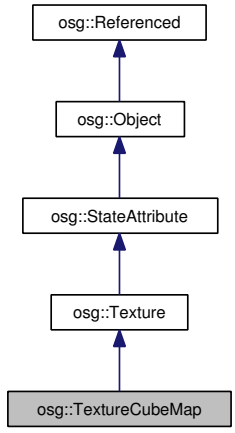
GLsizei osg::Texture3D::_textureWidth `[mutable, protected]`

Subloaded images can have different texture and image sizes.

GLsizei osg::Texture3D::_numMipmapLevels `[mutable, protected]`

Number of mip map levels the the texture has been created with,

4.400 osg::TextureCubeMap Class Reference



Public Types

- enum **Face** {
  POSITIVE_X,
  NEGATIVE_X,
  POSITIVE_Y,
  NEGATIVE_Y,
  POSITIVE_Z,
  NEGATIVE_Z }

Public Member Functions

- TextureCubeMap (const TextureCubeMap &cm, const CopyOp &copyop=CopyOp::SHALLOW_-COPY)
- **META_StateAttribute** (osg, TextureCubeMap, TEXTURE)

- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- virtual void setImage (unsigned int face, Image *image)
- virtual Image * getImage (unsigned int face)
- virtual const Image * getImage (unsigned int face) const
- virtual unsigned int getNumImages () const
- unsigned int & **getModifiedCount** (unsigned int face, unsigned int contextID) const
- void setTextureSize (int width, int height) const
- void **setTextureWidth** (int width)
- void **setTextureHeight** (int height)
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void setNumMipmapLevels (unsigned int num) const
- unsigned int getNumMipmapLevels () const
- void copyTexSubImageCubeMap (State &state, int face, int xoffset, int yoffset, int x, int y, int width, int height)
- virtual void apply (State &state) const

Static Public Member Functions

- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions
- class **SubloadCallback**

4.401 Detailed Description

TextureCubeMap state class which encapsulates OpenGL texture cubemap functionality.

4.402 Constructor & Destructor Documentation

osg::TextureCubeMap::TextureCubeMap (const TextureCubeMap & *cm*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.403 Member Function Documentation

virtual int osg::TextureCubeMap::compare (const StateAttribute & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual void osg::TextureCubeMap::setImage (unsigned int *face*, Image * *image*) `[virtual]`

Set the texture image for specified face.

Implements osg::Texture.

virtual Image* osg::TextureCubeMap::getImage (unsigned int *face*) `[virtual]`

Get the texture image for specified face.

Implements osg::Texture.

virtual const Image* osg::TextureCubeMap::getImage (unsigned int *face*) const `[virtual]`

Get the const texture image for specified face.

Implements osg::Texture.

virtual unsigned int osg::TextureCubeMap::getNumImages () const `[inline, virtual]`

Get the number of images that can be assigned to the Texture.

Implements osg::Texture.

void osg::TextureCubeMap::setTextureSize (int *width*, int *height*) const `[inline]`

Set the texture width and height. If width or height are zero then the repsective size value is calculated from the source image sizes.

void osg::TextureCubeMap::setNumMipmapLevels (unsigned int *num*) const `[inline]`

Set the number of mip map levels the the texture has been created with. Should only be called within an osg::Texuture::apply() and custom OpenGL texture load.

unsigned int osg::TextureCubeMap::getNumMipmapLevels () const [inline]

Get the number of mip map levels the the texture has been created with.

void osg::TextureCubeMap::copyTexSubImageCubeMap (State & *state*, int *face*, int *xoffset*, int *yoffset*, int *x*, int *y*, int *width*, int *height*)

Copies a two-dimensional texture subimage, as per glCopyTexSubImage2D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. Loads framebuffer data into the texture using offsets *xoffset* and *yoffset*. *width* and *height* must be powers of two.

virtual void osg::TextureCubeMap::apply (State & *state*) const [virtual]

On first apply (unless already compiled), create the mipmapped texture and bind it. Subsequent apply will simple bind to texture.

Implements osg::Texture.

static Extensions* osg::TextureCubeMap::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) [static]

Function to call to get the extension of a specified context. If the Exentsion object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID.

Reimplemented from osg::Texture.

static void osg::TextureCubeMap::setExtensions (unsigned int *contextID*, Extensions * *extensions*) [static]

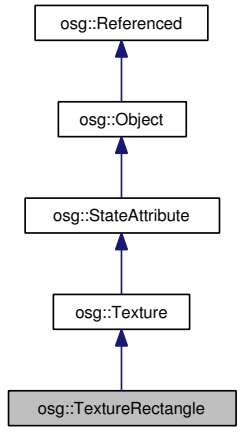
The setExtensions method allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

void osg::TextureCubeMap::allocateMipmap (State & *state*) const [protected, virtual]

Allocate mipmap levels of the texture by subsequent calling of glTexImage* function.

Implements osg::Texture.

4.404 osg::TextureRectangle Class Reference



Public Member Functions

- **TextureRectangle** (Image *image)
- TextureRectangle (const TextureRectangle &text, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, TextureRectangle, TEXTURE)
- virtual int compare (const StateAttribute &rhs) const
- virtual GLenum **getTextureTarget** () const
- void setImage (Image *image)
- Image * getImage ()
- const Image * getImage () const
- unsigned int & **getModifiedCount** (unsigned int contextID) const
- virtual void setImage (unsigned int, Image *image)
- virtual Image * getImage (unsigned int)
- virtual const Image * getImage (unsigned int) const
- virtual unsigned int getNumImages () const
- void setTextureSize (int width, int height) const
- void **setTextureWidth** (int width)
- void **setTextureHeight** (int height)
- virtual int **getTextureWidth** () const
- virtual int **getTextureHeight** () const
- virtual int **getTextureDepth** () const
- void **setSubloadCallback** (SubloadCallback *cb)
- SubloadCallback * **getSubloadCallback** ()
- const SubloadCallback * **getSubloadCallback** () const
- void copyTexImage2D (State &state, int x, int y, int width, int height)
- void copyTexSubImage2D (State &state, int xoffset, int yoffset, int x, int y, int width, int height)
- virtual void apply (State &state) const

Classes

- class **SubloadCallback**

4.405 Detailed Description

Texture state class which encapsulates OpenGL texture functionality.

4.406 Constructor & Destructor Documentation

osg::TextureRectangle::TextureRectangle (const TextureRectangle & *text*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.407 Member Function Documentation

virtual int osg::TextureRectangle::compare (const StateAttribute & *rhs*) const `[virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

void osg::TextureRectangle::setImage (Image * *image*)

Set the texture image.

Image* osg::TextureRectangle::getImage () `[inline]`

Get the texture image.

const Image* osg::TextureRectangle::getImage () const `[inline]`

Get the const texture image.

virtual void osg::TextureRectangle::setImage (unsigned int, Image * *image*) `[inline, virtual]`

Set the texture image, ignoring face value as there is only one image.

Implements osg::Texture.

virtual Image* osg::TextureRectangle::getImage (unsigned int) `[inline, virtual]`

Get the texture image, ignoring face value as there is only one image.

Implements osg::Texture.

virtual const Image* osg::TextureRectangle::getImage (unsigned int) const `[inline, virtual]`

Get the const texture image, ignoring face value as there is only one image.

Implements osg::Texture.

virtual unsigned int osg::TextureRectangle::getNumImages () const `[inline, virtual]`

Get the number of images that can be assigned to the Texture.

Implements osg::Texture.

void osg::TextureRectangle::setTextureSize (int *width*, int *height*) const `[inline]`

Set the texture width and height. If width or height are zero then the repsective size value is calculated from the source image sizes.

void osg::TextureRectangle::copyTexImage2D (State & *state*, int *x*, int *y*, int *width*, int *height*)

Copies pixels into a 2D texture image, as per glCopyTexImage2D. Creates an OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. *width* and *height* must be a power of two.

void osg::TextureRectangle::copyTexSubImage2D (State & *state*, int *xoffset*, int *yoffset*, int *x*, int *y*, int *width*, int *height*)

Copies a two-dimensional texture subimage, as per glCopyTexSubImage2D. Updates a portion of an existing OpenGL texture object from the current OpenGL background framebuffer contents at position *x*, *y* with width *width* and height *height*. Loads framebuffer data into the texture using offsets *xoffset* and *yoffset*. *width* and *height* must be powers of two.

virtual void osg::TextureRectangle::apply (State & *state*) const `[virtual]`

On first apply (unless already compiled), create and bind the texture, subsequent apply will simply bind to texture.

Implements osg::Texture.

void osg::TextureRectangle::allocateMipmap (State & *state*) const `[protected, virtual]`

Allocate mipmap levels of the texture by subsequent calling of glTexImage∗ function.

Implements osg::Texture.

void osg::TextureRectangle::applyTexParameters (GLenum *target*, State & *state*) const `[protected]`

Helper method. Sets texture paramters.

Reimplemented from osg::Texture.

4.408 osg::Timer Class Reference

Public Member Functions

- Timer_t tick () const
- void setStartTick ()
- void **setStartTick** (Timer_t t)
- Timer_t **getStartTick** () const
- double time_s () const
- double time_m () const
- double time_u () const
- double time_n () const
- double delta_s (Timer_t t1, Timer_t t2) const
- double delta_m (Timer_t t1, Timer_t t2) const
- double delta_u (Timer_t t1, Timer_t t2) const
- double delta_n (Timer_t t1, Timer_t t2) const
- double getSecondsPerTick () const

Static Public Member Functions

- static Timer ∗ **instance** ()

4.409 Detailed Description

Timer class is used for measuring elapsed time or time between two points.

4.410 Member Function Documentation

Timer_t osg::Timer::tick () const

Get the timers tick value.

void osg::Timer::setStartTick () `[inline]`

Set the start.

double osg::Timer::time_s () const `[inline]`

Get elapsed time in seconds.

double osg::Timer::time_m () const `[inline]`

Get elapsed time in milliseconds.

double osg::Timer::time_u () const `[inline]`

Get elapsed time in micoseconds.

double osg::Timer::time_n () const `[inline]`

Get elapsed time in nanoseconds.

double osg::Timer::delta_s (Timer_t *t1*, Timer_t *t2*) const `[inline]`

Get the time in seconds between timer ticks t1 and t2.

double osg::Timer::delta_m (Timer_t *t1*, Timer_t *t2*) const `[inline]`

Get the time in milliseconds between timer ticks t1 and t2.

double osg::Timer::delta_u (Timer_t *t1*, Timer_t *t2*) const `[inline]`

Get the time in microseconds between timer ticks t1 and t2.

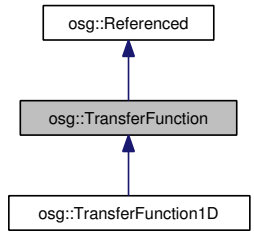
double osg::Timer::delta_n (Timer_t *t1*, Timer_t *t2*) const `[inline]`

Get the time in nanoseconds between timer ticks t1 and t2.

double osg::Timer::getSecondsPerTick () const `[inline]`

Get the the numer of ticks per second.

4.411 osg::TransferFunction Class Reference



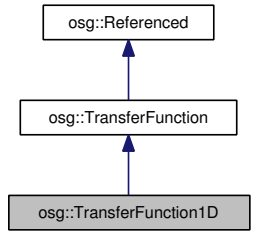
Public Member Functions

- osg::Image * **getImage** ()
- const osg::Image * **getImage** () const
- osg::Texture * **getTexture** ()
- const osg::Texture * **getTexture** () const
- osg::Shader * **getShader** ()
- const osg::Shader * **getShader** () const

4.412 Detailed Description

TransferFunction is a class that provide a 1D,2D or 3D colour look up table that can be used on the GPU as a 1D, 2D or 3D texture. Typically uses include mapping heights to colours when contouring terrain, or mapping intensities to colours when volume rendering.

4.413 osg::TransferFunction1D Class Reference



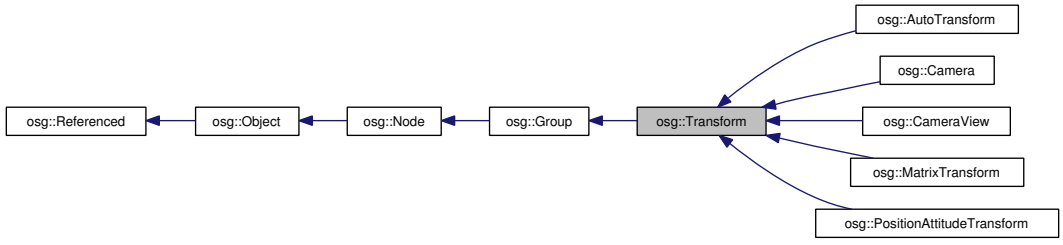
Public Member Functions

- void **setInputRange** (float minimum, float maximum)
- void **setMinimum** (float value)
- float **getMinimum** () const
- void **setMaximum** (float value)
- float **getMaximum** () const
- void **allocate** (unsigned int numX)
- void **clear** (const osg::Vec4 &color=osg::Vec4(1.0f, 1.0f, 1.0f, 1.0f))
- unsigned int **getNumberCellsX** () const
- void **setValue** (unsigned int i, const osg::Vec4 &color)
- const osg::Vec4 & **getValue** (unsigned int i) const

4.414 Detailed Description

1D variant of TransferFunction.

4.415 osg::Transform Class Reference



Public Types

- enum **ReferenceFrame** {
  RELATIVE_RF,
  ABSOLUTE_RF,
  ABSOLUTE_RF_INHERIT_VIEWPOINT }

Public Member Functions

- Transform (const Transform &, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_Node** (osg, Transform)
- virtual Transform * asTransform ()
- virtual const Transform * asTransform () const
- virtual MatrixTransform * **asMatrixTransform** ()
- virtual const MatrixTransform * **asMatrixTransform** () const
- virtual PositionAttitudeTransform * **asPositionAttitudeTransform** ()
- virtual const PositionAttitudeTransform * **asPositionAttitudeTransform** () const
- void setReferenceFrame (ReferenceFrame rf)
- ReferenceFrame **getReferenceFrame** () const
- virtual bool **computeLocalToWorldMatrix** (Matrix &matrix, NodeVisitor *) const
- virtual bool **computeWorldToLocalMatrix** (Matrix &matrix, NodeVisitor *) const
- virtual BoundingSphere computeBound () const

4.416 Detailed Description

A Transform is a group node for which all children are transformed by a 4x4 matrix. It is often used for positioning objects within a scene, producing trackball functionality or for animation.

Transform itself does not provide set/get functions, only the interface for defining what the 4x4 transformation is. Subclasses, such as MatrixTransform and PositionAttitudeTransform support the use of an osg::Matrix or a osg::Vec3/osgQuat respectively.

Note: If the transformation matrix scales the subgraph then the normals of the underlying geometry will need to be renormalized to be unit vectors once more. This can be done transparently through OpenGL's use of either GL_NORMALIZE and GL_RESCALE_NORMAL modes. For further background reading see the glNormalize documentation in the OpenGL Reference Guide (the blue book). To enable it in the OSG, you simply need to attach a local osg::StateSet to the osg::Transform, and set the appropriate mode to ON via stateset->setMode(GL_NORMALIZE, osg::StateAttribute::ON);

4.417 Constructor & Destructor Documentation

osg::Transform::Transform (const Transform &, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`**)**

Copy constructor using CopyOp to manage deep vs shallow copy.

4.418 Member Function Documentation

virtual Transform* osg::Transform::asTransform () `[inline, virtual]`

Convert 'this' into a Transform pointer if Node is a Transform, otherwise return 0. Equivalent to dynamic_cast<Transform*>(this).

Reimplemented from osg::Node.

virtual const Transform* osg::Transform::asTransform () const `[inline, virtual]`

convert 'const this' into a const Transform pointer if Node is a Transform, otherwise return 0. Equivalent to dynamic_cast<const Transform*>(this).

Reimplemented from osg::Node.

void osg::Transform::setReferenceFrame (ReferenceFrame *rf*)

Set the transform's ReferenceFrame, either to be relative to its parent reference frame, or relative to an absolute coordinate frame. RELATIVE_RF is the default. Note: Setting the ReferenceFrame to be ABSOLUTE_RF will also set the CullingActive flag on the transform, and hence all of its parents, to false, thereby disabling culling of it and all its parents. This is neccessary to prevent inappropriate culling, but may impact cull times if the absolute transform is deep in the scene graph. It is therefore recommended to only use absolute Transforms at the top of the scene, for such things as heads up displays. ABSOLUTE_RF_INHERIT_VIEWPOINT is the same as ABSOLUTE_RF except it adds the ability to use the parents view points position in world coordinates as its local viewpoint in the new coordinates frame. This is useful for Render to texture Cameras that wish to use the main views LOD range computation (which uses the viewpoint rather than the eye point) rather than use the local eye point defined by the this Transforms' abosolute view matrix.

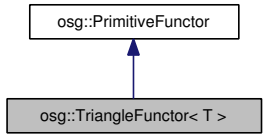
virtual BoundingSphere osg::Transform::computeBound () const `[virtual]`

Overrides Group's computeBound. There is no need to override in subclasses from osg::Transform since this computeBound() uses the underlying matrix (calling computeMatrix if required).

Reimplemented from osg::Group.

Reimplemented in osg::AutoTransform.

4.419 osg::TriangleFunctor< T > Class Template Reference



Public Member Functions

- void **setTreatVertexDataAsTemporary** (bool treatVertexDataAsTemporary)
- bool **getTreatVertexDataAsTemporary** () const
- virtual void setVertexArray (unsigned int, const Vec2 *)
- virtual void setVertexArray (unsigned int count, const Vec3 *vertices)

- virtual void setVertexArray (unsigned int, const Vec4 *)
- virtual void setVertexArray (unsigned int, const Vec2d *)
- virtual void setVertexArray (unsigned int, const Vec3d *)
- virtual void setVertexArray (unsigned int, const Vec4d *)
- virtual void drawArrays (GLenum mode, GLint first, GLsizei count)

  Mimics the OpenGL `glDrawArrays()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLubyte *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLushort *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLuint *indices)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void begin (GLenum mode)
- virtual void vertex (const Vec2 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec3 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const Vec4 &vert)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y, float z)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float x, float y, float z, float w)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void end ()

  Mimics the OpenGL `glEnd()` *function.*

4.420 Detailed Description

template<class T> class osg::TriangleFunctor< T >

Provides access to the triangles that compose an `osg::Drawable`. If the `Drawable` is not composed of triangles, the `TriangleFunctor` will convert the primitives to triangles whenever possible.

Notice that `TriangleFunctor` is a class template, and that it inherits from its template parameter `T`. This template parameter must implement `T::operator() (const osg::Vec3 v1, const osg::Vec3 v2, const osg::Vec3 v3, bool treatVertexDataAsTemporary)`, which will be called for every triangle when the functor is applied to a `Drawable`. Parameters `v1`, `v2`, and `v3` are the triangle vertices. The fourth parameter, `treatVertexDataAsTemporary`, indicates whether these vertices are coming from a "real" vertex array, or from a temporary vertex array, created by the `TriangleFunctor` from some other geometry representation.

See also:

`PrimitiveFunctor` for general usage hints.

4.421 Member Function Documentation

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned *count*, const Vec2 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned int *count*, const Vec3 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned *count*, const Vec4 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned *count*, const Vec2d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned *count*, const Vec3d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::setVertexArray (unsigned *count*, const Vec4d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

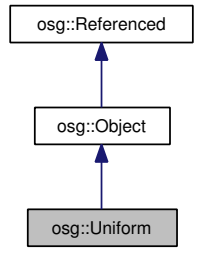
Implements osg::PrimitiveFunctor.

template<class T> virtual void osg::TriangleFunctor< T >::begin (GLenum *mode*) `[inline, virtual]`

Note: begin(..),vertex(..) & end() are convenience methods for adapting non vertex array primitives to vertex array based primitives. This is done to simplify the implementation of primtive functor subclasses - users only need override drawArray and drawElements.

Implements osg::PrimitiveFunctor.

4.422 osg::Uniform Class Reference



Public Types

- enum **Type** {
  FLOAT,
  FLOAT_VEC2,
  FLOAT_VEC3,
  FLOAT_VEC4,
  INT,
  INT_VEC2,

INT_VEC3,
INT_VEC4,
BOOL,
BOOL_VEC2,
BOOL_VEC3,
BOOL_VEC4,
FLOAT_MAT2,
FLOAT_MAT3,
FLOAT_MAT4,
SAMPLER_1D,
SAMPLER_2D,
SAMPLER_3D,
SAMPLER_CUBE,
SAMPLER_1D_SHADOW,
SAMPLER_2D_SHADOW,
SAMPLER_1D_ARRAY,
SAMPLER_2D_ARRAY,
SAMPLER_1D_ARRAY_SHADOW,
SAMPLER_2D_ARRAY_SHADOW,
UNDEFINED }

- typedef std::vector< StateSet * > ParentList

Public Member Functions

- **Uniform** (Type type, const std::string &name, int numElements=1)
- Uniform (const Uniform &rhs, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Object** (osg, Uniform)
- bool setType (Type t)
- Type getType () const
- void setName (const std::string &name)
- void setNumElements (unsigned int numElements)
- unsigned int getNumElements () const
- unsigned int getInternalArrayNumElements () const
- Uniform (const char *name, float f)
- **Uniform** (const char *name, int i)
- **Uniform** (const char *name, bool b)
- **Uniform** (const char *name, const osg::Vec2 &v2)
- **Uniform** (const char *name, const osg::Vec3 &v3)
- **Uniform** (const char *name, const osg::Vec4 &v4)
- **Uniform** (const char *name, const osg::Matrix2 &m2)

- **Uniform** (const char *name, const osg::Matrix3 &m3)
- **Uniform** (const char *name, const osg::Matrixf &m4)
- **Uniform** (const char *name, const osg::Matrixd &m4)
- **Uniform** (const char *name, int i0, int i1)
- **Uniform** (const char *name, int i0, int i1, int i2)
- **Uniform** (const char *name, int i0, int i1, int i2, int i3)
- **Uniform** (const char *name, bool b0, bool b1)
- **Uniform** (const char *name, bool b0, bool b1, bool b2)
- **Uniform** (const char *name, bool b0, bool b1, bool b2, bool b3)
- virtual int compare (const Uniform &rhs) const
- virtual int **compareData** (const Uniform &rhs) const
- bool **operator<** (const Uniform &rhs) const
- bool **operator==** (const Uniform &rhs) const
- bool **operator!=** (const Uniform &rhs) const
- void **copyData** (const Uniform &rhs)
- const ParentList & getParents () const
- ParentList getParents ()
- StateSet * **getParent** (unsigned int i)
- const StateSet * getParent (unsigned int i) const
- unsigned int getNumParents () const
- bool set (float f)
- bool **set** (int i)
- bool **set** (bool b)
- bool **set** (const osg::Vec2 &v2)
- bool **set** (const osg::Vec3 &v3)
- bool **set** (const osg::Vec4 &v4)
- bool **set** (const osg::Matrix2 &m2)
- bool **set** (const osg::Matrix3 &m3)
- bool **set** (const osg::Matrixf &m4)
- bool **set** (const osg::Matrixd &m4)
- bool **set** (int i0, int i1)
- bool **set** (int i0, int i1, int i2)
- bool **set** (int i0, int i1, int i2, int i3)
- bool **set** (bool b0, bool b1)
- bool **set** (bool b0, bool b1, bool b2)
- bool **set** (bool b0, bool b1, bool b2, bool b3)
- bool get (float &f) const
- bool **get** (int &i) const
- bool **get** (bool &b) const
- bool **get** (osg::Vec2 &v2) const
- bool **get** (osg::Vec3 &v3) const
- bool **get** (osg::Vec4 &v4) const
- bool **get** (osg::Matrix2 &m2) const
- bool **get** (osg::Matrix3 &m3) const
- bool **get** (osg::Matrixf &m4) const
- bool **get** (osg::Matrixd &m4) const

- bool **get** (int &i0, int &i1) const
- bool **get** (int &i0, int &i1, int &i2) const
- bool **get** (int &i0, int &i1, int &i2, int &i3) const
- bool **get** (bool &b0, bool &b1) const
- bool **get** (bool &b0, bool &b1, bool &b2) const
- bool **get** (bool &b0, bool &b1, bool &b2, bool &b3) const
- bool setElement (unsigned int index, float f)
- bool **setElement** (unsigned int index, int i)
- bool **setElement** (unsigned int index, bool b)
- bool **setElement** (unsigned int index, const osg::Vec2 &v2)
- bool **setElement** (unsigned int index, const osg::Vec3 &v3)
- bool **setElement** (unsigned int index, const osg::Vec4 &v4)
- bool **setElement** (unsigned int index, const osg::Matrix2 &m2)
- bool **setElement** (unsigned int index, const osg::Matrix3 &m3)
- bool **setElement** (unsigned int index, const osg::Matrixf &m4)
- bool **setElement** (unsigned int index, const osg::Matrixd &m4)
- bool **setElement** (unsigned int index, int i0, int i1)
- bool **setElement** (unsigned int index, int i0, int i1, int i2)
- bool **setElement** (unsigned int index, int i0, int i1, int i2, int i3)
- bool **setElement** (unsigned int index, bool b0, bool b1)
- bool **setElement** (unsigned int index, bool b0, bool b1, bool b2)
- bool **setElement** (unsigned int index, bool b0, bool b1, bool b2, bool b3)
- bool getElement (unsigned int index, float &f) const
- bool **getElement** (unsigned int index, int &i) const
- bool **getElement** (unsigned int index, bool &b) const
- bool **getElement** (unsigned int index, osg::Vec2 &v2) const
- bool **getElement** (unsigned int index, osg::Vec3 &v3) const
- bool **getElement** (unsigned int index, osg::Vec4 &v4) const
- bool **getElement** (unsigned int index, osg::Matrix2 &m2) const
- bool **getElement** (unsigned int index, osg::Matrix3 &m3) const
- bool **getElement** (unsigned int index, osg::Matrixf &m4) const
- bool **getElement** (unsigned int index, osg::Matrixd &m4) const
- bool **getElement** (unsigned int index, int &i0, int &i1) const
- bool **getElement** (unsigned int index, int &i0, int &i1, int &i2) const
- bool **getElement** (unsigned int index, int &i0, int &i1, int &i2, int &i3) const
- bool **getElement** (unsigned int index, bool &b0, bool &b1) const
- bool **getElement** (unsigned int index, bool &b0, bool &b1, bool &b2) const
- bool **getElement** (unsigned int index, bool &b0, bool &b1, bool &b2, bool &b3) const
- void setUpdateCallback (Callback *uc)
- Callback * getUpdateCallback ()
- const Callback * getUpdateCallback () const
- void setEventCallback (Callback *ec)
- Callback * getEventCallback ()
- const Callback * getEventCallback () const
- void dirty ()
- bool setArray (FloatArray *array)

- bool **setArray** (IntArray *array)
- FloatArray * getFloatArray ()
- const FloatArray * **getFloatArray** () const
- IntArray * getIntArray ()
- const IntArray * **getIntArray** () const
- void **setModifiedCount** (unsigned int mc)
- unsigned int **getModifiedCount** () const
- void **apply** (const GL2Extensions *ext, GLint location) const

Static Public Member Functions

- static const char * getTypename (Type t)
- static int getTypeNumComponents (Type t)
- static Uniform::Type getTypeId (const std::string &tname)
- static Type getGlApiType (Type t)
- static GLenum getInternalArrayType (Type t)

Friends

- class osg::StateSet

Classes

- struct **Callback**

4.423 Detailed Description

Uniform encapsulates glUniform values

4.424 Member Typedef Documentation

typedef std::vector<StateSet*> osg::Uniform::ParentList

A vector of osg::StateSet pointers which is used to store the parent(s) of this Uniform, the parents could be osg::Node or osg::Drawable.

4.425 Constructor & Destructor Documentation

osg::Uniform::Uniform (const Uniform & *rhs*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

osg::Uniform::Uniform (const char * *name*, float *f*) `[explicit]`

convenient scalar (non-array) constructors w/ assignment

4.426 Member Function Documentation

bool osg::Uniform::setType (Type *t*)

Set the type of glUniform, ensuring it is only set once.

Type osg::Uniform::getType () const `[inline]`

Get the type of glUniform as enum.

void osg::Uniform::setName (const std::string & *name*)

Set the name of the glUniform, ensuring it is only set once.

Reimplemented from osg::Object.

void osg::Uniform::setNumElements (unsigned int *numElements*)

Set the length of a uniform, ensuring it is only set once (1==scalar)

unsigned int osg::Uniform::getNumElements () const `[inline]`

Get the number of GLSL elements of the osg::Uniform (1==scalar)

unsigned int osg::Uniform::getInternalArrayNumElements () const

Get the number of elements required for the internal data array. Returns 0 if the osg::Uniform is not properly configured.

static const char* osg::Uniform::getTypename (Type *t*) `[static]`

Return the name of a Type enum as string.

static int osg::Uniform::getTypeNumComponents (Type *t*) `[static]`

Return the the number of components for a GLSL type.

static Uniform::Type osg::Uniform::getTypeId (const std::string & *tname*) `[static]`

Return the Type enum of a Uniform typename string

static Type osg::Uniform::getGlApiType (Type *t*) `[static]`

Return the GL API type corresponding to a GLSL type

static GLenum osg::Uniform::getInternalArrayType (Type *t*) `[static]`

Return the internal data array type corresponding to a GLSL type

virtual int osg::Uniform::compare (const Uniform & *rhs*) const `[virtual]`

return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

const ParentList& osg::Uniform::getParents () const `[inline]`

Get the parent list of this Uniform.

ParentList osg::Uniform::getParents () `[inline]`

Get the a copy of parent list of node. A copy is returned to prevent modification of the parent list.

const StateSet* osg::Uniform::getParent (unsigned int *i*) const `[inline]`

Get a single const parent of this Uniform.

Parameters:

i index of the parent to get.

Returns:

the parent i.

unsigned int osg::Uniform::getNumParents () const `[inline]`

Get the number of parents of this Uniform.

Returns:

the number of parents of this Uniform.

bool osg::Uniform::set (float *f*)

convenient scalar (non-array) value assignment

bool osg::Uniform::get (float & *f*) const

convenient scalar (non-array) value query

bool osg::Uniform::setElement (unsigned int *index*, float *f*)

value assignment for array uniforms

bool osg::Uniform::getElement (unsigned int *index*, float & *f*) const

value query for array uniforms

void osg::Uniform::setUpdateCallback (Callback * *uc*)

Set the UpdateCallback which allows users to attach customize the updating of an object during the update traversal.

Callback* osg::Uniform::getUpdateCallback () `[inline]`

Get the non const UpdateCallback.

const Callback* osg::Uniform::getUpdateCallback () const `[inline]`

Get the const UpdateCallback.

void osg::Uniform::setEventCallback (Callback * *ec*)

Set the EventCallback which allows users to attach customize the updating of an object during the Event traversal.

Callback* osg::Uniform::getEventCallback () `[inline]`

Get the non const EventCallback.

const Callback* osg::Uniform::getEventCallback () const `[inline]`

Get the const EventCallback.

void osg::Uniform::dirty () `[inline]`

Increment the modified count on the Uniform so Programs watching it know it update themselves. NOTE: autotomatically called during osg::Uniform::set*(); you must call if modifying the internal data array directly.

bool osg::Uniform::setArray (FloatArray * *array*)

Set the internal data array for a osg::Uniform

FloatArray* osg::Uniform::getFloatArray () `[inline]`

Get the internal data array for a float osg::Uniform.

IntArray* osg::Uniform::getIntArray () `[inline]`

Get the internal data array for an int osg::Uniform.

4.427 osg::Vec2b Class Reference

Public Types

- enum { **num_components** }
- typedef char value_type

Public Member Functions

- **Vec2b** (char r, char g)
- bool **operator==** (const Vec2b &v) const
- bool **operator!=** (const Vec2b &v) const
- bool **operator<** (const Vec2b &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y)
- void **set** (const Vec2b &rhs)
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type **x** () const
- value_type **y** () const
- value_type & **r** ()

- value_type & **g** ()
- value_type **r** () const
- value_type **g** () const
- Vec2b operator ∗ (float rhs) const
- Vec2b & operator ∗= (float rhs)
- Vec2b operator/ (float rhs) const
- Vec2b & operator/= (float rhs)
- Vec2b operator+ (const Vec2b &rhs) const
- Vec2b & operator+= (const Vec2b &rhs)
- Vec2b operator- (const Vec2b &rhs) const
- Vec2b & operator-= (const Vec2b &rhs)

Public Attributes

- value_type _v [2]

4.428 Detailed Description

General purpose float triple. Uses include representation of color coordinates. No support yet added for float ∗ Vec2b - is it necessary? Need to define a non-member non-friend operator∗ etc. Vec2b ∗ float is okay

4.429 Member Typedef Documentation

typedef char osg::Vec2b::value_type

Type of Vec class.

4.430 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.431 Member Function Documentation

Vec2b osg::Vec2b::operator ∗ (float *rhs*) const `[inline]`

Multiply by scalar.

Vec2b& osg::Vec2b::operator *= (float *rhs*) `[inline]`

Unary multiply by scalar.

Vec2b osg::Vec2b::operator/ (float *rhs*) const `[inline]`

Divide by scalar.

Vec2b& osg::Vec2b::operator/= (float *rhs*) `[inline]`

Unary divide by scalar.

Vec2b osg::Vec2b::operator+ (const Vec2b & *rhs*) const `[inline]`

Binary vector add.

Vec2b& osg::Vec2b::operator+= (const Vec2b & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec2b osg::Vec2b::operator- (const Vec2b & *rhs*) const `[inline]`

Binary vector subtract.

Vec2b& osg::Vec2b::operator-= (const Vec2b & *rhs*) `[inline]`

Unary vector subtract.

4.432 Member Data Documentation

value_type osg::Vec2b::_v[2]

Vec member varaible.

4.433 osg::Vec2d Class Reference

Public Types

- enum { **num_components** }
- typedef double value_type

Public Member Functions

- **Vec2d** (value_type x, value_type y)
- **Vec2d** (const Vec2f &vec)
- **operator Vec2f** () const
- bool **operator==** (const Vec2d &v) const
- bool **operator!=** (const Vec2d &v) const
- bool **operator**< (const Vec2d &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y)
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type **x** () const
- value_type **y** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec2d &rhs) const
- const Vec2d operator * (value_type rhs) const
- Vec2d & operator *= (value_type rhs)
- const Vec2d operator/ (value_type rhs) const
- Vec2d & operator/= (value_type rhs)
- const Vec2d operator+ (const Vec2d &rhs) const
- Vec2d & operator+= (const Vec2d &rhs)
- const Vec2d operator- (const Vec2d &rhs) const
- Vec2d & operator-= (const Vec2d &rhs)
- const Vec2d operator- () const
- value_type length () const
- value_type length2 (void) const
- value_type normalize ()

Public Attributes

- value_type _v [2]

4.434 Detailed Description

General purpose double pair, uses include representation of texture coordinates. No support yet added for double * Vec2d - is it necessary? Need to define a non-member non-friend operator* etc. BTW: Vec2d * double is okay

4.435 Member Typedef Documentation

typedef double osg::Vec2d::value_type

Type of Vec class.

4.436 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.437 Member Function Documentation

value_type osg::Vec2d::operator * (const Vec2d & *rhs*) const `[inline]`

Dot product.

const Vec2d osg::Vec2d::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec2d& osg::Vec2d::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

const Vec2d osg::Vec2d::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec2d& osg::Vec2d::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

const Vec2d osg::Vec2d::operator+ (const Vec2d & *rhs*) const `[inline]`

Binary vector add.

Vec2d& osg::Vec2d::operator+= (const Vec2d & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

const Vec2d osg::Vec2d::operator- (const Vec2d & *rhs*) const `[inline]`

Binary vector subtract.

Vec2d& osg::Vec2d::operator-= (const Vec2d & *rhs*) `[inline]`

Unary vector subtract.

const Vec2d osg::Vec2d::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec2d.

value_type osg::Vec2d::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec2d::length2 (void) const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec2d::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.438 Member Data Documentation

value_type osg::Vec2d::_v[2]

Vec member varaible.

4.439 osg::Vec2f Class Reference

Public Types

- enum { **num_components** }
- typedef float value_type

Public Member Functions

- **Vec2f** (value_type x, value_type y)

- bool **operator==** (const Vec2f &v) const
- bool **operator!=** (const Vec2f &v) const
- bool **operator<** (const Vec2f &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y)
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type **x** () const
- value_type **y** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec2f &rhs) const
- const Vec2f operator * (value_type rhs) const
- Vec2f & operator *= (value_type rhs)
- const Vec2f operator/ (value_type rhs) const
- Vec2f & operator/= (value_type rhs)
- const Vec2f operator+ (const Vec2f &rhs) const
- Vec2f & operator+= (const Vec2f &rhs)
- const Vec2f operator- (const Vec2f &rhs) const
- Vec2f & operator-= (const Vec2f &rhs)
- const Vec2f operator- () const
- value_type length () const
- value_type length2 (void) const
- value_type normalize ()

Public Attributes

- value_type _v [2]

4.440 Detailed Description

General purpose float pair. Uses include representation of texture coordinates. No support yet added for float * Vec2f - is it necessary? Need to define a non-member non-friend operator* etc. BTW: Vec2f * float is okay

4.441 Member Typedef Documentation

typedef float osg::Vec2f::value_type

Type of Vec class.

4.442 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.443 Member Function Documentation

value_type osg::Vec2f::operator * (const Vec2f & *rhs*) const `[inline]`

Dot product.

const Vec2f osg::Vec2f::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec2f& osg::Vec2f::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

const Vec2f osg::Vec2f::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec2f& osg::Vec2f::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

const Vec2f osg::Vec2f::operator+ (const Vec2f & *rhs*) const `[inline]`

Binary vector add.

Vec2f& osg::Vec2f::operator+= (const Vec2f & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

const Vec2f osg::Vec2f::operator- (const Vec2f & *rhs*) const `[inline]`

Binary vector subtract.

Vec2f& osg::Vec2f::operator-= (const Vec2f & *rhs*) `[inline]`

Unary vector subtract.

const Vec2f osg::Vec2f::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec2f.

value_type osg::Vec2f::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec2f::length2 (void) const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec2f::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.444 Member Data Documentation

value_type osg::Vec2f::_v[2]

Vec member varaible.

4.445 osg::Vec3b Class Reference

Public Types

- enum { **num_components** }
- typedef char value_type

Public Member Functions

- **Vec3b** (value_type r, value_type g, value_type b)
- bool **operator==** (const Vec3b &v) const
- bool **operator!=** (const Vec3b &v) const
- bool **operator<** (const Vec3b &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type r, value_type g, value_type b)
- void **set** (const Vec3b &rhs)
- value_type & **operator[ ]** (unsigned int i)
- value_type **operator[ ]** (unsigned int i) const

- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- value_type & **r** ()
- value_type & **g** ()
- value_type & **b** ()
- value_type **r** () const
- value_type **g** () const
- value_type **b** () const
- Vec3b operator ∗ (float rhs) const
- Vec3b & operator ∗= (float rhs)
- Vec3b operator/ (float rhs) const
- Vec3b & operator/= (float rhs)
- Vec3b operator+ (const Vec3b &rhs) const
- Vec3b & operator+= (const Vec3b &rhs)
- Vec3b operator- (const Vec3b &rhs) const
- Vec3b & operator-= (const Vec3b &rhs)

Public Attributes

- value_type _v [3]

4.446 Detailed Description

General purpose float triple. Uses include representation of color coordinates. No support yet added for float ∗ Vec3b - is it necessary? Need to define a non-member non-friend operator∗ etc. Vec3b ∗ float is okay

4.447 Member Typedef Documentation

typedef char osg::Vec3b::value_type

Type of Vec class.

4.448 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.449 Member Function Documentation

Vec3b osg::Vec3b::operator ∗ (float *rhs*) const `[inline]`

Multiply by scalar.

Vec3b& osg::Vec3b::operator ∗= (float *rhs*) `[inline]`

Unary multiply by scalar.

Vec3b osg::Vec3b::operator/ (float *rhs*) const `[inline]`

Divide by scalar.

Vec3b& osg::Vec3b::operator/= (float *rhs*) `[inline]`

Unary divide by scalar.

Vec3b osg::Vec3b::operator+ (const Vec3b & *rhs*) const `[inline]`

Binary vector add.

Vec3b& osg::Vec3b::operator+= (const Vec3b & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec3b osg::Vec3b::operator- (const Vec3b & *rhs*) const `[inline]`

Binary vector subtract.

Vec3b& osg::Vec3b::operator-= (const Vec3b & *rhs*) `[inline]`

Unary vector subtract.

4.450 Member Data Documentation

value_type osg::Vec3b::_v[3]

Vec member varaible.

4.451 osg::Vec3d Class Reference

Public Types

- enum { **num_components** }
- typedef double value_type

Public Member Functions

- **Vec3d** (const Vec3f &vec)
- **operator Vec3f** () const
- **Vec3d** (value_type x, value_type y, value_type z)
- **Vec3d** (const Vec2d &v2, value_type zz)
- bool **operator==** (const Vec3d &v) const
- bool **operator!=** (const Vec3d &v) const
- bool **operator**< (const Vec3d &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y, value_type z)
- void **set** (const Vec3d &rhs)
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec3d &rhs) const
- const Vec3d operator^ (const Vec3d &rhs) const
- const Vec3d operator * (value_type rhs) const
- Vec3d & operator *= (value_type rhs)
- const Vec3d operator/ (value_type rhs) const
- Vec3d & operator/= (value_type rhs)
- const Vec3d operator+ (const Vec3d &rhs) const
- Vec3d & operator+= (const Vec3d &rhs)
- const Vec3d operator- (const Vec3d &rhs) const
- Vec3d & operator-= (const Vec3d &rhs)
- const Vec3d operator- () const
- value_type length () const
- value_type length2 () const
- value_type normalize ()

Public Attributes

- value_type _v [3]

4.452 Detailed Description

General purpose double triple for use as vertices, vectors and normals. Provides general math operations from addition through to cross products. No support yet added for double * Vec3d - is it necessary? Need to define a non-member non-friend operator* etc. Vec3d * double is okay

4.453 Member Typedef Documentation

typedef double osg::Vec3d::value_type

Type of Vec class.

4.454 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.455 Member Function Documentation

value_type osg::Vec3d::operator * (const Vec3d & *rhs*) const `[inline]`

Dot product.

const Vec3d osg::Vec3d::operator^ (const Vec3d & *rhs*) const `[inline]`

Cross product.

const Vec3d osg::Vec3d::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec3d& osg::Vec3d::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

const Vec3d osg::Vec3d::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec3d& osg::Vec3d::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

const Vec3d osg::Vec3d::operator+ (const Vec3d & *rhs*) const `[inline]`

Binary vector add.

Vec3d& osg::Vec3d::operator+= (const Vec3d & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

const Vec3d osg::Vec3d::operator- (const Vec3d & *rhs*) const `[inline]`

Binary vector subtract.

Vec3d& osg::Vec3d::operator-= (const Vec3d & *rhs*) `[inline]`

Unary vector subtract.

const Vec3d osg::Vec3d::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec3d.

value_type osg::Vec3d::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec3d::length2 () const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec3d::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.456 Member Data Documentation

value_type osg::Vec3d::_v[3]

Vec member varaible.

4.457 osg::Vec3f Class Reference

Public Types

- enum { **num_components** }
- typedef float value_type

Public Member Functions

- **Vec3f** (value_type x, value_type y, value_type z)
- **Vec3f** (const Vec2f &v2, value_type zz)
- bool **operator==** (const Vec3f &v) const
- bool **operator!=** (const Vec3f &v) const
- bool **operator**< (const Vec3f &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y, value_type z)
- void **set** (const Vec3f &rhs)
- value_type & **operator[ ]** (int i)
- value_type **operator[ ]** (int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec3f &rhs) const
- const Vec3f operator^ (const Vec3f &rhs) const
- const Vec3f operator * (value_type rhs) const
- Vec3f & operator *= (value_type rhs)
- const Vec3f operator/ (value_type rhs) const
- Vec3f & operator/= (value_type rhs)
- const Vec3f operator+ (const Vec3f &rhs) const
- Vec3f & operator+= (const Vec3f &rhs)
- const Vec3f operator- (const Vec3f &rhs) const
- Vec3f & operator-= (const Vec3f &rhs)
- const Vec3f operator- () const
- value_type length () const
- value_type length2 () const
- value_type normalize ()

Public Attributes

- value_type _v [3]

4.458 Detailed Description

General purpose float triple for use as vertices, vectors and normals. Provides general math operations from addition through to cross products. No support yet added for float * Vec3f - is it necessary? Need to define a non-member non-friend operator* etc. Vec3f * float is okay

4.459 Member Typedef Documentation

typedef float osg::Vec3f::value_type

Type of Vec class.

4.460 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.461 Member Function Documentation

value_type osg::Vec3f::operator * (const Vec3f & *rhs*) const `[inline]`

Dot product.

const Vec3f osg::Vec3f::operator^ (const Vec3f & *rhs*) const `[inline]`

Cross product.

const Vec3f osg::Vec3f::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec3f& osg::Vec3f::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

const Vec3f osg::Vec3f::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec3f& osg::Vec3f::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

const Vec3f osg::Vec3f::operator+ (const Vec3f & *rhs*) const `[inline]`

Binary vector add.

Vec3f& osg::Vec3f::operator+= (const Vec3f & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

const Vec3f osg::Vec3f::operator- (const Vec3f & *rhs*) const `[inline]`

Binary vector subtract.

Vec3f& osg::Vec3f::operator-= (const Vec3f & *rhs*) `[inline]`

Unary vector subtract.

const Vec3f osg::Vec3f::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec3f.

value_type osg::Vec3f::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec3f::length2 () const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec3f::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.462 Member Data Documentation

value_type osg::Vec3f::_v[3]

Vec member varaible.

4.463 osg::Vec4b Class Reference

Public Types

- enum { **num_components** }
- typedef char value_type

Public Member Functions

- **Vec4b** (value_type x, value_type y, value_type z, value_type w)
- bool **operator==** (const Vec4b &v) const
- bool **operator!=** (const Vec4b &v) const
- bool **operator**< (const Vec4b &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y, value_type z, value_type w)
- value_type & **operator[ ]** (unsigned int i)
- value_type **operator[ ]** (unsigned int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type & **w** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- value_type **w** () const
- value_type & **r** ()
- value_type & **g** ()
- value_type & **b** ()
- value_type & **a** ()
- value_type **r** () const
- value_type **g** () const
- value_type **b** () const
- value_type **a** () const
- Vec4b operator * (float rhs) const
- Vec4b & operator *= (float rhs)
- Vec4b operator/ (float rhs) const
- Vec4b & operator/= (float rhs)
- Vec4b operator+ (const Vec4b &rhs) const
- Vec4b & operator+= (const Vec4b &rhs)
- Vec4b operator- (const Vec4b &rhs) const
- Vec4b & operator-= (const Vec4b &rhs)

Public Attributes

- value_type _v [4]

4.464 Detailed Description

General purpose float triple. Uses include representation of color coordinates. No support yet added for float * Vec4b - is it necessary? Need to define a non-member non-friend operator* etc. Vec4b * float is okay

4.465 Member Typedef Documentation

typedef char osg::Vec4b::value_type

Type of Vec class.

4.466 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.467 Member Function Documentation

Vec4b osg::Vec4b::operator * (float *rhs*) const `[inline]`

Multiply by scalar.

Vec4b& osg::Vec4b::operator *= (float *rhs*) `[inline]`

Unary multiply by scalar.

Vec4b osg::Vec4b::operator/ (float *rhs*) const `[inline]`

Divide by scalar.

Vec4b& osg::Vec4b::operator/= (float *rhs*) `[inline]`

Unary divide by scalar.

Vec4b osg::Vec4b::operator+ (const Vec4b & *rhs*) const `[inline]`

Binary vector add.

Vec4b& osg::Vec4b::operator+= (const Vec4b & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec4b osg::Vec4b::operator- (const Vec4b & *rhs*) const `[inline]`

Binary vector subtract.

Vec4b& osg::Vec4b::operator-= (const Vec4b & *rhs*) `[inline]`

Unary vector subtract.

4.468 Member Data Documentation

value_type osg::Vec4b::_v[4]

Vec member varaible.

4.469 osg::Vec4d Class Reference

Public Types

- enum { **num_components** }
- typedef double value_type

Public Member Functions

- **Vec4d** (value_type x, value_type y, value_type z, value_type w)
- **Vec4d** (const Vec3d &v3, value_type w)
- **Vec4d** (const Vec4f &vec)
- **operator Vec4f** () const
- bool **operator==** (const Vec4d &v) const
- bool **operator!=** (const Vec4d &v) const
- bool **operator**< (const Vec4d &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y, value_type z, value_type w)

- value_type & **operator[ ]** (unsigned int i)
- value_type **operator[ ]** (unsigned int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type & **w** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- value_type **w** () const
- value_type & **r** ()
- value_type & **g** ()
- value_type & **b** ()
- value_type & **a** ()
- value_type **r** () const
- value_type **g** () const
- value_type **b** () const
- value_type **a** () const
- unsigned int **asABGR** () const
- unsigned int **asRGBA** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec4d &rhs) const
- Vec4d operator * (value_type rhs) const
- Vec4d & operator *= (value_type rhs)
- Vec4d operator/ (value_type rhs) const
- Vec4d & operator/= (value_type rhs)
- Vec4d operator+ (const Vec4d &rhs) const
- Vec4d & operator+= (const Vec4d &rhs)
- Vec4d operator- (const Vec4d &rhs) const
- Vec4d & operator-= (const Vec4d &rhs)
- const Vec4d operator- () const
- value_type length () const
- value_type length2 () const
- value_type normalize ()

Public Attributes

- value_type _v [4]

4.470 Detailed Description

General purpose double quad. Uses include representation of color coordinates. No support yet added for double * Vec4d - is it necessary? Need to define a non-member non-friend operator* etc. Vec4d * double is okay

4.471 Member Typedef Documentation

typedef double osg::Vec4d::value_type

Type of Vec class.

4.472 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.473 Member Function Documentation

value_type osg::Vec4d::operator * (const Vec4d & *rhs*) const `[inline]`

Dot product.

Vec4d osg::Vec4d::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec4d& osg::Vec4d::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

Vec4d osg::Vec4d::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec4d& osg::Vec4d::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

Vec4d osg::Vec4d::operator+ (const Vec4d & *rhs*) const `[inline]`

Binary vector add.

Vec4d& osg::Vec4d::operator+= (const Vec4d & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec4d osg::Vec4d::operator- (const Vec4d & *rhs*) const `[inline]`

Binary vector subtract.

Vec4d& osg::Vec4d::operator-= (const Vec4d & *rhs*) `[inline]`

Unary vector subtract.

const Vec4d osg::Vec4d::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec4d.

value_type osg::Vec4d::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec4d::length2 () const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec4d::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.474 Member Data Documentation

value_type osg::Vec4d::_v[4]

Vec member varaible.

4.475 osg::Vec4f Class Reference

Public Types

- enum { **num_components** }
- typedef float value_type

Public Member Functions

- **Vec4f** (value_type x, value_type y, value_type z, value_type w)

- **Vec4f** (const Vec3f &v3, value_type w)
- bool **operator==** (const Vec4f &v) const
- bool **operator!=** (const Vec4f &v) const
- bool **operator**< (const Vec4f &v) const
- value_type * **ptr** ()
- const value_type * **ptr** () const
- void **set** (value_type x, value_type y, value_type z, value_type w)
- value_type & **operator[ ]** (unsigned int i)
- value_type **operator[ ]** (unsigned int i) const
- value_type & **x** ()
- value_type & **y** ()
- value_type & **z** ()
- value_type & **w** ()
- value_type **x** () const
- value_type **y** () const
- value_type **z** () const
- value_type **w** () const
- value_type & **r** ()
- value_type & **g** ()
- value_type & **b** ()
- value_type & **a** ()
- value_type **r** () const
- value_type **g** () const
- value_type **b** () const
- value_type **a** () const
- unsigned int **asABGR** () const
- unsigned int **asRGBA** () const
- bool **valid** () const
- bool **isNaN** () const
- value_type operator * (const Vec4f &rhs) const
- Vec4f operator * (value_type rhs) const
- Vec4f & operator *= (value_type rhs)
- Vec4f operator/ (value_type rhs) const
- Vec4f & operator/= (value_type rhs)
- Vec4f operator+ (const Vec4f &rhs) const
- Vec4f & operator+= (const Vec4f &rhs)
- Vec4f operator- (const Vec4f &rhs) const
- Vec4f & operator-= (const Vec4f &rhs)
- const Vec4f operator- () const
- value_type length () const
- value_type length2 () const
- value_type normalize ()

Public Attributes

- value_type _v [4]

4.476 Detailed Description

General purpose float quad. Uses include representation of color coordinates. No support yet added for float * Vec4f - is it necessary? Need to define a non-member non-friend operator* etc. Vec4f * float is okay

4.477 Member Typedef Documentation

typedef float osg::Vec4f::value_type

Type of Vec class.

4.478 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.479 Member Function Documentation

value_type osg::Vec4f::operator * (const Vec4f & *rhs*) const `[inline]`

Dot product.

Vec4f osg::Vec4f::operator * (value_type *rhs*) const `[inline]`

Multiply by scalar.

Vec4f& osg::Vec4f::operator *= (value_type *rhs*) `[inline]`

Unary multiply by scalar.

Vec4f osg::Vec4f::operator/ (value_type *rhs*) const `[inline]`

Divide by scalar.

Vec4f& osg::Vec4f::operator/= (value_type *rhs*) `[inline]`

Unary divide by scalar.

Vec4f osg::Vec4f::operator+ (const Vec4f & *rhs*) const `[inline]`

Binary vector add.

Vec4f& osg::Vec4f::operator+= (const Vec4f & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec4f osg::Vec4f::operator- (const Vec4f & *rhs*) const `[inline]`

Binary vector subtract.

Vec4f& osg::Vec4f::operator-= (const Vec4f & *rhs*) `[inline]`

Unary vector subtract.

const Vec4f osg::Vec4f::operator- () const `[inline]`

Negation operator. Returns the negative of the Vec4f.

value_type osg::Vec4f::length () const `[inline]`

Length of the vector = sqrt(vec . vec)

value_type osg::Vec4f::length2 () const `[inline]`

Length squared of the vector = vec . vec

value_type osg::Vec4f::normalize () `[inline]`

Normalize the vector so that it has length unity. Returns the previous length of the vector.

4.480 Member Data Documentation

value_type osg::Vec4f::_v[4]

Vec member varaible.

4.481 osg::Vec4ub Class Reference

Public Types

- enum { **num_components** }
- typedef unsigned char value_type

Public Member Functions

- **Vec4ub** (value_type x, value_type y, value_type z, value_type w)
- bool **operator==** (const Vec4ub &v) const
- bool **operator!=** (const Vec4ub &v) const
- bool **operator**< (const Vec4ub &v) const
- unsigned char * **ptr** ()
- const unsigned char * **ptr** () const
- void **set** (unsigned char r, unsigned char g, unsigned char b, unsigned char a)
- unsigned char & **operator[ ]** (unsigned int i)
- unsigned char **operator[ ]** (unsigned int i) const
- unsigned char & **r** ()
- unsigned char & **g** ()
- unsigned char & **b** ()
- unsigned char & **a** ()
- unsigned char **r** () const
- unsigned char **g** () const
- unsigned char **b** () const
- unsigned char **a** () const
- Vec4ub operator * (float rhs) const
- Vec4ub & operator *= (float rhs)
- Vec4ub operator/ (float rhs) const
- Vec4ub & operator/= (float rhs)
- Vec4ub operator+ (const Vec4ub &rhs) const
- Vec4ub & operator+= (const Vec4ub &rhs)
- Vec4ub operator- (const Vec4ub &rhs) const
- Vec4ub & operator-= (const Vec4ub &rhs)

Public Attributes

- value_type _v [4]

4.482 Detailed Description

General purpose float quad. Uses include representation of color coordinates. No support yet added for float ∗ Vec4ub - is it necessary? Need to define a non-member non-friend operator∗ etc. Vec4ub ∗ float is okay

4.483 Member Typedef Documentation

typedef unsigned char osg::Vec4ub::value_type

Type of Vec class.

4.484 Member Enumeration Documentation

anonymous enum

Number of vector components.

4.485 Member Function Documentation

Vec4ub osg::Vec4ub::operator ∗ (float *rhs*) const `[inline]`

Multiply by scalar.

Vec4ub& osg::Vec4ub::operator ∗= (float *rhs*) `[inline]`

Unary multiply by scalar.

Vec4ub osg::Vec4ub::operator/ (float *rhs*) const `[inline]`

Divide by scalar.

Vec4ub& osg::Vec4ub::operator/= (float *rhs*) `[inline]`

Unary divide by scalar.

Vec4ub osg::Vec4ub::operator+ (const Vec4ub & *rhs*) const `[inline]`

Binary vector add.

Vec4ub& osg::Vec4ub::operator+= (const Vec4ub & *rhs*) `[inline]`

Unary vector add. Slightly more efficient because no temporary intermediate object.

Vec4ub osg::Vec4ub::operator- (const Vec4ub & *rhs*) const `[inline]`

Binary vector subtract.

Vec4ub& osg::Vec4ub::operator-= (const Vec4ub & *rhs*) `[inline]`

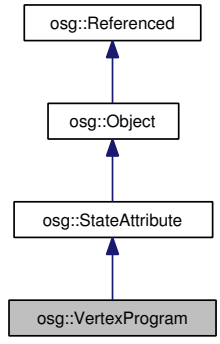
Unary vector subtract.

4.486 Member Data Documentation

value_type osg::Vec4ub::_v[4]

Vec member varaible.

4.487 osg::VertexProgram Class Reference



Public Types

- typedef std::map< GLuint, Vec4 > **LocalParamList**
- typedef std::map< GLenum, Matrix > **MatrixList**

Public Member Functions

- VertexProgram (const VertexProgram &vp, const CopyOp &copyop=CopyOp::SHALLOW_COPY)

- **META_StateAttribute** (osg, VertexProgram, VERTEXPROGRAM)
- virtual int compare (const osg::StateAttribute &sa) const
- virtual bool getModeUsage (StateAttribute::ModeUsage &usage) const
- GLuint & getVertexProgramID (unsigned int contextID) const
- void setVertexProgram (const char *program)
- void setVertexProgram (const std::string &program)
- const std::string & getVertexProgram () const
- void setProgramLocalParameter (const GLuint index, const Vec4 &p)
- void setLocalParameters (const LocalParamList &lpl)
- LocalParamList & getLocalParameters ()
- const LocalParamList & getLocalParameters () const
- void setMatrix (const GLenum mode, const Matrix &matrix)
- void setMatrices (const MatrixList &matrices)
- MatrixList & getMatrices ()
- const MatrixList & getMatrices () const
- void dirtyVertexProgramObject ()
- virtual void apply (State &state) const
- virtual void compileGLObjects (State &state) const
- virtual void resizeGLObjectBuffers (unsigned int maxSize)
- virtual void releaseGLObjects (State *state=0) const

Static Public Member Functions

- static void deleteVertexProgramObject (unsigned int contextID, GLuint handle)
- static void flushDeletedVertexProgramObjects (unsigned int contextID, double currentTime, double &availableTime)
- static Extensions * getExtensions (unsigned int contextID, bool createIfNotInitalized)
- static void setExtensions (unsigned int contextID, Extensions *extensions)

Classes

- class Extensions

4.488 Detailed Description

VertexProgram - encapsulates the OpenGL ARB vertex program state.

4.489 Constructor & Destructor Documentation

osg::VertexProgram::VertexProgram (const VertexProgram & *vp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`)

Copy constructor using CopyOp to manage deep vs shallow copy.

4.490 Member Function Documentation

virtual int osg::VertexProgram::compare (const osg::StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

virtual bool osg::VertexProgram::getModeUsage (StateAttribute::ModeUsage &) const `[inline, virtual]`

Return the modes associated with this StateAttribute.

Reimplemented from osg::StateAttribute.

GLuint& osg::VertexProgram::getVertexProgramID (unsigned int *contextID*) const `[inline]`

Get the handle to the vertex program ID for the current context.

void osg::VertexProgram::setVertexProgram (const char * *program*) `[inline]`

Set the vertex program using a C style string.

void osg::VertexProgram::setVertexProgram (const std::string & *program*) `[inline]`

Set the vertex program using C++ style string.

const std::string& osg::VertexProgram::getVertexProgram () const `[inline]`

Get the vertex program.

void osg::VertexProgram::setProgramLocalParameter (const GLuint *index*, const Vec4 & *p*) `[inline]`

Set Program Parameters

void osg::VertexProgram::setLocalParameters (const LocalParamList & *lpl*) `[inline]`

Set list of Program Parameters

LocalParamList& osg::VertexProgram::getLocalParameters () `[inline]`

Get list of Program Parameters

const LocalParamList& osg::VertexProgram::getLocalParameters () const `[inline]`

Get const list of Program Parameters

void osg::VertexProgram::setMatrix (const GLenum *mode*, const Matrix & *matrix*) `[inline]`

Matrix

void osg::VertexProgram::setMatrices (const MatrixList & *matrices*) `[inline]`

Set list of Matrices

MatrixList& osg::VertexProgram::getMatrices () `[inline]`

Get list of Matrices

const MatrixList& osg::VertexProgram::getMatrices () const `[inline]`

Get list of Matrices

void osg::VertexProgram::dirtyVertexProgramObject ()

Force a recompile on next apply() of associated OpenGL vertex program objects.

static void osg::VertexProgram::deleteVertexProgramObject (unsigned int *contextID*, GLuint *handle*) `[static]`

Use deleteVertexProgramObject instead of glDeletePrograms to allow OpenGL Vertex Program objects to cached until they can be deleted by the OpenGL context in which they were created, specified by contextID.

static void osg::VertexProgram::flushDeletedVertexProgramObjects (unsigned int *contextID*, double *currentTime*, double & *availableTime*) `[static]`

Flush all the cached vertex programs which need to be deleted in the OpenGL context related to contextID.

virtual void osg::VertexProgram::apply (State &) const `[virtual]`

apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

virtual void osg::VertexProgram::compileGLObjects (State &) const `[inline, virtual]`

Default to nothing to compile - all state is applied immediately.

Reimplemented from osg::StateAttribute.

virtual void osg::VertexProgram::resizeGLObjectBuffers (unsigned int *maxSize*) `[virtual]`

Resize any per context GLObject buffers to specified size.

Reimplemented from osg::StateAttribute.

virtual void osg::VertexProgram::releaseGLObjects (State * *state* = 0) const `[virtual]`

Release any OpenGL objects in specified graphics context if State object is passed, otherwise release OpenGL objexts for all graphics contexts if State object pointer is NULL.

Reimplemented from osg::StateAttribute.

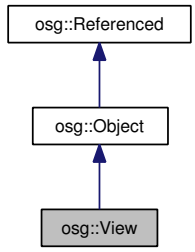
static Extensions* osg::VertexProgram::getExtensions (unsigned int *contextID*, bool *createIfNotInitalized*) `[static]`

Function to call to get the extension of a specified context. If the Exentsion object for that context has not yet been created and the 'createIfNotInitalized' flag been set to false then returns NULL. If 'createIfNotInitalized' is true then the Extensions object is automatically created. However, in this case the extension object will only be created with the graphics context associated with ContextID.

static void osg::VertexProgram::setExtensions (unsigned int *contextID*, Extensions * *extensions*) `[static]`

The setExtensions method allows users to override the extensions across graphics contexts. Typically used when you have different extensions supported across graphics pipes but need to ensure that they all use the same low common denominator extensions.

4.491 osg::View Class Reference



Public Types

- enum LightingMode {
  NO_LIGHT,
  HEADLIGHT,
  SKY_LIGHT }

Public Member Functions

- **View** (const osg::View &view, const osg::CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_Object** (osg, View)
- virtual void take (View &rhs)
- void setLightingMode (LightingMode lightingMode)
- LightingMode getLightingMode () const
- void setLight (osg::Light *light)
- osg::Light * getLight ()
- const osg::Light * getLight () const
- void setCamera (osg::Camera *camera)
- osg::Camera * getCamera ()
- const osg::Camera * getCamera () const
- bool **addSlave** (osg::Camera *camera, bool useMastersSceneData=true)
- bool **addSlave** (osg::Camera *camera, const osg::Matrix &projectionOffset, const osg::Matrix &viewOffse, bool useMastersSceneData=true)
- bool **removeSlave** (unsigned int pos)
- unsigned int **getNumSlaves** () const
- Slave & **getSlave** (unsigned int pos)
- const Slave & **getSlave** (unsigned int pos) const
- unsigned int **findSlaveIndexForCamera** (osg::Camera *camera)
- Slave * **findSlaveForCamera** (osg::Camera *camera)
- void **updateSlaves** ()
- void **updateSlave** (unsigned int i)

Classes

- struct Slave

4.492 Detailed Description

View - maintains a master camera view and a list of slave cameras that are relative to this master camera. Note, if no slave cameras are attached to the view then the master camera does both the control and implementation of the rendering of the scene, but if slave cameras are present then the master controls the view onto the scene, while the slaves implement the rendering of the scene.

4.493 Member Enumeration Documentation

enum osg::View::LightingMode

Options for controlling the global lighting used for the view.

4.494 Member Function Documentation

virtual void osg::View::take (View & *rhs*) `[virtual]`

Take all the settings, Camera and Slaves from the passed in view, leaving it empty.

void osg::View::setLightingMode (LightingMode *lightingMode*)

Set the global lighting to use for this view. Defaults to headlight.

LightingMode osg::View::getLightingMode () const `[inline]`

Get the global lighting used for this view.

void osg::View::setLight (osg::Light * *light*) `[inline]`

Get the global light.

osg::Light* osg::View::getLight () `[inline]`

Get the global lighting if assigned.

const osg::Light* osg::View::getLight () const `[inline]`

Get the const global lighting if assigned.

void osg::View::setCamera (osg::Camera * *camera*)

Set the master camera of the view.

osg::Camera* osg::View::getCamera () `[inline]`

Get the master camera of the view.

const osg::Camera* osg::View::getCamera () const `[inline]`

Get the const master camera of the view.

4.495 osg::View::Slave Struct Reference

Public Member Functions

- **Slave** (bool useMastersSceneData=true)
- **Slave** (osg::Camera *camera, const osg::Matrixd &projectionOffset, const osg::Matrixd &viewOffset, bool useMastersSceneData=true)
- **Slave** (const Slave &rhs)
- Slave & **operator=** (const Slave &rhs)

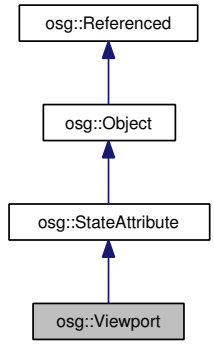
Public Attributes

- osg::ref_ptr< osg::Camera > **_camera**
- osg::Matrixd **_projectionOffset**
- osg::Matrixd **_viewOffset**
- bool **_useMastersSceneData**

4.496 Detailed Description

Slave allows one to up a camera that follows the master with a local offset to the project and view matrices.

4.497 osg::Viewport Class Reference



Public Types

- typedef double **value_type**

Public Member Functions

- **Viewport** (value_type x, value_type y, value_type width, value_type height)
- Viewport (const Viewport &vp, const CopyOp &copyop=CopyOp::SHALLOW_COPY)
- **META_StateAttribute** (osg, Viewport, VIEWPORT)
- virtual int compare (const StateAttribute &sa) const
- void **setViewport** (value_type x, value_type y, value_type width, value_type height)
- value_type & **x** ()
- value_type **x** () const
- value_type & **y** ()
- value_type **y** () const
- value_type & **width** ()
- value_type **width** () const
- value_type & **height** ()
- value_type **height** () const
- bool **valid** () const
- double aspectRatio () const
- const osg::Matrix computeWindowMatrix () const
- virtual void apply (State &state) const

4.498 Detailed Description

Encapsulate OpenGL glViewport.

4.499 Constructor & Destructor Documentation

osg::Viewport::Viewport (const Viewport & *vp*, const CopyOp & *copyop* = `CopyOp::SHALLOW_COPY`) `[inline]`

Copy constructor using CopyOp to manage deep vs shallow copy.

4.500 Member Function Documentation

virtual int osg::Viewport::compare (const StateAttribute & *sa*) const `[inline, virtual]`

Return -1 if *this < *rhs, 0 if *this==*rhs, 1 if *this>*rhs.

Implements osg::StateAttribute.

double osg::Viewport::aspectRatio () const `[inline]`

Return the aspectRatio of the viewport, which is equal to width/height. If height is zero, the potental division by zero is avoided by simply returning 1.0f.

const osg::Matrix osg::Viewport::computeWindowMatrix () const `[inline]`

Compute the Window Matrix which takes projected coords into Window coordinates. To convert local coordinates into window coordinates use v_window = v_local * MVPW matrix, where the MVPW matrix is ModelViewMatrix * ProjectionMatrix * WindowMatrix, the latter supplied by Viewport::computeWindowMatrix(), the ModelView and Projection Matrix can either be sourced from the current osg::State object, via osgUtil::SceneView or CullVisitor.

virtual void osg::Viewport::apply (State &) const `[virtual]`

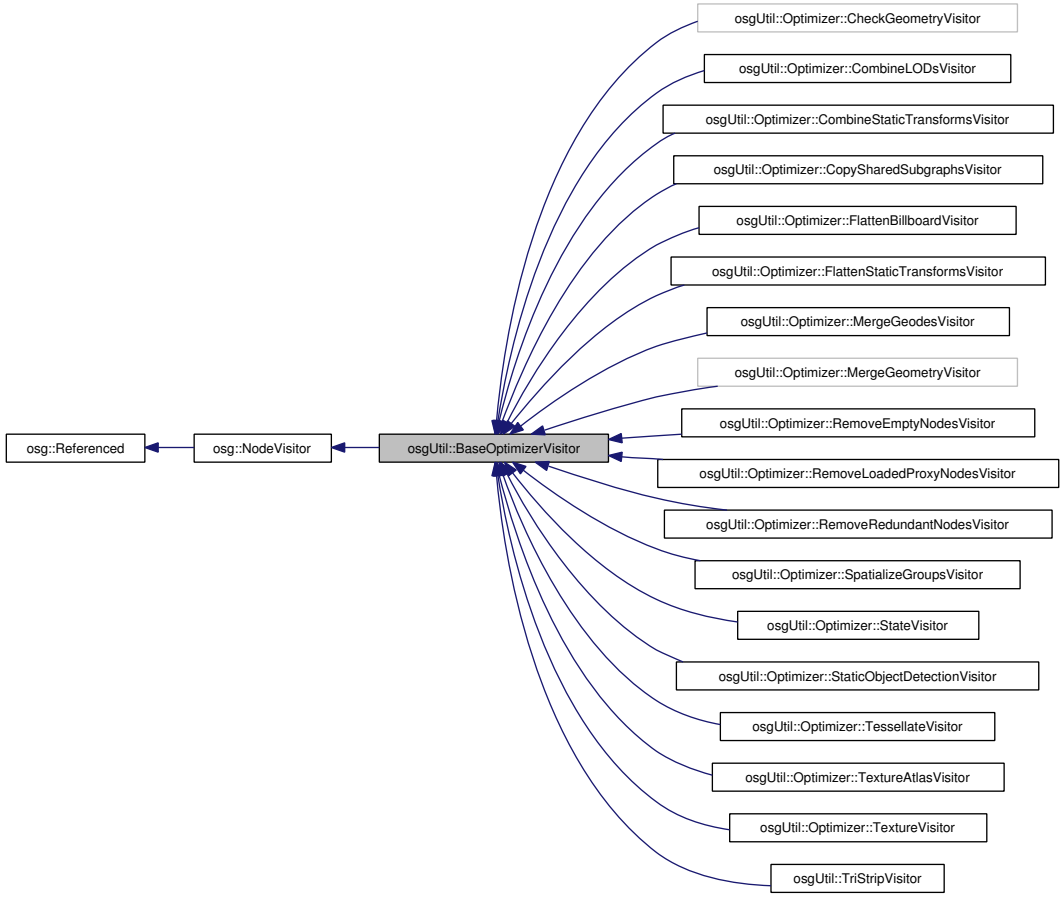
apply the OpenGL state attributes. The render info for the current OpenGL context is passed in to allow the StateAttribute to obtain details on the the current context and state.

Reimplemented from osg::StateAttribute.

Chapter 5

osgUtil Documentation

5.1 osgUtil::BaseOptimizerVisitor Class Reference



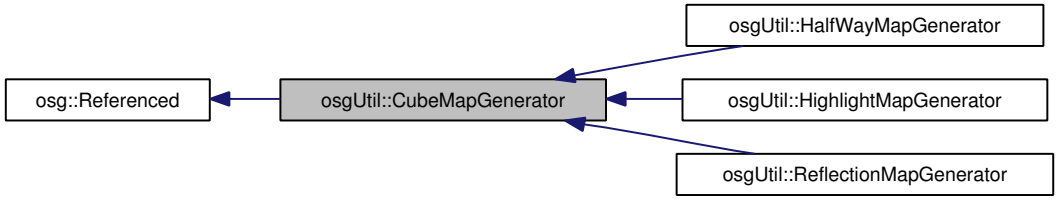
Public Member Functions

- **BaseOptimizerVisitor** (Optimizer *optimizer, unsigned int operation)
- bool **isOperationPermissibleForObject** (const osg::StateSet *object) const
- bool **isOperationPermissibleForObject** (const osg::StateAttribute *object) const
- bool **isOperationPermissibleForObject** (const osg::Drawable *object) const
- bool **isOperationPermissibleForObject** (const osg::Node *object) const

5.2 Detailed Description

Helper base class for implementing Optimizer techniques.

5.3 osgUtil::CubeMapGenerator Class Reference



Public Member Functions

- **CubeMapGenerator** (int texture_size=64)
- **CubeMapGenerator** (const CubeMapGenerator &copy, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- osg::Image * **getImage** (osg::TextureCubeMap::Face face)
- const osg::Image * **getImage** (osg::TextureCubeMap::Face face) const
- void generateMap (bool use_osg_system=true)

5.4 Detailed Description

This is the base class for cube map generators. It exposes the necessary interface to access the six generated images; descendants should only override the compute_color() method.

5.5 Member Function Documentation

void osgUtil::CubeMapGenerator::generateMap (bool *use_osg_system* = `true`)

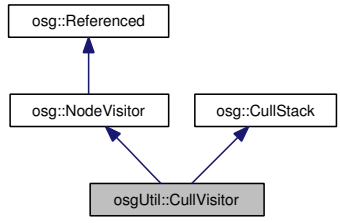
Generate the six cube images. If use_osg_system is true, then the OSG's coordinate system is used instead of the default OpenGL one.

virtual osg::Vec4 osgUtil::CubeMapGenerator::compute_color (const osg::Vec3 & *R*) const `[protected, pure virtual]`

Override this method to define how colors are computed. The parameter R is the reflection vector, pointing from the center of the cube. The return value should be the RGBA color associated with that reflection ray.

Implemented in osgUtil::HalfWayMapGenerator, osgUtil::HighlightMapGenerator, and osgUtil::ReflectionMapGenerator.

5.6 osgUtil::CullVisitor Class Reference



Public Types

- typedef osg::Matrix::value_type **value_type**

Public Member Functions

- CullVisitor (const CullVisitor &)

  Copy constructor that does a shallow copy.

- virtual CullVisitor * clone () const
- virtual void **reset** ()
- virtual osg::Vec3 getEyePoint () const
- virtual osg::Vec3 getViewPoint () const
- virtual float getDistanceToEyePoint (const osg::Vec3 &pos, bool withLODScale) const
- virtual float getDistanceFromEyePoint (const osg::Vec3 &pos, bool withLODScale) const

- virtual float getDistanceToViewPoint (const osg::Vec3 &pos, bool withLODScale) const
- virtual void **apply** (osg::Node &)
- virtual void **apply** (osg::Geode &node)
- virtual void **apply** (osg::Billboard &node)
- virtual void **apply** (osg::LightSource &node)
- virtual void **apply** (osg::ClipNode &node)
- virtual void **apply** (osg::TexGenNode &node)
- virtual void **apply** (osg::Group &node)
- virtual void **apply** (osg::Transform &node)
- virtual void **apply** (osg::Projection &node)
- virtual void **apply** (osg::Switch &node)
- virtual void **apply** (osg::LOD &node)
- virtual void **apply** (osg::ClearNode &node)
- virtual void **apply** (osg::Camera &node)
- virtual void **apply** (osg::OccluderNode &node)
- void **setClearNode** (const osg::ClearNode *earthSky)
- const osg::ClearNode * **getClearNode** () const
- void pushStateSet (const osg::StateSet *ss)
- void popStateSet ()
- void **setStateGraph** (StateGraph *rg)
- StateGraph * **getRootStateGraph** ()
- StateGraph * **getCurrentStateGraph** ()
- void **setRenderStage** (RenderStage *rg)
- RenderStage * **getRenderStage** ()
- RenderBin * **getCurrentRenderBin** ()
- void **setCurrentRenderBin** (RenderBin *rb)
- value_type **getCalculatedNearPlane** () const
- value_type **getCalculatedFarPlane** () const
- value_type **computeNearestPointInFrustum** (const osg::Matrix &matrix, const osg::Polytope::PlaneList &planes, const osg::Drawable &drawable)
- bool **updateCalculatedNearFar** (const osg::Matrix &matrix, const osg::BoundingBox &bb)
- bool **updateCalculatedNearFar** (const osg::Matrix &matrix, const osg::Drawable &drawable, bool isBillboard=false)
- void **updateCalculatedNearFar** (const osg::Vec3 &pos)
- void addDrawable (osg::Drawable *drawable, osg::RefMatrix *matrix)
- void addDrawableAndDepth (osg::Drawable *drawable, osg::RefMatrix *matrix, float depth)
- void addPositionedAttribute (osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- void addPositionedTextureAttribute (unsigned int textureUnit, osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- void computeNearPlane ()
- virtual void popProjectionMatrix ()
- virtual bool clampProjectionMatrixImplementation (osg::Matrixf &projection, double &znear, double &zfar) const
- virtual bool clampProjectionMatrixImplementation (osg::Matrixd &projection, double &znear, double &zfar) const
- bool clampProjectionMatrix (osg::Matrixf &projection, value_type &znear, value_type &zfar) const

- bool clampProjectionMatrix (osg::Matrixd &projection, value_type &znear, value_type &zfar) const
- void **setState** (osg::State *state)
- osg::State * **getState** ()
- const osg::State * **getState** () const
- void **setRenderInfo** (osg::RenderInfo &renderInfo)
- osg::RenderInfo & **getRenderInfo** ()
- const osg::RenderInfo & **getRenderInfo** () const

Static Public Member Functions

- static osg::ref_ptr< CullVisitor > & prototype ()
- static CullVisitor * create ()

Classes

- struct **MatrixPlanesDrawables**

5.7 Detailed Description

Basic NodeVisitor implementation for rendering a scene. This visitor traverses the scene graph, collecting transparent and opaque osg::Drawables into a depth sorted transparent bin and a state sorted opaque bin. The opaque bin is rendered first, and then the transparent bin is rendered in order from the furthest osg::Drawable from the eye to the one nearest the eye.

5.8 Member Function Documentation

virtual CullVisitor* osgUtil::CullVisitor::clone () const `[inline, virtual]`

Create a shallow copy of the CullVisitor, used by CullVisitor::create() to clone the prototype..

static osg::ref_ptr<CullVisitor>& osgUtil::CullVisitor::prototype () `[static]`

get the prototype singleton used by CullVisitor::create().

static CullVisitor* osgUtil::CullVisitor::create () `[static]`

create a CullVisitor by cloning CullVisitor::prototype().

virtual osg::Vec3 osgUtil::CullVisitor::getEyePoint () const `[inline, virtual]`

Get the eye point in local coordinates. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement.

Reimplemented from osg::NodeVisitor.

virtual osg::Vec3 osgUtil::CullVisitor::getViewPoint () const `[inline, virtual]`

Get the view point in local coordinates. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement.

Reimplemented from osg::NodeVisitor.

virtual float osgUtil::CullVisitor::getDistanceToEyePoint (const osg::Vec3 &, bool) const `[virtual]`

Get the distance from a point to the eye point, distance value in local coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceFromEyePoint(pos) is not implemented then a default value of 0.0 is returned.

Reimplemented from osg::NodeVisitor.

virtual float osgUtil::CullVisitor::getDistanceFromEyePoint (const osg::Vec3 &, bool) const `[virtual]`

Get the distance of a point from the eye point, distance value in the eye coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceFromEyePoint(pos) is not implemented than a default value of 0.0 is returned.

Reimplemented from osg::NodeVisitor.

virtual float osgUtil::CullVisitor::getDistanceToViewPoint (const osg::Vec3 &, bool) const `[virtual]`

Get the distance from a point to the view point, distance value in local coordinate system. Note, not all NodeVisitor implement this method, it is mainly cull visitors which will implement. If the getDistanceToViewPoint(pos) is not implemented then a default value of 0.0 is returned.

Reimplemented from osg::NodeVisitor.

void osgUtil::CullVisitor::pushStateSet (const osg::StateSet * *ss*) `[inline]`

Push state set on the current state group. If the state exists in a child state group of the current state group then move the current state group to that child. Otherwise, create a new state group for the state set, add it to the current state group then move the current state group pointer to the new state group.

void osgUtil::CullVisitor::popStateSet () `[inline]`

Pop the top state set and hence associated state group. Move the current state group to the parent of the popped state group.

void osgUtil::CullVisitor::addDrawable (osg::Drawable * *drawable*, osg::RefMatrix * *matrix*) `[inline]`

Add a drawable to current render graph.

void osgUtil::CullVisitor::addDrawableAndDepth (osg::Drawable * *drawable*, osg::RefMatrix * *matrix*, float *depth*) `[inline]`

Add a drawable and depth to current render graph.

void osgUtil::CullVisitor::addPositionedAttribute (osg::RefMatrix * *matrix*, const osg::StateAttribute * *attr*) `[inline]`

Add an attribute which is positioned relative to the modelview matrix.

void osgUtil::CullVisitor::addPositionedTextureAttribute (unsigned int *textureUnit*, osg::RefMatrix * *matrix*, const osg::StateAttribute * *attr*) `[inline]`

Add an attribute which is positioned relative to the modelview matrix.

void osgUtil::CullVisitor::computeNearPlane ()

compute near plane based on the polgon intersection of primtives in near plane candidate list of drawables. Note, you have to set ComputeNearFarMode to COMPUTE_NEAR_FAR_USING_PRIMITIVES to be able to near plane candidate drawables to be recorded by the cull traversal.

virtual void osgUtil::CullVisitor::popProjectionMatrix () `[virtual]`

Re-implement CullStack's popProjectionMatrix() adding clamping of the projection matrix to the computed near and far.

Reimplemented from osg::CullStack.

virtual bool osgUtil::CullVisitor::clampProjectionMatrixImplementation (osg::Matrixf & *projection*, double & *znear*, double & *zfar*) const `[virtual]`

CullVisitor's default clamping of the projection float matrix to computed near and far values. Note, do not call this method directly, use clampProjectionMatrix(..) instead, unless you want to bypass the callback.

virtual bool osgUtil::CullVisitor::clampProjectionMatrixImplementation (osg::Matrixd & *projection*, double & *znear*, double & *zfar*) const `[virtual]`

CullVisitor's default clamping of the projection double matrix to computed near and far values. Note, do not call this method directly, use clampProjectionMatrix(..) instead, unless you want to bypass the callback.

bool osgUtil::CullVisitor::clampProjectionMatrix (osg::Matrixf & *projection*, value_type & *znear*, value_type & *zfar*) const `[inline]`

Clamp the projection float matrix to computed near and far values, use callback if it exists, otherwise use default CullVisitor implementation.

bool osgUtil::CullVisitor::clampProjectionMatrix (osg::Matrixd & *projection*, value_type & *znear*, value_type & *zfar*) const `[inline]`

Clamp the projection double matrix to computed near and far values, use callback if it exists, otherwise use default CullVisitor implementation.

CullVisitor& osgUtil::CullVisitor::operator= (const CullVisitor &) `[inline, protected]`

Prevent unwanted copy operator.

5.9 osgUtil::DelaunayConstraint Class Reference

Public Member Functions

- void addtriangle (int i1, int i2, int i3)
- const osg::DrawElementsUInt * getTriangles () const
- osg::DrawElementsUInt * **getTriangles** ()
- osg::Vec3Array * getPoints (const osg::Vec3Array *points)
- osg::DrawElementsUInt * makeDrawable ()
- void merge (DelaunayConstraint *dco)
- void removeVerticesInside (const DelaunayConstraint *dco)
- float windingNumber (const osg::Vec3 &testpoint) const
- virtual bool contains (const osg::Vec3 &testpoint) const
- virtual bool **outside** (const osg::Vec3 &testpoint) const
- void handleOverlaps (void)

5.10 Detailed Description

DelaunayTriangulator: Utility class that triangulates an irregular network of sample points. Just create a DelaunayTriangulator, assign it the sample point array and call its triangulate() method to start the triangulation. Then you can obtain the generated primitive by calling the getTriangles() method.

Add DelaunayConstraints (or derived class) to control the triangulation edges.

5.11 Member Function Documentation

void osgUtil::DelaunayConstraint::addtriangle (int *i1*, int *i2*, int *i3*)

Each primitiveset is a list of vertices which may be closed by joining up to its start to make a loop. Constraints should be simple lines, not crossing themselves. Constraints which cross other constraints can cause difficulties - see the example for methods of dealing with them. collect up indices of triangle from delaunay triangles. The delaunay triangles inside the DelaunayConstraint area can be used to fill the area or generate geometry that terrain follows the area in some way. These triangles can form a canopy or a field.

const osg::DrawElementsUInt * osgUtil::DelaunayConstraint::getTriangles () const `[inline]`

Get the filling primitive. One: triangulate must have bneen called and two: triangle list is filled when DelaunayTriangulator::removeInternalTriangles is called. These return the triangles removed from the delaunay triangulation by DelaunayTriangulator::removeInternalTriangles.

osg::Vec3Array* osgUtil::DelaunayConstraint::getPoints (const osg::Vec3Array * *points*)

Call BEFORE makeDrawable to reorder points to make optimised set

osg::DrawElementsUInt* osgUtil::DelaunayConstraint::makeDrawable ()

converts simple list of triangles into a drawarray.

void osgUtil::DelaunayConstraint::merge (DelaunayConstraint * *dco*)

Add vertices and constraint loops from dco Can be used to generate extra vertices where dco crosses 'this' using osgUtil::Tessellator to insert overlap vertices.

void osgUtil::DelaunayConstraint::removeVerticesInside (const DelaunayConstraint * *dco*)

remove from line the vertices that are inside dco

float osgUtil::DelaunayConstraint::windingNumber (const osg::Vec3 & *testpoint*) const

return winding number as a float of loop around testpoint; may use multiple loops does not reject points on the edge or very very close to the edge

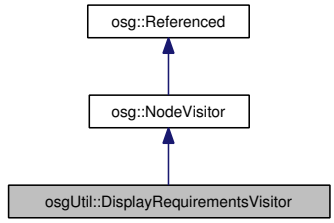
virtual bool osgUtil::DelaunayConstraint::contains (const osg::Vec3 & *testpoint*) const `[virtual]`

true if testpoint is internal (or external) to constraint.

void osgUtil::DelaunayConstraint::handleOverlaps (void)

Tessellate the constraint loops so that the crossing points are interpolated and added to the contraints for the triangulation.

5.12 osgUtil::DisplayRequirementsVisitor Class Reference



Public Member Functions

- DisplayRequirementsVisitor ()
- void setDisplaySettings (osg::DisplaySettings *ds)
- const osg::DisplaySettings * getDisplaySettings () const
- virtual void **applyStateSet** (osg::StateSet &stateset)
- virtual void **apply** (osg::Node &node)
- virtual void **apply** (osg::Geode &geode)

5.13 Detailed Description

A visitor for traversing a scene graph establishing which OpenGL visuals are required to support rendering of that scene graph. The results can then be used by applications to set up their windows with the correct visuals. Have a look at src/osgGLUT/Viewer.cpp's Viewer::open() method for an example of how to use it.

5.14 Constructor & Destructor Documentation

osgUtil::DisplayRequirementsVisitor::DisplayRequirementsVisitor ()

Default to traversing all children, and requiresDoubleBuffer, requiresRGB and requiresDepthBuffer to true and with alpha and stencil off.

5.15 Member Function Documentation

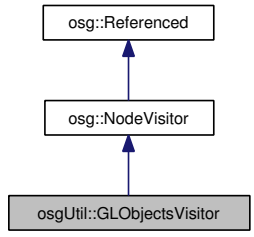
void osgUtil::DisplayRequirementsVisitor::setDisplaySettings (osg::DisplaySettings * *ds*) `[inline]`

Set the DisplaySettings.

const osg::DisplaySettings* osgUtil::DisplayRequirementsVisitor::getDisplaySettings () const `[inline]`

Get the DisplaySettings

5.16 osgUtil::GLObjectsVisitor Class Reference



Public Types

- enum ModeValues {

  SWITCH_ON_DISPLAY_LISTS,

  SWITCH_OFF_DISPLAY_LISTS,

  COMPILE_DISPLAY_LISTS,

  COMPILE_STATE_ATTRIBUTES,

  RELEASE_DISPLAY_LISTS,

  RELEASE_STATE_ATTRIBUTES,

  SWITCH_ON_VERTEX_BUFFER_OBJECTS,

  SWITCH_OFF_VERTEX_BUFFER_OBJECTS,

  CHECK_BLACK_LISTED_MODES }
- typedef unsigned int **Mode**

Public Member Functions

- GLObjectsVisitor (Mode mode=COMPILE_DISPLAY_LISTS|COMPILE_STATE_ATTRIBUTES|CHECK_BLACK_LISTED_MODES)
- virtual void reset ()
- void setMode (Mode mode)
- Mode getMode () const
- void setState (osg::State *state)
- osg::State * **getState** ()
- void **setRenderInfo** (osg::RenderInfo &renderInfo)
- osg::RenderInfo & **getRenderInfo** ()
- virtual void apply (osg::Node &node)
- virtual void apply (osg::Geode &node)
- void **apply** (osg::Drawable &drawable)
- void **apply** (osg::StateSet &stateset)

5.17 Detailed Description

Visitor for traversing scene graph and setting each osg::Drawable's _useDisplayList flag, with option to immediately compile osg::Drawable OpenGL Display lists and osg::StateAttribute's.

5.18 Member Enumeration Documentation

enum osgUtil::GLObjectsVisitor::ModeValues

Operation modes of the.

5.19 Constructor & Destructor Documentation

osgUtil::GLObjectsVisitor::GLObjectsVisitor (Mode *mode* = `COMPILE_DISPLAY_LISTS|COMPILE_STATE_ATTRIBUTES|CHECK_BLACK_LISTED_MODES`)

Construct a GLObjectsVisitor to traverse all children, operating on node according to specified mode, such as to compile or release display list/texture objects etc. Default mode is to compile GL objects.

5.20 Member Function Documentation

virtual void osgUtil::GLObjectsVisitor::reset () `[inline, virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented from osg::NodeVisitor.

void osgUtil::GLObjectsVisitor::setMode (Mode *mode*) `[inline]`

Set the operational mode of what operations to do on the scene graph.

Mode osgUtil::GLObjectsVisitor::getMode () const `[inline]`

Get the operational mode.

void osgUtil::GLObjectsVisitor::setState (osg::State * *state*) `[inline]`

Set the State to use during traversal.

virtual void osgUtil::GLObjectsVisitor::apply (osg::Node & *node*) `[virtual]`

Simply traverse using standard NodeVisitor traverse method.

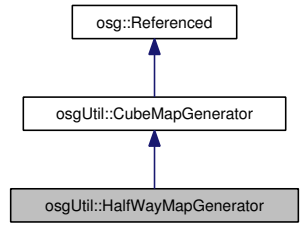
Reimplemented from osg::NodeVisitor.

virtual void osgUtil::GLObjectsVisitor::apply (osg::Geode & *node*) `[virtual]`

For each Geode visited set the display list usage according to the _displayListMode.

Reimplemented from osg::NodeVisitor.

5.21 osgUtil::HalfWayMapGenerator Class Reference



Public Member Functions

- **HalfWayMapGenerator** (const osg::Vec3 &light_direction, int texture_size=64)
- **HalfWayMapGenerator** (const HalfWayMapGenerator &copy, const osg::CopyOp &copyop)

5.22 Detailed Description

This cube map generator produces an Half-way vector map, useful for hardware-based specular lighting effects. It computes: C = normalize(R - L), where C is the resulting color, R is the reflection vector and L is the light direction.

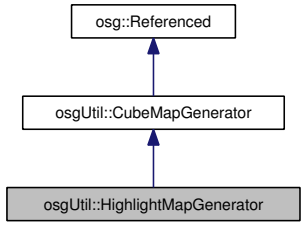
5.23 Member Function Documentation

osg::Vec4 osgUtil::HalfWayMapGenerator::compute_color (const osg::Vec3 & *R*) const `[inline, protected, virtual]`

Override this method to define how colors are computed. The parameter R is the reflection vector, pointing from the center of the cube. The return value should be the RGBA color associated with that reflection ray.

Implements osgUtil::CubeMapGenerator.

5.24 osgUtil::HighlightMapGenerator Class Reference



Public Member Functions

- **HighlightMapGenerator** (const osg::Vec3 &light_direction, const osg::Vec4 &light_color, float specular_exponent, int texture_size=64)
- **HighlightMapGenerator** (const HighlightMapGenerator &copy, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)

5.25 Detailed Description

This cube map generator produces a specular highlight map. The vector-color association is: C = (R dot (-L)) ^ n, where C is the resulting color, R is the reflection vector, L is the light direction and n is the specular exponent.

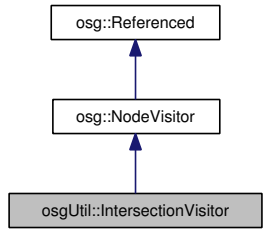
5.26 Member Function Documentation

osg::Vec4 osgUtil::HighlightMapGenerator::compute_color (const osg::Vec3 & *R*) const `[inline, protected, virtual]`

Override this method to define how colors are computed. The parameter R is the reflection vector, pointing from the center of the cube. The return value should be the RGBA color associated with that reflection ray.

Implements osgUtil::CubeMapGenerator.

5.27 osgUtil::IntersectionVisitor Class Reference



Public Member Functions

- **IntersectionVisitor** (Intersector *intersector=0, ReadCallback *readCallback=0)
- virtual void reset ()
- void setIntersector (Intersector *intersector)
- Intersector * getIntersector ()
- const Intersector * getIntersector () const
- void setReadCallback (ReadCallback *rc)
- ReadCallback * getReadCallback ()
- const ReadCallback * getReadCallback () const
- void **pushWindowMatrix** (osg::RefMatrix *matrix)
- void **pushWindowMatrix** (osg::Viewport *viewport)
- void **popWindowMatrix** ()
- osg::RefMatrix * **getWindowMatrix** ()
- const osg::RefMatrix * **getWindowMatrix** () const
- void **pushProjectionMatrix** (osg::RefMatrix *matrix)
- void **popProjectionMatrix** ()
- osg::RefMatrix * **getProjectionMatrix** ()
- const osg::RefMatrix * **getProjectionMatrix** () const
- void **pushViewMatrix** (osg::RefMatrix *matrix)
- void **popViewMatrix** ()
- osg::RefMatrix * **getViewMatrix** ()

- const osg::RefMatrix * **getViewMatrix** () const
- void **pushModelMatrix** (osg::RefMatrix *matrix)
- void **popModelMatrix** ()
- osg::RefMatrix * **getModelMatrix** ()
- const osg::RefMatrix * **getModelMatrix** () const
- virtual void **apply** (osg::Node &node)
- virtual void **apply** (osg::Geode &geode)
- virtual void **apply** (osg::Billboard &geode)
- virtual void **apply** (osg::Group &group)
- virtual void **apply** (osg::LOD &lod)
- virtual void **apply** (osg::PagedLOD &lod)
- virtual void **apply** (osg::Transform &transform)
- virtual void **apply** (osg::Projection &projection)
- virtual void **apply** (osg::Camera &camera)

Classes

- struct ReadCallback

5.28 Detailed Description

InteresectionVisitor is used to testing for intersections with the scene, traversing the scene using generic osgUtil::Intersector's to test against the scene. To implement different types of intersection techniques, one implements custom versions of the osgUtil::Intersector, and then pass the constructed intersector to the IntersectionVisitor.

5.29 Member Function Documentation

virtual void osgUtil::IntersectionVisitor::reset () `[virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented from osg::NodeVisitor.

void osgUtil::IntersectionVisitor::setIntersector (Intersector * *intersector*)

Set the intersector that will be used to intersect with the scene, and to store any hits that occur.

Intersector* osgUtil::IntersectionVisitor::getIntersector () `[inline]`

Get the intersector that will be used to intersect with the scene, and to store any hits that occur.

const Intersector* osgUtil::IntersectionVisitor::getIntersector () const `[inline]`

Get the const intersector that will be used to intersect with the scene, and to store any hits that occur.

void osgUtil::IntersectionVisitor::setReadCallback (ReadCallback * *rc*) `[inline]`

Set the read callback.

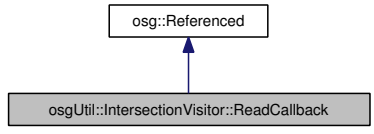
ReadCallback* osgUtil::IntersectionVisitor::getReadCallback () `[inline]`

Get the read callback.

const ReadCallback* osgUtil::IntersectionVisitor::getReadCallback () const `[inline]`

Get the const read callback.

5.30 osgUtil::IntersectionVisitor::ReadCallback Struct Reference



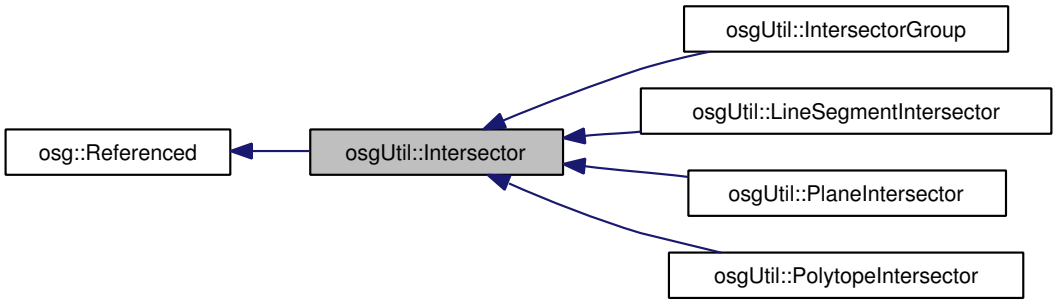
Public Member Functions

- virtual osg::Node * **readNodeFile** (const std::string &filename)=0

5.31 Detailed Description

Callback used to implement the reading of external files, allowing support for paged databases to be intergrated with IntersectionVisitor. A concrete implementation can be found in osgDB. Note, this loose coupling approach is required as osgUtil is independent from osgDB where the file reading is implemented, and osgDB itself is dependent upon osgUtil so a circular dependency would result from tighter integration.

5.32 osgUtil::Intersector Class Reference



Public Types

- enum **CoordinateFrame** {
  WINDOW,
  PROJECTION,
  VIEW,
  MODEL }

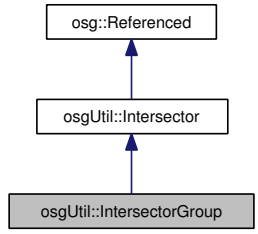
Public Member Functions

- **Intersector** (CoordinateFrame cf=MODEL)
- void **setCoordinateFrame** (CoordinateFrame cf)
- CoordinateFrame **getCoordinateFrame** () const
- virtual Intersector * **clone** (osgUtil::IntersectionVisitor &iv)=0
- virtual bool **enter** (const osg::Node &node)=0
- virtual void **leave** ()=0
- virtual void **intersect** (osgUtil::IntersectionVisitor &iv, osg::Drawable *drawable)=0
- virtual void **reset** ()
- virtual bool **containsIntersections** ()=0
- bool **disabled** () const
- void **incrementDisabledCount** ()
- void **decrementDisabledCount** ()

5.33 Detailed Description

Pure virtual base class for implementating custom intersection technique. To implement a specific intersection technique on must override all the pure virtue methods, concrete examples of how to do this can be seen in the LineSegmentIntersector.

5.34 osgUtil::IntersectorGroup Class Reference



Public Types

- typedef std::vector< osg::ref_ptr< Intersector > > **Intersectors**

Public Member Functions

- void addIntersector (Intersector *intersector)
- Intersectors & getIntersectors ()
- void clear ()
- virtual Intersector * **clone** (osgUtil::IntersectionVisitor &iv)
- virtual bool **enter** (const osg::Node &node)
- virtual void **leave** ()
- virtual void **intersect** (osgUtil::IntersectionVisitor &iv, osg::Drawable *drawable)
- virtual void **reset** ()
- virtual bool **containsIntersections** ()

5.35 Detailed Description

Concrent class for passing multiple intersectors through the scene graph. To be used in conjunction with IntersectionVisitor.

5.36 Member Function Documentation

void osgUtil::IntersectorGroup::addIntersector (Intersector * *intersector*)

Add an Intersector.

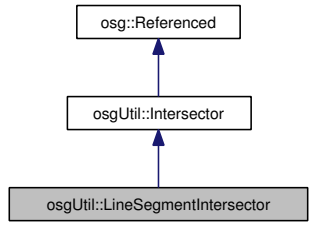
Intersectors& osgUtil::IntersectorGroup::getIntersectors () `[inline]`

Get the list of intersector.

void osgUtil::IntersectorGroup::clear ()

Clear the list of intersectors.

5.37 osgUtil::LineSegmentIntersector Class Reference



Public Types

- typedef std::multiset< Intersection > **Intersections**

Public Member Functions

- LineSegmentIntersector (const osg::Vec3d &start, const osg::Vec3d &end)
- LineSegmentIntersector (CoordinateFrame cf, const osg::Vec3d &start, const osg::Vec3d &end)
- LineSegmentIntersector (CoordinateFrame cf, double x, double y)
- void **insertIntersection** (const Intersection &intersection)
- Intersections & **getIntersections** ()
- Intersection **getFirstIntersection** ()
- void **setStart** (const osg::Vec3d &start)
- const osg::Vec3d & **getStart** () const
- void **setEnd** (const osg::Vec3d &end)
- const osg::Vec3d & **setEnd** () const
- virtual Intersector * **clone** (osgUtil::IntersectionVisitor &iv)
- virtual bool **enter** (const osg::Node &node)
- virtual void **leave** ()
- virtual void **intersect** (osgUtil::IntersectionVisitor &iv, osg::Drawable *drawable)
- virtual void **reset** ()
- virtual bool **containsIntersections** ()

Classes

- struct **Intersection**

5.38 Detailed Description

Concrent class for implementing line intersections with the scene graph. To be used in conjunction with IntersectionVisitor.

5.39 Constructor & Destructor Documentation

osgUtil::LineSegmentIntersector::LineSegmentIntersector (const osg::Vec3d & *start*, const osg::Vec3d & *end*)

Construct a LineSegmentIntersector the runs between the secified start and end points in MODEL coordinates.

osgUtil::LineSegmentIntersector::LineSegmentIntersector (CoordinateFrame *cf*, const osg::Vec3d & *start*, const osg::Vec3d & *end*)

Construct a LineSegmentIntersector the runs between the secified start and end points in the specified coordinate frame.

osgUtil::LineSegmentIntersector::LineSegmentIntersector (CoordinateFrame *cf*, double *x*, double *y*)

Convinience constructor for supporting picking in WINDOW, or PROJECTION coorindates In WINDOW coordinates creates a start value of (x,y,0) and end value of (x,y,1). In PROJECTION coordinates (clip space cube) creates a start value of (x,y,-1) and end value of (x,y,1). In VIEW and MODEL coordinates creates a start value of (x,y,0) and end value of (x,y,1).

5.40 osgUtil::Optimizer Class Reference

Public Types

- enum **OptimizationOptions** {
  FLATTEN_STATIC_TRANSFORMS,
  REMOVE_REDUNDANT_NODES,
  REMOVE_LOADED_PROXY_NODES,

COMBINE_ADJACENT_LODS,
SHARE_DUPLICATE_STATE,
MERGE_GEOMETRY,
CHECK_GEOMETRY,
SPATIALIZE_GROUPS,
COPY_SHARED_NODES,
TRISTRIP_GEOMETRY,
TESSELLATE_GEOMETRY,
OPTIMIZE_TEXTURE_SETTINGS,
MERGE_GEODES,
FLATTEN_BILLBOARDS,
TEXTURE_ATLAS_BUILDER,
STATIC_OBJECT_DETECTION,
DEFAULT_OPTIMIZATIONS,
ALL_OPTIMIZATIONS }

Public Member Functions

- void reset ()
- void optimize (osg::Node *node)
- virtual void optimize (osg::Node *node, unsigned int options)
- void setIsOperationPermissibleForObjectCallback (IsOperationPermissibleForObjectCallback *callback)
- IsOperationPermissibleForObjectCallback * getIsOperationPermissibleForObjectCallback ()
- const IsOperationPermissibleForObjectCallback * getIsOperationPermissibleForObjectCallback () const
- void **setPermissibleOptimizationsForObject** (const osg::Object *object, unsigned int options)
- unsigned int **getPermissibleOptimizationsForObject** (const osg::Object *object) const
- bool **isOperationPermissibleForObject** (const osg::StateSet *object, unsigned int option) const
- bool **isOperationPermissibleForObject** (const osg::StateAttribute *object, unsigned int option) const
- bool **isOperationPermissibleForObject** (const osg::Drawable *object, unsigned int option) const
- bool **isOperationPermissibleForObject** (const osg::Node *object, unsigned int option) const
- bool **isOperationPermissibleForObjectImplementation** (const osg::StateSet *stateset, unsigned int option) const
- bool **isOperationPermissibleForObjectImplementation** (const osg::StateAttribute *attribute, unsigned int option) const
- bool **isOperationPermissibleForObjectImplementation** (const osg::Drawable *drawable, unsigned int option) const
- bool **isOperationPermissibleForObjectImplementation** (const osg::Node *node, unsigned int option) const

Classes

- class **CheckGeometryVisitor**
- class CombineLODsVisitor
- class CombineStaticTransformsVisitor
- class CopySharedSubgraphsVisitor
- class FlattenBillboardVisitor
- class FlattenStaticTransformsVisitor
- struct IsOperationPermissibleForObjectCallback
- class MergeGeodesVisitor
- class **MergeGeometryVisitor**
- class RemoveEmptyNodesVisitor
- class RemoveLoadedProxyNodesVisitor
- class RemoveRedundantNodesVisitor
- class SpatializeGroupsVisitor
- class StateVisitor
- class StaticObjectDetectionVisitor
- class TessellateVisitor
- class TextureAtlasBuilder
- class TextureAtlasVisitor
- class TextureVisitor

5.41 Detailed Description

Traverses scene graph to improve efficiency. See OptimizationOptions. For example of usage see examples/osgimpostor or osgviewer.

5.42 Member Function Documentation

void osgUtil::Optimizer::reset ()

Reset internal data to initial state - the getPermissibleOptionsMap is cleared.

void osgUtil::Optimizer::optimize (osg::Node * *node*)

Traverse the node and its subgraph with a series of optimization visitors, specified by the OptimizationOptions.

virtual void osgUtil::Optimizer::optimize (osg::Node * *node*, unsigned int *options*) `[virtual]`

Traverse the node and its subgraph with a series of optimization visitors, specified by the OptimizationOptions.

void osgUtil::Optimizer::setIsOperationPermissibleForObjectCallback (IsOperationPermissibleForObjectCallback * *callback*) `[inline]`

Set the callback for customizing what operations are permitted on objects in the scene graph.

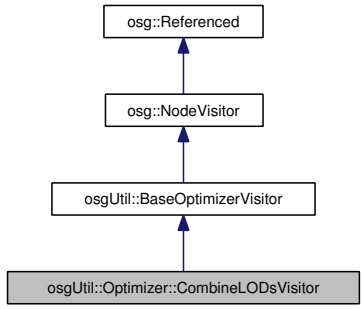
IsOperationPermissibleForObjectCallback* osgUtil::Optimizer::getIsOperationPermissibleForObjectCallback () `[inline]`

Get the callback for customizing what operations are permitted on objects in the scene graph.

const IsOperationPermissibleForObjectCallback* osgUtil::Optimizer::getIsOperationPermissibleForObjectCallback () const `[inline]`

Get the callback for customizing what operations are permitted on objects in the scene graph.

5.43 osgUtil::Optimizer::CombineLODsVisitor Class Reference



Public Types

- typedef std::set< osg::Group * > **GroupList**

Public Member Functions

- **CombineLODsVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::LOD &lod)
- void **combineLODs** ()

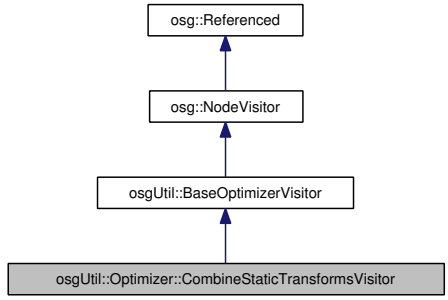
Public Attributes

- GroupList **_groupList**

5.44 Detailed Description

Optimize the LOD groups, by combining adjacent LOD's which have complementary ranges.

5.45 osgUtil::Optimizer::CombineStaticTransformsVisitor Class Reference



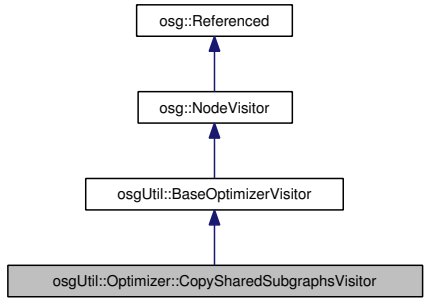
Public Member Functions

- **CombineStaticTransformsVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::MatrixTransform &transform)
- bool **removeTransforms** (osg::Node *nodeWeCannotRemove)

5.46 Detailed Description

Combine Static Transform nodes that sit above one another.

5.47 osgUtil::Optimizer::CopySharedSubgraphsVisitor Class Reference



Public Types

- typedef std::set< osg::Node * > **SharedNodeList**

Public Member Functions

- **CopySharedSubgraphsVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Node &node)
- void **copySharedNodes** ()

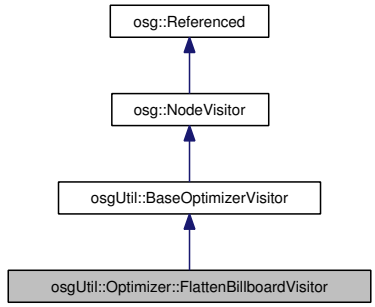
Public Attributes

- SharedNodeList **_sharedNodeList**

5.48 Detailed Description

Copy any shared subgraphs, enabling flattening of static transforms.

5.49 osgUtil::Optimizer::FlattenBillboardVisitor Class Reference



Public Types

- typedef std::vector< osg::NodePath > **NodePathList**
- typedef std::map< osg::Billboard *, NodePathList > **BillboardNodePathMap**

Public Member Functions

- **FlattenBillboardVisitor** (Optimizer *optimizer=0)
- virtual void reset ()
- virtual void **apply** (osg::Billboard &billboard)
- void **process** ()

Public Attributes

- BillboardNodePathMap **_billboards**

5.50 Detailed Description

Flatten MatrixTransform/Billboard pairs.

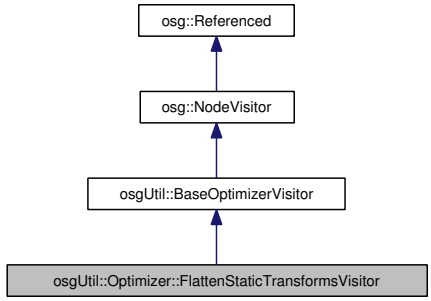
5.51 Member Function Documentation

virtual void osgUtil::Optimizer::FlattenBillboardVisitor::reset () `[virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented from osg::NodeVisitor.

5.52 osgUtil::Optimizer::FlattenStaticTransformsVisitor Class Reference



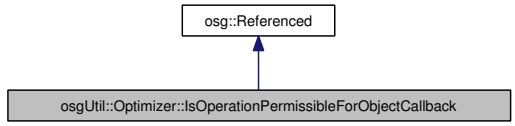
Public Member Functions

- **FlattenStaticTransformsVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Node &geode)
- virtual void **apply** (osg::Geode &geode)
- virtual void **apply** (osg::Billboard &geode)
- virtual void **apply** (osg::ProxyNode &node)
- virtual void **apply** (osg::PagedLOD &node)
- virtual void **apply** (osg::Transform &transform)
- bool **removeTransforms** (osg::Node *nodeWeCannotRemove)

5.53 Detailed Description

Flatten Static Transform nodes by applying their transform to the geometry on the leaves of the scene graph, then removing the now redundant transforms.

5.54 osgUtil::Optimizer::IsOperationPermissibleForObjectCallback Struct Reference



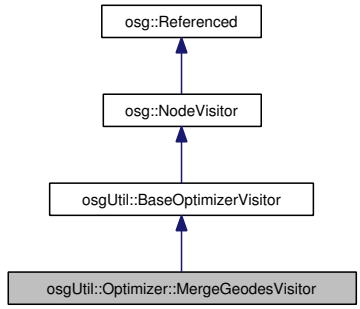
Public Member Functions

- virtual bool **isOperationPermissibleForObjectImplementation** (const Optimizer *optimizer, const osg::StateSet *stateset, unsigned int option) const
- virtual bool **isOperationPermissibleForObjectImplementation** (const Optimizer *optimizer, const osg::StateAttribute *attribute, unsigned int option) const
- virtual bool **isOperationPermissibleForObjectImplementation** (const Optimizer *optimizer, const osg::Drawable *drawable, unsigned int option) const
- virtual bool **isOperationPermissibleForObjectImplementation** (const Optimizer *optimizer, const osg::Node *node, unsigned int option) const

5.55 Detailed Description

Callback for customizing what operations are permitted on objects in the scene graph.

5.56 osgUtil::Optimizer::MergeGeodesVisitor Class Reference



Public Member Functions

- MergeGeodesVisitor (Optimizer *optimizer=0)

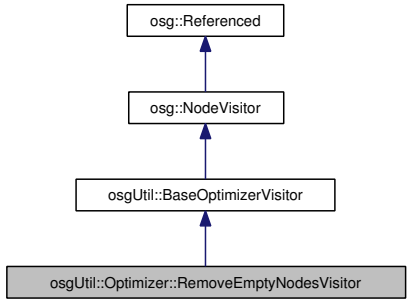
  default to traversing all children.

- virtual void **apply** (osg::Group &group)
- bool **mergeGeodes** (osg::Group &group)

5.57 Detailed Description

Combine geodes

5.58 osgUtil::Optimizer::RemoveEmptyNodesVisitor Class Reference



Public Types

- typedef std::set< osg::Node * > **NodeList**

Public Member Functions

- **RemoveEmptyNodesVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Geode &geode)
- virtual void **apply** (osg::Group &group)
- void **removeEmptyNodes** ()

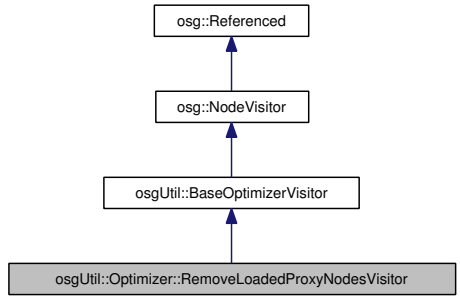
Public Attributes

- NodeList **_redundantNodeList**

5.59 Detailed Description

Remove rendundant nodes, such as groups with one single child.

5.60 osgUtil::Optimizer::RemoveLoadedProxyNodesVisitor Class Reference



Public Types

- typedef std::set< osg::Node * > **NodeList**

Public Member Functions

- **RemoveLoadedProxyNodesVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::ProxyNode &group)
- void **removeRedundantNodes** ()

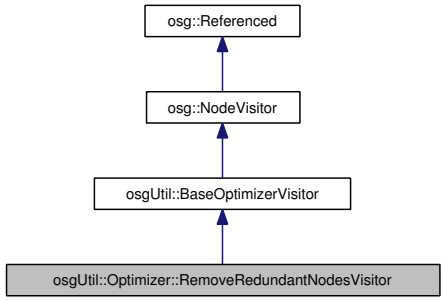
Public Attributes

- NodeList **_redundantNodeList**

5.61 Detailed Description

Remove loaded proxy nodes.

5.62 osgUtil::Optimizer::RemoveRedundantNodesVisitor Class Reference



Public Types

- typedef std::set< osg::Node * > **NodeList**

Public Member Functions

- **RemoveRedundantNodesVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Group &group)
- virtual void **apply** (osg::Transform &transform)
- bool **isOperationPermissible** (osg::Node &node)
- void **removeRedundantNodes** ()

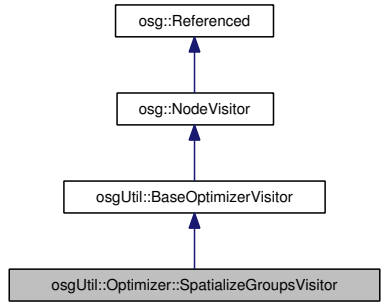
Public Attributes

- NodeList **_redundantNodeList**

5.63 Detailed Description

Remove redundant nodes, such as groups with one single child.

5.64 osgUtil::Optimizer::SpatializeGroupsVisitor Class Reference



Public Types

- typedef std::set< osg::Group * > **GroupsToDivideList**

Public Member Functions

- **SpatializeGroupsVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Group &group)
- bool **divide** (unsigned int maxNumTreesPerCell=8)
- bool **divide** (osg::Group *group, unsigned int maxNumTreesPerCell)

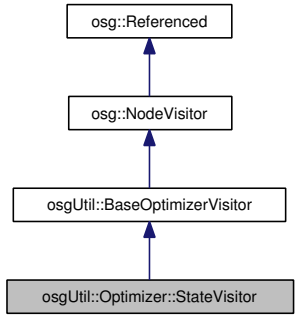
Public Attributes

- GroupsToDivideList **_groupsToDivideList**

5.65 Detailed Description

Spatialize scene into a balanced quad/oct tree.

5.66 osgUtil::Optimizer::StateVisitor Class Reference



Public Member Functions

- StateVisitor (Optimizer *optimizer=0)

  default to traversing all children.

- virtual void reset ()
- virtual void **apply** (osg::Node &node)
- virtual void **apply** (osg::Geode &geode)
- void **optimize** ()

5.67 Detailed Description

Optimize State in the scene graph by removing duplicate state, replacing it with shared instances, both for StateAttributes, and whole StateSets.

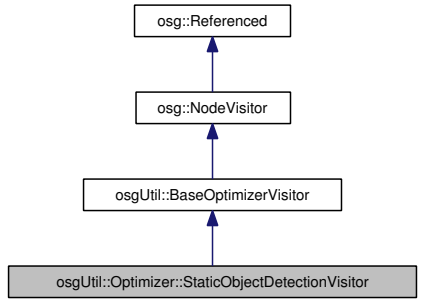
5.68 Member Function Documentation

virtual void osgUtil::Optimizer::StateVisitor::reset () `[virtual]`

empty visitor, make it ready for next traversal.

Reimplemented from osg::NodeVisitor.

5.69 osgUtil::Optimizer::StaticObjectDetectionVisitor Class Reference



Public Member Functions

- StaticObjectDetectionVisitor (Optimizer *optimizer=0)

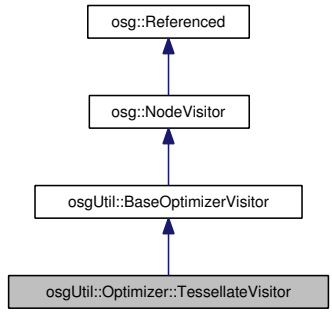
  default to traversing all children.

- virtual void **apply** (osg::Node &node)
- virtual void **apply** (osg::Geode &geode)

5.70 Detailed Description

Optimize the setting of StateSet and Geometry objects in scene so that they have a STATIC DataVariance when they don't have any callbacks associated with them.

5.71 osgUtil::Optimizer::TessellateVisitor Class Reference



Public Types

- typedef std::set< osg::Group * > **GroupList**

Public Member Functions

- **TessellateVisitor** (Optimizer *optimizer=0)
- virtual void **apply** (osg::Geode &geode)

Public Attributes

- GroupList **_groupList**

5.72 Detailed Description

Tessellate all geodes, to remove POLYGONS.

5.73 osgUtil::Optimizer::TextureAtlasBuilder Class Reference

Public Member Functions

- void **reset** ()
- void **setMaximumAtlasSize** (unsigned int width, unsigned int height)
- unsigned int **getMaximumAtlasWidth** () const
- unsigned int **getMaximumAtlasHeight** () const
- void **setMargin** (unsigned int margin)
- unsigned int **getMargin** () const
- void **addSource** (const osg::Image *image)
- void **addSource** (const osg::Texture2D *texture)
- unsigned int **getNumSources** () const
- const osg::Image * **getSourceImage** (unsigned int i)
- const osg::Texture2D * **getSourceTexture** (unsigned int i)
- void **buildAtlas** ()
- osg::Image * **getImageAtlas** (unsigned int i)
- osg::Texture2D * **getTextureAtlas** (unsigned int i)
- osg::Matrix **getTextureMatrix** (unsigned int i)
- osg::Image * **getImageAtlas** (const osg::Image *image)
- osg::Texture2D * **getTextureAtlas** (const osg::Image *image)

- osg::Matrix **getTextureMatrix** (const osg::Image *image)
- osg::Image * **getImageAtlas** (const osg::Texture2D *textue)
- osg::Texture2D * **getTextureAtlas** (const osg::Texture2D *texture)
- osg::Matrix **getTextureMatrix** (const osg::Texture2D *texture)

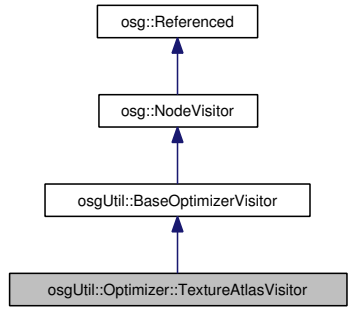
Classes

- class **Atlas**
- class **Source**

5.74 Detailed Description

Texture Atlas Builder creates a set of textures/images which each contain multiple images. Texture Atlas' are used to make it possible to use much wider batching of data.

5.75 osgUtil::Optimizer::TextureAtlasVisitor Class Reference



Public Member Functions

- TextureAtlasVisitor (Optimizer *optimizer=0)

  default to traversing all children.

- TextureAtlasBuilder & **getTextureAtlasBuilder** ()
- virtual void reset ()
- virtual void **apply** (osg::Node &node)
- virtual void **apply** (osg::Geode &geode)
- void **optimize** ()

5.76 Detailed Description

Optimize texture usage in the scene graph by combining textures into texture atlas Use of texture atlas cuts down on the number of seperate states in the scene, reducing state changes and improving the chances of use larger batches of geomertry.

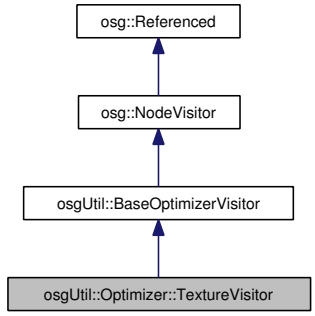
5.77 Member Function Documentation

virtual void osgUtil::Optimizer::TextureAtlasVisitor::reset () `[virtual]`

empty visitor, make it ready for next traversal.

Reimplemented from osg::NodeVisitor.

5.78 osgUtil::Optimizer::TextureVisitor Class Reference



Public Member Functions

- **TextureVisitor** (bool changeAutoUnRef, bool valueAutoUnRef, bool changeClientImageStorage, bool valueClientImageStorage, bool changeAnisotropy, float valueAnisotropy, Optimizer *optimizer=0)
- virtual void **apply** (osg::Geode &node)
- virtual void **apply** (osg::Node &node)
- void **apply** (osg::StateSet &stateset)
- void **apply** (osg::Texture &texture)

Public Attributes

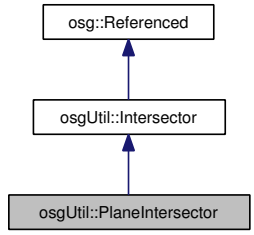
- bool **_changeAutoUnRef**

- bool **_valueAutoUnRef**
- bool **_changeClientImageStorage**
- bool **_valueClientImageStorage**
- bool **_changeAnisotropy**
- float **_valueAnisotropy**

5.79 Detailed Description

For all textures apply settings.

5.80 osgUtil::PlaneIntersector Class Reference



Public Types

- typedef std::vector< Intersection > **Intersections**

Public Member Functions

- PlaneIntersector (const osg::Plane &plane, const osg::Polytope &boundingPolytope=osg::Polytope())
- PlaneIntersector (CoordinateFrame cf, const osg::Plane &plane, const osg::Polytope &boundingPolytope=osg::Polytope())
- void **insertIntersection** (const Intersection &intersection)
- Intersections & **getIntersections** ()
- void **setRecordHeightsAsAttributes** (bool flag)
- bool **getRecordHeightsAsAttributes** () const
- void **setEllipsoidModel** (osg::EllipsoidModel *em)
- const osg::EllipsoidModel * **getEllipsoidModel** () const
- virtual Intersector * **clone** (osgUtil::IntersectionVisitor &iv)
- virtual bool **enter** (const osg::Node &node)
- virtual void **leave** ()

- virtual void **intersect** (osgUtil::IntersectionVisitor &iv, osg::Drawable *drawable)
- virtual void **reset** ()
- virtual bool **containsIntersections** ()

Classes

- struct **Intersection**

5.81 Detailed Description

Concrent class for implementing polytope intersections with the scene graph. To be used in conjunction with IntersectionVisitor.

5.82 Constructor & Destructor Documentation

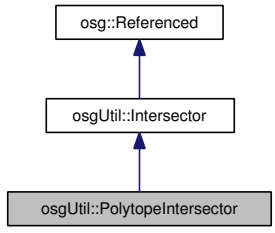
osgUtil::PlaneIntersector::PlaneIntersector (const osg::Plane & *plane*, const osg::Polytope & *boundingPolytope* = osg::Polytope ( ))

Construct a PolytopeIntersector using speified polytope in MODEL coordinates.

osgUtil::PlaneIntersector::PlaneIntersector (CoordinateFrame *cf*, const osg::Plane & *plane*, const osg::Polytope & *boundingPolytope* = osg::Polytope ( ))

Construct a PolytopeIntersector using speified polytope in specified coordinate frame.

5.83 osgUtil::PolytopeIntersector Class Reference



Public Types

- enum {

DimZero,

DimOne,

DimTwo,

AllDims }

dimension enum to specify primitive types to check.

- typedef std::set< Intersection > **Intersections**

Public Member Functions

- PolytopeIntersector (const osg::Polytope &polytope)
- PolytopeIntersector (CoordinateFrame cf, const osg::Polytope &polytope)
- PolytopeIntersector (CoordinateFrame cf, double xMin, double yMin, double xMax, double yMax)
- void **insertIntersection** (const Intersection &intersection)
- Intersections & **getIntersections** ()
- Intersection **getFirstIntersection** ()
- unsigned int **getDimensionMask** () const
- void setDimensionMask (unsigned int dimensionMask)
- const osg::Plane & **getReferencePlane** () const
- void setReferencePlane (const osg::Plane &plane)
- virtual Intersector * **clone** (osgUtil::IntersectionVisitor &iv)
- virtual bool **enter** (const osg::Node &node)
- virtual void **leave** ()
- virtual void **intersect** (osgUtil::IntersectionVisitor &iv, osg::Drawable *drawable)
- virtual void **reset** ()
- virtual bool **containsIntersections** ()

Classes

- struct **Intersection**

5.84 Detailed Description

Concrent class for implementing polytope intersections with the scene graph. To be used in conjunction with IntersectionVisitor.

5.85 Member Enumeration Documentation

anonymous enum

dimension enum to specify primitive types to check.

Enumerator:

DimZero check for points

DimOne check for lines

DimTwo check for triangles, quad

5.86 Constructor & Destructor Documentation

osgUtil::PolytopeIntersector::PolytopeIntersector (const osg::Polytope & *polytope*)

Construct a PolytopeIntersector using specified polytope in MODEL coordinates.

osgUtil::PolytopeIntersector::PolytopeIntersector (CoordinateFrame *cf*, const osg::Polytope & *polytope*)

Construct a PolytopeIntersector using specified polytope in specified coordinate frame.

osgUtil::PolytopeIntersector::PolytopeIntersector (CoordinateFrame *cf*, double *xMin*, double *yMin*, double *xMax*, double *yMax*)

Convenience constructor for supporting picking in WINDOW, or PROJECTION coordinates In WINDOW coordinates (clip space cube) creates a five sided polytope box that has a front face at 0.0 and sides around box xMin, yMin, xMax, yMax. In PROJECTION coordinates (clip space cube) creates a five sided polytope box that has a front face at -1 and sides around box xMin, yMin, xMax, yMax. In VIEW and MODEL coordinates (clip space cube) creates a five sided polytope box that has a front face at 0.0 and sides around box xMin, yMin, xMax, yMax.

5.87 Member Function Documentation

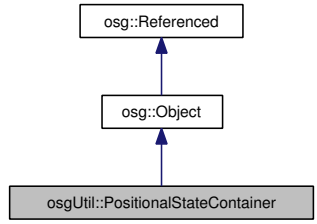
void osgUtil::PolytopeIntersector::setDimensionMask (unsigned int *dimensionMask*) `[inline]`

set the dimension mask. As polytope-triangle and polytope-quad intersections are expensive to compute it is possible to turn them off by calling setDimensionMask(DimZero | DimOne)

void osgUtil::PolytopeIntersector::setReferencePlane (const osg::Plane & *plane*) `[inline]`

set the plane used to sort the intersections. The intersections are sorted by the distance of the localIntersectionPoint and the reference plane. The default for the reference plane is the last plane of the polytope.

5.88 osgUtil::PositionalStateContainer Class Reference



Public Types

- typedef std::pair< osg::ref_ptr< const osg::StateAttribute >, osg::ref_ptr< osg::RefMatrix > > **AttrMatrixPair**
- typedef std::vector< AttrMatrixPair > **AttrMatrixList**
- typedef std::map< unsigned int, AttrMatrixList > **TexUnitAttrMatrixListMap**

Public Member Functions

- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &) const
- virtual bool **isSameKindAs** (const osg::Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const
- virtual void **reset** ()
- AttrMatrixList & **getAttrMatrixList** ()
- virtual void **addPositionedAttribute** (osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- TexUnitAttrMatrixListMap & **getTexUnitAttrMatrixListMap** ()
- virtual void **addPositionedTextureAttribute** (unsigned int textureUnit, osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- virtual void **draw** (osg::State &state, RenderLeaf *&previous, const osg::Matrix *postMultMatrix=0)

Public Attributes

- AttrMatrixList **_attrList**
- TexUnitAttrMatrixListMap **_texAttrListMap**

5.89 Detailed Description

PositionalStateContainer base class. Used in RenderStage class.

5.90 Member Function Documentation

virtual osg::Object* osgUtil::PositionalStateContainer::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual osg::Object* osgUtil::PositionalStateContainer::clone (const osg::CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Implements osg::Object.

virtual const char* osgUtil::PositionalStateContainer::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

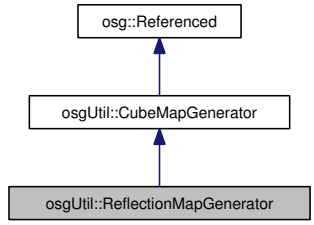
Implements osg::Object.

virtual const char* osgUtil::PositionalStateContainer::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

5.91 osgUtil::ReflectionMapGenerator Class Reference



Public Member Functions

- **ReflectionMapGenerator** (int texture_size=64)
- **ReflectionMapGenerator** (const ReflectionMapGenerator &copy, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)

5.92 Detailed Description

This is the most simple cube map generator. It performs a direct association between reflection vector and RGBA color (C = R).

5.93 Member Function Documentation

osg::Vec4 osgUtil::ReflectionMapGenerator::compute_color (const osg::Vec3 & *R*) const `[inline, protected, virtual]`

Override this method to define how colors are computed. The parameter R is the reflection vector, pointing from the center of the cube. The return value should be the RGBA color associated with that reflection ray.

Implements osgUtil::CubeMapGenerator.

5.94 osgUtil::RegisterRenderBinProxy Class Reference

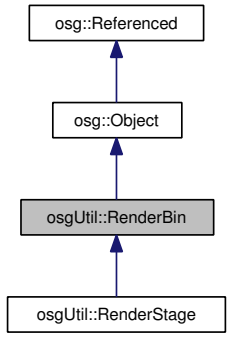
Public Member Functions

- **RegisterRenderBinProxy** (const std::string &binName, RenderBin *proto)

5.95 Detailed Description

Proxy class for automatic registration of renderbins with the RenderBin prototypelist.

5.96 osgUtil::RenderBin Class Reference



Public Types

- enum **SortMode** {
  SORT_BY_STATE,
  SORT_BY_STATE_THEN_FRONT_TO_BACK,
  SORT_FRONT_TO_BACK,
  SORT_BACK_TO_FRONT }
- typedef std::vector< RenderLeaf * > **RenderLeafList**
- typedef std::vector< StateGraph * > **StateGraphList**
- typedef std::map< int, osg::ref_ptr< RenderBin > > **RenderBinList**

Public Member Functions

- **RenderBin** (SortMode mode)
- RenderBin (const RenderBin &rhs, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_-COPY)
- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &copyop) const
- virtual bool **isSameKindAs** (const osg::Object *obj) const
- virtual const char * libraryName () const
- virtual const char * className () const

- virtual void **reset** ()
- void **setStateSet** (osg::StateSet *stateset)
- osg::StateSet * **getStateSet** ()
- const osg::StateSet * **getStateSet** () const
- RenderBin * **getParent** ()
- const RenderBin * **getParent** () const
- RenderStage * **getStage** ()
- const RenderStage * **getStage** () const
- int **getBinNum** () const
- StateGraphList & **getStateGraphList** ()
- const StateGraphList & **getStateGraphList** () const
- RenderBinList & **getRenderBinList** ()
- const RenderBinList & **getRenderBinList** () const
- RenderLeafList & **getRenderLeafList** ()
- const RenderLeafList & **getRenderLeafList** () const
- RenderBin * **find_or_insert** (int binNum, const std::string &binName)
- void **addStateGraph** (StateGraph *rg)
- virtual void **sort** ()
- virtual void **sortImplementation** ()
- void **setSortMode** (SortMode mode)
- SortMode **getSortMode** () const
- virtual void **sortByState** ()
- virtual void **sortByStateThenFrontToBack** ()
- virtual void **sortFrontToBack** ()
- virtual void **sortBackToFront** ()
- void **setSortCallback** (SortCallback *sortCallback)
- SortCallback * **getSortCallback** ()
- const SortCallback * **getSortCallback** () const
- virtual void **draw** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- virtual void **drawImplementation** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- void **setDrawCallback** (DrawCallback *drawCallback)
- DrawCallback * **getDrawCallback** ()
- const DrawCallback * **getDrawCallback** () const
- bool getStats (Statistics &primStats) const
- virtual unsigned int computeNumberOfDynamicRenderLeaves () const
- void **copyLeavesFromStateGraphListToRenderLeafList** ()

Static Public Member Functions

- static RenderBin * **createRenderBin** (const std::string &binName)
- static RenderBin * **getRenderBinPrototype** (const std::string &binName)
- static void **addRenderBinPrototype** (const std::string &binName, RenderBin *proto)
- static void **removeRenderBinPrototype** (RenderBin *proto)
- static void **setDefaultRenderBinSortMode** (SortMode mode)
- static SortMode **getDefaultRenderBinSortMode** ()

Classes

- struct **DrawCallback**
- struct **SortCallback**

5.97 Detailed Description

RenderBin base class. Renderbin contains geometries to be rendered as a group, renderbins are rendered once each. They can improve efficiency or use different rendering algorithms. A renderBin can contain further renderBins producing a tree hierarchy of renderBins.

5.98 Constructor & Destructor Documentation

osgUtil::RenderBin::RenderBin (const RenderBin & *rhs*, const osg::CopyOp & *copyop* = `osg::CopyOp::SHALLOW_COPY`**)**

Copy constructor using CopyOp to manage deep vs shallow copy.

5.99 Member Function Documentation

virtual osg::Object∗ osgUtil::RenderBin::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object∗ return type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osgUtil::RenderStage.

virtual osg::Object∗ osgUtil::RenderBin::clone (const osg::CopyOp &) const `[inline, virtual]`

Clone an object, with Object∗ return type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osgUtil::RenderStage.

virtual const char∗ osgUtil::RenderBin::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

virtual const char* osgUtil::RenderBin::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

Reimplemented in osgUtil::RenderStage.

bool osgUtil::RenderBin::getStats (Statistics & *primStats*) const

Extract stats for current draw list.

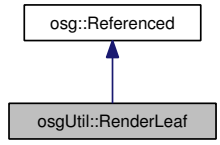
Reimplemented in osgUtil::RenderStage.

virtual unsigned int osgUtil::RenderBin::computeNumberOfDynamicRenderLeaves () const `[virtual]`

Compute the number of dynamic RenderLeaves.

Reimplemented in osgUtil::RenderStage.

5.100 osgUtil::RenderLeaf Class Reference



Public Member Functions

- **RenderLeaf** (osg::Drawable *drawable, osg::RefMatrix *projection, osg::RefMatrix *modelview, float depth=0.0f)
- void **set** (osg::Drawable *drawable, osg::RefMatrix *projection, osg::RefMatrix *modelview, float depth=0.0f)
- void **reset** ()
- virtual void **render** (osg::RenderInfo &renderInfo, RenderLeaf *previous)
- const osg::Drawable * **getDrawable** () const

Public Attributes

- StateGraph * **_parent**

- osg::ref_ptr< osg::Drawable > **_drawable**
- osg::ref_ptr< osg::RefMatrix > **_projection**
- osg::ref_ptr< osg::RefMatrix > **_modelview**
- float **_depth**
- bool **_dynamic**

Friends

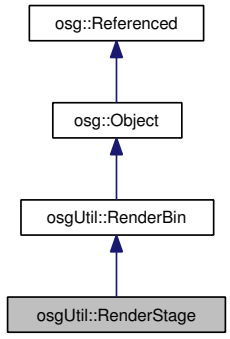
- class osgUtil::StateGraph

  Allow StateGraph to change the RenderLeaf's _parent.

5.101 Detailed Description

Container class for all data required for rendering of drawables.

5.102 osgUtil::RenderStage Class Reference



Public Member Functions

- **RenderStage** (SortMode mode)
- **RenderStage** (const RenderStage &rhs, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- virtual osg::Object * cloneType () const
- virtual osg::Object * clone (const osg::CopyOp &copyop) const
- virtual bool **isSameKindAs** (const osg::Object *obj) const
- virtual const char * className () const

- virtual void **reset** ()
- void setDrawBuffer (GLenum buffer)
- GLenum getDrawBuffer () const
- void setReadBuffer (GLenum buffer)
- GLenum getReadBuffer () const
- void setViewport (osg::Viewport *viewport)
- const osg::Viewport * getViewport () const
- osg::Viewport * getViewport ()
- void setClearMask (GLbitfield mask)
- GLbitfield getClearMask () const
- void **setColorMask** (osg::ColorMask *cm)
- osg::ColorMask * **getColorMask** ()
- const osg::ColorMask * **getColorMask** () const
- void setClearColor (const osg::Vec4 &color)
- const osg::Vec4 & getClearColor () const
- void setClearAccum (const osg::Vec4 &color)
- const osg::Vec4 & getClearAccum () const
- void setClearDepth (double depth)
- double getClearDepth () const
- void setClearStencil (int stencil)
- int getClearStencil () const
- void **setCamera** (osg::Camera *camera)
- osg::Camera * **getCamera** ()
- const osg::Camera * **getCamera** () const
- void **setCameraRequiresSetUp** (bool flag)
- bool **getCameraRequiresSetUp** () const
- void runCameraSetUp (osg::RenderInfo &renderInfo)
- void **setTexture** (osg::Texture *texture, unsigned int level=0, unsigned int face=0)
- osg::Texture * **getTexture** ()
- void **setImage** (osg::Image *image)
- osg::Image * **getImage** ()
- void **setImageReadPixelFormat** (GLenum format)
- GLenum **getImageReadPixelFormat** () const
- void **setImageReadPixelDataType** (GLenum type)
- GLenum **getImageReadPixelDataType** () const
- void **setFrameBufferObject** (osg::FrameBufferObject *fbo)
- osg::FrameBufferObject * **getFrameBufferObject** ()
- const osg::FrameBufferObject * **getFrameBufferObject** () const
- void **setGraphicsContext** (osg::GraphicsContext *context)
- osg::GraphicsContext * **getGraphicsContext** ()
- const osg::GraphicsContext * **getGraphicsContext** () const
- void **setInheritedPositionalStateContainerMatrix** (const osg::Matrix &matrix)
- const osg::Matrix & **getInheritedPositionalStateContainerMatrix** () const
- void **setInheritedPositionalStateContainer** (PositionalStateContainer *rsl)
- PositionalStateContainer * **getInheritedPositionalStateContainer** ()
- void **setPositionalStateContainer** (PositionalStateContainer *rsl)

- PositionalStateContainer * **getPositionalStateContainer** () const
- virtual void **addPositionedAttribute** (osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- virtual void **addPositionedTextureAttribute** (unsigned int textureUnit, osg::RefMatrix *matrix, const osg::StateAttribute *attr)
- void **copyTexture** (osg::RenderInfo &renderInfo)
- virtual void **sort** ()
- virtual void **drawPreRenderStages** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- virtual void **draw** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- virtual void **drawInner** (osg::RenderInfo &renderInfo, RenderLeaf *&previous, bool &doCopyTexture)
- virtual void **drawPostRenderStages** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- virtual void **drawImplementation** (osg::RenderInfo &renderInfo, RenderLeaf *&previous)
- void **addToDependencyList** (RenderStage *rs)
- void **addPreRenderStage** (RenderStage *rs, int order=0)
- void **addPostRenderStage** (RenderStage *rs, int order=0)
- bool getStats (Statistics &stats) const
- virtual unsigned int computeNumberOfDynamicRenderLeaves () const
- void **attach** (osg::Camera::BufferComponent buffer, osg::Image *image)

Classes

- struct **Attachment**

5.103 Detailed Description

RenderStage base class. Used for encapsulate a complete stage in rendering - setting up of viewport, the projection and model matrices and rendering the RenderBin's enclosed with this RenderStage. RenderStage also has a dependency list of other RenderStages, each of which must be called before the rendering of this stage. These 'pre' rendering stages are used for advanced rendering techniques like multistage pixel shading or impostors.

5.104 Member Function Documentation

virtual osg::Object* osgUtil::RenderStage::cloneType () const `[inline, virtual]`

Clone the type of an object, with Object* return type. Must be defined by derived classes.

Reimplemented from osgUtil::RenderBin.

virtual osg::Object* osgUtil::RenderStage::clone (const osg::CopyOp &) const `[inline, virtual]`

Clone an object, with Object* return type. Must be defined by derived classes.

Reimplemented from osgUtil::RenderBin.

virtual const char* osgUtil::RenderStage::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Reimplemented from osgUtil::RenderBin.

void osgUtil::RenderStage::setDrawBuffer (GLenum *buffer*) `[inline]`

Set the draw buffer used at the start of each frame draw.

GLenum osgUtil::RenderStage::getDrawBuffer () const `[inline]`

Get the draw buffer used at the start of each frame draw.

void osgUtil::RenderStage::setReadBuffer (GLenum *buffer*) `[inline]`

Set the read buffer for any required copy operations to use.

GLenum osgUtil::RenderStage::getReadBuffer () const `[inline]`

Get the read buffer for any required copy operations to use.

void osgUtil::RenderStage::setViewport (osg::Viewport * *viewport*) `[inline]`

Set the viewport.

const osg::Viewport* osgUtil::RenderStage::getViewport () const `[inline]`

Get the const viewport.

osg::Viewport* osgUtil::RenderStage::getViewport () `[inline]`

Get the viewport.

void osgUtil::RenderStage::setClearMask (GLbitfield *mask*) `[inline]`

Set the clear mask used in glClear(..). Defaults to GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_-BIT.

GLbitfield osgUtil::RenderStage::getClearMask () const `[inline]`

Get the clear mask.

void osgUtil::RenderStage::setClearColor (const osg::Vec4 & *color*) `[inline]`

Set the clear color used in glClearColor(..). glClearColor is only called if mask & GL_COLOR_BUFFER_-BIT is true

const osg::Vec4& osgUtil::RenderStage::getClearColor () const `[inline]`

Get the clear color.

void osgUtil::RenderStage::setClearAccum (const osg::Vec4 & *color*) `[inline]`

Set the clear accum used in glClearAccum(..). glClearAcumm is only called if mask & GL_ACCUM_-BUFFER_BIT is true.

const osg::Vec4& osgUtil::RenderStage::getClearAccum () const `[inline]`

Get the clear accum.

void osgUtil::RenderStage::setClearDepth (double *depth*) `[inline]`

Set the clear depth used in glClearDepth(..). Defaults to 1.0 glClearDepth is only called if mask & GL_-DEPTH_BUFFER_BIT is true.

double osgUtil::RenderStage::getClearDepth () const `[inline]`

Get the clear depth.

void osgUtil::RenderStage::setClearStencil (int *stencil*) `[inline]`

Set the clear stencil value used in glClearStencil(). Defaults to 0; glClearStencil is only called if mask & GL_STENCIL_BUFFER_BIT is true

int osgUtil::RenderStage::getClearStencil () const `[inline]`

Get the clear color.

void osgUtil::RenderStage::runCameraSetUp (osg::RenderInfo & *renderInfo*)

Attempt the set the RenderStage from the Camera settings.

bool osgUtil::RenderStage::getStats (Statistics & *stats*) const

Extract stats for current draw list.

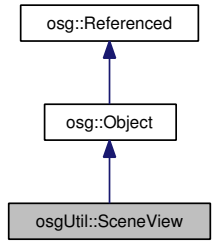
Reimplemented from osgUtil::RenderBin.

virtual unsigned int osgUtil::RenderStage::computeNumberOfDynamicRenderLeaves () const `[virtual]`

Compute the number of dynamic RenderLeaves.

Reimplemented from osgUtil::RenderBin.

5.105 osgUtil::SceneView Class Reference



Public Types

- enum **Options** {
  NO_SCENEVIEW_LIGHT,
  HEADLIGHT,
  SKY_LIGHT,
  COMPILE_GLOBJECTS_AT_INIT,
  STANDARD_SETTINGS }
- enum **ActiveUniforms** {
  FRAME_NUMBER_UNIFORM,
  FRAME_TIME_UNIFORM,
  DELTA_FRAME_TIME_UNIFORM,
  SIMULATION_TIME_UNIFORM,
  DELTA_SIMULATION_TIME_UNIFORM,
  VIEW_MATRIX_UNIFORM,
  VIEW_MATRIX_INVERSE_UNIFORM,

DEFAULT_UNIFORMS,

ALL_UNIFORMS }

- enum FusionDistanceMode {

  USE_FUSION_DISTANCE_VALUE,

  PROPORTIONAL_TO_SCREEN_DISTANCE }
- typedef Options **LightingMode**

Public Member Functions

- SceneView (osg::DisplaySettings *ds=NULL)
- **SceneView** (const SceneView &sceneview, const osg::CopyOp &copyop=osg::CopyOp())
- **META_Object** (osgUtil, SceneView)
- virtual void **setDefaults** ()
- virtual void setDefaults (unsigned int options)
- void setCamera (osg::Camera *camera, bool assumeOwnershipOfCamera=true)
- osg::Camera * getCamera ()
- const osg::Camera * getCamera () const
- void setSceneData (osg::Node *node)
- osg::Node * getSceneData (unsigned int childNo=0)
- const osg::Node * getSceneData (unsigned int childNo=0) const
- unsigned int getNumSceneData () const
- void setViewport (osg::Viewport *viewport)
- void setViewport (int x, int y, int width, int height)
- osg::Viewport * getViewport ()
- const osg::Viewport * getViewport () const
- void setDisplaySettings (osg::DisplaySettings *vs)
- const osg::DisplaySettings * getDisplaySettings () const
- osg::DisplaySettings * getDisplaySettings ()
- void setClearColor (const osg::Vec4 &color)
- const osg::Vec4 & getClearColor () const
- void setRedrawInterlacedStereoStencilMask (bool flag)
- bool getRedrawInterlacedStereoStencilMask () const
- void **setGlobalStateSet** (osg::StateSet *state)
- osg::StateSet * **getGlobalStateSet** ()
- const osg::StateSet * **getGlobalStateSet** () const
- void **setLocalStateSet** (osg::StateSet *state)
- osg::StateSet * **getLocalStateSet** ()
- const osg::StateSet * **getLocalStateSet** () const
- void setActiveUniforms (int activeUniforms)
- int getActiveUniforms () const
- void **updateUniforms** ()
- void **setLightingMode** (LightingMode mode)
- LightingMode **getLightingMode** () const

- void **setLight** (osg::Light *light)
- osg::Light * **getLight** ()
- const osg::Light * **getLight** () const
- void **setState** (osg::State *state)
- osg::State * **getState** ()
- const osg::State * **getState** () const
- void **setView** (osg::View *view)
- osg::View * **getView** ()
- const osg::View * **getView** () const
- void **setRenderInfo** (osg::RenderInfo &renderInfo)
- osg::RenderInfo & **getRenderInfo** ()
- const osg::RenderInfo & **getRenderInfo** () const
- void setProjectionMatrix (const osg::Matrixf &matrix)
- void setProjectionMatrix (const osg::Matrixd &matrix)
- void setProjectionMatrixAsOrtho (double left, double right, double bottom, double top, double zNear, double zFar)
- void setProjectionMatrixAsOrtho2D (double left, double right, double bottom, double top)
- void setProjectionMatrixAsFrustum (double left, double right, double bottom, double top, double zNear, double zFar)
- void setProjectionMatrixAsPerspective (double fovy, double aspectRatio, double zNear, double zFar)
- osg::Matrixd & getProjectionMatrix ()
- const osg::Matrixd & getProjectionMatrix () const
- bool getProjectionMatrixAsOrtho (double &left, double &right, double &bottom, double &top, double &zNear, double &zFar) const
- bool getProjectionMatrixAsFrustum (double &left, double &right, double &bottom, double &top, double &zNear, double &zFar) const
- bool getProjectionMatrixAsPerspective (double &fovy, double &aspectRatio, double &zNear, double &zFar) const
- void setViewMatrix (const osg::Matrixf &matrix)
- void setViewMatrix (const osg::Matrixd &matrix)
- void setViewMatrixAsLookAt (const osg::Vec3 &eye, const osg::Vec3 &center, const osg::Vec3 &up)
- osg::Matrixd & getViewMatrix ()
- const osg::Matrixd & getViewMatrix () const
- void getViewMatrixAsLookAt (osg::Vec3 &eye, osg::Vec3 &center, osg::Vec3 &up, float lookDistance=1.0f) const
- void **setInitVisitor** (osg::NodeVisitor *av)
- osg::NodeVisitor * **getInitVisitor** ()
- const osg::NodeVisitor * **getInitVisitor** () const
- void **setUpdateVisitor** (osg::NodeVisitor *av)
- osg::NodeVisitor * **getUpdateVisitor** ()
- const osg::NodeVisitor * **getUpdateVisitor** () const
- void **setCullVisitor** (osgUtil::CullVisitor *cv)
- osgUtil::CullVisitor * **getCullVisitor** ()
- const osgUtil::CullVisitor * **getCullVisitor** () const
- void **setCullVisitorLeft** (osgUtil::CullVisitor *cv)
- osgUtil::CullVisitor * **getCullVisitorLeft** ()

- const osgUtil::CullVisitor * **getCullVisitorLeft** () const
- void **setCullVisitorRight** (osgUtil::CullVisitor *cv)
- osgUtil::CullVisitor * **getCullVisitorRight** ()
- const osgUtil::CullVisitor * **getCullVisitorRight** () const
- void **setCollectOccludersVisitor** (osg::CollectOccludersVisitor *cov)
- osg::CollectOccludersVisitor * **getCollectOccludersVisitor** ()
- const osg::CollectOccludersVisitor * **getCollectOccludersVisitor** () const
- void **setStateGraph** (osgUtil::StateGraph *rg)
- osgUtil::StateGraph * **getStateGraph** ()
- const osgUtil::StateGraph * **getStateGraph** () const
- void **setStateGraphLeft** (osgUtil::StateGraph *rg)
- osgUtil::StateGraph * **getStateGraphLeft** ()
- const osgUtil::StateGraph * **getStateGraphLeft** () const
- void **setStateGraphRight** (osgUtil::StateGraph *rg)
- osgUtil::StateGraph * **getStateGraphRight** ()
- const osgUtil::StateGraph * **getStateGraphRight** () const
- void **setRenderStage** (osgUtil::RenderStage *rs)
- osgUtil::RenderStage * **getRenderStage** ()
- const osgUtil::RenderStage * **getRenderStage** () const
- void **setRenderStageLeft** (osgUtil::RenderStage *rs)
- osgUtil::RenderStage * **getRenderStageLeft** ()
- const osgUtil::RenderStage * **getRenderStageLeft** () const
- void **setRenderStageRight** (osgUtil::RenderStage *rs)
- osgUtil::RenderStage * **getRenderStageRight** ()
- const osgUtil::RenderStage * **getRenderStageRight** () const
- void setDrawBufferValue (GLenum drawBufferValue)
- GLenum getDrawBufferValue () const
- void setFusionDistance (FusionDistanceMode mode, float value=1.0f)
- FusionDistanceMode getFusionDistanceMode () const
- float getFusionDistanceValue () const
- void setPrioritizeTextures (bool pt)
- bool getPrioritizeTextures () const
- void **setComputeStereoMatricesCallback** (ComputeStereoMatricesCallback *callback)
- ComputeStereoMatricesCallback * **getComputeStereoMatricesCallback** ()
- const ComputeStereoMatricesCallback * **getComputeStereoMatricesCallback** () const
- bool projectWindowIntoObject (const osg::Vec3 &window, osg::Vec3 &object) const
- bool projectWindowXYIntoObject (int x, int y, osg::Vec3 &near_point, osg::Vec3 &far_point) const
- bool projectObjectIntoWindow (const osg::Vec3 &object, osg::Vec3 &window) const
- void setFrameStamp (osg::FrameStamp *fs)
- const osg::FrameStamp * getFrameStamp () const
- osg::Matrixd **computeLeftEyeProjection** (const osg::Matrixd &projection) const
- osg::Matrixd **computeLeftEyeView** (const osg::Matrixd &view) const
- osg::Matrixd **computeRightEyeProjection** (const osg::Matrixd &projection) const
- osg::Matrixd **computeRightEyeView** (const osg::Matrixd &view) const
- virtual osg::Matrixd **computeLeftEyeProjectionImplementation** (const osg::Matrixd &projection) const

- virtual osg::Matrixd **computeLeftEyeViewImplementation** (const osg::Matrixd &view) const
- virtual osg::Matrixd **computeRightEyeProjectionImplementation** (const osg::Matrixd &projection) const
- virtual osg::Matrixd **computeRightEyeViewImplementation** (const osg::Matrixd &view) const
- virtual void inheritCullSettings (const osg::CullSettings &settings)
- virtual void inheritCullSettings (const osg::CullSettings &settings, unsigned int inheritanceMask)
- virtual void init ()
- virtual void update ()
- virtual void cull ()
- virtual void draw ()
- unsigned int getDynamicObjectCount () const
- virtual void releaseAllGLObjects ()
- virtual void flushAllDeletedGLObjects ()
- virtual void flushDeletedGLObjects (double &availableTime)
- bool getStats (Statistics &primStats)

Classes

- struct ComputeStereoMatricesCallback

5.106 Detailed Description

SceneView is literaly a view of a scene, encapsulating the rendering of the scene. Provides methods for setting up the view and rendering it.

5.107 Member Enumeration Documentation

enum osgUtil::SceneView::FusionDistanceMode

FusionDistanceMode is used only when working in stereo.

Enumerator:

USE_FUSION_DISTANCE_VALUE Use fusion distance from the value set on the SceneView.

PROPORTIONAL_TO_SCREEN_DISTANCE Compute the fusion distance by multiplying the screen distance by the fusion distance value.

5.108 Constructor & Destructor Documentation

osgUtil::SceneView::SceneView (osg::DisplaySettings * *ds* = `NULL`)

Construct a default scene view.

5.109 Member Function Documentation

virtual void osgUtil::SceneView::setDefaults (unsigned int *options*) `[virtual]`

Set scene view to use default global state, light, camera and render visitor.

void osgUtil::SceneView::setCamera (osg::Camera * *camera*, bool *assumeOwnershipOfCamera* = `true`)

Set the camera used to represent the camera view of this SceneView.

osg::Camera* osgUtil::SceneView::getCamera () `[inline]`

Get the camera used to represent the camera view of this SceneView.

const osg::Camera* osgUtil::SceneView::getCamera () const `[inline]`

Get the const camera used to represent the camera view of this SceneView.

void osgUtil::SceneView::setSceneData (osg::Node * *node*)

Set the data to view. The data will typically be an osg::Scene but can be any osg::Node type.

osg::Node* osgUtil::SceneView::getSceneData (unsigned int *childNo* = `0`) `[inline]`

Get the scene data to view. The data will typically be an osg::Scene but can be any osg::Node type.

const osg::Node* osgUtil::SceneView::getSceneData (unsigned int *childNo* = `0`) const `[inline]`

Get the const scene data which to view. The data will typically be an osg::Scene but can be any osg::Node type.

unsigned int osgUtil::SceneView::getNumSceneData () const `[inline]`

Get the number of scene data subgraphs added to the SceneView's camera.

void osgUtil::SceneView::setViewport (osg::Viewport * *viewport*) `[inline]`

Set the viewport of the scene view to use specified osg::Viewport.

void osgUtil::SceneView::setViewport (int *x*, int *y*, int *width*, int *height*) `[inline]`

Set the viewport of the scene view to specified dimensions.

osg::Viewport∗ osgUtil::SceneView::getViewport () `[inline]`

Get the viewport.

const osg::Viewport∗ osgUtil::SceneView::getViewport () const `[inline]`

Get the const viewport.

void osgUtil::SceneView::setDisplaySettings (osg::DisplaySettings ∗ *vs*) `[inline]`

Set the DisplaySettings.

const osg::DisplaySettings∗ osgUtil::SceneView::getDisplaySettings () const `[inline]`

Get the const DisplaySettings

osg::DisplaySettings∗ osgUtil::SceneView::getDisplaySettings () `[inline]`

Get the DisplaySettings

void osgUtil::SceneView::setClearColor (const osg::Vec4 & *color*) `[inline]`

Set the color used in glClearColor(). Defaults to an off blue color.

const osg::Vec4& osgUtil::SceneView::getClearColor () const `[inline]`

Get the color used in glClearColor.

void osgUtil::SceneView::setRedrawInterlacedStereoStencilMask (bool *flag*) `[inline]`

Manually set the redraw interlaced stereo stencil mask request flag to control whether to redraw the stencil buffer on the next frame.

bool osgUtil::SceneView::getRedrawInterlacedStereoStencilMask () const `[inline]`

Get the redraw interlaced stereo stencil mask request flag.

void osgUtil::SceneView::setActiveUniforms (int *activeUniforms*) `[inline]`

Set the uniforms that SceneView should set set up on each frame.

int osgUtil::SceneView::getActiveUniforms () const `[inline]`

Get the uniforms that SceneView should set set up on each frame.

void osgUtil::SceneView::setProjectionMatrix (const osg::Matrixf & *matrix*) `[inline]`

Set the projection matrix. Can be thought of as setting the lens of a camera.

void osgUtil::SceneView::setProjectionMatrix (const osg::Matrixd & *matrix*) `[inline]`

Set the projection matrix. Can be thought of as setting the lens of a camera.

void osgUtil::SceneView::setProjectionMatrixAsOrtho (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*)

Set to an orthographic projection. See OpenGL glOrtho for documentation further details.

void osgUtil::SceneView::setProjectionMatrixAsOrtho2D (double *left*, double *right*, double *bottom*, double *top*)

Set to a 2D orthographic projection. See OpenGL glOrtho2D documentation for further details.

void osgUtil::SceneView::setProjectionMatrixAsFrustum (double *left*, double *right*, double *bottom*, double *top*, double *zNear*, double *zFar*)

Set to a perspective projection. See OpenGL glFrustum documentation for further details.

void osgUtil::SceneView::setProjectionMatrixAsPerspective (double *fovy*, double *aspectRatio*, double *zNear*, double *zFar*)

Create a symmetrical perspective projection, See OpenGL gluPerspective documentation for further details. Aspect ratio is defined as width/height.

osg::Matrixd& osgUtil::SceneView::getProjectionMatrix () `[inline]`

Get the projection matrix.

const osg::Matrixd& osgUtil::SceneView::getProjectionMatrix () const `[inline]`

Get the const projection matrix.

bool osgUtil::SceneView::getProjectionMatrixAsOrtho (double & *left*, double & *right*, double & *bottom*, double & *top*, double & *zNear*, double & *zFar*) const

Get the orthographic settings of the orthographic projection matrix. Returns false if matrix is not an orthographic matrix, where parameter values are undefined.

bool osgUtil::SceneView::getProjectionMatrixAsFrustum (double & *left*, double & *right*, double & *bottom*, double & *top*, double & *zNear*, double & *zFar*) const

Get the frustum setting of a perspective projection matrix. Returns false if matrix is not a perspective matrix, where parameter values are undefined.

bool osgUtil::SceneView::getProjectionMatrixAsPerspective (double & *fovy*, double & *aspectRatio*, double & *zNear*, double & *zFar*) const

Get the frustum setting of a symmetric perspective projection matrix. Returns false if matrix is not a perspective matrix, where parameter values are undefined. Note, if matrix is not a symmetric perspective matrix then the shear will be lost. Asymmetric matrices occur when stereo, power walls, caves and reality center display are used. In these configurations one should use the 'getProjectionMatrixAsFrustum' method instead.

void osgUtil::SceneView::setViewMatrix (const osg::Matrixf & *matrix*) `[inline]`

Set the view matrix. Can be thought of as setting the position of the world relative to the camera in camera coordinates.

void osgUtil::SceneView::setViewMatrix (const osg::Matrixd & *matrix*) `[inline]`

Set the view matrix. Can be thought of as setting the position of the world relative to the camera in camera coordinates.

void osgUtil::SceneView::setViewMatrixAsLookAt (const osg::Vec3 & *eye*, const osg::Vec3 & *center*, const osg::Vec3 & *up*)

Set to the position and orientation of view matrix, using the same convention as gluLookAt.

osg::Matrixd& osgUtil::SceneView::getViewMatrix () `[inline]`

Get the view matrix.

const osg::Matrixd& osgUtil::SceneView::getViewMatrix () const `[inline]`

Get the const view matrix.

void osgUtil::SceneView::getViewMatrixAsLookAt (osg::Vec3 & *eye*, osg::Vec3 & *center*, osg::Vec3 & *up*, float *lookDistance* = `1.0f`) const

Get to the position and orientation of a modelview matrix, using the same convention as gluLookAt.

void osgUtil::SceneView::setDrawBufferValue (GLenum *drawBufferValue*) `[inline]`

Set the draw buffer value used at the start of each frame draw. Note, overridden in quad buffer stereo mode

GLenum osgUtil::SceneView::getDrawBufferValue () const `[inline]`

Get the draw buffer value used at the start of each frame draw.

void osgUtil::SceneView::setFusionDistance (FusionDistanceMode *mode*, float *value* = 1.0f) `[inline]`

Set the FusionDistanceMode and Value. Note, is used only when working in stereo.

FusionDistanceMode osgUtil::SceneView::getFusionDistanceMode () const `[inline]`

Get the FusionDistanceMode.

float osgUtil::SceneView::getFusionDistanceValue () const `[inline]`

Get the FusionDistanceValue. Note, only used for USE_FUSION_DISTANCE_VALUE & PROPORTIONAL_TO_SCREEN_DISTANCE modes.

void osgUtil::SceneView::setPrioritizeTextures (bool *pt*) `[inline]`

Set whether the draw method should call renderer->prioritizeTexture.

bool osgUtil::SceneView::getPrioritizeTextures () const `[inline]`

Get whether the draw method should call renderer->prioritizeTexture.

bool osgUtil::SceneView::projectWindowIntoObject (const osg::Vec3 & *window*, osg::Vec3 & *object*) const

Calculate the object coordinates of a point in window coordinates. Note, current implementation requires that SceneView::draw() has been previously called for projectWindowIntoObject to produce valid values. Consistent with OpenGL windows coordinates are calculated relative to the bottom left of the window. Returns true on successful projection.

bool osgUtil::SceneView::projectWindowXYIntoObject (int *x*, int *y*, osg::Vec3 & *near_point*, osg::Vec3 & *far_point*) const

Calculate the object coordinates of a window x,y when projected onto the near and far planes. Note, current implementation requires that SceneView::draw() has been previously called for projectWindowIntoObject to produce valid values. Consistent with OpenGL windows coordinates are calculated relative to the bottom left of the window. Returns true on successful projection.

bool osgUtil::SceneView::projectObjectIntoWindow (const osg::Vec3 & *object*, osg::Vec3 & *window*) const

Calculate the window coordinates of a point in object coordinates. Note, current implementation requires that SceneView::draw() has been previously called for projectWindowIntoObject to produce valid values. Consistent with OpenGL windows coordinates are calculated relative to the bottom left of the window, whereas window API's normally have the top left as the origin, so you may need to pass in (mouseX,window_height-mouseY,...). Returns true on successful projection.

void osgUtil::SceneView::setFrameStamp (osg::FrameStamp * *fs*) `[inline]`

Set the frame stamp for the current frame.

const osg::FrameStamp* osgUtil::SceneView::getFrameStamp () const `[inline]`

Get the frame stamp for the current frame.

virtual void osgUtil::SceneView::inheritCullSettings (const osg::CullSettings & *settings*) `[inline, virtual]`

Inherit the local cull settings variable from specified CullSettings object, according to the inheritance mask.

virtual void osgUtil::SceneView::inheritCullSettings (const osg::CullSettings & *settings*, unsigned int *inheritanceMask*) `[virtual]`

Inherit the local cull settings variable from specified CullSettings object, according to the inheritance mask.

virtual void osgUtil::SceneView::init () `[virtual]`

Do init traversal of attached scene graph using Init NodeVisitor. The init traversal is called once for each SceneView, and should be used to compile display list, texture objects intialize data not otherwise intialized during scene graph loading. Note, is called automatically by update & cull if it hasn't already been called elsewhere. Also init() should only ever be called within a valid graphics context.

virtual void osgUtil::SceneView::update () `[virtual]`

Do app traversal of attached scene graph using App NodeVisitor.

virtual void osgUtil::SceneView::cull () `[virtual]`

Do cull traversal of attached scene graph using Cull NodeVisitor.

virtual void osgUtil::SceneView::draw () `[virtual]`

Do draw traversal of draw bins generated by cull traversal.

unsigned int osgUtil::SceneView::getDynamicObjectCount () const `[inline]`

Compute the number of dynamic objects that will be held in the rendering backend

virtual void osgUtil::SceneView::releaseAllGLObjects () `[virtual]`

Release all OpenGL objects from the scene graph, such as texture objects, display lists etc. These released scene graphs placed in the respective delete GLObjects cache, which then need to be deleted in OpenGL by SceneView::flushAllDeleteGLObjects().

virtual void osgUtil::SceneView::flushAllDeletedGLObjects () `[virtual]`

Flush all deleted OpenGL objects, such as texture objects, display lists etc.

virtual void osgUtil::SceneView::flushDeletedGLObjects (double & *availableTime*) `[virtual]`

Flush deleted OpenGL objects, such as texture objects, display lists etc within specified available time.

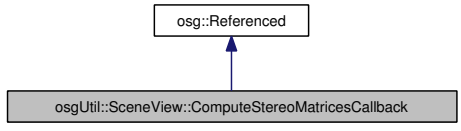
bool osgUtil::SceneView::getStats (Statistics & *primStats*)

Extract stats for current draw list.

virtual bool osgUtil::SceneView::cullStage (const osg::Matrixd & *projection*, const osg::Matrixd & *modelview*, osgUtil::CullVisitor * *cullVisitor*, osgUtil::StateGraph * *rendergraph*, osgUtil::RenderStage * *renderStage*) `[protected, virtual]`

Do cull traversal of attached scene graph using Cull NodeVisitor. Return true if computeNearFar has been done during the cull traversal.

5.110 osgUtil::SceneView::ComputeStereoMatricesCallback Struct Reference



Public Member Functions

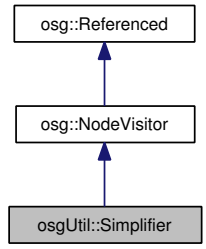
- virtual osg::Matrixd **computeLeftEyeProjection** (const osg::Matrixd &projection) const =0
- virtual osg::Matrixd **computeLeftEyeView** (const osg::Matrixd &view) const =0

- virtual osg::Matrixd **computeRightEyeProjection** (const osg::Matrixd &projection) const =0
- virtual osg::Matrixd **computeRightEyeView** (const osg::Matrixd &view) const =0

5.111 Detailed Description

Callback for overidding the default method for compute the offset projection and view matrices.

5.112 osgUtil::Simplifier Class Reference



Public Types

- typedef std::vector< unsigned int > **IndexList**

Public Member Functions

- **Simplifier** (float sampleRatio=1.0f, float maximumError=FLT_MAX, float maximumLength=0.0)
- void **setSampleRatio** (float sampleRatio)
- float **getSampleRatio** () const
- void setMaximumError (float error)
- float **getMaximumError** () const
- void setMaximumLength (float length)
- float **getMaximumLength** () const
- void **setDoTriStrip** (bool on)
- bool **getDoTriStrip** () const
- void **setSmoothing** (bool on)
- bool **getSmoothing** () const
- void **setContinueSimplificationCallback** (ContinueSimplificationCallback *cb)
- ContinueSimplificationCallback * **getContinueSimplificationCallback** ()
- const ContinueSimplificationCallback * **getContinueSimplificationCallback** () const

- bool **continueSimplification** (float nextError, unsigned int numOriginalPrimitives, unsigned int numRemainingPrimitives) const
- virtual bool **continueSimplificationImplementation** (float nextError, unsigned int numOriginalPrimitives, unsigned int numRemainingPrimitives) const
- virtual void **apply** (osg::Geode &geode)
- void simplify (osg::Geometry &geometry)
- void simplify (osg::Geometry &geometry, const IndexList &protectedPoints)

  a list of point indices

Classes

- class **ContinueSimplificationCallback**

5.113 Detailed Description

A simplifier for reducing the number of traingles in osg::Geometry.

5.114 Member Function Documentation

void osgUtil::Simplifier::setMaximumError (float *error*) `[inline]`

Set the maximum point error that all point removals must be less than to permit removal of a point. Note, Only used when down sampling. i.e. sampleRatio < 1.0

void osgUtil::Simplifier::setMaximumLength (float *length*) `[inline]`

Set the maximum length target that all edges must be shorted than. Note, Only used when up sampling i.e. sampleRatio > 1.0.

void osgUtil::Simplifier::simplify (osg::Geometry & *geometry*)

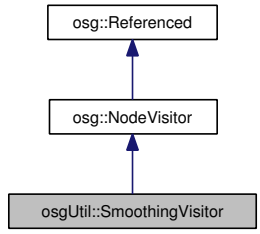
simply the geometry.

void osgUtil::Simplifier::simplify (osg::Geometry & *geometry*, const IndexList & *protectedPoints*)

a list of point indices

simply the geometry, whilst protecting key points from being modified.

5.115 osgUtil::SmoothingVisitor Class Reference



Public Member Functions

- SmoothingVisitor ()
  default to traversing all children.
- virtual void apply (osg::Geode &geode)
  apply smoothing method to all geode geosets.

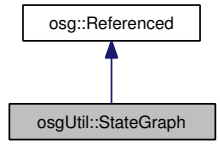
Static Public Member Functions

- static void smooth (osg::Geometry &geoset)
  smooth geoset by creating per vertex normals.

5.116 Detailed Description

A smoothing visitor for calculating smoothed normals for osg::GeoSet's which contains surface primitives.

5.117 osgUtil::StateGraph Class Reference



Public Types

- typedef std::map< const osg::StateSet *, osg::ref_ptr< StateGraph > > **ChildList**
- typedef std::vector< osg::ref_ptr< RenderLeaf > > **LeafList**

Public Member Functions

- **StateGraph** (StateGraph *parent, const osg::StateSet *stateset)
- StateGraph * **cloneType** () const
- void **setUserData** (osg::Referenced *obj)
- osg::Referenced * **getUserData** ()
- const osg::Referenced * **getUserData** () const
- const osg::StateSet * **getStateSet** () const
- bool empty () const
- bool **leaves_empty** () const
- float **getAverageDistance** () const
- float **getMinimumDistance** () const
- void **sortFrontToBack** ()
- void reset ()
- void clean ()
- void prune ()
- StateGraph * **find_or_insert** (const osg::StateSet *stateset)
- void addLeaf (RenderLeaf *leaf)

Static Public Member Functions

- static void **moveStateGraph** (osg::State &state, StateGraph *sg_curr, StateGraph *sg_new)
- static void **moveToRootStateGraph** (osg::State &state, StateGraph *sg_curr)
- static int **numToPop** (StateGraph *sg_curr)

Public Attributes

- StateGraph * **_parent**
- const osg::StateSet * **_stateset**
- int **_depth**
- ChildList **_children**
- LeafList **_leaves**
- float **_averageDistance**
- float **_minimumDistance**
- osg::ref_ptr< osg::Referenced > **_userData**
- bool **_dynamic**

5.118 Detailed Description

StateGraph - contained in a renderBin, defines the scene to be drawn.

5.119 Member Function Documentation

bool osgUtil::StateGraph::empty () const `[inline]`

return true if all of drawables, lights and children are empty.

void osgUtil::StateGraph::reset ()

Reset the internal contents of a StateGraph, including deleting all children.

void osgUtil::StateGraph::clean ()

Recursively clean the StateGraph of all its drawables, lights and depths. Leaves children intact, and ready to be populated again.

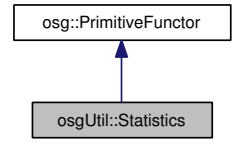
void osgUtil::StateGraph::prune ()

Recursively prune the StateGraph of empty children.

void osgUtil::StateGraph::addLeaf (RenderLeaf * *leaf*) `[inline]`

add a render leaf.

5.120 osgUtil::Statistics Class Reference



Public Types

- enum **StatsType** {
  STAT_NONE,

STAT_FRAMERATE,
STAT_GRAPHS,
STAT_PRIMS,
STAT_PRIMSPERVIEW,
STAT_PRIMSPERBIN,
STAT_DC,
STAT_RESTART }

- typedef std::pair< unsigned int, unsigned int > **PrimitivePair**
- typedef std::map< GLenum, PrimitivePair > **PrimitiveValueMap**
- typedef std::map< GLenum, unsigned int > **PrimitiveCountMap**

Public Member Functions

- void **reset** ()
- void **setType** (StatsType t)
- virtual void setVertexArray (unsigned int count, const osg::Vec3 *)
- virtual void setVertexArray (unsigned int count, const osg::Vec2 *)
- virtual void setVertexArray (unsigned int count, const osg::Vec4 *)
- virtual void setVertexArray (unsigned int count, const osg::Vec3d *)
- virtual void setVertexArray (unsigned int count, const osg::Vec2d *)
- virtual void setVertexArray (unsigned int count, const osg::Vec4d *)
- virtual void drawArrays (GLenum mode, GLint, GLsizei count)

  Mimics the OpenGL `glDrawArrays()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLubyte *)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLushort *)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void drawElements (GLenum mode, GLsizei count, const GLuint *)

  Mimics the OpenGL `glDrawElements()` *function.*

- virtual void begin (GLenum mode)

  Mimics the OpenGL `glBegin()` *function.*

- void **vertex** ()
- virtual void vertex (float, float, float)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const osg::Vec3 &)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const osg::Vec2 &)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (const osg::Vec4 &)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float, float)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void vertex (float, float, float, float)

  Mimics the OpenGL `glVertex()` *"family of functions".*

- virtual void end ()

  Mimics the OpenGL `glEnd()` *function.*

- void **addDrawable** ()
- void **addMatrix** ()
- void **addLight** (int np)
- void **addImpostor** (int np)
- int **getBins** ()
- void **setDepth** (int d)
- void **addBins** (int np)
- void **setBinNo** (int n)
- void **add** (const Statistics &stats)
- PrimitiveCountMap::iterator **GetPrimitivesBegin** ()
- PrimitiveCountMap::iterator **GetPrimitivesEnd** ()

Public Attributes

- int **numDrawables**
- int **nummat**
- int **nbins**
- int **nlights**
- int **depth**
- int **_binNo**
- StatsType **stattype**
- int **nimpostor**
- unsigned int **_vertexCount**
- PrimitiveValueMap **_primitiveCount**
- GLenum **_currentPrimitiveFunctorMode**

5.121 Detailed Description

Statistics base class. Used to extract primitive information from the renderBin(s). Add a case of getStats(osgUtil::Statistics *stat) for any new drawable (or drawable derived class) that you generate (eg see Geometry.cpp). There are 20 types of drawable counted - actually only 14 cases can occur in reality. these represent sets of GL_POINTS, GL_LINES GL_LINESTRIPS, LOOPS, TRIANGLES, TRI-fans, tristrips, quads, quadstrips etc The number of triangles rendered is inferred: each triangle = 1 triangle (number of vertices/3) each quad = 2 triangles (nverts/2) each trifan or tristrip = (length-2) triangles and so on.

5.122 Member Function Documentation

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec3 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec2 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec4 * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec3d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec2d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

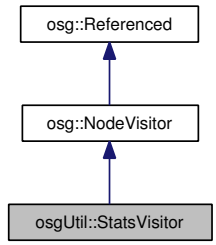
Implements osg::PrimitiveFunctor.

virtual void osgUtil::Statistics::setVertexArray (unsigned int *count*, const osg::Vec4d * *vertices*) `[inline, virtual]`

Sets the array of vertices used to describe the primitives. Somehow mimics the OpenGL `glVertexPointer()` function.

Implements osg::PrimitiveFunctor.

5.123 osgUtil::StatsVisitor Class Reference



Public Types

- typedef std::set< osg::Node * > **NodeSet**
- typedef std::set< osg::Drawable * > **DrawableSet**
- typedef std::set< osg::StateSet * > **StateSetSet**

Public Member Functions

- void reset ()
- void **apply** (osg::Node &node)
- void **apply** (osg::Group &node)
- void **apply** (osg::Transform &node)
- void **apply** (osg::LOD &node)
- void **apply** (osg::Switch &node)
- void **apply** (osg::Geode &node)
- void **apply** (osg::Drawable &drawable)
- void **totalUpStats** ()
- void **print** (std::ostream &out)

Public Attributes

- unsigned int **_numInstancedGroup**
- unsigned int **_numInstancedSwitch**
- unsigned int **_numInstancedLOD**
- unsigned int **_numInstancedTransform**
- unsigned int **_numInstancedGeode**
- unsigned int **_numInstancedDrawable**
- unsigned int **_numInstancedGeometry**
- unsigned int **_numInstancedStateSet**
- NodeSet **_groupSet**
- NodeSet **_transformSet**
- NodeSet **_lodSet**
- NodeSet **_switchSet**
- NodeSet **_geodeSet**
- DrawableSet **_drawableSet**
- DrawableSet **_geometrySet**
- StateSetSet **_statesetSet**
- osgUtil::Statistics **_uniqueStats**
- osgUtil::Statistics **_instancedStats**

5.124 Detailed Description

StatsVisitor for collecting statistics about scene graph.

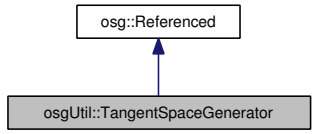
5.125 Member Function Documentation

void osgUtil::StatsVisitor::reset () `[virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented from osg::NodeVisitor.

5.126 osgUtil::TangentSpaceGenerator Class Reference



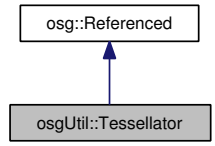
Public Member Functions

- **TangentSpaceGenerator** (const TangentSpaceGenerator &copy, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)
- void **generate** (osg::Geometry *geo, int normal_map_tex_unit=0)
- osg::Vec4Array * **getTangentArray** ()
- const osg::Vec4Array * **getTangentArray** () const
- void **setTangentArray** (osg::Vec4Array *array)
- osg::Vec4Array * **getNormalArray** ()
- const osg::Vec4Array * **getNormalArray** () const
- void **setNormalArray** (osg::Vec4Array *array)
- osg::Vec4Array * **getBinormalArray** ()
- const osg::Vec4Array * **getBinormalArray** () const
- void **setBinormalArray** (osg::Vec4Array *array)
- osg::IndexArray * **getIndices** ()

5.127 Detailed Description

The TangentSpaceGenerator class generates three arrays containing tangent-space basis vectors. It takes a texture-mapped Geometry object as input, traverses its primitive sets and computes Tangent, Normal and Binormal vectors for each vertex, storing them into arrays. The resulting arrays can be used as vertex program varying (per-vertex) parameters, enabling advanced effects like bump-mapping. To use this class, simply call the generate() method specifying the Geometry object you want to process and the texture unit that contains UV mapping for the normal map; then you can retrieve the TBN arrays by calling getTangentArray(), getNormalArray() and getBinormalArray() methods.

5.128 osgUtil::Tessellator Class Reference



Public Types

- enum WindingType {
  TESS_WINDING_ODD,
  TESS_WINDING_NONZERO,
  TESS_WINDING_POSITIVE,

TESS_WINDING_NEGATIVE,

TESS_WINDING_ABS_GEQ_TWO }

- enum TessellationType {

  TESS_TYPE_GEOMETRY,

  TESS_TYPE_DRAWABLE,

  TESS_TYPE_POLYGONS }
- typedef std::vector< osg::Vec3 * > **VertexPointList**
- typedef std::vector< osg::ref_ptr< Prim > > **PrimList**

Public Member Functions

- void setBoundaryOnly (const bool tt)
- const bool **getBoundaryOnly** ()
- void setWindingType (const WindingType wt)
- const WindingType **getWindingType** ()
- void setTessellationType (const TessellationType tt)
- const TessellationType **getTessellationType** ()
- void retessellatePolygons (osg::Geometry &cxgeom)
- void setTessellationNormal (const osg::Vec3 norm)
- osg::Geometry::PrimitiveSetList **getContours** ()
- void **beginTessellation** ()
- void **beginContour** ()
- void **addVertex** (osg::Vec3 *vertex)
- void **endContour** ()
- void **endTessellation** ()
- PrimList & **getPrimList** ()
- void **reset** ()

Classes

- struct **NewVertex**
- struct **Prim**
- struct **Vec3d**

5.129 Detailed Description

Originally a simple class for tessellating a single polygon boundary. Using old style glu tessellation functions for portability. Upgraded Jan 2004 to use the modern glu tessellation functions.

5.130 Member Enumeration Documentation

enum osgUtil::Tessellator::WindingType

The winding rule, see red book ch 11.

enum osgUtil::Tessellator::TessellationType

we interpret all contours in the geometry as a single set to be tessellated or each separate drawable's contours needs to be tessellated.

5.131 Member Function Documentation

void osgUtil::Tessellator::setBoundaryOnly (const bool *tt*) `[inline]`

Set and get tessellation request boundary only on/off

void osgUtil::Tessellator::setWindingType (const WindingType *wt*) `[inline]`

Set and get tessellation windong rule

void osgUtil::Tessellator::setTessellationType (const TessellationType *tt*) `[inline]`

Set and get tessellation type

void osgUtil::Tessellator::retessellatePolygons (osg::Geometry & *cxgeom*)

Change the contours lists of the geometry into tessellated primitives (the list of primitives in the original geometry is stored in the Tessellator for possible re-use. The name remains retessellatePolygons although it now handles trifans, strips, quads etc. as well as Polygons so as to not break old codes relying on this function name.

void osgUtil::Tessellator::setTessellationNormal (const osg::Vec3 *norm*) `[inline]`

Define the normal to the tessellated polygon - this provides a hint how to tessellate the contours; see gluTessNormal in red book or man pages. GWM July 2005. Can improve teselation "For example, if you know that all polygons lie in the x-y plane, call gluTessNormal(tess, 0.0, 0.0, 1.0) before rendering any polygons."

void osgUtil::Tessellator::reduceArray (osg::Array * *cold*, const unsigned int *nnu*) `[protected]`

remove unused parts of the array, eg for wehn retessellating tessellation can introduce extra vertices for concave or crossing boundaries, these will leak memory if not removed when retessellating.

5.132 Member Data Documentation

WindingType osgUtil::Tessellator::_wtype [protected]

winding rule, which parts will become solid

TessellationType osgUtil::Tessellator::_ttype [protected]

tessellation rule, which parts will become solid

unsigned int osgUtil::Tessellator::_numberVerts [protected]

number of vertices that are part of the 'original' set of contours

osg::Geometry::PrimitiveSetList osgUtil::Tessellator::_Contours [protected]

List of primitives that define the contours

unsigned int osgUtil::Tessellator::_index [protected]

count number of primitives in a geometry to get right no. of norms/colurs etc for per_primitive attributes.

osg::Vec3 osgUtil::Tessellator::tessNormal [protected]

the gluTessNormal for tessellation hint

unsigned int osgUtil::Tessellator::_extraPrimitives [protected]

count of number of extra primitives added

5.133 osgUtil::TransformAttributeFunctor Class Reference

Public Member Functions

- TransformAttributeFunctor (const osg::Matrix &m)
- virtual void apply (osg::Drawable::AttributeType type, unsigned int count, osg::Vec3 *begin)

Public Attributes

- osg::Matrix **_m**
- osg::Matrix **_im**

5.134 Detailed Description

Functor for transforming a drawable's vertex and normal attributes by specified matrix. typically used for flattening transform down onto drawable leaves.

5.135 Constructor & Destructor Documentation

osgUtil::TransformAttributeFunctor::TransformAttributeFunctor (const osg::Matrix & *m*)

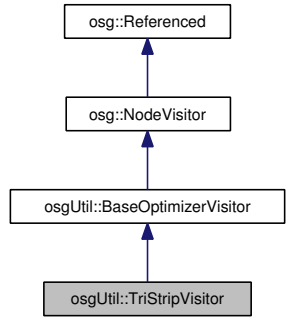
Construct a functor to transform a drawable's vertex and normal attributes by specified matrix.

5.136 Member Function Documentation

virtual void osgUtil::TransformAttributeFunctor::apply (osg::Drawable::AttributeType *type*, unsigned int *count*, osg::Vec3 * *begin*) `[virtual]`

Do the work of transforming vertex and normal attributes.

5.137 osgUtil::TriStripVisitor Class Reference



Public Member Functions

- TriStripVisitor (Optimizer *optimizer=0)

  default to traversing all children.

- void stripify (osg::Geometry &drawable)
- void stripify ()
- virtual void apply (osg::Geode &geode)

Accumulate the Geometry drawables to make into strips.

- void **setCacheSize** (unsigned int size)
- unsigned int **getCacheSize** ()
- const unsigned int **getCacheSize** () const
- void **setMinStripSize** (unsigned int size)
- unsigned int **getMinStripSize** ()
- const unsigned int **getMinStripSize** () const
- void **setGenerateFourPointPrimitivesQuads** (bool flag)
- bool **getGenerateFourPointPrimitivesQuads** () const

5.138 Detailed Description

A tri stripping visitor for converting Geometry surface primitives into tri strips. The current implemention is based upon Tanguy Fautre's triangulation code.

5.139 Member Function Documentation

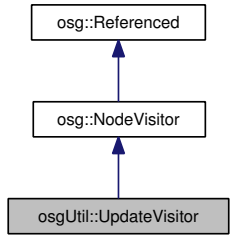
void osgUtil::TriStripVisitor::stripify (osg::Geometry & *drawable*)

Convert mesh primitives in Geometry into Tri Strips. Converts all primitive types except points and lines, linestrips which it leaves unchanged.

void osgUtil::TriStripVisitor::stripify ()

Stripify (make into strips of tria or quads) the accumulated list of Geometry drawables.

5.140 osgUtil::UpdateVisitor Class Reference



Public Member Functions

- virtual void reset ()
- virtual void apply (osg::Node &node)
- virtual void **apply** (osg::Geode &node)
- virtual void **apply** (osg::Billboard &node)
- virtual void **apply** (osg::LightSource &node)
- virtual void **apply** (osg::Group &node)
- virtual void **apply** (osg::Transform &node)
- virtual void **apply** (osg::Projection &node)
- virtual void **apply** (osg::Switch &node)
- virtual void **apply** (osg::LOD &node)
- virtual void **apply** (osg::OccluderNode &node)

5.141 Detailed Description

Basic UpdateVisitor implementation for animating a scene. This visitor traverses the scene graph, calling each nodes appCallback if it exists.

5.142 Member Function Documentation

virtual void osgUtil::UpdateVisitor::reset () `[virtual]`

Method to call to reset visitor. Useful if your visitor accumulates state during a traversal, and you plan to reuse the visitor. To flush that state for the next traversal: call reset() prior to each traversal.

Reimplemented from osg::NodeVisitor.

virtual void osgUtil::UpdateVisitor::apply (osg::Node & *node*) `[inline, virtual]`

During traversal each type of node calls its callbacks and its children traversed.

Reimplemented from osg::NodeVisitor.

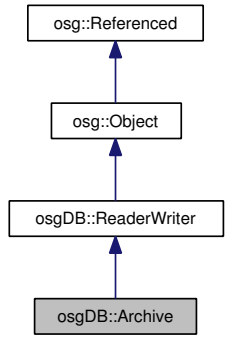
UpdateVisitor& osgUtil::UpdateVisitor::operator= (const UpdateVisitor &) `[inline, protected]`

Prevent unwanted copy construction. Prevent unwanted copy operator.

Chapter 6

osgDB Documentation

6.1 osgDB::Archive Class Reference



Public Types

- typedef std::vector< std::string > **FileNameList**

Public Member Functions

- virtual const char * libraryName () const
- virtual const char * className () const
- virtual bool **acceptsExtension** (const std::string &) const
- virtual void close ()=0
- virtual bool fileExists (const std::string &filename) const =0
- virtual std::string getMasterFileName () const =0

- virtual bool getFileNames (FileNameList &fileNameList) const =0
- virtual ReadResult **readObject** (const std::string &, const Options *=NULL) const =0
- virtual ReadResult **readImage** (const std::string &, const Options *=NULL) const =0
- virtual ReadResult **readHeightField** (const std::string &, const Options *=NULL) const =0
- virtual ReadResult **readNode** (const std::string &, const Options *=NULL) const =0
- virtual WriteResult **writeObject** (const osg::Object &, const std::string &, const Options *=NULL) const =0
- virtual WriteResult **writeImage** (const osg::Image &, const std::string &, const Options *=NULL) const =0
- virtual WriteResult **writeHeightField** (const osg::HeightField &, const std::string &, const Options *=NULL) const =0
- virtual WriteResult **writeNode** (const osg::Node &, const std::string &, const Options *=NULL) const =0

6.2 Detailed Description

Base class for implementing database Archives. See src/osgPlugins/osga for an example of a concrete implementation.

6.3 Member Function Documentation

virtual const char* osgDB::Archive::libraryName () const `[inline, virtual]`

return the name of the object's library. Must be defined by derived classes. The OpenSceneGraph convention is that the namespace of a library is the same as the library name.

Implements osg::Object.

virtual const char* osgDB::Archive::className () const `[inline, virtual]`

return the name of the object's class type. Must be defined by derived classes.

Implements osg::Object.

virtual void osgDB::Archive::close () `[pure virtual]`

close the archive.

virtual bool osgDB::Archive::fileExists (const std::string & *filename*) const `[pure virtual]`

return true if file exists in archive.

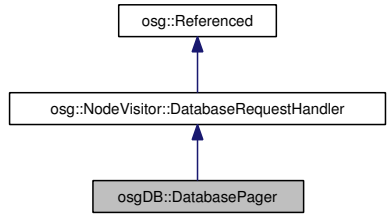
virtual std::string osgDB::Archive::getMasterFileName () const `[pure virtual]`

Get the file name which represents the master file recorded in the Archive.

virtual bool osgDB::Archive::getFileNames (FileNameList & *fileNameList*) const `[pure virtual]`

Get the full list of file names available in the archive.

6.4 osgDB::DatabasePager Class Reference



Public Types

- enum **DrawablePolicy** {
  DO_NOT_MODIFY_DRAWABLE_SETTINGS,
  USE_DISPLAY_LISTS,
  USE_VERTEX_BUFFER_OBJECTS,
  USE_VERTEX_ARRAYS }
- typedef OpenThreads::Thread::ThreadPriority **ThreadPriority**
- typedef std::list< osg::ref_ptr< osg::PagedLOD > > **PagedLODList**
- typedef std::set< osg::ref_ptr< osg::StateSet > > **StateSetList**
- typedef std::vector< osg::ref_ptr< osg::Drawable > > **DrawableList**
- typedef std::pair< StateSetList, DrawableList > **DataToCompile**
- typedef std::map< unsigned int, DataToCompile > **DataToCompileMap**
- typedef std::set< unsigned int > **ActiveGraphicsContexts**
- typedef std::vector< osg::observer_ptr< osg::GraphicsContext > > **CompileGraphicsContexts**

Public Member Functions

- **DatabasePager** (const DatabasePager &rhs)
- virtual DatabasePager * clone () const

- virtual void requestNodeFile (const std::string &fileName, osg::Group *group, float priority, const osg::FrameStamp *framestamp)
- virtual void **requestNodeFile** (const std::string &fileName, osg::Group *group, float priority, const osg::FrameStamp *framestamp, ReaderWriter::Options *loadOptions)
- virtual void run ()
- virtual int cancel ()
- virtual void clear ()
- void setDatabasePagerThreadPause (bool pause)
- bool getDatabasePagerThreadPause () const
- void setAcceptNewDatabaseRequests (bool acceptNewRequests)
- bool getAcceptNewDatabaseRequests () const
- int getNumFramesActive () const
- virtual void signalBeginFrame (const osg::FrameStamp *framestamp)
- virtual void signalEndFrame ()
- virtual void registerPagedLODs (osg::Node *subgraph)
- void setDoPreCompile (bool flag)
- bool getDoPreCompile () const
- void setTargetFrameRate (double tfr)
- double getTargetFrameRate () const
- void setMinimumTimeAvailableForGLCompileAndDeletePerFrame (double ta)
- double getMinimumTimeAvailableForGLCompileAndDeletePerFrame () const
- void setMaximumNumOfObjectsToCompilePerFrame (unsigned int num)
- unsigned int getMaximumNumOfObjectsToCompilePerFrame () const
- void setExpiryDelay (double expiryDelay)
- double getExpiryDelay () const
- void setDeleteRemovedSubgraphsInDatabaseThread (bool flag)
- bool getDeleteRemovedSubgraphsInDatabaseThread () const
- void setDrawablePolicy (DrawablePolicy policy)
- DrawablePolicy getDrawablePolicy () const
- void setUnrefImageDataAfterApplyPolicy (bool changeAutoUnRef, bool valueAutoUnRef)
- void getUnrefImageDataAfterApplyPolicy (bool &changeAutoUnRef, bool &valueAutoUnRef) const
- void setMaxAnisotropyPolicy (bool changeAnisotropy, float valueAnisotropy)
- void getMaxAnisotropyPolicy (bool &changeAnisotropy, float &valueAnisotropy) const
- bool requiresUpdateSceneGraph () const
- virtual void updateSceneGraph (double currentFrameTime)
- void setCompileGLObjectsForContextID (unsigned int contextID, bool on)
- bool getCompileGLObjectsForContextID (unsigned int contextID)
- bool requiresExternalCompileGLObjects (unsigned int contextID) const
- bool requiresCompileGLObjects () const
- virtual void compileGLObjects (osg::State &state, double &availableTime)
- virtual void compileAllGLObjects (osg::State &state)
- unsigned int getFileRequestListSize () const
- unsigned int getDataToCompileListSize () const
- double getMinimumTimeToMergeTile () const
- double getMaximumTimeToMergeTile () const

- double getAverageTimeToMergeTiles () const
- void resetStats ()
- void setMaximumNumOfRemovedChildPagedLODs (unsigned int number)
- unsigned int getMaximumNumOfRemovedChildPagedLODs () const
- void setMinimumNumOfInactivePagedLODs (unsigned int number)
- unsigned int getMinimumNumOfInactivePagedLODs () const

Static Public Member Functions

- static osg::ref_ptr< DatabasePager > & prototype ()
- static DatabasePager * create ()

Friends

- struct **DatabaseRequest**
- class **FindCompileableGLObjectsVisitor**
- class **FindPagedLODsVisitor**
- struct **SortFileRequestFunctor**

Classes

- struct **CompileOperation**
- struct **DatabaseRequest**

6.5 Detailed Description

Database paging class which manages the loading of files in a background thread, and syncronizing of loaded models with the main scene graph.

6.6 Member Function Documentation

virtual DatabasePager* osgDB::DatabasePager::clone () const `[inline, virtual]`

Create a shallow copy on the DatabasePager.

static osg::ref_ptr<DatabasePager>& osgDB::DatabasePager::prototype () `[static]`

get the prototype singleton used by DatabasePager::create().

static DatabasePager* osgDB::DatabasePager::create () `[static]`

create a DatabasePager by cloning DatabasePager::prototype().

virtual void osgDB::DatabasePager::requestNodeFile (const std::string & *fileName*, osg::Group * *group*, float *priority*, const osg::FrameStamp * *framestamp*) `[virtual]`

Add a request to load a node file to end the the database request list.

Implements osg::NodeVisitor::DatabaseRequestHandler.

virtual void osgDB::DatabasePager::run () `[virtual]`

Run does the database paging.

virtual int osgDB::DatabasePager::cancel () `[virtual]`

Cancel the database pager thread.

virtual void osgDB::DatabasePager::clear () `[virtual]`

Clear all internally cached structures.

void osgDB::DatabasePager::setDatabasePagerThreadPause (bool *pause*)

Set whether the database pager thread should be paused or not.

bool osgDB::DatabasePager::getDatabasePagerThreadPause () const `[inline]`

Get whether the database pager thread should is paused or not.

void osgDB::DatabasePager::setAcceptNewDatabaseRequests (bool *acceptNewRequests*) `[inline]`

Set whether new database request calls are accepted or ignored.

bool osgDB::DatabasePager::getAcceptNewDatabaseRequests () const `[inline]`

Get whether new database request calls are accepted or ignored.

int osgDB::DatabasePager::getNumFramesActive () const `[inline]`

Get the number of frames that are currently active.

virtual void osgDB::DatabasePager::signalBeginFrame (const osg::FrameStamp * *framestamp*) `[virtual]`

Signal the database thread that the update, cull and draw has begun for a new frame. Note, this is called by the application so that the database pager can go to sleep while the CPU is busy on the main rendering threads.

virtual void osgDB::DatabasePager::signalEndFrame () `[virtual]`

Signal the database thread that the update, cull and draw dispatch has completed. Note, this is called by the application so that the database pager can go to wake back up now the main rendering threads are iddle waiting for the next frame.

virtual void osgDB::DatabasePager::registerPagedLODs (osg::Node * *subgraph*) `[virtual]`

Find all PagedLOD nodes in a subgraph and register them with the DatabasePager so it can keep track of expired nodes. note, should be only be called from the update thread.

void osgDB::DatabasePager::setDoPreCompile (bool *flag*) `[inline]`

Set whether the database pager should pre compile OpenGL objects before allowing them to be merged into the scene graph. Pre compilation helps reduce the chances of frame drops, but also slows the speed at which tiles are merged as they have to be compiled first.

bool osgDB::DatabasePager::getDoPreCompile () const `[inline]`

Get whether the database pager should pre compile OpenGL objects before allowing them to be merged into the scene graph.

void osgDB::DatabasePager::setTargetFrameRate (double *tfr*) `[inline]`

Set the target frame rate that the DatabasePager should assume. Typically one would set this to the value refresh rate of your display system i.e. 60Hz. Default value is 100. Usage notes. The TargetFrameRate and the MinimumTimeAvailableForGLCompileAndDeletePerFrame parameters are not directly used by DatabasePager, but are should be used as a guide for how long to set aside per frame for compiling and deleting OpenGL objects - ie. the value to use when calling DatabasePager::compileGLObjectgs(state,availableTime,). The longer amount of time to set aside cthe faster databases will be paged in but with increased chance of frame drops, the lower the amount of time the set aside the slower databases will paged it but with better chance of avoid any frame drops. The default values are chosen to achieve the later when running on a modern mid to high end PC. The way to compute the amount of available time use a scheme such as : availableTime = maximum(1.0/targetFrameRate - timeTakenDuringUpdateCullAndDraw, minimumTimeAvailableForGLCompileAndDeletePerFrame).

double osgDB::DatabasePager::getTargetFrameRate () const `[inline]`

Get the target frame rate that the DatabasePager should assume.

void osgDB::DatabasePager::setMinimumTimeAvailableForGLCompileAndDeletePerFrame (double *ta*) `[inline]`

Set the minimum amount of time (in seconds) that should be made available for compiling and delete OpenGL objects per frame. Default value is 0.001 (1 millisecond). For usage see notes in setTargetFrameRate.

double osgDB::DatabasePager::getMinimumTimeAvailableForGLCompileAndDeletePerFrame () const `[inline]`

Get the minimum amount of time that should be made available for compiling and delete OpenGL objects per frame. For usage see notes in setTargetFrameRate.

void osgDB::DatabasePager::setMaximumNumOfObjectsToCompilePerFrame (unsigned int *num*) `[inline]`

Set the maximum number of OpenGL objects that the page should attempt to compile per frame. Note, Lower values reduces chances of a frame drop but lower the rate that database will be paged in at. Default value is 8.

unsigned int osgDB::DatabasePager::getMaximumNumOfObjectsToCompilePerFrame () const `[inline]`

Get the maximum number of OpenGL objects that the page should attempt to compile per frame.

void osgDB::DatabasePager::setExpiryDelay (double *expiryDelay*) `[inline]`

Set the amount of time that a subgraph will be kept without being visited in the cull traversal before being removed.

double osgDB::DatabasePager::getExpiryDelay () const `[inline]`

Get the amount of time that a subgraph will be kept without being visited in the cull traversal before being removed.

void osgDB::DatabasePager::setDeleteRemovedSubgraphsInDatabaseThread (bool *flag*) `[inline]`

Set whether the removed subgraphs should be deleted in the database thread or not.

bool osgDB::DatabasePager::getDeleteRemovedSubgraphsInDatabaseThread () const `[inline]`

Get whether the removed subgraphs should be deleted in the database thread or not.

void osgDB::DatabasePager::setDrawablePolicy (DrawablePolicy *policy*) `[inline]`

Set how loaded drawables should be handled w.r.t their display list/vertex buffer object/vertex array settings.

DrawablePolicy osgDB::DatabasePager::getDrawablePolicy () const `[inline]`

Get how loaded drawables should be handled w.r.t their display list/vertex buffer object/vertex array settings.

void osgDB::DatabasePager::setUnrefImageDataAfterApplyPolicy (bool *changeAutoUnRef*, bool *valueAutoUnRef*) `[inline]`

Set whether newly loaded textures should have their UnrefImageDataAfterApply set to a specified value.

void osgDB::DatabasePager::getUnrefImageDataAfterApplyPolicy (bool & *changeAutoUnRef*, bool & *valueAutoUnRef*) const `[inline]`

Get whether newly loaded textures should have their UnrefImageDataAfterApply set to a specified value.

void osgDB::DatabasePager::setMaxAnisotropyPolicy (bool *changeAnisotropy*, float *valueAnisotropy*) `[inline]`

Set whether newly loaded textures should have their MaxAnisotopy set to a specified value.

void osgDB::DatabasePager::getMaxAnisotropyPolicy (bool & *changeAnisotropy*, float & *valueAnisotropy*) const `[inline]`

Set whether newly loaded textures should have their MaxAnisotopy set to a specified value.

bool osgDB::DatabasePager::requiresUpdateSceneGraph () const

Return true if there are pending updates to the scene graph that require a call to updateSceneGraph(double).

virtual void osgDB::DatabasePager::updateSceneGraph (double *currentFrameTime*) `[inline, virtual]`

Merge the changes to the scene graph by calling calling removeExpiredSubgraphs then addLoadedDataToSceneGraph. Note, must only be called from single thread update phase.

void osgDB::DatabasePager::setCompileGLObjectsForContextID (unsigned int *contextID*, bool *on*)

Turn the compilation of rendering objects for specfied graphics context on (true) or off(false).

bool osgDB::DatabasePager::getCompileGLObjectsForContextID (unsigned int *contextID*)

Get whether the compilation of rendering objects for specfied graphics context on (true) or off(false).

bool osgDB::DatabasePager::requiresExternalCompileGLObjects (unsigned int *contextID*) const

Rerturn true if an external draw thread should call compileGLObjects(..) or not.

bool osgDB::DatabasePager::requiresCompileGLObjects () const

Return true if there are pending compile operations that are required. If requiresCompileGLObjects() return true the application should call compileGLObjects() .

virtual void osgDB::DatabasePager::compileGLObjects (osg::State & *state*, double & *availableTime*) `[virtual]`

Compile the rendering objects (display lists,texture objects, VBO's) on loaded subgraph. note, should only be called from the draw thread. Note, must only be called from a valid graphics context.

virtual void osgDB::DatabasePager::compileAllGLObjects (osg::State & *state*) `[virtual]`

Compile the rendering objects (display lists,texture objects, VBO's) on loaded subgraph. note, should only be called from the draw thread. Note, must only be called from a valid graphics context.

unsigned int osgDB::DatabasePager::getFileRequestListSize () const `[inline]`

Report how many items are in the _fileRequestList queue

unsigned int osgDB::DatabasePager::getDataToCompileListSize () const `[inline]`

Report how many items are in the _dataToCompileList queue

double osgDB::DatabasePager::getMinimumTimeToMergeTile () const `[inline]`

Get the minimum time between the first request for a tile to be loaded and the time of its merge into the main scene graph.

double osgDB::DatabasePager::getMaximumTimeToMergeTile () const `[inline]`

Get the maximum time between the first request for a tile to be loaded and the time of its merge into the main scene graph.

double osgDB::DatabasePager::getAverageTimeToMergeTiles () const `[inline]`

Get the average time between the first request for a tile to be loaded and the time of its merge into the main scene graph.

void osgDB::DatabasePager::resetStats ()

Reset the Stats variables.

void osgDB::DatabasePager::setMaximumNumOfRemovedChildPagedLODs (unsigned int *number*) `[inline]`

Set the maximum number of PagedLOD child to remove per frame

unsigned int osgDB::DatabasePager::getMaximumNumOfRemovedChildPagedLODs () const `[inline]`

Get the maximum number of PagedLOD child to remove per frame

void osgDB::DatabasePager::setMinimumNumOfInactivePagedLODs (unsigned int *number*) `[inline]`

Set the minimum number of inactive PagedLOD child to keep

unsigned int osgDB::DatabasePager::getMinimumNumOfInactivePagedLODs () const `[inline]`

Get the minimum number of inactive PagedLOD child to keep

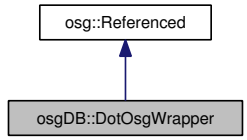
virtual void osgDB::DatabasePager::removeExpiredSubgraphs (double *currentFrameTime*) `[protected, virtual]`

Iterate through the active PagedLOD nodes children removing children which havn't been visited since specified expiryTime. note, should be only be called from the update thread.

void osgDB::DatabasePager::addLoadedDataToSceneGraph (double *currentFrameTime*) `[protected]`

Add the loaded data to the scene graph.

6.7 osgDB::DotOsgWrapper Class Reference



Public Types

- enum **ReadWriteMode** {

  READ_AND_WRITE,

  READ_ONLY }
- typedef std::vector< std::string > **Associates**
- typedef bool(∗ **ReadFunc**)(osg::Object &, osgDB::Input &)
- typedef bool(∗ **WriteFunc**)(const osg::Object &, osgDB::Output &)

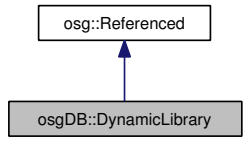
Public Member Functions

- **DotOsgWrapper** (osg::Object ∗proto, const std::string &name, const std::string &associates, ReadFunc readFunc, WriteFunc writeFunc, ReadWriteMode readWriteMode=READ_AND_WRITE)
- const osg::Object ∗ **getPrototype** () const
- const std::string & **getName** () const
- const Associates & **getAssociates** () const
- ReadFunc **getReadFunc** () const
- WriteFunc **getWriteFunc** () const
- ReadWriteMode **getReadWriteMode** () const

6.8 Detailed Description

Wrapper class for specifying read and write functions for extending the .osg file format.

6.9 osgDB::DynamicLibrary Class Reference



Public Types

- typedef void * **HANDLE**
- typedef void * **PROC_ADDRESS**

Public Member Functions

- const std::string & getName () const
- const std::string & getFullName () const
- HANDLE getHandle () const
- PROC_ADDRESS getProcAddress (const std::string &procName)

Static Public Member Functions

- static DynamicLibrary * loadLibrary (const std::string &libraryName)

6.10 Detailed Description

DynamicLibrary - encapsulates the loading and unloading of dynamic libraries, typically used for loading ReaderWriter plug-ins.

6.11 Constructor & Destructor Documentation

osgDB::DynamicLibrary::DynamicLibrary () `[inline, protected]`

disallow default constructor.

osgDB::DynamicLibrary::DynamicLibrary (const DynamicLibrary &) `[inline, protected]`

disallow copy constructor.

osgDB::DynamicLibrary::DynamicLibrary (const std::string & *name*, HANDLE *handle*) `[protected]`

Disallow public construction so that users have to go through loadLibrary() above which returns NULL on failure, a valid DynamicLibrary object on success.

6.12 Member Function Documentation

static DynamicLibrary* osgDB::DynamicLibrary::loadLibrary (const std::string & *libraryName*) `[static]`

returns a pointer to a DynamicLibrary object on successfully opening of library returns NULL on failure.

const std::string& osgDB::DynamicLibrary::getName () const `[inline]`

return name of library stripped of path.

const std::string& osgDB::DynamicLibrary::getFullName () const `[inline]`

return name of library including full path to it.

HANDLE osgDB::DynamicLibrary::getHandle () const `[inline]`

return handle to .dso/.dll dynamic library itself.

PROC_ADDRESS osgDB::DynamicLibrary::getProcAddress (const std::string & *procName*)

return address of function located in library.

static HANDLE osgDB::DynamicLibrary::getLibraryHandle (const std::string & *libraryName*) `[static, protected]`

get handle to library file

DynamicLibrary& osgDB::DynamicLibrary::operator= (const DynamicLibrary &) `[inline, protected]`

disallow copy operator.

6.13 osgDB::ImageOptions::PixelWindow Struct Reference

Public Member Functions

- void **set** (unsigned int x, unsigned int y, unsigned int w, unsigned int h)

Public Attributes

- unsigned int **windowX**
- unsigned int **windowY**
- unsigned int **windowWidth**
- unsigned int **windowHeight**

6.14 Detailed Description

PixelWindow stores the window (in exact pixels) from the overall imagery from which to extract the osg::Image

6.15 osgDB::ImageOptions::RatioWindow Struct Reference

Public Member Functions

- void **set** (double x, double y, double w, double h)

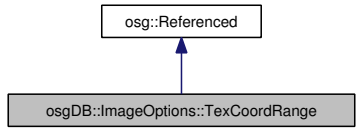
Public Attributes

- double **windowX**
- double **windowY**
- double **windowWidth**
- double **windowHeight**

6.16 Detailed Description

RatioWindow stores the window (as ratios of 0.0 to 1.0) from the overall imagery from which to extract the osg::Image

6.17 osgDB::ImageOptions::TexCoordRange Struct Reference



Public Member Functions

- void **set** (double x, double y, double w, double h)

Public Attributes

- double **_x**
- double **_y**
- double **_w**
- double **_h**

6.18 Detailed Description

Used as UserData attached to generated osg::Image's

6.19 osgDB::Input Class Reference

Public Types

- typedef osg::ArgumentParser::Parameter **Parameter**

Public Member Functions

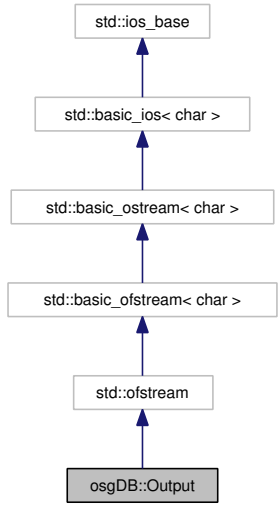
- void **setOptions** (const ReaderWriter::Options *options)
- const ReaderWriter::Options * **getOptions** () const
- virtual osg::Object * **readObjectOfType** (const osg::Object &compObj)
- virtual osg::Object * **readObjectOfType** (const basic_type_wrapper &btw)
- virtual osg::Object * **readObject** ()

- virtual osg::Image * **readImage** ()
- virtual osg::Drawable * **readDrawable** ()
- virtual osg::StateAttribute * **readStateAttribute** ()
- virtual osg::Uniform * **readUniform** ()
- virtual osg::Node * **readNode** ()
- virtual osg::Object * **readObject** (const std::string &fileName)
- virtual osg::Image * **readImage** (const std::string &fileName)
- virtual osg::Node * **readNode** (const std::string &fileName)
- virtual osg::Object * **getObjectForUniqueID** (const std::string &uniqueID)
- virtual void **registerUniqueIDForObject** (const std::string &uniqueID, osg::Object *obj)
- bool **read** (Parameter value1)
- bool **read** (Parameter value1, Parameter value2)
- bool **read** (Parameter value1, Parameter value2, Parameter value3)
- bool **read** (Parameter value1, Parameter value2, Parameter value3, Parameter value4)
- bool **read** (Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5)
- bool **read** (Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6)
- bool **read** (Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6, Parameter value7)
- bool **read** (Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6, Parameter value7, Parameter value8)
- bool **read** (const char *str)
- bool **read** (const char *str, Parameter value1)
- bool **read** (const char *str, Parameter value1, Parameter value2)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3, Parameter value4)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6, Parameter value7)
- bool **read** (const char *str, Parameter value1, Parameter value2, Parameter value3, Parameter value4, Parameter value5, Parameter value6, Parameter value7, Parameter value8)

6.20 Detailed Description

Class for managing the reading of ASCII .osg files.

6.21 osgDB::Output Class Reference



Public Types

- enum **PathNameHint** {
  AS_IS,
  FULL_PATH,
  RELATIVE_PATH,
  FILENAME_ONLY }

Public Member Functions

- **Output** (const char *name)
- void **setOptions** (const ReaderWriter::Options *options)
- const ReaderWriter::Options * **getOptions** () const
- void **setWriteOutDefaultValues** (bool flag)
- bool **getWriteOutDefaultValues** () const
- void **open** (const char *name)
- Output & **indent** ()
- std::string wrapString (const std::string &str)
- void **setIndentStep** (int step)
- int **getIndentStep** () const
- void **setIndent** (int indent)
- int **getIndent** () const

- void **setNumIndicesPerLine** (int num)
- int **getNumIndicesPerLine** () const
- void **moveIn** ()
- void **moveOut** ()
- virtual bool **writeObject** (const osg::Object &obj)
- virtual void **writeBeginObject** (const std::string &name)
- virtual void **writeEndObject** ()
- virtual void **writeUseID** (const std::string &id)
- virtual void **writeUniqueID** (const std::string &id)
- bool **getUniqueIDForObject** (const osg::Object *obj, std::string &uniqueID)
- bool **createUniqueIDForObject** (const osg::Object *obj, std::string &uniqueID)
- bool **registerUniqueIDForObject** (const osg::Object *obj, std::string &uniqueID)
- void **setPathNameHint** (const PathNameHint pnh)
- PathNameHint **getPathNameHint** () const
- virtual std::string **getFileNameForOutput** (const std::string &filename) const
- const std::string & **getFileName** () const
- void **setOutputTextureFiles** (bool flag)
- bool **getOutputTextureFiles** () const
- virtual std::string **getTextureFileNameForOutput** ()

6.22 Detailed Description

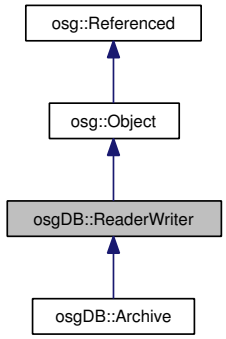
ofstream wrapper class for adding support for indenting. Used in output of .osg ASCII files to improve their readability.

6.23 Member Function Documentation

std::string osgDB::Output::wrapString (const std::string & *str*)

wrap a string with "" quotes and use \" for any internal quotes.

6.24 osgDB::ReaderWriter Class Reference



Public Types

- enum **ArchiveStatus** {
  READ,
  WRITE,
  CREATE }

Public Member Functions

- **ReaderWriter** (const ReaderWriter &rw, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_-COPY)
- **META_Object** (osgDB, ReaderWriter)
- virtual bool **acceptsExtension** (const std::string &) const
- virtual ReadResult openArchive (const std::string &, ArchiveStatus, unsigned int=4096, const Options *=NULL) const
- virtual ReadResult openArchive (std::istream &, const Options *=NULL) const
- virtual ReadResult **readObject** (const std::string &, const Options *=NULL) const
- virtual ReadResult **readImage** (const std::string &, const Options *=NULL) const
- virtual ReadResult **readHeightField** (const std::string &, const Options *=NULL) const
- virtual ReadResult **readNode** (const std::string &, const Options *=NULL) const
- virtual WriteResult **writeObject** (const osg::Object &, const std::string &, const Options *=NULL) const
- virtual WriteResult **writeImage** (const osg::Image &, const std::string &, const Options *=NULL) const
- virtual WriteResult **writeHeightField** (const osg::HeightField &, const std::string &, const Options *=NULL) const
- virtual WriteResult **writeNode** (const osg::Node &, const std::string &, const Options *=NULL) const

- virtual ReadResult **readObject** (std::istream &, const Options *=NULL) const
- virtual ReadResult **readImage** (std::istream &, const Options *=NULL) const
- virtual ReadResult **readHeightField** (std::istream &, const Options *=NULL) const
- virtual ReadResult **readNode** (std::istream &, const Options *=NULL) const
- virtual WriteResult **writeObject** (const osg::Object &, std::ostream &, const Options *=NULL) const
- virtual WriteResult **writeImage** (const osg::Image &, std::ostream &, const Options *=NULL) const
- virtual WriteResult **writeHeightField** (const osg::HeightField &, std::ostream &, const Options *=NULL) const
- virtual WriteResult **writeNode** (const osg::Node &, std::ostream &, const Options *=NULL) const

Classes

- class Options
- class **ReadResult**
- class **WriteResult**

6.25 Detailed Description

pure virtual base class for reading and writing of non native formats.

6.26 Member Function Documentation

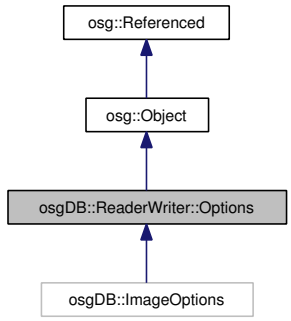
virtual ReadResult osgDB::ReaderWriter::openArchive (const std::string &, ArchiveStatus, unsigned *int* = `4096`, const Options * = `NULL`) const `[inline, virtual]`

open an archive for reading, writing, or to create an empty archive for writing to.

virtual ReadResult osgDB::ReaderWriter::openArchive (std::istream &, const Options * = `NULL`) const `[inline, virtual]`

open an archive for reading.

6.27 osgDB::ReaderWriter::Options Class Reference



Public Types

- enum CacheHintOptions {
  CACHE_NONE,
  CACHE_NODES,
  CACHE_IMAGES,
  CACHE_HEIGHTFIELDS,
  CACHE_ARCHIVES,
  CACHE_OBJECTS,
  CACHE_ALL }

  bit mask for setting up which object types get cached by readObject/Image/HeightField/Node(filename) calls

Public Member Functions

- **Options** (const std::string &str)
- **Options** (const Options &options, const osg::CopyOp &copyop=osg::CopyOp::SHALLOW_COPY)

- **META_Object** (osgDB, Options)
- void setOptionString (const std::string &str)
- const std::string & getOptionString () const
- void setDatabasePath (const std::string &str)
- FilePathList & getDatabasePathList ()
- const FilePathList & getDatabasePathList () const
- void setObjectCacheHint (CacheHintOptions useObjectCache)
- CacheHintOptions getObjectCacheHint () const
- void setPluginData (const std::string &s, void *v) const

- void * getPluginData (const std::string &s)
- const void * getPluginData (const std::string &s) const
- void removePluginData (const std::string &s) const

6.28 Detailed Description

Options base class used for passing options into plugins to control their operation.

6.29 Member Enumeration Documentation

enum osgDB::ReaderWriter::Options::CacheHintOptions

bit mask for setting up which object types get cached by readObject/Image/HeightField/Node(filename) calls

Enumerator:

CACHE_NONE do not cache objects of any type

CACHE_NODES cache nodes loaded via readNode(filename)

CACHE_IMAGES cache images loaded via readImage(filename)

CACHE_HEIGHTFIELDS cache heightfield loaded via readHeightField(filename)

CACHE_ARCHIVES cache heightfield loaded via readHeightField(filename)

CACHE_OBJECTS cache objects loaded via readObject(filename)

CACHE_ALL cache on all read*(filename) calls

6.30 Member Function Documentation

void osgDB::ReaderWriter::Options::setOptionString (const std::string & *str*) `[inline]`

Set the general Options string.

const std::string& osgDB::ReaderWriter::Options::getOptionString () const `[inline]`

Get the general Options string.

void osgDB::ReaderWriter::Options::setDatabasePath (const std::string & *str*) `[inline]`

Set the database path to use a hint of where to look when loading models.

FilePathList& osgDB::ReaderWriter::Options::getDatabasePathList () `[inline]`

Get the database path which is used a hint of where to look when loading models.

const FilePathList& osgDB::ReaderWriter::Options::getDatabasePathList () const `[inline]`

Get the const database path which is used a hint of where to look when loading models.

void osgDB::ReaderWriter::Options::setObjectCacheHint (CacheHintOptions *useObjectCache*) `[inline]`

Set whether the Registry::ObjectCache should be used by default.

CacheHintOptions osgDB::ReaderWriter::Options::getObjectCacheHint () const `[inline]`

Get whether the Registry::ObjectCache should be used by default.

void osgDB::ReaderWriter::Options::setPluginData (const std::string & *s*, void * *v*) const `[inline]`

Sets a plugindata value PluginData with a string

void* osgDB::ReaderWriter::Options::getPluginData (const std::string & *s*) `[inline]`

Get a value from the PluginData

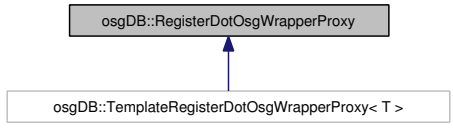
const void* osgDB::ReaderWriter::Options::getPluginData (const std::string & *s*) const `[inline]`

Get a value from the PluginData

void osgDB::ReaderWriter::Options::removePluginData (const std::string & *s*) const `[inline]`

Remove a value from the PluginData

6.31 osgDB::RegisterDotOsgWrapperProxy Class Reference



Public Member Functions

- **RegisterDotOsgWrapperProxy** (osg::Object *proto, const std::string &name, const std::string &associates, DotOsgWrapper::ReadFunc readFunc, DotOsgWrapper::WriteFunc writeFunc, DotOsg-

Wrapper::ReadWriteMode readWriteMode=DotOsgWrapper::READ_AND_WRITE)

6.32 Detailed Description

Proxy class for automatic registration of DotOsgWrappers with the Registry.

6.33 osgDB::RegisterReaderWriterProxy< T > Class Template Reference

Public Member Functions

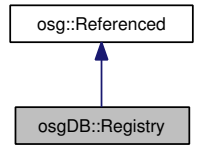
- T * **get** ()

6.34 Detailed Description

template<class T> class osgDB::RegisterReaderWriterProxy< T >

Proxy class for automatic registration of reader/writers with the Registry.

6.35 osgDB::Registry Class Reference



Public Member Functions

- void readCommandLine (osg::ArgumentParser &commandLine)
- void addFileExtensionAlias (const std::string mapExt, const std::string toExt)
- bool readPluginAliasConfigurationFile (const std::string &file)
- void **addDotOsgWrapper** (DotOsgWrapper *wrapper)
- void **removeDotOsgWrapper** (DotOsgWrapper *wrapper)
- void **addReaderWriter** (ReaderWriter *rw)

- void **removeReaderWriter** (ReaderWriter *rw)
- std::string createLibraryNameForFile (const std::string &fileName)
- std::string createLibraryNameForExtension (const std::string &ext)
- std::string createLibraryNameForNodeKit (const std::string &name)
- bool loadLibrary (const std::string &fileName)
- bool closeLibrary (const std::string &fileName)
- void closeAllLibraries ()
- ReaderWriter * getReaderWriterForExtension (const std::string &ext)
- osg::Object * **readObjectOfType** (const osg::Object &compObj, Input &fr)
- osg::Object * **readObjectOfType** (const basic_type_wrapper &btw, Input &fr)
- osg::Object * **readObject** (Input &fr)
- osg::Image * **readImage** (Input &fr)
- osg::Drawable * **readDrawable** (Input &fr)
- osg::Uniform * **readUniform** (Input &fr)
- osg::StateAttribute * **readStateAttribute** (Input &fr)
- osg::Node * **readNode** (Input &fr)
- bool **writeObject** (const osg::Object &obj, Output &fw)
- void setReadFileCallback (ReadFileCallback *cb)
- ReadFileCallback * getReadFileCallback ()
- const ReadFileCallback * getReadFileCallback () const
- ReaderWriter::ReadResult **openArchive** (const std::string &fileName, ReaderWriter::ArchiveStatus status, unsigned int indexBlockSizeHint, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **openArchiveImplementation** (const std::string &fileName, ReaderWriter::ArchiveStatus status, unsigned int indexBlockSizeHint, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readObject** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readObjectImplementation** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readImage** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readImageImplementation** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readHeightField** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readHeightFieldImplementation** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readNode** (const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::ReadResult **readNodeImplementation** (const std::string &fileName, const ReaderWriter::Options *options)
- void setWriteFileCallback (WriteFileCallback *cb)
- WriteFileCallback * getWriteFileCallback ()
- const WriteFileCallback * getWriteFileCallback () const
- ReaderWriter::WriteResult **writeObject** (const osg::Object &obj, const std::string &fileName, const ReaderWriter::Options *options)

- ReaderWriter::WriteResult **writeObjectImplementation** (const osg::Object &obj, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeImage** (const osg::Image &obj, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeImageImplementation** (const osg::Image &obj, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeHeightField** (const osg::HeightField &obj, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeHeightFieldImplementation** (const osg::HeightField &obj, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeNode** (const osg::Node &node, const std::string &fileName, const ReaderWriter::Options *options)
- ReaderWriter::WriteResult **writeNodeImplementation** (const osg::Node &node, const std::string &fileName, const ReaderWriter::Options *options)
- void **setCreateNodeFromImage** (bool flag)
- bool **getCreateNodeFromImage** () const
- void **setOptions** (ReaderWriter::Options *opt)
- ReaderWriter::Options * **getOptions** ()
- const ReaderWriter::Options * **getOptions** () const
- void initFilePathLists ()
- void initDataFilePathList ()
- void setDataFilePathList (const FilePathList &filepath)
- void setDataFilePathList (const std::string &paths)
- FilePathList & getDataFilePathList ()
- const FilePathList & getDataFilePathList () const
- void initLibraryFilePathList ()
- void setLibraryFilePathList (const FilePathList &filepath)
- void setLibraryFilePathList (const std::string &paths)
- FilePathList & getLibraryFilePathList ()
- const FilePathList & getLibraryFilePathList () const
- void updateTimeStampOfObjectsInCacheWithExternalReferences (double currentTime)
- void removeExpiredObjectsInCache (double expiryTime)
- void clearObjectCache ()
- void addEntryToObjectCache (const std::string &filename, osg::Object *object, double timestamp=0.0)
- osg::Object * getFromObjectCache (const std::string &fileName)
- void addToArchiveCache (const std::string &fileName, osgDB::Archive *archive)
- void removeFromArchiveCache (const std::string &fileName)
- osgDB::Archive * getFromArchiveCache (const std::string &fileName)
- void clearArchiveCache ()
- void releaseGLObjects (osg::State *state=0)
- DynamicLibrary * getLibrary (const std::string &fileName)
- void setDatabasePager (DatabasePager *databasePager)
- DatabasePager * getOrCreateDatabasePager ()
- DatabasePager * getDatabasePager ()
- void setSharedStateManager (SharedStateManager *SharedStateManager)
- SharedStateManager * getOrCreateSharedStateManager ()
- SharedStateManager * getSharedStateManager ()
- void addArchiveExtension (const std::string ext)

Static Public Member Functions

- static Registry * **instance** (bool erase=false)

Friends

- struct **ReadFunctor**
- struct **ReadObjectFunctor**
- struct **ReadImageFunctor**
- struct **ReadHeightFieldFunctor**
- struct **ReadNodeFunctor**
- struct **ReadArchiveFunctor**
- class **AvailableReaderWriterIterator**

Classes

- class **ReadFileCallback**
- struct ReadFunctor
- class **WriteFileCallback**

6.36 Detailed Description

Registry is a singleton factory which stores the reader/writers which are linked in at runtime for reading non-native file formats.

The RegisterDotOsgWrapperProxy can be used to automatically register DotOsgWrappers, at runtime with the Registry. A DotOsgWrapper encapsulates the functions that can read and write to the .osg for each osg::Object.

The RegisterReaderWriterProxy can be used to automatically register at runtime a reader/writer with the Registry.

6.37 Constructor & Destructor Documentation

osgDB::Registry::Registry () `[protected]`

constructor is private, as its a singleton, preventing construction other than via the instance() method and therefore ensuring only one copy is ever constructed

6.38 Member Function Documentation

void osgDB::Registry::readCommandLine (osg::ArgumentParser & *commandLine*)

read the command line arguments.

void osgDB::Registry::addFileExtensionAlias (const std::string *mapExt*, const std::string *toExt*)

register an .fileextension alias to mapExt toExt, the later should the the extension name of the readerwriter plugin library. For example to map .tif files to the tiff loader, use addExtAlias("tif","tiff") which will enable .tif to be read by the libdb_tiff readerwriter plugin.

bool osgDB::Registry::readPluginAliasConfigurationFile (const std::string & *file*)

Reads a file that configures extension mappings. File is ASCII text and each line contains the parameyters to the addFileExtensionAlias method. Lines can be commented out with an initial '#' character.

std::string osgDB::Registry::createLibraryNameForFile (const std::string & *fileName*)

create the platform specific library name associated with file.

std::string osgDB::Registry::createLibraryNameForExtension (const std::string & *ext*)

create the platform specific library name associated with file extension.

std::string osgDB::Registry::createLibraryNameForNodeKit (const std::string & *name*)

create the platform specific library name associated with nodekit library name.

bool osgDB::Registry::loadLibrary (const std::string & *fileName*)

find the library in the OSG_LIBRARY_PATH and load it.

bool osgDB::Registry::closeLibrary (const std::string & *fileName*)

close the attached library with specified name.

void osgDB::Registry::closeAllLibraries ()

close all libraries.

ReaderWriter* osgDB::Registry::getReaderWriterForExtension (const std::string & *ext*)

get a reader writer which handles specified extension.

void osgDB::Registry::setReadFileCallback (ReadFileCallback * *cb*) `[inline]`

Set the Registry callback to use in place of the default readFile calls.

ReadFileCallback* osgDB::Registry::getReadFileCallback () `[inline]`

Get the readFile callback.

const ReadFileCallback* osgDB::Registry::getReadFileCallback () const `[inline]`

Get the const readFile callback.

void osgDB::Registry::setWriteFileCallback (WriteFileCallback * *cb*) `[inline]`

Set the Registry callback to use in place of the default writeFile calls.

WriteFileCallback* osgDB::Registry::getWriteFileCallback () `[inline]`

Get the writeFile callback.

const WriteFileCallback* osgDB::Registry::getWriteFileCallback () const `[inline]`

Get the const writeFile callback.

void osgDB::Registry::initFilePathLists () `[inline]`

initilize both the Data and Library FilePaths, by default called by the constructor, so it should only be required if you want to force the re-reading of environmental variables.

void osgDB::Registry::initDataFilePathList ()

initilize the Data FilePath by reading the OSG_FILE_PATH environmental variable.

void osgDB::Registry::setDataFilePathList (const FilePathList & *filepath*) `[inline]`

Set the data file path using a list of paths stored in a FilePath, which is used when search for data files.

void osgDB::Registry::setDataFilePathList (const std::string & *paths*)

Set the data file path using a single string deliminated either with ';' (Windows) or ':' (All other platforms), which is used when search for data files.

FilePathList& osgDB::Registry::getDataFilePathList () `[inline]`

get the data file path which is used when search for data files.

const FilePathList& osgDB::Registry::getDataFilePathList () const `[inline]`

get the const data file path which is used when search for data files.

void osgDB::Registry::initLibraryFilePathList ()

initilize the Library FilePath by reading the OSG_LIBRARY_PATH and the appropriate system environmental variables

void osgDB::Registry::setLibraryFilePathList (const FilePathList & *filepath*) `[inline]`

Set the library file path using a list of paths stored in a FilePath, which is used when search for data files.

void osgDB::Registry::setLibraryFilePathList (const std::string & *paths*)

Set the library file path using a single string deliminated either with ';' (Windows) or ':' (All other platforms), which is used when search for data files.

FilePathList& osgDB::Registry::getLibraryFilePathList () `[inline]`

get the library file path which is used when search for library (dso/dll's) files.

const FilePathList& osgDB::Registry::getLibraryFilePathList () const `[inline]`

get the const library file path which is used when search for library (dso/dll's) files.

void osgDB::Registry::updateTimeStampOfObjectsInCacheWithExternalReferences (double *currentTime*)

For each object in the cache which has an reference count greater than 1 (and therefore referenced by elsewhere in the application) set the time stamp for that object in the cache to specified time. This would typically be called once per frame by applications which are doing database paging, and need to prune objects that are no longer required. Time value is time in sceonds.

void osgDB::Registry::removeExpiredObjectsInCache (double *expiryTime*)

Removed object in the cache which have a time stamp at or before the specified expiry time. This would typically be called once per frame by applications which are doing database paging, and need to prune objects that are no longer required, and called after the a called after the call to updateTimeStampOfObjectsInCacheWithExternalReferences(currentTime). Note, the currentTime is not the expiryTime, one would typically set the expiry time to a fixed amount of time before currentTime, such as expiryTime = currentTime-10.0. Time value is time in sceonds.

void osgDB::Registry::clearObjectCache ()

Remove all objects in the cache regardless of having external references or expiry times.

void osgDB::Registry::addEntryToObjectCache (const std::string & *filename*, osg::Object * *object*, double *timestamp* = 0.0)

Add a filename,object,timestamp tripple to the Registry::ObjectCache.

osg::Object* osgDB::Registry::getFromObjectCache (const std::string & *fileName*)

Get an object from the object cache

void osgDB::Registry::addToArchiveCache (const std::string & *fileName*, osgDB::Archive * *archive*)

Add archive to archive cache so that future calls reference this archive.

void osgDB::Registry::removeFromArchiveCache (const std::string & *fileName*)

Remove archive from cache.

osgDB::Archive* osgDB::Registry::getFromArchiveCache (const std::string & *fileName*)

Get an archive from the archive cache

void osgDB::Registry::clearArchiveCache ()

Remove all archives from the archive cache.

void osgDB::Registry::releaseGLObjects (osg::State * *state* = 0)

If State is non-zero, this function releases OpenGL objects for the specified graphics context. Otherwise, releases OpenGL objexts for all graphics contexts.

DynamicLibrary* osgDB::Registry::getLibrary (const std::string & *fileName*)

get the attached library with specified name.

void osgDB::Registry::setDatabasePager (DatabasePager * *databasePager*) `[inline]`

Set the DatabasePager.

DatabasePager* osgDB::Registry::getOrCreateDatabasePager ()

Get the DatabasePager, creating one if one is not already created.

DatabasePager* osgDB::Registry::getDatabasePager () `[inline]`

Get the DatabasePager. Return 0 if no DatabasePager has been assigned.

void osgDB::Registry::setSharedStateManager (SharedStateManager * *SharedStateManager*) `[inline]`

Set the SharedStateManager.

SharedStateManager* osgDB::Registry::getOrCreateSharedStateManager ()

Get the SharedStateManager, creating one if one is not already created.

SharedStateManager* osgDB::Registry::getSharedStateManager () `[inline]`

Get the SharedStateManager. Return 0 if no SharedStateManager has been assigned.

void osgDB::Registry::addArchiveExtension (const std::string *ext*)

Add an Archive extension.

DynamicLibraryList::iterator osgDB::Registry::getLibraryItr (const std::string & *fileName*) `[protected]`

get the attached library with specified name.

6.39 osgDB::Registry::ReadFunctor Struct Reference

Public Member Functions

- **ReadFunctor** (const std::string &filename, const ReaderWriter::Options *options)
- virtual ReaderWriter::ReadResult **doRead** (ReaderWriter &rw) const =0
- virtual bool **isValid** (ReaderWriter::ReadResult &readResult) const =0
- virtual bool **isValid** (osg::Object *object) const =0

Public Attributes

- std::string **_filename**
- const ReaderWriter::Options * **_options**

6.40 Detailed Description

Functor used in internal implementations.

Index