Open Scene Graph

Marc Schraffenberger
What is it?

- Portable, high level graphics toolkit
- Object oriented framework on top of OpenGL
- Used in development of
  - Flight simulators
  - Games
  - Virtual reality
  - Scientific visualization
How it started

- Started as a hobby by Don Burns in 1998 when porting an application from IRIX to Linux.
- In 1999, Robert Osfield ported it to Windows.
- Later in 1999, the Open Scene Graph library was released as open source.
- In 2001, Robert went fulltime on the library in response to a growing interest.
- Robert has now started consultant services and training for the library.
Who is using OSG?

Diosoft’s game “Frontier”
Who is using OSG?

BARC a virtual reality system
Who is using OSG?

Chalmers University with VR projects
Who is using OSG?

VideaLAB’s model of the Cathedral of Tui
Naming Conventions

- Modules - namespace \( x \), include/\( x \), src/\( x \), lib/lib\( x \).so
- Classes - Capital-Capital \textbf{ie.} \texttt{MatrixTransform}
- Templates - small-small \textbf{ie.} \texttt{ref_ptr}
- Methods - small-Capital \textbf{ie.} \texttt{Node::getStateSet()}
- Attributes - _small-Capital \textbf{ie.} \texttt{Shape::_vertexArray}
- General Variables - small-Capital \textbf{ie.} \texttt{int i}
- Static Variables - s_small-Capital \textbf{ie.} \texttt{s_someName}
Math Conventions

- Right Hand Coordinate System - \textit{like OpenGL}
- Transformation Order - 
  vertex * ( Model * View * Projection )
Distribution

- **doc/** - references guides, install instructions, etc
- **include/** - all include files
- **src/** - source code, including plugins
- **examples/** - all example code
- **Makefile** - unix, cygwin or mingw install script
- **VisualStudio/** - Windows VS install files
- **lib/** and **lib/osgPlugins** - compiled libraries
- **bin/** - demos get installed to
Modules/Libraries

- **osg** - core scene graph
- **osgUtil** - utility library with useful operations and traversers
- **osgDB** - plugin support library for managing dynamic plugins, both loaders and NodeKits
- **osgText** - NodeKit which adds TrueType font rendering support
- **osgParticle** - NodeKit adding particle systems support
Modules/Libraries (Cont.)

- **osgPlugins** - 33 plugins for reading and writing images and 3D databases

- **osgGA** - GUI adapter library which assists in viewer development

- **osgProducer** - windowing/viewer library built on top of OpenProducer

- **osgSim** - NodeKit for visual simulation support

- **osgFX** - NodeKit for special effect support

- **osgGL2** - NodeKit with OpenGL2 support
Modules/Libraries Diagram

- OSG
  - Scene Graph Rendering Elements
- OSGUtil
  - Traversers Enhancements
- OSGDB
  - Data Base Loading Plug-in Management
- Plug-Ins
- OSGText
- OSGSim
- Node Kits...
File Formats Supported

File Formats:

- 3dc
- 3ds
- ac3
- dds
- dw
- flt
- Freetype
- geo
- iv
- ive
- logo
- lwo
- md2
- obj
- osg
- osgtgz
- tgz
- txp
- DirectX
- zip

Image Formats:

- bmp
- dds
- jpeg
- pic
- png
- pnm
- qt
- rgb
- tiff
- tga
Core OSG Classes

- **Helper Classes** - memory management and math classes
- **osg::Nodes** - internal scene graph nodes
- **osg::Drawables** - leaves of scene graph which can be drawn
- **osg::State** - encapsulate OpenGL states
- **Traversers/Visitors** - classes for traversing and operations on the scene
Helper Classes

- **Memory Management**
  - `ref_ptr<>` - smart pointer that automatically ref/unref objects
  - `osg::Referenced` - ref(), unref() & protected virtual destructor
  - `osg::Object` - is the base for a `osg::Referenced` object
Helper Classes (Cont.)

- **Math**
  - `osg::Vec2` - 2D Vector (tex coords)
  - `osg::Vec3` - 3D Vector (coordinates & normals)
  - `osg::Vec4` - 4D Vector (RGBA color)
  - `osg::Quat` - Quaternions (rotations)
  - `osg::Matrix` - 4x4 float matrix (frames)
Node Classes

- **osg::Node** - base class for all internal nodes
- **osg::Group** - base node which can contain children
- **osg::Geode** - leaf node which drawable leaves hang
- **osg::Billboard** - orientates drawables to face viewer
- **osg::LOD** - level of detail node
Node Classes (Cont.)

- **osg::Switch** - controls which children are on or off
- **osg::Sequence** - automatically steps through children
- **osg::LightSource** - positions an osg::Light in scene
- **osg::ClipNode** - positions osg::ClipPlane planes in scene
- **osg::Projection** - overrides projection matrix (useful for HUDs)
Node Classes (Cont.)

- **osg::Transform** - base class for nodes which transform their subgraph
  - **osg::MatrixTransform** - uses a 4x4 matrix to transform
  - **osg::PositionAttitudeTransform** - uses a Vec3 position, Quat rotation, and Vec3 for pivot point
  - **osg::DOFTransform** - OpenFlight style Degree Of Freedom node
Drawables

- **osg::Drawable** - base class providing all draw methods
- **osg::Geometry** - adds real geometry to draw
- **osg::ShapeDrawable** - adds ability to render primitive shapes
- **osg::ImpostorSprite** - used by Impostor node as a billboard
- **osg::DrawPixels** - drawing images with glDrawPixels
State

- **osg::StateSet**
  - can use OpenGL enums modes directly
  - adding general state attribute
  - setting texture modes
  - adding texture attributes

- **osg::StateAttributes** - 34 subclasses covering most OpenGL states and extensions

- **osg::State** - used during draw traversals
Typical Scene Graph Structure

- **osg::Group** - at the top containing whole graph
- **osg::Groups** - LOD, Transform, Switches in middle
- **osg::Geode/Billboard** - leaf nodes which contain...
- **osg::Drawables** - leaves that contain geometry to draw
- **osg::StateSet** - attach to nodes and drawables, state inherits from parent only
Traversing a Scene Graph

- **osg::NodeCallback** - create custom callbacks for update, cull, draw, transform, ... , traversals.

- **osg::NodeVisitor** - traverse entire or sub graph with custom operations for different node types.

  look at osgcallback.cpp in examples/
Code Example: *Textures*

```cpp
osg::Geode* geode = new osg::Geode();
osg::StateSet* stateset = new osg::StateSet();
osg::Image* image = osgDB::readImageFile("lz.rgb");

if (image) {
    osg::Texture2D* texture = new osg::Texture2D;
    texture->setImage(image);
    stateset->setTextureAttributeAndModes(0, texture);
}

geode->setStateSet(stateset);
```
Code Example: *Shapes*

```cpp
osg::Geode* geode = new osg::Geode();
osg::Vec3 position(0.0f, 0.0f, 0.0f);
float radius = 2.0f;
osg::Sphere* sphere = new osg::Sphere(position, radius);

geode->addDrawable(new osg::ShapeDrawable(sphere));
```
Code Example: Callback

class UpdateCallback : public osg::NodeCallback
{
    virtual void operator()(osg::Node* node,
                            osg::NodeVisitor* nv)
    {
        std::cout << "pre traverse" << node << std::endl;
        traverse(node, nv);
        std::cout << "post traverse" << node << std::endl;
    }
};
Code Example: *Callback* (Cont.)

```java
... node.setUpdateCallback(new UpdateCallback());
...
viewer.update();
...
```
Code Example: Visitor

class CallbackVisitor : public NodeVisitor
{
    ...
    virtual void apply(osg::Transform& node)
    {
        node.setUpdateCallback(new UpdateCallback());
        traverse(node);
    }
};
...
rootnode->accept(visitor);
Code Example: Scene

viewer.setSceneData(rootnode);
viewer.realize();

while(!viewer.done())
{
    viewer.sync(); /* wait for all threads */
    viewer.update(); /* update traversal */
    viewer.frame(); /* cull and draw traversal */
}
viewer.sync();
Resources

- **HTML Docs** - doc/index.html
- **Examples** - examples/
- **Source** - include/* src/*
- **Bazaar** - http://openscenegraph.sf.net/bazaar
- **Mailing List** - http://openscenegraph.sf.net/support
- **OpenGL websites and books**
- **Real-Time Rendering** - Akenine-Möller & Haines