Tutorial 6: Defining materials in OSG

Franclin Foping

franclin@netcourrier.com

April 25, 2008

Abstract

In the previous tutorial, we have learnt how to create a shape in OSG. We focused on a rendering a capsule. We have also learnt how to tweak the viewer so that users can get statistics about program at runtime. This tutorial will focus on improving the visual facet of the capsule. By default, OSG fills it with a white color. What about if we want a red one instead?

1 Structure of the tutorial

The tutorial will commence with a description of **StateSets** in section 2. Section 3 will focus on materials, the scene graph of the scene will be presented in section 4. The tutorial will be closed by some exercises in section 6.

2 StateSets

We learnt from the first tutorial that OSG is built on top of OpenGL. We also know that OpenGL is a state machine. Examples of these state information include color of vertices, textures, fog. Well, in OSG all of these are wrapped in a set called *StateSet*. You should note that the state sets are part of the node object are also inherited from the *Referenced* class, so you should create them with smart pointers.

3 Material in OSG

As mentioned previously, OpenGL developers were used to call *glColor, glMaterial* in their application in order to change colour of the geometry. In OSG, all these calls are wrapped within a class called **osg::Material**. At first, you should create a **Material** object, you should also customize it and attach it to the StateSet of the node you want to apply material properties! That is it, isn't it? The following pseudo-code shows it in practice:

```
1 //First, we create the node StateSet object
2 osg::ref_ptr<osg::StateSet> nodess = node->getOrCreateStateSet();
3 //Secondly, the material object
4 osg::ref_ptr<osg::Material> nodematerial = new osg::Material;
5 //Specifying the yellow colour of the object
6 nodematerial->setDiffuse(osg::Material::FRONT,osg::Vec4(.9f,.9f,.2f,1.0f));
7 //Attaching the newly defined state set object to the node state set
8 nodess->setAttribute(mat.get());
```

As you can see on lines 2 and 3, the objects were created by the means of smart pointers. This is because the *Material* and *StateSet* classes are both inherited from the *Referenced* class so you should use smart pointers as mentioned in tutorial 3 to get a code without memory leaks!

4 Scene graph

Surprisingly enough, the scene graph is very similar as to the one of the previous tutorial. This is because we have only added a state set object to the geode, that is it!

5 Conclusion

Congratulations you have just completed another tutorial, if you compile and run the code of this tutorial, you should get the results as in figure 2. This tutorial was certainly the easiest so far, wasn't it? Unfortunately, in a

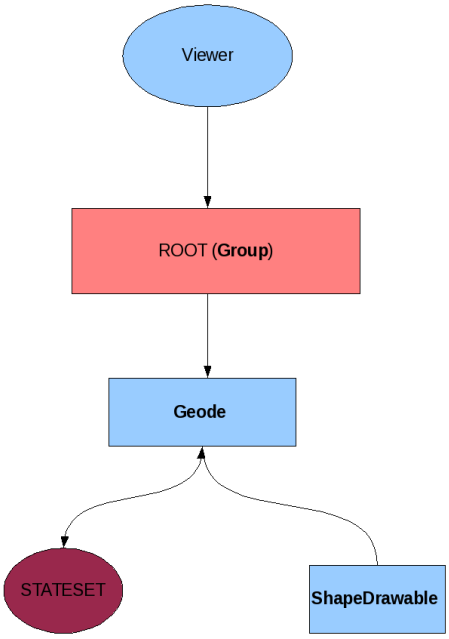

Figure 1: A scene graph

real program you are very unlikely to tweak material properties on objects, however it is good to know these things for they show you how easy is OSG. Next tutorial, we will learn how to add textures on the capsule, after that we are going to start the most interesting part of OSG: animations. For now, go on to the exercise section.

6 Exercise

1. Study the *osg::Material* class carefully and change the code by adding the lighting properties such as: **DIFFUSE, SPECULAR**, etc.

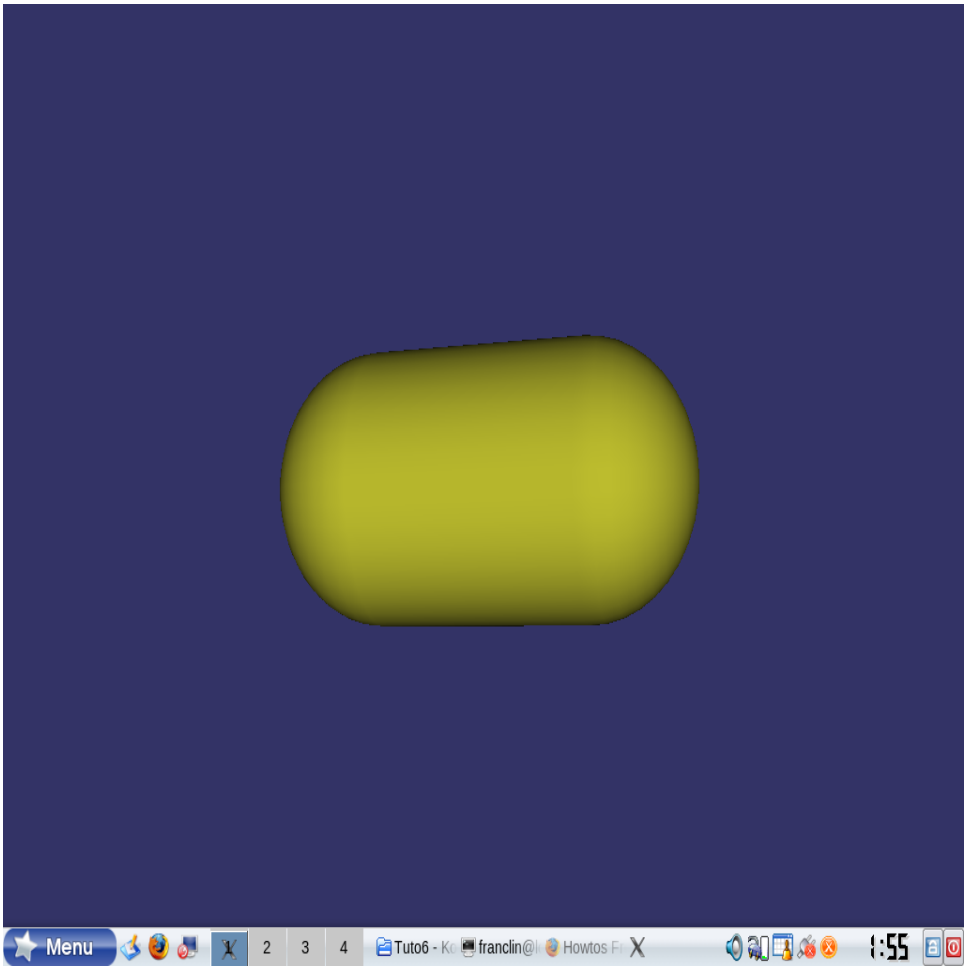

Figure 2: The final scene