Open Scene Graph Lighting

In this exercise you will create a scene populated with simple objects, apply materials to the objects, and add lighting to the scene.

- Create a new Win32 console project called *OSG lighting*
- You will need to set the project file for open scene graph. Remember to do this each time you make a new osg project.
  - under *C++> General* you will need to add the include directory for osg
  - under *Linker>General* you will need to add the library directory for osg
  - under *Linker > Input* you will need to add the osgd.lib osgviewerd.lib osgdbd.lib osgUtild.lib osgGAd.lib osgTextd.lib osgSimd.lib winmm.lib (if you are making a release rather than debug version, take the 'd' off the end of each lib name)
-

Create geometry and display in the viewer

- Set up the initial scene using the following code. You should run the project to test it before going any further.

```
#include "stdafx.h"

#include <osgViewer/Viewer>
#include <osgDB/ReadFile>
#include <osg/Node>
#include <osg/ref_ptr>
#include <osg/LightModel>
#include <osg/Geode>
#include <osg/ShapeDrawable>
#include <osg/Material>
#include <osg/Texture2D>
#include <osg/PositionAttitudeTransform>


int _tmain(int argc, _TCHAR* argv[])
{
	//Declare a node which will serve as the root node
	// the rest of the scene "hangs" on this node

	osg::ref_ptr<osg::Group>  myRoot  = new osg::Group();

	//*************************************************************//

	//********** create the objects and place them on  moveable nodes
**********************//

	osg::ref_ptr<osg::Geode> mySphere = new osg::Geode(); // this is a node to hold
a drawable object
```

```
	mySphere->addDrawable(new osg::ShapeDrawable(new
osg::Sphere(osg::Vec3(0.0,0.0,0.0), 0.25))); // the first vector is the centre
position, the second parameter is the radius

	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere1Xform = new
osg::PositionAttitudeTransform(); // this creates a moveable (transform) node
	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere2Xform = new
osg::PositionAttitudeTransform();
	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere3Xform = new
osg::PositionAttitudeTransform();
	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere4Xform = new
osg::PositionAttitudeTransform();
	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere5Xform = new
osg::PositionAttitudeTransform();
	osg::ref_ptr<osg::PositionAttitudeTransform>  sphere6Xform = new
osg::PositionAttitudeTransform();

	sphere1Xform->addChild(mySphere.get()); // add the sphere to the transform node
	sphere2Xform->addChild(mySphere.get()); // notice you can add the same object
to multiple nodes
	sphere3Xform->addChild(mySphere.get());
	sphere4Xform->addChild(mySphere.get());
	sphere5Xform->addChild(mySphere.get());
	sphere6Xform->addChild(mySphere.get());

	myRoot->addChild(sphere1Xform.get()); // adds it all to the root node
	myRoot->addChild(sphere2Xform.get()); // adds it all to the root node
	myRoot->addChild(sphere3Xform.get()); // adds it all to the root node
	myRoot->addChild(sphere4Xform.get()); // adds it all to the root node
	myRoot->addChild(sphere5Xform.get()); // adds it all to the root node
	myRoot->addChild(sphere6Xform.get()); // adds it all to the root node

		sphere1Xform->setPosition(osg::Vec3(-1,5,0.5 )); // set the desired xyz
position of the sphere in the scene
	sphere2Xform->setPosition(osg::Vec3(0,5,0.5 ));
	sphere3Xform->setPosition(osg::Vec3(1,5,0.5 ));
	sphere4Xform->setPosition(osg::Vec3(-1,5,-0.5 ));
	sphere5Xform->setPosition(osg::Vec3(0,5,-0.5 ));
	sphere6Xform->setPosition(osg::Vec3(1,5,-0.5 ));


	//*************************************************************//
```

Note that in the above code section you make the 6 transform nodes one at a time. It might be more efficient to use a loop to make them.

Also, note that we only make ONE sphere, and add copies of it to all 6 transform nodes.

```
	//*******set up the viewer to display your scene***************//

	osgViewer::Viewer viewer; // Declare a 'viewer' which will display the scene

	viewer.setSceneData( myRoot ); //assign the scene graph we created above to
this viewer

	 viewer.setLightingMode(osg::View::LightingMode::HEADLIGHT); // give the scene
some lighting
```

```
	viewer.getCamera()->setClearColor(osg::Vec4(0,0,1.0,1.0)); // sets background
colour of scene


	viewer.getCamera()-
>setViewMatrixAsLookAt(osg::Vec3(0.0,0.0,1.6),osg::Vec3(0.0,5.0,1.0), osg::Vec3(0.0,
0.0, 1.0));


	//*************************************************************//


	viewer.realize(); // start the viewer

	while (!viewer.done()) // until the end of the program
	{
		viewer.frame(); // update to next frame

	}

	return 0;
}
```

- You should see 6 spheres displayed in the viewport, but they don't yet have any material applied to them.

Add materials to the geometry

- Add the following code into your project. Compile and run.

```
sphere6Xform->setPosition(osg::Vec3(1,5,0.5 ));


	//*************************************************************//

	//********** create a material and apply it to the shapes
**********************//

	// create the material
	osg::ref_ptr<osg::Material> sphereMaterial = new osg::Material;

	sphereMaterial->setDiffuse(osg::Material::FRONT , osg::Vec4(0.6, 0.3, 0.5,
1.0));
	sphereMaterial->setSpecular(osg::Material::FRONT, osg::Vec4(0.0, 0.0, 0.0,
1.0));
	sphereMaterial->setAmbient(osg::Material::FRONT,  osg::Vec4(0.1, 0.1, 0.1,
1.0));
	sphereMaterial->setEmission(osg::Material::FRONT, osg::Vec4(0.0, 0.0, 0.0,
1.0));
	sphereMaterial->setShininess(osg::Material::FRONT, 25.0);

	//apply the material to each sphere
	sphere1Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
	sphere2Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
	sphere3Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
	sphere4Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
```

```
    sphere5Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
    sphere6Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);


    //*************************************************************//
//*******set up the viewer to display your scene***************//

    osgViewer::Viewer viewer; // Declare a 'viewer' which will display the scene
```

You should read up about the material properties and settings in more depth and then experiment with them.

Add two spotlights to the scene

- You are going to add two spotlights, lighting spheres 1 and 6. You will see that the default lighting will disappear, and only these two spheres are clearly lit.

```
sphere5Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);
    sphere6Xform->getOrCreateStateSet()->setAttribute(sphereMaterial);


    //*************************************************************//

    //********** Create Lights ************//
    /* You need to switch on the lighting in your root state set, then you need to
create a light, add it to a light source, and then attach it to your scene graph.
*/

    // create the lighting state set
    osg::StateSet* state = myRoot->getOrCreateStateSet();
    state->setMode( GL_LIGHTING, osg::StateAttribute::ON );
    state->setMode( GL_LIGHT0, osg::StateAttribute::ON ); // for first light
    state->setMode( GL_LIGHT1, osg::StateAttribute::ON ); // for second light

    // Create a white spot light
    osg::ref_ptr<osg::Light> light0 = new osg::Light;
    light0->setPosition( osg::Vec4( -1, 0, 0.5, 1.0 ));
    light0->setDirection( osg::Vec3(0,1,0)); // values between 0 and 1 to set the
axis of direction for the vector the light travels - use this if you want a spotlight
rather than a point light
    light0->setSpotCutoff(15 ); // spot cutoff gives us the size of the cone. 15
will give us a 30 degree cone. The max setting is 90 (180 degree cone)

    // create a light source object and add the light to it
    osg::ref_ptr<osg::LightSource> source0 = new osg::LightSource;
    source0->setLight( light0.get() );
    myRoot->addChild(source0);

    // and another light
    osg::ref_ptr<osg::Light> light1 = new osg::Light;
    light1->setAmbient( osg::Vec4( .1, .1, .1, 1 ));
    light1->setPosition( osg::Vec4( 1, 0, -0.5, 1.0 ));
```

```
	light1->setDirection( osg::Vec3( 0,1,0)); //point towards sphere 6
	light1->setSpotCutoff(10 ); // you will notice this cone is bigger and touches
adjacent objects
	light1->setLightNum( 1 ); // light number can be 0 - 7.  It is 0 by default

	// create another light source object and add the light to it
	osg::ref_ptr<osg::LightSource> source1 = new osg::LightSource;
	source1->setLight( light1.get() );
	myRoot->addChild(source1);



	//*************************************************************//


		//*******set up the viewer to display your scene***************//

	osgViewer::Viewer viewer; // Declare a 'viewer' which will display the scene
```

- Experiment with some of these settings

```
	light0->setAmbient(osg::Vec4d(0, 0.5, 0.5,0.5)); // first 3 values are RGB, the
last is 0 for off 1 for fully on

	light0->setDiffuse(osg::Vec4d(0, 0, 1.0, 0.5)); // ambient light is non-
directional "scatter", diffuse is the rays from the light

	light0->setSpecular(osg::Vec4d(0, 1.0, 0, 1.0)); // specular gives us the
amount of shininess of the light

	light0->setConstantAttenuation(0.1); // a value close to 0 makes an intense
light, close to 1 a much less intense - affects the overall intensity, regardless of
distance

	light0->setLinearAttenuation(0.5); // reduces linearly with distance from the
object

	light0->setQuadraticAttenuation(0.5); // reduces in proportion with the inverse
square of the distance from the light (similar to real world)
```

Now you have textured and lit a simple scene, experiment with attributes of the materials and the lights, then create a new scene, with different materials and some creative lighting.