Open Scene Graph Collisions

In this exercise you will learn how to handle collisions between objects in your scene. We are going to use simple collision detection between bounding boxes.

- Create a new Win32 console project called *OSG collisions*
- Set the project properties as in the lighting tutorial

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/PositionAttitudeTransform>
#include <osg/ref_ptr>


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();

	//*************************  load two objects in the scene
*********************//

	osg::ref_ptr<osg::Node> blueHex = osgDB::readNodeFile("models/lBlueHex.3ds");
	osg::ref_ptr<osg::PositionAttitudeTransform>  blueXform = new
osg::PositionAttitudeTransform();
	blueXform->setPosition(osg::Vec3(-0.2,2,0));
	blueXform->addChild(blueHex.get());
	myRoot->addChild(blueXform.get()); // adds it all to the root node

	osg::ref_ptr<osg::Node> redHex = osgDB::readNodeFile("models/redHex.3ds");
	osg::ref_ptr<osg::PositionAttitudeTransform>  redXform = new
osg::PositionAttitudeTransform();
	redXform->setPosition(osg::Vec3(0.2,2,0));
	redXform->addChild(redHex.get());
	myRoot->addChild(redXform.get()); // adds it all to the root node

		//*******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(1,1,1,1.0)); // sets background
colour of scene
```

```
	viewer.getCamera()->setViewMatrixAsLookAt(osg::Vec3(0.0,0.0,0),osg::Vec3(0.0,1.0,0), osg::Vec3(0.0, 0.0, 1.0));


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

	viewer.realize(); // start the viewer

	//************ add a title to the window ***********************//

	typedef osgViewer::Viewer::Windows Windows;
	Windows windows;

	viewer.getWindows(windows);

	windows[0]->setWindowName("UoP osg colllision example");


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

	}


	return 0;
}
```

- Test and run the project. At the moment you should see a red and blue box static on the screen.

Set up a function to move the boxes

- Now we will create a simple function to move the blue box towards the red one. Add this function definition to the top of the file:

```
#include <osg/ref_ptr>

void moveBox (osg::ref_ptr<osg::PositionAttitudeTransform> blueBox, osg::ref_ptr<osg::PositionAttitudeTransform> redBox); // this function is  going to move one of the boxes towards the other until we get a hit

int _tmain(int argc, _TCHAR* argv[])
```

- And add this to the bottom of the code:

```
while (!viewer.done()) // until the end of the program
	{
		moveBox(blueXform, redXform);
		viewer.frame(); // update to next frame

	}

	return 0;
}
```

```
void moveBox (osg::ref_ptr<osg::PositionAttitudeTransform> blueBox,
osg::ref_ptr<osg::PositionAttitudeTransform> redBox)
{
	osg::Vec3d boxPos = blueBox->getPosition();
	boxPos[0] +=0.002;
	blueBox->setPosition(boxPos);
}
```

- If you test the project you will see that the blue box moves towards the red box and passes right through it. Next we need to avoid a collision.

Create a function to detect a collision

- We are going to make a function to test for a collision, and then create a response to the collision so that we can see something has happened. Add the following code to your function (you could make a separate function for the collision, especially if you are wanting to test for multiple collisions in a project).

```
void moveBox (osg::ref_ptr<osg::PositionAttitudeTransform> blueBox,
osg::ref_ptr<osg::PositionAttitudeTransform> redBox)
{
    osg::Vec3d boxPos = blueBox->getPosition();
    boxPos[0] +=0.002;
    blueBox->setPosition(boxPos);
    const osg::BoundingSphere& bs1 = blueBox->getBound();
    const osg::BoundingSphere& bs2 = redBox->getBound();
    if(bs1.intersects(bs2))
    {
           boxPos[0] -= 0.1;
           blueBox->setPosition(boxPos); // bounce the box left after a collision
           redBox->setScale(redBox->getScale()*0.9); // decrease the size of the
red box after each hit
    }
}
```

You will notice that the boxes detect a hit before they appear to touch each other. This is because of the way bounding spheres are defined. It is generally considered acceptable but as you can see may appear slightly odd when viewed directly from the side. You may want to research more precise collision detection later on.

See if you can create a project with some more interesting collision behaviour.