Open Scene Graph Animation

In this exercise you will animate an object directly, and also use an animation path to move an object around your scene. I have re-used the blue hex from previous sessions, but of course you could use any compatible model you wish.

- Create a new Win32 console project called *OSG animation*
- Use the same 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 an object in the scene
*********************//

	osg::ref_ptr<osg::Node> blueHex = osgDB::readNodeFile("models/1BlueHex.3ds");
	osg::ref_ptr<osg::PositionAttitudeTransform>  hexXform = new
osg::PositionAttitudeTransform();
	hexXform->setPosition(osg::Vec3(0.0,1,0));
	hexXform->addChild(blueHex.get());
	myRoot->addChild(hexXform.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,-
1 ,0),osg::Vec3(0.0,1.0,0), 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;
}
```

- Now we are going to add a title to the window:

```
viewer.realize();

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

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

viewer.getWindows(windows);

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

	while (!viewer.done()) // until the end of the program
	{
```

Create a function to rotate the object

You are going to use a function to turn the object. Whilst a function is not strictly necessary for such a simple application, it is good practice to do this.

- Add the following function definition before your main program:

```
#include <osg/ref_ptr>

void turnObject (osg::ref_ptr<osg::PositionAttitudeTransform> objectToTurn); //
function to produce rotation of  object every frame

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

- Now add the function at the end of your program:

```
	return 0;
}
```

```
////////////////////////////////// functions/////////////////////

void turnObject (osg::ref_ptr<osg::PositionAttitudeTransform> objectToTurn) // rotates
the object a set amount each frame around the z axis
{

       osg::Quat::value_type tempAngle; // holds the angle in radians to rotate the
objects
       osg::Vec3 tempVector; // the axis of rotation
       objectToTurn->getAttitude().getRotate(tempAngle, tempVector); //  get initial
angle from the PAT node
       tempAngle += 0.01;

       if(tempAngle>0.999) // set it back to zero after each full rotation
       {
              tempAngle = 0;
       }
       objectToTurn->setAttitude(osg::Quat(tempAngle,tempVector)); // set the new
angle back to the PAT node

} // end of turnObject
```

- Next you need to add a call to this function. We will put the call in the viewer loop so that it is called every frame

```
       while (!viewer.done()) // until the end of the program
       {
              turnObject(hexXform); // calls a function to rotate the object each
frame
              viewer.frame(); // update to next frame

       }
```

- Now you can build and test this stage. You should see the hex box rotating constantly. Try changing the rate of rotation, or adding a more complex formula to move the object around.

Import an animation path.

Now we are going to import an animation path to move the object around the scene. Animation paths can be written directly into a file, or created via programs such as 3DS Max. The one we are using was recorded from 3DS Max using the MS file *recordPosition.ms*.

- Add the following code to your file. This loads in an animation path and assigns it to an OSG animation path. It may look complicated, but if you read it section by section you should be able to work out what each part does. Take particular care with the file open line - this is set up in an "if" statement so that the rest of the program will only execute if the file is successfully opened.

```
    myRoot->addChild(hexXform.get()); // adds it all to the root node

        //****************************************************************************/
    /
        //*************** load and run animation path************************//
        // set up the animation path

         osg::AnimationPath* animationPath = new osg::AnimationPath;
        animationPath->setLoopMode(osg::AnimationPath::NO_LOOPING);
        osg::ref_ptr<osg::AnimationPathCallback> hexCallback;

        FILE *readFile;
        if (readFile = fopen("animation/anim path2.txt", "r"))// checks that the files
has opened before running program
        {  // this opens the 'if' statement which is closed after the viewer loop

        // read and process each line of data
        char dataString[200];
        char * tokenString; // the token in this case is a tab character which delimits
    the columns
        int columnCount =1; // keeps count of the data columns to ensure the correct
    ones are processed
        int frameCount = 1;
        double xValue = 0;
        double yValue = 0;
        double zValue = 0;

        osg::Vec3 nextPos  = osg::Vec3d(0.0,1,0);;

        osg::AnimationPath::ControlPoint* myControl = new
    osg::AnimationPath::ControlPoint();

                while(! feof(readFile)) // continue until the end of the file is reached
                {
                        fgets(dataString, 200, readFile); // read in the next line of
    data from the file

                        tokenString = strtok (dataString,"\t"); // break it up into
    columns by identifying where the tabs appear

                        while (tokenString != NULL) // while there is still content to
    read from the line of data

                        {
                                switch(columnCount) // processes the x y and z data
                                {
                                case 1:
                                        xValue = atof(tokenString); // record the x column
    as a double

                                        break;
                                case 2:
                                        yValue = atof(tokenString); // record the y column
    as a double

                                        break;
                                case 3:
                                        zValue = atof(tokenString); // record the z column
    as a double

                                        break;
                                        break;
                                }
                                columnCount ++;
                                tokenString = strtok (NULL, "\t"); // set the pointer to
    the next block

                        } // end of while for processing one line
                        columnCount = 1;

                        nextPos[0] = xValue; // update the xyz position
                        nextPos[1] = yValue;
                        nextPos[2] = zValue;

                        myControl->setPosition(nextPos);
                        animationPath->insert(frameCount, *myControl);
                        frameCount++;
                        printf("%d\n", frameCount);
                }
                hexCallback = new osg::AnimationPathCallback(animationPath,0.0, 25.0);
                hexXform->setUpdateCallback(hexCallback);


                fclose(readFile); // close the animation read file

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



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


        osgViewer::Viewer viewer; // Declare a 'viewer' which will display the scene
    ..
    ..
    ..
    ..
    ..
while (!viewer.done()) // until the end of the program
        {
                //turnObject(hexXform); // calls a function to rotate the object each
frame
                viewer.frame(); // update to next frame

        }
                } // end if for file open
        else
        {
                printf("Can't find file\n");
                }

        return 0;
```

- Test the project. You may need to move your camera position back to view the entire animation; else the object will disappear off your screen at times. You will notice that the rotation animation you wrote earlier isn't running. The animation path has overridden it. You can combine the two, but this is a little more complex. Feel free to try!

Now try creating a simple animated model in 3DS max, and exporting both the model (in .3ds form) and it's animation path. Can you display it in OSG? For more complex animating you might want to try including

```
myControl->setScale(osg::Vec3(value,value,value));
```