Assignment 2: Modeling and Scene Setting

Out: 26th September 2017 **Due: 7th October 2017 at 8:59pm**

In this assignment, you will write a program that uses basic 3D shapes to set up a scene. You will also create a new 3D shape which will be a "surface of revolution.

Starting code

You have been given some starting code in the form of an IntelliJ/Qt project. Please download this code and make sure you can build and run it.

When you run the program, you will see a sphere, box and cone on the screen.

Part 1: Stage a scene (30 points)

In this part, you will write code to set up a similar but bigger scene using the given mesh files. You are free to set up whatever objects you like, so long as it is a meaningful scene with the following:

1. The scene must be compromised of a room. ~~The room has a floor, at least four walls and a ceiling.~~ The room has rectangular floor plan, a floor and three walls.
2. The room should contain a table. The table must have at least 4 legs and a rectangular or oval desktop. All the objects must be kept on this table.
3. In all, your scene should contain the room, the table and at least 2 objects on the table. Each "object" kept on the table must be something more interesting than just one of the basic shapes. That is, just keeping one sphere on the table by itself does not count as an object. Try to be creative!
4. Provide a different color to each object so that viewing is easier.

You may find it convenient to render in wire frame mode so that debugging your scene setup is easier.

You must place the camera so that you can see the room and its contents clearly. Look at how the draw function currently sets the camera.

Part 2: Create new object (30 points)

In this part, you must create a triangle mesh for a new type of object. This object must be a closed shape, must be physically constructible and must be a "surface of revolution".

A surface of revolution is a surface obtained by rotating a 3D curve about an axis. For example, a sphere is a surface of revolution because it can be obtained by drawing a semicircle in the X-Y plane such that its diameter is along the Y axis, and rotating it about the Y axis. Similarly, some chess pieces (e.g. pawn) can be obtained by drawing a similar curve that starts and ends on the Y axis in the X-Y plane, and rotating it about the Y axis.

How to achieve this

You must write a separate program that creates this shape, creates a PolygonMesh object out of it, and exports it to a ".obj" file using the OBJExporter class provided to you. Then, you can use your above program to read and render it like any other object instance. This small program will not use OpenGL or even open a window. It simply creates and exports an obj file.

Incorporate at least one instance of this object in your scene! This will be the 3rd object on the table.

Part 3: Trackball (30 points)

In this part, you will use the mouse to implement a track-ball interface in your program. To imagine how this should behave: imagine that your entire scene is enclosed in an invisible glass sphere. The scene is attached to the inside walls of this sphere using very thin wires, so that as you move the sphere like a globe, the entire scene moves with it.

You will "move" the trackball using the mouse (press, drag, release). Some examples of how your scene should behave with a trackball:

1. If you drag the mouse "up to down", the scene should rotate counter-clockwise about the +X axis. "Down to up" will make it go reverse.
2. If you drag the mouse "left to right", the scene should rotate counter-clockwise about the +Y axis. "Right to left" will make it go reverse.
3. A general drag of the mouse will cause the scene to rotate along the corresponding axis that depends on the drag direction.
4. The rotations should be cumulative. That is, subsequent drags should start at the last orientation and not reset the scene.
5. To control the speed of rotation, you can adjust the radius of this trackball. That is, a bigger glass sphere will require more movement to produce the same rotation and thus will be "slower".
6. You should not assume that the mouse is dragged in a straight line. A user can press the mouse button, then drag it around non-linearly before releasing the button. In this case the rotations are cumulative from each mouse position to the next one.

The View class already has a "trackballTransform" variable that should encode the transformations caused by the trackball. The "display" function currently uses it, and the main function is relaying mouse coordinates to the View class. Thus, your job is to simply fill in the details in the mouse functions of the View class.

Contents of the README file (10 points)

1. Explain how a user of your program will use it. This part should assume that the user is running your program through IntelliJ or Qt Creator. It should also point out where in your code (a) the scene is set up and (b) the user can change the camera before running the program and (c) how you have implemented the trackball.
2. Explain first in prose, and then in pseudo-code how to create an object of revolution. It should begin by discussing how one might identify objects of revolution in the real world (visually, not mathematically or technically. Imagine a reader has never heard of the term "object of revolution" before)

Program details

Prepare your program so that when we build and run, the entire scene should be visible upright. The mouse should work as a trackball on the entire scene (including the room, table and objects).

What to submit

Submit the entire IntelliJ project for your OpenGL program, the project that generates the object of revolution and your README file in a single zipped file.