**CSCE A385 Programming Assignment 5**

Computer Graphics

Shanan Almario

**Program Notes**

IDE: Visual Studios 2015

**Problem 1 Code Description**

The code outputs three objects and 3 lights. The objects are a cylinder with a brick texture, sphere with a gold texture, and ellipsoid with a copper texture, and the lights are a stationary light, orbiting light, and a moving spotlight attached to the camera.

**Interaction:**

*Keyboard*:

x: will move the space counter clockwise on the x-axis

X: will move the space clockwise on the x-axis

c: will move the space counter clockwise on the y-axis

C: will move the space clockwise on the y-axis

z: will move the space counter clockwise on the z-axis

Z: will move the space clockwise on the z-axis

q/Q: hides the current window being viewed

ESC: closes all windows

*Spec Key:*

Up: move closer to objects

Down: move away from objects

Right: move right

Left: move left

*Mouse*:

Pressing the left mouse button opens a window for just the cylinder.

Pressing the right mouse button opens a window for just the sphere.

Pressing the middle mouse button opens a window for just the ellipse.

**Problem 1 and 2**

The assignment was to draw a sphere with a gold texture, an ellipsoid with a copper texture, and a cylinder with a brick texture under three lighting scenarios, and then when each object is clicked, it’s supposed to open a window so that only the object is shown. I was unable to figure out how to implement picking, so I just made it so that each window for each isolated object will appear by left click, right click, and middle click.

This assignment was structured similar to the cannon assignment where each object that is not a copy of another is a sibling in the tree. Those that are a copy of an object are children. All siblings have their own draw and build methods. When isolating the objects, another display method was created where a new node is created containing the isolated object without siblings, and only that is displayed.

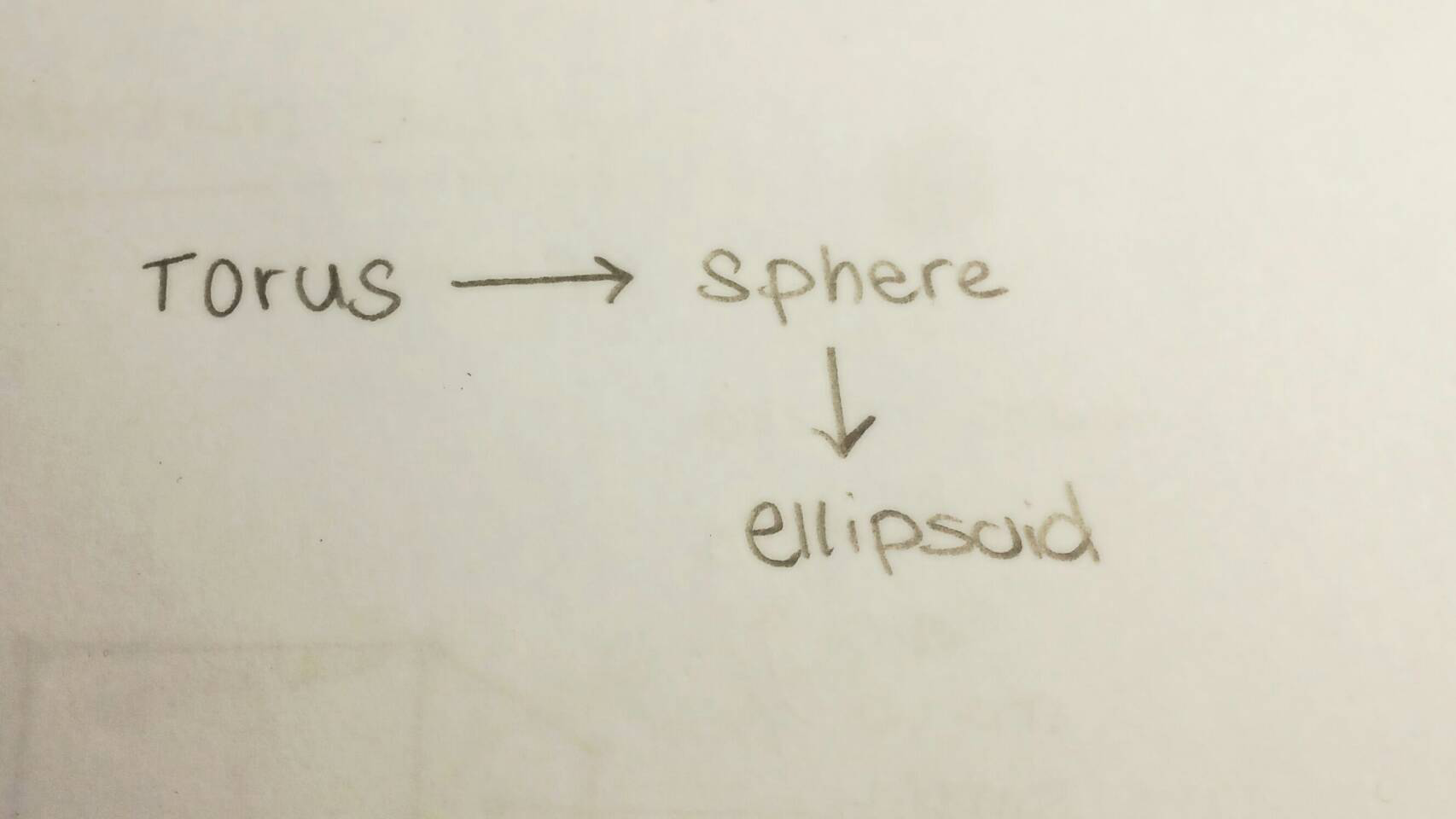
Textures were added to the code by creating extra buffer space for the textures, and a texture position to pass into the shaders, which were also changed by adding texture variables. Texture coordinates uses the normal coordinates of the objects. The sphere and ellipse use the x, and z coordinates of the normal while the cylinder uses the x, z from the normal coordinates for the top and bottom covers, and x, y from the normal coordinates for the rest. Texture buffers and binding inside the init function reference the TM\_stub\_Windows code provided by the instructor.

To create the separate windows, the build functions were put in a separate init function so that they could be called separately and only once so that the code doesn’t have to initialize the build functions again when trying to open another window.

The difference with the lighting that was made from assignment 4 is that all the objects were changed so that they are white objects now. Also, the different textures were activated in the draw methods. The following will be the description of how the lights were organized from assignment 4. The lights were called in the draw methods. Each object had their own light object placed at the same spot as the other objects for each of the three lights. The light variables were organized in two structs, lightData and materialData. The lightData struct held the light color data and light position data, while the materialData only held the material and shininess data. Each of the different light objects have their own computation and initialization functions, and each object have their own initialization functions for the lights. The objects shared the computation function for their lights.

The orbiting light was created by referencing the example code from the shaded sphere. The stationary light was created by using the same structure as the orbiting light, but the light\_position does not change. The moving spotlight attached to the camera references the example code given in the phong torus code, and follows its example in the shader. Its position is equal to the eye view. There is a small problem with this light because the specular light can be seen, but the ambient light doesn’t seem to follow the specular light. This is probably because there isn’t a calculation for the ambient light to change.

The resize routine will change the image of the scene so that it matches the size of the window.

*Scene Structure Tree*

**Cylinder**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**