**7-1 Final Project**

Noah Coleman

Computer Science, Southern New Hampshire University

CS-330-T2834

Professor Rodriguez

December 11, 2022

**7-1 Project Two**

**Development Justification**

When considering a 3D scene to study for this course I looked around my home for inspiration. I knew the intended result was for us to have multiple objects constructed using various primitive shapes. In setting up my scene I looked for objects I thought I could build in a virtual space. The golf ball would be built using a sphere, the candle a cylinder, the matches and pepper shaker using cubes, and the desk using a plane. I looked at these objects and felt they would present an interesting, but not impossible, project deliverable.

I was able to program these required shapes by following the guidelines of the OpenGL website. Vries (n.d.) talks about how complex shapes are created using a system of triangles. These triangles are programed using vertices. The more triangles you have the smoother a shaper looks. Most shapes can be made in this basic manner. For the cube and the cylinder, I was able to write the vertices directly. I passed them to the GPU in a certain order, and they displayed the expected shapes. For the sphere, I was able to utilize a function to calculate the vertices for me. This way the sphere looks almost completely smooth as there are many more triangles than I could possibly write by hand.

**3D Scene Navigation**

To navigate the 3D scene, it was important to be able to access the camera using available input devices. For this project the input devices were a keyboard and a mouse. The mouse was the most useful input for directing the directional viewpoint of the camera. The sweeping motion helps direct the camera where to look. The keyboard is more effective for moving the camera in the X, Y, or Z direction. Being able to move up, down, left, right, in, and out make the scene a treat to navigate. The W, A, S, D controls are very common in gaming controls, so they were an easy selection for navigating the scene. Q and E are located nearby, so they were a good accompaniment for up and down. It was also important to have a steady movement speed for the camera. Being able to smoothly navigate around the scene and observe every angle makes it more interesting.

**Custom Functionality**

There were two ways I made my code more modular and organized. One was by utilizing a function to render a sphere. The renderSphere function was pivotal to achieving my golf ball. When called, the function creates the necessary vertices to render a sphere in real time. It uses a for loop to calculate each X, Y, and Z coordinates, and then add them to the array of vertices. It also applies texture to the object. To render the sphere on the screen, I simply need to assign a shader, call the function, and then draw the triangles. With the proper shader activated the sphere renders using that shaders specifications. I could render different spheres with different shaders for different effects.

The other way I organized my code was by reusing the same cube vertices to render different objects. The vertices for a basic cube were assigned to a mesh. This mesh was then called each time I wanted to create a cube. To make the objects look different and appear in different places I simply had to modify them individually. By applying texture, then manipulating the transformation, scale and rotation, each cube becomes a different and independent object. Reusing this mesh made it much easier to render multiple cubes at once.

References

Vries, J. (n.d.). *Hello triangle*. LearnOpenGL. [https://learnopengl.com/Getting-started/Hello-Triangle](https://learnopengl.com/Getting-started/Hello-Triangle%20)