**Project 1 Reflection**

Victoria Kaloudis

Southern New Hampshire University

CS-330-H7827

Professor Kurt Diesch

October 20, 2022

**Project 1 Reflection**

Over the last six weeks, I have been preparing code for a 3D graphic in Visual Studio. This project consists of four objects, one of which is a complex object, based on a photograph I took earlier this term. The four objects in the photograph are a table, a cereal box, a lemon, and a mug. The reason I chose these objects was because they are part of my morning routine every day. Well, maybe not the cereal every day. The four primitive shapes I created to represent the objects were the plane, cube, pyramid, and cylinder, with the complex object made of a cube and pyramid. I initially planned to use a sphere, but quickly realized that the math behind it was too difficult to understand. Also, I planned to use my mug as the complex object but had difficulties with preparing the torus figure for the handle. Instead, I opted to make my cereal box a complex object. The box is a cube, and the flaps up top were open in the photograph which looked like a pyramid. With this newfound idea, I created the complex object of my cereal box. To correctly depict the objects, I created multiple meshs and called to them in my Urender function. This allowed me to apply different textures to the objects and keep my code more organized. With each mesh, I had to add in information into the namespace, the main function, Urender function, and the Ucreatemesh function associated with the mesh. In the namespace, I added in triangle mesh data and textureId data. In the main function, I made sure to call the function for all of the meshs I was working with. In the Urender function, I made sure to include the code for binding textures on corresponding texture units, activating the cube VAO that is used by the cube and lamp program, and code for drawing the triangles for each mesh. Finally, each Ucreatemesh function had all of the position, normal, and texture data needed to generate the objects.

To navigate my scene, you can use your mouse, your keyboard, or both. With both inputs, navigation is made easy. On your mouse, you can change your view by moving it around and zoom in or out by using the scroll wheel. On your keyboard, you can use the keys W, A, S, D, Q, E. Each key pertains to a different direction, and they are as follows: W is to zoom in, A is to move left, S is to zoom out, D is to move right, Q is to move up, and E is to move down. By pairing both the mouse and keyboard, you can easily rotate your camera view for easy viewing of all objects. To make this work, I used four different functions: 1) the process input function, 2) the mouse position call back, 3) the mouse scroll call back, and 4) the mouse button call back. Each of these functions worked to create actions once buttons were pressed on the keyboard and mouse, and whether the mouse scroll wheel was used. Once this was finished, I began working on the Ucreatemesh functions for each object. Within this function, I used a vertex to plot the triangles of each object and store the normals for lighting, and texture values to apply the texture I wanted. After this, I added in code so the vertex had the correct number of values for each, meaning the position had three values, the normal had three values, and the texture had two values. Next, I made sure to include code to be able to generate multiple VAOs and buffers at the same time. Then, I wanted to be able to create two buffers, the first one for the vertex data, and the second one for the indices. I added in a BindBuffer and BufferData function to activate the buffer and send the vertex data to the GPU. Next, I created strides between vertex coordinates and finally added in code to create the vertex attribute pointers to render my object. I was able to re-use this code in all my mesh functions with no errors. Overall, I am happy with my progress on this project and how much I learned about OpenGL.