**Project One**

Jonathan Campos

CS-330-T4214 Comp Graphic and Visualization

Professor Battersby

April 15, 2023

When I first read the project requirements regarding the shapes that needed to be included, I immediately started to look around my room to see what could be used. The first object I located was a jar that contained a shark. The jar is in the shape of a cylinder, which was one object less I needed to find to complete my submission. I then attempted to look for a torus and realized the lid of the jar could be recreated with the torus shape. At first, I had trouble loading the torus to be flat on the plane. The source code that was provided had the torus in an upright position, not flat. I was able to change the rotation of the object using the following: "rotation = glm::rotate(-90.05f, glm::vec3(1.0, 1.0f, 1.0f));". This did take a while to get right, as I had to make multiple adjustments. The cylinder object was fairly easy to place, as I simply left it at 0. The plane object was an easy implementation with the provided mesh, as it would have taken me a long time if I had been advised to create one from scratch. Since I had the objects on a desk, I applied a wood-type texture. I also scaled the plane so it could hold all of my objects. The next object created was a sphere, which was originally supposed to be a clay ball. After trying to implement a picture of clay, I realized it looked like just a simple white ball. My attempt to make it into a more realistic item was to apply a rubber band ball texture. This resulted in the texture being a bit distorted, but from a distance, it looks like a rubber band ball. The last object I decided to recreate was my computer. Although it was more of a rectangular object, I used the cube mesh and the following scale to create the main shape of the computer. " scale = glm::scale (glm::vec3(7.0f, 7.0f, 2.5f)); Now since I wanted the computer to actually look like one, I applied a texture that displays the internals of the computer from a side view. Applying the texture evidently caused the same texture to appear on each side of the object. My solution to remove these textures was to simply place another object as a cover to hide the duplicate texture. I had to use a total of five cube meshes to create a semi-realistic computer.

I used two different types of inputs to control the camera. The first input was the use of the following keys: W, A, S, D, Q, E, and P.t. types of inputs. The first input was the use of the following keys: "W,A,S,D,Q,E,P". The W key, when pressed, moved the camera closer to the object. The A key allowed a "backwards effect" to the camera. The S and D keys shifted the camera to the left and right, respectively. The Q and E keys allowed the camera to either move upwards or downwards. Lastly, the P key changed the perspective to an orthographic projection, using a boolean value to keep track of which projection it is currently in. When any of the keys are pressed, the camera position is updated using gCamera. ProcessKeyboard function. This function takes two arguments, one being the direction of the camera and the other being the time delta between frames.

To allow the user to control the view of the camera more percisly, the code implements the use of mouse movements. The camera's direction vectors are updated by mouse movements. The ProcessMouseMovement function is called with the offset values for x and y each time the mouse is moved by a user. The result is a more fluid camera movement in a 3d environment.

To be able to use of multiple textures, I included the UCreateTexture function, which loads textures from the file directory to create the texture for each object. We then included the UCreateMeshes function, which creates each object model. This function is defined in the mesh.h file, which was separated from the main source code.

The Meshes class encapsulates data relating to creating and destroying object meshes, making the code more modular and organized. Header files also allow you to separate interfaces from the main code source, which can allow one to reuse the code without having to rewrite it each time and is easier to manage. You also can reduce the risk of accidentally modifying code, which can break your application.