**Project One: Reflection**

Joshua Brown

Department of Computer Science, Southern New Hampshire University

CS – 330: Comp Graphic Visualization

Professor Gary Scott, M.S.

December 10, 2023

**Project One: Reflection**

* **Justify development choices for your 3D scene**. As you write, think about why you chose your selected objects. Also, consider how you were able to program for the required functionality.

When I initially selected my objects for my 3D scene, I figured creating the objects would be similar to creating objects in computer animation. I chose those objects because I thought it would be a good idea to select things that were sitting on my work desk, as I tended to see them or interact with them regularly, and as a result, I was more familiar with how they should look. In high school, I also had some experience with animating using a program called Lightwave, so I thought some concepts would essentially be the same; however, I couldn't have been more wrong. For example, one of the objects I selected in my first Milestone was a coffee cup. I had initially planned to create two cylinders, one larger and one smaller, and use the smaller cylinder to Boolean (or cut away the area of the larger cylinder), which is a common feature in animation programs. After learning more about OpenGL, I soon realized that shapes aren't things that you select from a menu and drag out to select a size; they are instead made from scratch by drawing triangles through code. Upon coming across this realization, I had to forget everything I thought I knew about making shapes, and instead, I had to be more realistic about what I was going to be able to make.

Despite this, I kept two original objects: the medicine bottle and the fork. After learning how to draw cylinders, I used two cylinders for the medicine bottle (one for the top and one for the bottle itself). The 2D fork came next; although it took a lot of time to plot the points, it was essentially a bunch of 2D cubes or squares of different sizes and shapes joined together. After some thought, I decided to ditch the coffee cup and cordless jump rope and instead went with my phone, which is usually always in front of me while working, and my wife's nail polish, which is always sitting in one of the corners of my desk at any given time. I created two cubes for the phone, one 3D and the other 2D. The 2D cube would sit on top of the 3D one and serve as the screen. The nail polish ended up being my object with more than one shape. I created it with a pyramid base to serve as the bottle and a cube to serve as the top. The last object was, of course, the desk itself, which I created using a plane.

* **Explain how a user can navigate your 3D scene**. As you compose your thoughts, discuss how you set up to control the virtual camera for your 3D scene using different input devices.

To navigate my 3D scene, you can use a mixture of keypad and mouse controls. The W and S keys are used to zoom in and out, respectively; the A and D keys are used to move left and right, and the Q and E keys are used to move down and up. The mouse can be used to control the view of the camera, so wherever you point the mouse the camera will face. Lastly, the scroll wheel on the mouse controls the speed of the camera movement; scrolling down on the wheel will significantly slow down the camera movement, while scrolling up on the wheel will make camera movements much faster. The P key is also used to switch back and forth from perspective to orthographic views.

* **Explain the custom functions in your program that you are using to make your code more modular and organized**. Ask yourself, what does the function you developed do and how is it reusable?

One function in particular that I customized to be reusable was the UCreateCylinder function. After I realized that I would be creating multiple cylinders, I immediately started to think about how to prevent redundant functions from all of the different cylinder meshes. Since the medicine bottle was going to be in the same x and z location, I would need to modify the code to take different y values for the start and end of the object. I would then use those coordinates in UCreatePrism to create the cylinders, so when I wanted to make the two cylinders, I could use the same function and just put in the different y coordinates.

Another function I customized to be reusable was calculating the lights in the fragment shader source code. Using one point light, the calculations for that light were all located in the main. When trying to add other lights, I didn't want to redo all of the calculations for each light I added, so I created a function called CalcPointLight that accepted a light's color, position, and intensity as parameters. I then put all of the math for the light in this function, so when I added more lights, I only needed to call the function and add them to the previous lights in the main (in the code, it was under a variable called Phong). Doing so made it easier for me to add more lights to my 3D scene.