Afahri Kerr

CS – 330

Professor Brian Battersby

08/11/2023

**Reflection**

The goal of this project was to create a 3D Scene from a 2D image. I felt that I could best understand how to do this by taking a picture of some school items that I had laying around the house. I wanted the objects to be somewhat of a challenge but not too hard since I was just learning. That is why I chose the objects that I did (A bottle, a book, a toy, and a pen). The objects seemed simple at first but as I began to design them, I noticed that I would have to combine different shapes in ways that I hadn’t planned for at first. I had to add tori as threads to my bottle to make it seem more realistic and combine different sized cylinders and a sphere to make my pen look realistic. It was a fun project to experiment with and see what really goes into designing a scene.

One of the objectives of the project was to create a camera to navigate the scene. I wanted my program to be used on a computer so I set my inputs up for a standard keyboard and mouse. I set the A and D keys to control camera movement from left to right while the Q and E keys can be used to go up and down. The W and S keys were used to zoom in and out while the mouse scroll wheel was used to control the speed of all the movements. The mouse was used to look around the area once you had the camera set up where you wanted it. All these inputs combined allowed you to navigate the scene to anywhere you wanted. We also added an ortho projection view using the O key, but you could revert back to the projection view by pressing the P key. All these commands are handled by my UProcessInput() function.

There is a lot going on to make this program work so I had to set everything up in functions to make my main function easy to follow. This also helps so that I can reuse the code when I make another program of this type. The first of these functions was the UInitialize() function. This function will create a window to store our scene, name it, set the viewport, and tell it how we will navigate that window using mouse. Then I had a UCreateShaderProgram() function which will take your vertex shader code and your fragment shader code and compile them into a single program to color your objects. Another useful function is the UCreateTexture() function. With this function all you have to do is enter a filename and the texture ID that you would like to bind it to. I used this function a lot inside my ULoadTexture() function to set up all the textures for my scene. The main function of this program though is the URender() function. This is where I would set and scale my meshes while applying the textures. It is also where I set my lighting for the scene. If I tried to put all of this code inside my main file then it alone would take up 475 lines. This is why it is important to create functions to organize a program.