Jessica McAlum

SNHU CS 330

February 19, 2023

**Final Project Reflection: Design Decisions**

**Justify development choices for your 3D scene:**

I selected objects for my 3D scene that seemed the easiest to recreate. Each object was simple and took no more than two primitive shapes to recreate. I used a plane to recreate the surface of the table. I used an elongated cube and a flattened pyramid to represent the chopstick. Two cylinders were used to recreate the candle. A sphere was used to recreate the tennis ball and a cube was used to recreate the box.I decided to develop the code in a way that I was able to manipulate the scale, location, and shininess of the objects from one section of the code located in the beginning of the code so I wouldn’t have to scroll so far down and search within the code to alter these attributes. This made it much easier for me to manipulate the scale and locations of the objects to best resemble the original scene. I also did the same for the light sources, so I could easily alter the lighting and the position of the light sources.

**Explain how a user can navigate your 3D scene:**

Moving the cursor around the screen will change the orientation of the camera so it can look up, down, left and right.Pressing the ‘A’ key will move the camera to the left, ‘D’ will move the camera to the right, ‘W’ will move the camera forward, and ‘S’ will move the camera backward. Pressing the ‘Q’ key will move the camera upward, and pressing the ‘E’ key will move the camera downward. Pressing ‘P’ will give you an orthographic (2D) display of the scene, and pressing ‘O’ will bring you back to viewing the (3D) perspective display of the scene. Scrolling the mouse up will decrease the movement speed of the camera, and scrolling the mouse down will increase the movement speed.

**Explain the custom functions in your program that you are using to make your code more modular and organized:**

The functions to draw a cylinder and sphere were put into their own separate files. This made the code more modular and vertex arrays and indices for those shapes could be created with ease. This also helped to keep the source code shorter and less complicated. I wasn’t able to figure out how to get the shader files to work from separate files in time, but I was able to figure out how to create a fragment shader source code that could be used by all of the objects in my scene. The object fragment shader source code also allows me to add multiple point lights if needed.