Nicole Penner

CS-330 Project Reflection

The scene I chose was a couch height table that had some objects on it. This table had a lamp on the far-left side, a tube of Chapstick, coasters, a stack of books, a tissue box, and of course the whole scene sits on a plane. The lamp was a complex object that I created out of multiple objects, since it is more difficult to recreate the shape. I recreated the lamp-shade by using a pyramid, since it was the most similar to the shape of the lamp-shade in real life. I used a prism, or another descriptor would be a tall and narrow cube, to recreate the “stalk” of the lamp. This was the closest to accurate of a shape that I could find. Finally, for the base I created a flattened cube. This is because the body of the lamp was not simply an elongated cube, but it had a small wider base. This was how I recreated the first object. For the tube of Chapstick, this was simple because it’s shape already resembles a primitive shape already – a cylinder. Therefore, to recreate the tube of Chapstick, that is the shape I used. To recreate the coasters on the table, I used a flattened cube, since the coasters were square and this was the most similar shape. I used a cube for the tissue box that was present on the table, which was again created by using a cube – another primitive shape that matched the objects shape. Finally, the whole scene is situated on a plane to mimic the table everything was on. All of these shapes are made using triangles – all of the sides of cubes for example have 2 triangles each which allow a four-point side to be drawn by the program. This is the case for how I drew all the other shapes as well, figuring out their vertices or indices and placing those 3D objects in the correct spot on the 3D plane.

A user is able to use both keys on a keyboard and their mouse to control the 3D scene. A user can press “W” to move forward, “A” to move left, “S” to move backwards, and “D” to move backwards. A user can also use “Q” to move upwards in the 3D scene and “E” to move downwards. Other functionality written to use keyboard buttons is that the letter “P” is able to switch between orthographic (2D) and perspective (3D) views. The mouse can also be used to change views of the 3D scene. The mouses cursor movement is able to change how the viewing screen is oriented, which could be up, down, left, or right. The scroll feature of the mouse is able to control how fast the different control features, namely the cursor, move the camera view of the 3D scene. All of these different features were written by controlling how the camera is viewing the scene. This was done by using the “camera.h” file, and I modified certain methods within it to be able to perform these functionalities. For example, to move using the “WASD” keys and “QE” keys the function uses velocity (speed multiplied by time) to find them. I added the functionality of moving up and down by using the current position and adding or subtracting movement up to get the camera angle to move up and down specifically. To change the viewport, I made use of glViewport() function for 3D viewing of the scene and glOrtho() function to switch to the 2D view. My understanding is the camera represents the lens in which the user is looking at the scene, that is why the movement of the scene uses the “camera.

I did not spend time to create and use as many custom functions as I would have liked. The more modular your code is, the easier it is to take and reuse that code in the future. Some of the functions I used like “camera.h” were already written, but I made changes to that code to be able to ensure my code had the correct camera controls required. In the future I could take my edited camera.h file and use that to impart camera controls in other projects I may do in the future – I would just need to put that function in the code and then call it in the correct way. My functions to draw shapes, shade, and all of the other necessary functionalities to draw the 3D scene correctly are all separate within one executable file. This ensures that I could take the functions I wrote and use them in the future. If code is modular, that means it is reusable and can be easily implemented in other projects in the future as the logic is called for and written correctly.