Nate Bennett

Eugenio Rodriguez

Comp Graphic and Visualization

12 August 2023

7-1 Submit Your Project

The objects that I chose for my OpenGL scene are a cube, plane, cylinder, and a torus. The scene that I was aiming to recreate was a desk with three sheets of paper, a cup, and a pen. The scene also had a phone, I thought about using multiple cubes for the base, screen, and buttons, but I thought that a phone mesh would be rather time-consuming to recreate, so I chose to omit it from the project. The plane represents the table in the scene, and the cubes represent the three sheets of paper on the table. The pen was a cylinder, and to create the vertices, indices, and texture coordinates for the mesh, I utilized for loops. The cup was my multi mesh object that was created using a cylinder for the cup and a torus for the handle. I used for loops to generate the vertices, indices, and the texture coordinates. I chose to texture the cup, handle, paper, and pen meshes. In my render loop, I translated and rotated many of the objects, tweaking them to try and match the format of the scene picture. Texturing took a lot of time to figure out, I almost decided to not use ‘stb’ image and thought about maybe attempting to use ‘SOIL’ instead for texturing. I tried editing the vertex and fragment shaders, mapping and binding textures over and over, and I even tried using ‘tbo’ as a texture buffer object. After many hours, I finally was able to see a texture appear on the cup mesh. After that, I adjusted the texture mapping for loop for the cup and handle meshes and got the texture to look decently clear. For the lighting, I had a directional light, and point lights illuminating the scene. When I read the requirements, and saw that a colored light should be used in the scene, I was not quite sure how I would implement that because the scene seemed like it had decently uniform and consistent lighting. I thought about maybe having blue point lights, or a different colored directional light, but what I ended up implementing was a blue-tinted spotlight that follows the camera like a flashlight.

The user can move around the scene, going forwards, backwards and side to side using WASD. The user can move up and down with Q and E which translates the camera on the y-axis. I used a ‘mouse\_callback’ function to let the user look around the scene, and a ‘scroll\_callback’ function that increases or decreases the user's movement speed variable to a certain point. The user can press P to switch cameras between the free roam camera and a bird's eye view camera. I put conditions in the ‘processInput’ function which check if the user is currently using the bird's eye view camera and does not let them use WASD, Q, E, or the mouse to move or look around if they are. When I first implemented the switching camera functionality, I was having an issue where it would keep switching back and forth really fast if the user held the P key. I did not want this because I just wanted the camera to switch once per time the user presses the P key. What I did to solve this was to create a new variable called ‘birdEyeKeyPressed’ that checked if the key was already pressed, and would not switch the camera another time until it was pressed again.

I believe that each mesh that I rendered in my scene is a separate function. I did this so that each mesh was separate and organized. Each mesh function creates an object that can be displayed when called, this can make the code more modular because the meshes could be called multiple times and several of the same objects could be created with the same parameters. I also implemented a ‘loadTexture’ function that loads textures when called. This can be helpful for organization and modularity because several textures are loaded within the scene, and rather than repeating the loading code each time for every one of them, the ‘loadTexture’ function can be called for each texture.