Janece Gates

Project Two

CS-330- Computerized Graphics and Visualization

December 11, 2022

The design decisions that I made were random initially. The pictures that I took in week two, were not planned properly. I was able to overcome the concerns of what I considered a complex photograph. The milestones leading up to this final project were very helpful. As I completed each milestone and became more familiar with OpenGL, I became increasingly grateful for the objects I chose in my scene. This is because, with exception of the white tube (which I omitted from the final scene) all of the objects were able to be created with a reasonable amount of effort using simple shapes.



The items used in Project Two meet the requirements of using four primitive shapes. The shapes that I used were a cube, a plane, a cylinder, and a torus. Creating these shapes was manageable once I understood the basic principles. The tutorials played a major role in my understanding of all of the required in Modules four, five, and six. My strategy was to use the tutorials as a starter code. The first step was to understand what the tutorial was doing and how it accomplished each week’s task. Then the next step was to deduce the differences in the tutorial’s goal and my goal, finally I was able to make the necessary changes that would result in my goal being realized, which in this case was to create each milestone model and eventually the Project Two model.   
 To navigate the 3D scene, one can make use of both keystrokes and the mouse position. First, the position of the mouse will correspond to the direction that the camera is facing. Once the mouse is in the desired position, the user can then utilize the keystrokes to move in a way that corresponds to the direction of the camera. If the user presses the “w” key, the camera will move forward in the direction it is facing. If the user presses the “s” key, the camera will move backwards in the 180-degree opposite of the direction it is facing. Pressing the “a”, “d”, “q”, and “e” keys moves the camera left, right, up, and down respectively. Again, these movements are executed in a way that corresponds to the direction of the front camera.

I have several custom functions in my code, however the most important ones are generateCylinderPoints(), generateRectangularPrism(), and generatePlane(). My approach with all three of these functions was very similar: make use of triangles to create the 3D components that each function was designed to create. The most extensive of these three functions was without a doubt the generateCylinderPoints() function. This function required the use of an additional helper function called rotXZ(). The rotXZ() function’s purpose was to take a point and rotate it given a specified angle in the XZ plane. I used this function to position the triangles of the pseudo-circles on each side of the cylinder. I had a central point which would be shared by all triangles and an external point which was then rotated continuously until I arrived at the starting position. The sides were created using upright triangles that were paired off such the hypotenuse of one is adjacent to that of the other. Then the pairs were stacked next to each other in a circular fashion. The Plane and the Rectangular Prism were much easier to create and had a strategy similar to that of my cylinder’s sides. Once these functions were complete, it was just a matter of calling the functions repeatedly any time I wanted to add a cylinder, rectangular prism or plane to the scene. Furthermore, to customize the position and dimensions of these objects, I simply needed to supply the corresponding information into the parameters of my functions.

Using the aforementioned custom functions and helper functions, I was able to specify the vertices of my objects. Once this was taken care of, all that was left to do was make use of the texture, shading, and lighting strategies already demonstrated in the tutorials.