Gabriel Feng

April 17, 2022

Computer Graphics and Visualization

Module 7 Reflection

**Development Choices**

While designing my three-dimensional scene, I had many choices to make regarding how to represent the shapes in the image I chose. In my understanding of objects when using three-dimensional modelling software, it would have made the most sense to me at the time to group all the mesh data of an object together. The tree is comprised of a cone and sphere in the scene which would mean that I would need all the vertex data for two primitive shapes in one object. However, the Sun was also a sphere like the leaves of the tree, meaning that I would need to duplicate the data for a sphere for both objects. In the end, I chose to leave the sphere mesh data in the “gLeaves” object and the trunk in “gTree”. This allowed me to call the sphere data to create the Sun with the light shader. With the houses, I also thought that I should separate the objects into a rectangular prism and triangular prism. The problem was that one face of the triangular prism used a separate texture, and with the way the shader was written I had troubles changing textures mid-render. I chose to render the two triangular faces with the rectangular prism in “gWalls” and the roof tiles in “gRoof” which allowed me to avoid the problem in the end. If I experimented a bit more, I may have been able to find a solution to making the primitive shapes one cohesive object but the method I used in the end also worked to finish the scene. The ground is a simple plane.

**Navigation**

To navigate the scene, the viewer uses the WASD keys to move. The “W” key moves the camera in its forward direction, not in any global direction. The same can be said for the “A”, “S”, and “D” keys; they move in the local left, backwards, and right local directions respectively. The camera’s forward direction is shifted with the corresponding vertical motion and horizontal motion of the mouse. The “Q” key moves the camera in the global +y direction, and the “E” key moves the camera in the global -y direction. The mouse scroll wheel controls the speed of movement around the scene. These collective controls are written in the “UProcessInput”, “UMousePositionCallback”, and “UMouseScrollCallback” functions.

**Custom Functions**

The functions that I altered for this project were “UCreateMesh” and “URender”. In the “UCreateMesh” function, I wrote all the vertex and triangle data for each of the three-dimensional shapes that I would render. I added a parameter that would load the specific data into the respective object I wanted. I could have written a new function that took the data from the object and generated the specific vertex arrays and buffers for the object which could have saved a lot of lines or procedurally generated the sphere for the “gLeaves” object. As for the “URender” function, I added many parts to it to render all the required objects. I would have liked to simplify it so that I could pass an object or list of objects and render them all that way, but I couldn’t figure out how to make that work. The “URender” function ended up having all the objects listed one by one rather than being passed to the function one at a time to render. However, the “UCreateTexture”, “UDestroyTexture”, “UDestroyMesh”, “UCreateShaderProgram”, and “UDestroyShaderProgram” are all functions capable of being used and called in a separate program without any problem.