Development Choices

Gary Landstrom

CS-330

June 20, 2022

While designing my 3D scene, I implemented cubes, pyramids, cylinders, and planes to best represent the different objects in my selected scene. Because my original scene only featured three objects that could be considered complex, I chose to spend extra time on the house lamp and use that as a fourth complex object. Only two of the complex objects in my scene are textured. I chose to color the remaining complex objects a solid red to help them stand out, and best show the lighting effect.

I chose to use a grid texture for the scene plane because I could not implement an outdoor texture for the ground without additional shapes and textures. For the house's roof, I chose to use a double-sided slope instead of one of the listed primitives to achieve a more realistic house design.

To draw faster, I defined vertices for primitives objects and used multiple draw calls to render them. Each additional object rendered was repositioned and rescaled with global variables. This drawing method was used for the truck and the lamp because these complex objects share a texture. When drawing the fountain, I defined all of the vertices for eight cubes and one plane. This method was not as efficient but allowed me to implement different textures on the object.

The user can explore the environment using the "W, A, S, D" in conjunction with the mouse to change the camera's angle. The user can also adjust the camera's movement speed with the scroll wheel and change the camera to two additional static angles using the "P" key.

When the "W, A, S, D" keys are used, they directly affect the camera's position. Each can move the camera on the x, y, or z-axis based on the camera angle. The camera angle is controlled by any mouse movement detected. Working together gives the user the illusion of flying around the environment.

I implemented my input controls based on the standard keyboard and mouse setup. Because anyone viewing this project would need a computer implementing input based on keyboard and mouse ensures that the user can effectively use the camera functionality.

I chose to implement a function dedicated to keyboard input scanners because six different key scanners are running while the program runs. Implementing this function helps maintain readability because the GLFW functions that scan for keyboard input are all located in one block instead of directly in the main function that drives the program.

All the functions used to hold scanners can easily be implemented in any program using the GLFW library to read user input. I also implemented separate functions for mouse movement and a scroll wheel scanner. Implementing these functions ensures that the scanners are separated based on type, enhancing the code's readability. If the code were recycled, all the designer would need to do is establish the camera speed and initial positioning.