Project Design Decisions

Jeremy Depenhart

10/11/2023

The objects I chose to depict in my scene were done so from the perspective of both simplicity and challenge. By this I mean that the objects I created were made up of simple primitives, but also contained several of these primitives to allow for a challenge in constructing them. The code itself became simple to write once an instance of each primitive was created. At that point, it was merely a matter of changing scale, position, and rotation for each primitive. I also chose to create parent objects which could affect all primitives in a single objects. This would allow me to adjust the full object once every primitive was positioned and scaled in accordance with the object. A great example of this was the Binder. It was made up of five cubes and three tori. Placing each individual primitive gave the challenge of finding the right size, rotation, and position. However, generating each primitive itself was easy once each object had been created at least once. After each primitive was placed, I then created a parent object in code to allow for the manipulation of the entire binder. I redid this process for each item. This workflow allowed me to go back and add more primitives and allow for more detail in the scene.

The only thing that would change this established workflow was texture application. My bottle object required two textures, a metallic green coat and a logo. While I had code to allow for more than one texture, applying it to shapes was not universal. In the example of my bottle, I had a base cylinder which was meant to have this extra texture. Applying the texture would make it appear on both the side and ends of the cylinder. To avoid this, I had to move code around to ensure that it only drew the secondary texture once both ends of the cylinder were already drawn.

My scene has a variety of navigational controls that allow for a full view of my objects. The user can freely move around the scene using the W, A, S, and D keys to move forward, left, backwards, and right. They can also use Q and E to move the camera straight up and down respectively. Moving the mouse allows the user to rotate the camera and look around the scene. These controls allow the user to move anywhere in the scene and see the objects at any angle. I also implemented code to allow for faster movement using the mouse scroll wheel. The user can scroll the mouse wheel up to move faster in the scene and scroll it down to move slower. I also implemented caps to ensure that the user cannot increase speed infinity of decrease speed to below a certain threshold. Lastly, I implemented the option for the user to hold down the P key and change their camera projection. Holding down the P key switched the camera to orthographic, while letting it go brings the camera back to perspective.

To modularize my code, I created several functions that serve different aspects of creating a 3D scene. While all functions are accessed through a main function, each action is fulfilled in its own separate function. A 3D scene needs to do several things. First, it needs to initialize the different libraries for OpenGL. It also needs to create a window and specify controls. It needs to load and unload shaders, texture, and models. Lastly, it needs to render the scene. All these actions are performed in their own function. Some of these functions can even be used multiple times. Functions that load and unload items, like textures, shaders, and models, can be reused for any number of additional items.

I also made sure to create separate files for certain functions to clean up my source file. The biggest example of this are the functions for loading and unloading each 3D primitive. Each of these functions could easily add a couple of hundred lines of code to the source file, which would make navigation more difficult. So, I created separate .cpp files for these functions and accessed them via header files.