Chris Ellis

CS330 - Professor Gray

04/21/2024

Final Project Reflection

**Final Project Reflection**

There were many choices made in the initial planning for the development of this 3D scene, most of which were made at the beginning of the term before attaining most of the knowledge about graphics programming learned later in the course. For this reason I initially selected objects that I could mostly represent with a single shape, and a few that could also be represented by complex shapes but only one that initially would likely completely require that. This was made to match the requirements on the rubric while not designing a scene that was so complex I would be forced to make multiple complicated objects beyond the requirements. Instead any more detail I wanted to add could be done at a later date, and things that I wanted to do that were too complex could be abandoned without risking not completing the project. Programming the required functionality mainly consisted of building on top of the code for each milestone, and expanding it to fit the scope of the final project. This was mainly combining all of the pieces into a functional whole (lighting, textures, materials, positioning, colors, etc).

For navigating the 3D scene, a user can use the W-A-S-D keys for forward, backward, left and right motion respectively. Q will lower the view while E will raise the view (in the vertical axis). Mouse control allows the user to ‘point’ the camera where the mouse cursor would be, and scrolling the mouse wheel out will speed up all of the key-based movement, while scrolling up on the mouse wheel will slow it down. The O-key can be pressed to enter an orthoscopic (front-facing) view of the scene, and P can be pressed to return to the perspective view.

I did not provide any custom functions, due to the development of the code being done alongside the development of the essential 3D components, which required being able to easily access each of the modifiers for all of the pieces in the scene. Now that the scene is complete, I would work on refactoring some of the code to be more DRY (don’t repeat yourself), and ensure that specific pieces were more readable. The most obvious piece that could use this treatment is the configuration of each mesh before it is drawn. The the simplest way to do this would be to provide some function like ConfigureMesh() that has parameters for changing things like scale, rotation, position, and the functions for texture UV scale and shader/material values. This could be done multiple ways but a function that could take this inputs and be configured from a larger configuration object like a HashTable of those values that also provides sane-defaults would reduce the footprint of the RenderScene function greatly. This configuration could be provided in one of more separate files as well. Obviously this example would be a complex function but it would be broken down further internally with wrappers for setting each of the different pieces. This practice of separating configuration values out from code avoids having ‘magic numbers’ inside many places in the code, like color values, positions, etc.