Tristan Blaisdell

Review

The scene that I made had a lot of shapes that can easily be generated with OpenGL. While I had a few complex shapes that required a combination of two or more meshes, most of the items were not complex shapes and just needed to be placed in the right positions. A lot of the shapes that I made also shared a similar color. Most of the objects in my scene are different shades of black which allowed for less complex texturing. While many shapes are black the computer screens themselves required a complex texture. I took a screenshot of my desktop and converted it to a .jpg file in order to use it effectively. Once I did this I had to work on getting it to be the exact right size to avoid stretching. The keyboard was also a bit complex. At first, I was going to individually place each key but with some research I was able to generate the keys in a grid pattern that allowed them to be evenly spaced and represent a real keyboard. The lighting that I used in my scene has three different light sources. I made a yellow light emulate from the front of the scene to represent the sun that is in my original image. I also added a red glow light coming from the keyboard to represent the lights on my keyboard. I was originally planning on putting a light source in between each key, however upon seeing that I could only use four light sources I made a red glow light instead that emulates from the keyboard. I was planning on adding another light coming from the monitors however I felt that made too many competing light sources, so by using a bright texture for the screen it gives the appearance of a light source without actually having one.

The user can navigate through the scene in a few different ways. The mouse itself will always control the direction that the camera is pointed in. The user can also use the ‘Q’ and ‘E’ keys to go up and down. The user also has access to ‘WASD’ which will move the camera in the direction they push. ‘WASD’ work like arrow keys so the user can go left, right, forward, and back. Lastly, the user has access to make the scene 3D and 2D by pressing ‘P’ and ‘O’. This will give the user a much different perspective on the scene.

The code base is very modular. The RenderScene() function was the function that I spent the most time with. This function is where all shapes and meshes are generated and placed in the world. This function also has access to setting the color, texture, and material the shapes are made out of. I split RenderScene() into multiple different smaller parts that are labeled for what part of the scene is being referenced. PrepareScene() is also a modular function that determines all shapes that will be loaded into the scene. In PrepareScene() you can pre-load the shapes into memory. SetupSceneLights() is also the location of all code relative to lighting. This is where the light location, intensity, color, and type is set. DefineObjectMaterials() is a modular function that allows all of the materials in the scene to be defined and later called in RenderScene(). Lastly, we have LoadSceneTextures() which allows us to load in different textures into the scene. These textures have to be located in the appropriate folder and then set in LoadSceneTextures(). All of these functions are then called in RenderScene().