Austin Scarinza

CS 330 Computer Graphic and Visualization

June 21, 2024

Reflection

When selecting my scene for the project, I wanted a scene that I would be able to create, but also pose some challenges along the way. The scene I chose had a couple of complex objects that allowed me to practice combining and manipulating several objects to create one cohesive thing. The main objects were the computer stand, the computer, and the vase. There were also some simple, smaller objects such as the keyboard, mouse, and keypad. I also incorporated a two-toned background that reflected the wall shown in the scene, and a table to help set the scene. To make the scene more realistic, I added textures and material characteristics to the complex main objects, the background, and the table.

Finding and selecting textures that worked with each object also created some challenges. For instance, the code was only set up to load and use textures that used three-channel RGB images, and I often found good textures that I liked, however, they were grayscale, single-channel pictures and could not be used. I would only find this out after loading and adding the texture to the code and getting errors thrown that they could not be used. Creating material profiles to match the textures was also difficult. Getting the different light types to act according to each material took some time. Seeing how wood acted differently from glass and metal made sense, it just took some trial and error. Adding the lighting on top helped finish the scene. Seeing how the materials and textures lit up in the light and faded out as the light panned away gave the scene some added realism. Adding colored ambient light to the scene was able to show off different hues in the textures and reflections of the objects.

Most of my scene was composed of boxes. The computer stand and computer were created from boxes, just all different sizes and orientations. Creating the programming to add the transform the boxes was quite simple, the challenge came when trying to move each object into the correct place to match the example scene. Sometimes the object would look to be in the correct place, but once I planned to either side or up and down, I could see that it either floated off the table or was too far forward. Getting the object to match the scene from all angles took the most time and was done by simple trial and error, moving it slightly into position over many executions.

When navigating and ensuring the objects were in the correct place, I utilized the camera controls that were incorporated to help move about the scene. The A, W, S, and D keys were used to move left, right, forward, and backward. The Q and E keys were used to move up and down in the scene and the mouse was used to look around. There was also the use of the mouse wheel which controlled the sensitivity of the speed at which all the other actions functioned.

Each action and operation in the code was written with a distinct purpose. The modularity of the codebase helped readability and refactoring when new components were added. Each function had one specific task and was named accordingly. For example, the LoadTextures function had one job, the open, load, and validate the textures being used in the scene. When textures were being opened, instead of having all the code each time in a larger function, having this one function to load the texture saves space, and makes the code clearer and more consistent. The whole codebase was written with this in mind and actibly reflects an error to keep the code clear and free of unneeded clutter. This made finding errors both logically and with syntax quicker because if there was an error loading a mesh, I knew where to look right away.