3D Scene: Gwen’s Toy Shelf

CS330-X6143 Comp Graphic and Visualization

Raphael Coloma

I chose to recreate my daughter’s toy shelf for the 3D scene final project. I admit I underestimated its difficulty. In one of our early assignments, we were asked how many objects from the picture we were planning on recreating. I was hopeful and said I would do everything. I was sadly mistaken. The final project required at least four completed objects. One of them needed to be comprised of two or more primitive shapes and two objects needed to be textured. It took a lot of research and effort, but I recreated six objects and applied texture to all of them. Three of the objects I created were “complex” objects made of two or more primitive shapes.

The objects I recreated were the two stuffed animals, two toy baskets, a set of books, and the shelf itself. I chose these objects because I wanted the 3D scene to be realistic and filled with character. I wanted to create objects that were complex, but not too complicated. The stuffed animals and one of the toy baskets were my “complex” objects. Their awkward shapes made it a good challenge. They required me to look deeper into how to create spheres and cylinders. It was interesting to break each object into primitive shapes and determine how they would be placed in the scene. In the program, I created meshes for five of the six primitive shapes that were suggested. As I started to code each object, it became increasingly difficult to keep track of each item; comments were a necessity. Once I had the meshes created, I was able to draw the shapes into my scene and manipulate them as needed.

The camera can be controlled using the keyboard and mouse. The mouse controls the direction the camera is looking and the WASD keys control the camera’s movement (forward, left, backward, right respectively). Additionally, the QE keys control the upward and downward movements of the camera, and the scroll wheel adjusts the speed of movement of the camera around the scene. I utilized the OpenGL camera header file to implement these controls. The UProcessInput function makes calls to the ProcessKeyboard function in the camera header file through assigned keystrokes. The directions are defined as forward, right, and up. When a movement key is pressed, the position of the camera is incremented/decremented respectively and multiplied by the movement speed and the time between frames. This allows movement of the camera through the 3D scene. The direction of the camera is tied to the movement of the mouse. Once the cursor position is established by the UMousePositionCallback function, it is passed to the ProcessMouseMovement function in the camera header file. The ProcessMouseMovement function then calls the updateCameraVector function to convert the cursor position into vector coordinates and orient the camera accordingly.

It was difficult to understand the code in the beginning. I researched different techniques and followed different tutorials that ended up making things more confusing. In the end, I decided to start from scratch and utilize the materials the class provided. Though the functions remain structurally the same, I incorporated classes that helped make the program more modular. As mentioned, I used and modified the OpenGL camera header file to allow freedom of movement in the 3D scene. To create and texture the spheres, I used a sphere header file. I just needed to pass the radius, number of sectors, and the number of stacks and the class would create and draw a sphere. The same holds true for the cylinder class, but the cylinder class requires the vertexBufferObject class and the staticMesh3D class to draw the object. I know there I could have refactored the code more to make it more modular. Looking at it, there are sections that could be adjusted to reduce the redundancy, but I am afraid to break the code and ruin all the hard work I currently put into it. I learned a lot from this class and I plan to delve deeper into the possibilities of OpenGL.