Chris Bowles CS-330-R3329

Module 7 Assignment

2-25-24

**Justify development choices for your 3D scene. As you write, think about why you chose your selected objects. Also consider how you were able to program for the required functionality.**

For my 3D scene, I used a selection of cubes, planes, cylinders, and cones to create the objects in it. I attempted to recreate a castle gate from the video game, World of Warcraft, so I thought it made sense to use cubes for the walls of the castle (by elongating them through adjusting object scale) as well as cylinders for the towers along the wall and then finally the cones would be representing the pointed roofs of the towers. Several of the objects were combined in the scene to give the perspective of a complex object like a tower with a roof on it, and the gates attached to the walls. To meet the requirements for the class, I chose at least four different types of shapes and wrote functions for creating them by using arrays of vertices and/or indices to relay positional, normal, and texture coordinates to the shader programs and then draw the elements themselves.

We also incorporated directional overhead lighting and point lighting in the form of a sun into the scene. The overhead lighting is just white in general which gives an even spread of light to the whole scene. The point light is an orange light square that gives a combination of diffused and specular lighting to the scene. It gives a perspective that there is currently a setting sun in the sky of orange color, casting orange rays against the walls and ground as it sets. It goes well with the skybox we implemented into the scene which is also reminiscent of sunset.

**Explain how a user can navigate your 3D scene. As you compose your thoughts, discuss how you set up to control the virtual camera for your 3D scene using different input devices.**

Our project includes functions to determine input from several sources, including keyboard, mouse clicks, and mouse cursor. These inputs are polled at the end of every frame to determine if any actions should be taken. We can use keyboard keys W, A, S, and D to move forward, left, back, and right respectively. We can also move up and down using the Q and E keys. The middle mouse button can be scrolled to increase or decrease the speed at which we move around the scene. We made sure to set limits for the speed so that it does not go above or below certain thresholds. We don’t want negative speed for example as this tends to switch the directions in which we move. The mouse cursor is what we use to change the pitch and yaw of the camera as we move around the scene. Finally, we can press the P key to switch our perspective of the scene between normal and orthographic views. Switching to this view does look strange using our skybox however, as the skybox position is directly tied to the camera view.

**Explain the custom functions in your program that you are using to make your code more modular and organized. Ask yourself, what does the function you developed do and how is it reusable?**

We have a custom function for key callback, which is designed to process when certain key presses are input. We designed this separately from the Process Input function as it was not recording an individual key press. In other words, every time we pressed P to switch to the orthographic view, it registered as 6-10 key presses depending on the sensitivity of the press. The key callback function can determine that the key is only being pressed once.

We also have the UCreatePrism function which was designed to take in a number of sides and create a rectangular prism based on that number. 4 sides would be a rectangle, and 100 sides would be representative of a cylinder. The more sides, the rounder the circumference. This function is very reusable for many objects, depending on how many sides are declared.

We also make use of several functions throughout the project such as the UCreateShaderProgram and UCreateTexture functions for combining shader programs and creating textures after loading in the image files. We use the UProcessInput function every single time the main loop runs to determine if keys were pressed. Obviously, we also have the Render function that binds textures and draws elements every main loop as well. In the future, we could probably make the Render function itself more organized as there are a lot of repeat calls for activating VAOs, binding textures, and drawing elements.