Project 1

Tyler Hollingsworth

Computer Science Department, Southern New Hampshire University

CS-330: Comp Graphic and Visualization

Instructor Malcolm Wabara

December 11th, 2022

My scene consists of a glasses case, a wooden pumpkin, a plastic pumpkin, and a bowl on an entry table. I chose these items because they initially stood out as I was looking for items to depict in a random collage and they met the object requirements. Another reason I chose these items was based on the perception that I could recreate them using triangles and maintain the original look. After selecting my objects, I began work on my scene by creating the glasses case object and centering it in the middle of the world. After I was able to get the glasses case rendering in the world view, I began to add texture to the object; place a plane beneath it representing the entry table; create methods to navigate the scene; and add light using the Phong lighting model.

After all the essential elements were set up, I began to add the other three objects to the scene one by one. The most difficult aspect of this was positioning the objects relative to each other and verifying that their size proportions were approximate to one another. Each of the objects also added more complexity to the code as they all needed different textures, vertices, and indices to render in the frame. Because of this, I had to add different meshes to my UcreateMesh function so that there was separation among the objects, and they were able to be rendered separately. By separating the objects, I gained more control over what shader was used to render them and what textures were applied to them.

Navigation was also a very large part of creating the program as the 3-D objects could not fully be seen without the ability to move throughout the scene and in between its objects. The first step in navigating the scene was by processing key inputs for basic movements. The W(forward), S(backward), A(left), D(right), Q(up), and E(down) keys act as input into the UProcessInput function and moves the entire scene according to the button press which represents the perception that the camera is navigating the scene. I then created the scroll\_callback function to take input from the mouse wheel and either speed up or slow down the camera movement depending on the direction that the mouse wheel is scrolled. Lastly, I created the mouse\_callback function to take mouse movement as an input that adjusts the target of the camera or in other words rotates the camera around its axis depending on the mouse movements.

To make the code easier to understand and maintain it was modularized into several functions that perform different tasks such as the navigational functions listed above. The main function calls upon all these other functions to run the program in a clearer and more separated way versus having all the code run together without any clear separation of ideas. The main function first calls the UcreateMesh function which creates the meshes for all the objects within the scene. It then calls the UcreateShader function twice to create the two shaders, one for the objects and the other for the light sources. After setting the background color to blue and entering the while loop the main function calls the UProcessInput function for navigating the scene based on key and mouse inputs.

The main function then calls the URender function to render the frame and draw the objects. It then exits the while loop and moves on to the tear-down process by calling the UDestroyMesh function which deletes the created meshes. The main function also calls a UDestroyShaderProgram function twice to delete the two created shaders. Finally, the main function calls the exit function to exit the program. There are many other functions within the program that perform various tasks including all the functions used from the imported libraries. Using these functions is a great way to break the code down into manageable pieces that perform specific tasks much like how the structure of this paper is broken down into paragraphs.