While developing this project, I had faced many challenges. These challenges cam in many forms considering at first, I did not understand the way OpenGL used code to generate a 3D scene. I had to research and watch tutorials outside of the materials provided for me to fully understand OpenGL. Even though I had faced many challenges I found making this project a lot of fun. Not only was I able to practice coding and using best practices but while doing so I was able to see a visual representation of what I was building.

When I first started this project, I thought it was going to be an easy feat. However, as the issues started to arise, I found that my project was not going to be easy, even in the slightest. This showed to be true when the first task was to create a 3D object. I did not understand at first how to generate the object. However, as I played with the code, I was quickly able to figure it out. After realizing how to generate object in the scene, I decided I wanted to use both vertices and indices to draw the scene. This was achieved through using the verts and indices arrays. In the indices array we took the index of each vertex in the verts array and match it to other indexes to form triangles. I decided to use this method because when I started my research for further understand of the material, I had found that using vertices was more efficient than using plain vertices. This is because using indices you minimize the number of draw calls. After completing many of the milestones using verts and indices, I quickly found that I needed to revert to using strictly vertices to draw the objects. This was found when we had to add texture to the objects. I could not find a way to add texture using the indices. In fact I couldn’t even figure out how to add the textures using just the vertices. After playing around with the code I finally figured everything out with the textures. I found using vertices to draw and texture the object to be easier than using the indices. This is because when drawing with vertices all you had to do was call the bindTexture() method and pass in the appropriate properties to have the texture binded to the selected vertices in the drawArrays() method.

Before adding textures to this project, we had to add functionality to allow the viewer to navigate the scene. For this functionality to work properly we first needed to add a header file in the project. This file was used for the camera movement functions and methods. Using the provided camera header file from the CS330 repository on GitHub, I was able to get the required functionality working properly. Even though it was working properly, the required functionality was not fully implemented. To fully implement the movement requirements, I had to add quite a few things to the header file. First I needed to declare the direction of movement in the Camera\_Movement() function. Next I had to add a conditional statement in the ProcessKeyboard() function to allow for the correct keys to be associated with the correct direction of movement. While implementing the movement, there was another functional requirement to implement. This requirement was using the scroll on the mouse to increase of decrease the speed of movement in the scene. To implement this functionality I needed to change the ProcessMouseScroll() function. I had to change it because it was previously coded to allow for the zoom in the scene. By changing the zoom variables and replacing them with the MovementSpeed variable I was able to get the required functionality. After making the changes to the camera header file, I ran the project and tested and realized that the movement requirements were not met. The previously implemented movements worked correctly, as well as the movement speed requirement. However, the up and down movement did not function as it should have. To allow for the up and down movement using the Q and E keys I had to make changes to the UProcessInput() function in the Source.cpp file. This was not necessarily a change rather more of just adding the requirements. To correctly implement the up and down movements I needed to add two more conditional statements in the UProcessInput() method. After adding this I tested the program and found that it functioned properly. However, there was one requirement that was yet to be implemented. This requirement allowed for the user to change the view of the scene using the P key. This meant that when the P key was pressed the using could toggle between an ortho view and a perspective view of the scene. This requirement was not too difficult to implement. For the proper implementation, I first needed to initialize a boolean variable for the ortho view and set it to false. Next, I needed to add the input. This was achieved by adding another conditional statement in the UProcessInput() function. After the input was implemented I then needed to add a conditional statement in the URender() function. This conditional statement would check if ortho is true or false when the P key is pressed. Then depending on the value of the variable the view of the scene would change between ortho and perspective views. After implementing the functionality requirements, the program was tested. Using the Q, E, A, S, W, and D keys the view can move through out the scene. The Q and E keys controlled the up and down movements in the scene. The W and S keys controlled the zoom. The A and D keys controlled the left and right movement. Finally, the scroll wheel on the mouse controlled the speed at which the movements were performed.

When looking back through the code and thinking of modularity and reusability I find that most, if not all the functions in my code are modular and reusable. For instance, looking at the UInitialize() function we can see that the use of this function could be used through out any other program. This can be done because having this function modular the way it is allows for any developer to reuse this function by only changing a few lines of code to allow it to function with their projects. Thinking more about the modularity and reusability of this program, I find that these are important in the development world. This is because it will cut down time and costs for companies to implement certain functionality and methods into their programs. I also find it important because it allows other developers the opportunity to implement your functions and functionality if they choose to do so.