For this project, I chose to try and model a small scene from an image I found of an outdoor eatery in Valletta, Malta. It quickly became complicated, so I chose to model some tables from the scene, as well as some walls and stairs.

The objects themselves are derived from a base class that holds data that all meshes will have, and there are three other classes: Plane, Cylinder, and Light. The Plane and Cylinder classes inherit from the Mesh base class, while adding their own functionalities as well. The Plane class, for example, allows you to create a plane with the specified width and length. The Cylinder class creates a cylinder based on a set of parameters, such as radius, height, and the number of triangles to use for the calculations. The higher the number of triangles you specify, the “higher resolution” or round the cylinder will be, because more triangles are being used. To save a bit of time, some objects in the scene (such as the base for the tables, and the square tabletops) are rendered as a cylinder with the number of triangles set to 4, as this creates a square. The math involved for creating the cylinders is kind of outside the scope of this project, but I decided to do it all myself because I wanted to learn how to do it. And so, I did. It could likely be done in a more efficient manner, such as by using recursion, but for the sake of the project, I decided to go with it.

The Light class is used for the two lights in the scene. The Light class holds properties such as the position of the light, the color of the light, some hard-coded values used to calculate the attenuation in the fragment shader, and mesh data such as vertices. The vertices for the lights are not dynamically generated based on any user-defined parameters – they are hard-coded as pyramids for simplicity. I went with two point lights for this scene. One of them is a really light shade of blue, as one of them had to be a different color. The other one is simply white light. It was a bit complicated getting the lighting to look right with multiple objects in the scene. I couldn’t quite get all of the objects to take both lights into consideration, so some of them only reflect light from one of the lights. The lighting works nonetheless. This made me realize how complicated lighting calculations can be, and how difficult it is.

Navigating around the scene is quite simple. You can use the W, A, S, and D keys to navigate forward, left, backward, and right, respectively. Holding the left shift key allows you to move the camera slower throughout the scene, and releasing it returns it to normal speed. The mouse scroll wheel also allows you to control the speed of the camera. The Q and E keys cause the camera to move straight up, and straight down. I modified the camera.h header file in order to implement this, as well as the camera speed controller using the scroll wheel. The modified header file is included in my project submission. This control set up is very typical these days, and is extremely common in video games and various other simulations. It was really cool to implement it, and now I know how much work goes into creating camera navigation through 3D space.

Finding textures for the project was really fun, and I found a lot of cool places to obtain textures that are licensed for any type of use. All of the textures that I used here were found from <https://ambientcg.com/>, and they do not require attribution and are free to use in any context. You can view their license from the homepage of the website to confirm.

This project, although stressful, was really fun, and I will certainly be improving upon this project in the future using my own time. I want to learn more about OpenGL and make a proper game or simulation using OpenGL.