During the development of my 3D scene, I employed several different types of programming. I employed many different controls and objects to populate the working area with objects from my OpenGL Scene that I chose. The functions of OpenGL are very neat and well laid out. This allows the readability of code from OpenGL to be easier. I made sure to minimize any of the code that was not in a function, as this leads to more secure programs. Custom functions were my favorite part of this class. I loved how it is possible to reuse meshes that are created from triangles.

I started strong with using the code from LearnOpenGL, but this was short-lived, as I discovered that everything that I had created was insufficient. I found it better to utilize the U\* functions that were provided with the weekly teachings. I found them to be much easier. So much easier, in fact, that it only took me roughly 5 hours to fully coordinate myself with all the functions and their uses. I found it was best to utilize cameras using pointers, as it allowed for easier manipulation of the vertices that the cameras use. In the end, it was in my best interest to learn the other code, as it has many useful functions.

The different objects that were created are a bittersweet thing. I would rather attach piece by piece, as they can get rather complex. The fact that you can manipulate these same created meshes to make other, more complicated meshes blows my mind. They are drawn with triangles. If they are only drawn with triangles, imagine how complex some characters are in games. Some even having upwards of thousands of triangles. I believe that the more intricate designs on these characters would take an extremely massive amount of time. However, even that pales in comparison to maps on larger open-world games. The sheer amount of debugging for the User Interface alone is bad enough. I imagine that most developers drive themselves mad until this all makes sense.

Controls are a very intuitive part of any interactive application. In my application, I made sure to account for input from the mouse and keyboard. I incorporated a WASD movement system, a way to raise and lower the camera, a way to show an orthogonal view and a way to exit the application quickly. If this were a professional development, the importance of the way to exit the application would be in a menu somewhere, so that it would not be so easy to exit an application without saving user-information somehow. I added the flying movement system, as well as the original buttons from Minecraft to OpenGL. I believe that I could have incorporated a gamepad, but the gamepad that I have requires specialized software to function.

Functions and their uses are an art among programmers. Modular building is something that is learned. It is best to keep everything in a function anyway, as variables can break, but variables that are used only during functions, have slightly less chance of causing an issue for the entire application, or worse, leaking information. Modular code makes your program easier to read, as you can follow all the steps and read the entire application MUCH more easily. This also makes it possible to call some functions more than once. As a great example for my scene, my table is made completely of cubes. They are made from the same cube, repeatedly, because this cube is modular. It can be reused, and this is the same for all the functions that I utilized during my development process.

My development process was interrupted by new items though. I chose to keep the carpet, table, book, headphones case, and the roll of medical tape. I chose to add a thumbtack, a sanding block, and my walls for better viewing pleasure. I used four of the shapes from the list to create my scene, and I believe that I did an awesome job. I chose to leave the corners of the table flat and not rounded, as this is majorly over-complicated. I chose to have a green light behind my walls to simulate a green reflection, and the white light in my code is to simulate a lightbulb.