Most of my choices in my scene revolved around being able to use a cuboid object. This is because a lot of the objects in the scene could be very easily based on multiple cube objects of different scales. The large coffee table was created as the center piece of the scene. This is what everything was based around. A laptop was created to put onto of the table, along with a decorative piece in the shape of a pyramid. These were put there to have something on top of the table. A bare surface doesn’t look that good. Couches on two sides of the table were created, one large and one small to give the scene a more roomy feel. They, also, provided a complex object that could be created. Lastly, the cabinet was assembled in the corner because another object was needed and I didn’t want something that wasn’t too simple.

The camera was set up per directions. The mouse provided a yaw and pitch for the camera, allowing the user to change angles of the camera. The scroll wheel allows the user to speed up the movement of the camera or slow it down. As always, WASD provide the directional movement so the user can actually move through the scene. The keys Q and E provide an up and down movement for the user, allowing the user to move up or down without needing to the change the angle of the camera. Lastly, the key P, changes the view of the camera. By default, the camera is set to perspective and once P is pressed, it changes to orthographic.

Aside from the needed functions and code that were created by following the guides, a few different things, including classes, were created for this scene. Firstly, the CubeObject and PyramidObject cpp files were created to house the vertices and indices needed for the objects. This was done to clean up the code and make it less messy, as they were originally in structs. I had, originally, wanted to make a master class and have other classes, which would populate the vertices and indices, inherit from the master class. Doing this proved to be a little difficult though, because, after doing some research with friends that are in the software engineering field, we found out OpenGL does not like inheritance classes. This is why some functions are basically copies with different object variables assigned.

More functions were created as well to help with the program. One function was the object scale, translation, and rotation function. It’s a simple function that just changes the objects scale, translation, and rotation in one line of code. This prevented me from having to write 3 or more lines each time I wanted to change something. Plus, it’s easy to change and work with when moving, scaling, or rotating and object. Another function was the draw object function. This function gave the program a way to draw an object just by passing the object variable and a few other parameters. So, anytime I wanted to draw something, I passed the object, the texture wanted and the required camera vars and it was drawn.

Lastly, the large code block functions at the bottom of the main file. These serve no other purpose than getting very large blocks of the same line of code together where it doesn’t clutter the main portion. All the mesh creation calls, mesh deletion calls, draw object calls, object scale/translation/rotation calls, and texture calls are grouped here. Some might say this is unneeded, but it makes it easier for me to understand and work with.

With all these, you would only have to write about four lines of code to create an object. One, declaring the variable by using the class imported. Calling the mesh creation. Calling the object scale/translation/rotation function. And lastly, calling the draw object function. Of course, it’s good practice to write in the deletion functions as well, but those aren’t need to draw the objects.