My 3D scene consisted of a plane, a sphere, a cube, a torus, and three cylinders. When originally choosing objects, I thought this was a good variety that would make a fun scene without being overly complicated. When I selected the primitive shapes, they each had a specific purpose: the sphere was a tennis ball, the cylinders were a candle and a mug (the mug later became two cylinders – a second one to form the *interior* of the mug/the coffee), the cube was a Rubik’s cube, the torus became the handle of the mug, and the plane was the desk the other objects were all situated on. The cube was straightforward to design, since were worked with cubes frequently – making sure the textures were correct was the most difficult part, as it took some tweaking to make sure the triangle slices forming each side matched up correctly. The objects utilizing formulas for elements of the object (sphere and cylinders) were definitely more challenging. It took me a lot of effort to find the right way to implement these; I felt I followed the logic of the formula relatively well, but would struggle to weave it into the overall program. I eventually found a good tutorial and was able to follow it to create separate classes and header files containing the logic to implement them. The torus was similar, though it took some tweaking to get the appearance right. The plane was actually fun to implement, as I just flattened out a cube because it matched the look of a real desk that way by having some depth. I textured it with a wood pattern and scaled the size of that texture to match and thought it looked really good.

I enjoyed the process of setting up the camera controls – the tutorials in our reading were easy enough to follow for this section for the basics. Using the existing camera class and adding a few modifications (such as additional directions to aid the dropping straight down and raising straight up functions) made this fairly straightforward. Finding the right logic of combining speed of the camera with the aforementioned “movement” added to the camera class completed the addition of those functions to the standard ‘WASD’ movement, which consists of using those four keys along with built-in functionality to cause that to code to map new positions based on camera speed, time, and direction/location. Additionally, free mouse movement was added so that the user can look around the screen simply by moving the mouse around and the view continuously updates to reflect that. The mouse scroll options were also mapped (to the camera speed), allowing it to speed up or slow down when moving around the scene – I set caps on both ends so that you stayed within reasonable limits. Finally, I set up a Boolean that matched the normal view projection (set to perspective) and toggled it off when hitting the ‘P’ key, which then converted to an orthographic view for the user to see the scene in a 2D perspective.

The code would have been a nightmare to handle without custom functions. While most of my functionality consisted of manipulating and reusing/redesigning functions from our previous assignments (hence using milestones to build upon each other until the final), the fact remains that these functions had to be added (primarily around the camera and sphere/cylinder objects) to *greatly* improve the code writing process and the reusability of these elements of code. For instance, I created a mouse position callback function that didn’t exist in our earlier projects. Without this, the ability to use the mouse to look around the scene would not exist. Similarly, a helpful breakthrough on the assignment was when I learned how easy it was to create multiple Vertex Array Objects and Vertex Buffer Objects, and how similarly easy it was to use those – this eliminated a ton of code by creating a lot of reusability. Adding textures via a custom function made that process *much* easier than having to run through large parts of that process each time, so that was another area where it was far more simple to add another texture thanks to that custom function. All in all, working with this much code and doing so many different things with it would be near impossible without adding levels of reusability. Even just being able to call the various function in our main and render functions eliminated a great deal of headache – and frankly made it much easier to troubleshoot and work through adding and changing elements of the code. So, I think I did a good job creating functions that enhanced the modularity and reusability of the code.