For my final project, I chose to recreate a wooden desktop with a deck of cards, notebook, pen, and coffee mug sitting on top of it. I chose my selected objects since I had actual versions of them on my real desk for reference, and they’re fairly easy to recognize from combining a few primitive shapes. In the case of the mug as an example, I was able to approximate this object by combining a torus for the handle, and cylinder for the body. The rest of the objects could be made using planes and cubes, other than the pen which was also another elongated cylinder. These objects were also fairly simple to texture for additional detail while preserving some level of realism to them such as in the case of the wooden desktop which is also flat in real life as well. I also didn’t want to scope a project too far out of reach for the time window given, since this is the first time I’ve used OpenGL to create any sort of 3D scene.

For scene navigation, I used the classic first-person movement control scheme of W, A, S, and D keyboard keys in combination with using the mouse movement for looking around. In this instance, rather than controlling a player with physics, the project uses a flying camera approach as seen in many 3D editors and game engines. In addition, you can also press the “Q” and “E” keys to move up and down vertically, as well as scroll with the mouse wheel forward or back to adjust your overall movement speed for more fine-tuned controls. While you could extend functionality to include other forms of input such as joysticks, or touch screens, this project sticks with mouse and keyboard for a wider range of user compatibility since most computers use these controls anyways.

To help make my code more modular, I broke it out into multiple functions to help me develop my scene more efficiently. The most used one was my “DrawMesh()” function which binds the given meshes VAO, texture ID, applies matrix transformations, and lastly calls either the “DrawArrays()” function or “DrawElements()” function depending on whether the mesh is using an index buffer or not. I included everything needed to draw a 3D mesh in this function such that it would only require one line of code in the render loop. This was especially important given that it gets called for every primitive draw call needed in the scene. Another example of reusable and modular code was my “CreateTexture(texturePath)” function which takes in a image files name/path and does everything needed to generate an OpenGL texture. This function returns its resulting ID when finished such that it could be used later on as an argument when I called the “DrawMesh()” function I just previously described. Given that I used at least 5 distinct textures in my project, and it’s common to use lots of different textures when developing 3D scenes, this function was an important addition. The use of both of these self-defined functions in my project made the code much easier to read, less bug prone, and overall faster to develop.