I chose the objects in my scene based on what I thought I would be able to create in code. Initially, I thought that I would be able to find a 3D modeling program that could spit out the vertex/normal/uv data that I could use for each object in my scene. This ended up being much more difficult than I anticipated and I decided not to waste too much time figuring out how to make it work. I chose the objects I did because they seemed relatively easy to represent in an OpenGL scene. I could not get a sphere to render correctly so I had to improvise with multiple primitive shapes. The most difficult part of the scene, for me, was figuring out all the vertices and normals for each object. Doing it manually with pen and paper was very time consuming (is there an easier way?). The creation of this scene took a multitude of effort, for me. Even though the final product does not look that great (in my opinion), I am ultimately happy that I was able to create something that (mostly) met the criteria for the final project.

The user can navigate the scene using the keyboard and mouse. The navigation is very similar to most popular first person video games played with the keyboard and mouse. The WASD keys on the keyboard move the camera forward, left, backward, and right, respectively, while the mouse is used to look in different directions. The mouse wheel can be used to change the speed at which the camera travels around the scene. Additionally, the Q and E keys can be used move the camera vertically within the scene.

I am going to be blunt and honest here. I had very high aspirations for my code in this class. I had thoughts and plans to make my own framework based on the tutorial code that was provided to us. In the end I was not able to accomplish this, I just did not have the time to do it. But I can talk about what I WOULD have done if given the time. The main places that I would refactor the code into my own solution would be in the UCreateMesh() and URender() functions. As soon as we were tasked with rendering multiple objects, I found that the current state of UCreateMesh was clunky and unwieldy when rendering multiple objects. I would have liked to define all of my vertex data in a separate .cpp file (or even better, create code that reads .obj files exported from 3D modeling software), and then at runtime I would have passed the vertex array variables defined within into a new create mesh function to set up the VAOs and VBOs for each object in the scene. This would drastically reduce the amount of code bloat that is currently present from having to repeat the multiple calls to glVertexAttribPointer, etc. for each object in the scene.

When considering the rendering loop, I would have liked to refactor even further, and make the vertex data for each object into its own class or struct, allowing me to also supply separate and unique translation, scale, and rotation data for each object in the scene. This would allow for a much more precise URender function, where calls to a separate ‘Model’ function are made with each object passed to it. The Model function would call the appropriate OpenGL functions to manipulate the model matrix and set the uniforms for the shader.