When initially making my choices for the scene, I chose simpler objects as I figured they’d be easier to create. For the most part, I was correct in this assessment. The first object I chose to render was an image holder, which consisted of a long cylinder and two spheres on both ends. This was my complex 3D shape which used multiple primitive shapes. I then chose a volleyball as my next shape, as it represents a simple sphere shape. Next, I used a phone charger block to represent a cube, along with two rectangular prisms on top of it as the outlet plugs. One object I had to change, though, was the glass cup to a thermos. This is because the glass cup was transparent, and including transparency with the shaders hadn’t been discussed. I recognize that I would have to change the alpha value of the texture, but this wasn’t something I could get working. I chose to instead render a thermos, as it uses the same shape (cylinder) without having to change the textures alpha channels. Finally, I used an oak textured plane to set the scene on top off. Many objects within the scene use both a diffuse map and a specular map, though some only use a diffuse map. This is because some of the objects lack a reflective surface. For instance, the charger blocks cube doesn’t use a specular map but the charger parts do.

Users have many options when navigating around the scene. The main mode of navigation would be the use of the WASD keys. ‘W’ allows users to move forward, ‘A’ moves them leftwards, ‘S’ moves them backwards, and ‘D’ moves them rightwards. Users can also move directly up and down relative to the y-coordinates using the QE keys. ‘Q’ moves the user up, and ‘E’ moves the user down. In order to look around the scene, the mouse was used to change the view of the scene relative to the position. If a user scrolled forward with their scroll button, they would be able to “zoom in” by reducing the field of view to a set point. If a user scrolled backwards, they would increase their field of view to a set point. If a user presses the ‘P’ key, they can toggle their view between perspective and orthographic. As users move around the scene, a spotlight will always be pointed in front of them to present how objects react as a light source changes.

One custom function used was loadTexture(), which allowed textures to be loaded in and returned. By not having to write the same code over and over, a lot of space was conserved which made the code look much more appealing. Another function used was scroll\_callback(), which handled the change of fov (field of view) feature. Every time the scroll button was used on the mouse, this function would be called. The mouse\_callback() function is similar, where every time the mouse moves this function is called. This function changes the camera view depending on how the mouse moves. The framebuffer\_size\_callback() function ensures that whenever the program’s window size changes, the program will change the view to keep the same proportions. The processInput() function is called whenever a keyboard key is pressed, and the camera will change accordingly depending on which key was pressed (though this only applies to the previously specified keys, any other key will have no effect).