I chose four objects to design for this project: a reed diffuser, a computer monitor, a coffee cup, and a desk lamp. The reason that I chose these four objects was that I believed they would allow me to design and create objects with varying levels of difficulty. I also believed that they would offer different challenges in the design phase. For example, the textures and materials that I utilized for the reed diffuser were different from those that I utilized for the computer monitor. Also, the physical placement of the objects had varying challenges; while the coffee cup was a relatively simple build with only two shapes to create the finished object, the desk lamp required six shapes and trigonometry to determine the angles and placements of the different objects in order to accurately represent the finished product. The design choices that I made when it came to developing the program were to ensure that it was centered on OOP principles. By separating the functionality of the program into their own functions and being able to call these functions from different spots in the program allowed the development process to be simpler and easier to modify at later points in the project.

The user is able to navigate the 3D scene utilizing the W and S keys to zoom in and out, the A and D keys to pan left and right, the Q and E keys to pan up and down, the O key to switch to an orthographic view, the P key to switch to a perspective view, the mouse scroll to adjust the speed at which the camera navigates, and the cursor to dynamically move around the scene. In order to perform these input-based actions, the program utilizes two different methods. The first method deals with key-based input; while the program is running it checks to see if a key is pressed and processes the appropriate function. The second method deals with mouse-based input; while the program is running it keeps track of the current mouse position and updates it every time the main program loops. If there has been any change in the mouse variables, position or scroll, it calls the necessary functions to update the interface. The main difference is that the first method with the key-based input is only called when an action occurs, while the second method is constantly tracking the details of the mouse-based input.

The custom functions that I incorporated into my program allow it to be more modular and organized through a sort of separation of concerns. These functions each deal with separate parts of the program that control different aspects, such as the lights, materials, textures, colors, and inputs. The main functions that I worked on dealt directly with these different aspects and initialized them to certain specifications in order to correctly replicate the given scene. Other functions in the program are helper functions that don’t directly interact with the interface and user, but are there to help the other functions interact with each other such as the ones dealt with finding specific textures or materials. While I did not explicitly work on these helper functions, these and my custom functions all play into the reasoning behind creating such a separation of concerns within the program – to create a more modular and easily maintained program. When creating an environment for a graphical user interface that will be displaying images such as this one, it can be extremely important to create a modular program. This way if certain aspects of the program want to be updated at a later date, it is easy to identify which sections of the program need to be changed and limit them to only these sections. For example, if I want to change the lighting of the scene, the only function that I need to edit is the SetupSceneLights function. If I wanted to change which objects are displayed or edit the dimensions of those already in the scene, the only function that I would need to update is RenderScene. This design choice allows for the program to be easily reused and changed in the future which can save a programmer and a company a lot of hours and money in the long run.