I decided to select a Dobby Lego picture based off a real Lego set my wife and I built together. With an infinite amount of time, I would have built each block one at a time, but I would have gotten too frustrated with the detail since creative artistic projects give me anxiety (I am never as perfect as I want to be). So I decided to add some non-Lego features and remove the book and sock that he would be holding from the original scene. I chose cylinders and squares and spheres all to remove the Lego aspect and create curves that would better imitate the reality of our CGI friend, Dobby from Harry Potter. I chose half torus’s for the arms and for fear of a lack of perfection in the hands, I fell back on the lego hands as they are more block like in structure. Programming was easy to get the objects on the canvas but difficult to get them to line up to eachother perfectly. I spent a lot of time moving them one pixel at a time until they were placed where they are now.

Movement for this is restricted to mouse and keyboard. I set it up that you can move up and down with the “q” and “e” keys respectively. You can navigate forward, backward, left and right with the “w”, “s”, “a”, and “d” keys respectively. The mouse can be used to change the viewing plane. It allows you to zoom in and look around in a more natural to understand type of way.

To enhance the visual realism of the scene, I implemented the Phong lighting model (to the best of my ability) with four light sources. The primary light source provides a balanced illumination that highlights the overall shape and texture of the objects. The secondary light source, adds depth and a warmer tone to the scene, ensuring that no part of the model is left in complete shadow -except for what looks appropriate given Dobby often lurks around in shadows. This approach allowed me to emphasize the 3D nature of the objects, making the scene more engaging and lifelike. It was really cool to learn how to do it, but I am sure I need much more practice to make it better.

In terms of modularity and organization, I created several custom functions to manage different aspects of the scene. Well, “created” or “copied” from the assignments for my project to build it out. For example, the SetTransformations function handles the scaling, rotation, and positioning of each object, ensuring they are correctly placed within the scene. This function is reusable for any object, making it easy to adjust the layout without duplicating code. Additionally, the SetShaderMaterial function sets the material properties for each object, such as ambient, diffuse, and specular colors, which allows for consistent lighting effects across different objects. It really helped to do the heavy lifting for the model and made it easier to figure out what I needed to change in my code. If everything was messed up, it was something that applied to everything. If one shape was messed up, it was probably just in the part of the code. Not sure if I am explaining that well. By separating these tasks into distinct functions, the code is easier to maintain and extend, and it promotes a clear structure that aligns with best practices in programming.

The SetupSceneLights function is another example of modularity, as it encapsulates the logic for initializing the light sources in the scene. By defining the light properties and positions in one place, it becomes straightforward to modify the lighting setup without affecting other parts of the code. This modular approach not only simplifies the development process but also enhances the readability and scalability of the program, making it easier to introduce new features or adjustments in the future.