For my project I chose particularly simple shapes because I didn’t have faith in being able to do this class. A lot of the objects seemed like they would be easy to create for the scene and to make it fit what the picture actually looked like in the end. I’ll be the first to admit that I had a tough time with this class, probably the toughest time of all of my classes and it had more to do with all of the vector math as well as new concepts being thrown at me. I’ve never been an artistic person and this, to me, seems like it comes more naturally to those who are art inclined. That may not be the case entirely considering it seems to be more about math than anything else. Math and experimentation in order to get things just right. That still doesn’t mean that the spatial portion of it is easy to understand. As a concept it is easy to understand the x, y, z axis and how that correlates to what is on your screen and how your camera correlates to it.

While these concepts aren’t necessarily difficult to understand, how to implement them and the math required as well as the way the computer interprets your inputs is a difficult concept to understand. So while I chose virtually 3 boxes and a cylinder, plus torus’ on one of those boxes, it became a much harder concept to put into play when you account for scalability, transformation, and textures.

For the lego brick it seemed pretty easy in order to make a box and then create torus’ to place atop it to interpret the connectors on a lego brick. This became harder to do as I realized that the rotations of the object of each torus didn’t work exactly as I thought it would. When I tried to rotate a torus on the x axis by 90 degrees, it didn’t work with simple numbers of 90 degrees. I actually had to play around with several different values to realize that the rotation around the x axis seemed to not be based entirely on 90 degrees, or at least not in the raw number of 90. I think I ended up with placing a value of about 2 or 3 in order to get it to rotate just enough to sit atop the brick like a lego brick connector would. Then you had to consider, for all objects, how they would sit on the plane and not clip through the plane which is supposed to be the ‘bottom’ of the scene. In order for it to be realistic, you can’t have an object poking out of the bottom fo the table. Where this tripped me up was actually with the Cylinder. While the boxes (starting with the lego brick) didn’t quite require a full transformation on the y-axis to match the plane’s thickness, I realized that the cylinder actually needed less of a transition upward. I assume this had to do with how the shader program was drawing the cylinder for the eye glass cleaner in the first place.

The next step that kind of was interesting to me, but probably the hardest of all was the lighting. Figuring out how to apply these in the best way was an interesting puzzle to complete. I settled on having two lights and having one of those lights create a light purple light in order to kind of simulate a bit more darkness coming from other angles but still lighting the object from the residual light in the room. Having an entire side be just completely dark doesn’t make sense, but it also doesn’t make sense for it to be super bright based on the direction the lighting is coming from or where other objects might be blocking the light. By making it a light purple light, it still looks somewhat natural but gives it a bit more of a darker look so that it looks more like shadows.

I will say one section I had exceptional difficulty on was creating the texture for the dice box. The lego brick was easy as the texture for it was just a solid color really, so applying the texture makes it look more realistic than just a flat color, but in the end I didn’t have to deal with stretching or duplicating a resolution for the lego brick texture. Just the natural settings worked. For the table, I used the original photo of the table I took for the plane, but it had a huge glare in it from the light above so it looked like I had spotlights ont he table. I opted to find a different texture that looked more natural without those glared spots and then I had to flip that texture to make it look more seamless, so I used mirrored repeat. The real trouble came with the dice box, and I’m sorry to say I still couldn’t get that fully right. I couldn’t figure out a way for it to NOT segment the picture into four repeats on each side of the dice box. And when I clamp to the edge, it just created one fourth of the side as the picture and then had stretched pixels going to the rest of the edges. It just looked ridiculous. So I opted for cutting the picture into four of the dice faces and then allowing it to repeat that texture to make it look more like the appropriate dice case. But unfortunately the top of the case still is completely wrong. I could not understand how to manipulate textures well enough to fix this unfortunately.

In my particular project, I also implemented the use of mouse and keyboard controls in order to navigate the scene from a ‘first person’ point of view. Using the mouse scroll wheel will also allow you to control the speed of the camera, by pushing the wheel forward you get a higher speed increase and pulling it backward will slow the camera down. One thing I realized in development is if I didn’t put a cap on the slow down, eventually you would reverse the values of the mouse and therefore cause all of the directions to reverse in kind. In order to fix this, I put a cap on how low you could actually slow the camera down so that it wouldn’t be immovable but also that it couldn’t be reversed in direction. With the Q and E keys, you can move the camera up and down and with the WASD keys, you can move the camera around the scene as if you were walking in those specific directions. Lastly if you press O, you can switch to an orthographic display, which also moves the camera to look directly at the scene from exactly at the x axis, which also makes the plane disappear. Pressing P will return the view to a perspective view which looks more realistic.

Some of the custom functions I created were just for the drawing of the different models. In this way I was able to separate them out and even collapse portions of the code in the Visual Studio IDE in order to make visibility cleaner on my screen. Instead of hardcoding everything into the render I could actually just call the individual functions in order to create the different objects in the scene. This allows for a level of modularity in that you can just set one function in order to draw the object, or even comment that function out in case you want to test something without that object, without having to comment out large blocks of code. It also provides you a simple work space to manipulate all of the values dealing with that one object without having to scroll your screen or see other code all around whatever you’re working on and getting confused.

I also considered separating the lighting functions from each of the objects in order to make it further modular, but it seemed to make less sense when you considered that you would then have to pass all of the values into the arguments of the function in order to make the lighting aspect more modular. It made more sense to go into each individual object and manipulate the coding manually from there. In the future I suppose it could be easier to separate the lighting into another function and then call that function from within the object model in order to manipulate those values ,but there are a lot of variables to consider in those cases that you have to pass through.