**Final Reflection**

For my scene, I went with making a 3D graphic representation of some of the items that would be found in a seven-layer parfait bowl. One of the first items that I decided to create was the model for the “Lady-Finger.” One of the key reasons for this was because of the fact I figured this shape would be the hardest to make. This ended up being true, as it was also the shape that helped cross off the checklist item of “making a complex shape”. I was able to accomplish this by merging some circles as well as a rectangle to help create the shape of the biscuit wafer.

I also chose to make a plane for one of my other objects. This helped set a scene for my creation and was a reference point for the other objects. The plan can help show the relationship of the object to the world, such as knowing if the item is supposed to be large or is it small? Without an object to use as reference, it can become difficult to try and understand how large or small an object within the world space is supposed to be. I also chose to throw in two more primitive shapes, a cube as well as a pyramid. The cube is used to represent the small bits of sponge cake that could also exist within a seven-layer parfait. Originally, I was going to make a cone instead of a triangle, however given the time restraints I had, I chose for the shape that was easier to make and somewhat representative of strawberries.

Regarding the textures used, I chose to use what I could find to be appropriate for the lady finger biscuit. The cube is textured to look like a sponge cake, respectively. The pyramid, as mentioned in the previous paragraph, is textured to look like a strawberry. I chose to wrap all my objects in a texture to make sure that my understanding of how to work the texture mapping was solid.

Navigating around the scene that I created is rather simple, and one could compare it almost to any standard form of FPS computer games in today’s time. There were nuanced controls that were added however, which would include the “Q” and “E” keys to control the camera’s position along the y-axis (assuming the mouse isn’t being used). The mouse can also be used to alter how the camera behaves within the scene. If the left mouse button is being held down, the user can use the mouse to be able to look in more specific directions. If no keys are being used, the mouse holding function can be used to cause the camera to “stand still” and be able to look all around the scene. Of course, it’s possible to use both the keys and the mouse buttons if the user felt comfortable with the two inputs simultaneously. Should the user want to see more than just what is being projected, pressing the “P” key will change the “perspective” of the camera to allow it switch to an orthographic view. This view type can display the whole image, or object rather, regardless of whether the object is completely within the camera’s current projection view or not.

When it comes to custom functions, I don’t believe that I really excelled in this field too much as I found myself having a difficult time understanding writing with this language. I was able to satisfy most if not all of the requirements needed for this project, so that is considered a win in my book. I believe that if I spent more time, I could have modularized my objects by putting them into a Mesh class of some kind. This method of modularization on my code was being done early on, and while some of the code is modulated (such as the fragment and light shaders as well as the VBO and VAO classes) there are areas of the code I wish I spent some more time in. One of the larger ones being the ability to have my objects be called in a function and be able to be drawn and translated into its proper spot. This would make the integration of objects within the world a much less tedious task.