**Reflection – Design Decisions**

**Development Choices Justification**

I chose my selected objects to mix complexity with simplicity for a roundabout beginning foray into OpenGL. I also picked these four objects to fulfill the project guidelines of including at least one complex object (i.e., an object requiring more than one type of shape). The yarn skein, lotion bottle, and coffee mug all utilize more than one type of 3D shape.

I found that the yarn skein was the optimal object for my skill level—it was difficult but doable, and I’m proud of it because it’s a culmination of several carefully arranged shapes and textures. The mug was simpler but still complex, just enough that it was satisfying to put together, especially with the implied coffee filling. The board game box was easy to implement, as it was one shape, but seeing the actual Splendor image as part of its texture added a lot to its visual attractiveness. The lotion bottle felt too complex for my current skill level, and I had to make a compromise between reality and the shapes/time I had to work with. I wish I’d used a different object in my photo reference, but at the time of developing the project proposal, I had no real OpenGL experience to base my choices off of.

**Scene Navigation**

As developed in prior module milestones, the scene can be navigated using keyboard and mouse controls. The WASD keyboard keys are used to move forward, left, backward, and right (respectively), while the QE keys are used to move up and down respectively. Mouse movements control the direction of the camera, and rolling the mouse wheel forward or backward determines the speed at which the camera moves through the scene, allowing the viewer to home in on any particular part of the scene. Additionally, the O key switches to a front-facing orthographic view in which the bottom plane is not visible, and the P key switches to a perspective view, which is the same as the one seen upon executing the scene.

**Custom Functions for Modularity**

Using the OpenGL sample as a reference, each main object was placed into its own function (like “RenderCoffeeMug()”) and called in RenderScene(). Because most of the objects developed here use more than one shape, this kept the code more organized than if every shape was lumped into one function. This is because the code gets visually confusing or overwhelming quickly with so many blocks dedicated to the smaller shapes that make up larger objects. Splitting larger objects up into functions breaks up the complexity.

When a program is well-organized, it’s easier for *all* developers to pinpoint specific locations in the code, resulting in an efficient workflow. I further organized my code with comments that note what shapes correspond to what parts of the photo reference, like a cylinder representing the lotion bottle cap, or why some textures are shinier than others. Modular, well-commented code means that anyone, whether it’s me or a team member, can understand and edit it, and that’s what it means for it to be reusable.