Final Project Reflection

# 9. Justify development choices for your 3D scene.

For this 3D scene, I selected a realistic everyday setup—an orange, bottle, mug, and knife—based on a still life reference. These objects were chosen to demonstrate a variety of shapes (sphere, cylinder, tapered cylinder, torus, prism, box) and texture mapping techniques. The goal was to balance complexity with clarity while ensuring the scene showcased proficiency in transformations, material settings, and lighting. I programmed object construction using modular reusable transformation and rendering functions, which made it easier to place, scale, and rotate objects precisely. Additionally, I made deliberate lighting adjustments to avoid overexposure while still illuminating all components, ensuring a visually compelling result.

# 10. Explain how a user can navigate your 3D scene.

User navigation was enabled through camera control using keyboard and/or mouse input handled in the SceneManager or view class. This allows the user to rotate the camera around the scene, zoom in and out, and pan across the objects. These interactions are typically managed by capturing input events (such as mouse movement for rotation and scroll for zoom) and applying them to the camera’s view matrix using transformation math (e.g., gluLookAt or a custom view matrix setup). This interactivity enhances the user experience by allowing dynamic exploration of the scene from multiple angles.

# 11. Explain the custom functions in your program that you are using to make your code more modular and organized.

Several reusable helper functions were implemented to streamline the scene's construction. For example:  
- SetTransformations(): applies scaling, rotation, and position in a unified way to any object.  
- SetShaderMaterial(): modularizes how materials (ambient, diffuse, specular) are assigned.  
- SetShaderTexture(): handles the binding of texture identifiers to shapes.  
- Draw[Shape]Mesh(): functions encapsulate how each geometry type is rendered.  
  
By isolating these responsibilities, the program becomes much easier to maintain, debug, and extend. Any new object can be added with just a few lines using these modular calls, without rewriting transformation or shader logic.