**Project Reflection**

In developing my 3D scene using OpenGL, I chose a cereal bowl, eggs, and a rolling pin on a tabletop as my primary objects to simulate a kitchen setting, aiming for both realism and functional interactivity. The cereal bowl was represented by a hemisphere for the bowl's body and a torus for its rim, chosen for their geometric resemblance to real-world objects. I utilized detailed parameterization (e.g., stacks, sectors, and radii) to accurately model the bowl's curvature and depth, ensuring a visually convincing representation. For the eggs, I employed scaled ellipsoids to capture their distinctive shape, tweaking the x, y, and z scale factors to achieve an egg-like appearance. This approach allowed for a more natural look, essential for enhancing the scene's realism. The rolling pin was modeled using cylinders, with one representing the body and a thinner, elongated one for the handles, mirroring the typical structure of rolling pins. This choice was driven by the need for recognizable kitchen utensils that users could intuitively interact with, further immersing them in the virtual environment. My programming choices, including the manipulation of geometric transformations and careful adjustment of object positions, were aimed at achieving a cohesive scene where objects not only looked authentic but were also positioned in a way that reflected natural spatial relationships, enhancing the overall user experience.

In my 3D, I have set up intuitive navigation controls to move around seamlessly. By initializing GLFW and GLEW, I created a window where users can explore the scene. The camera is controlled through keyboard inputs for movement—'W', 'A', 'S', 'D' for forward, backward, left, and right, respectively, and 'Q' and 'E' for up and down movements, all of which are adjusted for consistent speed with `gDeltaTime`. Additionally, the mouse plays a critical role: moving the mouse changes the camera's view direction, while scrolling adjusts the zoom, offering a first-person exploration experience. Mouse buttons serve for extra interactions, like acknowledging clicks without affecting navigation. This setup, combined with GLFW's callback functions for mouse and keyboard events, allows users to intuitively navigate and interact with the objects on the tabletop, enhancing the immersive experience of the scene.

To render my 3D scene with a cereal bowl, eggs, and a rolling pin on a tabletop, I developed several custom functions to enhance modularity and organization, ensuring each aspect of the scene setup and rendering process is neatly compartmentalized. Functions like `UInitialize`, `UResizeWindow`, `UProcessInput`, `UMousePositionCallback`, `UMouseScrollCallback`, and `UMouseButtonCallback` are foundational, handling window initialization, input processing, and dynamic adjustments based on user interactions. For graphics rendering, `UCreateMesh` and `UDestroyMesh` manage the lifecycle of mesh objects, while `UCreateTexture` and `UDestroyTexture` deal with texture creation and cleanup, ensuring efficient memory use. The `URender` function orchestrates the drawing of the scene, and shader management is streamlined through `UCreateShaderProgram` and `UDestroyShaderProgram`, facilitating the use of GLSL for realistic lighting and texturing. Additionally, `flipImageVertically` addresses the discrepancy between image loading conventions and OpenGL's coordinate system, making texture application more intuitive. These functions are reusable across various elements of the scene and can be adapted or expanded for future projects, demonstrating a scalable approach to 3D graphics programming.