## Development Choices

The design of the 3D scene was based on creating a realistic and immersive environment. The selected objects were chosen to evoke the feel of a study or laboratory setting, with items such as a table, test tubes, a test tube rack, and various accessories like a candle, beaker, inkwell, and papers. These objects contribute to the scene’s overall narrative, which can be interpreted as a space for experimentation, research, or even artistic creation. The candle, in particular, introduces a dynamic element by providing flickering light, enhancing the realism of the scene. The table and wall serve as basic structural elements that ground the other objects in the scene. Programming the scene required breaking down the rendering process into manageable functions, such as loading meshes, applying textures, and adding light sources. The *PrepareScene*() function, for example, ensures that all textures, materials, and meshes are properly loaded into memory, while the *RenderScene*() function handles the drawing of the objects, using modular drawing functions for individual items like the table, test tubes, and candle. This approach ensures that each object can be easily added or modified without disrupting the entire scene, making the code more efficient and maintainable.

## Navigation

User navigation within the 3D scene is facilitated through both mouse and keyboard inputs, which control the movement and orientation of the camera. The mouse controls are implemented through the *Mouse\_Position\_Callback()* and *Mouse\_Scroll\_Callback()* functions. When the user moves the mouse, the camera’s orientation is updated based on the difference in the mouse’s X and Y positions, which allows the user to freely look around the scene. Additionally, the mouse scroll functionality is used to zoom the camera in or out by adjusting the camera’s zoom level. Keyboard controls, handled by the *ProcessKeyboardEvents()* function, allow the user to move the camera forward, backward, left, right, up, and down using the WASD keys and Q/E for vertical movement. Furthermore, the camera’s projection mode can be switched between perspective and orthographic views using the P and O keys. These input mechanisms together provide an intuitive and responsive navigation system, allowing the user to explore the 3D environment by adjusting the camera's position and view in real-time.

## Modularity and Organization

In this code, modularity and organization play a crucial role in maintaining readability, flexibility, and scalability, especially for a complex OpenGL scene creation task. Modularity is achieved by encapsulating distinct tasks into individual functions, such as *DrawTestTubes()* and *DrawTestTube()*, which handle specific drawing operations for the test tubes in the scene. The *DrawTestTubes()* function is responsible for positioning and rendering multiple test tubes along an angled rack, using trigonometry to calculate each test tube's position based on the rack's angle and spacing. This function iterates through each test tube, calculating its X and Z offsets using cosine and sine, respectively, to distribute the tubes along the rack in a precise and organized manner. The *DrawTestTube()* function, on the other hand, focuses on rendering the individual test tube, including its liquid base, glass top, and torus, while adjusting transformations, textures, and materials accordingly.

By encapsulating the logic for drawing components into distinct functions, the code becomes highly modular, allowing for easier debugging and maintenance. Changes to how test tubes are drawn or how their positioning is calculated can be made in the respective functions without affecting the rest of the codebase. This modular approach also enhances reusability, as the drawing functions can be reused for other objects or scenes, and it makes the code easier to extend with additional objects or features in the future. Furthermore, the clear separation of concerns between functions like drawing the table, walls, and individual items ensures that each piece of the scene is organized and that the responsibilities of each function remain distinct, which ultimately makes the code more maintainable and scalable.