1. **Justify development choices for your 3D scene**. As you write, think about why you chose your selected objects. Also consider how you were able to program for the required functionality. The items or scene I chose to replicate in 3D consist of a table, a candle vase, a cup, a ball, and a Bible. The project required at least four primitive shapes or meshes to be used to represent the different items in 3D. Also, the scene should include a complex item which would feature more than one primitive shape to render in 3D. I selected the table, candle vase (with lid), a ball, a glass cup, and a Bible for my class project. The complex items here are the candle vase and the lid and the Bible. The rest of the items can be represented using a single primitive shape. For the most part, a combination of the same mesh or primitive shape was used to achieve the complex shapes. For the table, I used a plane to represent it in 3D. The candle vase and lid were replicated in 3D using a combination of cylinder meshes. The ball was represented in 3D using a sphere. The glass cup was also created using a cylinder mesh and the Bible was created using a host of box meshes (cubes). A meshes.cpp file and a meshes.h files were created as external files which I imported into my source file to draw the various shapes. To achieve the required functionality, especially dealing with the location of the items, I applied my knowledge of transformations to translate and move objects to the correct positions as depicted in my project proposal. It was mostly try and error and experimenting with figures to achieve a more intuitive and meaningful result.
2. **Explain how a user can navigate your 3D scene**. As you compose your thoughts, discuss how you set up to control the virtual camera for your 3D scene using different input devices. As can be seen throughout the class, it is easier to navigate the project with both the mouse and the keyboard. It is more personable using that approach. A camera.h header file was created with all camera movements implemented in code. Forward, backward, left, right, up, and down movement of the camera are all captured here. This file is then imported into the main source file to use in our UProcessInput() function. This function is defined to call the camera.h file movement implementations when certain keyboard keys are entered. For instance, when the key S is pressed, the camera.h backward movement implementation is called gCamera.ProcessKeyboard(BACKWARD, gDeltaTime). The mouse is also set up to call the camera movement functions in the camera.h so that should the user move the mouse in certain direction, the camera follows.
3. **Explain the custom functions in your program that you are using to make your code more modular and organized**. Ask yourself, what does the function you developed do and how is it reusable? As mentioned in this reflection, some important implementations were outsourced to some external files such as meshes.cpp and the meshes.h including the camera.h header files. By creating and importing these external files, the possibility of running into naming issues can be avoided. Also, by doing this, the source code was kept clean, well-structured, and organized. Additionally, files like stb\_image.h were imported into the main source code .cpp file to use textures.