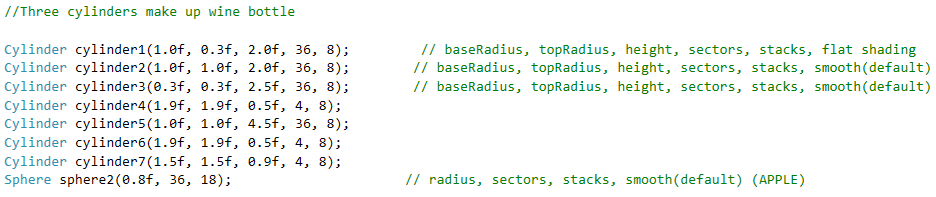
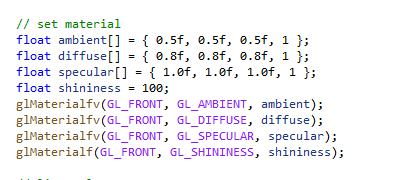
**Development Choices**

Objects in Scene are: A bottle, a can, an apple, a book, and a box. The wine bottle was the hardest object to make. In order to do so I used a combination of three cylinders. For example, the neck of the bottle is just a cylinder with a different radius on top and bottom, so it cones to a smaller radius. The neck has a very small radius while the base of the bottle has the biggest radius. The can was a standard cylinder, the apple a sphere, and for the book and box I actually used my cylinder class to create them. I could only make a square with that method, so I just made two of them for the book, to make it rectangular. The box was just a square using that method. The cylinder class was used with a method call in the main.cpp class that took in values of base radius, top radius, height, sectors, stacks, and shading. The Sphere class was similar and took in values of radius, sectors, stacks, and shading. An example of those method calls is shown here below:

**User Controls**

The most convenient user control for my scene is the mouse. I set it up so when you left click and hold the button down, you can move the mouse in any direction around the scene (up, down, left, right). There is also Keyboard controls the user can use. The key ‘a’ moves user to the left of the scene, while key ‘d’ moves user to the right of the screen. The key ‘e’ moves user down the screen, while ‘q’ moves them up the scene.

**Lighting**

As the user moves around the scene, you will notice that the lighting changes. There is specular, ambient, and diffuse light set up in the scene. Methods were made to control this with an example of those method calls below.

**Custom Coded Functions**

The use of methods and classes to create Spheres, Cylinders and lighting in my code make it much neater in the main.cpp class. It is nice to be able to call the cylinder or sphere class to create a cylinder or sphere for each use and not have to write the code to do so, but just call the method and pass it the radius values. The code is also commented to show what is happening which makes it much easier to follow. I also included simple texture names to make the texture view more obvious about which image/texture goes with which object. This was a great learning experience in this class.