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.

* Overly ambitious scene with oddly structured objects, and multi-level/stair to window. Would have chosen an easier scene if I had anticipated difficulties. Chose to not implement raised window due to time constraints. The tape and stapler object were implemented as simple cuboid shapes but would be better represented as complex shapes.
* Applying texture is difficult. I opted to use multiple planes for tape and stapler to allow for different textures to be applied to the sides. The two cylindrical objects are created using a for loop for rotations, so I am not sure how you would get a non-rotation texture on them.
* Used a single light source with high ambient light because I believe this is true to my scene.

When I first chose the image for my scene, I was aware that the scene had some complexities that would be difficult to implement. Unfortunately, I had a lot of difficulty with OpenGL. I struggled to add multiple shapes, and to figure out how to use the model and view matrices to correctly place objects. I have taken a few shortcuts in my scene to deliver a finished product. I did not try to implement the multi-level window ledge. With my current understanding, I don’t think adding the additional planes would be too difficult, but I don’t have pictures for the textures, and reorienting the objects could get tedious. My shapes are very rough approximations of the objects they represent. I also took some liberties with the textures to best represent the objects. In summation, many specificities of the scene have been overlooked to be able to deliver a complete scene on time.

1. **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.

* Complete 3d control using six keyboard buttons (Q, W, E, A, S, D) and mouse.
* Movement speed adjustable with mouse wheel
* Able to change between perspective and orthographic view with P button.

Implementing the camera was by far the easiest part of the project. Using the camera.h file from learnopenGL made the process very smooth. I added some functionality to the camera file to allow for movement on the y axis based on milestone requirements.

1. **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?

* Expanded learnopenGL’s camera.h file to process movement on the Y axis (up and down).
* Added function to change between perspective and orthographic view.
* Updated and created new shaders for processing textures and light sources.
* Used multiple function definitions for organizational purposes based on classes’ tutorial structure.

The code in the project would benefit from refactoring to make it more reusable. I would start by moving most of the large global variables (like the vertices arrays) into separate files. I would also like to see the create mesh function reworked to do the same thing for each mesh, and take a mesh in as an argument. This would eliminate a lot of the redundant code in that function. Also, the drawing phase in the render function is structured in repetitive processes (clear model, transformations, update shader, draw) that could probably be made into a function.