For my project I had chosen to replicate a scene displaying a fast-food meal. The scene contains a burger, fries in a container, a fountain drink cup, and a box with chicken nuggets. I chose the objects in my scene because they allowed me to use several basic low-polygon 3D shapes. The burger in the scene is made up of several spheres for the two buns and the beef, as well as a flattened box to represent the cheese. Originally the burger buns were meant to be made from half-spheres. However, I found implementing a half sphere to be significantly more difficult than I anticipated and replaced the buns with slightly flattened spheres instead. The fries are composed of tall, thin boxes for each individual fry and the fry box is made using an upside-down tapered cylinder. The fry box was initially meant to be made using a box based off the original 2D scene, however I found that a tapered cylinder looked more like a realistic fry box. The nuggets are made using flattened spheres and the cardboard container is made using two boxes (with one at a -45-degree angle to appear like an open box). The fountain drink cup was made using an upside-down tapered cylinder for the cup and a torus and flattened sphere for the lid, as well as a tall thin cylinder for the straw. I used the transformation functions: scaleXYZ, X, Y, and Z rotationDegrees, and positionXYZ to set the scale, rotation, and position of each basic shape so that the shapes aligned with the original 2D scene.

The user can navigate the 3D scene using keyboard and mouse inputs. The user can move forward, backwards, and side to side in the scene using the W,A,S,D keys. The user can move up and down using the Q and E keys. With the mouse cursor the user can look in any direction while remaining in the same position. This was done using mouse position callback functions that record the changes in cursor movement and use those movements to adjust the camera’s view. The mouse scroll wheel is used to adjust the movement speed of the camera when using the W,A,S,D keys; scrolling forward increases movement speed, while scrolling backwards decreases speed. The user can also switch to perspective or orthographic views of the scene by pressing P and O respectively. The perspective view displays the scene in an immersive view which shows depth in the scene whereas the orthographic view displays the scene in a top-down view that shows the scene and objects in a 2D less immersive view that allows the user to see the objects size and placement similar to a blueprint.

To ensure the program was organized and modular, I used several custom functions. I developed the loadSceneTextures() function to load all of the textures for my objects into GPU memory. The function makes the code more organized and makes it easy to find and modify the textures that are being loaded into memory. I also created the DefineObjectMaterials() function which stores the different materials used for the objects in the scene. Similar to the loadsceneTextures() function, this function keeps code organized and makes it easy to find and modify the materials for objects without affecting other parts of the code. Another function I created was the SetupSceneLights() function which initializes the three light sources in the scene. This function contributes to organized code by holding all of the code to modify the lights and separating it from the rest of the code. Lastly I created the SetCamera() function to set the camera’s position, orientation and zoom when switching between perspective or orthographic camera views. This function keeps code organized as it separates the functionality to reset the camera’s when switching from perspective to orthographic allowing the ProcessKeyboardEventes function to stay clear and concise making it easier to understand.