For this assignment, I chose to recreate a scene one would find as they are mini golfing. I recreated a scene of a putting green, with a blue golf ball, a windmill for the ball to go through, and two flowerpots on the sides of the green. Some of these objects included multiple shapes, whereas some only required one. For example, the golf ball was created with a sphere mesh and the green and ground were created using plane meshes. The windmill was created using two box meshes scaled to fit more of a rectangular prism shape and a tapered cylinder. For the flowerboxes I used box meshes. For the flowers in the boxes, I did experience a little difficulty getting the shapes to accurately resemble that of a flower. I used both a cone mesh and tapered cylinder to represent the flowers. I used two different meshes to help replicate the uniqueness of flowers in real life, but I was unsure of what shape would most accurately represent the flowers in the picture. Overall, I think with more practice or knowledge I could've done better with the shape, but I still think they are able to resemble the flowers.

While I was completing this assignment, I did experience some issues when adding the lighting for the scene. Before adding any light sources, the user could see the entire scene and all the objects within it from any angle. However, as I added the light sources, my scene became dark, and half of the objects were hidden. I played around with the position and intensity of the

light sources, as well as trying to change the textures and materials of the objects, but I was still facing this issue.

```
void SceneManager::SetupSceneLights()
{
    // this line of code is NEEDED for telling the shaders to render
    // the 3D scene with custom lighting - to use the default rendered
    // lighting then comment out the following line
    m_pShaderManager->setBoolValue(g_UseLightingName, true);

    //First light source
    m_pShaderManager->setVec3Value("lightSources[0].position", 5.0f, 0.0f, 0.0f);
    m_pShaderManager->setVec3Value("lightSources[0].ambientColor", 0.1f, 0.1f, 0.1f);
    m_pShaderManager->setVec3Value("lightSources[0].diffuseColor", 0.5f, 0.5f, 0.5f);
    m_pShaderManager->setVec3Value("lightSources[0].specularColor", 0.2f, 0.2f, 0.2f);
    m_pShaderManager->setFloatValue("lightSources[0].focalStrength", 70.0f);
    m_pShaderManager->setFloatValue("lightSources[0].specularIntensity", 0.2f);
```

In addition, I included textures for certain objects in my scene. For example, I used a green shingle texture for the material on the windmill. I also used pink and purple textures to represent pink and purple flowers in the boxes. Lastly, I used a stone texture on the flower boxes. These textures help add realistic features to my scene and help light sources reflect off them in different, realistic ways.

```
void SceneManager::DefineObjectMaterials()
{
    OBJECT_MATERIAL windmillMaterial;
    windmillMaterial.ambientColor = glm::vec3(0.2f, 0.2f, 0.2f);
    windmillMaterial.ambientStrength = 0.3f;
    windmillMaterial.diffuseColor = glm::vec3(0.2f, 0.2f, 0.2f);
    windmillMaterial.specularColor = glm::vec3(0.5f, 0.5f, 0.5f);
    windmillMaterial.shininess = 22.0;
    windmillMaterial.tag = "greenshingle";

    m_objectMaterials.push_back(windmillMaterial);
```

There are multiple ways for users to navigate throughout my 3D scene. Navigating can be done with just a couple keyboard clicks. To move forward in the scene, a user can press W and to move backward, a user would press S. To move left, press A, and to move right, press D. Additionally, Q is pressed to move up and E is pressed to move down. There is also an

orthographic view, which can be accessed by clicking O on your keyboard, and a projection view, which can be accessed by clicking P on your keyboard. Although I believe I have the code right for the orthographic view, I am not entirely positive I did this step correctly. Lastly, you can also use the mouse to control some of the camera movements and features.

```
// process camera zooming in and out
//If w key is hit, move forward
if (glfwGetKey(m_pWindow, GLFW_KEY_W) == GLFW_PRESS)
{
    g_pCamera->ProcessKeyboard(FORWARD, gDeltaTime);
}
//If s key is hit, move backward
if (glfwGetKey(m_pWindow, GLFW_KEY_S) == GLFW_PRESS)
{
    g_pCamera->ProcessKeyboard(BACKWARD, gDeltaTime);
}
```

There are many functions used in this code that help make it more readable and modular. For example, there is a renderScene() function where all the objects are described and declared. For each object in this function, the scale, rotation, position, transformation, shader texture or color, UV scale, and shader material are all described, and then the mesh is declared. Another example is the LoadSceneTextures() function, which houses all of the jpg files for the textures needed in the scene. The SetUpSceneLights() method is used to declare and call the light sources and set each of their ambient, diffuse, and specular colors. Finally, the DefineObjectMaterials() function helps to define each material and how the lighting will reflect off it. Including functions helps make the program easier to read, more organized, and easier to replicate in future projects.

```
//right flower box shape

// set the XYZ scale for the mesh
scaleXYZ = glm::vec3(3.0f, 3.0f, 3.0f);

// set the XYZ rotation for the mesh
XrotationDegrees = 0.0f;
YrotationDegrees = 0.0f;
ZrotationDegrees = 0.0f;

// set the XYZ position for the mesh
positionXYZ = glm::vec3(7.0f, -2.0f, -5.0f);

// set the transformations into memory to be used on the drawn meshes
SetTransformations(
    scaleXYZ,
    XrotationDegrees,
    YrotationDegrees,
    ZrotationDegrees,
    positionXYZ);

//SetShaderColor(0.827, 0.686, 0.494, 1);
SetShaderTexture("flowerBox");
SetTextureUVScale(1.0, 1.0);
SetShaderMaterial("greenshingle");

// draw the mesh with transformation values
m_basicMeshes->DrawBoxMesh();
/**************************************************************/
```