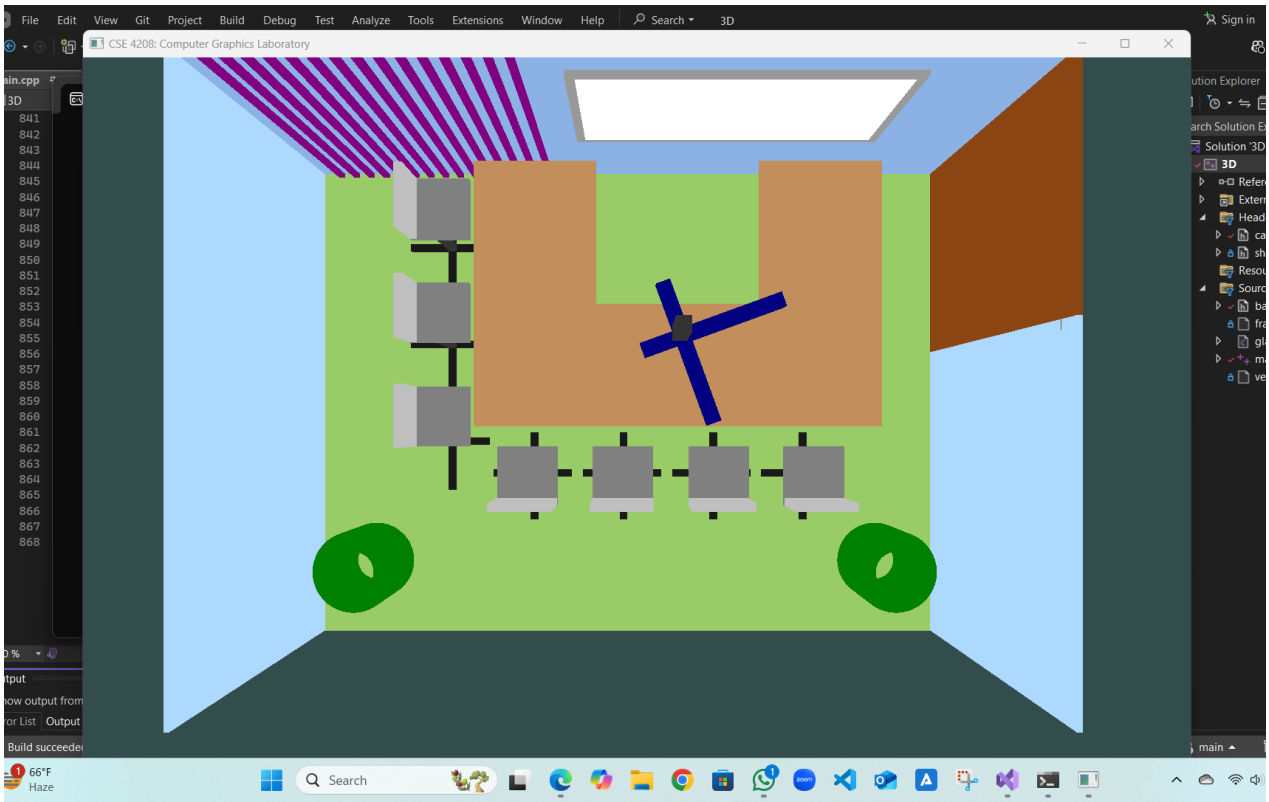
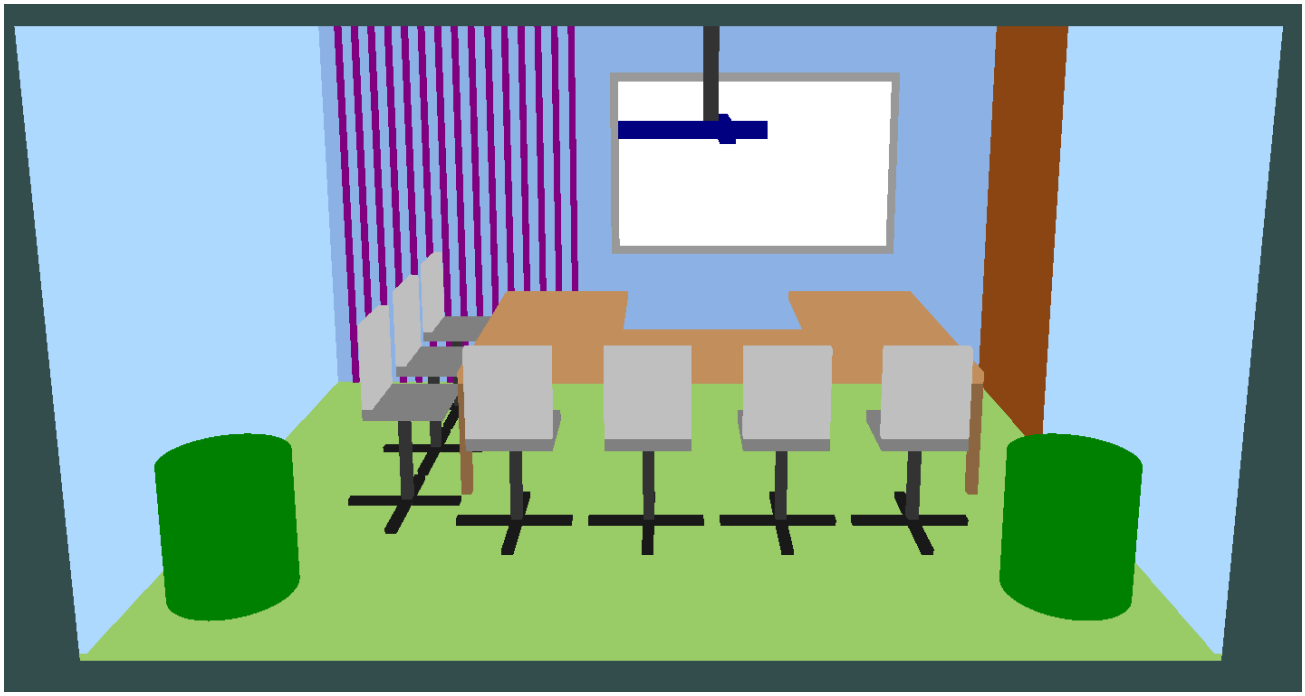
```
278         if (birdEyeView) {
279             glm::vec3 birdEyePosition(0.0f, 10.0f, 0.0f); // Camera above the scene
280             glm::vec3 birdEyeTarget(0.0f, 0.0f, 0.0f);    // Focus on origin
281             glm::vec3 upVector(0.0f, 0.0f, 1.0f);        // Adjust the up vector
282             view = customLookAt(birdEyePosition, birdEyeTarget, upVector);
283         }
284         // Rotate around a point logic
285         else if (glfwGetKey(window, GLFW_KEY_F) == GLFW_PRESS) {
286             view = rotateCameraAroundPoint(); // Rotate around the look-at point
287         }
288         else if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS) {
289             view = customLookAt(eye, lookAtPoint, upVector);
290         }
291         else if (glfwGetKey(window, GLFW_KEY_Y) == GLFW_PRESS) {
292             view = customLookAt(eye, lookAtPoint, upVector);
293         }
294         else if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS) {
295             view = customLookAt(eye, lookAtPoint, upVector);
296         }
297         else {
298             view = basic_camera.createViewMatrix();
299         }
```



```
654         //cylinder-1
655         translateMatrix = glm::translate(identityMatrix, glm::vec3(4.5f, 0.0f, -1.8f));
656         scaleMatrix = glm::scale(identityMatrix, glm::vec3(0.5f, 2.0f, 0.1f));
657         rotateYMatrix = glm::rotate(identityMatrix, glm::radians(r), glm::vec3(0.0f, 1.0f, 0.0f));
658         model = globalTranslationMatrix * translateMatrix * glm::translate(identityMatrix, glm::vec3(0.5f, 0
659         ourShader.setMat4("model", model);
660         ourShader.setVec4("color", glm::vec4(0.0f, 0.5f, 0.0f, 1.0f)); // Deep green color
661         glBindVertexArray(VAO);
662         glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
663         int x = 1;
664         for (int i = 1; i < 360; i++)
665         {
666             translateMatrix = glm::translate(identityMatrix, glm::vec3(4.5f, 0.0f, -1.8f));
667             scaleMatrix = glm::scale(identityMatrix, glm::vec3(0.5f, 2.0f, 0.1f));
668             rotateYMatrix = glm::rotate(identityMatrix, glm::radians(r - i), glm::vec3(0.0f, 1.0f, 0.0f));
669             model = globalTranslationMatrix * translateMatrix * glm::translate(identityMatrix, glm::vec3(0.5
670             ourShader.setMat4("model", model);
671             ourShader.setVec4("color", glm::vec4(0.0f, 0.5f, 0.0f, 1.0f)); // Deep green color
672             glBindVertexArray(VAO);
673             glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
674         }
675
```



```cpp
glm::mat4 RotationMatricesX(float theta) {
    float cosTheta = cos(theta);
    float sinTheta = sin(theta);
    return glm::mat4(
        1.0f, 0.0f, 0.0f, 0.0f,
        0.0f, cosTheta, -sinTheta, 0.0f,
        0.0f, sinTheta, cosTheta, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    );
}

glm::mat4 RotationMatricesY(float theta) {
    float cosTheta = cos(theta);
    float sinTheta = sin(theta);
    return glm::mat4(
        cosTheta, 0.0f, sinTheta, 0.0f,
        0.0f, 1.0f, 0.0f, 0.0f,
        -sinTheta, 0.0f, cosTheta, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    );
}

glm::mat4 RotationMatricesZ(float theta) {
    float cosTheta = cos(theta);
    float sinTheta = sin(theta);
    return glm::mat4(
        cosTheta, -sinTheta, 0.0f, 0.0f,
        sinTheta, cosTheta, 0.0f, 0.0f,
        0.0f, 0.0f, 1.0f, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    );
}
```