**Description:**

In this assignment, a 3D dining room scene has been designed using various geometric shapes, including two toruses, a cone, and a cylinder, which are used as point lights within the scene. Directional and spotlight effects have also been incorporated. Interactive controls through the keyboard allow the lights to be toggled on and off, and the lighting properties, such as ambient, diffuse, and specular components, can be adjusted.

**Program and Results:**

**// Sphere**

Class Sphere:

Attributes:

Material: ambient, diffuse, specular, shininess

Geometry: radius, sectorCount, stackCount

Buffers: sphereVAO, vertices, normals, indices

Constructor(radius, sectorCount, stackCount, ambient, diffuse, specular, shininess):

Set material and geometry attributes

Call buildCoordinatesAndIndices()

Call buildVertices()

Initialize VAO, VBO, EBO for rendering

Method buildCoordinatesAndIndices():

For each stack (latitude):

For each sector (longitude):

Compute vertex positions and normals

Generate indices for triangles

Method drawSphere(shader, model):

Set material and model in shader

Bind VAO and render sphere

End Class

Sphere sphere = Sphere();

translateMatrix = glm::translate(identityMatrix, glm::vec3(3.0f, 0.3f, 0.0f));

scaleMatrix = glm::scale(translateMatrix, glm::vec3(0.3f, 0.3f, 0.3f));

model = globalTranslationMatrix \* scaleMatrix;

sphere.drawSphere(lightingShader, model);

**// Cone**

float cylinder\_vertices[] = {

0.505450490764242,0.0f,0.494505494505494f,0.0f,0.0f,0.0f,

0.512152956078189,0.0f,0.797488226059654f,0.0f,0.0f,0.0f,

… }

unsigned int cone\_indices[] = {

0, 1, 2,

58, 59, 60,

58, 60, 61,

…}

unsigned int lightCubeVAO2;

glGenVertexArrays(1, &lightCubeVAO2);

glBindVertexArray(lightCubeVAO2);

glBindBuffer(GL\_ARRAY\_BUFFER, VBO\_cone);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, EBO\_cone);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 6 \* sizeof(float), (void\*)0);

glEnableVertexAttribArray(0);

glBindVertexArray(lightCubeVAO2);

translateMatrix = glm::translate(identityMatrix, glm::vec3(1.3f, 2.95f, 1.36f));

scaleMatrix = glm::scale(translateMatrix, glm::vec3(-0.6f, -0.26f, -0.6f));

model = globalTranslationMatrix \* scaleMatrix;

ourShader.setMat4("model", model);

ourShader.setVec3("color", glm::vec3(0.8f, 0.8f, 0.8f));

glDrawElements(GL\_TRIANGLES, 5000, GL\_UNSIGNED\_INT, 0);

**// Torus**

Set outerRadius, innerRadius, numOuterSegments, numInnerSegments

For i in 0 to numOuterSegments:

Compute outerAngle, centerX, centerZ

For j in 0 to numInnerSegments:

Compute innerAngle

Calculate x, y, z based on center and radius

Apply translation and scaling

Set model matrix in shader

Draw object

End Inner Loop

End Outer Loop

**// Point Light**

Class PointLight:

Properties:

position, ambient, diffuse, specular, k\_c, k\_l, k\_q, lightNumber

Methods:

setUpPointLight(lightingShader): Set light attributes based on lightNumber

turnOff(): Set ambientOn, diffuseOn, specularOn to 0.0

turnOn(): Set ambientOn, diffuseOn, specularOn to 1.0

turnAmbientOn(): Set ambientOn to 1.0

turnAmbientOff(): Set ambientOn to 0.0

turnDiffuseOn(): Set diffuseOn to 1.0

turnDiffuseOff(): Set diffuseOn to 0.0

turnSpecularOn(): Set specularOn to 1.0

turnSpecularOff(): Set specularOn to 0.0

**// Directional Light**

lightingShader.setVec3("diectionalLight.directiaon", 0.0f, 3.0f, 0.0f);

lightingShader.setVec3("diectionalLight.ambient", .2, .2, .2);

lightingShader.setVec3("diectionalLight.diffuse", .8f, .8f, .8f);

lightingShader.setVec3("diectionalLight.specular", 1.0f, 1.0f, 1.0f);

lightingShader.setBool("dlighton", directionalLightOn);

**// Spot Light**

lightingShader.setVec3("spotlight.position", 0.0f, 0.0f, 0.0f);

lightingShader.setVec3("spotlight.direction", 0, -1, 0);

lightingShader.setVec3("spotlight.ambient", .2, .2, .2);

lightingShader.setVec3("spotlight.diffuse", .8f, .8f, .8f);

lightingShader.setVec3("spotlight.specular", 1.0f, 1.0f, 1.0f);

lightingShader.setFloat("spotlight.k\_c", 1.0f);

lightingShader.setFloat("spotlight.k\_l", 0.09);

lightingShader.setFloat("spotlight.k\_q", 0.032);

lightingShader.setFloat("cos\_theta", glm::cos(glm::radians(5.5f)));

lightingShader.setBool("spotlighton", spotLightOn);

**// Light On Off**

if (glfwGetKey(window, GLFW\_KEY\_1) == GLFW\_PRESS) {

// All Point Light, Directional Light, Spot Light Off }

if (glfwGetKey(window, GLFW\_KEY\_2) == GLFW\_PRESS) {

// All Point Light, Directional Light, Spot Light On }

if (glfwGetKey(window, GLFW\_KEY\_3) == GLFW\_PRESS) { // All Point Light Off }

if (glfwGetKey(window, GLFW\_KEY\_4) == GLFW\_PRESS) { // All Point Light On }

if (glfwGetKey(window, GLFW\_KEY\_5) == GLFW\_PRESS) { // Directional Light Off }

if (glfwGetKey(window, GLFW\_KEY\_6) == GLFW\_PRESS){ // Directional Light On }

if (glfwGetKey(window, GLFW\_KEY\_7) == GLFW\_PRESS) { // Spot Light Off }

if (glfwGetKey(window, GLFW\_KEY\_8) == GLFW\_PRESS) { // Spot Light On }

if (glfwGetKey(window, GLFW\_KEY\_0) == GLFW\_PRESS) { // Point Light1 On, 2, 3, 4 Off }

if (glfwGetKey(window, GLFW\_KEY\_KP\_1) == GLFW\_PRESS) { // All Ambient Light Off }

if (glfwGetKey(window, GLFW\_KEY\_KP\_2) == GLFW\_PRESS) { // All Ambient Light On }

if (glfwGetKey(window, GLFW\_KEY\_KP\_3) == GLFW\_PRESS) { // All Diffuse Light Off }

if (glfwGetKey(window, GLFW\_KEY\_KP\_4) == GLFW\_PRESS) { // All Diffuse Light On }

if (glfwGetKey(window, GLFW\_KEY\_KP\_5) == GLFW\_PRESS) { // All Specular Light Off }

if (glfwGetKey(window, GLFW\_KEY\_KP\_6) == GLFW\_PRESS) { // All Specular Light On }

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Figure 4.1: Keyboard Controls for Light and Lighting Property Adjustments According to Light On-Off Algorithm*