**Name: Israt Tasnim Esha, Roll: 1907090**

**Description:**

This assignment features various furniture elements, including two types of chairs (one made of cones, the other using a cylinder and cube), tables (made of cubes), sofa (made of cubes), mirrors (using cylinders), a sink (using cubes), three walls (made of cubes), a floor (made of cubes), interior design elements (using cylinders and hexagons), and lighting (using cylinders). This assignment utilizes cubes, hexagons, cylinders, and cones with textures and lighting.

**Program and Results:**

**// cube.h**

Class Cube:

Attributes:

texture (diffuseMap, specularMap) // Texture maps for the cube

TXmin, TXmax, TYmin, TYmax // Texture coordinates for mapping

Methods:

setTextureCoordinates():

- Define UV coordinates for cube faces

- UV coordinates range from (TXmin, TYmin) to (TXmax, TYmax)

setTextureWrapping():

- Set texture wrapping mode:

- Repeat: Texture tiles if UV > [0,1]

- Clamp: Texture clamps to edge if UV > [0,1]

- Mirrored Repeat: Texture mirrors when UV > [0,1]

applyTexture():

- Bind diffuseMap and specularMap to texture units

- Set texture wrapping mode in OpenGL

drawCubeWithTexture(shader, modelMatrix):

- Use shader program

- Bind textures with applyTexture

- Draw cube with texture mapping based on UV coordinates

**// Cone2.h**

Class Cone2:

Attributes:

radius // Radius of the cone base

height // Height of the cone

sectorCount // Number of sectors for the cone base

ambient, diffuse, specular // Material properties

TXmin, TXmax, TYmin, TYmax // Texture coordinates for mapping

diffuseMap, specularMap // Diffuse and specular texture maps

shininess // Shininess factor for material

Methods:

set(radius, height, sectorCount, amb, diff, spec, shiny, dMap, sMap, textureXmin, textureYmin, textureXmax, textureYmax):

- Set the attributes: radius, height, sectorCount, ambient, diffuse, specular, shininess

- Set texture maps: diffuseMap, specularMap

- Set texture coordinates: TXmin, TXmax, TYmin, TYmax

- Call buildCoordinatesAndIndices() to prepare vertex and index data

buildCoordinatesAndIndices():

- Calculate sectorStep based on sectorCount

- Define vertices for bottom center, bottom circle, and apex

- Define indices for bottom circle and side triangles

drawConeWithTexture(shader, model):

- Use shader program

- Set material properties (ambient, diffuse, specular, shininess)

- Bind the diffuse and specular textures

- Set the model matrix for the shader

- Draw the cone using vertex and index data

setUpConeVertexDataAndConfigureVertexAttribute():

- Generate buffers and vertex array for cone

- Bind buffers and configure vertex attribute pointers for position, normal, and texture coordinates

- Enable the vertex attribute arrays for position, normal, and texture

**// cylinder.h**

Class CylinderWithTexture:

Attributes:

baseRadius // Radius of the base of the cylinder

topRadius // Radius of the top of the cylinder

height // Height of the cylinder

sectorCount // Number of sectors (around the cylinder)

stackCount // Number of stacks (along the height of the cylinder)

ambient, diffuse, specular // Material properties

shininess // Shininess factor for material

diffuseMap, specularMap // Diffuse and specular texture maps

Methods:

set(baseRadius, topRadius, height, sectorCount, stackCount, amb, diff, spec, shiny):

- Set the attributes: baseRadius, topRadius, height, sectorCount, stackCount

- Set the material properties: ambient, diffuse, specular, shininess

buildCoordinatesAndIndices():

- Calculate sectorStep and stackStep for positioning

- Generate coordinates for the cylinder surface and caps

- Generate normals (for lighting) and texture coordinates

- Generate indices for the side surface and caps

buildCap(radius, y, isTop):

- Generate vertices for the top or bottom cap

- Add center point and calculate the sector angles for each segment of the cap

- Set normals and texture coordinates

- Create indices for the cap triangles

buildVertices():

- Prepare final vertex data combining position, normal, and texture coordinates

- Push data into the vertices array

setupVAO():

- Generate and bind a Vertex Array Object (VAO)

- Create and bind Vertex Buffer Object (VBO) and Element Buffer Object (EBO)

- Set up vertex attribute pointers for position, normal, and texture coordinates

- Enable vertex attribute arrays for rendering

drawCylinder(lightingShader, model, viewPos):

- Use the lighting shader

- Set material properties in the shader (ambient, diffuse, specular, shininess)

- Bind the diffuse and specular texture maps to texture units

- Pass the transformation matrices (model, view)

- Draw the cylinder using the configured VAO and indices

**// hexagon.h**

Class Hexagon:

Attributes:

ambient, diffuse, specular // Material properties

TXmin, TXmax, TYmin, TYmax // Texture coordinates

diffuseMap, specularMap // Texture maps

shininess // Shininess factor

hexagonVAO, lightHexagonVAO, lightTexHexagonVAO // Vertex Array Objects

hexagonVBO, hexagonEBO // Vertex and Element Buffer Objects

Methods:

constructor():

- Initialize material properties and texture properties

- Call setUpHexagonVertexDataAndConfigureVertexAttribute()

constructor(ambient, diffuse, specular, shininess):

- Set material properties

- Call setUpHexagonVertexDataAndConfigureVertexAttribute()

constructor(diffuseMap, specularMap, shininess, TXmin, TYmin, TXmax, TYmax):

- Set texture properties

- Call setUpHexagonVertexDataAndConfigureVertexAttribute()

destructor():

- Delete VAOs and VBOs for hexagon and light objects

drawHexagonWithTexture(lightingShaderWithTexture, model):

- Use lighting shader with texture

- Set shader material properties (diffuse, specular, shininess)

- Bind diffuse and specular textures

- Set model matrix

- Draw hexagon with texture

drawHexagonWithMaterialisticProperty(lightingShader, model):

- Use lighting shader

- Set material properties (ambient, diffuse, specular, shininess)

- Set model matrix

- Draw hexagon with material properties

drawHexagon(shader, model, r, g, b):

- Use shader

- Set color for hexagon

- Set model matrix

- Draw hexagon with specified color

setMaterialisticProperty(ambient, diffuse, specular, shininess):

- Set the material properties

setTextureProperty(diffuseMap, specularMap, shininess):

- Set texture properties

Private Methods:

setUpHexagonVertexDataAndConfigureVertexAttribute():

- Define hexagon vertices and indices based on texture coordinates and geometry

- Create and bind VAOs, VBOs, and EBOs for hexagon

- Set vertex attribute pointers for position, normal, and texture coordinates

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