Shivcharan T
205001100
CSE - B

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Lab Exercise 10: : Creating a 3D Scene in C++ using OpenGL

Write a C++ program using Opengl to draw atleast four 3D objects. Apply lighting and texture and render the scene. Apply transformations to create a simple 3D animation. [Use built-in transformation functions.

Aim:

To create a 3D Scene in C++ using OpenGL

Algorithm:

1. Initialize OpenGL, set window dimensions, and create a window using GLUT.
2. Set clear color and enable depth testing for accurate rendering.
3. Load a texture and set its parameters for later use in the scene using SOIL.
4. Enable lighting and set light parameters (position, ambient, diffuse, specular).
5. Enable texture mapping and set the shading model to GL_FLAT.
6. Set up the perspective projection using gluPerspective.
7. Define the display function to clear buffers, set the projection and modelview matrices, and draw 3D objects.
8. Inside the display function, draw a teapot, a scaled sphere, a scaled and translated cone, and a torus using built-in functions.
9. Apply transformations to each object (translation, rotation, scaling) to create the desired arrangement using built-in functions..
10. Implement an update function to control the rotation angle for animation.
11. Set up the main function, specify the display and update functions, and initialize the scene.
12. Enter the GLUT main loop to handle events and continuously render the scene.

10.cpp:

```
#include <GL/glut.h>
#include <SOIL/SOIL.h>

const int windowWidth = 800;
const int windowHeight = 600;

GLfloat angle = 0.0f; // Initial rotation angle

// Texture variables
GLuint textureID;

// Rotation angles
float angleX = 0.0f;
float angleY = 0.0f;
```

```
// Texture coordinates
float texCoordX = 0.0f;
float texCoordY = 0.0f;

void init()
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glEnable(GL_DEPTH_TEST);

    // Load texture
    glGenTextures(1, &textureID);
    glBindTexture(GL_TEXTURE_2D, textureID);
    int width, height;
    unsigned char *image = SOIL_load_image("texture1.jpg", &width, &height, 0,
SOIL_LOAD_RGB);
    glTexImage2D(GL_TEXTURE_2D,   0,   GL_RGB,   width,   height,   0,   GL_RGB,
GL_UNSIGNED_BYTE, image);
    SOIL_free_image_data(image);

    // Set texture parameters
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

    // Enable lighting and set light parameters
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
    GLfloat ambient[] = {0.2, 0.2, 0.2, 1.0};
    GLfloat diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);

    // Enable texture and set shading model
    glEnable(GL_TEXTURE_2D);
    glShadeModel(GL_FLAT);

    // Set up perspective projection
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0f, 1.0f, 1.0f, 100.0f);
    glMatrixMode(GL_MODELVIEW);
}

void drawTeapot()
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    glutSolidTeapot(1.0); // Draw a teapot
```

```
    glDisable(GL_TEXTURE_2D);
}

// glusolid

void drawSphere()
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    GLUquadricObj *sphere = gluNewQuadric();
    gluQuadricTexture(sphere, GL_TRUE);

    gluSphere(sphere, 0.5, 20, 20);

    gluDeleteQuadric(sphere);
    glDisable(GL_TEXTURE_2D);
}

void drawConeInit()
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    GLUquadricObj *cone = gluNewQuadric();
    gluQuadricTexture(cone, GL_TRUE);

    gluCylinder(cone, 0.0, 0.5, 1.0, 20, 20); // Draw a cone

    gluDeleteQuadric(cone);
    glDisable(GL_TEXTURE_2D);
}

void drawCone()
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    GLUquadricObj *cone = gluNewQuadric();
    gluQuadricTexture(cone, GL_TRUE);

    gluCylinder(cone, 0.0, 0.5, 1.0, 20, 20); // Draw the cone

    glPushMatrix();
    glTranslatef(0.0, 0.0, 1.0);     // Move to the base of the cone
    gluDisk(cone, 0.0, 0.5, 20, 20); // Draw the base circle
    glPopMatrix();

    gluDeleteQuadric(cone);
    glDisable(GL_TEXTURE_2D);
}

void drawCylinder()
{
    glEnable(GL_TEXTURE_2D);
```

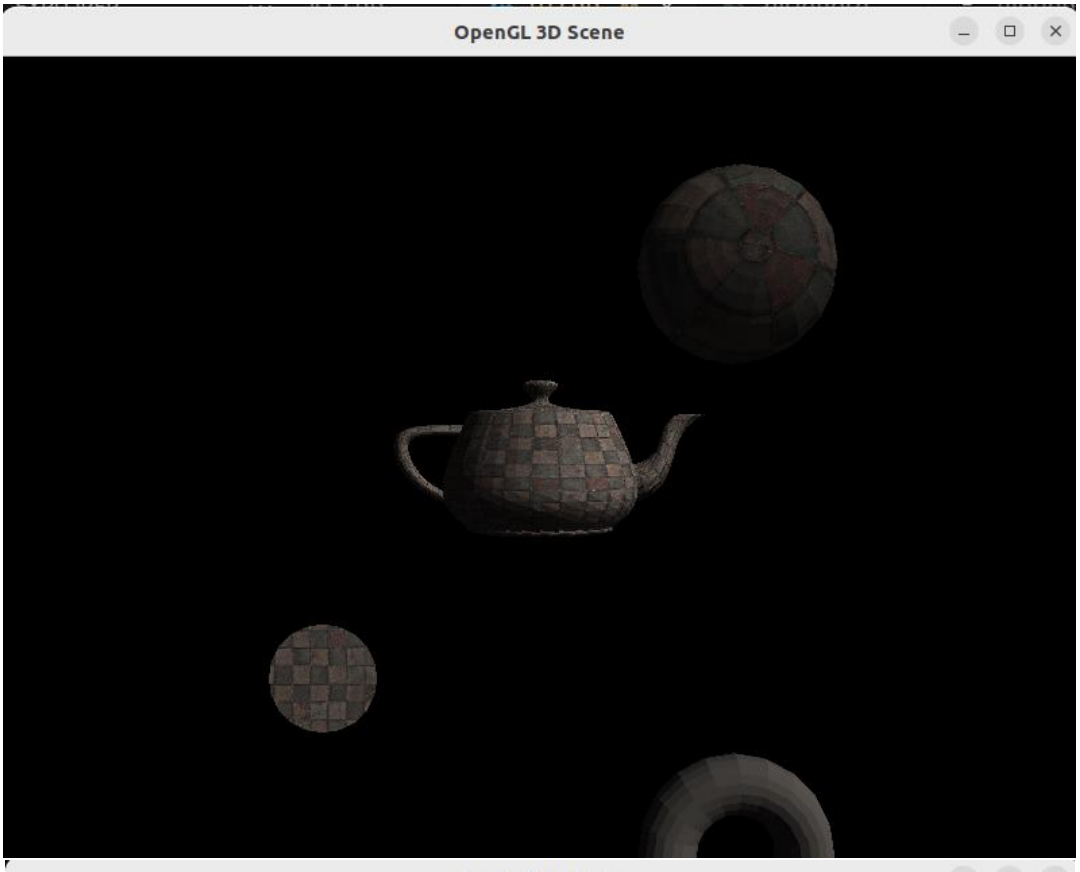
```
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    GLUquadricObj *cylinder = gluNewQuadric();
    gluQuadricTexture(cylinder, GL_TRUE);

    gluCylinder(cylinder, 0.5, 0.5, 1.0, 20, 20);

    gluDeleteQuadric(cylinder);
    glDisable(GL_TEXTURE_2D);
}

void drawTorus()
{
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureID); // Bind the texture

    glutSolidTorus(0.3, 0.7, 20, 20);

    glDisable(GL_TEXTURE_2D);
}

void display()
{
    glViewport(0, 0, windowWidth, windowHeight); // Set the viewport size

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0,  static_cast<double>(windowWidth)  /  windowHeight,  0.1,
100.0); // Adjusted near and far planes

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // Adjusted camera
position

    glTranslatef(0.0f, 0.0f, -5.0f);    // Move the scene back along the z-axis
    glRotatef(angle, 1.0f, 1.0f, 0.0f); // Rotate around the x and y-axis

    // Apply material properties (color, etc.)
    GLfloat material_diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f};
    glMaterialfv(GL_FRONT, GL_DIFFUSE, material_diffuse);

    // Draw 3D objects
    drawTeapot(); // Draw a teapot
    glTranslatef(2.0f, 2.0f, 0.0f);
    glScalef(2.0f, 2.0f, 2.0f);  // Scale
    drawSphere(); // Draw a sphere
    glScalef(0.5f, 0.5f, 0.5f);  // Scale
    glTranslatef(-4.0f, -4.0f, 0.0f);
    drawCone(); // Draw a cylinder
    glTranslatef(4.0f, -2.0f, 0.0f);
    drawTorus(); // Draw a torus
```

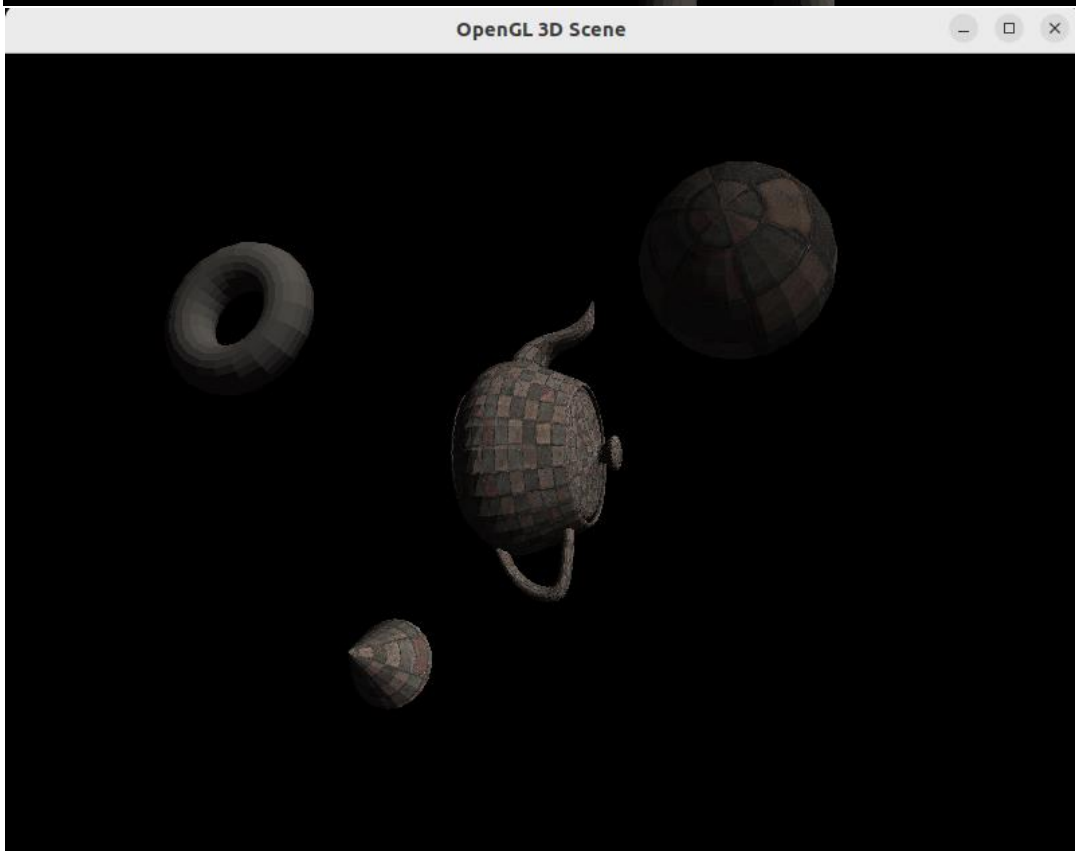
```
    glTranslatef(0.0f, 0.0f, -5.0f);
    glutSwapBuffers();
}

void update(int value)
{
    angle += 2.0f; // Update rotation angle
    if (angle > 360)
    {
        angle -= 360; // Keep the angle within 0 to 360 degrees
    }

    glutPostRedisplay();          // Trigger a redraw
    glutTimerFunc(16, update, 0); // Call update function every 16 milliseconds
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(windowWidth, windowHeight); // Set window size
    glutCreateWindow("OpenGL 3D Scene");

    // Add these lines for proper initialization
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutTimerFunc(25, update, 0);

    init();

    glutMainLoop();
    return 0;
}
```

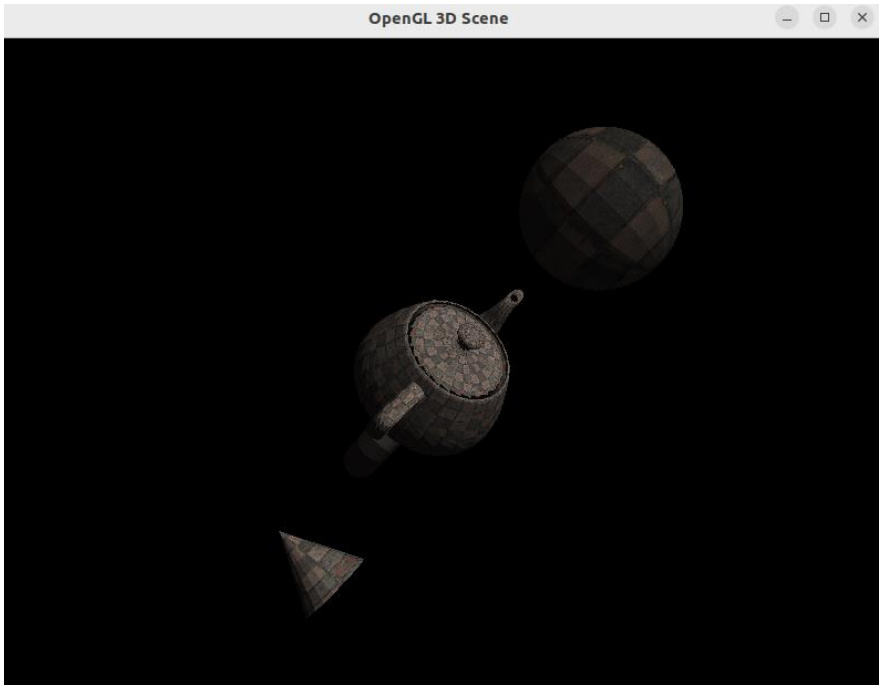
run.sh:

```
g++ 10.cpp -lGL -lglut -lGLU -lSOIL
./a.out
```

Sample I/O:


OpenGL 3D Scene


OpenGL 3D Scene



Learning Outcomes:

Thus, 3D objects were drawn and lighting and textures were applied and the scene was rendered in C++ using OpenGL.