Shivcharan T
205001100
CSE - B

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

Lab Exercise 9: : 3-Dimensional Projections in C++ using OpenGL

Write a menu driven program to perform Orthographic parallel projection and Perspective projection on any 3D object. Set the camera to any position on the 3D space. Have (0,0,0) at the center of the screen. Draw X, Y and Z axis. You can use gluPerspective() to perform perspective projection. Use keyboard functions to rotate and show different views of the object. [Can use built-in functions for 3D transformations].

Aim:
To implement 3D projections on objects using C++ using OpenGL

Algorithm:

1. Include necessary OpenGL and GLUT libraries.

2. Declare global variables for rotation angles and camera position.

3. Specify the vertices of the 3D object.

4. Implement a function to draw X, Y, and Z axes.

5. Create a function to draw the 3D object using glBegin(GL_QUADS) and glVertex3fv.

6. Set up the display function. Use gluLookAt for camera positioning, incorporate rotation transformations using glRotatef, call drawAxes and drawObject within the display function, and use glutSwapBuffers to swap the front and back buffers.

7. Implement a keyboard function to handle user input. Update rotation angles based on key presses (e.g., 'x', 'y', 'z') and allow the user to exit the program (e.g., 'q').

8. Initialize OpenGL and GLUT, set up the window and callback functions for display and keyboard input, configure perspective or orthographic projection, and enter the main event loop.

Code:

9.cpp:

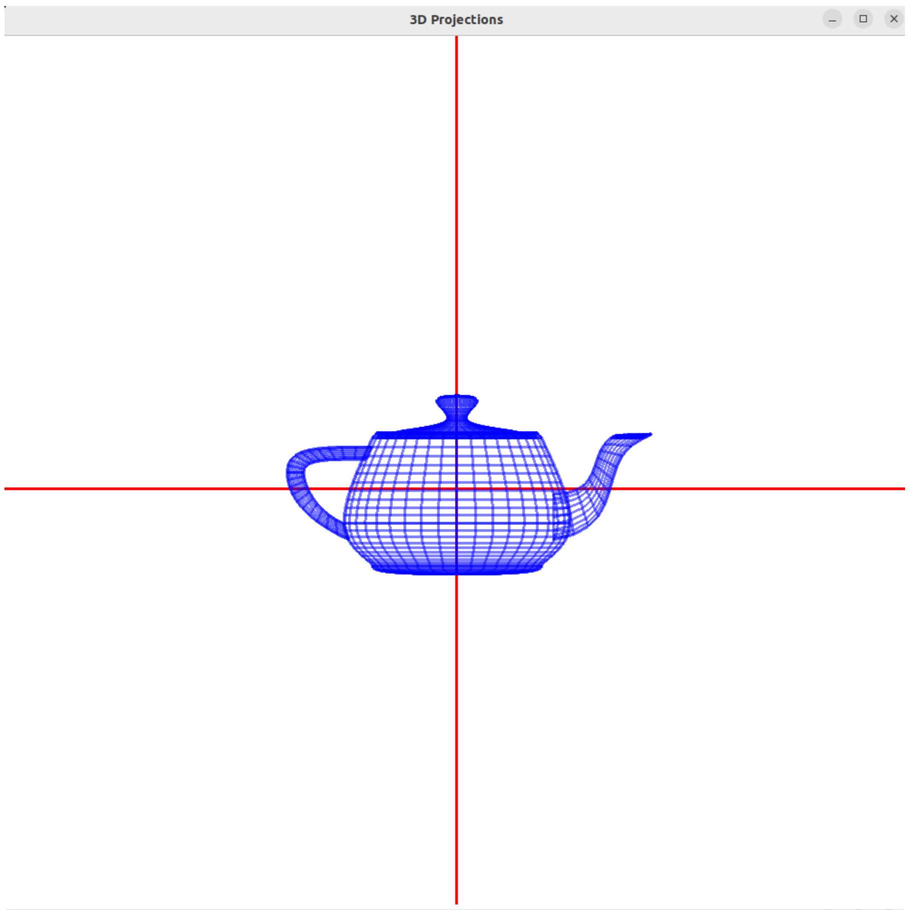
```
#include <iostream>
#include <stdio.h>
#include <cmath>
#include <cstring>
#include <GL/glut.h>
```

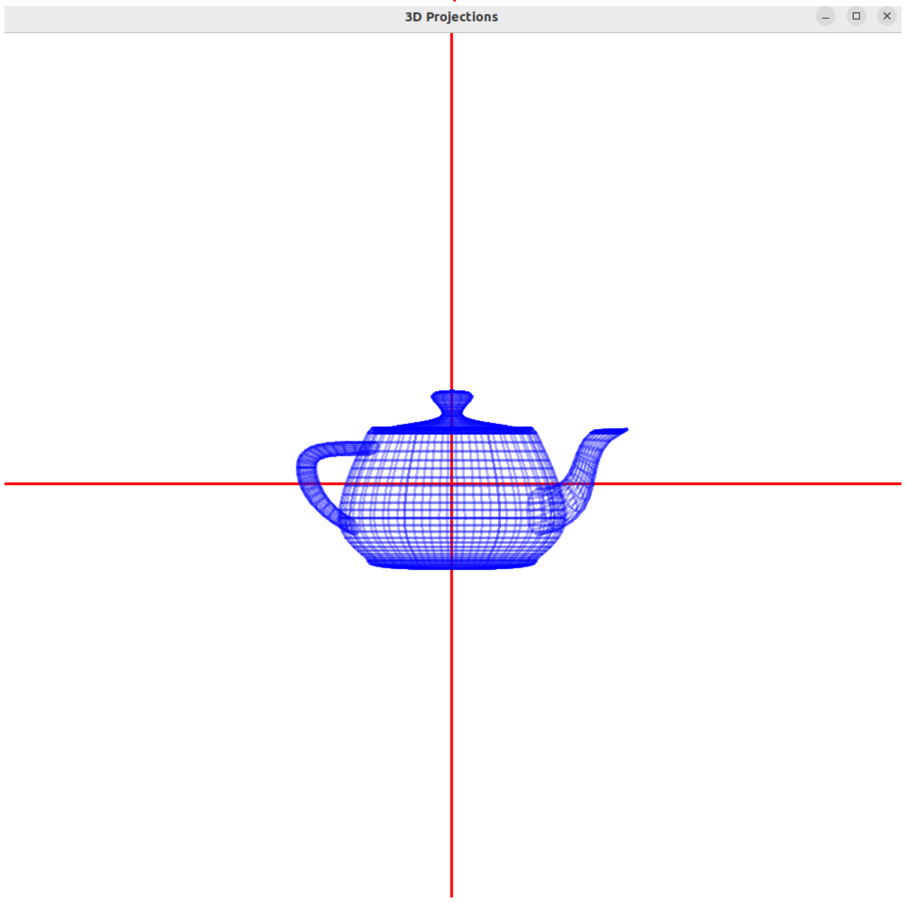
```
using namespace std;

// Global constants
const float windowHeight = 1000;
const float windowWidth = 1000;

const float X_MIN = -500;
const float X_MAX = 500;
const float Y_MIN = -500;
const float Y_MAX = 500;
const int FPS = 60;

// Global variables to handle rotation
GLfloat x_rotate = 0;
GLfloat y_rotate = 0;

// Global variable for projection
bool isOrthoProjection = true;
void initializeDisplay();
void keyboardKeys(unsigned char key, int x, int y);
void drawAxes();

void myInit(void)
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(0.0, 0.0, 1.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-windowWidth / 2, windowWidth / 2, -windowHeight / 2, windowHeight
/ 2);
}

void drawAxes()
{
    // To draw X and Y axis
    glColor3d(1, 0, 0);
    glBegin(GL_LINES);
    glVertex2f(-2, 0);
    glVertex2f(2, 0);
    glVertex2f(0, -2);
    glVertex2f(0, 2);
    glEnd();
    glFlush();
}

void keyboardKeys(unsigned char key, int x, int y)
{
    // Callback function for keyboard interactivity
    key = tolower(key);
    switch (key)
    {
    case 'w':
    {
        // glLoadIdentity(); // Reset transformations
        x_rotate += 5;
        break;
    }
    case 's':
    {
        x_rotate -= 5;
        break;
    }
    case 'd':
    {
        y_rotate += 5;
        break;
    }
    case 'a':
    {
        y_rotate -= 5;
        break;
    }
    case 27:
        exit(0);
    case 32:
    {
        // Spacebar for changing projections
        isOrthoProjection = !isOrthoProjection;
        x_rotate = 0;
        y_rotate = 0;
        break;
    }
    }
    // Update the display
    glutPostRedisplay();
}

void display()
{
    // Initialize display parameters
    glClearColor(1, 1, 1, 1);
    glClear(GL_COLOR_BUFFER_BIT);
    // Translucency
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    // Line width
    glLineWidth(3);
    // Apply the transformations & drawing on the model view matrix
    glMatrixMode(GL_MODELVIEW);
    // Draw the X and Y axis
    drawAxes();
    // Transform only the drawn object, so use the matrix stack accordingly
    glPushMatrix();
    if (isOrthoProjection)
    {
        // Parallel Projection
        glOrtho(-2, 2, -2, 2, -2, 2);
    }
    else
    {
        // Perspective Projection
        gluPerspective(120, 1, 0.1, 50); // FoVy = 120, Aspect Ratio = 1
    }
    gluLookAt(0, 0, 1, 0, 0, 0, 0, 1, 0); // Camera, Center & Up Vector
    glPushMatrix(); // Create a separate transformation matrix
    glRotatef(x_rotate, 1, 0, 0);         // Keyboard based rotations
    glRotatef(y_rotate, 0, 1, 0);
    glColor4f(0, 0, 1, 0.3); // Draw the object
    glutWireTeapot(0.5);
    glPopMatrix(); // Pop the transformation matrix
    glPopMatrix(); // Pop the matrix back into the model view stack
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(windowWidth, windowHeight);
    glutCreateWindow("3D Projections");
    printf("Enter (1) for orthographic and (0) for perspective: ");
    int oop;
    scanf("%d", &oop);
    isOrthoProjection = oop;
    // Register the callback functions
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboardKeys);

    glutMainLoop();
    return 0;
}
```

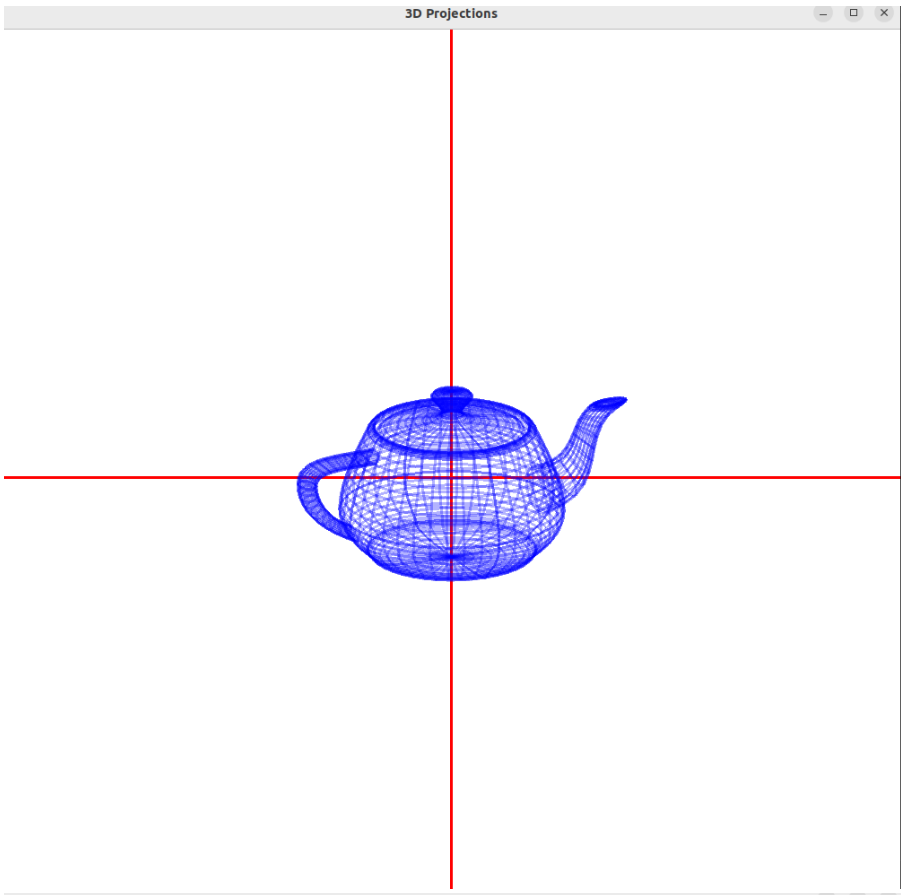
run.sh:

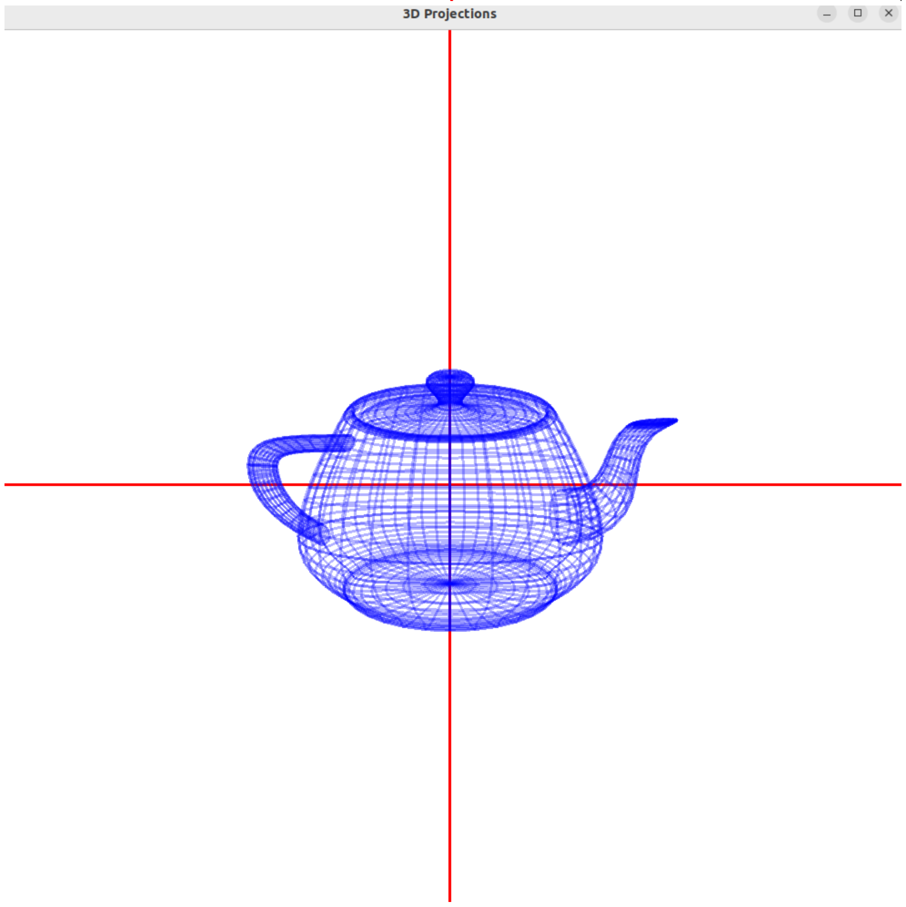
```
g++ 9.cpp -lGL -lglut -lGLU
./a.out
```

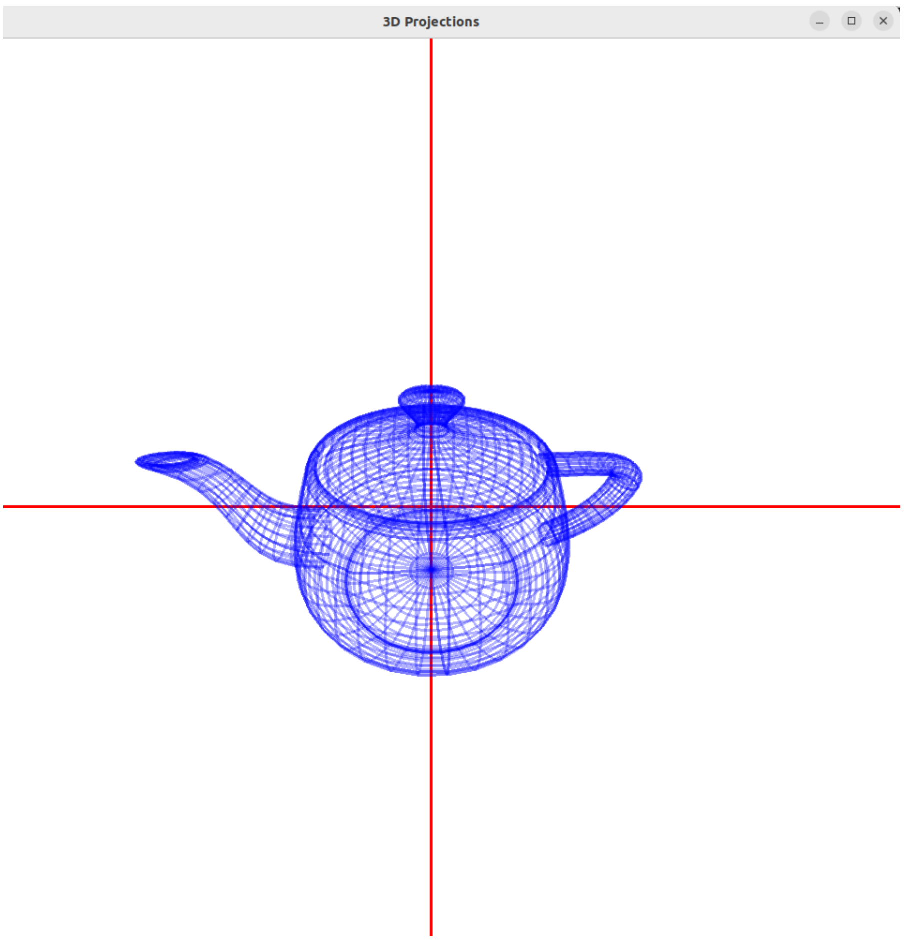
Sample I/O:


3D Projections


3D Projections


3D Projections


3D Projections



Learning Outcomes:

Thus, 3D projections has been implemented on objects using OpenGL.