SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1712-Graphics and Multimedia Lab

Programming Assignment 6

3-Dimensional Projections in C++ using OpenGL

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Write a menu driven program to perform Orthographic 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.

Use gluPerspective() to perform perspective projection. Also, can use inbuilt functions for 3D trasformations.

Source code:

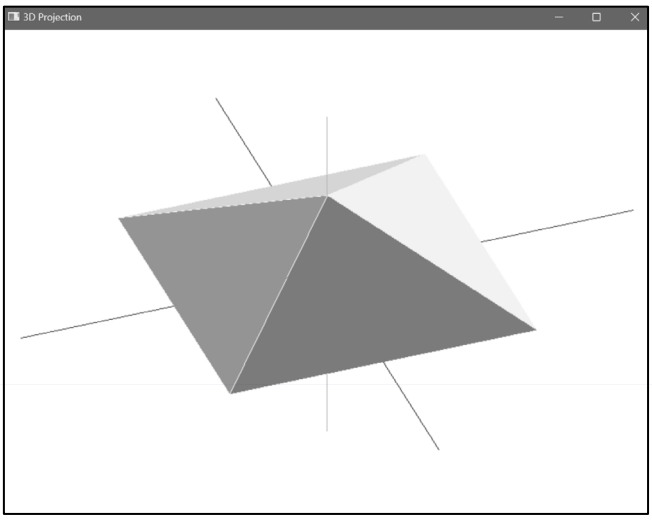
```
#include <GL/glut.h>
int projectionMode = 0;
float rotateX = 0.0f;
float rotateY = 0.0f;
float cameraX = 0.0f;
float cameraY = 0.0f;
float cameraZ = 5.0f;
void draw3DObject()
{
    glBegin(GL_TRIANGLES);
    glColor3f(0.682, 0.871, 0.988);
    glVertex3f(0, 1, 0);
    glVertex3f(-1, 0, -1);
    glVertex3f(1, 0, -1);
    glColor3f(1, 0.984, 0.451);
    glVertex3f(0, 1, 0);
    glVertex3f(1, 0, -1);
    glVertex3f(1, 0, 1);
    glColor3f(0.694, 0.369, 1);
    glVertex3f(0, 1, 0);
    glVertex3f(1, 0, 1);
    glVertex3f(-1, 0, 1);
    glColor3f(0.972, 0.459, 0.667);
    glVertex3f(0, 1, 0);
    glVertex3f(-1, 0, 1);
    glVertex3f(-1, 0, -1);
    glEnd();
```

```
    glColor3f(0.914, 0.722, 0.141);
    glBegin(GL_QUADS);
    glVertex3f(-1, 0, -1);
    glVertex3f(1, 0, -1);
    glVertex3f(-1, 0, 1);
    glVertex3f(-1, 0, -1);
    glEnd();
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINES);
    glVertex3f(0, 1, 0);
    glVertex3f(-1, 0, -1);
    glVertex3f(0, 1, 0);
    glVertex3f(1, 0, -1);
    glVertex3f(0, 1, 0);
    glVertex3f(1, 0, 1);
    glVertex3f(0, 1, 0);
    glVertex3f(-1, 0, 1);
    glVertex3f(-1, 0, -1);
    glVertex3f(1, 0, -1);
    glVertex3f(1, 0, -1);
    glVertex3f(1, 0, 1);
    glVertex3f(1, 0, 1);
    glVertex3f(-1, 0, 1);
    glVertex3f(-1, 0, -1);
    glVertex3f(-1, 0, 1);
    glEnd();
}
void setPerspectiveProjection()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 1.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}
void setOrthographicProjection()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}
void display()
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 0, 1, 0);
    glRotatef(rotateX, 1.0f, 0.0f, 0.0f);
    glRotatef(rotateY, 0.0f, 1.0f, 0.0f);
    glBegin(GL_LINES);
    glColor3f(1.0f, 0.0f, 0.0f);
    glVertex3f(-2, 0, 0);
    glVertex3f(2, 0, 0);
    glColor3f(0.0f, 1.0f, 0.0f);
```

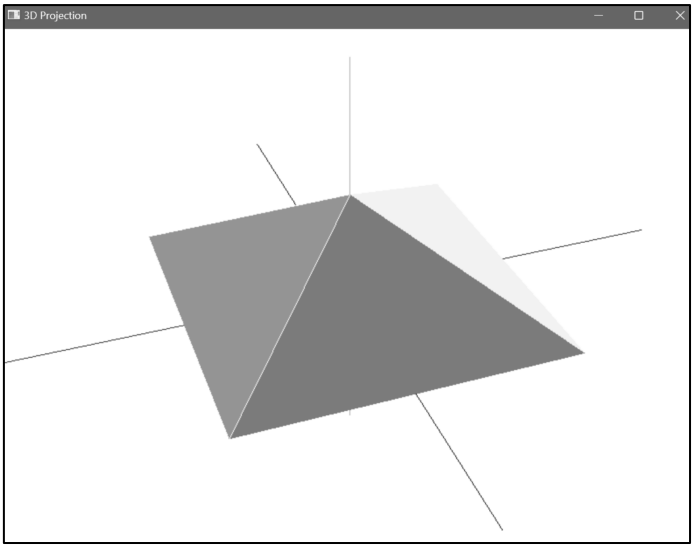
```
    glVertex3f(0, -2, 0);
    glVertex3f(0, 2, 0);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex3f(0, 0, -2);
    glVertex3f(0, 0, 2);
    glEnd();
    glColor3f(1.0f, 1.0f, 1.0f);
    draw3DObject();
    glutSwapBuffers();
}
void keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'o':
        projectionMode = 0;
        setOrthographicProjection();
        break;
    case 'p':
        projectionMode = 1;
        setPerspectiveProjection();
        break;
    case 'r':
        rotateX += 10.0f;
        break;
    case 'l':
        rotateY += 10.0f;
        break;
    case 'f':
        cameraZ -= 0.25f;
        break;
    case 'b':
        cameraZ += 0.25f;
        break;
    case 27:
        exit(0);
    }
    glutPostRedisplay();
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutCreateWindow("3D Projection");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    setOrthographicProjection();
    glutMainLoop();
    return 0;
}
```

Output

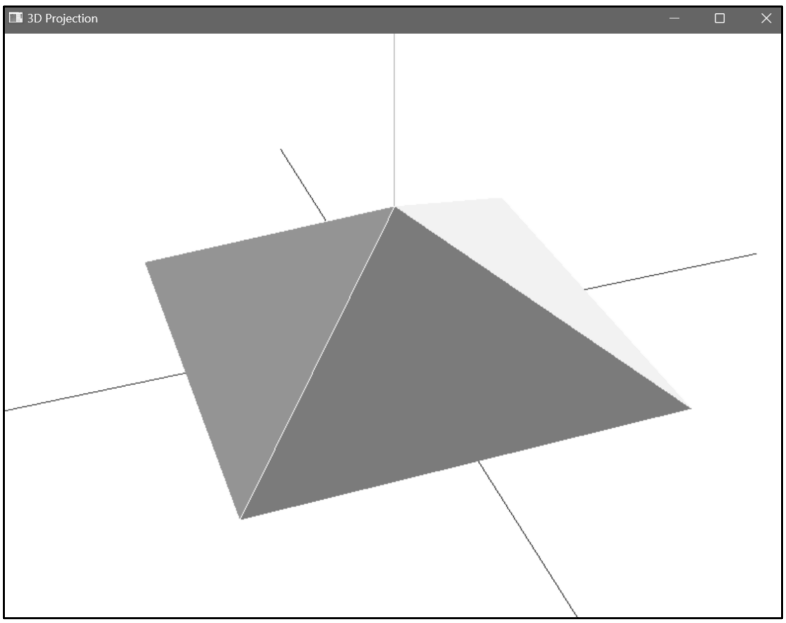
1 Orthographic Projection



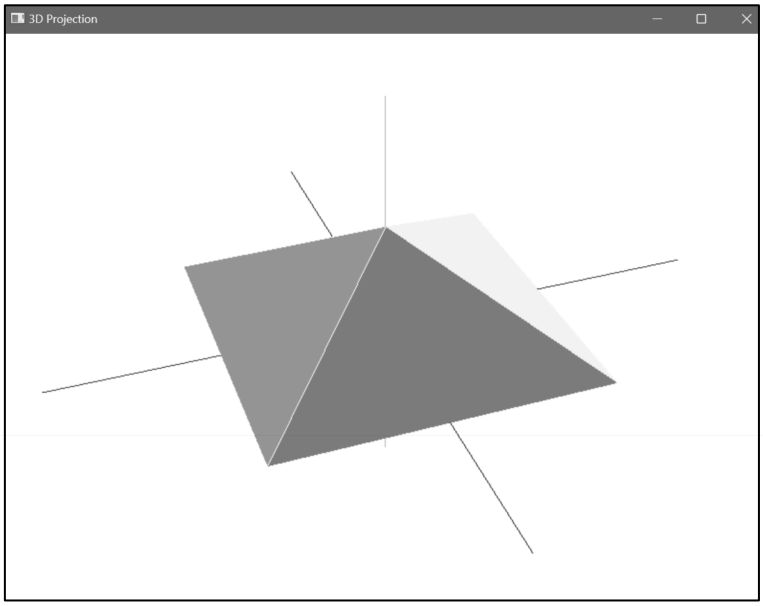
2 Perspective Projection



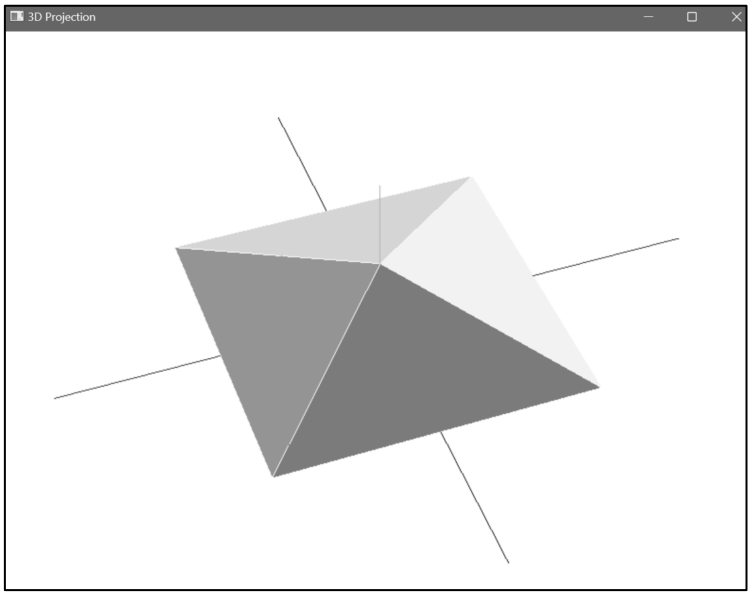
3 Moving Camera Forward



4 Moving Camera Backward



5 Rotate Right



6 Rotate Left