EXERCISE - 9: 3-DIMENSIONAL PROJECTIONS IN C++ USING OPENGL

AIM:

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].

ALGORITHM:

```
1.   Initialise frame buffer using glutInit()
2.   Set display mode as single buffer for 2D graphics with RGB colour using glutInitDisplayMode()
3.   Set output window size as 640, 640 pixels using glutWindowSize()
4.   Create the output window using glutCreateWindow()
5.   Call function to draw using glutDisplayFunc()
6.   Set visibility of faces using depth test parameter using glEnable()
7.   In the display function
    7.1.   Set background colour (RGB, opacity) as white using glClear()
    7.2.   Clear frame buffer using  glClear()
    7.3.   Set matrix mode to manipulate matrix values using glMatrixMode()
    7.4.   Load identity matrix using glLoadIdentity()
    7.5.   Set camera position
    7.6.   Plot coordinate axis
    7.7.   Draw the object
    7.8.   Get the key input and adjust the rotation and translation values.
    7.9.   Use gluOrtho() and gluPerspective() for orthographic and perspective projections respectively.
   7.10.   Display the object
   7.11.   End using glEnd()
   7.12.   Flush frame buffer using glFlush()
8.   Refresh the screen repeatedly while calling the function to draw using glutMainLoop()
```

CODE:

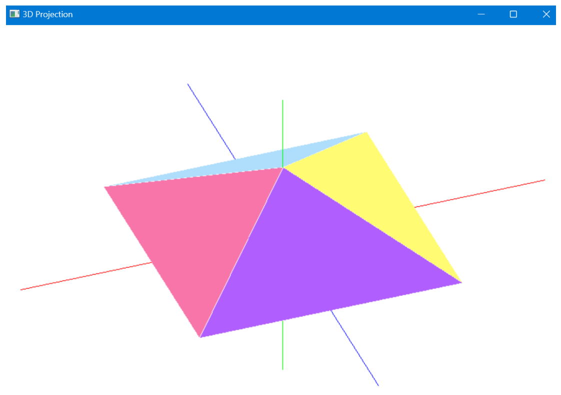
```
#include <GL/glut.h>

// Projection mode (0 for Orthographic, 1 for Perspective)
int projectionMode = 0;

// Rotation angles
float rotateX = 0.0f;
float rotateY = 0.0f;

// Camera position
float cameraX = 0.0f;
float cameraY = 0.0f;
float cameraZ = 5.0f;

// Function to draw the 3D object
void draw3DObject() {
    // Replace this with your 3D object drawing code
    //glutSolidCube(1.0);

    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();

    //Vertices
    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();

}
```

```
// Function to set up the perspective projection
void setPerspectiveProjection() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 1.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}

// Function to set up the orthographic projection
void setOrthographicProjection() {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}

// Display function
void display() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    // Set camera position and look at (0, 0, 0)
    gluLookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 0, 1, 0);

    // Apply rotation
    glRotatef(rotateX, 1.0f, 0.0f, 0.0f);
    glRotatef(rotateY, 0.0f, 1.0f, 0.0f);

    // Draw X, Y, and Z axes

    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();

    // Draw the 3D object
    glColor3f(1.0f, 1.0f, 1.0f);  // White object
    draw3DObject();

    glutSwapBuffers();
}

// Keyboard function to handle menu-driven options
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 'o':
        projectionMode = 0;  // Switch to Orthographic projection
        setOrthographicProjection();
        break;
    case 'p':
        projectionMode = 1;  // Switch to Perspective projection
        setPerspectiveProjection();
        break;
    case 'r':
        rotateX += 10.0f;  // Rotate object
        break;
    case 'l':
        rotateY += 10.0f;  // Rotate object
        break;
    case 'f':
        cameraZ -= 0.25f;  // Move camera forward
        break;
    case 'b':
        cameraZ += 0.25f;  // Move camera backward
        break;
    case 27:  // ESC key to exit
        exit(0);
    }

    glutPostRedisplay();
```

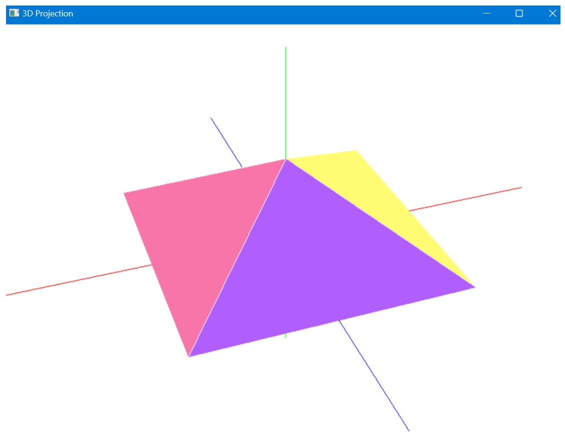
```
}

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();  // Initial projection mode

    glutMainLoop();

    return 0;
}
```

OUTPUT:

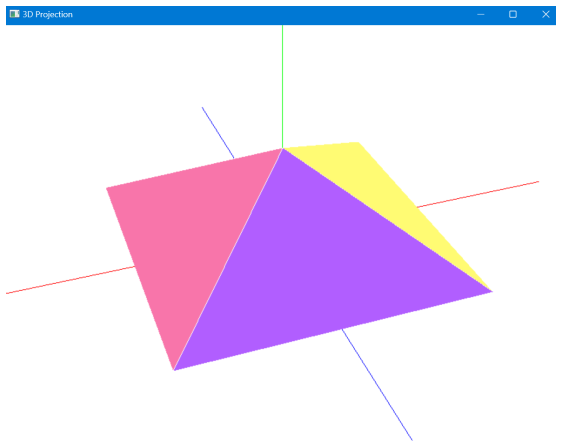
Orthographic Projection



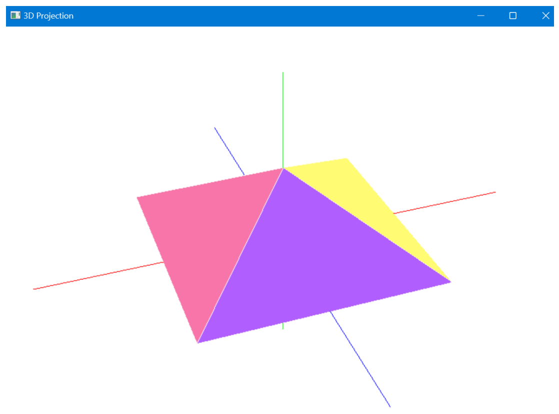
Perspective Projection



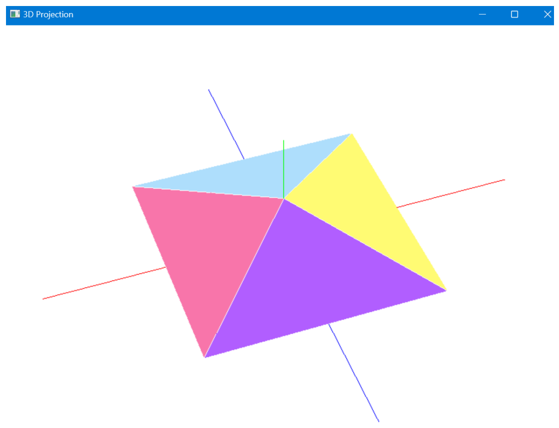
Moving Camera Forward



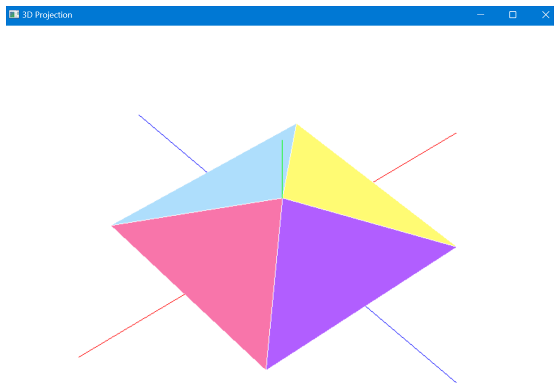
Moving Camera Backward



Rotate Right



Rotate Left



LEARNING OUTCOMES:

1. Learnt to adjust the parameters of the frames
2. Learnt to plot points and mark coordinates
3. Learnt to plot points in 3-Dimensions
4. Learnt about parallel and perspective projections