SSN College of Engineering
Department of Computer Science and Engineering
UCS1712 – GRAPHICS AND MULTIMEDIA LAB
Assignment V B – 2D Transformation

Name: Prasannakumaran D
Registration Number: 185001110
Semester: VII

September 8, 2021

1 AIM

1) To write a C++ menu-driven program using OPENGL to perform 2D transformations – reflection and shearing for polygons

2 ALGORITHM

- Read the no. of edges of the polygon from the user.
- Read the vertices of the polygon.
- Plot the original polygon.
- Read the transformation from the user given the menu
- If option is Reflection along X axis, multiply -1 to the Y coordinates of the original polygon and plot
- If option is Reflection along Y axis, multiply -1 to the X coordinates of the original polygon and plot
- If option is Reflection about the origin, multiply -1 to both the X and Y coordinates of the original polygon and plot
- If option is Reflection along Y = X line, swap the X and Y coordinates of the original polygon and plot
- If the option is shearing along the X axis then, read the shearing factor and add the shearing factor to the topmost vertices of the polygon and plot. You may also translate the sheared polygon so that it doesn't overlap on the original polygon
- If the option is shearing along the Y axis then, read the shearing factor and add the shearing factor to the rightmost vertices of the polygon and plot.You may also translate the sheared polygon so that it doesn't overlap on the original polygon

3 Code : 2D Transformation

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <vector>
```

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

using namespace std;
int pntX1, pntY1, op = 0, edges;
vector<int> pntX;
vector<int> pntY;
int shearingX, shearingY;
double round(double d) {
    return floor(d + 0.5);
}

void drawPolygon() {
    glBegin(GL_POLYGON);
    glColor3f(1, 0.01, 0.18);
    for (int i = 0; i < edges; i++) {
        glVertex2i(pntX[i], pntY[i]);
    }
    glEnd();
}

void reflection(int option) {
    if (option == 4) {
    glBegin(GL_LINES);
    glVertex2i(-640, -640);
    glVertex2i(640, 640);
    glEnd();
    }
    glBegin(GL_POLYGON);
    glColor3f(0.02, 0.72, 0.09);
    //Reflection about X axis
    if (option == 1) {
        for (int i = 0; i < edges; i++) {
            glVertex2i(round(pntX[i]), round(pntY[i] * -1));
        }
    }
    //Reflection about Y axis
    else if (option == 2 ) {
        for (int i = 0; i < edges; i++) {
            glVertex2i(round(pntX[i] * -1), round(pntY[i]));
        }
    }
    //origin reflection
    else if (option == 3) {
        for (int i = 0; i < edges; i++) {
            glVertex2i(round(pntX[i] * -1), round(pntY[i])*-1);
        }
    }
        //Y=X reflection
    else if (option == 4) {
        for (int i = 0; i < edges; i++) {
        glVertex2i(round(pntY[i]), round(pntX[i]));
        }
    }
    glEnd();
}
void shearing(int option) {
    glBegin(GL_POLYGON);
    glColor3f(0.02, 0.72, 0.09);
    //translating the transformed polygon so that it doesn't overlap on the orig
    if (option == 5) {
        glVertex2i(pntX[0]+200, pntY[0]);
        glVertex2i(pntX[1] + shearingX + 200, pntY[1]);
        glVertex2i(pntX[2] + shearingX + 200, pntY[2]);
        glVertex2i(pntX[3] + 200, pntY[3]);
    }
    else if (option == 6) {
        glVertex2i(pntX[0] + 200, pntY[0]);
        glVertex2i(pntX[1] + 200, pntY[1]);
        glVertex2i(pntX[2] + 200, pntY[2] + shearingY);
        glVertex2i(pntX[3] + 200, pntY[3] + shearingY);
    }
    glEnd();
}

void myInit(void) {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(4.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-640.0, 640.0, -640, 640.0);
}

void myDisplay(void) {
    while (true) {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2i(-640, 0);
    glVertex2i(640, 0);
    glEnd();
    glBegin(GL_LINES);
    glVertex2i(0, -640);
    glVertex2i(0, 640);
    glEnd();
    drawPolygon();
    cout << "1. Reflection (X axis)\n";
    cout << "2. Reflection (Y axis)\n";
    cout << "3. Reflection (Origin)\n";
    cout << "4. Reflection (Y = X)\n";
    cout << "5. Shearing (X axis) \n";
    cout << "6. Shearing (Y axis) \n";
    cout << "7. Exit\n";
    cout << "Enter your choice : ";
    cin >> op;
    if (op >= 1 && op <= 4) {
        reflection(op);
    }
    else if (op == 5 ) {
        cout << "Enter the shearing factor for X: ";cin>> shearingX;
        shearing(op);
    }
    else if(op == 6) {
        cout << "Enter the shearing factor for Y: ";cin>> shearingY;
        shearing(op);
    }
    else{
        break;
    }
    glFlush();
    }
}

int main(int argc, char** argv) {
    cout << "For Polygon:\n" << endl;
    cout << "Enter no of edges: "; cin >> edges;
    cout << "\nEnter Polygon Coordinates : \n";
    for (int i = 0; i < edges; i++) {
        cout << "Vertex " << i + 1 << " : "; cin >> pntX1 >> pntY1;
        pntX.push_back(pntX1);
        pntY.push_back(pntY1);
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 640);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Transformations - 2");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
}
```

3.1 Output

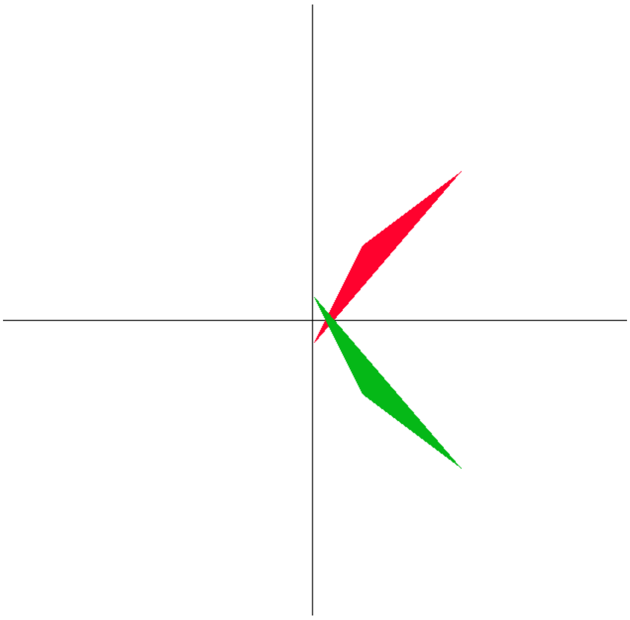

Figure 1: Reflection X axis

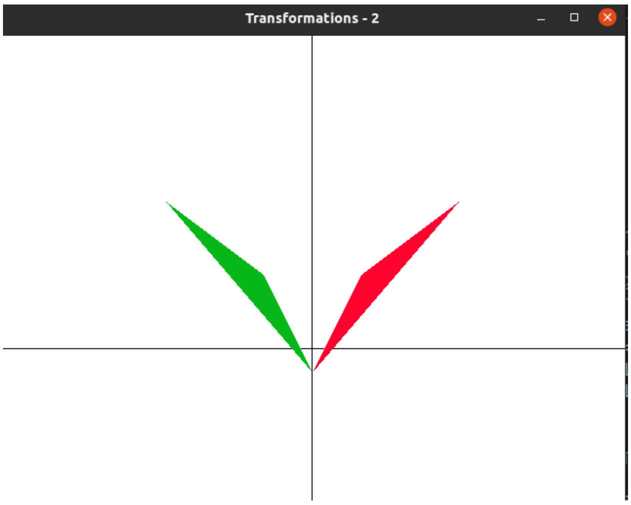

Figure 2: Reflection Y axis

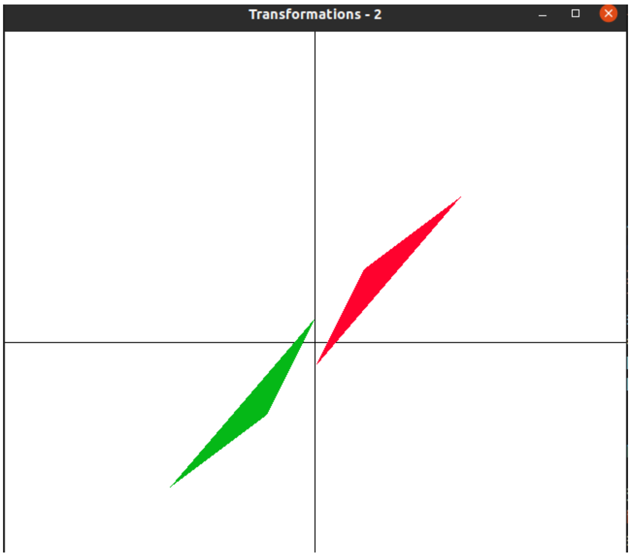

Figure 3: Reflection Origin

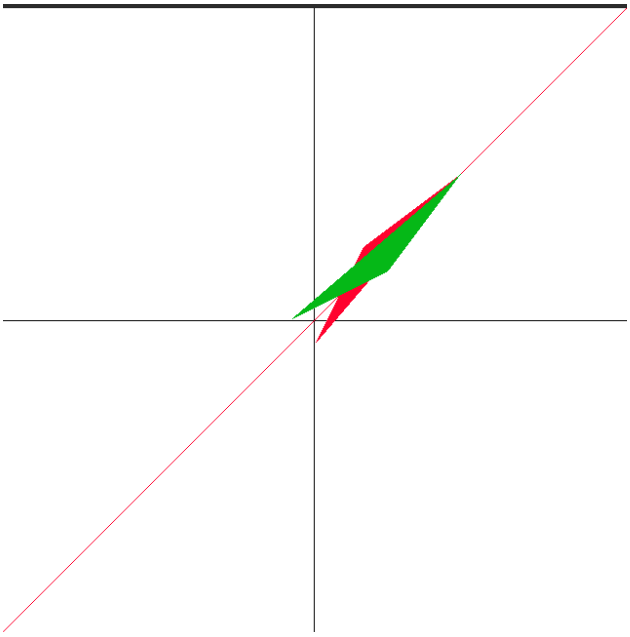

Figure 4: Reflection Y = X

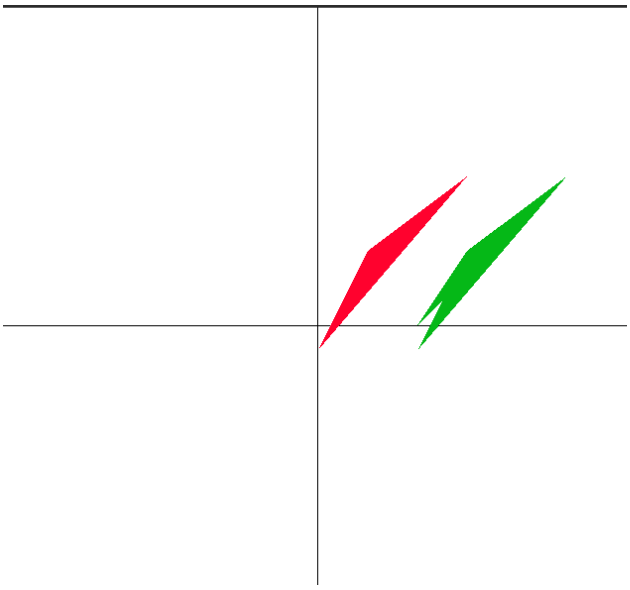

Figure 5: Shearing X axis

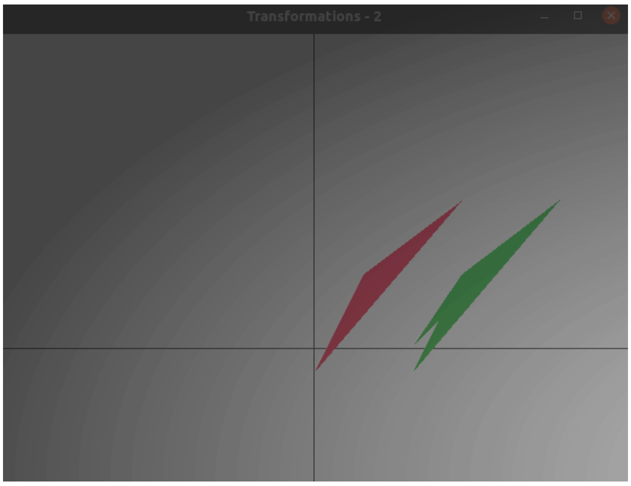

Figure 6: Shearing Y axis

4 RESULT

Thus compiled and executed a C++ menu-driven program using OPENGL to perform 2D transformations –reflection and shearing for polygon successfully.