# SSN COLLEGE OF ENGINEERING
## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
## UCS1712 – GRAPHICS AND MULTIMEDIA LAB
## EX NO: 5b – 2D Transformations – Reflection and Shearing

---

Name : Mohamed Hashim G
RegNo : 185001094

---

**AIM:**

To write a C++ program to perform rotation and shearing on polygon.

**ALGORITHM:**

1. Read the vertices for polygon be transformed as input.

2. Read the choice of operation to be performed.

3. For Reflection:

    a. For each vertex (x,y) apply reflection as follows:

        i. For reflection along x axis: y = -y

        ii. For reflection along y axis: x = -x

        iii. For reflection along origin: x = -x and y = -y

        iv. For reflection along x=y : x,y = y,x

    b. For polygon, draw the reflection of polygon using the four new vertices.

4. For Shearing:

    a. Read shearing axis and shearing factor sf.

    b. If shearing axis is along X-axis:

        i. Add the shearing factor to the x-coordinates of the 2 nd and 3 rd vertex.

    c. If shearing axis is along Y-axis:

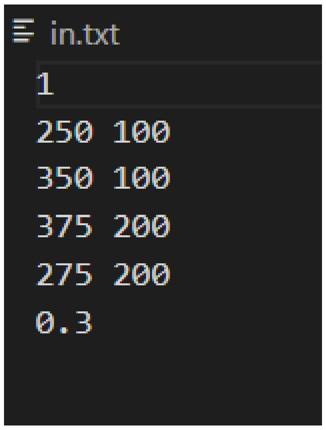
        i. Add the shearing factor to the y-coordinates of the 3 rd and 4 th vertex.

d. For polygon, draw the sheared polygon using the four new vertices.

**CODE:**

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <vector>
#include <GL/glut.h>

using namespace std;
int choice;

// Polygon
vector<int> pntX;
vector<int> pntY;

int xaxis=500, yaxis=500;
float shear;

double round(double d)
{
    return floor(d + 0.5);
}

// Polygon
void drawPolygon()
{
    glBegin(GL_QUADS);
    glColor3f(1.0, 1.0, 1.0);
    for (int i = 0; i < 4; i++)
    {
        glVertex2i(xaxis+pntX[i], yaxis+pntY[i]);
    }
    glEnd();
}

// REFLECTION
void drawPolygonReflection(int rx, int ry)
{
    glBegin(GL_QUADS);
    for (int i = 0; i < 4; i++)
    {
        glVertex2i(xaxis+pntX[i]*rx, yaxis+pntY[i]*ry);
    }
    glEnd();
}

// SHEAR
void drawPolygonShearX()
{
    glBegin(GL_QUADS);
    glColor3f(1.0, 0.0, 0.0);

    glVertex2f(xaxis+pntX[0] , yaxis+pntY[0]);
    glVertex2f(xaxis+pntX[1] , yaxis+pntY[1]);
    glVertex2f(xaxis+pntX[2] + (pntY[2]*shear), yaxis+pntY[2]);
    glVertex2f(xaxis+pntX[3] + (pntY[3]*shear), yaxis+pntY[3]);
    glEnd();
}

void drawPolygonShearY()
{
    glBegin(GL_QUADS);
    glColor3f(0.0, 0.0, 1.0);

    glVertex2f(xaxis+pntX[0] -100, yaxis+pntY[0] );
    glVertex2f(xaxis+pntX[1] -100, yaxis+pntY[1] + (pntX[1]*shear) );
    glVertex2f(xaxis+pntX[2] -100, yaxis+pntY[2] + (pntX[2]*shear) );
    glVertex2f(xaxis+pntX[3] -100, yaxis+pntY[3] );
    glEnd();
}

void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    // draw axes
    // Y-axis
    glBegin(GL_LINES);
        glVertex2f(500, 0);
        glVertex2f(500, 1000);
    glEnd();

    // X-axis
    glBegin(GL_LINES);
        glVertex2f(0, 500);
        glVertex2f(1000, 500);
    glEnd();
    drawPolygon();

    if(choice==0) // REFLECTION
    {
        // X=Y
        glBegin(GL_LINES);
            glVertex2f(0, 0);
            glVertex2f(1000, 1000);
        glEnd();

        glColor3f(0.0, 0.0, 1.0);
        drawPolygonReflection(1, -1);

        glColor3f(1.0, 0.0, 0.0);
        drawPolygonReflection(-1, 1);

        glColor3f(0.0, 1.0, 1.0);
        drawPolygonReflection(-1, -1);

        glBegin(GL_QUADS);
            glColor3f(0.8, 0.3, 0.9);
            for (int i = 0; i < 4; i++)
            {
                glVertex2i(yaxis+pntY[i], xaxis+pntX[i]);
            }
        glEnd();
    }
    else // SHEAR
    {
        // cout<<"shear";
        drawPolygonShearX();
        drawPolygonShearY();
    }
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(700, 0);
    glutInitWindowSize(750, 750);
    glutCreateWindow("5b-Reflection & Shear");
    glClearColor(0,0,0,1);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 1000, 0.0, 1000);

    int i, pntX1, pntY1;
    freopen("in.txt", "r", stdin);
    cin>>choice;
    for (i = 0; i < 4; i++)
    {
        cin >> pntX1 >> pntY1;
        pntX.push_back(pntX1);
        pntY.push_back(pntY1);
    }
    // POLYGON
    if(choice==1)
    {
        cin>>shear;
    }

    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

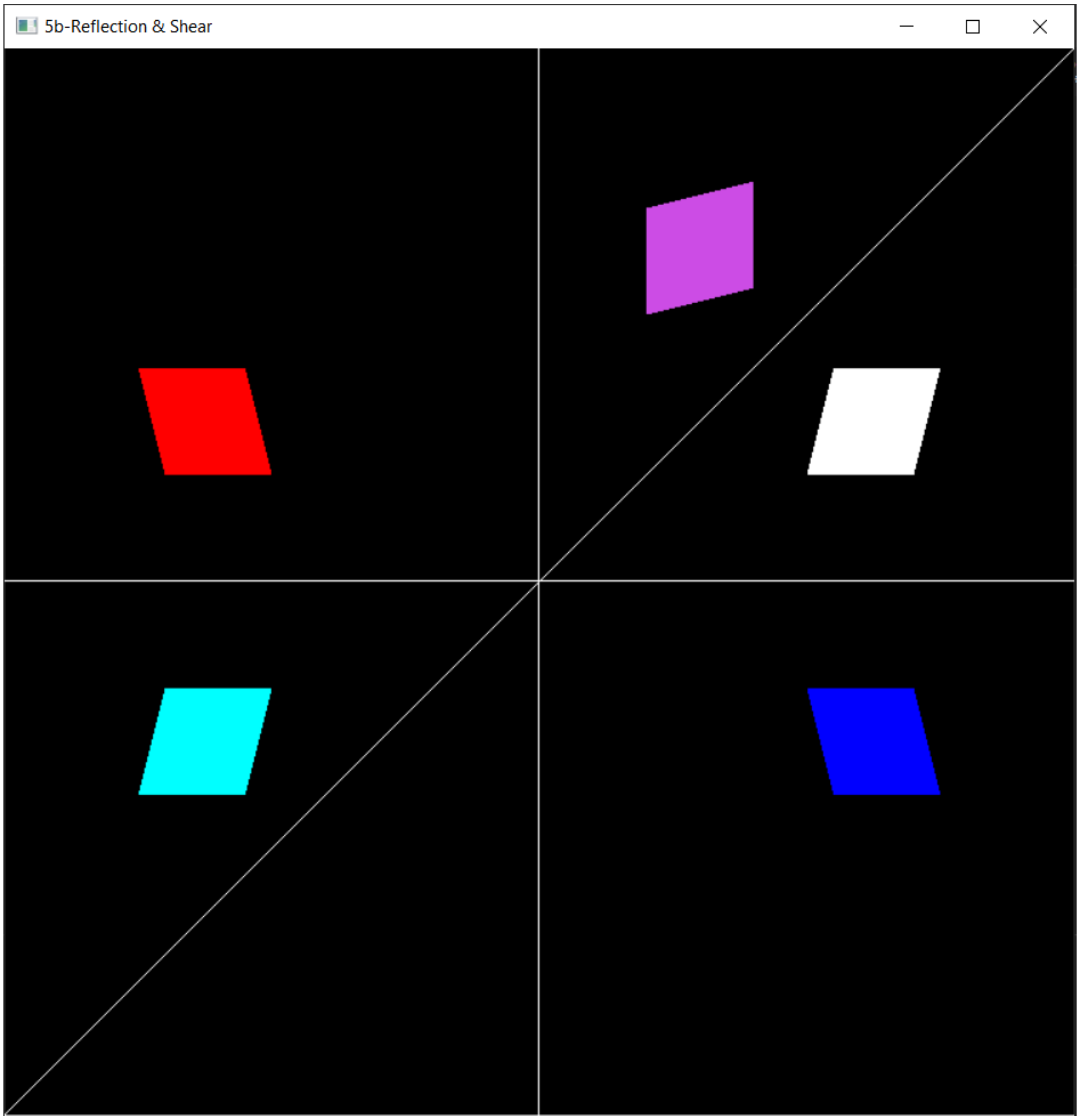
**OUTPUT:**



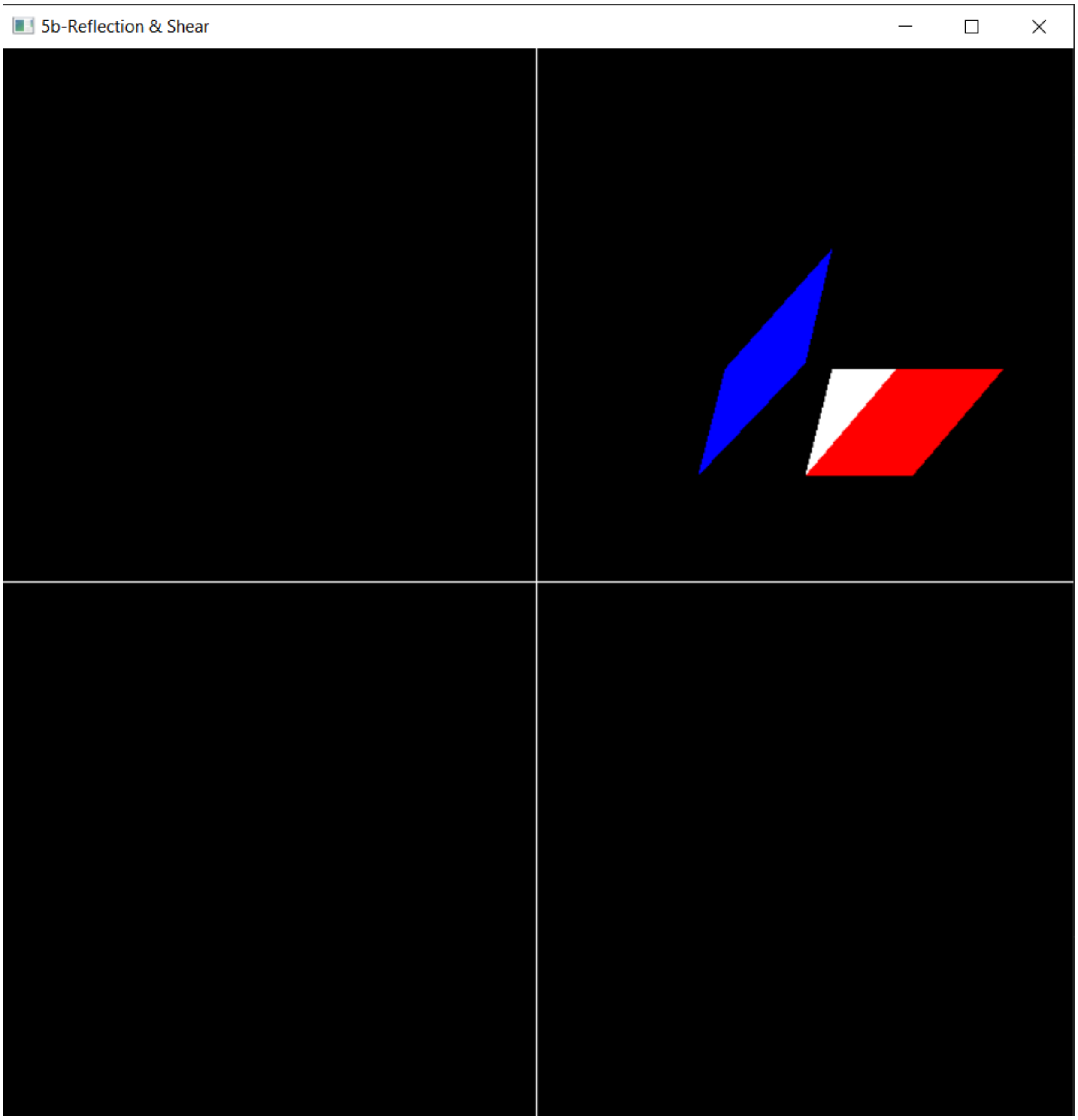
in.txt

```
1
250 100
350 100
375 200
275 200
0.3
```

## REFLECTION



**SHEAR**



**RESULT:**

Thus 2D Transformations like rotation, shearing have been performed on polygons.