**SSN College of Engineering**

**Department of Computer Science and Engineering**

## UCS1712 – GRAPHICS AND MULTIMEDIA LAB

**EX NO: 5b – 2D Transformations – Reflection and Shearing**

SATHEESH KUMAR G R

185001136

**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:
   1. For each vertex (x,y) apply reflection as follows:
      1. For reflection along x axis: y = -y
      2. For reflection along y axis: x = -x
      3. For reflection along origin: x = -x and y = -y
      4. For reflection along x=y : x,y = y,x
   2. For polygon, draw the reflection of polygon using the four new vertices.
4. For Shearing:
   1. Read shearing axis and shearing factor sf.
   2. If shearing axis is along X-axis:
      1. Add the shearing factor to the x-coordinates of the 2nd and 3rd vertex.
   3. If shearing axis is along Y-axis:
      1. Add the shearing factor to the y-coordinates of the 3rd and 4th vertex.
   4. 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 pntX1, pntY1, choice = 0, edges=4;

vector<int> pntX;

vector<int> pntY;

vector<int> sxpntX;

vector<int> sxpntY;

vector<int> sypntX;

vector<int> sypntY;

char reflectionAxis, shearingAxis;

int shearingX, shearingY;

double round(double d)

{

    return floor(d + 0.5);

}

void drawPolygon(vector<int> tX,vector<int> tY)

{

    glBegin(GL\_POLYGON);

    glColor3f(1.0, 0.0, 0.0);

    for (int i = 0; i < edges; i++)

    {

        glVertex2i(tX[i], tY[i]);

    }

    glEnd();

}

void drawPolygonMirrorReflection()

{

    {

        glBegin(GL\_POLYGON);

        glColor3f(0.0, 0.0, 1.0);

        for (int i = 0; i < edges; i++)

        {

            glVertex2i(round(pntX[i]), round(pntY[i] \* -1));

        }

        glEnd();

    }

    {

        glBegin(GL\_POLYGON);

        glColor3f(0.0, 1.0, 0.0);

        for (int i = 0; i < edges; i++)

        {

            glVertex2i(round(pntX[i] \* -1), round(pntY[i]));

        }

        glEnd();

    }

    {

        glBegin(GL\_POLYGON);

        glColor3f(1.0, 1.0, 0.0);

        for (int i = 0; i < edges; i++)

        {

            glVertex2i(round(pntX[i] \* -1), round(pntY[i]\* -1));

        }

        glEnd();

    }

    {

        glBegin(GL\_POLYGON);

        glColor3f(0.0, 1.0, 1.0);

        for (int i = 0; i < edges; i++)

        {

            glVertex2i(round(pntY[i]), round(pntX[i]));

        }

        glEnd();

    }

}

    void drawPolygonShearingX()

    {

        glBegin(GL\_POLYGON);

        glColor3f(0.5, 0.0, 1.0);

        glVertex2i(sxpntX[0], sxpntY[0]);

        glVertex2i(sxpntX[1] + shearingX, sxpntY[1]);

        glVertex2i(sxpntX[2] + shearingX, sxpntY[2]);

        glVertex2i(sxpntX[3], sxpntY[3]);

        glEnd();

    }

    void drawPolygonShearingY()

    {

        glBegin(GL\_POLYGON);

        glColor3f(0.5, 0.0, 1.0);

        glVertex2i(sypntX[0], sypntY[0]);

        glVertex2i(sypntX[1], sypntY[1]);

        glVertex2i(sypntX[2], sypntY[2] + shearingY);

        glVertex2i(sypntX[3], sypntY[3] + shearingY);

        glEnd();

    }

void myInit(void)

{

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glColor3f(2.0f, 0.0f, 1.0f);

    glPointSize(4.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-500,500,-500,500);

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0.0, 0.0, 0.0);

}

void myDisplay(void)

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0.0, 0.0, 0.0);

    drawPolygon(pntX,pntY);

    drawPolygonMirrorReflection();

    drawPolygon(sxpntX,sxpntY);

    drawPolygonShearingX();

    drawPolygon(sypntX,sypntY);

    drawPolygonShearingY();

    glFlush();

}

int main(int argc, char\*\* argv)

{

    cout << "Enter your choice:\n\n" << endl;

    cout << "1. Mirror Reflection" << endl;

    cout << "2. Shearing along x" << endl;

    cout << "3. Shearing along y" << endl;

    cout << "0. Exit" << endl;

    cin >> choice;

    while(choice>0)

    {

    if (choice == 1)

    {

        cout << "Reflection ";

        cout << "\nFor Polygon:" << endl;

        cout << "No of edges: "; cin >> edges;

        for (int i = 0; i < edges; i++)

        {

            cout << "vertex  " << i + 1 << " : "; cin >> pntX1 >> pntY1;

            pntX.push\_back(pntX1);

            pntY.push\_back(pntY1);

        }

    }

    else if (choice == 2)

    {

            cout << "Shearing along x\n";

            cout << "Shearing factor for X: "; cin >> shearingX;

                cout << "\nFor Polygon:" << endl;

                cout << "No of edges: "; cin >> edges;

                for (int i = 0; i < edges; i++)

                {

                    cout << "vertex  " << i + 1 << " : "; cin >> pntX1 >> pntY1;

                    sxpntX.push\_back(pntX1);

                    sxpntY.push\_back(pntY1);

                }

    }

    else

    {

            cout << "Shearing along y\n";

            cout << "Shearing factor for Y: "; cin >> shearingY;

                cout << "\nFor Polygon:" << endl;

                for (int i = 0; i < edges; i++)

                {

                    cout << "vertex  " << i + 1 << " : "; cin >> pntX1 >> pntY1;

                    sypntX.push\_back(pntX1);

                    sypntY.push\_back(pntY1);

                }

    }

    cout << "Enter your choice:" << endl;

cin >> choice;

    }

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(1000,1000);

    glutInitWindowPosition(0,500);

    glutCreateWindow("2D Transformations");

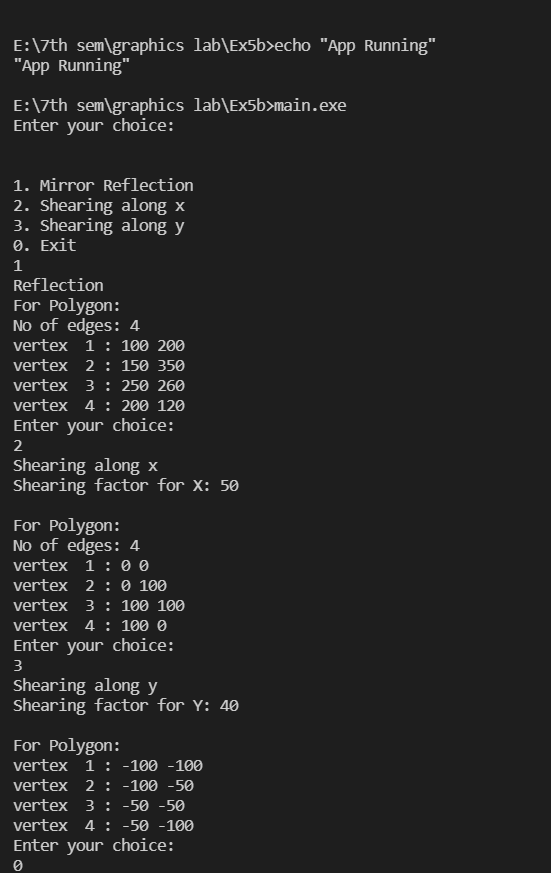
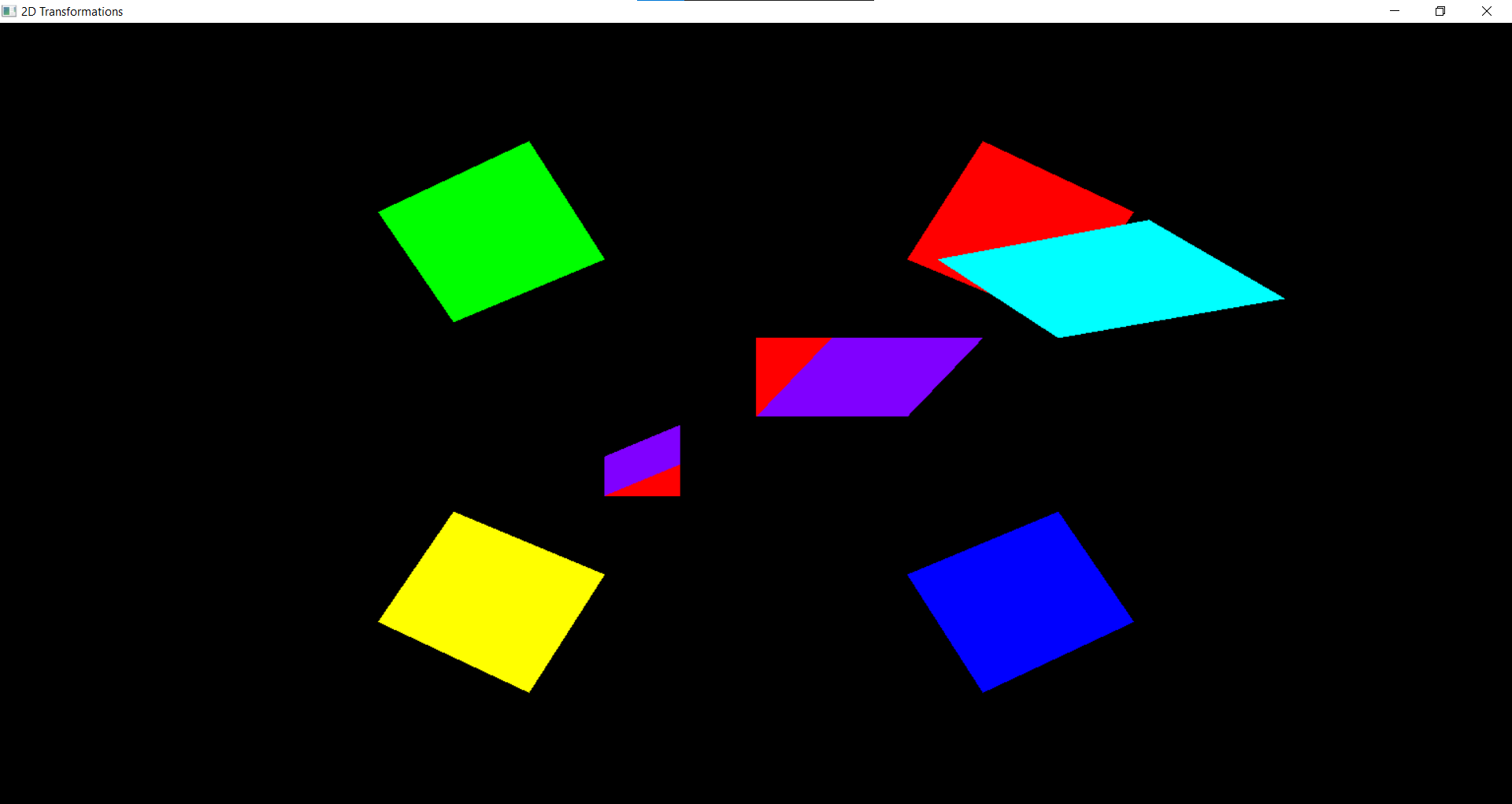
    glutDisplayFunc(myDisplay);

    myInit();

    glutMainLoop();

    return 0;

}

**OUTPUT:** ****