**SSN COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**UCS1712 – GRAPHICS AND MULTIMEDIA LAB**

**EX NO: 5a – 2D Transformations – Translation, Rotation and Scaling**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Name   : Mohamed Hashim G

RegNo : 185001094

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AIM:**

To draw lines as a series of points using DDA line drawing algorithm.

**ALGORITHM:**

1. Read the vertices for polygon and line to be transformed as input.

2. Read the choice of operation to be performed.

3. For Translation:

a. Read translation factor (tx,ty).

b. For each vertex (x,y) apply translation as follows:

i. x = x+tx

ii. y = y+ty

c. For polygon, draw the translated polygon using the four new vertices.

d. For line, draw the translated line using the two new vertices.

4. For Rotation:

a. Read the degree of rotation Ɵ for polygon and line.

b. Set the fixed point (fx,fy) as the first vertex.

c. For each vertex (x,y) the new vertex is computed as follows:

i. x = fx + (x-fx)cos(Ɵ) – (y-fy)sin(Ɵ)

ii. y = fy + (x-fx)sin(Ɵ) + (y-fy)cos(Ɵ)

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

e. For line, draw the rotated line using the two new vertices.

5. For Scaling:

a. Read scaling factor (sx,sy).

b. Set the fixed point (fx,fy) as the first vertex.

c. For each vertex (x,y) apply scaling as follows:

i. x = (x\*sx) + fx\*(1-sx)

ii. y = (y\*sy) + fy\*(1-sy)

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

e. For line, draw the scaled line using the two 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 transX, transY;

double scaleX, scaleY;

double angle, angleRad;

// Line

vector<int> lpntX;

vector<int> lpntY;

int ltransX, ltransY;

double lscaleX, lscaleY;

double langle, langleRad;

double round(double d)

{

    return floor(d + 0.5);

}

// line

void drawLine()

{

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 1.0);

    for (int i = 0; i < 2; i++)

    {

        glVertex2i(lpntX[i], lpntY[i]);

    }

    glEnd();

}

void drawLineTrans(int x, int y)

{

    glBegin(GL\_LINES);

    glColor3f(0.0, 1.0, 0.0);

    for (int i = 0; i < 2; i++)

    {

        glVertex2i(lpntX[i] + x, lpntY[i] + y);

    }

    glEnd();

}

void drawLineScale(double x, double y)

{

    glBegin(GL\_LINES);

    glColor3f(1.0, 1.0, 0.0);

    for (int i = 0; i < 2; i++)

    {

        glVertex2i(round(lpntX[i] \* x), round(lpntY[i] \* y));

    }

    glEnd();

}

void drawLineRotation(double angleRad)

{

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 1.0);

    for (int i = 0; i < 2; i++)

    {

        glVertex2i(round((lpntX[i] \* cos(angleRad)) - (lpntY[i] \* sin(angleRad))), round((lpntX[i] \* sin(angleRad)) + (lpntY[i] \* cos(angleRad))));

    }

    glEnd();

}

// Polygon

void drawPolygon()

{

    glBegin(GL\_QUADS);

    glColor3f(1.0, 1.0, 1.0);

    for (int i = 0; i < 4; i++)

    {

        glVertex2i(pntX[i], pntY[i]);

    }

    glEnd();

}

void drawPolygonTrans(int x, int y)

{

    glBegin(GL\_QUADS);

    glColor3f(0.0, 1.0, 0.0);

    for (int i = 0; i < 4; i++)

    {

        glVertex2i(pntX[i] + x, pntY[i] + y);

    }

    glEnd();

}

void drawPolygonScale(double x, double y)

{

    glBegin(GL\_QUADS);

    glColor3f(1.0, 1.0, 0.0);

    for (int i = 0; i < 4; i++)

    {

        glVertex2i(round(pntX[i] \* x), round(pntY[i] \* y));

    }

    glEnd();

}

void drawPolygonRotation(double angleRad)

{

    glBegin(GL\_QUADS);

    glColor3f(0.0, 0.0, 1.0);

    for (int i = 0; i < 4; i++)

    {

        glVertex2i(round((pntX[i] \* cos(angleRad)) - (pntY[i] \* sin(angleRad))), round((pntX[i] \* sin(angleRad)) + (pntY[i] \* cos(angleRad))));

    }

    glEnd();

}

void myDisplay(void)

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0.0, 0.0, 0.0);

    if(choice==0)

    {

        drawPolygon();

        drawPolygonTrans(transX, transY);

        if(scaleX > 1 || scaleY > 1)

        {

            drawPolygonScale(scaleX, scaleY);

            drawPolygon();

        }

        else

        {

            drawPolygonScale(scaleX, scaleY);

        }

        drawPolygonRotation(angleRad);

    }

    else

    {

        drawLine();

        drawLineTrans(ltransX,ltransY);

        if(lscaleX > 1 || lscaleY > 1)

        {

            drawLineScale(lscaleX, lscaleY);

            drawLine();

        }

        else

        {

            drawLineScale(lscaleX, lscaleY);

        }

        drawLineRotation(langleRad);

    }

    glFlush();

}

int main(int argc, char\*\* argv)

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowPosition(700, 0);

    glutInitWindowSize(750, 750);

    glutCreateWindow("5a-Geometic Transforamtion");

    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;

    // POLYGON

    if(choice==0)

    {

        for (i = 0; i < 4; i++)

        {

            cin >> pntX1 >> pntY1;

            pntX.push\_back(pntX1);

            pntY.push\_back(pntY1);

        }

        cin >> transX >> transY;

        cin >> scaleX >> scaleY;

        cin >> angle;

        angleRad = angle \* 3.1416 / 180;

    }

    else

    {

        for (i = 0; i < 2; i++)

        {

            cin >> pntX1 >> pntY1;

            lpntX.push\_back(pntX1);

            lpntY.push\_back(pntY1);

        }

        cin >> ltransX >> ltransY;

        cin >> lscaleX >> lscaleY;

        cin >> langle;

        langleRad = langle \* 3.1416 / 180;

    }

    glutDisplayFunc(myDisplay);

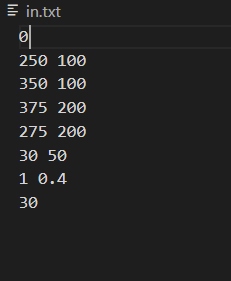
    glutMainLoop();

    return 0;

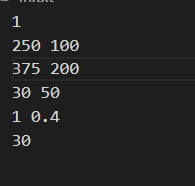
}

**OUTPUT:**

**Polygon**



**Line**



White- original line

Blue- Rotation

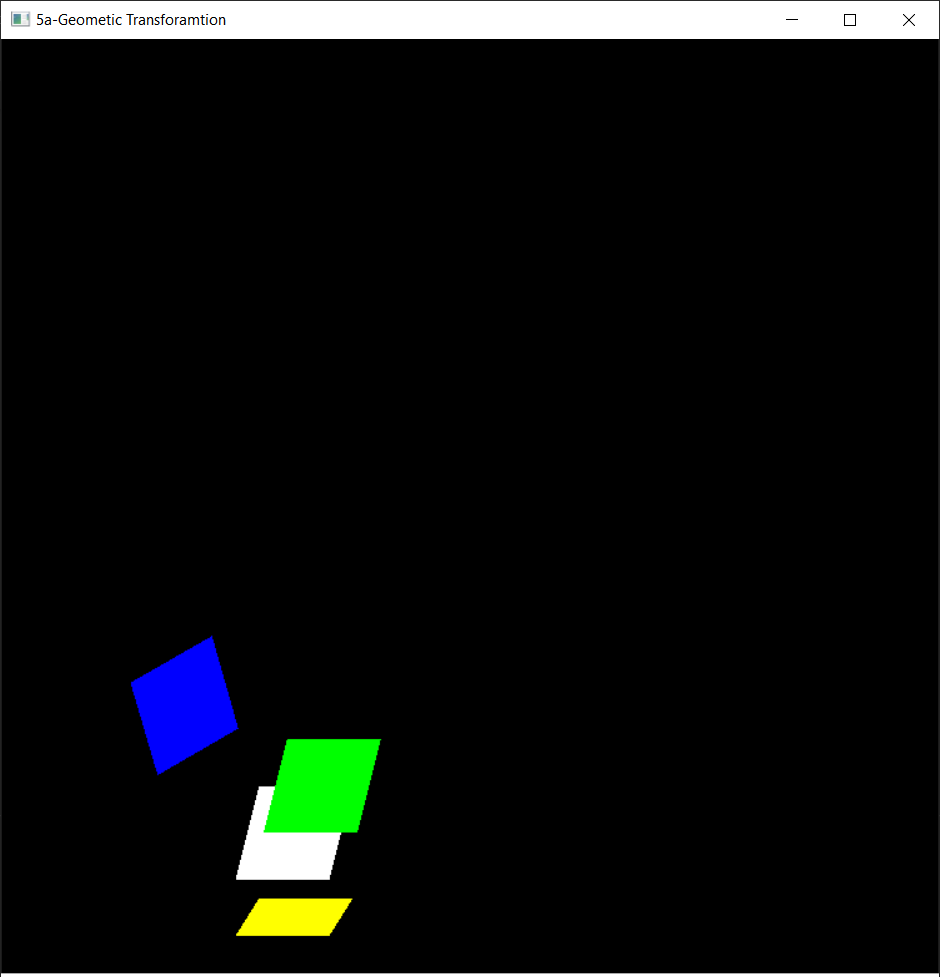
Green- Translation

Yellow- Scaling

**For Line**



**For Polygon**



**RESULT:**

Thus 2D Transformations like Translation, Rotation and Scaling have been performed on a

polygon and a line.