**SSN College of Engineering**

**Department of Computer Science and Engineering**

## UCS1712 – GRAPHICS AND MULTIMEDIA LAB

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

SATHEESH KUMAR G R

185001136

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:
   1. Read translation factor (tx,ty).
   2. For each vertex (x,y) apply translation as follows:
      1. x = x+tx
      2. y = y+ty
   3. For polygon, draw the translated polygon using the four new vertices.
   4. For line, draw the translated line using the two new vertices.
4. For Rotation:
   1. Read the degree of rotation Ɵ for polygon and line.
   2. Set the fixed point (fx,fy) as the first vertex.
   3. For each vertex (x,y) the new vertex is computed as follows:
      1. x = fx + (x-fx)cos(Ɵ) – (y-fy)sin(Ɵ)
      2. y = fy + (x-fx)sin(Ɵ) + (y-fy)cos(Ɵ)
   4. For polygon, draw the rotated polygon using the four new vertices.
   5. For line, draw the rotated line using the two new vertices.
5. For Scaling:
   1. Read scaling factor (sx,sy).
   2. Set the fixed point (fx,fy) as the first vertex.
   3. For each vertex (x,y) apply scaling as follows:
      1. x = (x\*sx) + fx\*(1-sx)
      2. y = (y\*sy) + fy\*(1-sy)
   4. For polygon, draw the scaled polygon using the four new vertices.
   5. 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 pntX1, pntY1, choice = 0, vertices;

vector<int> pntX;

vector<int> pntY;

vector<int> lpntX;

vector<int> lpntY;

int transX, transY;

double scaleX, scaleY;

double angle, angleRad;

int ltransX, ltransY;

double lscaleX, lscaleY;

double langle, langleRad;

double round(double d)

{

    return floor(d + 0.5);

}

void drawLine()

{

    glBegin(GL\_LINES);

    glColor3f(1.0, 0.0, 0.0);

    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++)

    {

        int m = round((lpntX[i] \* x) + (lpntX[0]\*(1-x)));

        int n = round((lpntY[i] \* y) + (lpntY[0]\*(1-y)));

        glVertex2i(m,n);

    }

    glEnd();

}

void drawLineRotation(double langleRad)

{

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 1.0);

    for (int i = 0; i < 2; i++)

    {

        int x =lpntX[0]+round(((lpntX[i]-lpntX[0]) \* cos(langleRad)) - ((lpntY[i]-lpntY[0]) \* sin(langleRad)));

        int y =lpntY[0]+round(((lpntX[i]-lpntX[0]) \* sin(langleRad)) + ((lpntY[i]-lpntY[0]) \* cos(langleRad)));

        glVertex2i(x,y);

    }

    glEnd();

}

void drawPolygon()

{

    glBegin(GL\_QUADS);

    glColor3f(1.0, 0.0, 0.0);

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

    {

        int m = round((pntX[i] \* x) + (pntX[0]\*(1-x)));

        int n = round((pntY[i] \* y) + (pntY[0]\*(1-y)));

        glVertex2i(m,n);

    }

    glEnd();

}

void drawPolygonRotation(double angleRad)

{

    glBegin(GL\_QUADS);

    glColor3f(0.0, 0.0, 1.0);

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

    {

        int x =pntX[0]+round(((pntX[i]-pntX[0]) \* cos(angleRad)) - ((pntY[i]-pntY[0]) \* sin(angleRad)));

        int y =pntY[0]+round(((pntX[i]-pntX[0]) \* sin(angleRad)) + ((pntY[i]-pntY[0]) \* cos(angleRad)));

        glVertex2i(x,y);

    }

    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)

{

        drawPolygonScale(scaleX, scaleY);

        drawPolygonTrans(transX, transY);

        drawPolygonRotation(angleRad);

        drawPolygon();

        if(scaleX<1 || scaleY<1)

            drawPolygonScale(scaleX, scaleY);

        drawLineScale(lscaleX, lscaleY);

        drawLineTrans(ltransX,ltransY);

        drawLineRotation(langleRad);

        drawLine();

        if(lscaleX<1 || lscaleY<1)

            drawLineScale(lscaleX, lscaleY);

    glFlush();

}

int main(int argc, char\*\* argv)

{

    cout << "\nPOLYGON\nEnter no of vertices: ";

    cin >> vertices;

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

    {

        cout << "Enter co-ordinates for vertex " << i + 1 << " : "; cin >> pntX1 >> pntY1;

        pntX.push\_back(pntX1);

        pntY.push\_back(pntY1);

    }

    cout << "\nLINE\nEnter the vertices\n";

    for (int i = 0; i < 2; i++)

    {

        cout << "Enter co-ordinates for vertex " << i + 1 << " : "; cin >> pntX1 >> pntY1;

        lpntX.push\_back(pntX1);

        lpntY.push\_back(pntY1);

    }

    cout <<  "POLYGON" << endl;

    cout << "1. Translation for Polygon" << endl;

    cout << "2. Scaling for Polygon" << endl;

    cout << "3. Rotation for Polygon" << endl;

    cout <<  "LINE" << endl;

    cout << "4. Translation for Line "<<endl;

    cout << "5. Scaling for Line" << endl;

    cout << "6. Rotation for Line" << endl;

    cout << "0. Exit\nEnter Choice:";

    cin >> choice;

    while(choice>0)

    {

    switch(choice)

    {

        case 1 : cout << "Enter the translation factor (tx,ty) for polygon: "; cin >> transX >> transY; break;

        case 2 : cout << "Enter the scaling factor (sx,sy) (First vertex is the fixed point) for polygon: "; cin >> scaleX >> scaleY;drawPolygonScale(scaleX, scaleY); break;

        case 3 : cout << "Enter the angle for rotation for polygon: "; cin >> angle;angleRad = angle \* 3.1416 / 180;break;

        case 4 : cout << "Enter the translation factor (tx,ty) for line: "; cin >> ltransX >> ltransY;break;

        case 5 : cout << "Enter the scaling factor (sx,sy) (First vertex is the fixed point) for line: "; cin >> lscaleX >> lscaleY;break;

        case 6 : cout << "Enter the angle for rotation for line: "; cin >> langle;langleRad = langle \* 3.1416 / 180;break;

        default : break;

    }

    cout << "\nEnter Choice:";

    cin >> choice;

    }

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(1000,1000);

    glutInitWindowPosition(0,500);

    glutCreateWindow("2D Transformations");

    glutDisplayFunc(myDisplay);

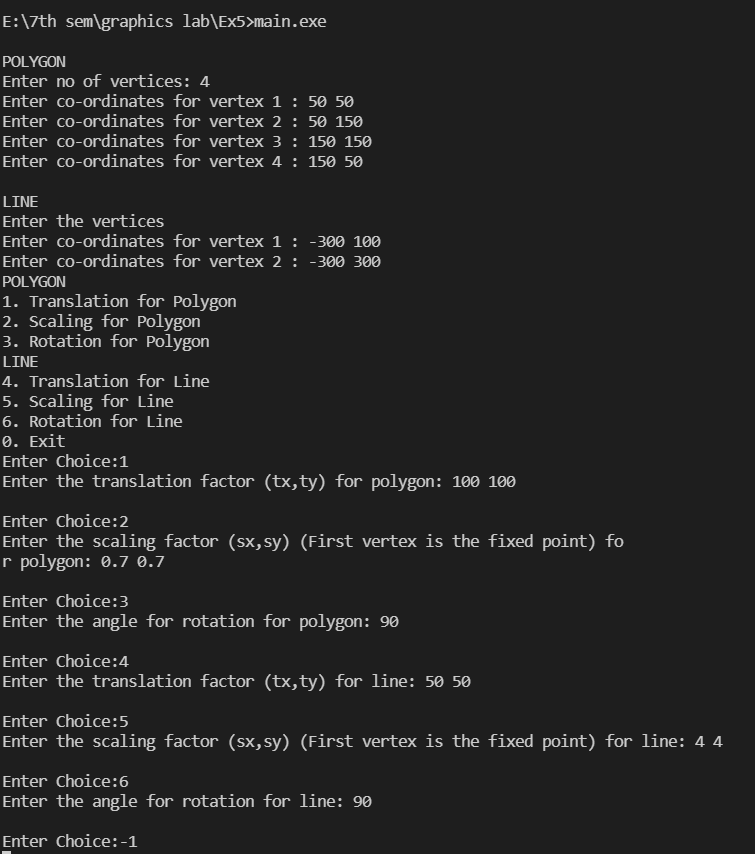
    myInit();

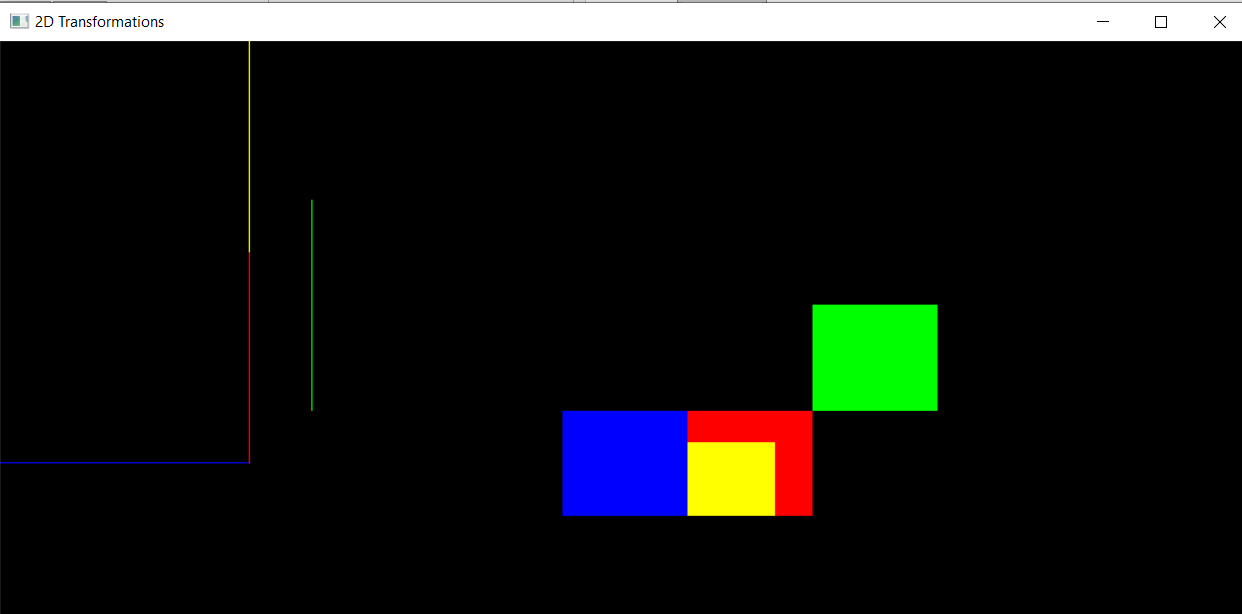
    glutMainLoop();

    return 0;

}

**OUTPUT:**

****



**RESULT:**

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