**SSN College of Engineering**

**Department of Computer Science and Engineering**

## UCS1712 – GRAPHICS AND MULTIMEDIA LAB

**EX6A - 2D Composite Transformations**

SATHEESH KUMAR G R

185001136

**1]AIM**

To write a program in C++ using openGL to perform the following 2Dimensional composite transformations.

**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 the choice is rotation and scaling

Get the rotation axis, pivot point and scaling factor from the user Perform rotation and save it to the polygon

Perform scaling and plot the polygon

* + If the choice is reflection and shearing

Get the reflection axis and shearing factor from the user perform reflection save it to the polygon

Perform shearing and plot the polygon

**3]CODE:**

#include <stdio.h>

#include <math.h>

#include <iostream>

#include <vector>

#include <GL/glut.h>

using namespace std;

int pntX1, pntY1, op = 0, edges, op1, op2;

int shearingX, shearingY;

vector<int> pntX, tempX;

vector<int> pntY, tempY;

int transX, transY;

double scaleX, scaleY;

double angle, angleRad;

char reflectionAxis;

double round(double d)

{

    return floor(d + 0.5);

}

void drawPolygon()

{

    glBegin(GL\_POLYGON);

    glColor3f(0.4, 0.6, 0.0);

    for (int i = 0; i < edges; i++)

    {

        glVertex2i(pntX[i], pntY[i]);

    }

    glEnd();

}

void translate(int x, int y)

{

    glBegin(GL\_POLYGON);

    glColor3f(0.7, 0.3, 0.4);

    for (int i = 0; i < edges; i++)

    {

        pntX[i] += x;

        pntY[i] += y;

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

    }

    glEnd();

}

void scale(double x, double y)

{

    glBegin(GL\_POLYGON);

    glColor3f(0.7, 0.3, 0.4);

    for (int i = 0; i < edges; i++)

    {

        pntX[i] = round(pntX[i] \* x) + 300;

        pntY[i] = round(pntY[i] \* y);

        glVertex2i(pntX[i], pntY[i]);

    }

    glEnd();

}

void rotate(double theta)

{

    glBegin(GL\_POLYGON);

    glColor3f(0.88, 0.1, 1.0);

    for (int i = 0; i < edges; i++)

    {

        int pntX1 = pntX[i];

        int pntY1 = pntY[i];

        pntX[i] = round((pntX1 \* cos(theta)) - (pntY1 \* sin(theta))); pntY[i] = round((pntX1 \* sin(theta)) + (pntY1 \* cos(theta))); //glVertex2i(pntX[i],pntY[i]);

    }

    glEnd();

}

void reflectX()

{

    for (int i = 0; i < edges; i++)

    {

        pntY[i] = pntY[i] \* -1;

    }

}

void reflectY()

{

    for (int i = 0; i < edges; i++)

    {

        pntX[i] = pntX[i] \* -1;

    }

}

void reflectOrigin()

{

    for (int i = 0; i < edges; i++)

    {

        pntX[i] = pntX[i] \* -1;

        pntY[i] = pntY[i] \* -1;

    }

}

void reflectDiag()

{

    for (int i = 0; i < edges; i++)

    {

        int temp = pntX[i];

        pntX[i] = pntY[i];

        pntY[i] = temp;

    }

    glEnd();

}

void shearX()

{

    glBegin(GL\_POLYGON);

    glColor3f(0.3, 0.3, 0.3);

    glVertex2i(pntX[0] + 150, pntY[0]);

    glVertex2i(pntX[1] + shearingX + 150, pntY[1]);

    glVertex2i(pntX[2] + shearingX + 150, pntY[2]);

    glVertex2i(pntX[3] + 150, pntY[3]);

    glEnd();

}

void shearY()

{

    glBegin(GL\_POLYGON);

    glColor3f(0.3, 0.3, 0.3);

    glVertex2i(pntX[0] + 150, pntY[0]);

    glVertex2i(pntX[1] + 150, pntY[1]);

    glVertex2i(pntX[2] + 150, pntY[2] + shearingY); glVertex2i(pntX[3] + 150, pntY[3] + shearingY); glEnd();

}

void myInit(void)

{

    glClearColor(0.0, 0.0, 0.0, 0.0);

    glColor3f(0.0f, 0.0f, 0.0f);

    glPointSize(4.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-500, 500, -500, 500);

}

void myDisplay(void)

{

    while (true) {

        glClear(GL\_COLOR\_BUFFER\_BIT);

        glColor3f(0.0, 0.0, 0.0);

        drawPolygon();

        cout << "\nSelect the required Composite Transformation:\n";

        cout << "1. Rotation & Scaling\n";

        cout << "2. Reflection & Shearing\n";

        cout << "3. Exit\n";

        cout << "Enter your choice : ";

        cin >> op;

        if (op == 3) {

            break;

        }

        if (op == 1)

        {

            cout << "Enter the angle for rotation: "; cin >> angle;

            angleRad = angle \* 3.1416 / 180;

            cout << "Enter fixed point: "; cin >> transX >> transY;

            translate(-transX, -transY);

            rotate(angleRad);

            translate(transX, transY);

            cout << "Enter the scaling factor for X and Y: "; cin >> scaleX  >> scaleY;

            scale(scaleX, scaleY);

        }

        else if (op == 2)

        {

            cout << "\nChoose reflection axis: \n";

            cout << "1. Reflect along X axis\n";

            cout << "2. Reflect along Y axis\n";

            cout << "3. Reflect about origin\n";

            cout << "4. Reflect along X=Y\n";

            cout << "Enter your choice : ";

            cin >> op1;

            if (op1 == 1)

            {

                reflectX();

            }

            else if (op1 == 2)

            {

                reflectY();

            }

            else if (op1 == 3)

            {

                reflectOrigin();

            }

            else if (op1 == 4)

            {

                reflectDiag();

            }

            cout << "\nChoose shearing axis: \n";

            cout << "1. Shear along X axis\n";

            cout << "2. Shear along Y axis\n";

            cout << "Enter your choice : ";

            cin >> op2;

            if (op2 == 1)

            {

                cout << "Enter the shearing factor for X: "; cin >>

                shearingX;

                shearX();

            }

            else if (op2 == 2)

            {

                cout << "Enter the shearing factor for Y: "; cin >>

                shearingY;

                shearY();

            }

        }

        pntX = tempX;

        pntY = tempY;

        glFlush();

    }

}

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

{

    cout << "\n2D-Transformations\n" << endl;

    cout << "\nFor 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);

        tempX.push\_back(pntX1);

        pntY.push\_back(pntY1);

        tempY.push\_back(pntY1);

    }

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(640, 480);

    glutInitWindowPosition(100, 150);

    glutCreateWindow("Composite Transformations");

    glutDisplayFunc(myDisplay);

    myInit();

    glutMainLoop();

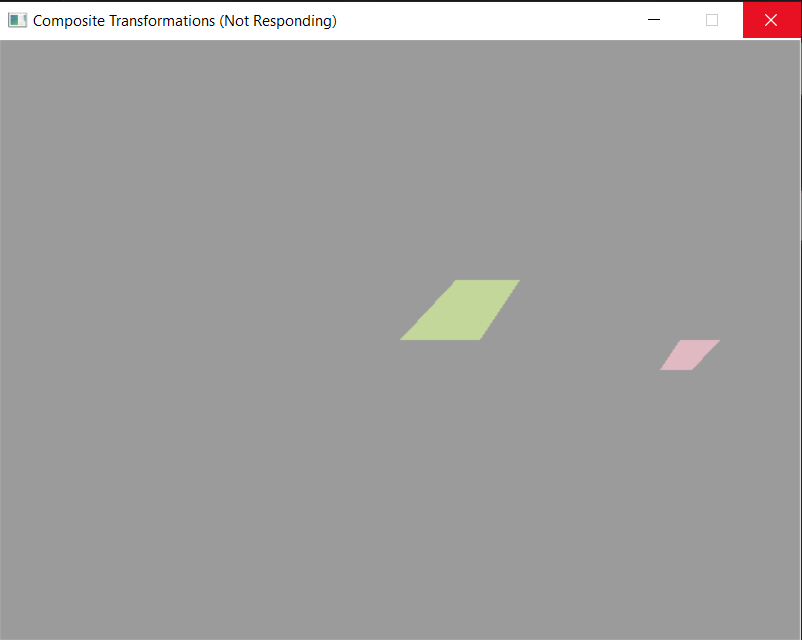
    return 0;

}

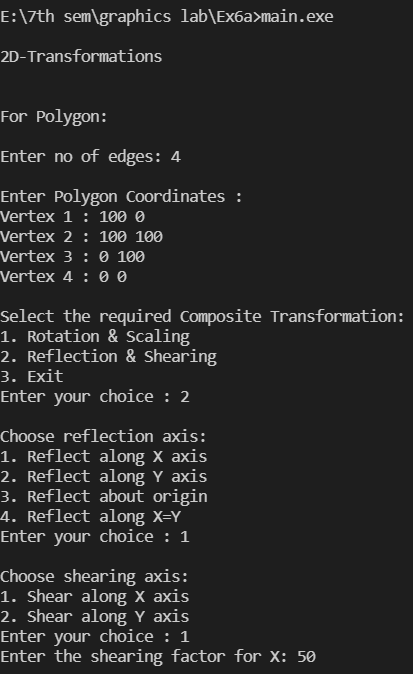
**4]OUTPUT:**

**ROTATION AND SCALING:**

****

****

**REFLECTION AND SHEARING:**

****

