SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 09/09/2021

EX NO: 6a

2D Transformations – Composite Transformation

Aim:

To perform rotation and scaling of an object Input: Rotation angle θ, Fixed point (xf,yf) and scaling factors sx and sy.

To perform reflection and shearing of an object Input: The reflecting axis and the shearing factor s.

Algorithm:

1. Read all the vertices from the user.
2. For Rotation and Scaling:
    a. Read angle of rotation, fixed point and scaling factor from the user.
    b. First move the polygon to the origin based on the fixed point by subtracting the fixed point coordinates on all the vertices.
    c. Now rotate the polygon by the given angle of rotation around the origin.
    d. Now scale the polygon by using the scaling factors.
    e. Now move the polygon back to the initial position by adding the fixed point coordinate value to all the vertices.
3. For Reflection and Shearing:
    a. Read the shearing factor from the user.
    b. Along X axis:

i. Perform shearing operation on the polygon by adding the shearing parameter on selected vertices based on the axis of reflection..
ii. Now multiply with -1 on all the x coordinates to reflect the sheared polygon along the X axis.

c. Along Y axis:
i. Perform shearing operation on the polygon by adding the shearing parameter on selected vertices based on the axis of reflection.
ii. Now multiply with -1 on all the y coordinates to reflect the sheared polygon along the Y axis.

Code:

```
#include <gl/glut.h>
#include <math.h>
#include <iostream>
#include <vector>
using namespace std;

double degree;
vector<int> pntX;
vector<int> pntY;
vector<int> tpntX;
vector<int> tpntY;

int fx, fy;
int vertices;

void myInit() {

    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-800, 800, -800, 800);
}

void drawPolygon(int flag, double r, double g, double b) {
    glBegin(GL_QUADS);
    glColor3f(r, g, b);
```

```
    if (flag == 0)
    {
        for (int i = 0; i < vertices; i++) {
            glVertex2i(pntX[i], pntY[i]);
        }
    }
    else
    {
        for (int i = 0; i < vertices; i++) {
            glVertex2i(tpntX[i], tpntY[i]);
        }
    }

    glEnd();

}

void rotatePolygon(double angleRad) {
    int temp_x = 0, temp_y = 0;
    for (int i = 0; i < vertices; i++) {
        temp_x = floor((tpntX[i] * cos(angleRad)) - (tpntY[i] *
sin(angleRad)));
        temp_y = floor((tpntX[i] * sin(angleRad)) + (tpntY[i] *
cos(angleRad)));
        tpntX[i] = temp_x;
        tpntY[i] = temp_y;
    }
}

void scalePolygon(double x, double y) {
    for (int i = 0; i < vertices; i++) {
        tpntX[i] = round(tpntX[i] * x);
        tpntY[i] = round(tpntY[i] * y);
    }
}

void shearReflectXaxis(int x)
{
    for (int i = 2; i < vertices; i++)
```

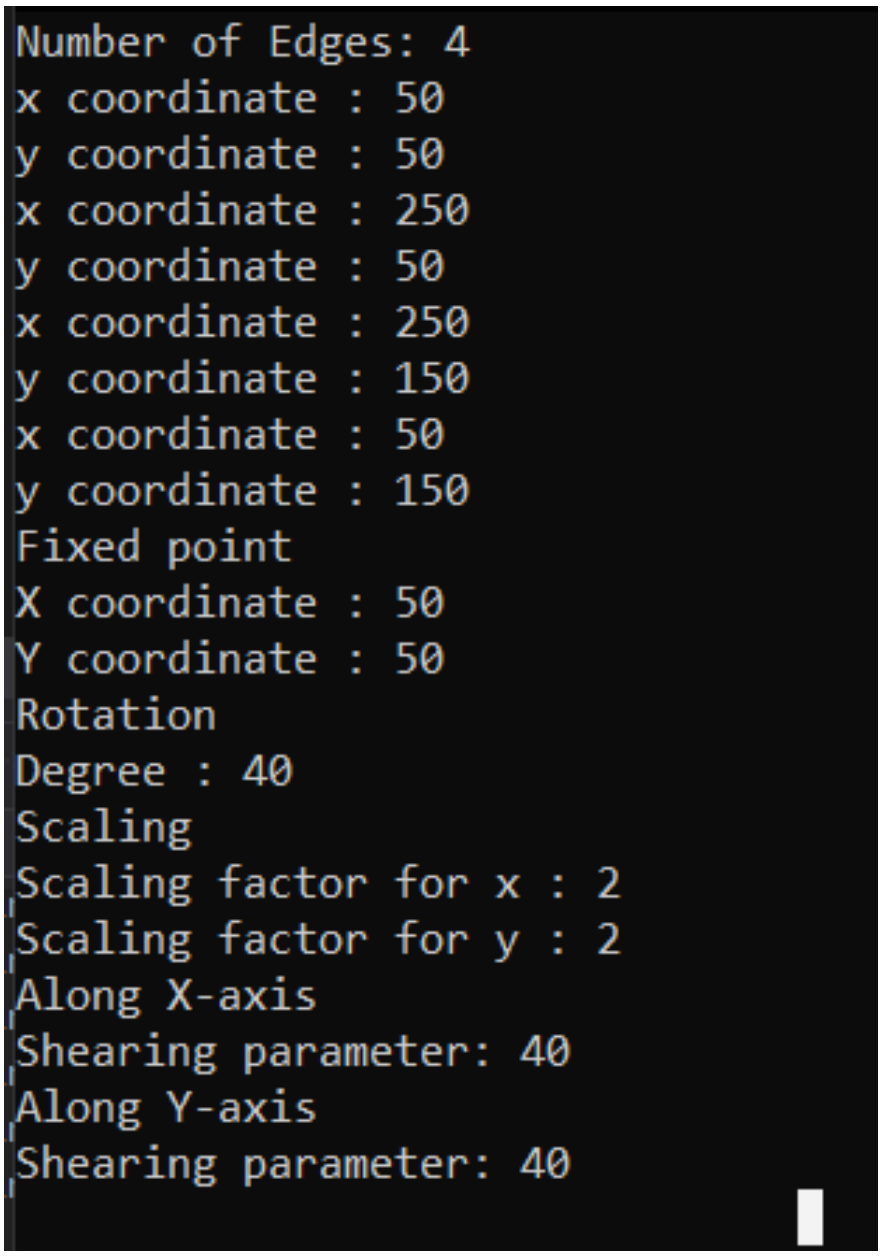
```
    {
        tpntX[i] += x;
    }

    for (int i = 0; i < vertices; i++)
    {
        tpntY[i] *= -1;
    }
}

void shearReflectYaxis(int y)
{
    for (int i = 1; i < 3; i++)
    {
        tpntY[i] += y;
    }

    for (int i = 0; i < vertices; i++)
    {
        tpntX[i] *= -1;
    }
}

void printMenu() {
    cout << "1. Translation" << "\n";
    cout << "2. Rotation" << "\n";
    cout << "3. Scaling" << "\n";
    cout << "-1. exit" << "\n";
    cout << "Choose : " << "\n";
}
void display(void) {

    glClear(GL_COLOR_BUFFER_BIT);
    int x, y;

    cout << "Number of Edges: ";
    cin >> vertices;
    for (int i = 0; i < vertices; i++)
    {
```

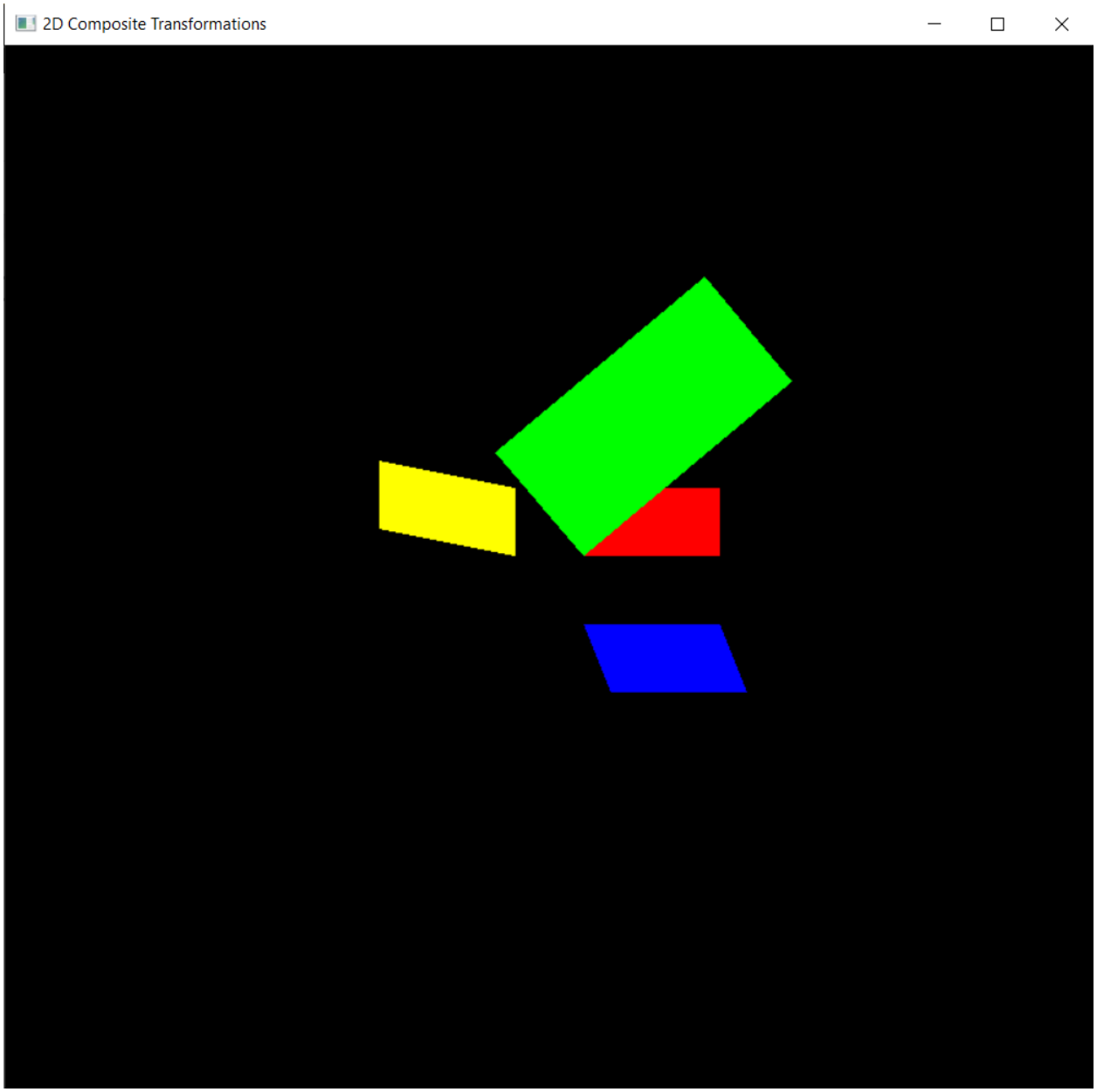
```
    cout << "x coordinate : ";
    cin >> x;
    cout << "y coordinate : ";
    cin >> y;
    pntX.push_back(x);
    pntY.push_back(y);
    tpntX.push_back(x);
    tpntY.push_back(y);

}

drawPolygon(0, 1.0, 0.0, 0.0);

cout << "Fixed point\n";
cout << "X coordinate : "; cin >> fx;
cout << "Y coordinate : "; cin >> fy;

for (int i = 0; i < vertices; i++)
{
    tpntX[i] = pntX[i] - fx;
    tpntY[i] = pntY[i] - fy;
}

cout << "Rotation\n";
cout << "Degree : "; cin >> degree;
rotatePolygon(degree * 3.14 / 180);

cout << "Scaling\n";
cout << "Scaling factor for x : "; cin >> x;
cout << "Scaling factor for y : "; cin >> y;
scalePolygon(x, y);

for (int i = 0; i < vertices; i++)
{
    tpntX[i] += fx;
    tpntY[i] += fy;
}

drawPolygon(1, 0.0, 1.0, 0.0);
```

```
    for (int i = 0; i < vertices; i++)
    {
        tpntX[i] = pntX[i];
        tpntY[i] = pntY[i];
    }

    cout << "Along X-axis\n";
    cout << "Shearing parameter: ";
    cin >> x;
    shearReflectXaxis(x);

    drawPolygon(1, 0.0, 0.0, 1.0);

    for (int i = 0; i < vertices; i++)
    {
        tpntX[i] = pntX[i];
        tpntY[i] = pntY[i];
    }

    cout << "Along Y-axis\n";
    cout << "Shearing parameter: ";
    cin >> y;
    shearReflectYaxis(y);

    drawPolygon(1, 1.0, 1.0, 0.0);


    glFlush();

}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("2D Composite Transformations"); // Create a
window with the given title
    myInit();
```

```
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;

}

// 4 50 50 250 50 250 150 50 150 50 50 40 2 2 40 40
```

Input/Output Screenshot:





Original Polygon
Rotated and Scaled
Reflected and sheared along Y axis
Reflected and sheared along X axis

Result:

Thus 2D composite Transformatiosn on a polygon is created and executed using opengl and c++.