SSN COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

EX NO: 6a – 2D Transformations – Composite Transformation

Name: Nachammai Devi Pooja S

RollNo:185001096

Date:9/09/2020

Aim:

To write a C++ menu-driven program using OPENGL to perform 2D composite transformations for polygons.

Algorithm:

Step 1: Obtain no. of edges of polygon from user
Step 2: Obtain coordinates of vertices
Step 3: Plot the original polygon and line
Step 4: Obtain transformation option from user
Step 5: option 1 – Rotation & Scaling:

- ➔ Get angle of rotation(theta), fixed point (x,y) and scaling factors as input from user
- ➔ Translate the polygon by -x and -y
- ➔ Rotate polygon by theta
- ➔ Translate the rotated polygon back by x and y
- ➔ Scale the polygon by scaling factors and plot final polygon multiply -1 to the Y coordinates of the original polygon and plot

Step 6: option 2-Reflection & Shearing:

-> Get reflection axis, shearing axis and shearing factor as input from user
-> Reflect the original polygon along the given reflection axis
-> Shearing the reflected polygon along the given shearing axis by the given shearing factor and plot final polygon

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)
```

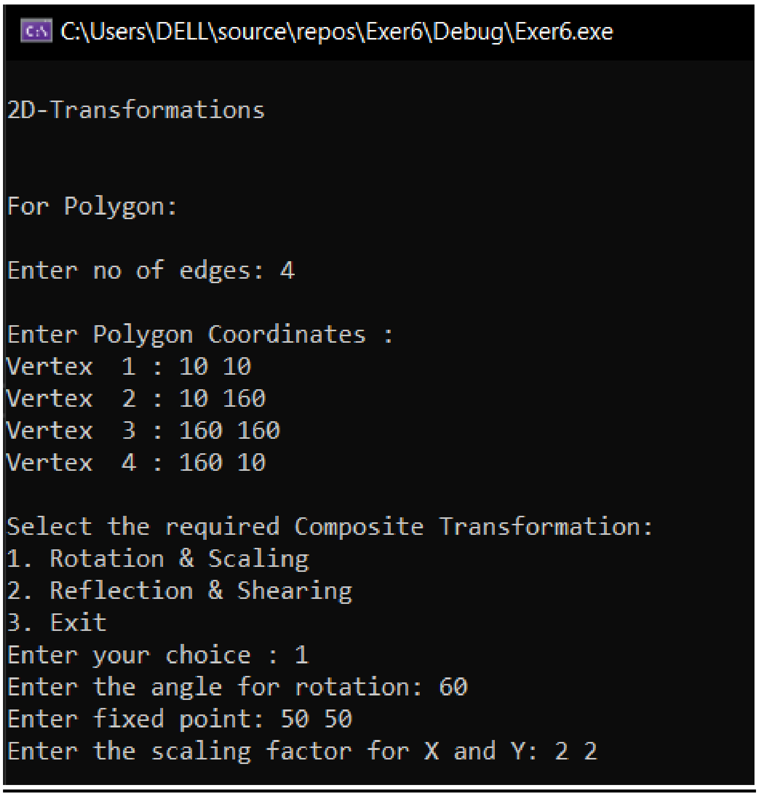
```
{
        return floor(d + 0.5);
}

void drawPolygon()
{
        glBegin(GL_POLYGON);
        glColor3f(0.4, 0, 0.2);
        for (int i = 0; i < edges; i++)
        {
                glVertex2i(pntX[i], pntY[i]);
        }
        glEnd();
}


void translate(int x, int y)
{
        glBegin(GL_POLYGON);
        glColor3f(6.08, 0.67, 1.0);
        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(6.08, 0.67, 1.0);
        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(6.08, 0.67, 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, 0.4, 0.7);

	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, 0.4, 0.7);

	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(1.0, 1.0, 1.0, 0.0);
       glColor3f(0.0f, 0.0f, 0.0f);
       glPointSize(4.0);
       glMatrixMode(GL_PROJECTION);
       glLoadIdentity();
       gluOrtho2D(-640.0, 640.0, -480.0, 480.0);
}

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();
    }
}

void 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();

}
```

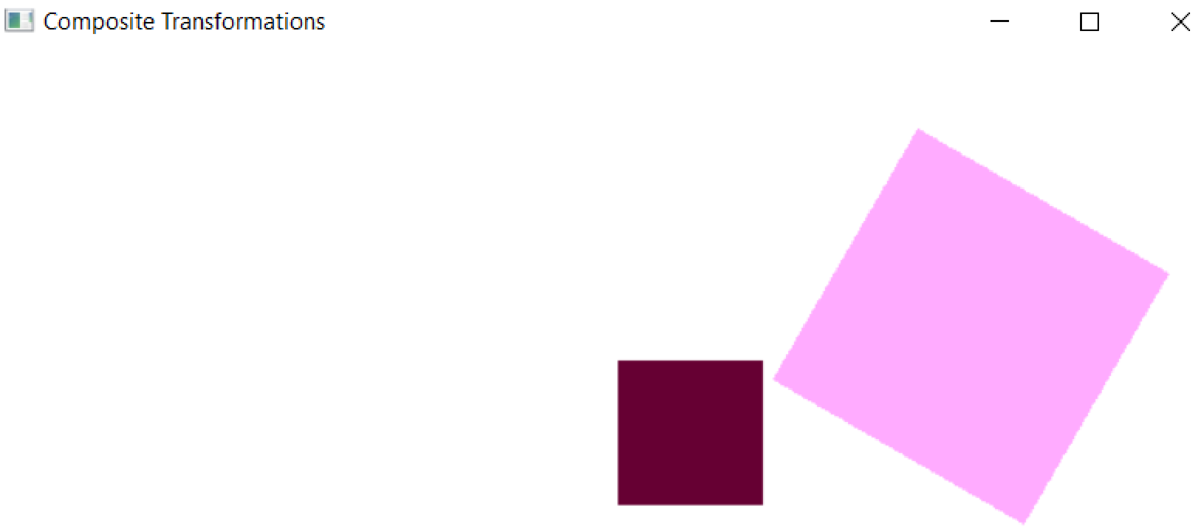
OUTPUT:



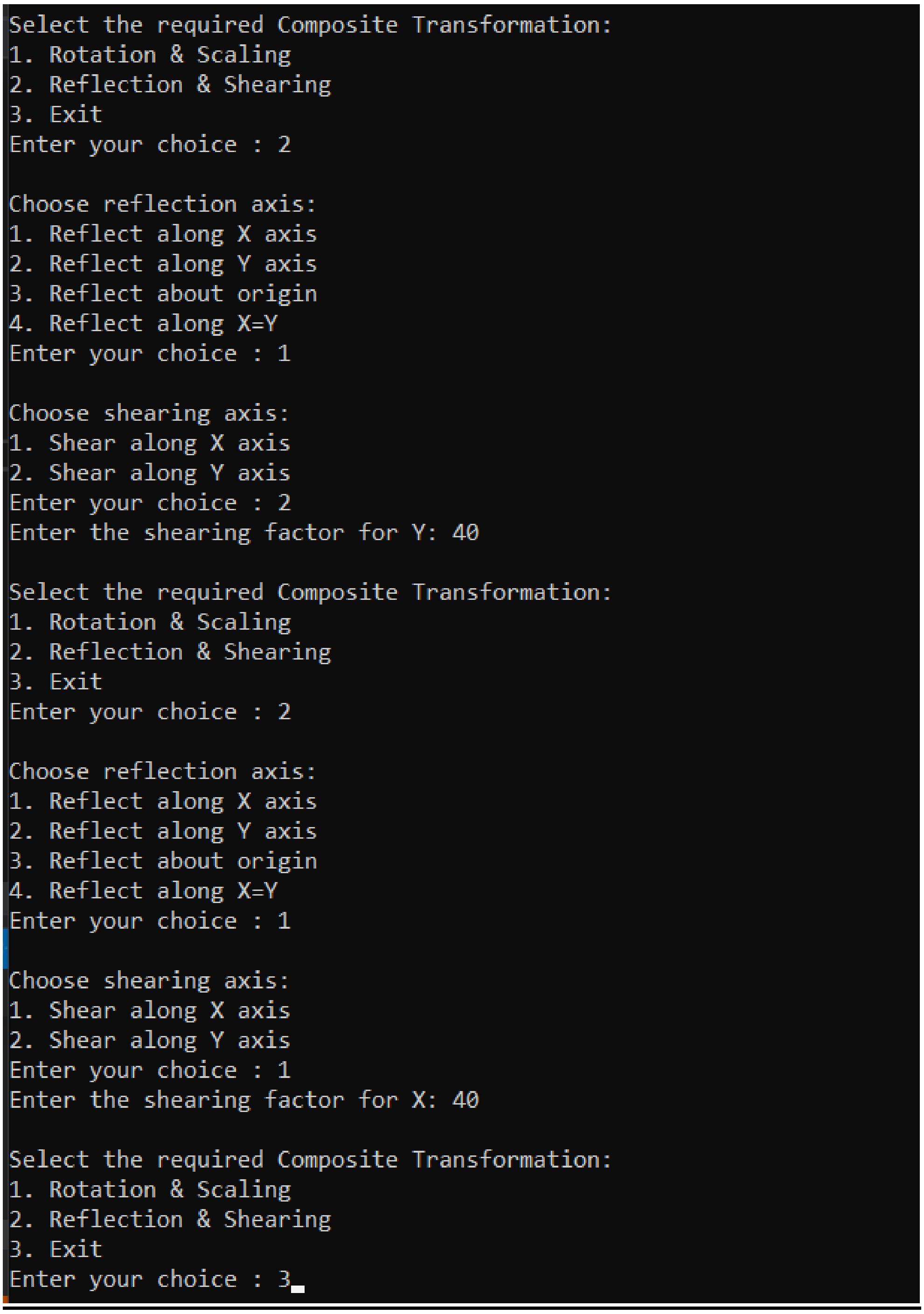
C:\Users\DELL\source\repos\Exer6\Debug\Exer6.exe

```
2D-Transformations


For Polygon:

Enter no of edges: 4

Enter Polygon Coordinates :
Vertex  1 : 10 10
Vertex  2 : 10 160
Vertex  3 : 160 160
Vertex  4 : 160 10

Select the required Composite Transformation:
1. Rotation & Scaling
2. Reflection & Shearing
3. Exit
Enter your choice : 1
Enter the angle for rotation: 60
Enter fixed point: 50 50
Enter the scaling factor for X and Y: 2 2
```

Rotation & Scaling:

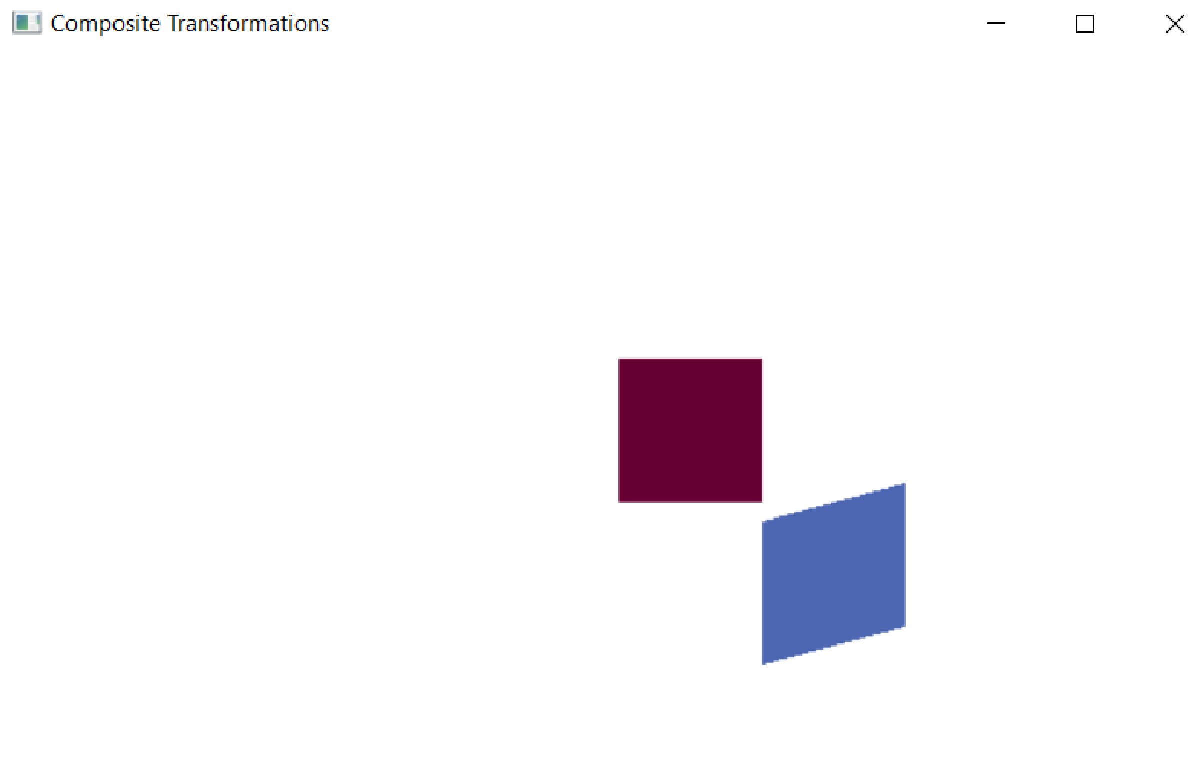


2)Reflection & Shearing:

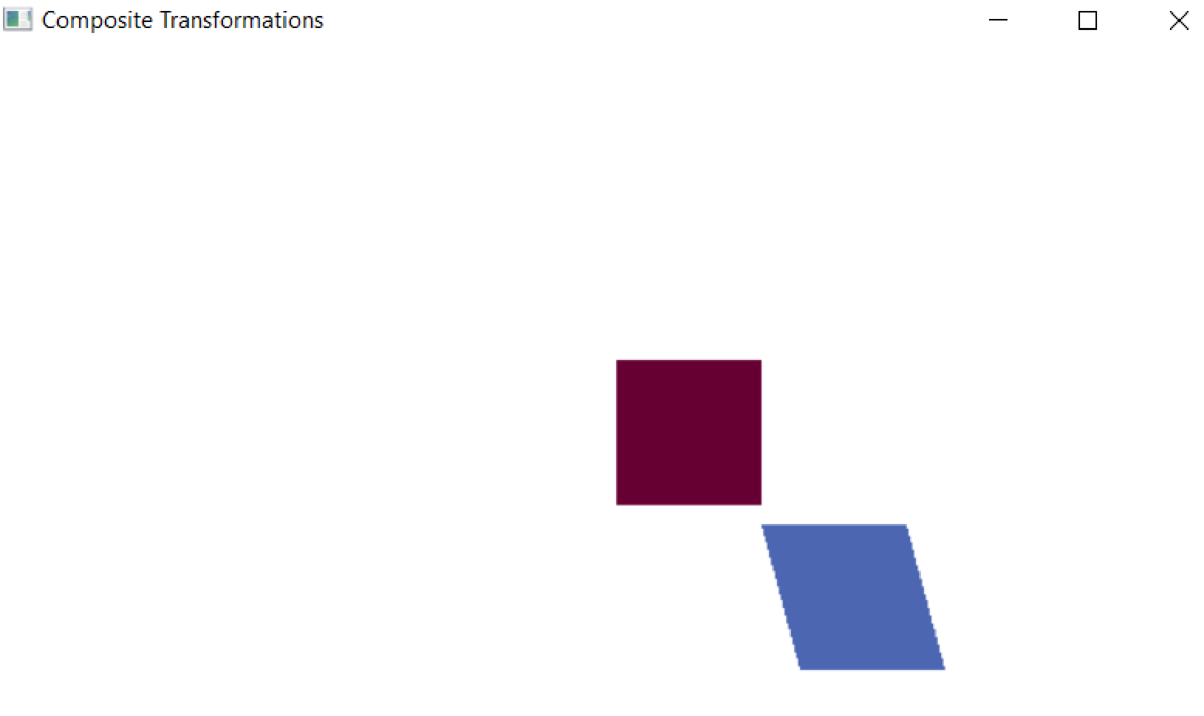


```
Select the required Composite Transformation:
1. Rotation & Scaling
2. Reflection & Shearing
3. Exit
Enter your choice : 2

Choose reflection axis:
1. Reflect along X axis
2. Reflect along Y axis
3. Reflect about origin
4. Reflect along X=Y
Enter your choice : 1

Choose shearing axis:
1. Shear along X axis
2. Shear along Y axis
Enter your choice : 2
Enter the shearing factor for Y: 40

Select the required Composite Transformation:
1. Rotation & Scaling
2. Reflection & Shearing
3. Exit
Enter your choice : 2

Choose reflection axis:
1. Reflect along X axis
2. Reflect along Y axis
3. Reflect about origin
4. Reflect along X=Y
Enter your choice : 1

Choose shearing axis:
1. Shear along X axis
2. Shear along Y axis
Enter your choice : 1
Enter the shearing factor for X: 40

Select the required Composite Transformation:
1. Rotation & Scaling
2. Reflection & Shearing
3. Exit
Enter your choice : 3
```

Shearing Along Y:



Shearing Along X:



Result:

A C++ menu-driven program using OPENGL to perform 2D composite transformations for polygon was written and implemented successfully.