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

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

Name : Likhitha Verma A
Reg : 185001084
Date : 07/09/2021

AIM :

Write a program in C++ using OPENGL to perform the following 2-Dimensional composite transformations - Perform rotation and scaling of an object Input, Perform reflection and shearing of an object Input.

ALGORITHM :

1. Read the no. of edges of the polygon from the user.
2. Read the vertices of the polygon.
3. Plot the original polygon.
4. Read the transformation from the user given the menu
5. If the choice is rotation and scaling
    a. get the rotation axis, pivot point and scaling factor from the user
    b. perform rotation and save it to the polygon
    c. perform scaling
    d. Plot the polygon
6. If the choice is reflection and shearing
    a. get the reflection axis and shearing factor from the user
    b. perform reflection and save it to the polygon
    c. perform shearing
    d. Plot the polygon

CODE :

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

int pntX1, pntY1, op = 0, edges;
vector<int> pntX, tempX;
vector<int> pntY, tempY;
int transX, transY;
double scaleX, scaleY;
double angle, angleRad;
char reflectionAxis;
int shearingX, shearingY;

double round(double d)
{
	return floor(d + 0.5);
}

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


void translate(int x, int y)
{
	for (int i = 0; i < edges; i++)
	{
		pntX[i] += x;
		pntY[i] += y;
	}
}
```

```
void scale(double x, double y)
{
	for (int i = 0; i < edges; i++)
	{
		pntX[i] = round(pntX[i] * x) + 300;
		pntY[i] = round(pntY[i] * y);
	}
}

void rotate(double theta)
{
	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)));
	}
}

void reflection(int option)
{
	//X axis reflection
	if (option == 1)
	{
		for (int i = 0; i < edges; i++)
		{
			pntY[i] *= -1;
		}
	}//Y axis reflection
	else if (option == 2)
	{
		for (int i = 0; i < edges; i++)
		{
			pntX[i] *= -1;
		}
	}//origin reflection
	else if (option == 3) {
		for (int i = 0; i < edges; i++) {
			pntX[i] *= -1;
			pntY[i] *= -1;
		}
	}//Y=X reflection
```

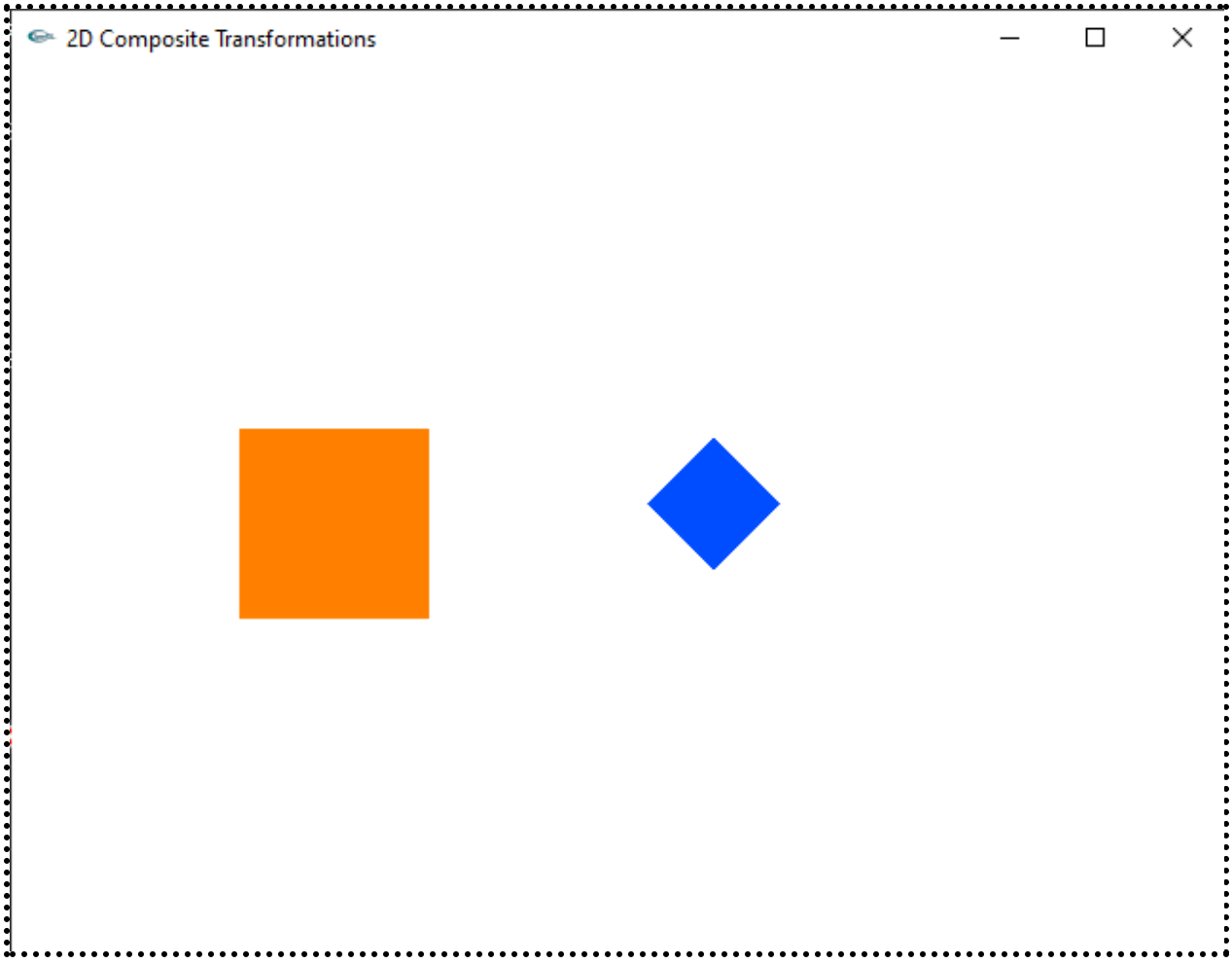
```
	else if (option == 4) {
		for (int i = 0; i < edges; i++) {
			int temp = pntX[i];
			pntX[i] = pntY[i];
			pntY[i] = temp;
		}
	}
}

void shearing(int option)
{
	//translating the tranformed polygon so that it doesn't overlap on the
original polygon
	if (option == 1)
	{
		cout << "Enter the shearing factor for X: "; cin >> shearingX;
		pntX[0] = pntX[0] + 100;
		pntX[1] = pntX[1] + shearingX + 100;
		pntX[2] = pntX[2] + shearingX + 100;
		pntX[3] = pntX[3] + 100;
	}
	else if (option == 2)
	{
		cout << "Enter the shearing factor for Y: "; cin >> shearingY;
		pntX[0] = pntX[0] + 100;
		pntX[1] = pntX[1] + 100;
		pntX[2] = pntX[2] + 100;
		pntX[3] = pntX[3] + 100;

		pntY[2] = pntY[2] + shearingY;
		pntY[3] = pntY[3] + shearingY;
	}
}

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);
		cout << "1. Rotation and scaling\n";
		cout << "2. Reflection and 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 the pivot point X and Y: "; cin >> transX >>
transY;
			cout << "Enter the scaling factor for X and Y: "; cin >> scaleX
>> scaleY;
			glColor3f(1.0, .5, 0.0);
			drawPolygon();
			translate(-transX, -transY);
			rotate(angleRad);
			translate(transX, transY);
			scale(scaleX, scaleY);
			glColor3f(0.0, 0.3, 1.0);
			drawPolygon();
		}
		else if (op == 2)
		{
			glColor3f(1.0, .5, 0.0);
			drawPolygon();
			cout << "1. Reflection (X axis)\n";
			cout << "2. Reflection (Y axis)\n";
			cout << "3. Reflection (Origin)\n";
			cout << "4. Reflection (Y = X)\n";
```

```
                cout << "Enter choice of reflection axis : ";
                cin >> op;
                reflection(op);
                cout << "1. Shearing (X axis) \n";
                cout << "2. Shearing (Y axis) \n";
                cout << "Enter choice of shearing  axis : ";
                int option;
                cin >> option;

                shearing(option);
                glColor3f(0.0, 0.3, 1.0);
                drawPolygon();
            }
            pntX = tempX;
            pntY = tempY;
            glFlush();
      }

}

void main(int argc, char** argv)
{
      cout << "For 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("2D Composite Transformations");
      glutDisplayFunc(myDisplay);
      myInit();
      glutMainLoop();
}
```

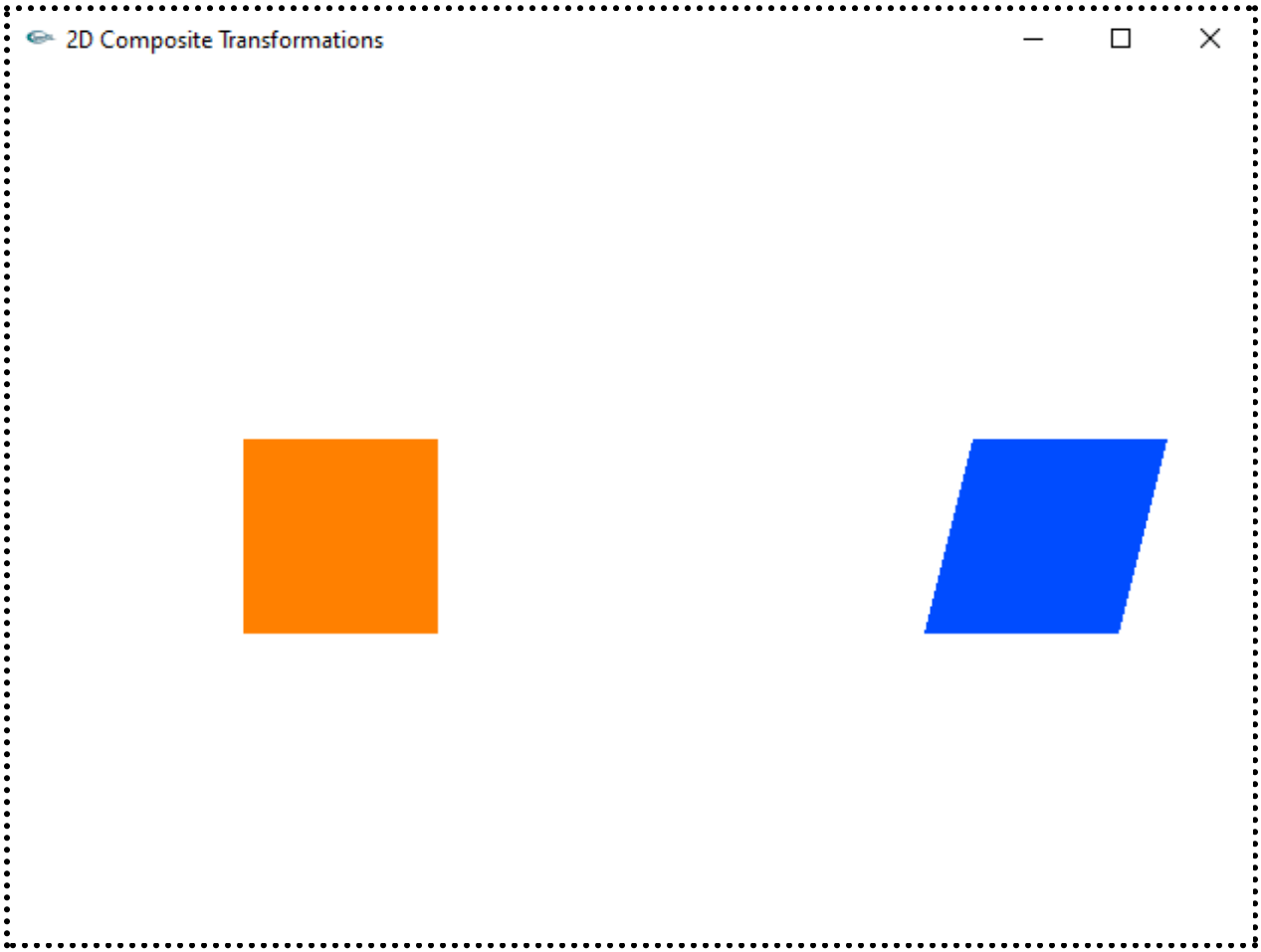
OUTPUT : i) Rotation and scaling

```
For Polygon:

Enter no of edges: 4

Enter Polygon Coordinates :
Vertex  1 : -400 -100
Vertex  2 : -400 100
Vertex  3 : -200 100
Vertex  4 : -200 -100
1. Rotation and scaling
2. Reflection and shearing
3. Exit
Enter your choice : 1
Enter the angle for rotation: 45
Enter the pivot point X and Y: -400 -100
Enter the scaling factor for X and Y: 0.5 0.5
```



ii) Reflection and shearing :

```
1. Rotation and scaling
2. Reflection and shearing
3. Exit
Enter your choice : 2
1. Reflection (X axis)
2. Reflection (Y axis)
3. Reflection (Origin)
4. Reflection (Y = X)
Enter choice of reflection axis : 2
1. Shearing (X axis)
2. Shearing (Y axis)
Enter choice of shearing  axis : 1
Enter the shearing factor for X: 50
```



RESULT :

Thus compiled and executed a C++ menu-driven program using OPENGL to perform 2D composite transformations successfully.