Name:	MAHEK SHAH
Roll No:	
Class/Sem:	SE/III
Experiment No.:	6
Title:	Implement 2D Transformation on an object.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No. 6

Aim :Write a program to implement 2D Transformation on an object -Translation, Rotation, Reflection, Scaling and Shear in C.

Objective :

Description: We have to perform 2D transformations on 2D objects. Here we perform transformations on a line segment. The 2D transformations are:

1. Translation
2. Scaling
3. Rotation
4. Reflection
5. Shear

1. Translation: Translation is defined as moving the object from one position to another position along straight line path.

We can move the objects based on translation distances along x and y axis. tx denotes translation distance along x-axis and ty denotes translation distance along y axis.

Translation Distance: It is nothing but by how much units we should shift the object from one location to another along x, y-axis.

Consider (x,y) are old coordinates of a point. Then the new coordinatesof that same point (x',y') can be obtained as follows:

$$X' = x + tx$$
$$Y' = y + ty$$

2. **Scaling**: scaling refers to changing the size of the object either by increasing or decreasing. We will increase or decrease the size of the object based on scaling factors along x and y-axis.

If (x, y) are old coordinates of object, then new coordinates of object after applying scaling transformation are obtained as:

x'=x*sx y'=y*sy.

sx and sy are scaling factors along x-axis and y-axis.

3. Rotation: A rotation repositions all points in an object along a circular path in

the plane centered at the pivot point. We rotate an objectby an angle theta.
New coordinates after rotation depend on *both x* and *y* • x'=
$x\cos\theta - y\sin\theta$
• $y' = x\sin\theta + y\cos\theta$

4. Reflection: Reflection is nothing but producing mirror image of an object. Reflection can be done just by rotating the object about given axis of reflection with an angle of 180 degrees.

5. Shear:
1. Shear is the translation along an axis by an amount that increases linearly with another axis (**Y**). It produces shape distortions as if objects were composed of layers that are caused to slide over each other.
2. Shear transformations are very useful in creating italic letters and slanted letters from regular letters.
3. Shear transformation changes the shape of the object to a slant position.
4. Shear transformation is of 2 types:
a. X-shear: changing x-coordinate value and keeping y constant $x'=x+shx*y$
$y'=y$
b. Y-shear: changing y coordinates value and keeping x constant $x'=x$
$y'=y+shy*x$ shx and shy are shear factors along x and y-axis.

Algorithm :

1. Start

2. Read the coordinate values to draw the 2D figure and perform the following steps to achieve the necessary transformations
3. Translation : Read the translation vector tx and ty and compute the new position using x1 = x + tx , y1 = y + ty and display the new position
4. Rotation : Read the rotation angle Θ and compute the new pos using the formula x1 = x Cos Θ – y Sin Θ , y1 = x Sin Θ + y Cos Θ and display the new position

5. Scaling : Read the scaling factor sx and sy and compute the newposition using x1 = x. sx , y1 = y. sy and display the new position

6. Shearing : Shearing is done in x direction and y direction using the factor shx and shy for X direction: x1 = x + shx (y – yref), y1 = y & forY direction x1 = x , y1 = shy (x – xref)

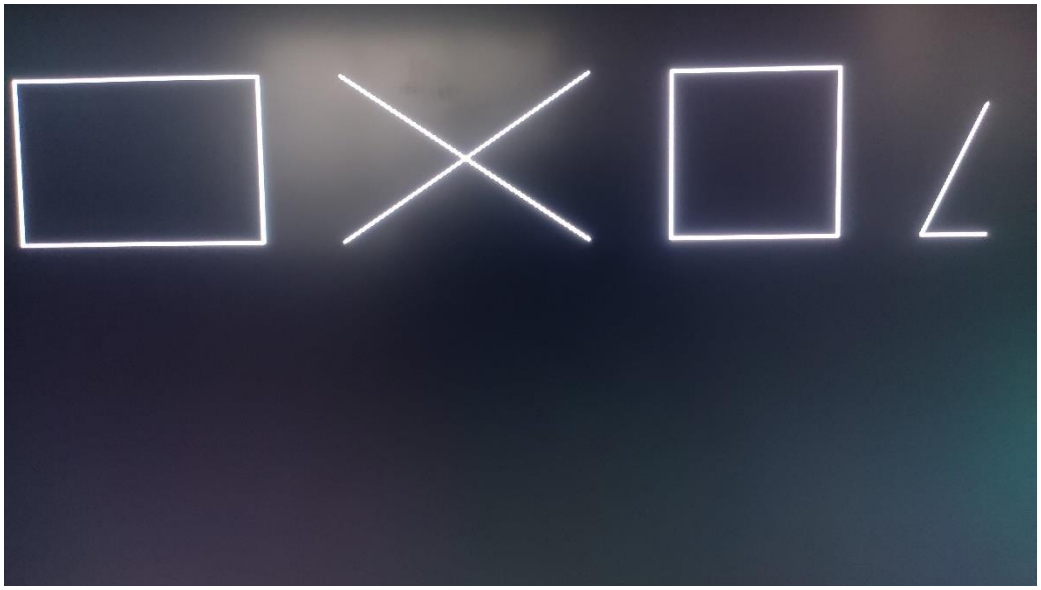
7. Reflection : Reflection depends upon the reflection axis . If reflection axis is y = x , then x1 = y and y1 = x
8. Stop

Code :

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void drawObject(int x[], int y[], int edges) {
    for (int i = 0; i < edges - 1; i++) {
        line(x[i], y[i], x[i + 1], y[i + 1]);
    }
    line(x[edges - 1], y[edges - 1], x[0], y[0]);
}
void translate(int x[], int y[], int edges, int tx, int ty) {
    for (int i = 0; i < edges; i++) {
        x[i] = x[i] + tx;
        y[i] = y[i] + ty;
    }
}
void rotate(int x[], int y[], int edges, float angle) {
    float radian = angle * (M_PI / 180.0);
    int cx = 0, cy = 0;
    for (int i = 0; i < edges; i++) {
        int tempX = x[i];
        x[i] = cx + (x[i] - cx) * cos(radian) - (y[i] - cy) * sin(radian);
        y[i] = cy + (tempX - cx) * sin(radian) + (y[i] - cy) * cos(radian);
    }
}
void reflect(int x[], int y[], int edges, int reflectionAxis) {
    for (int i = 0; i < edges; i++) {
        if (reflectionAxis == 0)
            y[i] = -y[i];
        else if (reflectionAxis == 1)
            x[i] = -x[i];
    }
}
void scale(int x[], int y[], int edges, float sx, float sy) {
    for (int i = 0; i < edges; i++) {
        x[i] = round(x[i] * sx);
        y[i] = round(y[i] * sy);
    }
}
```

```
void shear(int x[], int y[], int edges, float shx, float shy) {
    for (int i = 0; i < edges; i++) {
        x[i] = round(x[i] + shx * y[i]);
        y[i] = round(y[i] + shy * x[i]);
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int x[] = {50, 100, 100, 50, 50};
    int y[] = {50, 50, 100, 100, 50};
    int edges = sizeof(x) / sizeof(x[0]);
    drawObject(x, y, edges);
    translate(x, y, edges, 50, 50);
    drawObject(x, y, edges);
    rotate(x, y, edges, 45);
    drawObject(x, y, edges);
    reflect(x, y, edges, 1);
    drawObject(x, y, edges);
    scale(x, y, edges, 1.5, 1.5);
    drawObject(x, y, edges);
    shear(x, y, edges, 0.5, 0);
    drawObject(x, y, edges);
    getch();
    closegraph();
    return 0;
}
```

Output:



Conclusion:

The 2D transformation algorithms implemented in this program provide a foundation for manipulating objects in computer graphics. Each transformation serves a specific purpose:

- **Translation:** Moves the object by a specified distance in the x and y directions.
- **Rotation:** Rotates the object by a given angle around the origin.
- **Reflection:** Mirrors the object with respect to the X-axis or Y-axis.
- **Scaling:** Enlarges or shrinks the object by scaling its coordinates.
- **Shearing:** Skews the object by displacing its points along one of the axes.

These transformations are essential for creating dynamic and visually appealing graphics in applications such as computer-aided design, gaming, and simulations. While the provided program demonstrates the transformations individually, combining multiple transformations can produce more complex effects. Understanding and implementing 2D transformations are fundamental skills in computer graphics and are often extended to 3D transformations in more advanced graphics applications.