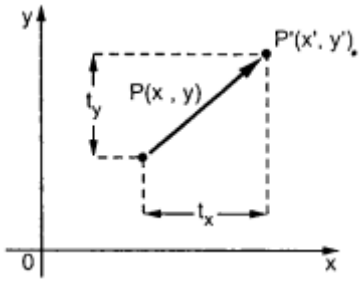
Experiment No. 6
Topic: To implement 2D Transformations
Name: Vinith Shetty
Roll Number: 55
Date of Performance:
Date of Submission:

Experiment No. 6

Aim: To implement 2D Transformations: Translation, Scaling, Rotation.

Objective:
To understand the concept of transformation, identify the process of transformation and application of these methods to different object and noting the difference between these transformations.

Theory:
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.



Consider (x,y) are old coordinates of a point. Then the new coordinates of that same point (x',y') can be obtained as follows:
x' = x + tx
y' = y + ty
We denote translation transformation as P. we express above equations in matrix form as:
P' = P + T , where

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

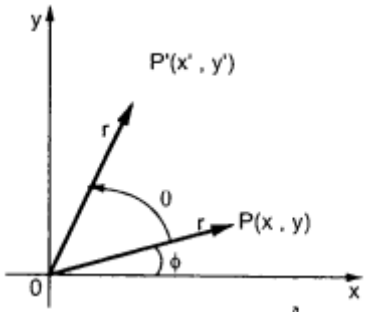
Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,gm;
int x1,y1,x2,y2,tx,ty,x3,y3,x4,y4;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter the starting point of line segment:");
scanf("%d %d",&x1,&y1);
printf("Enter the ending point of line segment:");
scanf("%d %d",&x2,&y2);
printf("Enter translation distances tx,ty:\n");
scanf("%d%d",&tx,&ty);
setcolor(5);
line(x1,y1,x2,y2);
outtextxy(x2+2,y2+2,"Original line");
x3=x1+tx;
y3=y1+ty;
x4=x2+tx;
y4=y2+ty;
setcolor(7);
line(x3,y3,x4,y4);
outtextxy(x4+2,y4+2,"Line after translation");
getch();
}
```

Output –

2) Rotation –

A rotation repositions all points in an object along a circular path in the plane centered at the pivot point. We rotate an object by an angle theta. New coordinates after rotation depend on both x and y.



$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

The above equations can be represented in the matrix form as given below

$$[x'\ y'] = [x\ \ y]\begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}$$

$$P' = P \cdot R$$

where R is the rotation matrix and it is given as

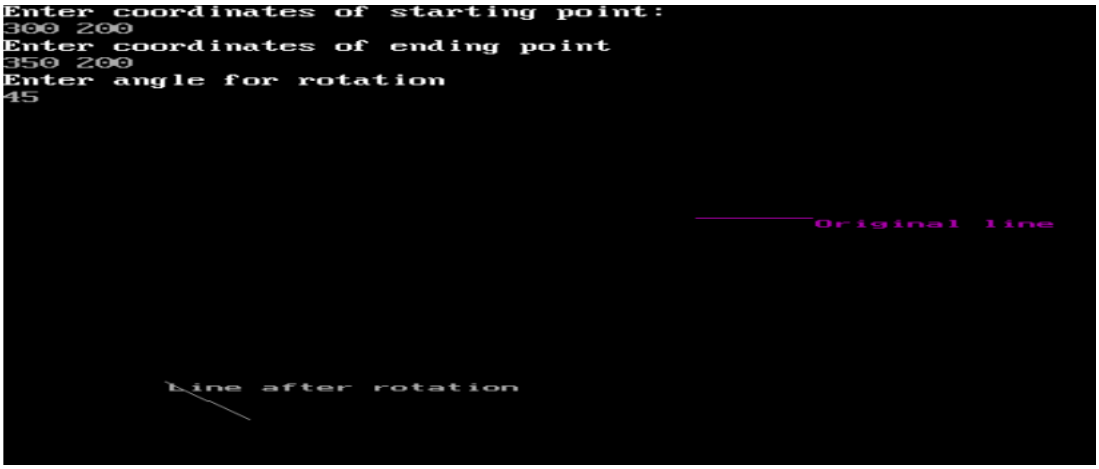
$$R = \begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}$$

Program:

```
#include<stdio.h>
#include<conio.h>
```

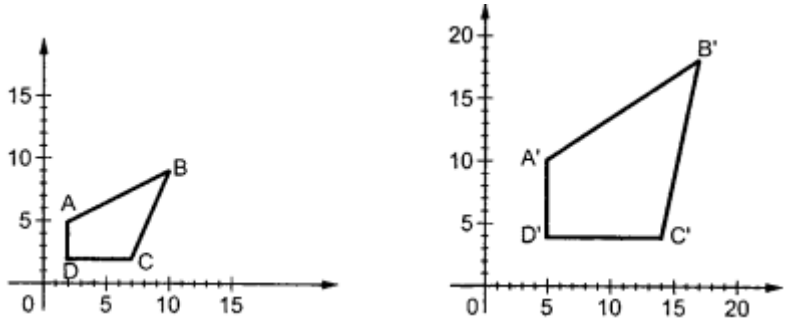
```
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,gm;
float x1,y1,x2,y2,x3,y3,x4,y4,a,t;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter coordinates of starting point:\n");
scanf("%f%f",&x1,&y1);
printf("Enter coordinates of ending point\n");
scanf("%f%f",&x2,&y2);
printf("Enter angle for rotation\n");
scanf("%f",&a);
setcolor(5);
line(x1,y1,x2,y2);
outtextxy(x2+2,y2+2,"Original line");
t=a*(3.14/180);
x3=(x1*cos(t))-(y1*sin(t));
y3=(x1*sin(t))+(y1*cos(t));
x4=(x2*cos(t))-(y2*sin(t));
y4=(x2*sin(t))+(y2*cos(t));
setcolor(7);
line(x3,y3,x4,y4);
outtextxy(x3+2,y3+2,"Line after rotation");
getch();
```

Output:



3) 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. we express the above equations in matrix form as:

$$[x' \; y'] = [x \; y] \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$= [x \cdot S_x \quad y \cdot Sy]$$

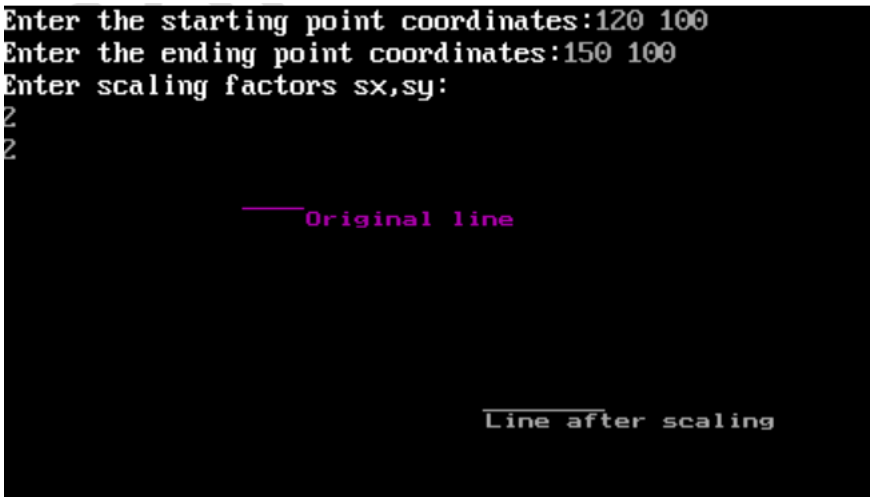
$$= P \cdot S$$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,gm;
float x1,y1,x2,y2,sx,sy,x3,y3,x4,y4;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter the starting point coordinates:");
scanf("%f %f",&x1,&y1);
printf("Enter the ending point coordinates:");
scanf("%f %f",&x2,&y2);
printf("Enter scaling factors sx,sy:\n");
scanf("%f%f",&sx,&sy);
setcolor(5);
line(x1,y1,x2,y2);
outtextxy(x2+2,y2+2,"Original line");
```

```
x3=x1*sx;
y3=y1*sy;
x4=x2*sx;
y4=y2*sy;
setcolor(7);
line(x3,y3,x4,y4);
outtextxy(x3+2,y3+2,"Line after scaling");
getch();
}
```

Output –



Conclusion: Comment on :

1. Application of transformation

2. Difference noted between methods
3. Application on different object

Application of Transformation: In 2D Transformations, transformations are fundamental operations applied to shapes, objects, or images to alter their position, orientation, size, or appearance. It finds applications in various domains, facilitating the manipulation, positioning, and alteration of shapes, objects, images, and elements within a 2D space for diverse purposes ranging from design and animation to engineering and image processing.

Difference Noted between Translation, Rotation, Scaling: Translation moves objects without changing their size or orientation, rotation changes the orientation of an object around a fixed point, and scaling modifies the size of an object while retaining its shape and orientation. These transformations serve different purposes in manipulating objects within computer graphics applications, enabling various visual effects, animations, and positional changes in designs and interfaces.

Application of Translation, Rotation, Scaling on Different Objects: The application of translation, rotation, and scaling on different objects is crucial in computer graphics for various purposes, including design, animation, and image manipulation. In each case, these transformations (translation, rotation, and scaling) serve specific purposes on different objects, allowing for movement, repositioning, orientation changes, size adjustments, and various visual effects, catering to the requirements of different applications within computer graphics, design, animation, and simulation.