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

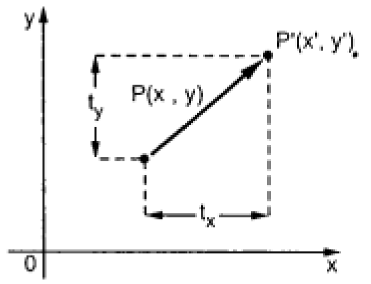
Objective:

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

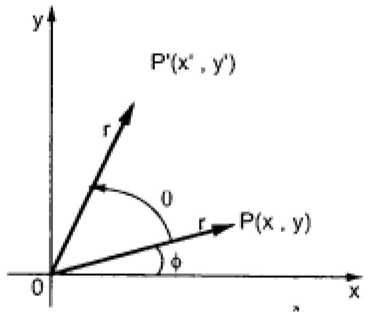
We denote translation transformation as P. we express above equations in matrix form as:

P' = P + T , where

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

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

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

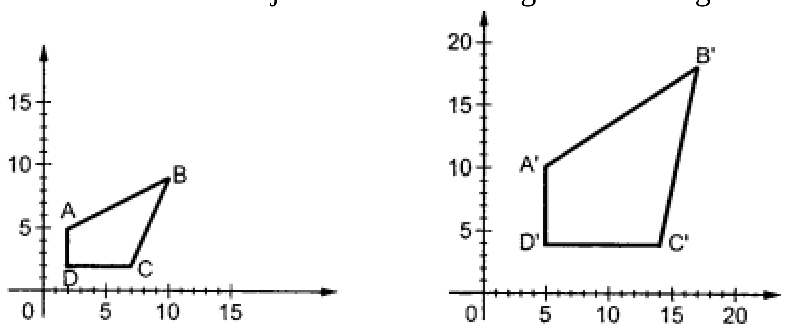
$$P' = P \cdot R$$

where R is the rotation matrix and it is given as

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

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:

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

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

$$= P \cdot S$$

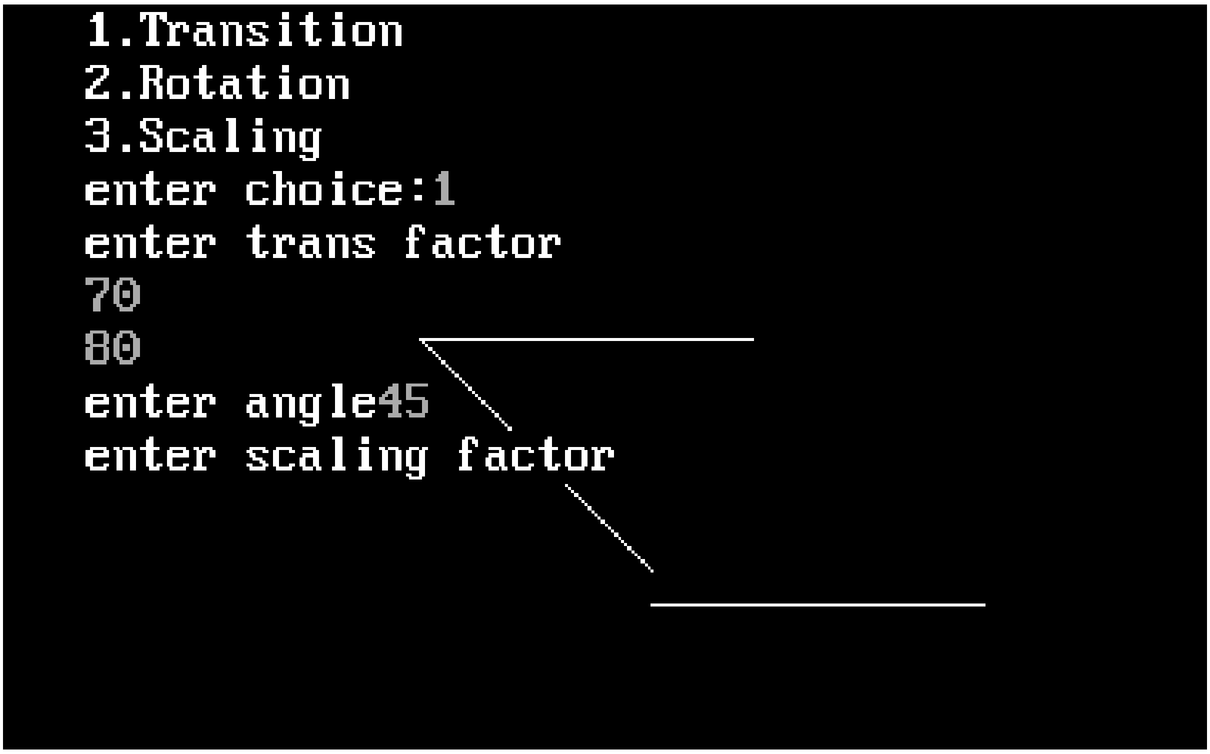
Program:

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

void main()
{
int gd =DETECT,gm,ch,sx,sy,tx,ty,nx1,nx2,ny1,ny2;
double r,t;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
line(100,100,200,100);
printf("1.Transition\n2.Rotation\n3.Scaling\n");
printf("enter choice:");
scanf("%d",&ch);
switch(ch)
{
case 1 : printf("enter trans factor \n");
         scanf("%d%d",&tx,&ty);
         nx1=100+tx;
         ny1=100+ty;
         nx2=200+tx;
         ny2=100+ty;
         line(nx1,ny1,nx2,ny2);
         getch();
case 2 : printf("enter angle");
         scanf("%lf",&r);
         t=(3.14*r)/180;
         nx1=(int)(100+(200-100)*cos(t)-(100-100)*sin(t));
         ny1=(int)(100+(200-100)*sin(t)+(100-100)*cos(t));
         line(100,100,nx1,ny1);
         getch();
case 3 : printf("enter scaling factor \n");
         scanf("%d%d",&sx,&sy);
         nx1=100*sx;
         ny1=100*sy;
         nx2=200*sx;
         ny2=100*sy;
         line(nx1,ny1,nx2,ny2);
         getch();
default : printf("invalid\n");
```

```
}
getch();
closegraph();
}
```

Output –



Conclusion: Comment on :

1. Application of transformation:-To achieve more complex effects.
2. Difference noted between methods:-In translation size of object remains same it is just translated. In rotation object is rotated. In scaling size of object gets enlarged or reduced.