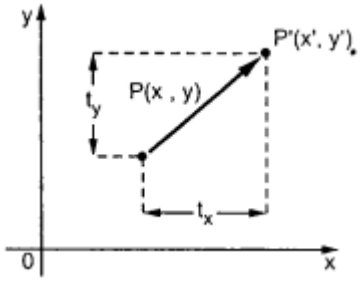
Experiment No. 6

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

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

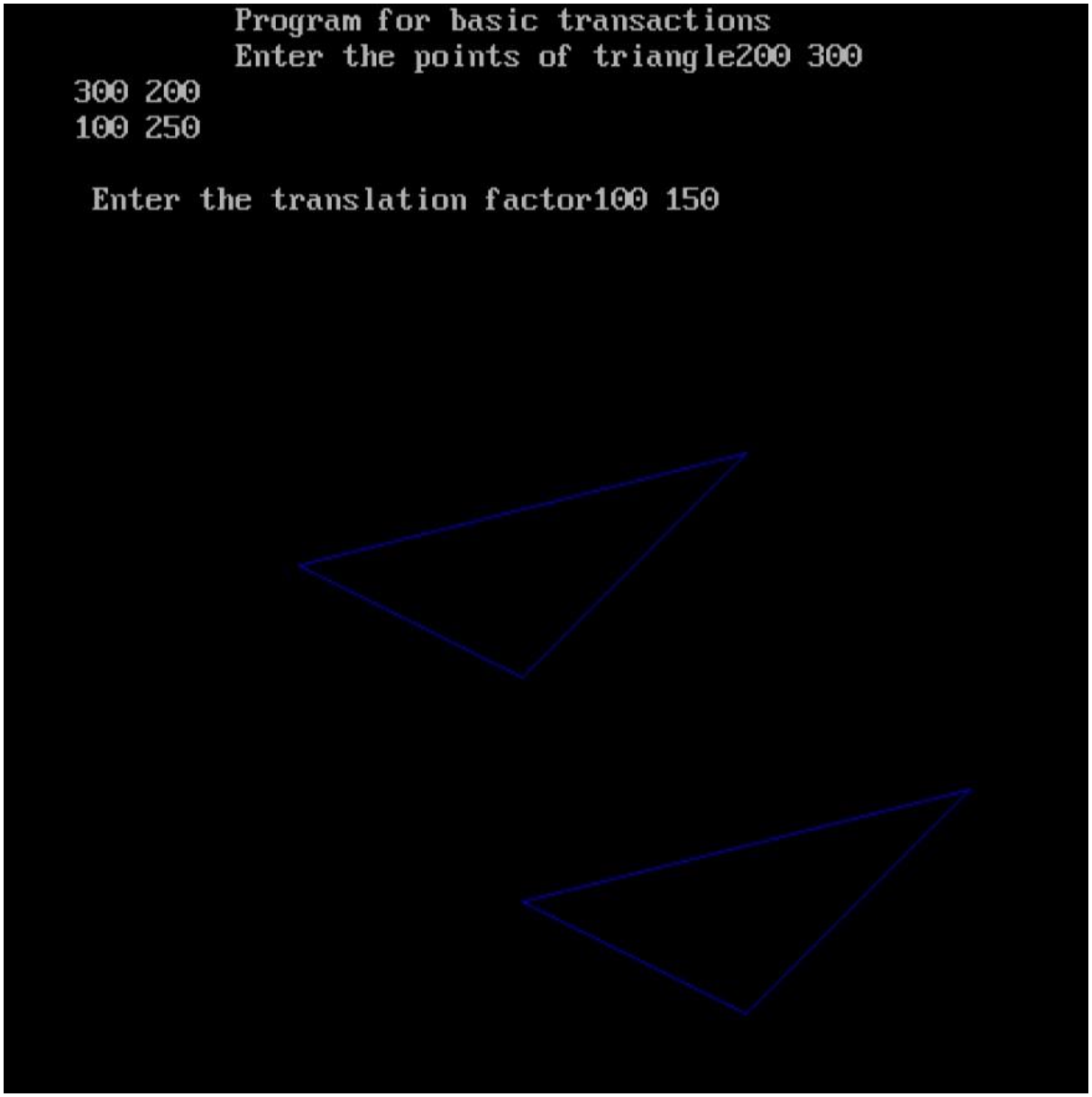
Program:

```
#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include<math.h>

int main()
```

```
{
        int gm;
        int gd=DETECT;
        int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;
        int sx,sy,xt,yt,r;
        float t;
        initgraph(&gd,&gm," ");
        printf("\t Program for basic transactions");
        printf("\n\t Enter the points of triangle");
        setcolor(1);
        scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
        printf("\n Enter the translation factor");
        scanf("%d%d",&xt,&yt);
         nx1=x1+xt;
         ny1=y1+yt;
         nx2=x2+xt;
         ny2=y2+yt;
         nx3=x3+xt;
         ny3=y3+yt;
         line(nx1,ny1,nx2,ny2);
         line(nx2,ny2,nx3,ny3);
         line(nx3,ny3,nx1,ny1);
getch();
closegraph();
```

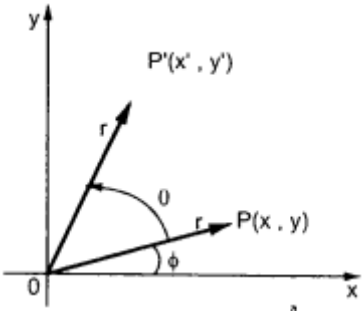
}

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

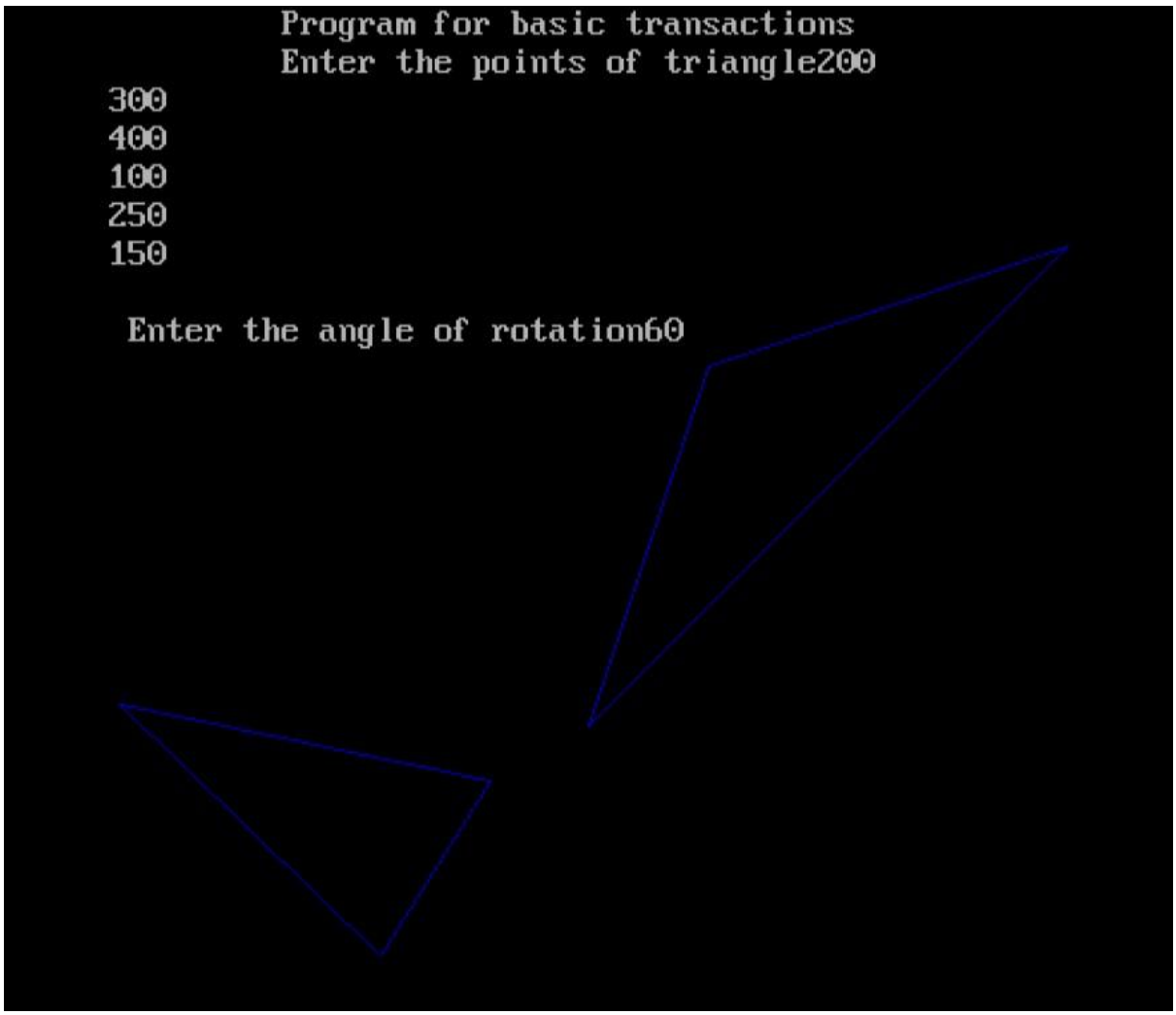
$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Program:

```
#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include<math.h>

int main()

{

        int gm;

        int gd=DETECT;

        int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;

        int sx,sy,xt,yt,r;
```

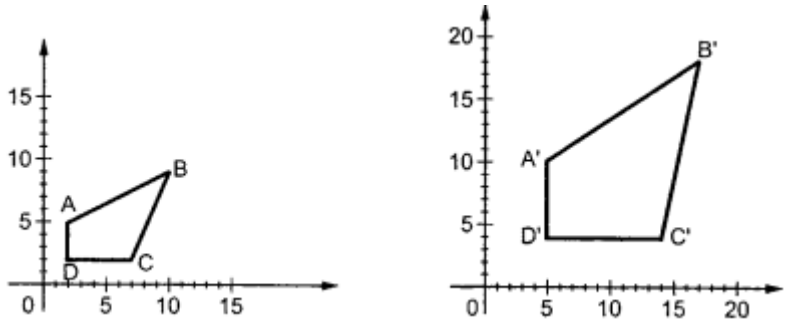
```
        float t;
        initgraph(&gd,&gm," ");
        printf("\t Program for basic transactions");
        printf("\n\t Enter the points of triangle");
        setcolor(1);
        scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
 printf("\n Enter the angle of rotation");
                        scanf("%d",&r);
                        t=3.14*r/180;
                        nx1=abs(x1*cos(t)-y1*sin(t));
                        ny1=abs(x1*sin(t)+y1*cos(t));
                        nx2=abs(x2*cos(t)-y2*sin(t));
                        ny2=abs(x2*sin(t)+y2*cos(t));
                        nx3=abs(x3*cos(t)-y3*sin(t));
                        ny3=abs(x3*sin(t)+y3*cos(t));
                        line(nx1,ny1,nx2,ny2);
                        line(nx2,ny2,nx3,ny3);
                        line(nx3,ny3,nx1,ny1);
                        getch();
  closegraph();
return 0;
}
```

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:

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

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

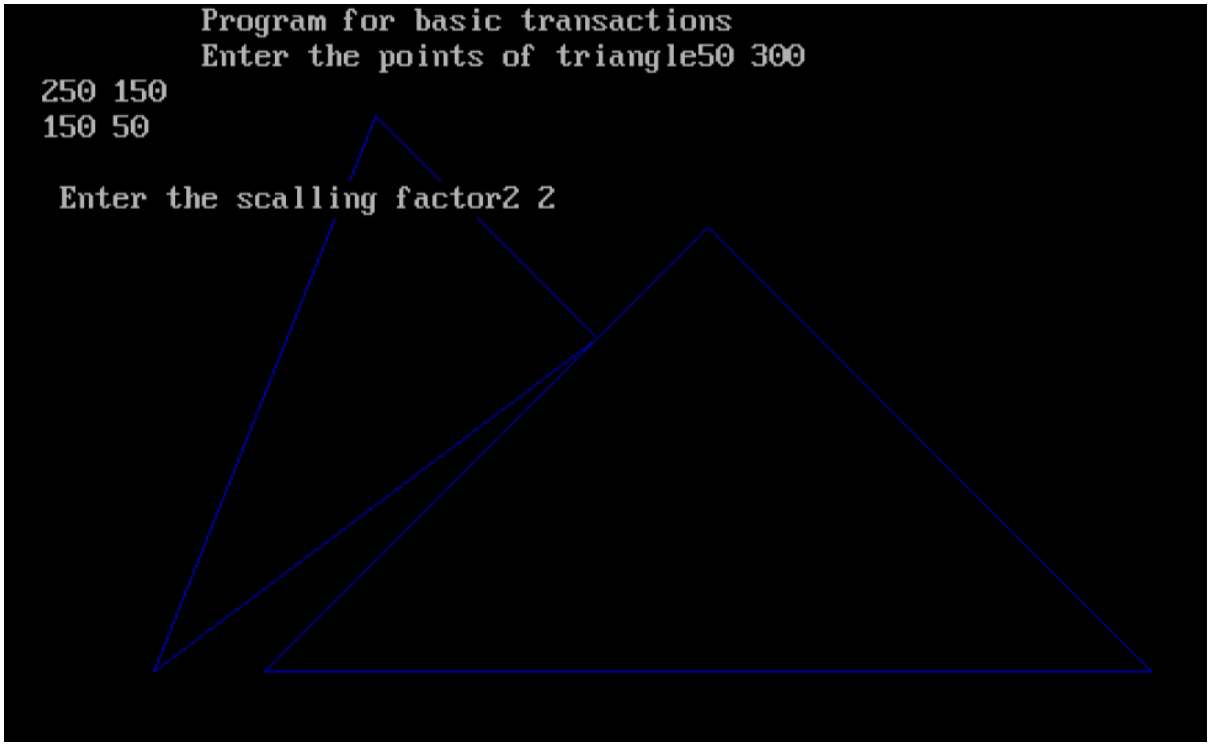
$$= P \cdot S$$

Program:

```
#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include<math.h>

int main()

{

        int gm;

        int gd=DETECT;

        int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3,c;

        int sx,sy,xt,yt,r;

        float t;
```

```
        initgraph(&gd,&gm," ");
        printf("\t Program for basic transactions");
        printf("\n\t Enter the points of triangle");
        setcolor(1);
        scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        line(x3,y3,x1,y1);
printf("\n Enter the scalling factor");
                        scanf("%d%d",&sx,&sy);
                        nx1=x1*sx;
                        ny1=y2*sy;
                        nx2=x2*sx;
                        ny2=y2*sy;
                        nx3=x3*sx;
                        ny3=y3*sy;
                        line(nx1,ny1,nx2,ny2);
                        line(nx2,ny2,nx3,ny3);
                        line(nx3,ny3,nx1,ny1);
                        getch();
closegraph();
}
```

Output –



Conclusion: Comment on :

1. Application of transformation
2. Difference noted between methods
3. Application t different object