**Experiment No.: 1**

Name of Experiment: **Line, Circle and Ellipse attributes**

Aim: Implementation of Line, Circle and Ellipse attributes.

**Draw Line, Circle and Ellipse**

```
#include <graphics.h>
#include <conio.h>  // For getch()

void drawLine() {
   // Drawing a line from point (100, 150) to (300, 150)
   line(100, 150, 300, 150);
}

void drawCircle() {
   // Drawing a circle with center (200, 200) and radius 50
   circle(200, 200, 50);
}

void drawEllipse() {
   // Drawing an ellipse with center (200, 200), x-radius 100, y-radius 50
   ellipse(200, 200, 0, 360, 100, 50);
}

int main() {
   // Initialize the graphics mode and driver
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

   // Set a background color (optional)
   setbkcolor(WHITE);
   cleardevice();

   // Set line, circle, and ellipse color
   setcolor(BLUE);
```

```
    // Draw shapes
    drawLine();
    drawCircle();
    drawEllipse();

    // Hold the output screen
    getch();

    // Close the graphics mode
    closegraph();
    return 0;
}
```

**Experiment No.: 2**

Name of Experiment: **Pre Defined Patterns**

Aim: To produce a pixel and pre specified patterns on screen. (Same to all groups)

Algorithm: (Step wise):

1. Draw a rectangle using predefined functions from (0, 0) to (getmaxx(), getmaxy())
2. Draw two lines which partitioned the above rectangle into 4 quadrants.
3. Find the centre of each quadrant, and glow them with different colors.
4. Draw the following predefined patterns using colors as follows
5. Draw a circle in 1st quadrant
6. Draw a square in 2nd quadrant
7. Draw an ellipse in 3rd quadrant
8. Draw a triangle in 4th quadrant

**Draw A Pixel**

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

void main()
{
   int gd = DETECT, gm, color;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   putpixel(85, 35, GREEN);
   putpixel(30, 40, RED);
   putpixel(115, 50, YELLOW);
   putpixel(135, 50, CYAN);
   putpixel(45, 60, BLUE);
   putpixel(20, 100, WHITE);
   putpixel(200, 100, LIGHTBLUE);
   putpixel(150, 100, LIGHTGREEN);
   putpixel(200, 50, YELLOW);
   putpixel(120, 70, RED);
   getch();
   closegraph();
}
```

Viva Questions:

1. What is the function for drawing a line?

2. How to find the centre of 3rd quadrant?
3. How to get the resolution of your screen?
4. What is the use of initgraph() function.
5. What is the use of closegraph() function?
6. Why restorecrtmode() function is used?
7. How to change the color of your background screen?
8. Name the arguments that are to be passed to ellipse function.
9. How to get the coordinates of the bottom-right pixel on your screen?
10. Name the function used to put a pixel on your screen.

**Experiment No.: 3**

Name of Experiment**: <u>Digital Differential Analyzer (DDA) Algorithm.</u>**

Aim: SHG1 & 5: Draw a line having (X1<X2) using DDA Algorithm having slope < than 1.

SHG2 & 6: Draw a line having (X1<X2) using DDA Algorithm having slope > than 1.

SHG3: Draw a line having (X1>X2) using DDA Algorithm having slope < than 1.

SHG4: Draw a line having (X1>X2) using DDA Algorithm having slope > than 1.

Algorithm: (Step wise):

1. Accept two end points as (x1, y1) and (x2, y2)
2. Find dx = x2 – x1 and dy = y2 – y1 and m = abs (dy/dx).
3. The deference with the greater magnitude determines the value of the parameter size.
   If abs (dx) > abs (dy) then size = abs(dx).
   Otherwise the size = abs (dy).
4. Start from the pixel (x1, y1) and determine increment or decrement which is needed to generate the next pixel at each step.
5. loop the following process for size times
   a. if (x1<x2) (line is to be drawn from left to right)
      if ( m <1) then Xinc = 1 and Yinc = m
      else Xinc = 1/m and Yinc = 1
   b. else (Line is to be drawn from right to left)
      if ( m >1) then Xinc = -1 and Yinc = -m
      else Xinc = -1/m and Yinc = -1

Test Data: (x1, y1) = (2, 7) and (x2, y2) = (15, 10)

Results for different data sets:

dx = 13 and dy = 3

size = 13

m = abs (3/13) = 0.2

so Xinc = 1 and Yinc = m = 0.2

| Step | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Y | 7.2 | 7.4 | 7.6 | 7.8 | 8 | 8.2 | 8.4 | 8.6 | 8.8 | 9 | 9.2 | 9.4 | 9.6 |
| (x,y) | (3,7) | (4,7) | (5,8) | (6,8) | (7,8) | (8,8) | (9,8) | (10,9) | (11,9) | (12,9) | (13,9) | (14,9) | (15,10) |

rogram:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
   intgd = DETECT, gm, i;
   float x, y, dx, dy, steps;
   int x0, x1, y0, y1;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setbkcolor(WHITE);
   x0 = 100 , y0 = 200, x1 = 500, y1 = 300;
   dx = (float)(x1 - x0);
   dy = (float)(y1 - y0);
   if(dx>=dy)
        {
      steps = dx;
   }
   else
        {
      steps = dy;
   }
   dx = dx/steps;
   dy = dy/steps;
   x = x0;
   y = y0;
   i = 1;
```

```
    while(i<= steps)
    {
       putpixel(x, y, RED);
       x += dx;
       y += dy;
       i=i+1;
    }
    getch();
    closegraph();
}
```

Viva Questions:

1. What does DDA stands for?
2. How do you find the slope of a line?
3. What is the difference between floor() and ceil() functions?
4. In which header file abs () function is found?
5. How we can round off any value?
6. What is the equation of line having end points (x1, y1) and (x2, y2)?

**Experiment No.: 4**

Name of Experiment**: Bresenham's Line Drawing Algorithm**

Aim: **SHG 1, 3, 5:** Line of slope $|m| < 1$

**SHG 2, 4, 6**: Line of slope $|m| >= 1$

Algorithm: (Step wise):

1. Insert two I/P line end points and store the left end point in $(X_o, Y_o)$.
2. Plot the first point. (Load the point into the frame buffer)
3. Calculate constants $\Delta x$, $\Delta y$, $2\Delta x$, $2\Delta y$ and obtain the starting value for the decision parameter as

   $P_o = 2\Delta y - \Delta x$
4. At each $X_k$ along the line, starting at K=0 perform the following test:
   a. if $P_k<0$ the next point to plot is $(X_K + 1, Y_k)$ and

      $P_{k+1} = P_k + 2\Delta y$

   b. Otherwise the next point to plot is $(X_K + 1, Y_k+1)$ and

      $P_{k+1} = P_k + 2\Delta y - 2\Delta x$
5. Repeat the step 4 for $\Delta x$ times.

Test Data: (x1, y1) = (20, 10) and (x2, y2) = (30, 18)

Results for different data sets:

$\Delta x = 10$ and $\Delta y = 8$

$P_o = 2\Delta y - \Delta x = 6$

Results for different data sets:

| K | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Pk | 6 | 2 | -2 | 14 | 10 | 6 | 2 | -2 | 14 | 10 |
| X,Y | 21,11 | 22,12 | 23,12 | 24,13 | 25,14 | 26,15 | 27,16 | 28,16 | 29,17 | 30,18 |

Program:

```
#include<stdio.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;
   dx=x1-x0;
   dy=y1-y0;
   x=x0;
   y=y0;
   p=2*dy-dx;
   while(x<x1)
   {
      if(p>=0)
      {
         putpixel(x,y,7);
         y=y+1;
         p=p+2*dy-2*dx;
      }
      else
      {
         putpixel(x, y, 7);
         p=p+2*dy;}
         x=x+1;
      }
}
int main()
{
   int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
   initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
   printf("Enter co-ordinates of first point: ");
   scanf("%d %d", &x0, &y0);
   printf("Enter co-ordinates of second point: ");
   scanf("%d %d", &x1, &y1);
```

```
    drawline(x0, y0, x1, y1);
    return 0;
}
```

Viva Questions:

1. What is decision parameter in this algorithm?
2. What is the value of initial decision parameter?
3. What is the value of decision parameter at Kth step?
4. What is the value of decision parameter at (K+1)th if Pk is <0; step?
5. What is the value of decision parameter at (K+1)th if Pk is >0; step?
6. What is the value of decision parameter at (K+1)th if Pk is =0; step
7. What is the equation of line having end points (x1, y1) and (x2, y2)?
8. What is the equation of line in slope-intercept form?

**Experiment No.: 5**

**Name of Experiment: Mid Point Circle Drawing Algorithm**

**Aim:**

1. Draw an arc between two points
2 Draw a circle

**Algorithm:**

1. Input radius r and circle centre (Xc, Yc) and obtain the first point on the circumference of the circle centered at (0,0).

   X = 0 and y = r.

2. Calculate initial decision parameter

   Po = 5/4 – r (or 1 – r for integer calculation)

3. At each X k, starting at K=0, perform the following test:

   a. if $P_k<0$ the next point to plot is $(X_K + 1, Y_k)$ and

   $P_{k+1} = P_k + 2 X_{K+1} + 1$

   b. Otherwise the next point to plot is $(X_K + 1, Y_k - 1)$ and

   $P_{k+1} = P_k + 2 X_{K+1} - 2 Y_{K+1} + 1$

4. Determine symmetry points in the other seven octants.
5. Move each calculated position (X, Y) onto the circular path centered at (Xc, Yc) and plot the coordinate values.

   X = Xc + X, Y = Yc + Y

6. Repeat steps 3 to 5 until X>=Y.

**Test Data:** Yc = 0, Yc = 0 and radius = 10

X0 = 0 and Y0 = 10

P0 = 1 – 10 = -9

**Results for different data sets:**

| K | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Pk | -9 | -6 | -1 | 6 | -3 | 8 | 5 |
| X,Y | 1, 10 | 2, 10 | 3, 10 | 4, 9 | 5, 9 | 6, 8 | 7, 7 |

Program

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
int gd=DETECT, gm;
int i, r, x=0, y, xc, yc;
float d;
clrscr();
initgraph(&gd, &gm, "c://tc//bgi");
printf("Enter Radius\n");
scanf("%d", &r);
printf("Enter Center of circle\n");

scanf("%d", &xc);
scanf("%d", &yc);
d=1.25-r;
y=r;
do
{
if(d<0.0)
{
x=x+1;
d=d+2*x+1;
}
else
{
x=x+1;
y=y-1;
d=d+2*x-2*y+1;
}
putpixel(xc+x, yc+y, 5);
putpixel(xc-y, yc-x, 5);
```

```
putpixel(xc+y, yc-x, 5);
putpixel(xc-y, yc+x, 5);
putpixel(xc+y, yc+x, 5);
putpixel(xc-x, yc-y, 5);
putpixel(xc+x, yc-y, 5);
putpixel(xc-x, yc+y, 5);
}
while(x<y);
getch();
}
```

**Viva Questions:**

1. What is equation of circle?
2. How you transfer the circle to its original canter?
3. What is 8 way symmetry property?
4. What is the slope of circle at the point on the line Y=X ?
5. What is the slope of circle at the point on the line Y=0 ?
6. What is the value of initial decision parameter?
7. What is the value of decision parameter at (K+1)th if Pk is >0; step?
8. What is the value of decision parameter at (K+1)th if Pk is <0; step?
9. What is the value of decision parameter at (K+1)th if Pk is =0; step?
10. What is the decision parameter in this algorithm?

**Experiment No.: 6**

**Name of Experiment: Mid Point Ellipse Drawing Algorithm**

**Aim:**

1. Draw an arc between two points
2. Draw an ellipse

**Algorithm:**

1. Input radius $r_x$ & $r_y$ and ellipse center (Xc, Yc) and obtain the first point on the circumference of the ellipse centered at (0, 0).

   $X_0 = 0$ and $y_0 = r_x$.

2. Calculate initial decision parameter in the region 1

   $P1o = r_y^2 - r_x^2 r_y + (1/4) r_x^2$

3. At each $X_k$, starting at K=0, perform the following test:

   a. if $P1_k<0$ the next point along the ellipse centered at (0. 0) is $(X_{K+1}, Y_k)$ and

   $$P1_{k+1} = P1_k + 2 r_y^2 X_{K+1} + r_y^2$$

   b. Otherwise the next point along the ellipse is $(X_K + 1, Y_k - 1)$ and

   $$P1_{k+1} = P1_k + 2 r_y^2 X_{K+1} - 2 r_x^2 Y_{K+1} + r_y^2$$

   With

   $$2 r_y^2 X_{K+1} = 2 r_y^2 X_K + 2 r_y^2 \text{ and } 2 r_x^2 Y_{K+1} = 2 r_x^2 Y_K - 2 r_x^2$$

   And continue until $2 r_y^2 X >= 2 r_x^2 Y$.

4. Calculate initial decision parameter in the region 2 using the last point (Xo, Yo) calculated in region 1 as

   $$P2o = r_y^2 (X_0 + 1/2)^2 - r_x^2 (Y_0 - 1)^2 + r_x^2 r_y^2$$

5. At each $Y_k$ in region 2, starting at K=0, perform the following test:

   a. if $P2_k<0$ the next point along the ellipse centered at (0. 0) is $(X_K, Y_k - 1)$ and

   $$P2_{k+1} = P2_k - 2 r_x^2 Y_{K+1} + r_x^2$$

   b. Otherwise the next point along the ellipse is $(X_K + 1, Y_k - 1)$ and

   $$P2_{k+1} = P2_k + 2 r_y^2 X_{K+1} - 2 r_x^2 Y_{K+1} + r_x^2$$

   Using the same incremental calculations for x and y as in region 1.

6. Determine symmetry points in the other three octants.

5. Move each calculated position (X, Y) onto the elliptical path centered at (Xc, Yc) and plot the coordinate values.

   $X = Xc + X, \quad Y = Yc + Y$

6. Repeat steps for region 2 until y <=0

**Test Data:** $r_x = 8$ & $r_y = 6$

$2r_y^2 x = 0$

$2r_x^2 y = 2r_x^2 r_y$

**Results for different data sets:**

For region 1: P1o = -332

| K | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| P1k | -332 | -224 | -44 | 208 | -108 | 288 | 244 |
| $(X_{K+1}, Y_{k+1})$ | (1, 6) | (2, 6) | (3, 6) | (4, 5) | (5, 5) | (6, 46) | (7, 3) |
| $2 r_y^2 X_{K+1}$ | 72 | 144 | 216 | 288 | 360 | 432 | 504 |
| $2 r_x^2 Y_{K+1}$ | 768 | 768 | 768 | 640 | 640 | 512 | 384 |

Since $2 r_y^2 X >= 2 r_x^2 Y$ ao we move out of the region 1.

For region 1: P2o = -151

| K | 0 | 1 | 2 |
|---|---|---|---|
| P2k | -151 | 233 | 745 |
| $(X_{K+1}, Y_{k+1})$ | (8, 2) | (8, 1) | (8, 0) |
| $2 r_y^2 X_{K+1}$ | 576 | 576 | - |
| $2 r_x^2 Y_{K+1}$ | 256 | 128 | - |

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

int main() {
    /* request auto detection */
    int gdriver = DETECT, gmode, err;
    long midx, midy, xradius, yradius;
    long xrad2, yrad2, twoxrad2, twoyrad2;
    long x, y, dp, dpx, dpy;

    /* initialize graphic mode */
    initgraph(&gdriver, &gmode, "C:/TC/BGI");
    err = graphresult();

    if (err != grOk) {
        /* error occurred */
        printf("Graphics Error: %s\n",
                grapherrormsg(err));
        return 0;
    }

    /* x axis radius and y axis radius of ellipse */
    xradius = 100, yradius = 50;

    /* finding the center postion to draw ellipse */
    midx = getmaxx() / 2;
    midy = getmaxy() / 2;

    xrad2 = xradius * xradius;
    yrad2 = yradius * yradius;

    twoxrad2 = 2 * xrad2;
```

```
twoyrad2 = 2 * yrad2;
x = dpx = 0;
y = yradius;
dpy = twoxrad2 * y;

putpixel(midx + x, midy + y, WHITE);
putpixel(midx - x, midy + y, WHITE);
putpixel(midx + x, midy - y, WHITE);
putpixel(midx - x, midy - y, WHITE);

dp = (long) (0.5 + yrad2 - (xrad2 * yradius) + (0.25 * xrad2));

while (dpx < dpy) {
      x = x + 1;
      dpx = dpx + twoyrad2;
      if (dp < 0) {
            dp = dp + yrad2 + dpx;
      } else {
            y = y - 1;
            dpy = dpy - twoxrad2;
            dp = dp + yrad2 + dpx - dpy;
      }

      /* plotting points in y-axis(top/bottom) */
      putpixel(midx + x, midy + y, WHITE);
      putpixel(midx - x, midy + y, WHITE);
      putpixel(midx + x, midy - y, WHITE);
      putpixel(midx - x, midy - y, WHITE);
      delay(100);
}

delay(500);
```

```
    dp = (long)(0.5 + yrad2 * (x + 0.5) * (x + 0.5) + xrad2 * (y - 1) * (y - 1) -
xrad2 * yrad2);

        while (y > 0) {
            y = y - 1;
            dpy = dpy - twoxrad2;

            if (dp > 0) {
                dp = dp + xrad2 - dpy;
            } else {
                x = x + 1;
                dpx = dpx + twoyrad2;
                dp = dp + xrad2 - dpy + dpx;
            }
            /* plotting points at x-axis(left/right) */
            putpixel(midx + x, midy + y, WHITE);
            putpixel(midx - x, midy + y, WHITE);
            putpixel(midx + x, midy - y, WHITE);
            putpixel(midx - x, midy - y, WHITE);
            delay(100);
        }
        getch();
        /* deallocate memory allocated for graphic screen */
        closegraph();
        return 0;
    }
```

**Viva Questions:**

1. What is equation of ellipse?
2. How you transfer the ellipse to its original canter?
3. How we can draw the same points in other 3 Quadrants?
4. What is the condition to move out of the region 1?
5. What is the value of initial decision parameter for region 1?
6. What is the value of initial decision parameter for region 2?
7. What is the value of decision parameter for R1 at (K+1)th if Pk is >0; step?
8. What is the value of decision parameter for R2at (K+1)th if Pk is <0; step?
9. What is the value of decision parameter for R2at (K+1)th if Pk is =0; step?
10. What is Aspect Ratio?

**Experiment-7**

**Aim: Two dimensional transformations- Translation**

Translation: It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

Translation of point: To translate a point from coordinate position (x, y) to another (x1 y1), we add algebraically the translation distances Tx and Ty to original coordinate.

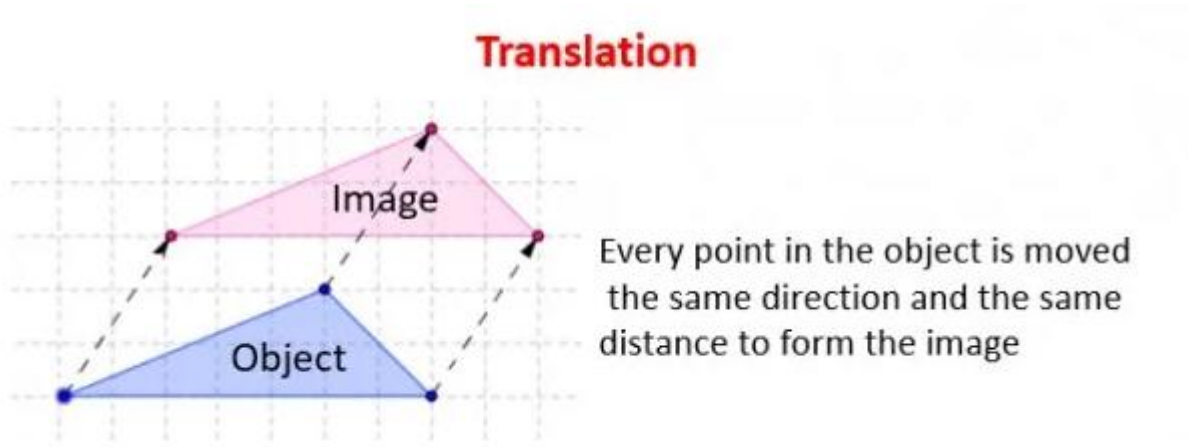
$x1=x+Tx$

$y1=y+Ty$

The translation pair (Tx, Ty) is called as translation vectors.

Translation is a movement of objects without deformation. Every position or point is translated by the same amount. When the straight line is translated, then it will be drawn using endpoints.

Example:



```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT, gm ;
int x3, y3, x1, y1, x2, y2, tx, ty;
clrscr();
initgraph(&gd, &gm , "c:\\turboc3\\bgi");
printf("\n Enter first coordinate : ");
scanf("%d %d", &x1, &y1);
printf("\n Enter second coordinate: ");
scanf("%d %d", &x2, &y2);
printf("\n Enter third coordinate: ");
scanf("%d %d", &x3, &y3);
line(x1, y1, x2, y2);
line(x2, y2, x3, y3);
line(x3, y3, x1, y1);
printf("\n enter translation vectors : ");
scanf("%d %d", &tx, &ty);
setcolor(BLUE);
line(x1+tx, y1+ty, x2+tx, y2+ty);
line(x2+tx, y2+ty, x3+tx, y3+ty);
line(x3+tx, y3+ty, x1+tx, y1+ty);
getch();
closegraph();
}
```

## Experiment-8

## Aim: Two dimensional transformations- Rotation

**Rotation**: It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

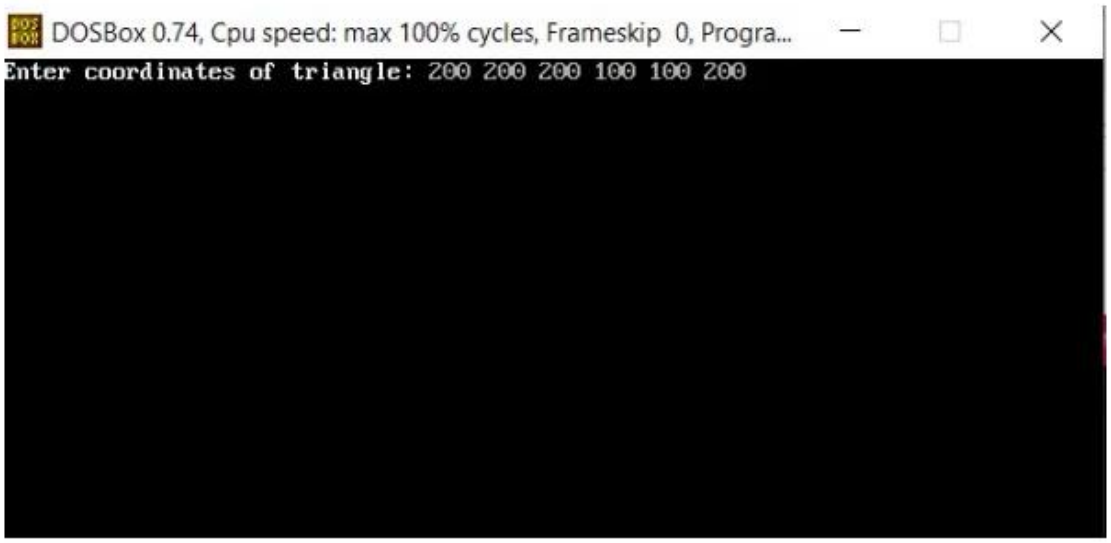
**Types of Rotation**

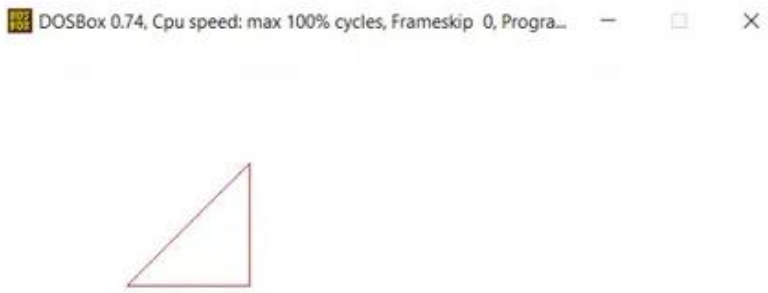
Anticlockwise-The positive value of the rotation angle rotates an object in an anti-clockwise direction.

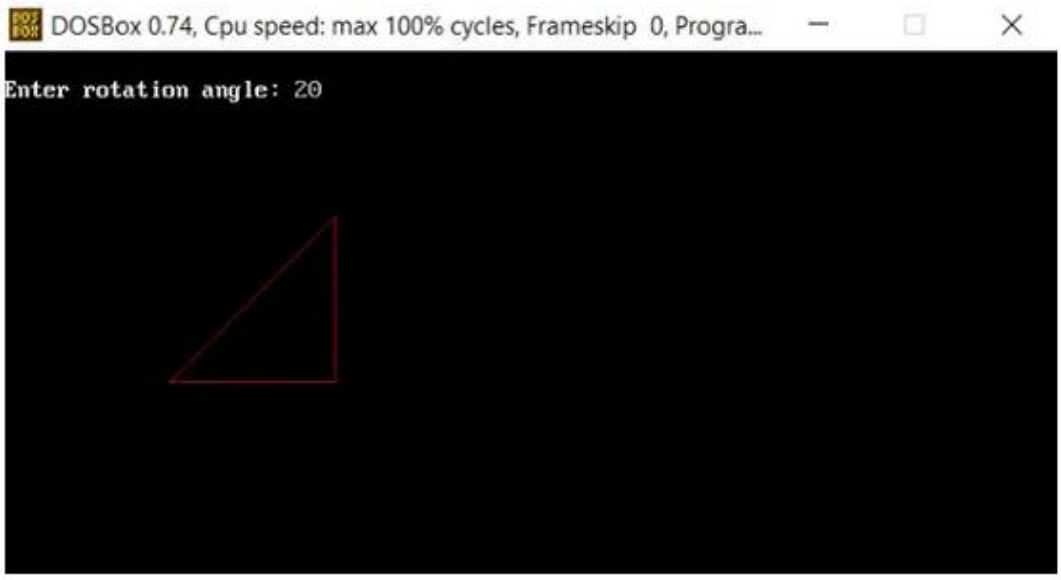
Counterclockwise- The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
   int gd=0, gm, x1, y1, x2, y2, x3, y3;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
   scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);
   setbkcolor(WHITE);
   cleardevice();
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   getch();
   setbkcolor(BLACK);  //setting background color
   printf("Enter rotation angle: ");
   scanf("%lf", &angle);
   setbkcolor(WHITE);  //setting background color
  //M_PI : Pi, the ratio of a circle's circumference to its diameter.
   c = cos(angle *M_PI/180);
   s = sin(angle *M_PI/180);
   x1 = floor(x1 * c + y1 * s);
   y1 = floor(-x1 * s + y1 * c);
   x2 = floor(x2 * c + y2 * s);
   y2 = floor(-x2 * s + y2 * c);
   x3 = floor(x3 * c + y3 * s);
   y3 = floor(-x3 * s + y3 * c);
   cleardevice();
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   getch();
   closegraph();
```

Output


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter coordinates of triangle: 200 200 200 100 100 200


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

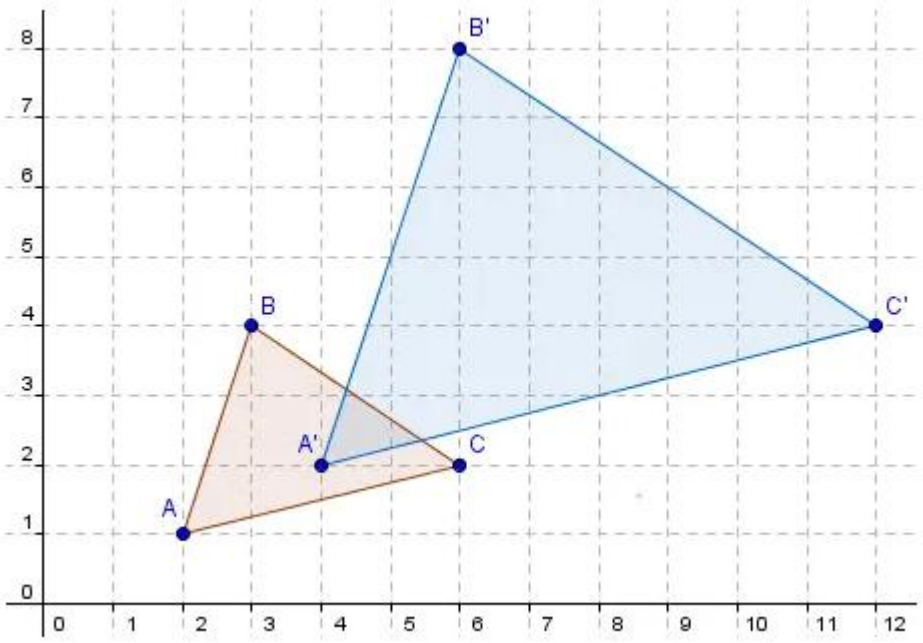

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
Enter rotation angle: 20

## Experiment-9

## Aim: Two dimensional transformations- Scaling

**Scaling of triangle**

Scaling: It is used to change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. Sx in x direction Sy in y-direction. If the original position is x and y. Scaling factors are Sx and Sy.




Program

// Matrix Multiplication to find new Coordinates.

// s[ ][ ] is scaling matrix.

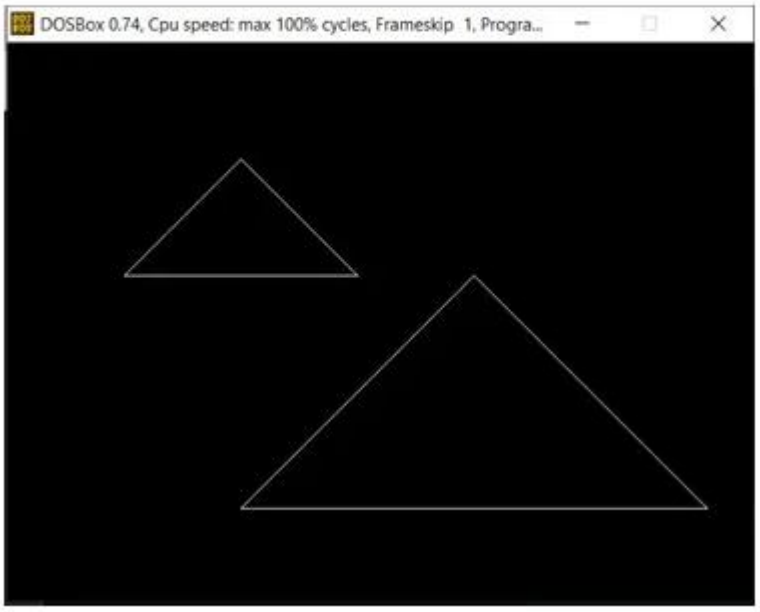
// p[ ][ ] is to store points that needs to be scaled.

// p[0][0] is x coordinate of point.

// p[1][0] is y coordinate of given point.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void fC(int s[ ][2], int p[ ][1])
{
   int temp[2][1] = { 0 };
   for (int i = 0; i < 2; i++)
      for (int j = 0; j < 1; j++)
         for (int k = 0; k < 2; k++)
            temp[i][j] += (s[i][k] * p[k][j]);
   p[0][0] = temp[0][0];
   p[1][0] = temp[1][0];
}
void scale(int x[ ], int y[ ], int sx, int sy)
{
   // Triangle before Scaling
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
   // Initializing the Scaling Matrix.
   int s[2][2] = { sx, 0, 0, sy };
   int p[2][1];
   for (int i = 0; i < 3; i++)
   {
      p[0][0] = x[i];
      p[1][0] = y[i];
      fC(s, p);
      x[i] = p[0][0];
      y[i] = p[1][0];
   }
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
}
void main()
{
   int x[] = { 100, 200, 300 };
```

**Output**




---

## Experiment-10

### Aim: Two dimensional transformations- Reflection

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>
void main()
{
clrscr();
int graphdriver=DETECT, graphmode;
initgraph(&graphdriver, &graphmode,"C://TC/BGI");
int x, y, x1, a[3][3];
double b[3][3], c[3][3];
printf("\n Enter Ist coordinates of triangle:");
scanf(%d %d, &a[0][0], &a[1][0])
printf("\n Enter 2nd coordinates of triangle:");
scanf(%d %d, &a[0][1], &a[1][1]);
printf("\n Enter 3rd coordinates of triangle:");
scanf(%d %d, &a[0][2], &a[1][2]);
printf("\n Enter 1. for reflection in x-axis:\n");
printf("\n Enter 2. for reflection in y-axis:\n");
printf("\n Enter 3. for reflection in both the axis:\n");
scanf(%d, &x);
cleardevice();
```

```
line(320, 0, 320, 479);
line(0, 240, 639, 240);
line(a[0][0], a[1][0], a[0][1], a[1][1]);
line(a[0][1], a[1][1], a[0][2], a[1][2]);
line(a[0][0], a[1][0], a[0][2], a[1][2]);
switch(x)
{
case 1:
        b[0][0]=640-a[0][0];
        b[0][1]=640-a[0][1];
        b[0][2]=640-a[0][2];
        b[1][0]=a[1][0];
        b[1][1]=a[1][1];
        b[1][2]=a[1][2];
                line(320, 0, 320, 479);
                line(0, 240, 639, 240);
                line(b[0][0], b[1][0], b[0][1], b[1][1]);
                line(b[0][1], b[1][1], b[0][2], b[1][2]);
                line(b[0][0], b[1][0], b[0][2], b[1][2]);
                getch();
                break;
case 2:
        b[1][0]=480-a[1][0];
        b[1][1]=480-a[1][1];
        b[1][2]=480-a[1][2];
        b[0][0]=a[0][0];
        b[0][1]=a[0][1];
        b[0][2]=a[0][2];
                line(320, 0, 320, 479);
                line(0, 240, 639, 240);
                line(b[0][0], b[1][0], b[0][1], b[1][1]);
                line(b[0][1], b[1][1], b[0][2], b[1][2]);
                line(b[0][0], b[1][0], b[0][2], b[1][2]);
                getch();
                break;
case 3:
        b[0][0]=640-a[0][0];
        b[0][1]=640-a[0][1];
        b[0][2]=640-a[0][2];
        b[1][0]=a[1][0];
        b[1][1]=a[1][1];
        b[1][2]=a[1][2];
                line(320, 0, 320, 479);
                line(0, 240, 639, 240);
                line(b[0][0], b[1][0], b[0][1], b[1][1]);
                line(b[0][1], b[1][1], b[0][2], b[1][2]);
                line(b[0][0], b[1][0], b[0][2], b[1][2]);
        b[1][0]=480-a[1][0];
        b[1][1]=480-a[1][1];
        b[1][2]=480-a[1][2];
```

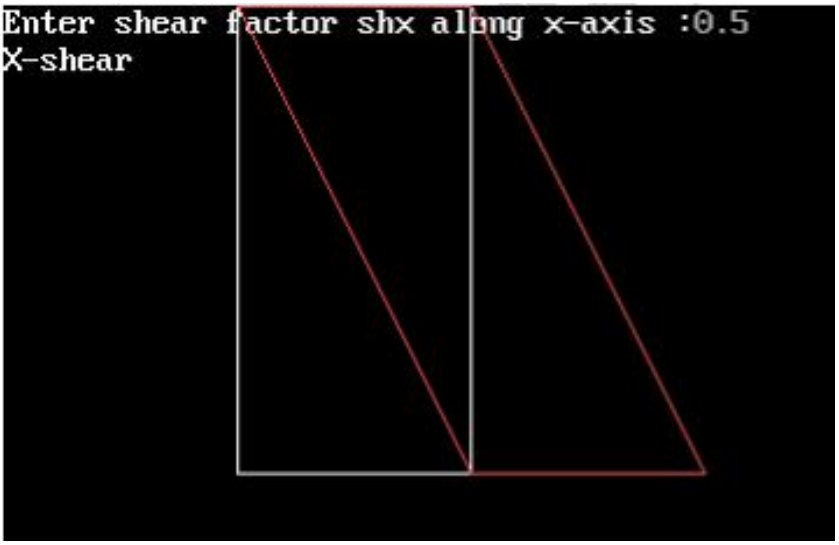
```
        b[0][0]=a[0][0];
        b[0][1]=a[0][1];
        b[0][2]=a[0][2];
                line(320, 0, 320, 479);
                line(0, 240, 639, 240);
                line(b[0][0], b[1][0], b[0][1], b[1][1]);
                line(b[0][1], b[1][1], b[0][2], b[1][2]);
                line(b[0][0], b[1][0], b[0][2], b[1][2]);
                getch();
                break;
}
getch();
closegraph();
}
```

## Experiment-11

## Aim: Two dimensional transformations- Shear

**PROGRAM FOR X-SHEAR**

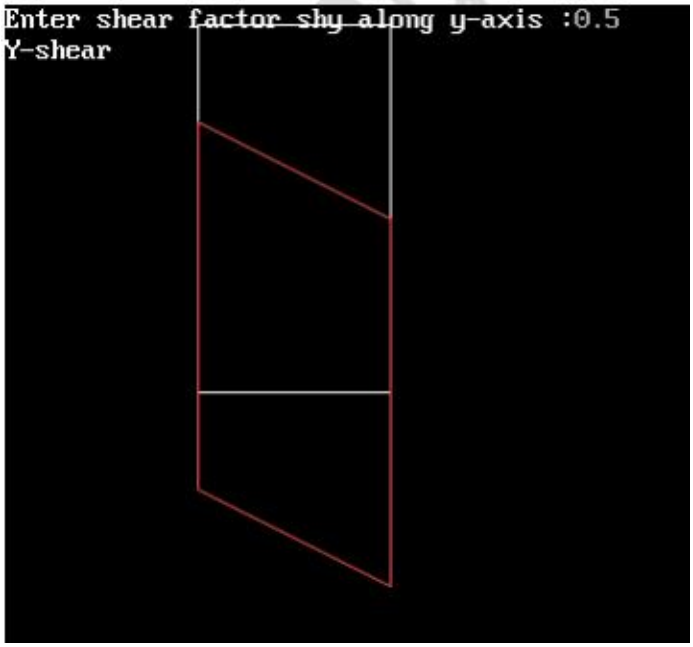
```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void main()
{
        int gd=DETECT, gm;
        float shx, shy;
        initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
        printf("Enter shear factor shx along x-axis :");
        scanf("%f", &shx);
        line(100, 0, 200, 0);
        line(200, 0, 200, 200);
        line(200, 200, 100, 200);
        line(100, 200, 100, 0);
        printf("X-shear");
        setcolor(12);
        line((100+(0*shx)), 0, (200+(0*shx)), 0);
        line((200+(0*shx)), 0, (200+(200*shx)), 200);
        line((200+(200*shx)), 200, (100+(200*shx)), 200);
        line((100+(200*shx)), 200, (100+(0*shx)), 0);
        getch();
}
```




**Output:** In above output, red lined rectangle denotes object after x-shear transformation

**PROGRAM FOR Y-SHEAR**

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void main()
{
        int gd=DETECT, gm;
        float shx, shy;
        initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
        printf("Enter shear factor shy along y-axis :");
        scanf("%f",);
        line(100, 10, 200, 10);
        line(200, 10, 200, 200);
        line(200, 200, 100, 200);
        line(100, 200, 100, 10);
        printf("Y-shear");
        setcolor(12);
        line(100, 10+(shy*100), 200, 10+(shy*200));
        line(200, 10+(shy*200), 200, 200+(shy*200));
        line(200, 200+(shy*200), 100, 200+(shy*100));
        line(100, 200+(shy*100), 100, 10+(shy*100));
        getch();
}
```




**Output:** In above output, red lined rectangle denotes object after y-shear transformation