Experiment No. 4

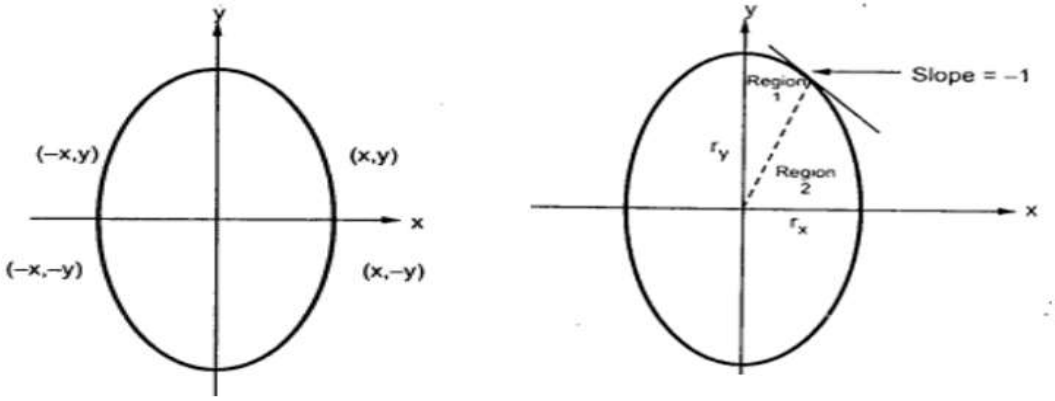
Aim-To implement midpoint Ellipse algorithm

Objective:

Draw the ellipse using Mid-point Ellipse algorithm in computer graphics. Midpoint ellipse algorithm plots (finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Theory:

Midpoint ellipse algorithm uses four way symmetry of the ellipse to generate it. Figure shows the 4-way symmetry of the ellipse.



Here the quadrant of the ellipse is divided into two regions as shown in the fig. Fig. shows the division of first quadrant according to the slope of an ellipse with rx < ry. As ellipse is drawn from 90 0 to 0 0 , x moves in positive direction and y moves in negative direction and ellipse passes through two regions 1 and 2.

The equation of ellipse with center at (xc, yc) is given as -

[(x – xc) / rx] 2 + [(y – yc) / ry] 2 = 1

Therefore, the equation of ellipse with center at origin is given as -

[x / rx] 2 + [y / ry] 2 = 1

i.e. x 2 ry 2 + y 2 rx 2 = rx 2 ry 2

Let, f ellipse (x, y) = x2 ry2 + y2 rx2 - rx2 ry2

Algorithm:

Step 1: Start

Step 2: Declare r_x , r_y, x , y , m , dx , dy , P , P2.

Step 3: Initialize initial point of region1 as

x=0 , y = r_y

Step 4: Calculate P= $r_y^2 + r_x^2 / 4 - r_y r_x^2$

dx = 2 r_y^2 x

dy = 2 r_x^2 y

Step 5: Update values of dx and dy after each iteration.

Step 6: Repeat steps while (dx < dy):

Plot (x,y)

if(P < 0)

Update x = x+1 ;

P += r_y^2 [2x + 3]

Else

Update x = x + 1

y= y - 1

Step 7: When dx ≥ dy, plot region 2:

Step 8: Calculate P2 = r_y^2(x+1 / 2)^2 + r_x^2(y -1)^2- $r_x^2 r_y^2$

Step 9: Repeat till (y > 0)

If (P2 > 0)

Update y = y-1 (x will remain same)

P2 =P2 -2 y r_x^2 + r_x^2

else

x = x+1

y = y-1

$P2 = P2 + 2\ r_{y^2}[2x] - 2\ y\ r_{x^2} + r_{x^2}$

Step 10: End

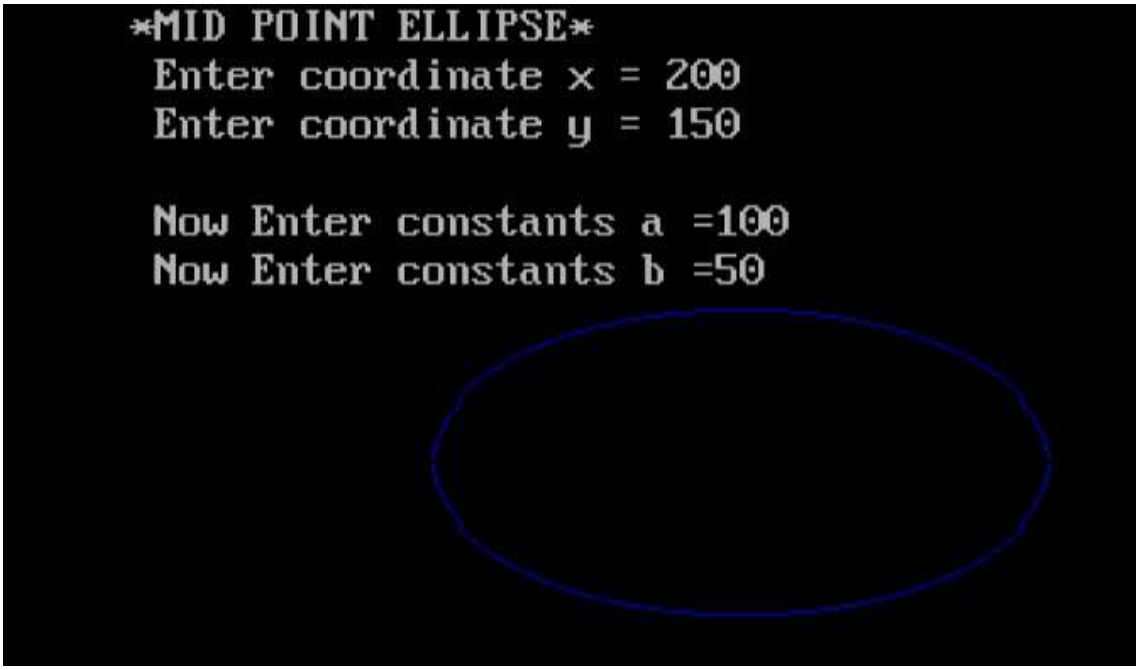
Program:

```
#include<stdio.h>
#include<graphics.h>
int main()
{
        long x,y,x_center,y_center;
        long a_sqr,b_sqr,fx,fy,d,a,b,tmp1,tmp2;
        int g_driver=DETECT,g_mode;

        initgraph(&g_driver,&g_mode,"");
        printf("*MID POINT ELLIPSE*");
        printf("\n Enter coordinate x = ");
        scanf("%ld",&x_center);
        printf(" Enter coordinate y = ");
        scanf("%ld",&y_center);
        printf("\n Now Enter constants a =");
        scanf("%ld",&a,&b);
        printf(" Now Enter constants b =");
        scanf("%ld",&b);
        x=0;
        y=b;
        a_sqr=a*a;
```

```
b_sqr=b*b;
fx=2*b_sqr*x;
fy=2*a_sqr*y;
d=b_sqr-(a_sqr*b) + (a_sqr*0.25);
do
{
        putpixel(x_center+x,y_center+y,1);
        putpixel(x_center-x,y_center-y,1);
        putpixel(x_center+x,y_center-y,1);
        putpixel(x_center-x,y_center+y,1);

        if(d<0)
        {
                d=d+fx+b_sqr;
        }
        else
        {
                y=y-1;
                d=d+fx+-fy+b_sqr;
                fy=fy-(2*a_sqr);
        }
        x=x+1;
        fx=fx+(2*b_sqr);
        delay(10);
}
while(fx<fy);
tmp1=(x+0.5)*(x+0.5);
```

```
        tmp2=(y-1)*(y-1);
        d=b_sqr*tmp1+a_sqr*tmp2-(a_sqr*b_sqr);

        do
        {
                putpixel(x_center+x,y_center+y,1);
                putpixel(x_center-x,y_center-y,1);
                putpixel(x_center+x,y_center-y,1);
                putpixel(x_center-x,y_center+y,1);

                if(d>=0)
                d=d-fy+a_sqr;
                else
                {
                        x=x+1;
                        d=d+fx-fy+a_sqr;
                        fx=fx+(2*b_sqr);
                }
                y=y-1;
                fy=fy-(2*a_sqr);
        }
        while (y>0);
        getch();
        closegraph();
}
```

Output:



Conclusion: Comment on

1. Slow or fast
2. Difference with circle
3. Importance of object