EXP 04:- MID-POINT-ELLIPSE ALGORITHM

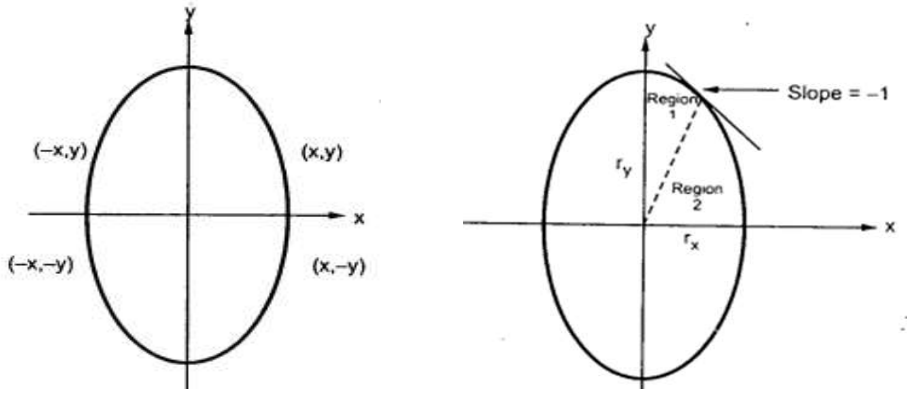
Aim: To implement midpoint Ellipse algorithm

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 divison 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 -

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

Therefor, 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:

1) Read rx and ry.

2) Initialise starting point as

x = 0

y = ry

3) Calculate the initial value of decision parameter in region 1 as

$d1 = ry^2 - rx^2\ ry + (rx^2 / 4)$

4) Initialise dx and dy as

$dx = 2\ ry^2\ x$

$dy = 2\ rx^2\ y$

5) do

```
        {
        plot (x, y)
        if (d1 < 0)
                {
                x =  x+ 1
                y = y
                dx = dx + 2 ry^2
                d1 = d1 + dx + ry^2
                }
        else
                {
                x =  x+ 1
                y = y - 1
                dx = dx + 2 ry^2
                dy = dy - 2 rx^2
                d1 = d1 + dx - dy + ry^2
                }
        }
```

while (dx < dy)

6) Calculate the initial value of decision parameter in region 2 as

$d2 = ry^2\ [x + (1/2)]^2 + rx^2\ (y - 1)^2 + rx^2\ ry^2$

7) do

```
        {
        plot (x, y)
        if (d2 > 0)
                {
                x =  x
                y = y - 1
                dy = dy + 2 rx^2
                d2 = d2 - dy + rx^2
                }
        else
                {
                x =  x+ 1
```

y = y - 1
dy = dy - 2 rx^2
dx = dx + 2 ry^2
d2 = d2 + dx – dy + rx^2
}
}
while (y > 0)

8) Determine the symmetry points in other three quadrants.
9) Stop.

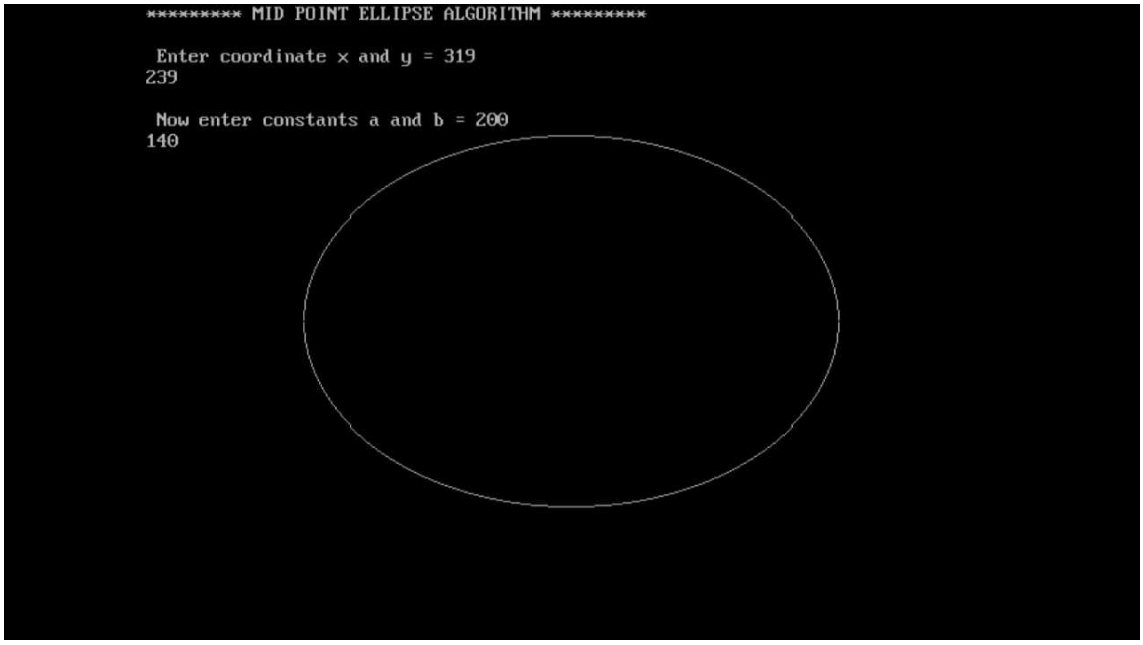
Program:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
void 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;
      clrscr();

    initgraph(&g_driver,&g_mode,"C:\\TURBOC3\\BGI");
    printf("********* MID POINT ELLIPSE ALGORITHM *********");
    printf("\n\n Enter coordinate x and y = ");
    scanf("%ld%ld",&x_center,&y_center);
    printf("\n Now enter constants a and b = ");
    scanf("%ld%ld",&a,&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,7);
  putpixel(x_center-x,y_center-y,7);
  putpixel(x_center+x,y_center-y,7);
```

```
putpixel(x_center-x,y_center+y,7);

 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,7);
putpixel(x_center-x,y_center-y,7);
putpixel(x_center+x,y_center-y,7);
putpixel(x_center-x,y_center+y,7);

 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: The Mid-point-ellipse Algorithm efficiently plots ellipses by utilizing symmetry and pixel selection based on midpoint calculations. It avoids unnecessary floating-point operations, making it faster than the traditional approach. However, it may still suffer from pixelated or jagged appearance due to discrete grid limitations