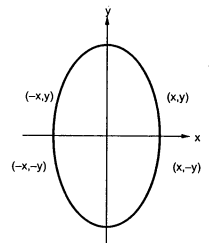
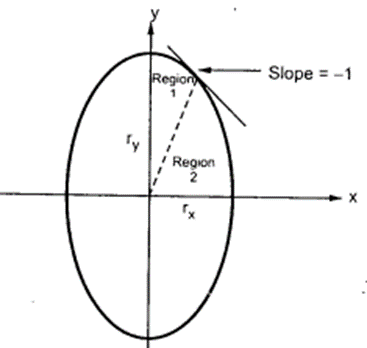
**Aim:**  To implement midpoint Ellipse algorithm

**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 900 to 00 , 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. x2 ry2 + y2 rx2 = rx2 ry2

Let, fellipse (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 = ry2 – rx2 ry + (rx2 / 4)

4) Initialise dx and dy as

dx = 2 ry2 x

dy = 2 rx2 y

5) do

{

plot (x, y)

if (d1 < 0)

{

x = x+ 1

y = y

dx = dx + 2 ry2

d1 = d1 + dx + ry2

}

else

{

x = x+ 1

y = y - 1

dx = dx + 2 ry2

dy = dy – 2 rx2

d1 = d1 + dx – dy + ry2

}

}

while (dx < dy)

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

d2 = ry2 [x + (1/2)]2 + rx2 (y - 1)2 + rx2 ry2

7) do

{

plot (x, y)

if (d2 > 0)

{

x = x

y = y - 1

dy = dy + 2 rx2

d2 = d2 - dy + rx2

}

else

{

x = x+ 1

y = y - 1

dy = dy - 2 rx2

dx = dx + 2 ry2

d2 = d2 + dx – dy + rx2

}

}

while (y > 0)

8) Determine the symmetry points in other three quadrants.

9) Stop.

**Program:**

#include<stdio.h>

#include<conio.h>

#include<graphics.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 gd = DETECT, gm;

clrscr();

initgraph(&gd,&gm,"..//BGI");

printf("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,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 = fy+(2\*b\_sqr);

delay(50);

}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;

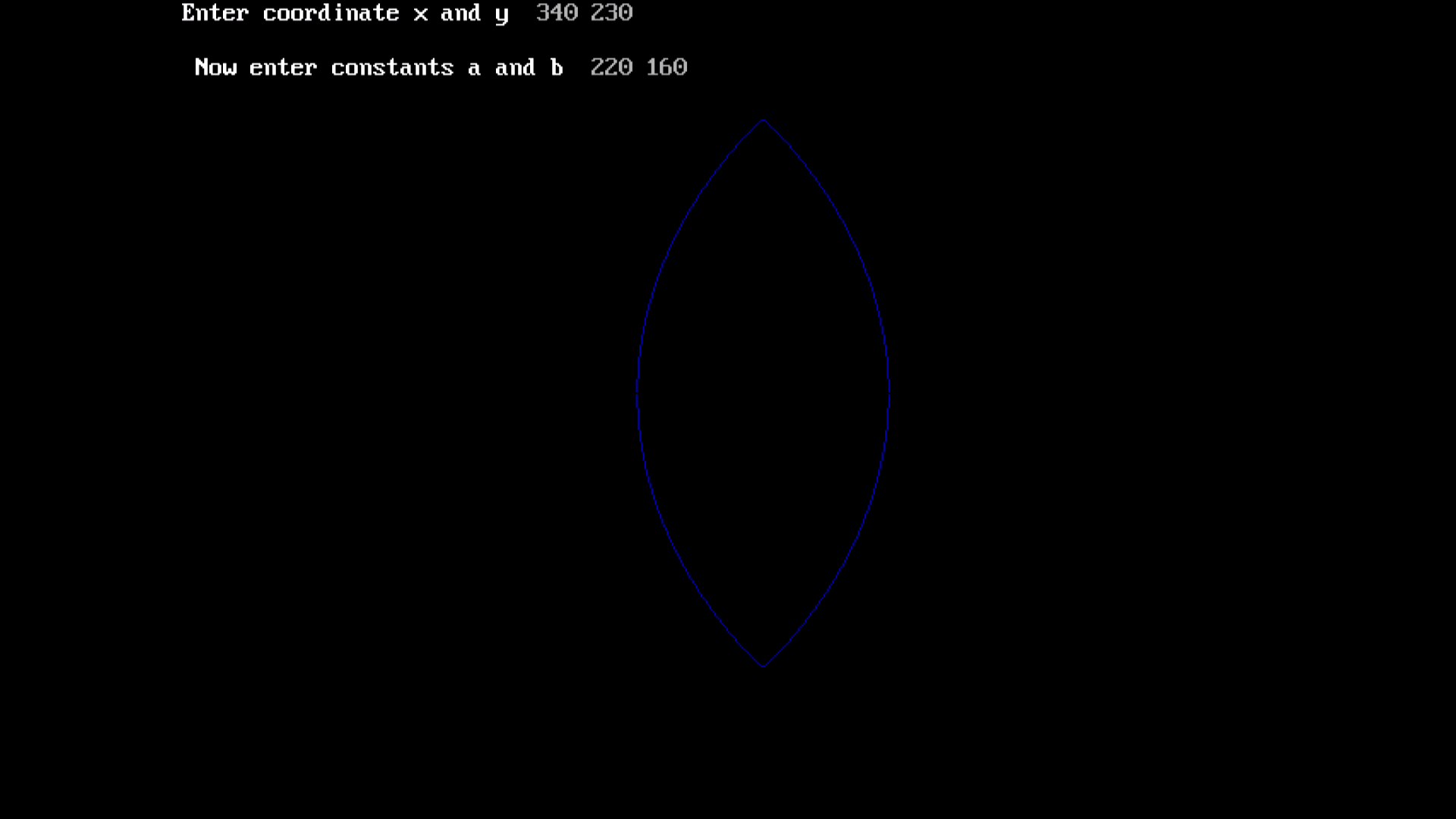
fy = fy-(2\*a\_sqr);

}while(y>0);

getch();

closegraph();

}

**Output:** 

**Conclusion:** Comment on

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