Experiment No. 4

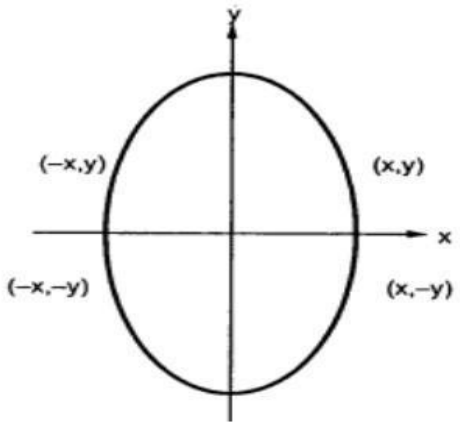
Aim-To implement midpoint Ellipse algorithm

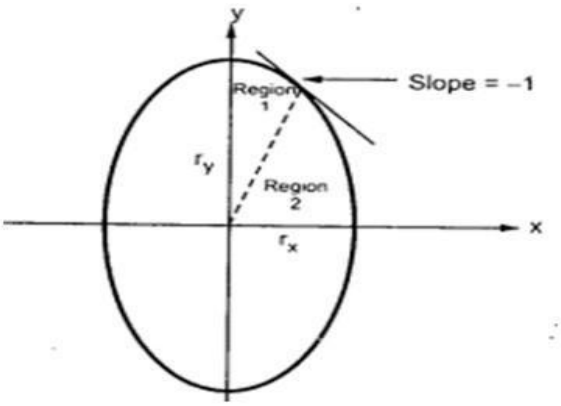
Objective:

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 2Let, f ellipse (x, y) = x2 ry2 + y2 rx2 - rx2 ry2

Algorithm:

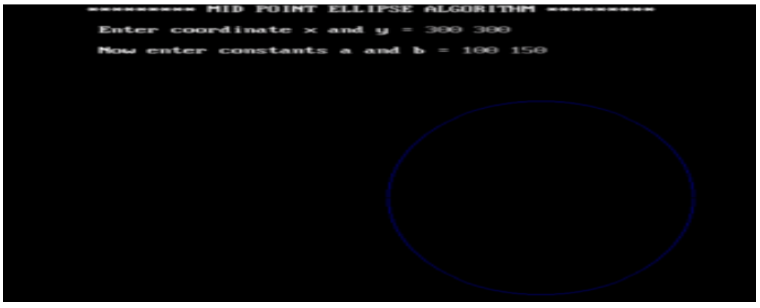
Step 1: Start
Step 2: Declare r_x , r_y , x , y , m , dx , dy , P , P2.
Step 3: Initialize initial point of region1 as
x=0 , y = r_y
Step 4: Calculate P= $r_y^2 + r_x^2 / 4 - r_y r_x^2$
dx = $2 r_y^2 x$
dy = $2 r_x^2 y$
Step 5: Update values of dx and dy after each iteration.
Step 6: Repeat steps while (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<conio.h>
#include<graphics.h>
void pixel(int x,int y,int xc,int yc)
{
putpixel(x+xc,y+yc,BLUE);
putpixel(x+xc,-y+yc,BLUE);
putpixel(-x+xc,y+yc,BLUE);
putpixel(-x+xc,-y+yc,BLUE);
putpixel(y+xc,x+yc,BLUE);
putpixel(y+xc,-x+yc,BLUE);
putpixel(-y+xc,x+yc,BLUE);
putpixel(-y+xc,-x+yc,BLUE);
}
```

```
main()
{
int gd=DETECT,gm=0,r,xc,yc,x,y;
float p;
//detectgraph(&gd,&gm);
initgraph(&gd,&gm," ");
        printf("\n ********* MID POINT ELLIPSE ALGORITHM ********* ")
printf("\n Enter the radius of the circle:");
scanf("%d",&r);
printf("\n Enter the center of the circle:");
scanf("%d %d",&xc,&yc);
y=r;
x=0;
p=(5/4)-r;
while(x<y)
{
        if(p<0)
        {
                x=x+1;
                y=y;
                p=p+2*x+3;
        }
        else
        {
                x=x+1;
                y=y-1;
                p=p+2*x-2*y+5;
        }
        pixel(x,y,xc,yc);
}
getch();
closegraph();
}
```

Output:



Conclusion: Comment on

1. Slow or fast

2. Difference with circle

3. Importance of object

1. The Midpoint Ellipse Algorithm is considered a fast and efficient method for drawing ellipses, especially when compared to other algorithms that involve complex mathematical calculations like trigonometric functions. The algorithm's efficiency stems from its ability to make incremental decisions about which pixel to plot next without the need for costly operations like square roots or trigonometric functions.

2. The Midpoint Ellipse Algorithm and the Midpoint Circle Algorithm are both efficient techniques for drawing curved shapes, but they differ in their application and the type of curves they are designed for. The Midpoint Ellipse Algorithm is specifically tailored for drawing ellipses, which are elongated, oval shapes. It uses decision parameters to plot points on the ellipse efficiently, making it a preferred choice for ellipse drawing. On the other hand, the Midpoint Circle Algorithm is designed for drawing circles, which are a special case of ellipses with equal semi-major and semi-minor axes. While the underlying principles of these two algorithms are similar, the key distinction lies in their primary use: one for ellipses and the other for circles.

3. In the context of computer graphics and algorithms like the Midpoint Ellipse Algorithm, "a" and "b" represent the semi-major and semi-minor axes of the ellipse. These parameters define the size and shape of the ellipse to be drawn. The importance of "a" and "b" lies in their ability to allow the algorithm to adapt to different ellipse sizes and aspect ratios. By adjusting these values, you can control the dimensions and orientation of the resulting ellipse. This flexibility is crucial for various applications, as it allows you to draw ellipses of different sizes and proportions, making the algorithm versatile and widely applicable in graphics and design, where ellipses are used for various visual elements.