Experiment No. 4
Implement midpoint Ellipse algorithm.
Name: SHUBHAM SANJAY MOHANTY
Roll Number: 30
Date of Performance:
Date of Submission:

Experiment No. 4

Aim-To implement midpoint Ellipse algorithm

Objective:

Draw the ellipse using Mid-point Ellipse algorithm in computer graphics. Midpoint ellipse

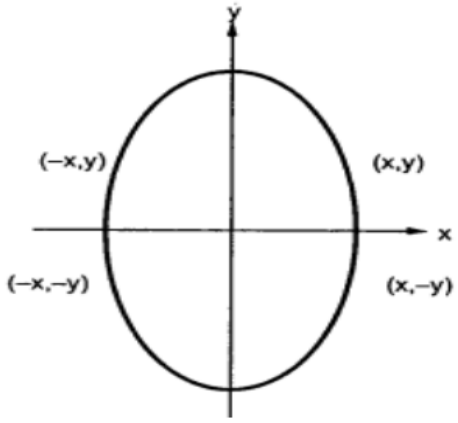
algorithm plots (finds) points of an ellipse on the first quadrant by dividing the quadrant into

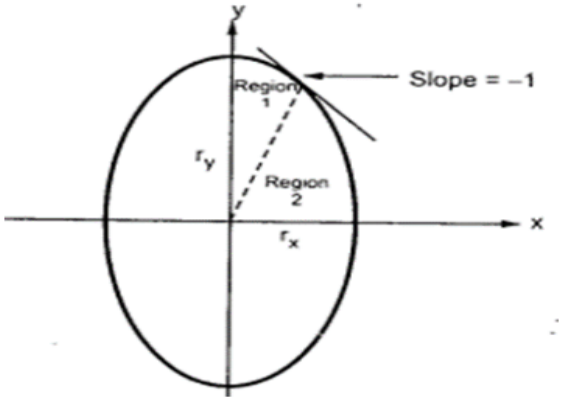
two regions.

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:

```
int x=0, y=b; [starting point]
int fx=0, fy=2a2 b [initial partial derivatives]
int p = b2-a2 b+a2/4
while (fx<="" 1="" {="" set="" pixel="" (x,="" y)="" x++;="" fx="fx" +="" 2b2;
        if (p<0)
        p = p + fx +b2;
        else
        {
                y--;
                fy=fy-2a2
```

```
                    p = p + fx +b2-fy;
            }
  }
  Setpixel (x, y);
  p=b2(x+0.5)2+ a2 (y-1)2- a2 b2
  while (y>0)
  {
            y--;
            fy=fy-2a2;
            if (p>=0)
            p=p-fy+a2
           else
            {
                    x++;
                    fx=fx+2b2
                    p=p+fx-fy+a2;
            }
            Setpixel (x,y);
  }
```

Program:

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
#include<conio.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,"C:\\TurboC3\\BGI");
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,4);
   putpixel(x_center-x,y_center-y,3);
   putpixel(x_center+x,y_center-y,2);
   putpixel(x_center-x,y_center+y,1);

   if(d<0)
   {
      d=d+fx+b_sqr;
```

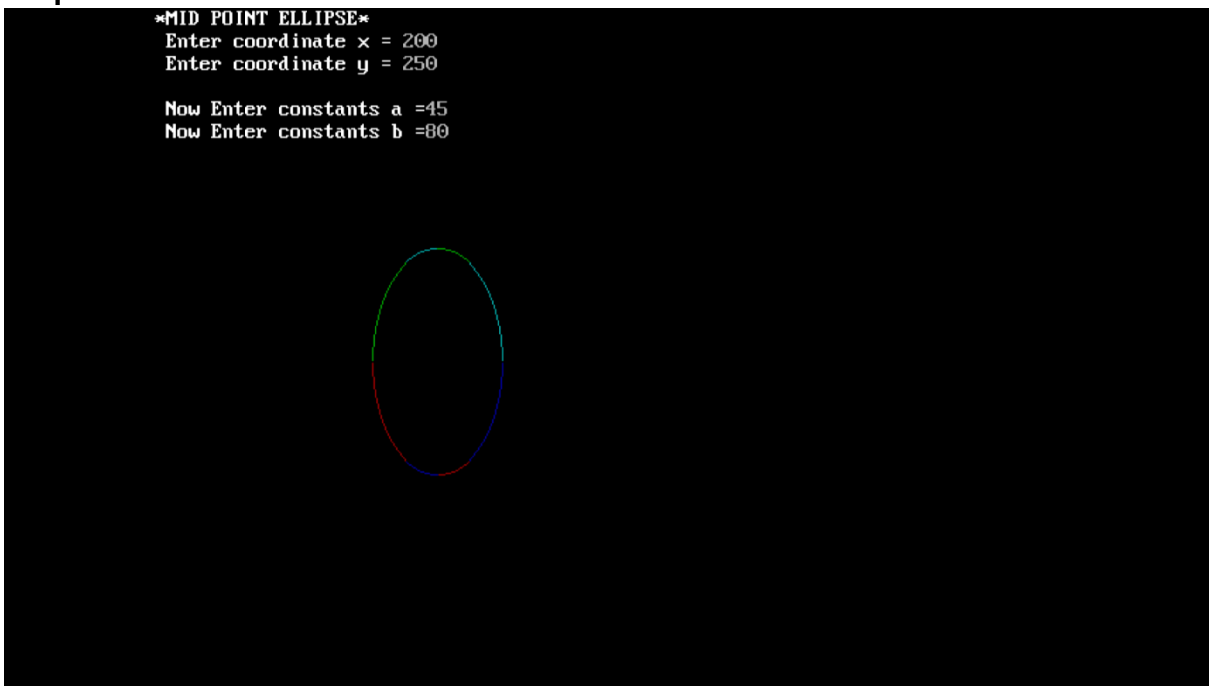
```
    }
    else
    {
        y=y-1;
        d=d+fx+-fy+b_sqr;
        fy=fy-(2*a_sqr);
    }
    x=x+1;
    fx=fx+(2*b_sqr);
    delay(10);
}
while(fx<fy);
tmp1=(x+0.5)*(x+0.5);
tmp2=(y-1)*(y-1);
d=b_sqr*tmp1+a_sqr*tmp2-(a_sqr*b_sqr);

do
{
    putpixel(x_center+x,y_center+y,1);
    putpixel(x_center-x,y_center-y,2);
    putpixel(x_center+x,y_center-y,3);
    putpixel(x_center-x,y_center+y,4);

    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();
  return 0;
}
```

Output:



Conclusion:

The algorithm used to draw an ellipse is notably different from that of a circle due to the fact that ellipses are not symmetric in the same way that circles are. The primary distinction lies in the process of calculating and plotting the ellipse's points, which involves varying both the horizontal and vertical radii as it moves along the curve. In contrast, circles have a constant radius.

The importance of ellipse drawing algorithms lies in their applicability to various real-world objects. Ellipses are commonly encountered in fields such as engineering, computer graphics, and mathematics. They represent not only simple geometric shapes but also many practical objects like wheels, orbits of celestial bodies, and even the shape of the human eye's cornea. Precisely rendering ellipses is essential for accurately representing these objects in computer graphics, engineering drawings, or scientific simulations. Hence, a robust and efficient algorithm for drawing ellipses is valuable for creating realistic and accurate depictions of objects and phenomena in these domains.