Experiment No. 4
Implement midpoint Ellipse algorithm.
Name: Jaffari Mohammed Ali Sayyed Naqi Ali
Roll Number: 16
Date of Performance:
Date of Submission:

Experiment No. 4

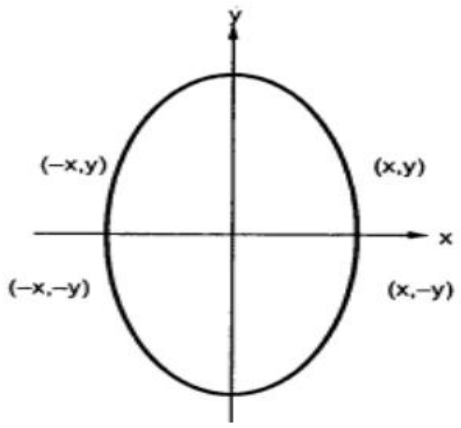
Aim-To implement midpoint Ellipse algorithm

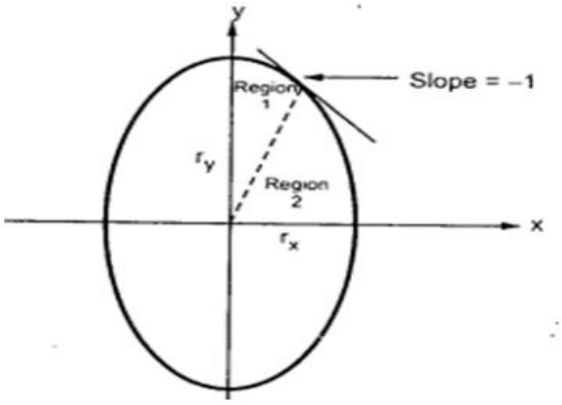
Objective:

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=$2a^2$ b [initial partial derivatives]
int p = b^2-a^2 b+a^2/4
while (fx<="" 1="" {="" set="" pixel="" (x,="" y)="" x++;="" fx="fx" +="" $2b^{2;}$

```
        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);
}
```

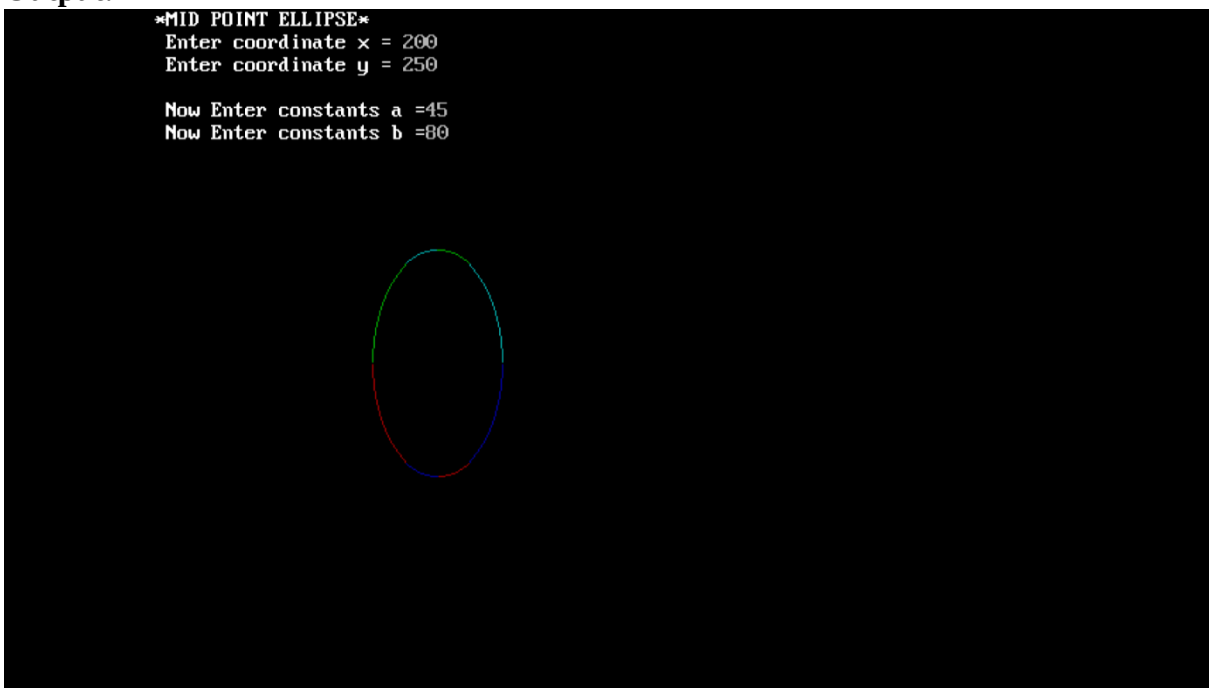
Program:

```
#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 technique employed for drawing ellipses distinguishes itself from circle drawing algorithms due to the inherent asymmetry of ellipses. The crux of the dissimilarity lies in the process of calculating and plotting points along the ellipse's contour, which entails adjusting both the horizontal and vertical radii as it traverses the curve. This is a stark contrast to circles, where the radius remains constant throughout.The significance of algorithms designed for drawing ellipses is underscored by their wide-ranging applicability in the real world. Ellipses feature prominently in fields like engineering, computer graphics, and mathematics, as they represent not only elementary geometric shapes but also a plethora of practical objects: from the wheels of vehicles to the orbits of celestial bodies and even the corneal shape of the human eye. Achieving precise rendering of ellipses is paramount for faithfully representing these objects in computer graphics, engineering blueprints, or scientific simulations. Thus, a robust and efficient ellipse drawing algorithm holds paramount importance, enabling the creation of realistic and meticulously accurate depictions of objects and phenomena across these domains.