|  |  |
| --- | --- |
| **Name:** | **RAHUL RAMMILAN YADAV** |
| **Roll No:** |  |
| **Class/Sem:**  **Experiment No.:**  **Title:**  **Date of**  **Performance:**  **Date of Submission:** | DSE/III  4  Midpoint Ellipse Drawing Algorithm |
|  |
| **Marks:** |  |
| **Sign of Faculty:** |  |

**Experiment No. 4**

Aim :Write a program to implement Midpoint Ellipse DrawingAlgorithm in C.

Objective : To implement midpoint ellipse drawing algorithm for drawing anellipse with radii rx and ry.

**Midpoint EllipseAlgorithm**

This is an incremental method for scan converting an ellipse that is centered at the origin in standard position i.e., with the major and minoraxis parallel to coordinate system axis. It is very similar to the midpointcircle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.

Let's first rewrite the ellipse equation and define the function f that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

Now divide the elliptical curve from (0, b) to (a, 0) into two parts atpoint Q where the slope of the curve is -1.

Slope of the curve is defined by the f(x, y) = 0 is where fx &fy are partial derivatives of f(x, y) with respect to x & y.

We have fx = 2b2 x, fy=2a2 y & Hence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is (0, b)

Suppose that the coordinates of the last scan converted pixel uponenteringstep i are(xi,yi). We areto select either T (xi+1),yi) or S (xi+1,yi-1)to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

pi = f(xi+1),yi-1/2)

pi=b2 (xi+1)2+a2 (yi-1/2)2-a2 b2

If pi<0, the midpoint is inside the curve and we choose pixel T.

If pi>0, the midpoint is outside or on the curve and we choose pixel S.

Decision parameter for the next step is:

pi+1=f(xi+1+1,yi+1-1/2)

= b2 (xi+1+1)2+a2 (yi+1-1/2)2-a2 b2 Since xi+1=xi+1,we have

pi+1-pi=b2[((xi+1+1)2+a2 (yi+1-1/2)2-(yi -1/2)2]

pi+1=pi+2b2 xi+1+b2+a2 [(yi+1-1/2)2-(yi -1/2)2] If T is chosen pixel (pi<0), we have yi+1=yi.

If S is chosen pixel (pi>0) we have yi+1=yi-1. Thus we can express pi+1in terms of pi and (xi+1,yi+1):

pi+1= pi+2b2 xi+1+b2 if pi<0

= pi+2b2 xi+1+b2-2a2 yi+1 if pi>0

The initial value for the recursive expression can be obtained by the evaluating the original definition of pi with (0, b):

p1 = (b2+a2 (b-1/2)2-a2 b2 = b2-a2 b+a2/4

Suppose the pixel (xj yj) has just been scan converted upon entering step j. The next pixel is either U (xj ,yj-1) or V (xj+1,yj-1). The midpointof the horizontal line connecting U & V is used to define the decision parameter: qj=f(xj+1/2,yj-1)

qj=b2 (xj+1/2)2+a2 (yj -1)2-a2 b2

If qj<0, the midpoint is inside the curve and we choose pixel V.

If qj≥0, the midpoint is outside the curve and we choose pixel U. Decision parameter for the next step is:

qj+1=f(xj+1+1/2,yj+1-1)

= b2 (xj+1+1/2)2+ a2 (yj+1-1)2- a2 b2 Since yj+1=yj-1,we have

qj+1-qj=b2 [(xj+1+1/2)2-(xj +1/2)2 ]+a2 (yj+1-1)2-( yj+1)2 ]

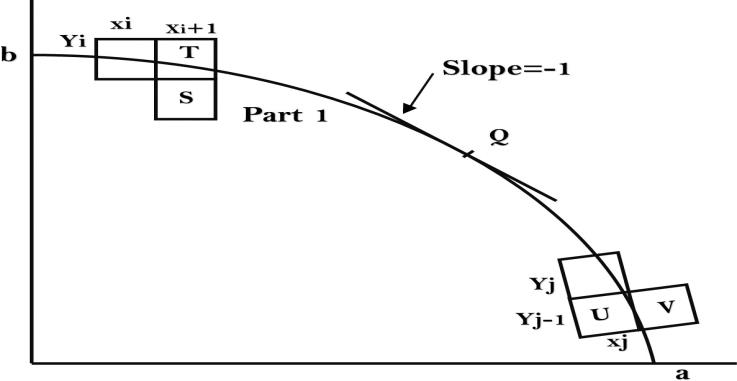
qj+1=qj+b2 [(xj+1+1/2)2-(xj +1/2)2]-2a2 yj+1+a2 If V is chosen pixel (qj<0), we have xj+1=xj.

If U is chosen pixel (pi>0) we have xj+1=xj. Thus we can express qj+1in terms of qj and (xj+1,yj+1 ):

qj+1=qj+2b2 xj+1-2a2 yj+1+a2 if qj < 0 =qj-2a2 yj+1+a2 if qj>0

The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of (xk yk) of the last pixel choosen for the part 1 of the curve:

q1 = f(xk+1/2,yk-1)=b2 (xk+1/2)2-a2 (yk-1)2- a2 b2



**Algorithm:**

int x=0, y=b; [starting point]

int fx=2x𝑏2, fy=2𝑎2y [initial partial derivatives]

int p = 𝑏2-𝑎2 b+𝑎2/4 while (fx<fy); Setpixel (x, y);

if (p<0) fx=fx+2𝑏2;

p = p + fx +𝑏2; else

{

y--; fx=fx+2𝑏2; fy=fy-2𝑎2;

p = p + fx +𝑏2-fy; }

x++;

p=𝑏2 (x+0.5)2+ 𝑎2 (y-1)2- 𝑎2 𝑏2 while (y>0)

Setpixel (x, y); {

y--;

fy=fy-2𝑎2; if (p>=0) p=p-fy+𝑎2 else

{

y--; x++;

fx=fx+2𝑏2 ; fy=fy-2𝑎2; p=p+fx-fy+𝑎2;} }

Code :

#include <stdio.h>

#include <graphics.h>

void drawEllipseMidpoint(int xc, int yc, int rx, int ry) {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, NULL);

    int x = 0, y = ry;

    int rx\_sq = rx \* rx;

    int ry\_sq = ry \* ry;

    int two\_rx\_sq = 2 \* rx\_sq;

    int two\_ry\_sq = 2 \* ry\_sq;

    int p;

    int px = 0, py = two\_rx\_sq \* y;

    putpixel(xc + x, yc - y, WHITE);

    putpixel(xc - x, yc - y, WHITE);

    putpixel(xc + x, yc + y, WHITE);

    putpixel(xc - x, yc + y, WHITE);

    p = round(ry\_sq - (rx\_sq \* ry) + (0.25 \* rx\_sq));

    while (px < py) {

        x++;

        px += two\_ry\_sq;

        if (p < 0)

            p += ry\_sq + px;

        else {

            y--;

            py -= two\_rx\_sq;

            p += ry\_sq + px - py;

        }

        putpixel(xc + x, yc - y, WHITE);

        putpixel(xc - x, yc - y, WHITE);

        putpixel(xc + x, yc + y, WHITE);

        putpixel(xc - x, yc + y, WHITE);

    }

    p = round(ry\_sq \* (x + 0.5) \* (x + 0.5) + rx\_sq \* (y - 1) \* (y - 1) - rx\_sq \* ry\_sq);

    while (y > 0) {

        y--;

        py -= two\_rx\_sq;

        if (p > 0)

            p += rx\_sq - py;

        else {

            x++;

            px += two\_ry\_sq;

            p += rx\_sq - py + px;

        }

        putpixel(xc + x, yc - y, WHITE);

        putpixel(xc - x, yc - y, WHITE);

        putpixel(xc + x, yc + y, WHITE);

        putpixel(xc - x, yc + y, WHITE);

    }

    delay(5000);

    closegraph();

}

int main() {

    int xc, yc, rx, ry;

    printf("Enter the center of the ellipse (xc, yc): ");

    scanf("%d %d", &xc, &yc);

    printf("Enter the major and minor radii (rx, ry): ");

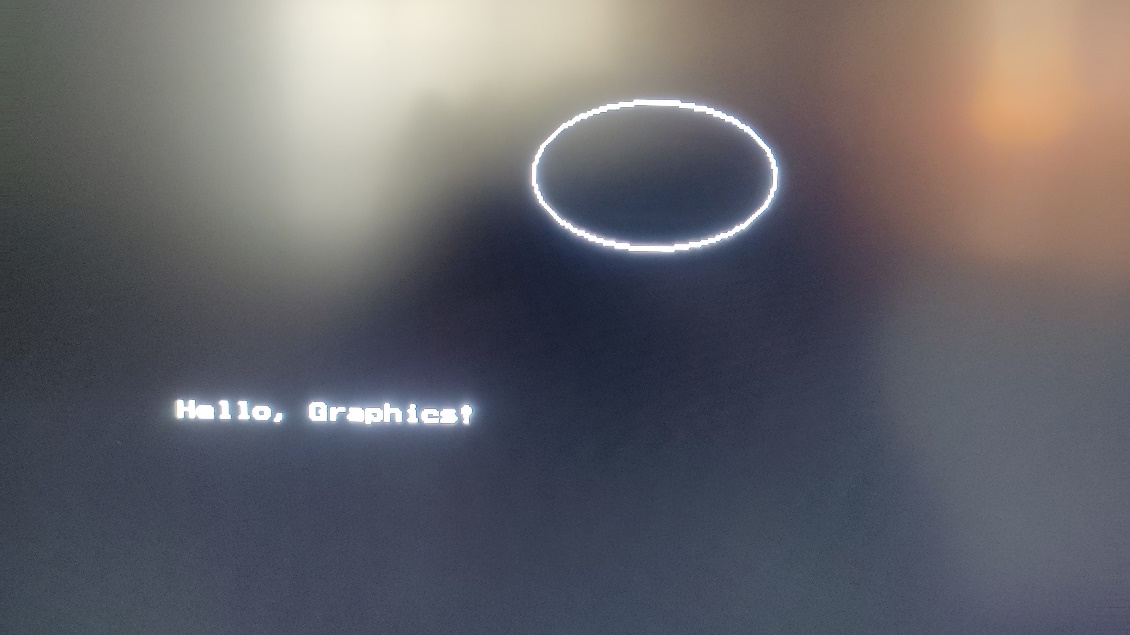
    scanf("%d %d", &rx, &ry);

    drawEllipseMidpoint(xc, yc, rx, ry);

    return 0;

}

**Output**



**Conclusion :**

The Midpoint Ellipse Drawing Algorithm is an efficient method for drawing ellipses, providing a balance between accuracy and computational complexity. The algorithm is based on selecting points along the ellipse using a decision parameter and updating it at each step to determine the next pixel to be plotted.

One notable advantage of the Midpoint Ellipse Drawing Algorithm is its ability to handle ellipses of various sizes and orientations without significant modifications. It is also relatively efficient compared to more straightforward algorithms, especially when dealing with large ellipses.

However, like many algorithms, the Midpoint Ellipse Drawing Algorithm has its limitations. It may not provide pixel-perfect results for all ellipses, especially when dealing with extreme aspect ratios. Additionally, the algorithm relies on floating-point arithmetic, which may be computationally expensive on systems with limited resources.