Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

Name:	MAHEK SHAH
Roll No:	
Class/Sem:	SE/III
Experiment No.:	4
Title:	Midpoint Ellipse Drawing Algorithm
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No. 4

Aim :Write a program to implement Midpoint Ellipse Drawing Algorithm 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 $\frac{dy}{dx} = -\frac{fx}{fy}$ where fx &fy are partial derivatives of f(x, y) with respect to x & y.

We have fx = 2b2 x, fy=2a2 y & $\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$ 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:

$p_{i+1}=f(x_{i+1}+1,y_{i+1}-1/2)$

$= b^2 (x_{i+1}+1)^2+a^2 (y_{i+1}-1/2)^2-a^2 b^2$ Since x_{i+1}=xi+1,we have

p_{i+1}-pi=$b^2[((x_{i+1}+1)^2+a^2 (y_{i+1}-1/2)^2-(y_i -1/2)^2]$

p_{i+1}= pi+$2b^2 x_{i+1}+b^2+a^2 [(y_{i+1}-1/2)^2-(y_i -$
$1/2)^2]$ If T is chosen pixel (pi<0), we have
y_{i+1}=yi.
If S is chosen pixel (pi>0) we have y_{i+1}=yi-1. Thus we can
express p_{i+1}in terms of pi and (x_{i+1},y_{i+1}):
p_{i+1}= pi+$2b^2 x_{i+1}+b^2$ if pi<0
= pi+$2b^2 x_{i+1}+b^2-2a^2 y_{i+1}$ 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 = $(b^2+a^2 (b-$
$1/2)^2-a^2 b^2 = b^2-a^2$
$b+a^2/4$
Suppose the pixel $(x_j\ y_j)$ has just been scan converted upon entering step j. The next pixel is either U (x_j ,y_j-1) or V (x_j+1,y_j-1). The midpointof the horizontal line connecting U & V is used to define the decision parameter: $q_j=f(x_j+1/2,y_j-1)$

$q_j=b^2 (x_j+1/2)^2+a^2 (y_j -1)^2-a^2 b^2$
If $q_j<0$, the midpoint is inside the curve and we choose pixel V.
If $q_j\geq0$, the midpoint is outside the curve and we choose pixel U. Decision parameter for the next step is:

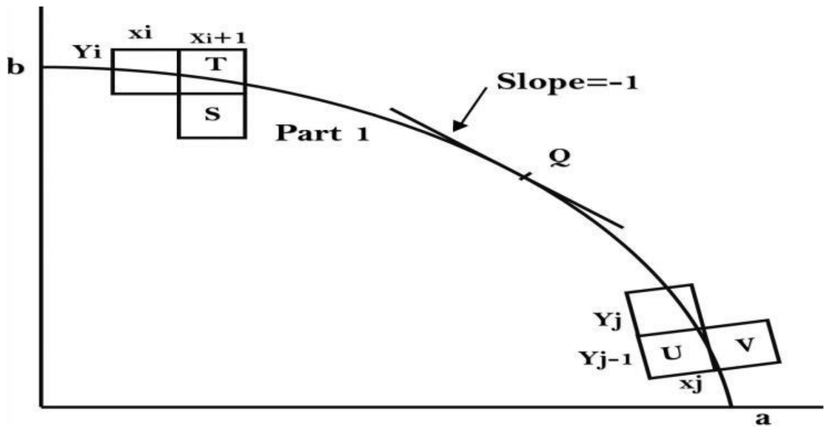
$q_{j+1}=f(x_{j+1}+1/2,y_{j+1}-1)$

$= b^2 (x_{j+1}+1/2)^2+ a^2 (y_{j+1}-$
$1)^2- a^2 b^2$ Since $y_{j+1}=y_j-1$,we have

$q_{j+1}-q_j=b^2 [(x_{j+1}+1/2)^2-(x_j +1/2)^2]+a^2 (y_{j+1}-1)^2-(y_{j+1})^2]$

$q_{j+1}=q_j+b^2 [(x_{j+1}+1/2)^2-(x_j +1/2)^2]-2a^2$
$y_{j+1}+a^2$ If V is chosen pixel (qj<0), we
have x_{j+1}=xj.
If U is chosen pixel (pi>0) we have x_{j+1}=xj. Thus we can express q_{j+1}in terms of q_j and (x_{j+1},y_{j+1}):
$q_{j+1}=q_j+2b^2 x_{j+1}-2a^2 y_{j+1}+a^2$ if
qj < 0 =$q_j-2a^2 y_{j+1}+a^2$ if qj>0
The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of $(x_k\ y_k)$ of the last pixel choosen for the part 1 of the curve:
q1 = $f(x_k+1/2,y_k-1)=b^2 (x_k+1/2)^2-a^2 (y_k-1)^2- a^2 b^2$

$$f(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{bmatrix} < 0\,(x,y)\text{inside} \\ 0\,(x,y)\text{on} \\ > 0\,(x,y)\text{outside} \end{bmatrix}$$



Algorithm:

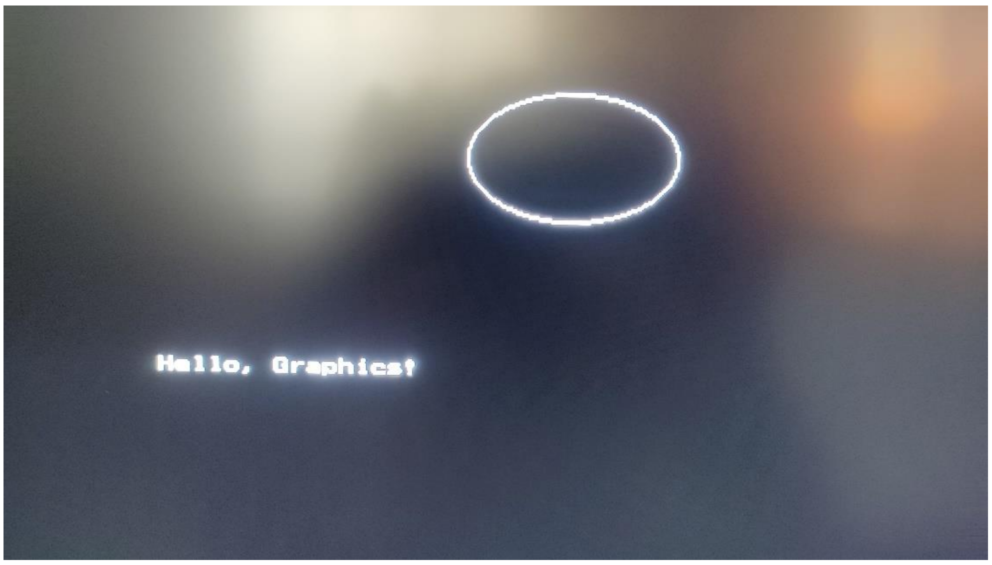
```
int x=0, y=b; [starting point]
int fx=2x𝑏2, fy=2𝑎2y [initial partial derivatives]
int p = 𝑏2-𝑎2 b+𝑎2/4 while (fx<fy); Setpixel (x, y);
if (p<0) fx=fx+2𝑏2;
p = p + fx +𝑏2; else
{
y--; fx=fx+2𝑏2; fy=fy-2𝑎2;
p = p + fx +𝑏2-fy; }
x++;

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.