



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

<b>Name:</b>	TANVI MANGESH SURVE
<b>Roll No:</b>	
<b>Class/Sem:</b>	DSE/III
<b>Experiment No.:</b>	4
<b>Title:</b>	Midpoint Ellipse Drawing Algorithm
<b>Date of Performance:</b>	
<b>Date of Submission:</b>	
<b>Marks:</b>	
<b>Sign of Faculty:</b>	



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

### Experiment No. 4

**Aim :** Write a program to implement Midpoint Ellipse Drawing Algorithm in C.

**Objective :** To implement midpoint ellipse drawing algorithm for drawing an ellipse with radii  $r_x$  and  $r_y$ .

#### **Midpoint Ellipse Algorithm**

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 minor axis parallel to coordinate system axis. It is very similar to the midpoint circle 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 at point  $Q$  where the slope of the curve is  $-1$ .

$$\frac{dy}{dx} = -\frac{f_x}{f_y}$$

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

$$\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$$

We have  $f_x = 2b^2x$ ,  $f_y = 2a^2y$  & 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 upon entering step  $i$  are  $(x_i, y_i)$ . We are to select either  $T(x_{i+1}, y_i)$  or  $S(x_{i+1}, y_{i-1})$  to be the next pixel. The midpoint of  $T$  &  $S$  is used to define the following decision parameter.

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

$$p_i = b^2(x_{i+1})^2 + a^2(y_{i-1/2})^2 - a^2b^2$$

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

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

Decision parameter for the next step is:

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

$$= b^2(x_{i+1}+1)^2 + a^2(y_{i+1-1/2})^2 - a^2b^2$$

Since  $x_{i+1} = x_i + 1$ , we have



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

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

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

If T is chosen pixel ( $p_i < 0$ ), we have  $y_{i+1}=y_i$ .

If S is chosen pixel ( $p_i > 0$ ) we have  $y_{i+1}=y_i-1$ . Thus we can express  $p_{i+1}$  in terms of  $p_i$  and  $(x_{i+1}, y_{i+1})$ :

$$p_{i+1}=p_i+2b^2x_{i+1}+b^2 \quad \text{if } p_i < 0$$
$$=p_i+2b^2x_{i+1}+b^2-2a^2y_{i+1} \quad \text{if } p_i > 0$$

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

$$p_1 = (b^2+a^2(b-1/2)^2-a^2b^2) = b^2-a^2b+a^2/4$$

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 midpoint of the horizontal line connecting U & V is used to define the decision parameter:  $q_j=f(x_j+1/2, y_j-1)$

$$q_j=b^2(x_j+1/2)^2+a^2(y_j-1)^2-a^2b^2$$

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

If  $q_j \geq 0$ , the midpoint is outside the curve and we choose pixel U. Decision parameter for the next step is:

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

Since  $y_{j+1}=y_j-1$ , we have

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

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

If V is chosen pixel ( $q_j < 0$ ), we have  $x_{j+1}=x_j$ .

If U is chosen pixel ( $p_i > 0$ ) we have  $x_{j+1}=x_j$ . Thus we can express  $q_{j+1}$  in terms of  $q_j$  and  $(x_{j+1}, y_{j+1})$ :

$$q_{j+1}=q_j+2b^2x_{j+1}-2a^2y_{j+1}+a^2 \quad \text{if } q_j < 0$$
$$=q_j-2a^2y_{j+1}+a^2 \quad \text{if } q_j > 0$$

The initial value for the recursive expression is computed using the original definition of  $q_j$ . And the coordinates of  $(x_k, y_k)$  of the last pixel chosen for the part 1 of the curve:

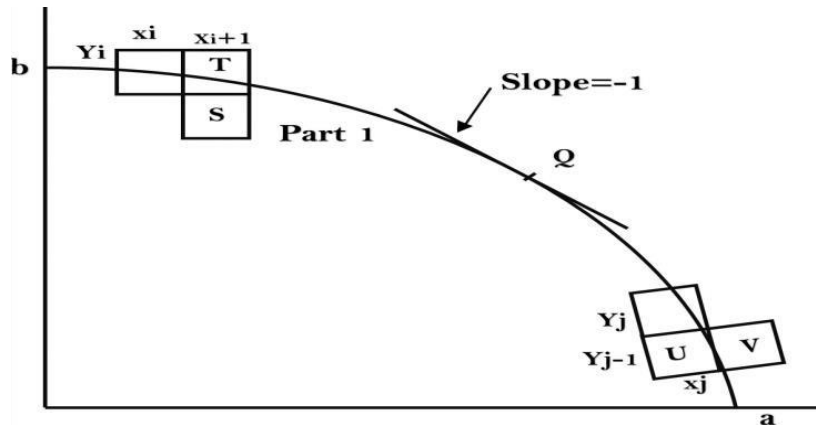
$$q_1 = f(x_k+1/2, y_k-1)=b^2(x_k+1/2)^2-a^2(y_k-1)^2-a^2b^2$$



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

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



### 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;
        }
    }
}
```

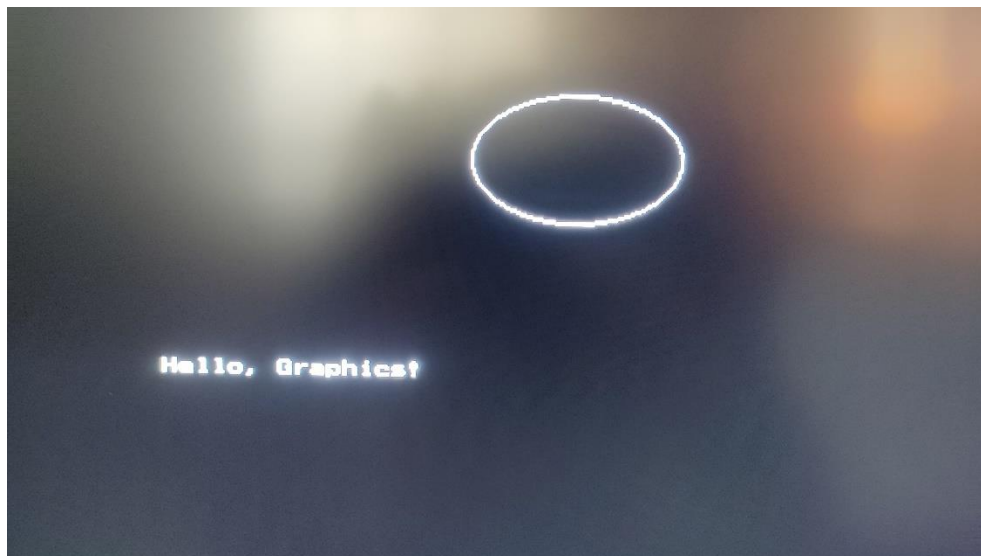


# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

```
    }  
    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





# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

### **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.