Midpoint Ellipse Algorithm

Midpoint ellipse algorithm is a method for drawing ellipses in computer graphics. This method is modified from Bresenham’s algorithm. The advantage of this modified method is that only addition operations are required in the program loops. This leads to simple and fast implementation in all processors.

Let us consider one quarter of an ellipse. The curve is divided into two regions. In region I, the slope on the curve is greater than –1 while in region II less than –1.

Consider the general equation of an ellipse,
\[ b^2x^2 + a^2y^2 - a^2b^2 = 0 \]
where \( a \) is the horizontal radius and \( b \) is the vertical radius, we can define an function \( f(x,y) \) by which the error due to a prediction coordinate \((x,y)\) can be obtained. The appropriate pixels can be selected according to the error so that the required ellipse is formed. The error can be confined within half a pixel.

Set \( f(x,y) = b^2x^2 + a^2y^2 - a^2b^2 \)

In region I \((dy/dx > -1)\),

\[ (x_{k+1}, y_{k}) \]

\[ E \]

Prediction \((x_k+1, y_k-\frac{1}{2})\)

Region I

\[ x \text{ is always incremented in each step, i.e. } x_{k+1} = x_k + 1. \]

\[ y_{k+1} = y_k \text{ if E is selected, or } y_{k+1} = y_k - 1 \text{ if SE is selected.} \]

In order to make decision between S and SE, a prediction \((x_k+1, y_k-\frac{1}{2})\) is set at the middle between the two candidate pixels. A prediction function \( P_k \) can be defined as follows:
\[ P_k = f(x_{k+1}, y_{k-1/2}) \]
\[ = b^2(x_{k+1})^2 + a^2(y_{k-1/2})^2 - a^2b^2 \]
\[ = b^2(x_k^2 + 2x_k + 1) + a^2(y_k^2 - y_k + 1/4) - a^2b^2 \]

If \( P_k < 0 \), select E:

\[ P_{k+1}^E = f(x_{k+2}, y_{k-1/2}) \]
\[ = b^2(x_{k+2})^2 + a^2(y_{k-1/2})^2 - a^2b^2 \]
\[ = b^2(x_k^2 + 4x_k + 4) + a^2(y_k^2 - y_k + 1/4) - a^2b^2 \]

Change of \( P_k \) is: \( \Delta P_k^E = P_{k+1}^E - P_k = b^2(2x_k + 2) \)

If \( P_k > 0 \), select SE:

\[ P_{k+1}^{SE} = f(x_{k+2}, y_{k-3/2}) \]
\[ = b^2(x_{k+2})^2 + a^2(y_{k-3/2})^2 - a^2b^2 \]
\[ = b^2(x_k^2 + 4x_k + 4) + a^2(y_k^2 - 3y_k + 9/4) - a^2b^2 \]

Change of \( P_k \) is \( \Delta P_k^{SE} = P_{k+1}^{SE} - P_k = b^2(2x_k + 3) - 2a^2(y_k - 1) \)

Calculate the changes of \( \Delta P_k \):

If E is selected,
\[ \Delta P_{k+1}^E = b^2(2x_k + 5) \]
\[ \Delta^2 P_k^E = \Delta P_{k+1}^E - \Delta P_k^E = 2b^2 \]
\[ \Delta P_{k+1}^{SE} = b^2(2x_k + 5) - 2a^2(y_k - 1) \]
\[ \Delta^2 P_k^{SE} = \Delta P_{k+1}^{SE} - \Delta P_k^{SE} = 2b^2 \]

If SE is selected,
\[ \Delta P_{k+1}^E = b^2(2x_k + 5) \]
\[ \Delta^2 P_k^E = \Delta P_{k+1}^E - \Delta P_k^E = 2b^2 \]
\[ \Delta P_{k+1}^{SE} = b^2(2x_k + 5) - 2a^2(y_k - 2) \]
\[ \Delta^2 P_k^{SE} = \Delta P_{k+1}^{SE} - \Delta P_k^{SE} = 2(a^2 + b^2) \]

Initial values:
\[ x_0 = 0, \; y_0 = b, \; P_0 = b^2 + 1/4a^2(1 - 4b) \]
\[ \Delta P_0^E = 3b^2, \; \Delta P_0^{SE} = 3b^2 - 2a^2(b - 1) \]

In region II (\( dy/dx < -1 \)), all calculations are similar to that in region I except that \( y \) is decremented in each step.
Mid-point Ellipse Algorithm

Prediction

\((x_{k+1/2}, y_{k-1})\)

Region II

\(y\) is always decremented in each step, i.e. \(y_{k+1} = y_k - 1\).

\(x_{k+1} = x_k\) if \(S\) is selected, or \(x_{k+1} = x_k + 1\) if \(SE\) is selected.

\[ P_k = f(x_{k+1/2}, y_{k-1}) \]

\[ = b^2(x_{k+1/2})^2 + a^2(y_{k-1})^2 - a^2b^2 \]

\[ = b^2(x_k^2 + x_k + 1/4) + a^2(y_k^2 - 2y_k + 1) - a^2b^2 \]

If \(P_k > 0\), select \(S\):

\[ P_{k+1}^S = f(x_{k+1/2}, y_{k-2}) \]

\[ = b^2(x_{k+1/2})^2 + a^2(y_{k-2})^2 - a^2b^2 \]

\[ = b^2(x_k^2 + x_k + 1/4) + a^2(y_k^2 - 4y_k + 4) - a^2b^2 \]

Change of \(P_k^S\) is: \(\Delta P_k^S = P_{k+1}^S - P_k = a^2(3 - 2y_k)\)

If \(P_k < 0\), select \(SE\):

\[ P_{k+1}^{SE} = f(x_{k+3/2}, y_{k-2}) \]

\[ = b^2(x_{k+3/2})^2 + a^2(y_{k-2})^2 - a^2b^2 \]

\[ = b^2(x_k^2 + 3x_k + 9/4) + a^2(y_k^2 - 4y_k + 4) - a^2b^2 \]

Change of \(P_k^{SE}\) is \(\Delta P_k^{SE} = P_{k+1}^{SE} - P_k = 2b^2(x_k + 1) + a^2(3 - 2y_k)\)

Calculate the changes of \(\Delta P_k^S\):

If \(S\) is selected,

\[ \Delta P_{k+1}^S = a^2(5 - 2y_k) \]

\[ \Delta^2 P_k^S = \Delta P_{k+1}^S - \Delta P_k^S = 2a^2 \]

\[ \Delta P_{k+1}^{SE} = 2b^2(x_k + 1) + a^2(5 - 2y_k) \]

\[ \Delta^2 P_k^{SE} = \Delta P_{k+1}^{SE} - \Delta P_k^{SE} = 2a^2 \]

If \(SE\) is selected,

\[ \Delta P_{k+1}^S = a^2(5 - 2y_k) \]

\[ \Delta^2 P_k^S = \Delta P_{k+1}^S - \Delta P_k^S = 2a^2 \]

\[ \Delta P_{k+1}^{SE} = 2b^2(2x_k + 2) - a^2(5 - 2y_k) \]
\[ \Delta^2 P_k^{SE} = \Delta P_{k+1}^{SE} - \Delta^2 P_k^{SE} = 2(a^2 + b^2) \]

Determine the boundary between region I and II:

Set \( f(x, y) = 0 \),

\[ \frac{dy}{dx} = \frac{-bx}{a^2 \sqrt{1 - x^2 / a^2}}. \]

When \( \frac{dy}{dx} = -1 \), \( x = \frac{a^2}{\sqrt{a^2 + b^2}} \) and \( y = \frac{b^2}{\sqrt{a^2 + b^2}} \).

At region I, \( \frac{dy}{dx} > -1 \), \( x < \frac{a^2}{\sqrt{a^2 + b^2}} \) and \( y > \frac{b^2}{\sqrt{a^2 + b^2}} \), therefore

\[ \Delta P_k^{SE} < b^2 \left( \frac{2a^2}{\sqrt{a^2 + b^2}} + 3 \right) - 2a^2 \left( \frac{b^2}{\sqrt{a^2 + b^2}} - 1 \right) = 2a^2 + 3b^2. \]

Initial values at region II:

\[ x_0 = \frac{a^2}{\sqrt{a^2 + b^2}} \quad \text{and} \quad y_0 = \frac{b^2}{\sqrt{a^2 + b^2}} \]

\( x_0 \) and \( y_0 \) will be the accumulative results from region I at the boundary. It is not necessary to calculate them from values of \( a \) and \( b \).

\[ P_0 = P_1^{I} - \frac{1}{4}[a^2(4y_0 - 3) + b^2(4x_0 + 3)] \]

where \( P_1^{I} \) is the accumulative result from region I at the boundary.

\[ \Delta P_0^{E} = b^2(2x_0 + 3) \]

\[ \Delta P_0^{SE} = 2a^2 + 3b^2 \]

Implementation of the algorithm:

The algorithm described above shows how to obtain the pixel coordinates in the first quarter only. The ellipse centre is assumed to be at the origin. In actual implementation, the pixel coordinates in other quarters can be simply obtained by use of the symmetric characteristics of an ellipse. For a pixel \((x, y)\) in the first quarter, the corresponding pixels in other three quarters are \((x, -y)\), \((-x, y)\) and \((-x, -y)\) respectively. If the centre is at \((x_C, y_C)\), all calculated coordinate \((x, y)\) should be adjusted by adding the offset \((x_C, y_C)\). For easy implementation, a function `PlotEllipse()` is defined as follows:

```cpp
PlotEllipse(x_C, y_C, x, y)
    putpixel(x_C+x, y_C+y)
    putpixel(x_C-x, y_C+y)
    putpixel(x_C+x, y_C-y)
    putpixel(x_C-x, y_C-y)
end PlotEllipse
```

The function to draw an ellipse is described in the following pseudo-codes:

```cpp
DrawEllipse(x_C, y_C, a, b)
    Declare integers x, y, P, \( \Delta P^E, \Delta P^S, \Delta^2 P^E, \Delta^2 P^S \) and \( \Delta^2 P^{SE} \)
    // Set initial values in region I
```
Set \( x = 0 \) and \( y = b \)

\[ P = b^2 + (a^2(1 - 4b) - 2)/4 \]  // Intentionally \(-2\) to round off the value

\[ \Delta P^E = 3b^2 \]

\[ \Delta P^E = 2b^2 \]

\[ \Delta P^SE = \Delta P^E - 2a^2(b - 1) \]

\[ \Delta P^S^2 = \Delta P^E + 2a^2 \]

// Plot the pixels in region I

```
PlotEllipse(xc, yc, x, y)
```

while \( \Delta P^SE < 2a^2 + 3b^2 \)

if \( P < 0 \) then  // Select E

\[ P = P + \Delta P^E \]

\[ \Delta P^E = \Delta P^E + \Delta^2 P^E \]

\[ \Delta P^SE = \Delta P^SE + \Delta^2 P^E \]

else  // Select SE

\[ P = P + \Delta P^SE \]

\[ \Delta P^E = \Delta P^E + \Delta^2 P^E \]

\[ \Delta P^SE = \Delta P^SE + \Delta^2 P^SE \]

end if

increment x

```
PlotEllipse(xc, yc, x, y)
```

end while

// Set initial values in region II

\[ P = P - (a^2(4y - 3) + b^2(4x + 3) + 2)/4 \]  // Intentionally \(+2\) to round off the value

\[ \Delta P^S = a^2(3 - 2y) \]

\[ \Delta P^SE = 2b^2 + 3a^2 \]

\[ \Delta P^S^2 = 2a^2 \]

// Plot the pixels in region II

while \( y > 0 \)

if \( P > 0 \) then  // Select S

\[ P = P + \Delta P^S \]

\[ \Delta P^E = \Delta P^E + \Delta^2 P^S \]

\[ \Delta P^SE = \Delta P^SE + \Delta^2 P^S \]

else  // Select SE

\[ P = P + \Delta P^SE \]

\[ \Delta P^E = \Delta P^E + \Delta^2 P^S \]

\[ \Delta P^SE = \Delta P^SE + \Delta^2 P^SE \]

end if

decrement y

```
PlotEllipse(xc, yc, x, y)
```

end while

end

end DrawEllipse