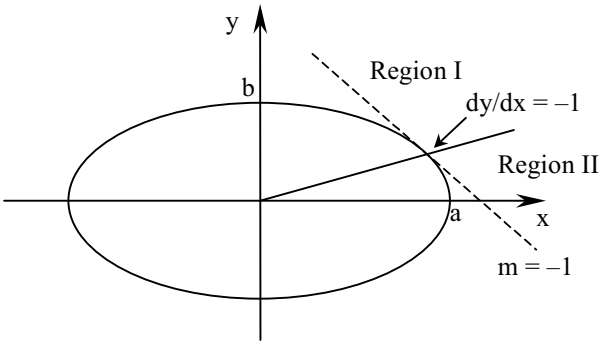
Midpoint Ellipse Algorithm

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.

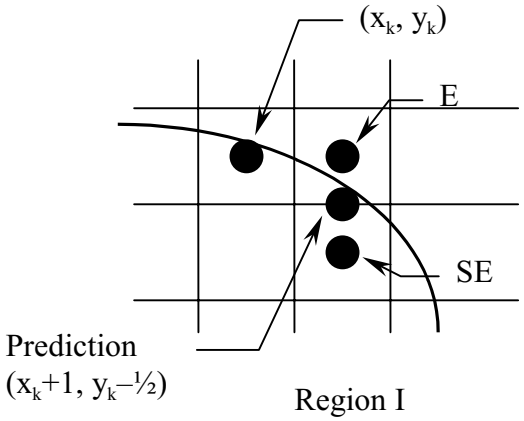
Let us consider one quarter of an ellipse. The curve is divided into two regions. In region I, the slope on the curve is greater than –1 while in region II less than –1.



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 is always incremented in each step, i.e. $x_{k+1} = x_k + 1$.
$y_{k+1} = y_k$ if E is selected, or $y_{k+1} = y_k – 1$ if SE is selected.
In order to make decision between S and SE, a prediction $(x_k+1, y_k–½)$ is set at

the middle between the two candidate pixels. A prediction function P_k can be defined as follows:

$$\begin{aligned} P_k &= f(x_k+1, y_k-½) \\ &= b^2(x_k+1)^2 + a^2(y_k-½)^2 - a^2b^2 \\ &= b^2(x_k^2 + 2x_k + 1) + a^2(y_k^2 - y_k + ¼) - a^2b^2 \end{aligned}$$

If $P_k < 0$, select E :

$$\begin{aligned} P_{k+1}^{E} &= f(x_k+2, y_k-½) \\ &= b^2(x_k+2)^2 + a^2(y_k-½)^2 - a^2b^2 \\ &= b^2(x_k^2 + 4x_k + 4) + a^2(y_k^2 - y_k + ¼) - a^2b^2 \end{aligned}$$

Change of P_k^E is: $\Delta P_k^E = P_{k+1}^E - P_k = b^2(2x_k + 3)$

If Pk > 0, select SE :

$$\begin{aligned} P_{k+1}^{SE} &= f(x_k+2, y_k-3/2) \\ &= b^2(x_k+2)^2 + a^2(y_k-3/2)^2 - a^2b^2 \\ &= b^2(x_k^2 + 4x_k + 4) + a^2(y_k^2 - 3y_k + 9/4) - a^2b^2 \end{aligned}$$

Change of P_k^{SE} 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 P_k:

If E is selected,
$\Delta P_{k+1}^E = b^2(2x_k + 5)$
$\Delta^2 P_k^E = \Delta P_{k+1}^E - \Delta P_k^E = 2b^2$

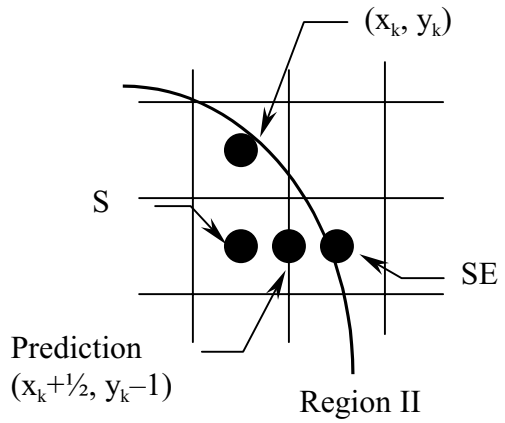
$\Delta P_{k+1}^{SE} = b^2(2x_k + 5) - 2a^2(y_k - 1)$
$\Delta^2 P_k^{SE} = \Delta P_{k+1}^{SE} - \Delta P_k^{SE} = 2b^2$

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:
$x_0 = 0$, $y_0 = b$, $P_0 = b^2 + ¼a^2(1 - 4b)$
$\Delta P_0^E = 3b^2$, $\Delta P_0^{SE} = 3b^2 - 2a^2(b - 1)$

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



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.

$$\begin{aligned} P_k &= f(x_k+½, y_k-1) \\ &= b^2(x_k+½)^2 + a^2(y_k-1)^2 - a^2b^2 \\ &= b^2(x_k^2 + x_k + ¼) + a^2(y_k^2 - 2y_k + 1) - a^2b^2 \end{aligned}$$

If $P_k > 0$, select S :

$$\begin{aligned} P_{k+1}^S &= f(x_k+½, y_k-2) \\ &= b^2(x_k+½)^2 + a^2(y_k-2)^2 - a^2b^2 \\ &= b^2(x_k^2 + x_k + ¼) + a^2(y_k^2 - 4y_k + 4) - a^2b^2 \end{aligned}$$

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

If Pk < 0, select SE :

$$\begin{aligned} P_{k+1}^{SE} &= f(x_k+3/2, y_k-2) \\ &= b^2(x_k+3/2)^2 + a^2(y_k-2)^2 - a^2b^2 \\ &= b^2(x_k^2 + 3x_k + 9/4) + a^2(y_k^2 - 4y_k + 4) - a^2b^2 \end{aligned}$$

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 ΔP_k:

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,
$\Delta P_{k+1}^S = a^2(5 - 2y_k)$
$\Delta^2 P_k^S = \Delta P_{k+1}^S - \Delta P_k^S = 2a^2$

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

Determine the boundary between region I and II:

Set f(x, y) = 0, dy/dx = $-bx / [a^2\sqrt{(1 - x^2/a^2)}]$.
When dy/dx = –1, $x = a^2\sqrt{(a^2 + b^2)}$ and $y = b^2\sqrt{(a^2 + b^2)}$.

At region I, dy/dx > –1, $x < a^2\sqrt{(a^2 + b^2)}$ and $y > b^2\sqrt{(a^2 + b^2)}$, therefore
$\Delta P_k^{SE} < b^2(2a^2/(a^2 + b^2) - 2a^2(b^2/(a^2 + b^2) - 1) = 2a^2 + 3b^2$

Initial values at region II:
$x_0 = a^2\sqrt{(a^2 + b^2)}$
$y_0 = b^2\sqrt{(a^2 + b^2)}$
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.
$P_0 = P_k^I - ¼[a^2(4y_0 - 3) + b^2(4x_0 + 3)]$ where P_k^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:

```
PlotEllipse (xC, yC, x, y)
    putpixel(xC+x, yC+y)
    putpixel(xC+x, yC–y)
    putpixel(xC–x, yC+y)
    putpixel(xC–x, yC–y)
end PlotEllipse
```

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

```
DrawEllipse (xC, yC, a, b)
    Declare integers x, y, P, ΔP^E, ΔP^S, ΔP^SE, Δ²P^E, Δ²P^S and Δ²P^SE
    // Set initial values in region I
    Set x = 0 and y = b
    P = b² + (a²(1 – 4b) – 2)/4        // Intentionally –2 to round off the value
    ΔP^E = 3b²
    Δ²P^E = 2b²
```

```
        ΔP^SE = ΔP^E – 2a^2(b – 1)
        Δ^2P^SE = Δ^2P^E + 2a^2
        // Plot the pixels in region I
        PlotEllipse(x_C, y_C, x, y)
        while ΔP^SE < 2a^2 + 3b^2
            if P < 0 then     // Select E
                P = P + ΔP^E
                ΔP^E = ΔP^E + Δ^2P^E
                ΔP^SE = ΔP^SE + Δ^2P^E
            else          // Select SE
                P = P + ΔP^SE
                ΔP^E = ΔP^E + Δ^2P^E
                ΔP^SE = ΔP^SE + Δ^2P^SE
                decrement y
            end if
            increment x
            PlotEllipse(x_C, y_C, 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
        ΔP^S = a^2(3 – 2y)
        ΔP^SE = 2b^2 + 3a^2
        Δ^2P^S = 2a^2
        // Plot the pixels in region II
        while y > 0
            if P > 0 then     // Select S
                P = P + ΔP^E
                ΔP^E = ΔP^E + Δ^2P^S
                ΔP^SE = ΔP^SE + Δ^2P^S
            else          // Select SE
                P = P + ΔP^SE
                ΔP^E = ΔP^E + Δ^2P^S
                ΔP^SE = ΔP^SE + Δ^2P^SE
                increment x
            end if
            decrement y
            PlotEllipse(x_C, y_C, x, y)
        end while
    end DrawEllipse
```