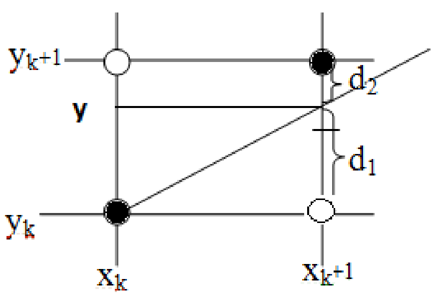
Bresenham's Line Drawing Algorithm for Lines with Slope <=1

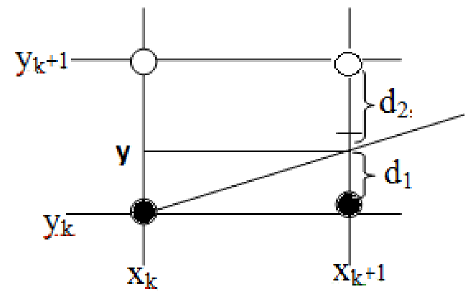
For the line with slope less than equal to one, the pixel positions are determined by sampling at unit 'x' interval i.e. $x_{k+1} = x_k + 1$, with the starting pixel at (x_0, y_0) from left hand.

For any k[th] step, assuming position (x_k , y_k) has been selected at previous step, we determine next position (x_{k+1} , y_{k+1}) as either $(x_k + 1 , y_k)$ or $(x_k + 1 , y_k + 1)$

At x_{k+1} label vertical pixel separations from ideal line path as d_1 and d_2, 'y' coordinate at x_{k+1} will be

$y = m (x_k+1) + c$





The distance of lower pixel from the ideal location $d_1 = y - y_k$ or $d_1 = m (x_k+1) + c - y_k$

The distance of the ideal location from the upper pixel $d_2 = y_k +1 - y$ or $d_2 = y_k +1 - m (x_k+1) - c$

Thus the difference between the separations of two pixel positions from the actual line path,

$d_1 - d_2 = m (x_k+1) + c - y_k - y_k -1 + m (x_k+1) + c$

$d_1 - d_2 = 2m (x_k+1) + 2c - 2y_k - 1$

Substituting $m = \Delta y/\Delta x$, we get

$\Delta x (d_1 - d_2) = 2 \Delta y. x_k + 2\Delta y + \Delta x.2c - \Delta x.2y_k - \Delta x$

$P_k = \Delta x.(d_1 - d_2) = 2 \Delta y. x_k - 2\Delta x. y_k + b$ …. (i) where $b = 2\Delta y + \Delta x.2c - \Delta x$ and P_k is the decision parameter at the k[th] step

At the k+1[th] step

$P_{k+1} = 2 \Delta y.x_{k+1} - 2\Delta x.y_{k+1} + b$ …. (ii)

Now subtracting (i) and (ii)

$P_{k+1} = P_k + 2\Delta y. (x_{k+1} - x_k) - 2\Delta x. (y_{k+1} - y_k)$

Since the slope of the line is less than one, we sample in 'x' direction i.e. $x_{k+1} - x_k = 1$ so,

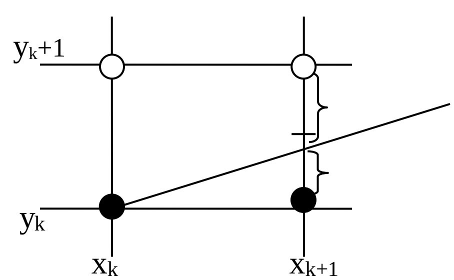
$P_{k+1} = P_k + 2\Delta y - 2\Delta x. (y_{k+1} - y_k)$ …. (iii)

Case 1:

if $P_k <= 0$ then the pixel on scanline 'y_k' is closer to the line path and $y_{k+1} = y_k$

i.e. from equation (iii)

$\mathbf{P_{k+1} = P_k + 2\Delta y}$

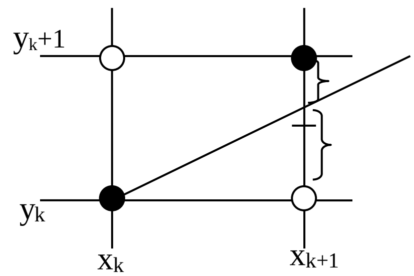


Case 2:

if $P_k > 0$ then the pixel on scanline '$y_k + 1$' is closer to the line path and $y_{k+1} = y_k + 1$

i.e. from equation (iii)

$$\mathbf{P_{k+1} = P_k + 2\Delta y - 2\Delta x}$$



The Initial Decision Parameter P_0 = ?

We have,

$d_1 - d_2 = 2m(x_k+1) + 2c - 2y_k - 1$

if the line passes thru (x_0, y_0) then

$d_1 - d_2 = 2m(x_0+1) + 2c - 2y_0 - 1$

$= 2mx_0 + 2c - 2y_0 + 2m - 1$

or $d_1 - d_2 = 2m - 1$ since, $2mx_0 + 2c - 2y_0 = 0$

or $\mathbf{P_0 = \Delta x\,(d_1 - d_2) = 2\Delta y - \Delta x}$

Algorithm

For |m| <=1

```
i.    Read xa,ya,xb,yb (Assume -1 <= m <=1)
ii.   Load (x0, y0) into the frame buffer (i.e. plot the first point)
iii.  Calculate constants Δy,Δx, 2Δy and 2Δy - 2Δx
      Obtain the first decision parameter p0 = 2Δy - Δx
iv.   At each xk along the line starting at k = 0 perform
      the following tests:
      If pk < 0 then the next point to plot is (xk +1 , yk) and
             pk+1 = pk + 2Δy
      else the next point to plot is (xk +1 , yk+1) and
            pk+1 = pk + 2Δy - 2Δx
v.      Repeat step iv Δx times
```

Here first we initialize the decision parameter and set the first pixel. Next, during each iteration, we increment 'x' to the next horizontal position, then use the current value of the decision parameter to select the bottom or top pixel (`increment y`) and update the decision parameter and at the end set the chosen pixel.

Advantages

It is a faster incremental algorithm that makes use of integer arithmetic calculations only , avoids floating point computations

So it uses faster operations such as addition/subtraction and bit shifting

CPU intensive rounding off operations are avoided

Disadvantages

It is used for drawing basic lines, antialiasing is not a part of this algorithm so for drawing smooth lines it is not suitable

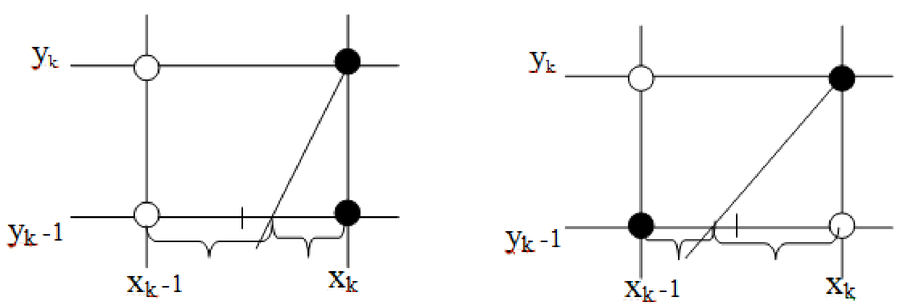
Bresenham's Line Drawing Algorithm for Lines with Positive slope, Magnitude of the Slope Greater Than One and Moving from Right to Left

For the line with slope greater than equal to one and negative slope, the pixel positions are determined by sampling at unit 'y' interval i.e. $y_{k+1} = y_k + 1$, with the starting pixel at (x_0, y_0) from right hand.

For any k^{th} step, assuming position (x_k , y_k) has been selected at previous step, we determine next position (x_{k+1} , y_{k+1}) as either (x_k, y_k-1) or $(x_k -1, y_k - 1)$

At x_{k+1} label vertical pixel separations from ideal line path as d_1 and d_2, 'y' coordinate at x_{k+1} will be

$(y_k -1)= m\, x_k + c$



The distance of right pixel from the ideal location $d_1 = x - x_k$ or $d_1 = ((y_k-1) + c)/m - x_k$

The distance of the ideal location from the left pixel $d_2 = x_k -1 - x$ or $d_2 = x_k -1 - ((y_k-1) - c)/m$

Thus the difference between the separations of two pixel positions from the actual line path,

$d_1 - d_2 = ((y_k-1) - c)/m - x_k - x_k +1 + ((y_k-1) - c)/m$

$d_1 - d_2 = 2 ((y_k-1) - c)/m - 2x_k + 1$

$d_1 - d_2 = 2\, \Delta x\, y_k- 2\, \Delta x - 2c\, \Delta x - \Delta y\, 2x_k + \Delta y$

Substituting $m = \Delta y/\Delta x$, we get

$\Delta y\, (d_1 - d_2) = 2.\Delta x.\, y_k - 2.\Delta y.x_k + \Delta y - 2\Delta x - \Delta x.2c$

$P_k = \Delta y.(d_1 - d_2) = 2.\Delta x.\ y_k - 2.\Delta y.x_k + b$ …. (i) where $b = -2\Delta x + \Delta x.2c + \Delta y$ and P_k is the decision parameter at the k^{th} step

At the $k+1^{th}$ step

$P_{k+1} = 2.\Delta x.\ y_{k+1} - 2.\Delta y.x_{k+1} + b$ …. (ii)

Now subtracting (i) and (ii)

$P_{k+1} = P_k + 2\Delta x.\ (y_{k+1} - y_k) - 2\Delta y.\ (x_{k+1} - x_k)$

Since the slope of the line is greater than one, we sample in decreasing 'y' direction i.e. $y_{k+1} - y_k = -1$ so,

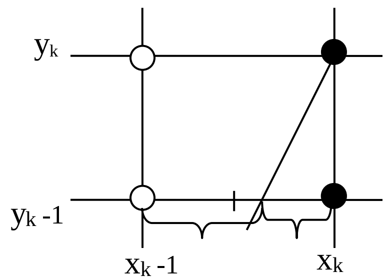
$P_{k+1} = P_k - 2\Delta x - 2\Delta y\ (x_{k+1} - x_k)$ …. (iii)

Case 1:

if $P_k < 0$ then the pixel on scanline 'x_k' is closer to the line path and $x_{k+1} = x_k$

i.e. from equation (iii)

$\mathbf{P_{k+1} = P_k - 2\Delta x}$

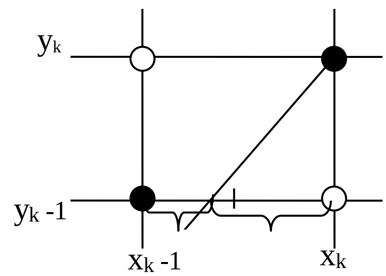


Case 2:

if $P_k >= 0$ then the pixel on scanline 'x_k - 1' is closer to the line path and $x_{k+1} = x_k - 1$

i.e. from equation (iii)

$\mathbf{P_{k+1} = P_k - 2\Delta x + 2\Delta y}$



The Initial Decision Parameter $P_0 = ?$

We have,

$d_1 - d_2 = 2\ ((y_k\text{-}1) - c)/m - 2x_k + 1$

if the line passes thru (x_0, y_0) then

$d_1 - d_2 = 2\ ((y_0\text{-}1) - c)/m - 2x_0 + 1$

$= 2y_0/m - 2/m - c/m - 2x_0 + 1$

or $d_1 - d_2 = -2/m + 1$ since, $2y_0/m - c/m - 2x_0 = 0$

or $\mathbf{P_0 = \Delta y\ (d_1 - d_2) = -\ 2\Delta x + \Delta y}$

Digitize a line with end points A(2,6) B(4,9)

Digitize a line with end points A(-2,-6) B(-4,-9)