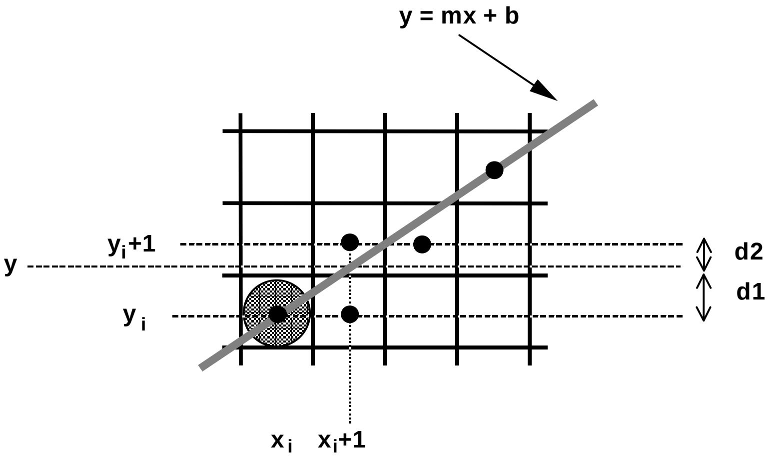
DERIVATION OF THE BRESENHAM'S LINE ALGORITHM

Assumptions:
- input: line endpoints at (X1,Y1) and (X2, Y2)
- X1 < X2
- line slope ≤ 45^o, i.e. $0 < m \le 1$
- x coordinate is incremented in steps of 1, y coordinate is computed
- generic line equation: $y = mx + b$



Derivation

Assume that we already have a location of pixel (x_i, y_i) and have plotted it. The question is, what is the location of the next pixel.

Geometric location of the line at x-coordinate $x_{i+1} = x_i + 1$ is:

$$y = m(x_i + 1) + b \quad (1)$$

where:

$$m = \Delta y / \Delta x \text{ (slope)} \quad (2)$$

b – intercept

$$\Delta x = X2 - X1 \text{ (from the assumption above that } X1 < X2 \text{)} \quad (3)$$

$$\Delta y = Y2 - Y1$$

Define:

$$d1 = y - y_i = m(x_i + 1) + b - y_i$$

$$d2 = (y_i + 1) - y = y_i + 1 - m(x_i + 1) - b$$

Calculate:

$$d1 - d2 = m(x_i + 1) + b - y_i - y_i - 1 + m(x_i + 1) + b$$

$$= 2m(x_i + 1) - 2y_i + 2b - 1 \quad (4)$$

if $d1 - d2 < 0$ then $y_{i+1} \leftarrow y_i$ (5)

if $d1 - d2 > 0$ then $y_{i+1} \leftarrow y_i + 1$ (6)

We want integer calculations in the loop, but m is not an integer. Looking at definition of m ($m = \Delta y / \Delta x$) we see that if we multiply m by Δx, we shall remove the denominator and hence the floating point number.

For this purpose, let us multiply the difference (d1 - d2) by Δx and call it p_i:

$$p_i = \Delta x(d1 - d2)$$

The sign of p_i is the same as the sign of d1 – d2, because of the assumption (3).

Expand p_i:

$$\begin{aligned}
p_i &= \Delta x(d1 - d2) & \\
&= \Delta x[\, 2m(x_i + 1) - 2y_i + 2b - 1 \,] & \text{from (4)} \\
&= \Delta x[\, 2 \cdot (\Delta y / \Delta x) \cdot (x_i + 1) - 2y_i + 2b - 1 \,] & \text{from (2)} \\
&= 2\cdot\Delta y\cdot (x_i + 1) - 2\cdot\Delta x\cdot y_i + 2\cdot\Delta x\cdot b - \Delta x & \text{result of multiplication by } \Delta x \\
&= 2\cdot\Delta y\cdot x_i + 2\cdot\Delta y - 2\cdot\Delta x\cdot y_i + 2\cdot\Delta x\cdot b - \Delta x & \\
&= 2\cdot\Delta y\cdot x_i - 2\cdot\Delta x\cdot y_i + \underline{2\cdot\Delta y + 2\cdot\Delta x\cdot b - \Delta x} & \text{(7)}
\end{aligned}$$

Note that the underlined part is constant (it does not change during iteration), we call it c, i.e.

$$c = 2\cdot\Delta y + 2\cdot\Delta x\cdot b - \Delta x$$

Hence we can write an expression for p_i as:

$$p_i \quad = 2\cdot\Delta y\cdot x_i - 2\cdot\Delta x\cdot y_i + c \qquad (8)$$

Because the sign of p_i is the same as the sign of d1 – d2, we could use it inside the loop to decide whether to select pixel at $(x_i + 1, y_i)$ or at $(x_i + 1, y_i +1)$. Note that the loop will only include integer arithmetic. There are now 6 multiplications, two additions and one selection in each turn of the loop.

However, we can do better than this, by defining p_i recursively.

$$\begin{aligned}
p_{i+1} &= 2\cdot\Delta y\cdot x_{i+1} - 2\cdot\Delta x\cdot y_{i+1} + c & \text{from (8)} \\
p_{i+1} - p_i &= 2\cdot\Delta y\cdot x_{i+1} - 2\cdot\Delta x\cdot y_{i+1} + c & \\
&\quad - (2\cdot\Delta y\cdot x_i - 2\cdot\Delta x\cdot y_i + c) & \\
&= 2\Delta y \cdot (x_{i+1} - x_i) - 2\Delta x \cdot (y_{i+1} - y_i) & x_{i+1} - x_i = 1 \text{ always}
\end{aligned}$$

$$p_{i+1} - p_i = 2\Delta y - 2\Delta x \cdot (y_{i+1} - y_i)$$

Recursive definition for p_i:

$$p_{i+1} = p_i + 2\Delta y - 2\Delta x \cdot \underline{(y_{i+1} - y_i)}$$

If you now recall the way we construct the line pixel by pixel, you will realise that the underlined expression: $y_{i+1} - y_i$ can be either 0 (when the next pixel is plotted at the same y-coordinate, i.e. d1 – d2 < 0 from (5)); or 1 (when the next pixel is plotted at the next y-coordinate, i.e. d1 – d2 > 0 from (6)). Therefore the final recursive definition for p_i will be based on choice, as follows (remember that the sign of p_i is the same as the sign of d1 – d2):

if $p_i < 0$, $p_{i+1} = p_i + 2\Delta y$ because $2\Delta x \cdot (y_{i+1} - y_i) = 0$
if $p_i > 0$, $p_{i+1} = p_i + 2\Delta y - 2\Delta x$ because $(y_{i+1} - y_i) = 1$

At this stage the basic algorithm is defined. We only need to calculate the initial value for parameter p_o.

$$p_i = 2\cdot\Delta y\cdot x_i - 2\cdot\Delta x\cdot y_i + 2\cdot\Delta y + 2\cdot\Delta x\cdot b - \Delta x \quad \text{from (7)}$$

$$p_0 = 2\cdot\Delta y\cdot x_0 - 2\cdot\Delta x\cdot y_0 + 2\cdot\Delta y + 2\Delta x\cdot b - \Delta x \quad (9)$$

For the initial point on the line:

$$y_0 = mx_0 + b$$

therefore

$$b = y_0 - (\Delta y/\Delta x) \cdot x_0$$

Substituting the above for b in (9)we get:

$$\begin{aligned}
p_0 &= 2\cdot\Delta y\cdot x_0 - 2\cdot\Delta x\cdot y_0 + 2\cdot\Delta y + 2\Delta x\cdot [\, y_0 - (\Delta y/\Delta x) \cdot x_0 \,] - \Delta x \\
&= 2\cdot\Delta y\cdot x_0 - 2\cdot\Delta x\cdot y_0 + 2\cdot\Delta y + 2\Delta x\cdot y_0 - 2\Delta x\cdot (\Delta y/\Delta x) \cdot x_0 - \Delta x && \text{simplify} \\
&= 2\cdot\Delta y\cdot x_0 - 2\cdot\Delta x\cdot y_0 + 2\cdot\Delta y + 2\Delta x\cdot y_0 - 2\Delta y\cdot x_0 - \Delta x && \text{regroup} \\
&= 2\cdot\Delta y\cdot x_0 - 2\Delta y\cdot x_0 - 2\cdot\Delta x\cdot y_0 + 2\Delta x\cdot y_0 + 2\cdot\Delta y - \Delta x && \text{simplify} \\
&= 2\cdot\Delta y - \Delta x
\end{aligned}$$

We can now write an outline of the complete algorithm.

Algorithm

1. Input line endpoints, (X1,Y1) and (X2, Y2)
2. Calculate constants:
   $\Delta x = X2 - X1$
   $\Delta y = Y2 - Y1$
   $2\Delta y$
   $2\Delta y - \Delta x$
3. Assign value to the starting parameters:
   $k = 0$
   $p_0 = 2\Delta y - \Delta x$
4. Plot the pixel at ((X1,Y1)
5. For each integer x-coordinate, x_k, along the line
   if $p_k < 0$ plot pixel at ($x_k + 1$, y_k)
   $p_{k+1} = p_k + 2\Delta y$ (note that $2\Delta y$ is a pre-computed constant)

   else plot pixel at ($x_k + 1$, $y_k + 1$)
   $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
   (note that $2\Delta y - 2\Delta x$ is a pre-computed constant)

   increment k

   while $x_k < X2$