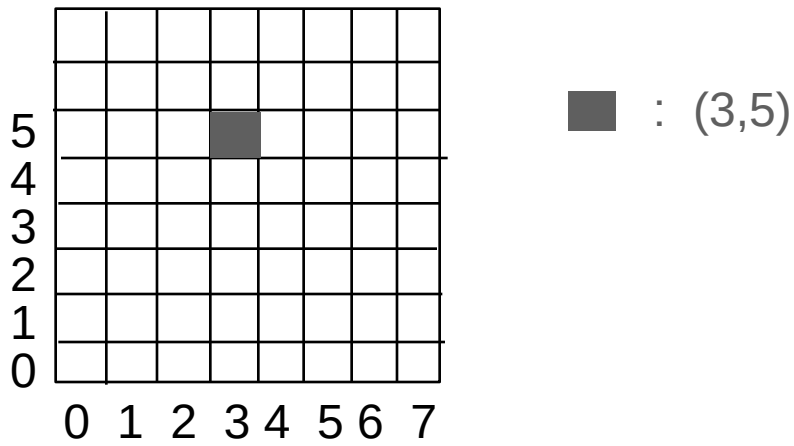
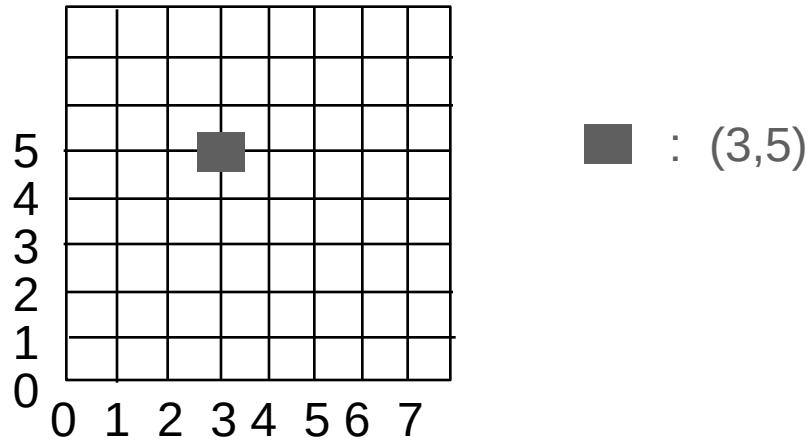
Line and Circle Rasterization Algorithms

2D screen coordinate systems (0−based)

(1) pixels as grid cells

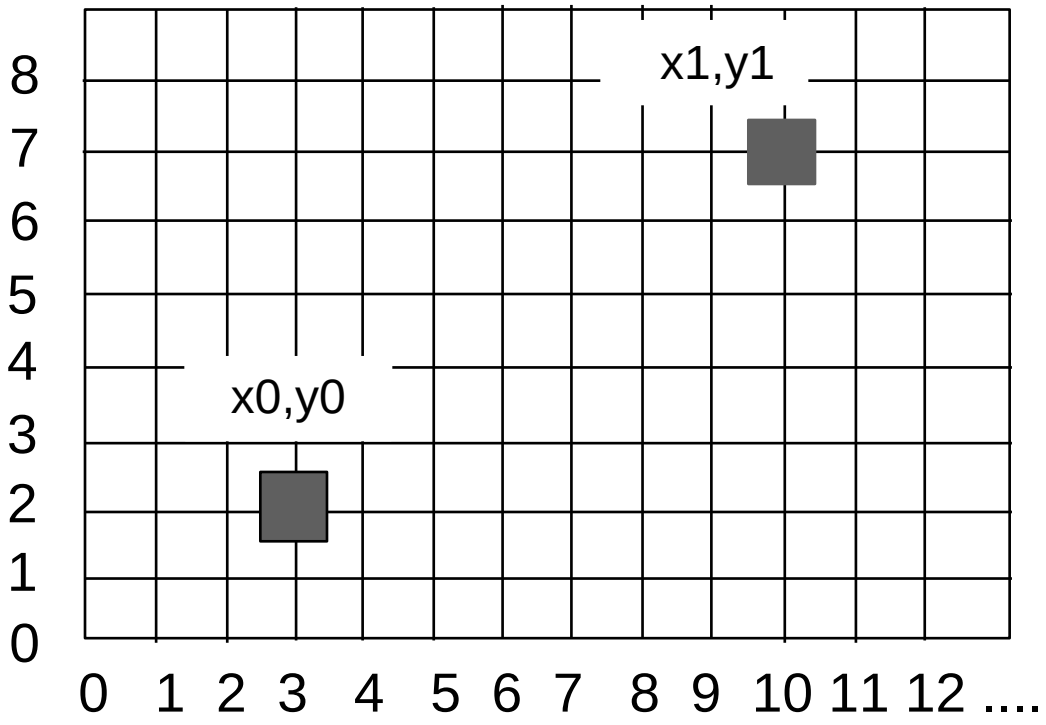


√ (2) pixels as intersection points

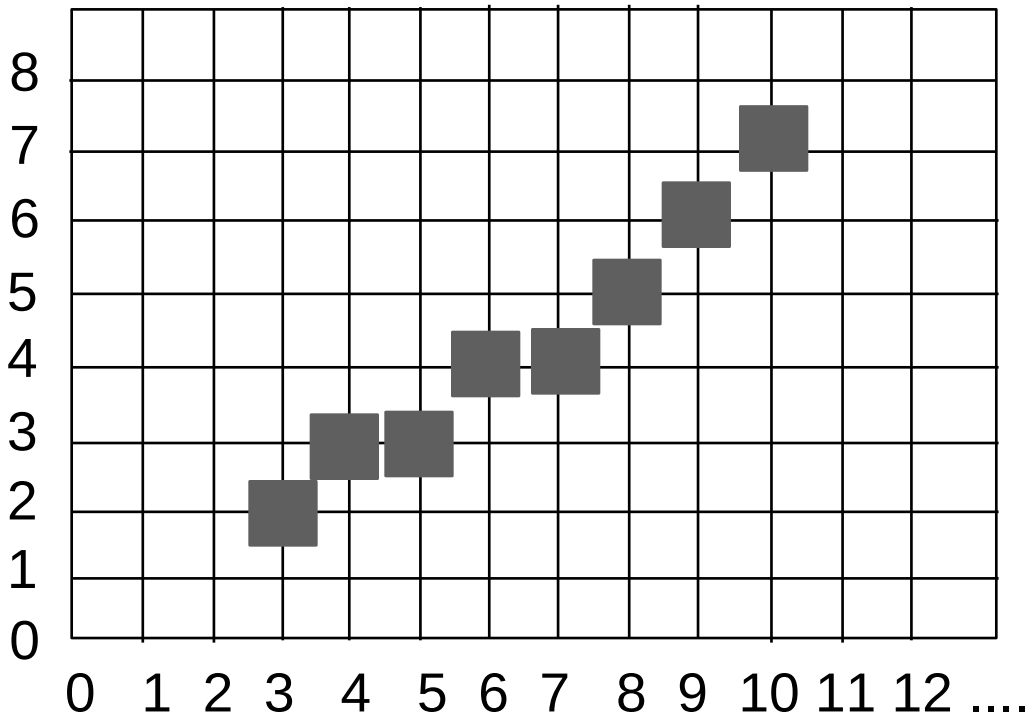


We use convention (2)

Line−Drawing Algorithm



rasterize_line_segment(x0, y0, x1, y1);



DDA Line−Drawing Algorithm

Slope−Intercept Line Equation

$$y = \underset{\text{slope}}{m} \cdot x + \underset{\text{intercept}}{b}$$

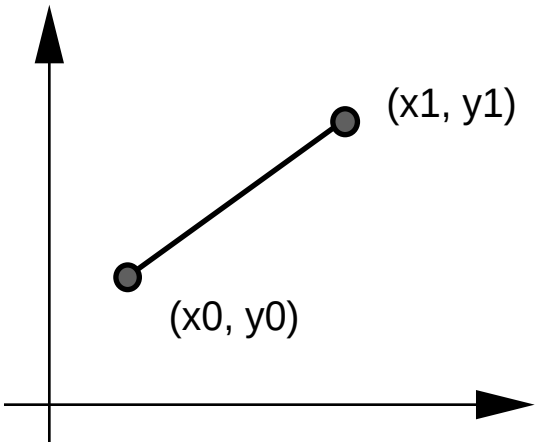
For a given pair of points (x0,y0) and (x1,y1), let
dx = x1−x0, and dy = y1−y0

we can compute the slope as:

m = (y1−y0)/(x1−x0) = dy / dx

and

intercept b = y1 − m*x1



Given the slope, we have:

dy = m . dx

also

dx = dy . (1/m)

We can use the above equation to compute
a sequence of discrete points along the line
segment:

DDA Line−Drawing Algorithm (cont'd)

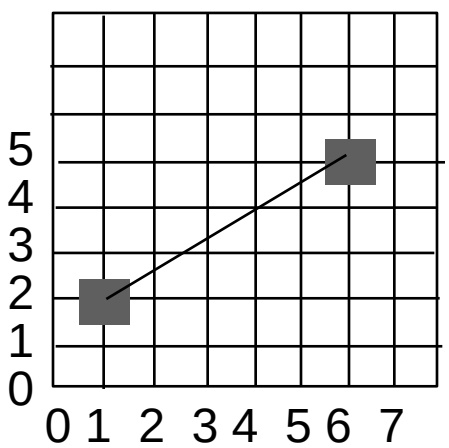
Starting from X = x0 Y = y0

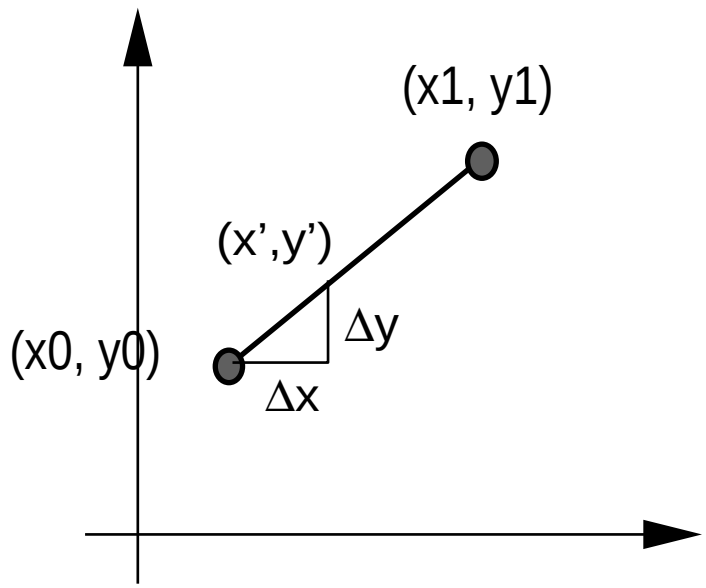
If we choose the next point's x value at:

X' = X + ΔX

Then we have:

Y' = Y + ??





Now consider the screen (discrete) space ...

our goal: finding the closest pixels (integer x and y) along the line path

To do this, we can set Δx = 1, and we will have:

$X_1 = x_0 + 1$ → ... $X_k = X_{k-1} + 1$

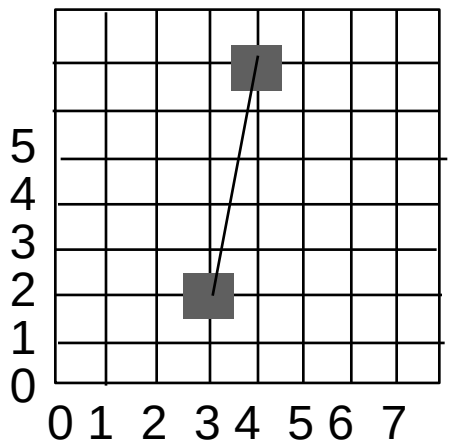
$Y_1 = y_0 + m$ → ... $Y_k = Y_{k-1} + m$

↓ ↓

plot (X_1, int(Y_1)) ... plot (X_k, int(Y_k))

DDA Line−Drawing Algorithm (cont'd)

What about a line looks like this ?



Can we still use $\Delta X = 1$ and compute y ?

No! Because if we do so, we will end up with .....

Instead, we should increment Y by 1 in this case:

Set $\Delta y = 1$, and we will have:

$X_1 = x_0 + 1/m$ → ... $X_k = X_{k-1} + 1/m$

$Y_1 = y_0 + 1$ → ... $Y_k = Y_{k-1} + 1$

↓ ↓

plot (int(X_1), Y_1) ... plot (int(X_k),Y_k)

DDA Line−Drawing Algorithm (cont'd)

The above algorithm is called DDA

(Digital Differential Analyzer) algorithm because

it is based on ΔX and ΔY

So far in our discussions the slope m is greater

than 0, what to do if m is negative?

Read page 87−88 in the textbook

DDA Algorithm has two problems:

1) Numerical errors (could be bad for long line segments)

2) Floating point operations −− Too slow

Bresenham's Line Drawing Algorithm

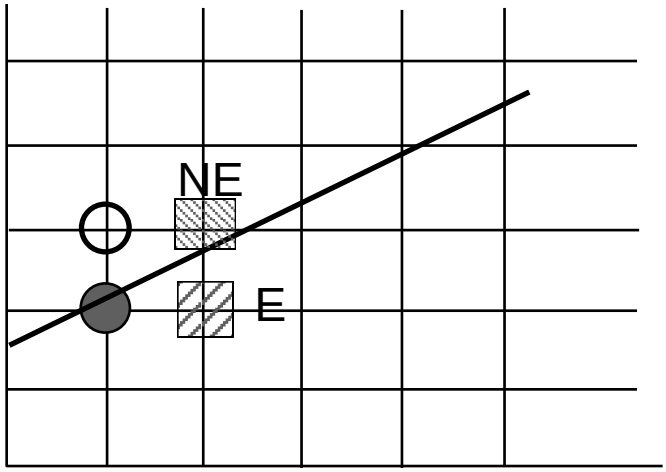
Only incremental integer operations are used

In the case of $0 < m < 1$,

we sample the line at unit X intervals, what we

really need to decide is −−

Should we increment Y now?



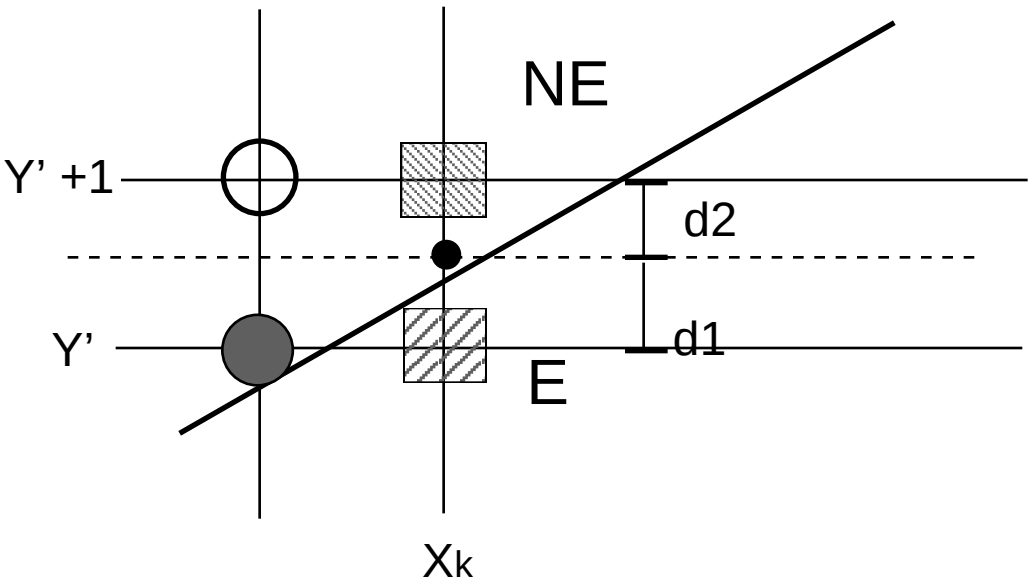
: pixel has been drawn

: draw northeast pixel?

: draw east pixel?

Bresenham's Line Drawing Algorithm (cont'd)

The key idea:



Should we increment Y?

If d1 > d2 → Use NE pixel (y = Y'+1)

if d2 > d1 → Use E pixel (y = Y')

Can we compute and compare d1 and d2
using pure integer operations ?

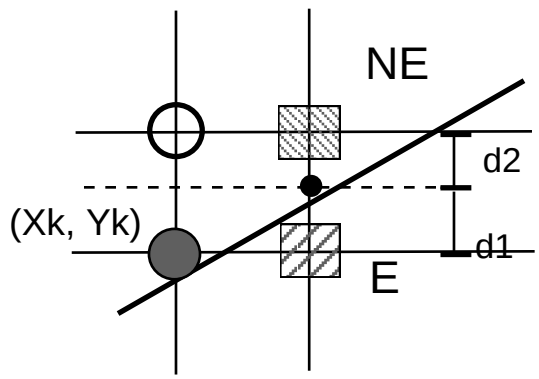
Bresenham's Line Drawing Algorithm (cont'd)

Assuming that the k−th pixels (X_k, Y_k) along the line has been drawn, and $0<m<1$:

Now consider the k+1 pixel (X,Y)

We know: $X = X_k + 1$

We want to decide whether



$$\boxed{\begin{array}{lll} Y = Y_k & (d2 > d1) & \text{or} \\ Y = Y_k + 1 & (d1 > d2) & \end{array}}$$

Remember the line equation: $Y = m*X + b$

so we have $Y = m (X_k + 1) + b$

$$\boxed{\begin{array}{l} d1 = Y - Y_k = m (X_k + 1) + b - Y_k \\ d2 = (Y_k + 1) - Y = Y_k + 1 - m (X_k + 1) - b \end{array}}$$

$d1 - d2 = 2 m (X_k + 1) - 2 Y_k + 2 b - 1$

Substitute $m = dy / dx$

$d1 - d2 = 2 (dy / dx) (X_k + 1) - 2 Y_k + 2 b - 1$

$dx (d1-d2) = 2dy (X_k + 1) - 2 dx Y_k +$

$dx (2b-1)$

let $2 dy + dx (2b-1) = c$ We have:

$dx (d1-d2) = 2dy . X_k - 2 dx . Y_k + C$

Bresenham's Line Drawing Algorithm (cont'd)

$dx\,(d1-d2) \quad = \quad 2dy\,.\,X_k \;-\; 2\,dx\,.\,Y_k + C$

Remember $C = 2\,dy + dx\,(\,2b-1)$ and

$dy = y1 - y0$ (y distance of two end points)
$dx = x1 - x0$ (x distance of two end points)

Let $\boxed{P_k = dx\,(d1 - d2)}$

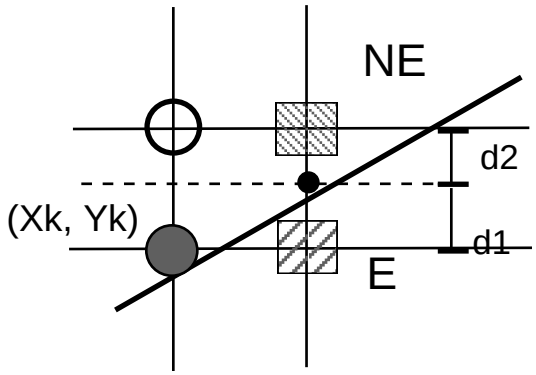
If we only consider the $dx > 0$ (draw the line from left to right):

then P_k has the same sign as $(d1 - d2)$

That is:

if $P_k < 0 \Rightarrow d1 < d2 \Rightarrow Y_{k+1} = Y_k$ (take the E Pixel)

else $d1 > d2 \Rightarrow Y_{k+1} = Y_k + 1$ (take the NE pixel)



Bresenham's Line Drawing Algorithm (cont'd)

Given P_k, we can compute P_{k+1} incrementally:

Consider

$P_k = 2\,dy \,.\, X_k - 2\,dx \,.\, Y_k + C$

We have:

$P_{k+1} = 2\,dy \,.\, X_{k+1} - 2\,dx \,.\, Y_{k+1} + C$ where

(X_{k+1}, Y_{k+1}) is the (k+1)th pixel

$$P_{k+1} - P_k = 2\,dy\,(X_{k+1} - X_k) - 2\,dx\,(Y_{k+1} - Y_k)$$

$$= 2\,dy - 2\,dx\,(Y_{k+1} - Y_k)$$

So,

$$\boxed{P_{k+1} = P_k + 2\,dy - 2\,dx \,.\, (Y_{k+1} - Y_k)}$$

The initial condition P_0:

$$P_0 = 2\,dy.\,X0 - 2\,dx \,.\, Y0 + 2\,dy$$

$$+ dx\,(2\,Y0 - 2\,dy/dx \,.\, X0 - 1)$$

$$= 2dy - dx$$

Bresenham's Line Algorithm

Given end points (x_0,y_0) (x_1,y_1)

$dx = x_1 - x_0$, $dy = y_1 - y_0$

Starting with an end point (x_0, y_0):

1. Compute $P_0 = 2dy - dx$

2. For each K, starting with k = 0

   if ($P_k < 0$)

   the next point is $(X_k + 1, Y_k)$ // E pixel

   $P_{k+1} = P_k + 2\,dy$

   else

   the next point is $(X_k + 1, Y_k + 1)$ // NE pixel

   $P_{k+1} = P_k + 2(dy - dx)$

3 Repeat step 2 $x_1 - x_0$ times

Ask yourself ...

In the class, we assumed that $0 < m < 1$ and $dx > 0$, where m is the slope of the line segment and $dx = x_1 - x_0$

Describe how we should modify the algorithm at the bottom of page 90 in the textbook for an arbitrary slope m and dx ?

(hint: read page 91, 92)