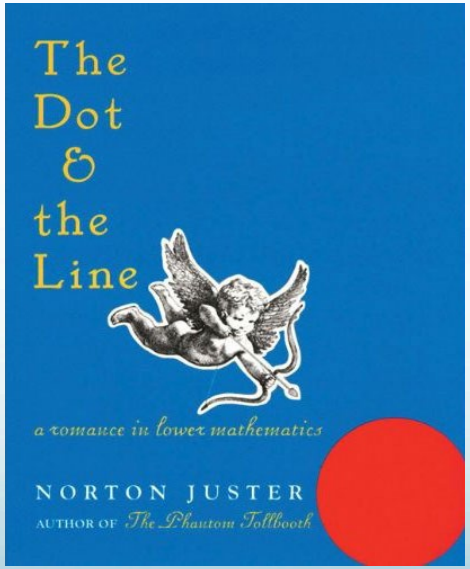
CSC 240 Computer Graphics
Lecture 2: Line Algorithms

Nick Howe
Smith College

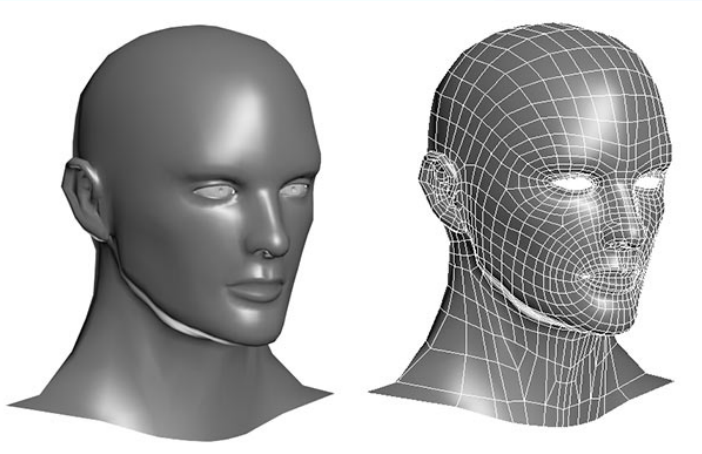
Portions based on slides & content courtesy Sara Mathieson

Line Algorithm



Why Lines?

- Even 3D shapes are drawn eventually in 2D
- 3D shapes made of polygons
- Each polygon made of lines
- Compute endpoints & draw in 2D



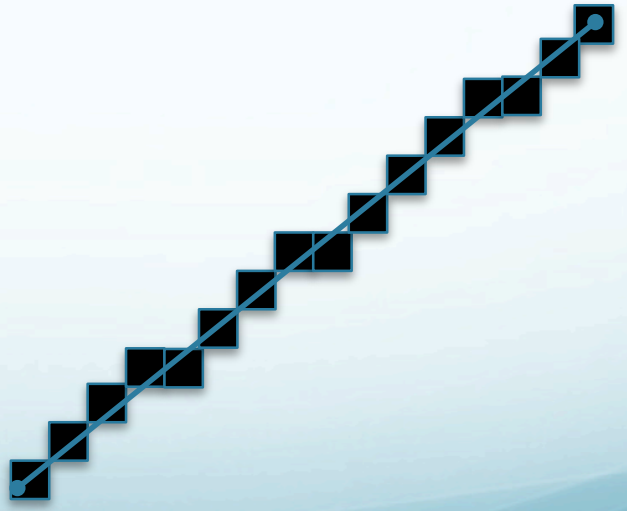
☞ Given endpoints (x_1,y_1) and (x_2,y_2), what pixels should be colored to draw the line segment connecting them?

Image: http://socialsharing.info/polygons-in-3d-modelling/polygon-modeling-practical-basics-ebal-studios-polygons-in-3d-modelling/

Math vs. Graphics

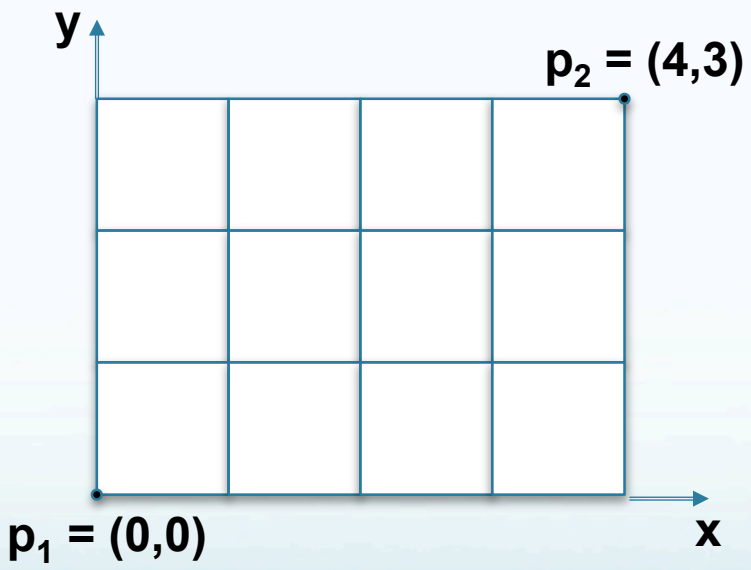
- In mathematics, a line is infinitely long and infinitely thin

  - An infinitely thin line would be invisible!
  - We'll have to allow some thickness to see it
  - We only want to draw one segment of a line
- Two points determine a line
  - For a segment, we can use the endpoints
  - Fill in (just) enough pixels to connect the endpoints



Drawing a Line

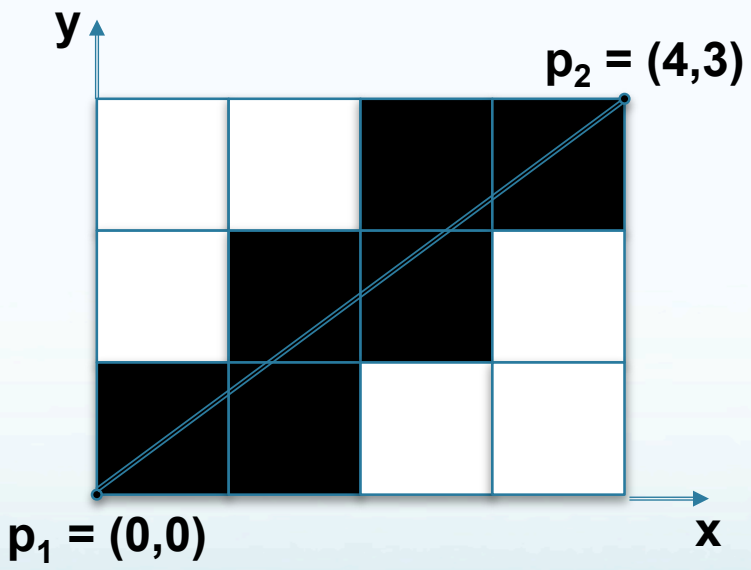
- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?



** Remember that on the screen, the y axis is actually inverted!*

Drawing a Line

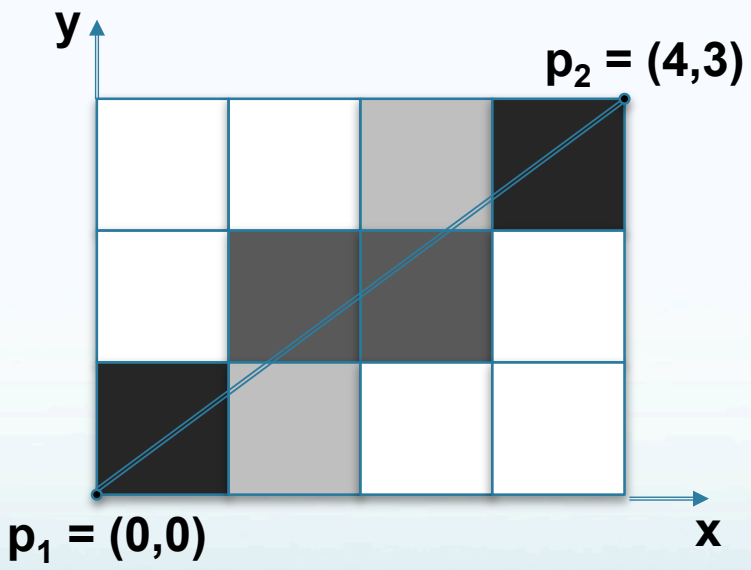
- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?



Too much: line barely touches some pixels; result is too thick

Drawing a Line

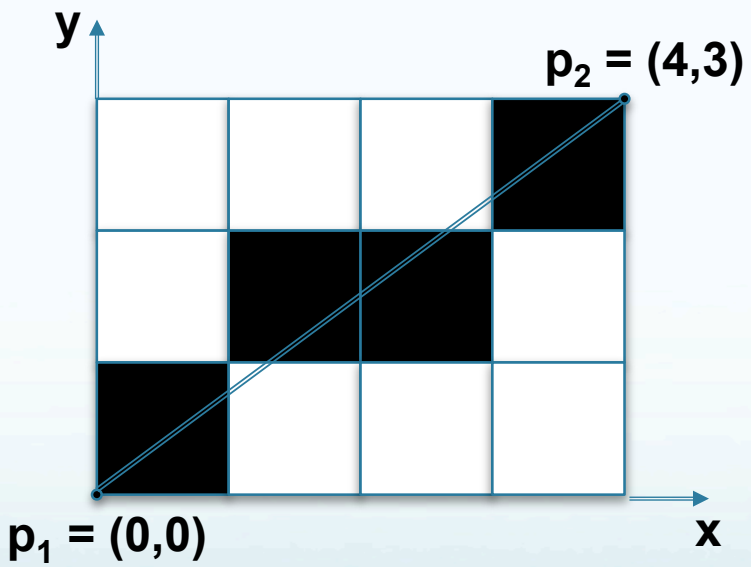
- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?



Anti-aliased line:
more advanced topic

Drawing a Line

- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?

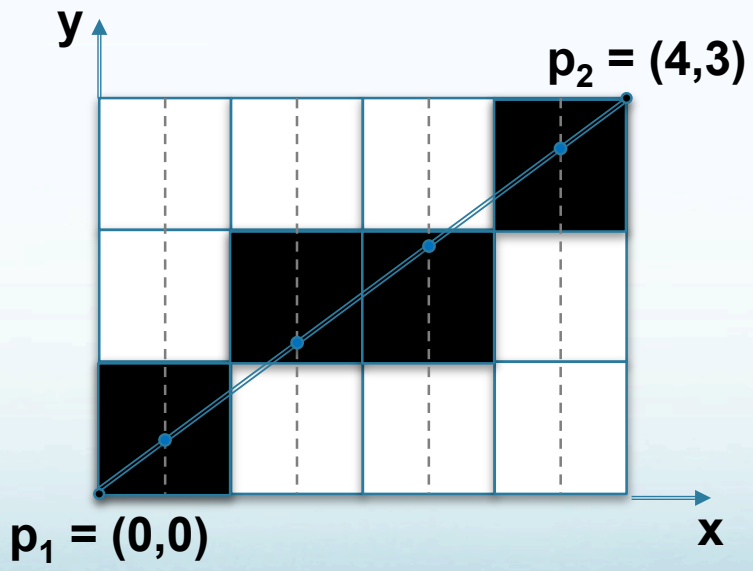


Simple line: minimally connected; exactly one pixel filled per column

What algorithm will give this?

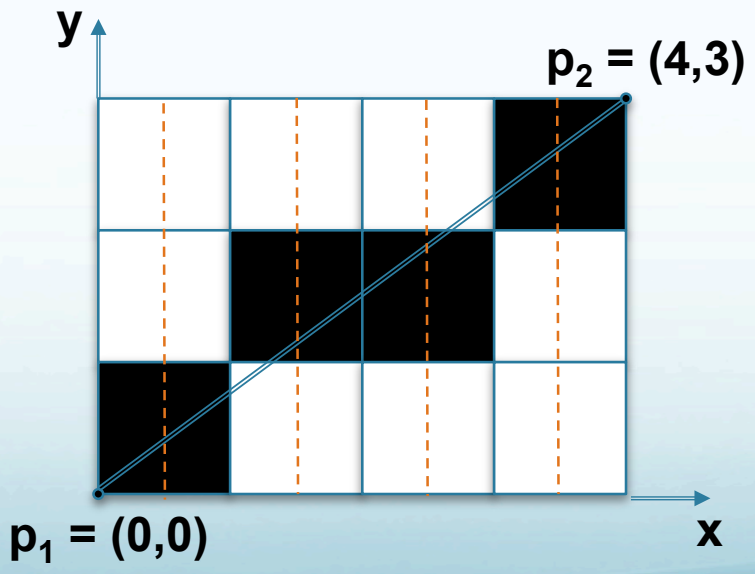
Simple Line Algorithm

- Loop over the columns:
  - Compute the intersection of the line with the center of each column
  - Fill in the pixel closest to each intersection point



Simple Line Algorithm

```
loop for x from x1 to x2
   compute corresponding y using line equation
   color pixel (floor(x),floor(y))
endloop
```



Math for Lines

- Slope intercept form: $y = mx + b$
- General form: $Ax + By + C = 0$

  (Multiplying by any nonzero constant gives equivalent eqn.)
- Form given any two points (x_1,y_1) and (x_2,y_2):

$$(y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0$$

- Converted back to slope-intercept, this gives:

$$y = \frac{y_2 - y_1}{x_2 - x_1}x + \frac{x_2y_1 - x_1y_2}{x_2 - x_1}$$

Slope!

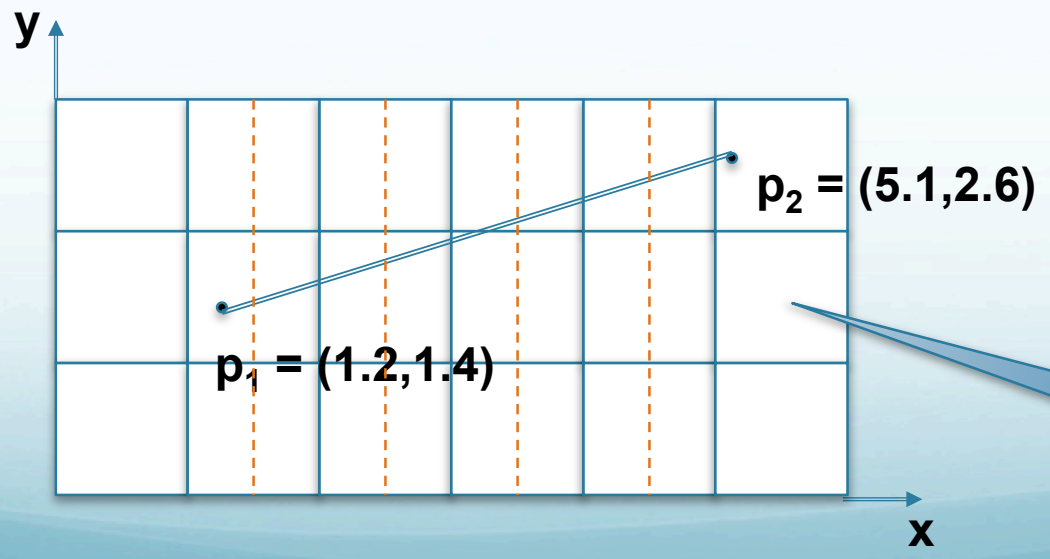
Questions

PAUSE NOW & ANSWER

- What is the slope of the line that passes through (0,0) and (4,3)?

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3 - 0}{4 - 0} = \frac{3}{4}$$

- Suppose you want to draw a line between (1.2,1.4) and (5.1,2.6). List the x values of the column centers you would need to intersect with the line?

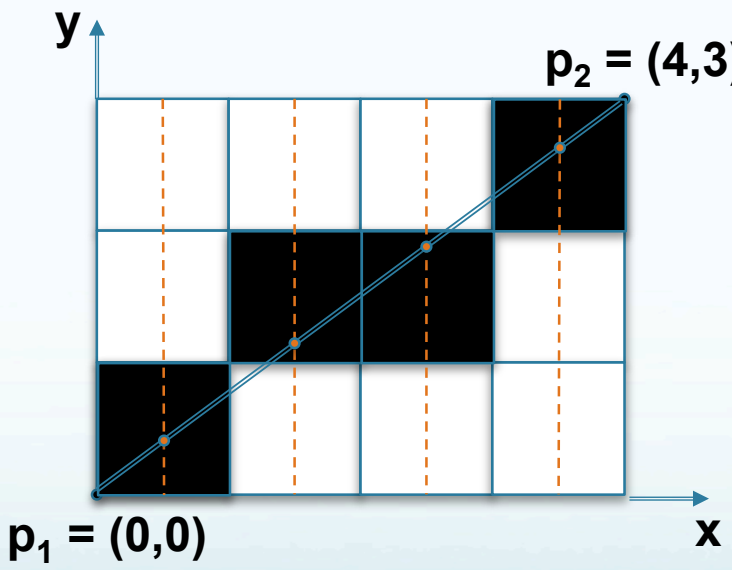


1.5, 2.5, 3.5, 4.5

Not 5.5 because the line stops at 5.1

Drawing a Line

- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?



$$Slope = \frac{rise}{run} = \frac{y_1 - y_2}{x_1 - x_2} = \frac{3}{4}$$

x	y	ink
0.5	0.375	(0,0)
1.5	1.125	(1,1)
2.5	1.875	(2,1)
3.5	2.625	(3,2)

Note: inked pixel is **floor** of x and y.

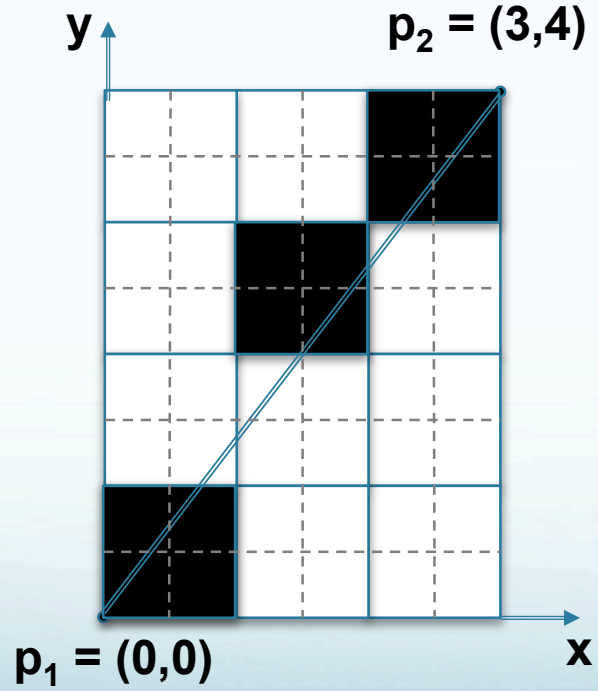
$$(y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0$$

$$-3x + 4y = 0$$

Remember: we are using corner-origin coordinates

Another Line

- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (3,4)}$?



What's the problem here?

By looping over columns, we only filled in three pixels. But the endpoints are more separated vertically, so we end up with gaps.

How should we solve it?

If we loop over rows instead of columns, then we will fill in four pixels and the line will be connected.

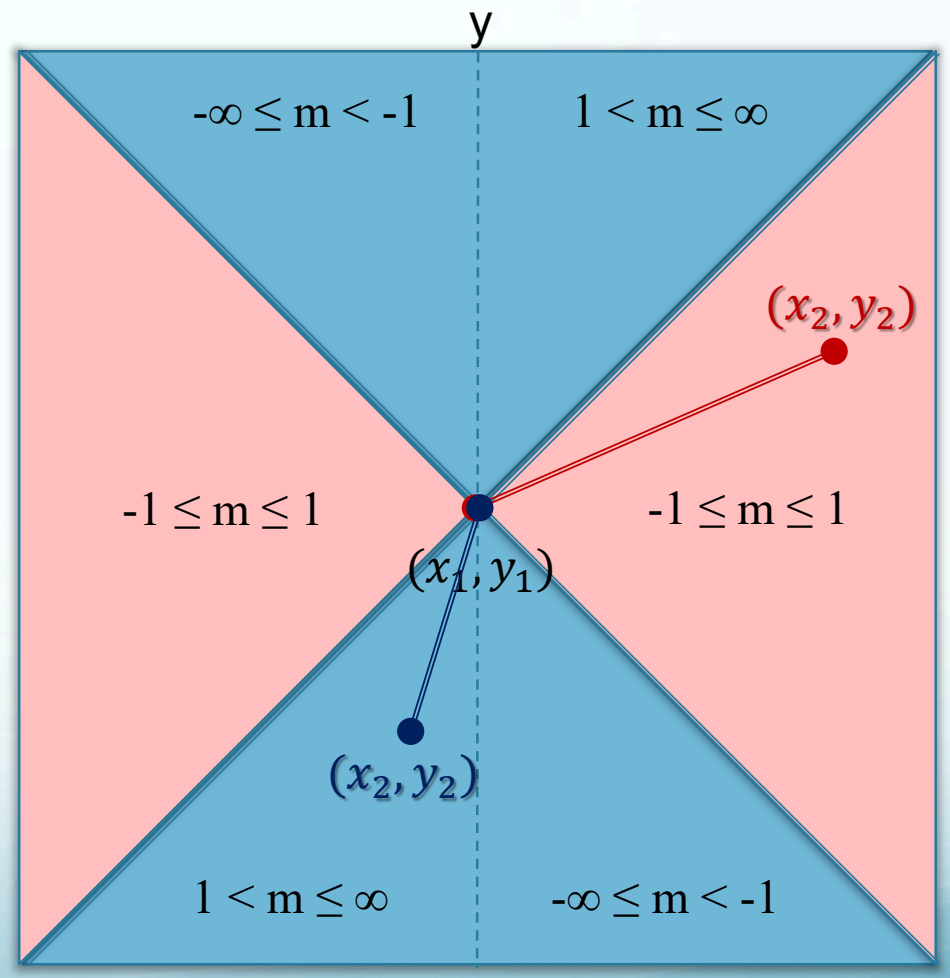
The line's slope determines whether we should loop over rows or columns

Line Drawing Cases (Simple Algorithm)

Two zones depending on slope:

I. Slope between -1 and 1:
loop over columns (step in x)

II. Slope more than 1,
or less than -1:
loop over rows (step in y)



Questions

PAUSE NOW & ANSWER

1. If we always looped over x regardless of slope, which lines would possibly appear disconnected? (Describe their range of slopes)

   Lines with slope greater than 1 or less than -1

2. What if we always looped over y?

   Lines with slope between 1 and -1

3. Suppose we are looping over rows, and our line intersects with the current row at the point (3.9,6.5). What pixel should we color in?

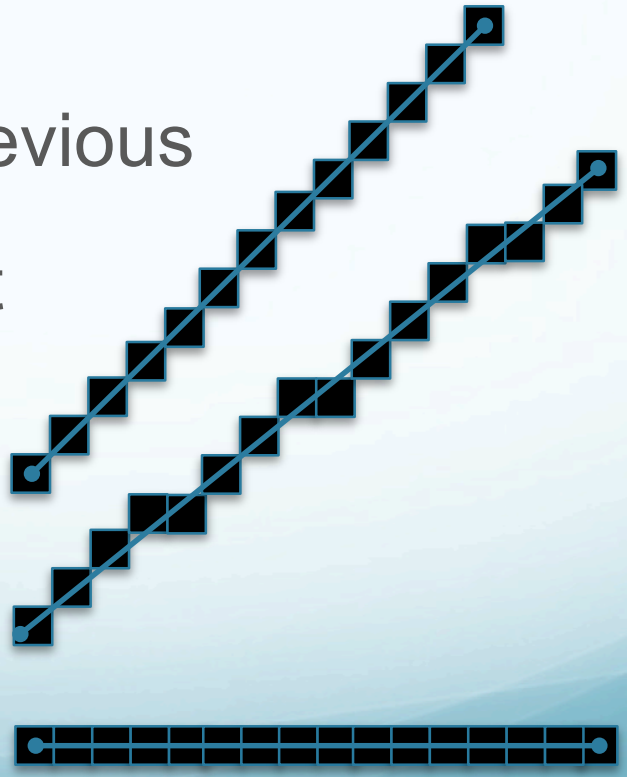
   The pixel can be found by rounding down, so the answer is (3,6)

Midpoint Algorithm

Consider cases for moderate slope, $0 \leq m \leq 1$.

- For $m = 0$ each pixel is to the right of previous
- For $m = 1$ each pixel is up and to the right of previous
- In between, next pixel is **either** up or to the right

Clever Trick: avoid computing line equation by finding a rule to decide which way to move



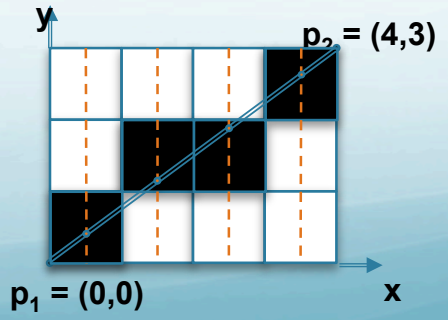
Midpoint Algorithm

- Loop over columns, moving upwards as necessary to follow line (pseudocode below assumes $0 \leq m \leq 1$; similar loop in other cases)

```
y ← y1
loop for x = x1 to x2 do
        if (some_test) then
                y ← y+1
        ink(x,y)
endloop
```

When should we move?

- Test whether line is above or below pixel boundary at middle of next column



Math for Lines (2)

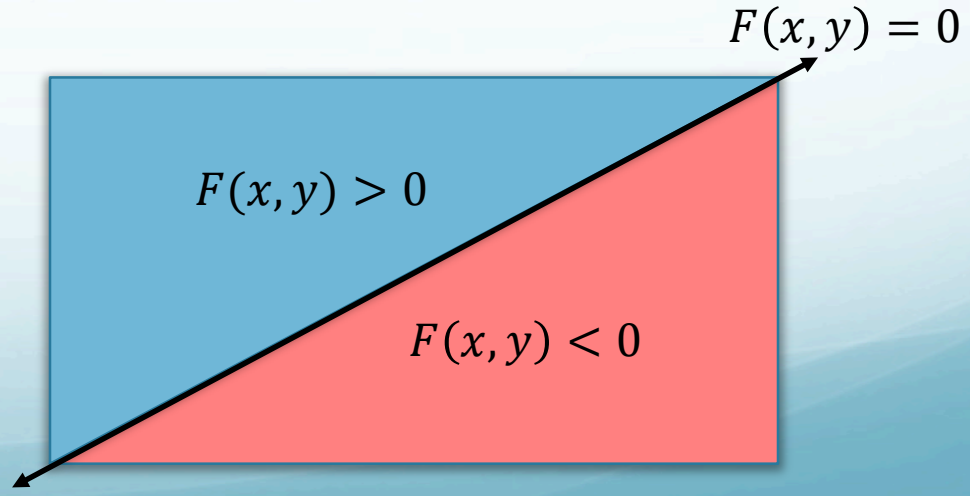
Recall that the equation of a line through two points (x_1, y_1) and (x_2, y_2) is:

$$(y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0.$$

- Let's call that left hand side F:

$$F(x, y) = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1$$

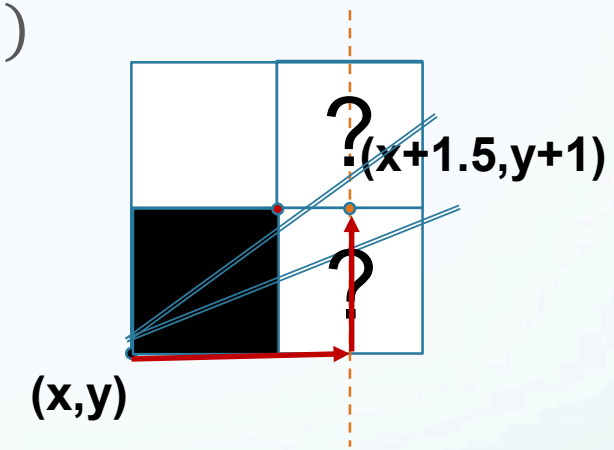
- The line is all (x, y) for which $F(x, y)$ evaluates to zero.
- What about other points?
  - Points above the line: $F(x, y) > 0$
  - Points below the line: $F(x, y) < 0$



Midpoint Algorithm

Use $F(x, y)$ to see if line is above or below boundary for next pixel

- If current pixel is (x, y) then next boundary is$(x + 1.5, y + 1)$
- If $F(x + 1.5, y + 1) > 0$ then boundary point is above line
  - Need to keep y the same
- If $F(x + 1.5, y + 1) < 0$ then boundary point is below line
  - Need to increment y

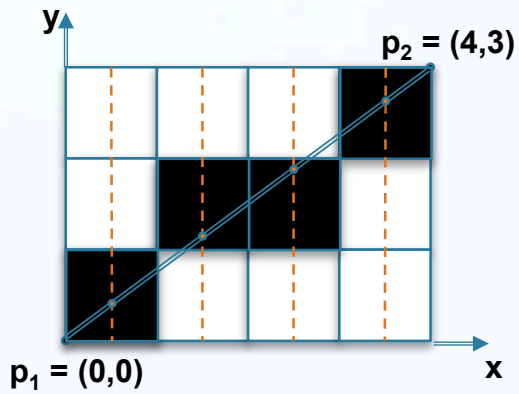


After determining new y, increment x and begin again for next column

$$F(x,y) = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1$$

Midpoint Algorithm

```
y ← 0
loop for x = 0 to 3 do
        ink(x,y)
        if F(x+1.5,y+1)<0 then
                y ← y+1
endloop
```



$F(x,y) = -3x + 4y$

x	y	F(x+1.5,y+1)	Decision

Midpoint Algorithm

```
y ← 0
loop for x = 0 to 3 do
        ink(x,y)
        if F(x+1.5,y+1)<0 then
                y ← y+1
endloop
```

We're still doing as much work as before, since this line computes the full line equation

Not much changes between adjacent points. Let's use the previously computed F to get the new value!

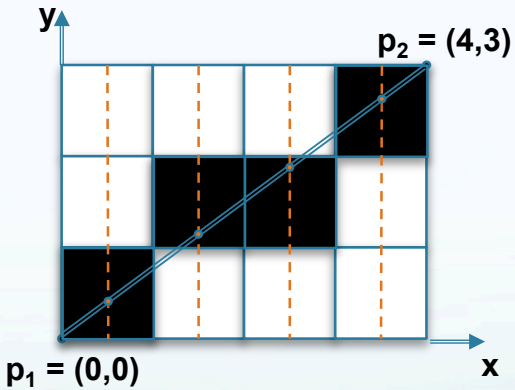
Incremental Midpoint Algorithm*

```
y ← y1
d ← F(x1+1.5,y+1)
for x = x1 to x2 do
    ink(x,y)
    if d<0 then
        y ← y+1
        d ← d +(x2-x1) +(y1-y2)
    else
        d ← d+(y1-y2)
endfor
```

Keep track of running value of F(x,y)

Instead of the whole line equation, we do just one addition per column. Win!

Both x and y change here, so we add two terms



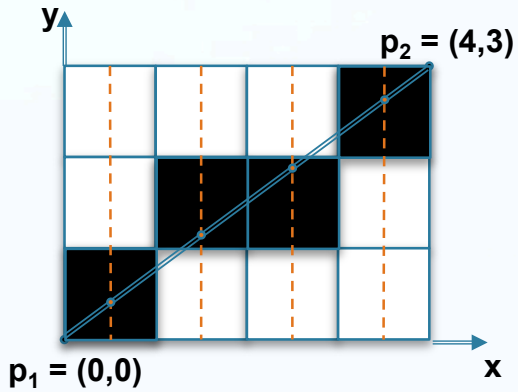
Only x changes in this case, so add the x term to running total

$$F(x,y) = (y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1$$

* AKA Bresenham's algorithm

Incremental Midpoint Algorithm

```
y ← 0
d ← F(1.5,1)
for x = 0 to 4 do
    ink(x,y)
    if d<0 then
        y ← y+1
        d ← d +4-3
    else
        d ← d-3
endfor
```



$F(x,y) = -3x + 4y$

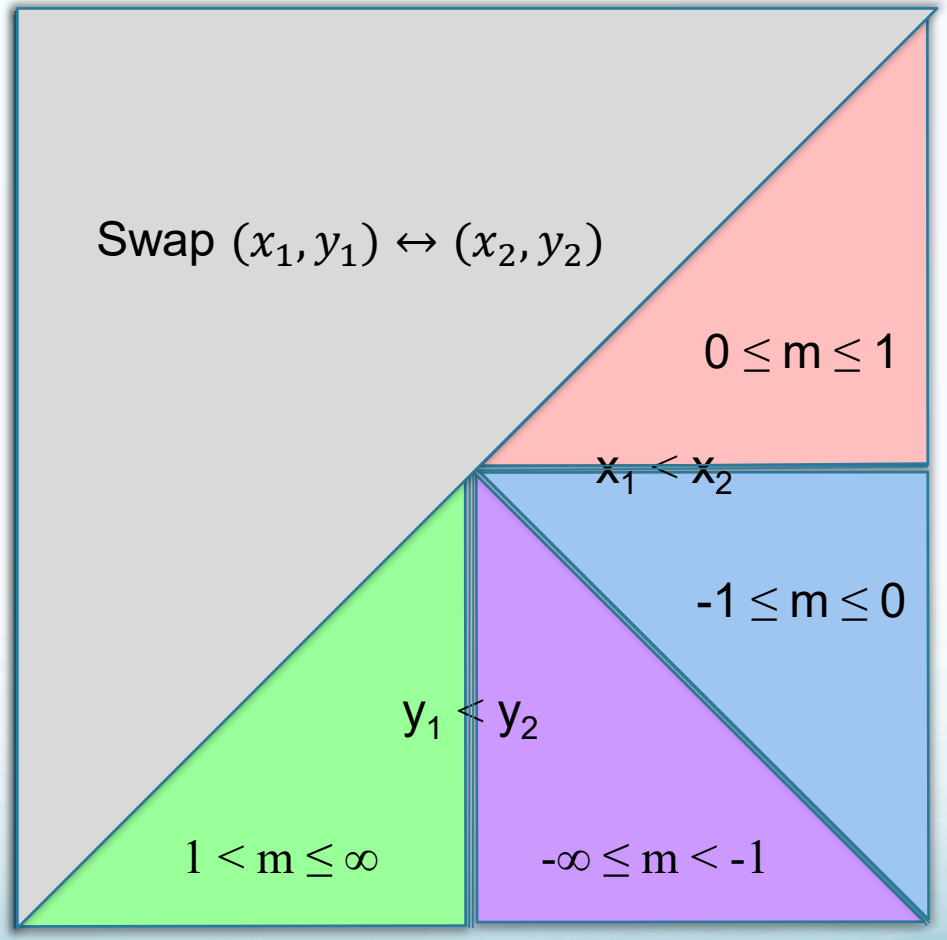
Note d holds the same value as computed F before!

x	y	F(x+1.5,y+1)	d	Decision

Incremental Line Drawing Cases

- Eight zones depending on relative values of x_1, x_2, y_1 & y_2
- Can swap $x_1 \leftrightarrow x_2$, $y_1 \leftrightarrow y_2$ to fold eight cases into four

I: step x, increment y
II: step x, decrement y
III: step y, increment x
IV: step y, decrement x



$-\infty < m < -1$

Questions

PAUSE NOW & ANSWER

1. What are the effects of different pixel grid conventions on a line drawing implementation?

   Changes to the offsets

2. What are the advantages of the incremental line drawing algorithm?

   Fewer arithmetic operations per pixel

3. What are the disadvantages of the incremental line drawing algorithm?

   More complicated to program

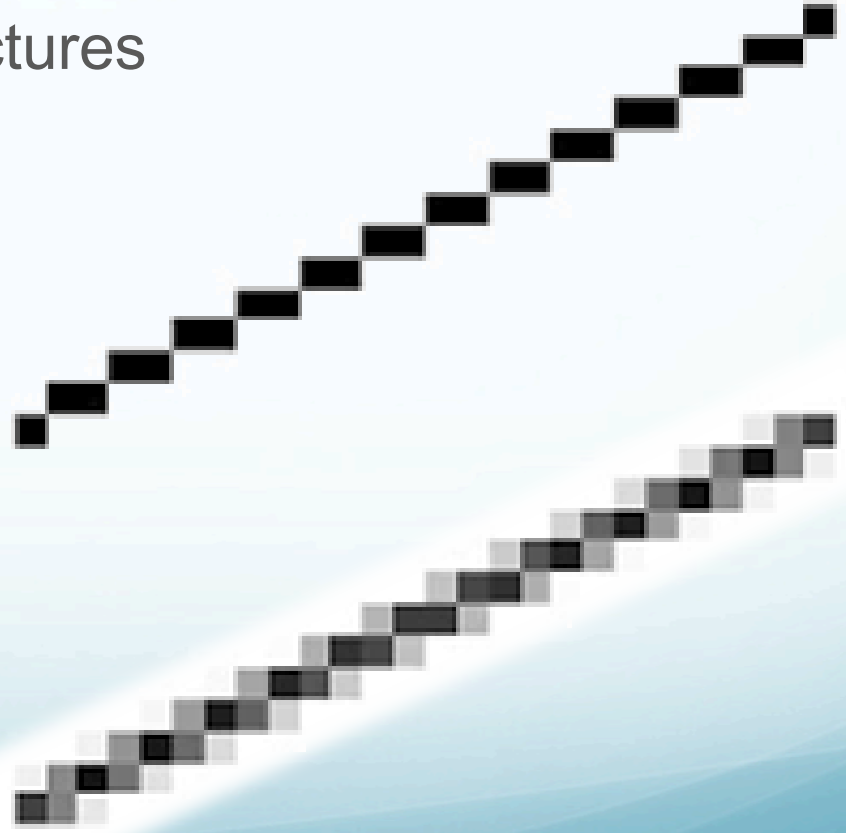
Antialiasing

The simple line algorithm and its relatives all suffer from **aliasing**

- Jaggedness/discontinuity due to pixel grid strictures
- Visually unappealing

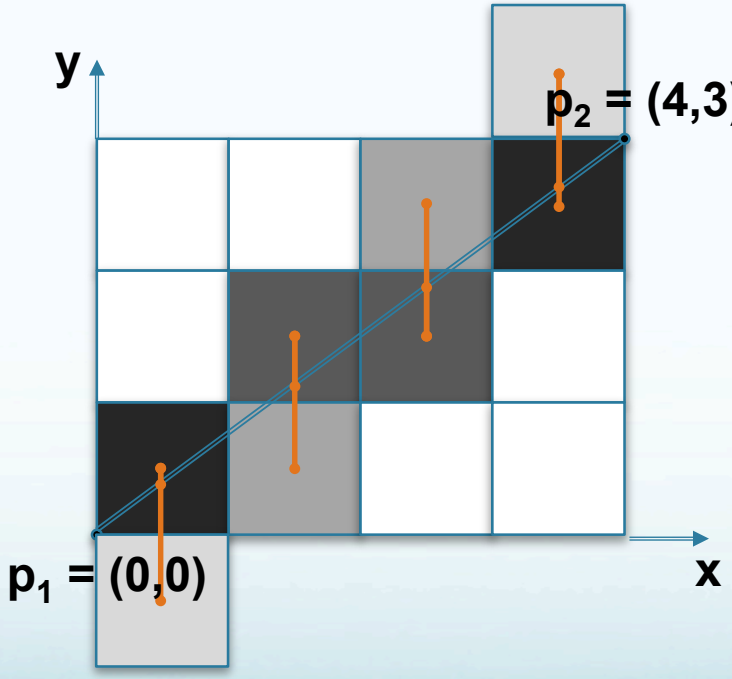
Solution is a strategic, deliberate blurring!

- Known as **antialiasing**
- Idea: use variable shading
  to create a smoother look
- Weight shading based on path of line



Antialiased Line

- What pixels should be colored for a line with endpoints $\mathbf{p_1 = (0,0)}$ and $\mathbf{p_2 = (4,3)}$?



Xiaolin Wu's algorithm: Shade pixels according to vertical distance from pixel centers to the line

x	y	lower	upper
0.5	0.375	0.125	0.875
1.5	1.125	0.375	0.625
2.5	1.875	0.625	0.375
3.5	2.625	0.875	0.125

Review

After this video, you should know how to:

- Give pseudocode for a simple line algorithm, and implement it if need be
- Given endpoints of a line, determine the proper loop for rendering it
- Carry out by hand the calculations for the simple line algorithm
- Give pseudocode for the midpoint line algorithm
- Explain the advantages of the midpoint algorithm over the simple one
- Define antialiasing and how it applies to drawing lines.

Music: https://www.bensound.com