Scan Conversion

Rasterization of Primitives

- How to draw primitives?
  - Convert from geometric definition to pixels
  - *rasterization* = selecting the pixels
- Will be done frequently
  - must be fast:
    - use integer arithmetic
    - use addition instead of multiplication

Rasterization Algorithms

- Algorithmic:
  - Line-drawing: Bresenham, 1965
  - Polygons: uses line-drawing
  - Circles: Bresenham, 1977
- Currently implemented in *all* graphics libraries
  - You'll probably never have to implement them yourself

Why should I know them?

- Excellent example of efficiency:
  - no superfluous computations
- Possible extensions:
  - efficient drawing of parabolas, hyperbolas
- Applications to similar areas:
  - robot movement

Point plotting

- Application program furnish the co-ordinates.
- If the co-ordinates are real no's, they are rounded off to nearest integer, as device co-ordinates support only integer co-ordinates.

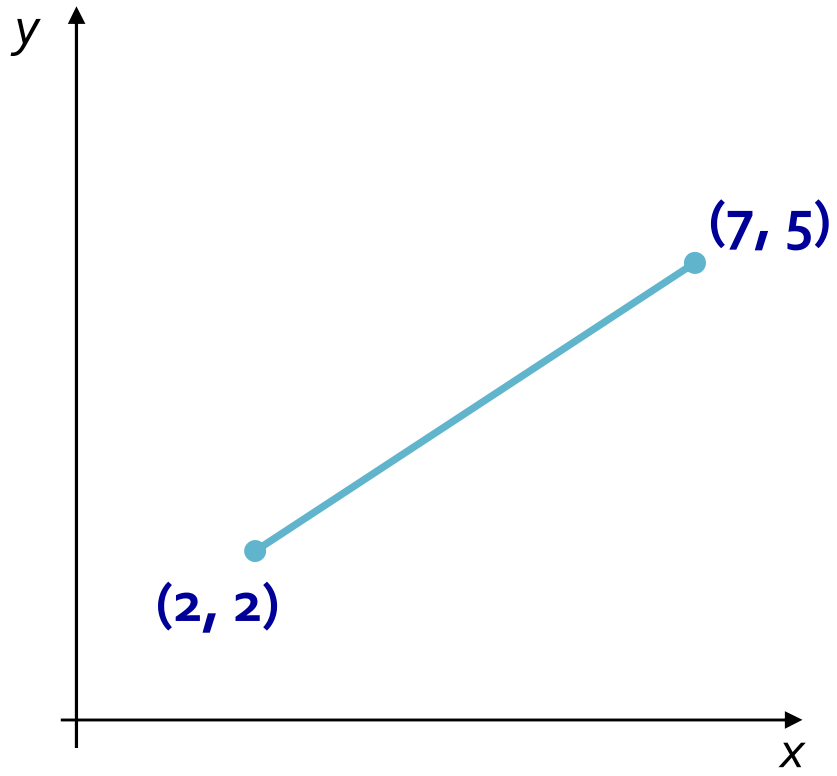
Line Scan Conversion

Line Plotting

- Two end points (co-ordinates) are specified
- Intermediate positions between end points are calculated (using line eqn)
- The intermediate and end points are plotted
- Pixel level color management to draw line on color CRT

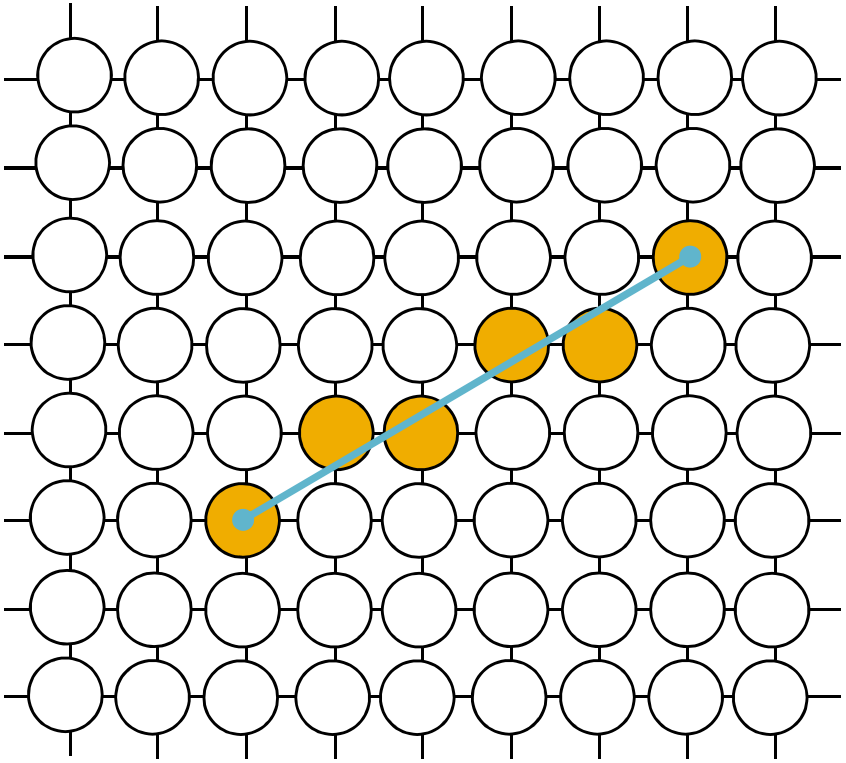
Scan Conversion Problem of Lines

- A line segment in a scene is defined by the coordinate positions of the line end-points



The Problem (cont…)

- But what happens when we try to draw this on a pixel based display?

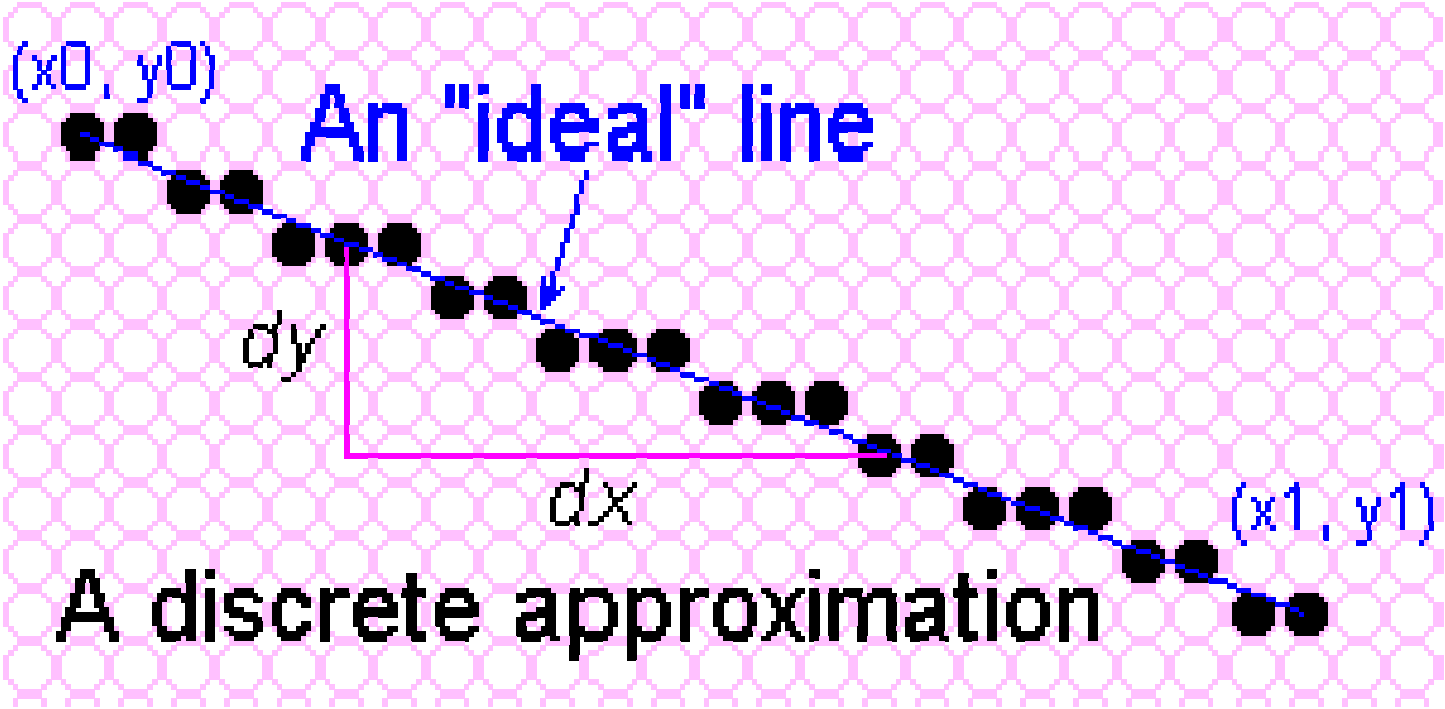


- How do we choose which pixels to turn on?

Line Plotting

- jaggies
  - Screen location have integer values
  - If all the points on a line have integer co-ordinate, no problem
  - Rarely the case
  - In practice, non integer co-ordinates are approximated to integer co-ordinates (e.g. [10.48, 20.51] → [10, 20])
  - Such approximation makes the line displayed with a stairstep appearance, called "jaggies"
- Jaggies reduce with increase in resolution

Jaggies

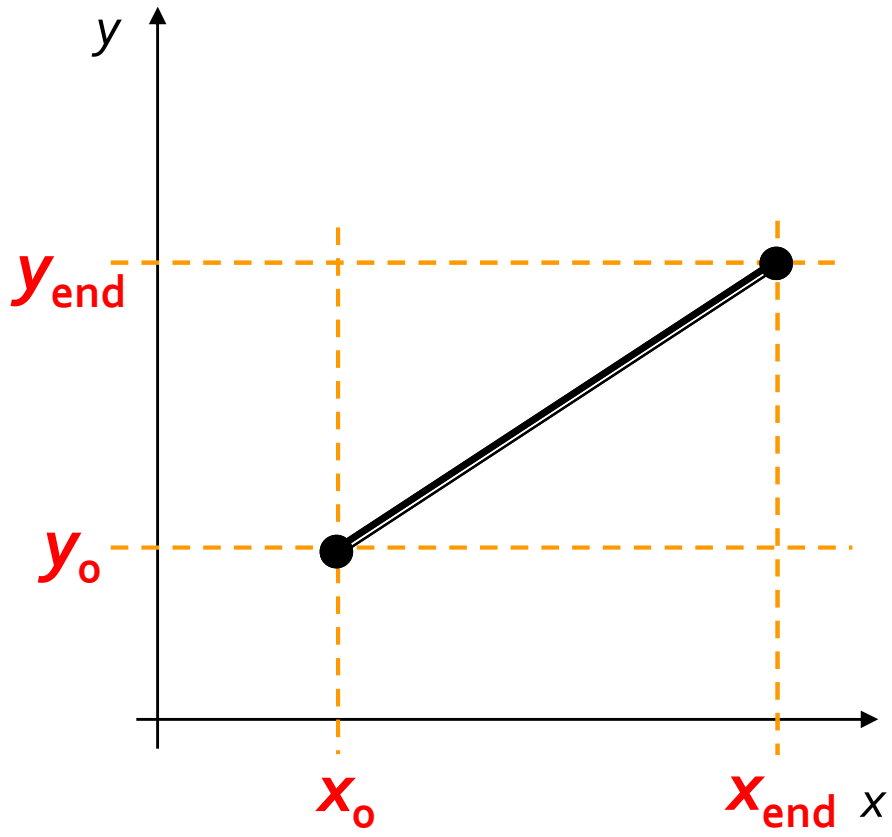


Considerations

- Considerations to keep in mind:
  - The line has to look good
    - Avoid *jaggies*
  - It has to be lightening fast!
    - How many lines need to be drawn in a typical scene?
    - This is going to come back to bite us again and again

Line Equations

- Let's quickly review the equations involved in drawing lines



- Slope-intercept line equation:

$$y = m \cdot x + b$$

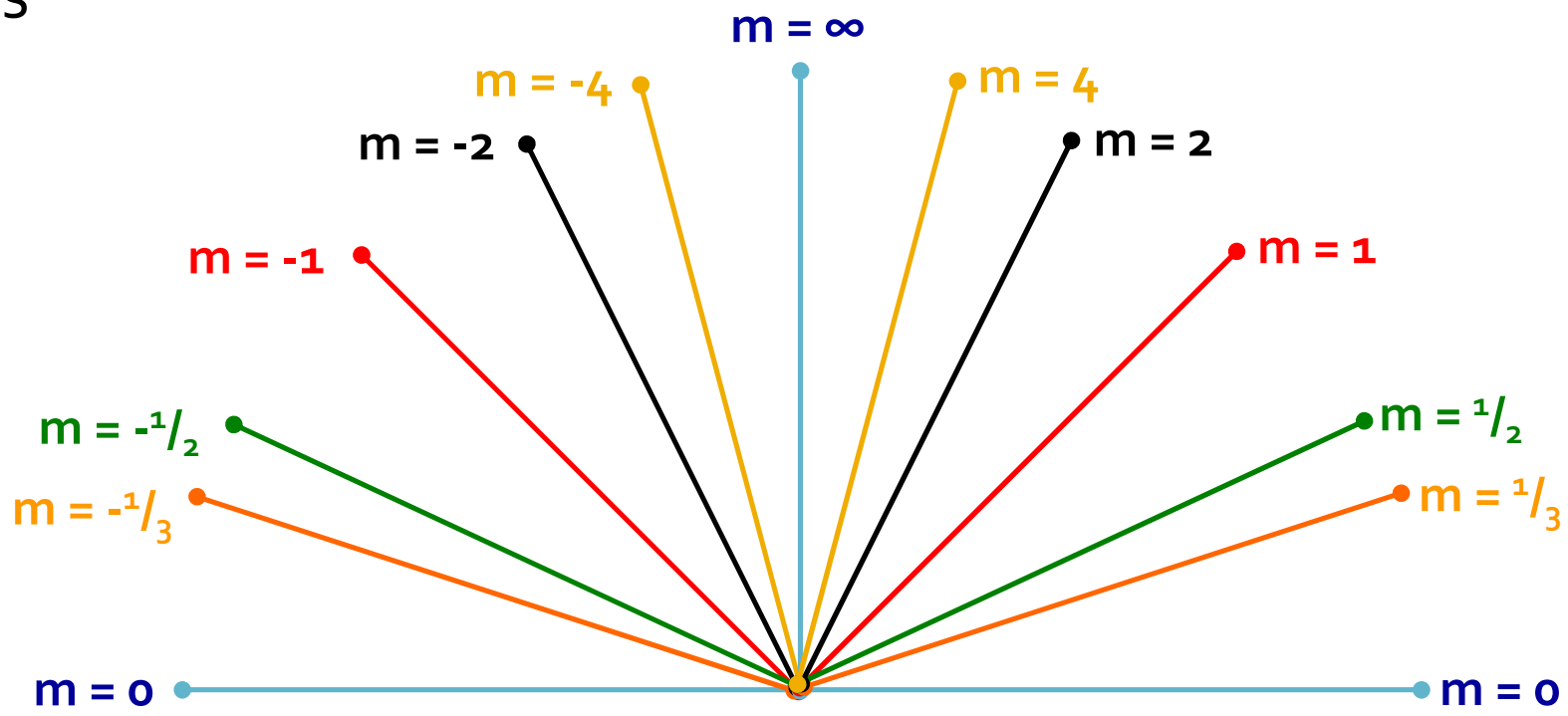
- where:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

$$b = y_0 - m \cdot x_0$$

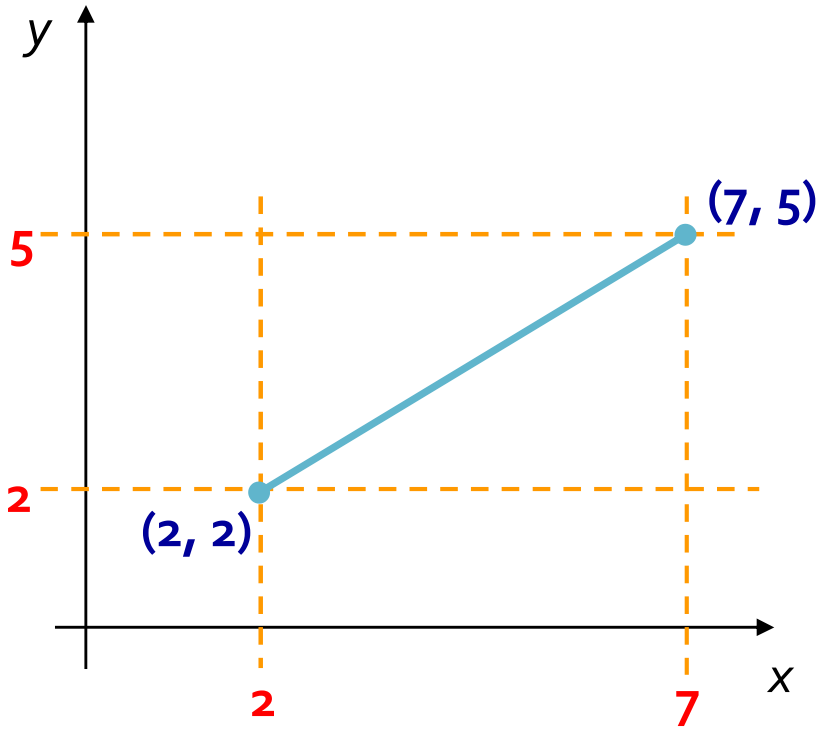
Lines & Slopes

- The slope of a line (*m*) is defined by its start and end coordinates
- The diagram below shows some examples of lines and their slopes

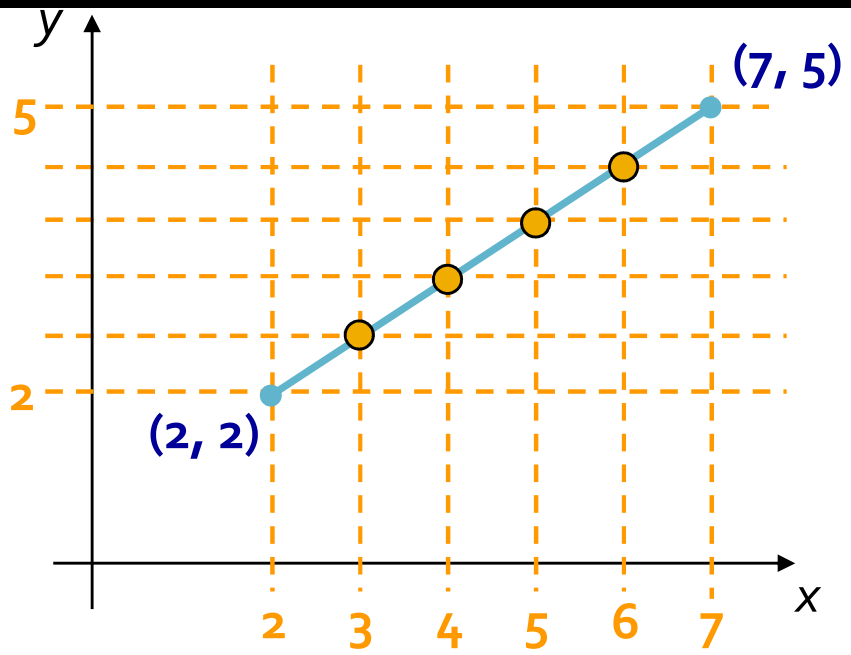


A Very Simple Solution

- We could simply work out the corresponding *y* coordinate for each unit *x* coordinate
- Let's consider the following example:



A Very Simple Solution (cont...)



- First work out *m* and *b*:

$$m = \frac{5-2}{7-2} = \frac{3}{5}$$

$$b = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

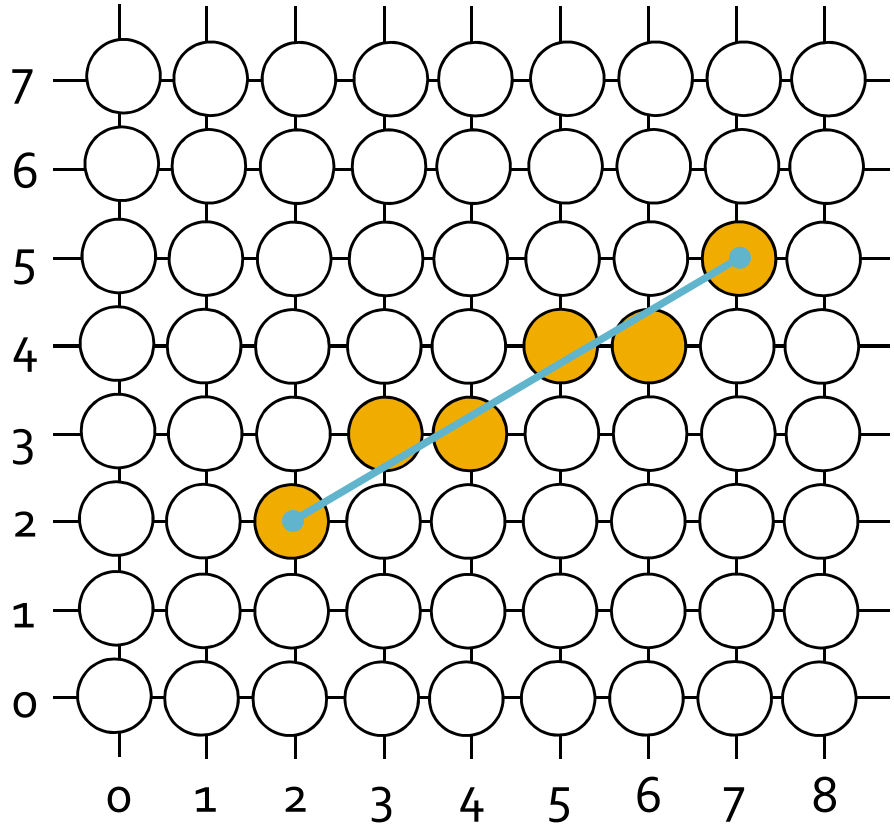
- Now for each *x* value work out the *y* value:

$$y(3) = \frac{3}{5} \cdot 3 + \frac{4}{5} = 2\frac{3}{5} \qquad y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5}$$

$$y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5} \qquad y(6) = \frac{3}{5} \cdot 6 + \frac{4}{5} = 4\frac{2}{5}$$

A Very Simple Solution (cont…)

- Now just round off the results and turn on these pixels to draw our line



$$y(3) = 2\frac{3}{5} \approx 3$$

$$y(4) = 3\frac{1}{5} \approx 3$$

$$y(5) = 3\frac{4}{5} \approx 4$$

$$y(6) = 4\frac{2}{5} \approx 4$$

A Very Simple Solution (cont…)

- However, this approach is just way too slow
- In particular look out for:
  - The equation $y = mx + b$ requires the multiplication of m by x
  - Rounding off the resulting y coordinates
- We need a faster solution