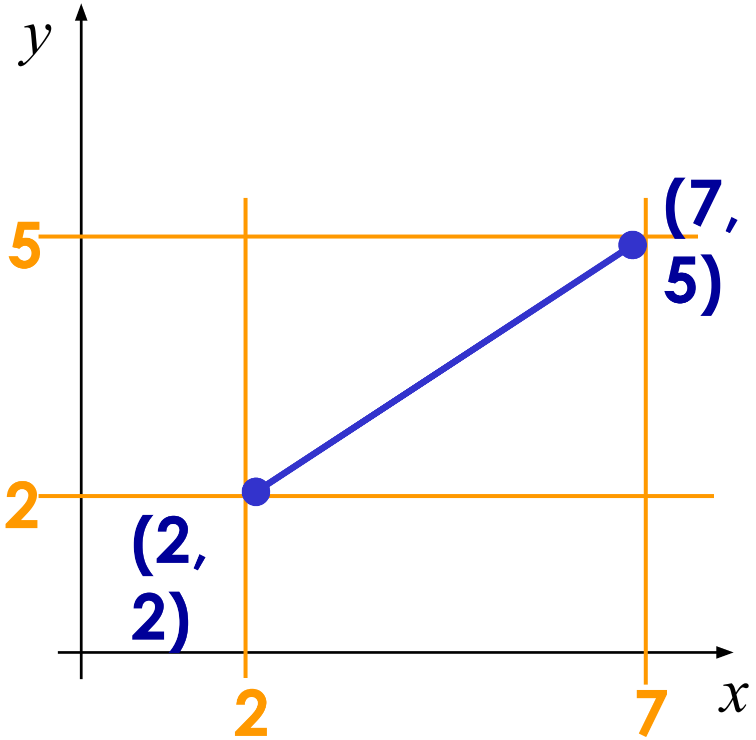
CSE- 4105

Md. Manowarul Islam
Lecturer
Department of CSE
Jagannath University

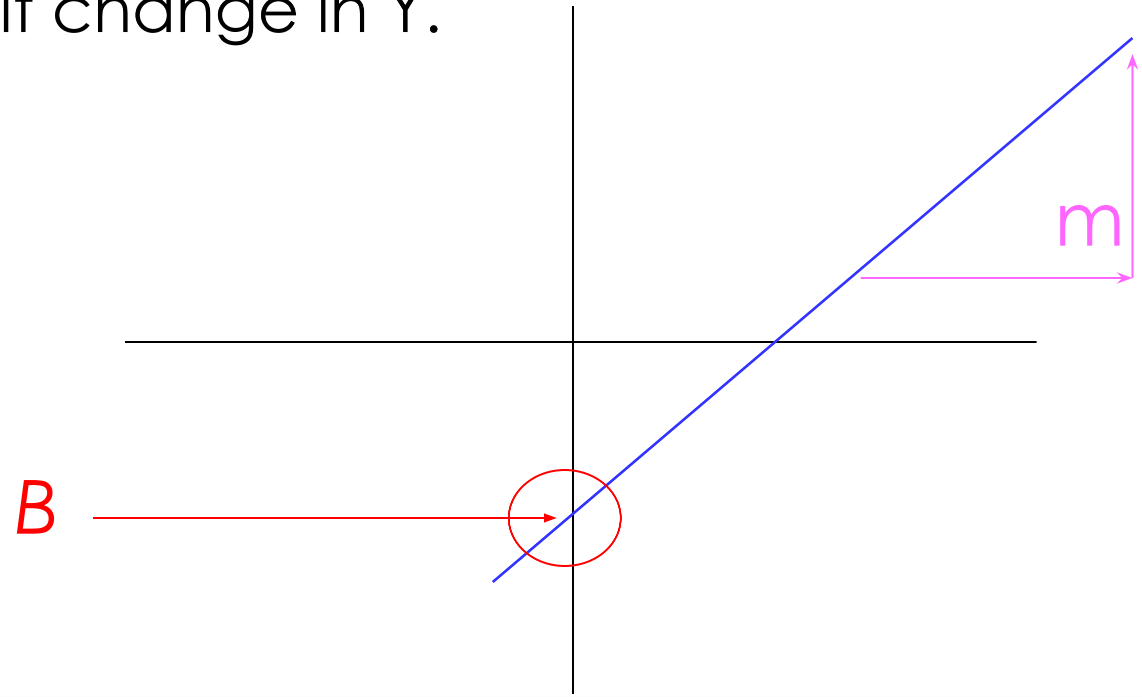
Basic of line

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



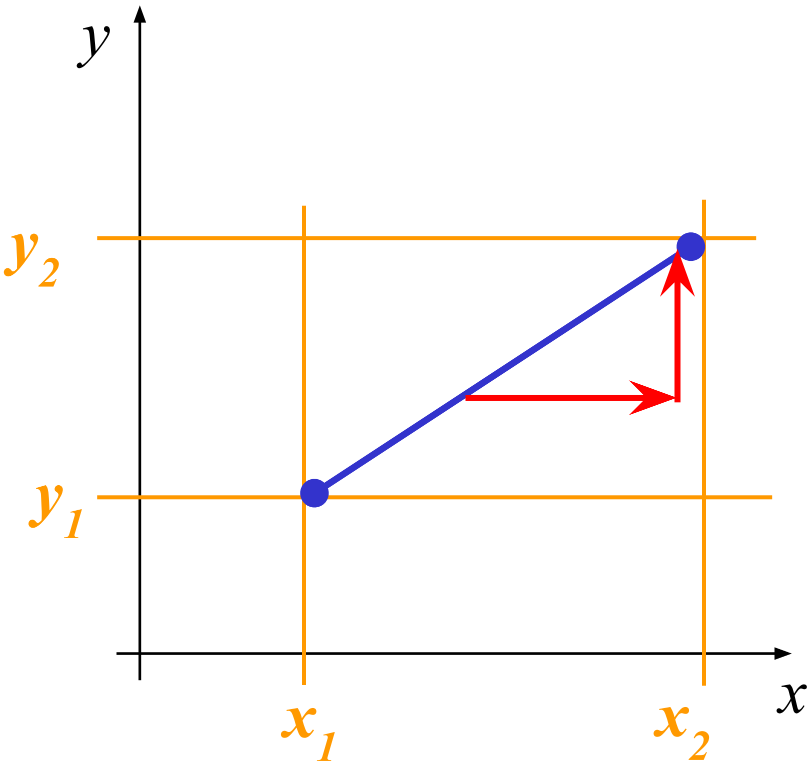
Basic of line

- Line drawing is accomplished by calculating intermediate positions along the line path between specified end points.
- $Y=mX+B$
- A slope of 2 means that every 1-unit change in X yields a 2-unit change in Y.



Line Equations

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



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

where:

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

$$b = y_1 - m \cdot x_1$$

Algorithm

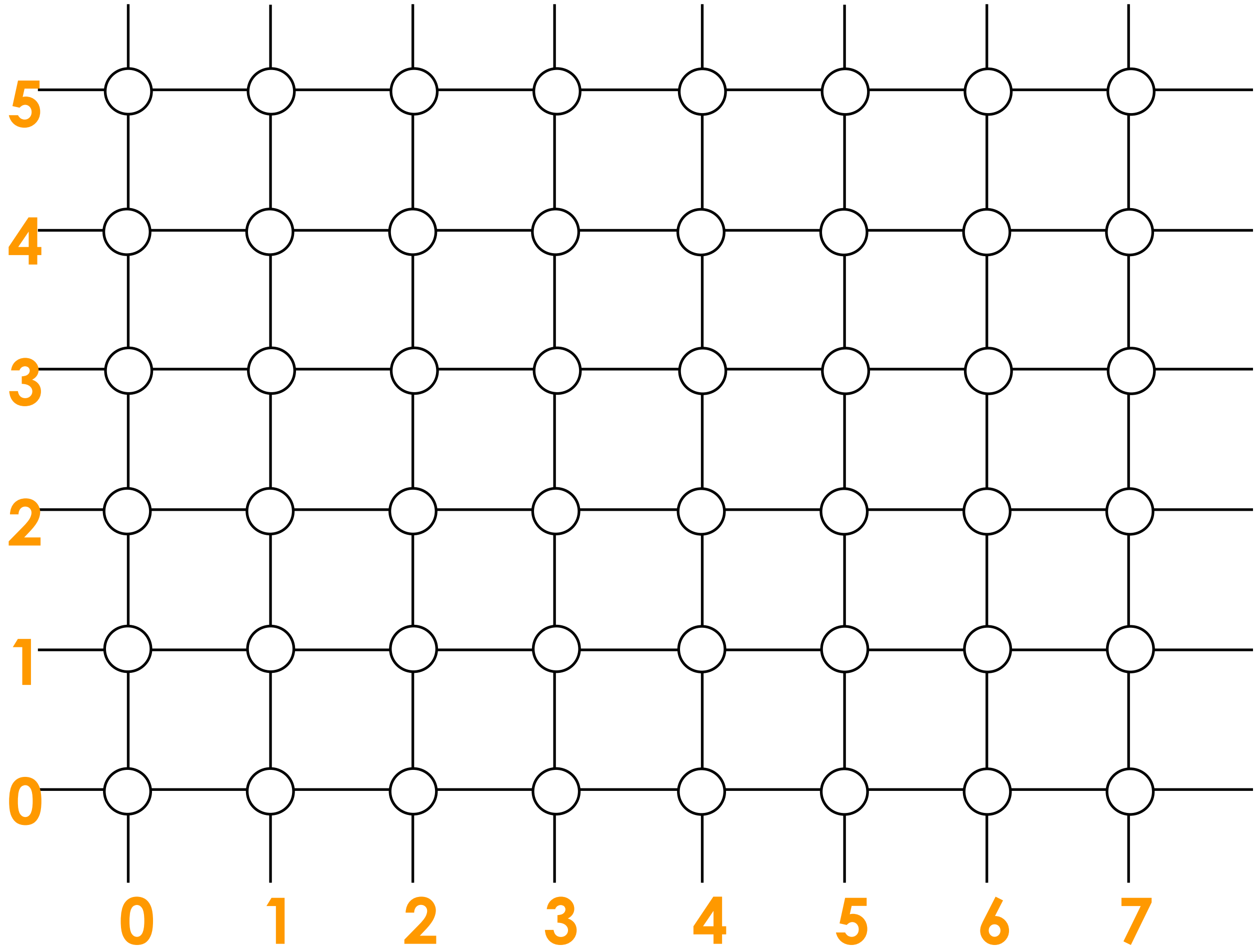
- Start at the pixel for the left-hand endpoint x_1
- Step along the pixels horizontally until reach the right-hand end point x_2
- For each value of x compute corresponding y value
- Round this value to nearest integer to select the nearest pixel

Basic Algorithm:
For x= x_1 to x_2
    y=mx+b
    PlotPixel(x,round(y))
End;

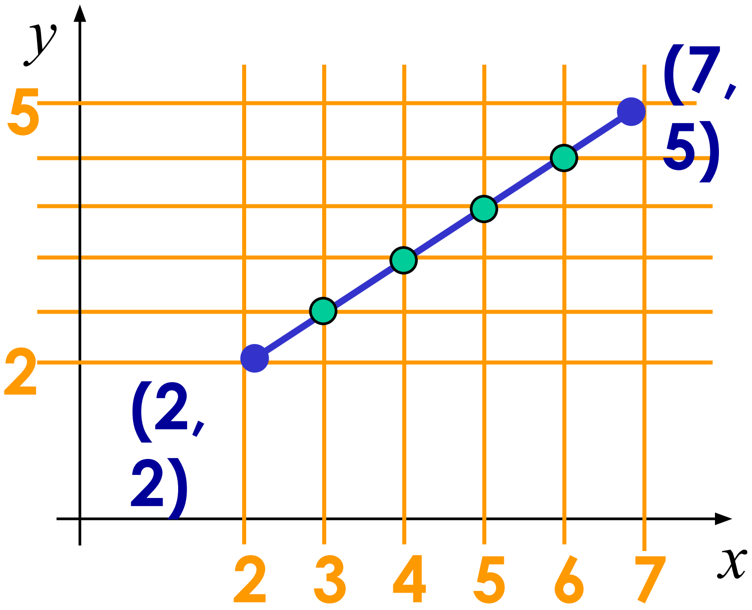
Problems: Lot of computation and inefficient.

For each iteration: 1 float multiplication, 1 addition, 1 round

A Very Simple Solution



A Very Simple Solution



- First work out m and b:

$$m = \frac{5-2}{7-2} = \frac{3}{5} \qquad b = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

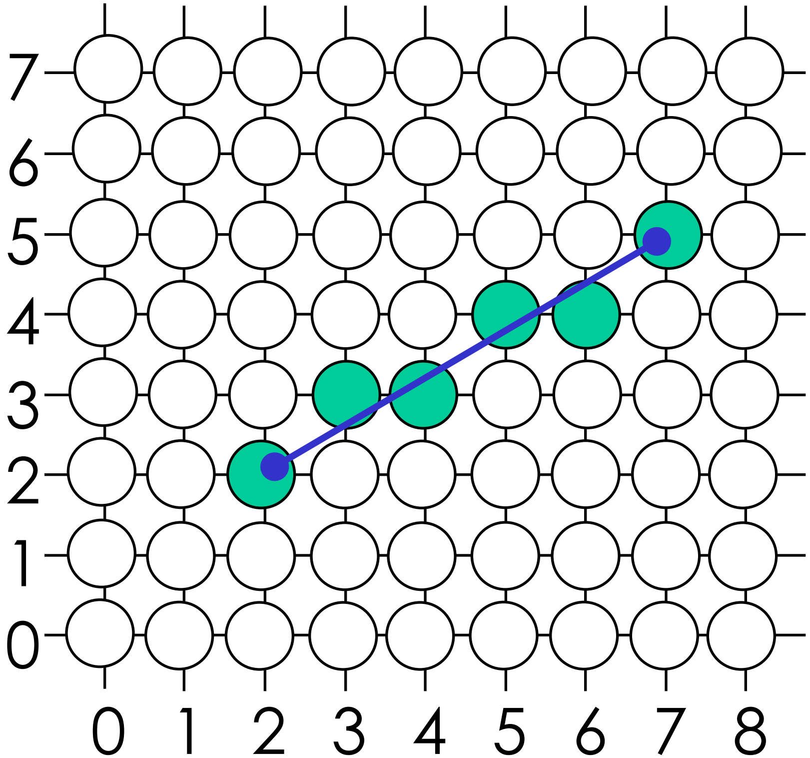
Now for each x value work out the y value:

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

$$y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5} \qquad y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5} \qquad y(6) = \frac{3}{5} \cdot 6 + \frac{4}{5} = 4\frac{2}{5}$$

A Very Simple Solution

- 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

- 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

Problems: Lot of computation and inefficient.

For each iteration: 1 float multiplication, 1 addition, 1 round

Line Equations

- The equation of a simple line is: $y_i = m \cdot x_i + b$

- Modify the equation:

$$y_{i+1} = m \cdot x_{i+1} + b$$

$$= m(x_i + 1) + b$$

$$= mx_i + m + b$$

Replace by y_i $\longrightarrow$ $\boxed{= (mx_i + b)} + m$

$$= y_i + m$$

The DDA Algorithm

- The *digital differential analyzer* (DDA) algorithm takes an incremental approach in order to speed up scan conversion
- Simply calculate y_{k+1} based on y_k

The DDA Algorithm (cont…)

- Consider the list of points that we determined for the line in our previous example:
- $(2, 2), (3, 2^3/_5), (4, 3^1/_5), (5, 3^4/_5), (6, 4^2/_5), (7, 5)$
- Notice that as the x coordinates go up by one, the y coordinates simply go up by the slope of the line
- This is the key insight in the DDA algorithm

Basic Algorithm:
For x= x_1 to x_2
    PlotPixel(x,round(y))
    y=y+m
End;

The DDA Algorithm (cont…)

- When the slope of the line is between -1 and 1 begin at the first point in the line and, by incrementing the x coordinate by 1, calculate the corresponding y coordinates as follows:

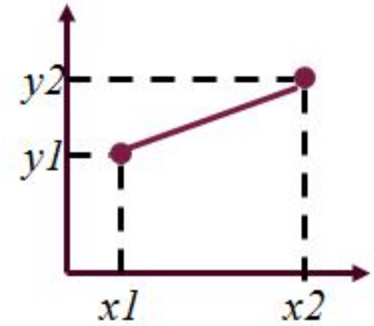
$$y_{k+1} = y_k + m$$

- When the slope is outside these limits, increment the y coordinate by 1 and calculate the corresponding x coordinates as follows:

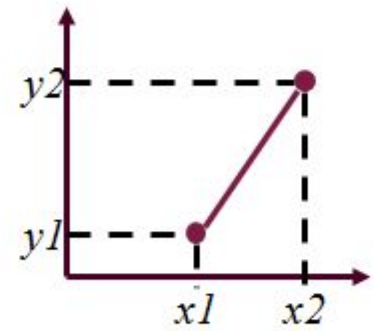
$$x_{k+1} = x_k + \frac{1}{m}$$

DDA Algorithm

- **Digital Differential Analyzer**
  - 0 < Slope <= 1
    - Unit x interval = 1
  - Slope > 1
    - Unit y interval = 1



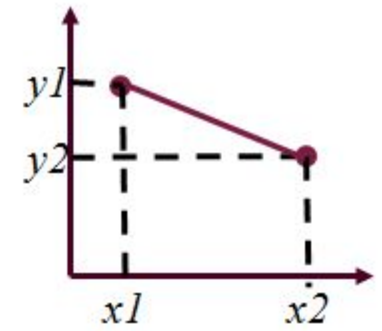
$$y_{k+1} = y_k + m$$



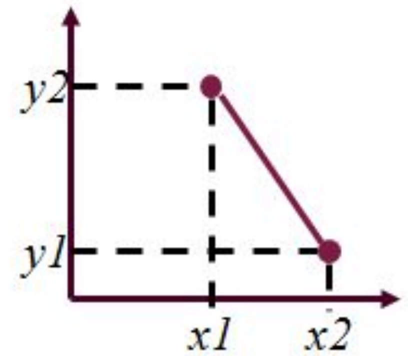
$$x_{k+1} = x_k + \frac{1}{m}$$

DDA Algorithm

- **Digital Differential Analyzer**
  - -1 <= Slope < 0
    - Unit x interval = -1
  - Slope < -1
    - Unit y interval = -1



$$y_{k+1} = y_k - m$$



$$x_{k+1} = x_k - \frac{1}{m}$$

The DDA Algorithm Summary

- The DDA algorithm is much faster than our previous attempt
  - In particular, there are no longer any multiplications involved
- However, there are still two big issues:
  - Accumulation of round-off errors can make the pixelated line drift away from what was intended
  - The rounding operations and floating point arithmetic involved are time consuming

Thank You