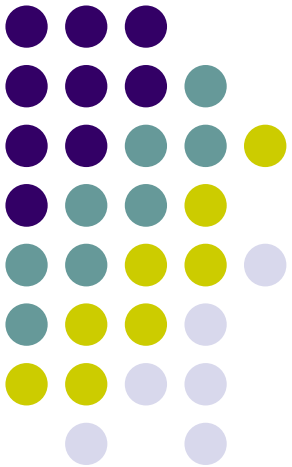
Computer Graphics (CS 543) Lecture 9: Rasterization and Antialiasing

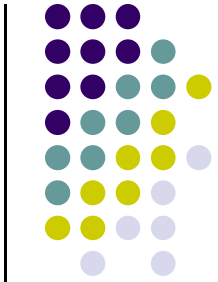
Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

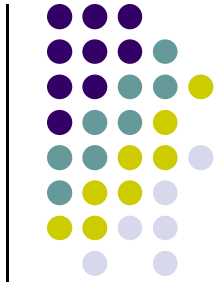


Rasterization

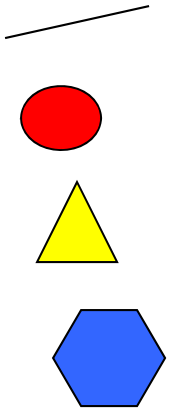


- Rasterization (scan conversion)
  - Determine which pixels that are inside primitive specified by a set of vertices
  - Produces a set of fragments
  - Fragments have a location (pixel location) and other attributes such color and texture coordinates that are determined by interpolating values at vertices
- Pixel colors determined later using color, texture, and other vertex properties

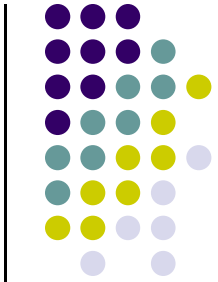
Rasterization



- Implemented by graphics hardware
- Rasterization algorithms
  - Lines
  - Circles
  - Triangles
  - Polygons



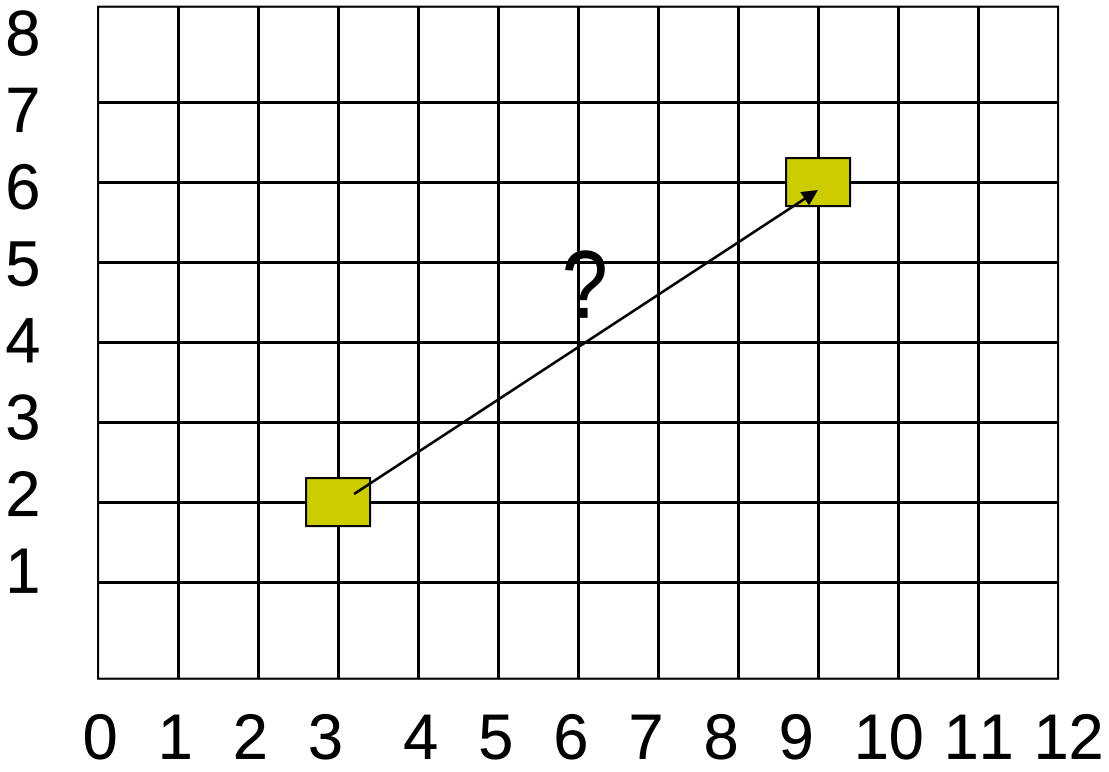
Line drawing algorithm



- Programmer specifies (x,y) values of end pixels
- Need algorithm to figure out which intermediate pixels are on line path
- Pixel (x,y) values constrained to integer values
- Actual computed intermediate line values may be floats
- Rounding may be required. E.g. computed point

  (10.48, 20.51) rounded to (10, 21)
- Rounded pixel value is off actual line path (jaggy!!)
- Sloped lines end up having jaggies
- Vertical, horizontal lines, no jaggies

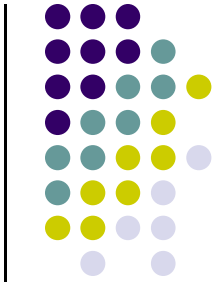
Line Drawing Algorithm



Line: (3,2) -> (9,6)

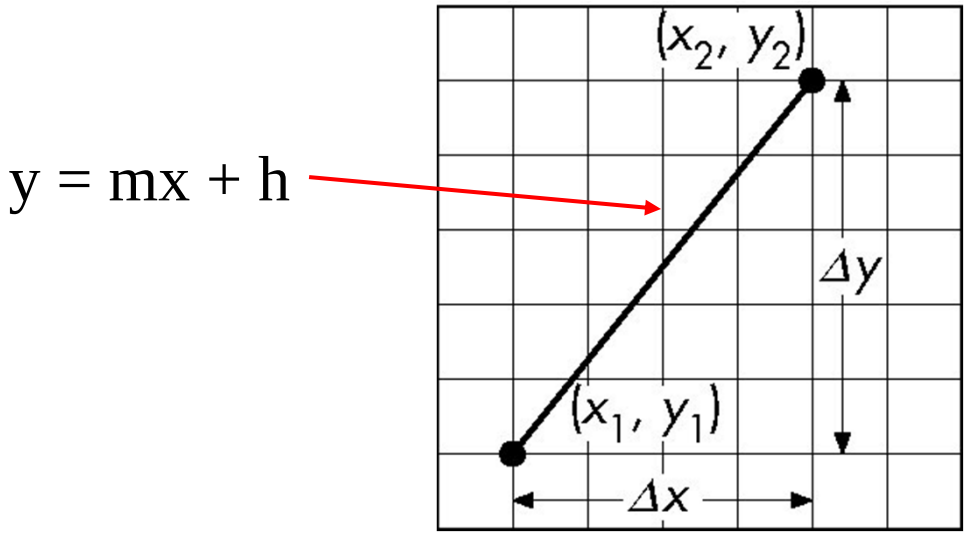
Which intermediate pixels to turn on?

Scan Conversion of Line Segments

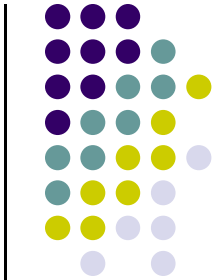


- Start with line segment in window coordinates with integer values for endpoints
- Assume implementation has a **`write_pixel`** function

$$m = \frac{\Delta y}{\Delta x}$$

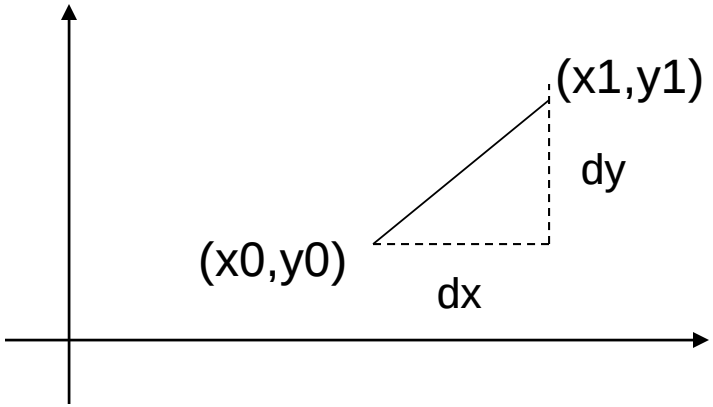


Line Drawing Algorithm

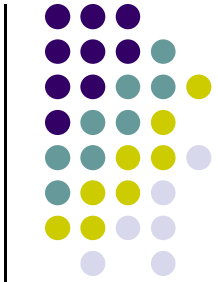


- Slope-intercept line equation
  - y = mx + b
  - Given two end points (x0,y0), (x1, y1), how to compute m and b?

$$m = \frac{dy}{dx} = \frac{y1 - y0}{x1 - x0} \qquad b = y0 - m * x0$$



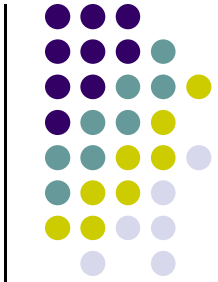
Line Drawing Algorithm



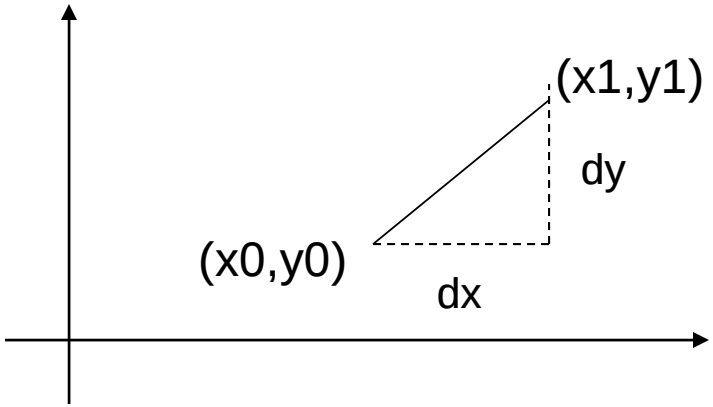
- Numerical example of finding slope m:
  - (Ax, Ay) = (23, 41), (Bx, By) = (125, 96)

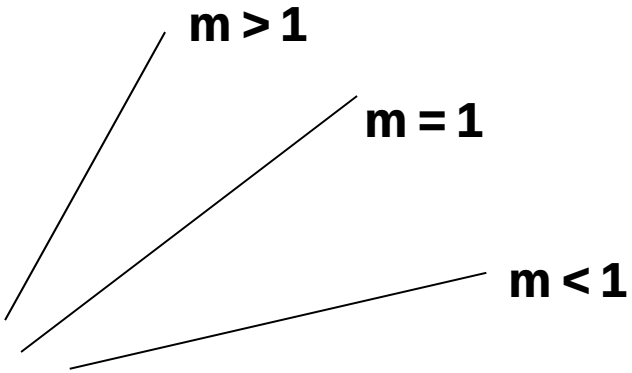
$$m = \frac{By - Ay}{Bx - Ax} = \frac{96 - 41}{125 - 23} = \frac{55}{102} = 0.5392$$

Digital Differential Analyzer (DDA): Line Drawing Algorithm



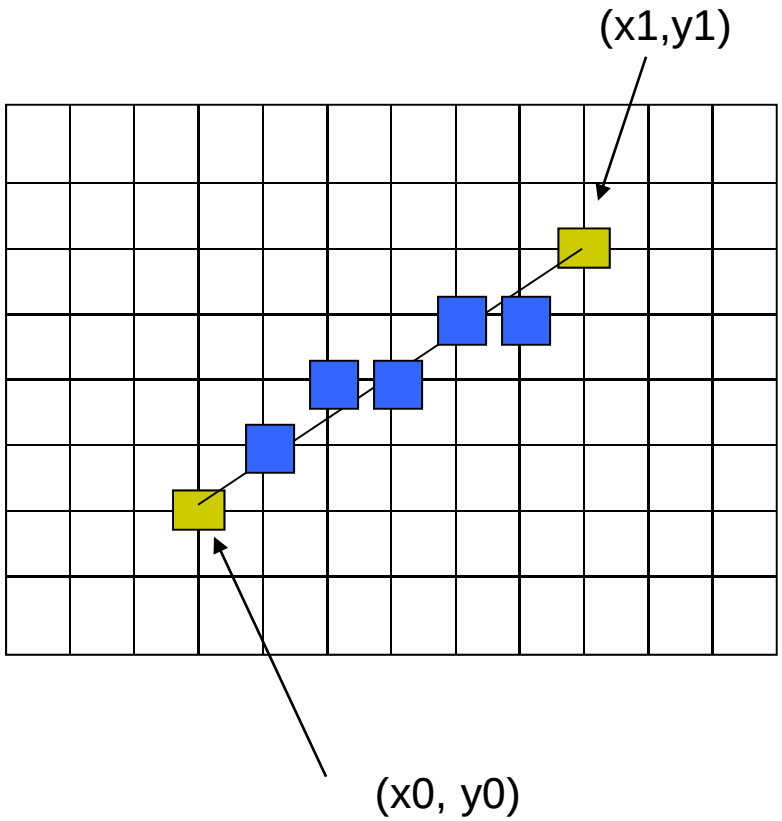
- Walk through the line, starting at (x0,y0)
- Constrain x, y increments to values in [0,1] range
- Case a: x is incrementing faster (m < 1)
  - Step in x=1 increments, compute and round y
- Case b: y is incrementing faster (m > 1)
  - Step in y=1 increments, compute and round x





DDA Line Drawing Algorithm (Case a: m < 1)

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



x = x0 y = y0

Illuminate pixel (x, round(y))

x = x0 + 1 y = y0 + 1 * m

Illuminate pixel (x, round(y))

x = x + 1 y = y + 1 * m

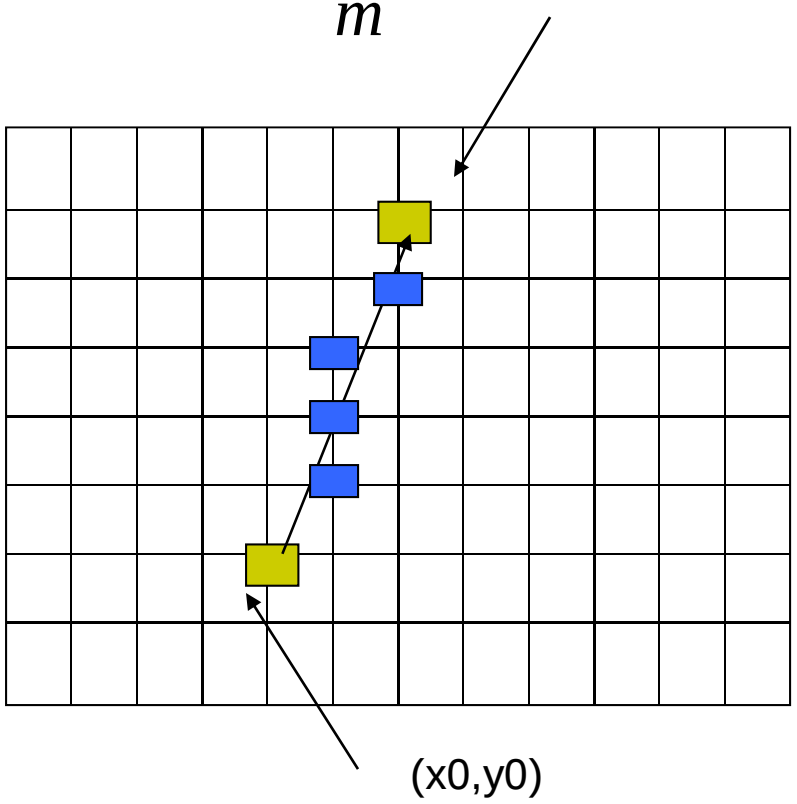
Illuminate pixel (x, round(y))

...

Until x == x1

DDA Line Drawing Algorithm (Case b: m > 1)

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



x = x0 y = y0

Illuminate pixel (round(x), y)

y = y0 + 1 x = x0 + 1 * 1/m

Illuminate pixel (round(x), y)

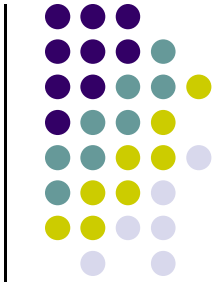
y = y + 1 x = x + 1 /m

Illuminate pixel (round(x), y)

...

Until y == y1

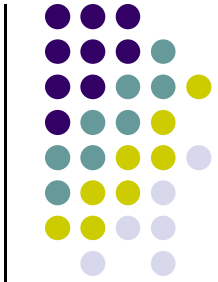
DDA Line Drawing Algorithm Pseudocode



```
compute m;
if m < 1:
{
   float y = y0;                // initial value
   for(int x = x0;x <= x1; x++, y += m)
                                setPixel(x, round(y));
}
else // m > 1
{
   float x = x0;        // initial value
   for(int y = y0;y <= y1; y++, x += 1/m)
               setPixel(round(x), y);
}
```

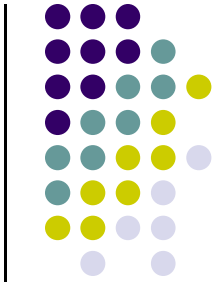
- Note: **`setPixel(x, y)`** writes current color into pixel in column x and row y in frame buffer

Line Drawing Algorithm Drawbacks



- DDA is the simplest line drawing algorithm
  - Not very efficient
  - Round operation is expensive
- Optimized algorithms typically used.
  - Integer DDA
  - E.g.Bresenham algorithm (Hill)
- Bresenham algorithm
  - Incremental algorithm: current value uses previous value
  - Integers only: avoid floating point arithmetic
  - Several versions of algorithm: we'll describe midpoint version of algorithm

References



- Angel and Shreiner, Interactive Computer Graphics, 6th edition
- Hill and Kelley, Computer Graphics using OpenGL, 3rd edition, Chapter 9