Low-level algorithms

Curve generation
Polygon fill
Flood fill

Curve generation

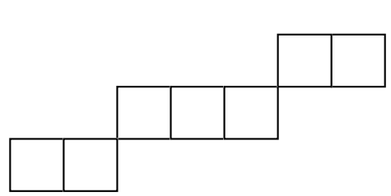
Problem: Generate a digital curve

Find a connected sequence of discrete pixels that follows the curve as closely as possible

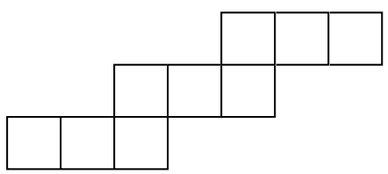
The curve should be either 4-connected or 8-connected, one pixel wide

Connectivity

8-connected: horizontal, vertical and diagonal moves are allowed:

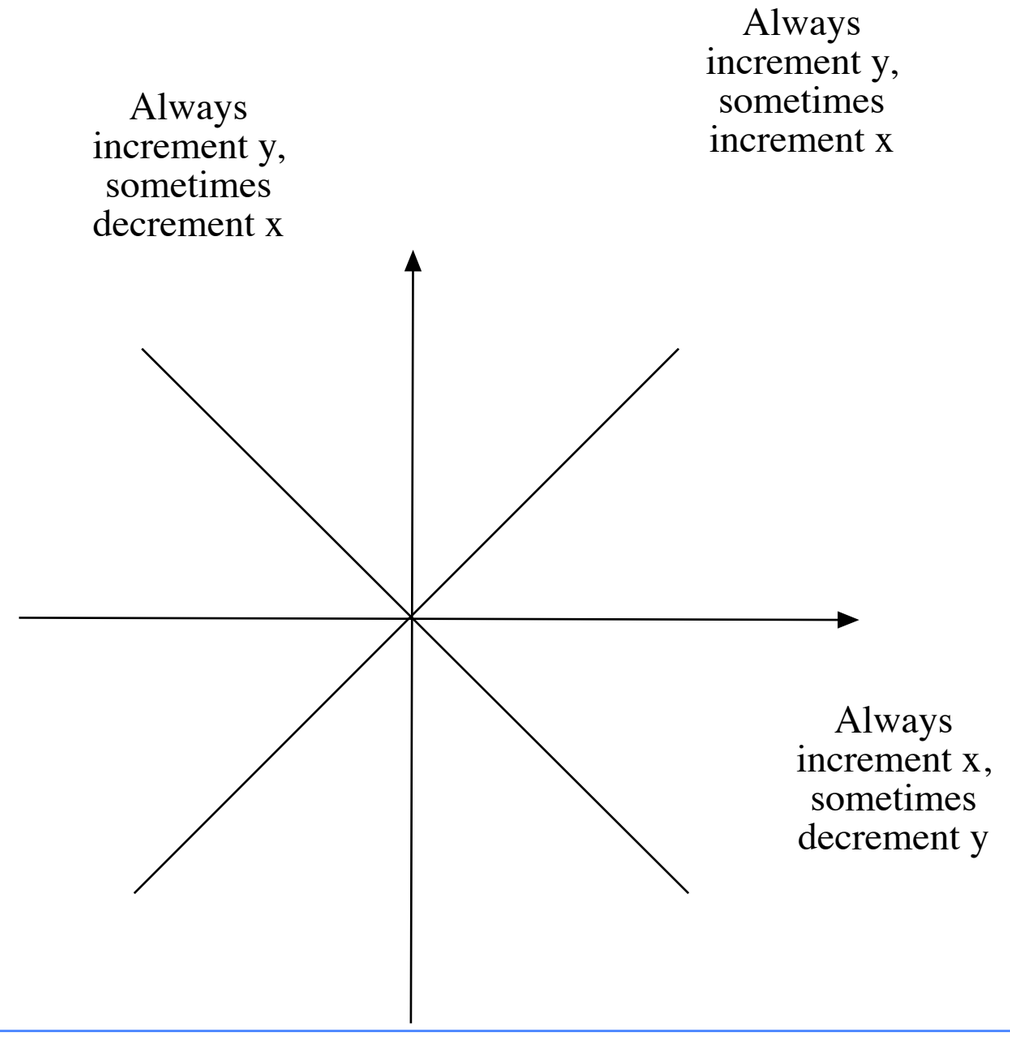


4-connected: diagonal moves are not allowed:



Choose one, don’t mix them! 8-connectedness most common for curve generation.

We need to move differently in different directions



Two line drawing algorithms:

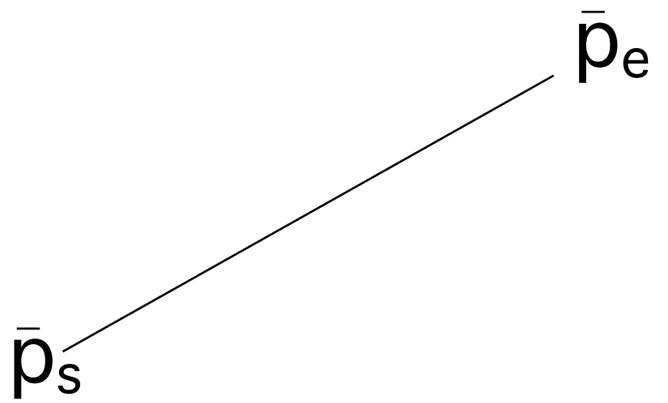
The DDA algorithm:

Simple but slow on low-end hardware

The Bresenham algorithm:

Extremely fast on low-end hardware

Line definition



$\bar{p}_s = (x_s, y_s)$
$\bar{p}_e = (x_e, y_e)$

$y = mx + b$

$m = (y_s - y_e) / (x_s - x_e) = \Delta y / \Delta x$

$b = y_s - mx_s$

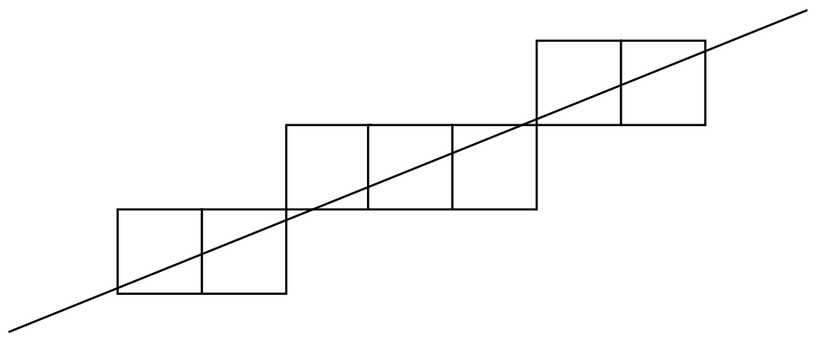
DDA (Digital Differential Algorithm)

Assume $-1 < m - 1$, $x_e > x_s$

$x := x_s$
$y := y_s$

repeat

DrawPixel(x, round(y)
$x \mathrel{+}= 1$
$y \mathrel{+}= m$
until $x > x_e$



Bresenham's line drawing algorithm

Bresenham’s algorithm

Integer-based
Incremental; Additions and shifts only

Exceptionally fast on low-end hardware.

Manipulate the line equation to find integer-based "decision variable".

Bresenham's line drawing algorithm

$$p_0 = 2\Delta y - \Delta x$$

Inspect p to decide step (if y should change)

Increment by

Horizontal move: $2\Delta y$ (p goes up)

Diagonal move: $2\Delta y - 2\Delta x$ (p goes down)

Line drawing, summary

DDA algorithm

Floating-point
Simple and straightforward

Bresenham’s algorithm

Integer-based
Incremental; Additions and shifts only
Ideal for low-power hardware

When do I need a line drawing algorithm?

Drawing lines: Rarely. You probably have a well optimized algorithm in any library.

BUT it can be used for other purposes. For example ray marching! (Ray-casting in grid!) Very similar to Bresenham's algorithm!

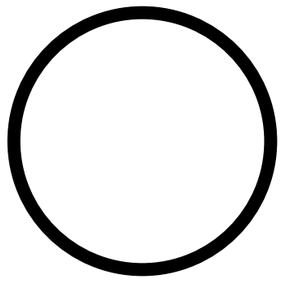
Other curves

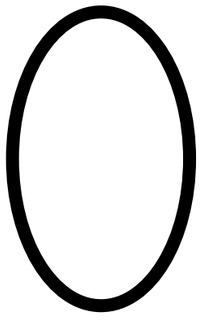
Midpoint algorithm

Any curve that can be expressed by polynomial

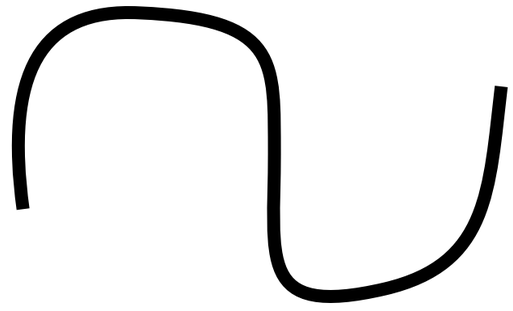
”Midpoint” refers to measurements at the midpoint between candidates

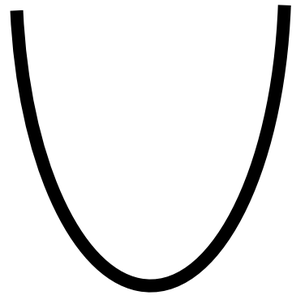
The midpoint algorithm can draw (with excellent speed)





- Circles
- Ellipses
- Parabolas
- Most splines





Related simplifications of polynomial functions

1) Horner's Rule

2) Forward-difference calculations

Horner's Rule

$$a \cdot u^3 + b \cdot u^2 + c \cdot u + d = ((a \cdot u + b) \cdot u + c) \cdot u + d$$

6 multiplications became 3

Forward-difference calculations

Like with Bresenham, calculate the difference of a function $f(x+\Delta) - f(x)$

Repeat to get higher order differentials.

Can reduce all polynomial functions to additions only (if Δ is constant)!

OpenGL vs line and point drawing

For OpenGL, everything are polygons!

Even lines and points are drawn with polygons.

-> Simplifies the optimized OpenGL kernel