Lines

Equation of a line : $y = mx + b$ m = slope, b = y-intercept

If we have start of (x_1, y_1) and end of (x_2, y_2), we have...

Slope : $m = \frac{y_2 - y_1}{x_2 - x_1}$ Y-Intercept : $b = y_1 - mx_1$

Y Interval : $\Delta y = m \Delta x$ X Interval : $\Delta x = \frac{\Delta y}{m}$

Types of Line Drawing Algorithms

Drawing a line is also known as rasterization; convert a conceptual line to a raster display. A term we have seen before, and will use again when speaking of drawing other primitives.

Scan Conversion Techniques

- Slope-Intercept (DeltaX = 1 or DeltaY = 1)
- Bresenham (plus Symmetric)
- Midpoint
- Two-Step
- Symmetric Two-Step

Digital Differential Analyzer (DDA)

Sample the equation of a line at unit intervals in either the X or Y coordinate.

Step 1 : Compute ΔX & ΔY

- $\Delta X = X_2 - X_1$
- $\Delta Y = Y_2 - Y_1$

Step 2 : Determine How Many Points to Compute

if $|(\Delta X)| > |(\Delta Y)|$ then
NumPoints = $|(\Delta X)| + 1$
else
NumPoints = $|(\Delta Y)| + 1$

Step 3 : Compute X & Y Increments

$\delta x = \Delta X / NumPoints$

$\delta y = \Delta Y / NumPoints$

Step 4 : Plot The Points

```
plotX = x1
plotY = y1
while PointCurrent <= NumPoints
    plotX = x1 + PointCurrent * δx
    plotY = y1 + PointCurrent * δy
```

Problems With DDA

1. Initial Divisions
2. Round off error can lead to drift : $Y_{k+1} = Y_k + m$ where Y_{k+1} (int) Y_k (int) m (float)
3. Memory overhead of floating point
4. Float to Integer conversion

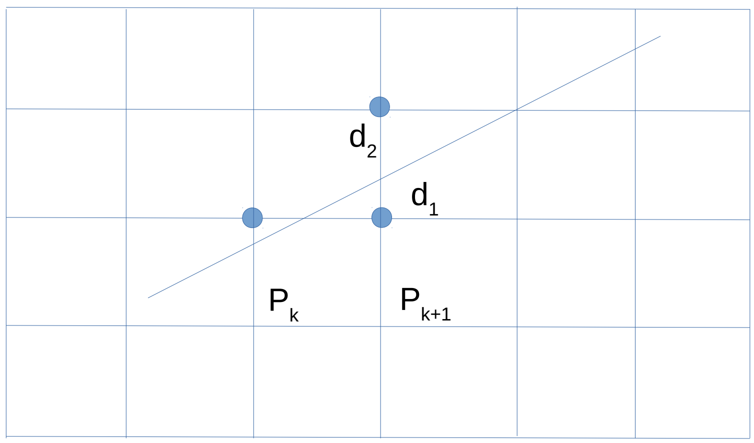
We don’t care too much about 1 and 3 now, computers are FAST; float and int operations happen at same speed on today’s CPUs. Regardless, there are still faster approaches.

Bresenham Algorithm

Line scan conversion that uses only incremental integer calculations.

Concept: Difference of distances, using a decision parameter P_k

Again, equation of a line: $y = mx + b$ This algorithm valid where: $0 \leq m \leq 1$



P_k is the current decision parameter

P_{k+1} is the next decision parameter

d_1 is the distance from the y-intercept at P_{x+1} : y - y_k or $m(x_k + 1) + b - y_k$

d_2 is the distance from the y-intercept at P_{x+1} : (y_k + 1) - y or $y_k + 1 + m(x_k + 1) - b$

The next pixel to draw is:

$$x_{k+1} = x_k + 1$$

$$y_{k+1} = d_1 - d_2 \geq 0 \,?\, y_k + 1 : y_k$$

if $d_1 - d_2$ is positive, then y_k+1 else y_k

We use this to create a Decision Parameter, P_k, below...

Difference of Distances

$d_1 - d_2 = (y - y_k) - (y_k + 1) - y$ (substituting from above)

$$d_1 = y - y_k \rightarrow m(x_k + 1) + b - y_k$$

$$d_2 = (y_k + 1) - y \rightarrow y_k + 1 - m(x_k + 1) - b$$

$$d_1 - d_2 = m(x_k + 1) + b - y_k - y_k - 1 + m(x_k + 1) + b$$

$$d_1 - d_2 = 2m(x_k + 1) - 2_{y_k} + 2b - 1$$

where

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

$d_1 - d_2$ is floating point arithmetic, we want it (P_k) in terms of integer arithmetic. We can do this by substituting m with $\frac{\Delta y}{\Delta x}$

$$P_k=\Delta x(d_1-d_2)=\Delta x(\frac{\Delta y}{\Delta x}(x_k+1)+b)-\Delta x\, y_k-\Delta x\, y_k-\Delta x+\Delta x(\frac{\Delta y}{\Delta x}(x_k+1)+b)$$

$$P_k=2\Delta x(\frac{\Delta y}{\Delta x}(x_k+1)+b)-2\Delta x\, y_k-\Delta x$$

...eventually…

$$P_k=2\Delta y\, x_k-2\Delta x\, y_k+2\Delta y+\Delta x(2b-1)$$

Notice that $2\Delta y+\Delta x(2b-1)$ is constant for all X, can compute one time for all pixels. We will call this c : $c=2\Delta y+\Delta x(2b-1)$

$$P_k=2\Delta y\, x_k-2\Delta x\, y_k+c$$

$$P_{k+1}=2\Delta y\, x_{k+1}-2\Delta x\, y_{k+1}+c$$

$P_{k+1}-P_k=2\Delta y(x_{k+1}-x_k)-2\Delta x(y_{k+1}-y_k)$ Remember $x_{k+1}=x_k+1$

$$P_{k+1}-P_k=2\Delta y(x_k-x_k+1)-2\Delta x(y_{k+1}-y_k)$$

$$P_{k+1}-P_k=2\Delta y-2\Delta x(y_{k+1}-y_k)$$

$P_{k+1}=P_k+2\Delta y(x_k-x_k+1)-2\Delta x(y_{k+1}-y_k)$ ← Decision Parameter!

Here is how we use it (again)

if $P_k\geq 0\, then\; y_{k+1}=y_k+1$

$$P_{k+1}=P_k+2\Delta y-2\Delta x$$

if $P_k<0\, then\; y_{k+1}=y_k$

$$P_{k+1}=P_k+2\Delta y$$

What about P_0 ?

$$P_k=2\Delta y\, x_k-2\Delta x\, y_k+c$$

$$b=y_1-m x_1 \qquad m=\frac{\Delta y}{\Delta x}$$

$$P_0=2\Delta y-\Delta x$$

Guess what else? The algorithm is symmetric! Can draw two pixels from a single calculation.