Scan Conversion - Lines

- Idea: Use Explicit Line Formula

Explicit - one coordinate as function of the others

\[ y = f(x) \]
\[ z = f(x,y) \]

(line)

\[ y = mx + b \]
\[ y = \frac{(y_2 - y_1)}{(x_2 - x_1)} (x - x_1) + y_1 \]

Typically separate into 4 (or 8) cases (why?)
Basic Line Drawing

Assume \( s \leq t \) & line slope absolute value is \( \leq 1 \)

```plaintext
Line \((x_1, y_1, x_2, y_2)\):
begin
int dx = |\(x_2 - x_1\)|, dy = |\(y_2 - y_1\)|;
int x = \(x_1\), y = \(y_1\);
PlotPixel(x,y); // starts at \((x_1, y_1)\)
for \( i = 0 \) to \( dx \) do
begin
  \( y = y + \text{sign}(dy) \);
  PlotPixel(x,y);
  \( x = x + \text{sign}(dx) \);
end
end
```

Questions:
Can this algorithm use integer arithmetic?

Midpoint (Bresenham) Algorithm

- **Key Observation 1:**
  - At each step have ONLY 2 choices
  - East/North-East

- **Key Observation 2:**
  - Can decide based on whether midpoint is above/below line
  - How?
    - Evaluate implicit line equation at \((x+1, y+1/2)\)

Bresenham (Midpoint) Algorithm

- Starting point satisfies \( d(x_1, y_1) = 0 \)
- Each step moves right (east) or upper right (northeast)
- Sign of \( d(x+1; y+\frac{1}{2}) \) indicates if to move east or northeast

Midpoint (Bresenham) Algorithm

- **Key Observation 1:**
  - At each step have ONLY 2 choices
  - East/North-East

- **Key Observation 2:**
  - Can decide based on whether midpoint is above/below line
  - How?
    - Evaluate implicit line equation at \((x+1, y+1/2)\)

Bresenham Algorithm

- Implicit formulation = distance (up to scale)
  \[ d(x, y) = 2(\text{scale} \cdot y - ydx + c) \]

- Given point \( P = (x, y) \) \( d(x, y) \) is signed distance of \( P \) to \( e \) (up to scale)
- \( d \) is zero for \( P \in e \)
Computer Graphics

Scan Conversion

Bresenham (Midpoint) Algorithm

- Insanely efficient version (less computations inside the loop)
  - compute $d$ incrementally
- At $(x_i, y_i)$
  $$d_{new} = d_{old} + 2y - dx$$
- Increment in $d$ (after each step)
  - If move east
  $$d_{new} = d_{old} + 2y - dx$$
  - If move northeast
  $$d_{new} = d_{old} + 2y - dx$$

Bresenham Examples

- Intensity depends on angle
- Comment: extends to higher order curves - e.g. circles

Comparison: float/integer

Assume $x_i < x_f$, & line slope is $< 1$

Scan Conversion of Lines

Discussion

- Integer: Bresenham
  - Good for hardware implementations (integer!)
- Floating Point
  - May be faster for software (depends on system!)
  - Easier to parallelize

Implicit test

- Instead of clipping line in continuous space
  - For each integer value of $(x,y)$ test if inside window just before drawing
  - Inefficient on CPU
  - On a parallel (GPU) processor can be surprisingly fast