**Outline**

- Line generation problem
- Straightforward approach
- Midpoint line algorithm

**Line Generation Problem**

- How to display a straight line from $(x_0, y_0)$ to $(x_1, y_1)$ on a monitor?
  - Select pixels that are closest to the line
  - Do it efficiently
- Simplifications:
  - Lines are 1 pixel wide
  - No need for anti-aliasing

**Line Equations**

- Explicit representation:
  - $y = m \cdot x + b$
  - $m = \frac{\Delta y}{\Delta x}$ (slope)
  - $b = y_0 - m \cdot x_0$ (Y intercept)
- Parametric function:
  - $x = x_0 + k \cdot \Delta x$
  - $y = y_0 + k \cdot \Delta y$
- Implicit representation:
  - $F(x, y) = ax + by + c = 0$
  - $a = \Delta y, b = -\Delta x$
  - $c = \Delta x \cdot y_0 - \Delta y \cdot x_0$

**Straightforward Approach I**

- Based on explicit line representation
- Algorithm:
  - float $m = \frac{\Delta y}{\Delta x}$;
  - for ( int $x=x_0$; $x<=x_1$; $x++$ ) {
    - float $y = y_0 + (x - x_0) \cdot m$;
    - draw($x$, round($y$));
  }
- Computational cost:
  - $n$ times multiplication
  - $2n$ times addition/subtraction

**Incremental Version**

- Algorithm:
  - float $m = \frac{\Delta y}{\Delta x}$;
  - float $y = y_0$;
  - for ( int $x=x_0$; $x<=x_1$; $x++$ ) {
    - $y += m$;
    - draw($x$, round($y$));
  }
- Computational cost:
  - $n$ times addition
Limitations

- Missing pixels
- Works great if \(|m| < 1\)
- Must reverse the roles of \(x\) and \(y\) if \(|m| > 1\)
- Error accumulation
- Slope \(m\) is a fractional number
- May not be able to precisely represented in a computer

Straightforward Approach II

- Based on parametric representation
- Incremental algorithm:
  - float \(dx = dk \times \Delta x\);
  - float \(dy = dk \times \Delta y\);
  - for ( float \(k = 1\) ; \(k = 1\) ; \(k += dk, x += dx, y += dy\) )
  - draw(\(\text{round}(x)\), \(\text{round}(y)\));
- Limitations:
  - Computational cost depends on the value of \(dk\)
  - Set \(dk = 1 / \max(\Delta x, \Delta y)\)
  - No missing pixel problem but error accumulation still exists

Midpoint Line Algorithm

- Origin:
  - First proposed by Bresenham in 1965.
  - Improved by Pitteway in 1967
- Advantage:
  - Incremental approach
  - Use integer arithmetic only
  - Fast
  - No error accumulation

Eight Cases

- \(\Delta x > 0, \Delta y > 0, |\Delta x| > |\Delta y|\)
- \(\Delta x > 0, \Delta y < 0, |\Delta x| > |\Delta y|\)
- \(\Delta x < 0, \Delta y > 0, |\Delta x| > |\Delta y|\)
- \(\Delta x < 0, \Delta y < 0, |\Delta x| > |\Delta y|\)
- \(\Delta x > 0, \Delta y > 0, |\Delta y| > |\Delta x|\)
- \(\Delta x < 0, \Delta y < 0, |\Delta y| > |\Delta x|\)
- \(\Delta x > 0, \Delta y < 0, |\Delta y| > |\Delta x|\)
- \(\Delta x < 0, \Delta y > 0, |\Delta y| > |\Delta x|\)

Consider Case 1 First

- \(\Delta x > 0, \Delta y > 0, \Delta x > \Delta y, |\Delta x| > |\Delta y|\)
- Others can be handled by suitable reflections about the principle axes
  - Case 8 is reflection over \(x\) axis
  - Case 4 is reflection over \(y\) axis
- ...

Basic Idea

- Assume \((u, v)\) is selected
- Need to chose between \((u+1, v)\) and \((u+1, v+1)\)
  - If \(M\) is above the line, pick \((u+1, v)\)
  - Otherwise, pick \((u+1, v+1)\)
Relative Position of a Point

- The relative position of point \((u,v)\) to the line can be determined using the sign of the implicit line function.
- Since \(a=\Delta y>0\) & \(b=-\Delta x<0\)
- On the line: \(F(u,v)=0\)
- Above the line: \(F(u,v)<0\)
- Below the line: \(F(u,v)>0\)

First Decision

- Line starts from \((x_0,y_0)\)
  - \((x_0,y_0)\) should be picked.
  - \(M\) is \((x_0+1,y_0+\frac{1}{2})\)
- Let \(d=F(M)\):
  - \(=a(x_0+1)+b(y_0+\frac{1}{2})+c\)
  - \(=a\Delta y-b\Delta x+c-a-b/2\)
  - \(=F(x_0,y_0)+a+b/2\)
  - \(=\Delta y-\Delta x/2\)

Next Decision (Scenario 1)

- If \((u+1,v)\) is picked
  - \(M'\) is \((u+2,v+\frac{1}{2})\)
- Current \(d=F(M)\)
  - \(=a(u+1)+b(v+\frac{1}{2})+c\)
- Next \(d'=F(M')\)
  - \(=a(u+2)+b(v+\frac{1}{2})+c\)
- Difference:
  - \(d'-d=a=\Delta y\)
- Update function:
  - \(d'=d+\Delta y\)

Next Decision (Scenario 2)

- If \((u+1,v+1)\) is picked
  - \(M'\) is \((u+2,v+\frac{1}{2})\)
- Current \(d=F(M)\)
  - \(=a(u+1)+b(v+\frac{1}{2})+c\)
- Next \(d'=F(M')\)
  - \(=a(u+2)+b(v+\frac{1}{2})+c\)
- Difference:
  - \(d'-d=a+b=\Delta y-\Delta x\)
- Update function:
  - \(d'=d+\Delta y-\Delta x\)

Overall Algorithm

- int \(x=x0, y=y0\);
- draw(\(x, y\));
- float \(d=\Delta y-\Delta x / 2.0\);
- while (\(x < x1\)) {
  if (\(d <= 0\)) {
    \(d += \Delta y\);
    \(x ++\);
  } else {
    \(d += \Delta y-\Delta x\);
    \(x ++; y ++\);
  }
  draw(\(x, y\));
}

Pure Integer Implementation

- int \(x=x0, y=y0\);
- draw(\(x, y\));
- int \(d2=\Delta y * 2 - \Delta x\);
- while (\(x < x1\)) {
  if (\(d2 <= 0\)) {
    \(d2 += \Delta y * 2\);
    \(x ++\);
  } else {
    \(d2 += (\Delta y - \Delta x) * 2\);
    \(x ++; y ++\);
  }
  draw(\(x, y\));
}