### Line Drawing

- **Outline**
  - Line drawing
  - Digital differential analyzer
  - Bresenham’s algorithm

### Scan-Conversion Algorithms

- **Line Drawing**
  - Scan-Conversion: Computing pixel coordinates for ideal line on 2D raster grid
  - Pixels best visualized as circles/dots
    - Why? Monitor hardware

### Drawing a Line

- \( y = mx + B \)
- \( m = \Delta y / \Delta x \)
- Start at leftmost \( x \) and increment by 1
  - \( \Delta x = 1 \)
- \( y_i = \text{Round}(mx_i + B) \)
- This is expensive and inefficient
  - Since \( \Delta x = 1 \), \( y_{i+1} = y_i + \Delta y = y_i + m \)
  - No more multiplication!
- This leads to an incremental algorithm

### Digital Differential Analyzer (DDA)

- If [slope] is less than 1
  - \( \Delta x > \Delta y \)
- Check for vertical line
  - \( \Delta y = 0 \)
- Compute corresponding \( \Delta y (\Delta x) = m (1/\Delta x) \)
  - \( x_{i+1} = x_i + \Delta x \)
  - \( y_{i+1} = y_i + \Delta y \)
- Round \((x,y)\) for pixel location
- Issue: Would like to avoid floating point operations
Generalizing DDA

- If |slope| is less than or equal to 1
  - Ending point should be right of starting point
- If |slope| is greater than 1
  - Ending point should be above starting point
- Keep x and y as floating point values
- Vertical line is a special case
  \[ \Delta x = 0 \]

Bresenham’s Algorithm

- 1965 @ IBM
- Basic Idea:
  - Only integer arithmetic
  - Incremental
- Consider the implicit equation for a line:
  \[ f(x,y) = ax + by + c = 0 \]

The Algorithm

Assumptions: \( Q_x < Q_y \)
\( 0 \leq \text{slope} \leq 1 \)

Bresenham’s Algorithm

Given:
- implicit line equation: \( f(x,y) = ax + by + c = 0 \)
- Let: \( d_x = r_x - q_x, d_y = r_y - q_y \)
  where \( r \) and \( q \) are points on the line and \( d_x \) is positive
  \[ i = d_x, b = -d_y, c = -(q_x r_y - r_x q_y) \]
Then:
- Observe that all of these are integers
- if \( f(x,y) < \zeta \) for points above the line
- \( f(x,y) > 0 \) for points below the line
Now:....

Bresenham’s Algorithm

Assume:
- \( Q = \) exact \( y \) value at \( x = p_y + 1 \)
- \( y \) midway between \( E \) and \( NE \): \( M = p_y + 1/2 \)
Observe:
- If \( Q < M \), then pick \( E \)
- Else pick \( NE \)
- If \( Q = M \), it doesn’t matter
Bresenham’s Algorithm

- Create “modified” implicit function (2x)
  \[ f(x, y) = 2ax + 2by + 2c = 0 \]
- Create a decision variable \( D \) to select, where \( D \) is the value of \( f \) at the midpoint:
  \[
  D = f(p_x + 1, p_y + (1/2)) \\
  = 2a(p_x + 1) + 2b\left(p_y + \frac{1}{2}\right) + 2c \\
  = 2ap_x + 2bp_y + (2a + b + 2c)
  \]
- If \( D > 0 \) then \( M \) is below the line \( f(x, y) \)
  - \( NE \) is the closest pixel
- If \( D \leq 0 \) then \( M \) is above the line \( f(x, y) \)
  - \( E \) is the closest pixel
- Note: because we multiplied by 2x, \( D \) is now an integer—which is very good news
- How do we make this incremental??

Case I: When \( E \) is next

- What increment for computing a new \( D \)?
- Next midpoint is: \( (p_x + 2, p_y + (1/2)) \)
  \[
  D_{\text{new}} = f(p_x + 2, p_y + (1/2)) \\
  = 2a(p_x + 2) + 2b\left(p_y + \frac{1}{2}\right) + 2c \\
  = 2ap_x + 2bp_y + (4a + 3b + 2c) \\
  = 2ap_x + 2bp_y + (2a + b + 2c) + (2a + 2b) \\
  = D + 2(a + b) = D + 2d_x \\
  \]
- Hence, increment by: \( 2d_x \)

Case II: When \( NE \) is next

- What increment for computing a new \( D \)?
- Next midpoint is: \( (p_x + 2, p_y + (1/2)) \)
  \[
  D_{\text{new}} = f(p_x + 2, p_y + (1/2)) \\
  = 2a(p_x + 2) + 2b\left(p_y + \frac{1}{2}\right) + 2c \\
  = 2ap_x + 2bp_y + (4a + 3b + 2c) \\
  = 2ap_x + 2bp_y + (2a + b + 2c) + (2a + 2b) \\
  = D + 2(a + b) = D + 2d_x \\
  \]
- Hence, increment by: \( 2(d_x - d_y) \)

How to get an initial value for \( D \)?

- Suppose we start at: \( (q_x, q_y) \)
- Initial midpoint is: \( (q_x + 1, q_y + 1/2) \)

Then:
  \[
  D_{\text{init}} = f(q_x + 1, q_y + 1/2) \\
  = 2a(q_x + 1) + 2b\left(q_y + \frac{1}{2}\right) + 2c \\
  = (2aq_x + 2bq_y + 2c) + (2a + b) \\
  = 0 + 2a + b \\
  = 2d_x - d_y
  \]
The Algorithm

Assumptions:

\[ 0 \leq \text{slope} \leq 1 \]

Pre-computed:

\[ 2d_y, \quad 2(d_y - d_x) \]

Generalize Algorithm

- If \( q_x > r_x \), swap points
- If \( \text{slope} > 1 \), always increment \( y \), conditionally increment \( x \)
- If \( -1 \leq \text{slope} < 0 \), always increment \( x \), conditionally decrement \( y \)
- If \( \text{slope} < -1 \), always decrement \( y \), conditionally increment \( x \)
- Rework \( D \) increments

Bresenham’s Algorithm: Example

Reflect line into first case
Calculate pixels
Reflect pixels back into original orientation
Bresenham’s Algorithm: Example

Bresenham’s Algorithm: Example

Bresenham’s Algorithm: Example

Bresenham’s Algorithm: Example

Bresenham’s Algorithm: Example

Bresenham’s Algorithm: Example
Some issues with Bresenham’s Algorithms

- Pixel “density” varies based on slope
  - straight lines look darker, more pixels per unit length
- Endpoint order
- Line from P1 to P2 should match P2 to P1
- Always choose E when hitting M, regardless of direction

Questions?
Go to Assignment 1