DDA Algorithm / Digital Differential Analyzer

-dy/dx = m (differential equation)

check:
  a) -1 ≤ m ≤ 1 (if not, loop over y)
  b) x₁ < x₂ (if not, switch vertices)

x₁, x₂, y₁, y₂ all doubles (computed screen coordinates for two vertices)
m = (y₂ − y₁) / (x₂ − x₁)
b = y₂ − m*x₂

xₘᵗᵣₜ = (int) (x₁ + 0.5) // if <0 subtract 0.5 (find closest starting pixel)
xₑₐₙₜ = (int) (x₂ + 0.5) (find closest ending pixel)
y = m*xₘᵗᵣₜ + b //this is a double

while (xₘᵗᵣₜ ≤ xₑₐₙₜ)
{
  yᵢ = (int) (y + 0.5) //if <0 subtract 0.5
  plot(xₘᵗᵣₜ, yᵢ)
  y = y + m
  xₘᵗᵣₜ++
}

(379.974, -80.2012) to (369.548, -61.7504)
note that this line will be drawn off screen
m = -1.7697
1/m = -0.56507 (loop over y)

ystart = -80 (ystart < yend)
yend = -62

369.548 = -0.56507*-61.7504 + b
b = 334.6547

x = -80*-0.56507 + 334.6547 = 379.8603 (align x with ystart)
f = (x − xi) + 0.5
yi = -80, x = 379.8603, xi = 380, plot (380, -80), f = .3603
f = f + m
x = 379.8603 − 0.56507 = 379.2952 if (f < 0) f = f + 1
yi = -79, x = 379.2952, xi = 379, plot (379, -79), f = .7952

x = 379.2952 − 0.56507 = 378.7301
yi = -78, x = 378.7301, xi = 379, plot (379, -78), f = .2301

x = 379.7301 − 0.56507 = 378.1650
yi = -77, x = 378.1650, xi = 378, plot (378, -77), f = .6650

x = 379.1650 − 0.56507 = 377.6000
yi = -76, x = 377.6000, xi = 378, plot (378, -76), f = .1000
Bresenham’s Algorithm
The DDA algorithm appears efficient. However, it requires a floating-point addition for each pixel generated. Bresenham derived a line-rasterization algorithm, which, remarkably, avoids all floating-point calculations and has become the standard algorithm used in rasterizers.

Optimizations:
1. Integer Operations
2. Eliminate Divisions
3. Compare to 0

First Improvement:
Let y increase by 0 or 1
Keep track of the fraction separately

\[ m = \frac{dy}{dx} \leq 1 \]
\[ x_{\text{curr}} = (\text{int}) (x_1 + 0.5) \]
\[ x_{\text{end}} = (\text{int}) (x_2 + 0.5) \]
\[ y = mx_{\text{curr}} + b \quad \text{//this is a double} \]

\[ y_i = (\text{int}) (y + 0.5) \quad \text{//if <0 subtract 0.5} \]
\[ \text{fraction} = y - y_i + 0.5 \quad \text{//fraction is a double and is set to include correct rounding} \]
while \((x_{\text{curr}} \leq x_{\text{end}})\) \text{//above line: “align” y with xcurr (since xcurr rounded, what is y?)}
{
    plot(x_{\text{curr}}, y_i);
    x_{\text{curr}}++;

    \text{fraction} += m; \quad \text{//m can be positive or negative here}

    \text{if (fraction} > 1) \}
    \{
        y_i++;
        \text{fraction} -= 1
    \}
else if (fraction < 0)
    \{
        y_i--;
        \text{fraction} += 1
    }
}

yi = -80, f = .3603, xi = 380, plot (380, -80)

yi = -79, f = -0.20477, xi = 379, f = .7952, plot (379, -79)
yi = -78, f = 0.23016, xi = 379, f = plot (379, -78)
yi = -77, f = -0.33491, xi = 378, f = .66509, plot (378, -77)
yi = -76, f = 0.10002, xi = 378, plot (378, -76)
Second Improvement:

**dy/dx is rational**  //don’t keep track of any floats anymore

**let dy = yi – yi**
**and dx = xi – xi**

Initial fraction = 0.5 (again, include correct rounding)  //there is some error here

Multiply through by 2dx

Initial fraction = dx

If (dx > dy) then slope < 1, loop over x

x' = (int) (x1 + 0.5)  //if < 0 subtract 0.5
x' = (int) (x2 + 0.5)

//repeat above for yi and y' end

dx = x' - x' end  //make always positive (use stepx to store the sign)
dy = y' - y' end  //make always positive (use stepy to store the sign)

Fraction = dx;  //there is some error here as fraction = yi – yi + 0.5

//since we are no longer keeping doubles, don’t keep track of y
//thus, fraction = 0.5 since y = y' = yi initially

I made dx and dy positive, and stored the sign in stepx and stepy variables.

While (x' != x' end + stepx)

{
    plot(x', y')
    x' = x' + stepx

    Fraction += 2dy  //multiply through by 2dx (was dy/dx)

    If (fraction > 2dx)  //multiply through by 2dx (was 1) (**left shift to mult by 2**) { 
        y' = y' + stepy
        Fraction -= 2dx  //multiply through by 2dx (was 1)
    }
}
Third Improvement:  
**Test against zero**

Subtract 2dx from fraction

if \( dx > dy \) then slope < 1, loop over x

\[ x_{\text{curr}} = \text{(int)} (x_1 + 0.5) \]  //if < 0 subtract 0.5

\[ x_{\text{end}} = \text{(int)} (x_2 + 0.5) \]

//repeat above for \( y_{\text{curr}} \) and \( y_{\text{end}} \)

\[ dx = x_{\text{curr}} - x_{\text{end}} \quad \text{//make always positive (use stepx)} \]

\[ dy = y_{\text{curr}} - y_{\text{end}} \quad \text{//make always positive (use stepy)} \]

\[ \text{fraction} = -dx; \quad \text{//subtract 2dx (was dx)} \]

while \( x_{\text{curr}} \neq x_{\text{end}} + \text{stepx} \)
{
    plot\( (x_{\text{curr}}, y_{\text{curr}}) \);
    \[ x_{\text{curr}} = x_{\text{curr}} + \text{stepx} \]

    \[ \text{fraction} += 2dy \]

    if \( \text{fraction} > 0 \)  //test against zero (subtract 2dx)
    {
        \[ y_{\text{curr}} = y_{\text{curr}} + \text{stepy} \]
        \[ \text{fraction} -= 2dx \]
    }
}

dx and dy both positive

\[ dx = x_{\text{curr}} - x_{\text{end}} = 380 - 370 = 10 \]

stepx = -1 (x decreases)

\[ dy = y_{\text{curr}} - y_{\text{end}} = -80 - (-62) = -18 \Rightarrow dy = 18 \]

stepy = +1 (y increases)

\( dy > dx \), loop over y

plot\( (380, -80), f = -dy = -18 \)

\( y_{\text{curr}} = -79, f = 2 \Rightarrow x_{\text{curr}} = 379, f = -34, \) plot\( (379, -79) \)

\( y_{\text{curr}} = -78, f = -14, \) plot\( (379, -78) \)

\( y_{\text{curr}} = -77, f = 6 \Rightarrow x_{\text{curr}} = 378, f = -30, \) plot\( (378, -77) \)

\( y_{\text{curr}} = -76, f = -10, \) plot\( (378, -76) \)

compares well with DDA