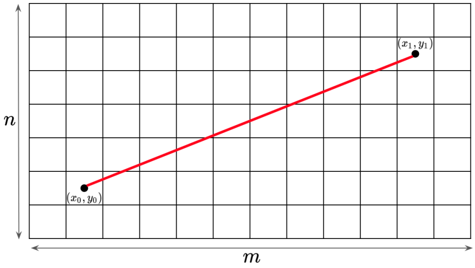
CSL7450: Computer Graphics

Rajendra Nagar

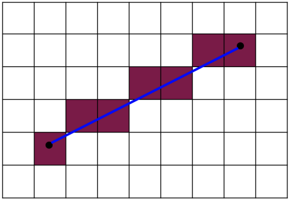
Trimester 2, 2020-2021
Indian Institute of Technology Jodhpur
sites.google.com/iitj.ac.in/csl7450/

December 5, 2020

How to Draw a Line on a screen?



Algorithm 1: $(x_{i+1}, \texttt{round}(m \times x_{i+1} + c))$
Algorithm 2: $(x_{i+1}, \texttt{round}(y_i + m))$



Jaggies or Staircasing

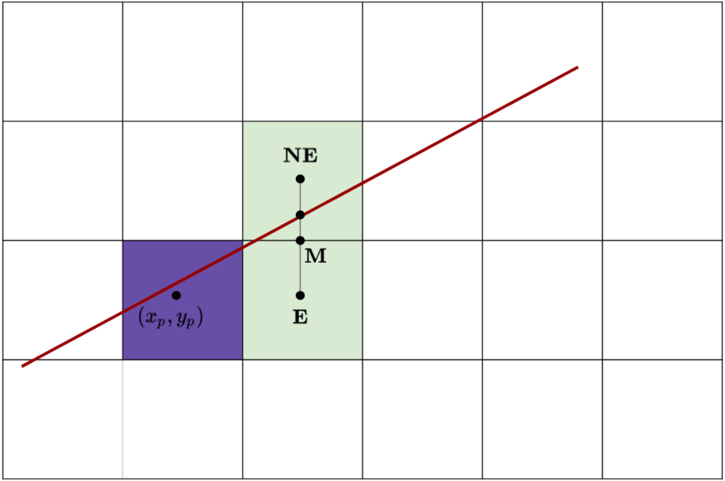
Can we eliminate the rounding operation?

Draw at $(x_{i+1}, \texttt{round}(y_i + m))$.

- Bresenham developed a classic algorithm that is attractive because it uses only integer arithmetic [1].
- Thus eliminating the round operation.
- Allows the calculation for (x_{i+1}, y_{i+1}) by using the calculation already done at (x_i, y_i).

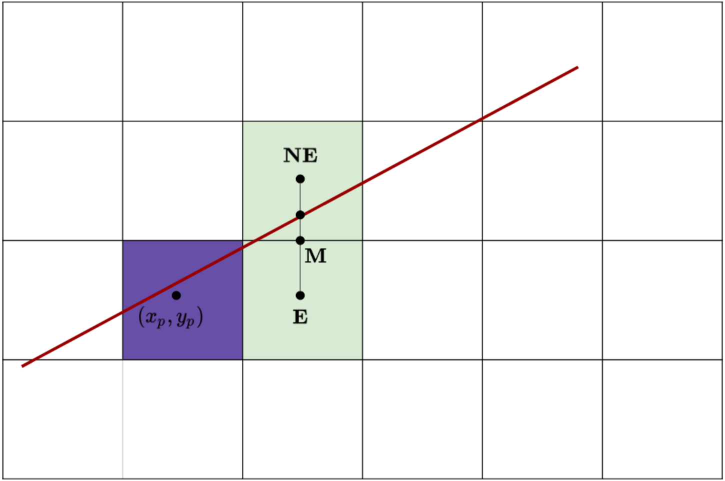
[1] Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. IBM Systems journal, 4(1), 25-30.

Midpoint Line Algorithm



- We have to determine on which side of the line the midpoint $\mathbf{M}$ lies.
- If the midpoint lies above the line, pixel $\mathbf{E}$ is closer to the line.
- If the midpoint lies below the line, pixel $\mathbf{NE}$ is closer to the line.

Midpoint Line Algorithm



- The algorithm chooses $\mathbf{NE}$ as the next pixel for this example.
- Now, we need to come up with a way to calculate on which side of the line the midpoint lies.

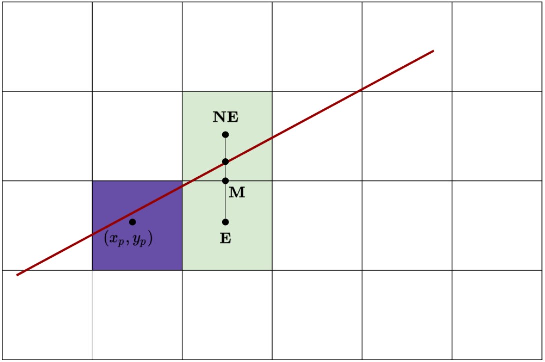
Midpoint Line Algorithm

- Let $F(x, y) = ax + by + c = 0$ be the equation of the line and $\Delta x = x_1 - x_0$ and $\Delta y = y_1 - y_0$, then slope-intercept form can be written as

$$\begin{aligned} y &= \frac{\Delta y}{\Delta x} x + c_0 \\ y\Delta x &= x\Delta y + c_0 \Delta x \\ 0 &= x\Delta y - y\Delta x + c_0 \Delta x. \end{aligned}$$

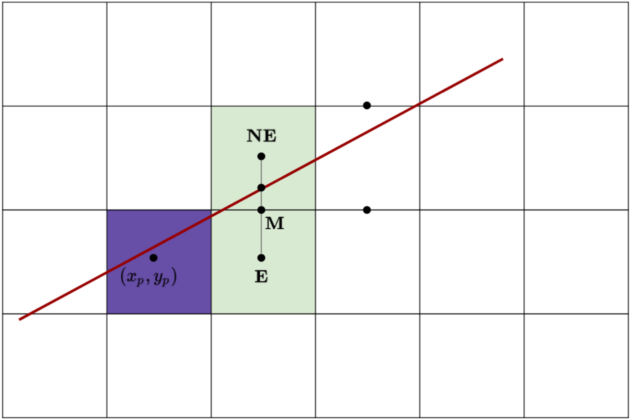
$\Rightarrow a = \Delta y, b = -\Delta x, c = c_0 \Delta x.$

Midpoint Line Algorithm



- $F(x, y) = 0$ for points on the line, $F(x, y) > 0$ for points below the line, and $F(x, y) < 0$ for points above the line.
- Define $d = F(\mathbf{M}) = F(x_p + 1, y_p + \frac{1}{2}) = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$.
- If $d > 0$, we choose pixel $\mathbf{NE}$, if $d < 0$, we choose $\mathbf{E}$ and if $d = 0$, we can choose either.

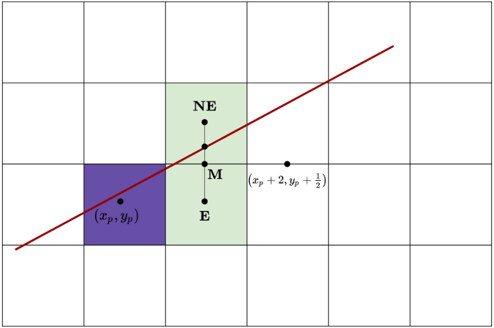
Midpoint Line Algorithm



- Next, we ask what happens to the location of $\mathbf{M}$ and therefore to the value of d for the next grid line , both depend, of course, on whether we chose $\mathbf{E}$ or $\mathbf{NE}$.

Midpoint Line Algorithm

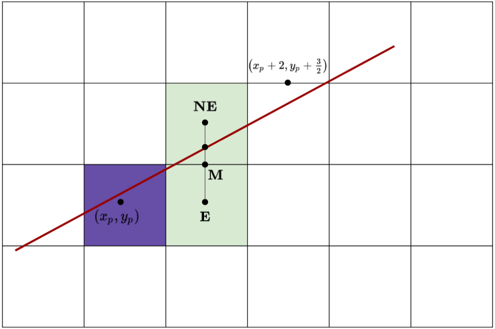
- If $\mathbf{E}$ is chosen , $\mathbf{M}$ is incremented by one step in the x direction. Then,



$$
\begin{aligned}
d_{\text{new}} &= F\left(x_p + 2, y_p + \frac{1}{2}\right) \\
&= a(x_p + 2) + b(y_p + \frac{1}{2}) + c \\
d_{\text{old}} &= a(x_p + 1) + b(y_p + \frac{1}{2}) + c \\
d_{\text{new}} &= d_{\text{old}} + a \\
&= d_{\text{old}} + \Delta_{\mathbf{E}}.
\end{aligned}
$$

Midpoint Line Algorithm

- If $\mathbf{NE}$ is chosen , $\mathbf{M}$ is incremented by one step in both the directions.



$$
\begin{aligned}
d_{\text{new}} &= F\left(x_p + 2, y_p + \frac{3}{2}\right) \\
&= a(x_p + 2) + b(y_p + \frac{3}{2}) + c \\
d_{\text{old}} &= a(x_p + 1) + b(y_p + \frac{1}{2}) + c \\
d_{\text{new}} &= d_{\text{old}} + a + b \\
&= d_{\text{old}} + \Delta_{\mathbf{NE}}.
\end{aligned}
$$

- Since the first pixel is simply the first endpoint (x_0, y_0), we can directly calculate the initial value of d for choosing between $\mathbf{E}$ and $\mathbf{NE}$. The first midpoint is at $(x_0 + 1, y_0 + \frac{1}{2})$, and

$$\begin{aligned} d_{\text{start}} &= F(x_0 + 1, y_0 + \frac{1}{2}) \\ &= a(x_0 + 1) + b(y_0 + \frac{1}{2}) + c \\ &= ax_0 + by_0 + c + a + \frac{b}{2} \\ &= a + \frac{b}{2}. \\ &= \Delta y - \frac{\Delta x}{2}. \end{aligned}$$

- To eliminate the fraction in d_{start}, we redefine our original F by multiplying it by 2; $F(x, y) = 2(ax + by + c)$.
- This multiplies each constant and the decision variable by 2, but does not affect the sign of the decision variable, which is all that matters for the midpoint test.

Input: (x_0, y_0) and (x_1, y_1)

Initialize: $\Delta x = x_1 - x_0$, $\Delta y = y_1 - y_0$, $d = 2\Delta y - \Delta x$, $\Delta_{\mathbf{E}} = 2\Delta y$, $\Delta_{\mathbf{NE}} = 2(\Delta y - \Delta x)$, $x = x_0$, and $y = y_0$.

Draw at (x, y)

while $x < x_1$ **do**

    if $d \leq 0$ **then**

        $d \leftarrow d + \Delta_{\mathbf{E}}$

        $x \leftarrow x + 1$

    else

        $d \leftarrow d + \Delta_{\mathbf{NE}}$

        $x \leftarrow x + 1$

        $y \leftarrow y + 1$

    end if

    Draw at (x, y)

end while