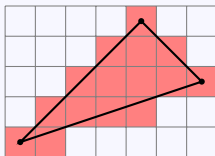


Line Rasterization

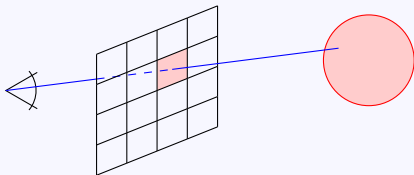
University of California Riverside

Raster Image

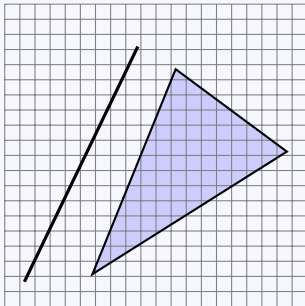
- Object oriented
 - for each object...



- Image oriented
 - for each pixel...



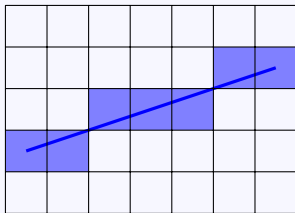
What is rasterization?



Rasterization is the process of determining which pixels are “covered” by the primitive

Rasterization

- In: 2D primitives (floating point)
- Out: covered pixels (integer)
- Must be fast (called **many times**)
- Visually pleasing
 - lines have constant width
 - lines have no gaps



DDA algorithm for lines

- DDA = “digital differential analyzer”

DDA algorithm for lines

- DDA = “digital differential analyzer”
- Plot line $y = mx + b$

DDA algorithm for lines

- DDA = “digital differential analyzer”
- Plot line $y = mx + b$
- For each x :

DDA algorithm for lines

- DDA = “digital differential analyzer”
- Plot line $y = mx + b$
- For each x :
 - $y = mx + b$

DDA algorithm for lines

- DDA = “digital differential analyzer”
- Plot line $y = mx + b$
- For each x :
 - $y = mx + b$
 - turn on pixel $(x, \text{round}(y))$

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

$$\begin{aligned}y_{i+1} &= mx_{i+1} + b \\&= m(x_i + 1) + b \\&= y_i + m\end{aligned}$$

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

$$\begin{aligned}y_{i+1} &= mx_{i+1} + b \\&= m(x_i + 1) + b \\&= y_i + m\end{aligned}$$

- Each time:

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

$$\begin{aligned}y_{i+1} &= mx_{i+1} + b \\&= m(x_i + 1) + b \\&= y_i + m\end{aligned}$$

- Each time:
 - Increment x

DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

$$\begin{aligned}y_{i+1} &= mx_{i+1} + b \\&= m(x_i + 1) + b \\&= y_i + m\end{aligned}$$

- Each time:
 - Increment x
 - Add m to y

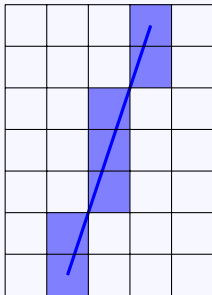
DDA algorithm for lines

- Assume $|m| \leq 1$
- March from left to right
 - $x_0 = \text{start}$, $x_{i+1} = x_i + 1$, $x_n = \text{end}$

$$\begin{aligned}y_{i+1} &= mx_{i+1} + b \\&= m(x_i + 1) + b \\&= y_i + m\end{aligned}$$

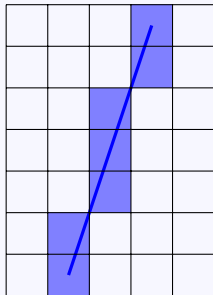
- Each time:
 - Increment x
 - Add m to y
 - turn on pixel $(x_i, \text{round}(y_i))$

DDA algorithm for lines



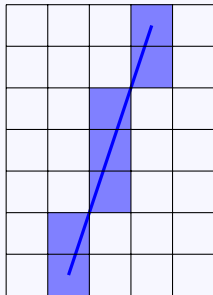
- What if $|m| > 1$?

DDA algorithm for lines



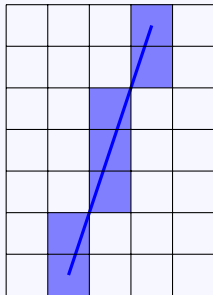
- What if $|m| > 1$?
- Increment y by m

DDA algorithm for lines



- What if $|m| > 1$?
- Increment y by m
- $\text{round}(y)$ may skip an integer
 - gap in the line

DDA algorithm for lines



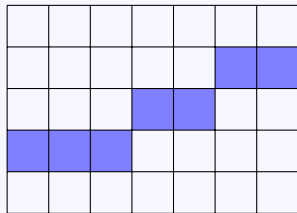
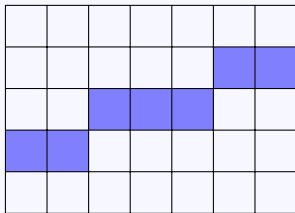
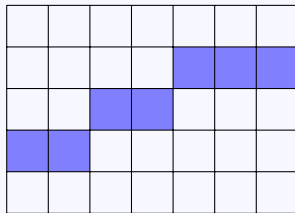
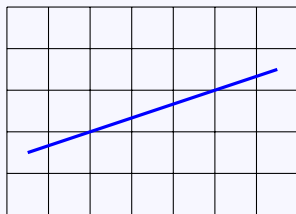
- What if $|m| > 1$?
- Increment y by m
- $\text{round}(y)$ may skip an integer
 - gap in the line
- Swap the roles of x and y
 - Loop over y , compute and round x

DDA algorithm for lines - limitations

- Must round for each pixel
 - very slow
- Only use ops: $+$, $-$, $\times$
 - Even better: $+$, $-$

Rasterization choices

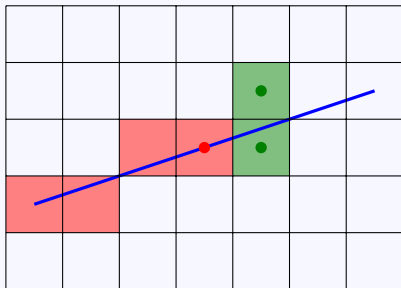
- Thin, no gaps
- Still have choices



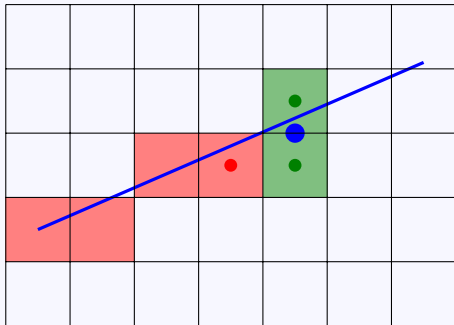
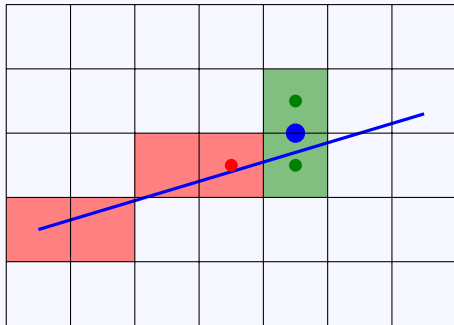
Midpoint algorithm

- Assume $0 \leq m \leq 1$
- Move from left to right
- Choose between $(x + 1, y)$ and $(x + 1, y + 1)$

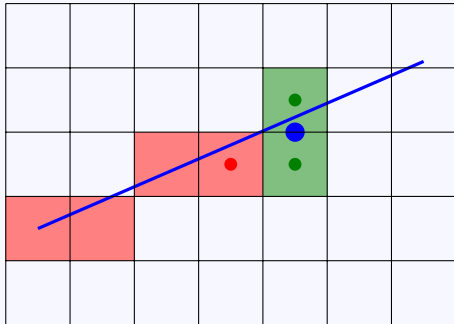
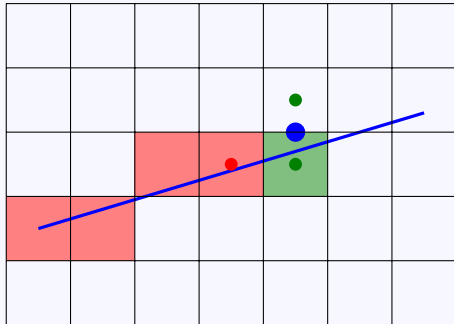
```
 $y = y_0$   
for  $x = x_0, \dots, x_1$  do  
  draw( $x, y$ )  
  if  $\langle \text{condition} \rangle$  then  
     $y \leftarrow y + 1$ 
```



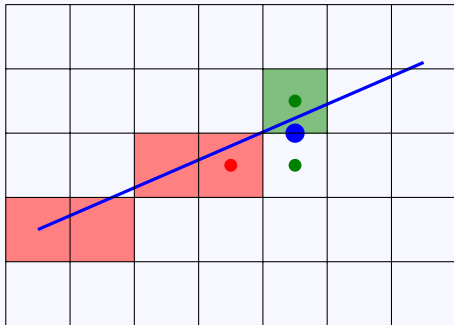
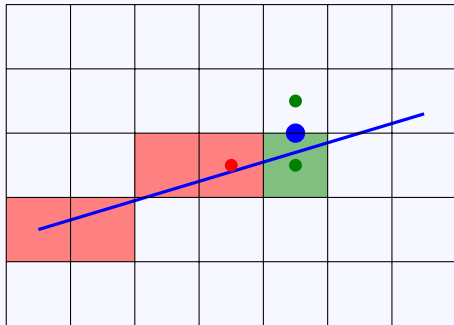
Check midpoint location



Check midpoint location



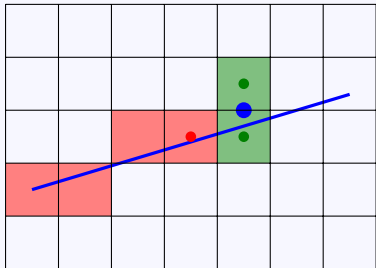
Check midpoint location



Criterion

Implicit line equation:

$$f(\mathbf{x}) = \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) = 0$$



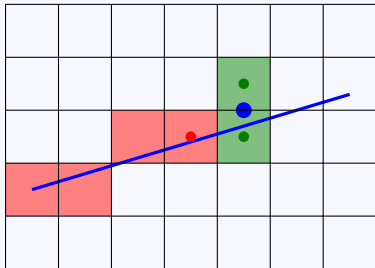
Criterion

Implicit line equation:

$$f(\mathbf{x}) = \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) = 0$$

Evaluate f at midpoint:

$$f\left(x + 1, y + \frac{1}{2}\right) \stackrel{?}{<} 0$$



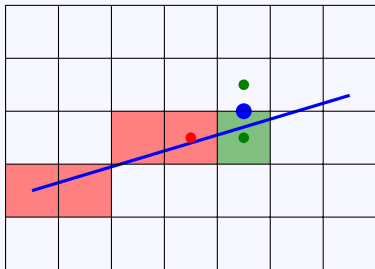
Criterion

Implicit line equation:

$$f(\mathbf{x}) = \mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) = 0$$

Evaluate f at midpoint:

$$f\left(x + 1, y + \frac{1}{2}\right) < 0$$



Midpoint algorithm ($0 \leq m \leq 1$)

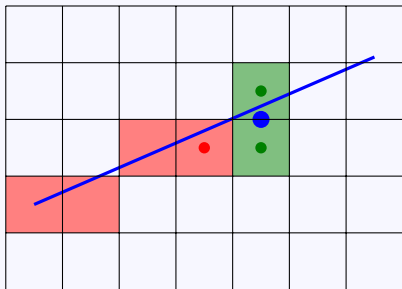
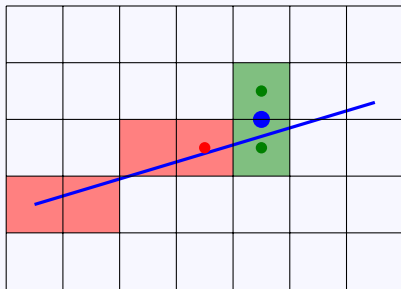
$y \leftarrow y_0$

for $x = x_0, \dots, x_1$ **do**

draw(x, y)

if $f(x + 1, y + \frac{1}{2}) < 0$ **then**

$y \leftarrow y + 1$



Efficiency: incremental update

- Compute initial $f(x, y)$
- Compute next by updating previous
- Update with *one* addition

$$f(x, y) = (y_0 - y_1)x + (x_1 - x_0)y + (x_0y_1 - x_1y_0)$$

Efficiency: incremental update

- Compute initial $f(x, y)$
- Compute next by updating previous
- Update with *one* addition

$$f(x, y) = (y_0 - y_1)x + (x_1 - x_0)y + (x_0y_1 - x_1y_0)$$

$$f(x + 1, y) = f(x, y) + (y_0 - y_1)$$

Efficiency: incremental update

- Compute initial $f(x, y)$
- Compute next by updating previous
- Update with *one* addition

$$f(x, y) = (y_0 - y_1)x + (x_1 - x_0)y + (x_0y_1 - x_1y_0)$$

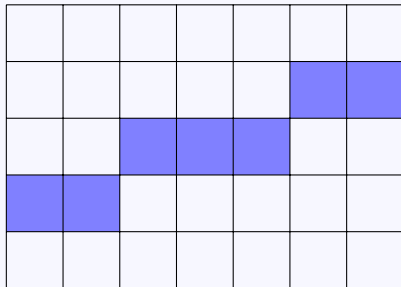
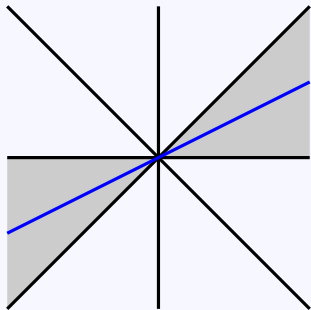
$$f(x + 1, y) = f(x, y) + (y_0 - y_1)$$

$$f(x + 1, y + 1) = f(x, y) + (y_0 - y_1) + (x_1 - x_0)$$

Efficiency: incremental update

```
 $y \leftarrow y_0$   
 $d \leftarrow f(x_0 + 1, y_0 + \frac{1}{2})$   
for  $x = x_0, \dots, x_1$  do  
  draw( $x, y$ )  
  if  $d < 0$  then  
     $y \leftarrow y + 1$   
     $d \leftarrow d + (y_0 - y_1) + (x_1 - x_0)$   
  else  
     $d \leftarrow d + (y_0 - y_1)$ 
```

Other cases: $0 \leq m \leq 1$



Other cases: $-1 \leq m \leq 0$

