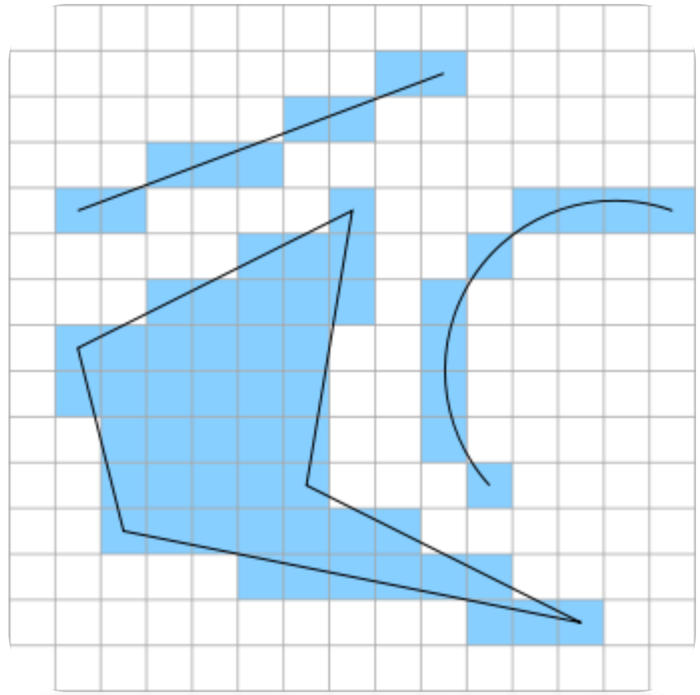


Rasterization

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

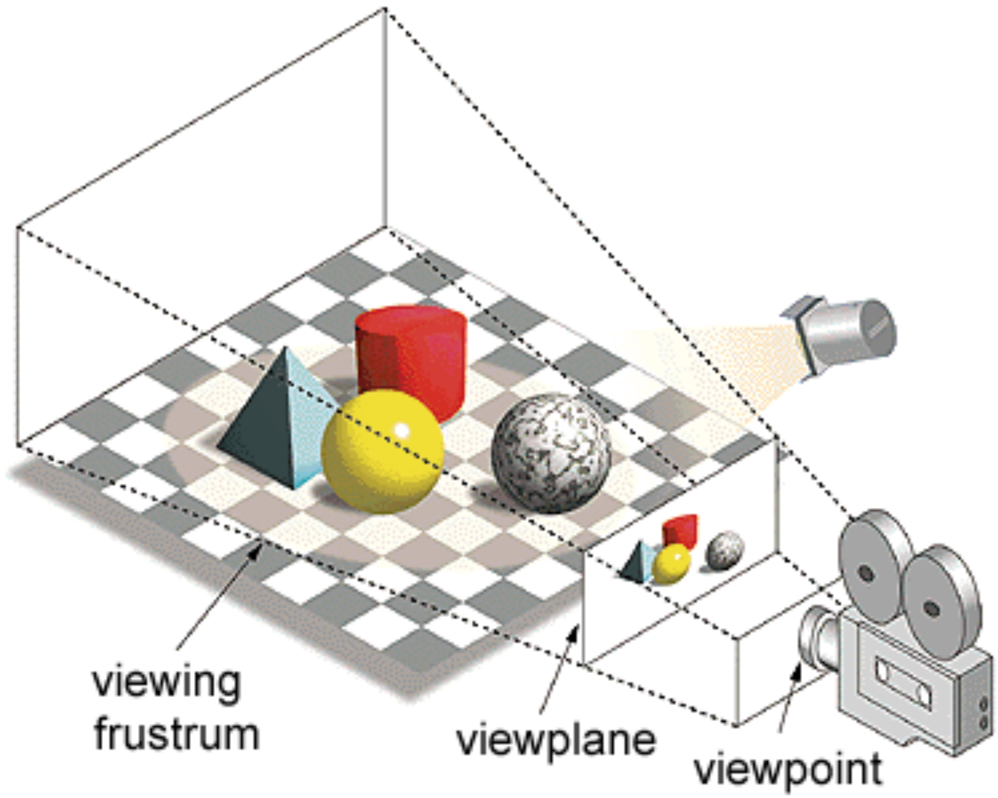
http://iloveshaders.blogspot.com/2011/05/how-rasterization-process-works.html

Render [ren-der]:

The process of generating an image from a description of a scene by means of a computer program

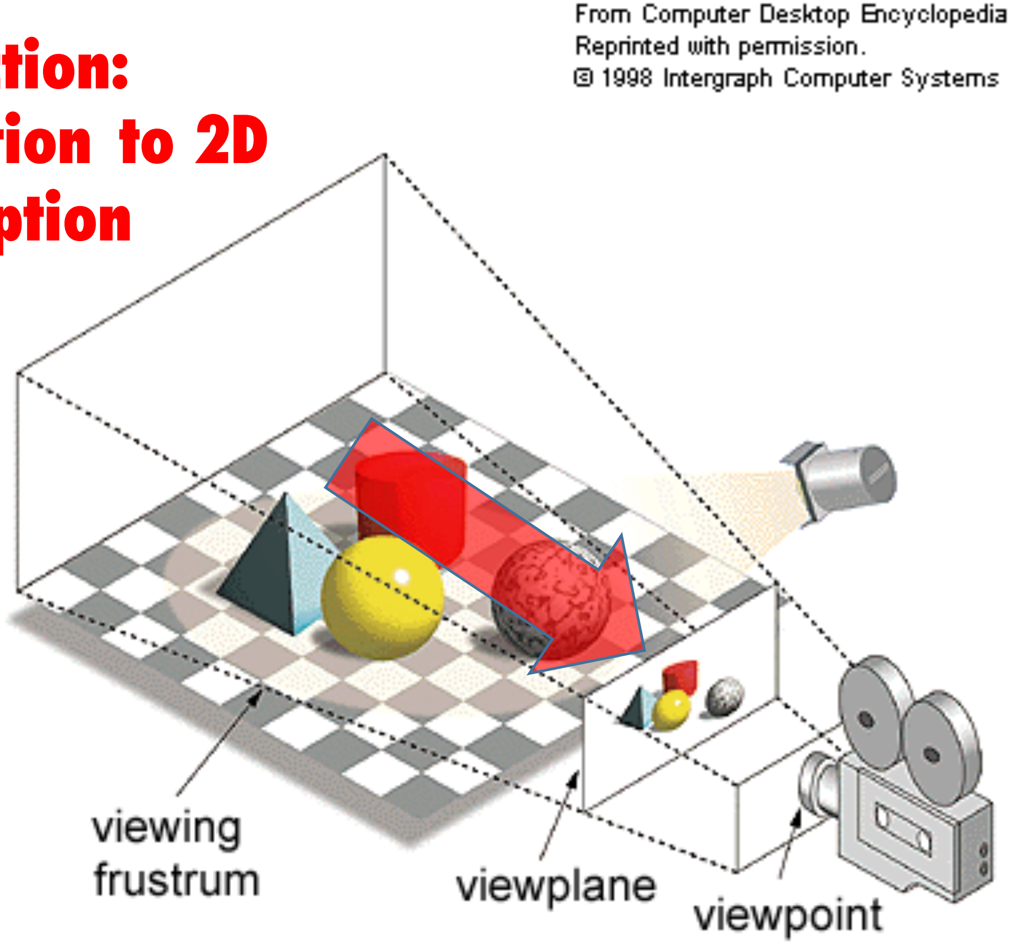
https://en.wikipedia.org/wiki/Rendering_(computer_graphics)

Two Ways to Render an Image

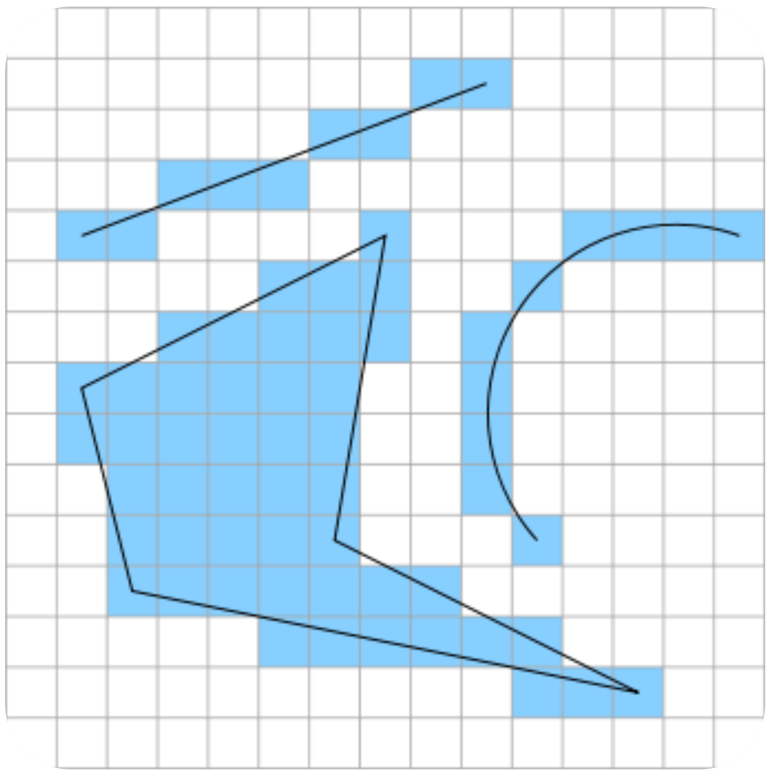


Two Ways to Render an Image

Projection:
3D description to 2D description



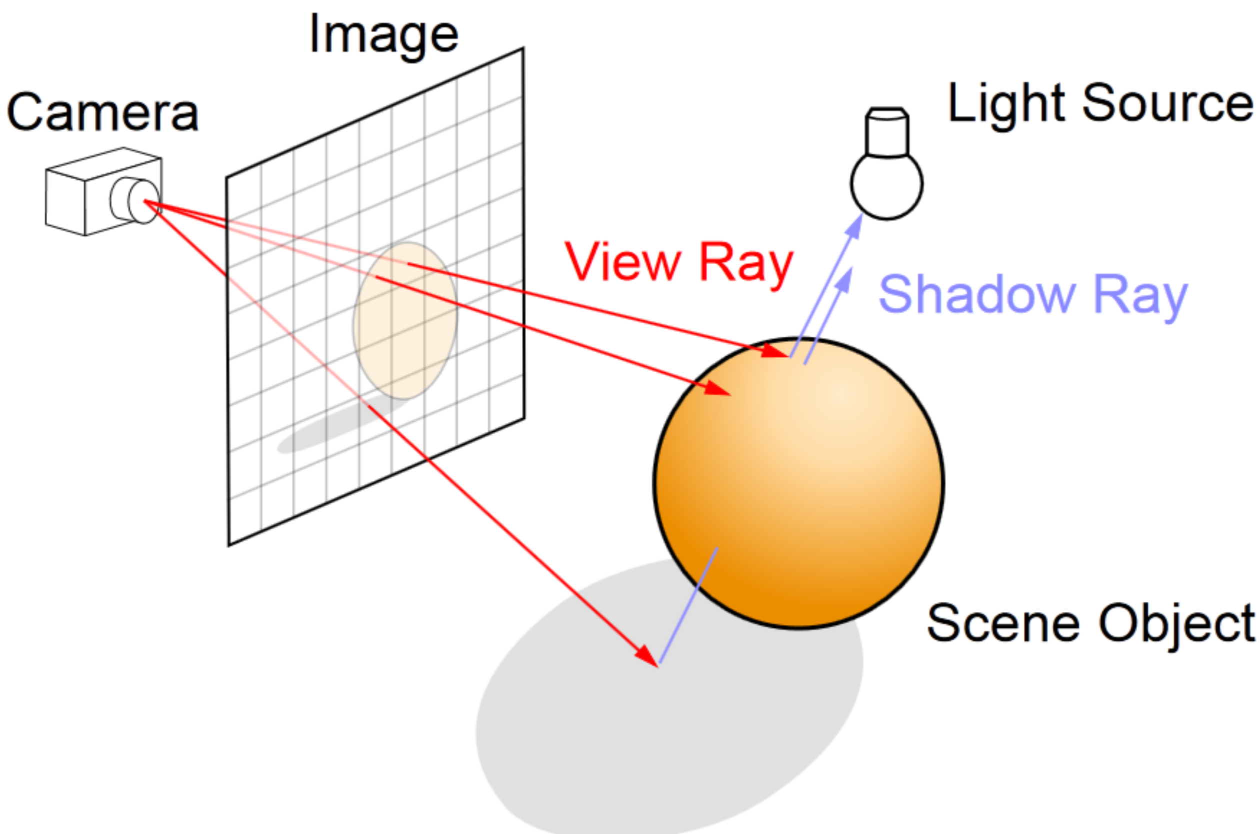
Two Ways to Render an Image



Rasterization

http://iloveshaders.blogspot.com/2011/05/how-rasterization-process-works.html

Two Ways to Render an Image



Raytracing

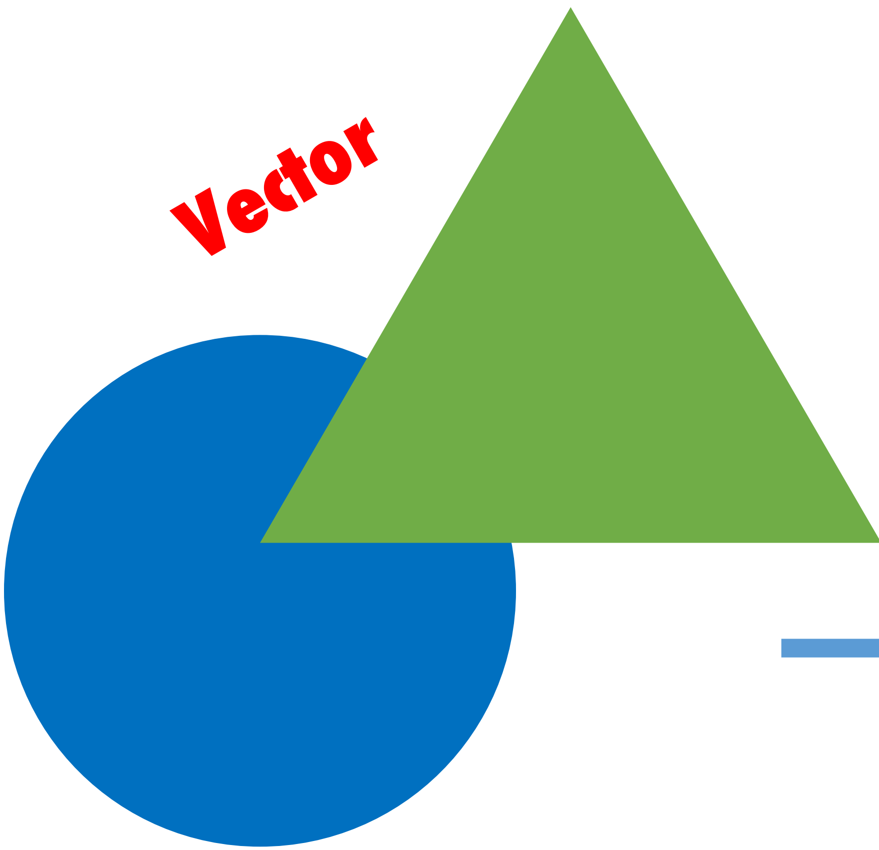
http://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg

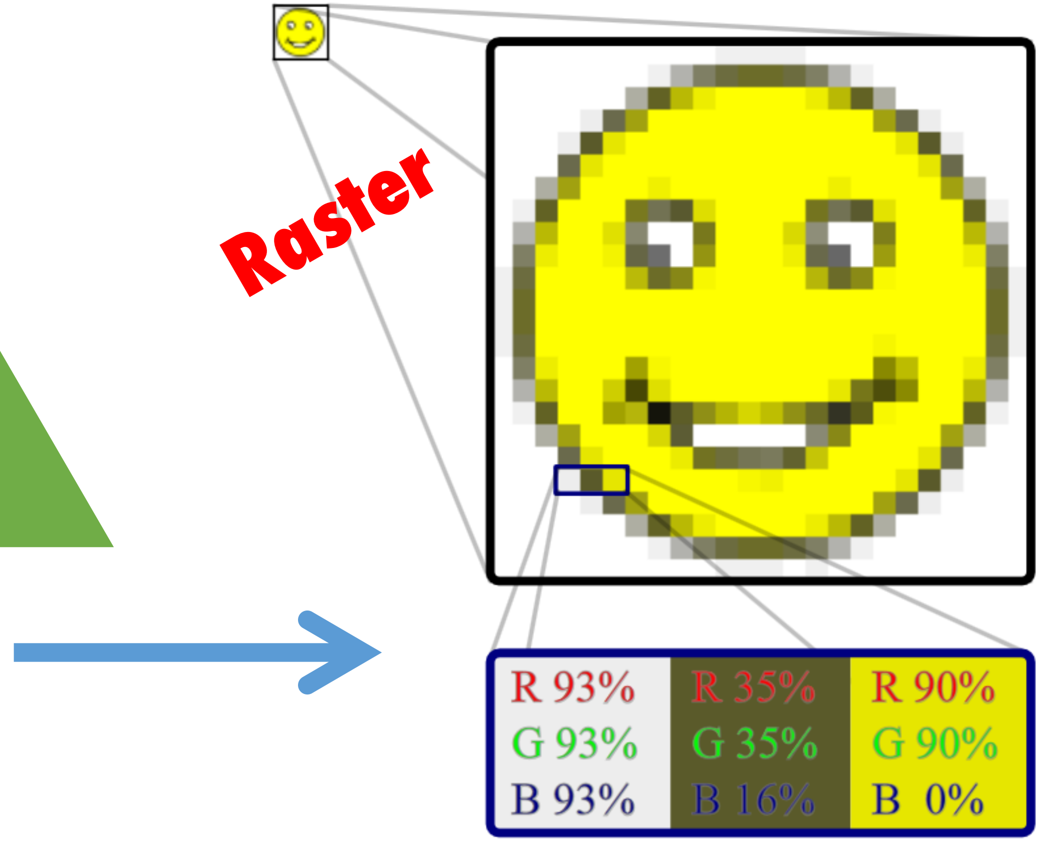
Rasterization

Rasterize [rastərʌɪz]:

To convert vector data to raster – pixel or dot – format.

Rasterization Process





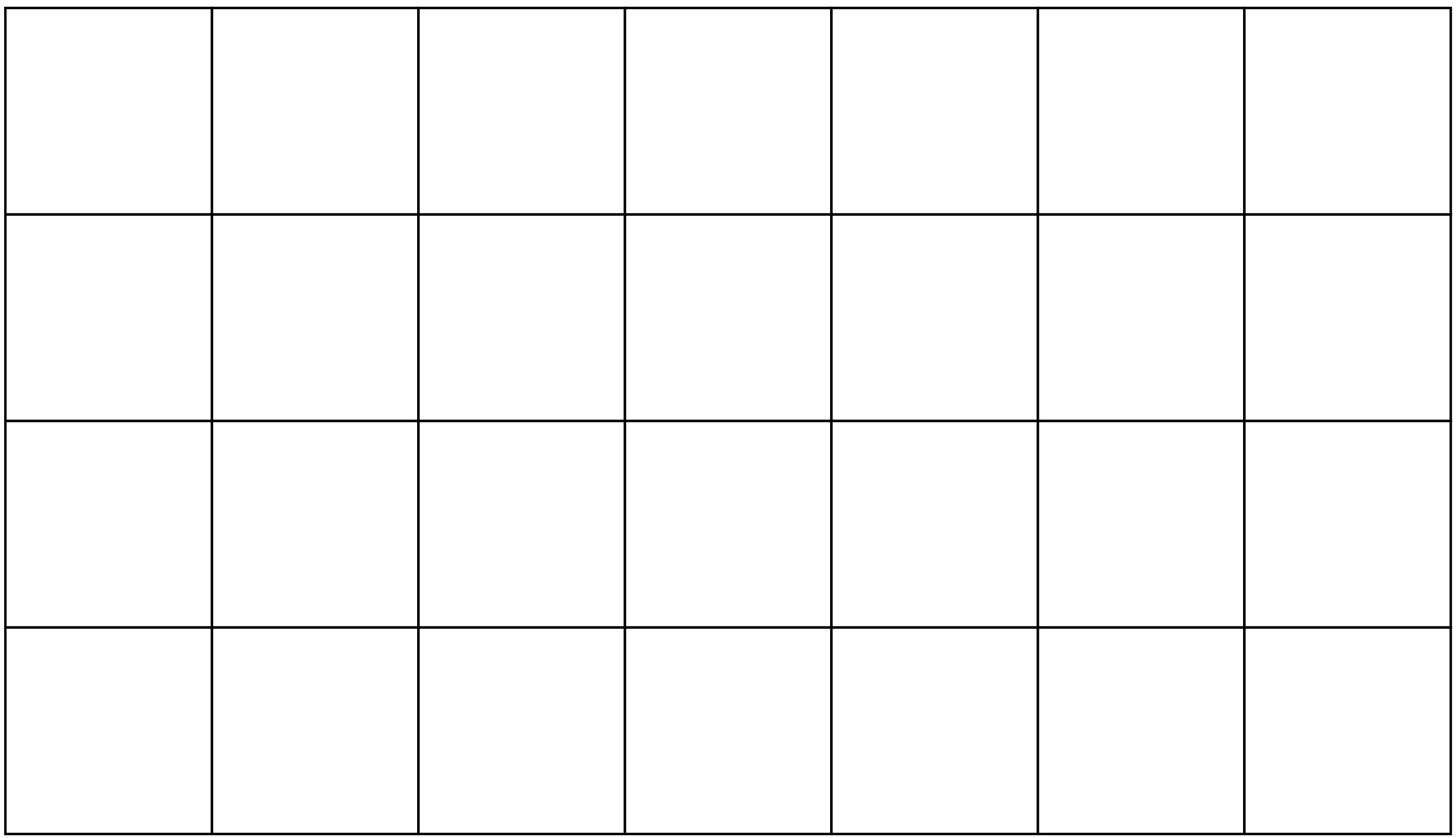
"A triangle is here, a circle is there, …"

"This pixel is yellow…"

Scan Conversion

Figuring out which pixels to shade.

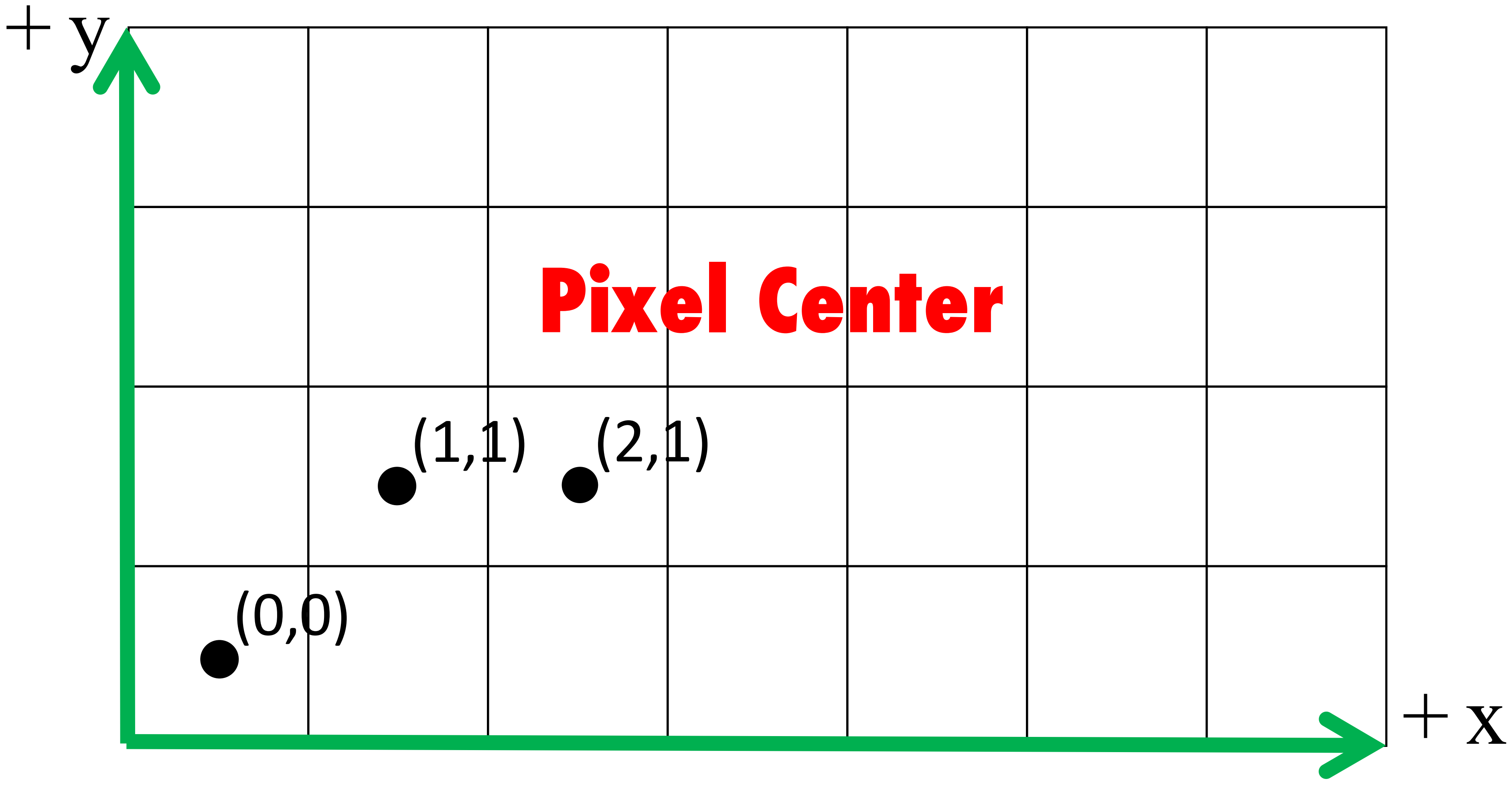
The Basics: Framebuffer



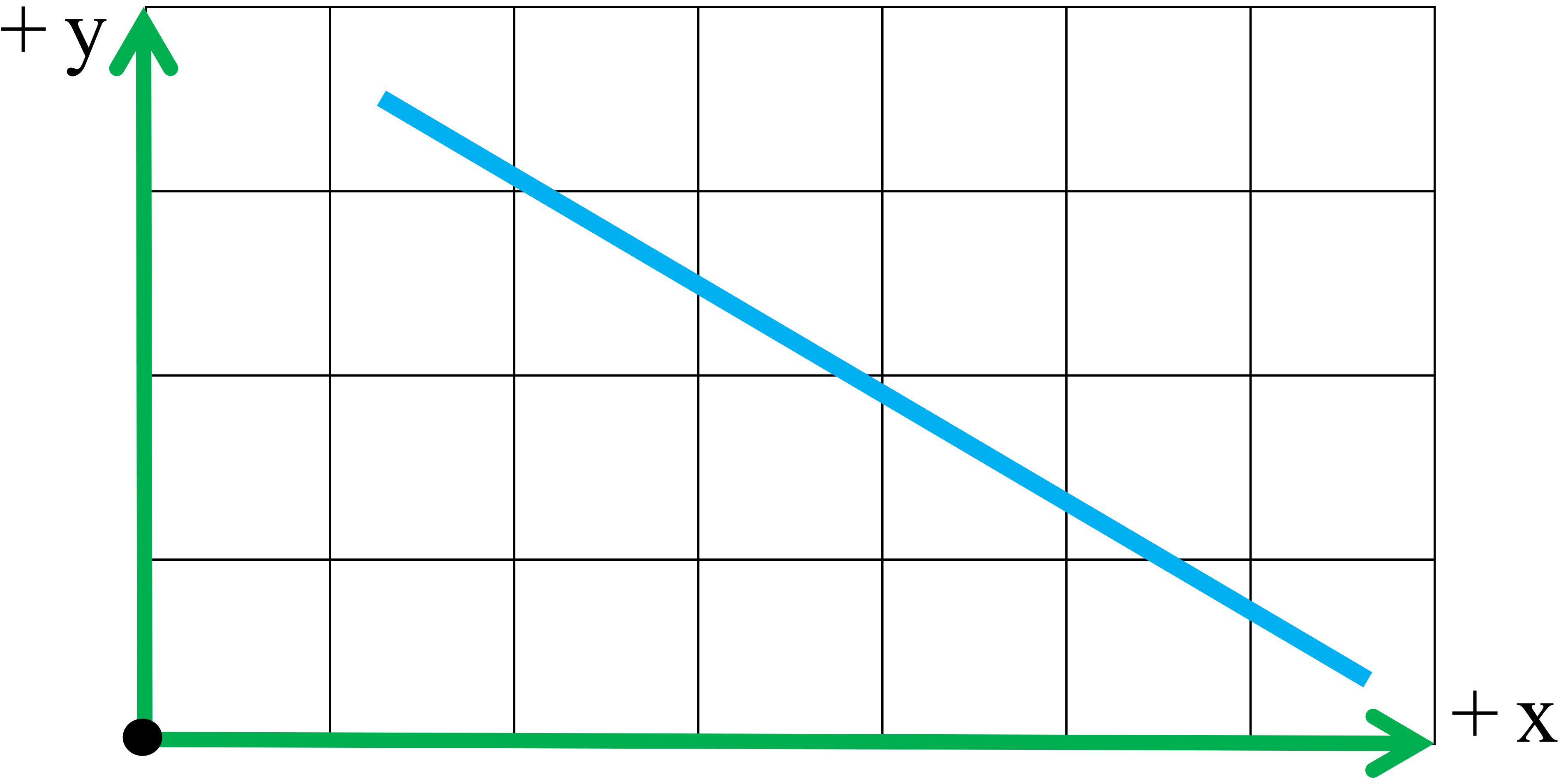
The Basics: Framebuffer

Pixel	Pixel	Pixel	Pixel	Pixel	Pixel	Pixel
Pixel	Pixel	Pixel	Pixel	Pixel	Pixel	Pixel
Pixel	Pixel	Pixel	Pixel	Pixel	Pixel	Pixel
Pixel	Pixel	Pixel	Pixel	Pixel	Pixel	Pixel

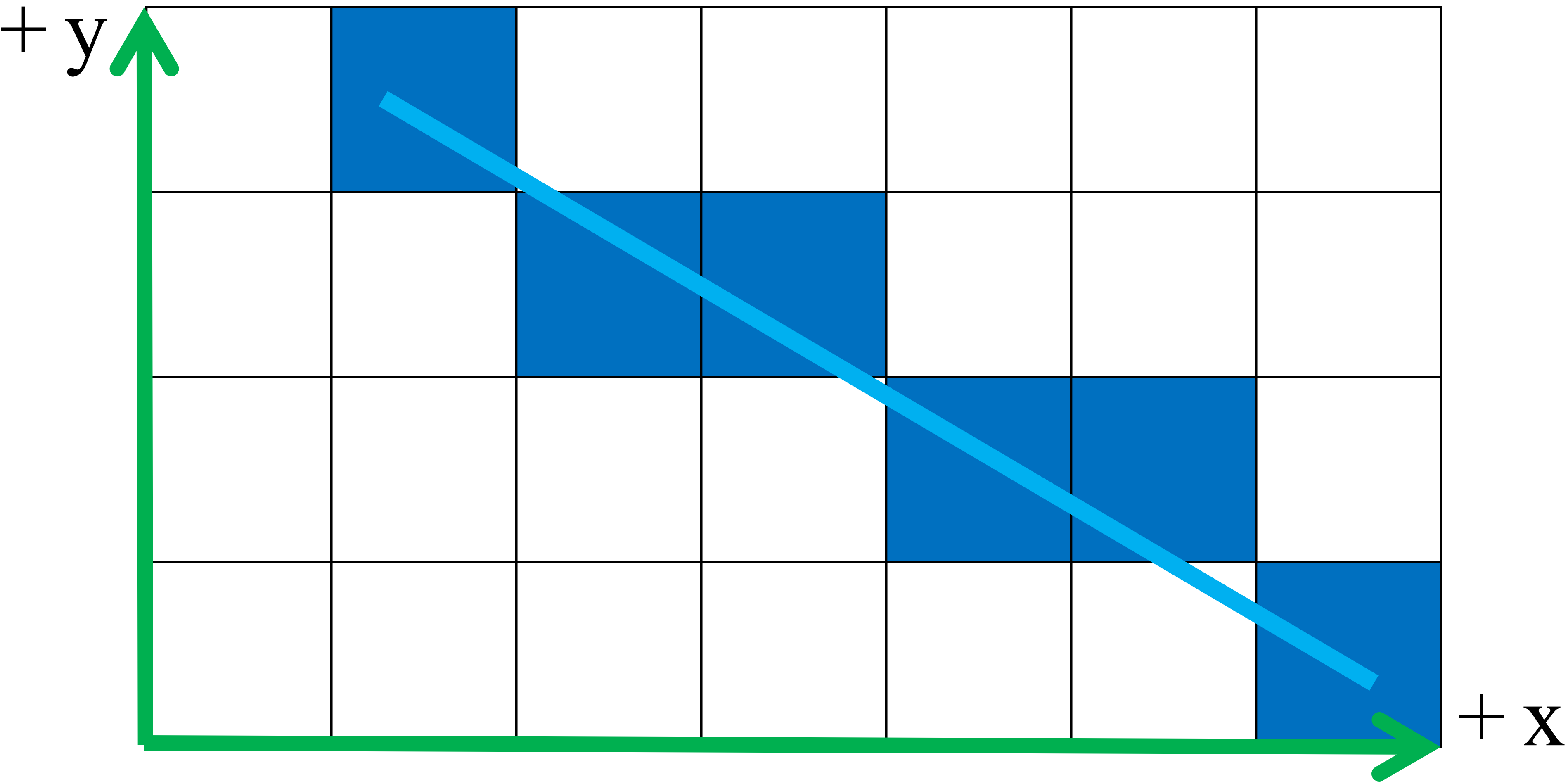
Framebuffer Coordinates



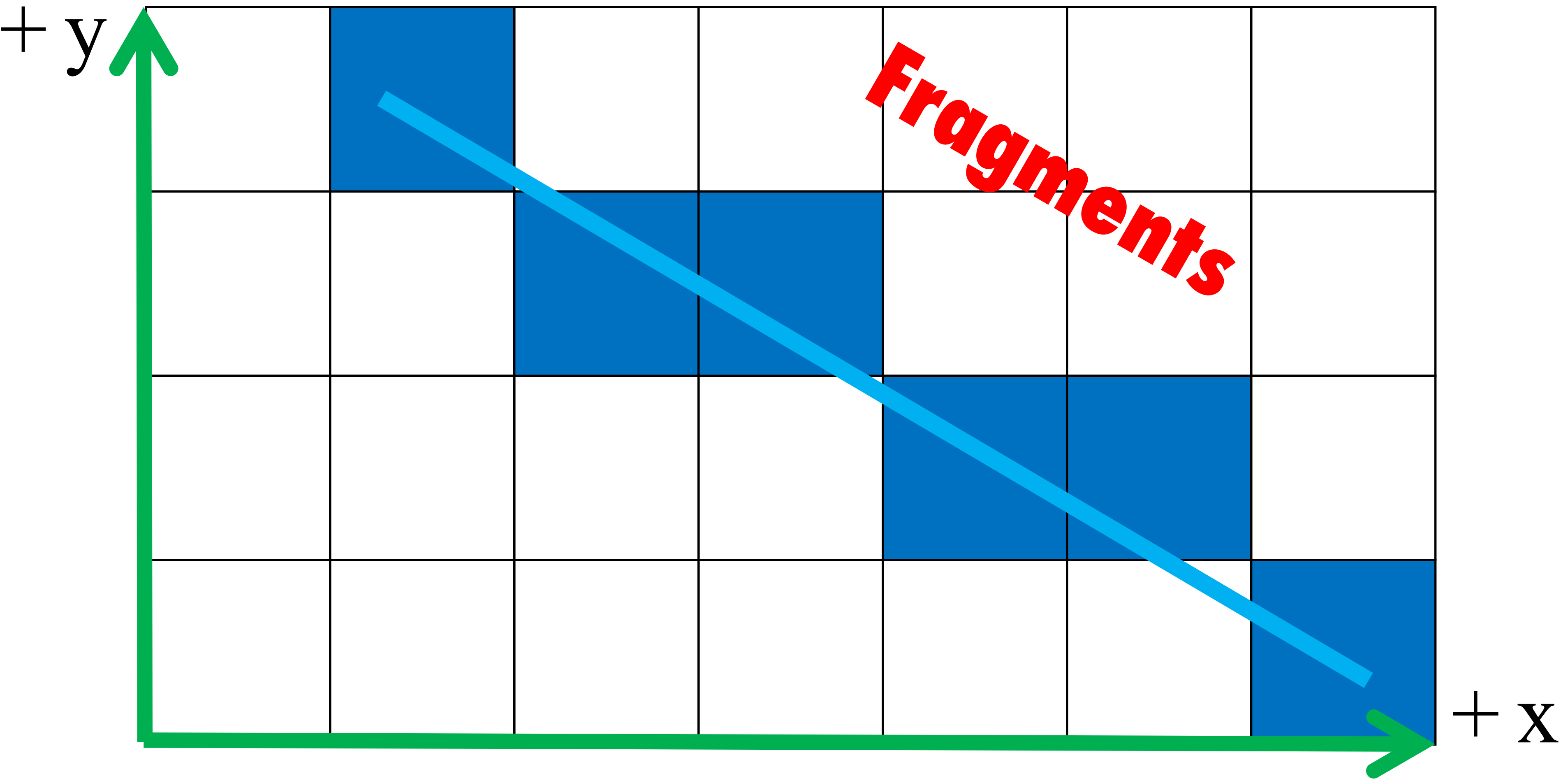
Problem



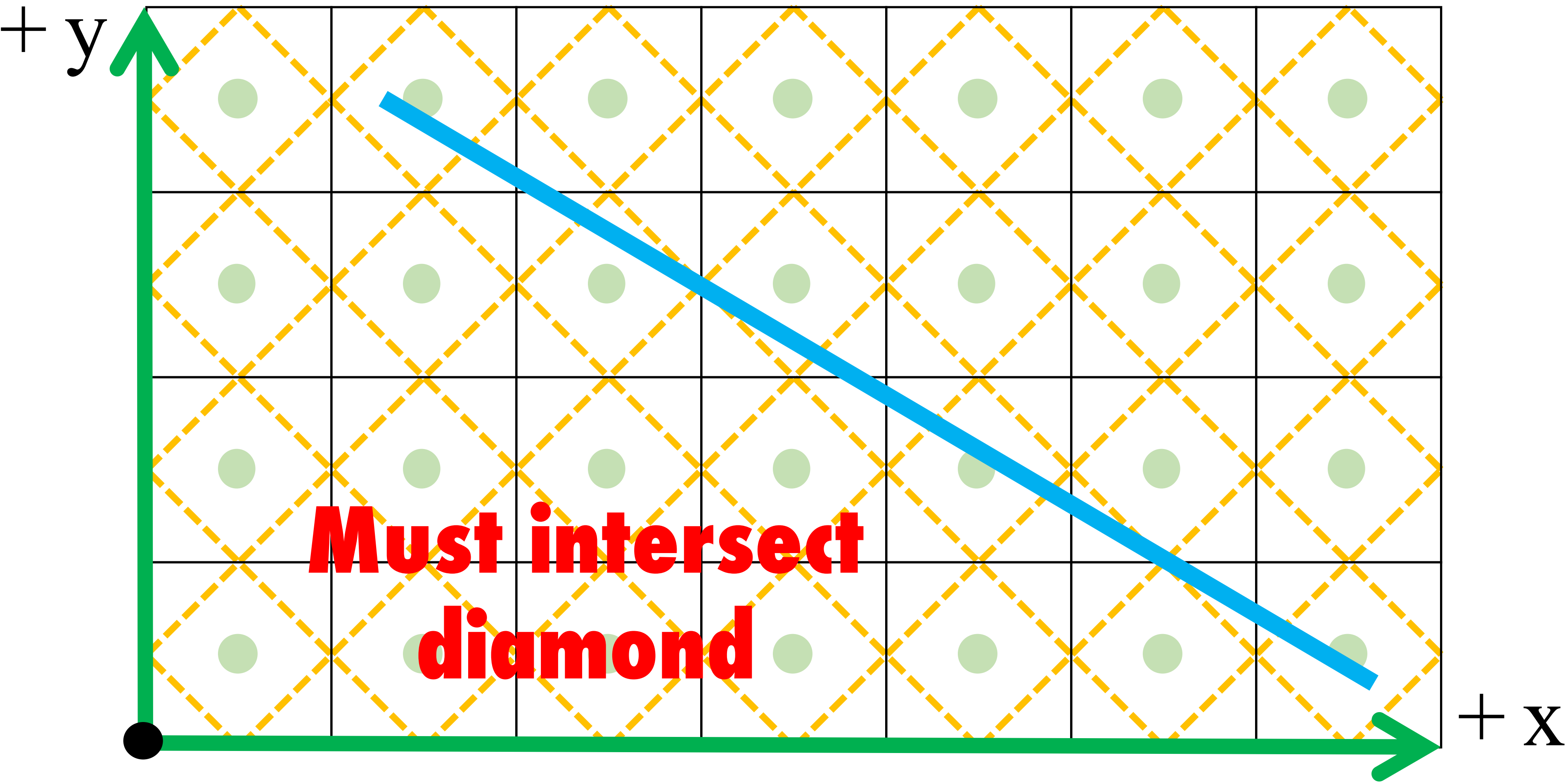
The Basics: Scan Conversion



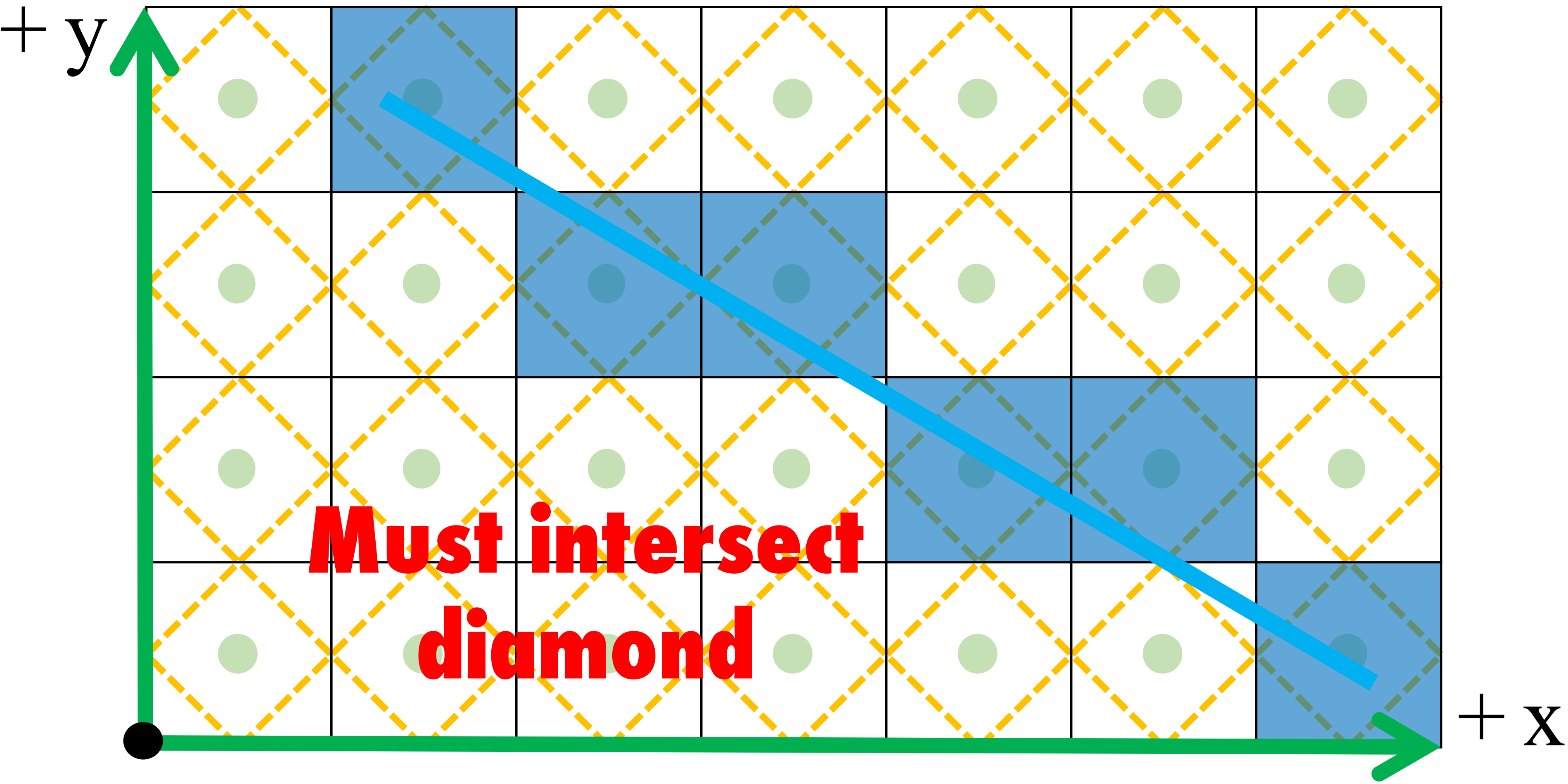
The Basics: Scan Conversion



Convention



Convention



Desirable Properties

- Fast
- Simple
- Integer arithmetic

Bresenham’s Algorithm

- Introduced in 1967

- Best-fit approximation under some conditions

Bresenham's Algorithm

- Introduced in 1967

Variation by Pitteway (1967)
"Midpoint Algorithm"

- Best-fit approximation under some conditions

Bresenham's Algorithm

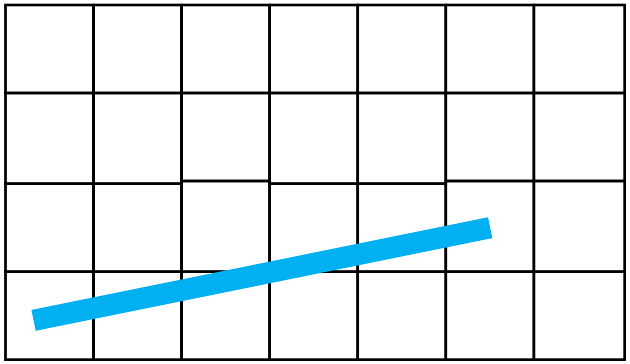
- Equation of a line

$$y = mx + b$$

Bresenham's Algorithm

- Equation of a line

$$y = mx + b$$

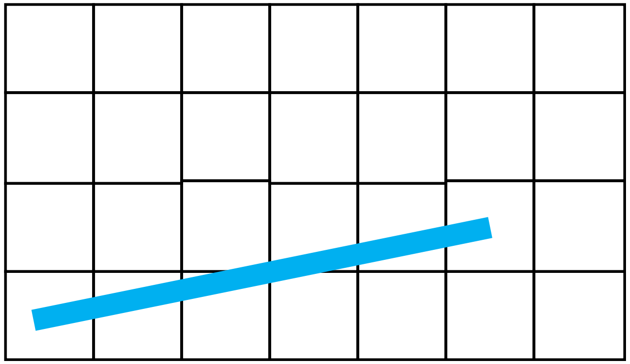


Assumption: $0 \leq m < 1$

Bresenham's Algorithm

- Equation of a line

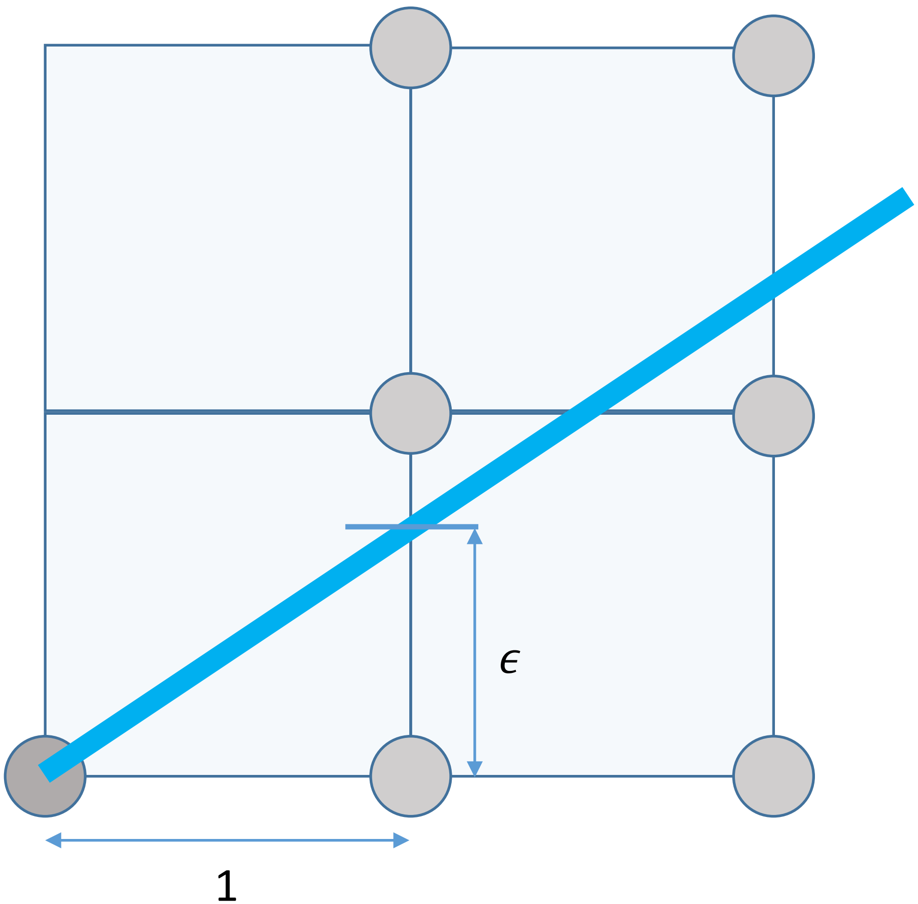
$$y = mx + b$$



Assumption: $0 \leq m < 1$

Easy fix via rotation/reflection

Bresenham's Algorithm

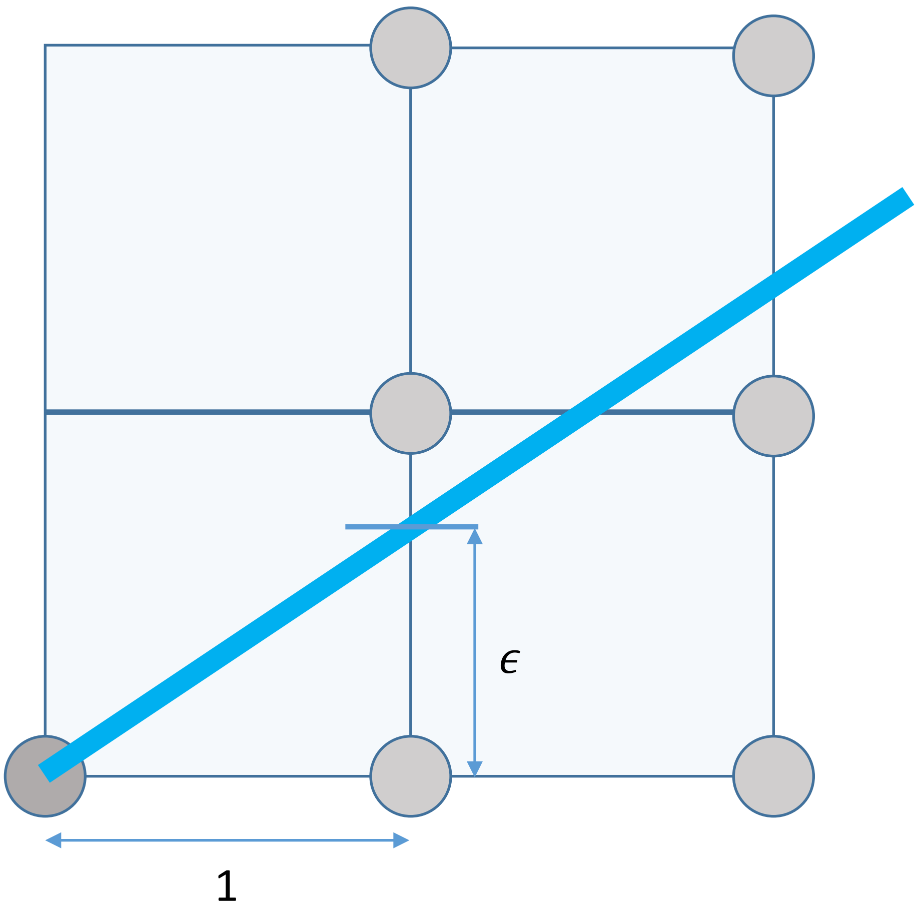


$$y = mx + b$$

Stategy

- Every step: increment x
- Keep track of "error" ϵ
- At every step accumulate
  - $\epsilon \leftarrow \epsilon + m$
- If $\epsilon \geq \frac{1}{2}$
  - Increment y
  - Reset $\epsilon \leftarrow \epsilon + m - 1$
- Start with $\epsilon = m$

Bresenham's Algorithm

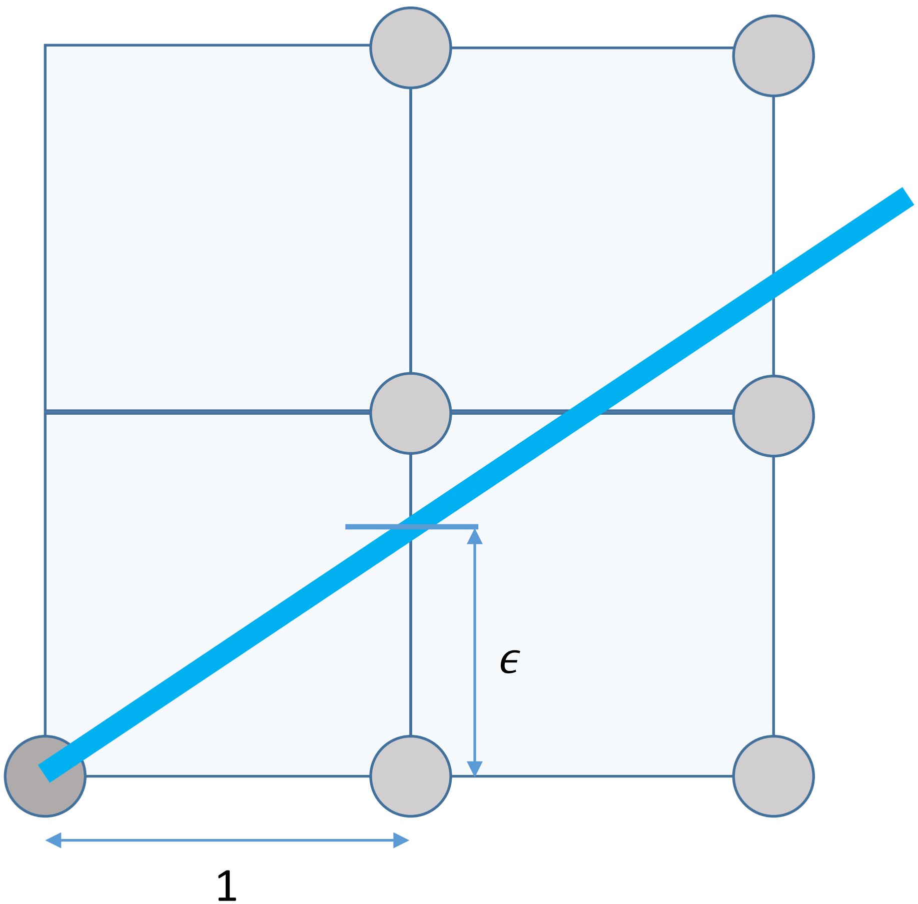


$$y = mx + b$$

Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    ϵ = m
    while x < x1
      pixel(x, y)
      if ϵ > 1/2
          y ← y + 1
          ϵ ← ϵ + m − 1
      else
          ϵ ← ϵ + m
      x ← x + 1
```

Bresenham's Algorithm

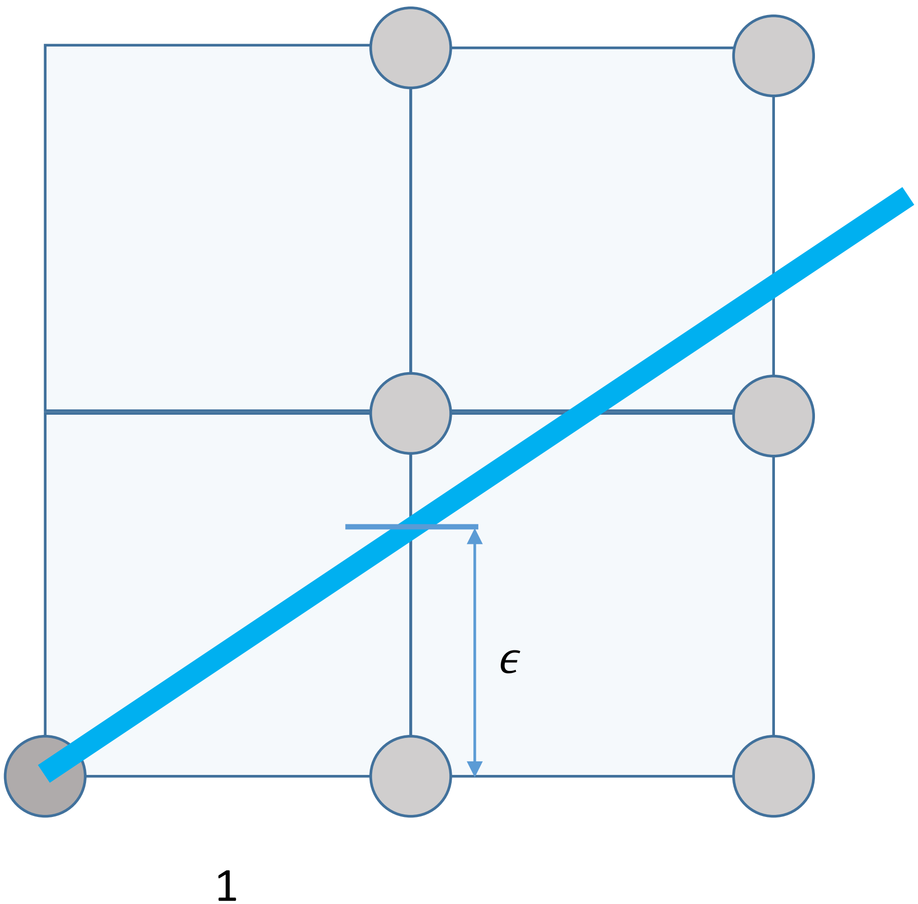


$$y = mx + b$$

Algorithm

line$(x0, y0, x1, y1)$
  $x \leftarrow x0$
  $y \leftarrow y0$
  $\epsilon = m$
  while $x < x1$
    pixel(x, y)
    if $\epsilon > \frac{1}{2}$
      $y \leftarrow y + 1$
      $\epsilon \leftarrow \epsilon + \boxed{m} - 1$
    else
      $\epsilon \leftarrow \epsilon + \boxed{m}$
    $x \leftarrow x + 1$

Bresenham's Algorithm

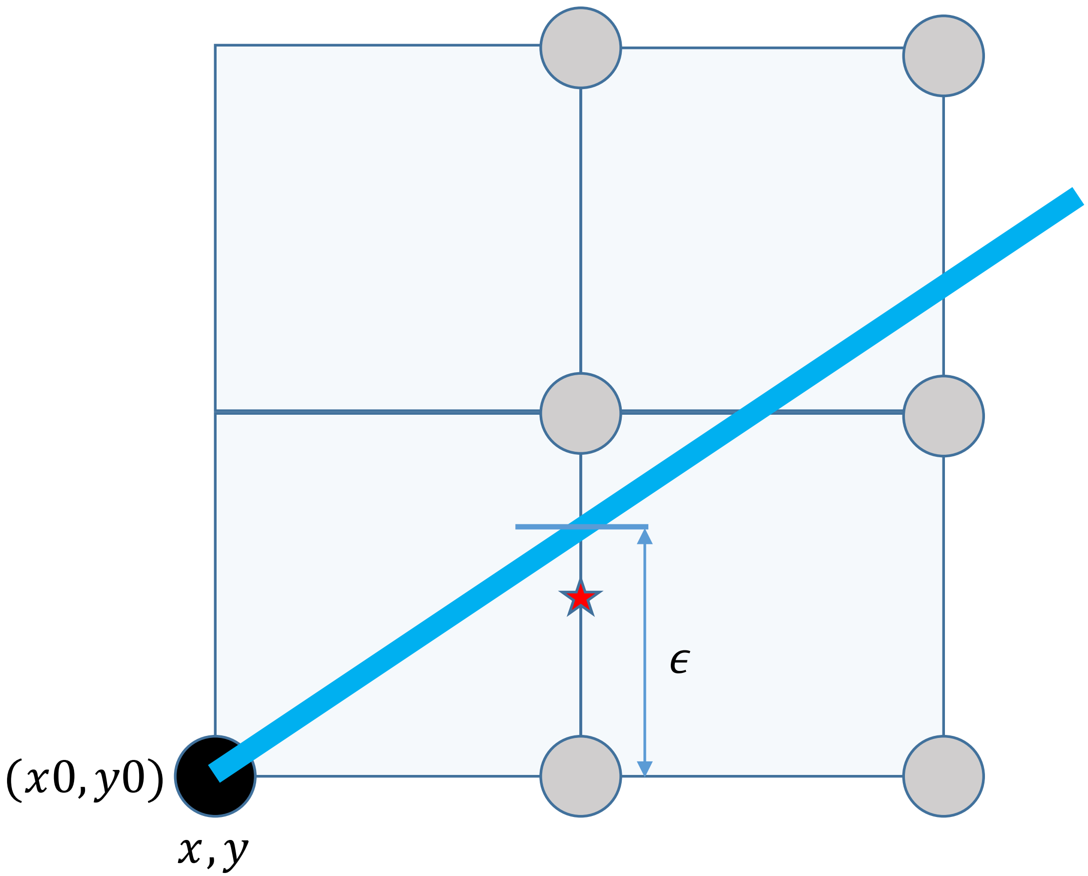


$$y = mx + b$$

Algorithm

line($x0, y0, x1, y1$)
  $x \leftarrow x0$
  $y \leftarrow y0$
  $\epsilon = m$
  while $x < x1$
    pixel(x, y)
    if $\epsilon > \frac{1}{2}$
      $y \leftarrow y + 1$
      $\epsilon \leftarrow \epsilon - 1$
    $\epsilon \leftarrow \epsilon + m$
    $x \leftarrow x + 1$

Bresenham's Algorithm

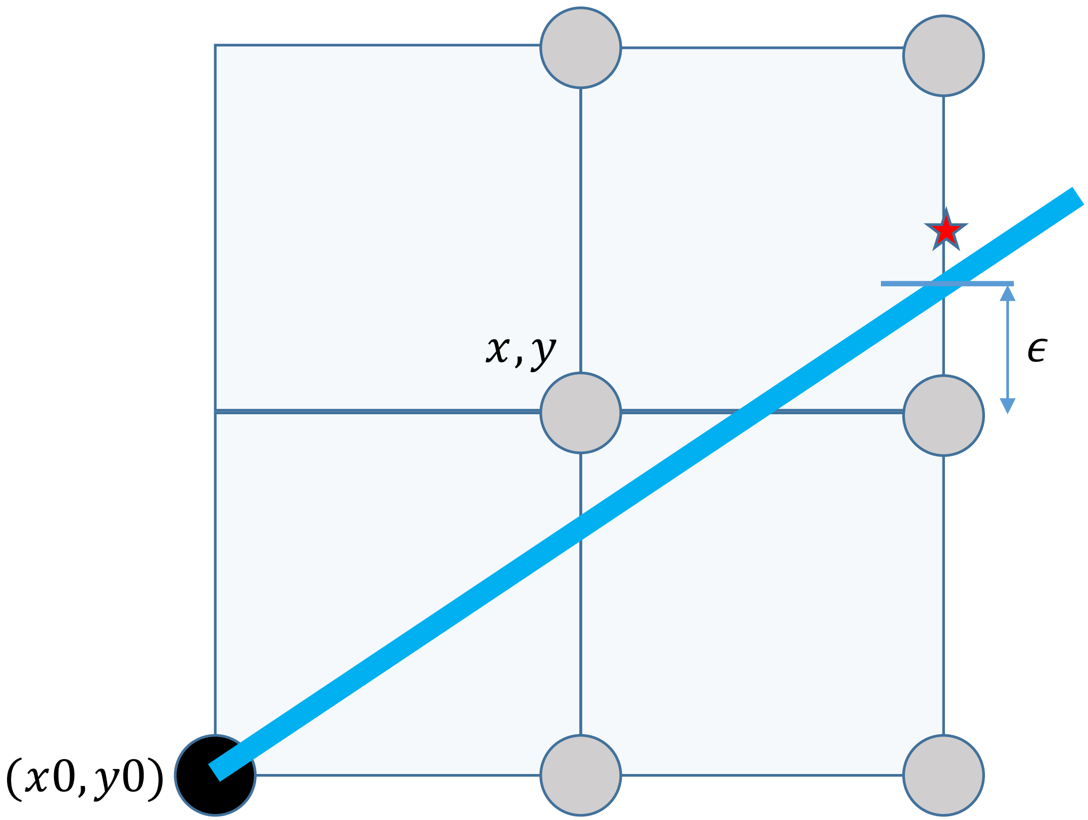


$$y = mx + b$$

Algorithm

```
line(x0,y0,x1,y1)
    x ← x0
    y ← y0
    ϵ = m
    while x < x1
      pixel(x,y)
      if ϵ > 1/2
          y ← y + 1
          ϵ ← ϵ − 1
      ϵ ← ϵ + m
      x ← x + 1
```

Bresenham's Algorithm

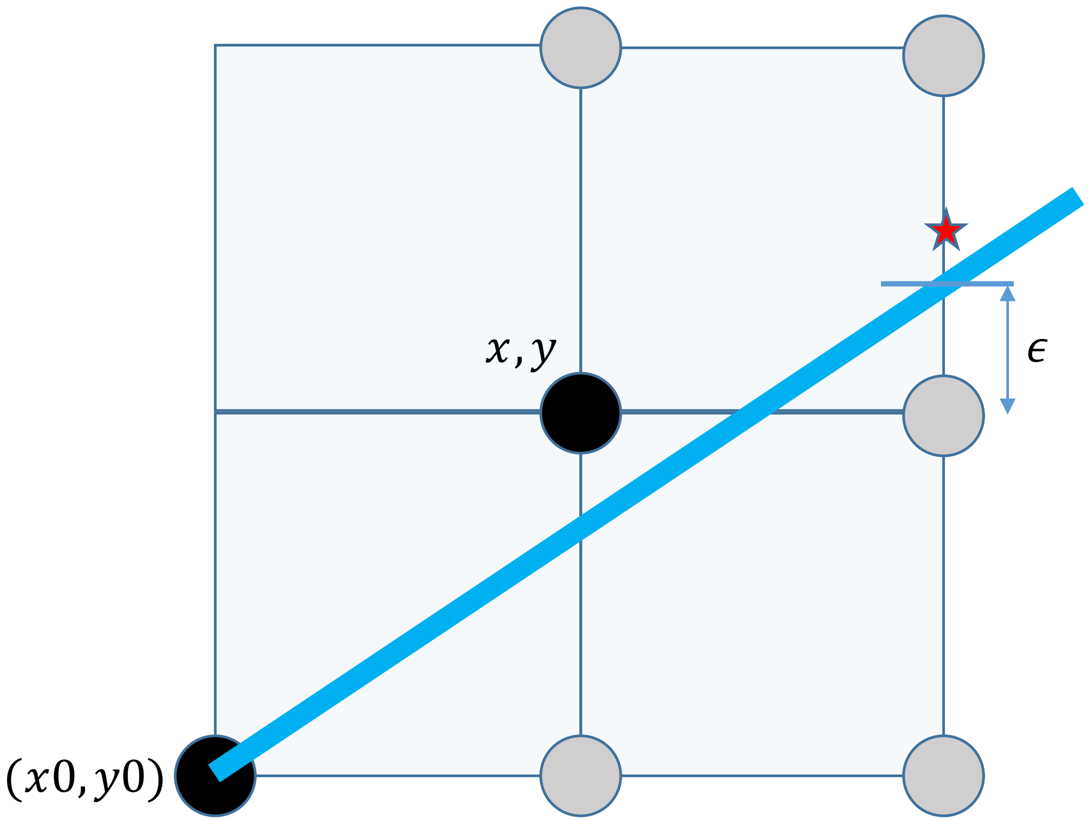


$$y = mx + b$$

Algorithm

```
line(x0,y0,x1,y1)
    x ← x0
    y ← y0
    ϵ = m
    while x < x1
      pixel(x,y)
      if ϵ > 1/2
          y ← y + 1
          ϵ ← ϵ − 1
      ϵ ← ϵ + m
      x ← x + 1
```

Bresenham's Algorithm

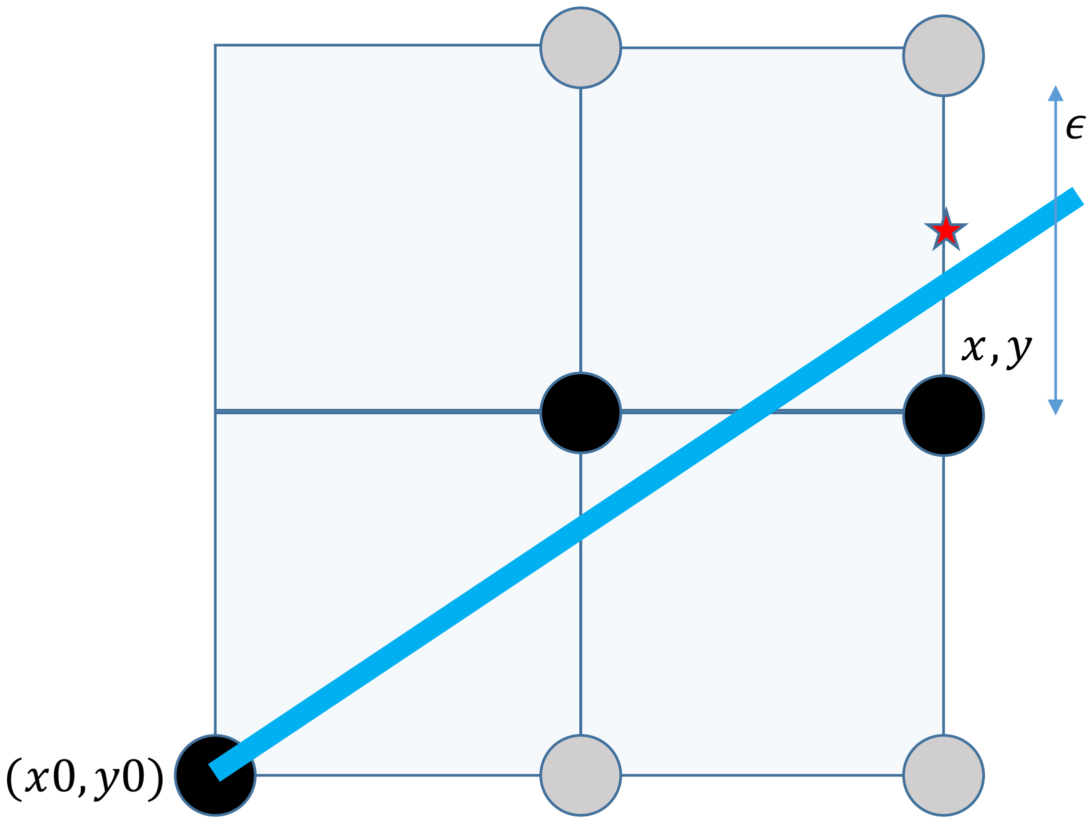


$$y = mx + b$$

Algorithm

```
line(x0,y0,x1,y1)
    x ← x0
    y ← y0
    ε = m
    while x < x1
      pixel(x,y)
      if ε > 1/2
          y ← y + 1
          ε ← ε − 1
      ε ← ε + m
      x ← x + 1
```

Bresenham's Algorithm



$$y = mx + b$$

Algorithm

```
line(x0,y0,x1,y1)
    x ← x0
    y ← y0
    ε = m
    while x < x1
      pixel(x,y)
      if ε > 1/2
          y ← y + 1
          ε ← ε − 1
      ε ← ε + m
      x ← x + 1
```

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    ϵ = m
    while x < x1
      pixel(x, y)
      if ϵ > 1/2
          y ← y + 1
          ϵ ← ϵ − 1
      ϵ ← ϵ + m
      x ← x + 1
```

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    ϵ = m
    while x < x1
        pixel(x, y)
        if ϵ > 1/2
            y ← y + 1
            ϵ ← ϵ − 1
        ϵ ← ϵ + m
        x ← x + 1
```

$\epsilon = m = \frac{\Delta y}{\Delta x}$

$\Rightarrow \Delta x\, \epsilon = \Delta y$

$\Rightarrow 2\Delta x\, \epsilon = 2\Delta y$

Use $d = 2\Delta x\, \epsilon$

Bresenham's Algorithm

$$
\begin{aligned}
&\texttt{line}(x0, y0, x1, y1)\\
&\quad x \leftarrow x0\\
&\quad y \leftarrow y0\\
&\quad \boxed{d = 2\Delta y}\\
&\quad \texttt{while}\ \ x < \ x1\\
&\qquad \texttt{pixel}(x, y)\\
&\qquad \texttt{if}\ \ \epsilon > \tfrac{1}{2}\\
&\qquad\qquad y \leftarrow y + 1\\
&\qquad\qquad \epsilon \leftarrow \epsilon - 1\\
&\qquad \epsilon \leftarrow \epsilon + m\\
&\qquad x \leftarrow x + 1
\end{aligned}
$$

$$
\begin{aligned}
&\epsilon = m = \frac{\Delta y}{\Delta x}\\
&\Rightarrow \Delta x\, \epsilon = \Delta y\\
&\Rightarrow 2\Delta x\, \epsilon = 2\Delta y
\end{aligned}
$$

Use $d = 2\Delta x\, \epsilon$

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    d = 2Δy
    while x < x1
        pixel(x, y)
        if ε > 1/2
            y ← y + 1
            ε ← ε − 1
        ε ← ε + m
        x ← x + 1
```

$\epsilon > \frac{1}{2}$

$\Rightarrow 2\Delta x\, \epsilon > \Delta \mathrm{x}$

$\Rightarrow d > \Delta x$

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    d = 2Δy
    while x < x1
        pixel(x, y)
        if d > Δx
            y ← y + 1
            ε ← ε − 1
        ε ← ε + m
        x ← x + 1
```

$\epsilon > \frac{1}{2}$

$\Rightarrow 2\Delta x\, \epsilon > \Delta \mathrm{x}$

$\Rightarrow d > \Delta x$

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    d = 2Δy
    while x < x1
       pixel(x, y)
       if d > Δx
           y ← y + 1
           ϵ ← ϵ − 1
       ϵ ← ϵ + m
       x ← x + 1
```

$\epsilon > \frac{1}{2}$

$\Rightarrow 2\Delta x\, \epsilon > \Delta \mathrm{x}$

$\Rightarrow d > \Delta x$

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    d = 2Δy
    while x < x1
        pixel(x, y)
        if d > Δx
            y ← y + 1
            d ← d − 2Δx
        ε ← ε + m
        x ← x + 1
```

$\epsilon > \frac{1}{2}$

$\Rightarrow 2\Delta x\, \epsilon > \Delta x$

$\Rightarrow d > \Delta x$

Bresenham's Algorithm

$$\texttt{line}(x0, y0, x1, y1)$$
$$x \leftarrow x0$$
$$\mathrm{y} \leftarrow y0$$
$$d = 2\Delta\mathrm{y}$$
$$\texttt{while}\ \ x < \ x1$$
$$\texttt{pixel}(x, y)$$
$$\texttt{if}\ \ d > \Delta\mathrm{x}$$
$$y \leftarrow y + 1$$
$$d \leftarrow d - 2\Delta\mathrm{x}$$
$$\epsilon \leftarrow \epsilon + m$$
$$x \leftarrow x + 1$$

$$\epsilon > \frac{1}{2}$$
$$\Rightarrow 2\Delta x\, \epsilon > \Delta\mathrm{x}$$
$$\Rightarrow d > \Delta x$$

Bresenham's Algorithm

```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    d = 2Δy
    while x < x1
        pixel(x, y)
        if d > Δx
            y ← y + 1
            d ← d − 2Δx
        d ← d + 2Δy
        x ← x + 1
```

$\epsilon > \frac{1}{2}$

$\Rightarrow 2\Delta x\, \epsilon > \Delta \mathrm{x}$

$\Rightarrow d > \Delta x$

Bresenham's Algorithm

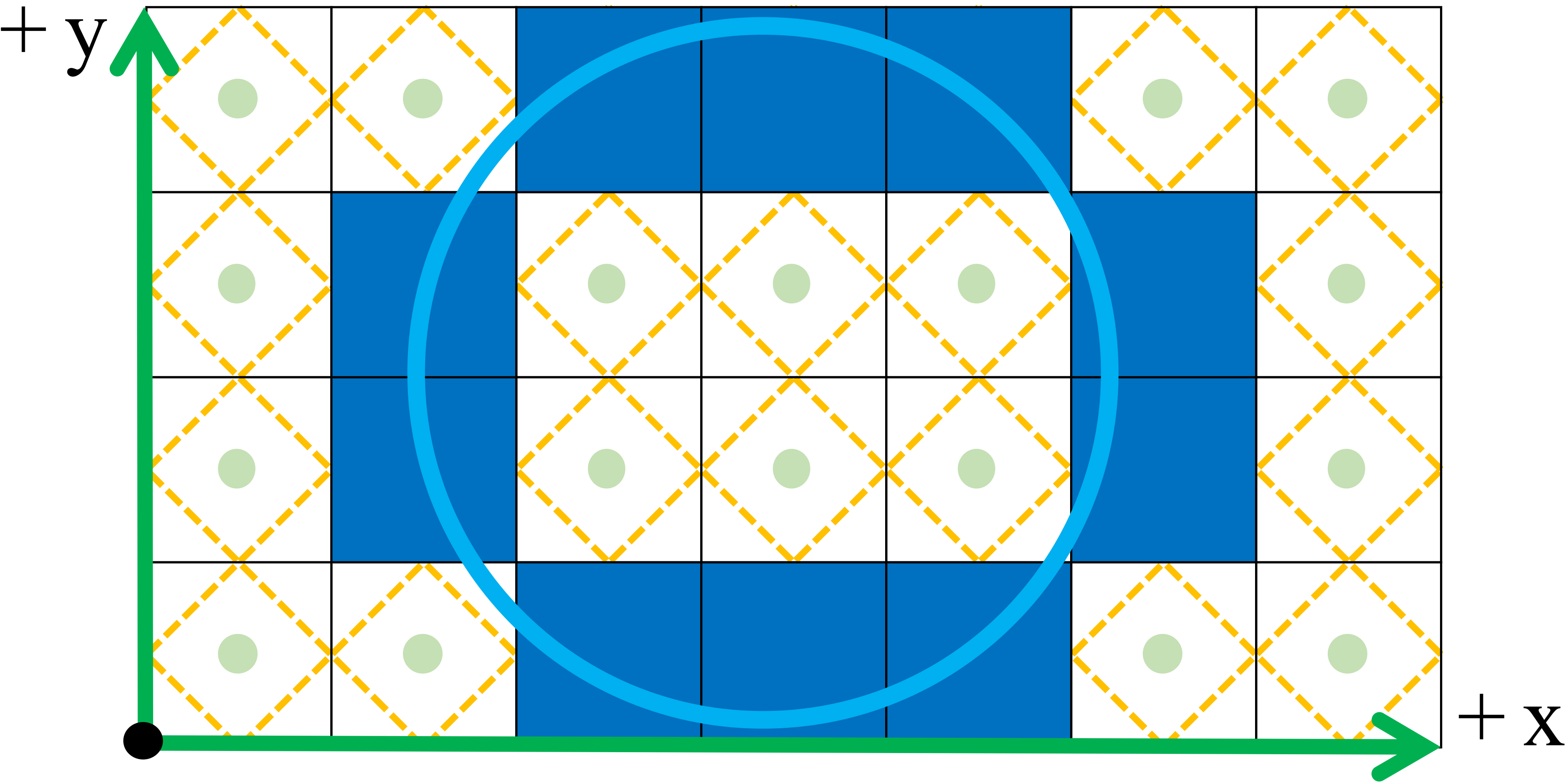
```
line(x0, y0, x1, y1)
    x ← x0
    y ← y0
    Δy ← y1 − y0
    Δx ← x1 − x0
    twoDY ← Δy ≪ 1
    twoDX ← Δx ≪ 1
    d = twoDY
    while x < x1
      pixel(x, y)
      if d > Δx
          y ← y + 1
          d ← d − twoDX
      d ← d + twoDY
      x ← x + 1
```

Different Method: Same Output

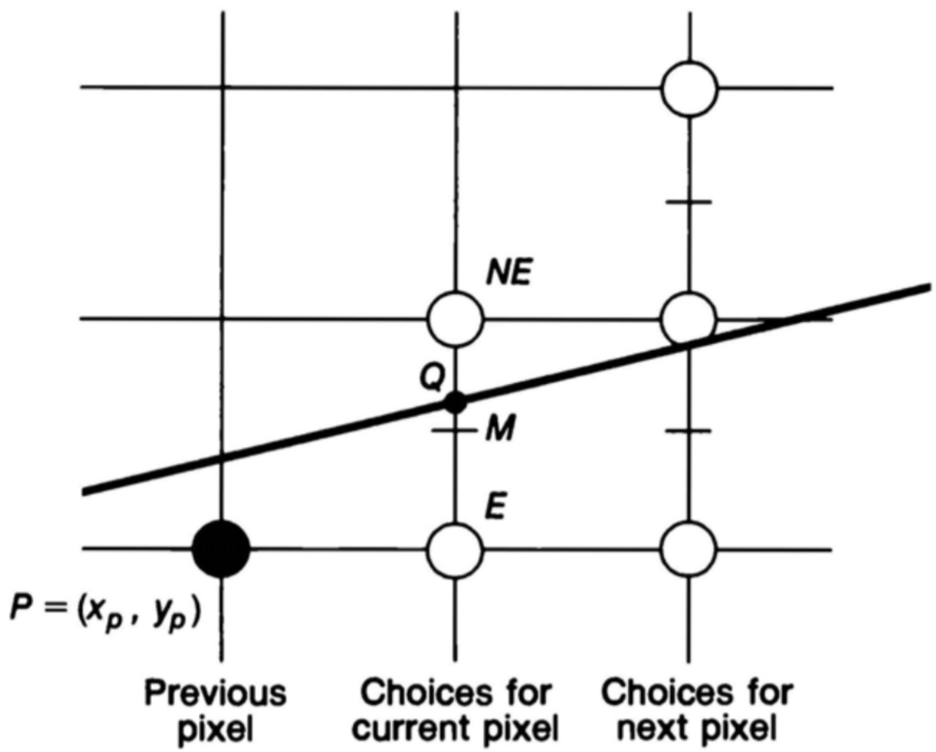
Pitteway (1967)
"Midpoint Algorithm"

READING ASSIGNMENT !

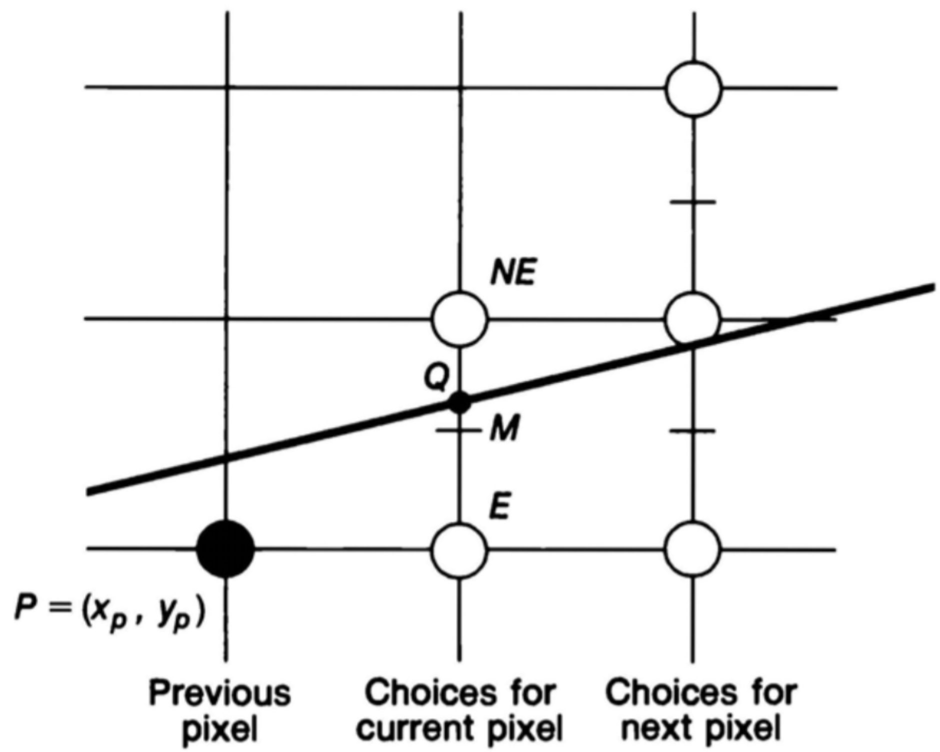
Rasterizing Circle



Midpoint Algorithm: Idea



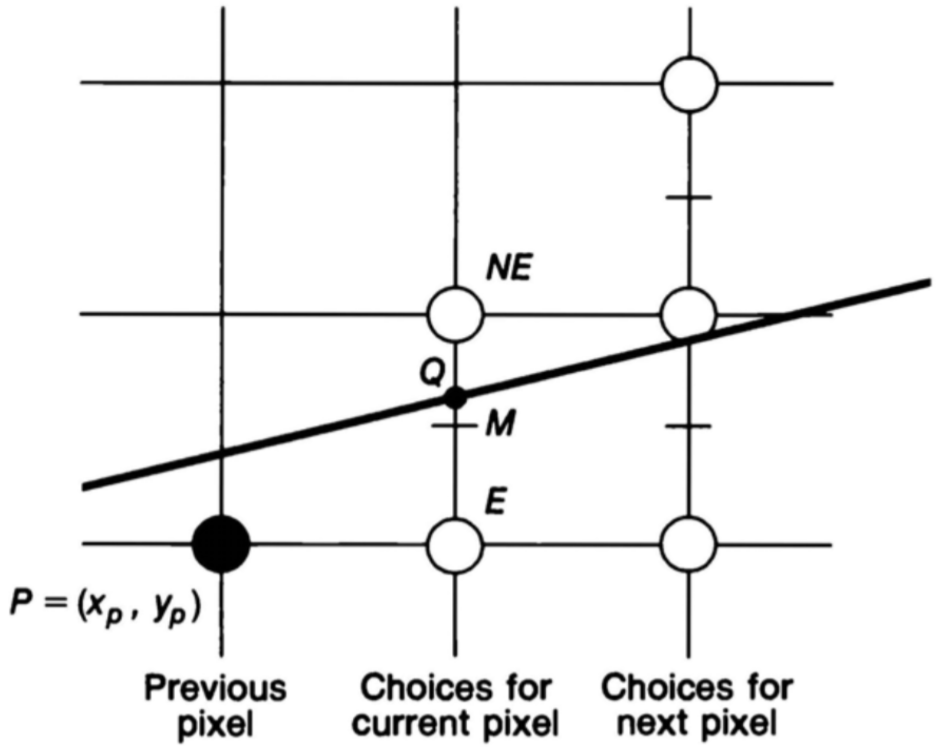
Midpoint Algorithm: Idea



Equation of a line:

$$ax + by + c = 0$$

Midpoint Algorithm: Idea



Equation of a line: Implicit Form

$$F(x, y) = ax + by + c$$

Midpoint Algorithm: Idea

$$y = \frac{\Delta y}{\Delta x} x + B$$

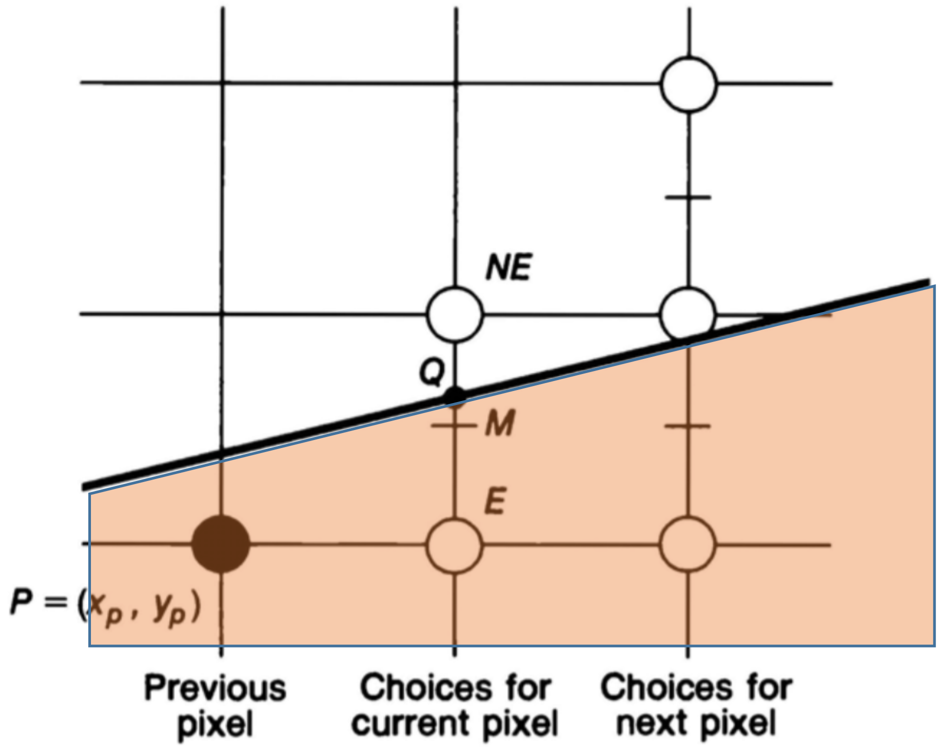
$$\implies \Delta x \cdot y = \Delta y \cdot x + \Delta x \cdot B$$

$$\implies 0 = \Delta y \cdot x - \Delta x \cdot y + \Delta x \cdot B$$

Equation of a line: Implicit Form

$$F(x, y) = \underbrace{\Delta y}_{a} \cdot x + \underbrace{(-\Delta x)}_{b} \cdot y + \underbrace{\Delta x \cdot B}_{c}$$

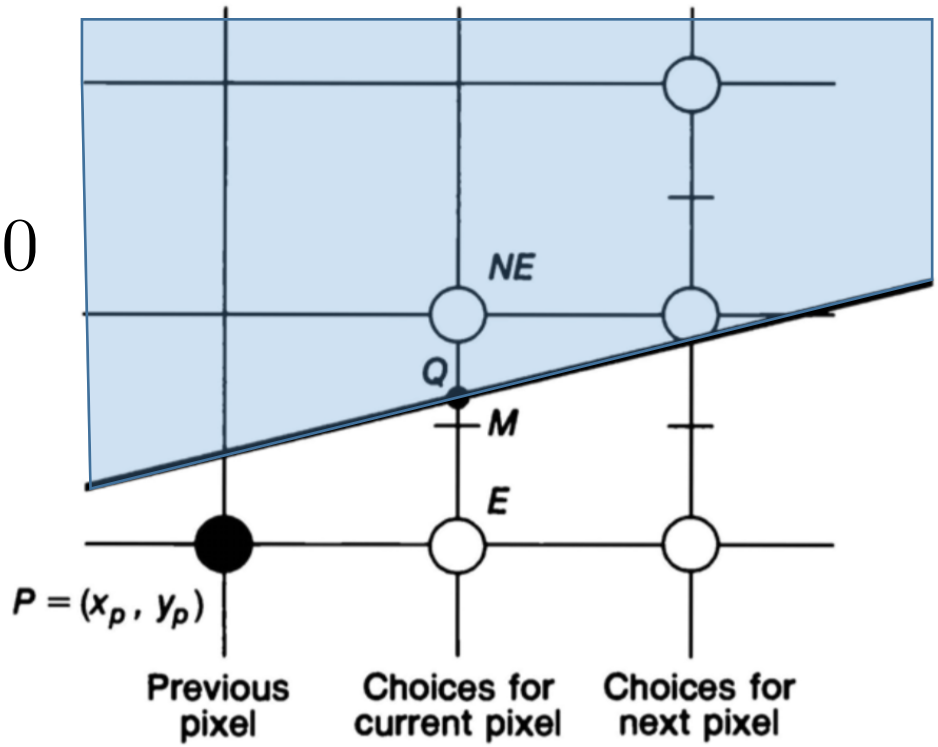
Midpoint Algorithm: Idea



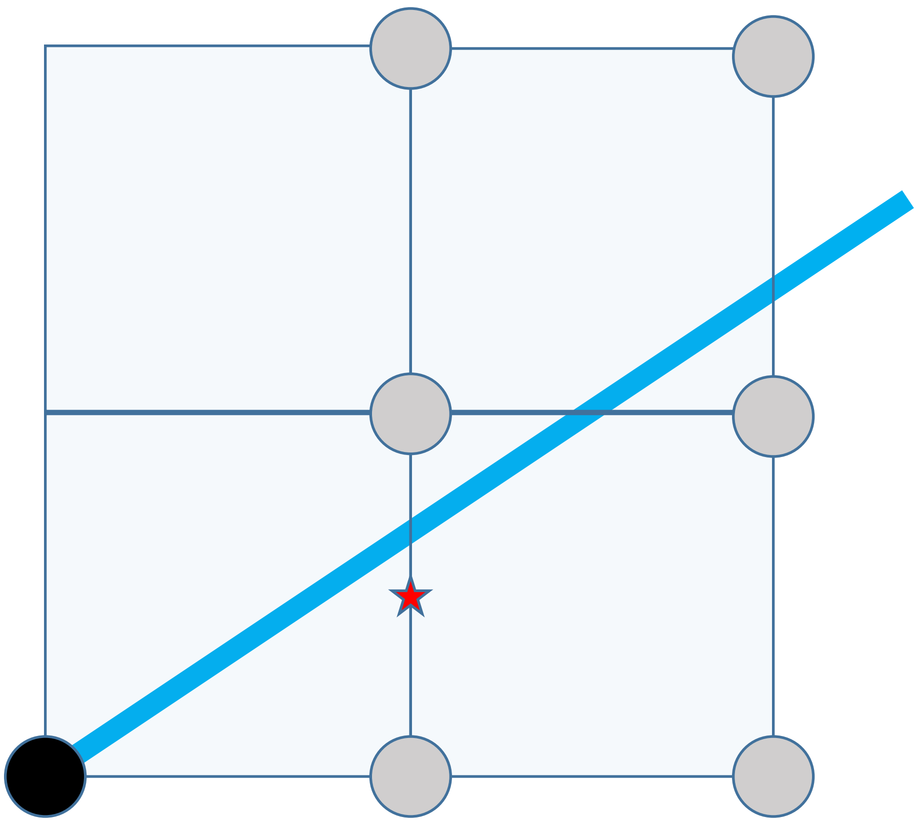
$F(x, y) > 0$

Midpoint Algorithm: Idea

$F(x, y) < 0$



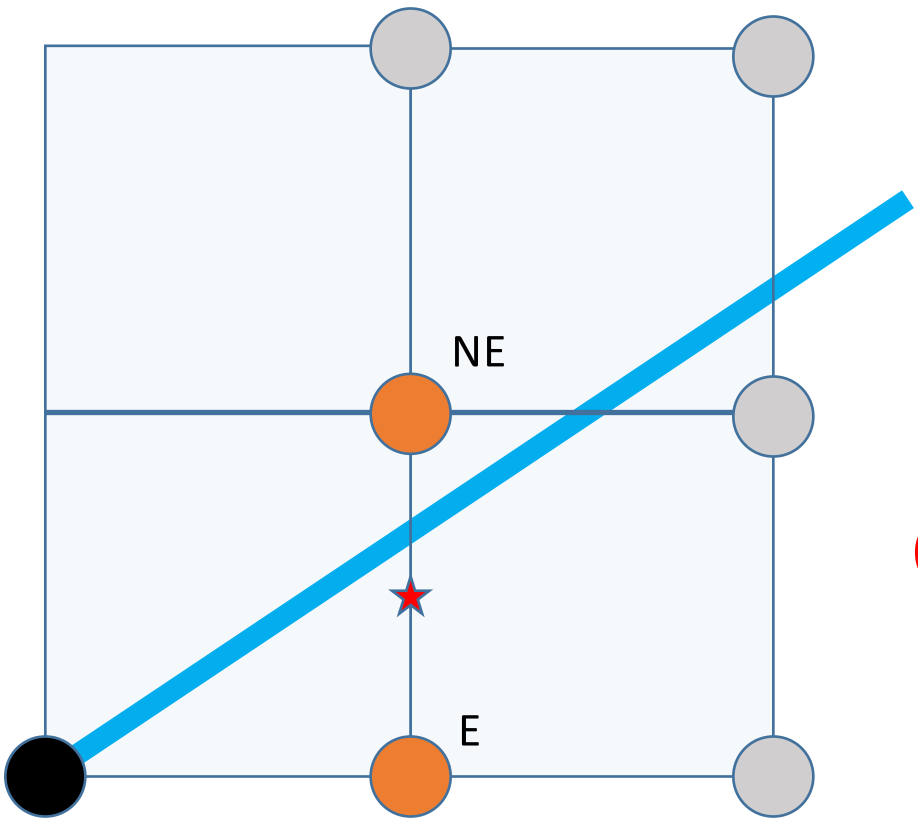
Midpoint Algorithm: Idea



Assume

$$0 \le m < 1$$

Midpoint Algorithm: Idea

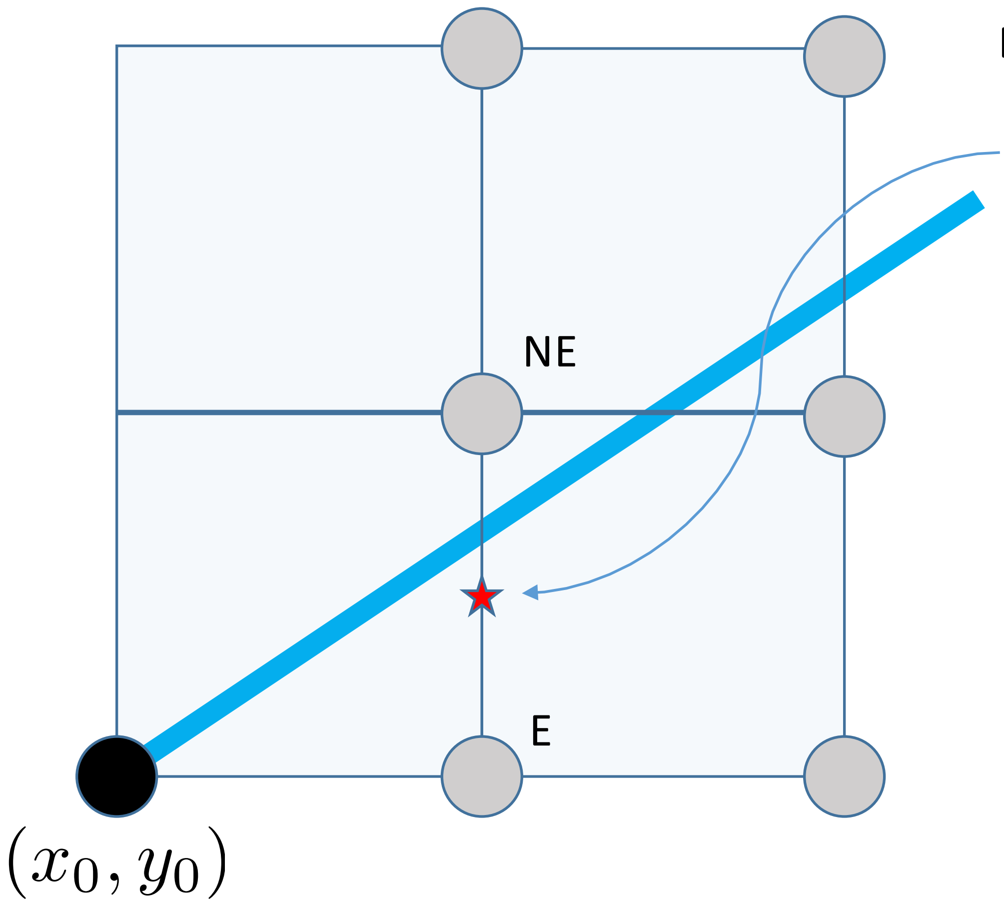


Assume

$$0 \leq m < 1$$

Choose between E and NE

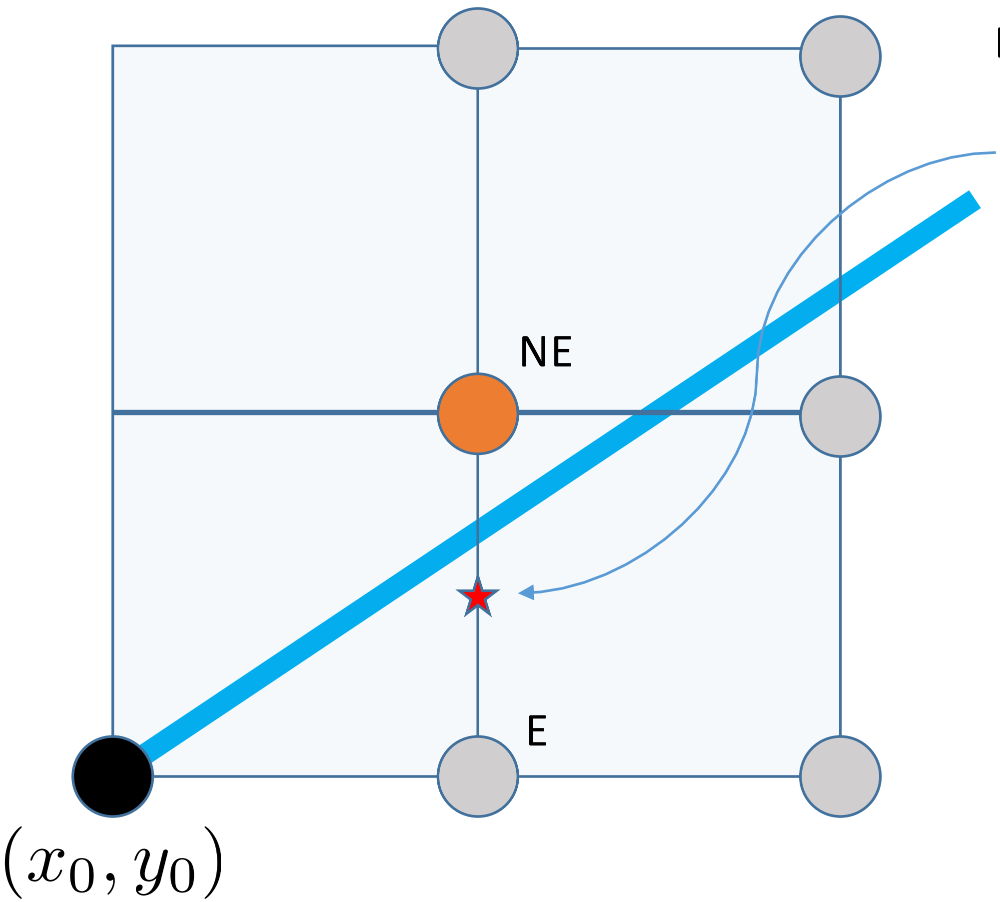
Midpoint Algorithm: Idea



Decision variable: for next move

$$d = F\left(x_0 + 1, y_0 + \frac{1}{2}\right)$$
$$= a \cdot (x_0 + 1) + b \cdot \left(y_0 + \frac{1}{2}\right) + c$$
$$= F(x_0, y_0) + a + \frac{b}{2}$$
$$= a + \frac{b}{2}, \quad (x_0, y_0) \text{ is on the line}$$

Midpoint Algorithm: Idea



Decision variable: for next move

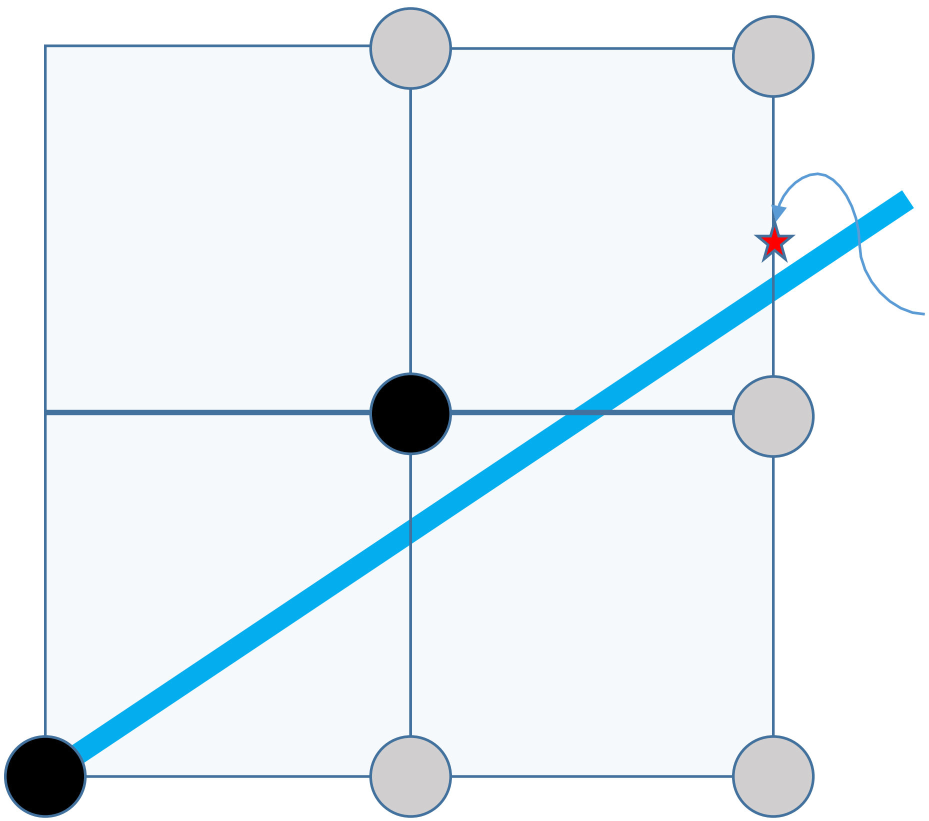
$$d = F\left(x_0 + 1, y_0 + \frac{1}{2}\right)$$
$$= a \cdot (x_0 + 1) + b \cdot \left(y_0 + \frac{1}{2}\right) + c$$
$$= F(x_0, y_0) + a + \frac{b}{2}$$
$$= a + \frac{b}{2}, \quad (x_0, y_0) \text{ is on the line}$$

In this case

$$d > 0$$

Choose NE

Midpoint Algorithm: Idea



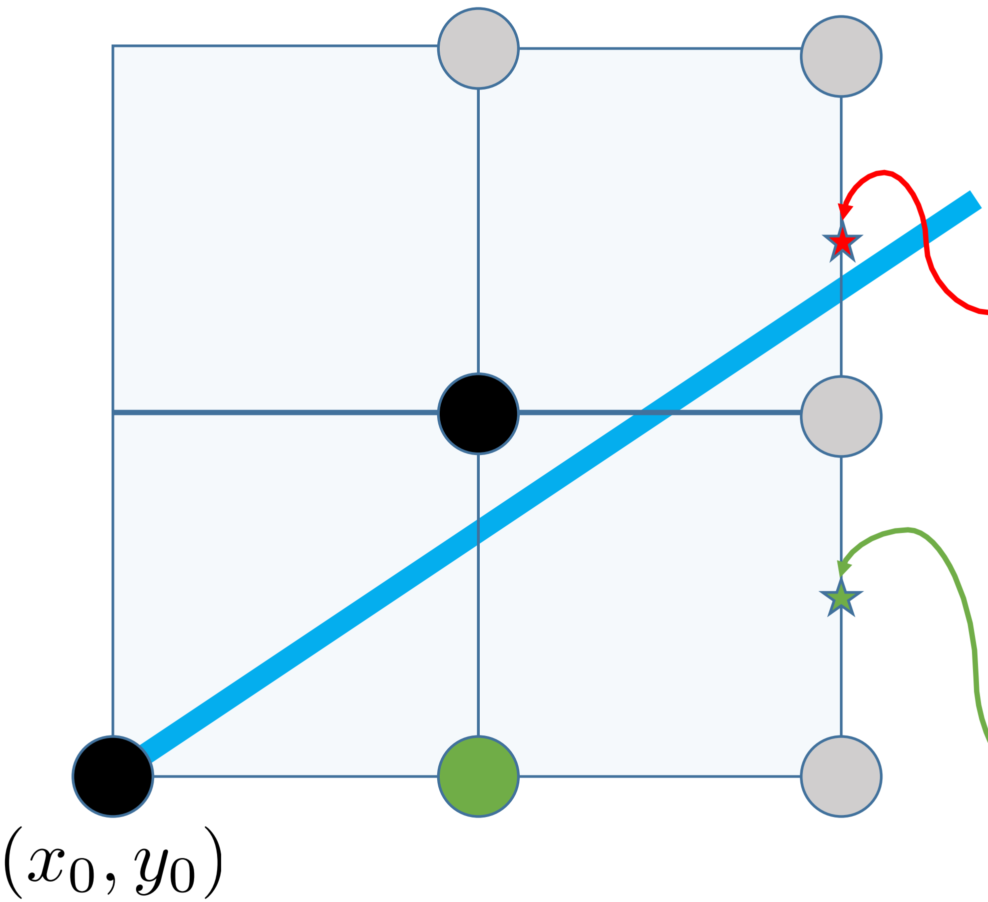
Decision variable update

When NE was chosen

$$d = F\left(x_0 + 2, y_0 + \frac{3}{2}\right)$$
$$= a \cdot (x_0 + 2) + b \cdot \left(y_0 + \frac{3}{2}\right) + c$$
$$= d_{old} + \underbrace{(a + b)}_{\Delta d_{NE}}$$

(x_0, y_0)

Midpoint Algorithm: Idea



Decision variable update

When NE was chosen

$$d = F\left(x_0 + 2, y_0 + \frac{3}{2}\right)$$

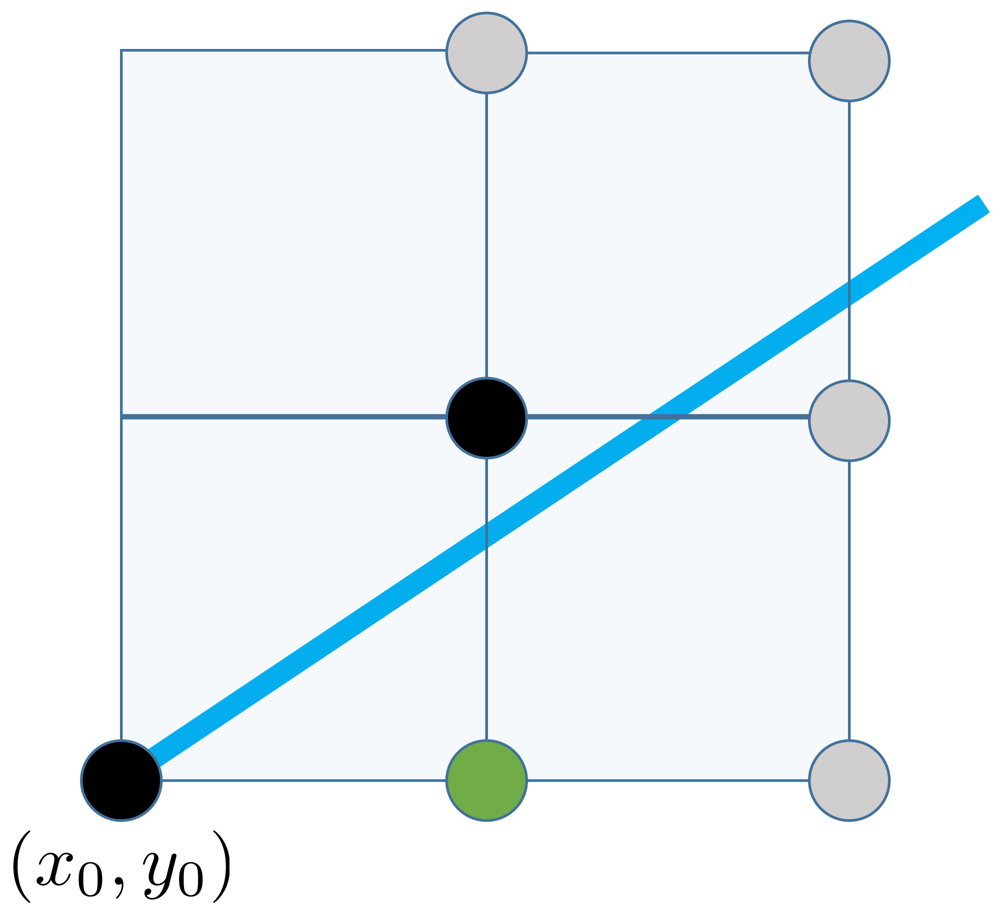
$$= a \cdot (x_0 + 2) + b \cdot \left(y_0 + \frac{3}{2}\right) + c$$

$$= d_{old} + \underbrace{(a + b)}_{\Delta d_{NE}}$$

If E were chosen instead

$$d = d_{old} + \underbrace{a}_{\Delta d_E}$$

Midpoint Algorithm: Idea

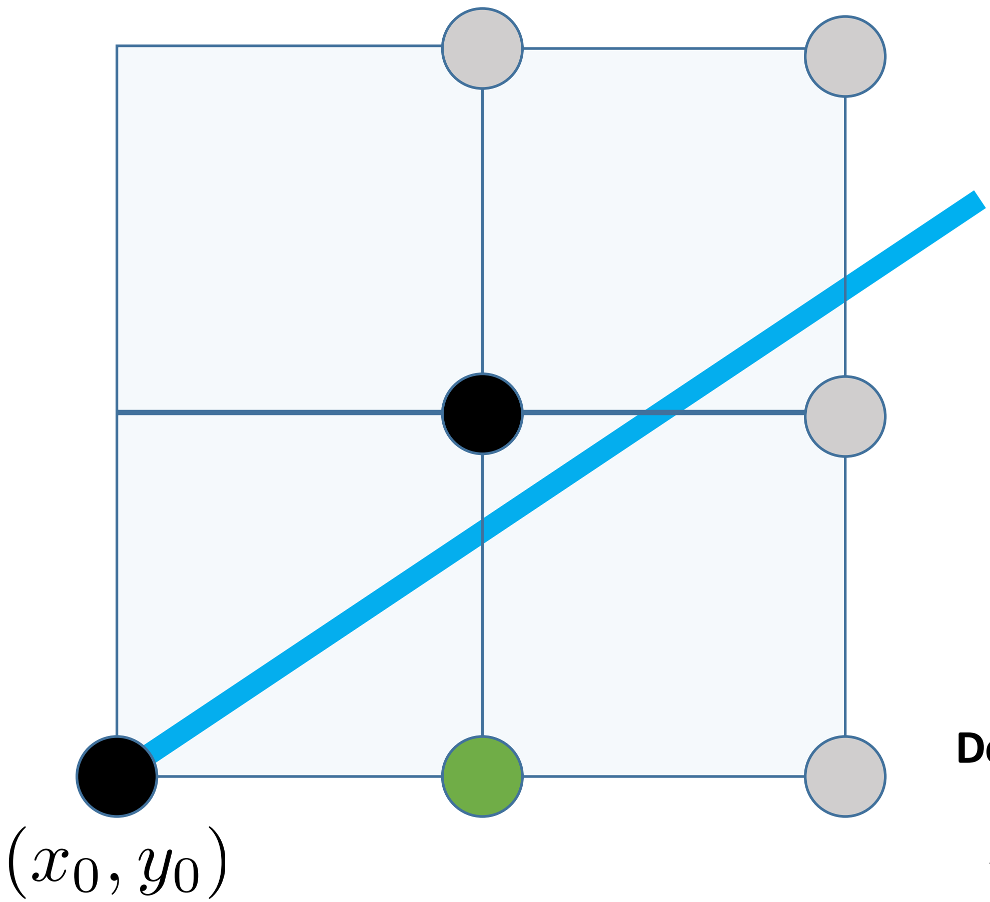


Recall

$$d_{start} = a + \boxed{\frac{b}{2}}$$

Fraction!

Midpoint Algorithm: Idea



Recall

$$d_{start} = a + \boxed{\frac{b}{2}}$$

Fraction!

Define Implicit Function as:

$$F(x, y) = 2\,(ax + by + c)$$

Midpoint Algorithm: Summary

$$y = \frac{\Delta y}{\Delta x} x + B$$

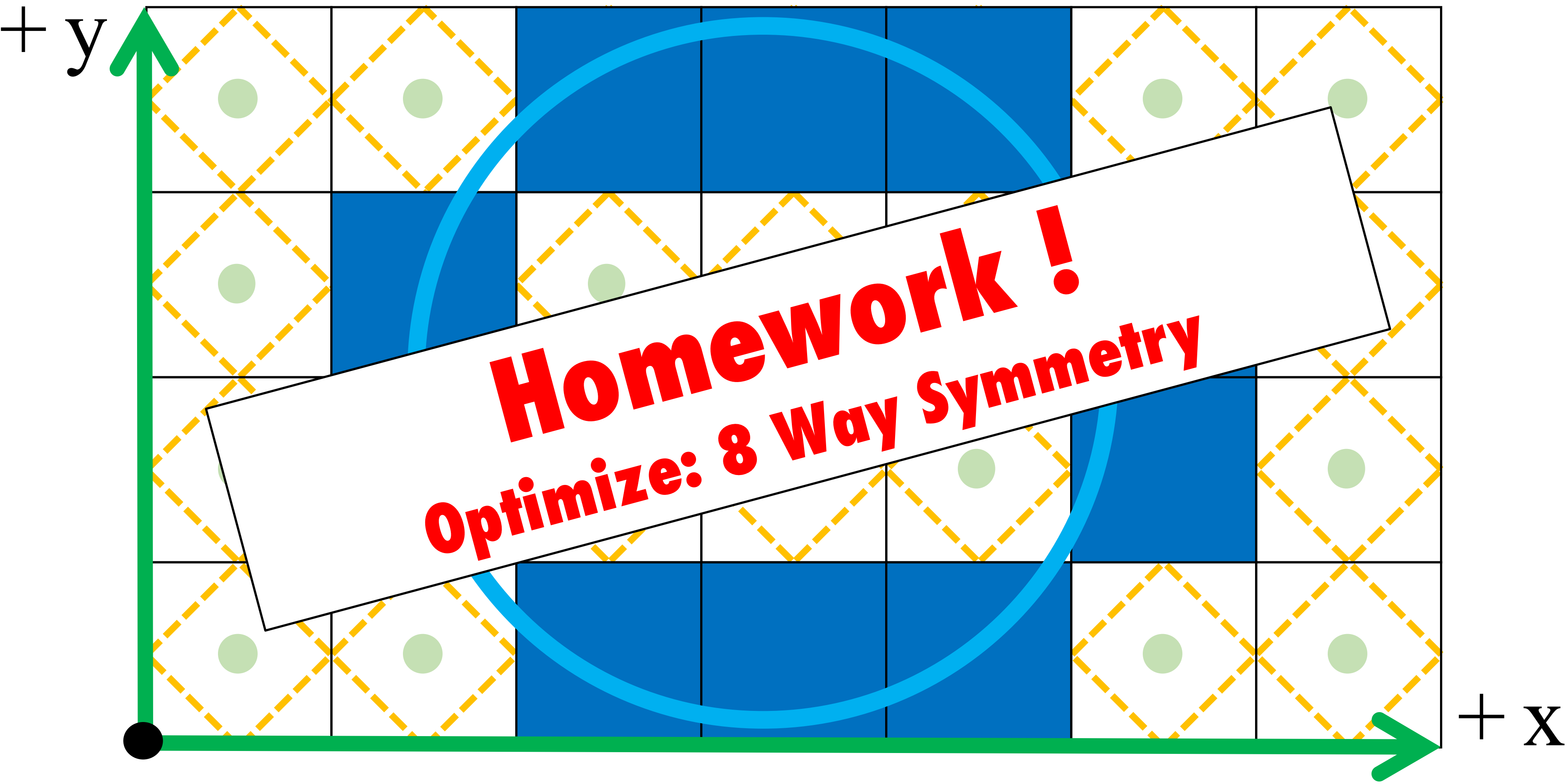
$$a = \Delta y, \quad b = -\Delta x, \quad c = \Delta x \cdot B$$

$$d_0 = 2a + b$$

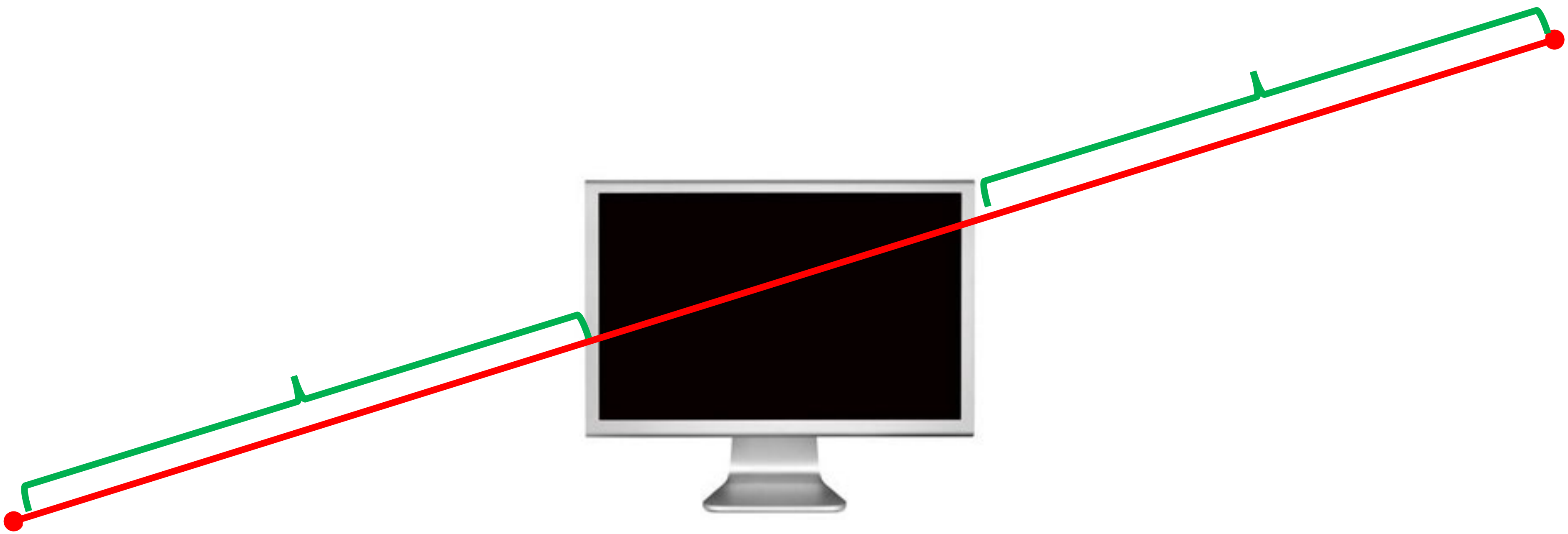
$$\text{if } d_i \leq 0 \text{ Choose E and } d_{i+1} = d_i + 2a$$

$$\text{if } d_i > 0 \text{ Choose NE and } d_{i+1} = d_i + 2(a + b)$$

Circle with Midpoint Algorithm



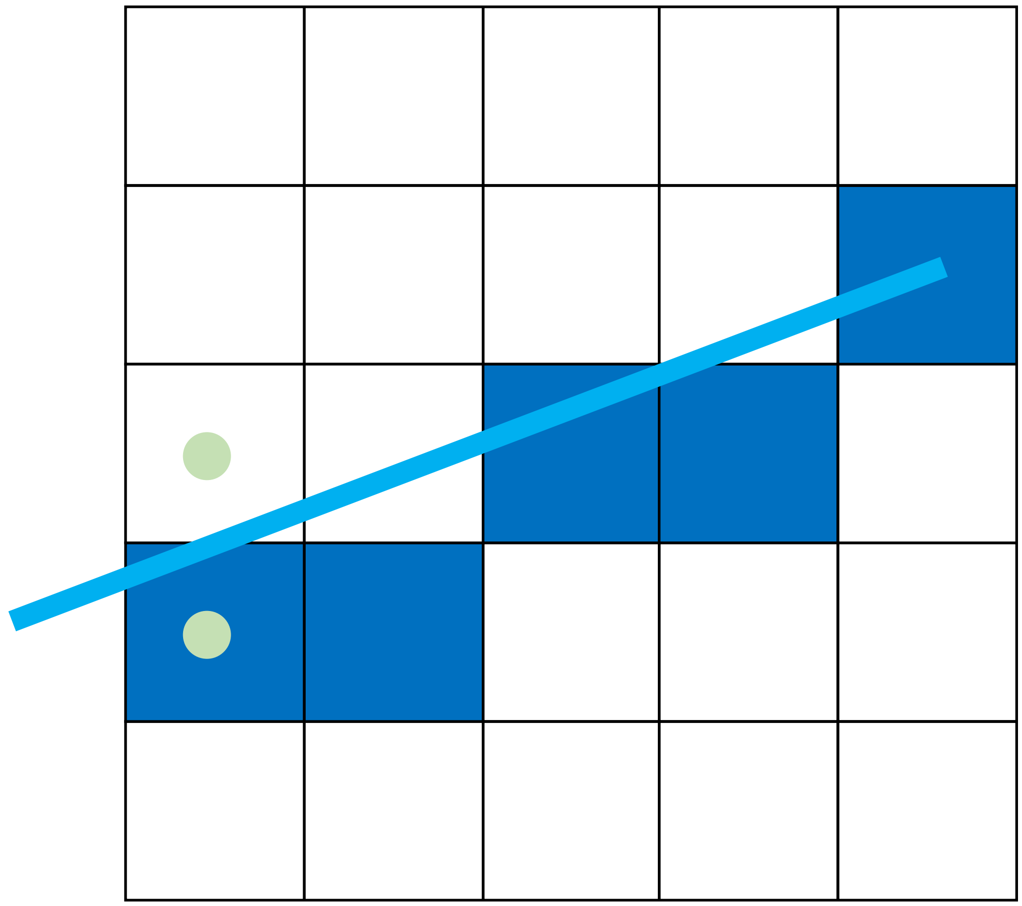
Important Issues We've Ignored



Clipping

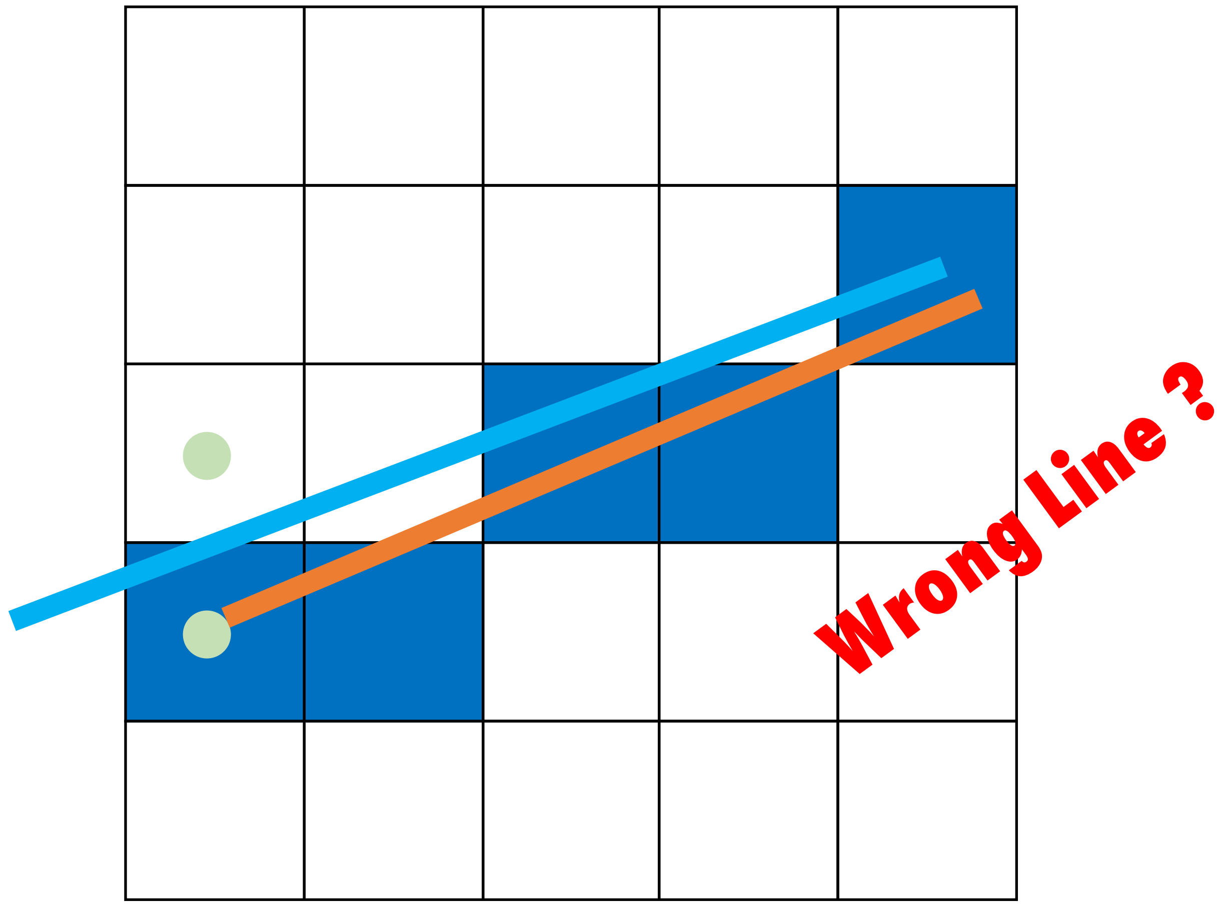
http://www.cheap-computermonitors.com/images/1-cheap-flat-screen-computer-monitors.jpg

Important Issues We've Ignored



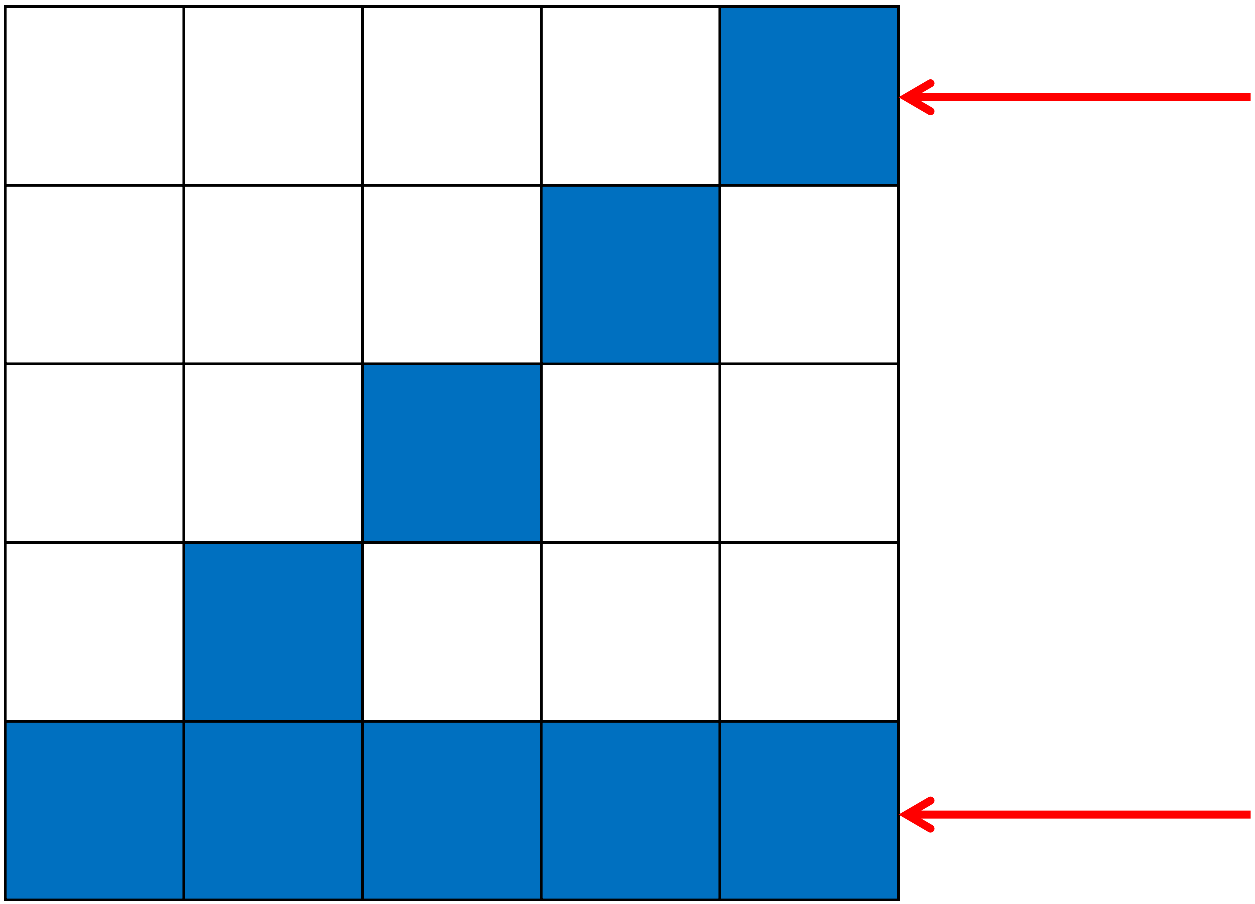
Clipping

Important Issues We've Ignored



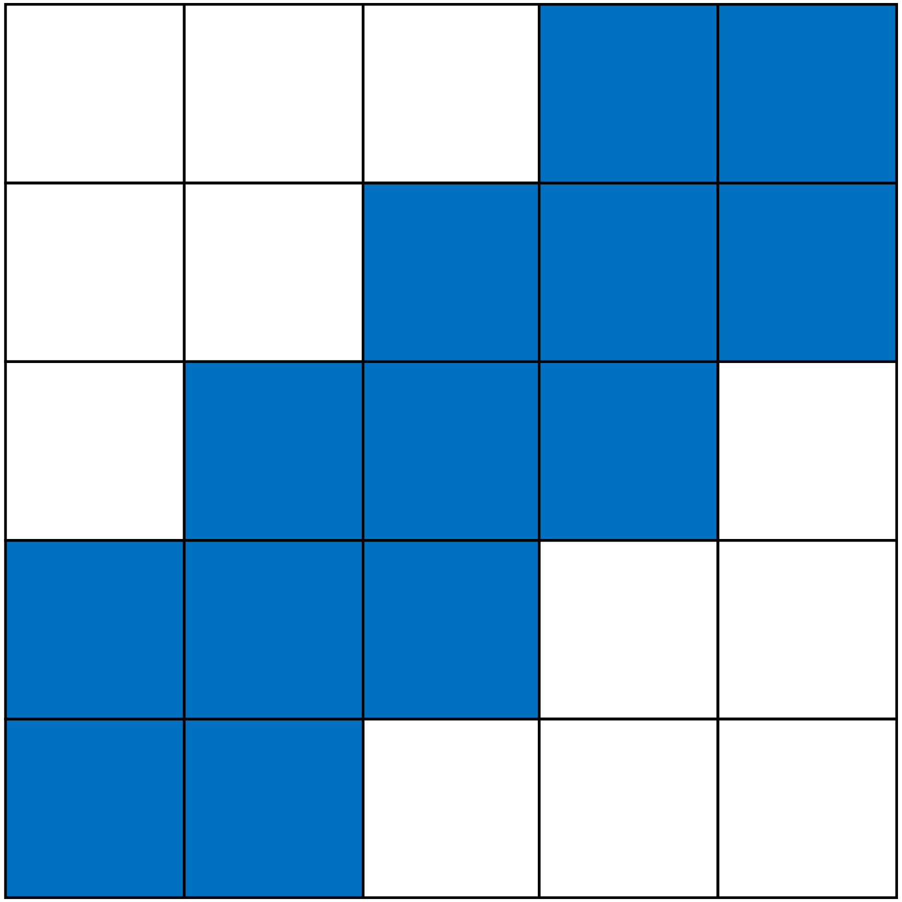
Clipping

Important Issues We've Ignored



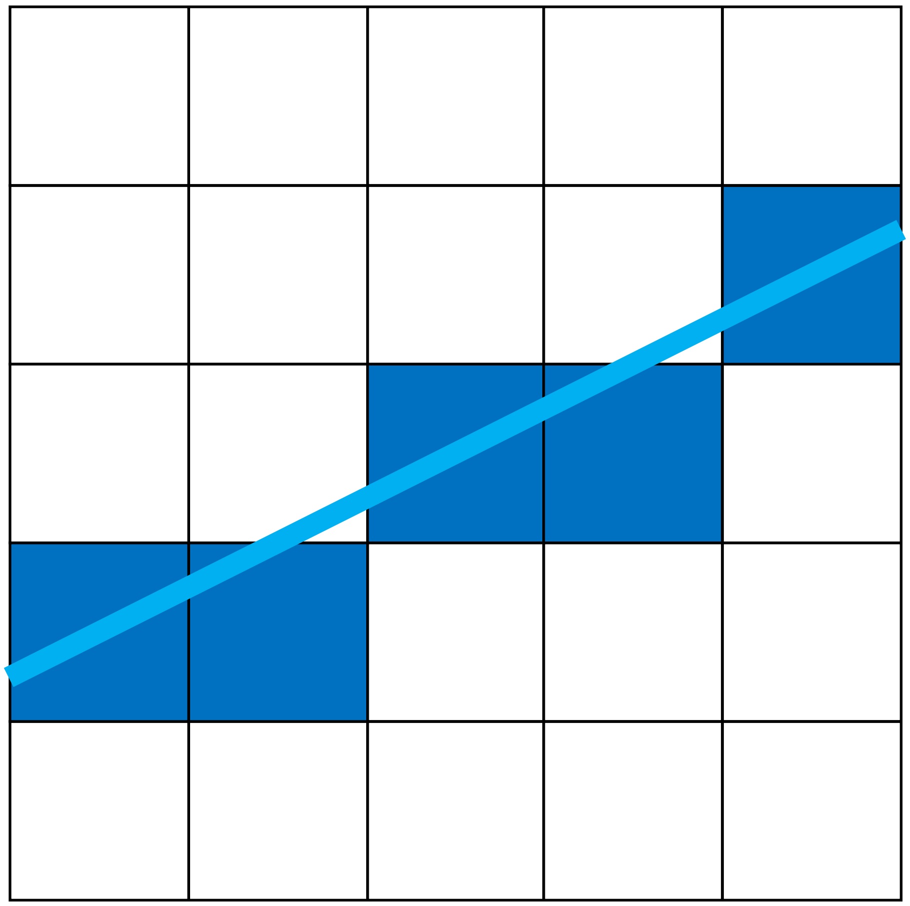
Line intensity

Important Issues We’ve Ignored



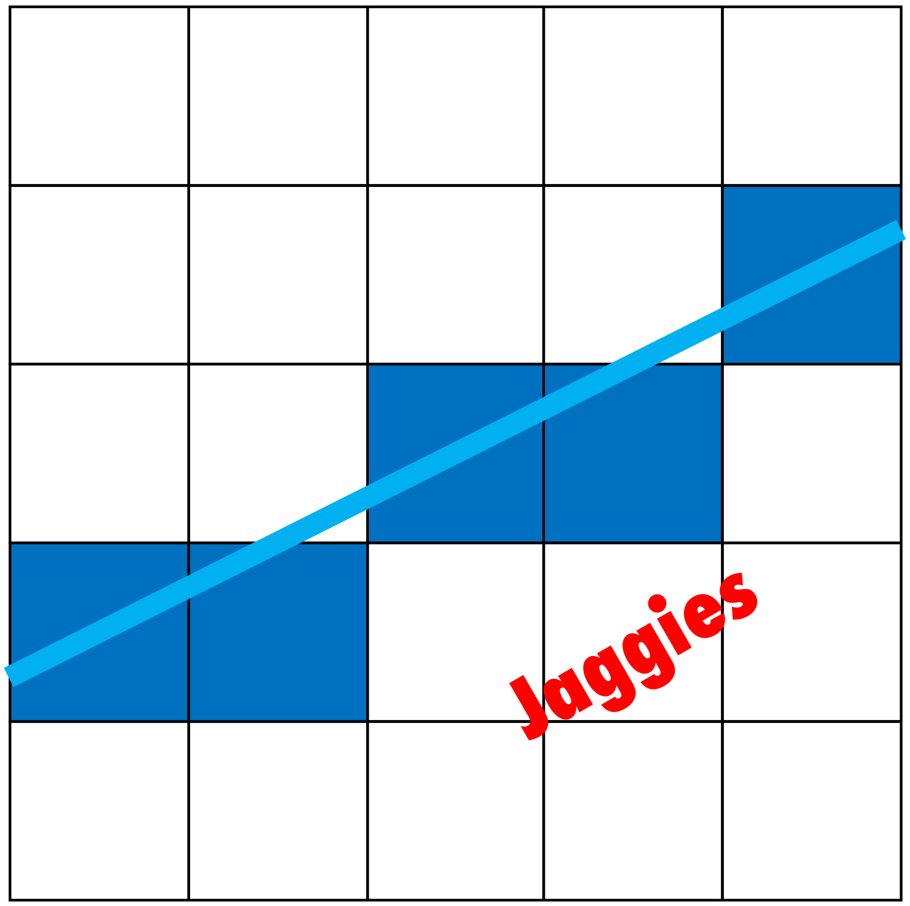
Line Thickness

Important Issues We've Ignored



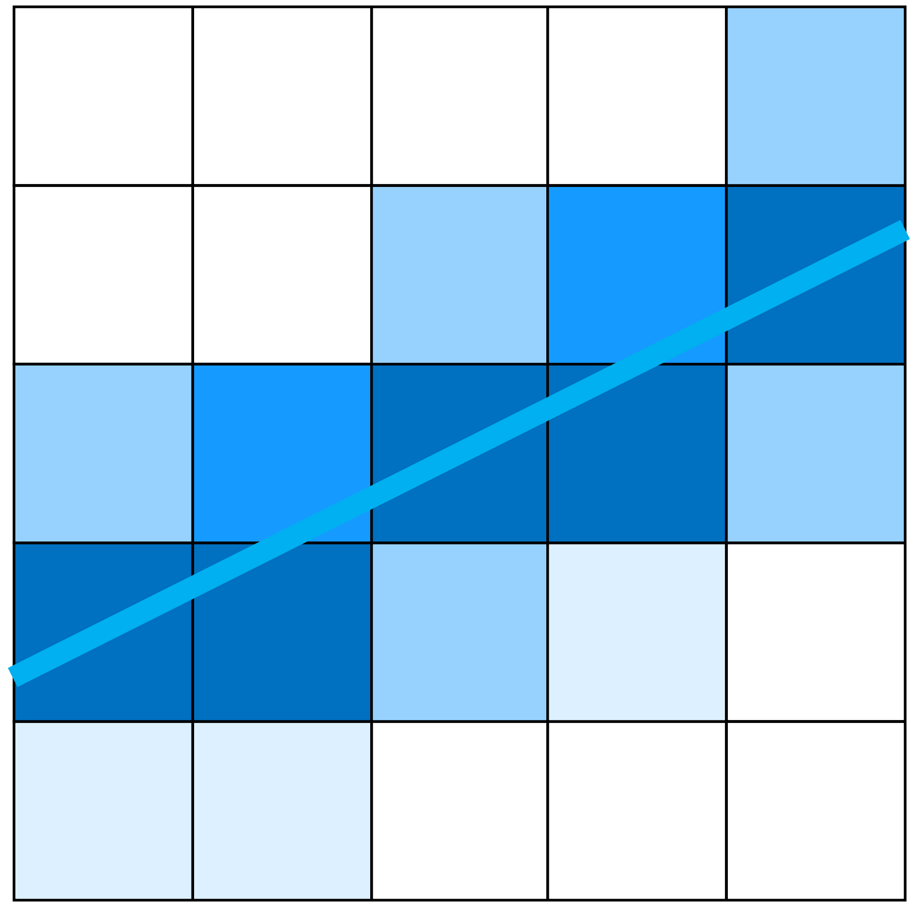
Antialiasing

Important Issues We've Ignored



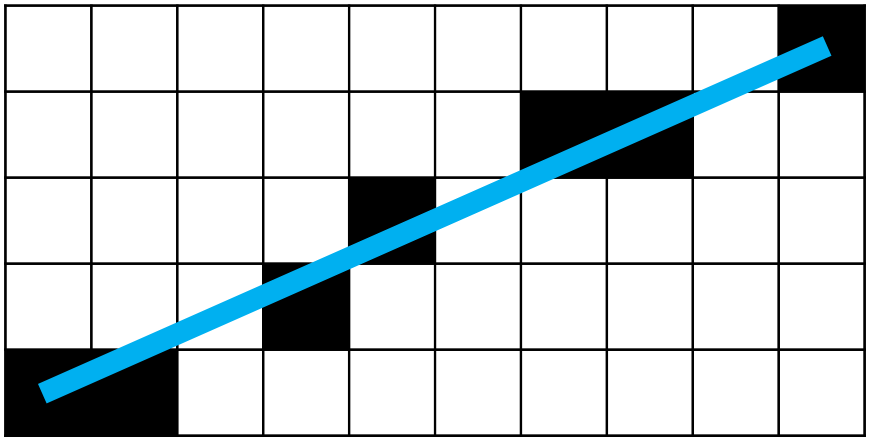
Antialiasing

Important Issues We've Ignored



Antialiasing

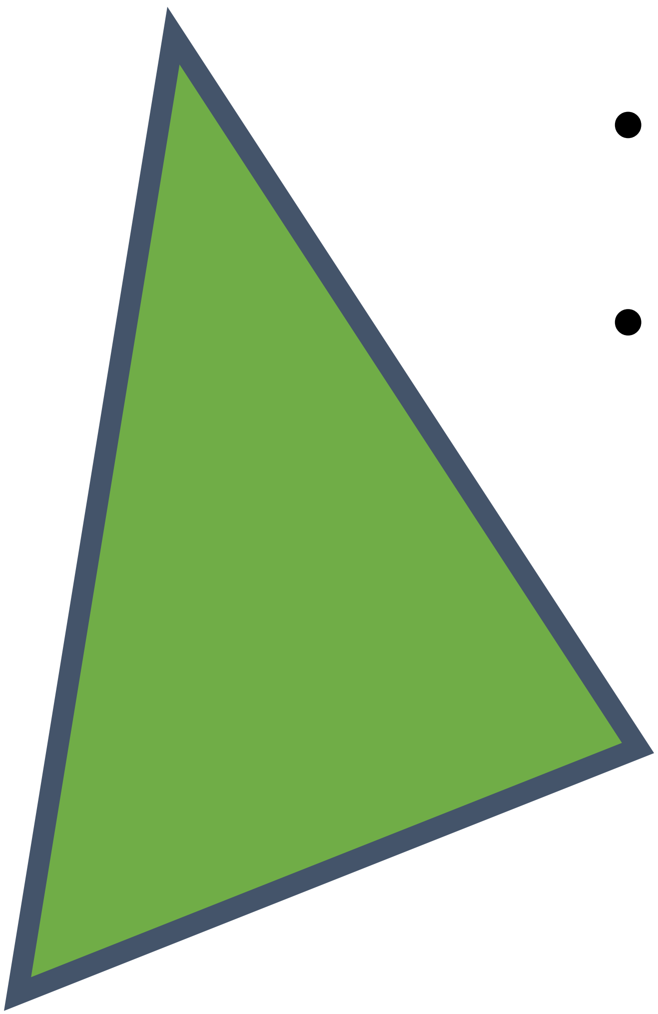
Important Issues We've Ignored



Problems with drawing in reverse ?

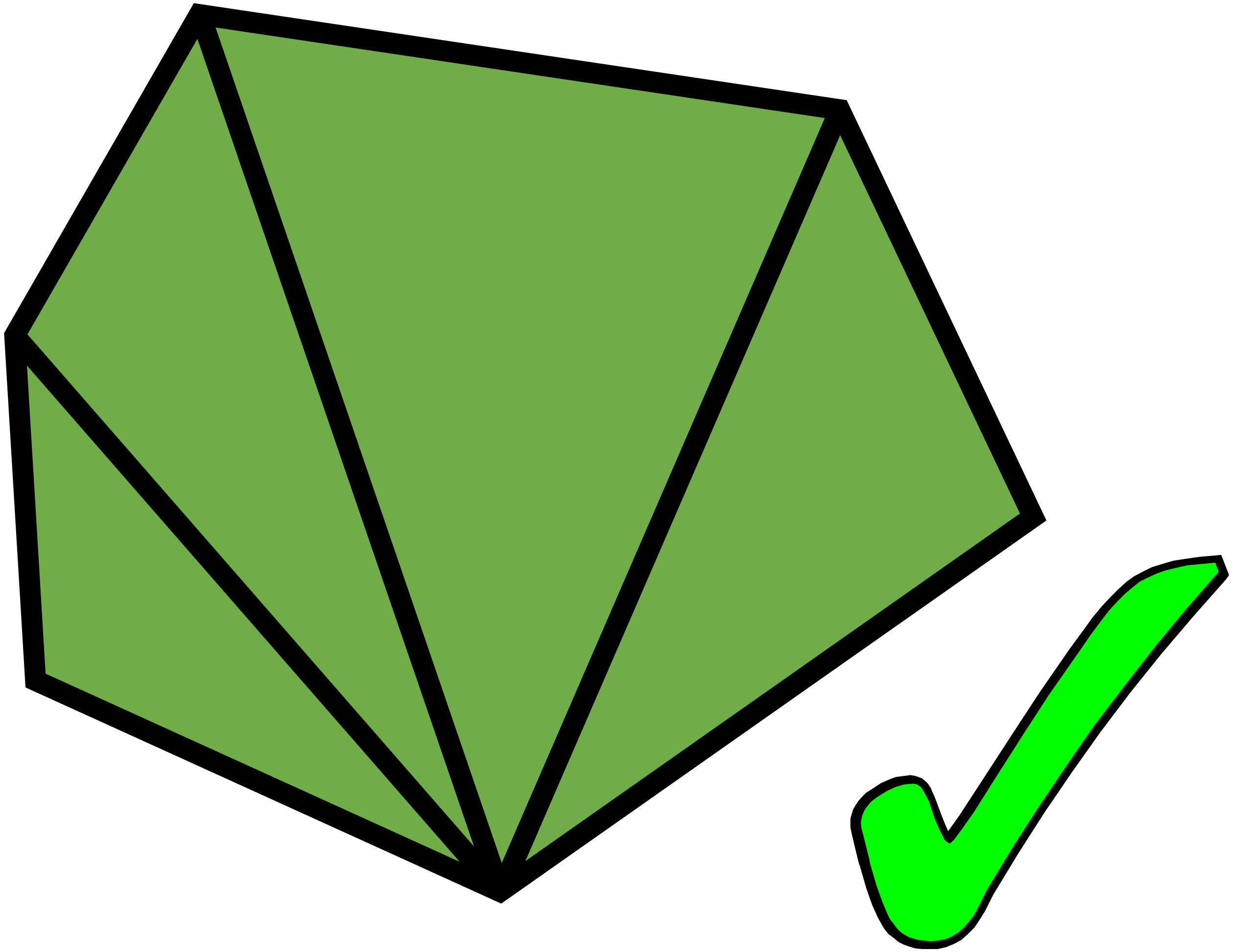
Stippling Patterns

Filling Triangles

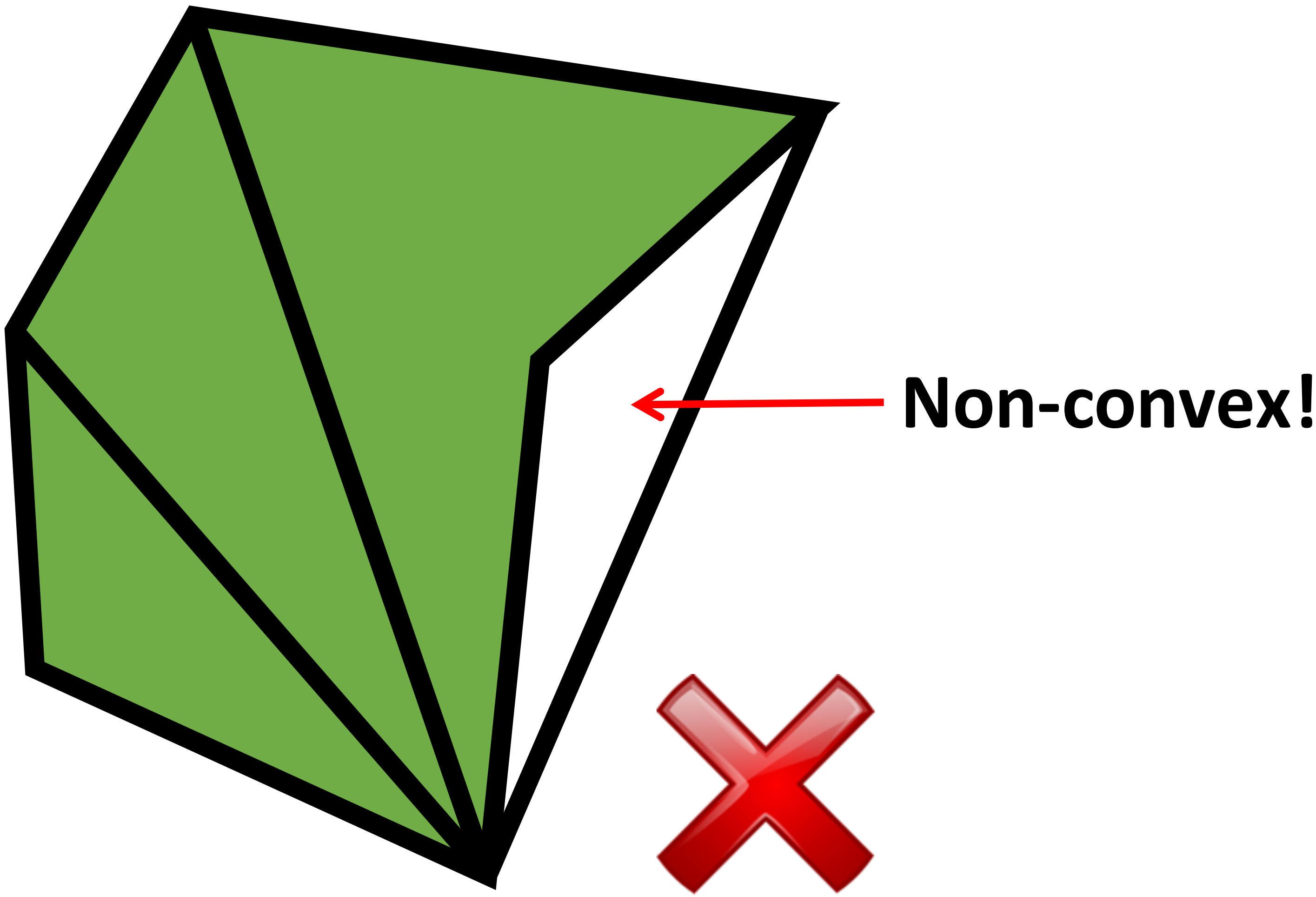


- Optimize a single primitive
- Nice properties:
  - Planar
  - “Inside” well-defined
  - Straightforward shading

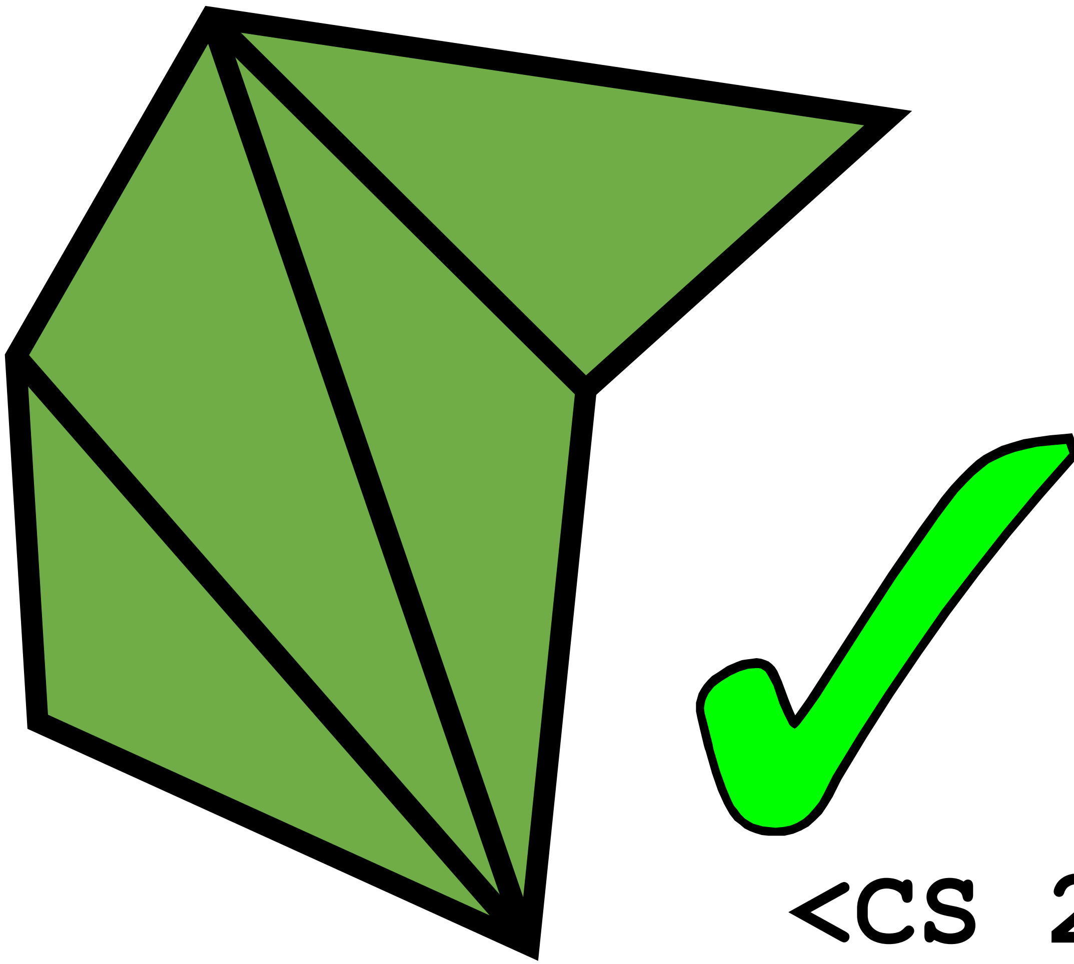
Have We Lost Anything?



Have We Lost Anything?

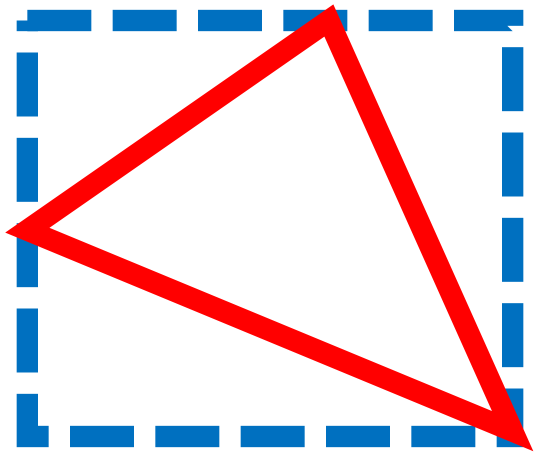


Have We Lost Anything?



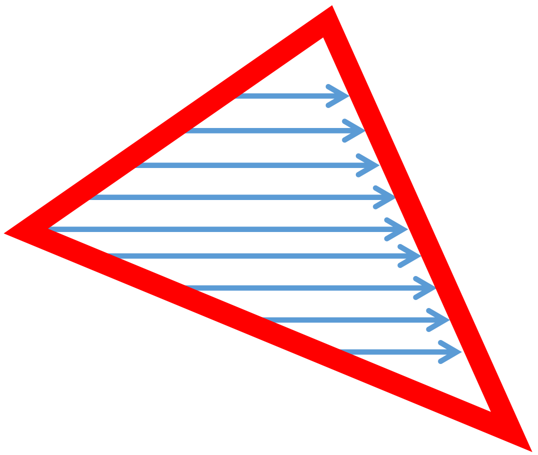
<CS 268/>

Two Methods for Filling



Check if each pixel in bounding box is inside the triangle.

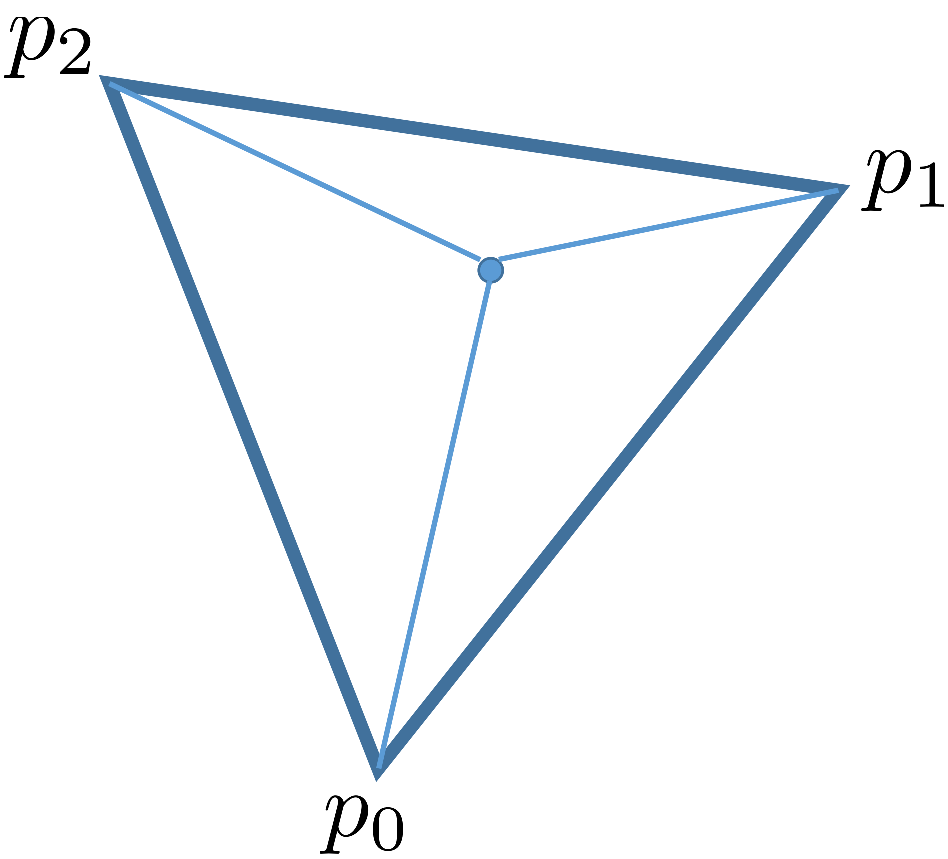
Parallelizable



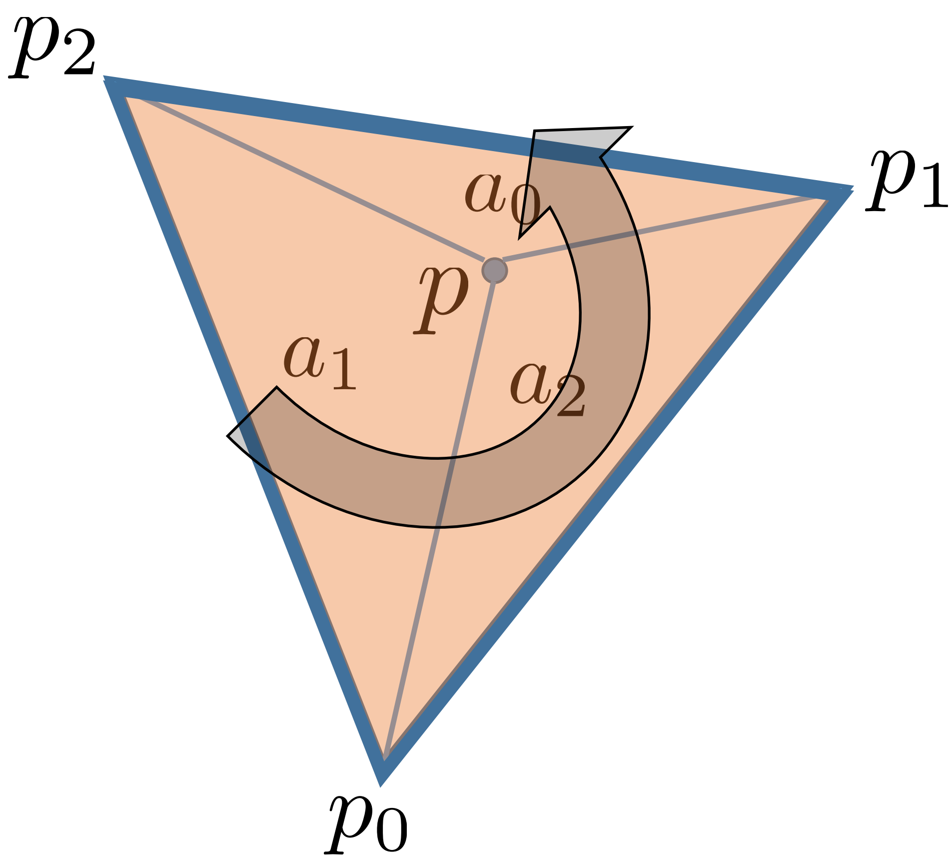
Rasterize border; sweep from left to right.

Less Computation

Barycentric Coordinates

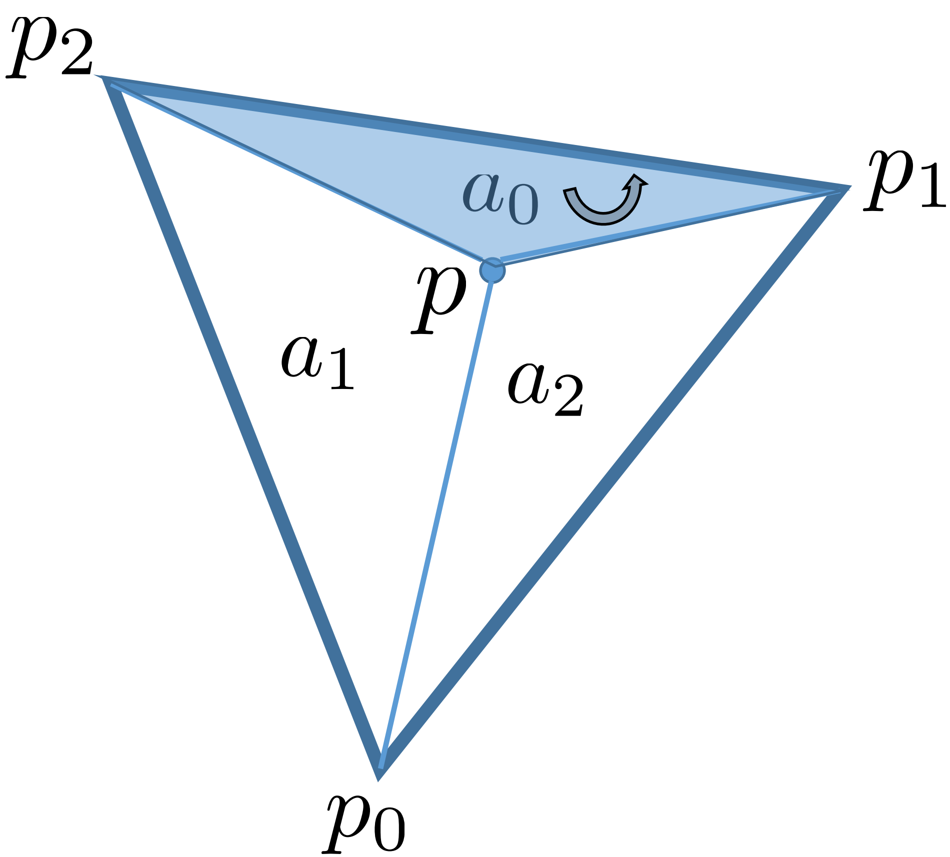


Barycentric Coordinates



$$A = Area(p_0, p_1, p_2)$$

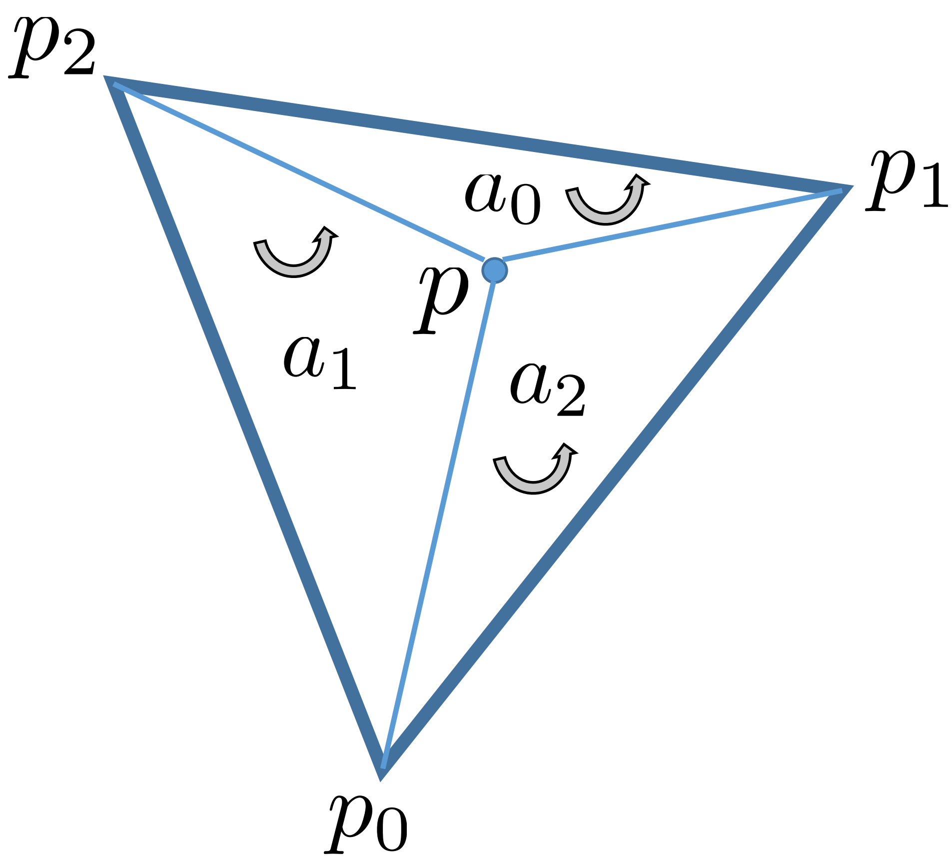
Barycentric Coordinates



$$A = Area(p_0, p_1, p_2)$$

$$a_0 = \frac{Area(p, p_1, p_2)}{A}$$

Barycentric Coordinates



$$A = Area(p_0, p_1, p_2)$$

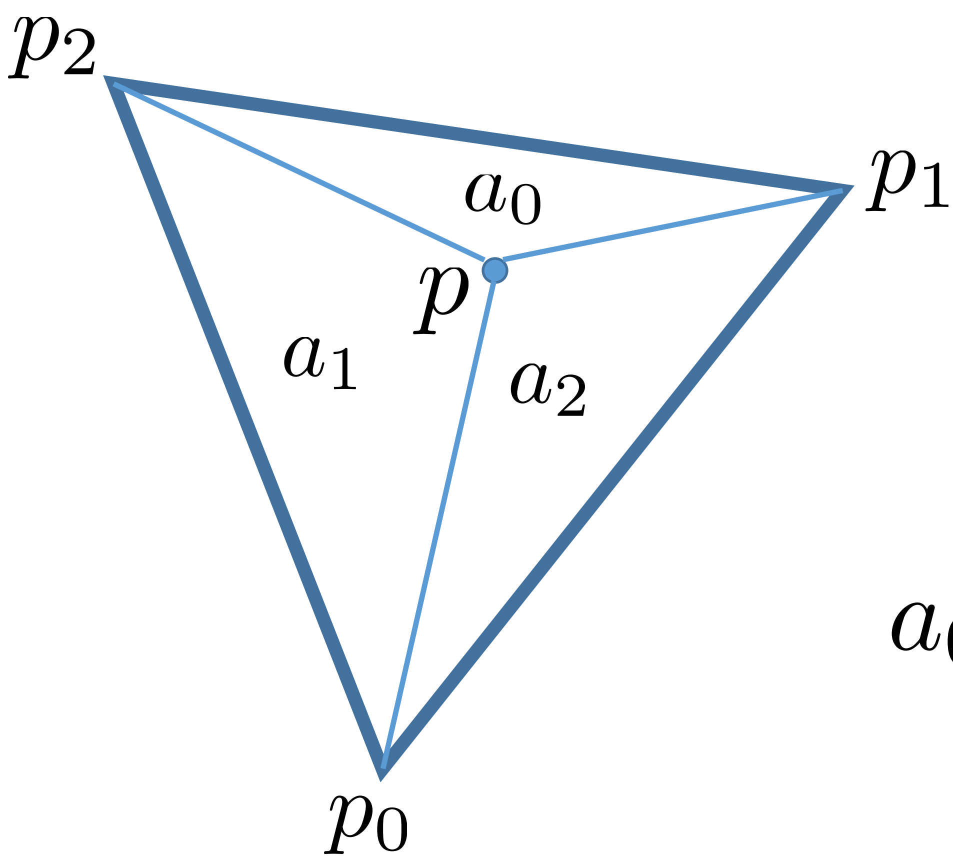
$$a_0 = \frac{Area(p, p_1, p_2)}{A}$$

$$a_1 = \frac{Area(p, p_2, p_0)}{A}$$

$$a_2 = \frac{Area(p, p_1, p_2)}{A}$$

$$a_0 + a_1 + a_2 = 1$$

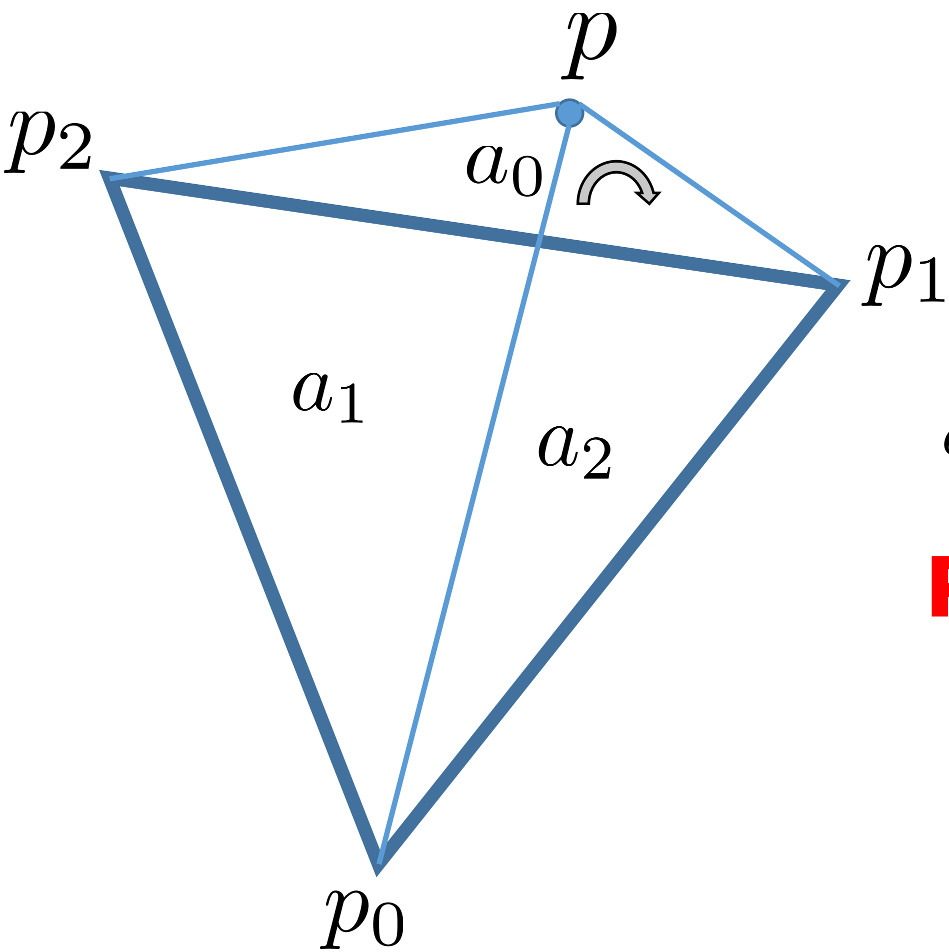
Barycentric Coordinates



Point inside

$$a_0, a_1, a_2 >= 0$$

Barycentric Coordinates

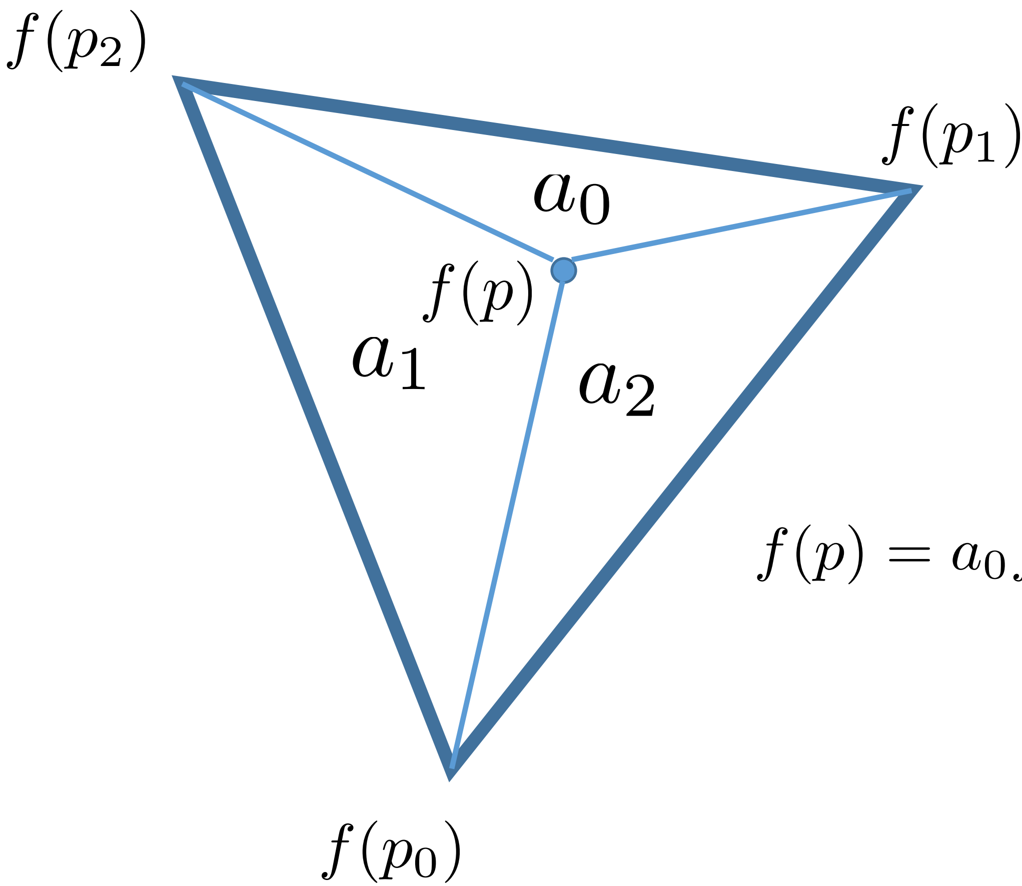


$$a_0 = \frac{Area(p, p_1, p_2)}{A}$$

Point outside

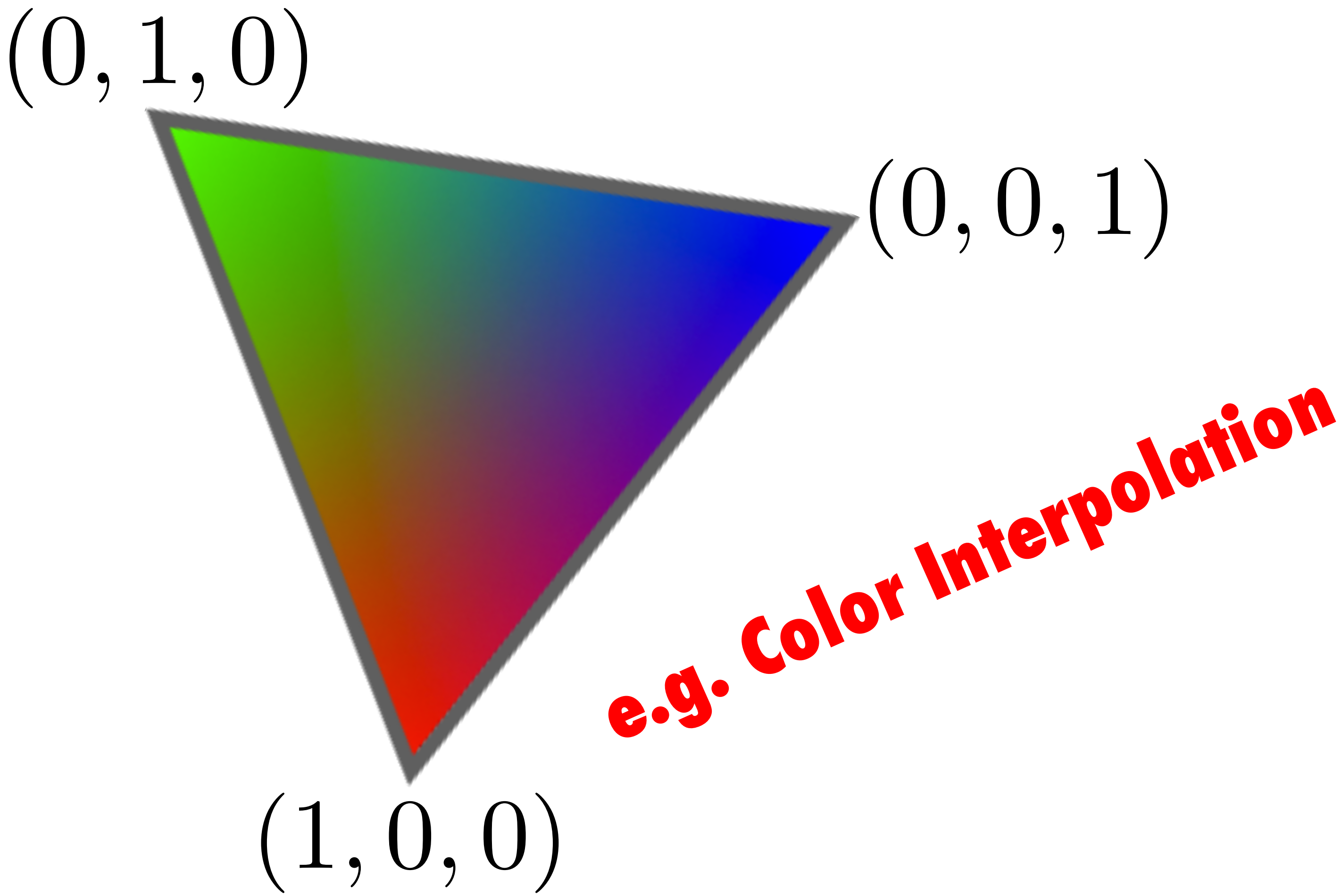
$$\exists i : a_i < 0$$

Barycentric Interpolation

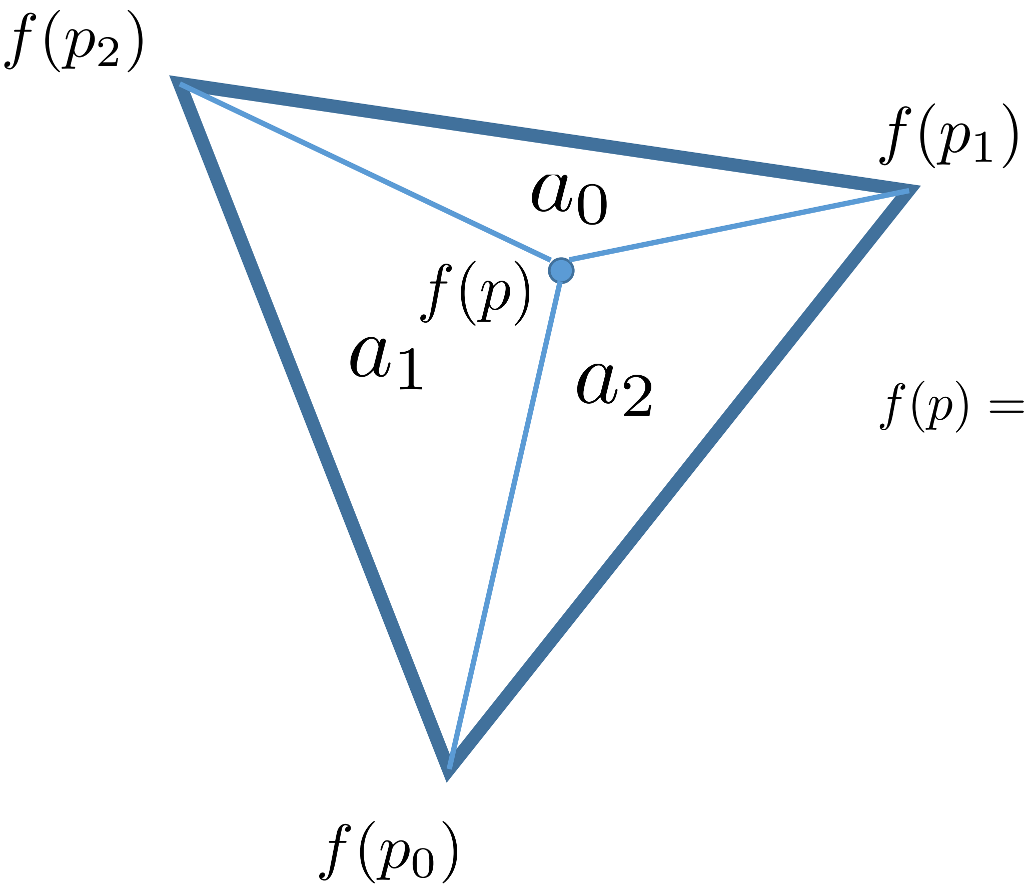


$$f(p) = a_0 f(p_0) + a_1 f(p_1) + a_2 f(p_2)$$

Barycentric Interpolation



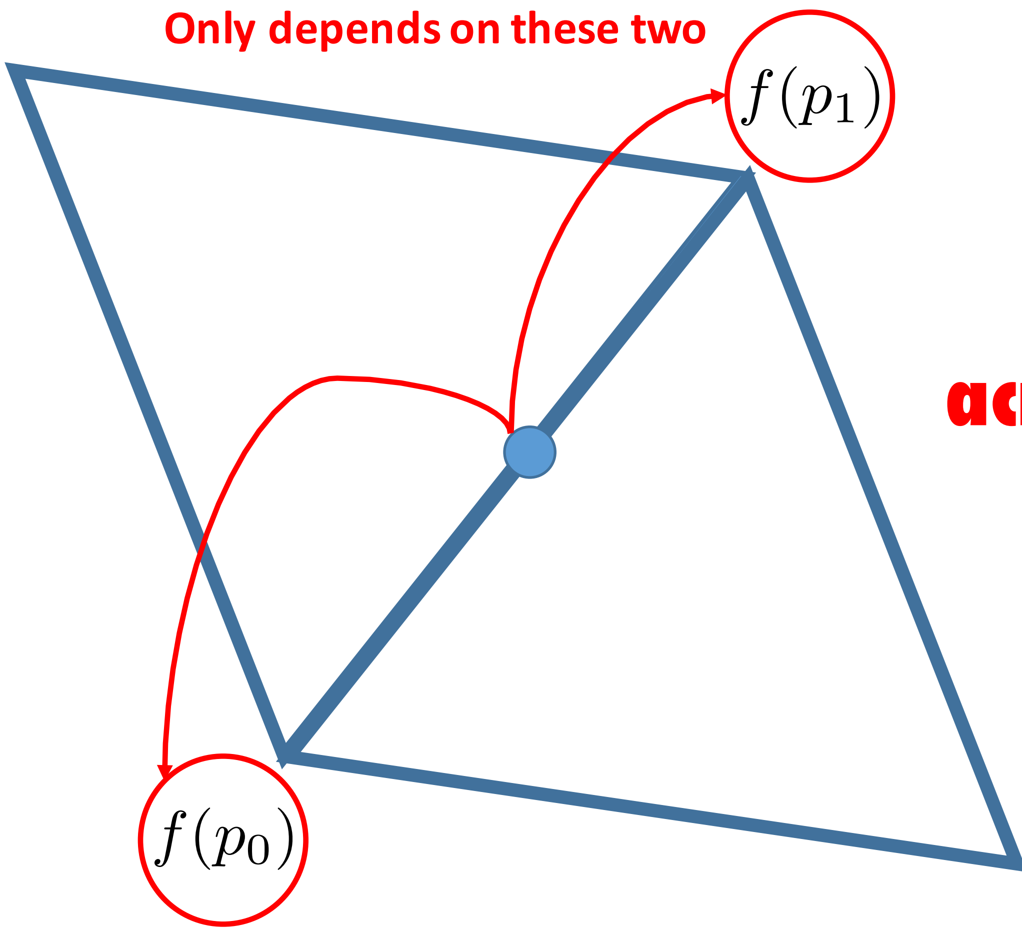
Barycentric Interpolation



$$f(p) = a_0 f(p_0) + a_1 f(p_1) + a_2 f(p_2)$$

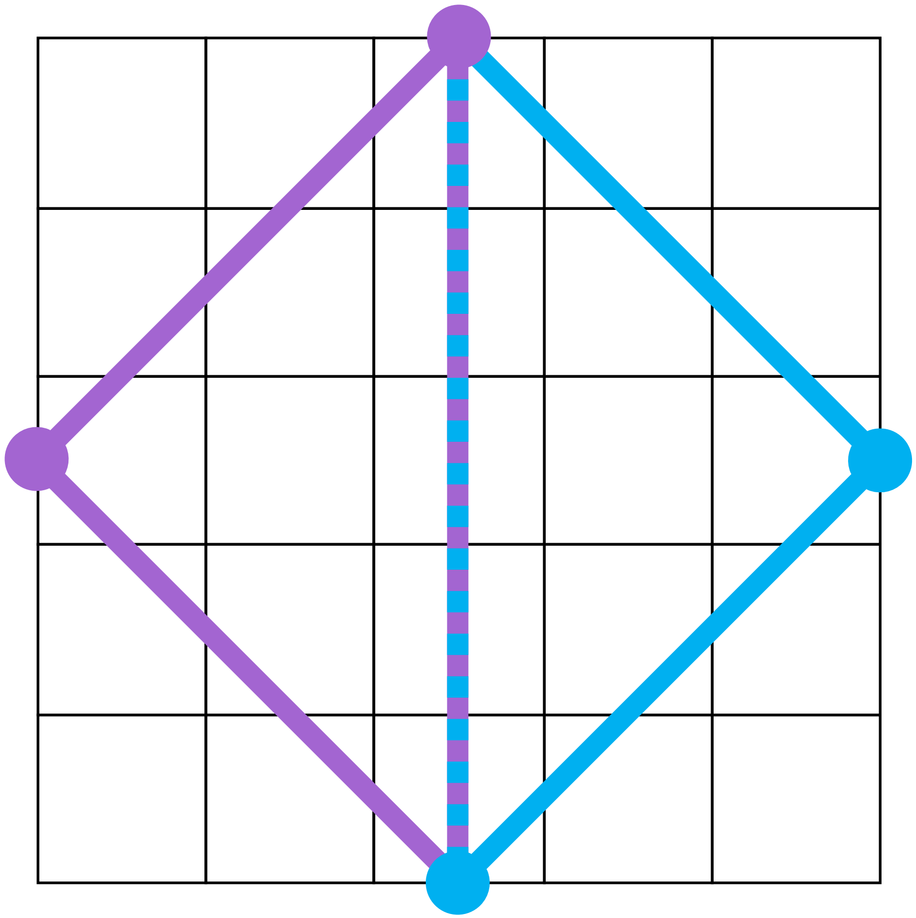
Linear

Barycentric Interpolation



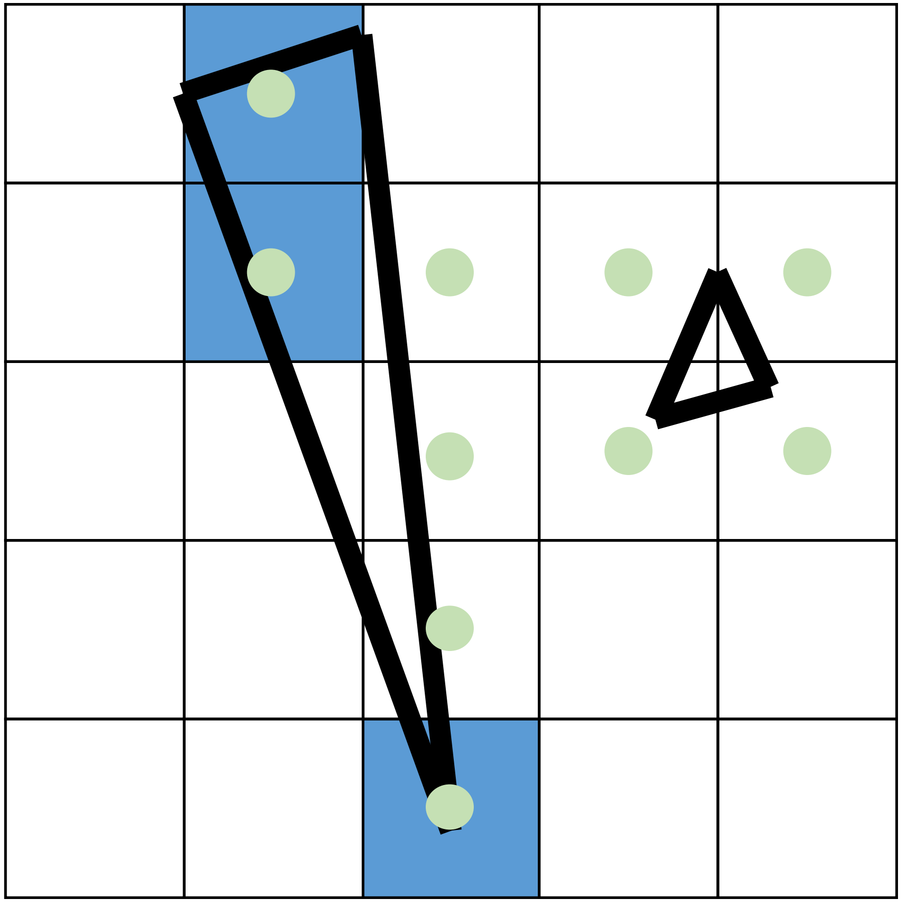
Continuous across boundaries

Issues

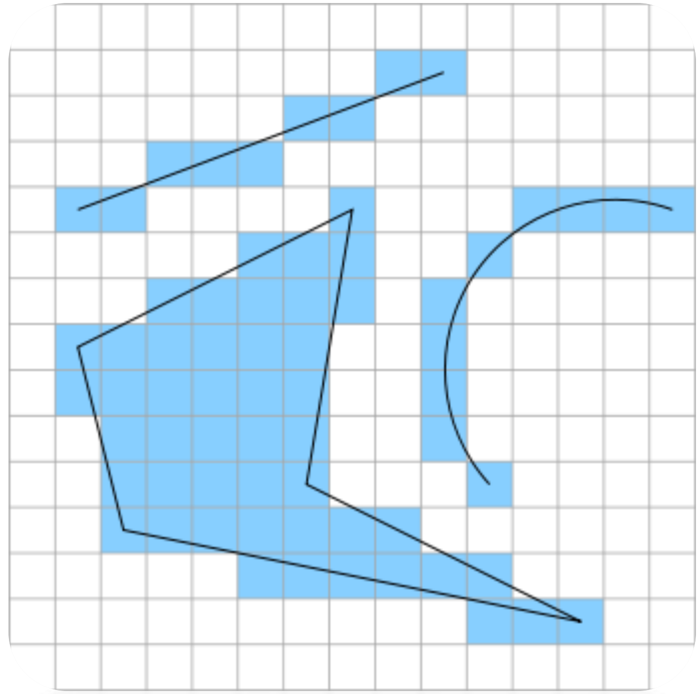


Adjacency and Singularity

More Issues



Small triangle and slivers



Rasterization

CS 148: Summer 2016
Introduction of Graphics and Imaging
Zahid Hossain

http://iloveshaders.blogspot.com/2011/05/how-rasterization-process-works.html