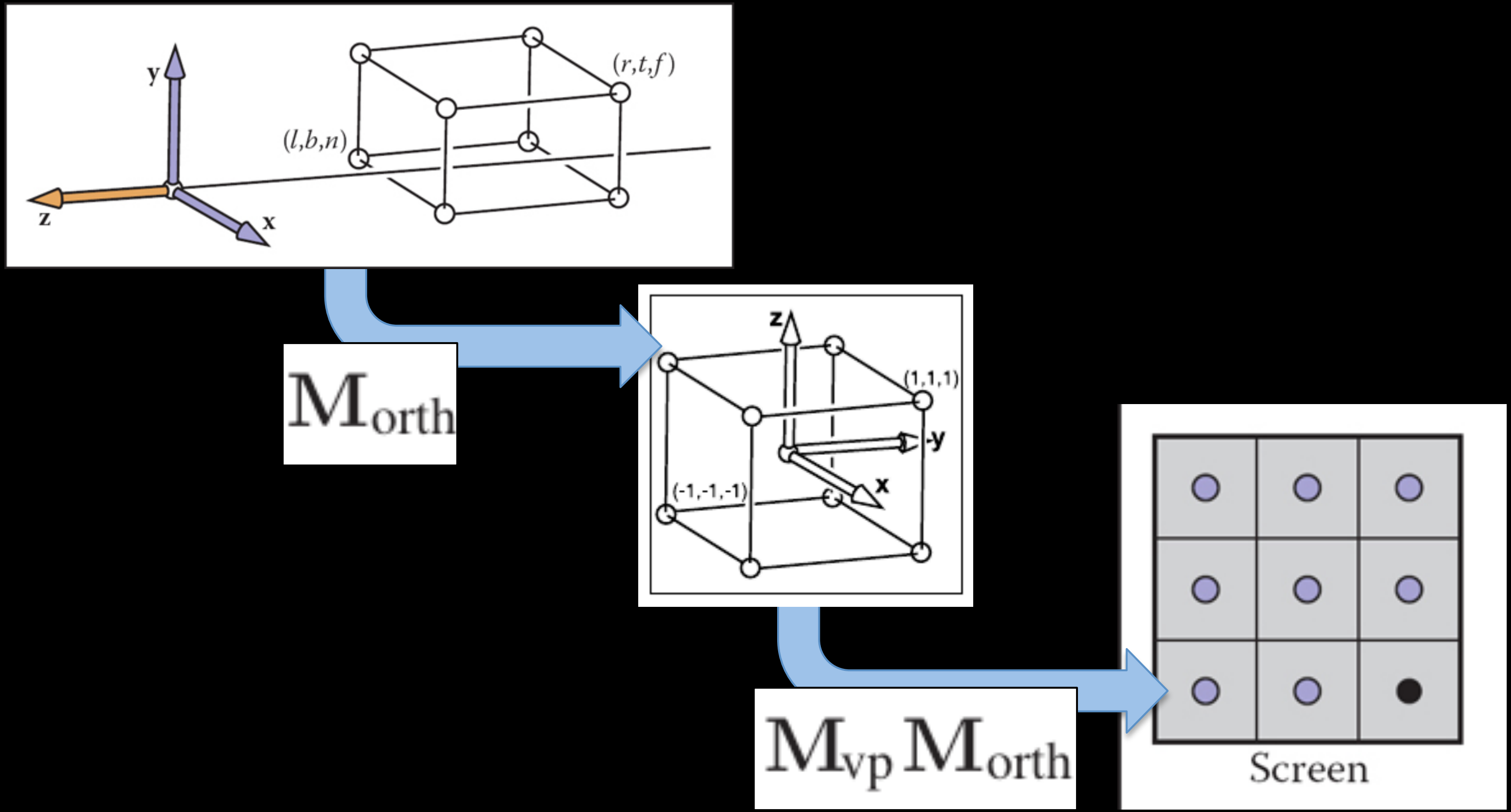
Rasterization

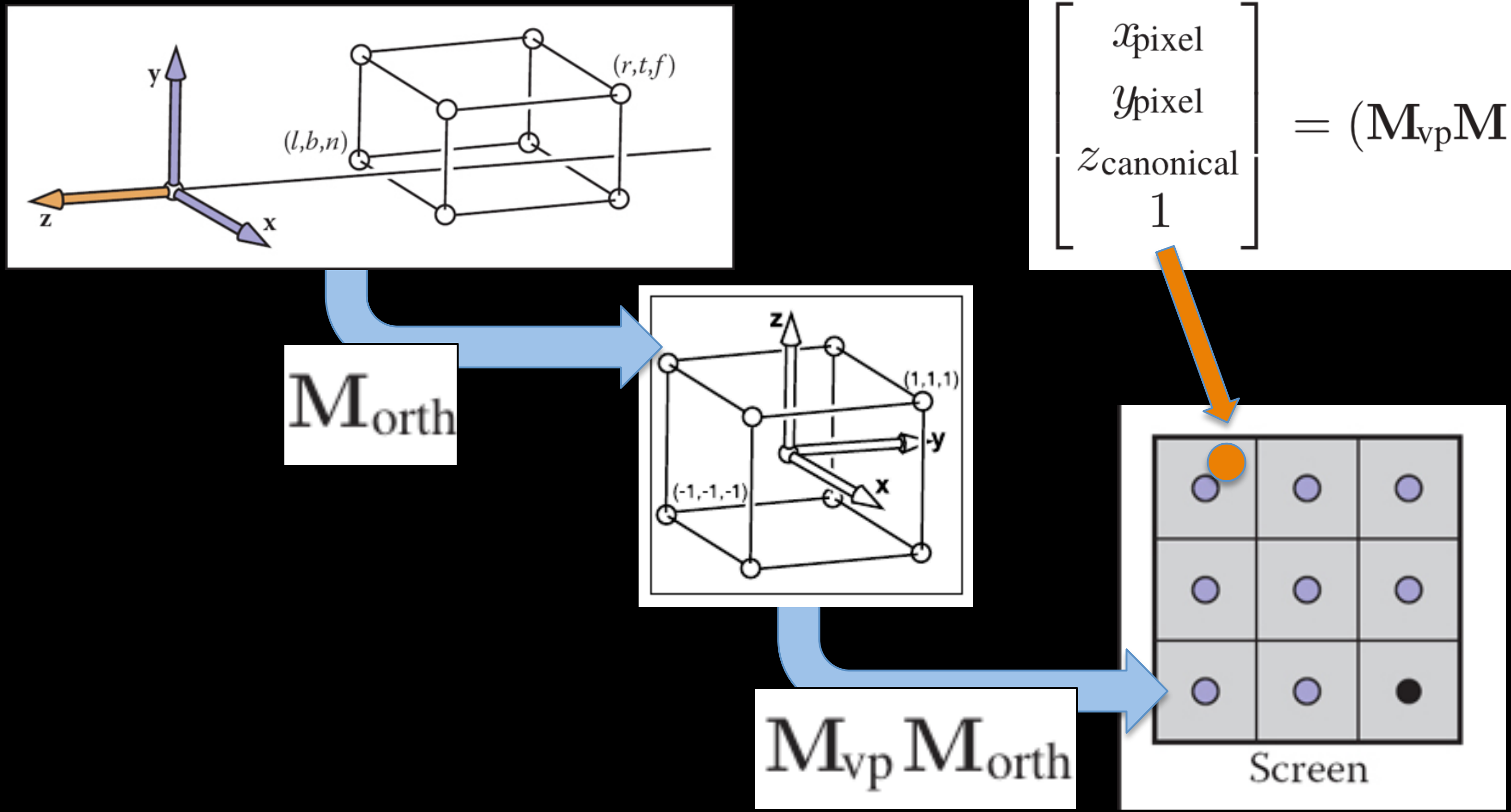
CPSC 453

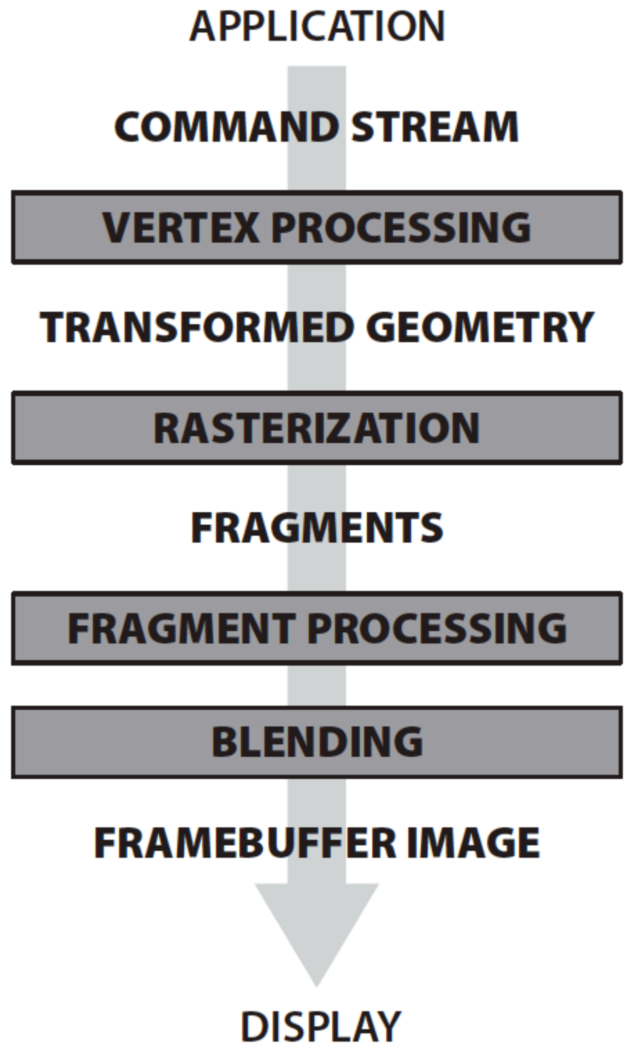
Transform to 2D

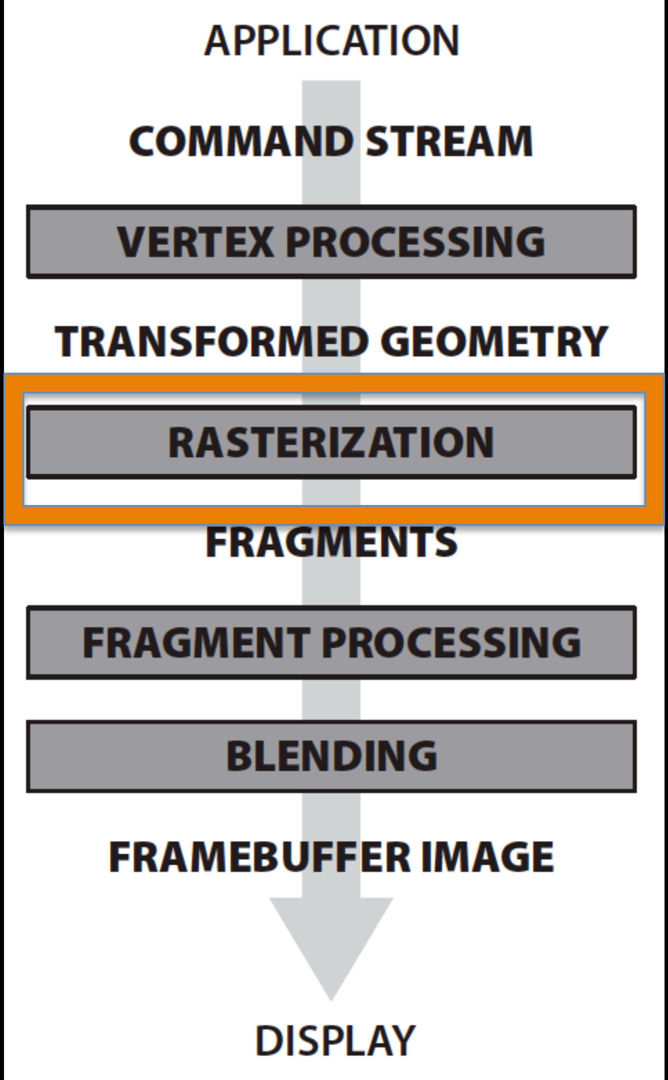


Transform to 2D

$$\begin{bmatrix} x_{\text{pixel}} \\ y_{\text{pixel}} \\ z_{\text{canonical}} \\ 1 \end{bmatrix} = (\mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$




APPLICATION
COMMAND STREAM
VERTEX PROCESSING
TRANSFORMED GEOMETRY
RASTERIZATION
FRAGMENTS
FRAGMENT PROCESSING
BLENDING
FRAMEBUFFER IMAGE
DISPLAY

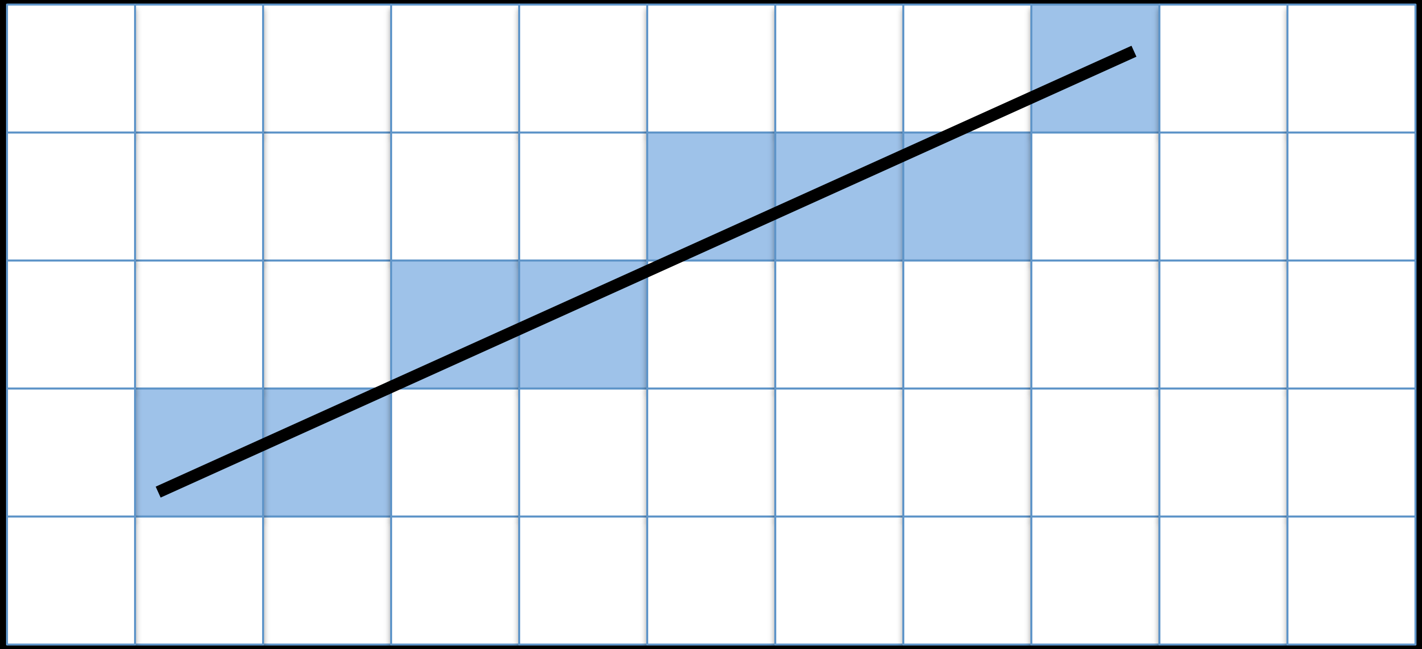


We want:

- *Fast*
- *Efficient*
- *Simple*

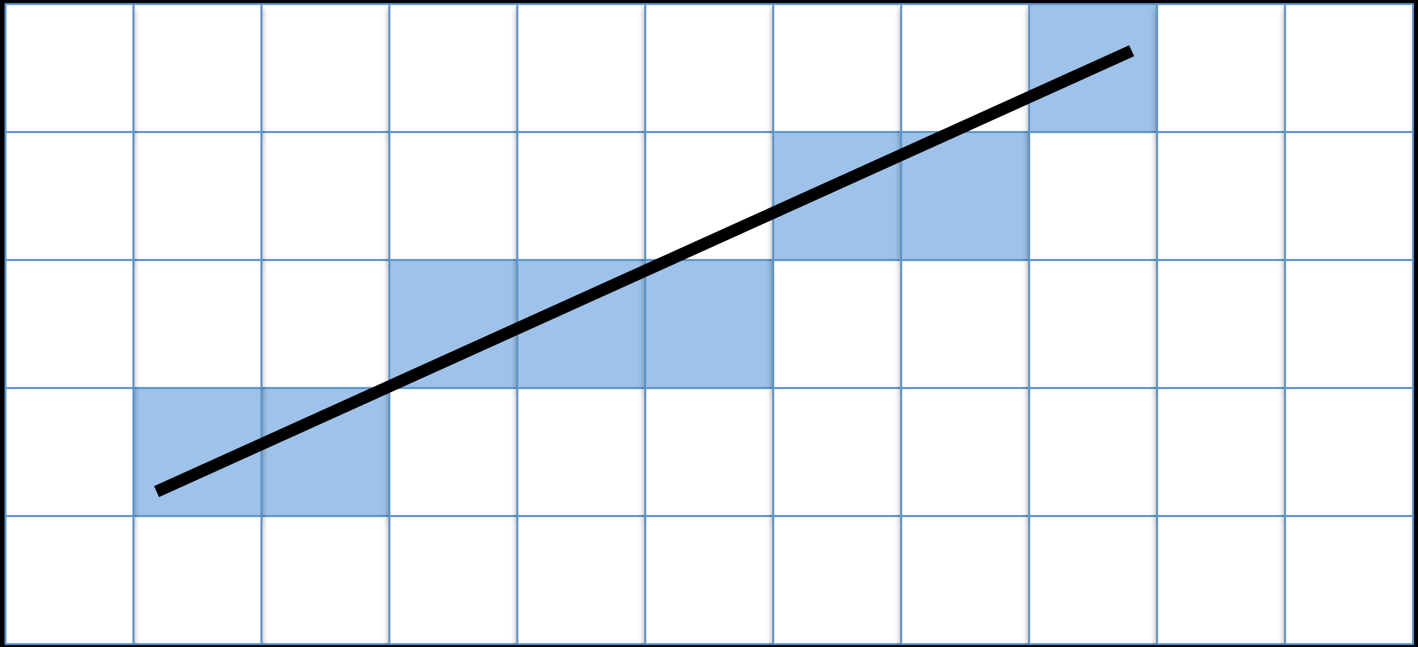
Rasterization *aka Scanline renders*

- Finding all pixels in an image occupied by a geometric primitive



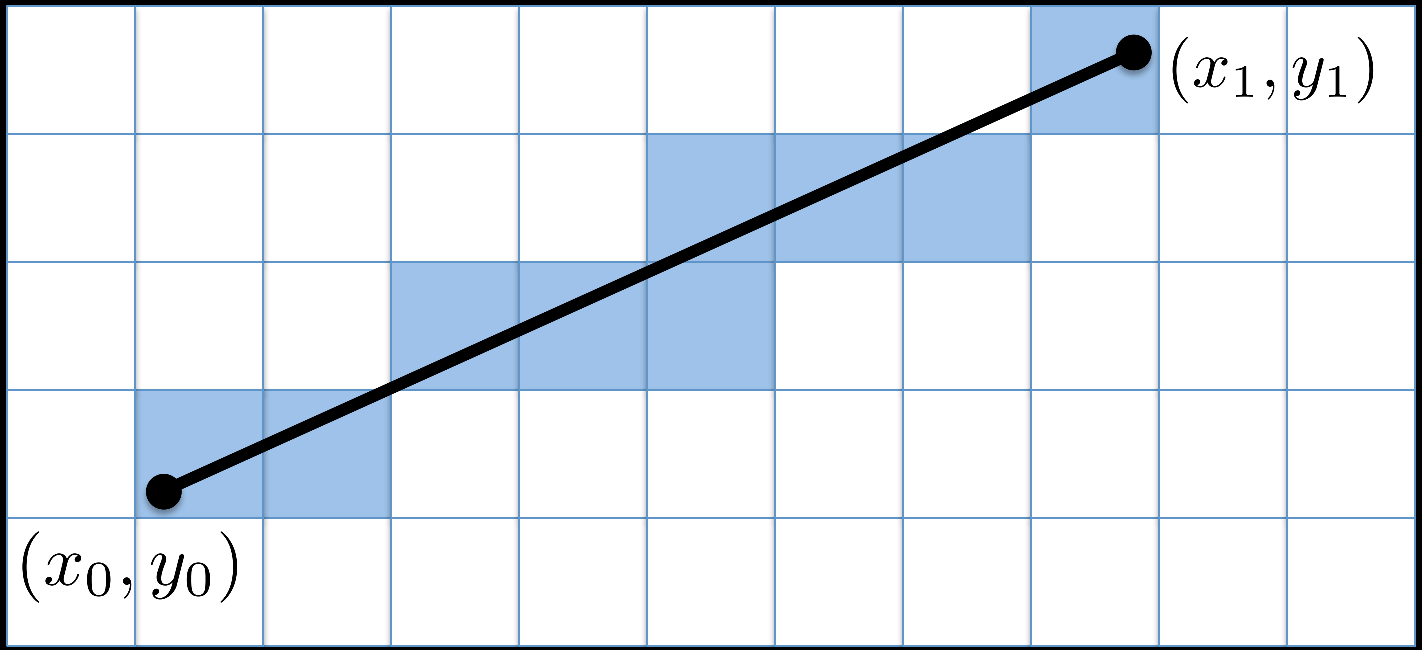
Rasterization

- Finding all pixels in an image occupied by a geometric primitive

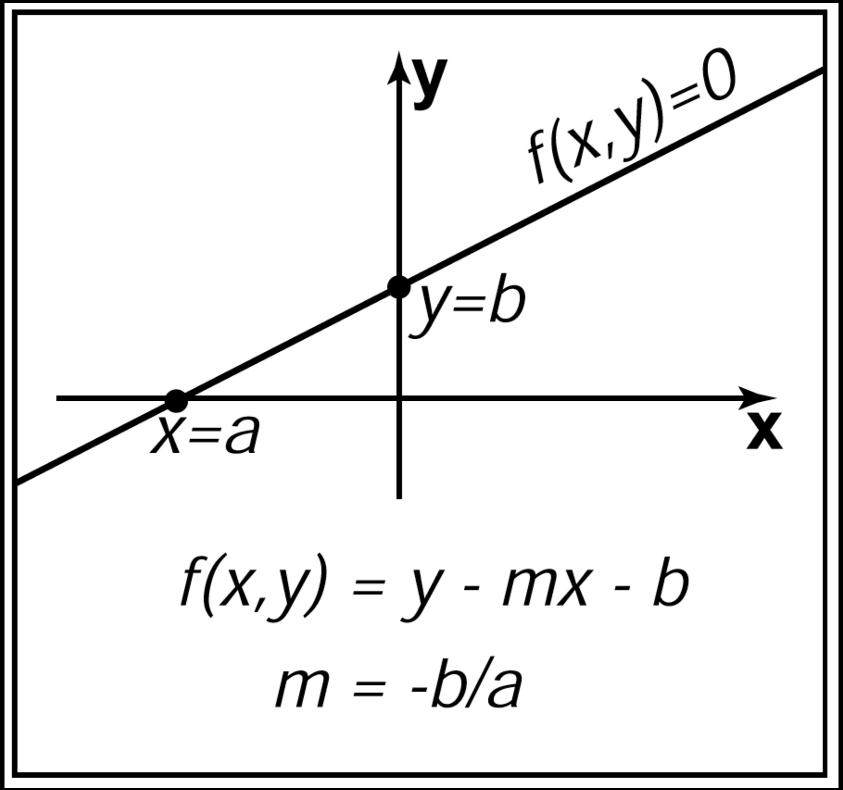


Rasterization

- Finding all pixels in an image occupied by a geometric primitive

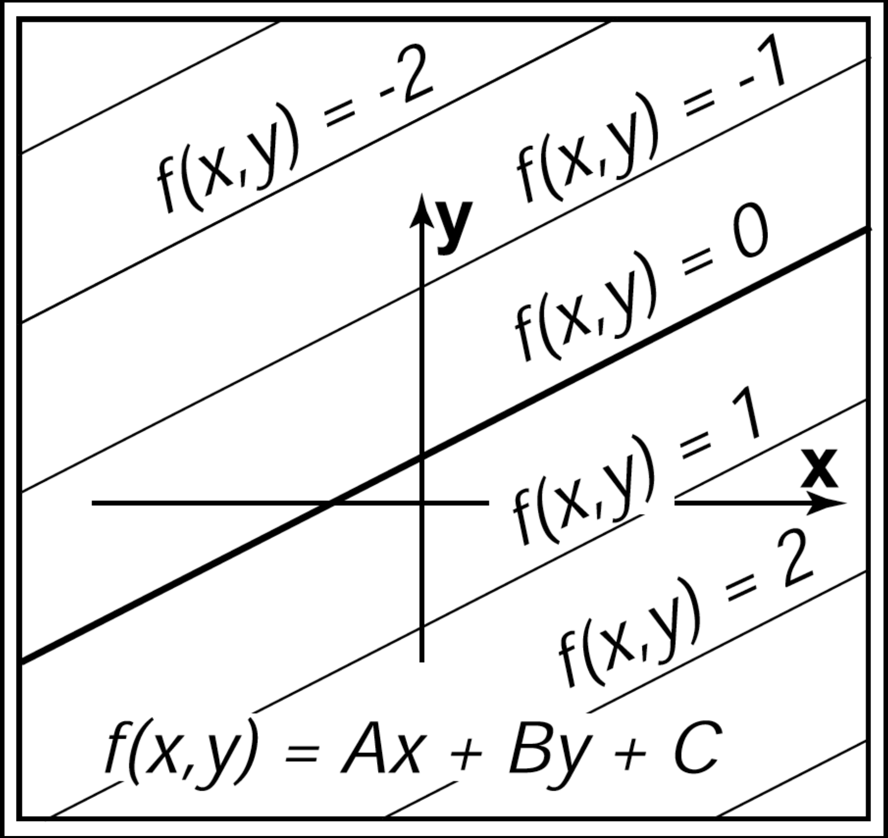


Implicit Line equation

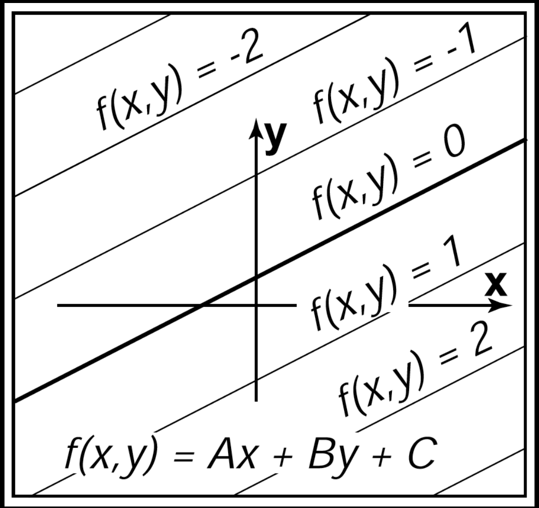


Hard to represent line *x=0*

Implicit Line equation



Implicit Line equation

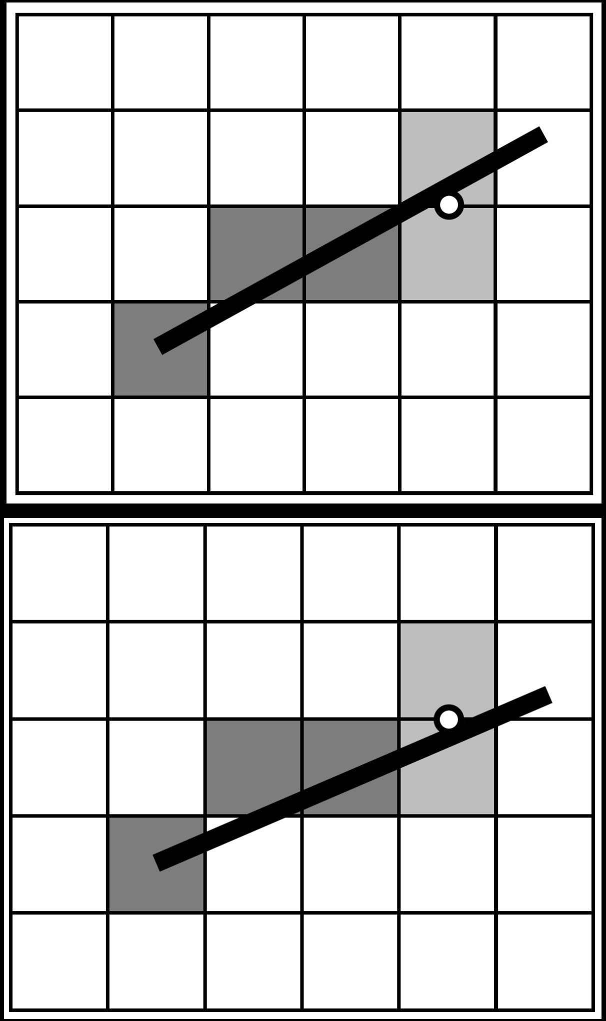


$$f(x,y) \equiv \underbrace{(y_0 - y_1)}_{A}x + \underbrace{(x_1 - x_0)}_{B}y + \underbrace{x_0y_1 - x_1y_0}_{C} = 0$$

Midpoint algorithm $m \in (0, 1]$

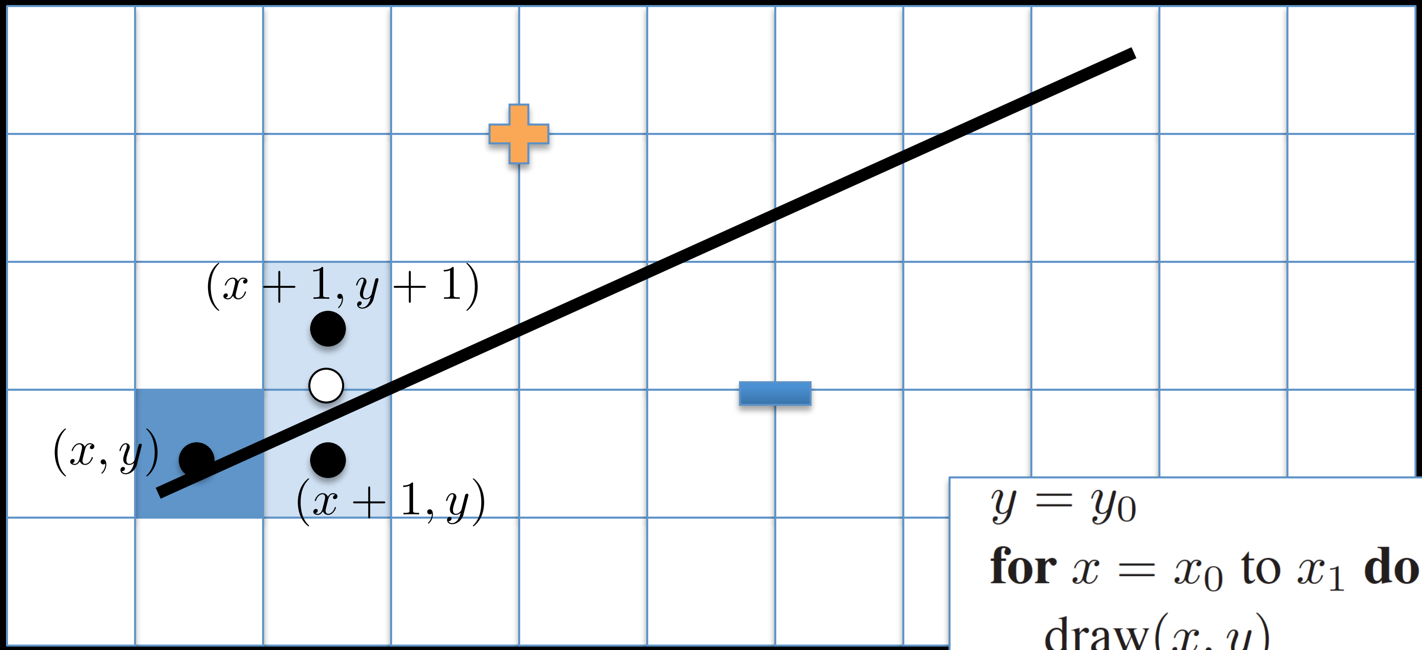
$y = y_0$
for $x = x_0$ to x_1 **do**
  draw(x, y)
  if (some condition) **then**
    $y = y + 1$

- *"thinnest line" (1 pixel)*
- *no gaps*



Line drawing: *midpoint algorithm*

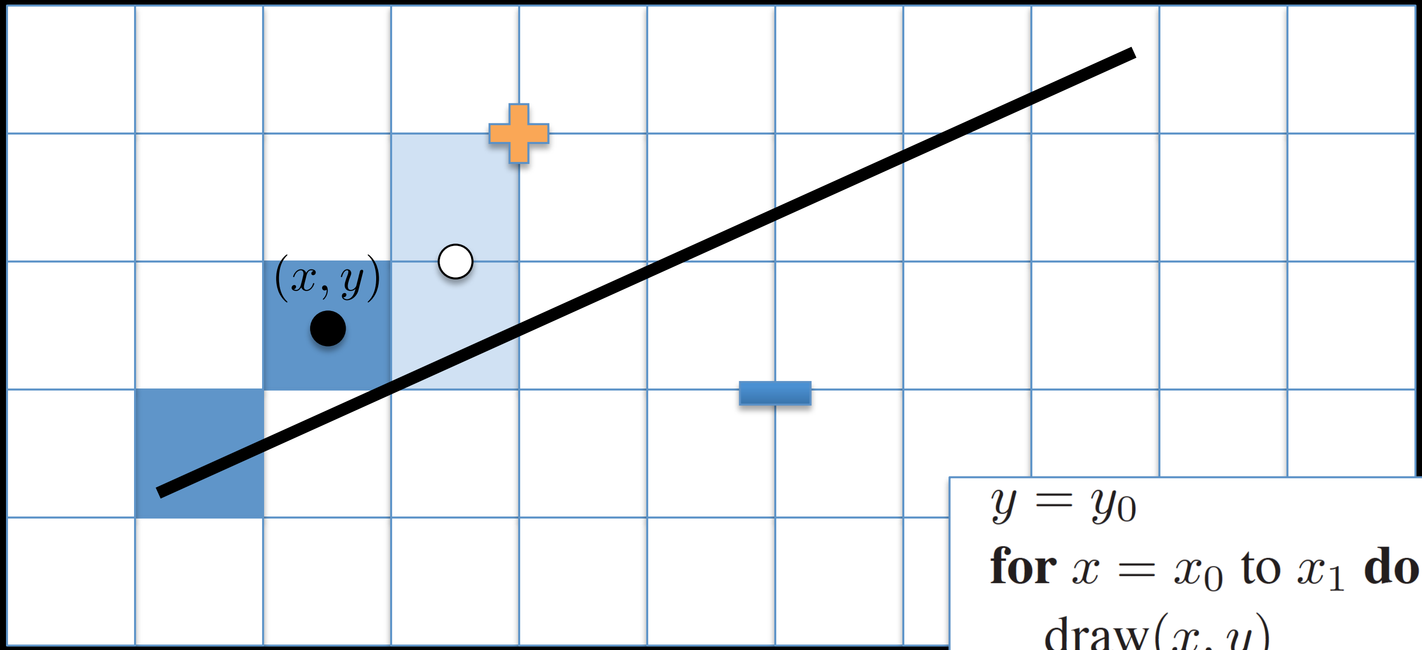
- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$
for $x = x_0$ to x_1 **do**
  draw(x, y)
  if $f(x+1, y+0.5) < 0$ **then**
    $y = y + 1$

Line drawing: *midpoint algorithm*

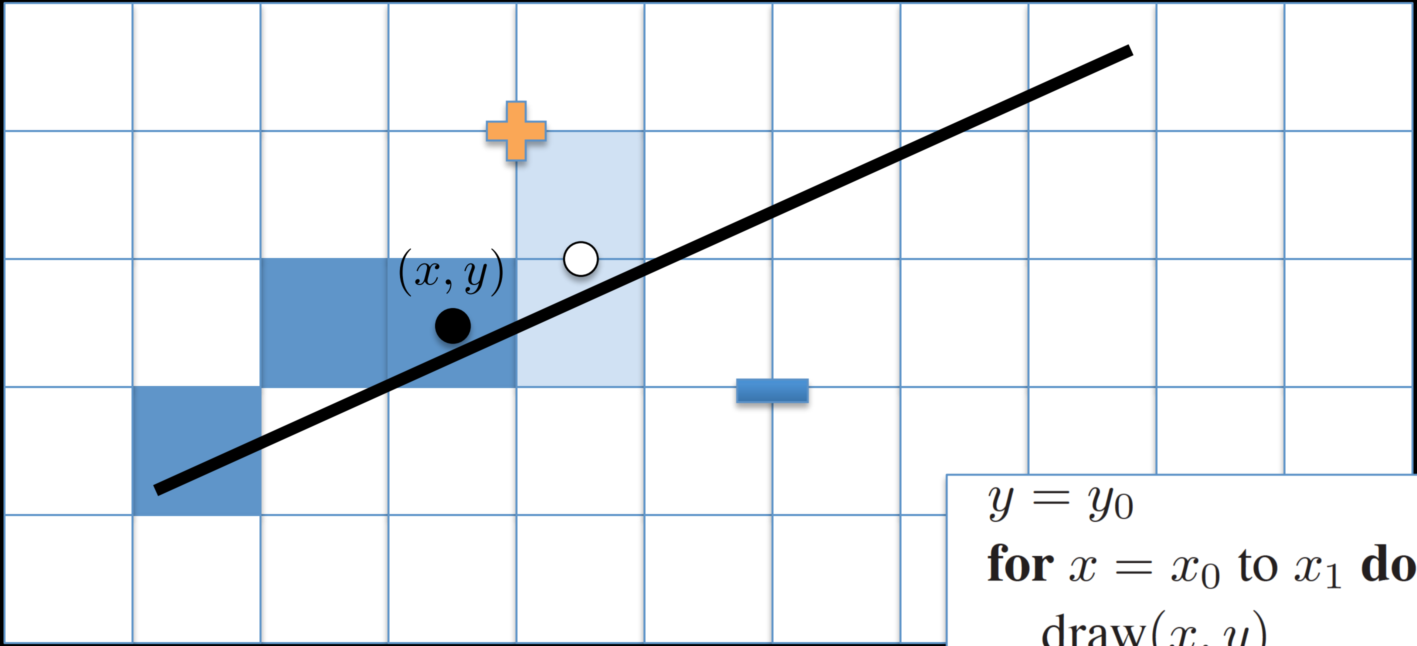
- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$
for $x = x_0$ to x_1 **do**
  draw(x, y)
  if $f(x + 1, y + 0.5) < 0$ **then**
    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$

for $x = x_0$ to x_1 **do**

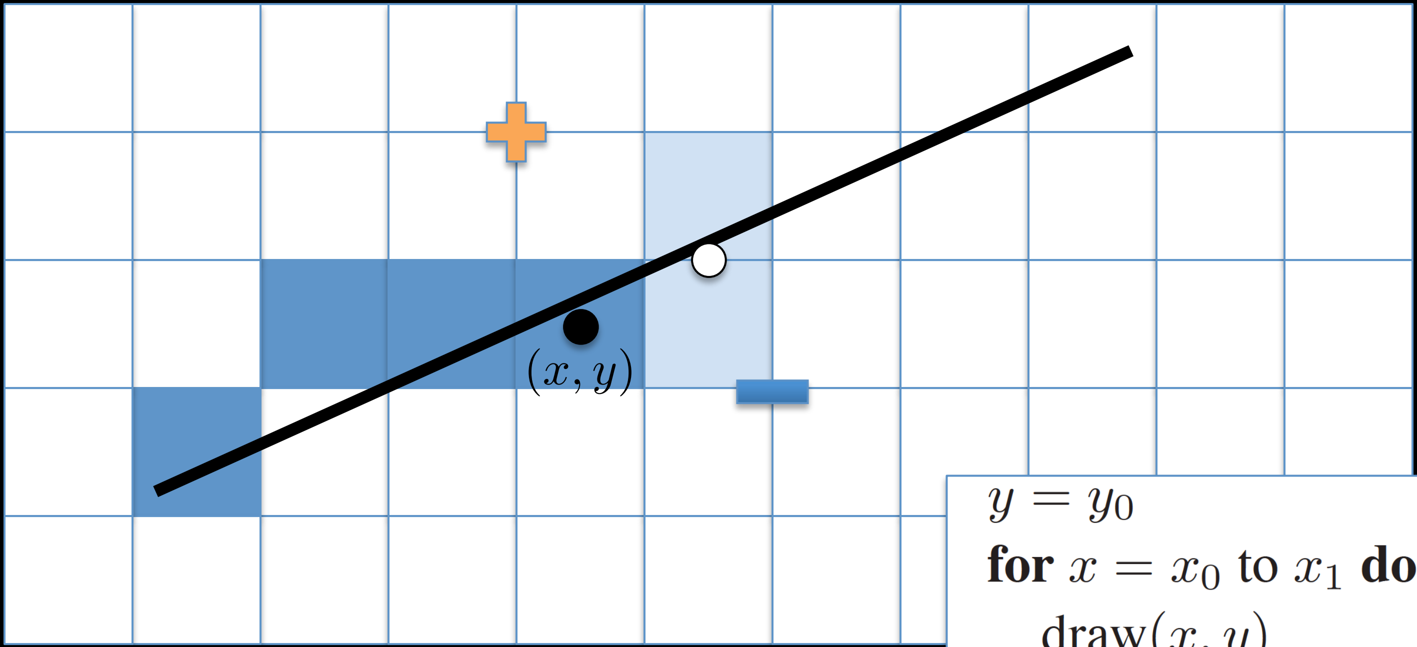
  draw(x, y)

  if $f(x+1, y+0.5) < 0$ **then**

    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$

for $x = x_0$ to x_1 **do**

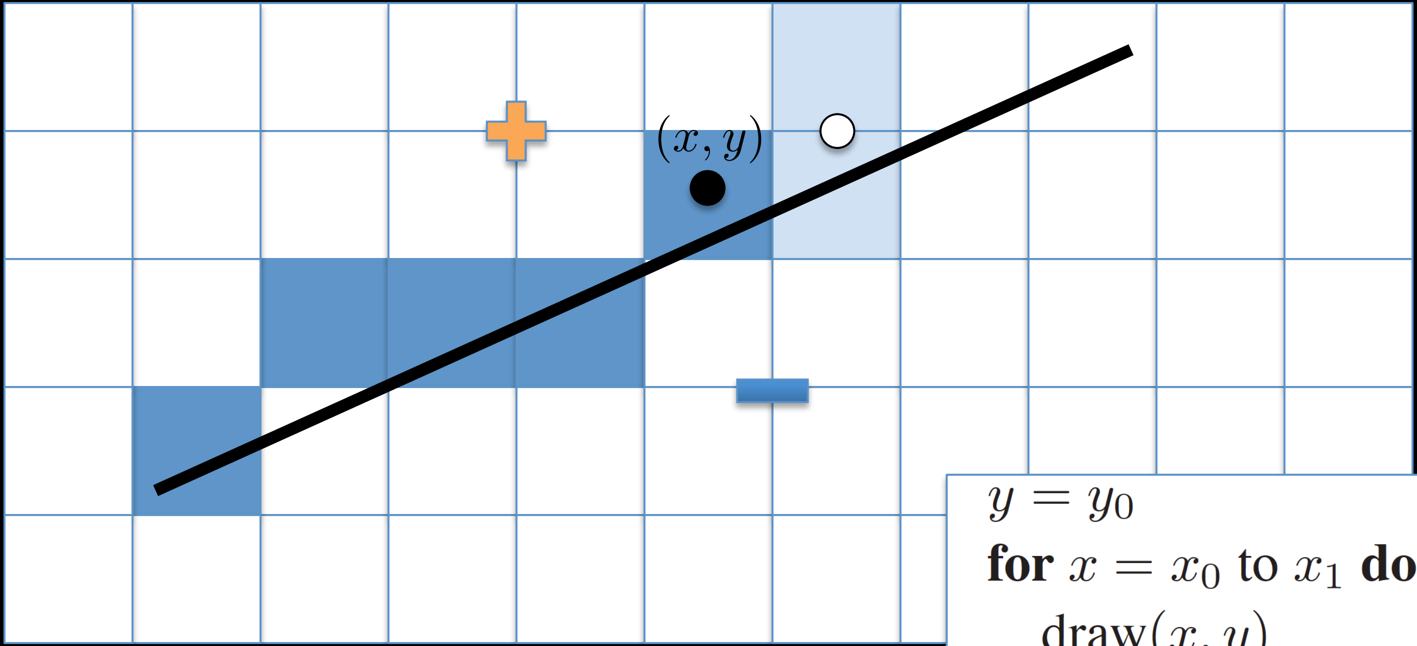
  draw(x, y)

  if $f(x + 1, y + 0.5) < 0$ **then**

    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$

for $x = x_0$ to x_1 **do**

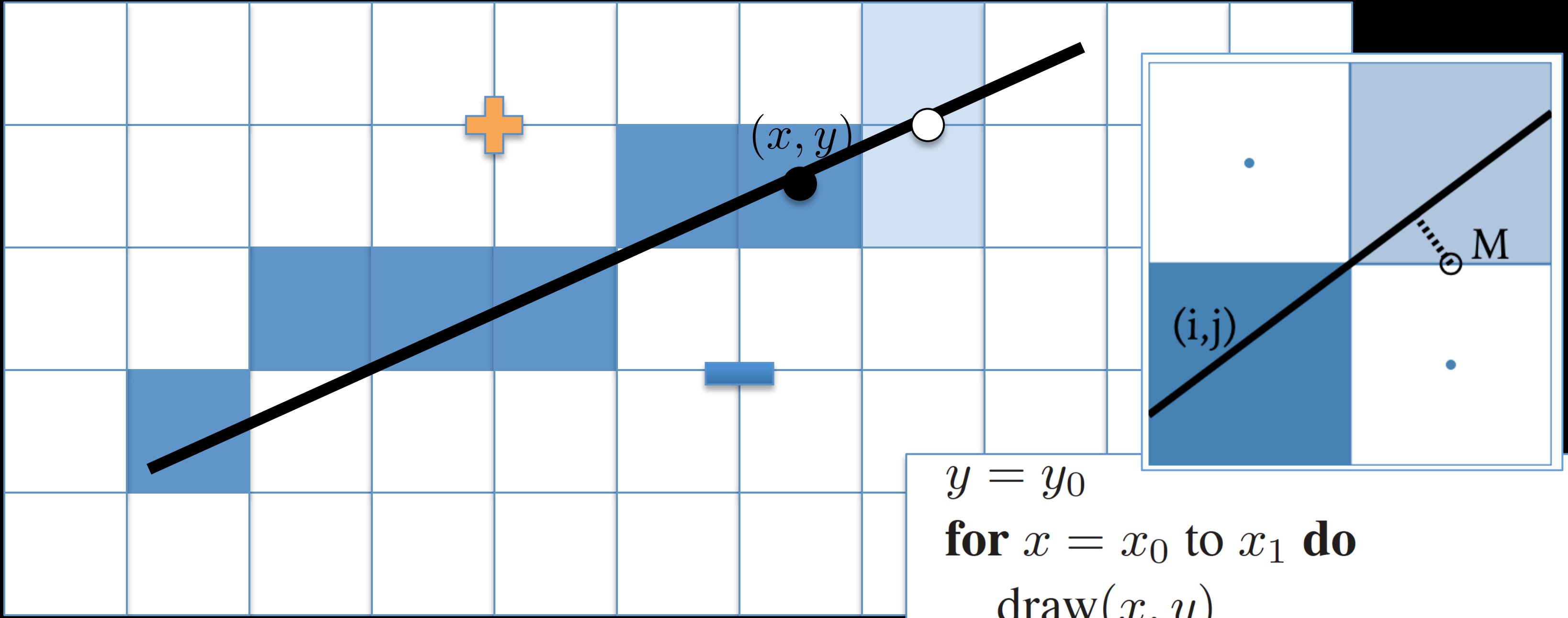
  draw(x, y)

  if $f(x+1, y+0.5) < 0$ **then**

    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$

for $x = x_0$ to x_1 **do**

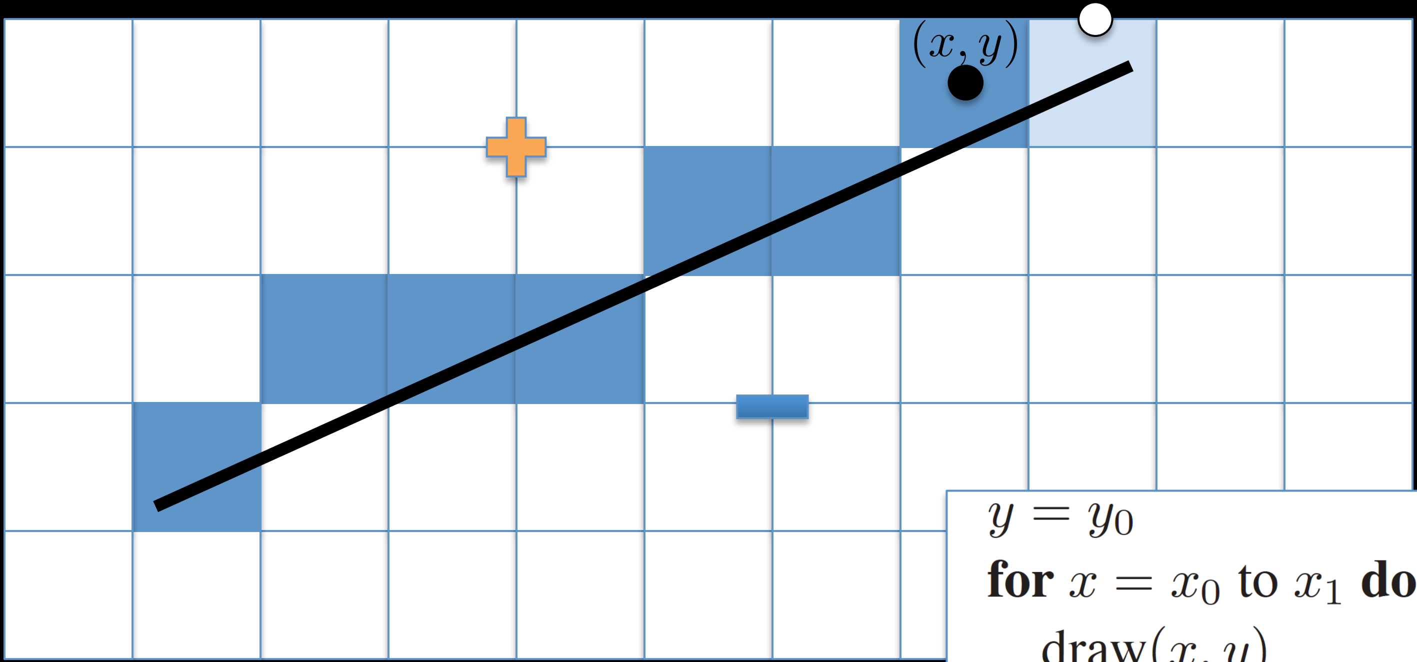
  $\text{draw}(x, y)$

  if $f(x+1, y+0.5) < 0$ **then**

    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



$y = y_0$

for $x = x_0$ to x_1 **do**

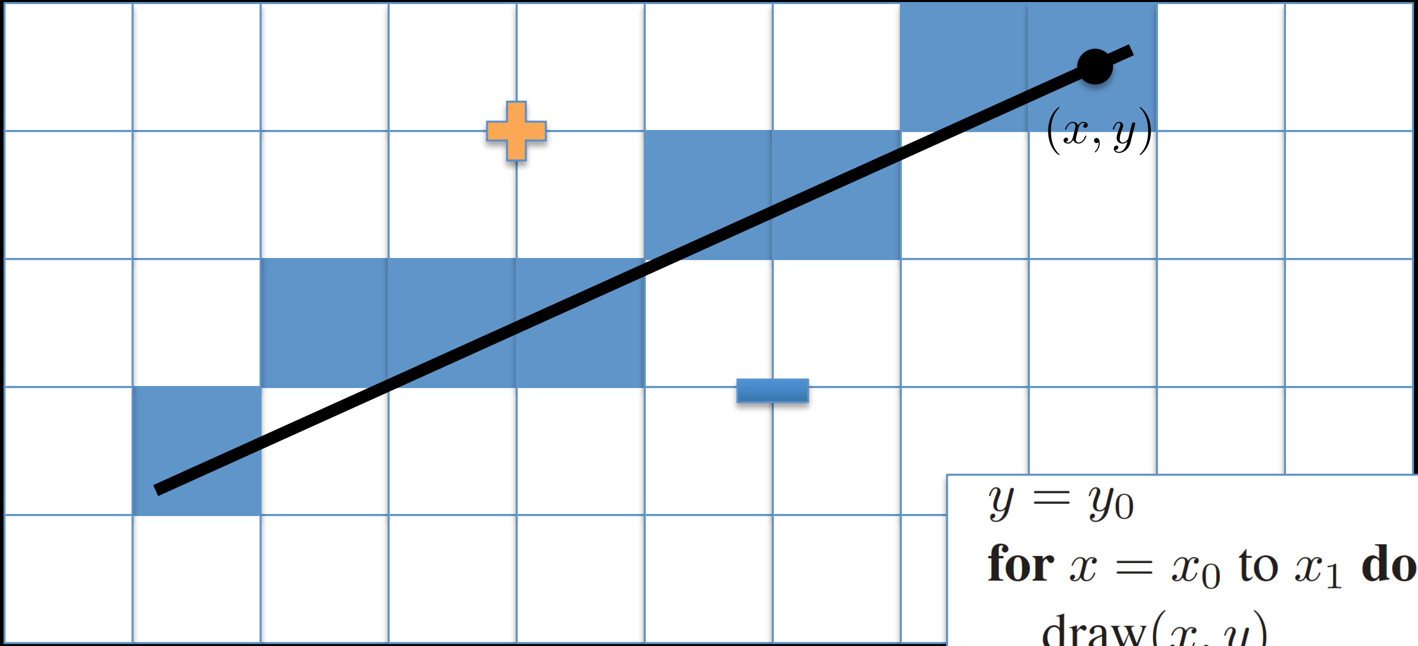
  draw(x, y)

  if $f(x+1, y+0.5) < 0$ **then**

    $y = y + 1$

Line drawing: *midpoint algorithm*

- Finding all pixels in an image occupied by a geometric primitive



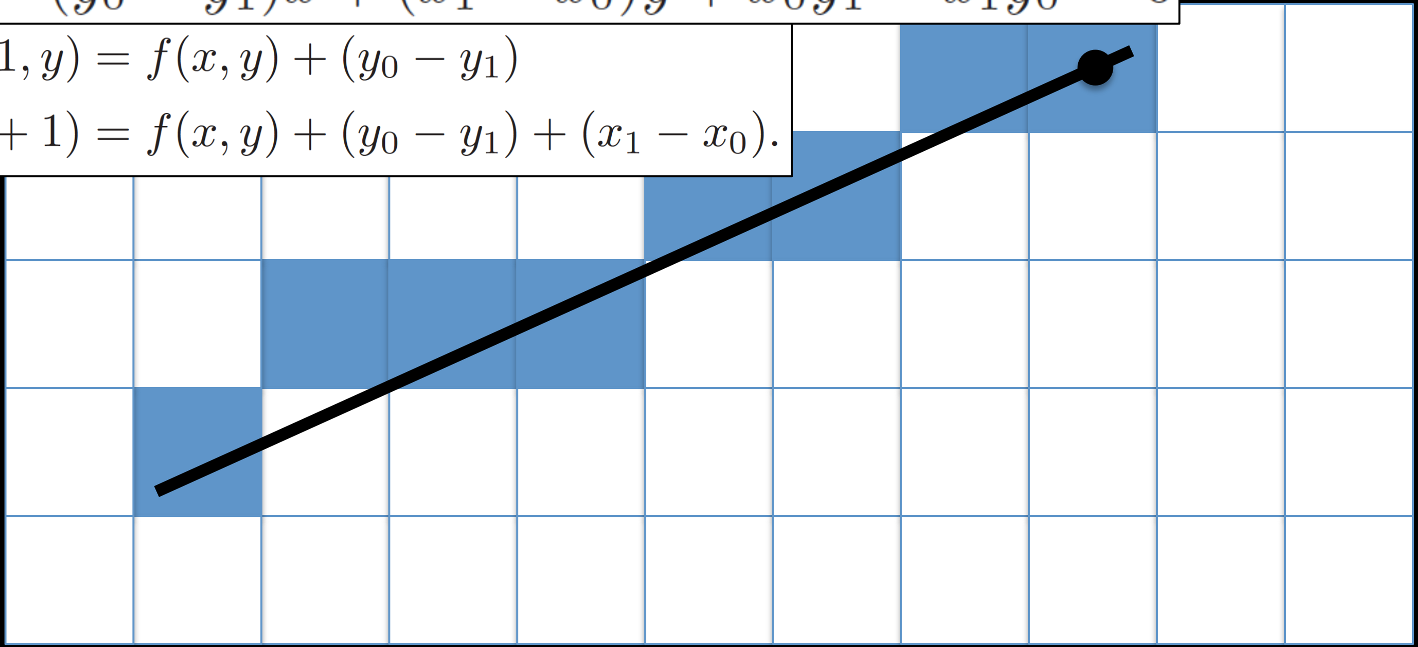
$y = y_0$
for $x = x_0$ to x_1 **do**
  draw(x, y)
  if $f(x+1, y+0.5) < 0$ **then**
    $y = y + 1$

Line drawing: *midpoint algorithm (incremental)*

$$f(x,y) \equiv (y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0 = 0$$

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

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

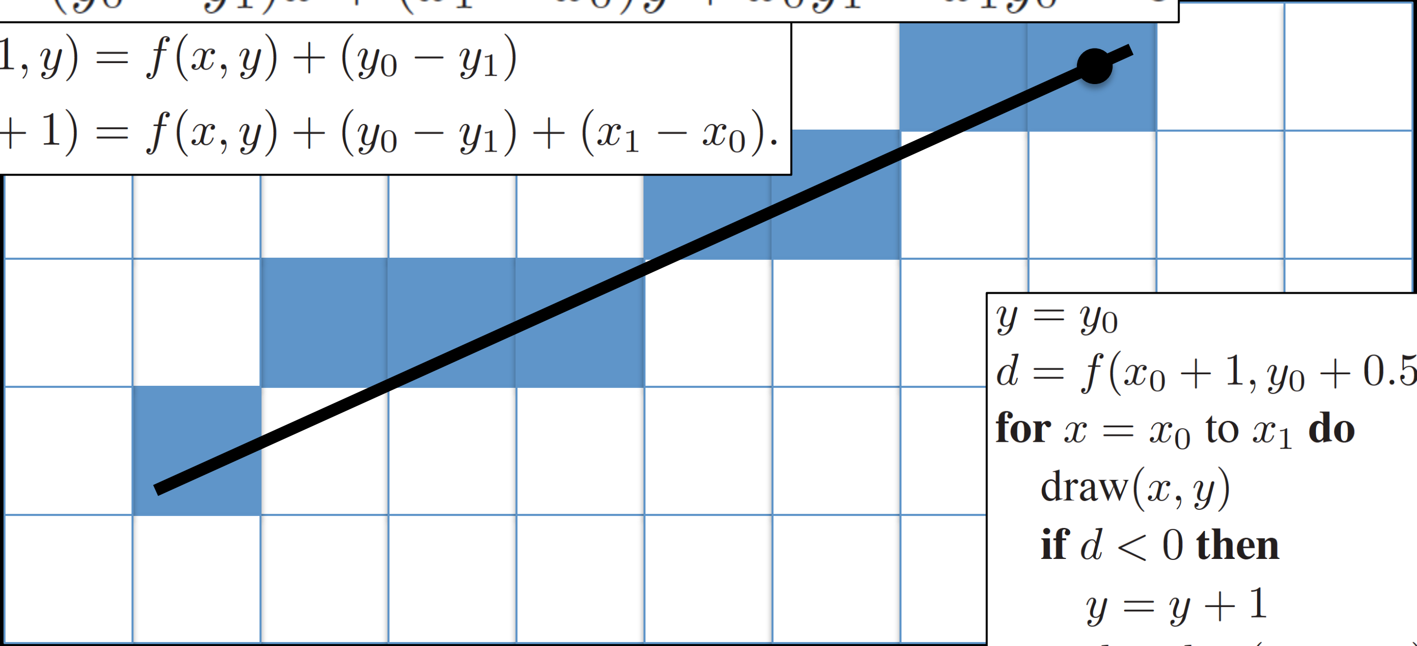


Line drawing: *midpoint algorithm (incremental)*

$$f(x,y) \equiv (y_0 - y_1)x + (x_1 - x_0)y + x_0 y_1 - x_1 y_0 = 0$$

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

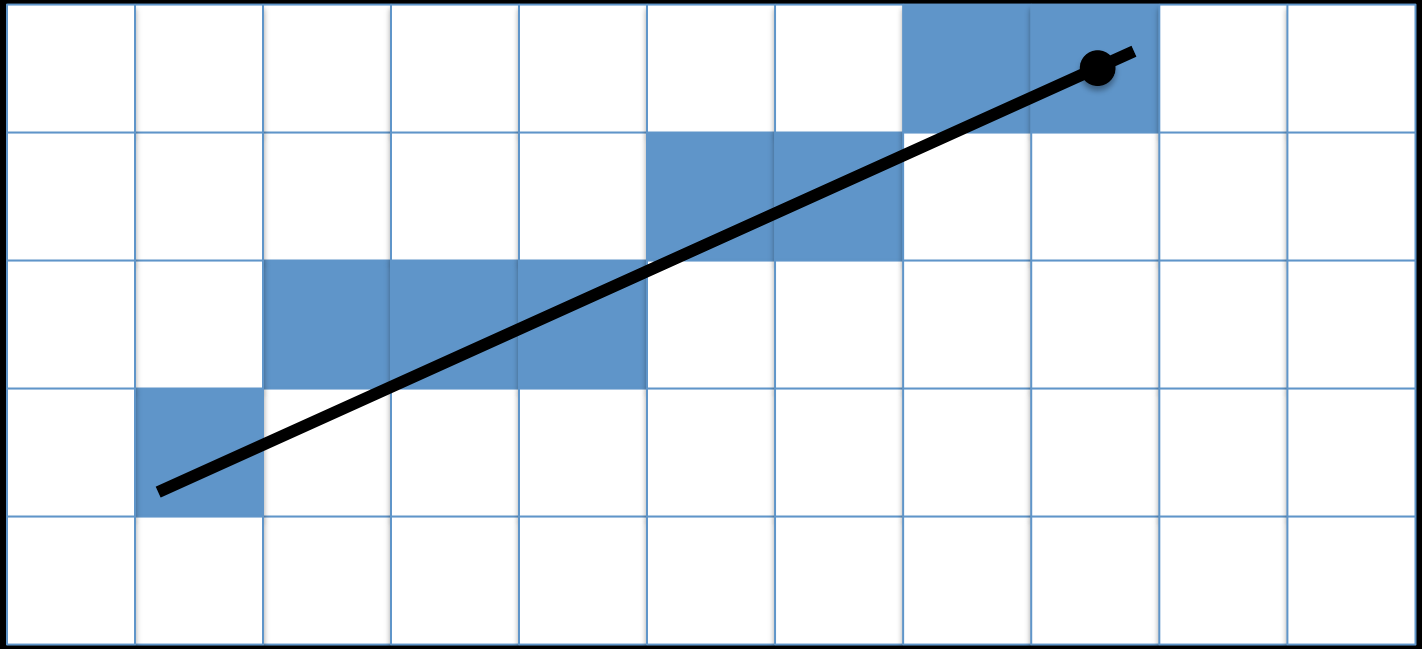
$$f(x+1, y+1) = f(x,y) + (y_0 - y_1) + (x_1 - x_0).$$



$y = y_0$
$d = f(x_0 + 1, y_0 + 0.5)$
for $x = x_0$ to x_1 **do**
  draw(x, y)
  if $d < 0$ **then**
    $y = y + 1$
    $d = d + (x_1 - x_0) + (y_0 - y_1)$
  else
    $d = d + (y_0 - y_1)$

Potential *numerical* issues?

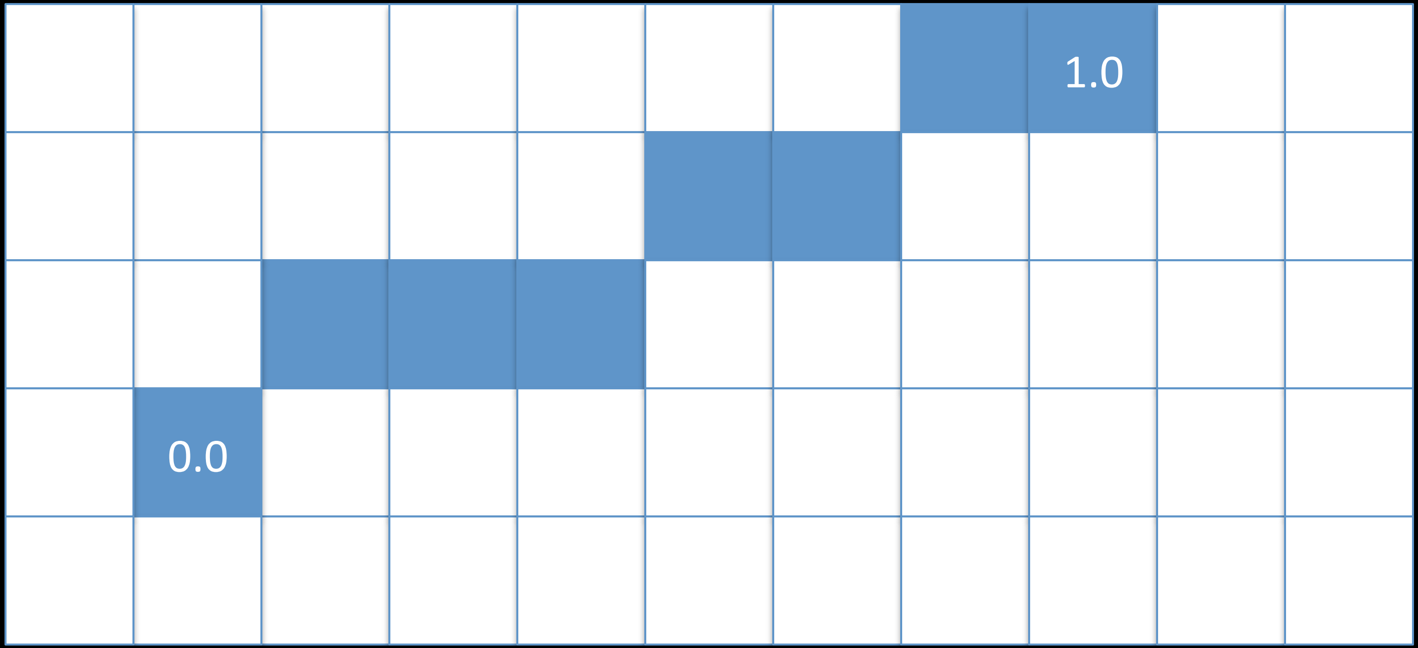
Line drawing: *midpoint algorithm*



Similar arguments for $m \notin (0, 1]$

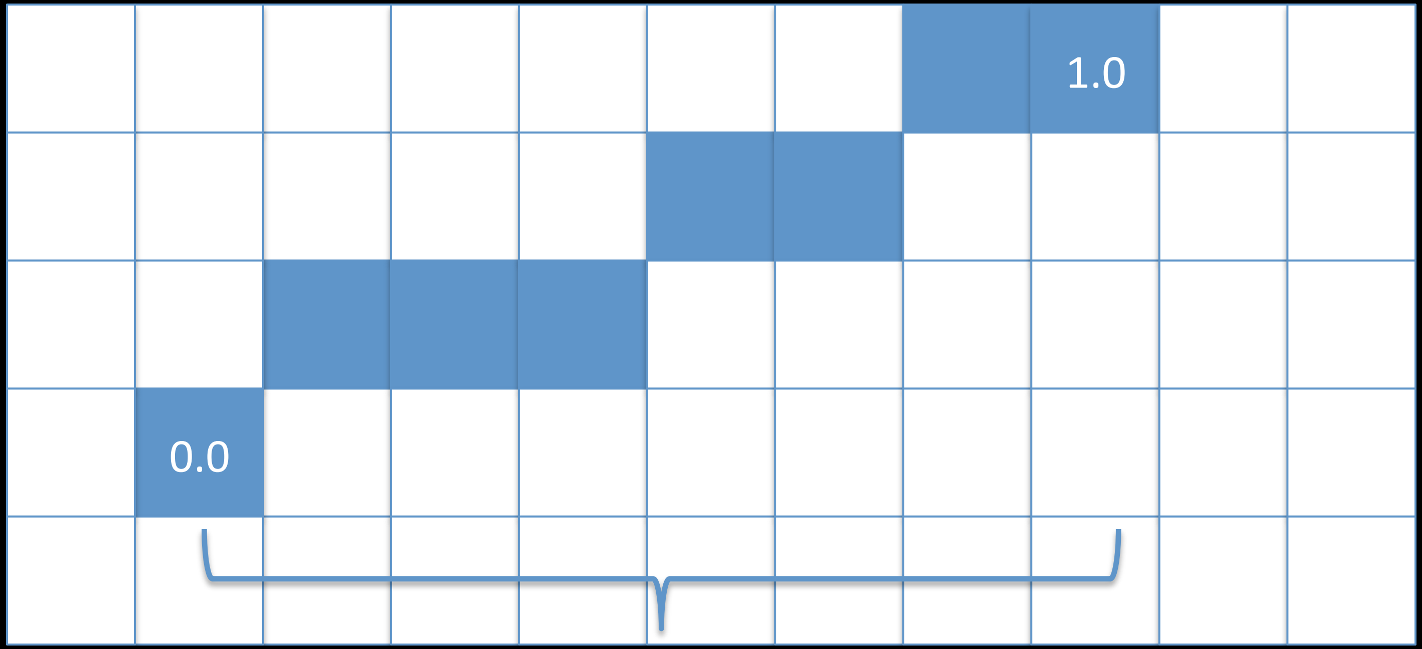
Interpolating values

- Finding all pixels in an image occupied by a geometric primitive



Interpolating values

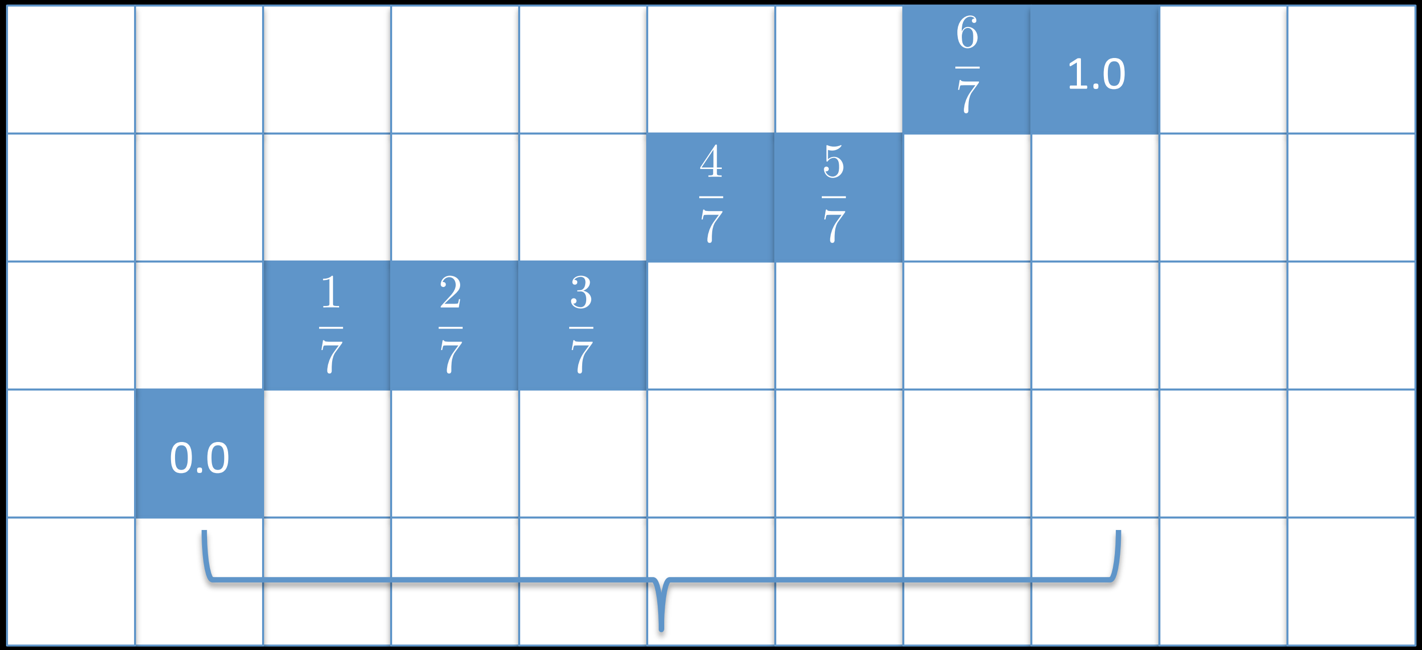
- Finding all pixels in an image occupied by a geometric primitive



7 steps – $\Delta = \frac{1}{7}$

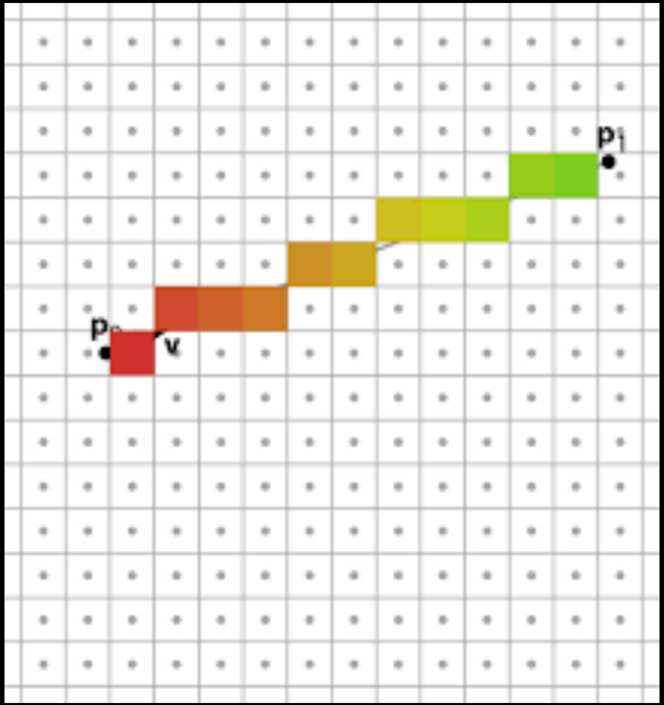
Interpolating values

- Finding all pixels in an image occupied by a geometric primitive



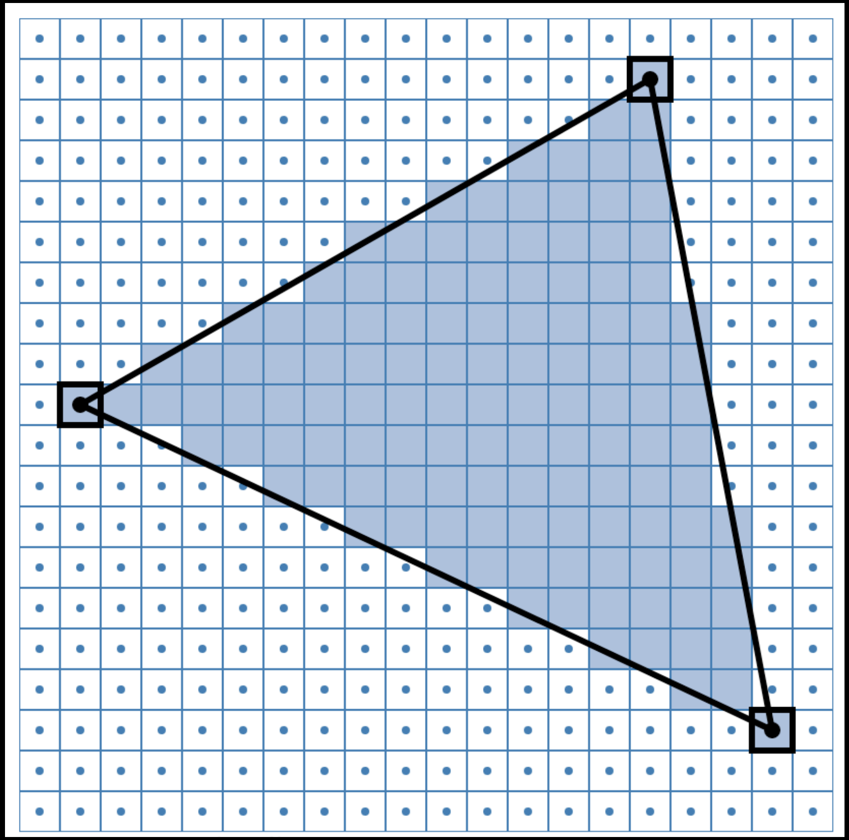
7 steps – $\Delta = \frac{1}{7}$

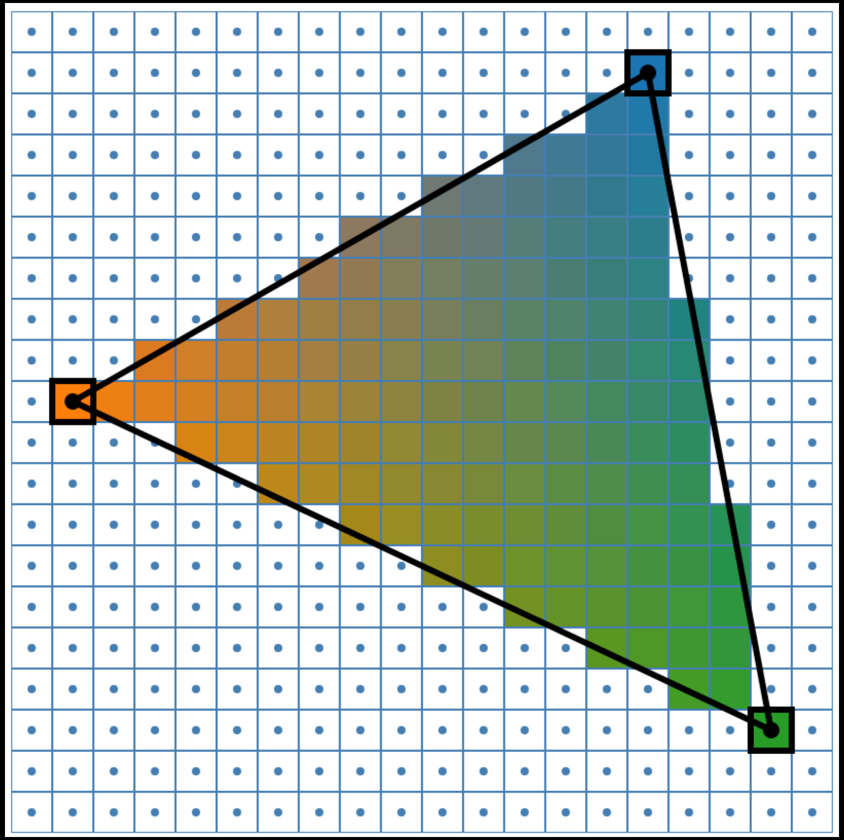
Interpolating values



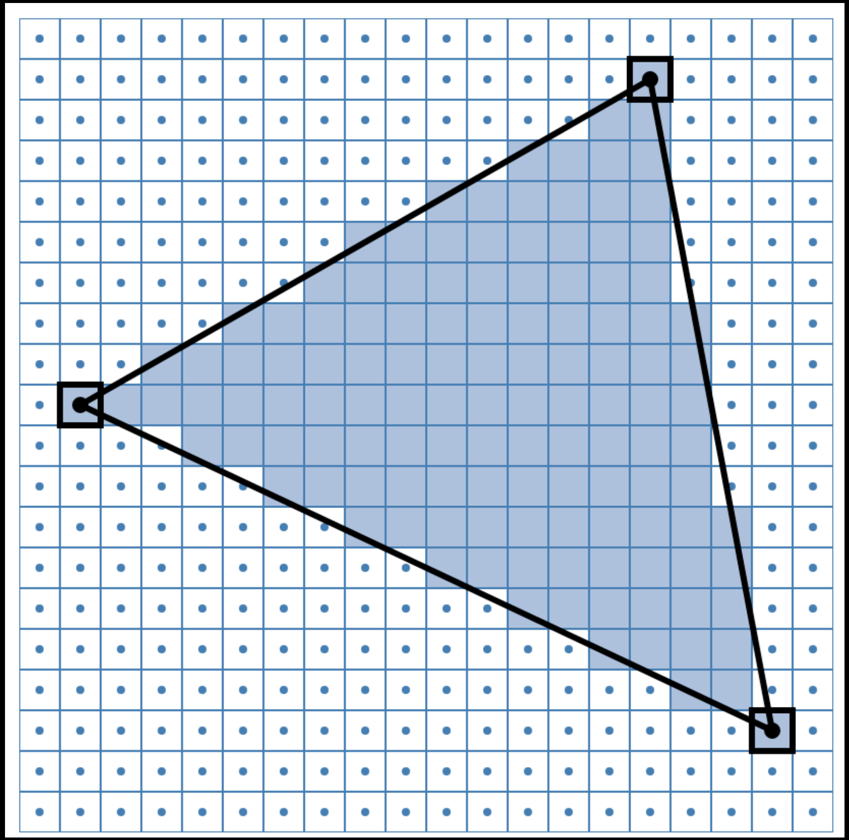
https://observablehq.com/@infowantstobeseen/drawing-lines

Triangle Rasterization: Raster each line?

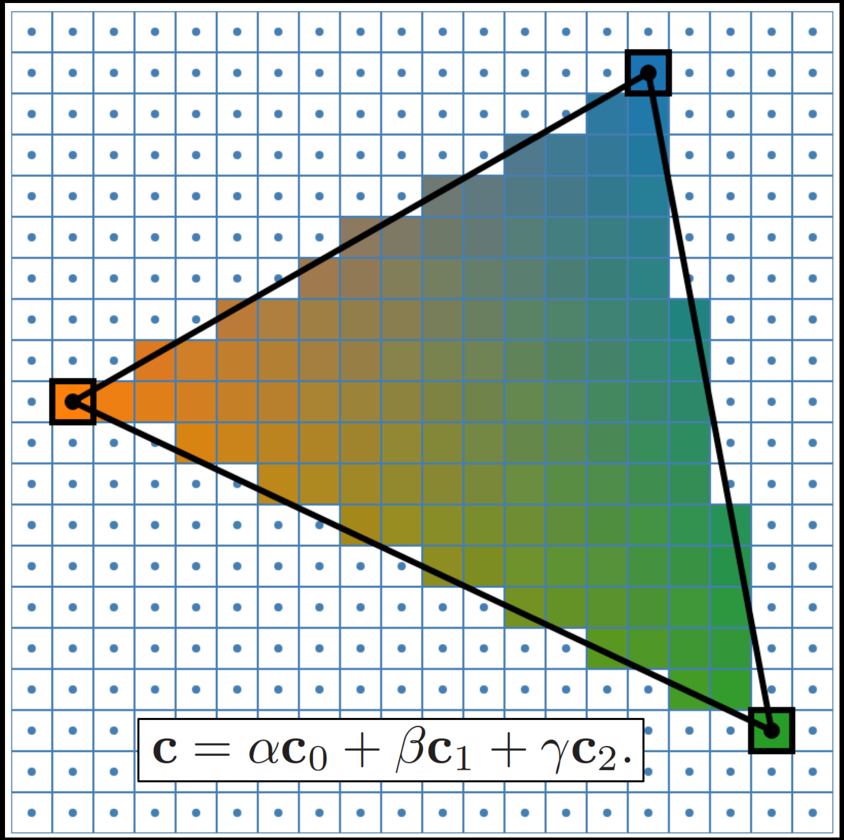




Triangle Rasterization

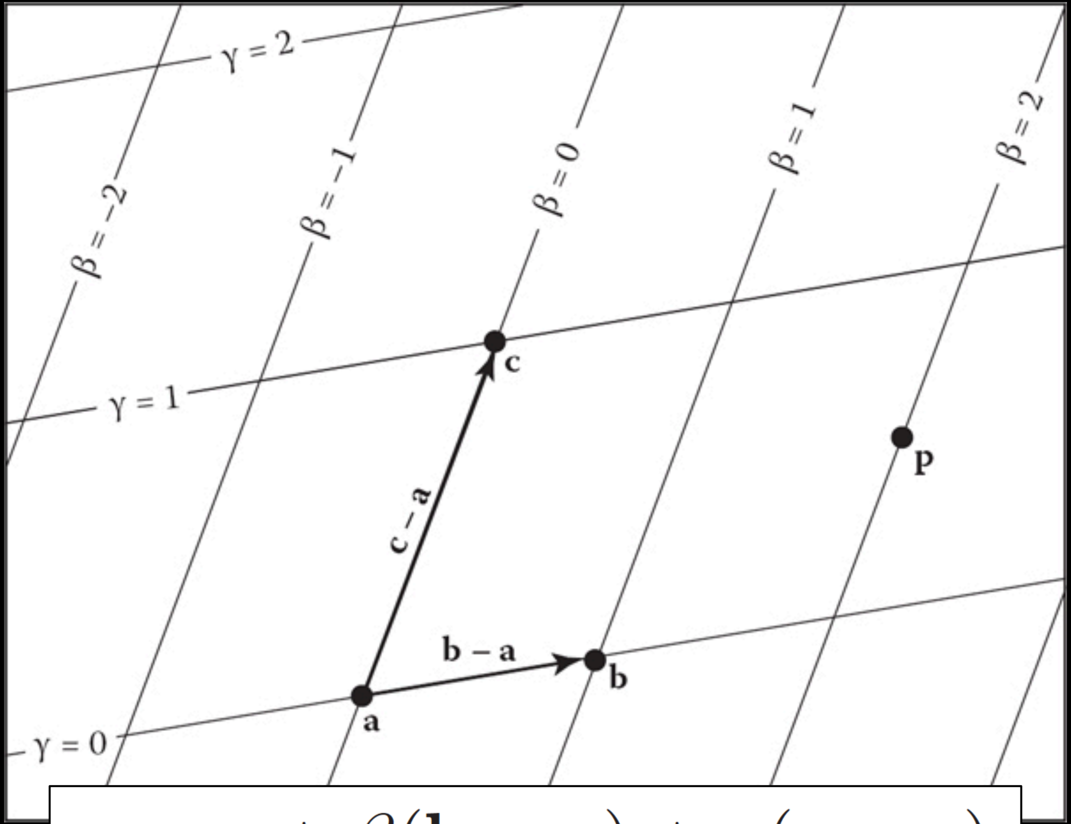


Inside-outside test



Interpolation

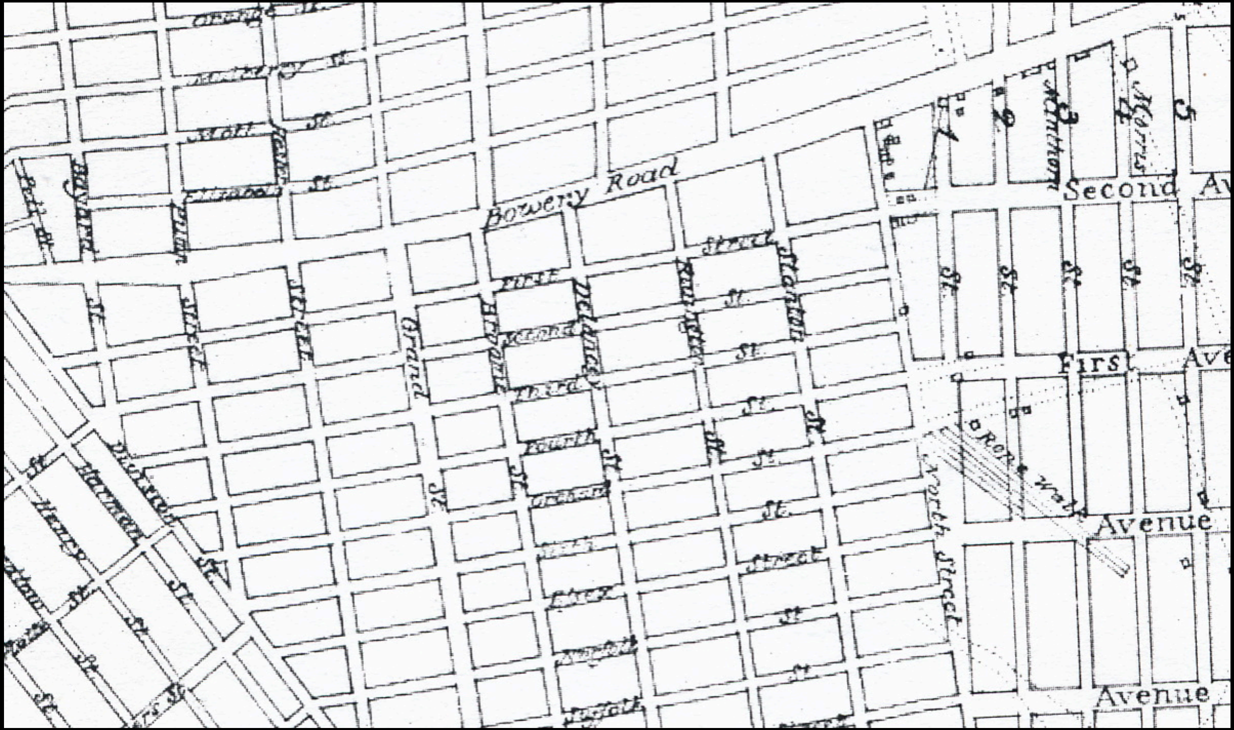
Triangle Rasterization: barycentric coordinates



$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

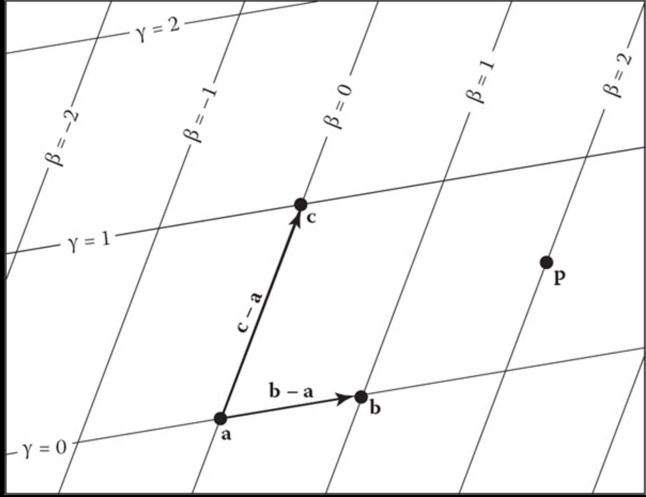
$\beta = 2, \gamma = 0.5$

Non-orthogonal coordinates



Triangle Rasterization: barycentric coordinates

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

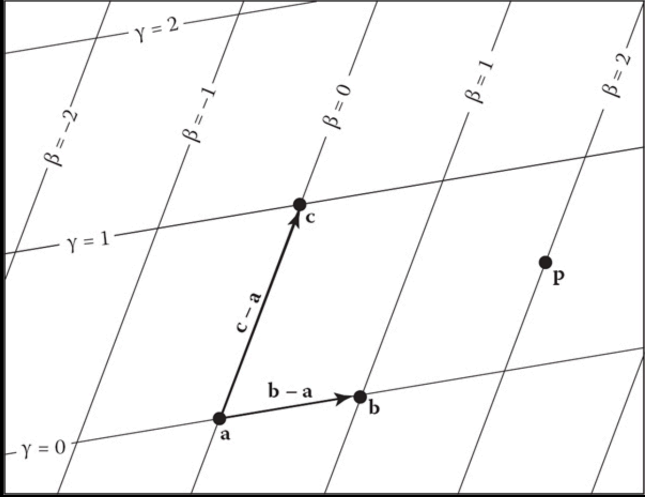


$$\beta = 2, \gamma = 0.5$$

Triangle Rasterization: barycentric coordinates

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$



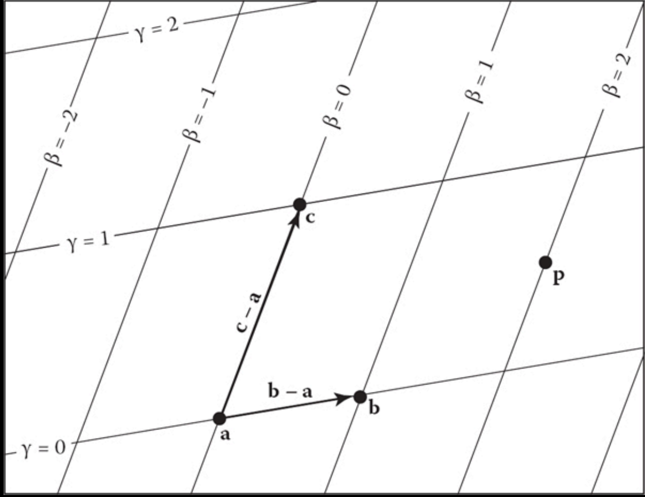
$$\beta = 2, \gamma = 0.5$$

Triangle Rasterization: barycentric coordinates

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

$$\alpha \equiv 1 - \beta - \gamma,$$



$$\beta = 2, \gamma = 0.5$$

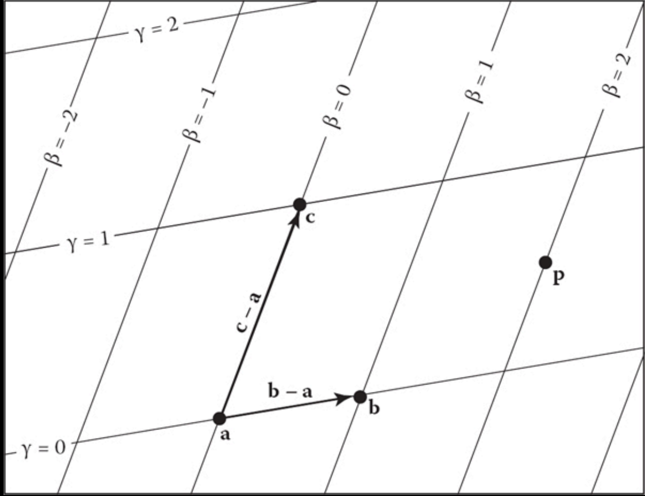
Triangle Rasterization: barycentric coordinates

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

$$\alpha \equiv 1 - \beta - \gamma,$$

$$\alpha + \beta + \gamma = 1.$$



$$\beta = 2, \gamma = 0.5$$

Triangle Rasterization: barycentric coordinates

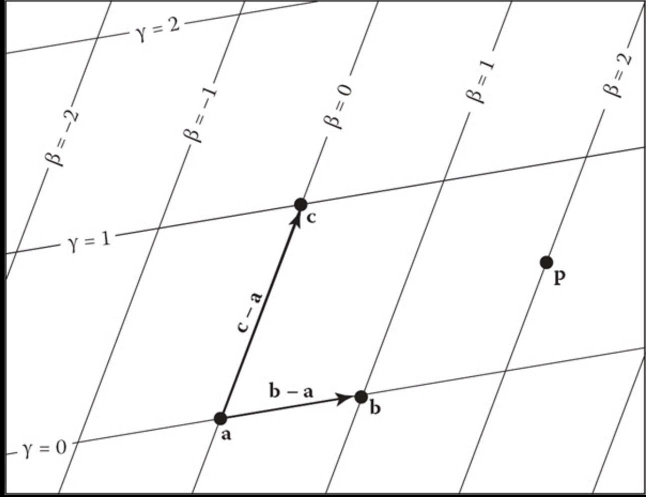
$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

$$\alpha \equiv 1 - \beta - \gamma,$$

$$\alpha + \beta + \gamma = 1.$$

$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c},$$



$$\beta = 2, \gamma = 0.5$$

Triangle Rasterization: barycentric coordinates

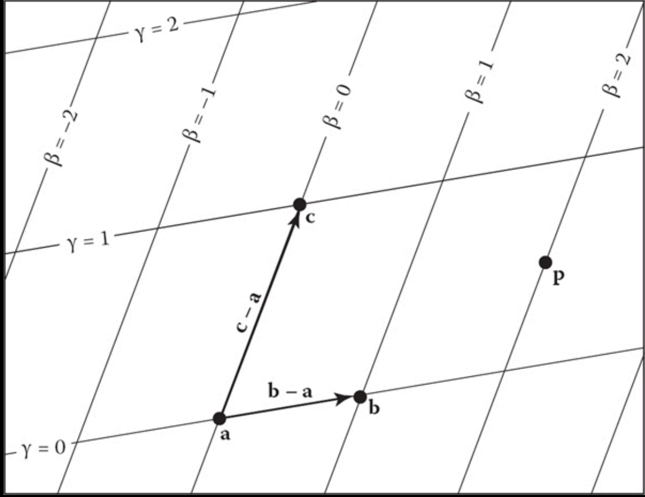
$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}).$$

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

$$\alpha \equiv 1 - \beta - \gamma,$$

$$\alpha + \beta + \gamma = 1.$$

$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c},$$

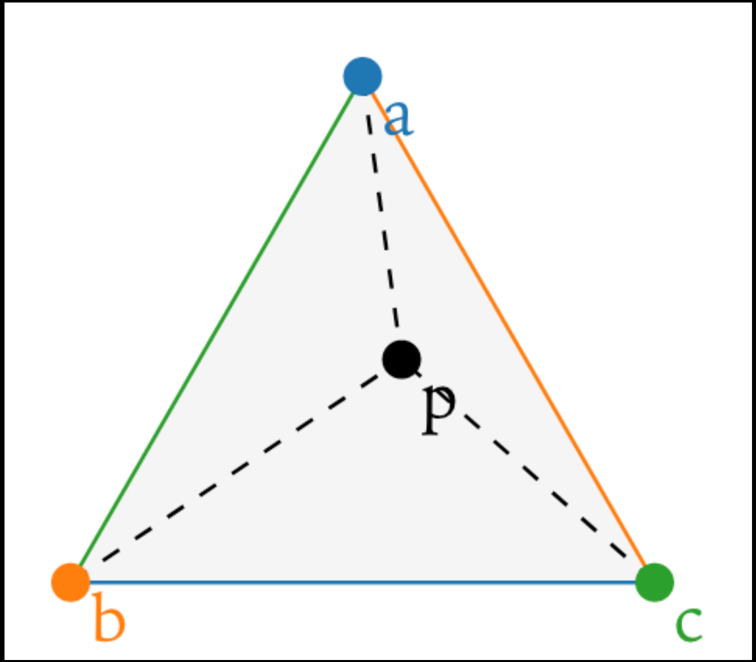


$\beta = 2, \gamma = 0.5$

barycentric coordinates

All points *inside* the triangle have:

$$0 < \alpha < 1,$$
$$0 < \beta < 1,$$
$$0 < \gamma < 1.$$



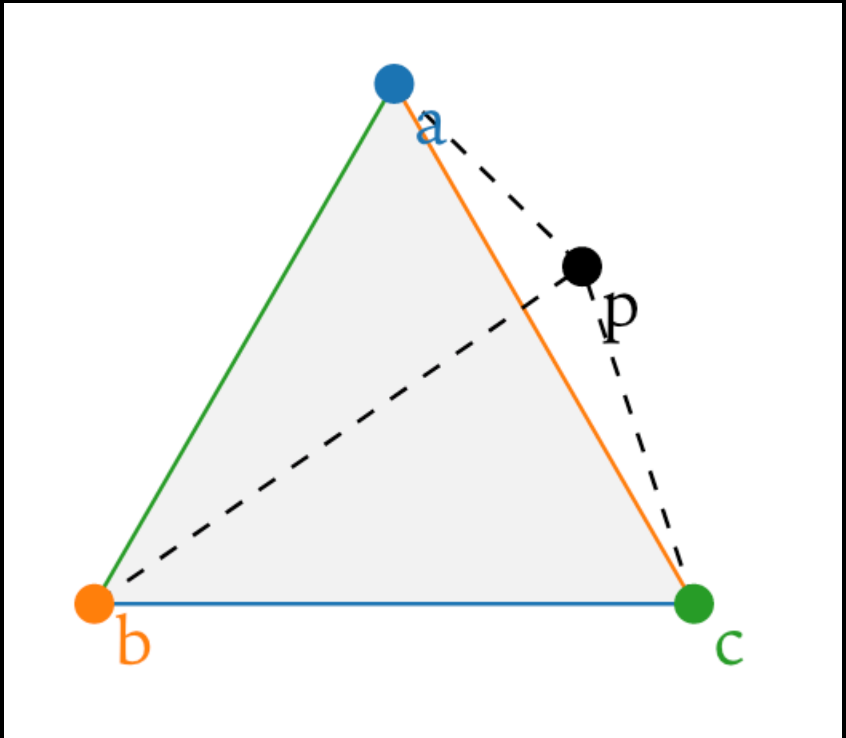
Barycentric Coords. for Δabc: $\alpha = 0.44$, $\beta = 0.21$, $\gamma = 0.35$

observablehq.com/@infowantstobeseen/barycentric-coordinates

barycentric coordinates

All points *inside* the triangle have:

$$0 < \alpha < 1,$$
$$0 < \beta < 1,$$
$$0 < \gamma < 1.$$



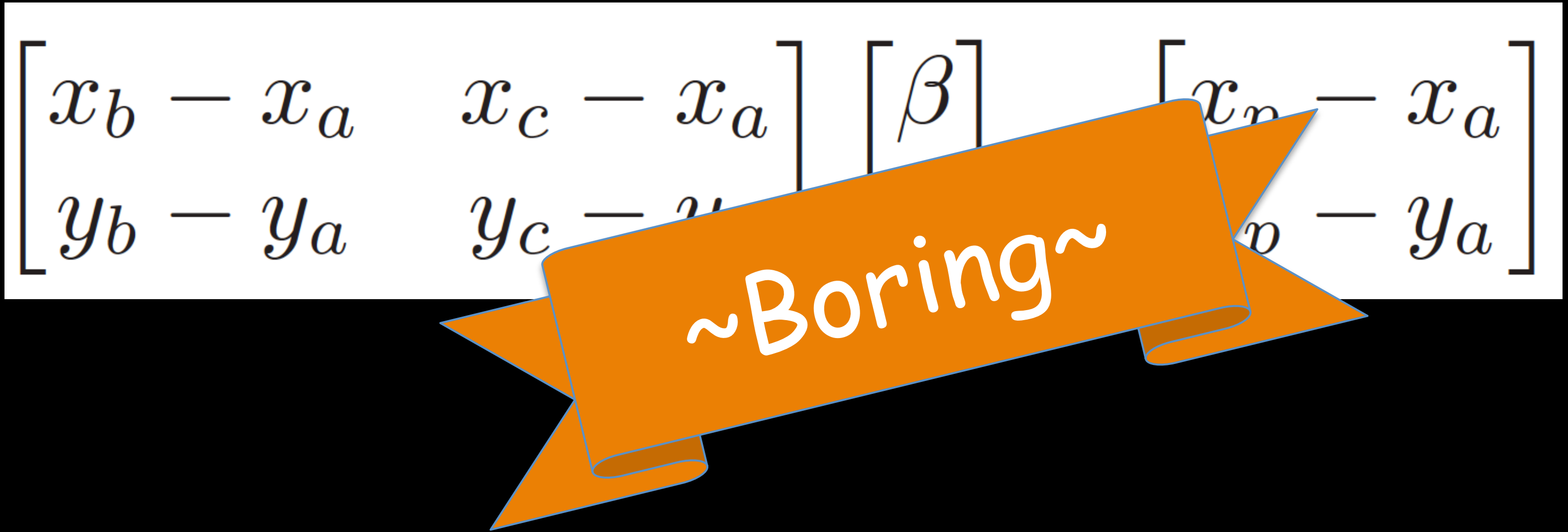
Barycentric Coords. for Δabc: $\alpha = 0.65$, $\beta = -0.14$, $\gamma = 0.49$

observablehq.com/@infowantstobeseen/barycentric-coordinates

Calculate barycentric coordinates

$$\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x_p - x_a \\ y_p - y_a \end{bmatrix}$$

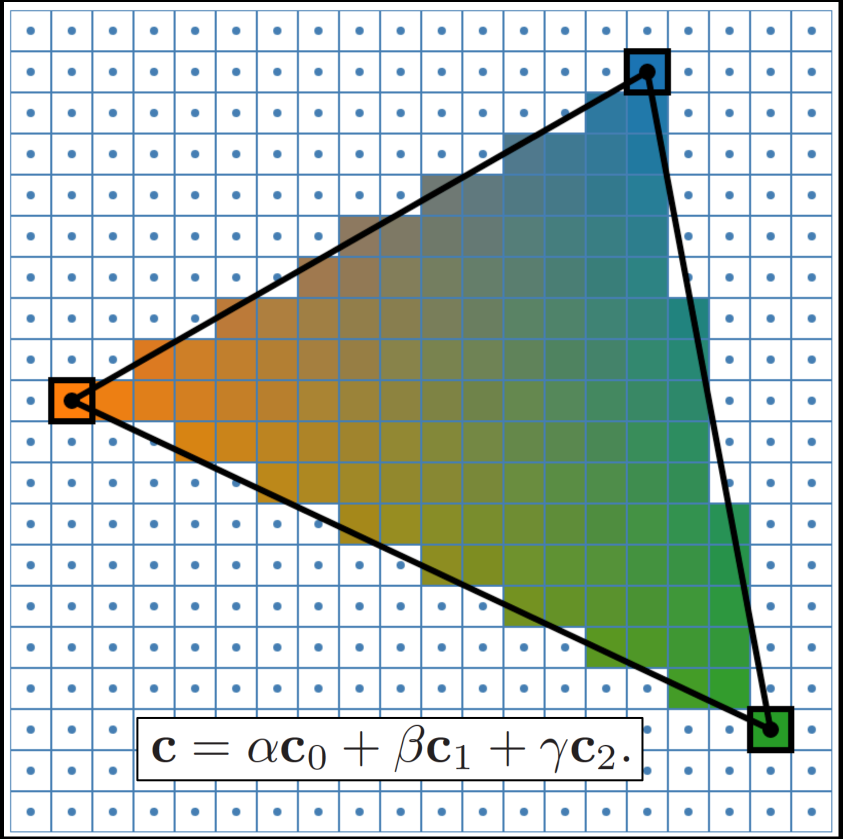
Calculate barycentric coordinates



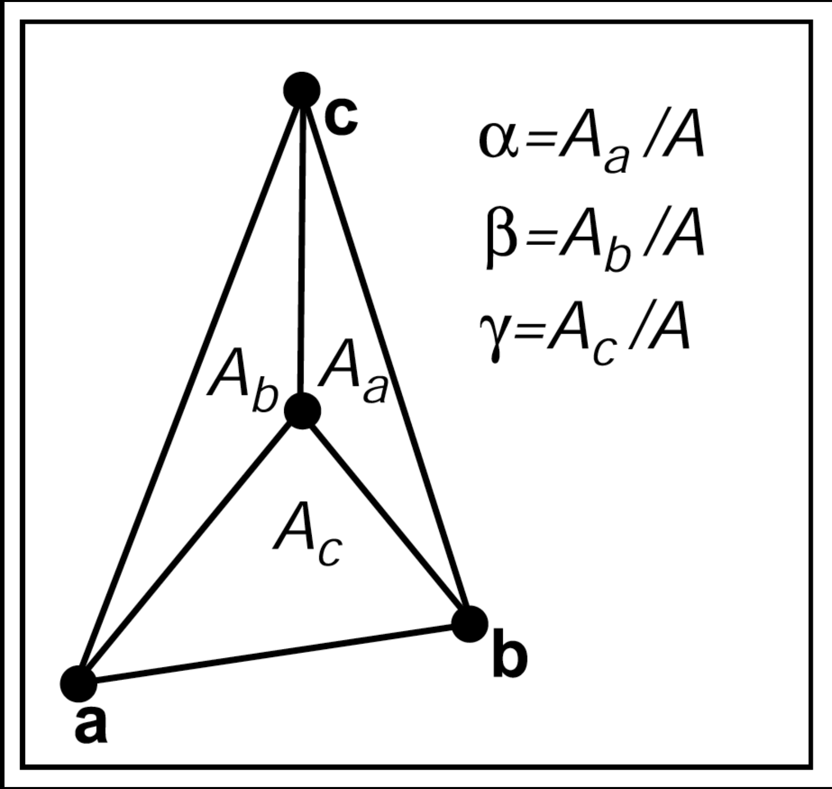
Use geometric reasoning…

Triangle Rasterization

$x_{\min} = \text{floor}(x_i)$
$x_{\max} = \text{ceiling}(x_i)$
$y_{\min} = \text{floor}(y_i)$
$y_{\max} = \text{ceiling}(y_i)$
for $y = y_{\min}$ to $y_{\max}$ **do**
  for $x = x_{\min}$ to $x_{\max}$ **do**
    $\alpha = f_{12}(x, y)/f_{12}(x_0, y_0)$
    $\beta = f_{20}(x, y)/f_{20}(x_1, y_1)$
    $\gamma = f_{01}(x, y)/f_{01}(x_2, y_2)$
    if $(\alpha > 0$ and $\beta > 0$ and $\gamma > 0)$ **then**
      $\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$
      drawpixel (x, y) with color $\mathbf{c}$

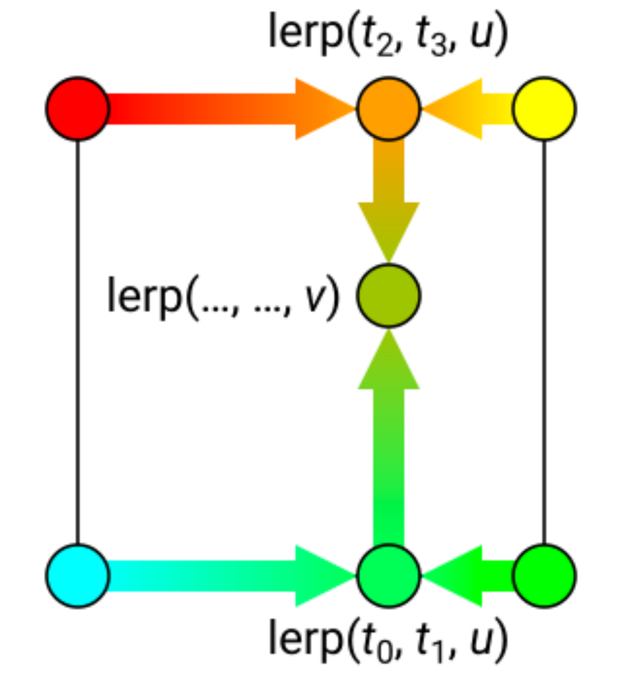


Option 2) Barycentric coordinates *via* areas



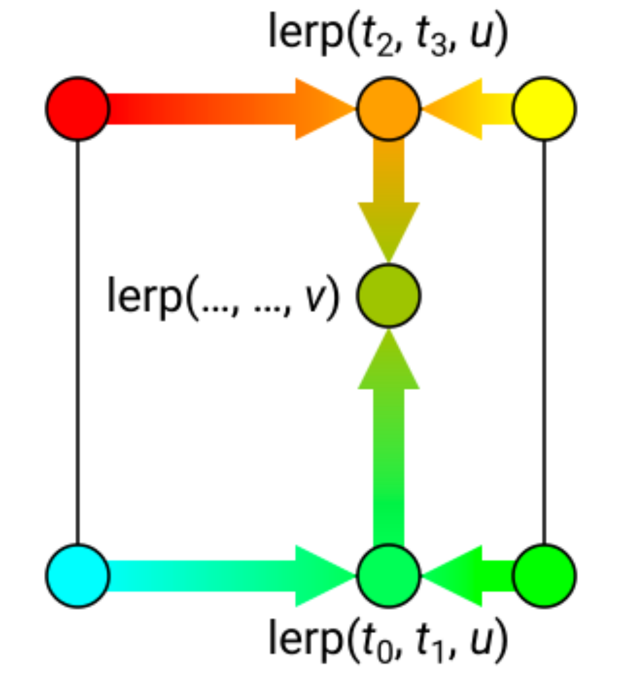
Quad Rasterization?

- *Bilinear interpolation*

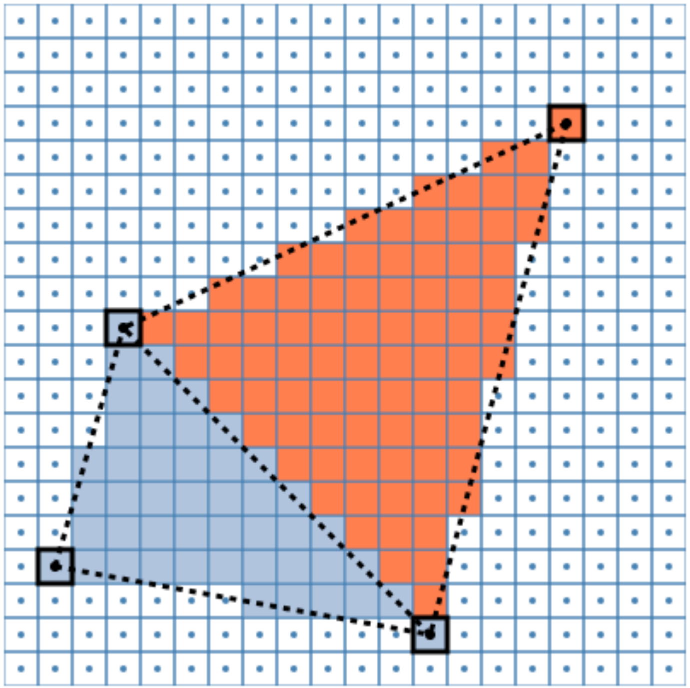


Quad Rasterization?

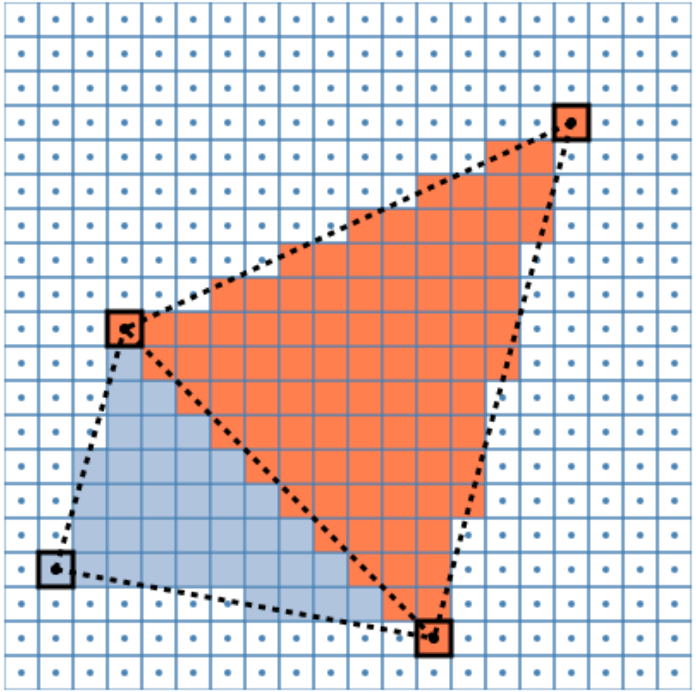
- *Bilinear interpolation… but is not unique (e.g. mean value)*
- *Hardware is specifically optimized for triangles*
- *Graphics drivers typically split input geometry into triangles*



Shared Edges



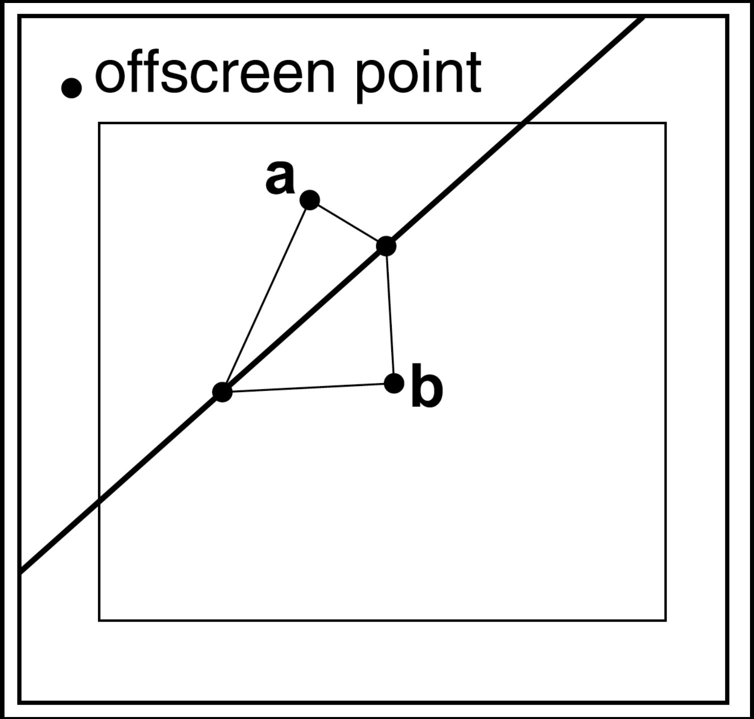
Issue with triangles that overlap.



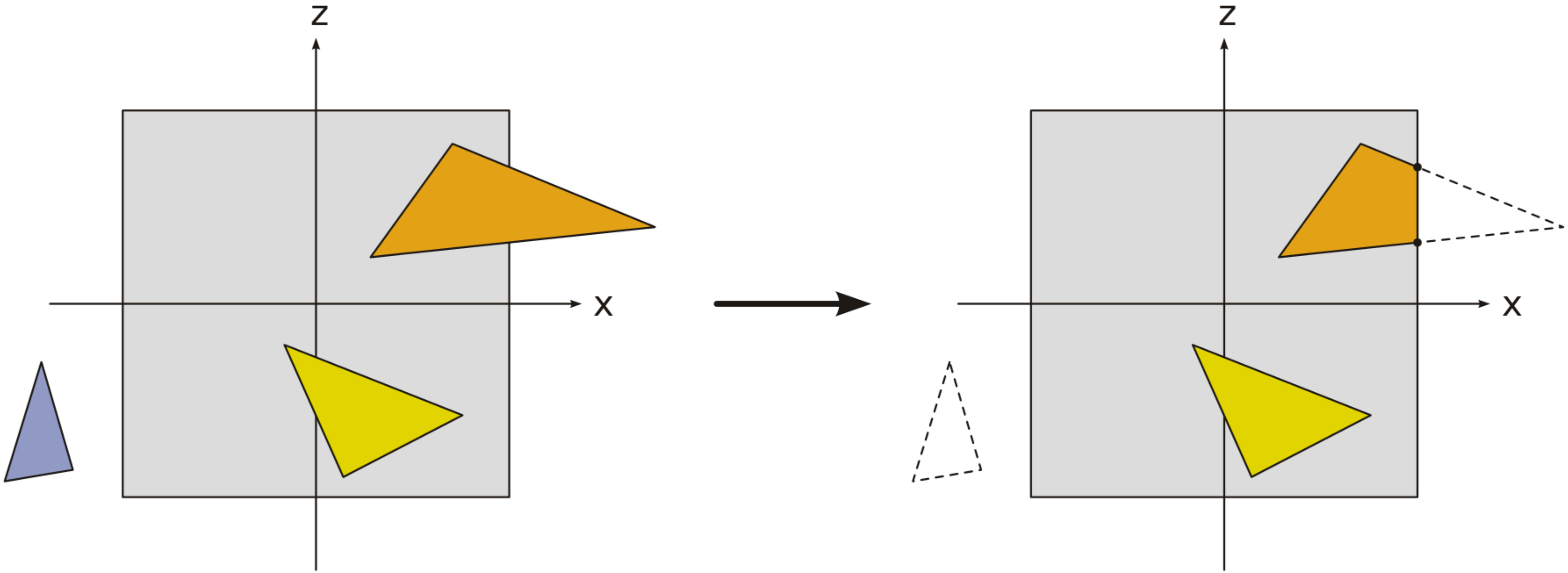
Issue with triangles that overlap.

https://observablehq.com/@infowantstobeseen/drawing-triangles

Shared Edges

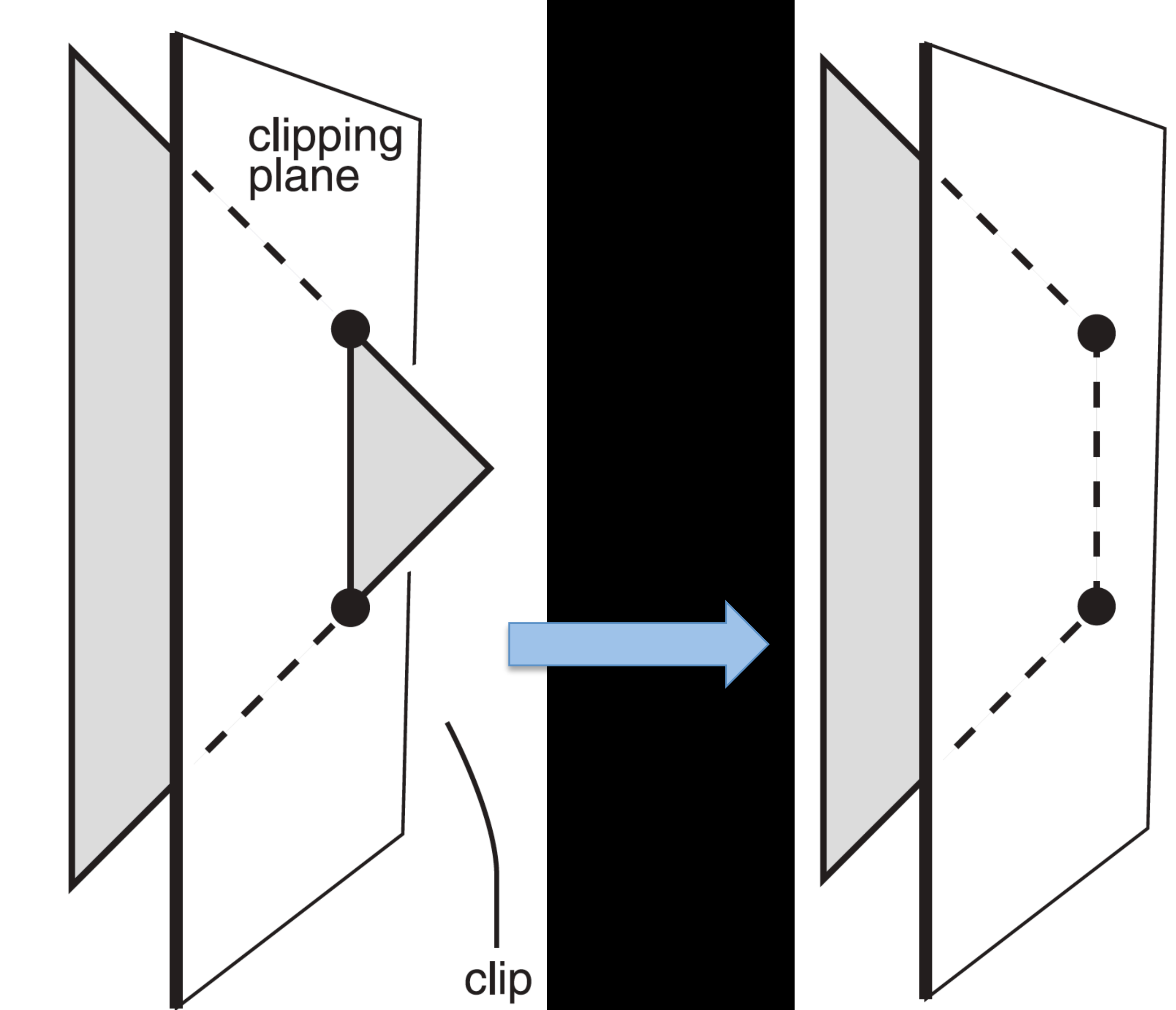


Clipping

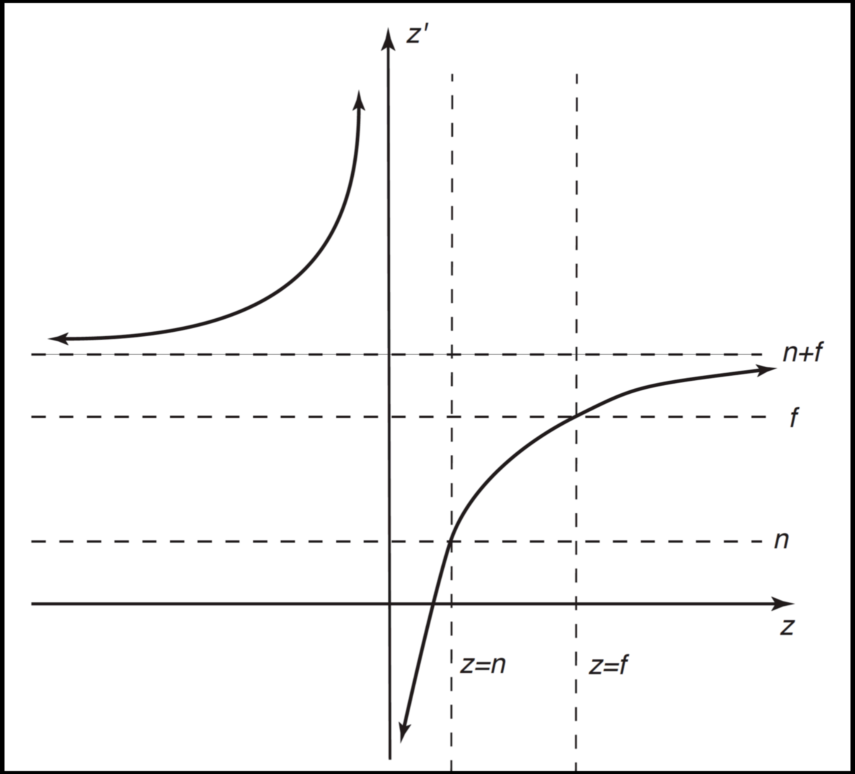


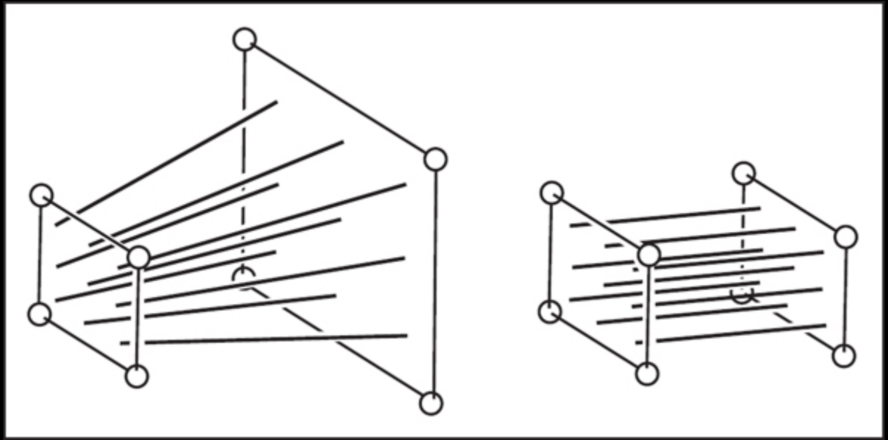
wikipedia.org (graphics pipeline)

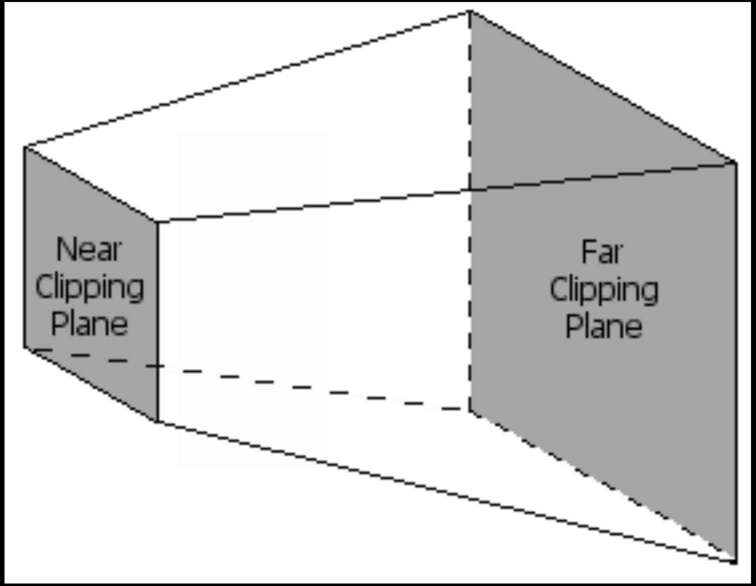
Clipping



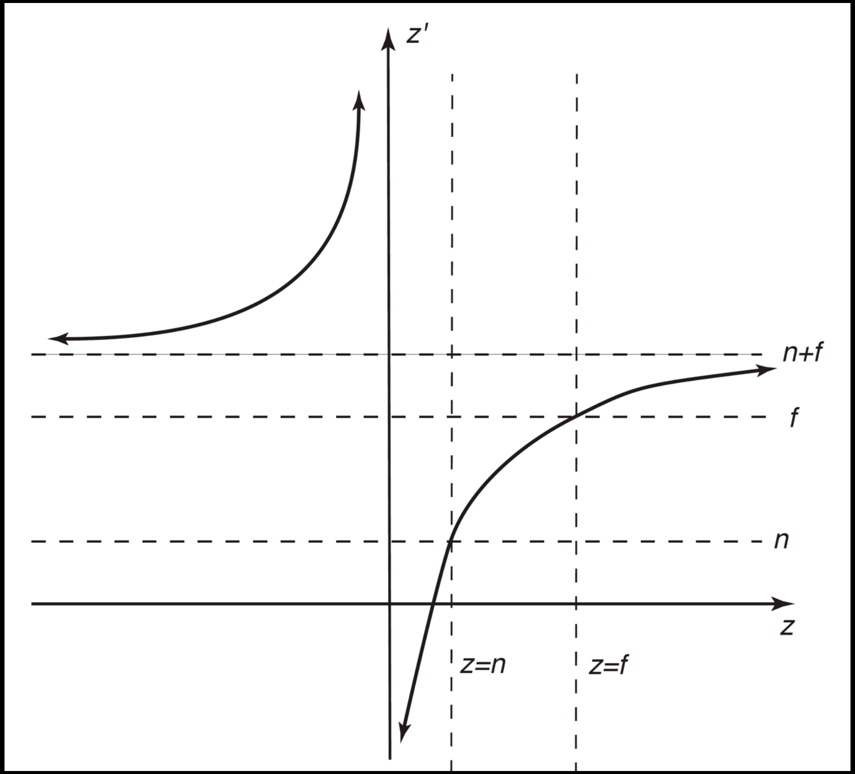
Most pernicious: near plane clipping

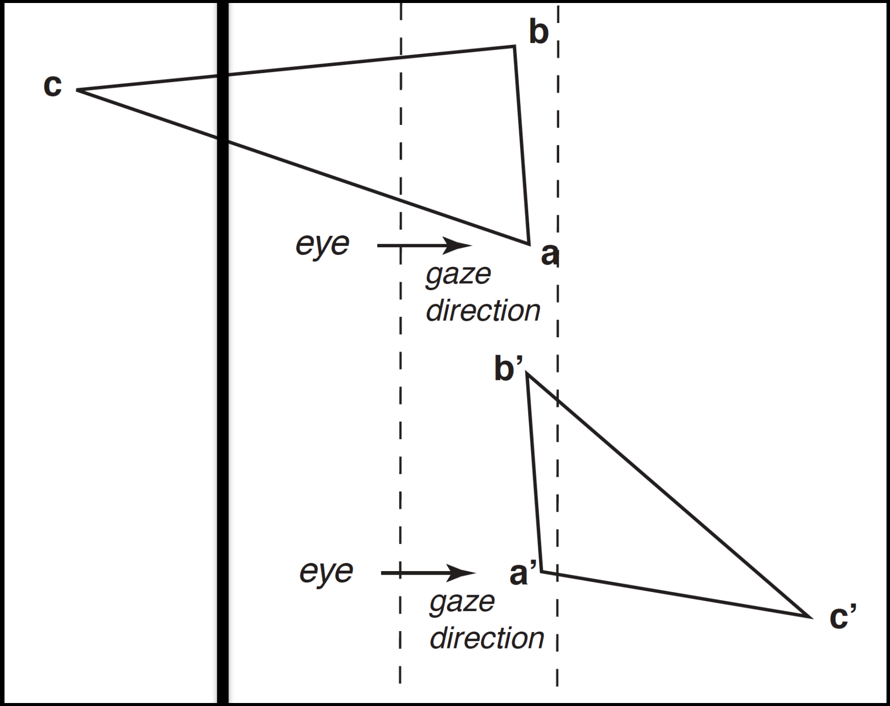




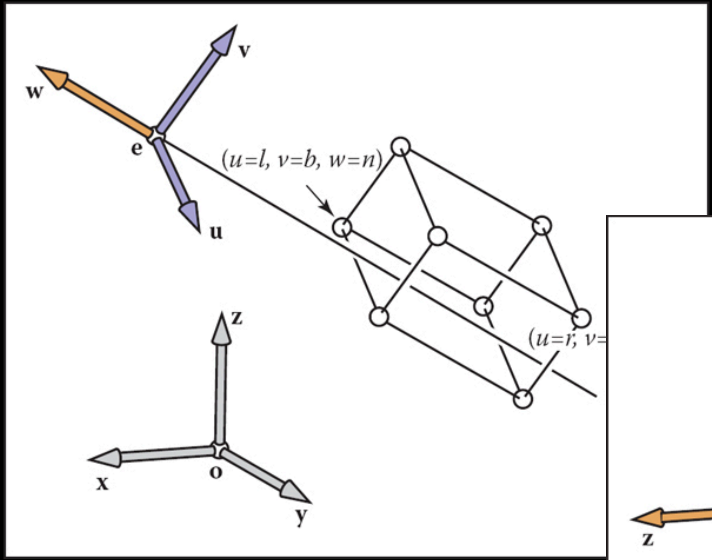


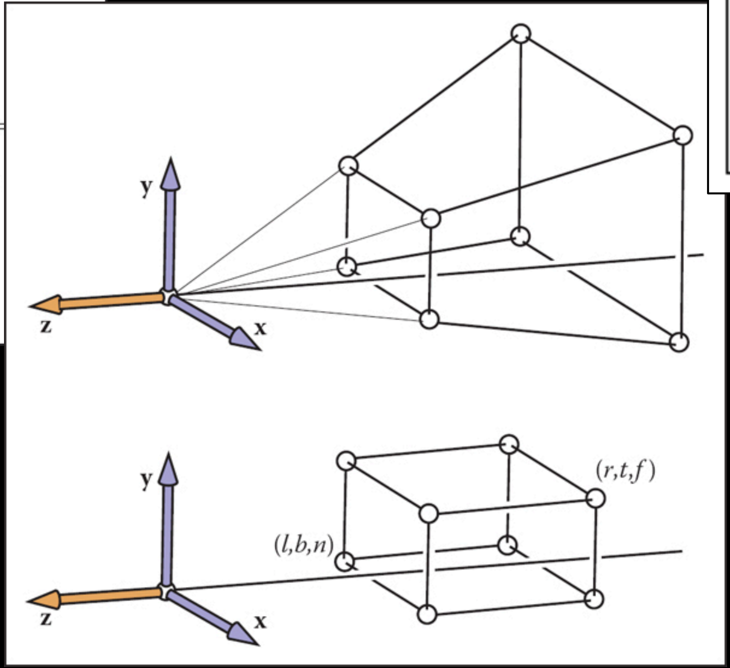
Most pernicious: near plane clipping

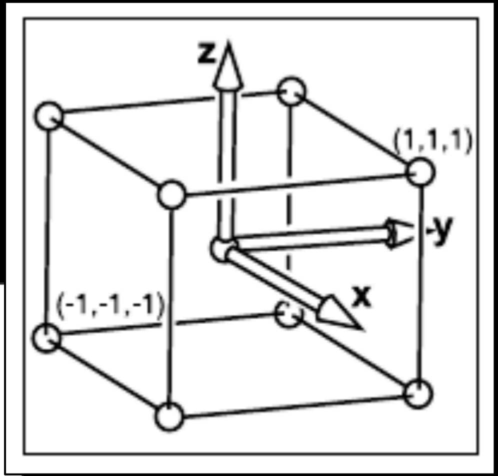




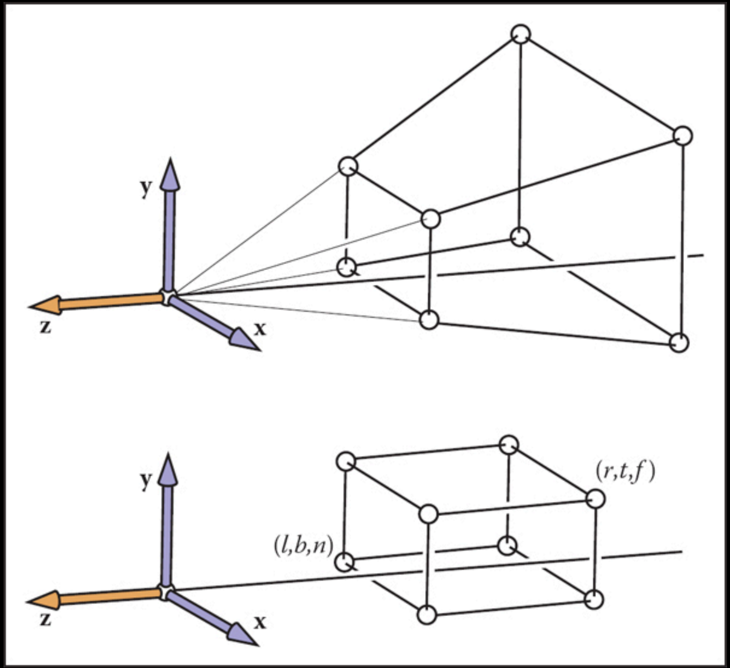
Choose the space to clip within





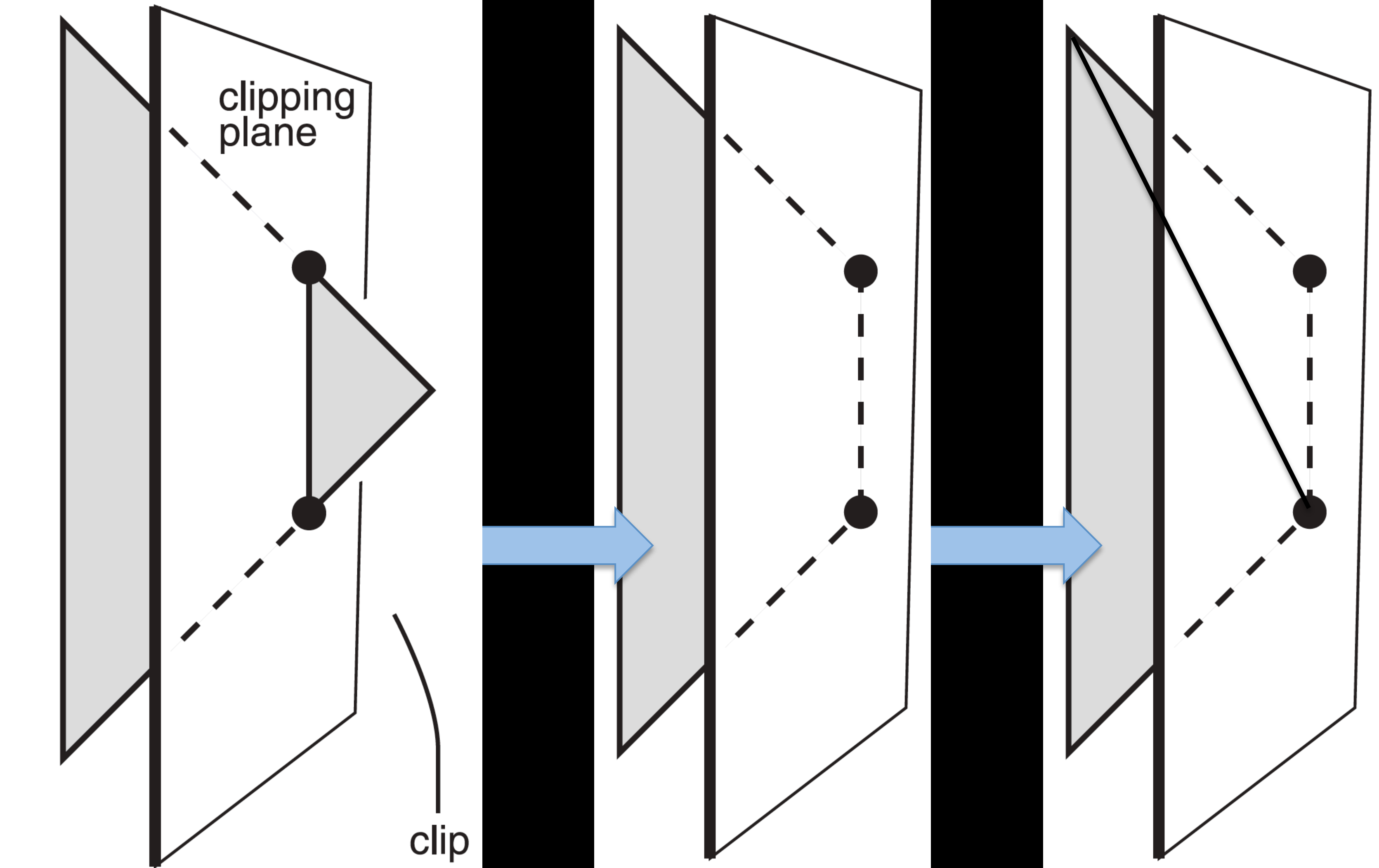


Choose the space to clip within

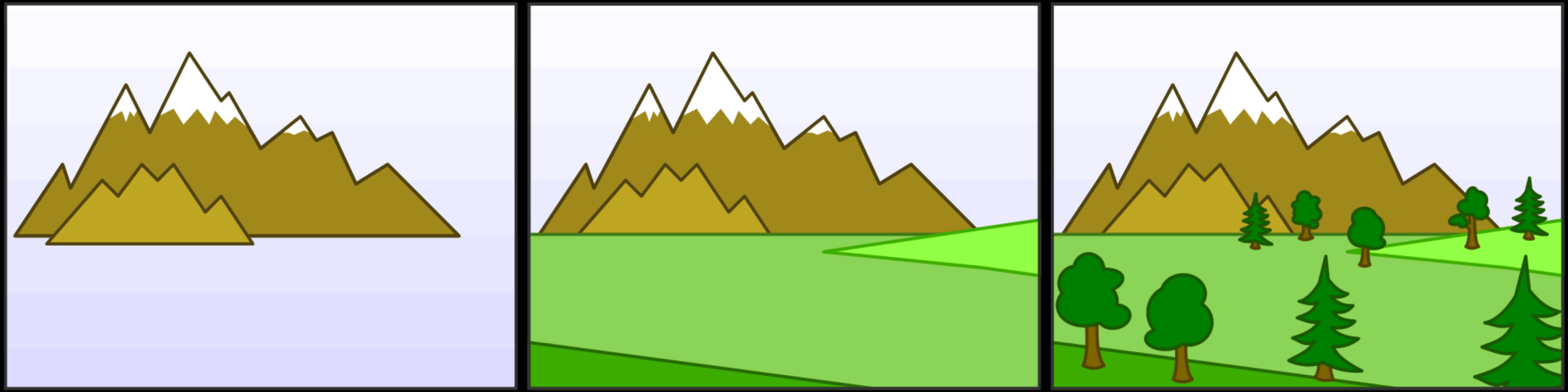


for each of six planes **do**
  if (triangle entirely outside of plane) **then**
    break (triangle is not visible)
  else if triangle spans plane **then**
    clip triangle
    if (quadrilateral is left) **then**
      break into two triangles

Use geometric reasoning…


clipping
plane
clip

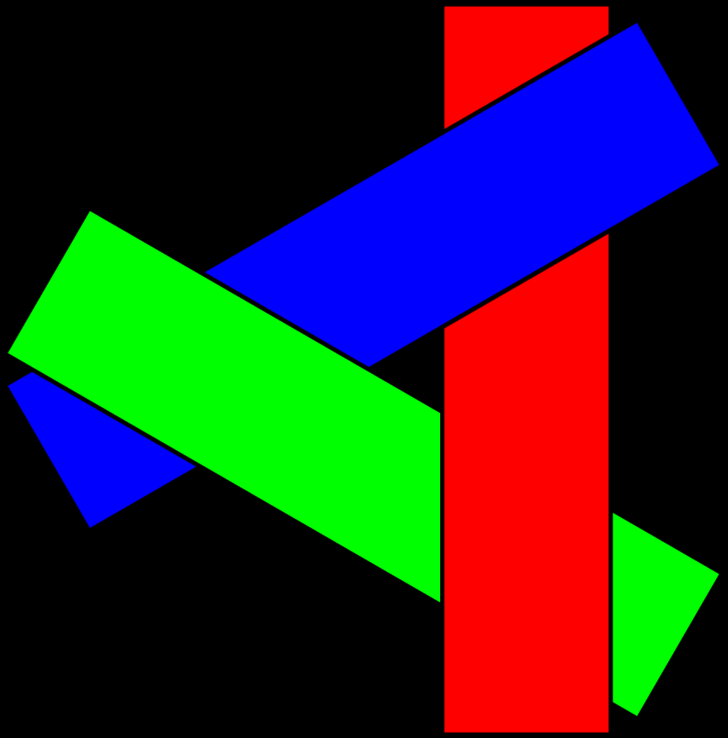
Minimal 3D Pipeline



Sort 3D rendering by object depth

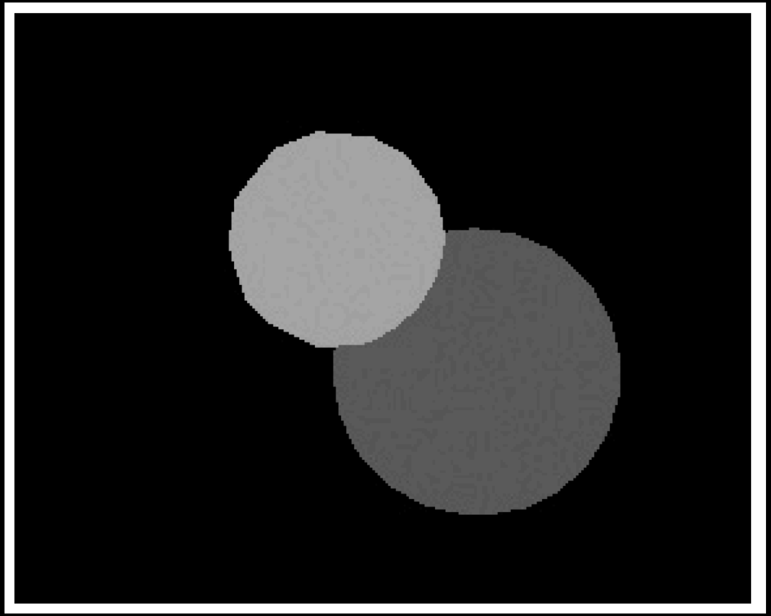
Wikipedia.org

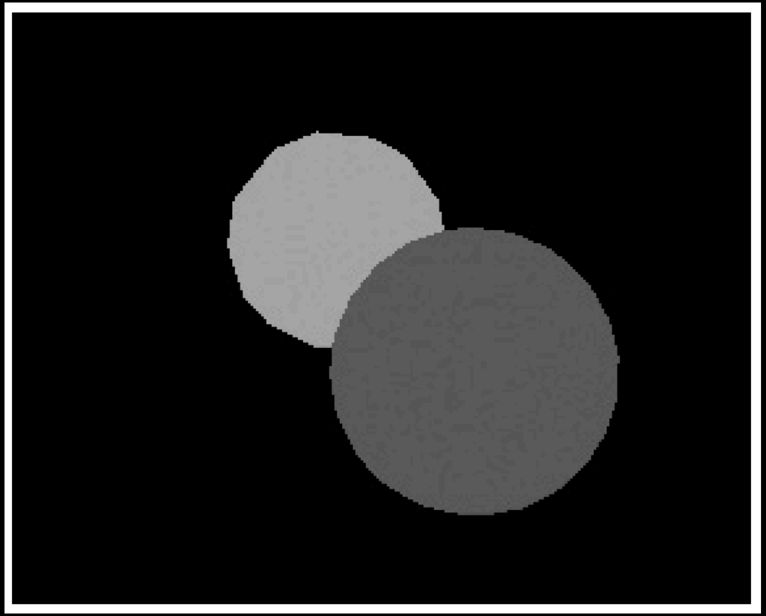
Occlusion cycle: *painter's algorithm breaks down…*



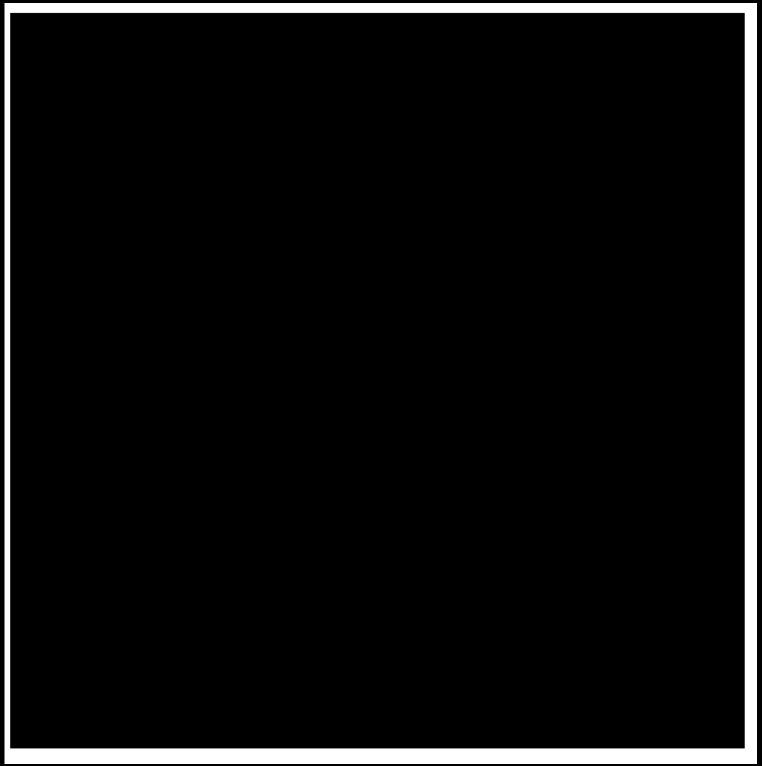
Wikipedia.org

Sort 3D rendering by depth



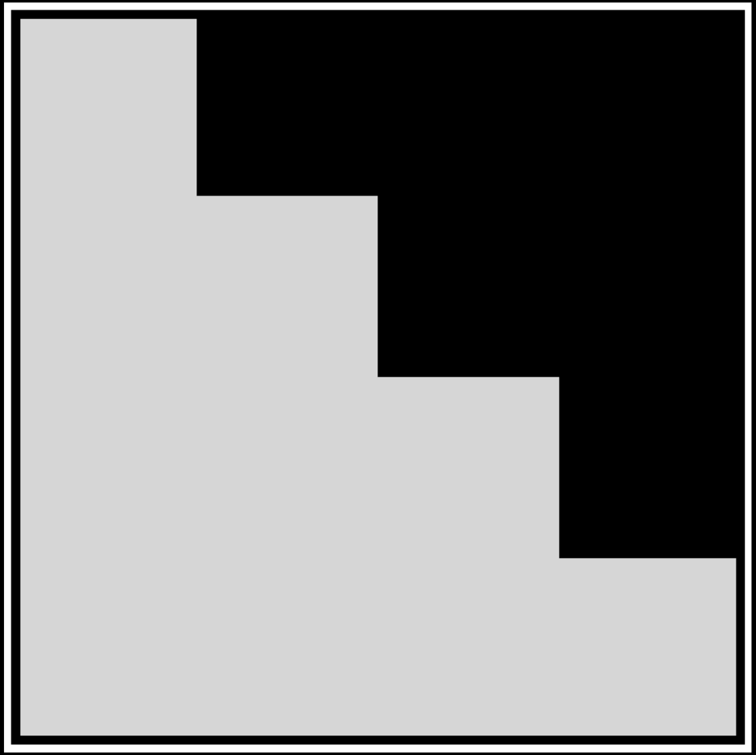


z-Buffer



∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

z-Buffer

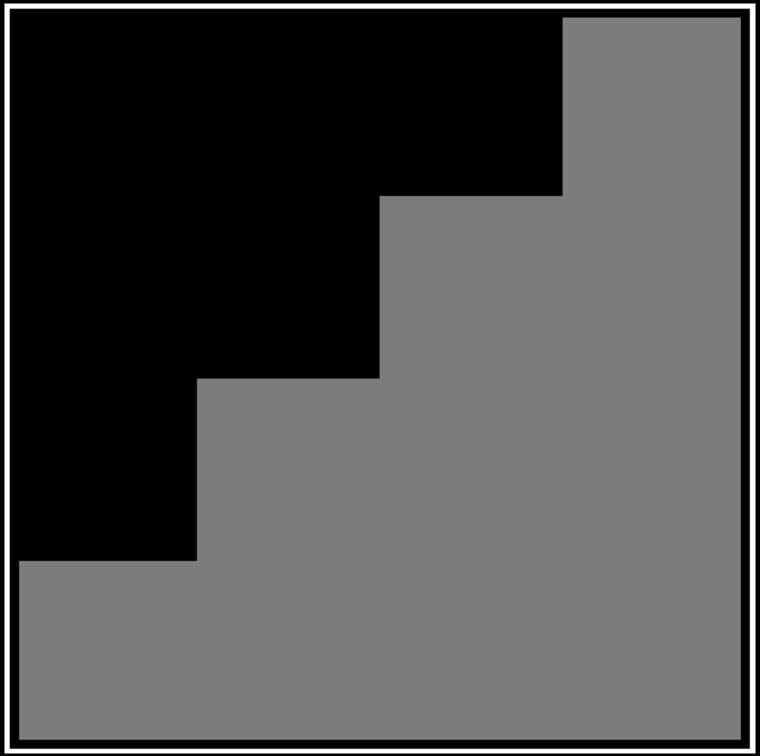


Triangle A

1	∞	∞	∞
1	3	∞	∞
1	3	5	∞
1	3	5	7

Triangle A depth values

z-Buffer

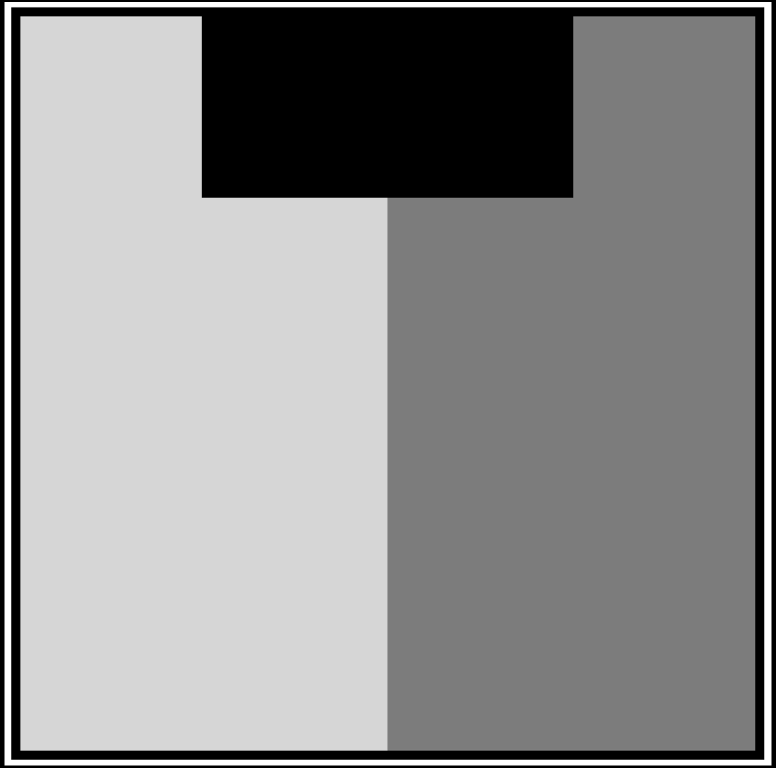


Triangle B

∞	∞	∞	1
∞	∞	3	1
∞	5	3	1
7	5	3	1

Triangle B depth values

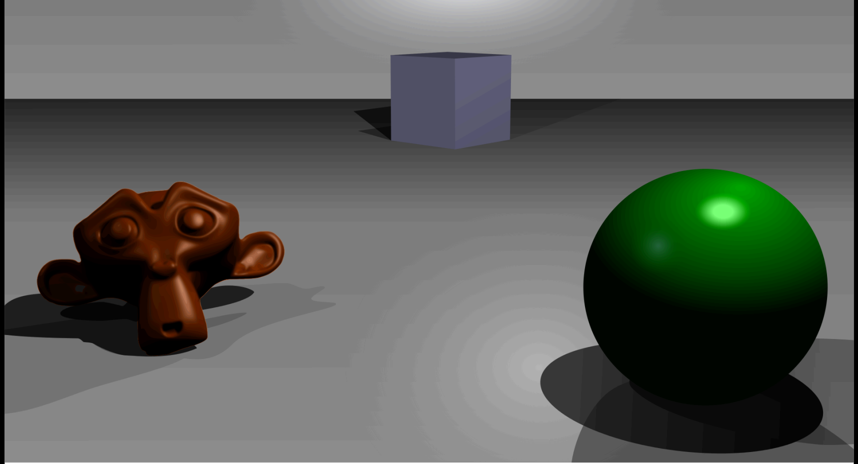
z-Buffer



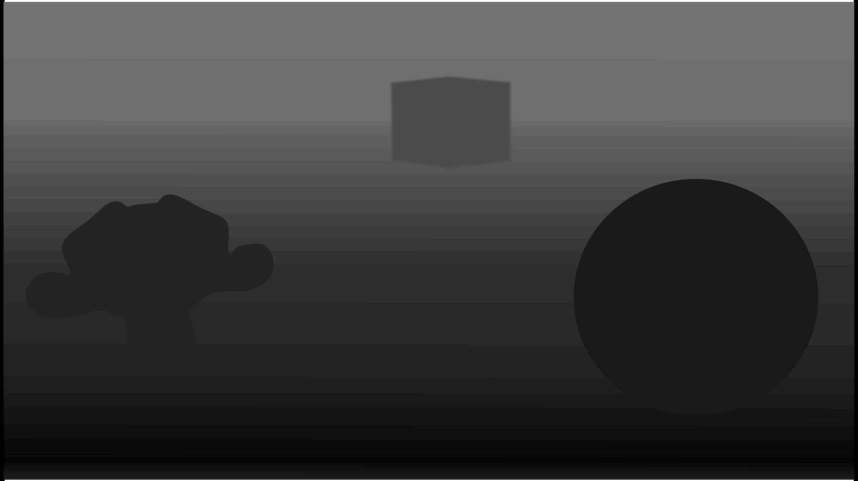
Triangle A & Triangle B

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

Combined z-buffer

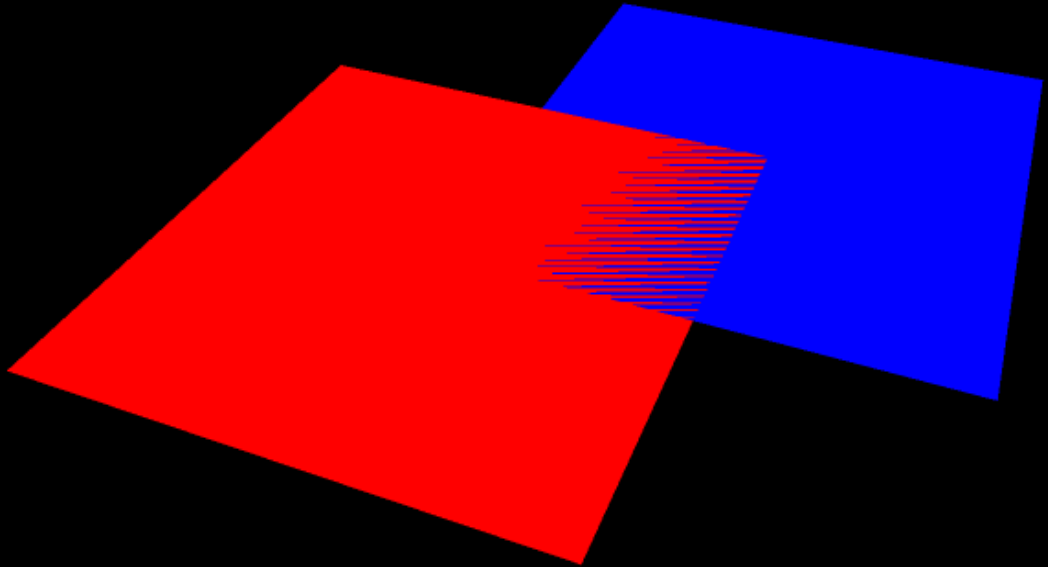


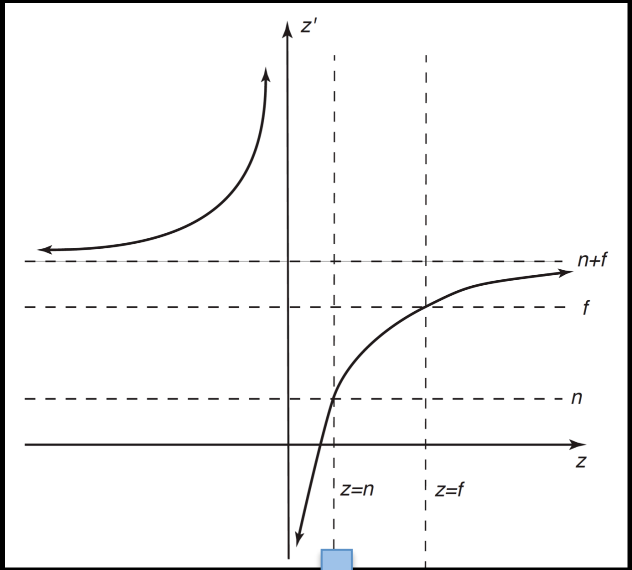
A simple three-dimensional scene



Z-buffer representation

z-Buffer: “z-fighting”



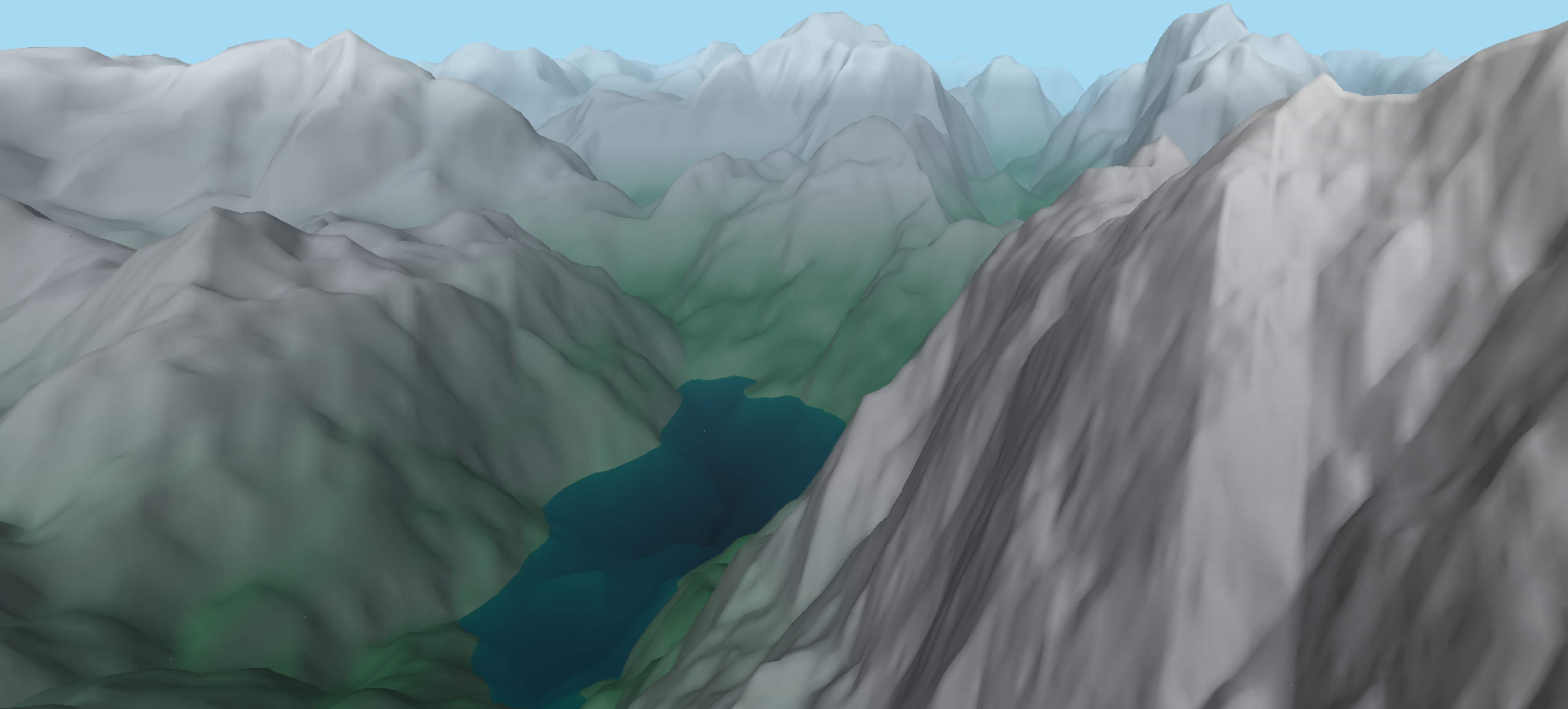


1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

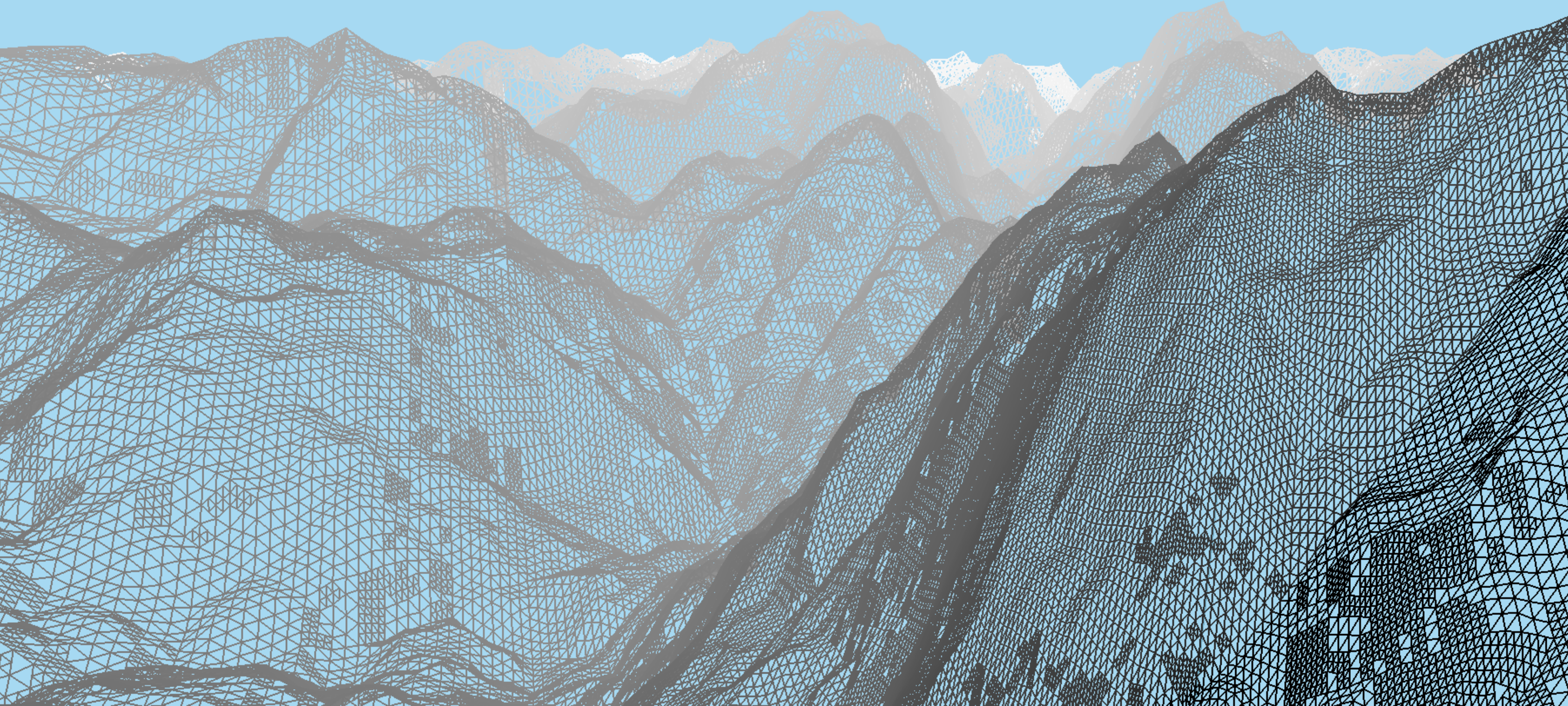
Culling primitives

- *View volume culling*
- *Backface culling*
- *Occlusion culling*

Culling primitives



Culling primitives



Culling primitives