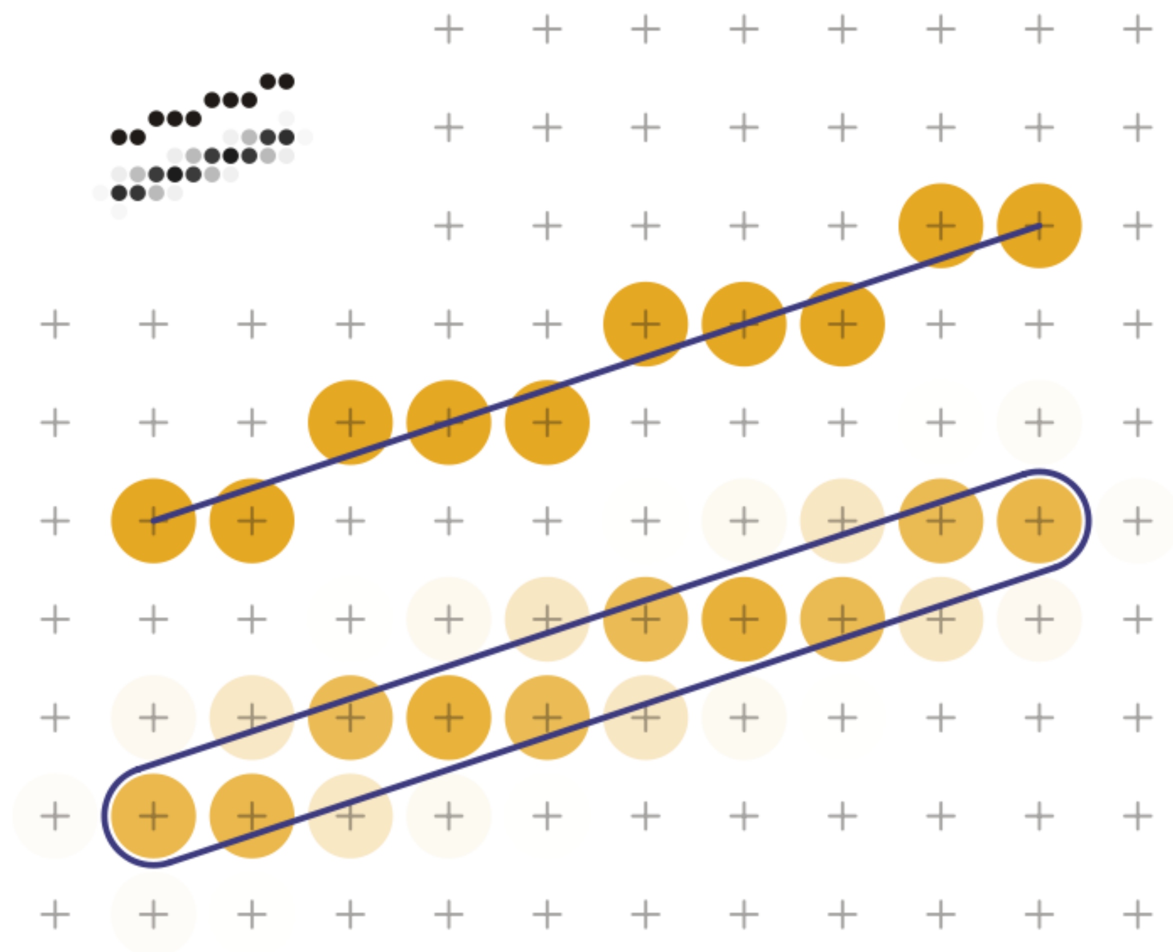


Scan Conversion

Lines and Triangles

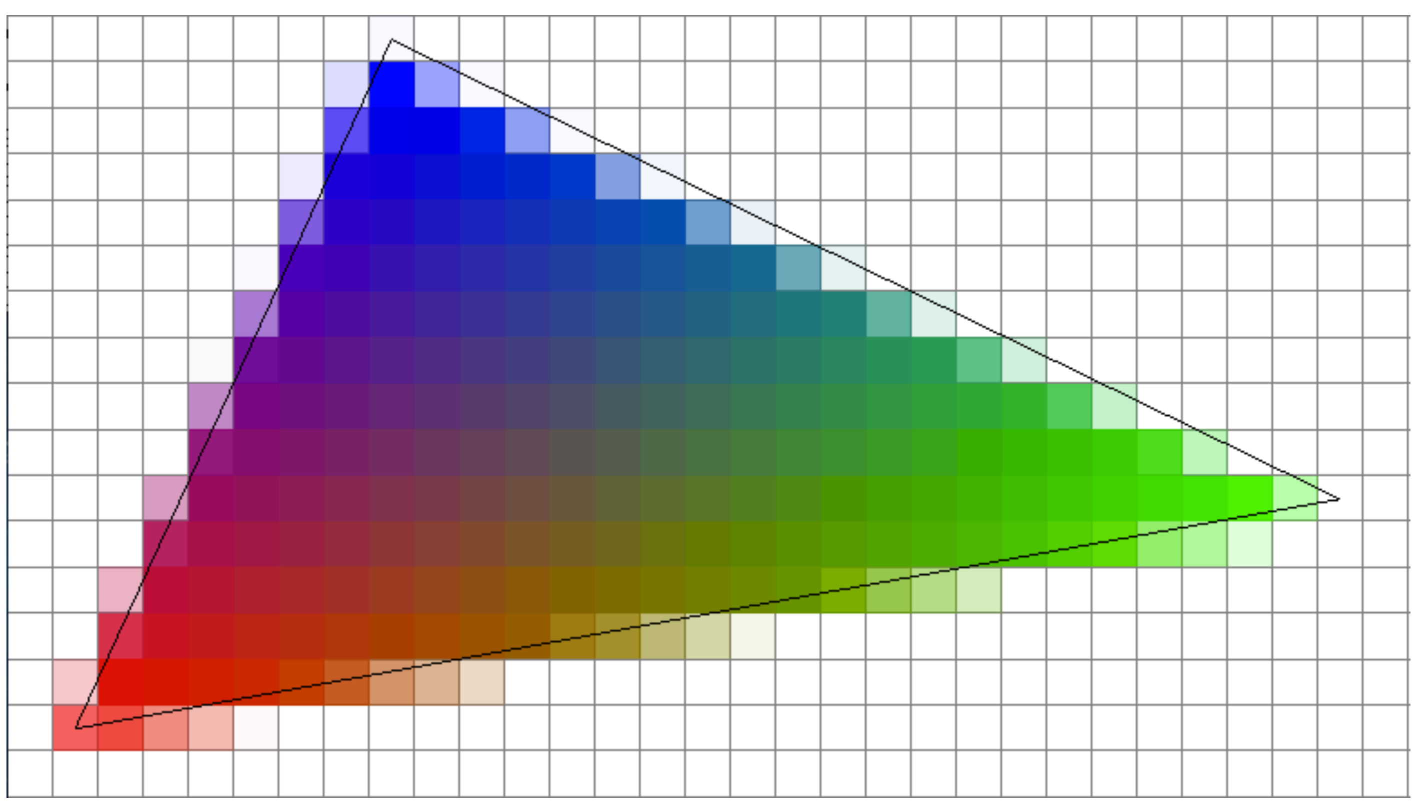
Introduction to Computer Graphics
CSE 533 / 333

Ojaswa Sharma

Rasterization

Rasterization is a central operation in graphics.

1. *Enumerates* the pixels that are covered by a primitive
2. *Interpolates* values (or attributes) across the primitive

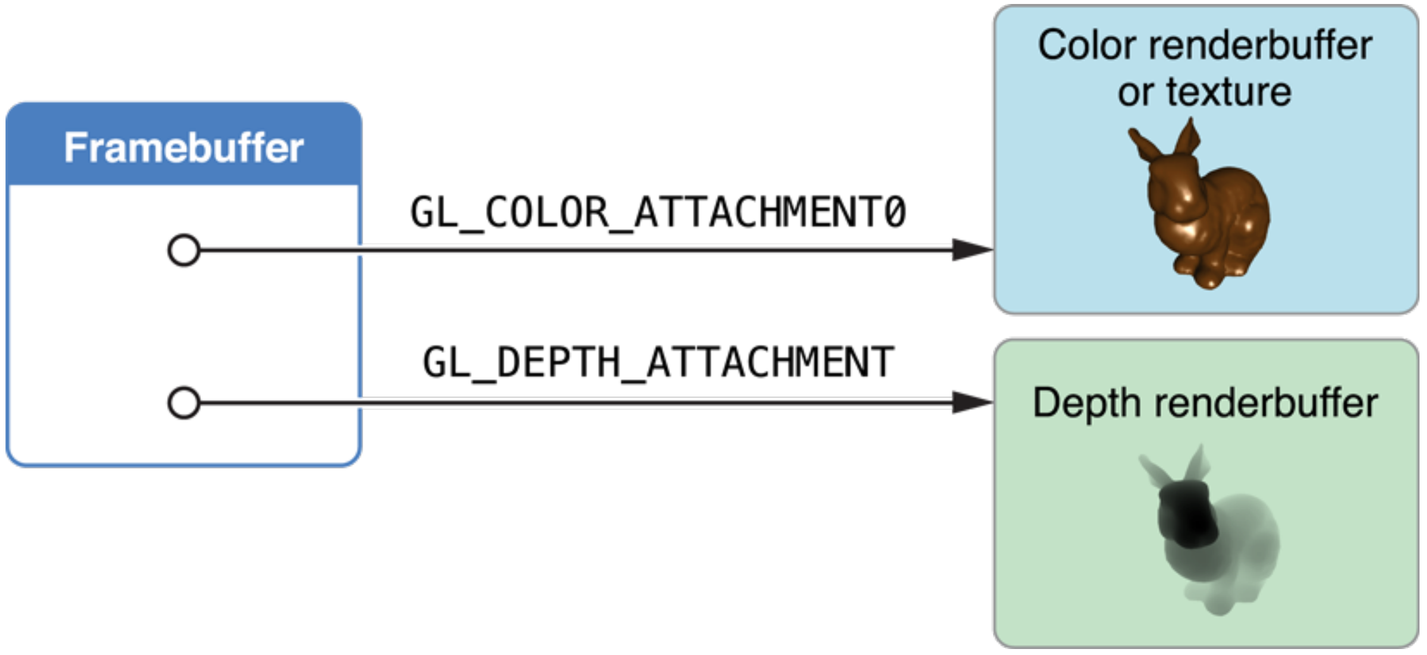


Source: http://web.eecs.umich.edu/~sugih/courses/eecs487/pa1.html

Rasterization

The output of a rasterizer is a set of *fragments*, one for each pixel covered by the primitive.

- Each fragment lives at a pixel position
- Carries its own set of attribute values (color, depth, etc.)



Source: https://developer.apple.com

Rasterizing Lines

DDA Algorithm

After, *digital differential analyzer*, an early electro-mechanical device for digital simulation of differential equations.

$$\frac{dy}{dx} = m, \ \ m\text{: slope}$$

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x}, 0 \leq m \leq 1.$$

For Δx change in x-coordinate, change in y-coordinate is

$$\Delta y = m\Delta x$$

Beware: floating point computations!

Midpoint Line Algorithm

- Drawing lines using the implicit equation

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

- Pitteway, 1967; van Aken and Novak, 1985
- Produces same lines as the Bresenham's algorithm (Bresenham, 1965), but is straightforward

Midpoint Line Algorithm

- Assume: $x_0 \leq x_1$, if not then swap end points.
- $m = \dfrac{y_1 - y_0}{x_1 - x_0}$
- Consider the case $m \in (0, 1]$
  Other cases can be analogously derived.
- More *run* than *rise* for this case.
- Integer computations
- Draw the thinnest possible line that has no gaps.

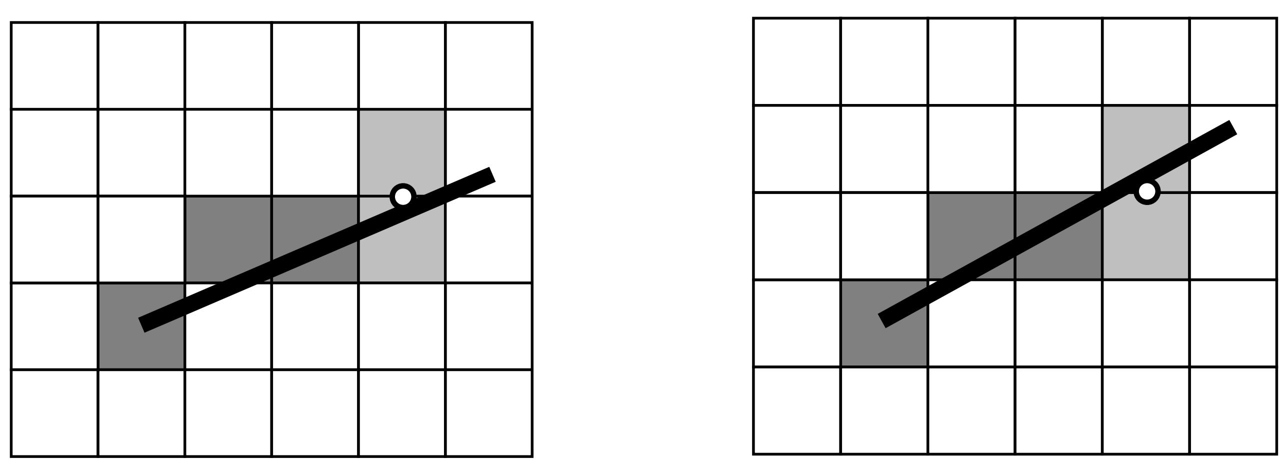
Other cases of
m \in (-inf, -1]
m \in (-1, 0]
m \in (1, inf]

Midpoint Line Algorithm

Basic Idea:

- Next potential candidates: (x+1, y) or (x+1, y+1)
- Midpoint: (x+1, y+0.5)
- If line passes below midpoint, then choose the bottom pixel, else choose the top pixel.

Points above the line are all positive



Midpoint Line Algorithm

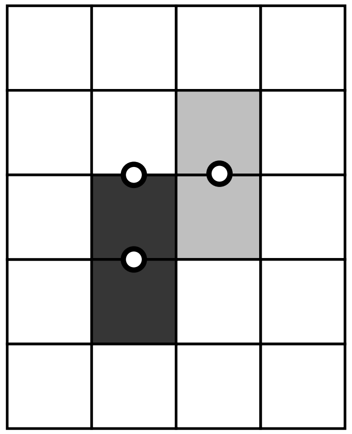
$y = y_0$
for $x = x_0$ *to* x_1 **do**
  draw (x, y)
  if $f(x+1, y+0.5) < 0$ **then**
    $y = y + 1$

- Need improvisations
- Integer computations

Midpoint Line Algorithm

- More efficient loop
- Reuse computation from previous step
- Inside the loop, one of these is already evaluated in last step:

$$f(x-1, y+0.5) \text{ or } f(x-1, y-0.5)$$



- Use the relationship:

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

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

Midpoint Line Algorithm

$y = y_0$

$d = 2(y_0 - y_1)(x_0 + 1) + (x_1 - x_0)(2y_0 + 1) +$
$\quad 2x_0 y_1 - 2x_1 y_0$ $\equiv$ $2f(x_0+1, y_0+0.5)$

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

  draw (x, y)

  if $d < 0$ **then**

    $y = y + 1$

    $d = d + 2(x_1 - x_0) + 2(y_0 - y_1)$

  else

    $d = d + 2(y_0 - y_1)$

$((y_1 \geq y_0)$ and $(x_1 - x_0 > y_1 - y_0)) \equiv (m \in [0, 1))$

Midpoint Line Algorithm

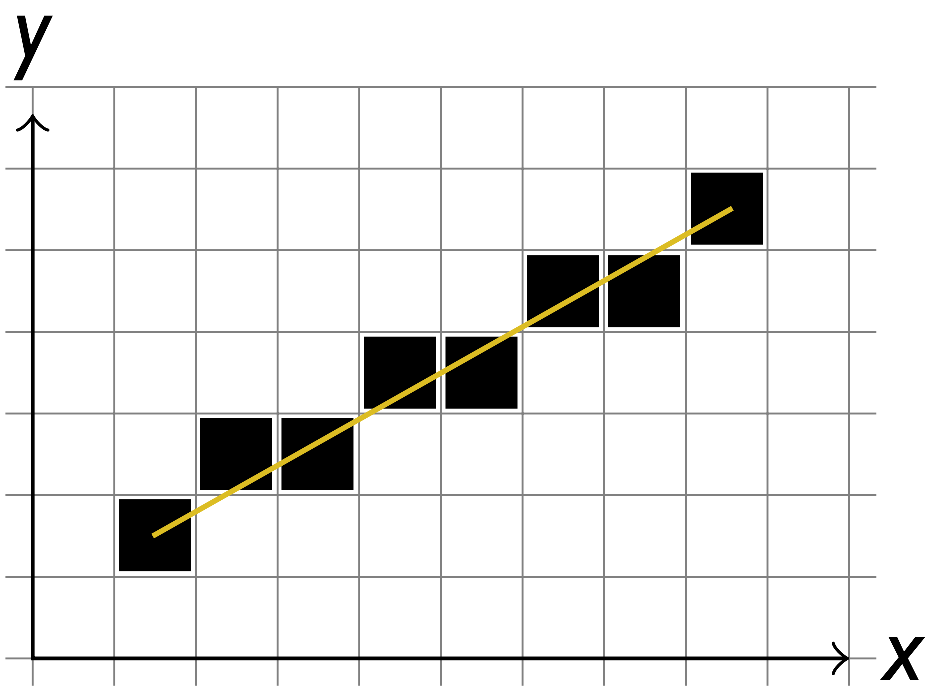
Example:

Rasterize and plot the line segment:

p_0: (1, 1)

p_1: (8, 5)

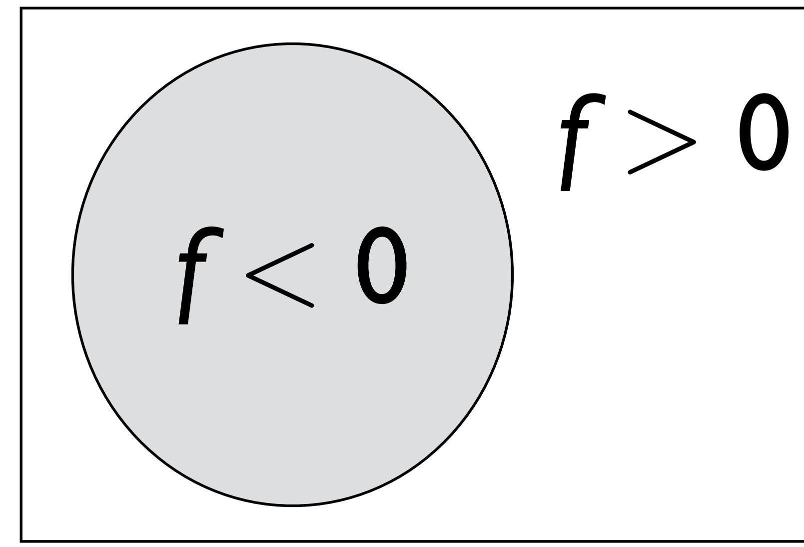
x	y
1	1
2	2
3	2
4	3
5	3
6	4
7	4
8	5



Rasterizing Circles

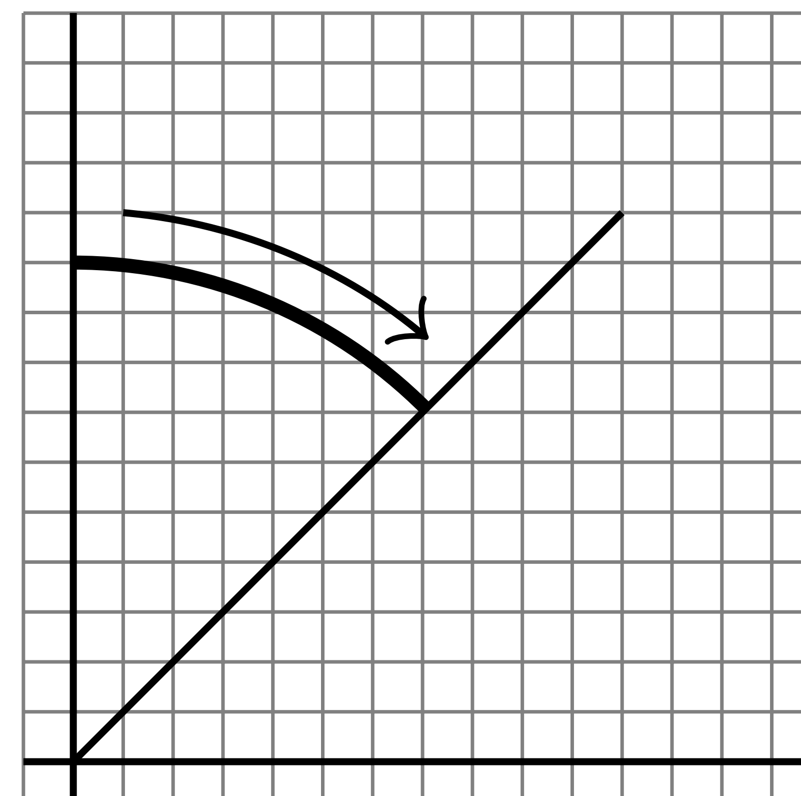
Midpoint Circle Algorithm

Circle: $f(x, y) \equiv x^2 + y^2 - r^2 = 0$



- Draw an octant of the circle and replicate using symmetry
- Do
$$x_{i+1} = x_i + 1$$
$$d_i \geq 0 : y_{i+1} = y_i - 1$$
$$d_i < 0 : y_{i+1} = y_i$$

where $$d_i = f\left(x_i + 1, y_i - \frac{1}{2}\right)$$



Midpoint Circle Algorithm

Determine d_i

$$d_{i+1} - d_i$$

$$= f(x_i + 2, y_{i+1} - ½) - f(x_i + 1, y_i - ½)$$

$$= (x_i + 2)^2 + (y_{i+1} - ½)^2 - r^2 - (x_i + 1)^2 - (y_i - ½)^2 + r^2$$

$$= 2x_i + 3 + (y_{i+1}^2 - y_{i+1}) - (y_i^2 - y_i)$$

$$y_{i+1} = y_i \qquad \Longrightarrow \quad d_{i+1} - d_i = 2x_i + 3$$

$$y_{i+1} = y_i - 1 \qquad \Longrightarrow \quad d_{i+1} - d_i = 2x_i - 2y_i + 5$$

Midpoint Circle Algorithm

Initialization

- Compute

$$x_0 = 0, y_0 = r$$

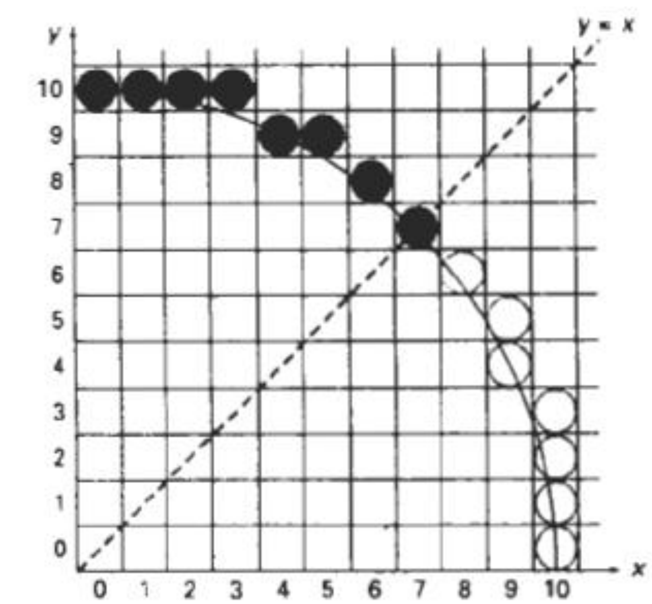
$$d_0 = \left(1, r - \tfrac{1}{2}\right) = 1 + \left(r - \tfrac{1}{2}\right)^2 - r^2$$

$$1 + r^2 - r + \tfrac{1}{4} - r^2 = \tfrac{5}{4} - r$$

- Integer computation $d_0 = \text{round}\left(\tfrac{5}{4} - r\right)$
- If r is integer value $d_0 = 1 - r$

Midpoint Circle Algorithm

$(x, y) = (0, r)$
$d = \text{round}\left(\frac{5}{4} - r\right)$
while $y > x$ **do**
    $\text{draw\_all\_octants}\,(x, y)$
    $x = x + 1$
    if $d \geq 0$ **then**
        $y = y - 1$
        $d = d + 2x - 2y + 5$
    else
        $d = d + 2x + 3$



Rasterizing Triangles

Triangle Rasterization

Drawing a 2D triangle with points

$$P_0(x_0, y_0), P_1(x_1, y_1), \text{ and } P_2(x_2, y_2)$$

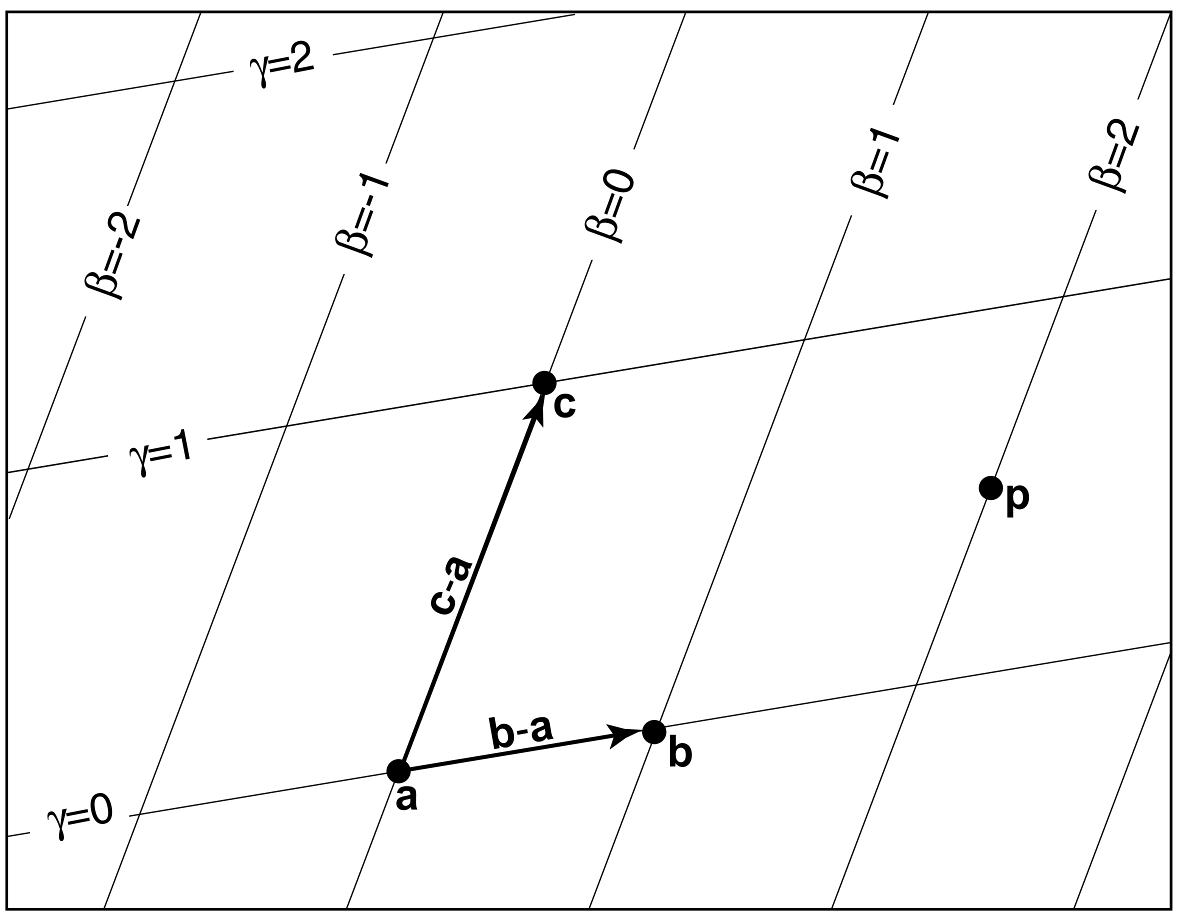
- Color interpolation† using *Barycentric coordinates* $c = \alpha c_0 + \beta c_1 + \gamma c_2$
- Rasterize adjacent triangles so that there are no holes

† Gouraud interpolation (Gouraud, 1971)

Barycentric Coordinates

$$p = \alpha a + \beta b + \gamma c$$

$$0 \leq \alpha, \beta, \gamma \leq 1$$



$$\beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)}$$

$$\gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)}$$

$$\alpha = 1 - \beta - \gamma$$

$$f_{ab}(x, y) \equiv (y_a - y_b)x + (x_b - x_a)y + x_a y_b - x_b y_a$$

Triangle Rasterization

$\{x_{min}, y_{min}, x_{max}, y_{max}\} = \text{bbox}(P_0, P_1, P_2)$

for $y = y_{min}$ *to* y_{max} **do**

  for $x = x_{min}$ *to* x_{max} **do**

$$\alpha = \frac{f_{12}(x, y)}{f_{12}(x_0, y_0)}$$

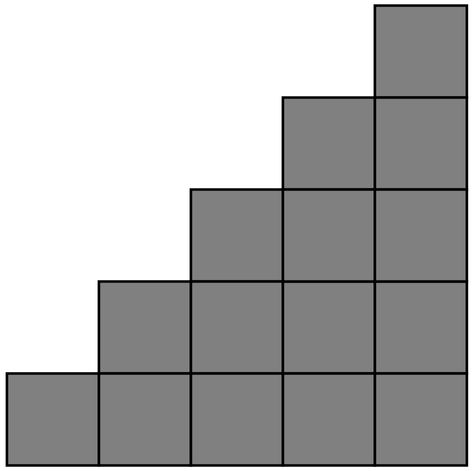
$$\beta = \frac{f_{20}(x, y)}{f_{20}(x_1, y_1)}$$

$$\gamma = 1 - \alpha - \beta$$

  if $(\alpha > 0 \wedge \beta > 0 \wedge \gamma > 0)$ **then**

   $c = \alpha c_0 + \beta c_1 + \gamma c_2$

   drawpixel(x, y) with color *c*



Triangle Rasterization

What about pixels on the boundary?

- In order to make sure no gaps remain between adjacent triangles,
  - Either choose to draw the edge by one of the triangles
  - Or draw the edge twice by both triangles.

Reading

- FCG: 2.7, 8.1.1, 8.1.2

ICG: Interactive Computer Graphics, E. Angel, and D. Shreiner, 6th ed.
FCG: Fundamentals of Computer Graphics, P. Shirley, M. Ashikhmin, and S. Marschner, 3rd ed.