CS130 : Computer Graphics

Lecture 5: Rasterizing Triangles

Tamar Shinar
Computer Science & Engineering
UC Riverside

What is rasterization?

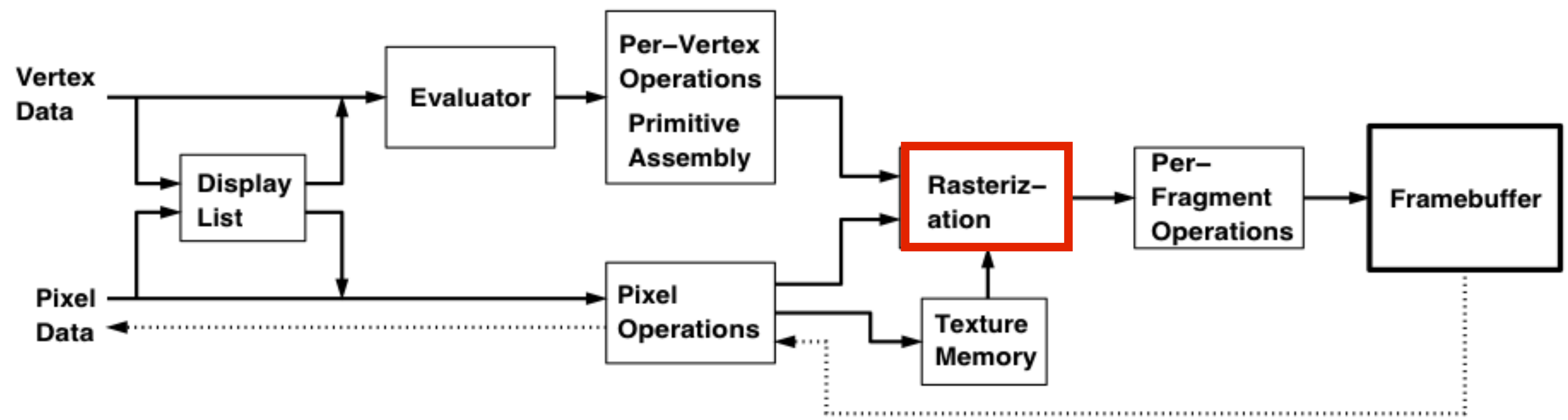


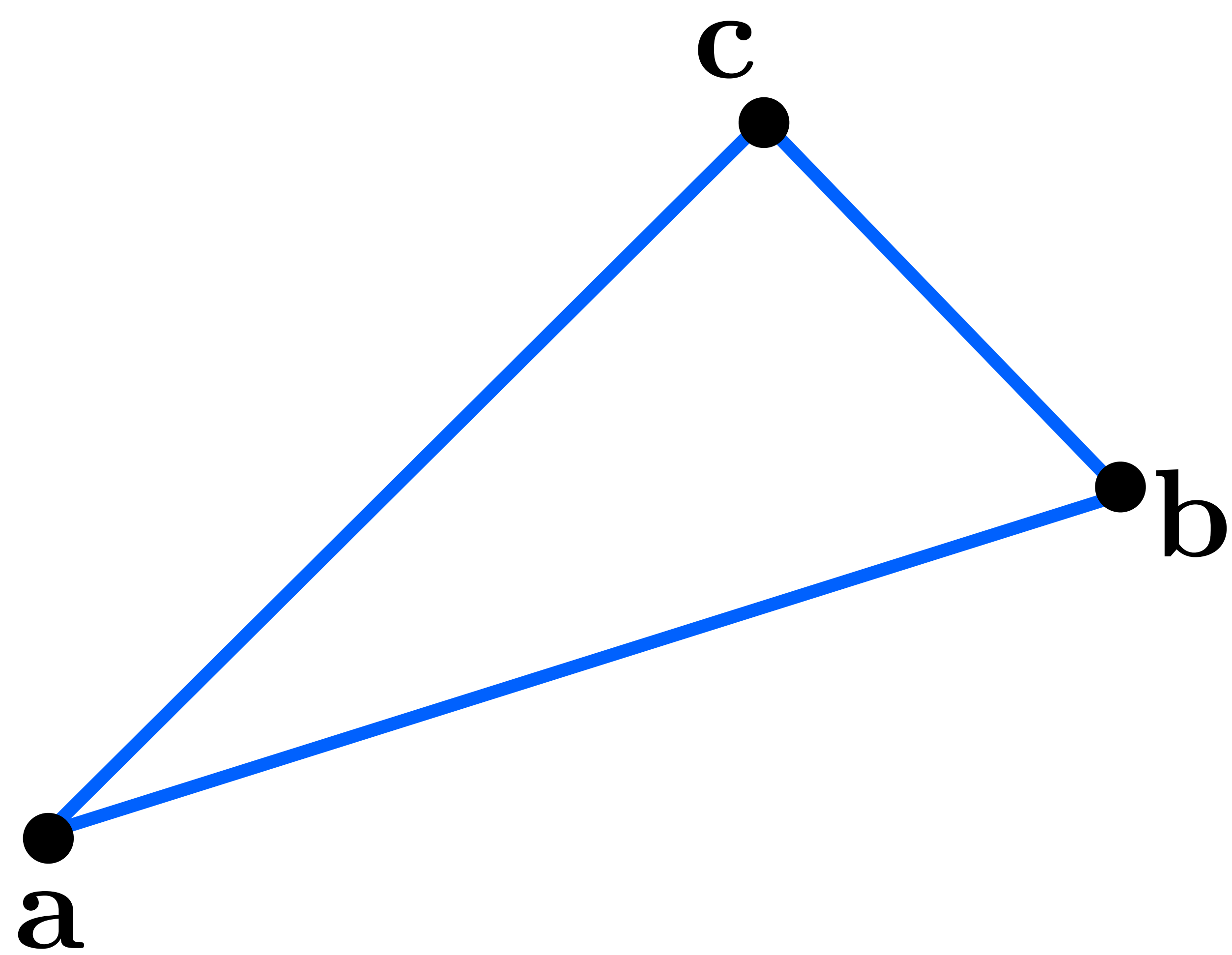
Figure 1. Block diagram of OpenGL.

- input: primitives, output: fragments
- enumerate the pixels covered by a primitive
- interpolate attributes across the primitive

– **output** 1 fragment per pixel covered by the primitive

Triangles

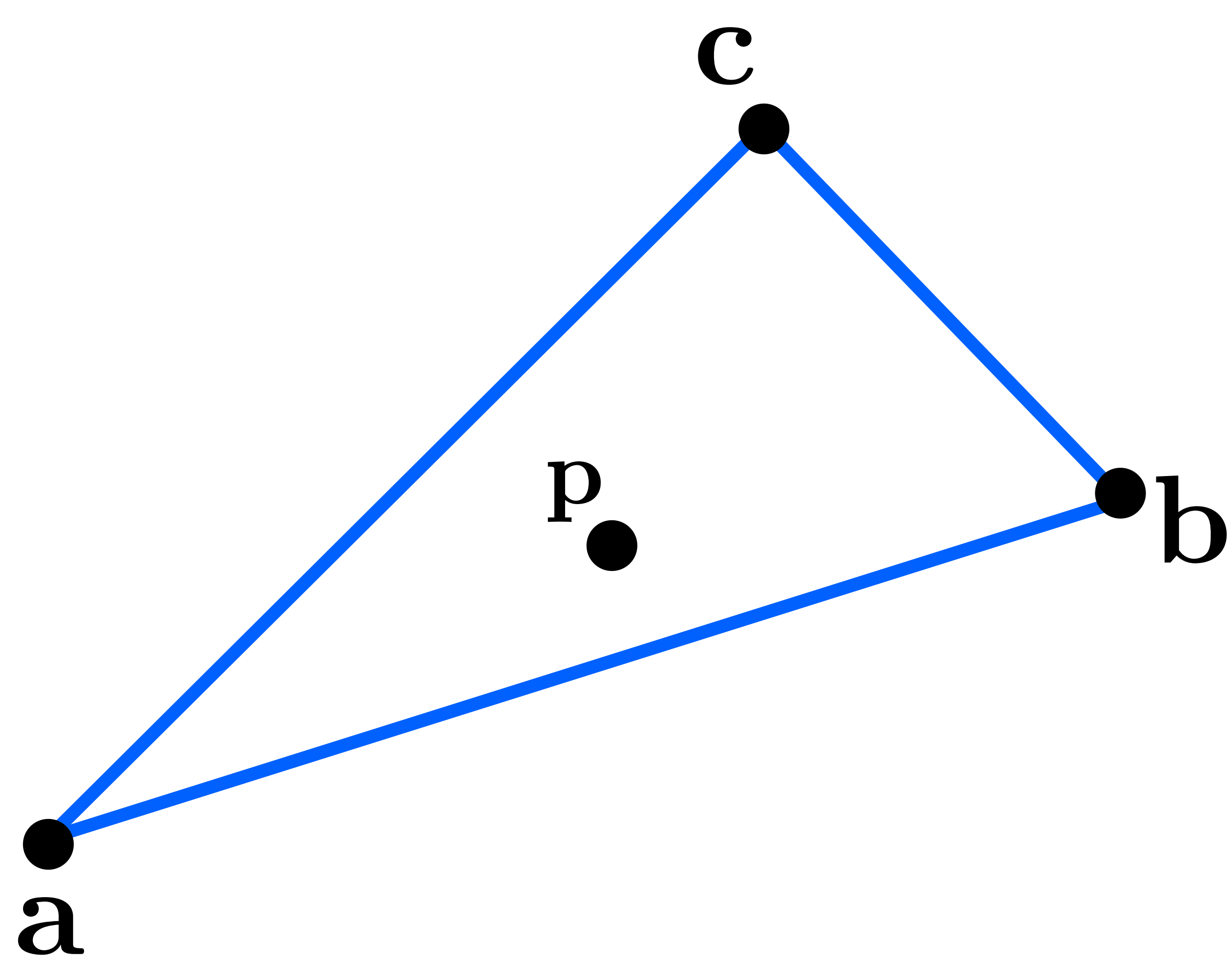
barycentric coordinates



barycentric coordinates

$$\mathbf{p} = f(\mathbf{a}, \mathbf{b}, \mathbf{c})$$

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

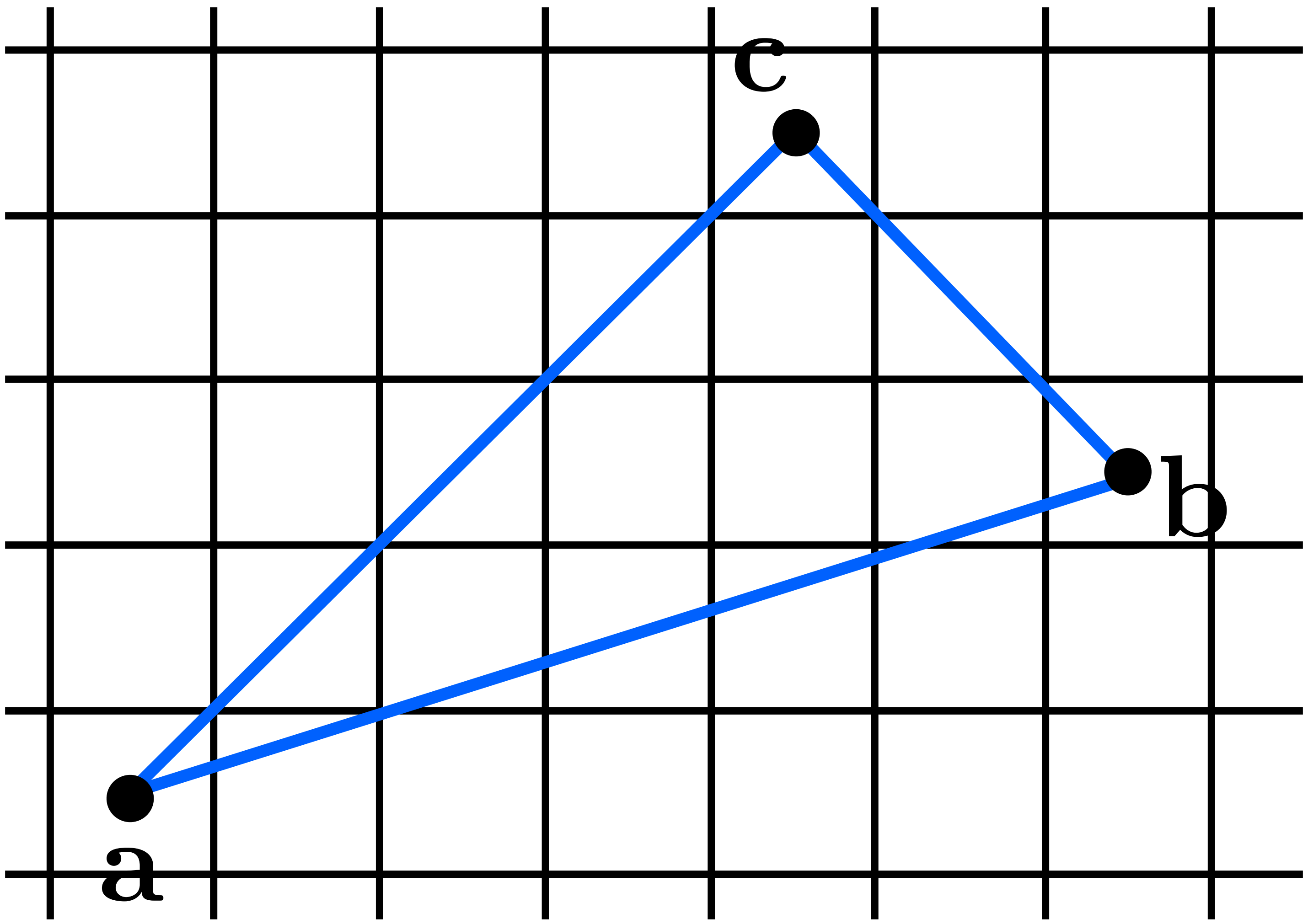


What are (α, β, γ) ?

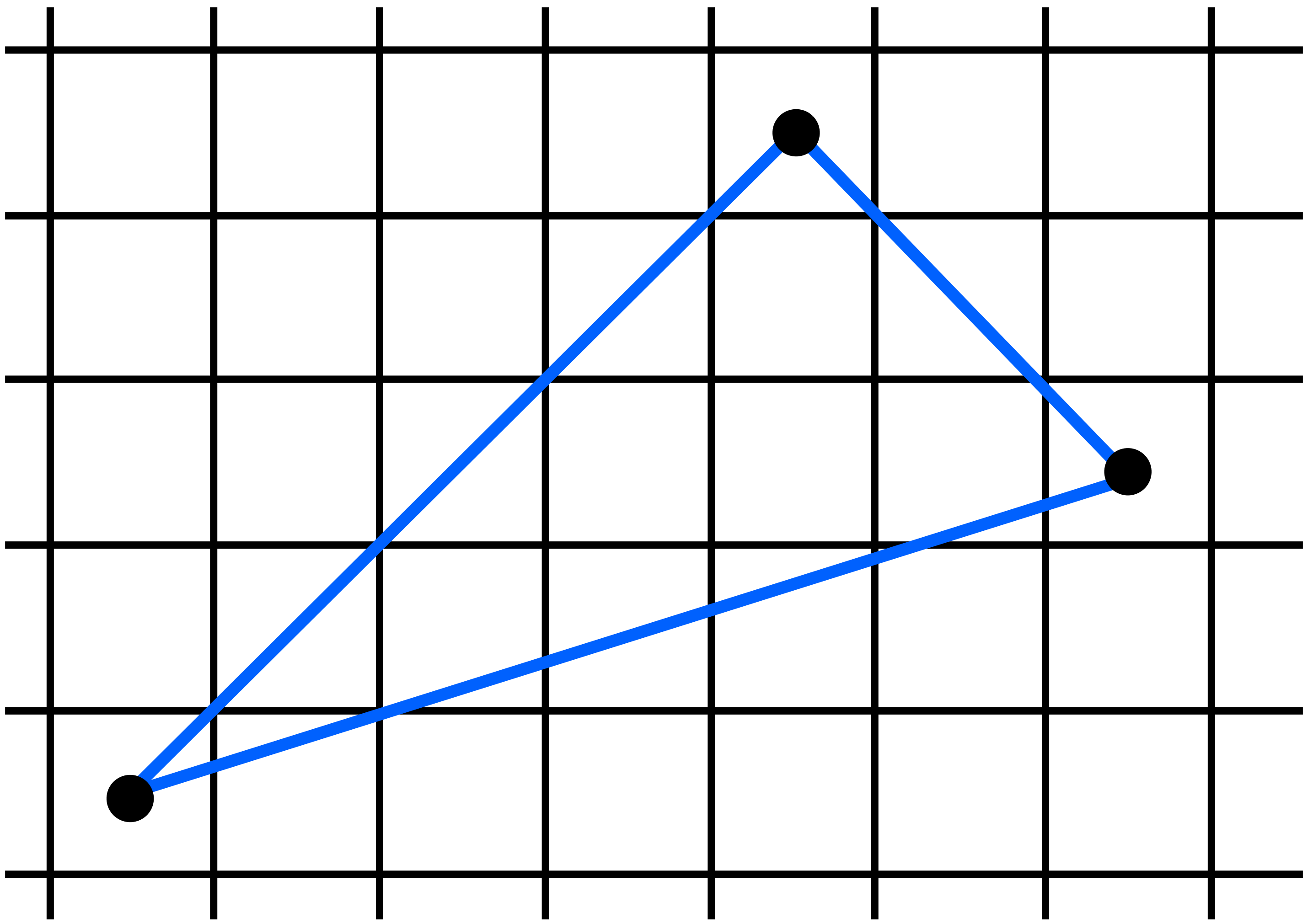
<whiteboard>

Triangle rasterization

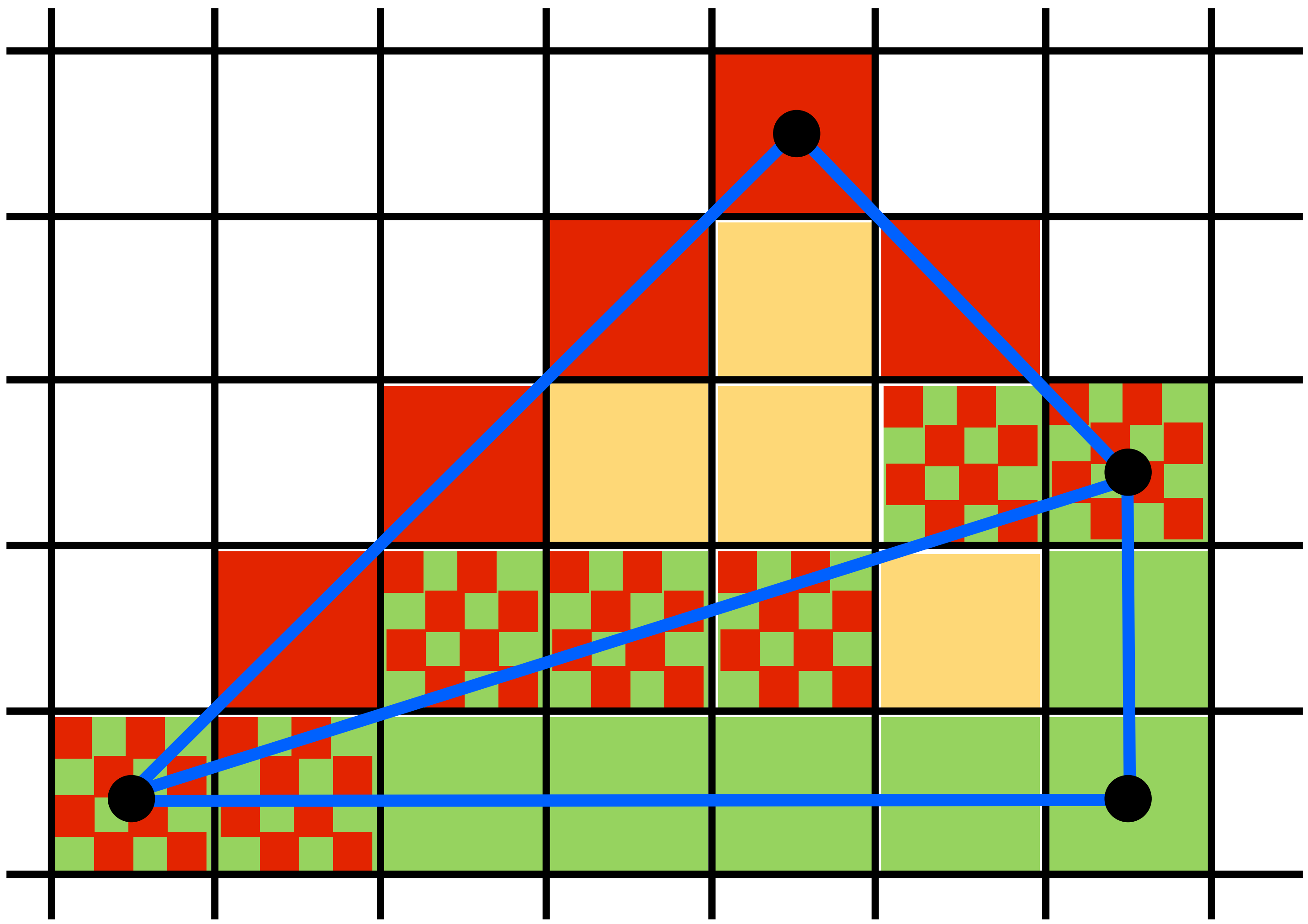
Which pixels should be used to approximate a triangle?



Triangle rasterization issues



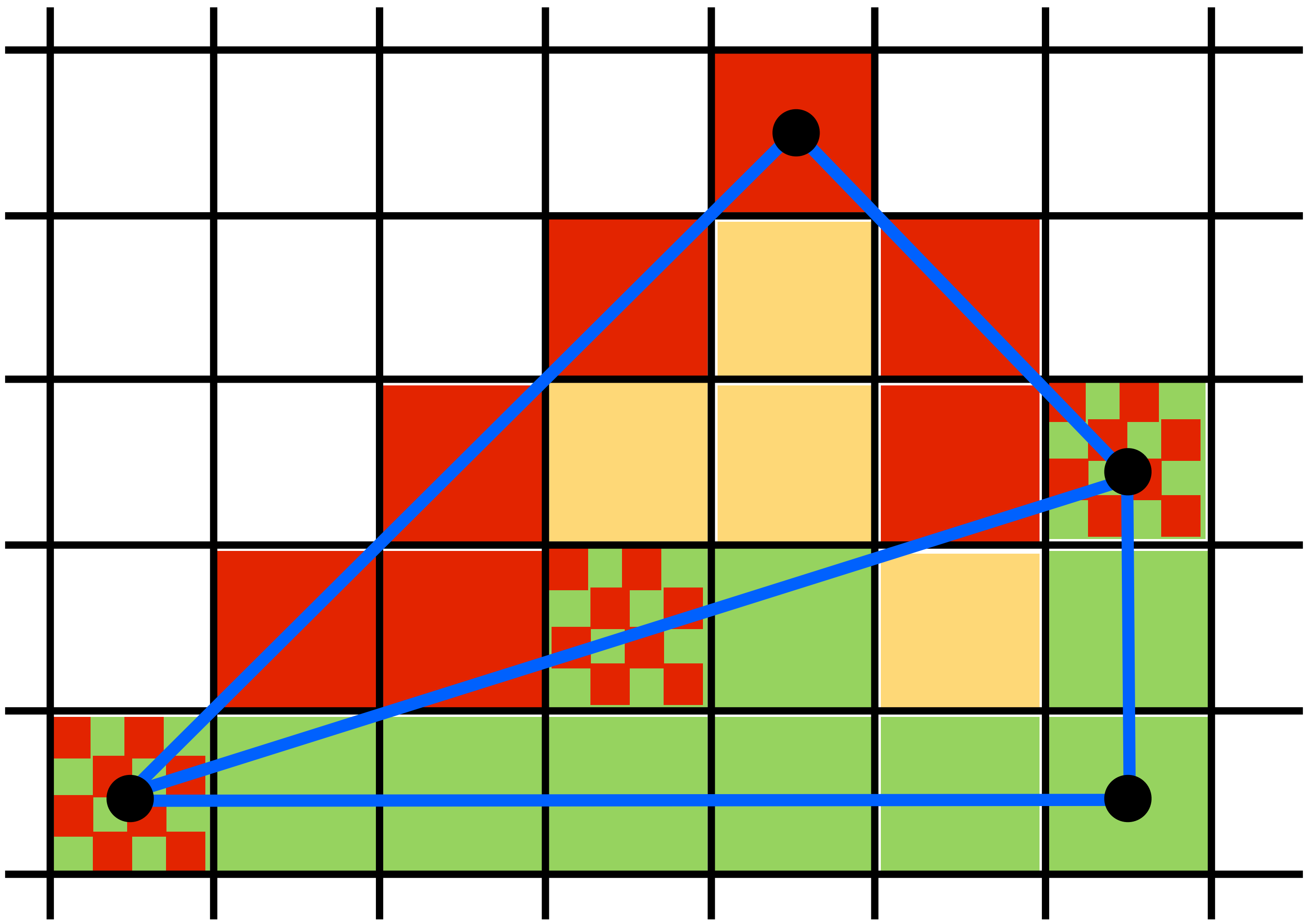
Which pixels should be used to approximate a triangle?



Who should fill in shared edge?

but who should fill in pixels for a shared edge?

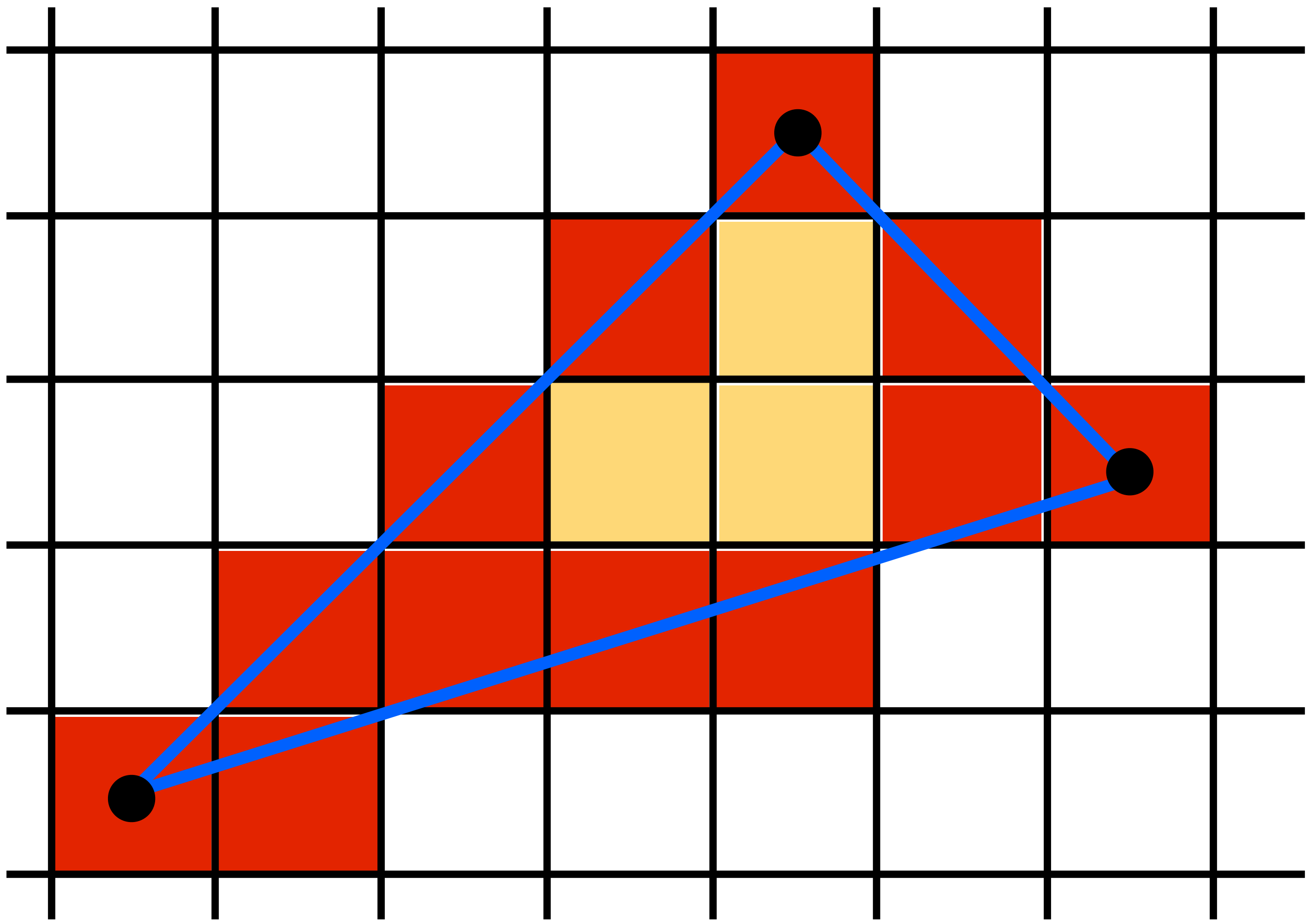
Which pixels should be used to approximate a triangle?



Who should fill in shared edge?

give to triangle that contains pixel center
– but we have some **ties**
why can't neither/both triangles draw the pixel?
 neither: gaps
 both: indeterminacy (due to indeterminate drawing order), incorrect, e.g., if both triangles are partially transparent
we want a **unique** assignment

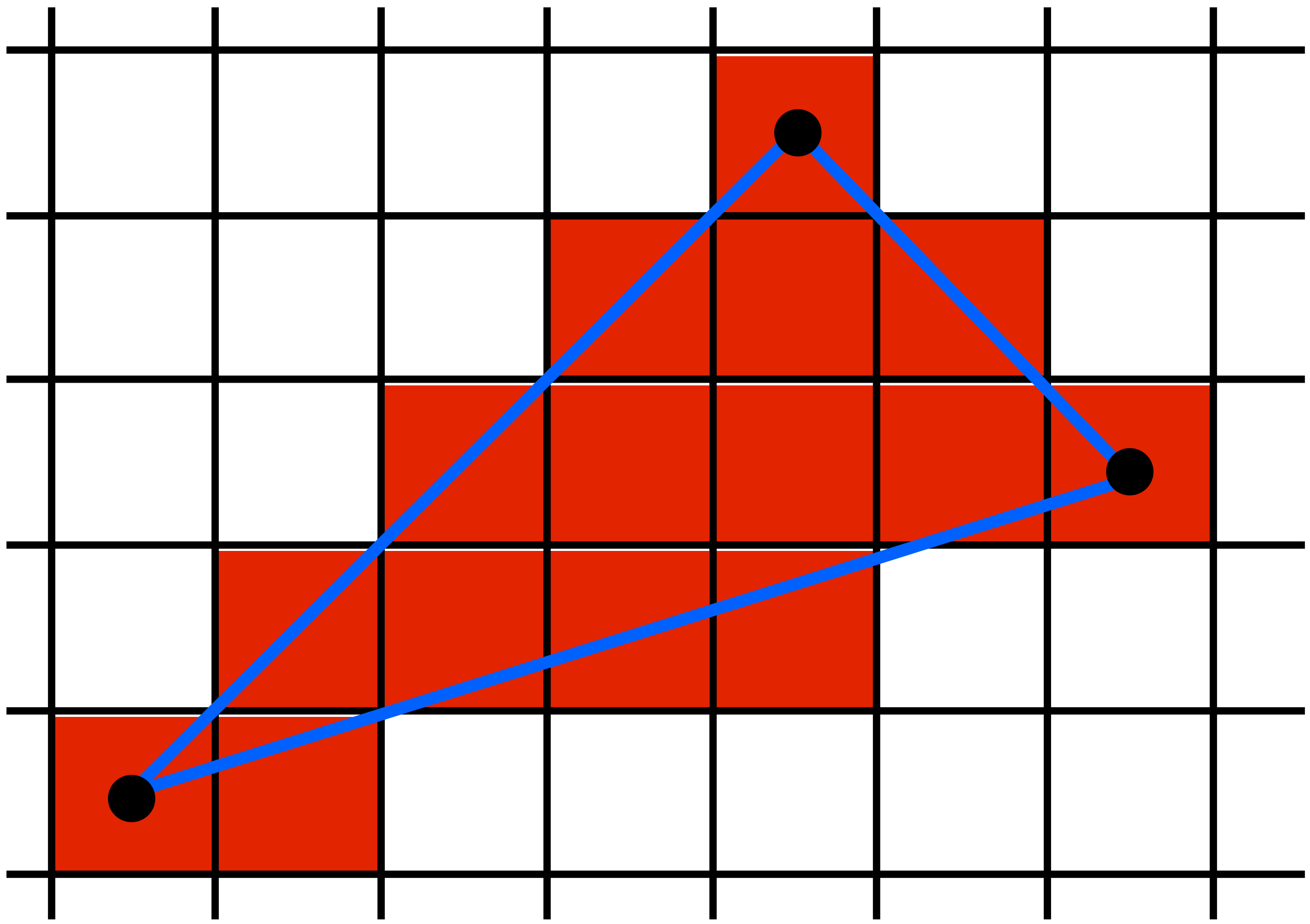
Which pixels should be used to approximate a triangle?



Use Midpoint Algorithm for edges and fill in?

That could be one possibility but we use a different approach based on barycentric coordinates

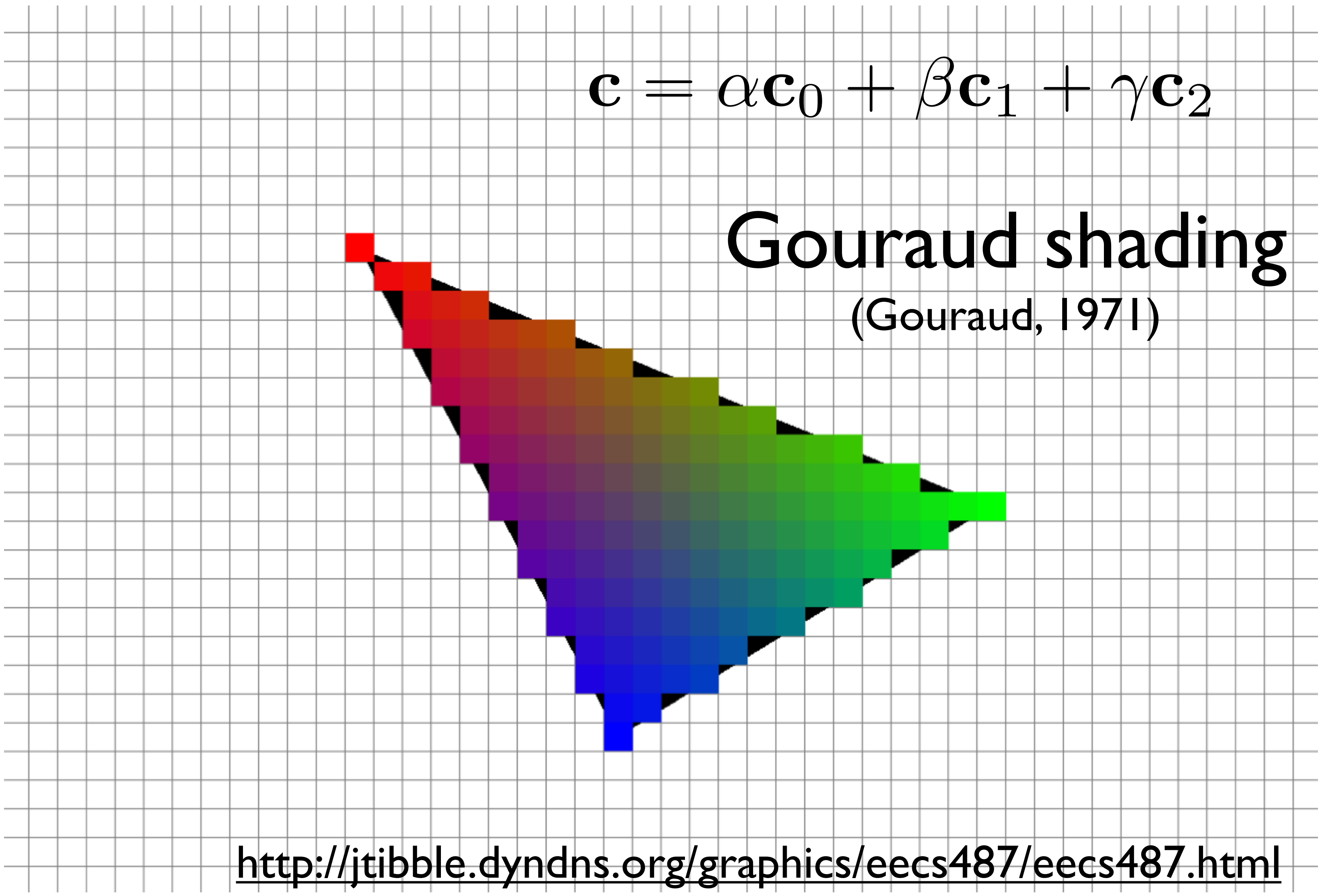
Which pixels should be used to approximate a triangle?



Use an approach based on barycentric coordinates

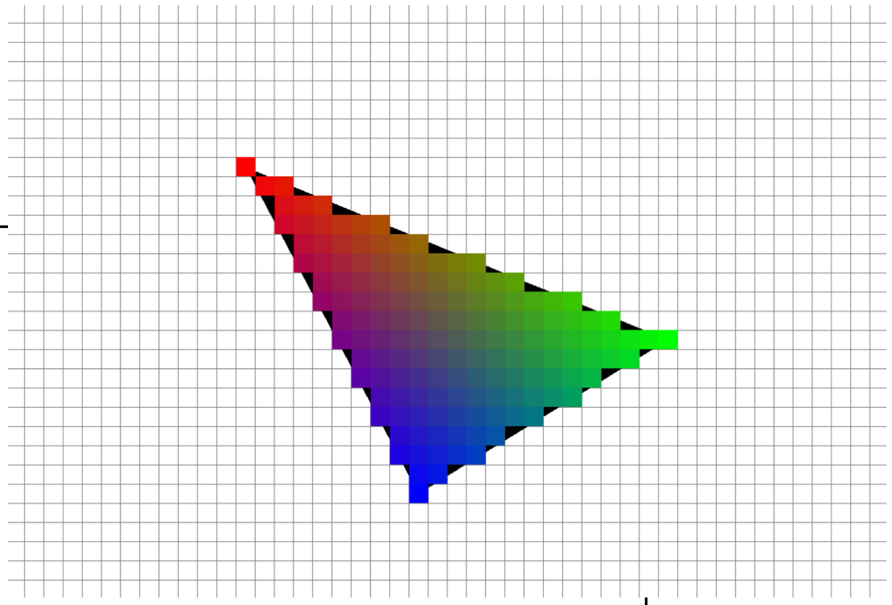
For each pixel, we compute its barycentric coordinates
If the coordinates are all >= 0, then the pixel is covered by the triangle

We can interpolate attributes using barycentric coordinates



Using barycentric coordinates also has the advantage that we can easily interpolate colors or other attributes from triangle vertices

Triangle rasterization algorithm



for all x do

    for all y do

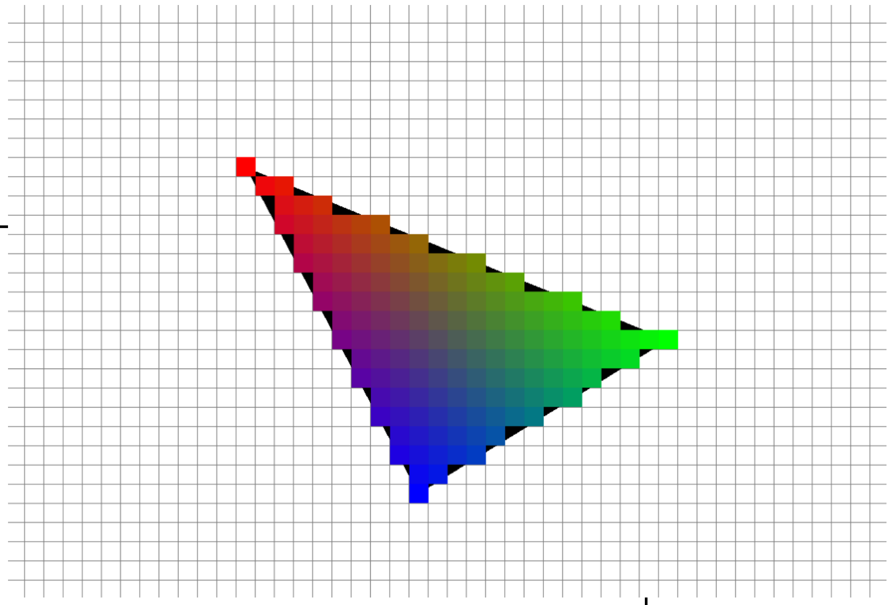
        compute (α, β, γ) for (x,y)

        if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ then

            $\mathbf{c} = \alpha \mathbf{c}_0 + \beta \mathbf{c}_1 + \gamma \mathbf{c}_2$

            drawpixel(x,y) with color c

Triangle rasterization algorithm

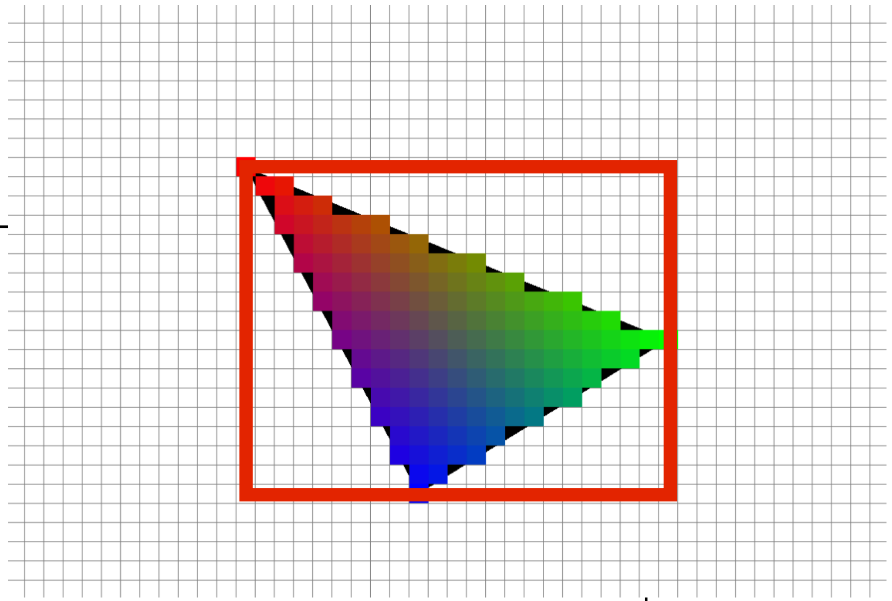


for all x do
    for all y do
        compute (α, β, γ) for (x,y)
        if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ then
            $\mathbf{c} = \alpha \mathbf{c}_0 + \beta \mathbf{c}_1 + \gamma \mathbf{c}_2$
            drawpixel(x,y) with color c

the rest of the algorithm is to make the steps in red more **efficient**

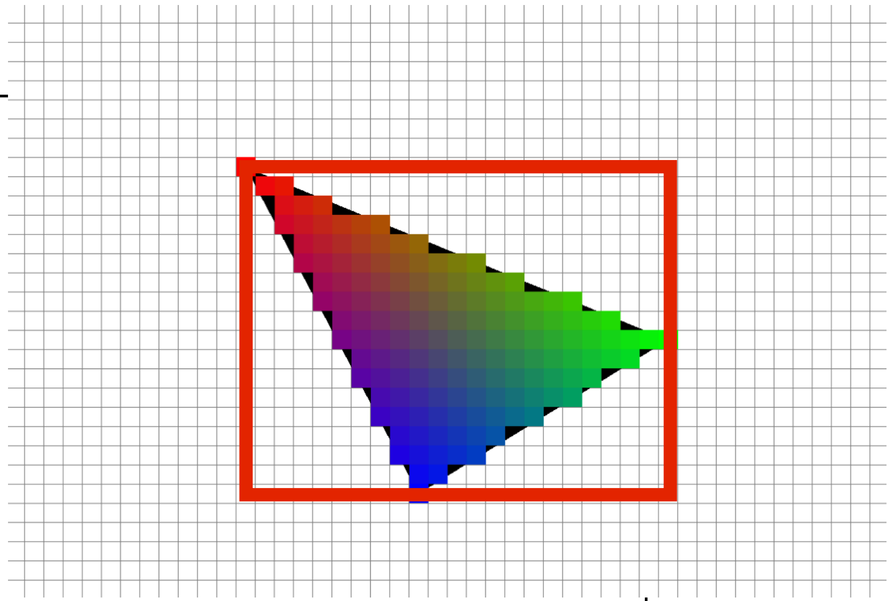
Triangle rasterization algorithm

use a bounding rectangle



for x in [x_min, x_max]
    for y in [y_min, y_max]
        compute (α, β, γ) for (x,y)
        if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ then
            $\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$
            drawpixel(x,y) with color c

Triangle rasterization algorithm



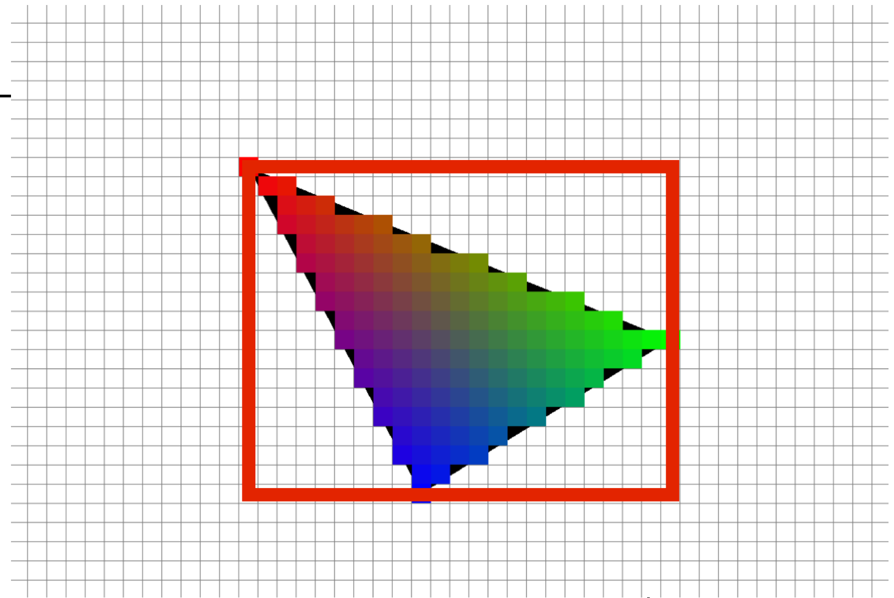
for x in [x_min, x_max]
    for y in [y_min, y_max]
        $\alpha = f_{bc}(x,y)/f_{bc}(x_a,y_a)$
        $\beta = f_{ac}(x,y)/f_{ac}(x_b,y_b)$
        $\gamma = f_{ab}(x,y)/f_{ab}(x_c,y_c)$
        if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ then
            $\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$
            drawpixel(x,y) with color c

<whiteboard>

<whiteboard> : computing alpha, beta, and gamma

Triangle rasterization algorithm

Optimizations?



for x in [x_min, x_max]

  for y in [y_min, y_max]

$$\alpha = f_{bc}(x,y)/f_{bc}(x_a,y_a)$$

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

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

if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ **then**

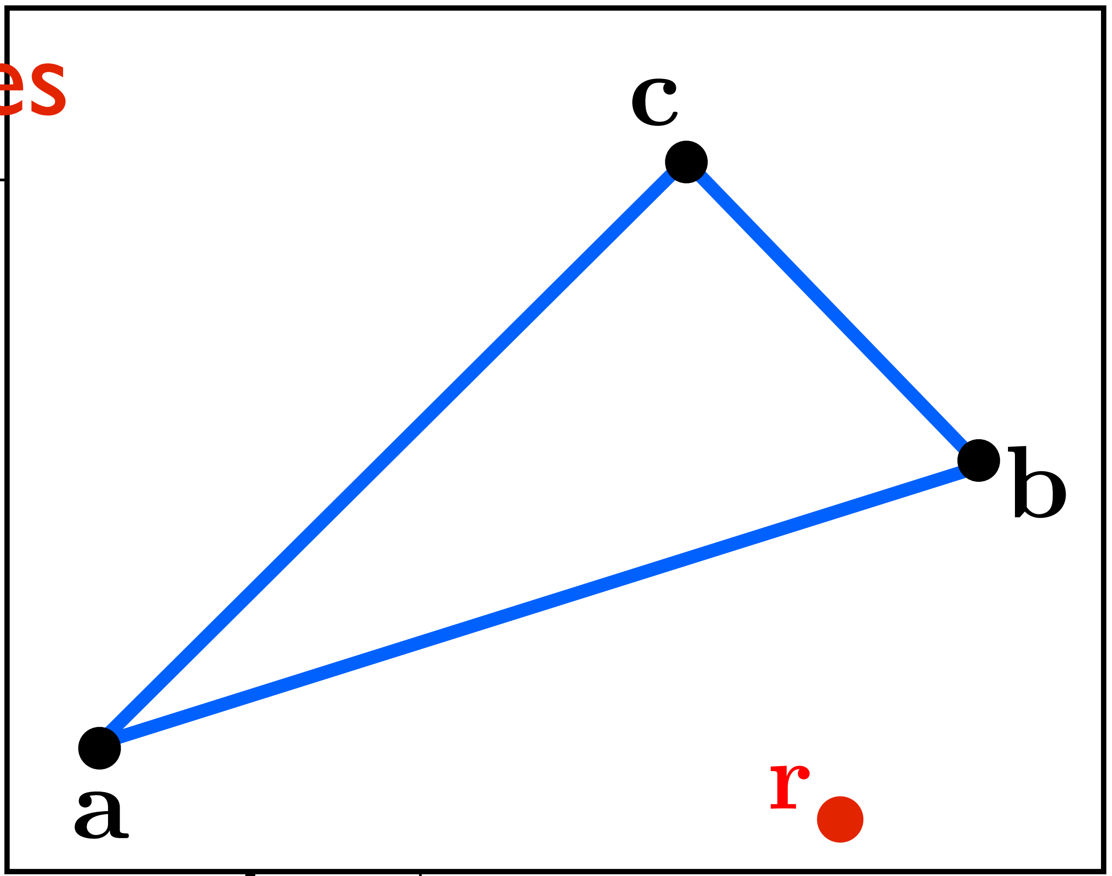
$$\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$$

drawpixel(x,y) with color c

1. can make computation of bary. coords. **incremental**
- f(x,y) = Ax+By+C
- f(x+1,y) = f(x,y) + A
2. **color** computation can also be made **incremental**
3. **alpha > 0 and beta > 0 and gamma > 0** (if true => they are also less than one)

Triangle rasterization algorithm

dealing with shared triangle edges



for x in [x_min, x_max]

for y in [y_min, y_max]

$$\alpha = f_{bc}(x,y)/f_{bc}(x_a,y_a)$$

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

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

if $(\alpha \geq 0 \text{ and } \beta \geq 0 \text{ and } \gamma \geq 0)$ then

if $(\alpha > 0 \text{ or } f_{12}(\mathbf{p}_0)f_{12}(\mathbf{r}) > 0)$ and
$(\beta > 0 \text{ or } f_{20}(\mathbf{p}_1)f_{20}(\mathbf{r}) > 0)$ and
$(\gamma > 0 \text{ or } f_{01}(\mathbf{p}_2)f_{01}(\mathbf{r}) > 0)$ then

$$\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$$

drawpixel(x,y) with color c

- compute f_12(r), f_20(r) and f_01(r) and make sure r doesn't hit a line