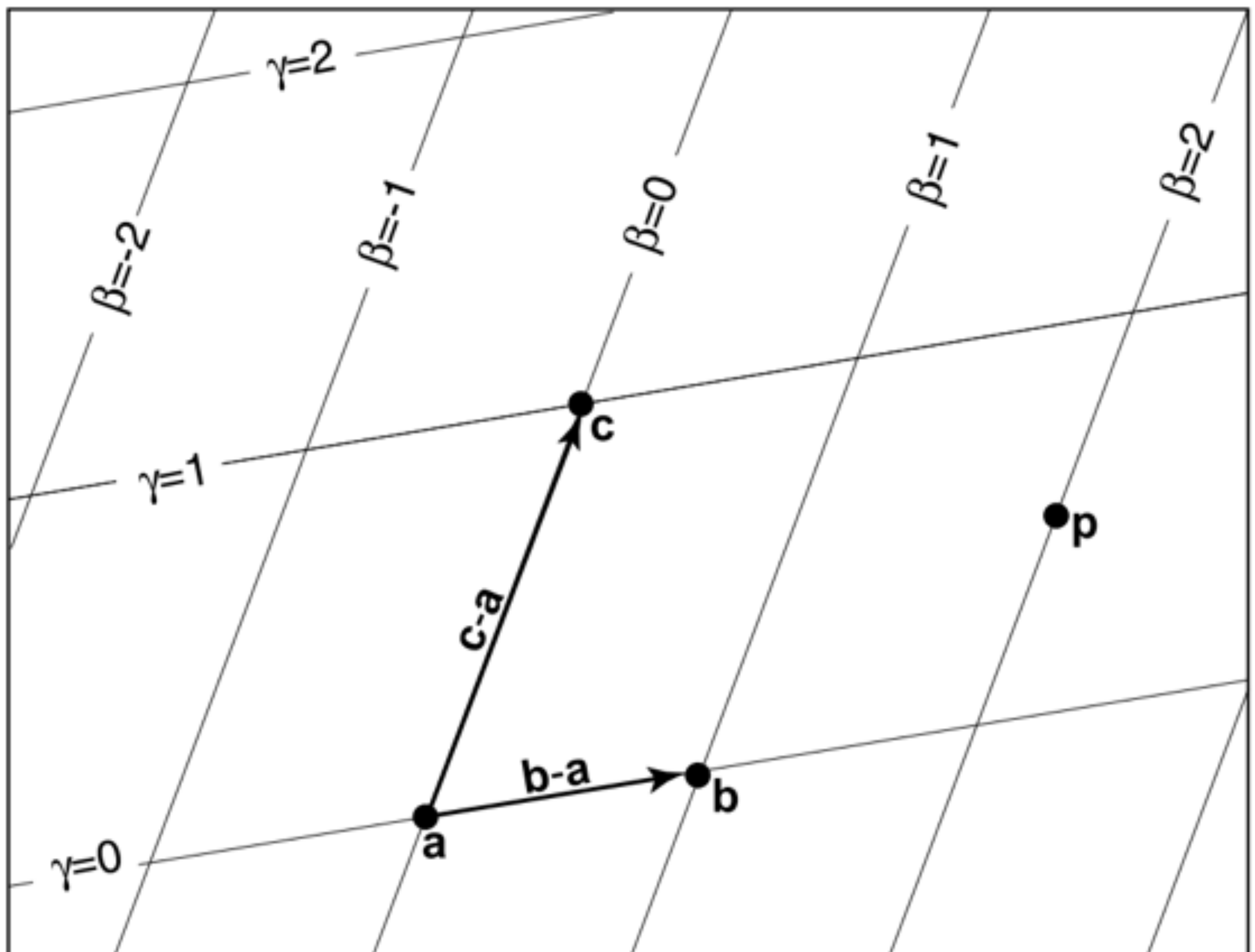
Triangles

barycentric coordinates

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



barycentric coordinates

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

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

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

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

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

barycentric coordinates

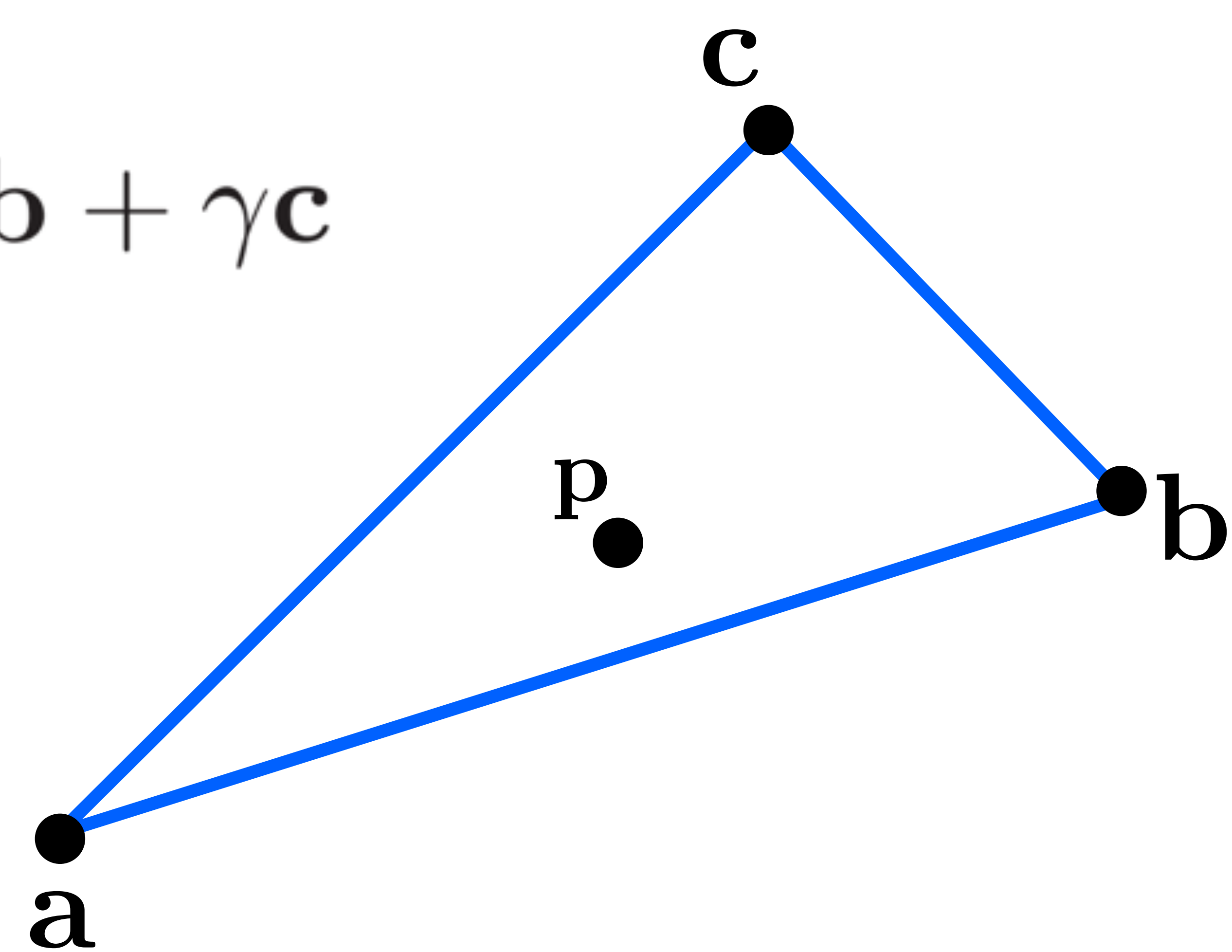
If **p** <u>inside the triangle,</u>

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

$$0 < \alpha < 1$$

$$0 < \beta < 1$$

$$0 < \gamma < 1.$$

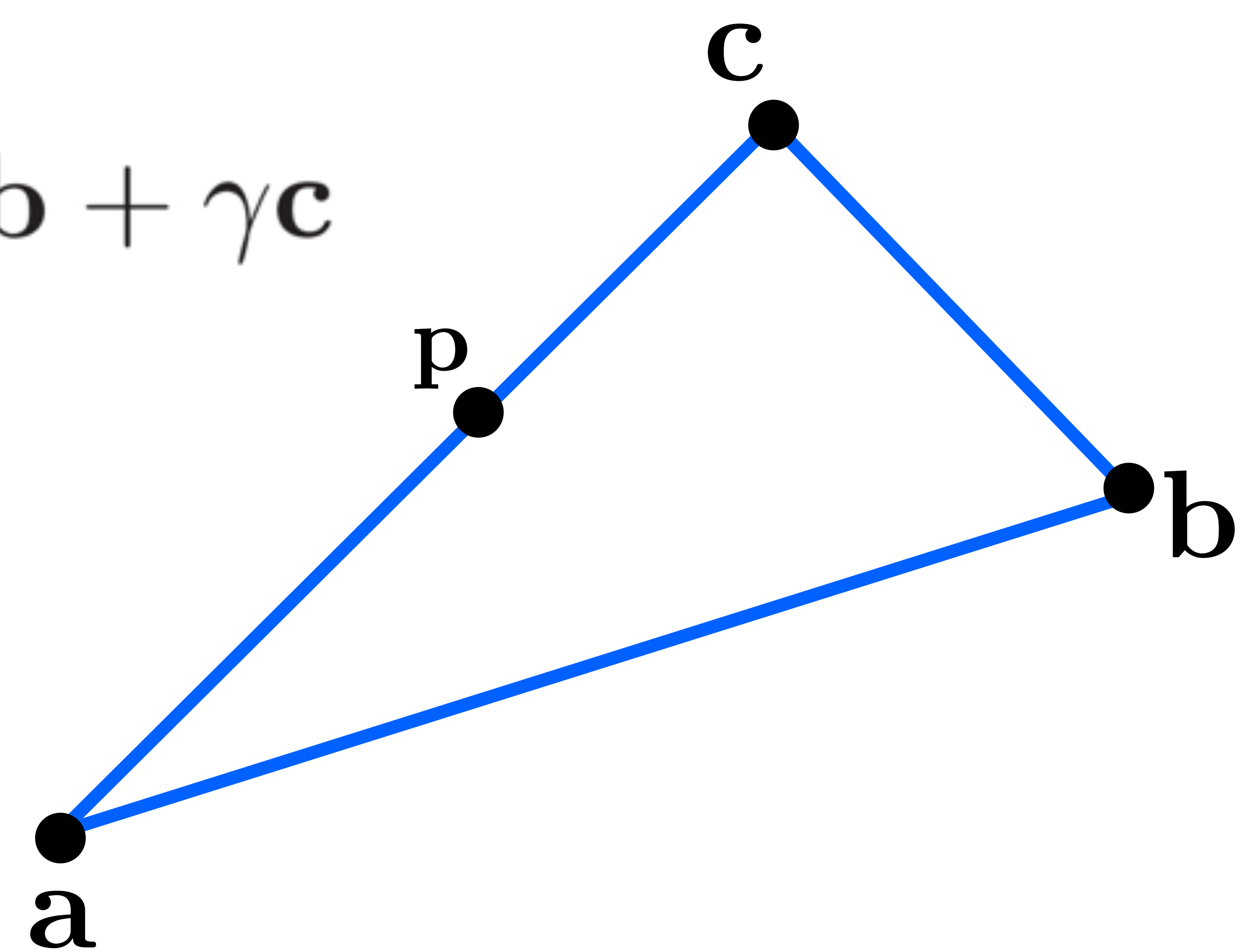


barycentric coordinates

If **p** <u>on an edge</u>, e.g.,

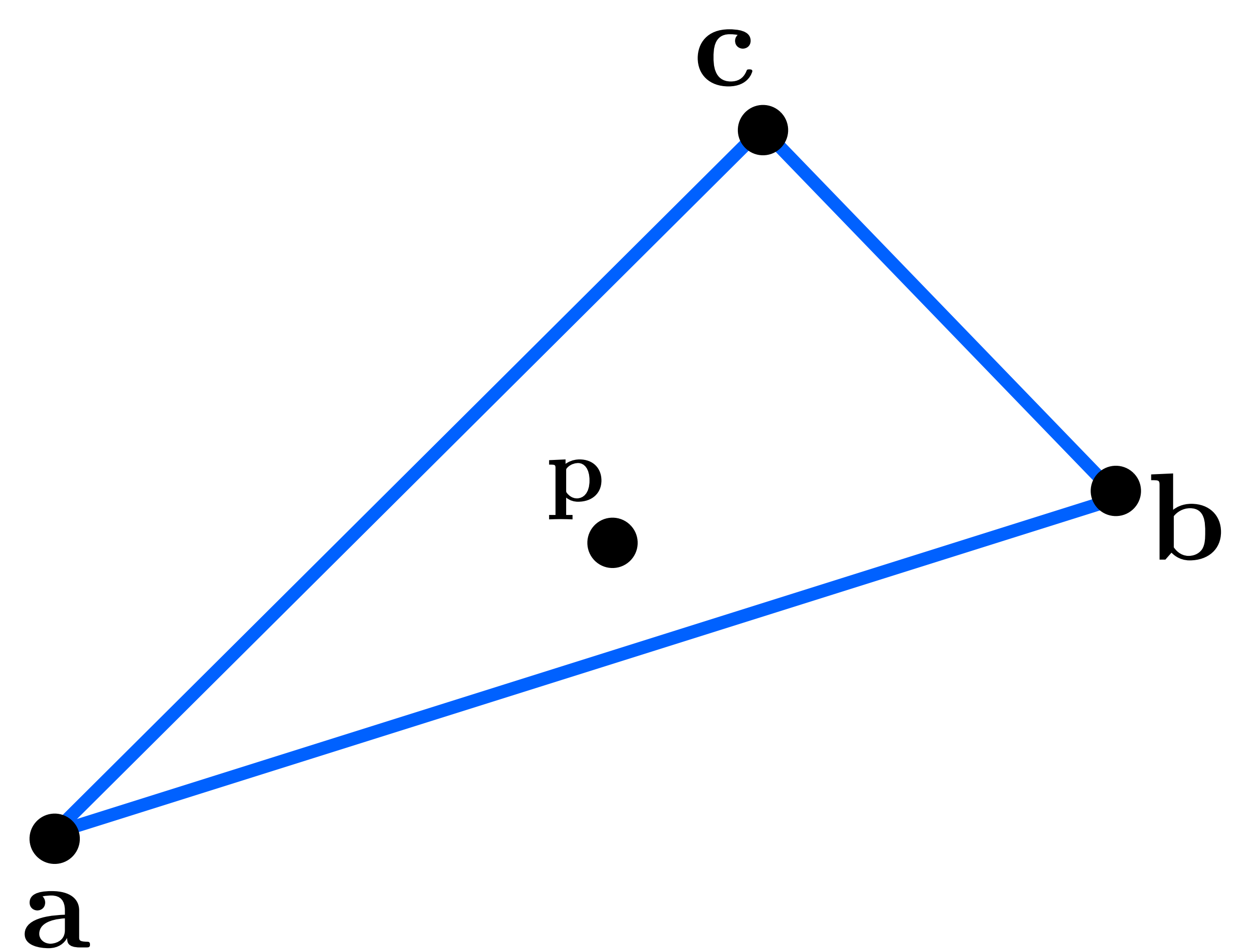
$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

$$\beta = 0$$
$$\alpha + \gamma = 1$$



barycentric coordinates

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

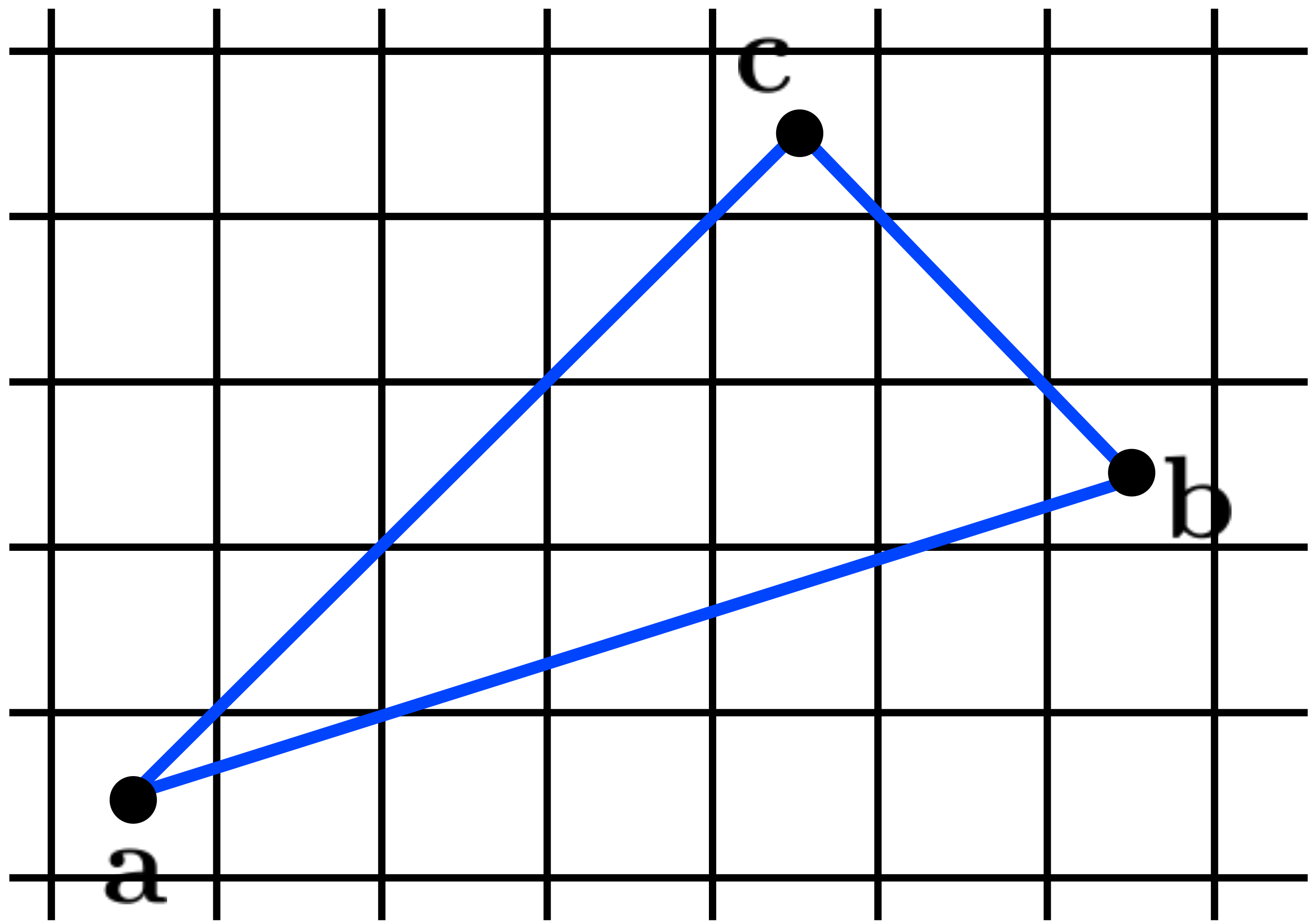


What are (α, β, γ) ?

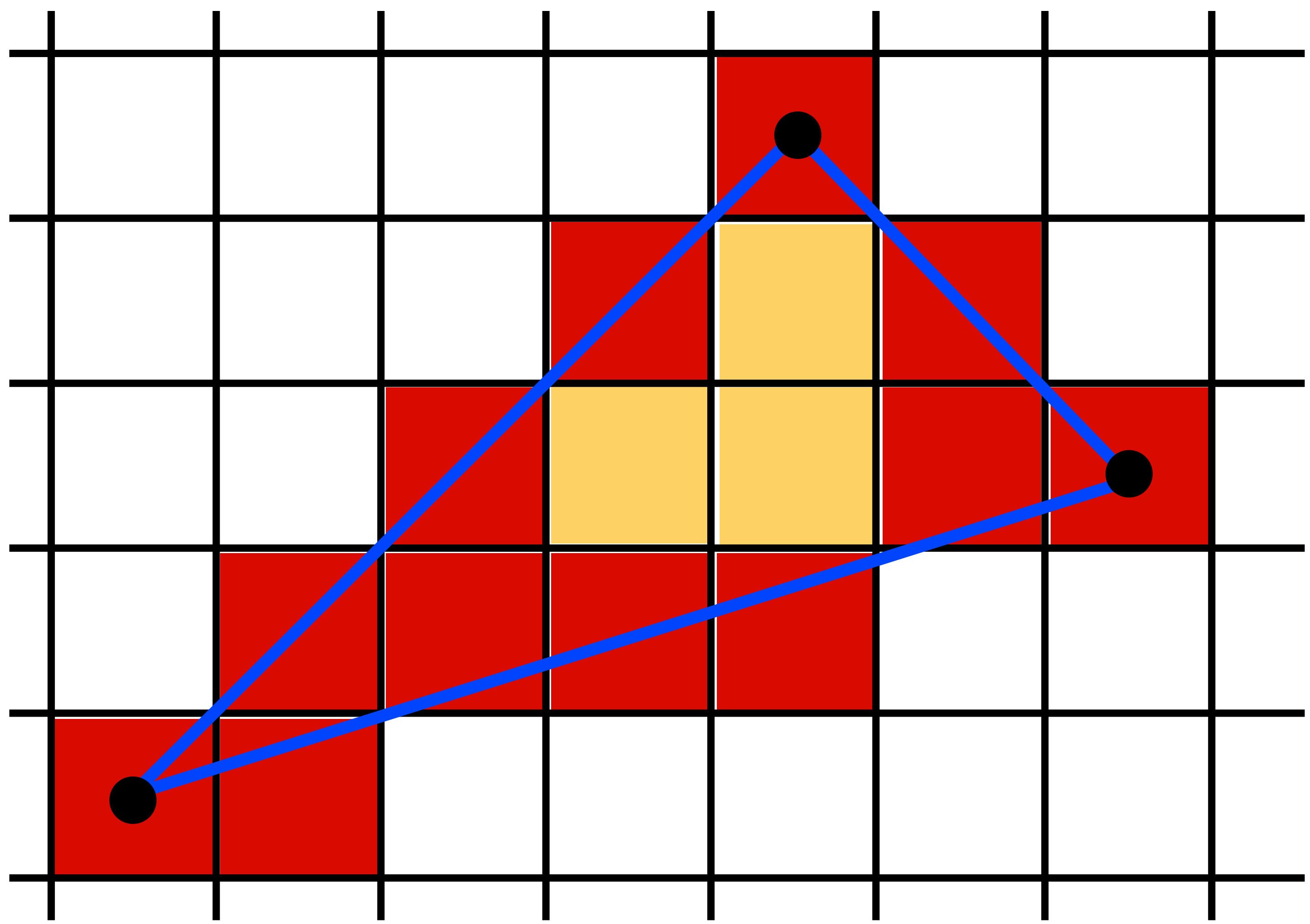
<whiteboard>

Triangle rasterization

Which pixels should be used to approximate a triangle?

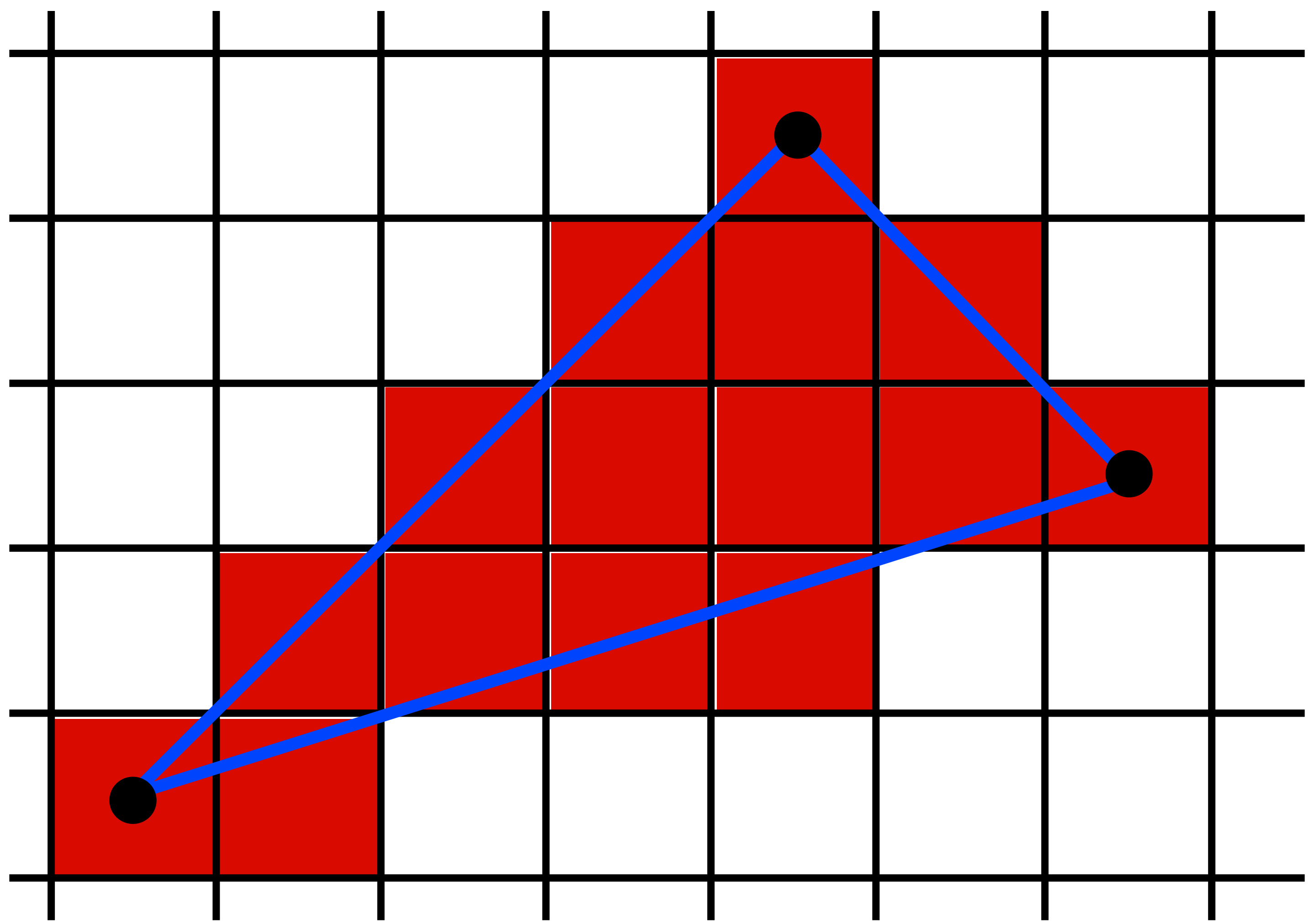


Which pixels should be used to approximate a triangle?



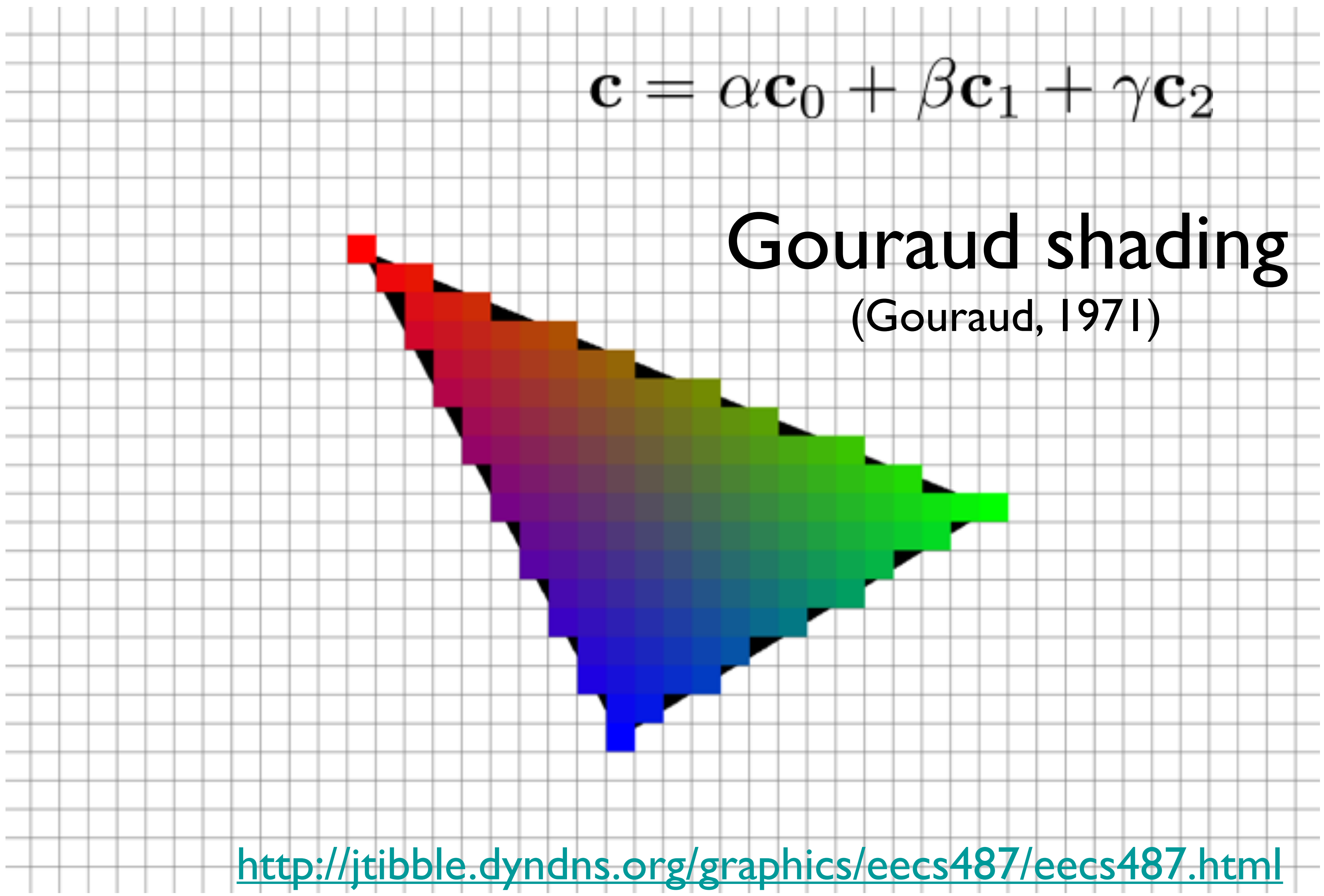
Use Midpoint Algorithm for edges and fill in?

Which pixels should be used to approximate a triangle?

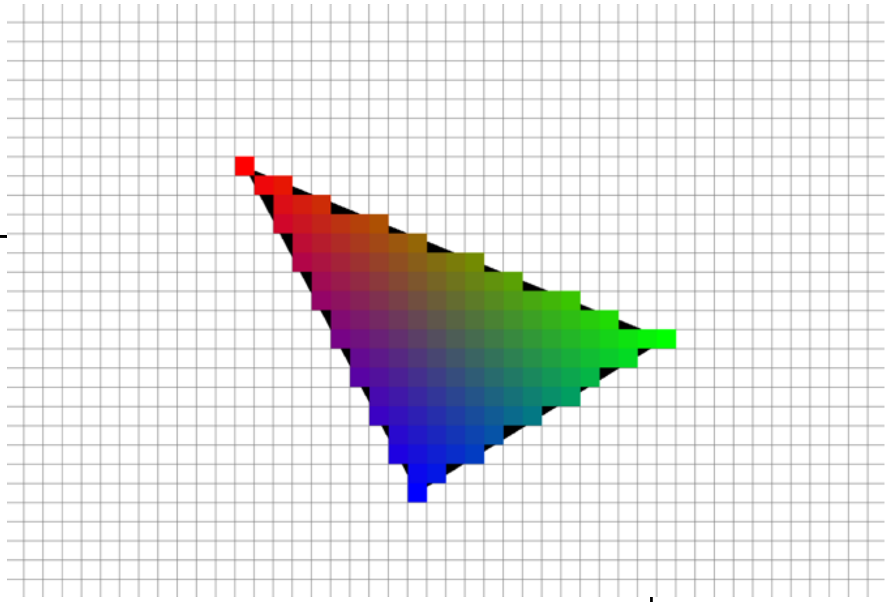


Use an approach based on barycentric coordinates

We can interpolate attributes using barycentric coordinates

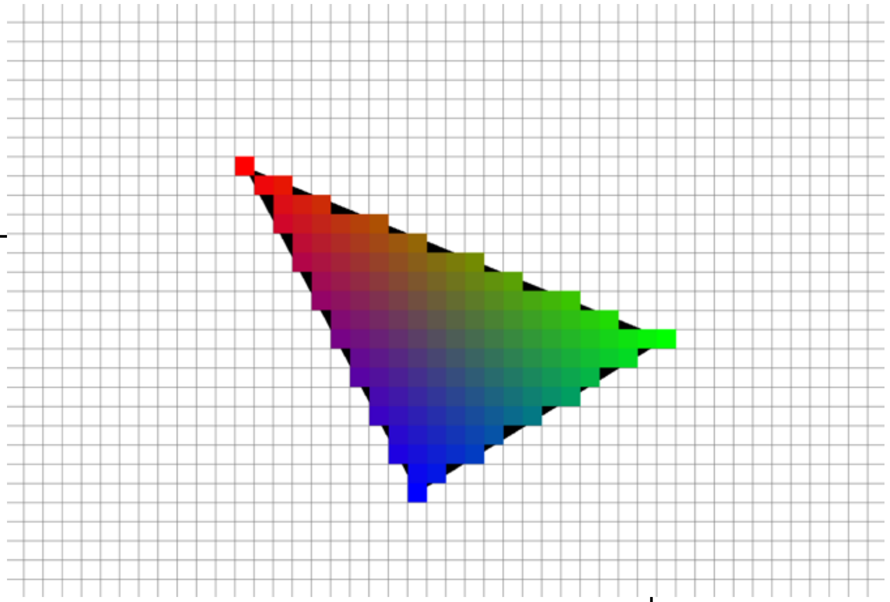


Triangle rasterization algorithm



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

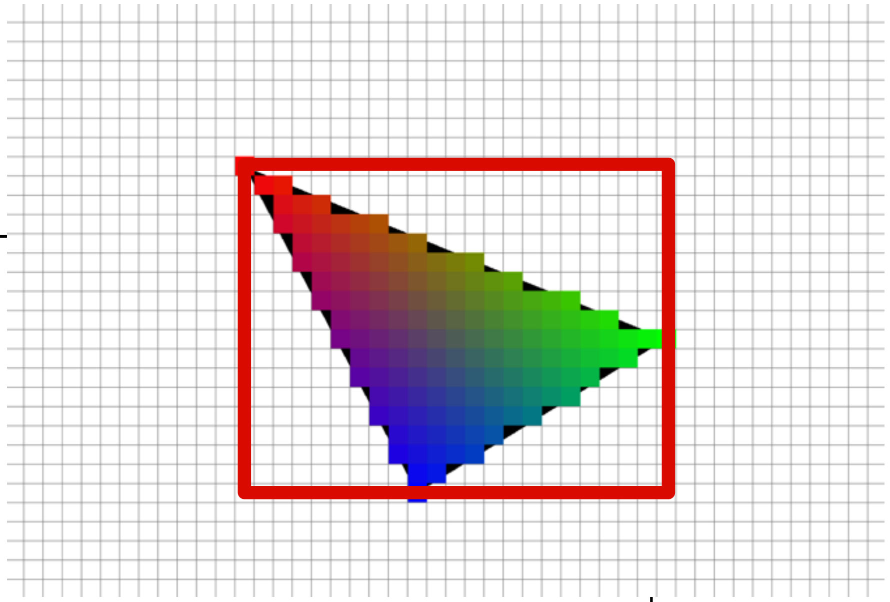
Triangle rasterization algorithm



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

Triangle rasterization algorithm

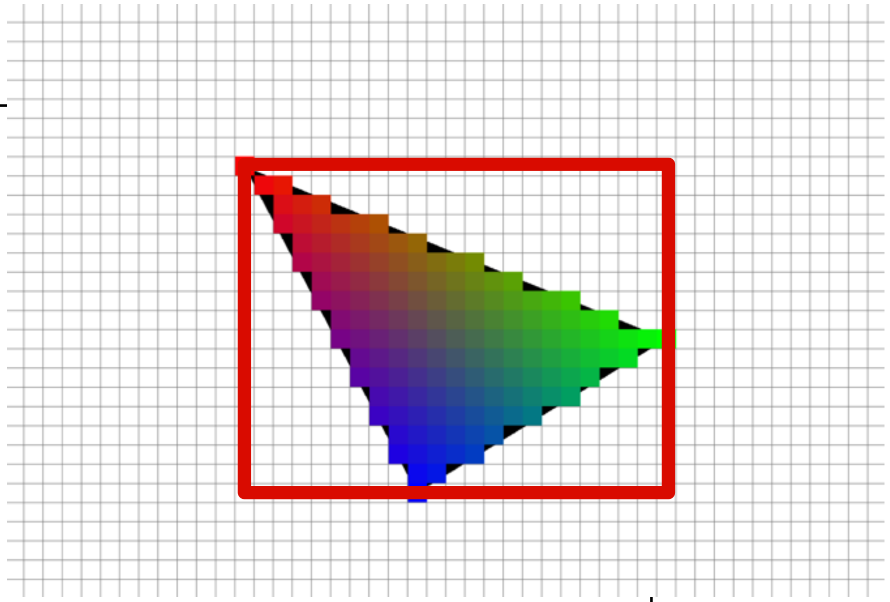
use a bounding rectangle



```
for x in [x_min, x_max]
    for y in [y_min, y_max]
        compute (α, β, γ) for (x,y)
        if (α ∈ [0,1] and β ∈ [0,1] and γ ∈ [0,1]) then
            c = αc_0 + βc_1 + γc_2
            drawpixel(x,y) with color c
```

$$\text{compute } (\alpha,\beta,\gamma) \text{ for } (x,y)$$
$$\text{if } (\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1]) \text{ then}$$
$$\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$$

Triangle rasterization algorithm



for x in [x_min, x_max]

    for y in [y_min, y_max]

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

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

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

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

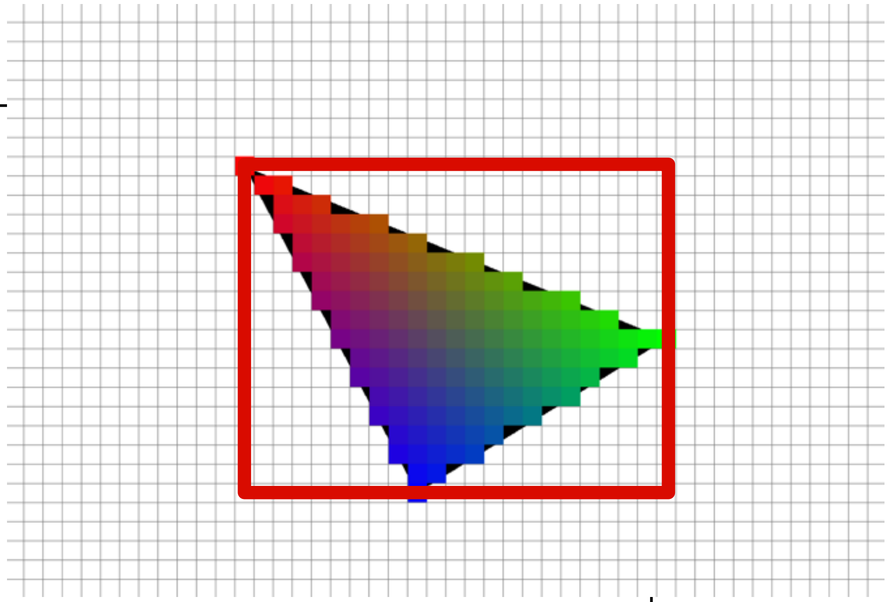
            $\mathbf{c} = \alpha\mathbf{c}_0 + \beta\mathbf{c}_1 + \gamma\mathbf{c}_2$

            drawpixel(x,y) with color c

<whiteboard>

Triangle rasterization algorithm

Optimizations?



for x in [x_min, x_max]

    for y in [y_min, y_max]

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

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

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

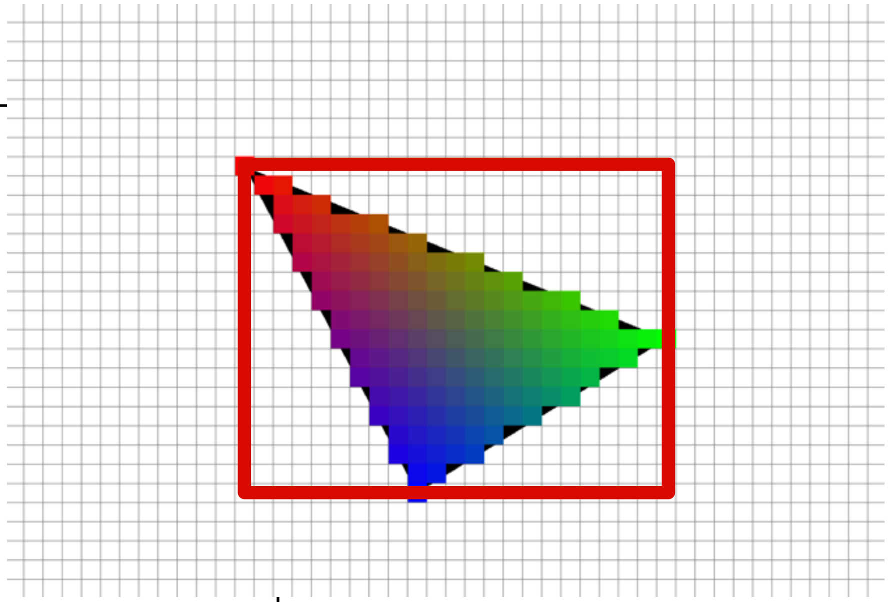
if $(\alpha \in [0,1] \text{ and } \beta \in [0,1] \text{ and } \gamma \in [0,1])$ then

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

drawpixel(x,y) with color c

Triangle rasterization algorithm

Optimizations?



for x in [x_min, x_max]

for y in [y_min, y_max]

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

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

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

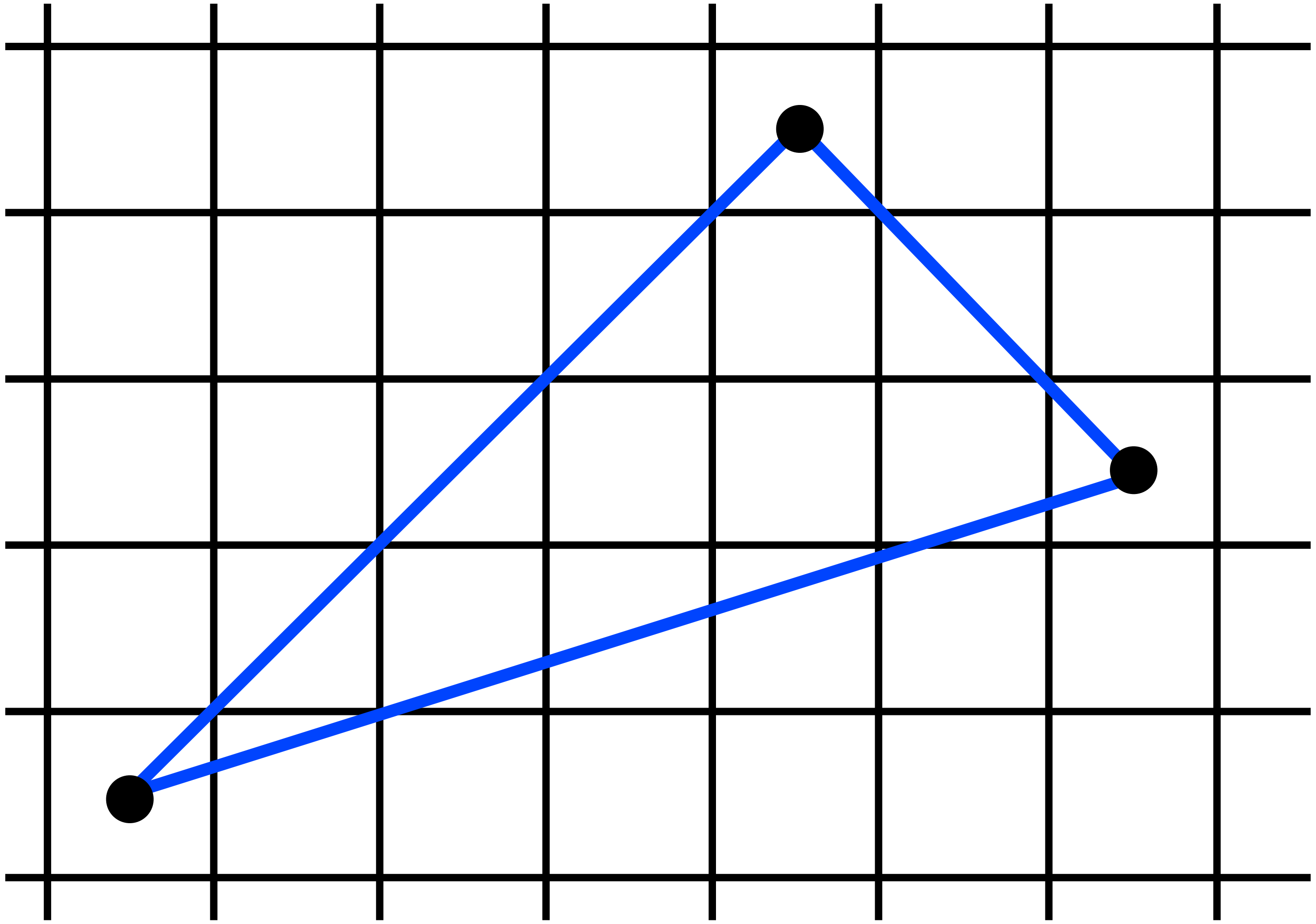
if $(\alpha \geq 0 \text{ and } \beta \geq 0 \text{ and } \gamma \geq 0)$then

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

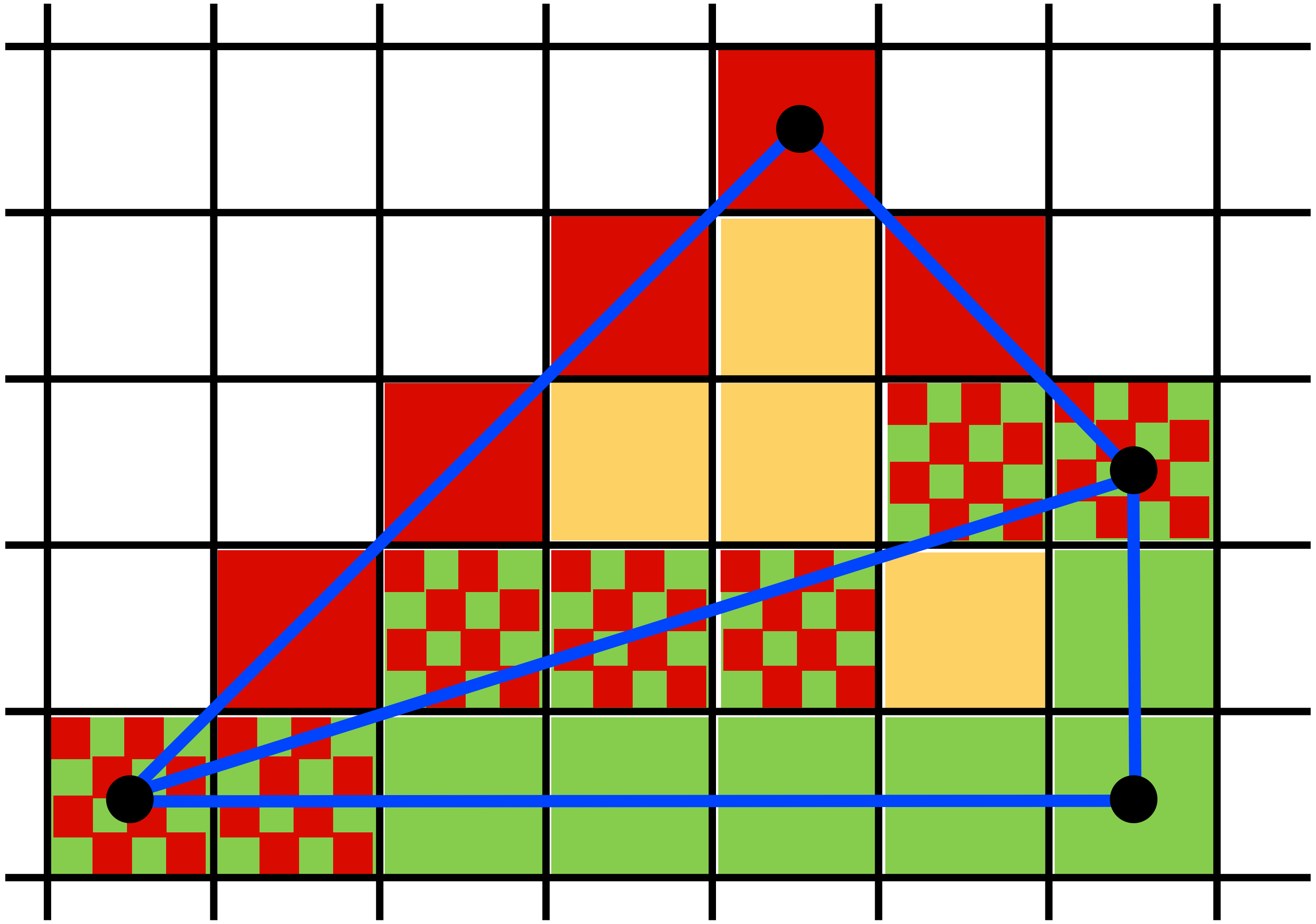
drawpixel(x,y) with color c

make computation of bary. coords. incremental
color can also be computed incrementally
don't need to check upper bound

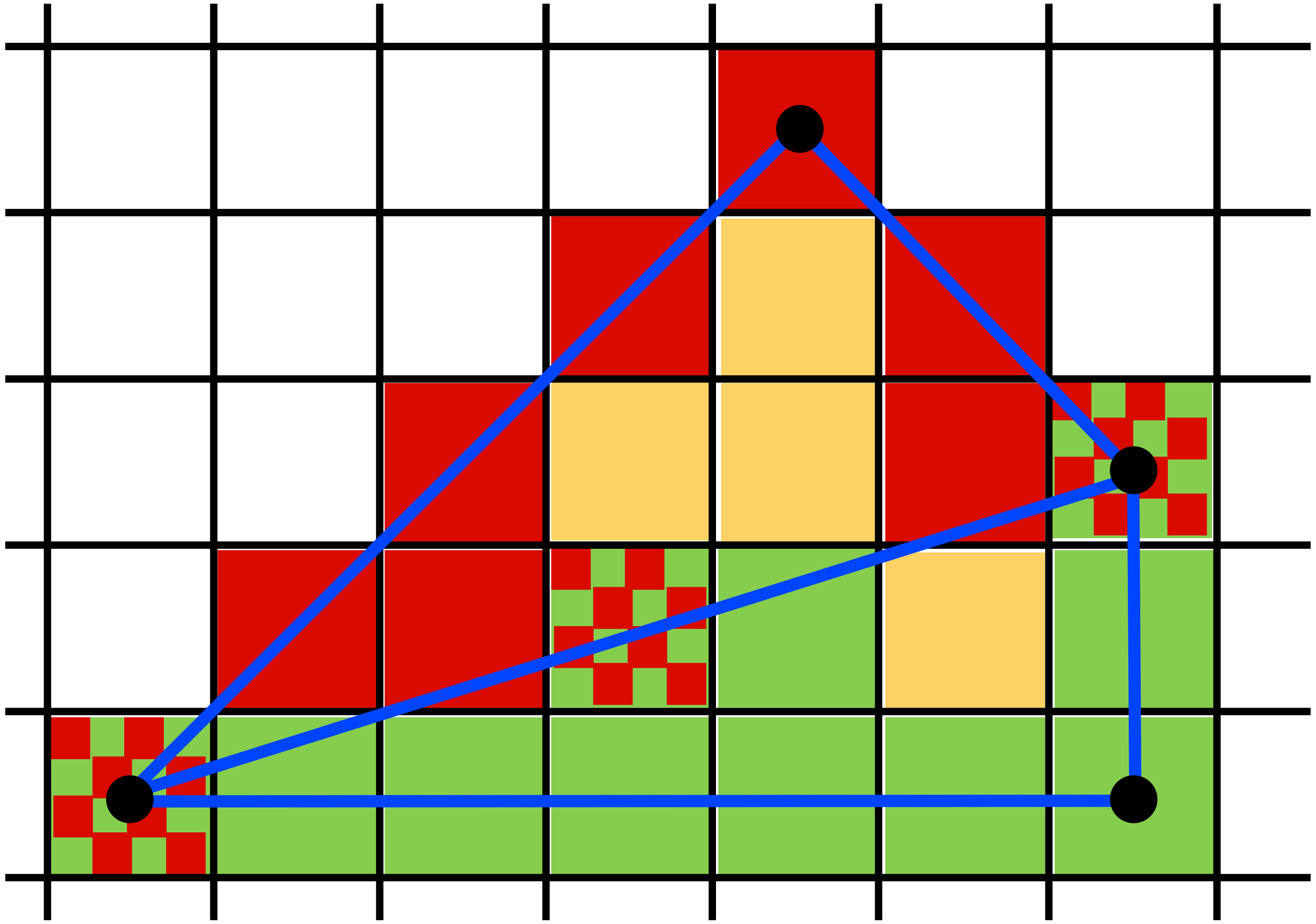
Triangle rasterization issues



Who should fill in shared edge?

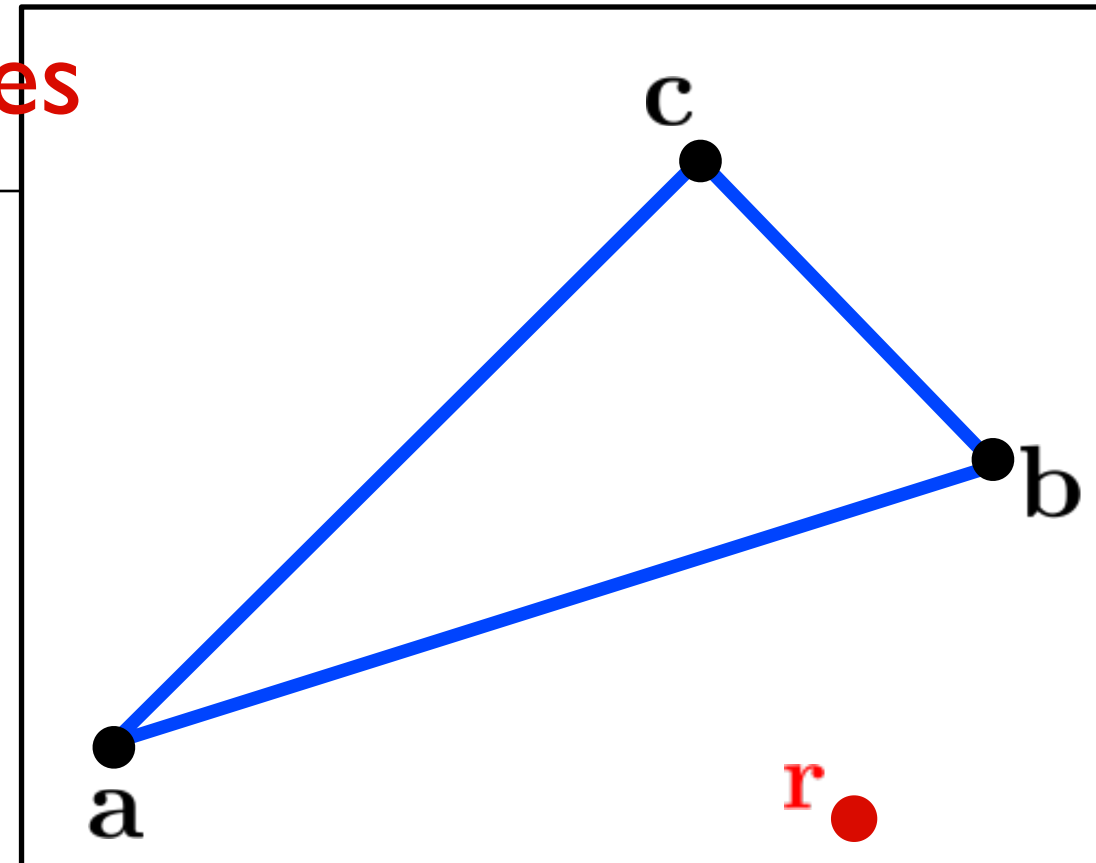


Who should fill in shared edge?



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

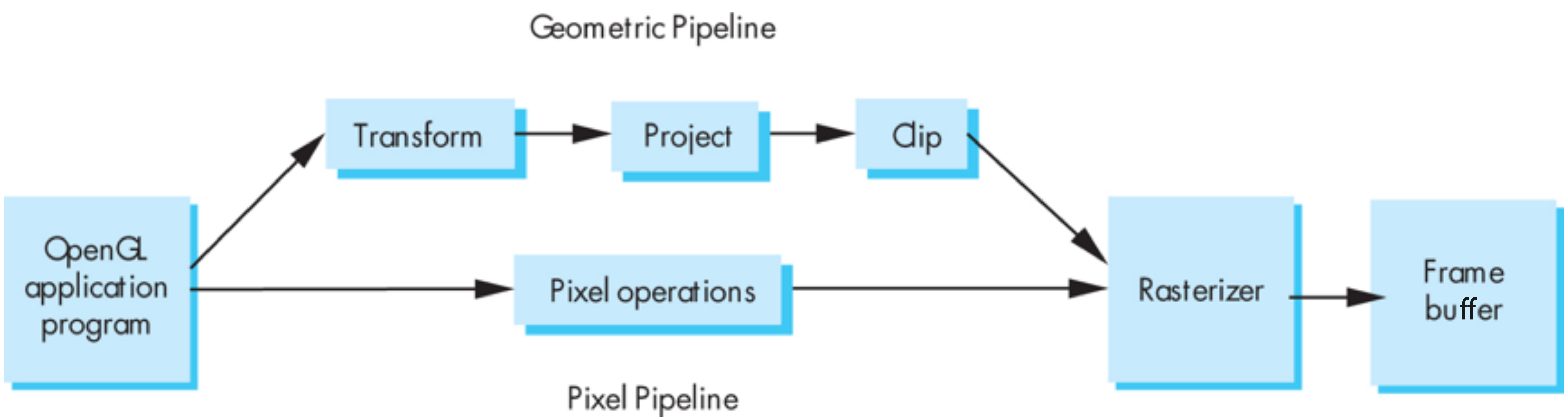
if $(\alpha > 0 \text{ or } f_{bc}(\mathbf{a})f_{bc}(\mathbf{r}) > 0)$ and $(\beta > 0 \text{ or } f_{ca}(\mathbf{b})f_{ca}(\mathbf{r}) > 0)$ and $(\gamma > 0 \text{ or } f_{ab}(\mathbf{c})f_{ab}(\mathbf{r}) > 0)$ then

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

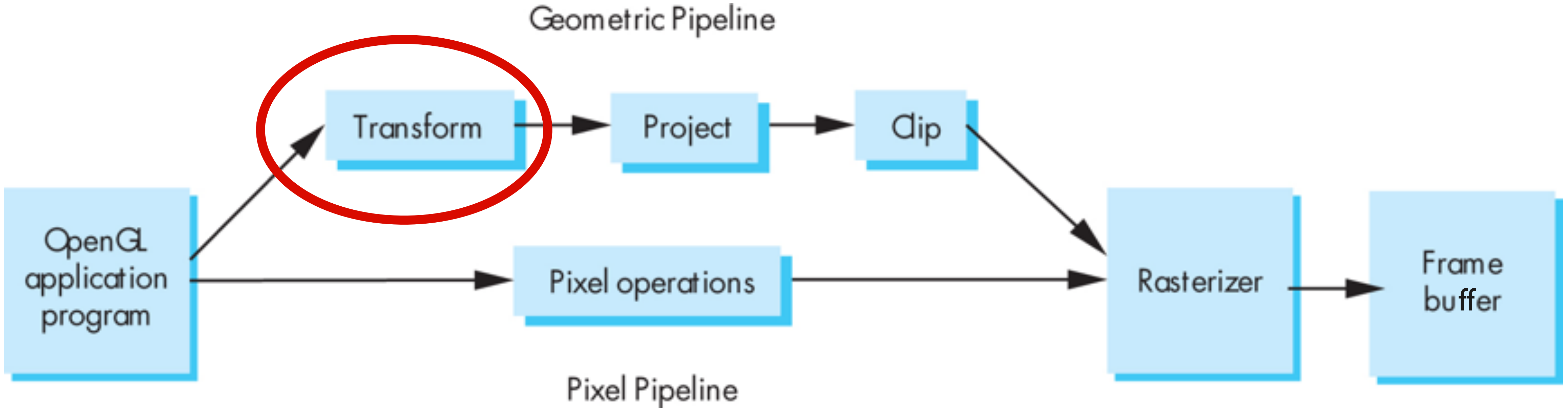
drawpixel(x,y) with color c

Graphics Pipeline (cont.)

Graphics Pipeline

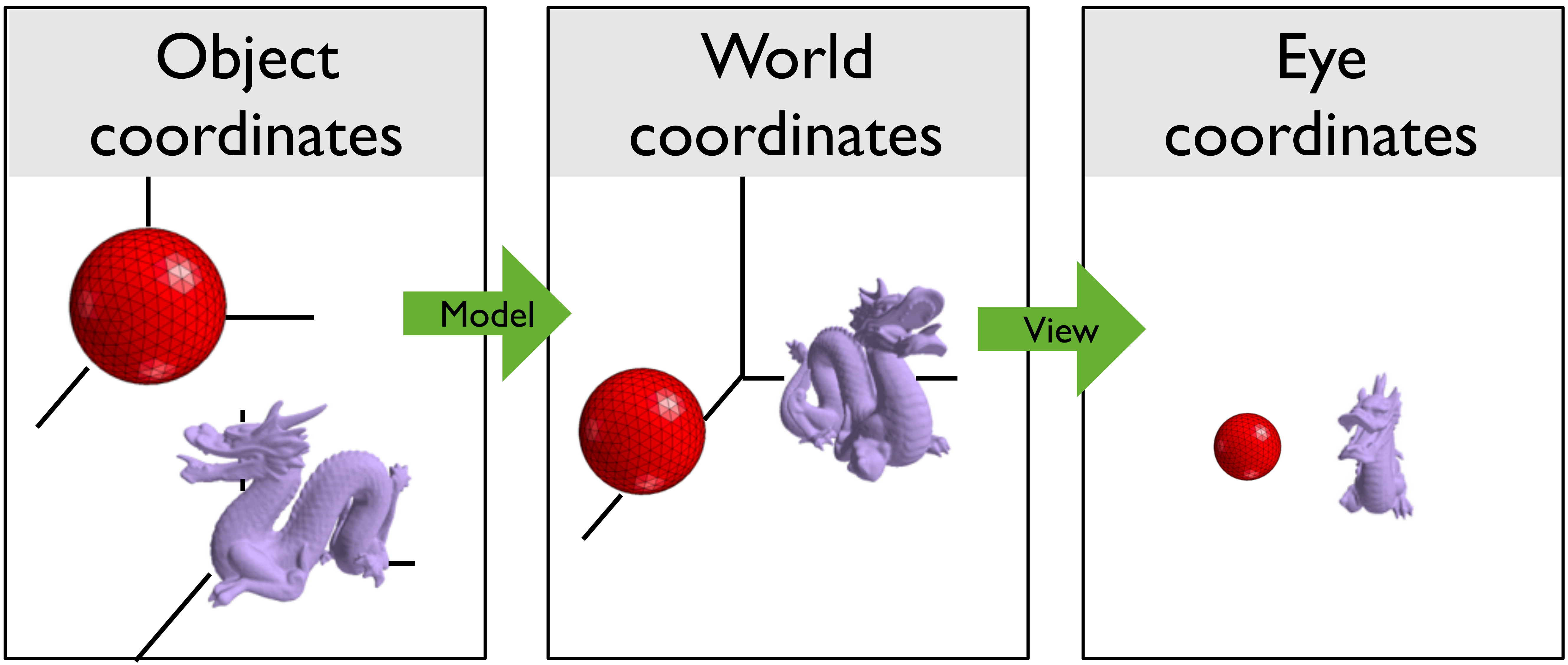


Transform

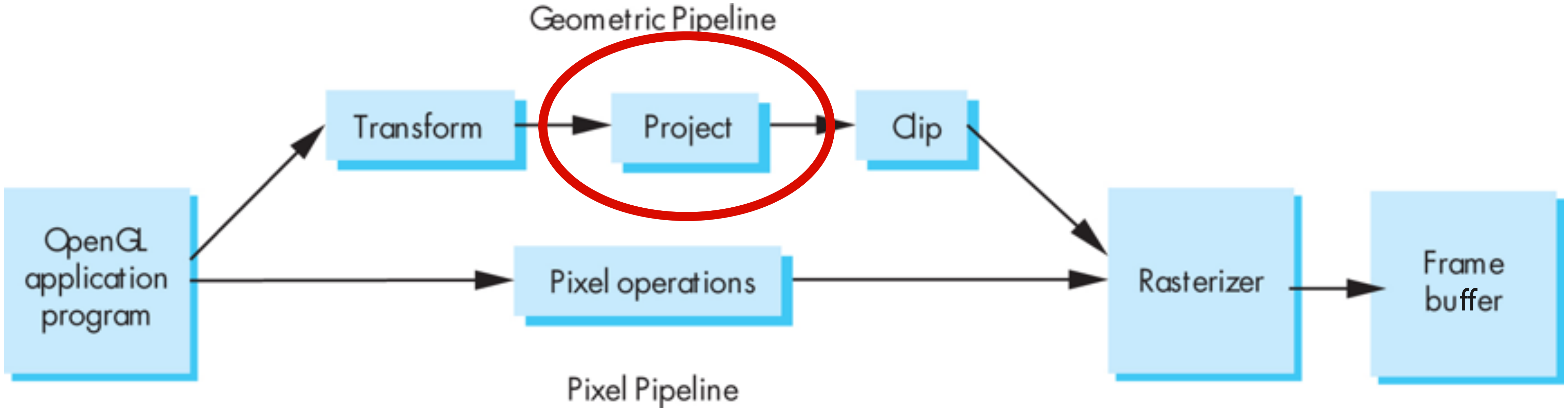


Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

“Modelview” Transformation

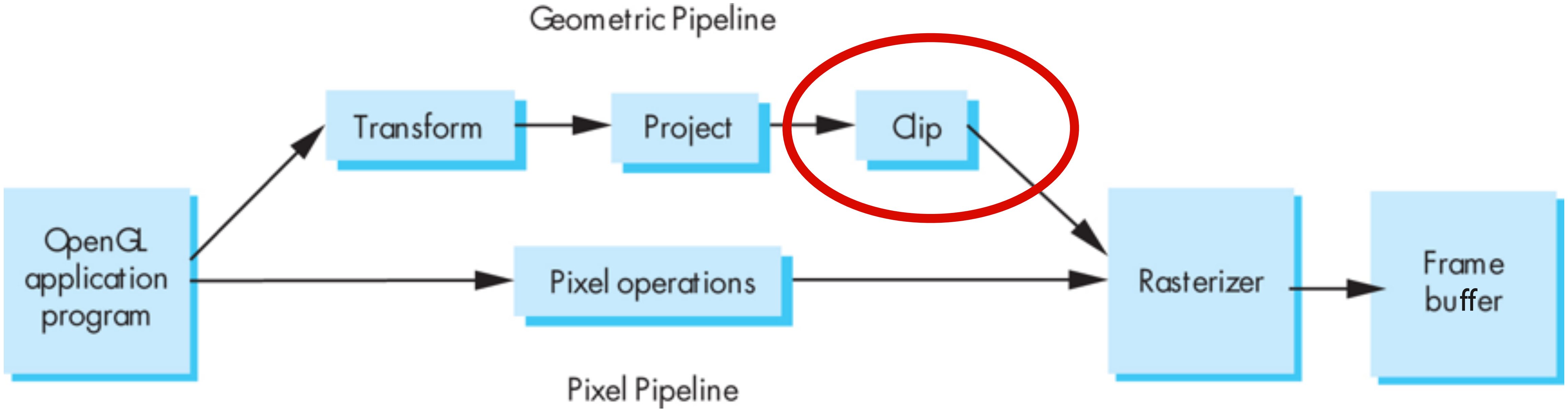


Project



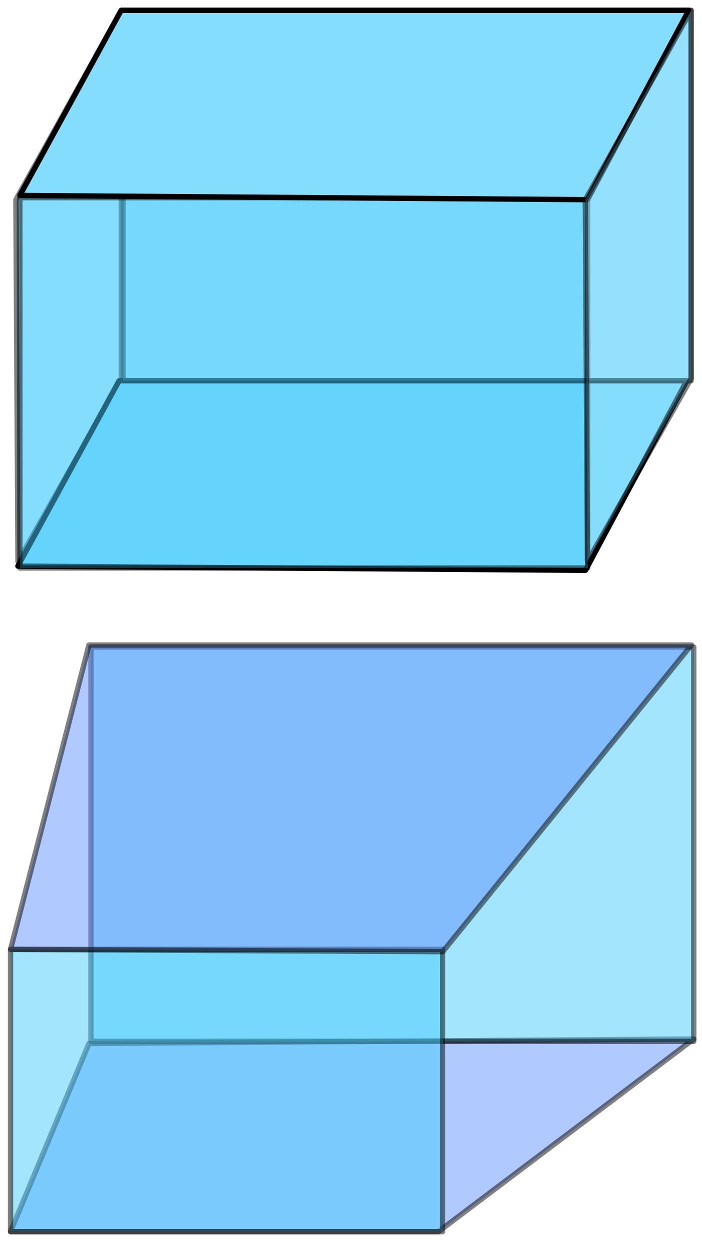
Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

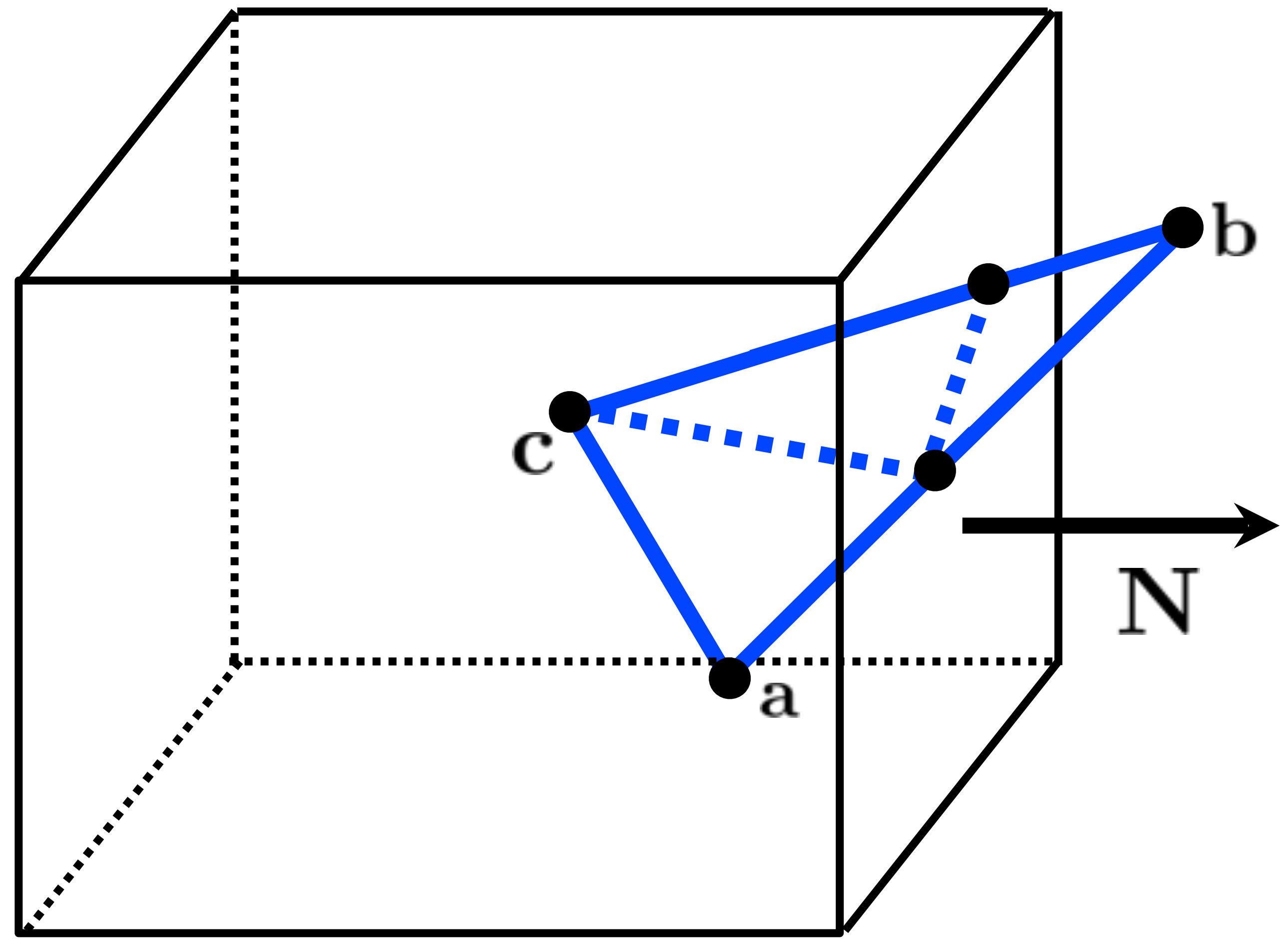
Clip



Geometric Pipeline
Transform
Project
Clip
OpenGL application program
Pixel operations
Rasterizer
Frame buffer
Pixel Pipeline

Clip against view volume

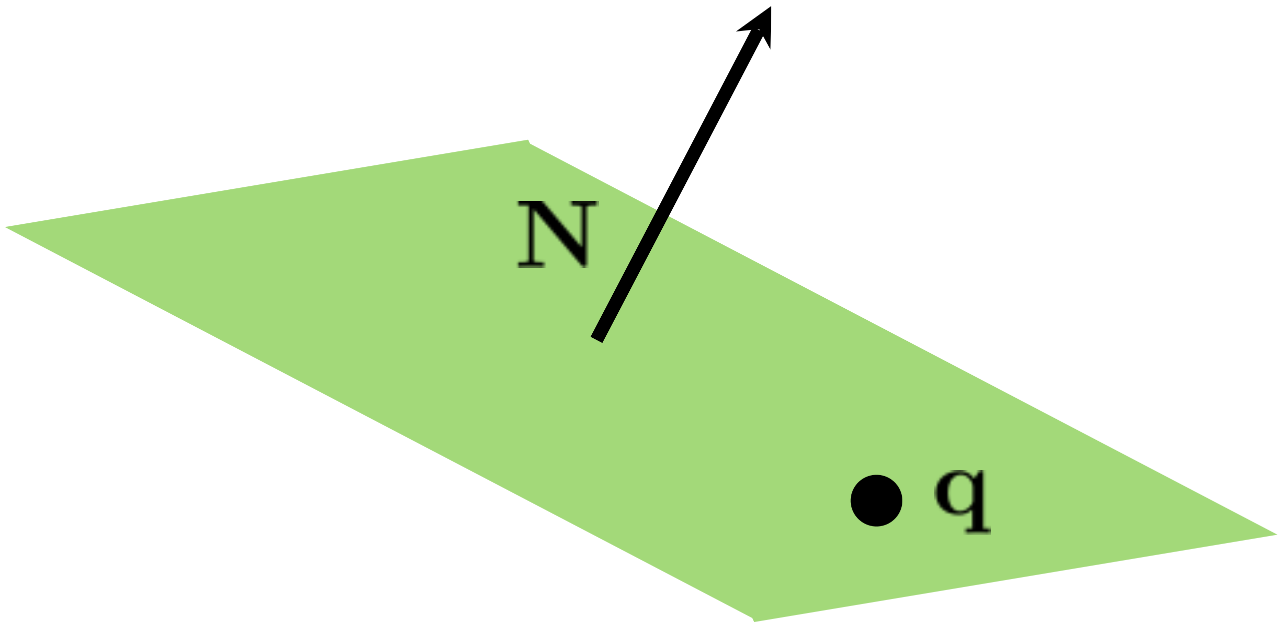




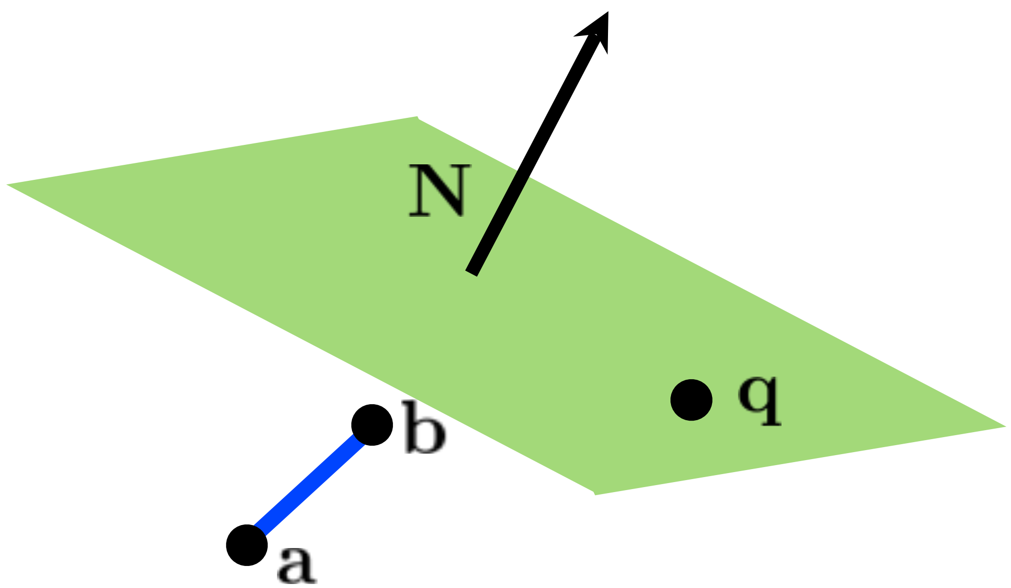
Clipping against a plane

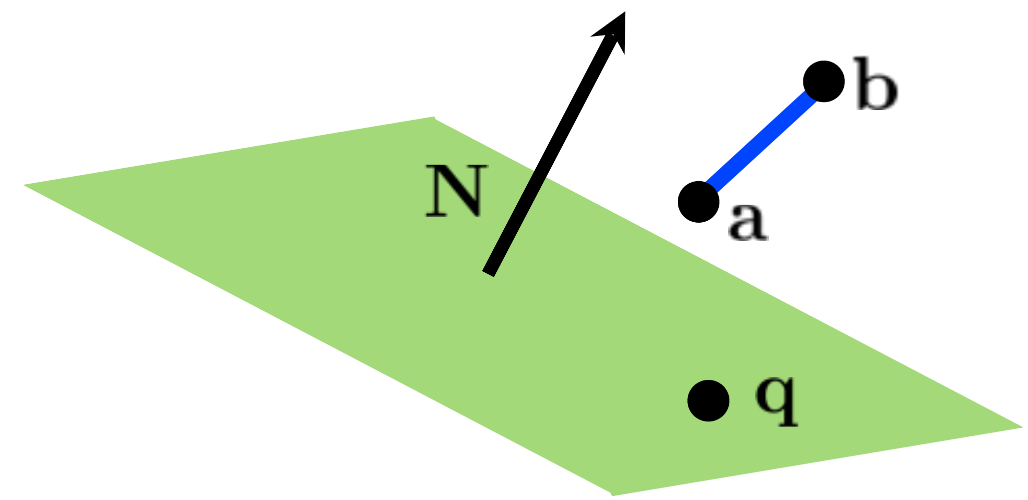
What's the equation for the plane through **q** with normal **N**?

$$f(\mathbf{p}) = \mathbf{N} \cdot (\mathbf{p} - \mathbf{q}) = 0$$

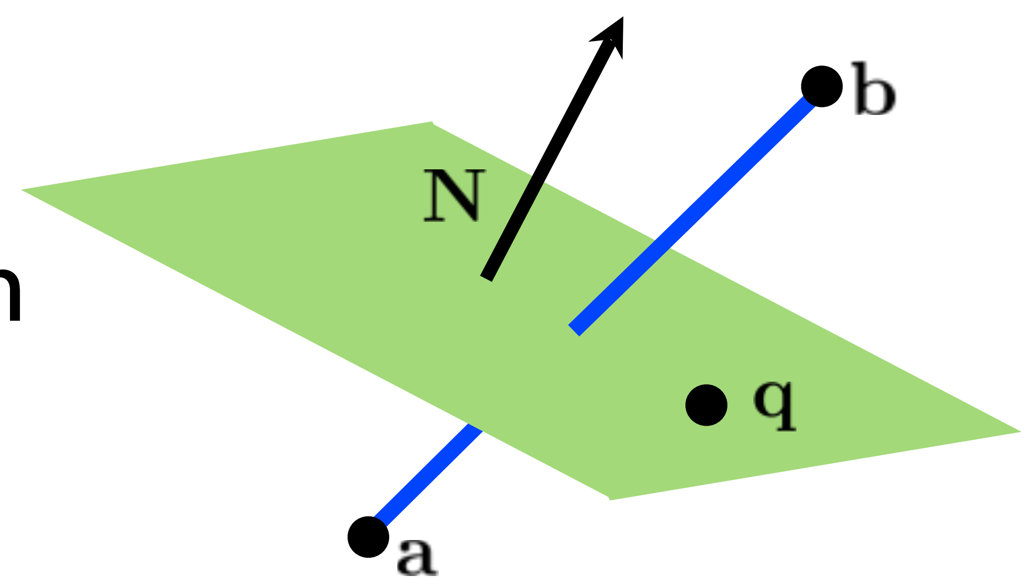


Intersection of line and plane



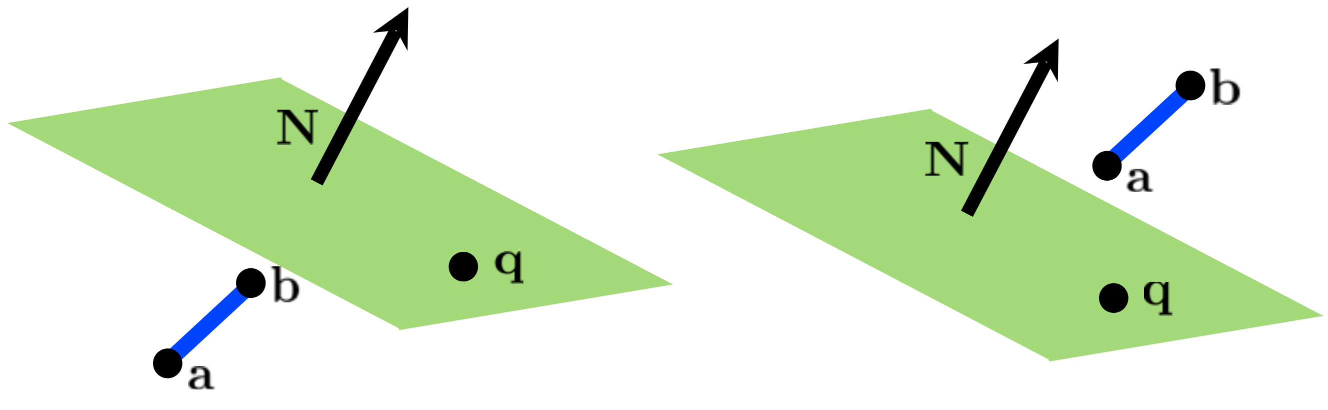


How can we distinguish between these cases?

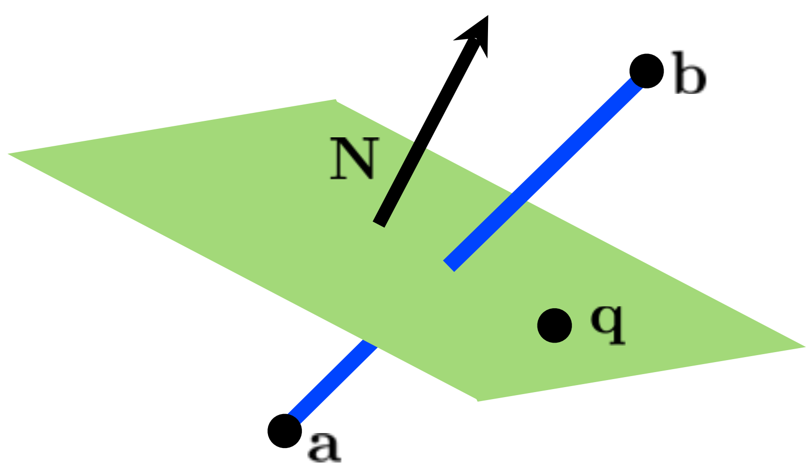


Intersection of line and plane

$$f(\mathbf{a})f(\mathbf{b}) \geq 0$$

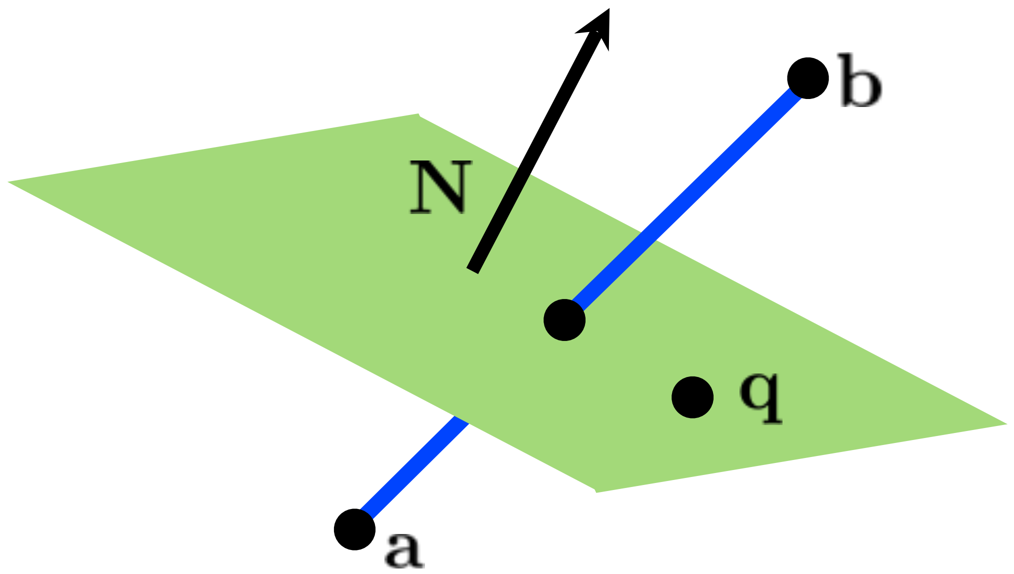


$$f(\mathbf{a})f(\mathbf{b}) < 0$$



Intersection of line and plane

How can we find the intersection point?



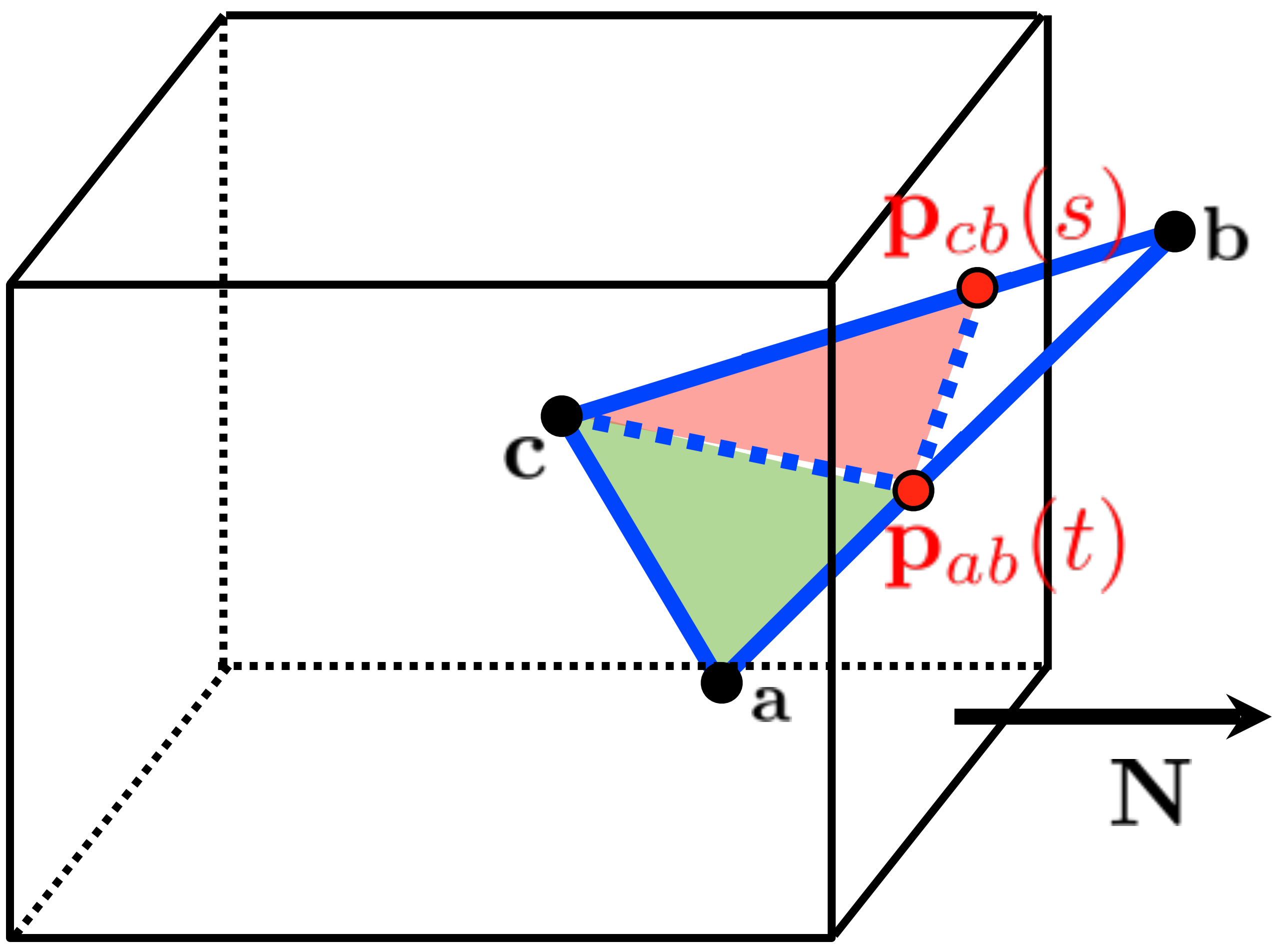
<whiteboard>

Clip against view volume

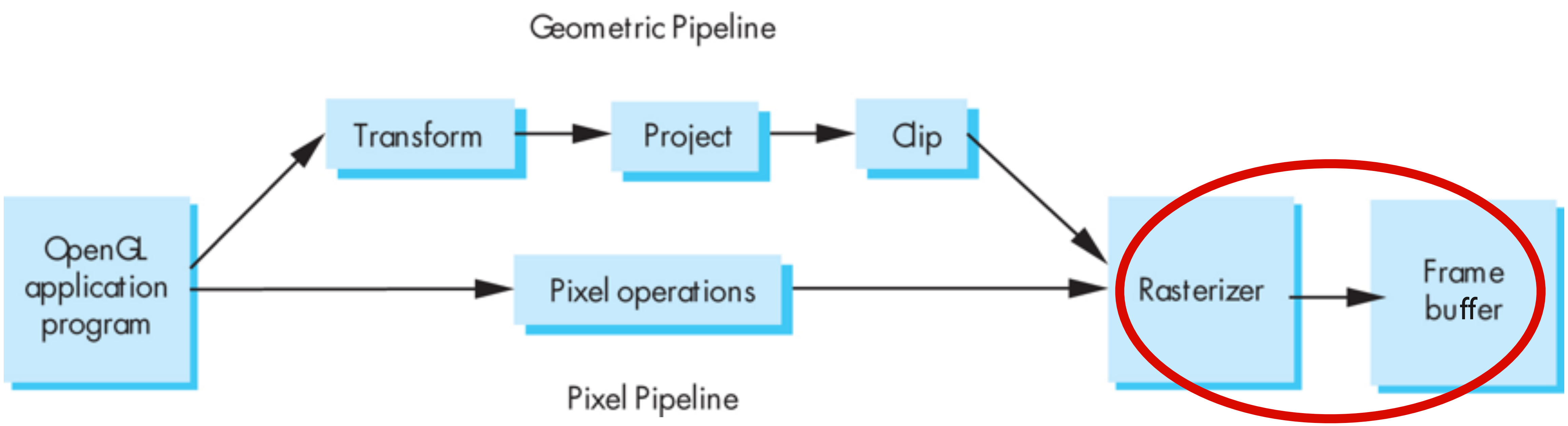
$$s = \frac{\mathbf{N} \cdot (\mathbf{q} - \mathbf{c})}{\mathbf{N} \cdot (\mathbf{b} - \mathbf{c})}$$

$$t = \frac{\mathbf{N} \cdot (\mathbf{q} - \mathbf{a})}{\mathbf{N} \cdot (\mathbf{b} - \mathbf{a})}$$

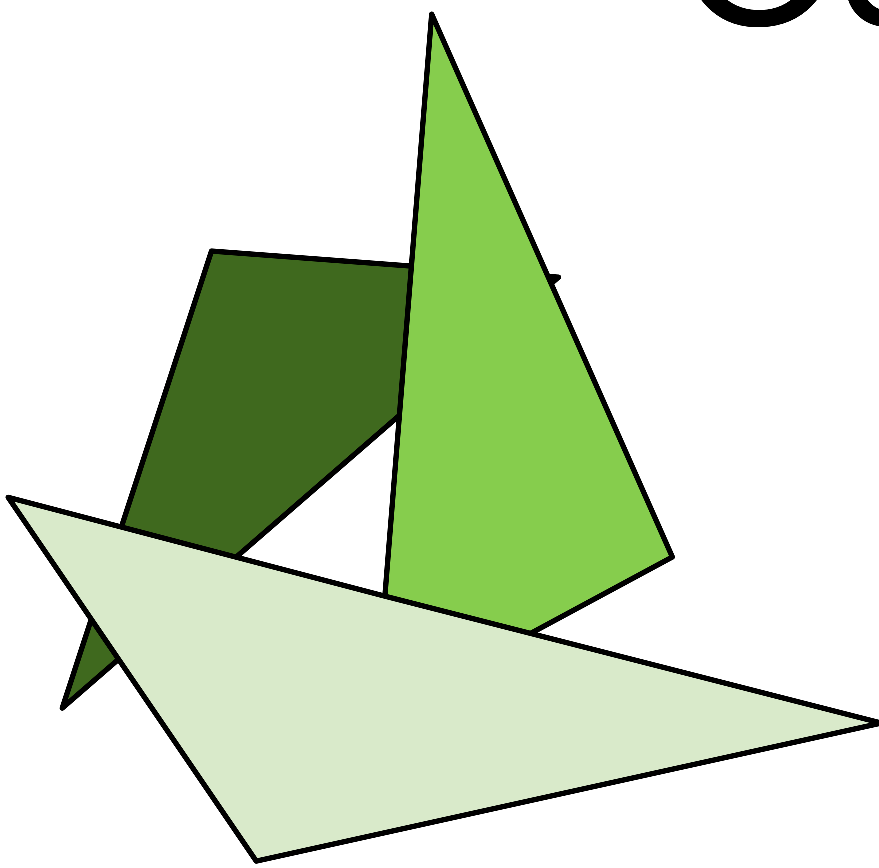
need to generate new triangles



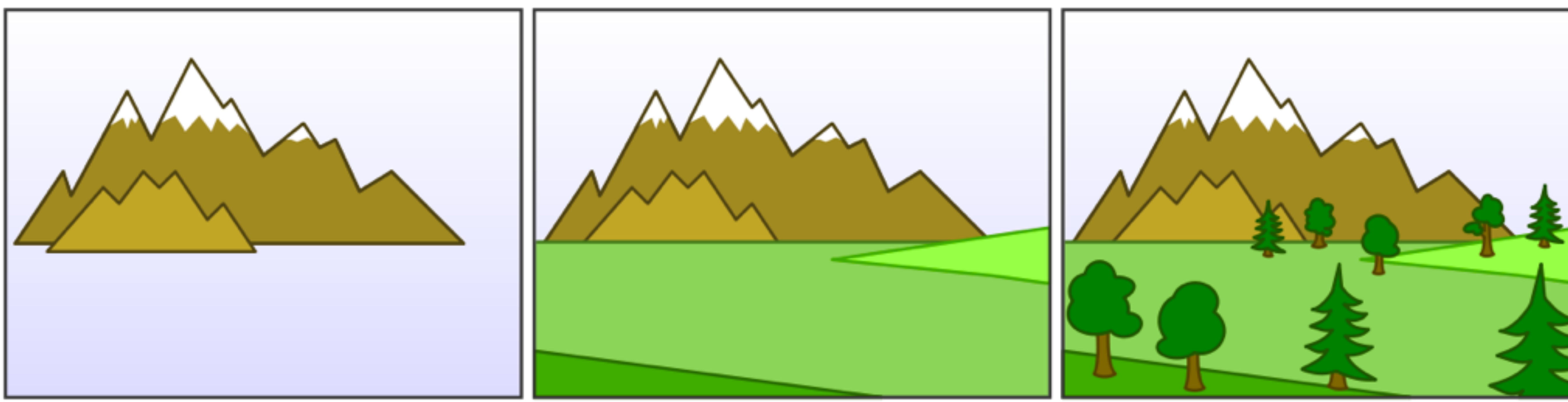
Hidden Surface Removal



Occlusion

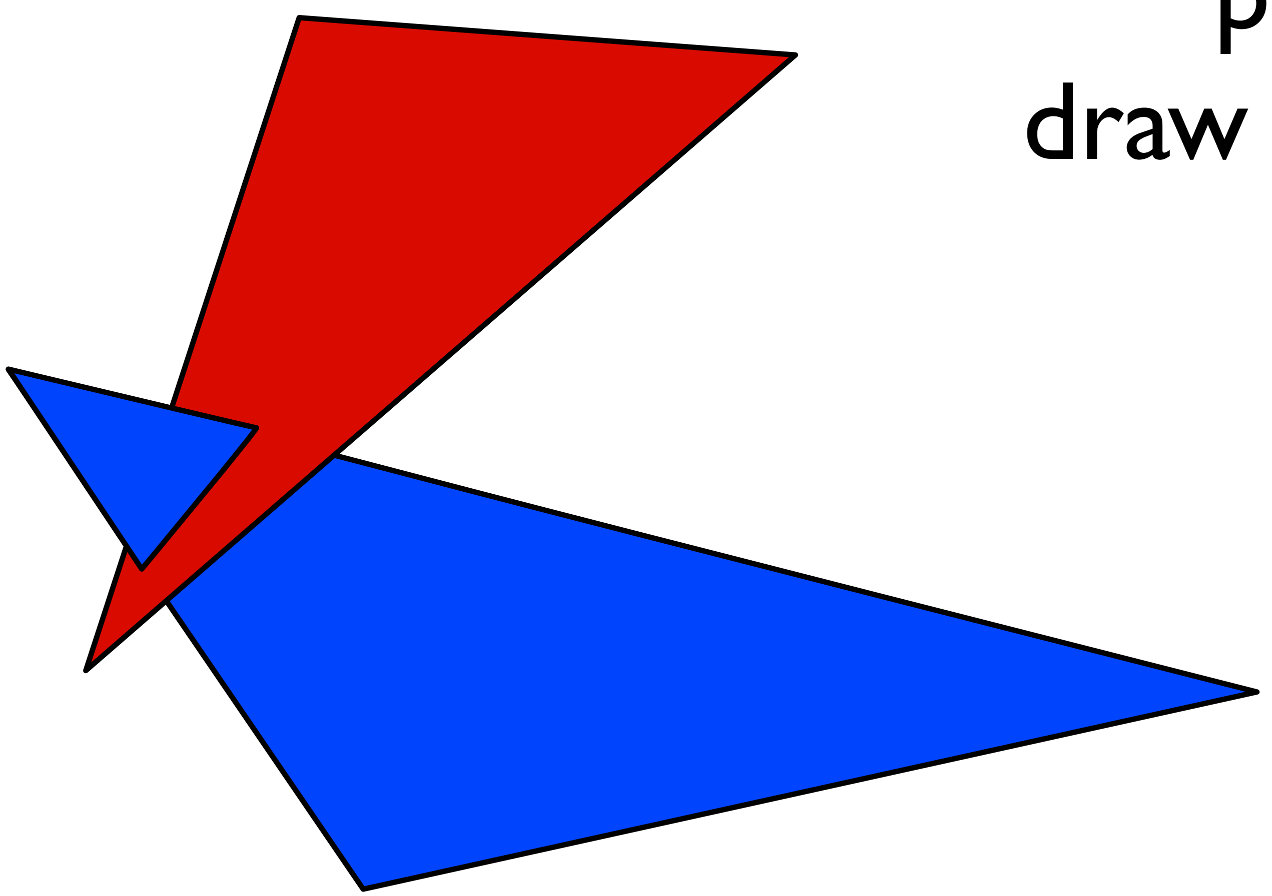


“painter’s algorithm”
draw primitives in
back-to-front order



[Wikimedia Commons]

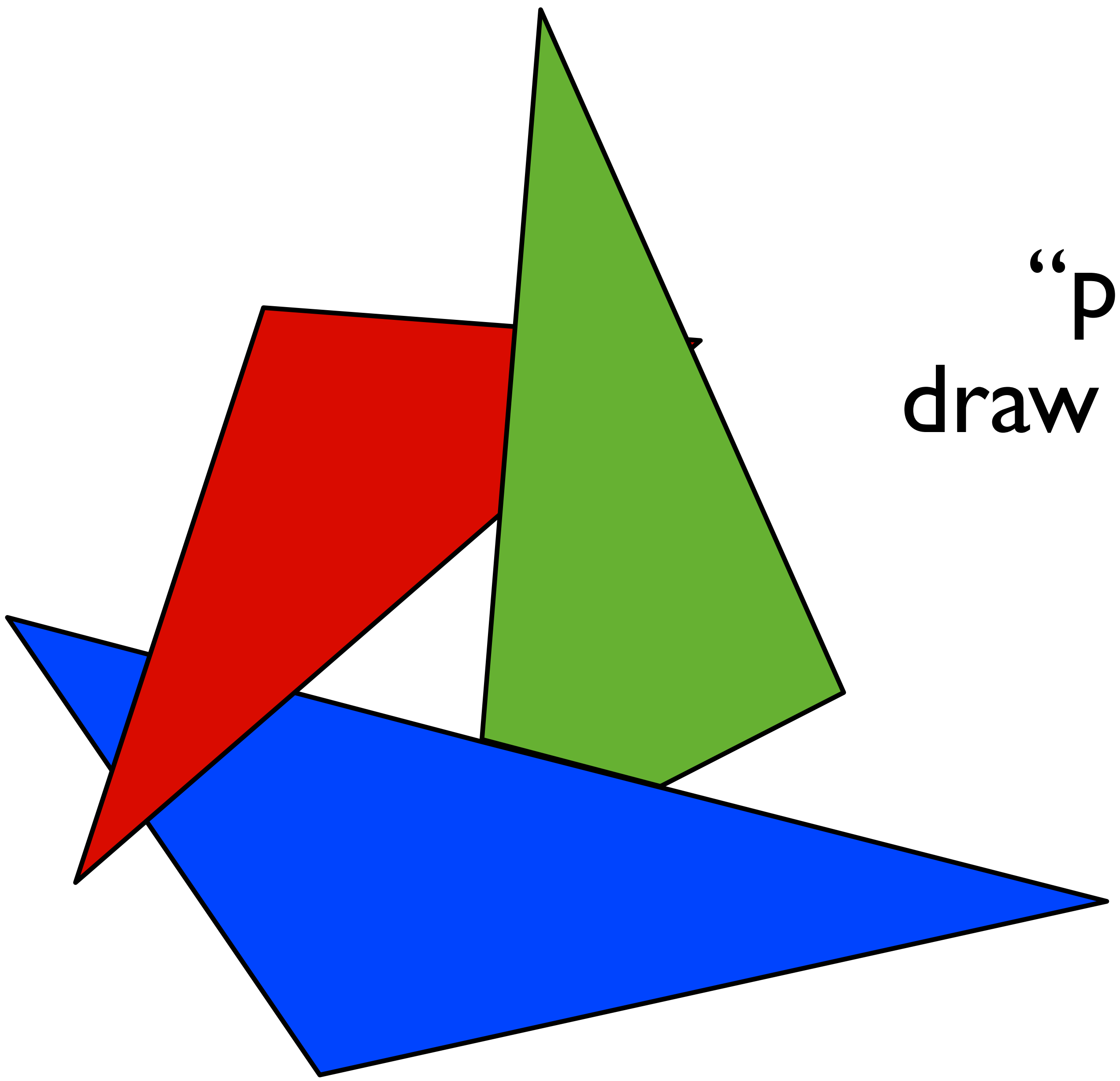
Occlusion



“painter’s algorithm”
draw primitives in back-to-front order

problem:
triangle intersection

Occlusion



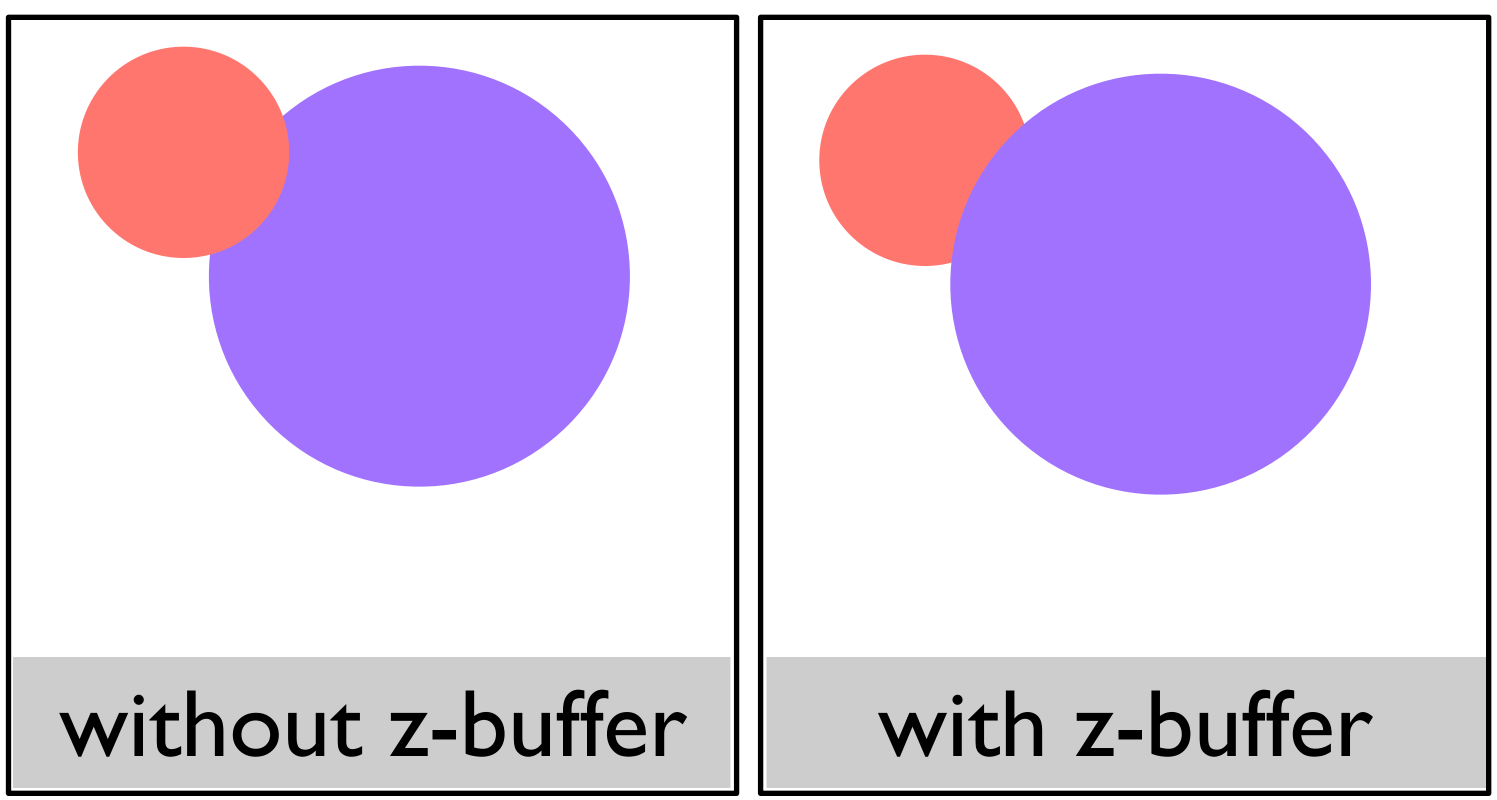
“painter’s algorithm”
draw primitives in back-to-front order

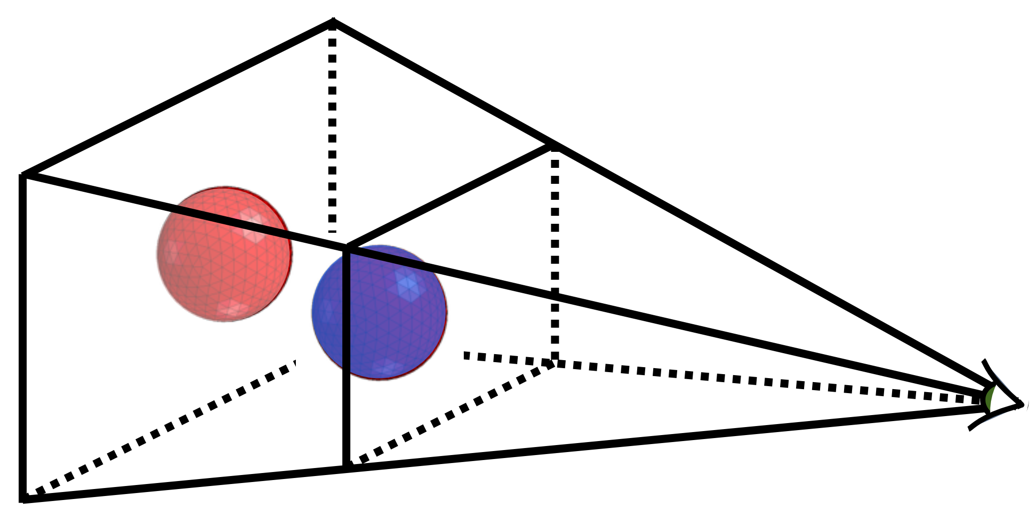
problem:
occlusion cycle

Use a *z-buffer* for hidden surface removal

test depth on a pixel by pixel basis

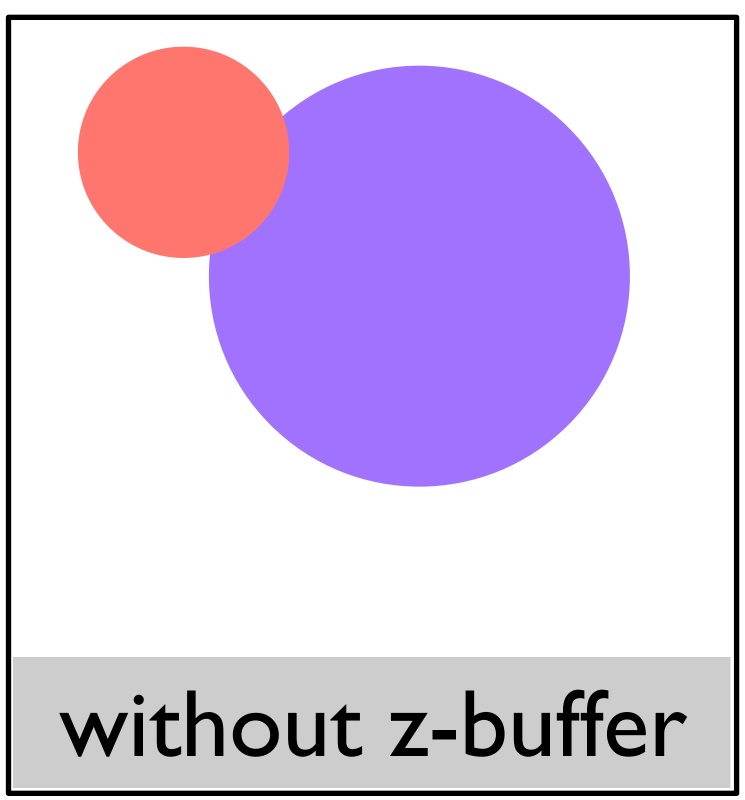
red drawn last



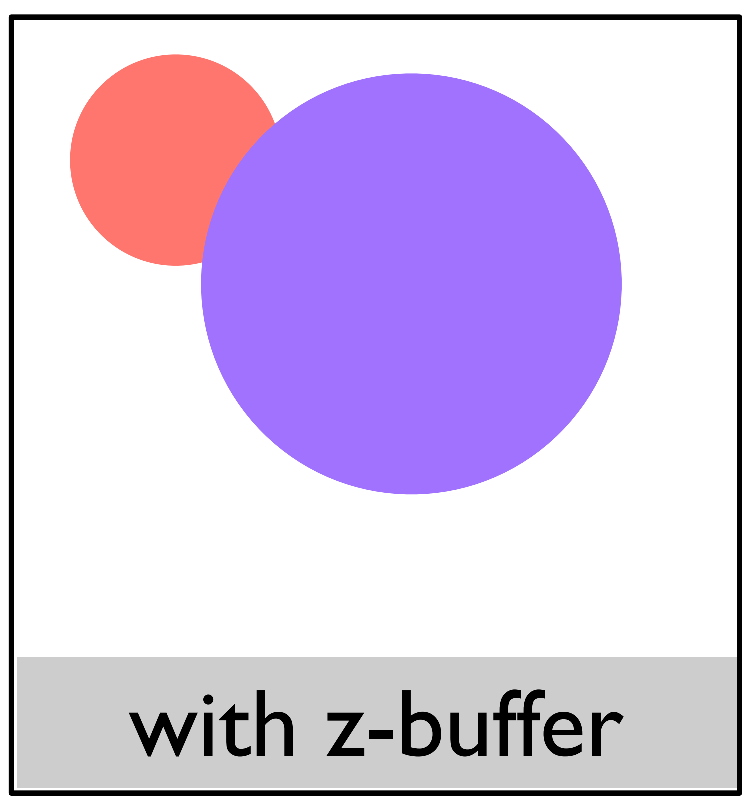


Use a *z-buffer* for hidden surface removal

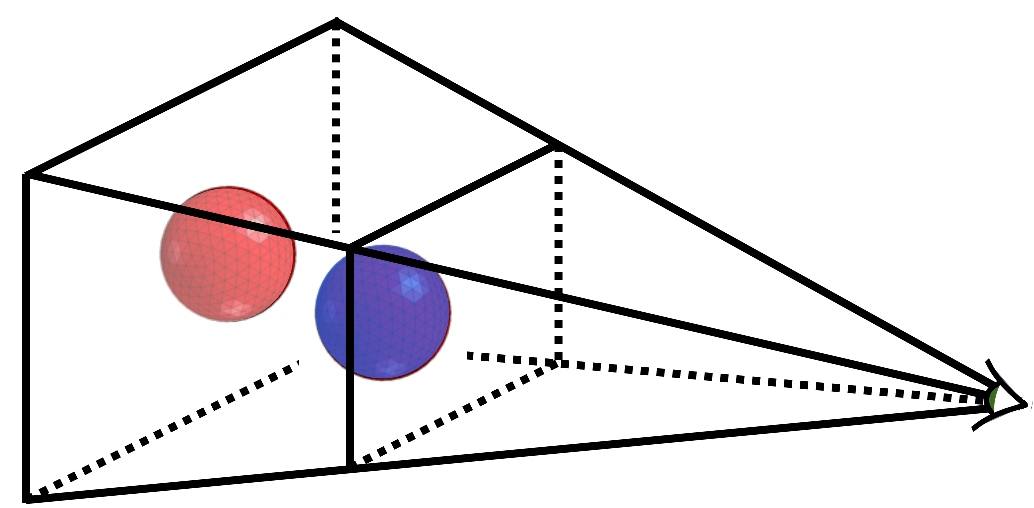
at each pixel, record distance to the closest object that has been drawn in a *depth* buffer



without z-buffer



with z-buffer



Use a *z-buffer* for hidden surface removal

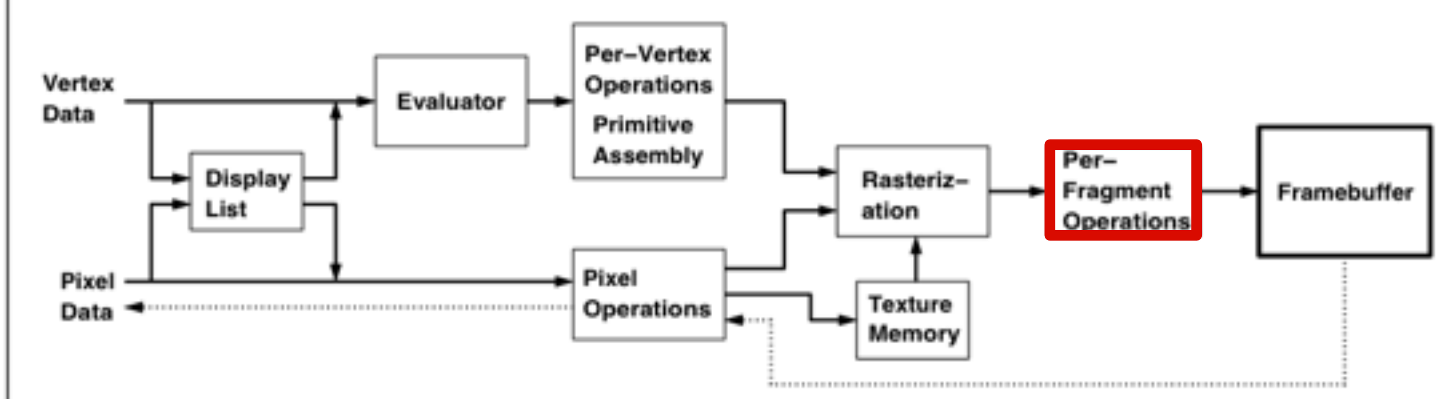
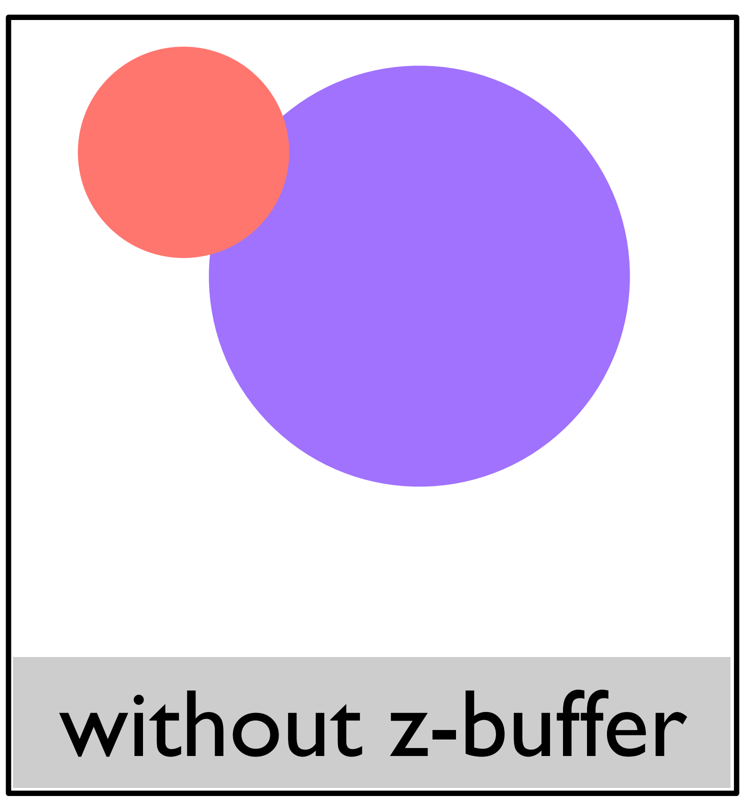
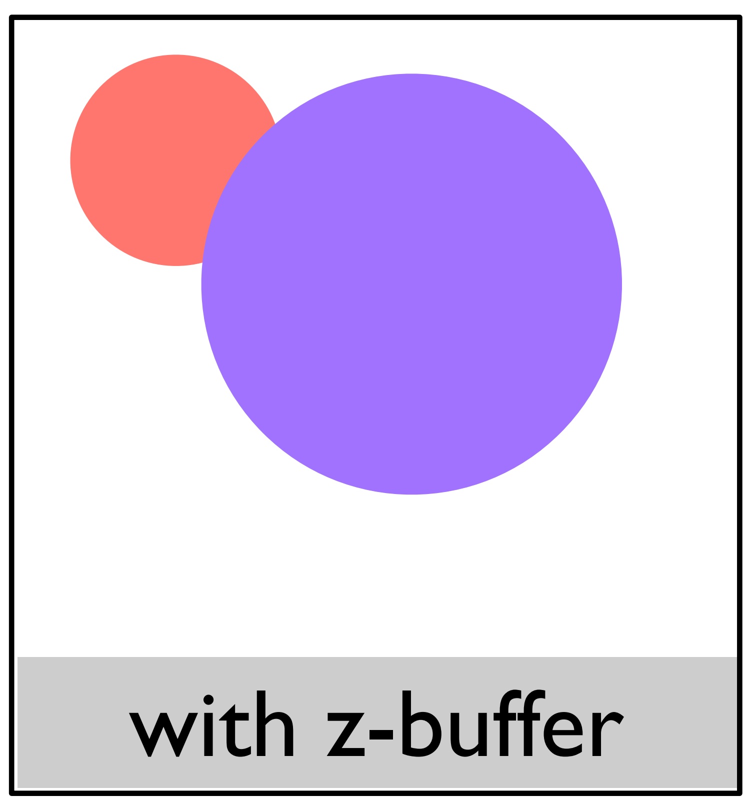
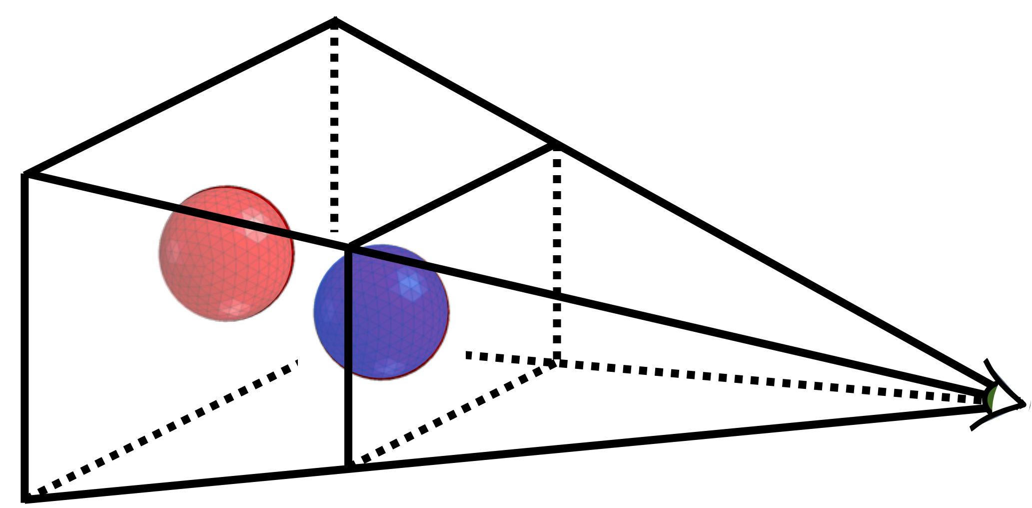


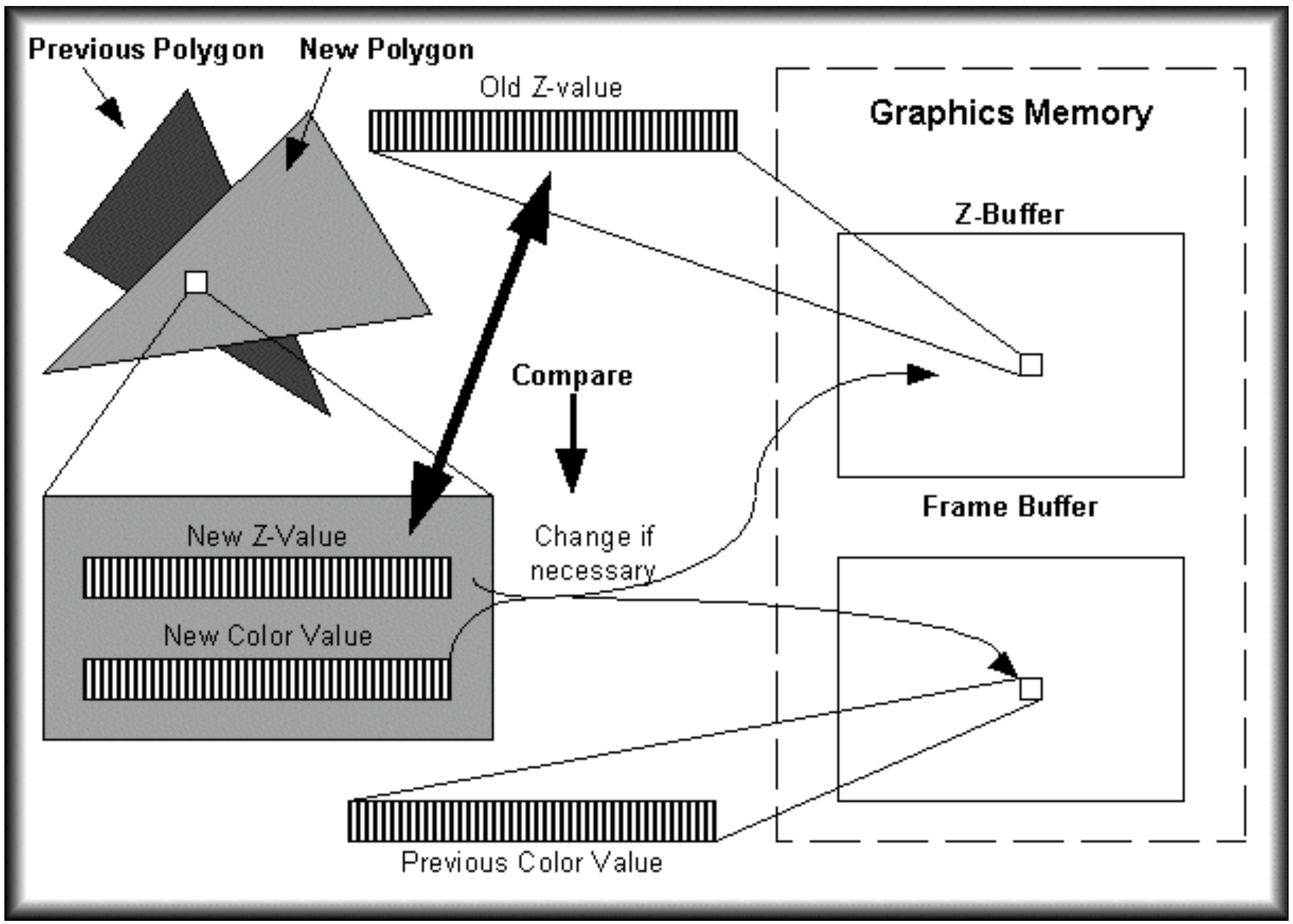
Figure 1. Block diagram of OpenGL.





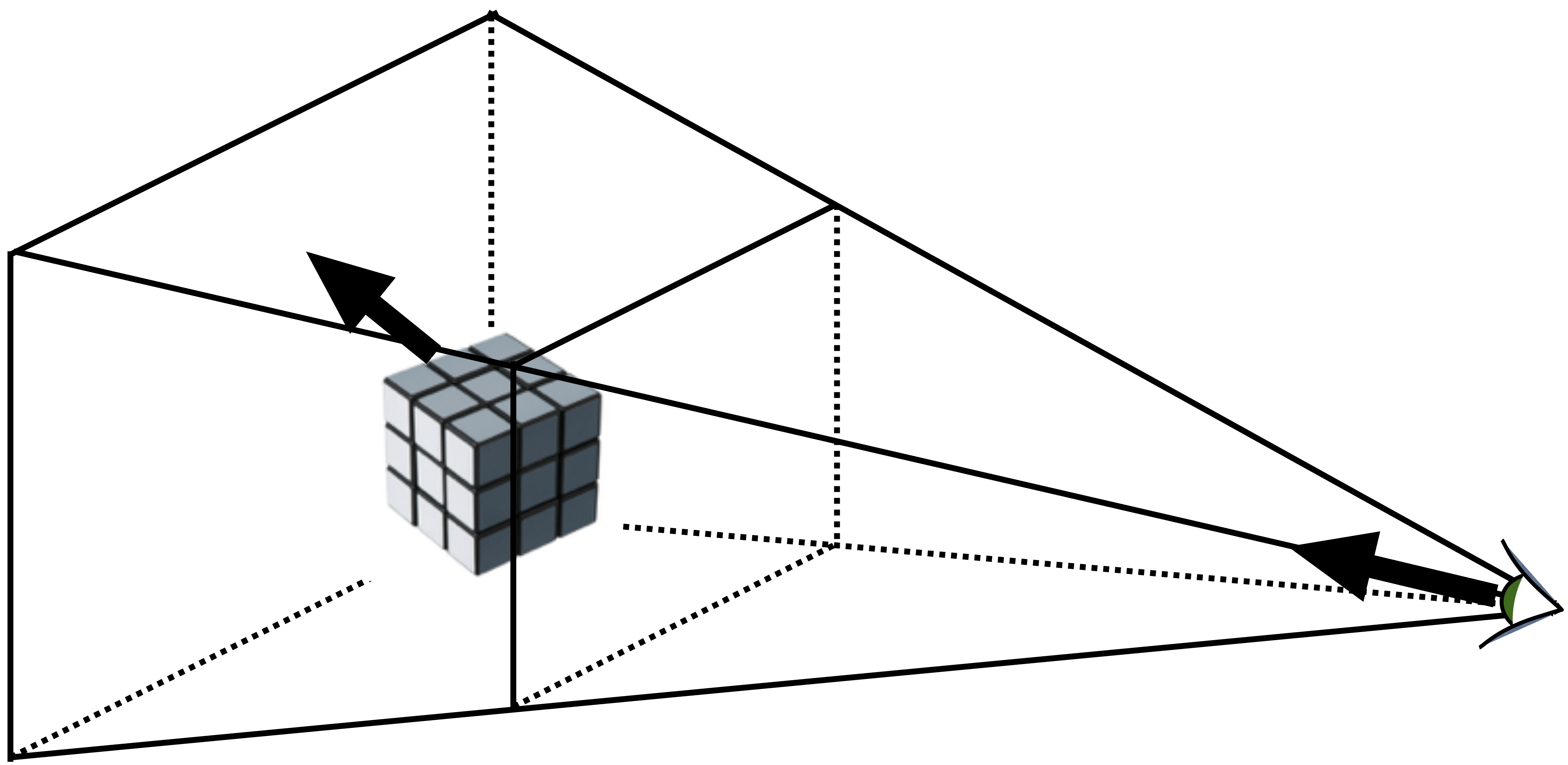


Use a *z-buffer* for hidden surface removal



http://www.beyond3d.com/content/articles/41/

Backface culling: another way to eliminate hidden geometry



Hidden Surface Removal in OpenGL

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

glEnable(GL_DEPTH_TEST);

glEnable(GL_CULL_FACE);
```

For a perspective transformation, there is more precision in the depth buffer for z-values closer to the near plane