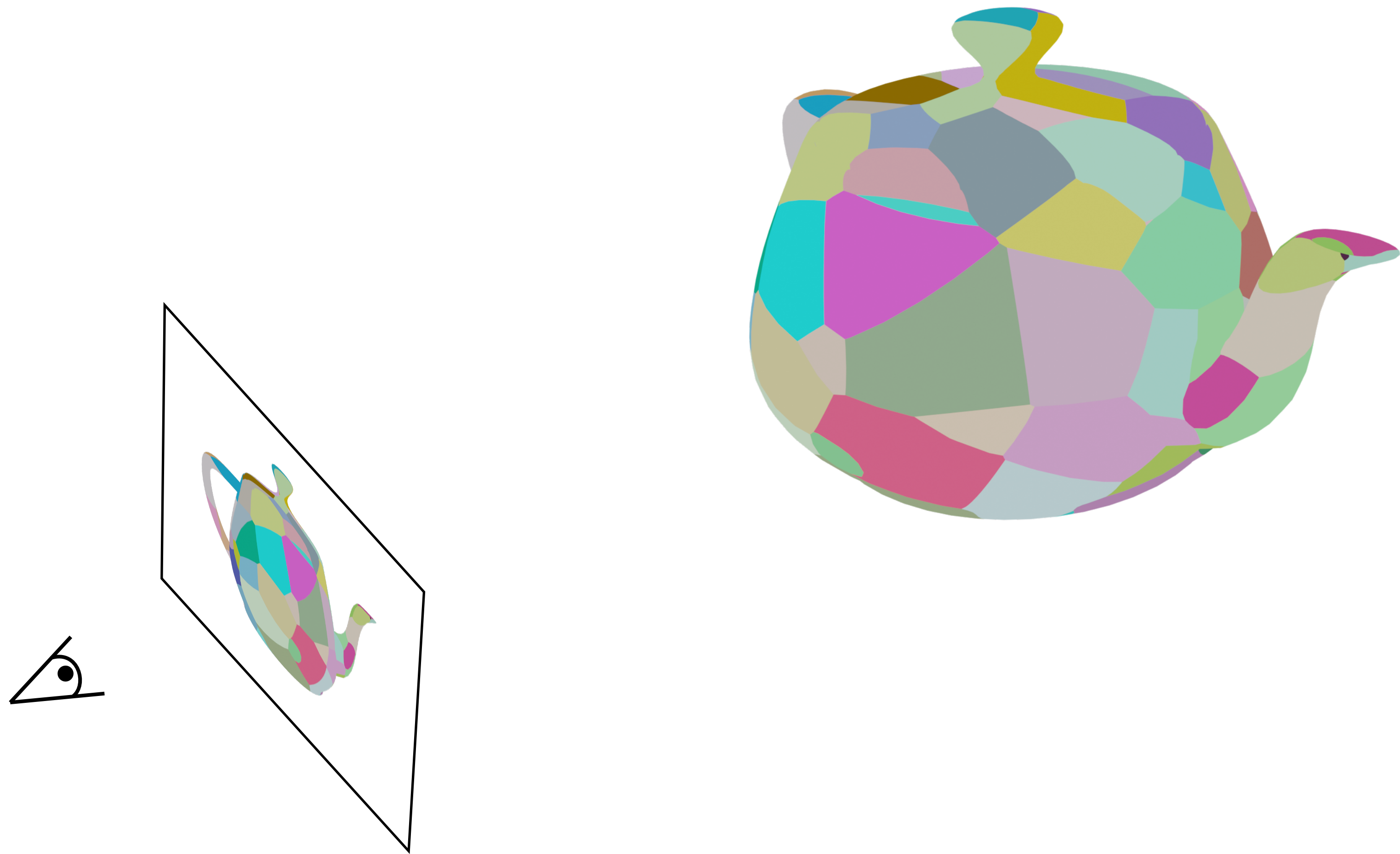


Rasterization

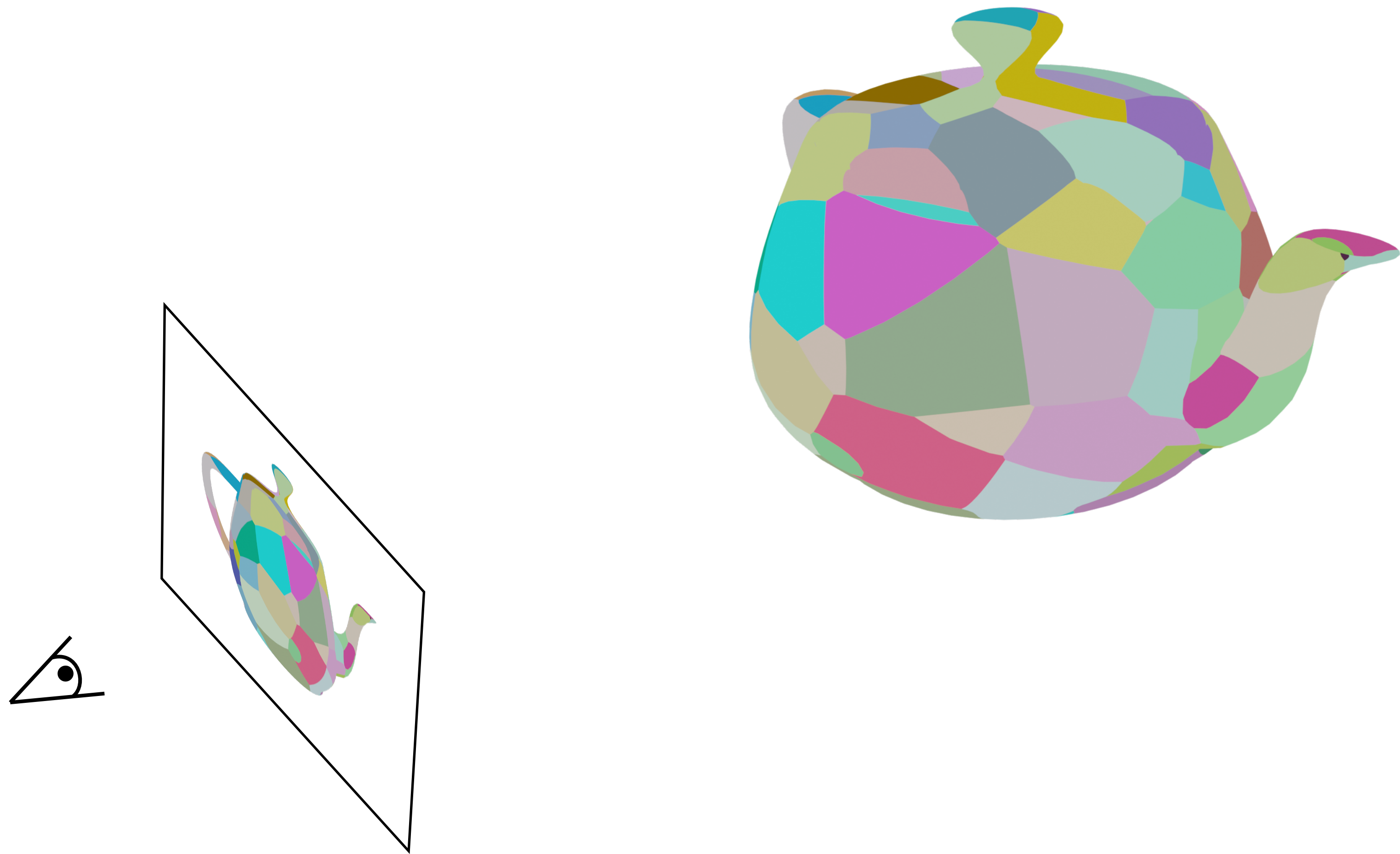
UCSD CSE 167

Tzu-Mao Li

Goal: given 3D shapes, project them
to an image

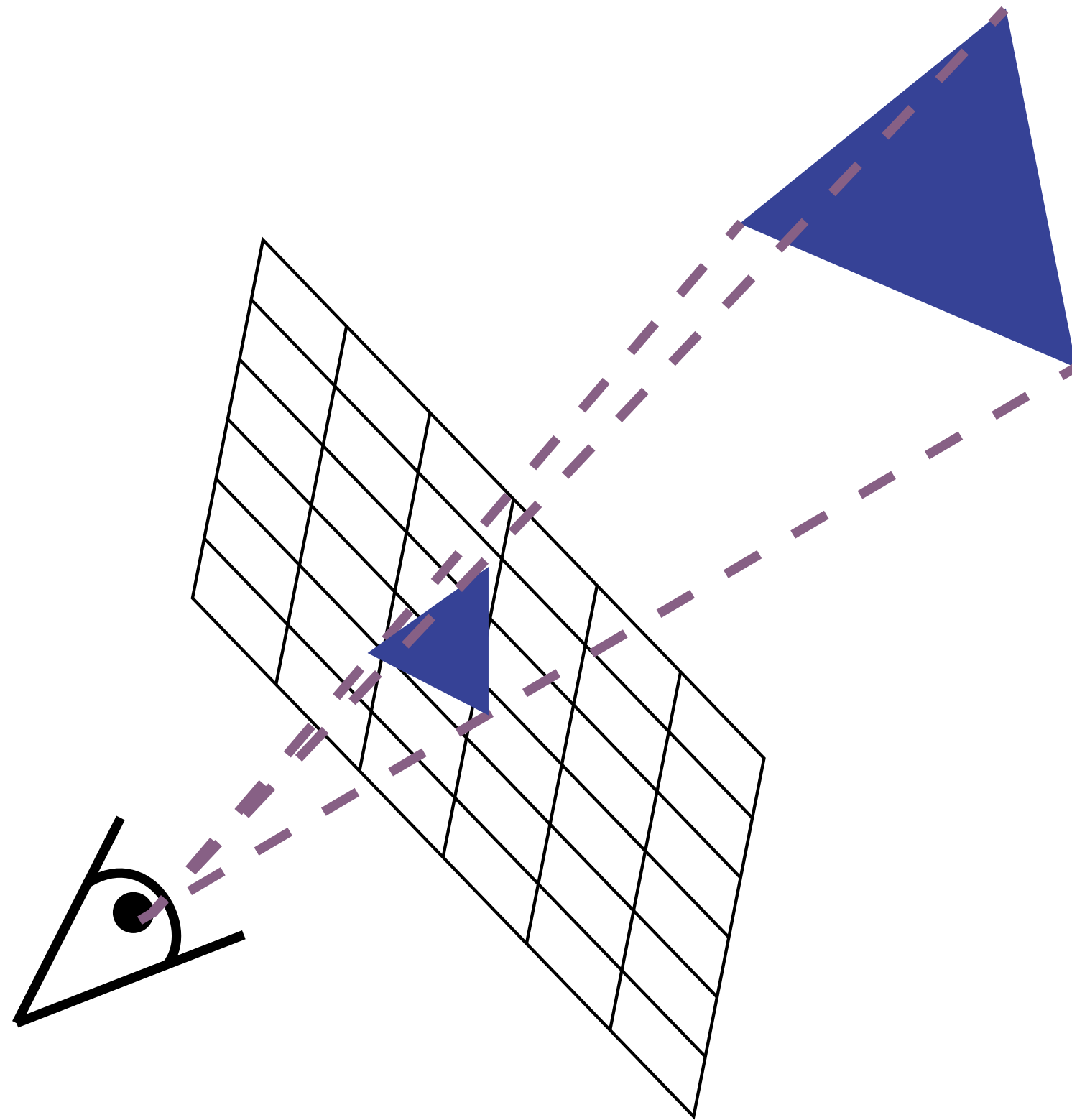


Goal: given **3D shapes**, project them
to an image



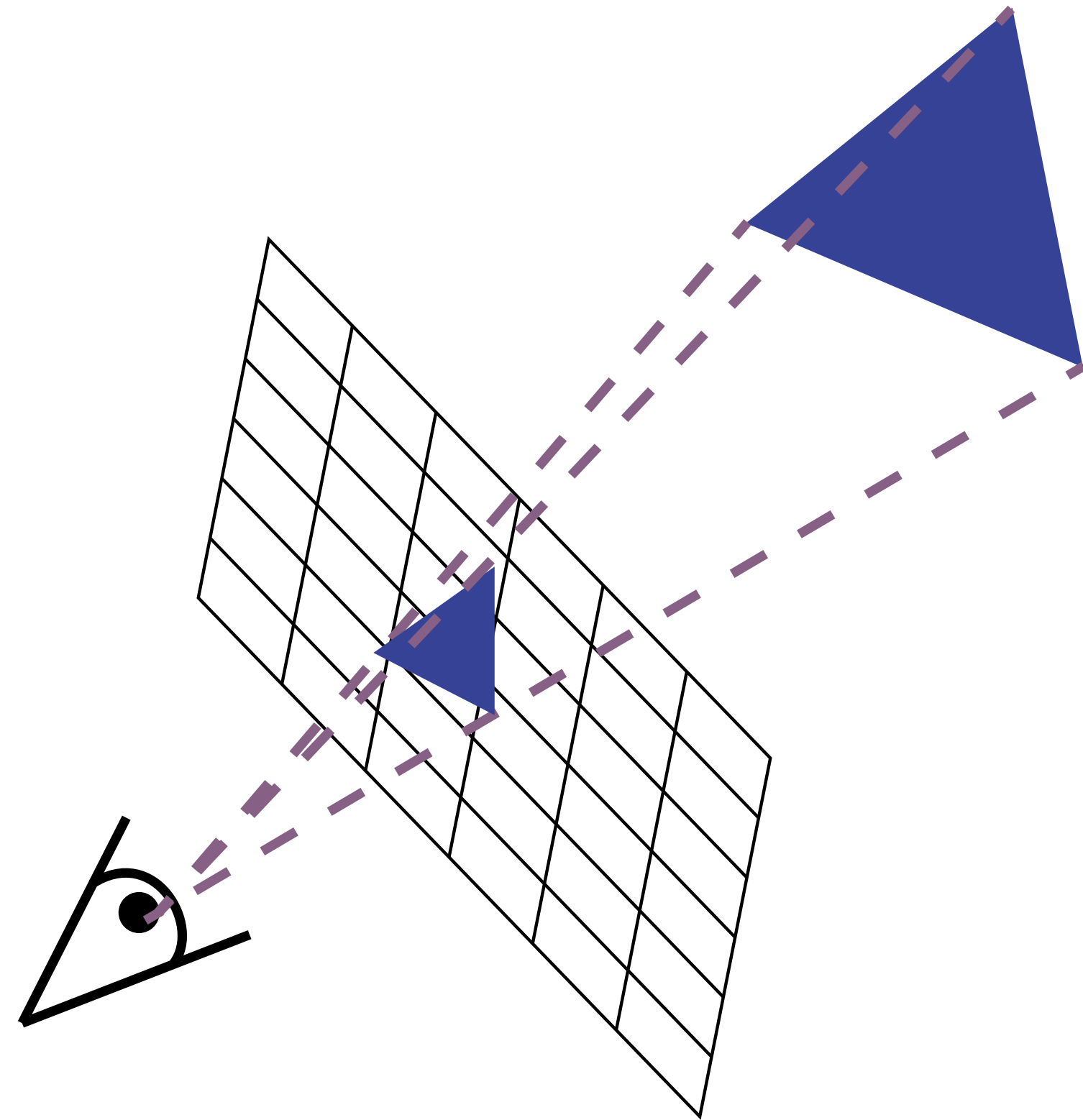
We will focus on triangles

we will talk about other shapes next time



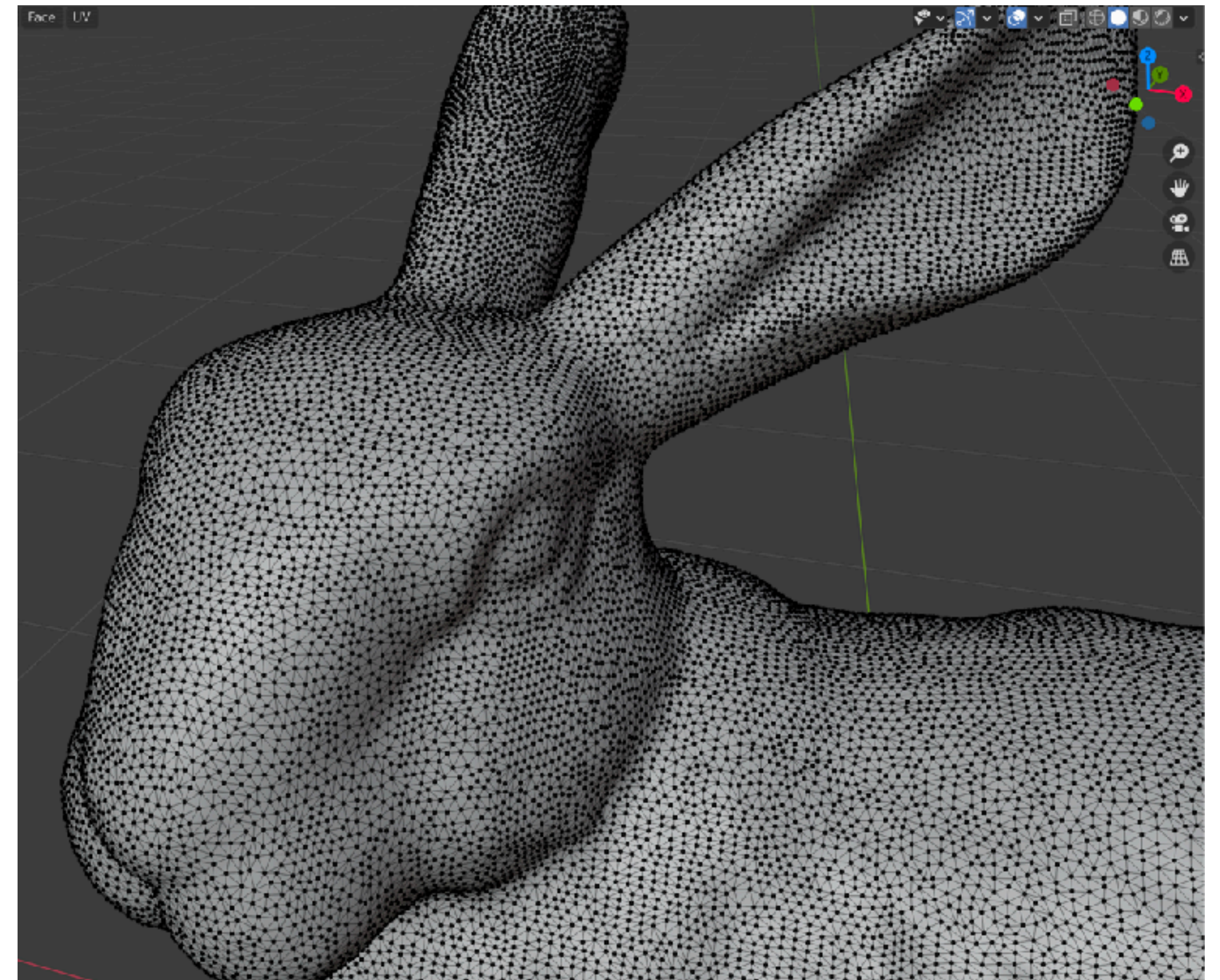
Why triangles?

- after perspective projection, a triangle is still a triangle
- so they are easier to render



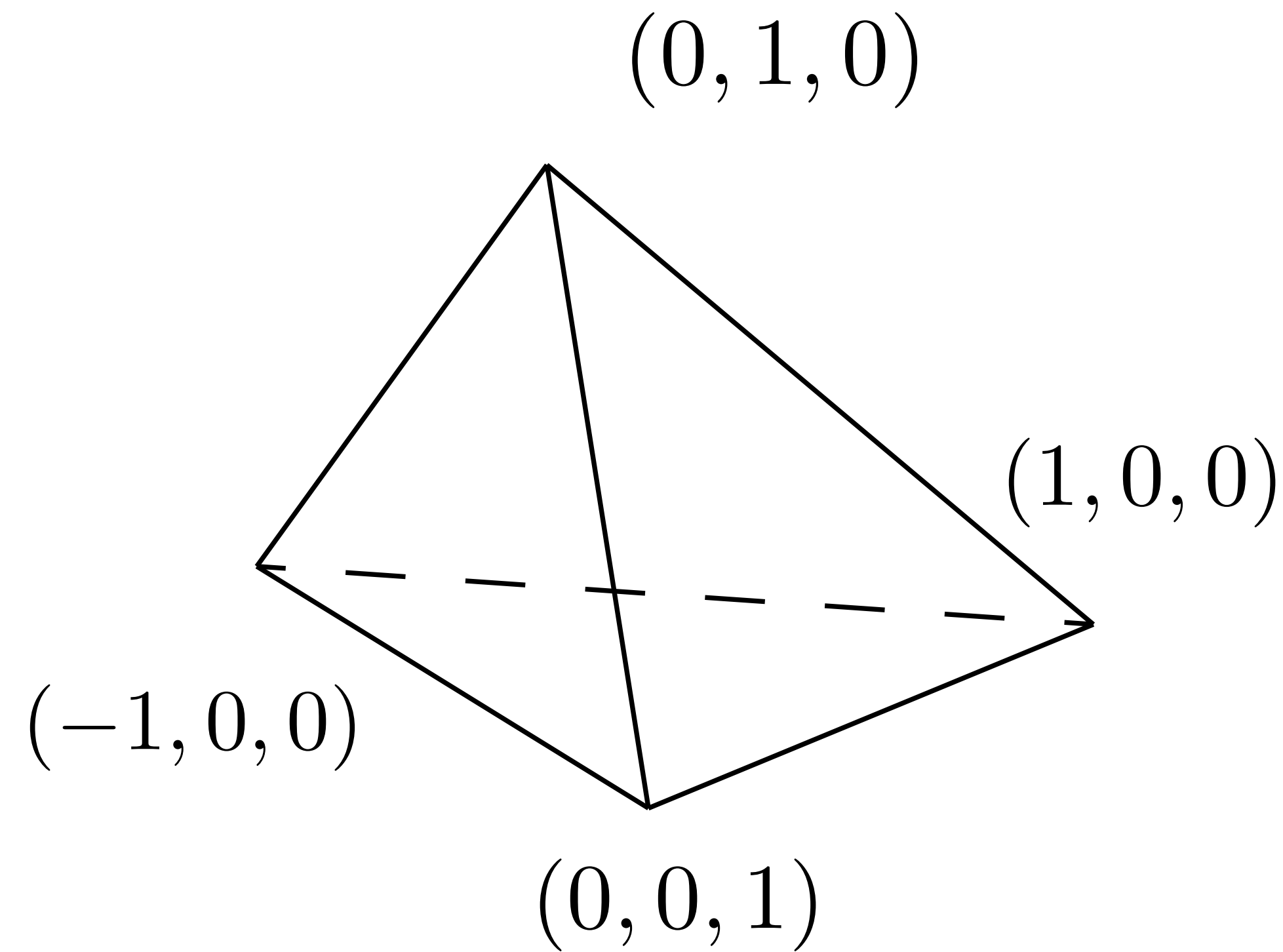
Why triangles?

- after perspective projection, a triangle is still a triangle
- so they are easier to render
- triangles can represent very complex shapes!



We use “triangle meshes” to store triangles

usually more efficient than storing individual triangles (why?)



vertices:

0: $(0, 1, 0)$

1: $(1, 0, 0)$

2: $(0, 0, 1)$

3: $(-1, 0, 0)$

faces:

0: $(0, 1, 2)$

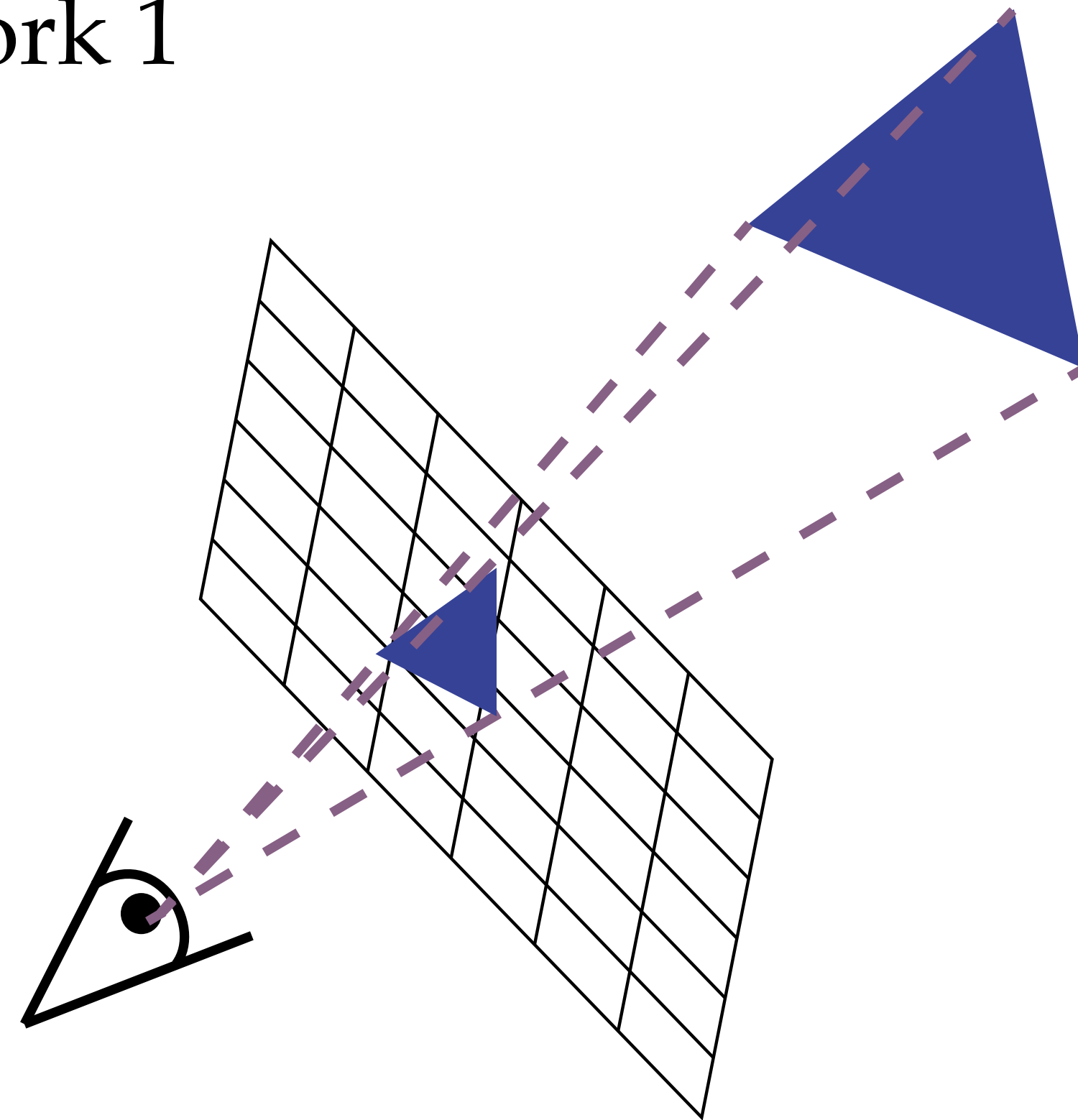
1: $(0, 1, 3)$

2: $(0, 2, 3)$

3: $(1, 2, 3)$

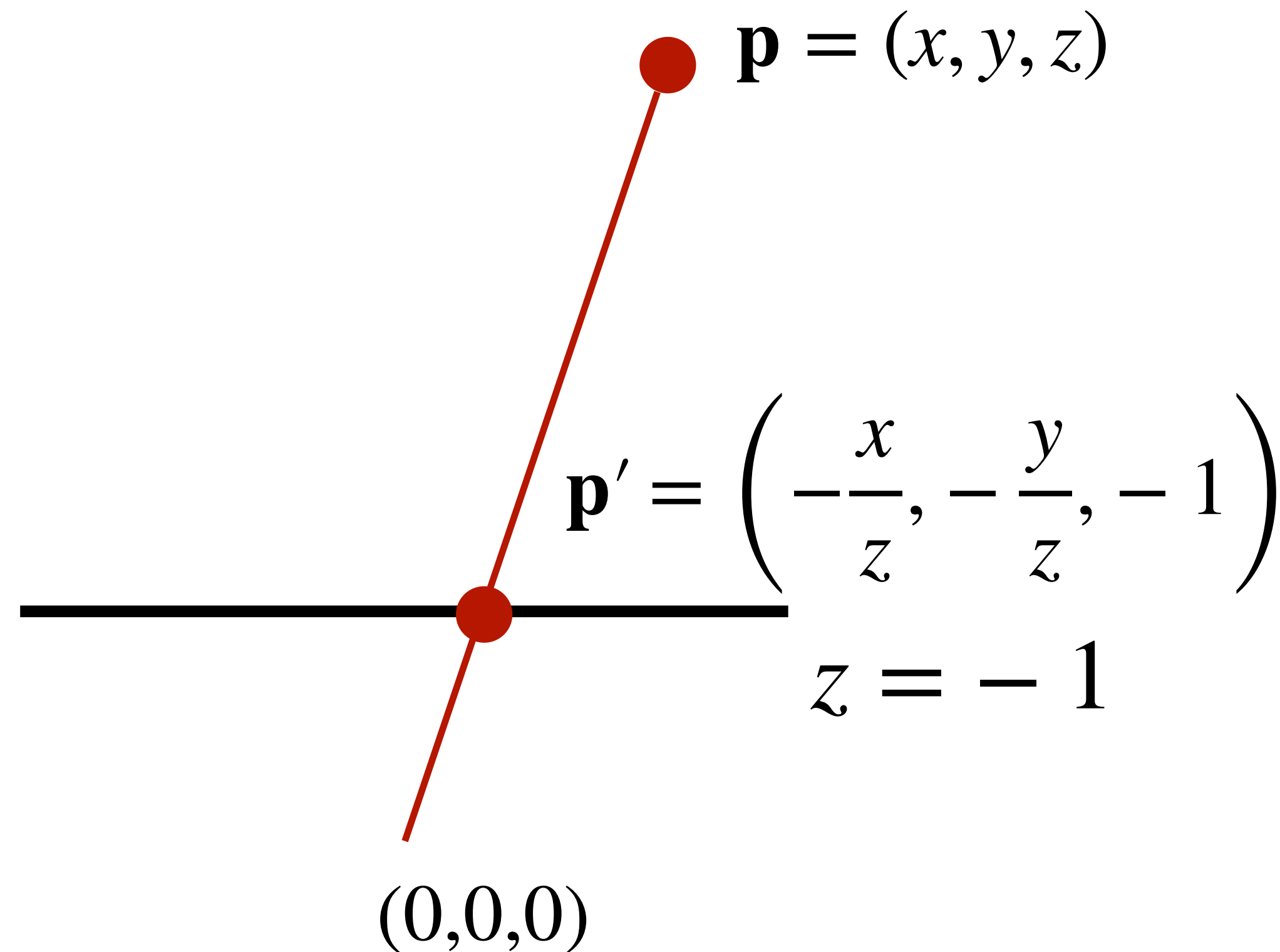
Let's start with a single triangle

plan: project the 3 vertices of the triangle to the screen,
render the 2D triangle using Homework 1



Projecting one vertex to the screen

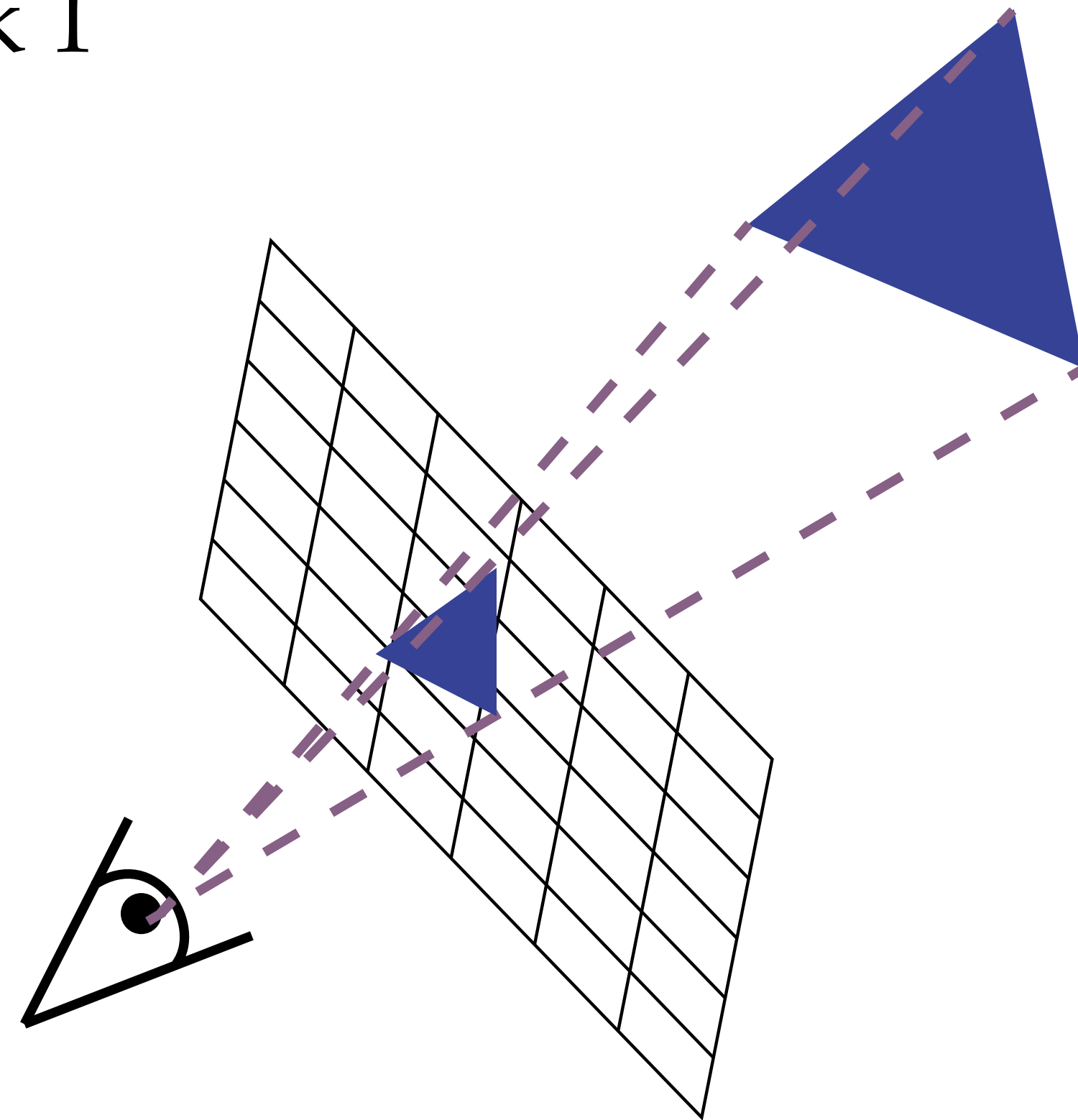
we covered it last time!



we will generalize to arbitrary camera poses this Friday

Rendering a single triangle

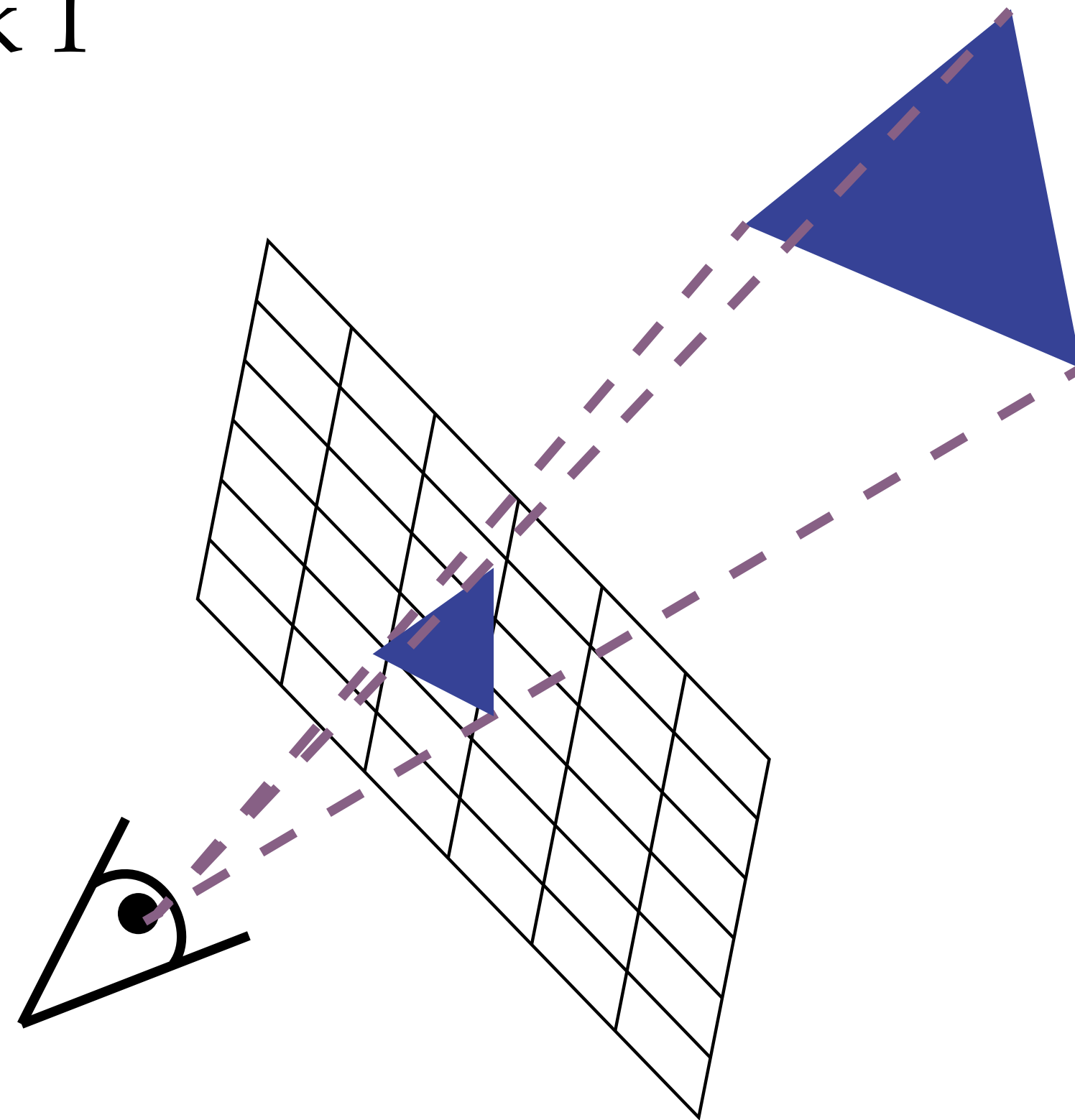
1. take the three vertices, divide their coordinates by $-z$
2. render the 2D triangle using Homework 1



Rendering a single triangle

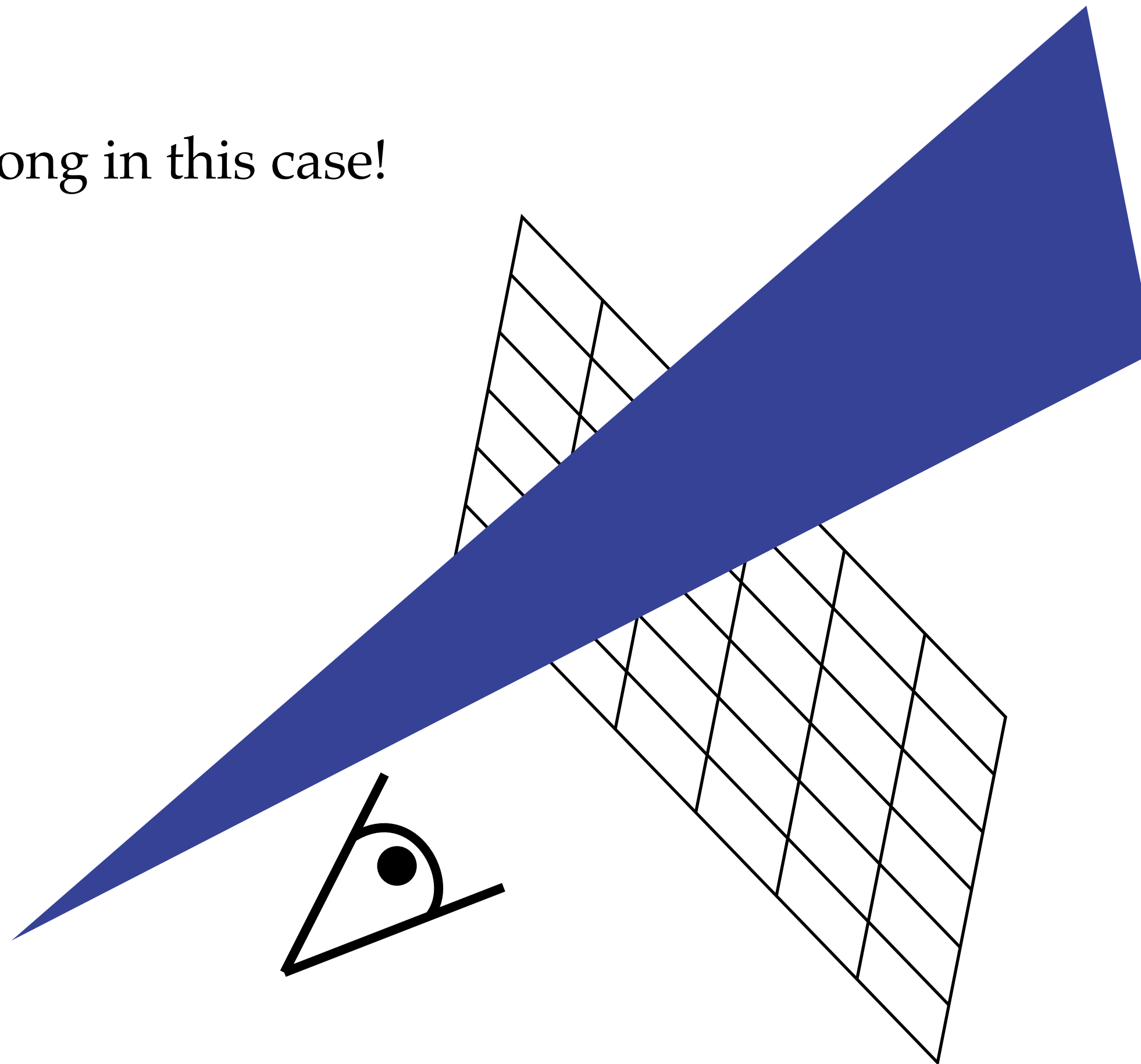
1. take the three vertices, divide their coordinates by $-z$
2. render the 2D triangle using Homework 1

are we done?



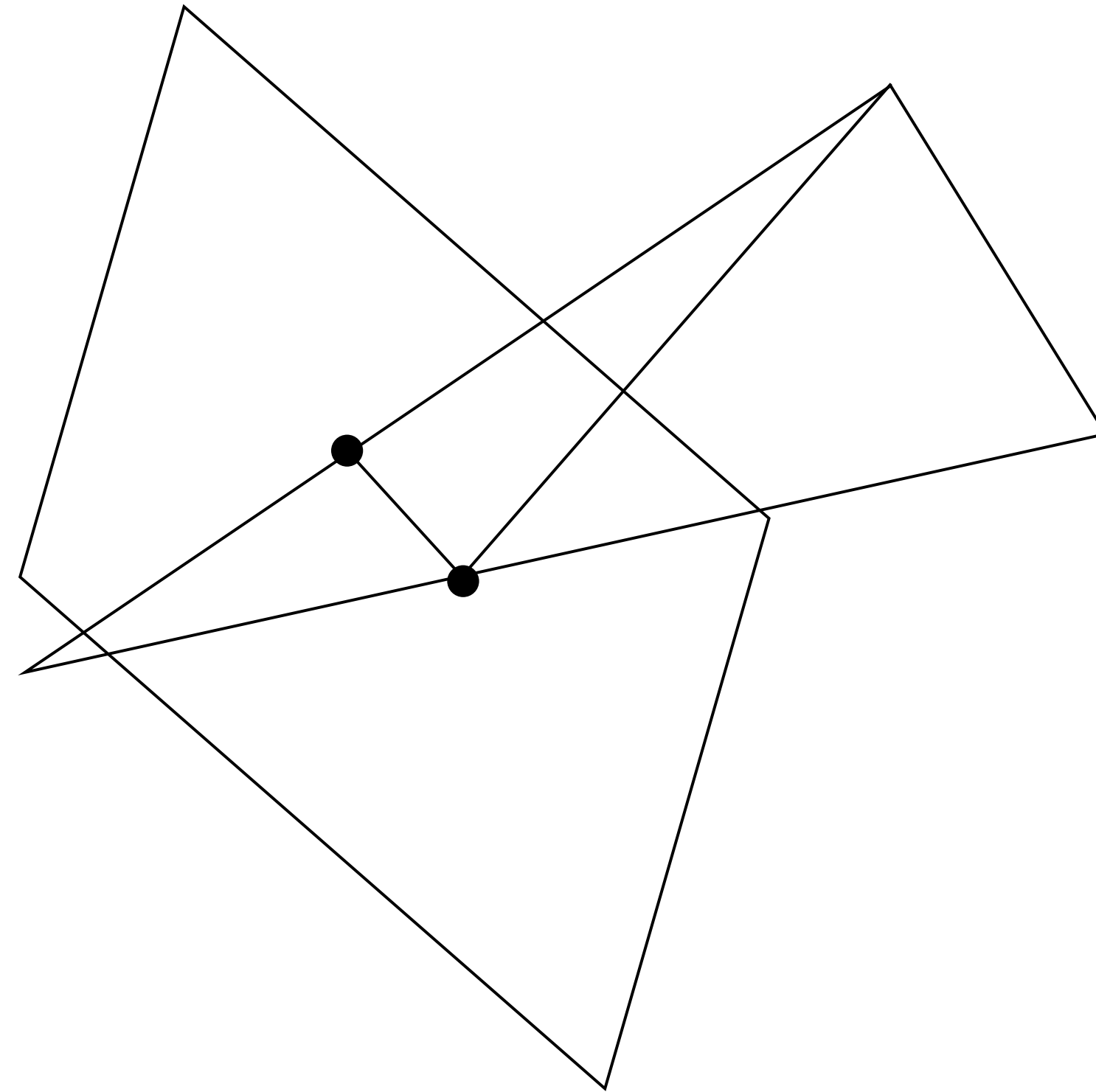
What if one or more vertices are behind the camera?

the perspective projection would be wrong in this case!



We need to “clip” the triangle

near clipping plane (e.g., $z = -0.00001$)

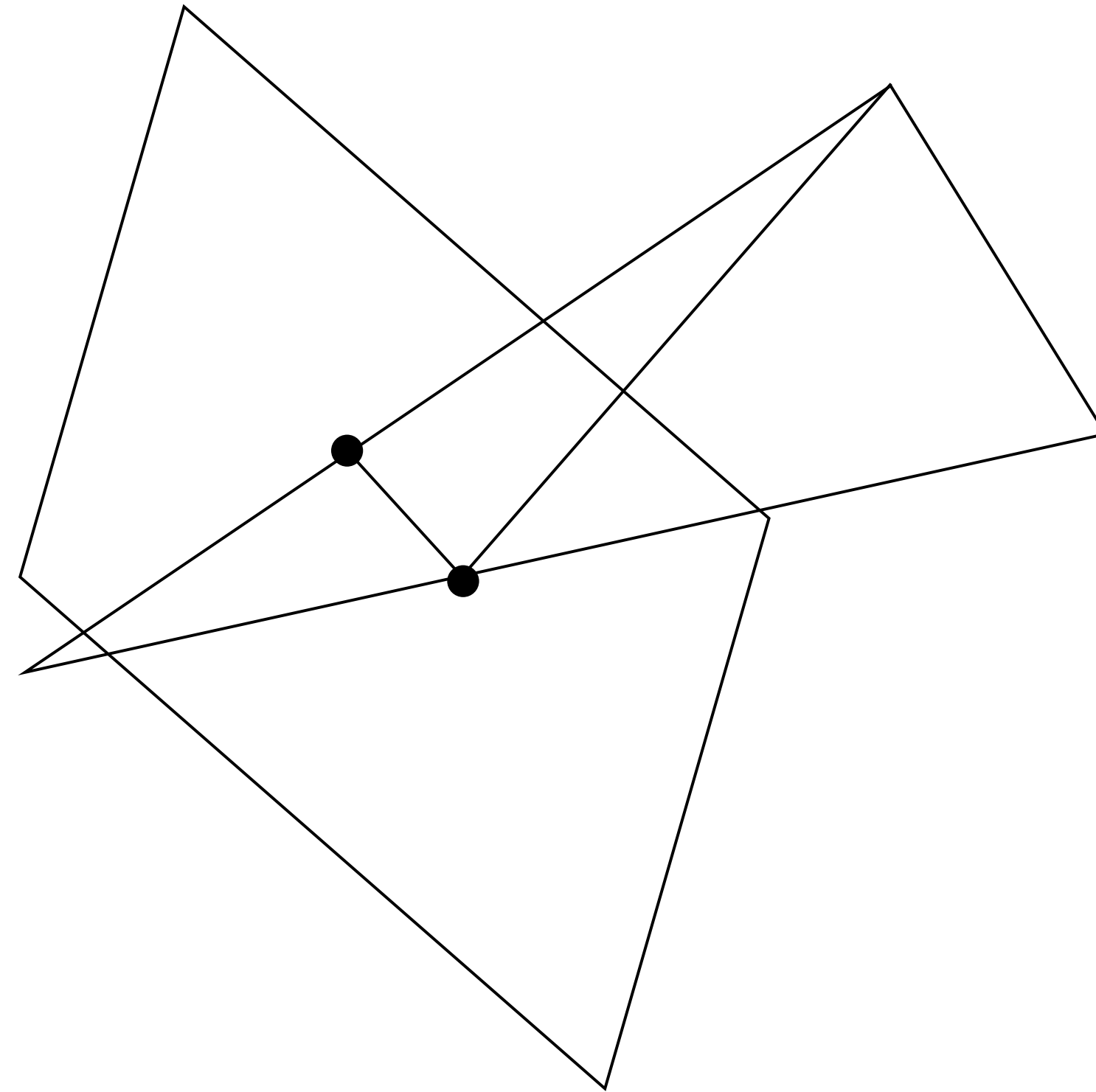


reading: Sutherland-Hodgman algorithm

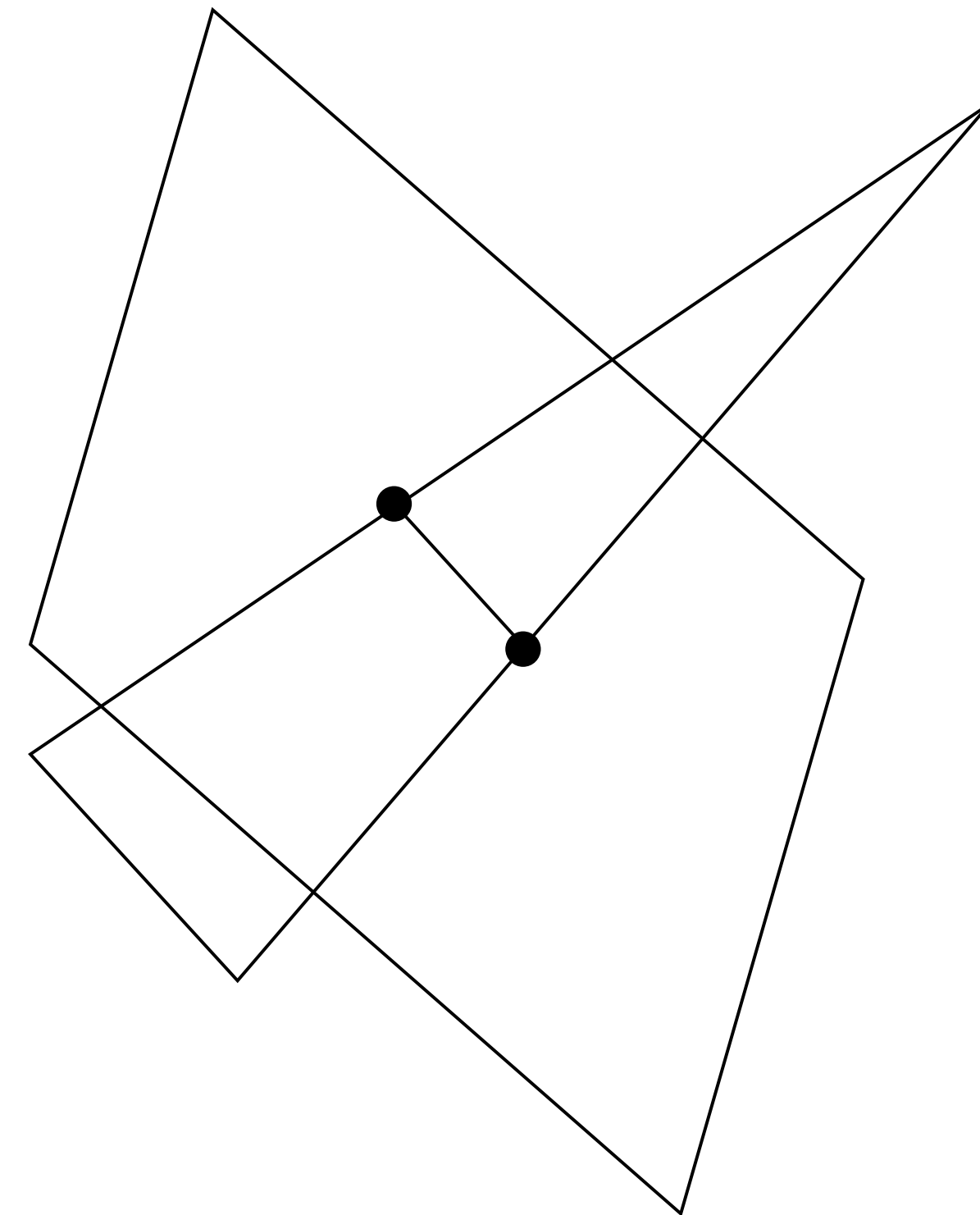
https://en.wikipedia.org/wiki/Sutherland%E2%80%93Hodgman_algorithm

We need to “clip” the triangle

near clipping plane (e.g., $z = -0.00001$)



near clipping plane

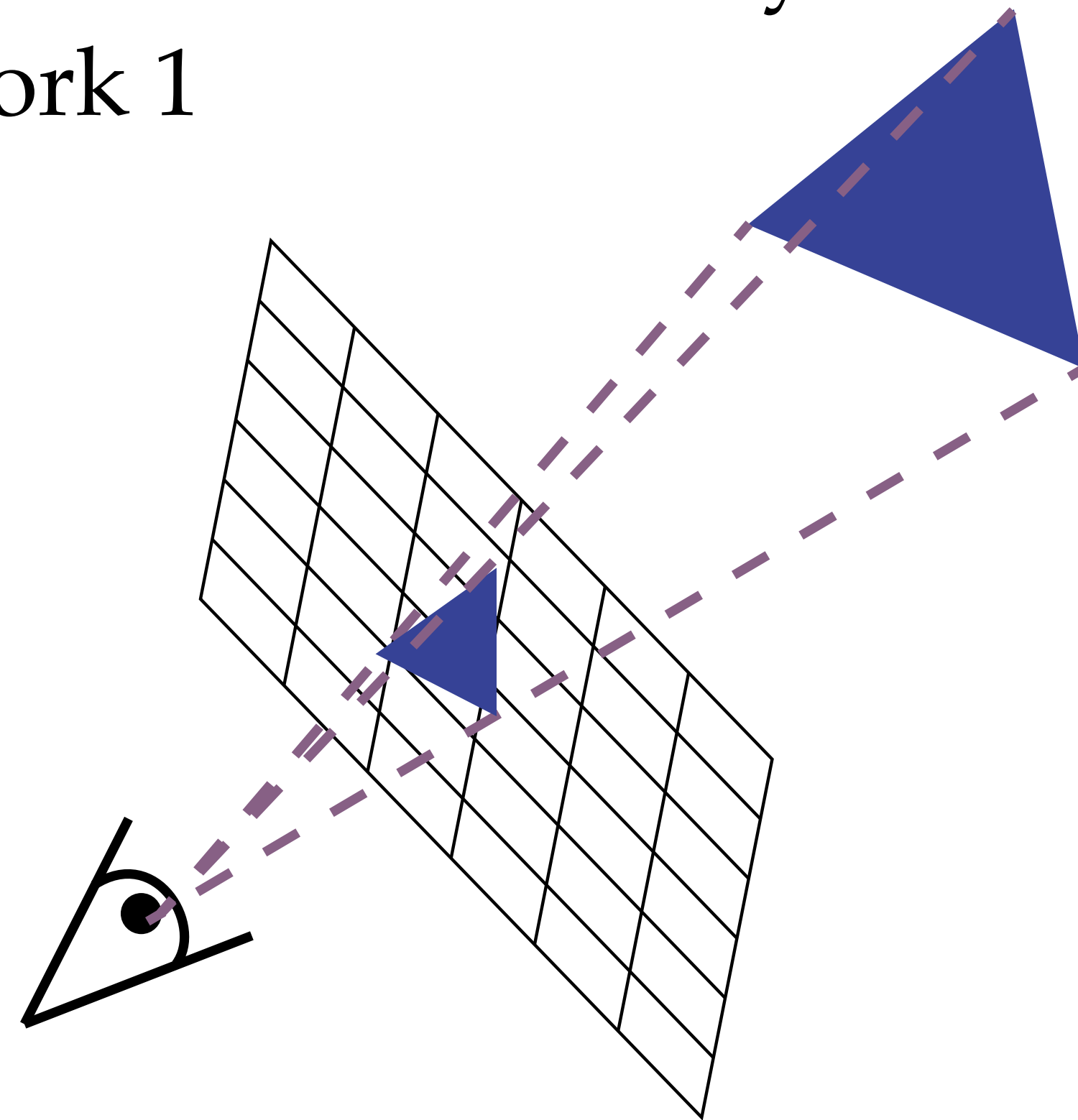


reading: Sutherland-Hodgman algorithm

https://en.wikipedia.org/wiki/Sutherland%E2%80%93Hodgman_algorithm

Rendering a single triangle

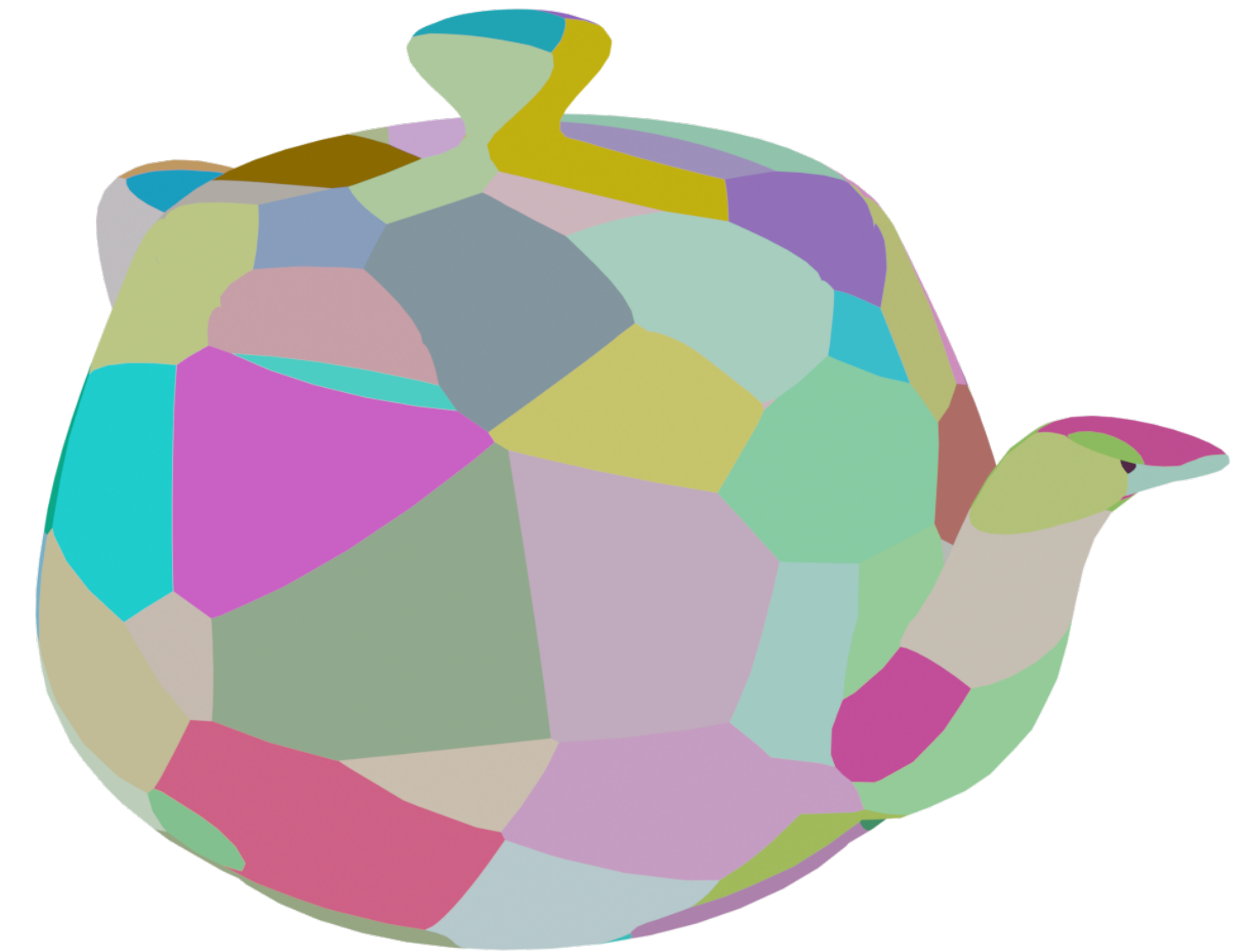
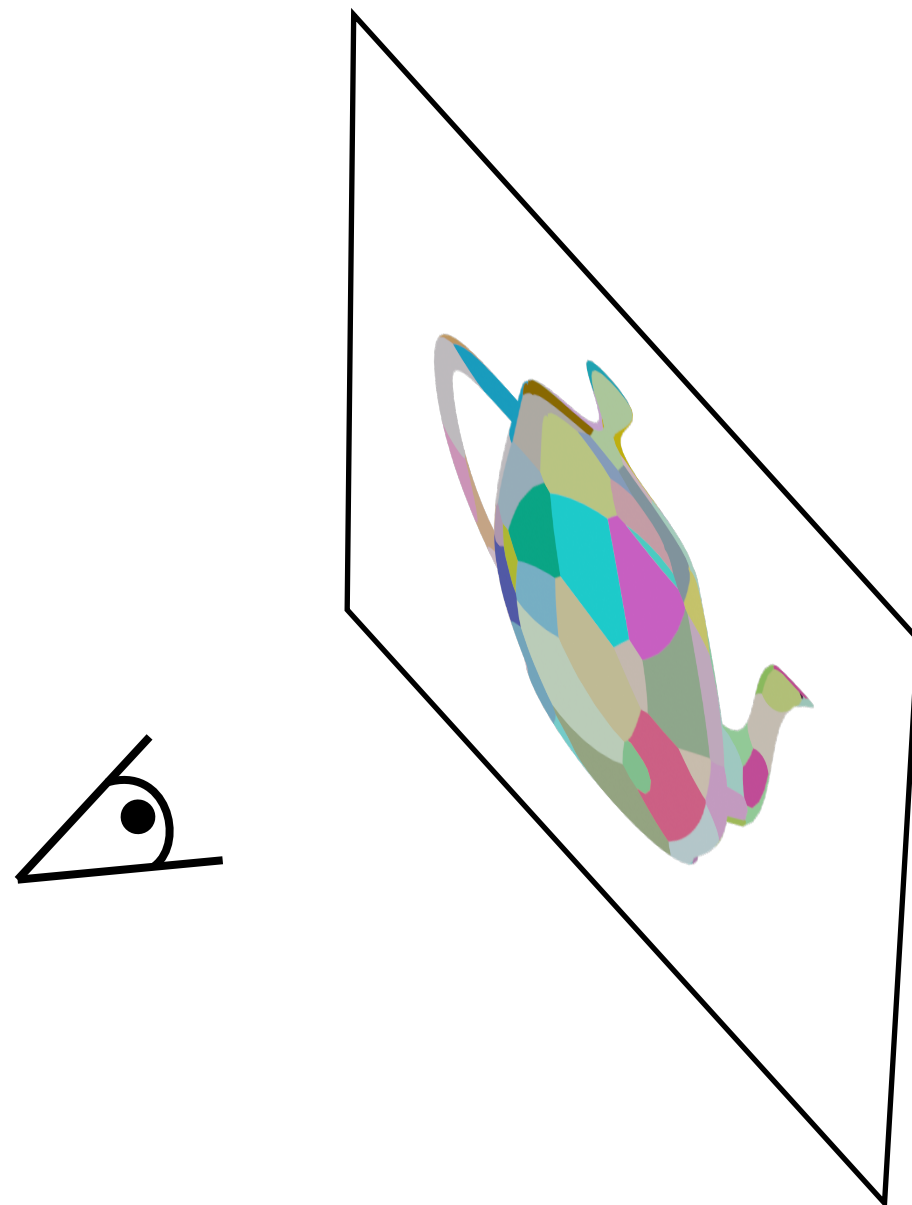
1. clip the triangle against near clipping plane
2. take the three (or four) vertices, divide their coordinates by $-z$
3. render the 2D triangle(s) using Homework 1



What about multiple triangles?

```
2d_triangles = []  
for each triangle:  
    clip the triangle  
    take the 3-4 vertices and divide them by  $-z$   
    2d_triangles.push(triangles)  
render 2d_triangles using Homework 1
```

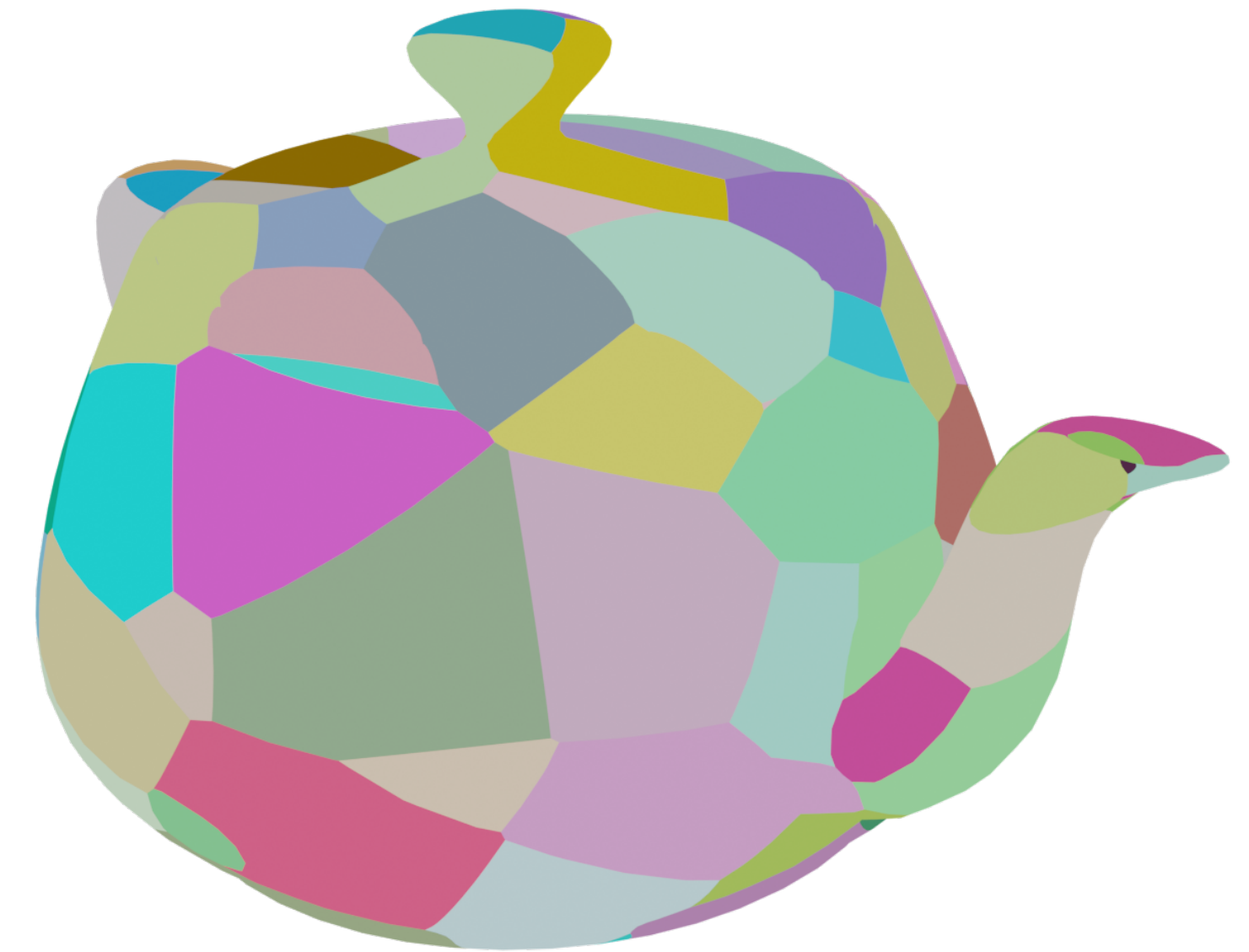
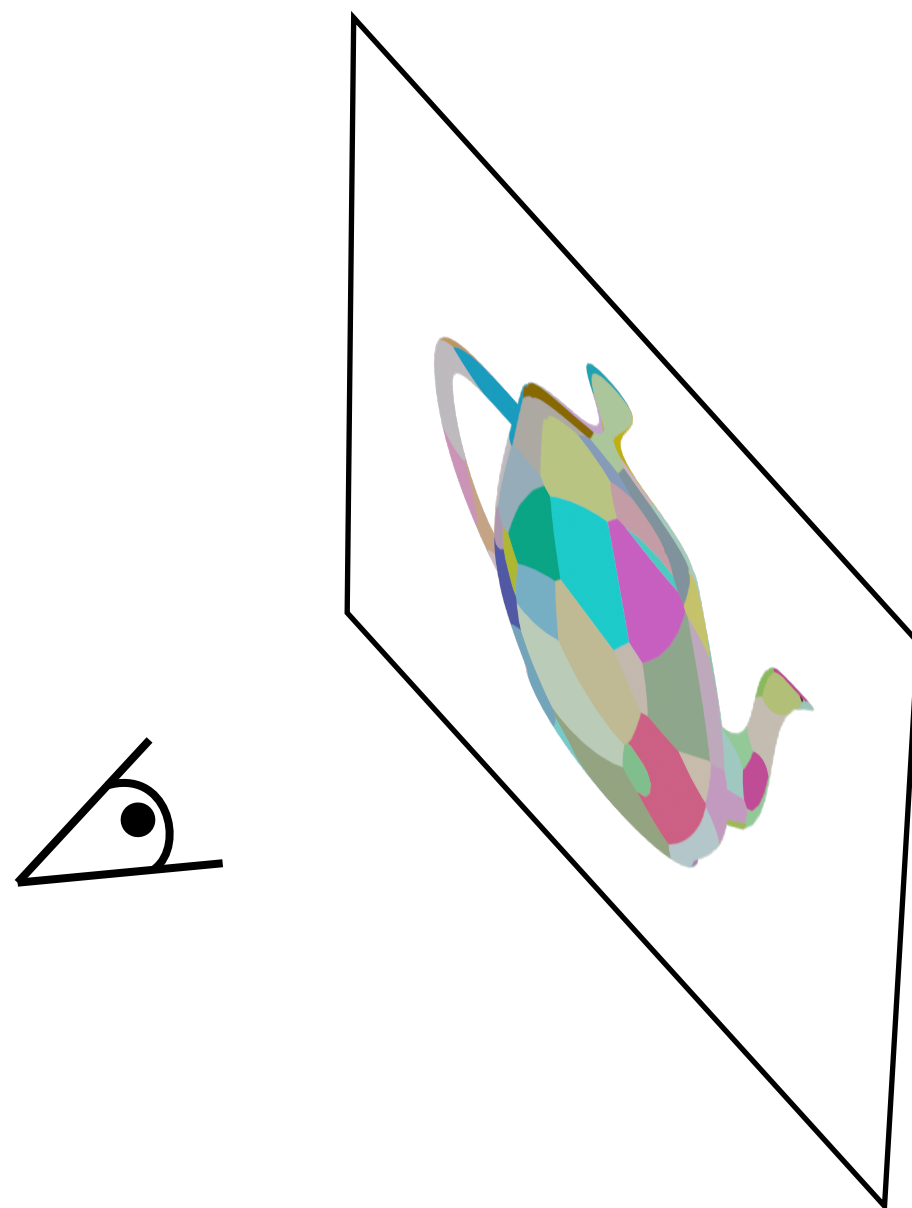
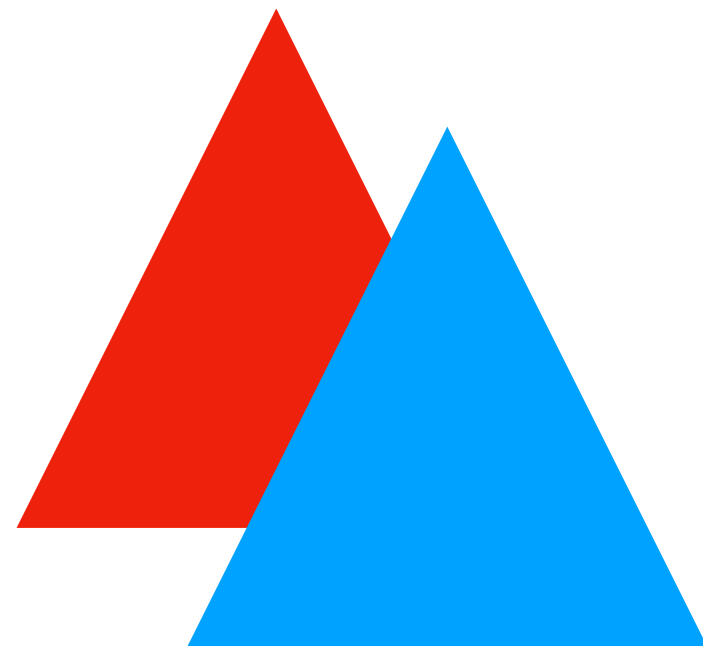
are we done?



What about multiple triangles?

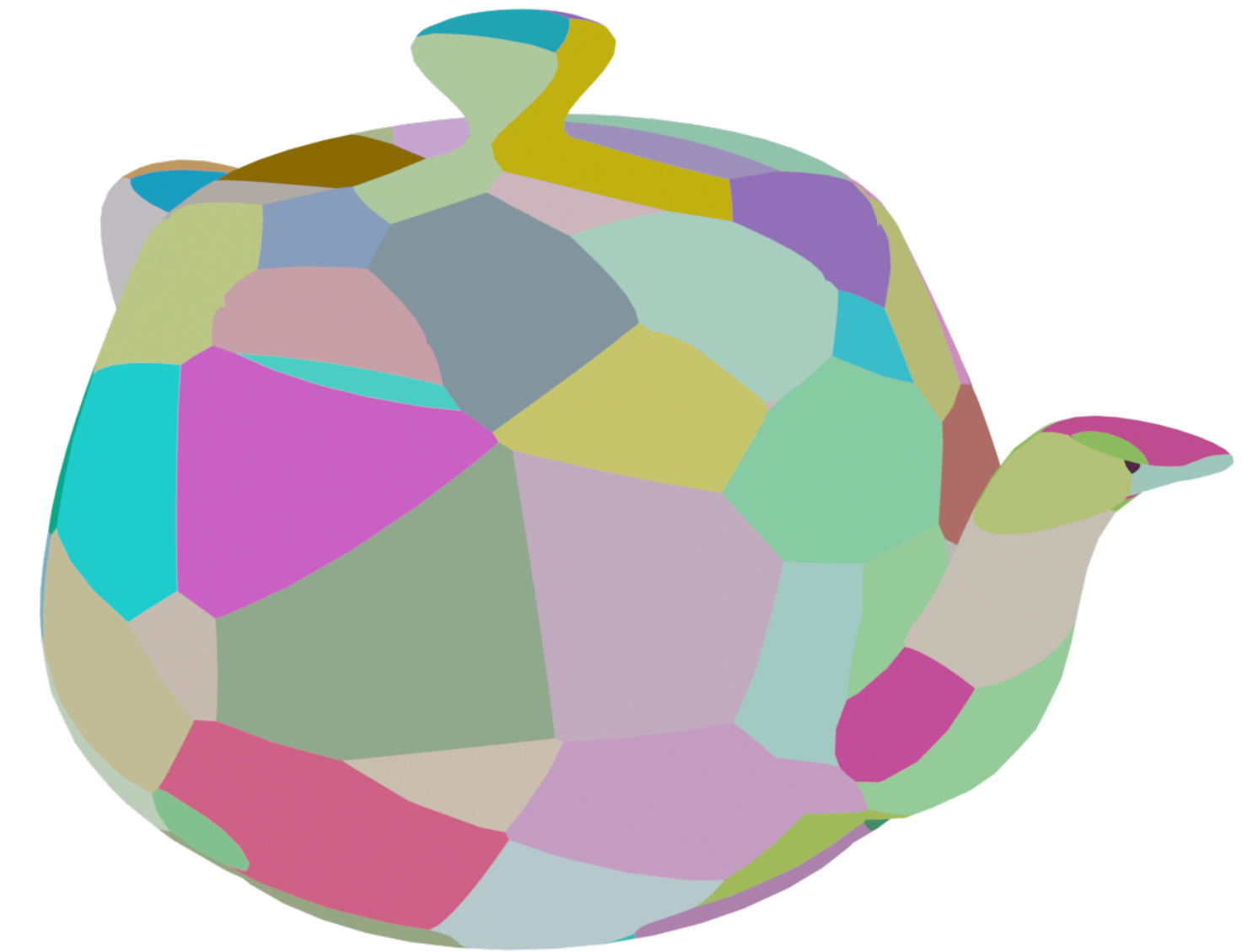
```
2d_triangles = []  
for each triangle:  
    clip the triangle  
    take the 3-4 vertices and divide them by  $-z$   
    2d_triangles.push(triangles)  
render 2d_triangles using Homework 1
```

triangles can block each other!

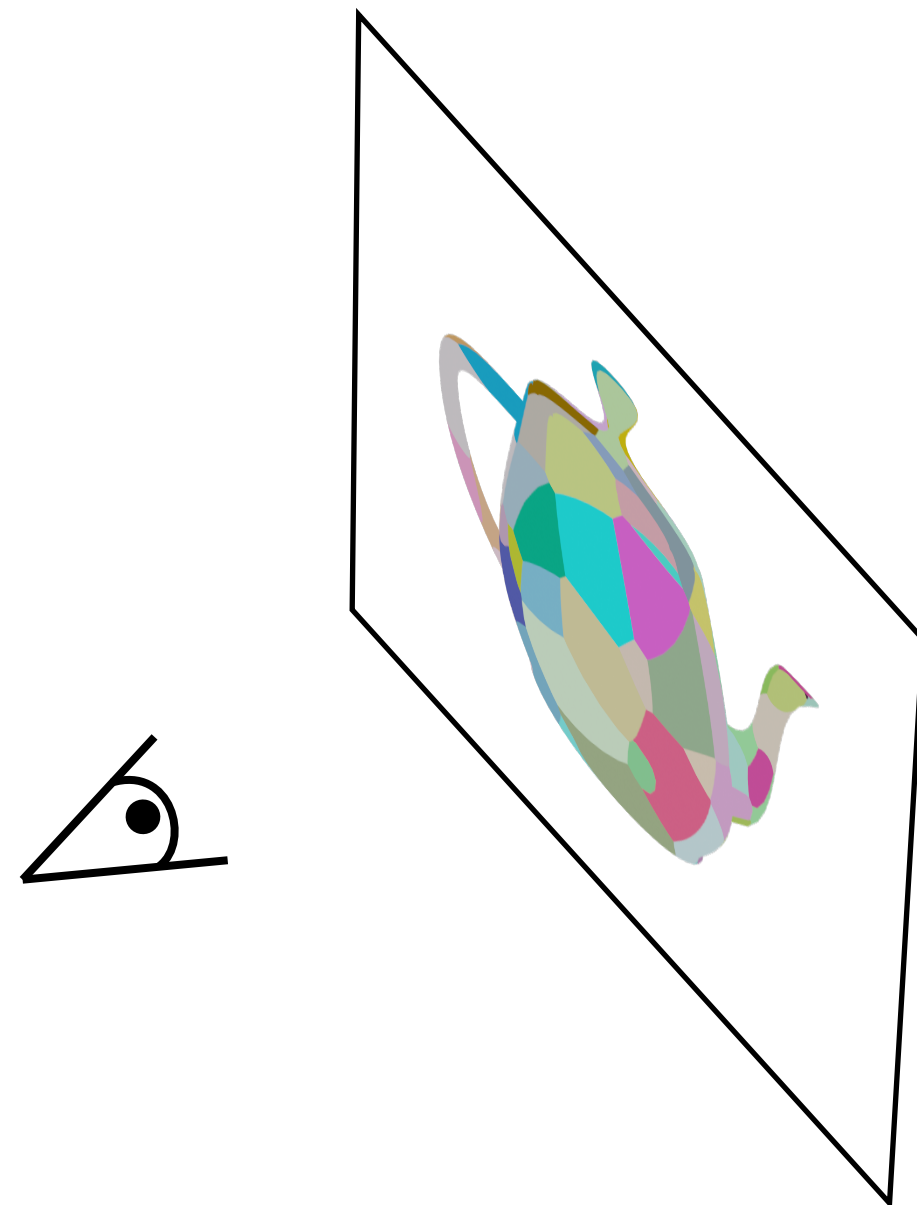


Attempt 1: painter's algorithm

```
2d_triangles = []  
for each triangle:  
    clip the triangle  
    take the 3-4 vertices and divide them by  $-z$   
    2d_triangles.push(triangles)  
sort 2d_triangles by their mean z coordinates  
render 2d_triangles using Homework 1
```



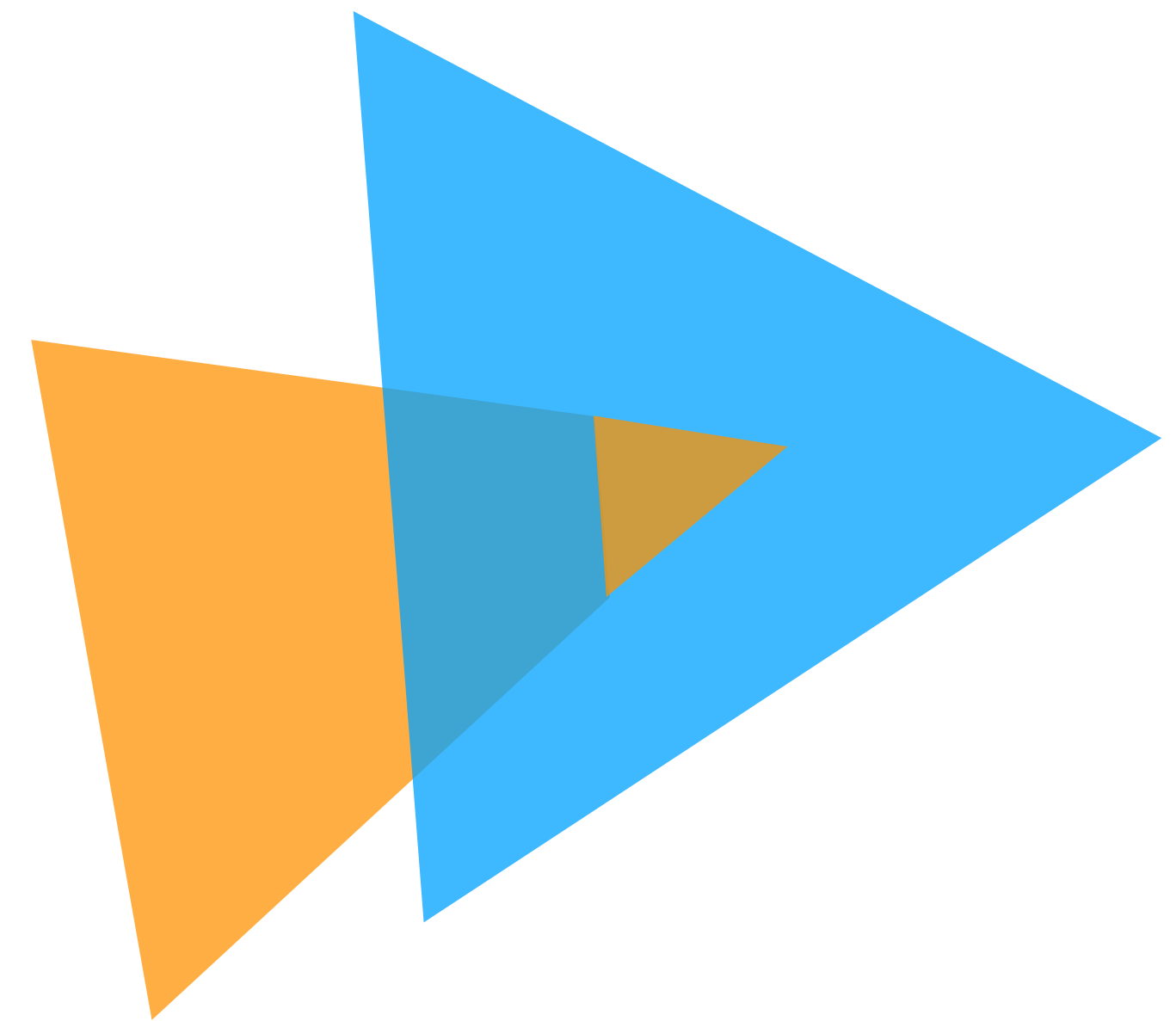
will this work?



Painter's algorithm can often fail

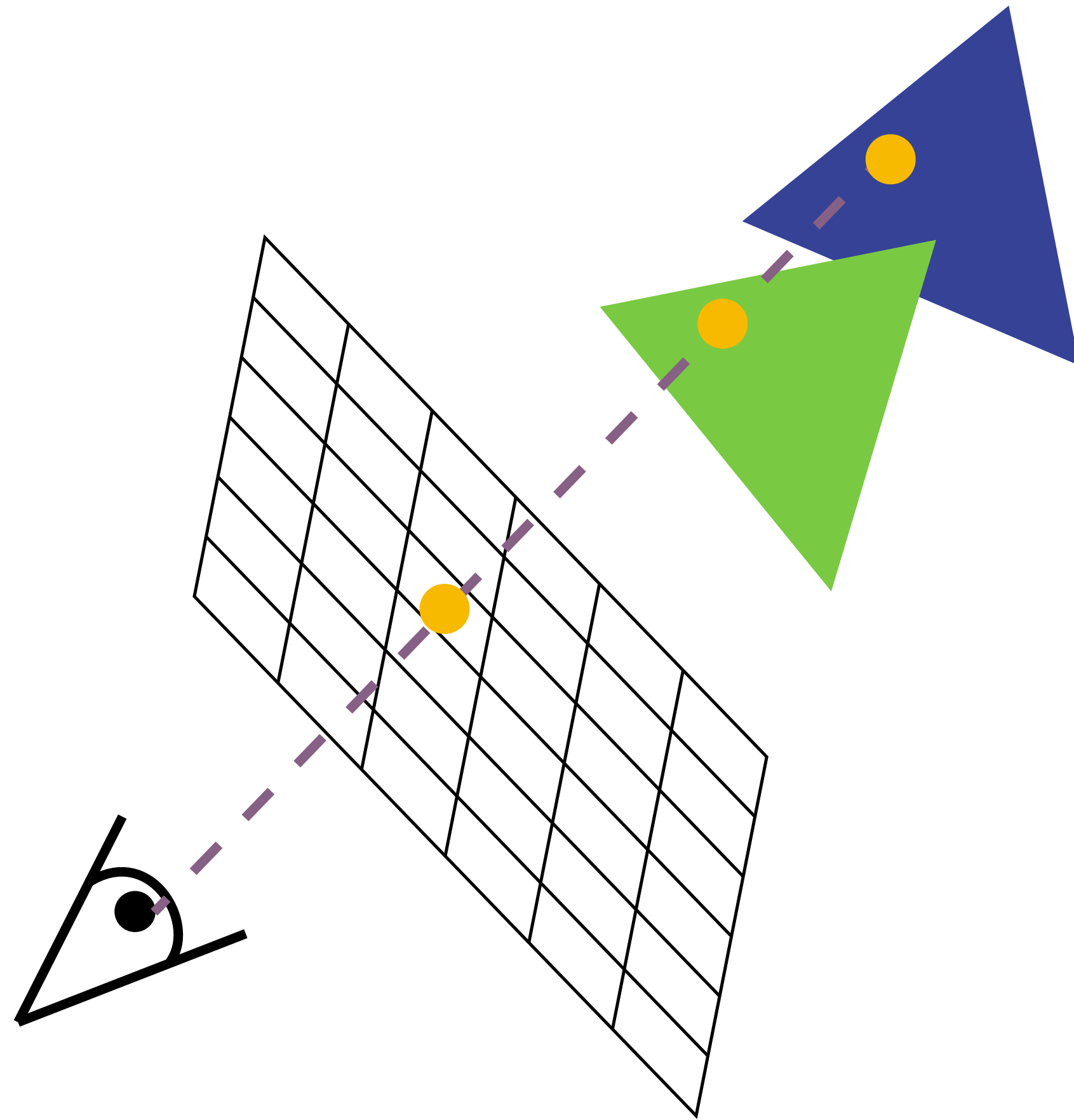


cyclic order



interpenetration

Instead, we need to figure out
the depth correspond to the pixel sample



We need barycentric coordinates

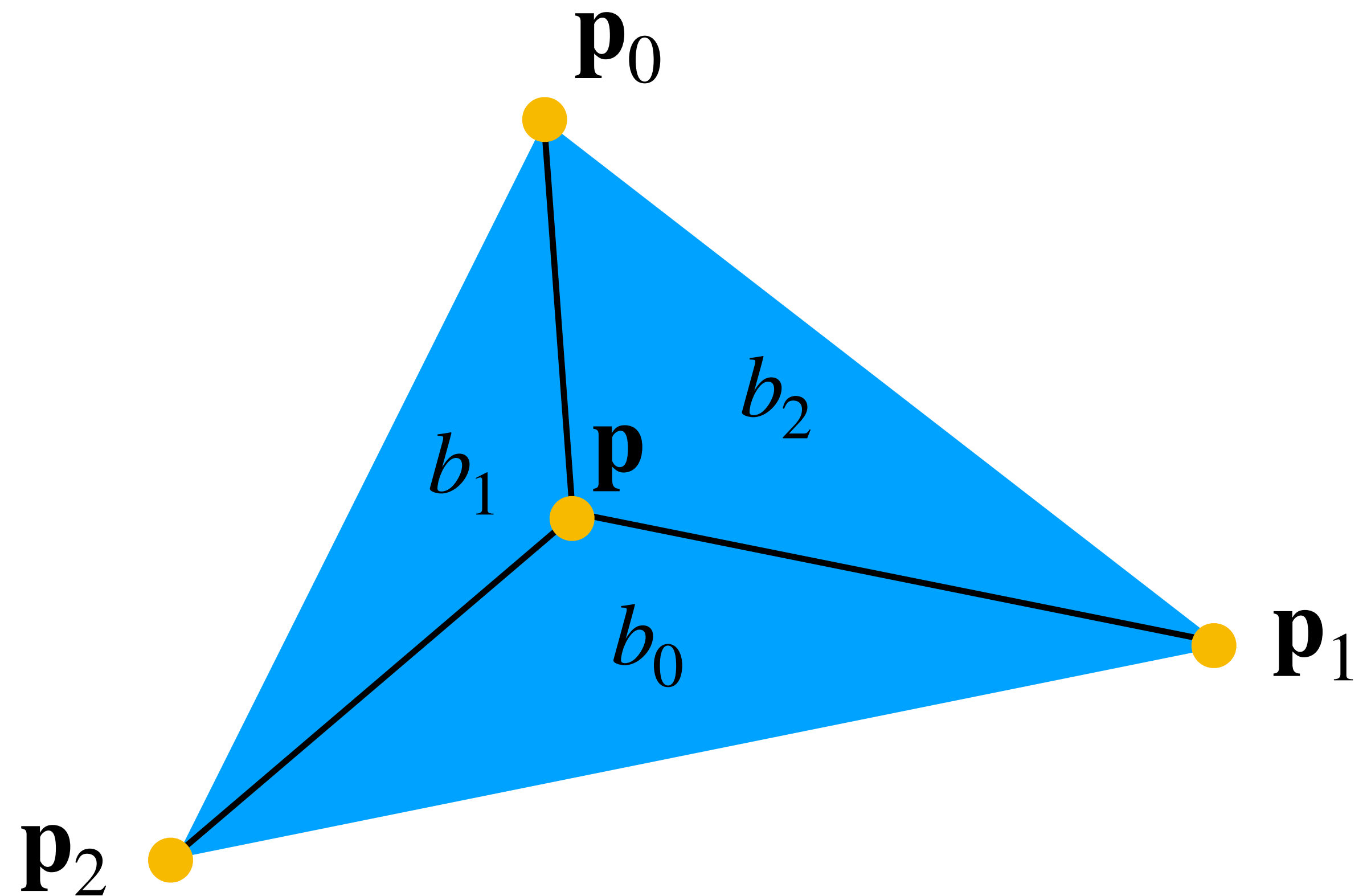
each point \mathbf{p} inside the triangle can be written as

$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

$$b_0, b_1, b_2 \geq 0$$

$$b_0 + b_1 + b_2 = 1$$

if we have the barycentric coordinates,
we get the depth value



We need barycentric coordinates

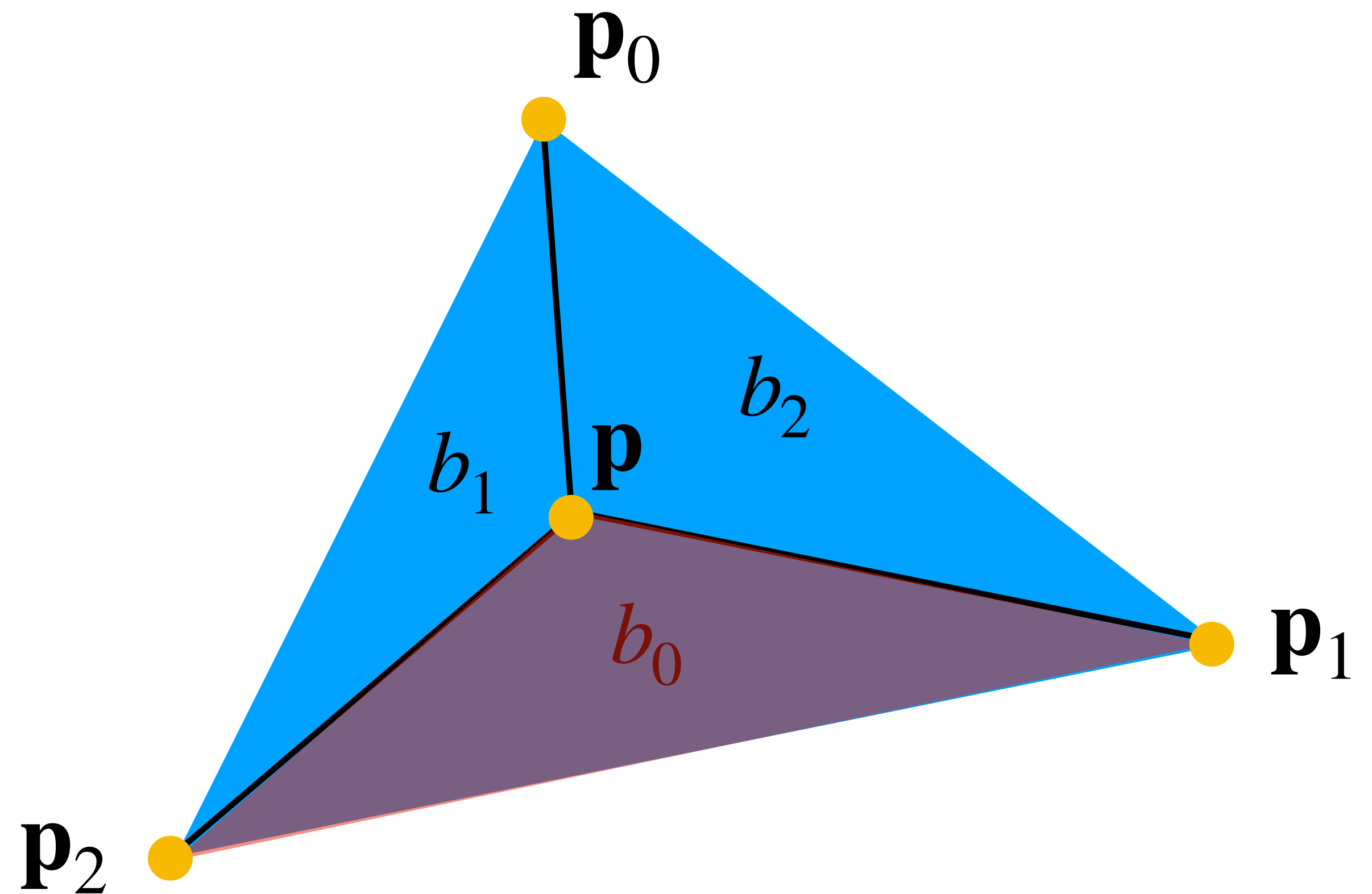
each point \mathbf{p} inside the triangle can be written as

$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

$$b_0, b_1, b_2 \geq 0$$

$$b_0 + b_1 + b_2 = 1$$

$$b_0 = \frac{\text{area}(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$



We need barycentric coordinates

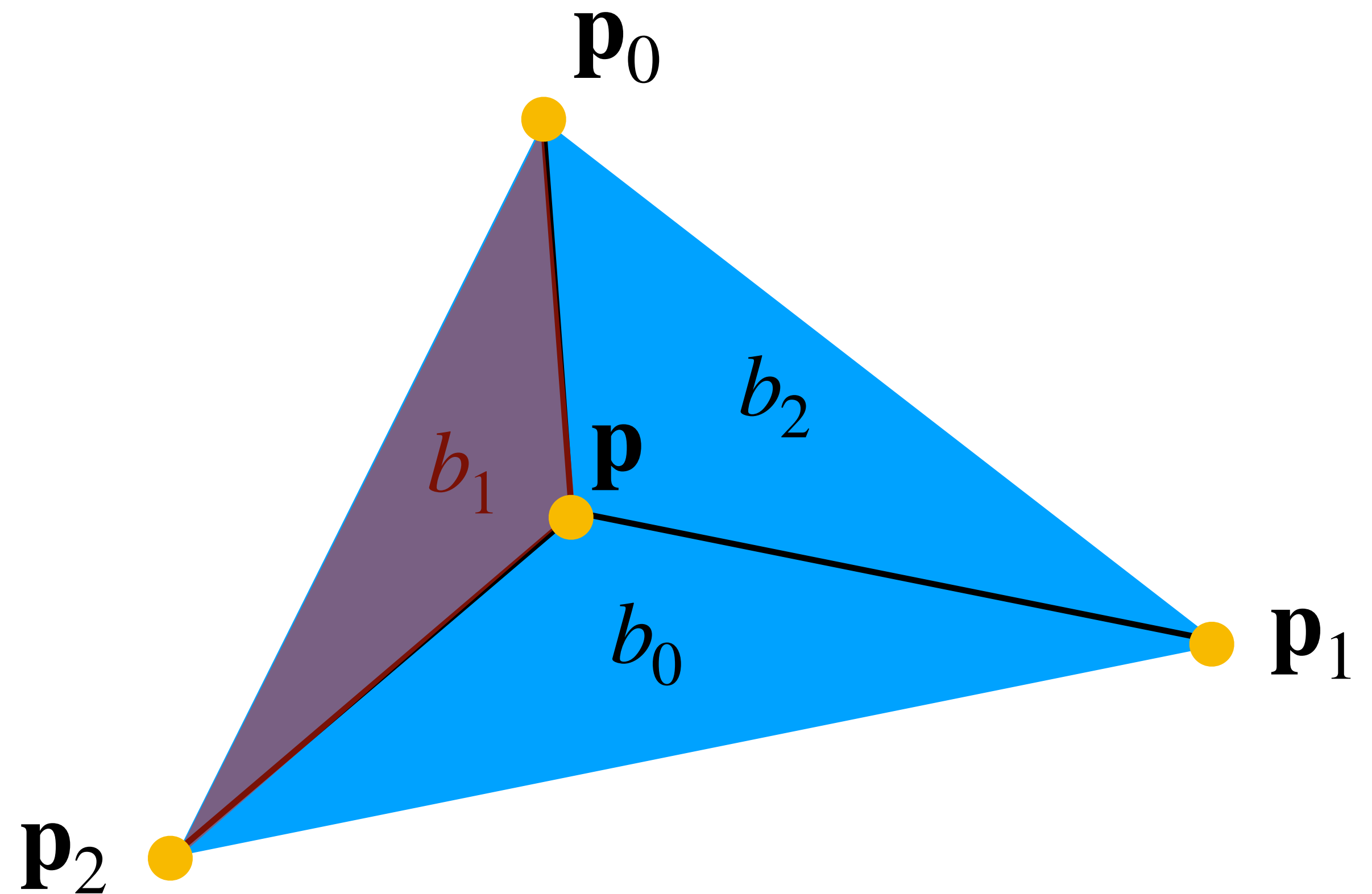
each point \mathbf{p} inside the triangle can be written as

$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

$$b_0, b_1, b_2 \geq 0$$

$$b_0 + b_1 + b_2 = 1$$

$$b_0 = \frac{\text{area}(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)} \quad b_1 = \frac{\text{area}(\mathbf{p}_0, \mathbf{p}, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$



We need barycentric coordinates

each point \mathbf{p} inside the triangle can be written as

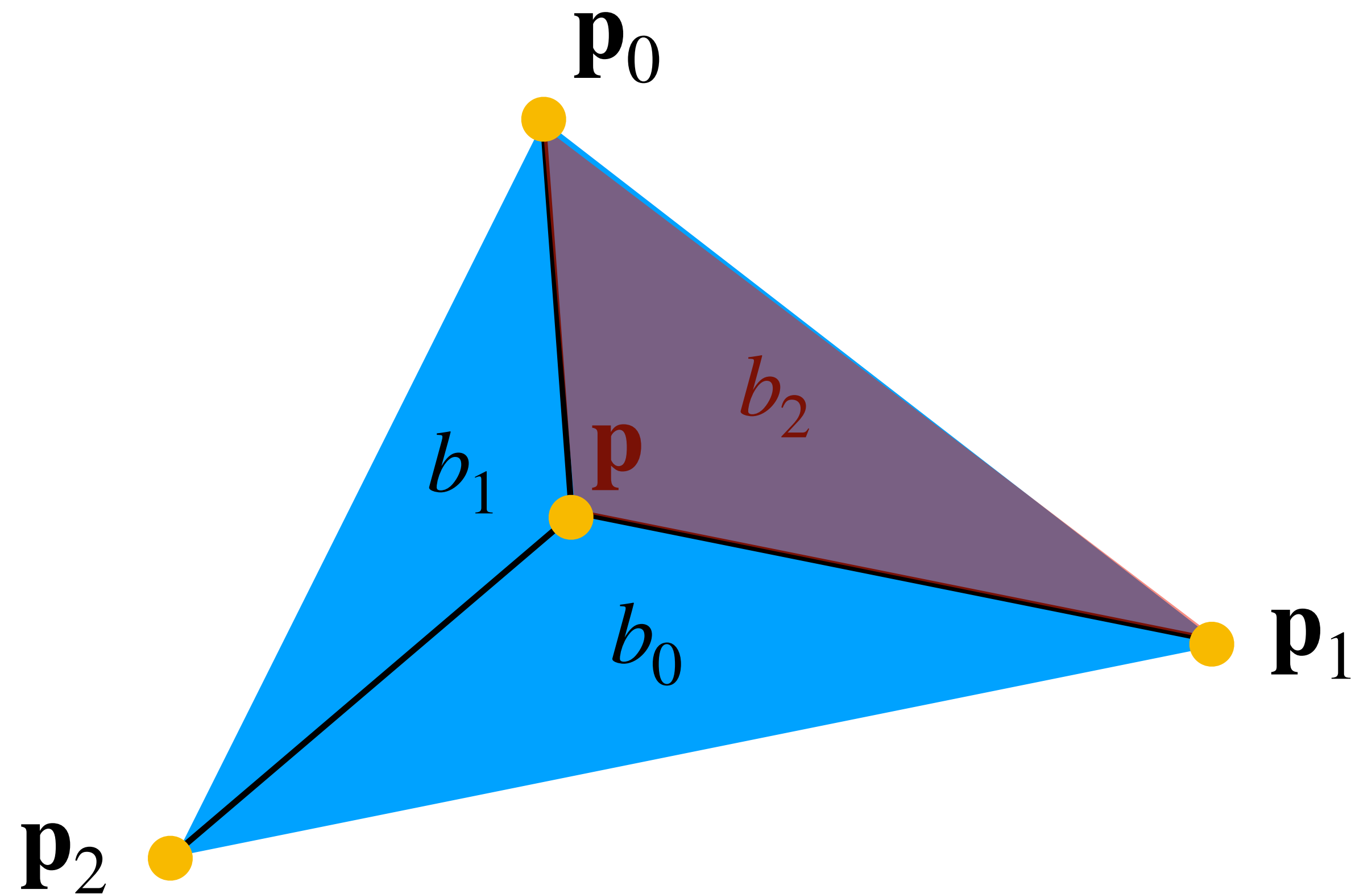
$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

$$b_0, b_1, b_2 \geq 0$$

$$b_0 + b_1 + b_2 = 1$$

$$b_0 = \frac{\text{area}(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)} \quad b_1 = \frac{\text{area}(\mathbf{p}_0, \mathbf{p}, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$

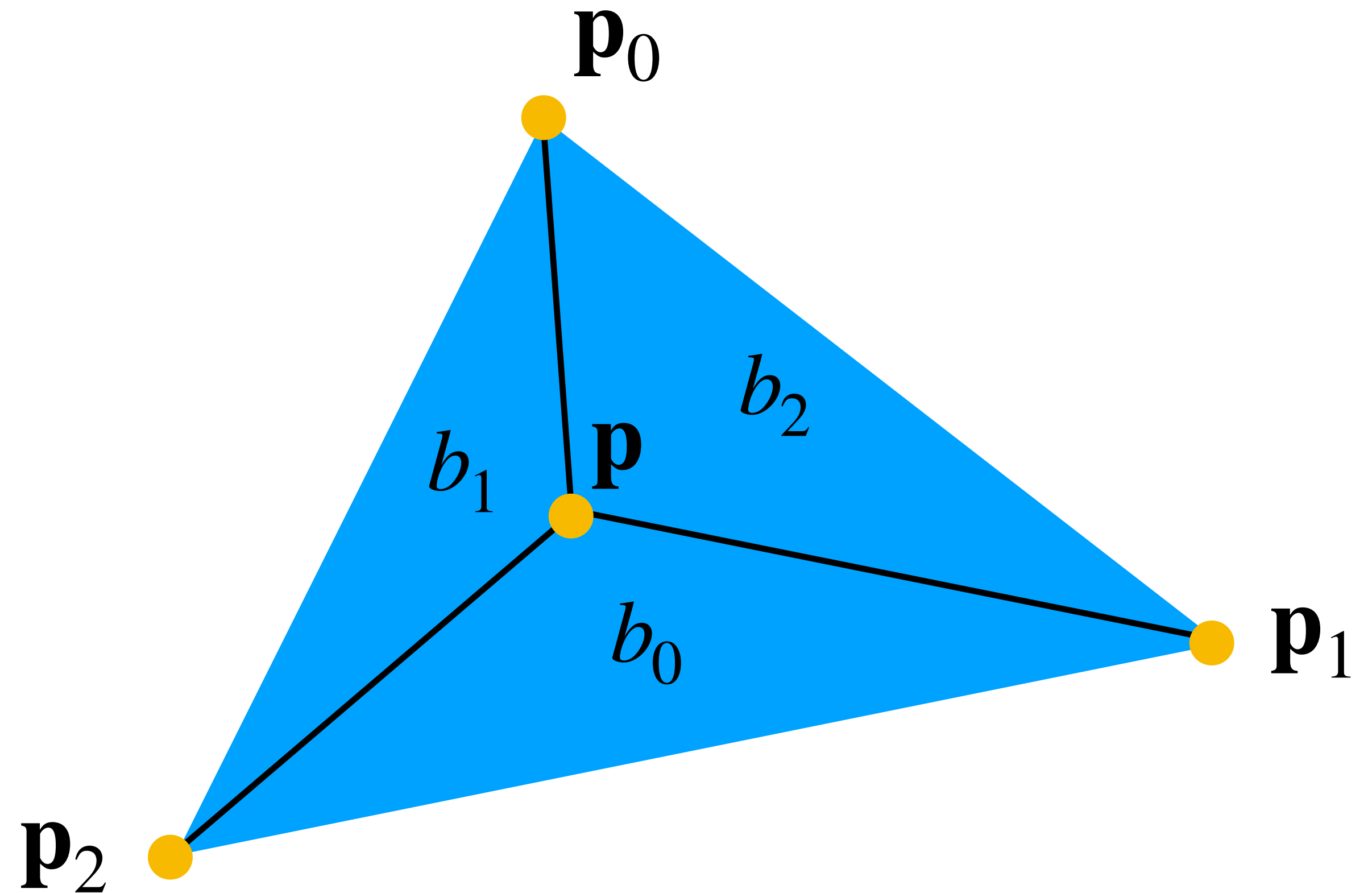
$$b_2 = \frac{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p})}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$



We need barycentric coordinates

proof for 2D: given \mathbf{p} , \mathbf{p}_0 , \mathbf{p}_1 , \mathbf{p}_2 , we can solve for b_0 , b_1 , b_2

$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

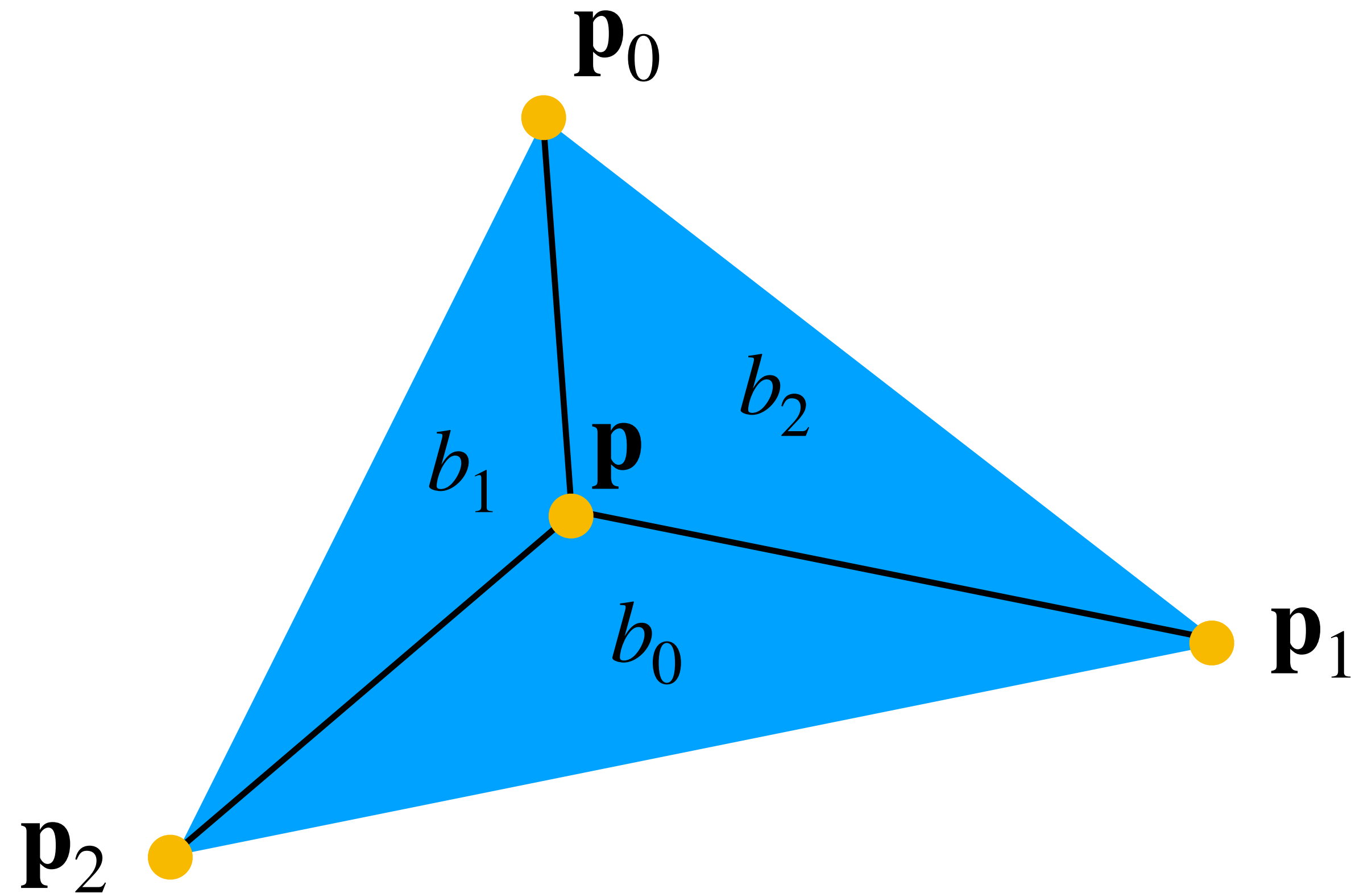


We need barycentric coordinates

proof for 2D: given \mathbf{p} , \mathbf{p}_0 , \mathbf{p}_1 , \mathbf{p}_2 , we can solve for b_0 , b_1 , b_2

$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

$$\begin{bmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \cdot x \\ \mathbf{p} \cdot y \\ 1 \end{bmatrix}$$

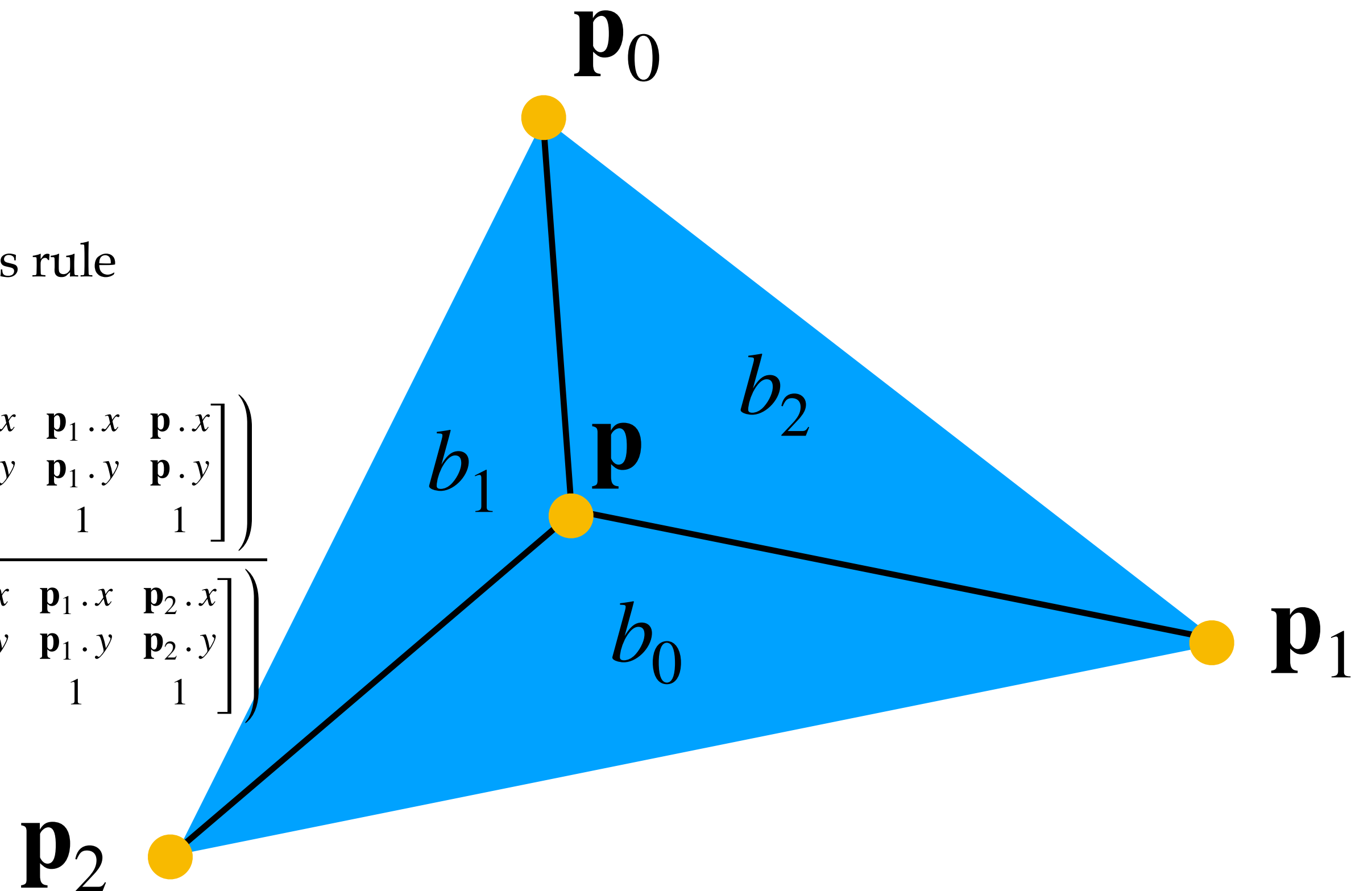


We need barycentric coordinates

proof for 2D: given \mathbf{p} , \mathbf{p}_0 , \mathbf{p}_1 , \mathbf{p}_2 , we can solve for b_0 , b_1 , b_2

$$\begin{bmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \cdot x \\ \mathbf{p} \cdot y \\ 1 \end{bmatrix} \quad \text{Cramer's rule}$$

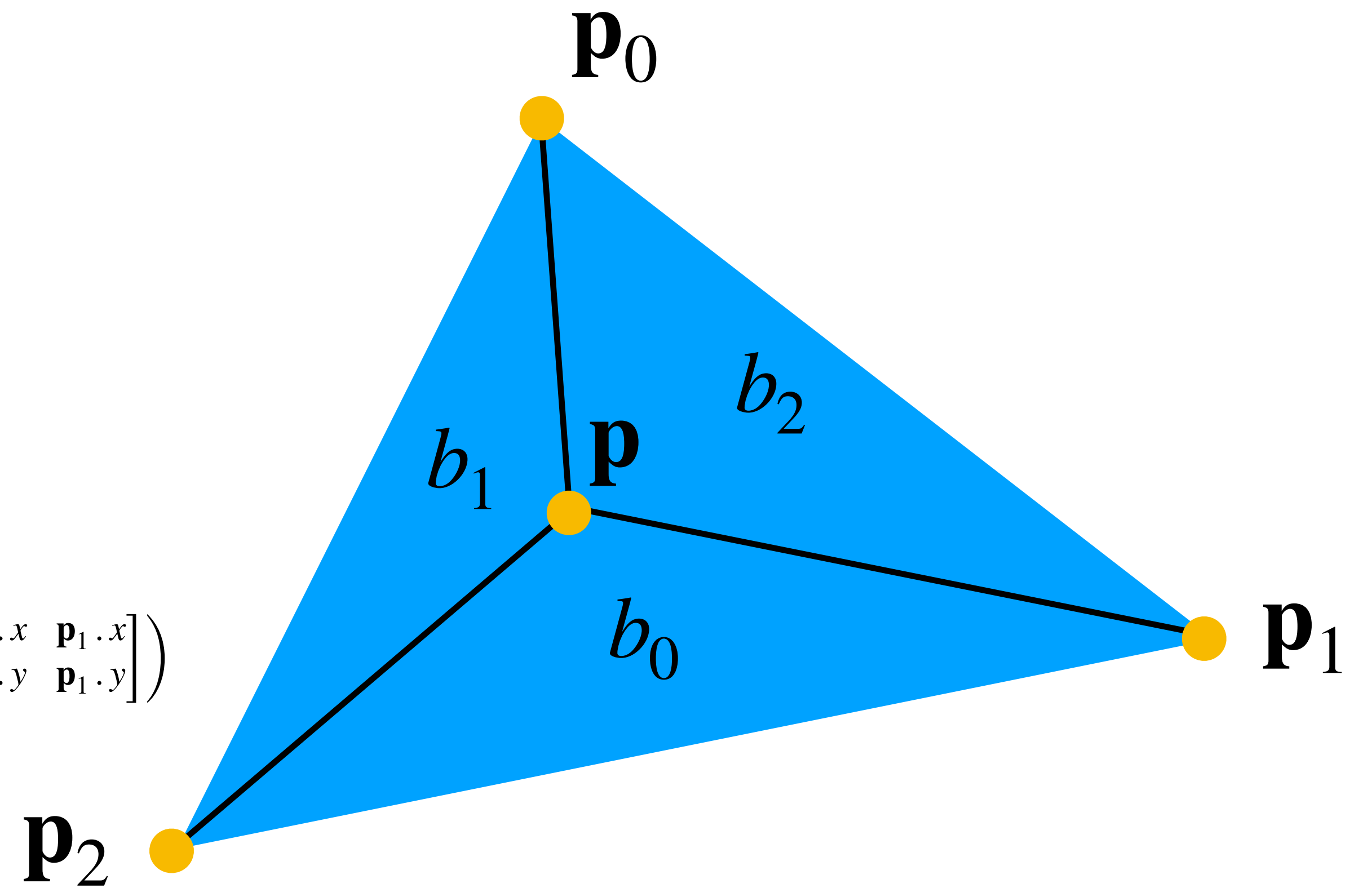
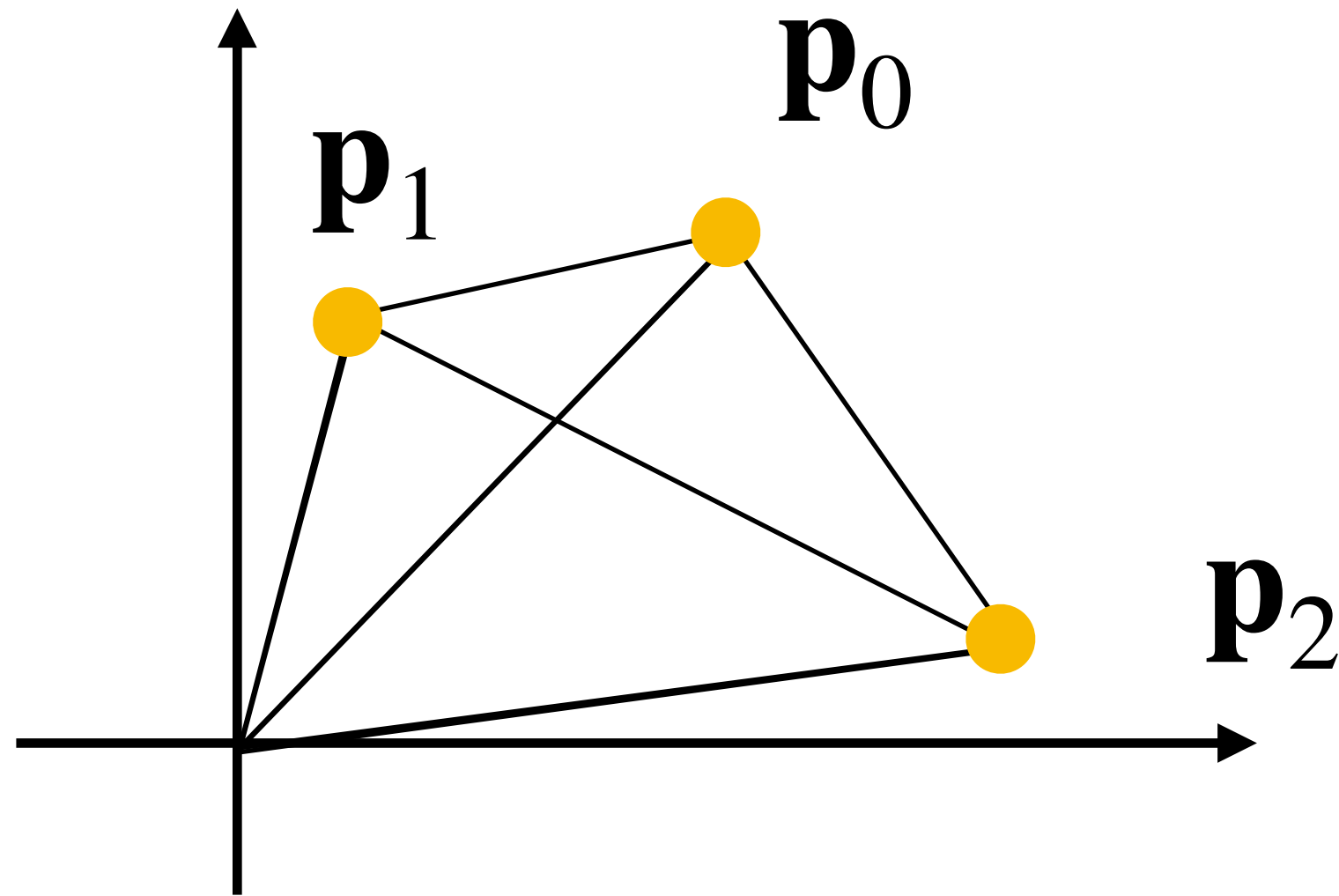
$$b_0 = \frac{\det \begin{pmatrix} \mathbf{p} \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p} \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix}}{\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix}} \quad b_1 = \frac{\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p} \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p} \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix}}{\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix}} \quad b_2 = \frac{\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p} \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p} \cdot y \\ 1 & 1 & 1 \end{pmatrix}}{\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix}}$$



check out the 3Blue1Brown video for Cramer's rule
<https://www.youtube.com/watch?v=jBsC34PxzoM>

We need barycentric coordinates

proof for 2D: given \mathbf{p} , \mathbf{p}_0 , \mathbf{p}_1 , \mathbf{p}_2 , we can solve for b_0 , b_1 , b_2



$$\det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \\ 1 & 1 & 1 \end{pmatrix} = \det \begin{pmatrix} \mathbf{p}_1 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_1 \cdot y & \mathbf{p}_2 \cdot y \end{pmatrix} - \det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_2 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_2 \cdot y \end{pmatrix} + \det \begin{pmatrix} \mathbf{p}_0 \cdot x & \mathbf{p}_1 \cdot x \\ \mathbf{p}_0 \cdot y & \mathbf{p}_1 \cdot y \end{pmatrix}$$

2D determinant = area of the parallelogram

check out the 3Blue1Brown video for determinant
<https://www.youtube.com/watch?v=Ip3X9LOh2dk>

Barycentric coordinates can be used for inside-outside tests as well!

probably faster than the ray casting we used

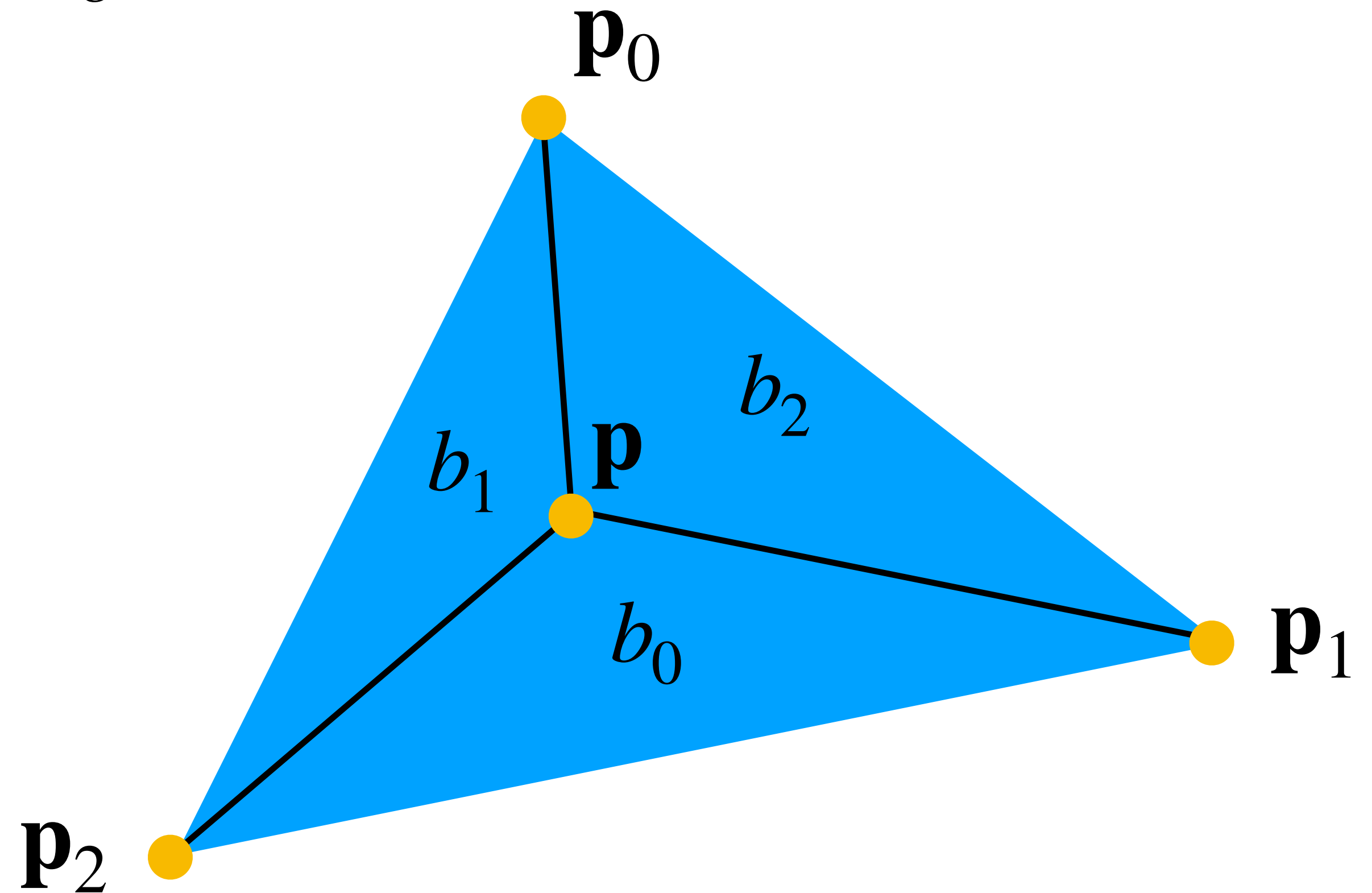
$$\mathbf{p} = b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$$

\mathbf{p} is inside the triangle if and only if

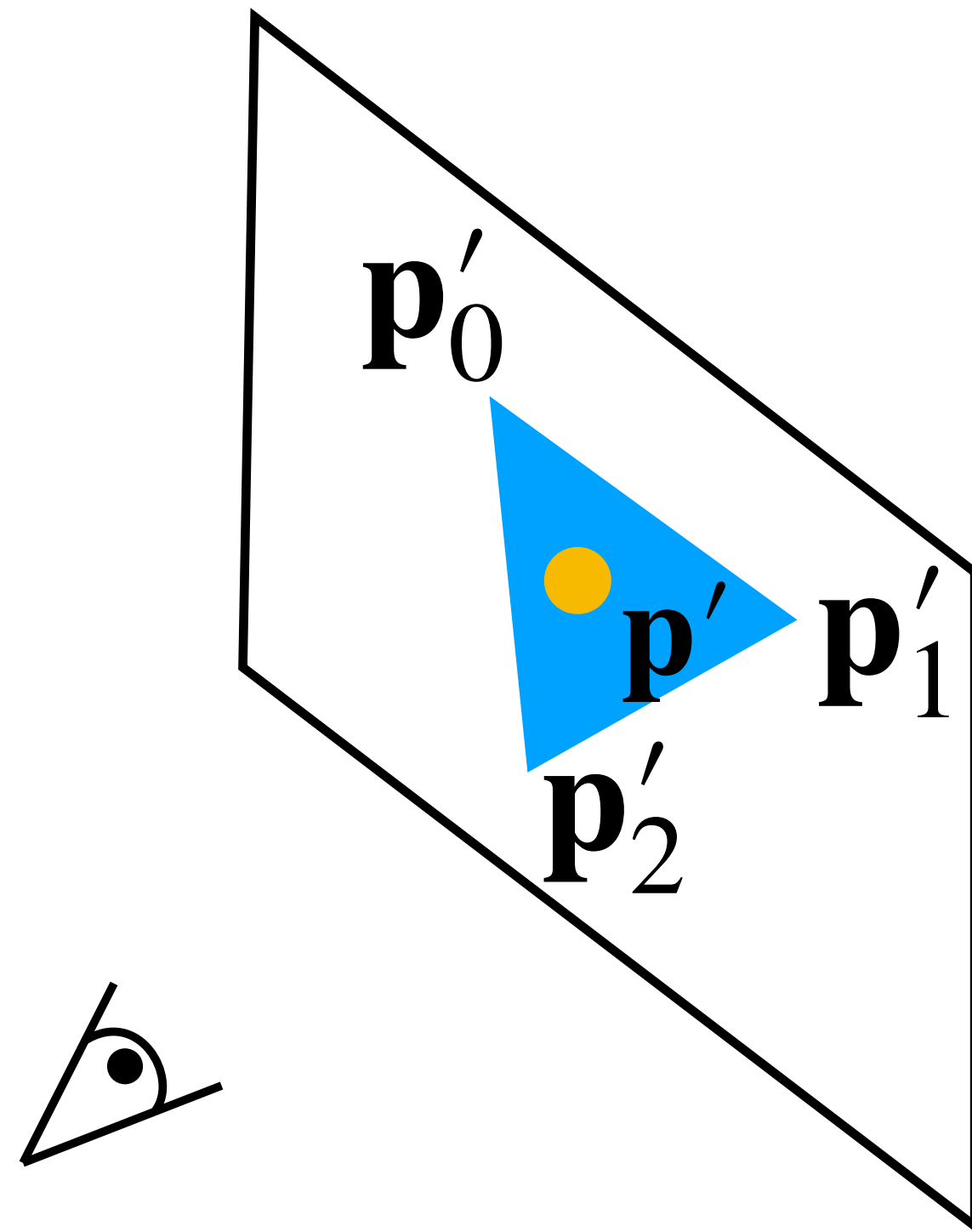
$$0 \leq b_0 \leq 1$$

$$0 \leq b_1 \leq 1$$

$$0 \leq b_2 \leq 1$$



Given a pixel sample and a screen triangle, we can figure out its barycentric coordinates



$$b'_0 = \frac{\text{area}(\mathbf{p}', \mathbf{p}'_1, \mathbf{p}'_2)}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

$$b'_1 = \frac{\text{area}(\mathbf{p}'_0, \mathbf{p}', \mathbf{p}'_2)}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

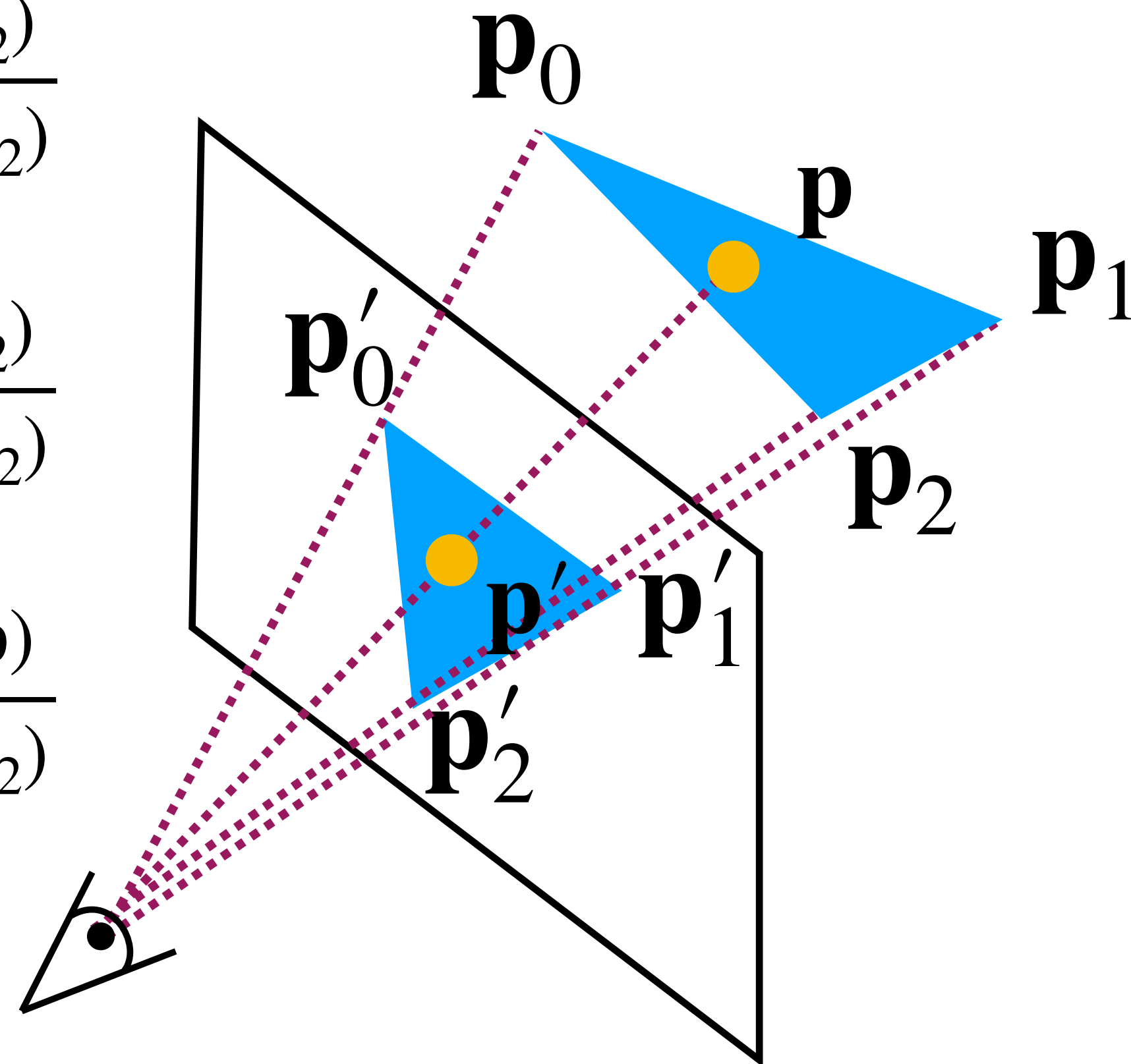
$$b'_2 = \frac{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}')}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

Our goal is to figure out the barycentric coordinates of the 3D triangle

$$b_0 = \frac{\text{area}(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$

$$b_1 = \frac{\text{area}(\mathbf{p}_0, \mathbf{p}, \mathbf{p}_2)}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$

$$b_2 = \frac{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p})}{\text{area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$



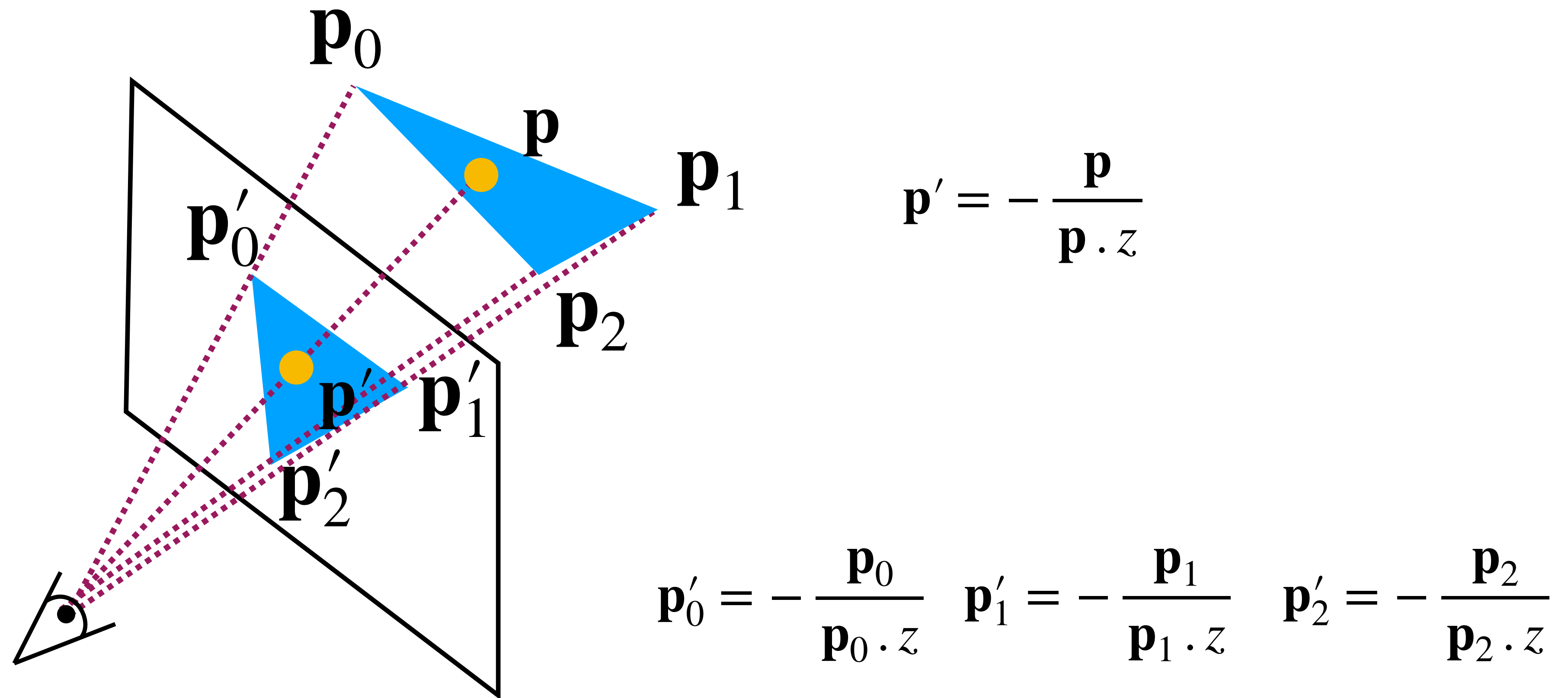
$$b'_0 = \frac{\text{area}(\mathbf{p}', \mathbf{p}'_1, \mathbf{p}'_2)}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

$$b'_1 = \frac{\text{area}(\mathbf{p}'_0, \mathbf{p}', \mathbf{p}'_2)}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

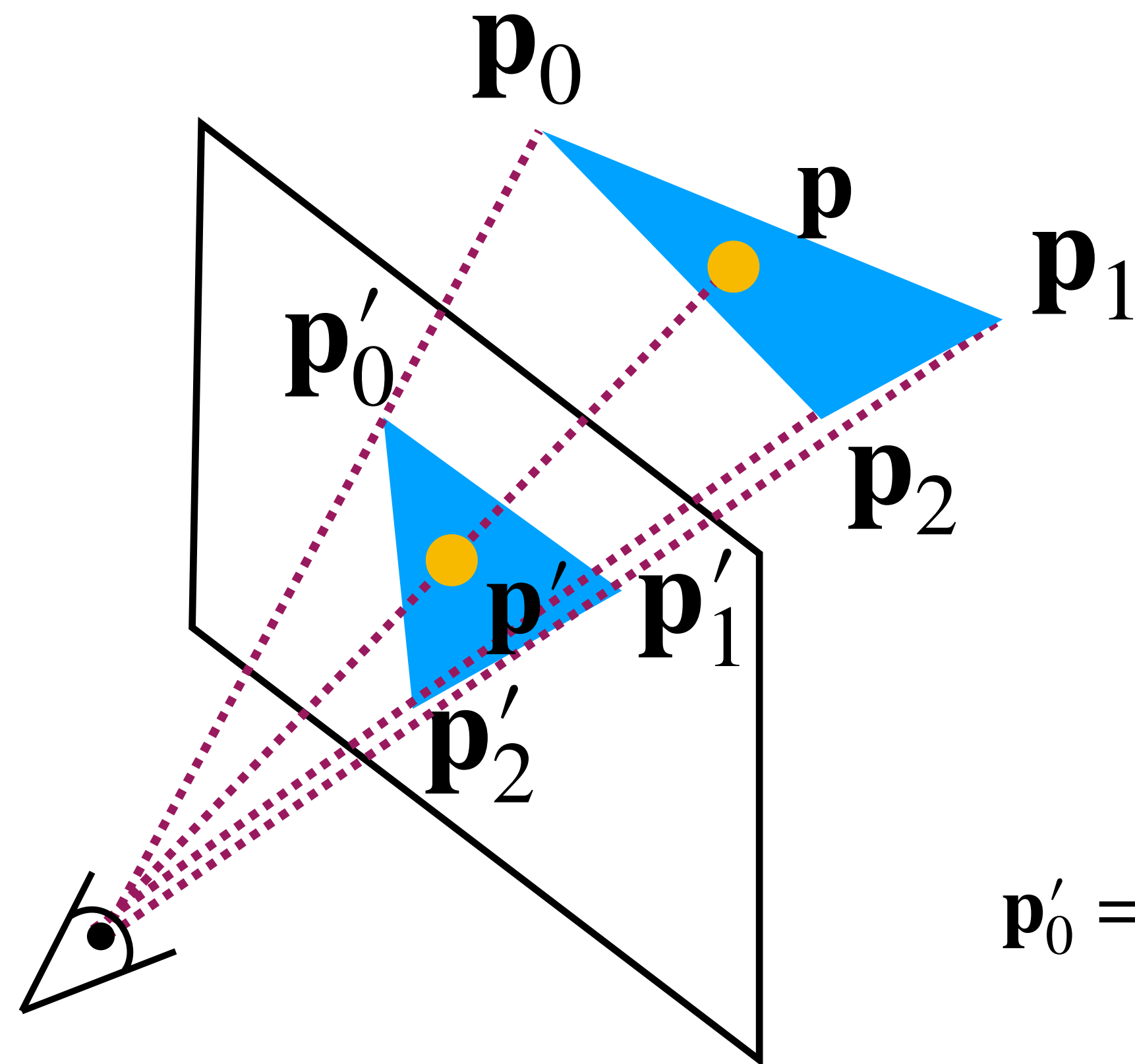
$$b'_2 = \frac{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}')}{\text{area}(\mathbf{p}'_0, \mathbf{p}'_1, \mathbf{p}'_2)}$$

are they the same?

We solve for the 3D barycentric coordinates using the projection relation



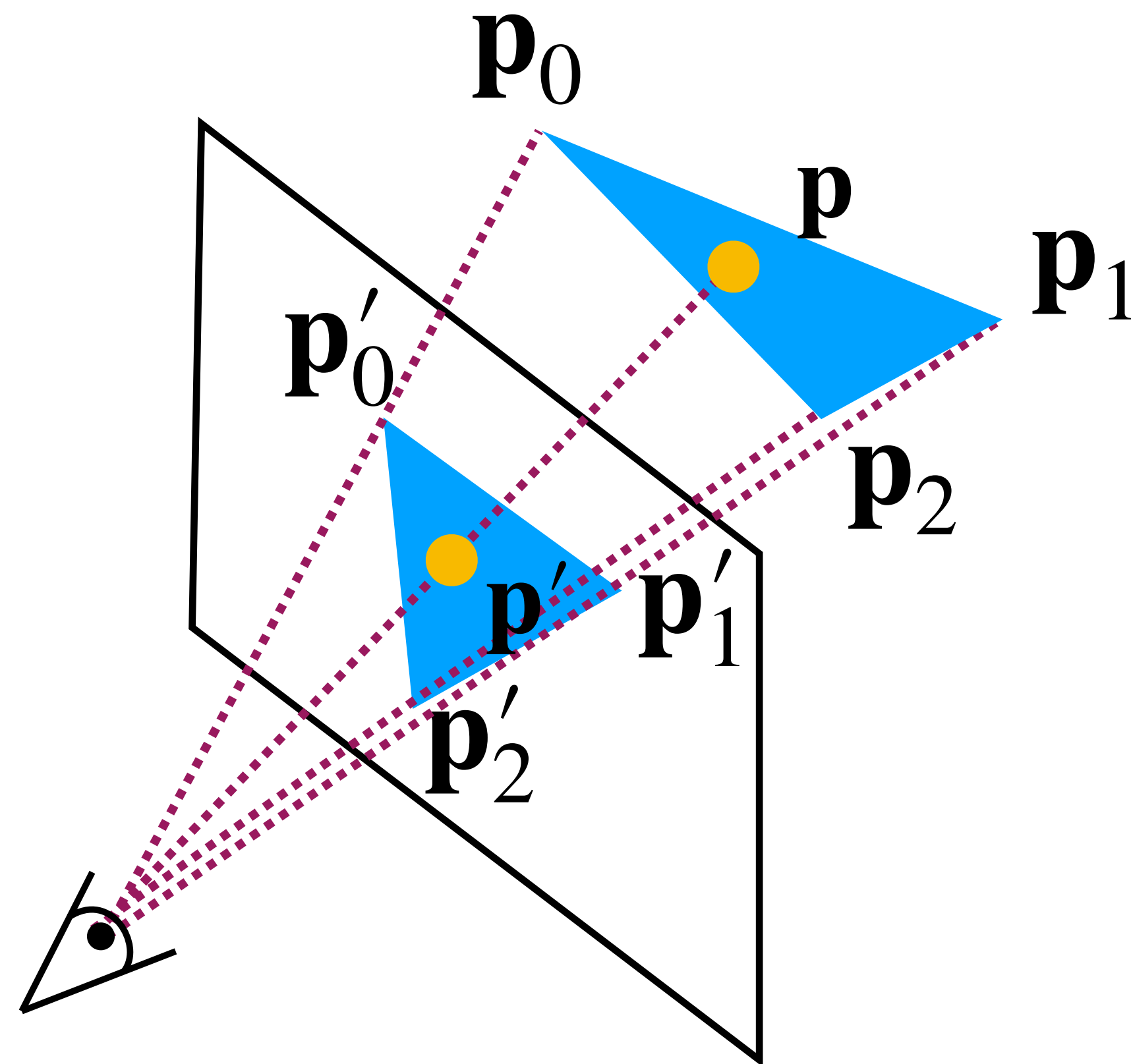
We solve for the 3D barycentric coordinates using the projection relation



$$\mathbf{p}' = - \frac{b_0 \mathbf{p}_0 + b_1 \mathbf{p}_1 + b_2 \mathbf{p}_2}{b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}}$$

$$\mathbf{p}'_0 = - \frac{\mathbf{p}_0}{\mathbf{p}_0 \cdot \mathbf{z}} \quad \mathbf{p}'_1 = - \frac{\mathbf{p}_1}{\mathbf{p}_1 \cdot \mathbf{z}} \quad \mathbf{p}'_2 = - \frac{\mathbf{p}_2}{\mathbf{p}_2 \cdot \mathbf{z}}$$

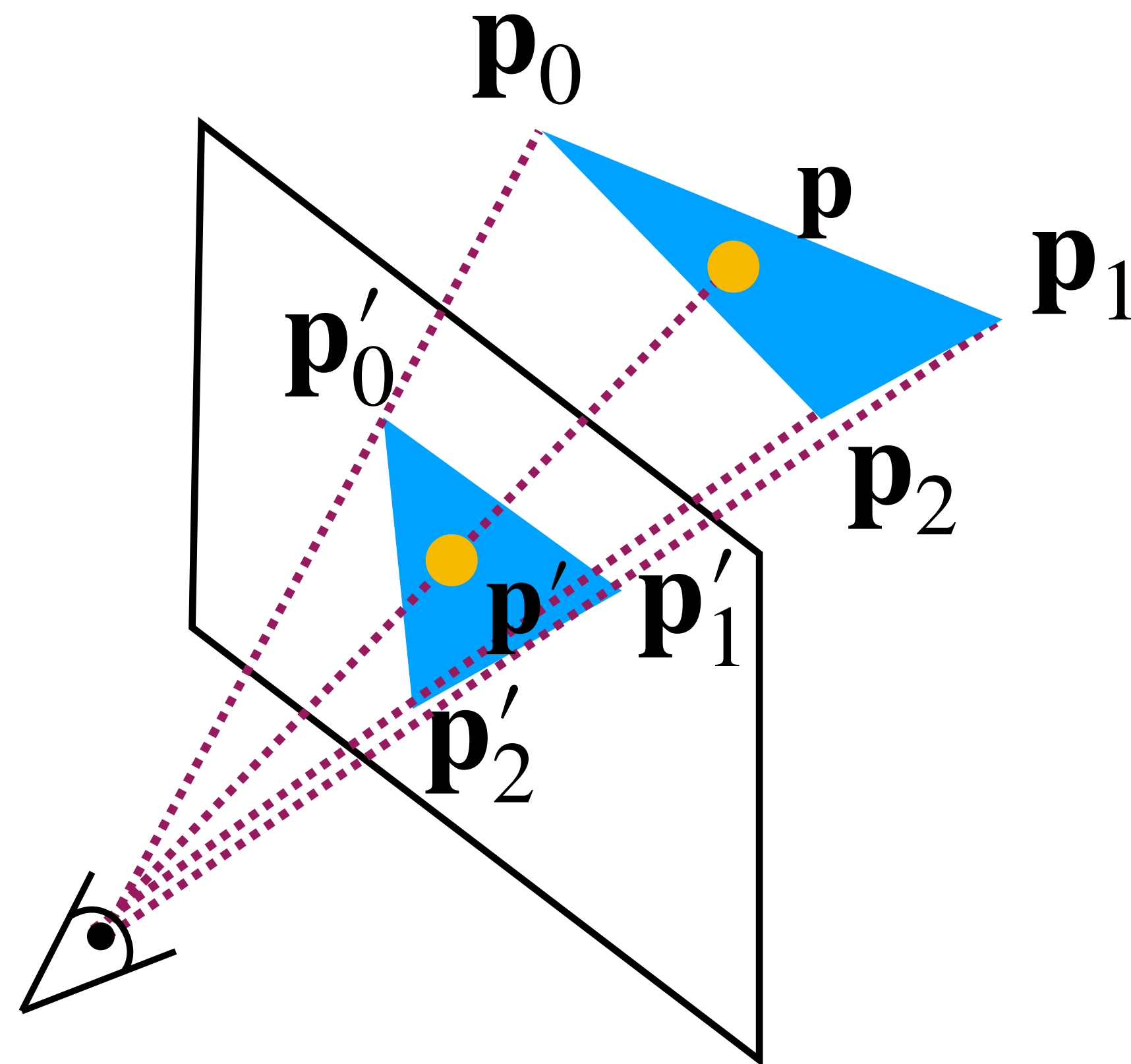
We solve for the 3D barycentric coordinates using the projection relation



$$\mathbf{p}' = \frac{b_0 (\mathbf{p}_0 \cdot \mathbf{z}) \mathbf{p}'_0 + b_1 (\mathbf{p}_1 \cdot \mathbf{z}) \mathbf{p}'_1 + b_2 (\mathbf{p}_2 \cdot \mathbf{z}) \mathbf{p}'_2}{b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}}$$

$$\mathbf{p}' = b'_0 \mathbf{p}'_0 + b'_1 \mathbf{p}'_1 + b'_2 \mathbf{p}'_2$$

We solve for the 3D barycentric coordinates using the projection relation

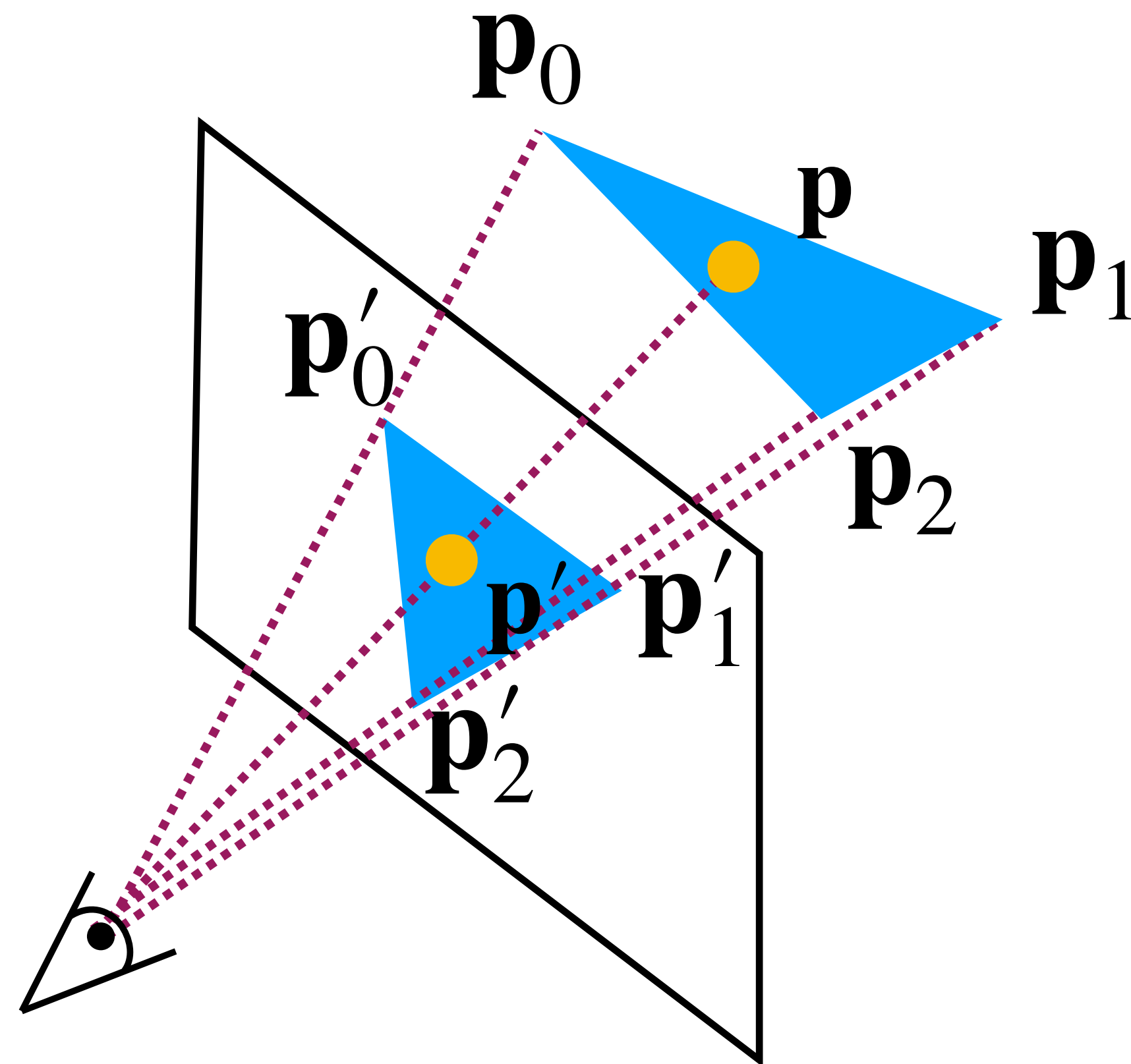


$$b'_0 = \frac{b_0 \mathbf{p}_0 \cdot \mathbf{z}}{b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}}$$

$$b'_1 = \frac{b_1 \mathbf{p}_1 \cdot \mathbf{z}}{b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}}$$

$$b'_2 = \frac{b_2 \mathbf{p}_2 \cdot \mathbf{z}}{b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}}$$

We solve for the 3D barycentric coordinates using the projection relation



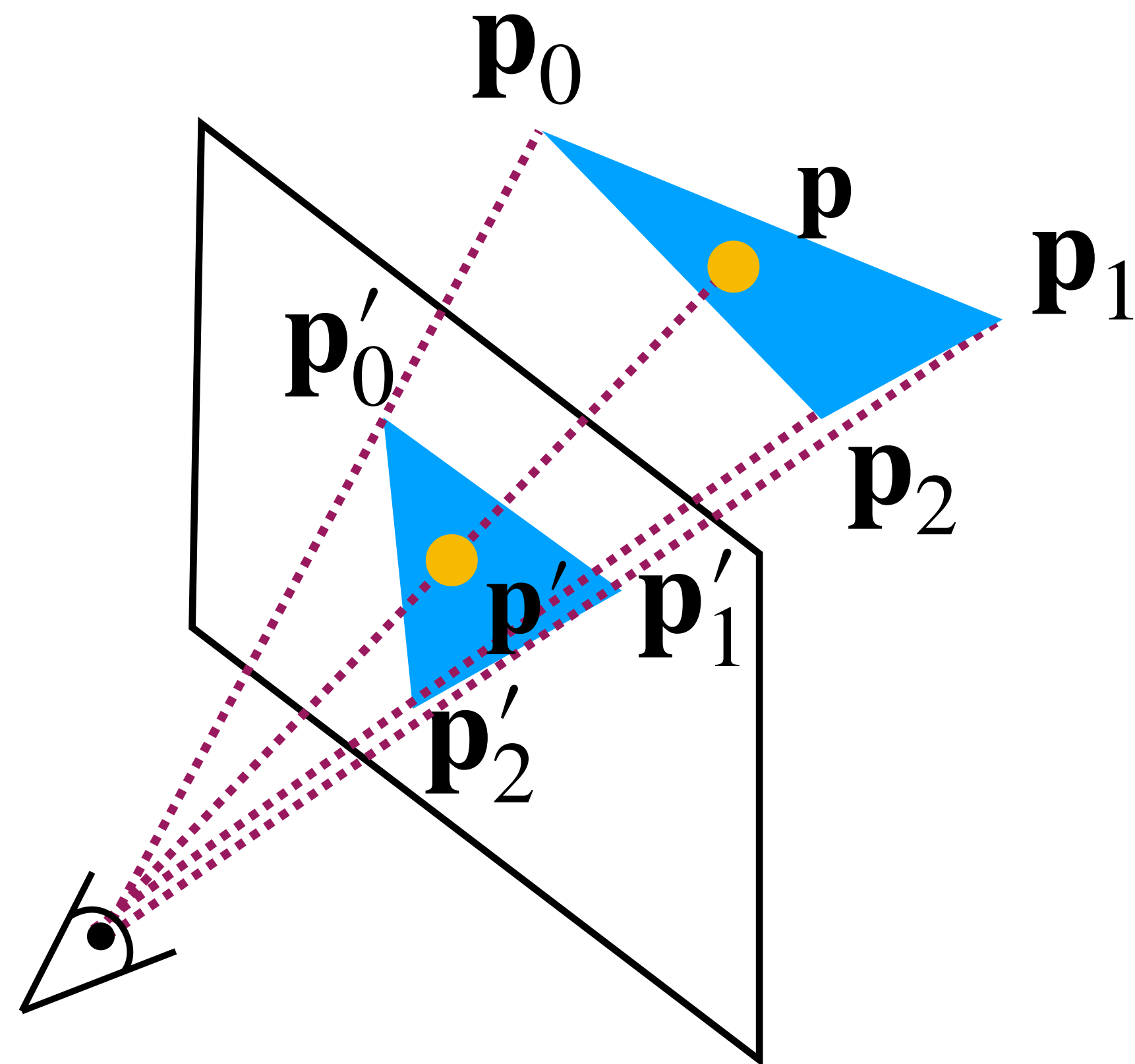
$$b'_0 = \frac{b_0 \mathbf{p}_0 \cdot \mathbf{z}}{Z}$$

$$b'_1 = \frac{b_1 \mathbf{p}_1 \cdot \mathbf{z}}{Z}$$

$$b'_2 = \frac{b_2 \mathbf{p}_2 \cdot \mathbf{z}}{Z}$$

$$Z = b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}$$

We solve for the 3D barycentric coordinates using the projection relation



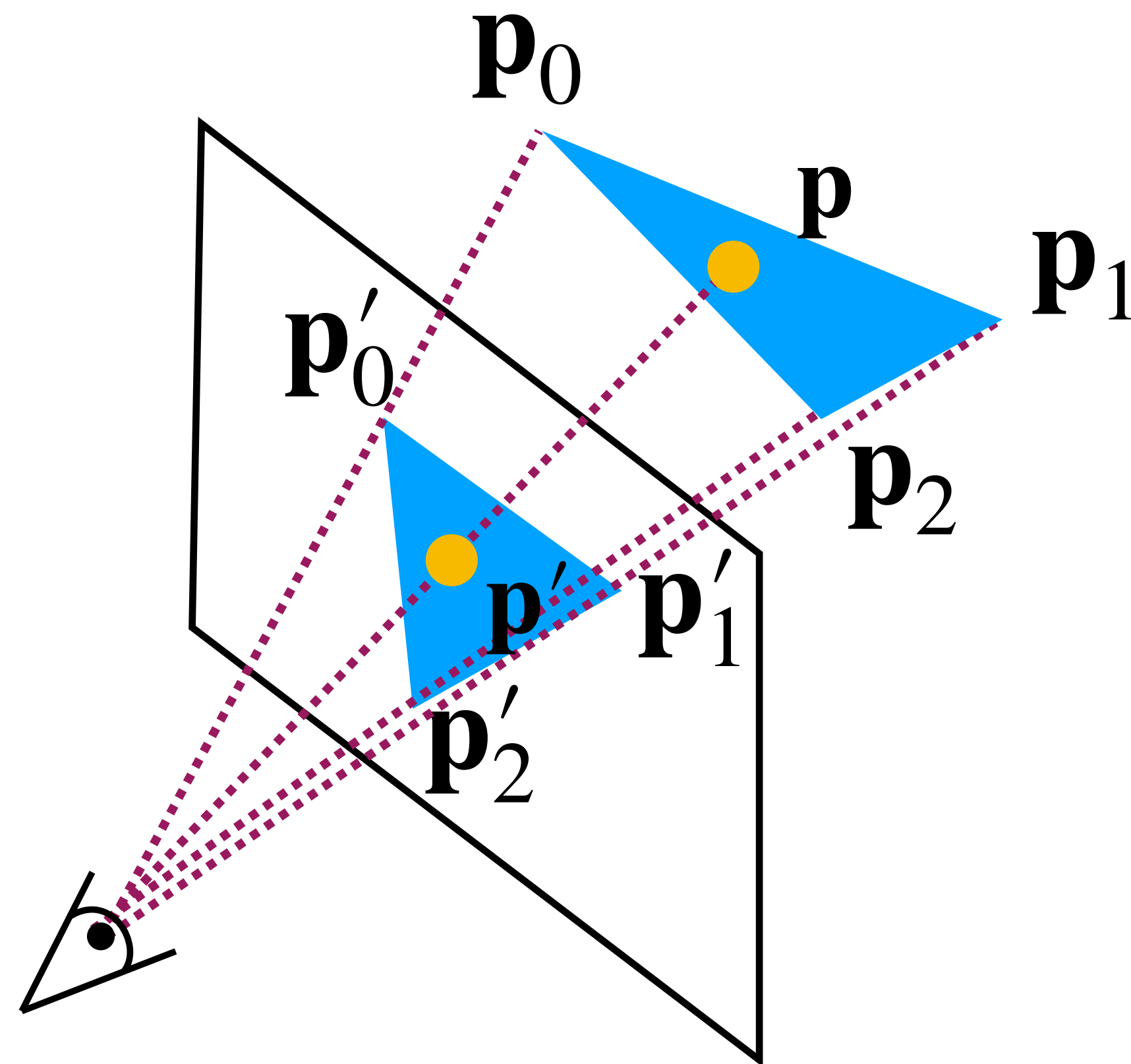
$$b_0 = \frac{Zb'_0}{\mathbf{p}_0 \cdot \mathbf{z}}$$

$$b_1 = \frac{Zb'_1}{\mathbf{p}_1 \cdot \mathbf{z}}$$

$$b_2 = \frac{Zb'_2}{\mathbf{p}_2 \cdot \mathbf{z}}$$

$$Z = b_0\mathbf{p}_0 \cdot \mathbf{z} + b_1\mathbf{p}_1 \cdot \mathbf{z} + b_2\mathbf{p}_2 \cdot \mathbf{z}$$

We solve for the 3D barycentric coordinates using the projection relation

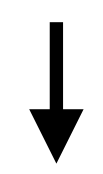


$$b_0 = \frac{Zb'_0}{\mathbf{p}_0 \cdot \mathbf{z}}$$

$$b_1 = \frac{Zb'_1}{\mathbf{p}_1 \cdot \mathbf{z}}$$

$$b_2 = \frac{Zb'_2}{\mathbf{p}_2 \cdot \mathbf{z}}$$

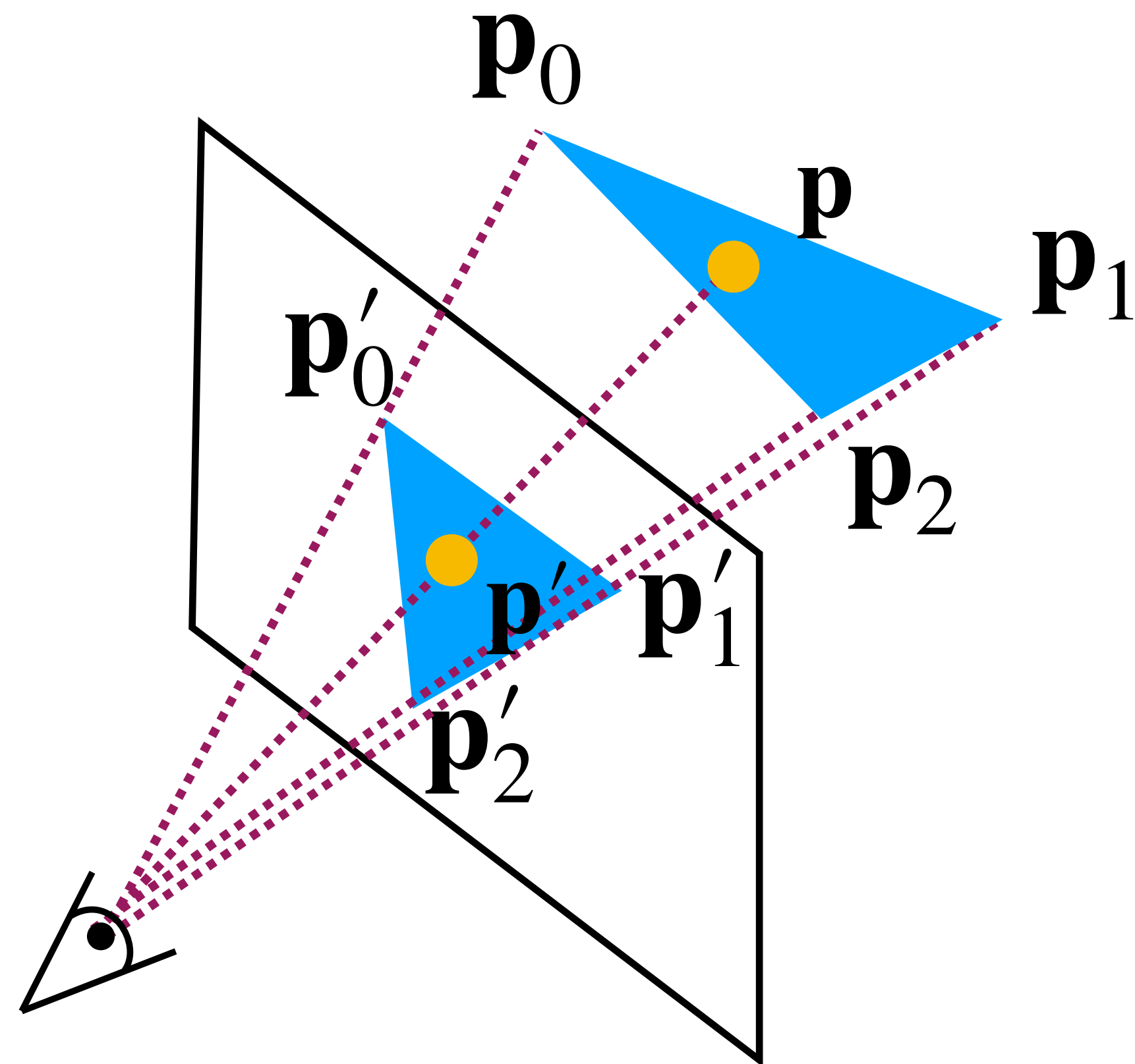
$$b_0 + b_1 + b_2 = 1$$



$$Z = \frac{1}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$

$$Z = b_0\mathbf{p}_0 \cdot \mathbf{z} + b_1\mathbf{p}_1 \cdot \mathbf{z} + b_2\mathbf{p}_2 \cdot \mathbf{z}$$

We solve for the 3D barycentric coordinates using the projection relation

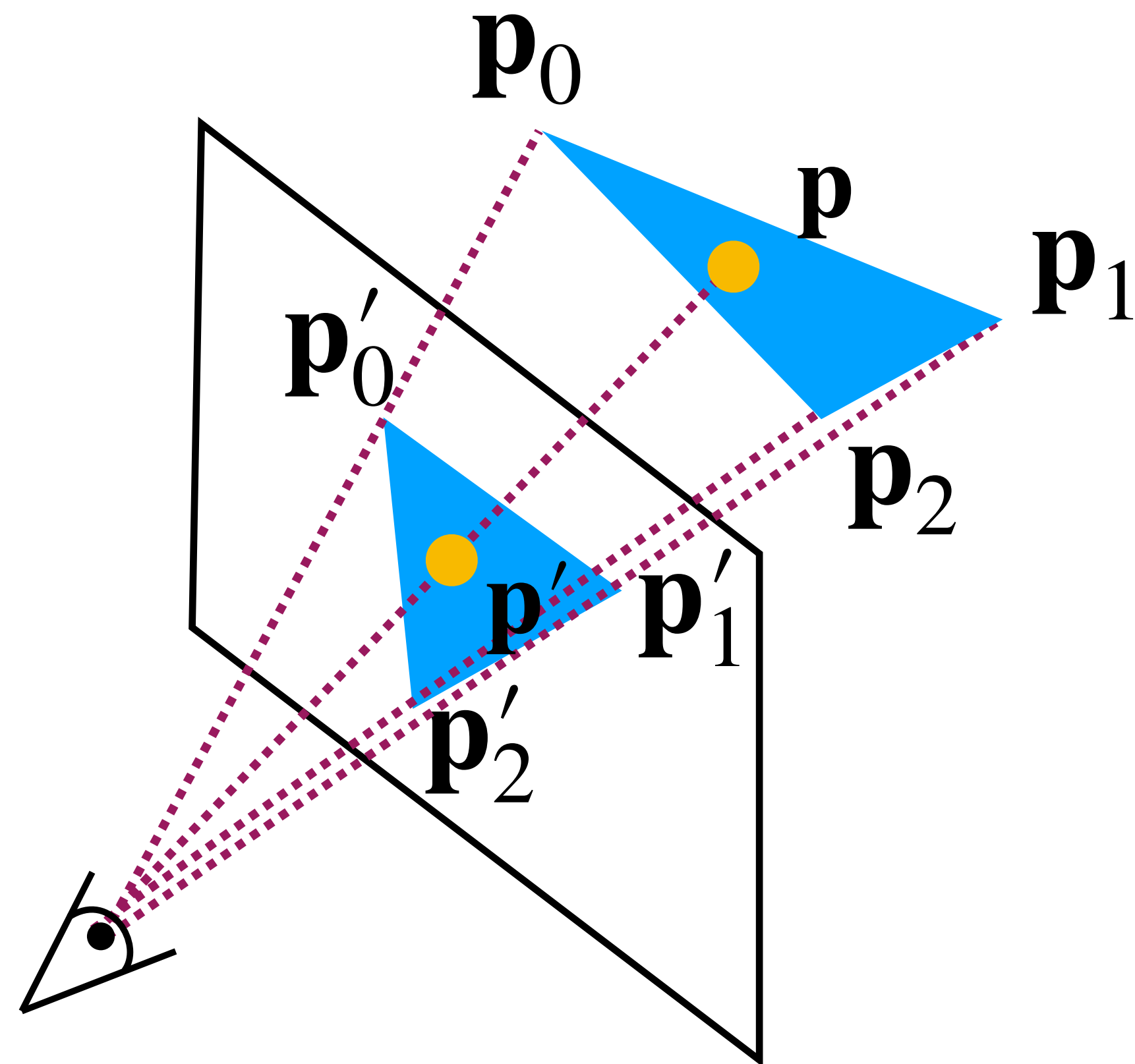


$$b_0 = \frac{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$

$$b_1 = \frac{\frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$

$$b_2 = \frac{\frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$

The 3D barycentric coordinates are the 2D ones weighted by inverse depth

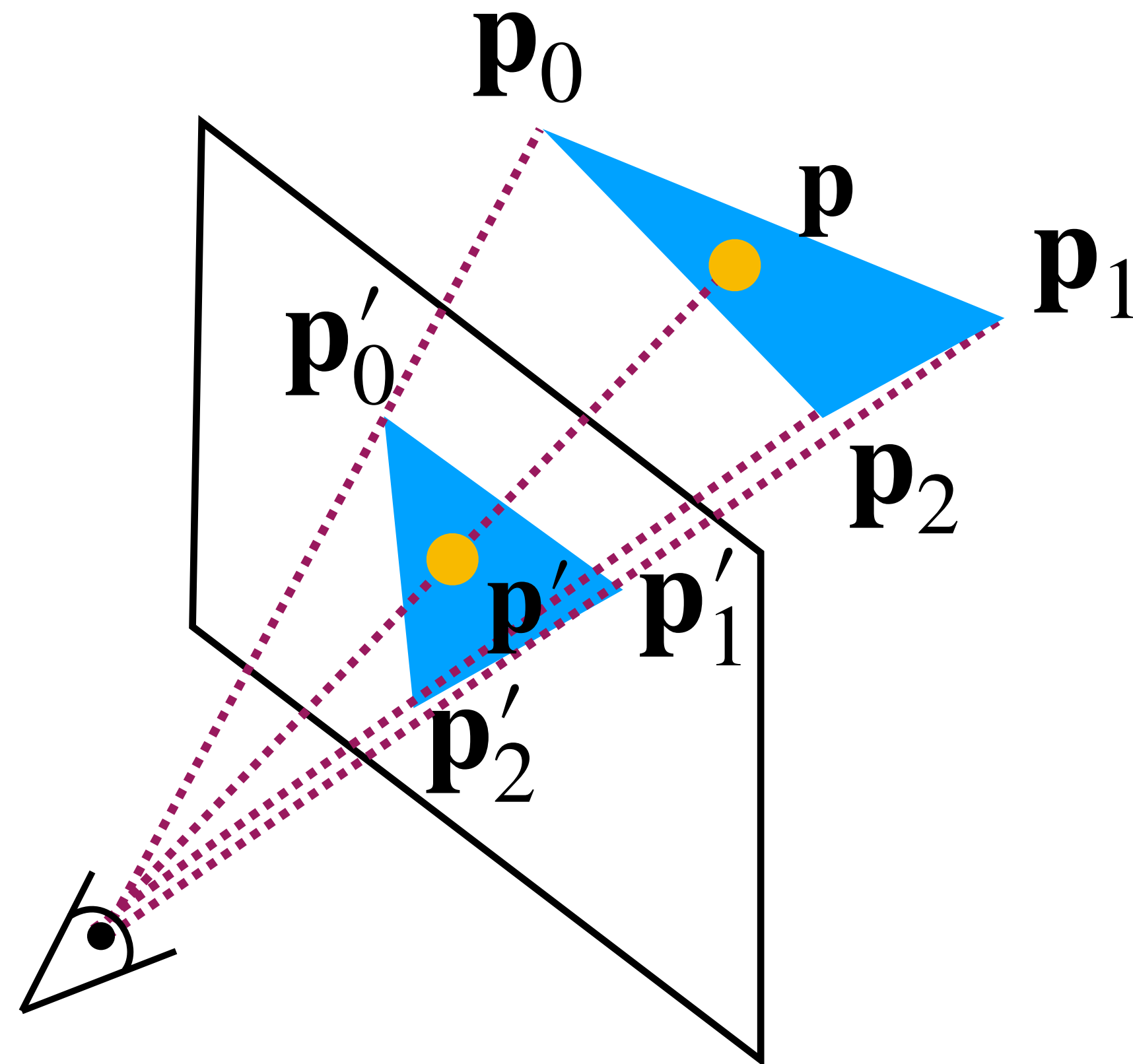


$$b_0 = \frac{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$
$$b_1 = \frac{\frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$
$$b_2 = \frac{\frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}{\frac{b'_0}{\mathbf{p}_0 \cdot \mathbf{z}} + \frac{b'_1}{\mathbf{p}_1 \cdot \mathbf{z}} + \frac{b'_2}{\mathbf{p}_2 \cdot \mathbf{z}}}$$

We can now interpolate any values on the 3D triangle

this is called the “perspective-corrected interpolation”

e.g., color

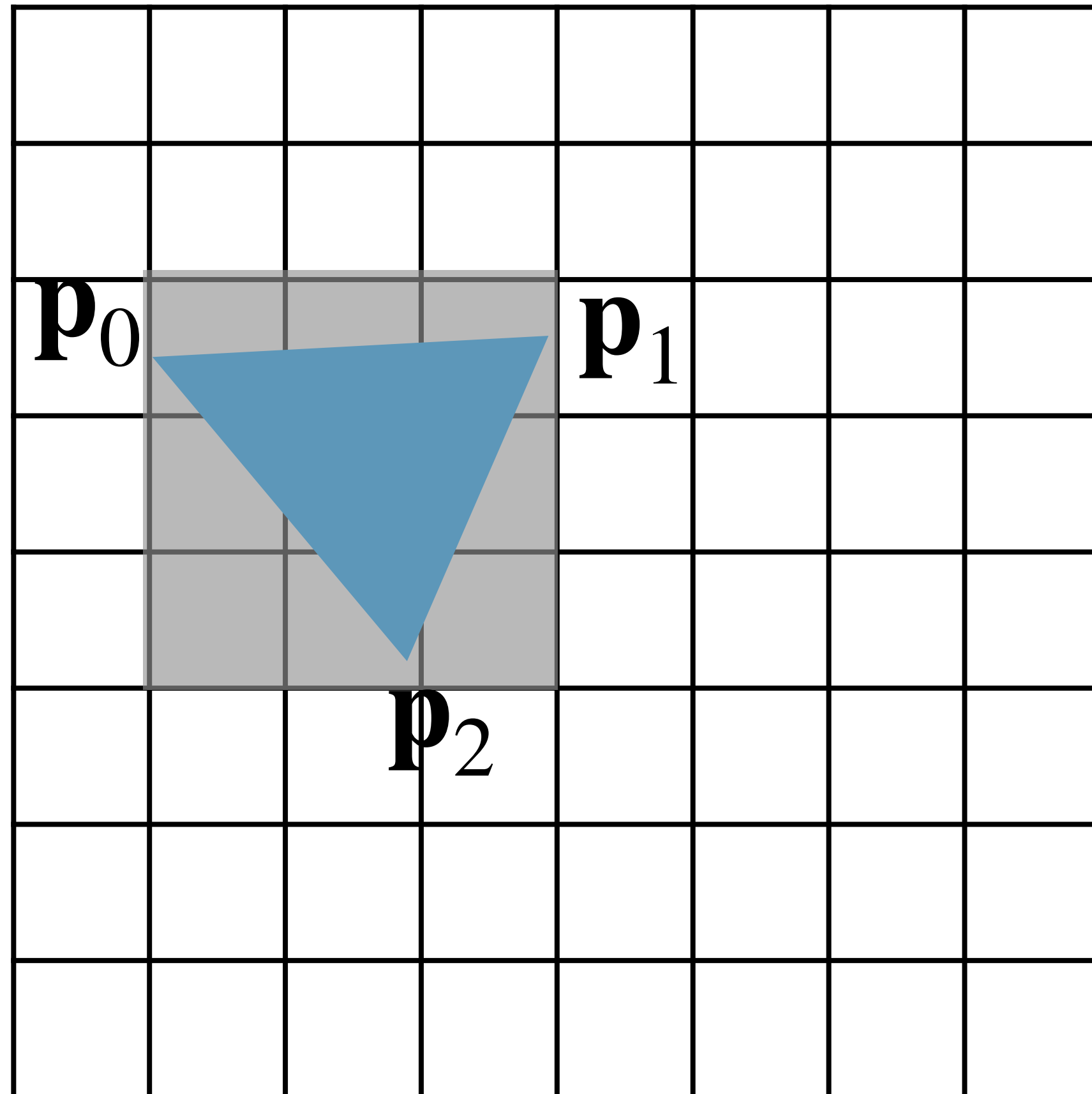


$$Z = b_0 \mathbf{p}_0 \cdot \mathbf{z} + b_1 \mathbf{p}_1 \cdot \mathbf{z} + b_2 \mathbf{p}_2 \cdot \mathbf{z}$$

$$v = b_0 v_0 + b_1 v_1 + b_2 v_2$$

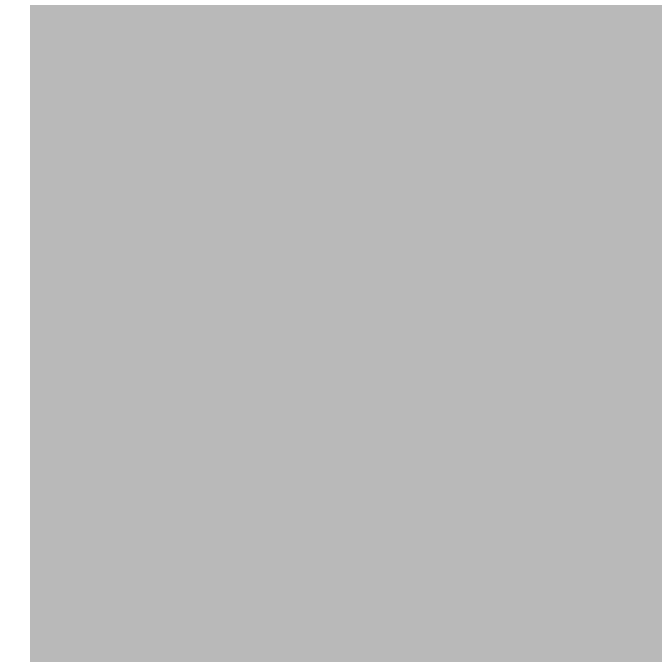
Speedup trick:

For each triangle, only tests pixels within the **bounding box**



$(\mathbf{p}_{\min_x}, \mathbf{p}_{\min_y})$

$(\mathbf{p}_{\max_x}, \mathbf{p}_{\min_y})$



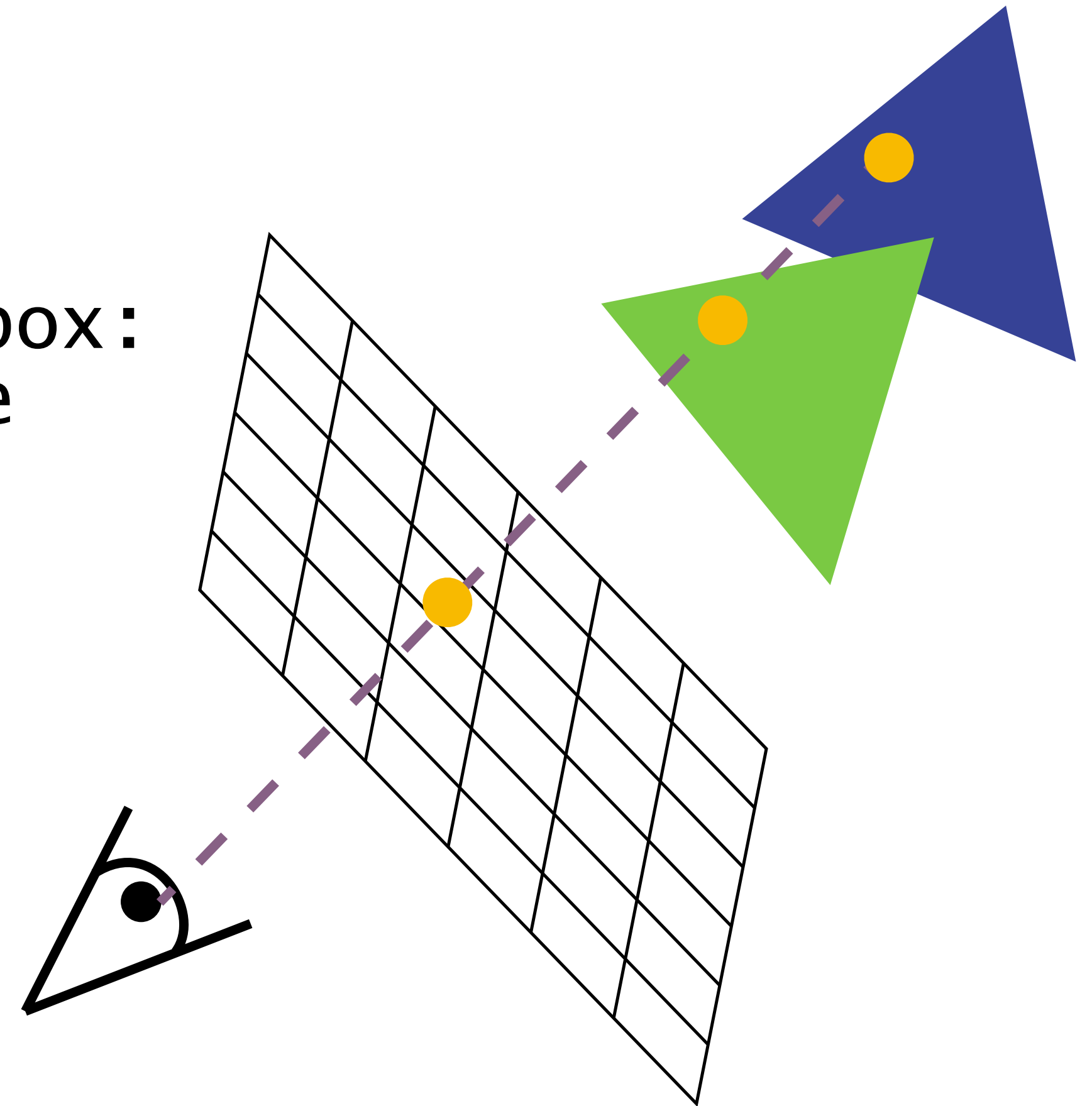
$(\mathbf{p}_{\min_x}, \mathbf{p}_{\max_y})$

$(\mathbf{p}_{\max_x}, \mathbf{p}_{\max_y})$

Q: this will be more helpful when there are many **small** or **large** triangles?

The rasterization algorithm

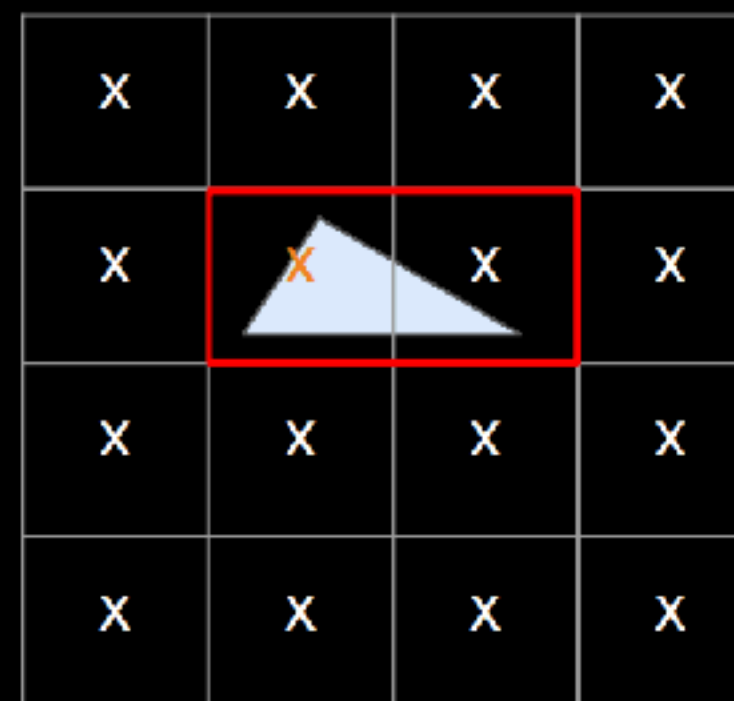
```
z_buffer = inf(w, h)
img = zeros(w, h)
for each clipped 3D triangle:
  get the 2D triangle by dividing -z
  compute image space bounding box
  for each pixel (x, y) in the bounding box:
    if the pixel center hits the triangle
      interpolate z
      if (|z| < z_buffer[x, y]):
        z_buffer[x, y] = |z|
        img[x, y] = ...
```



This is basically the rasterization algorithm used in Unreal Engine 5!

Micropoly software rasterizer

- 128 triangle clusters => threadgroup size 128
- 1 thread per vertex
 - Transform position
 - Store in groupshared
 - If more than 128 verts loop (max 2)
- 1 thread per triangle
 - Fetch indexes
 - Fetch transformed positions
 - Calculate edge equations and depth gradient
 - Calculate screen bounding rect
 - For all pixels in rect
 - If inside all edges then write pixel

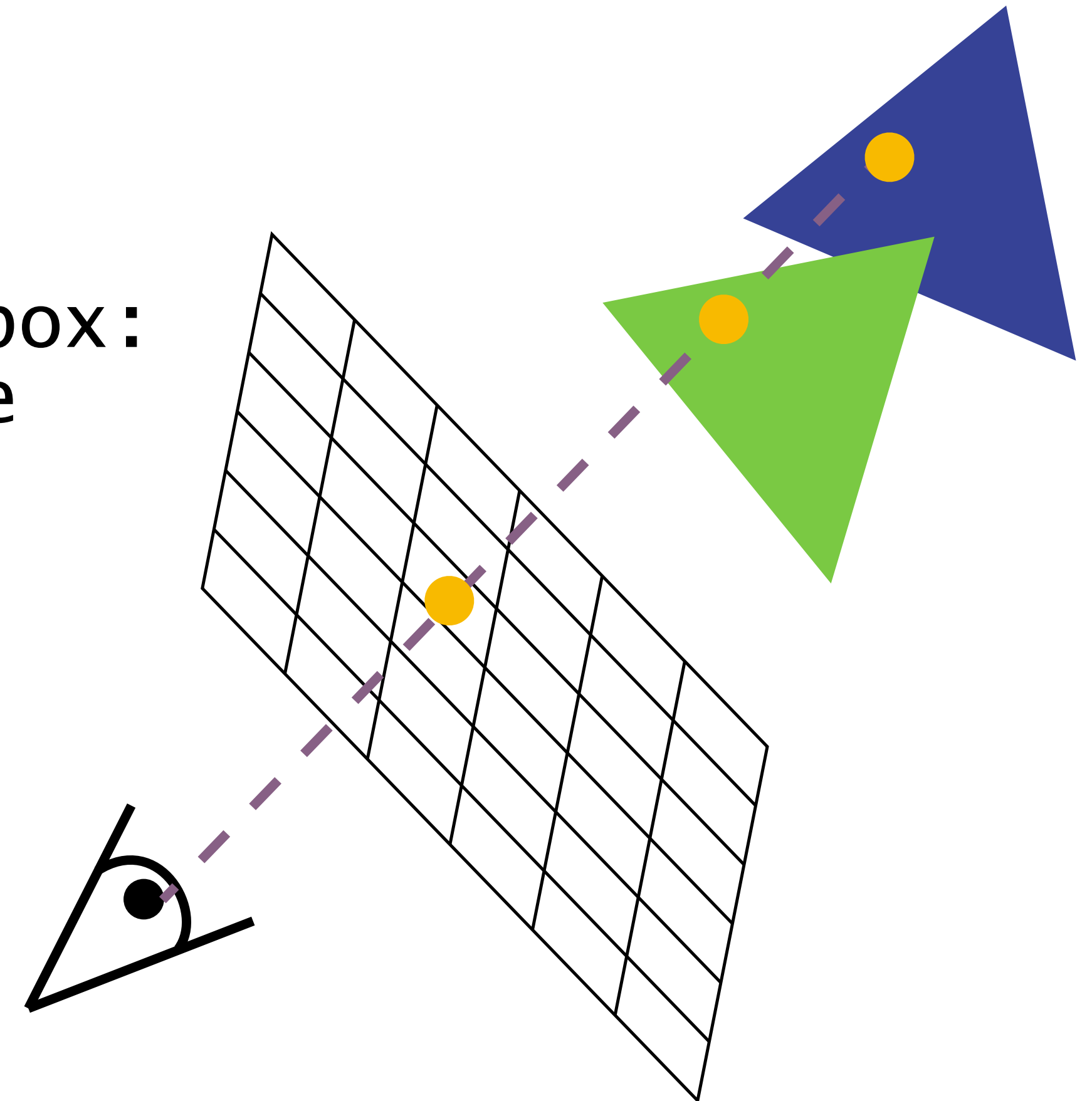


Speeding up rasterization

```
Z_buffer = inf(w, h)
img = zeros(w, h)
for each clipped 3D triangle:
  get the 2D triangle by dividing -z
  compute image space bounding box
  for each pixel (x, y) in the bounding box:
    if the pixel center hits the triangle
      interpolate Z
      if (Z < Z_buffer[x, y]):
        Z_buffer[x, y] = Z
        img[x, y] = ...
```

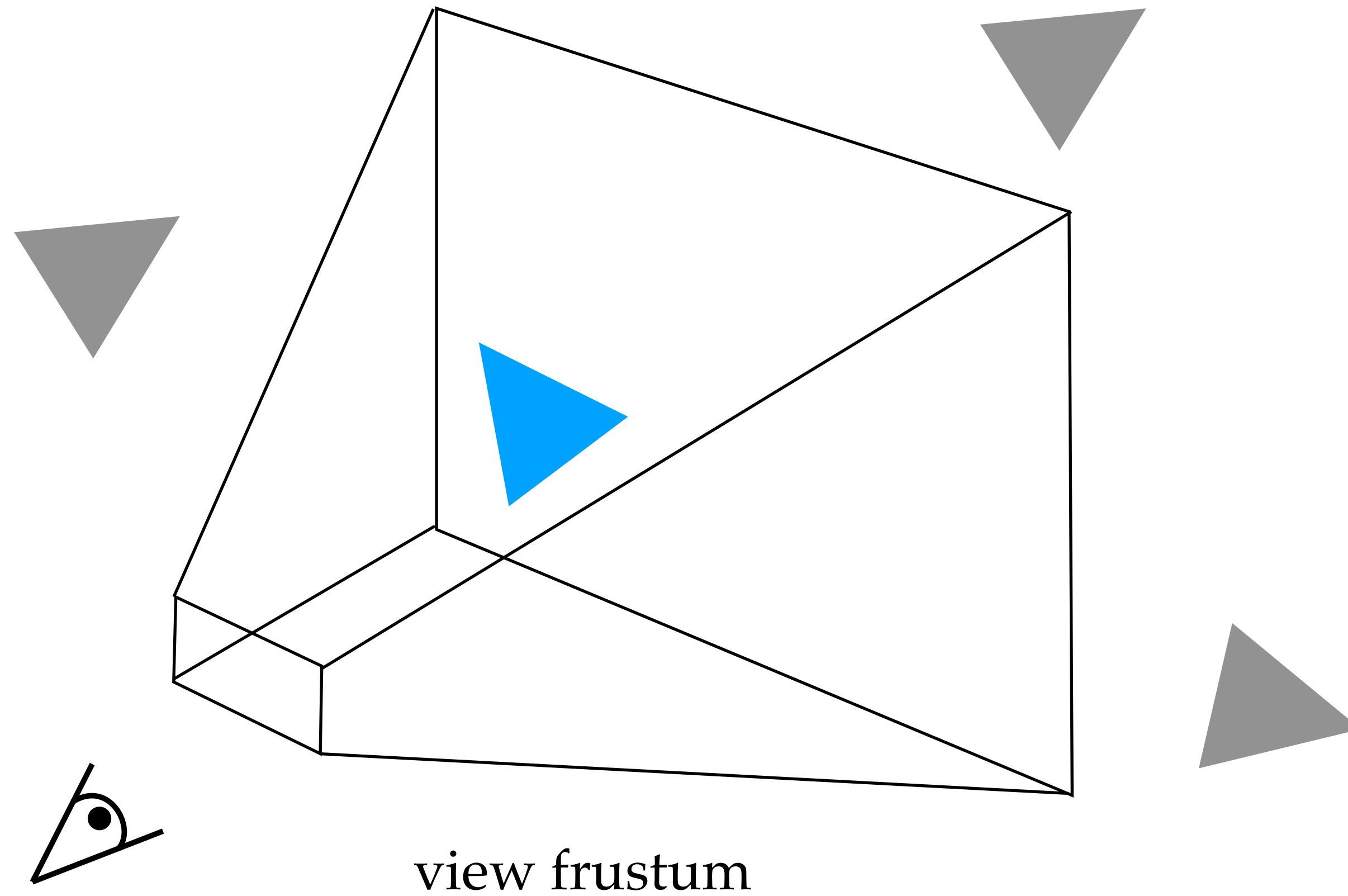
we can often skip triangles by

1. frustum culling, and 2. occlusion culling



Frustum culling

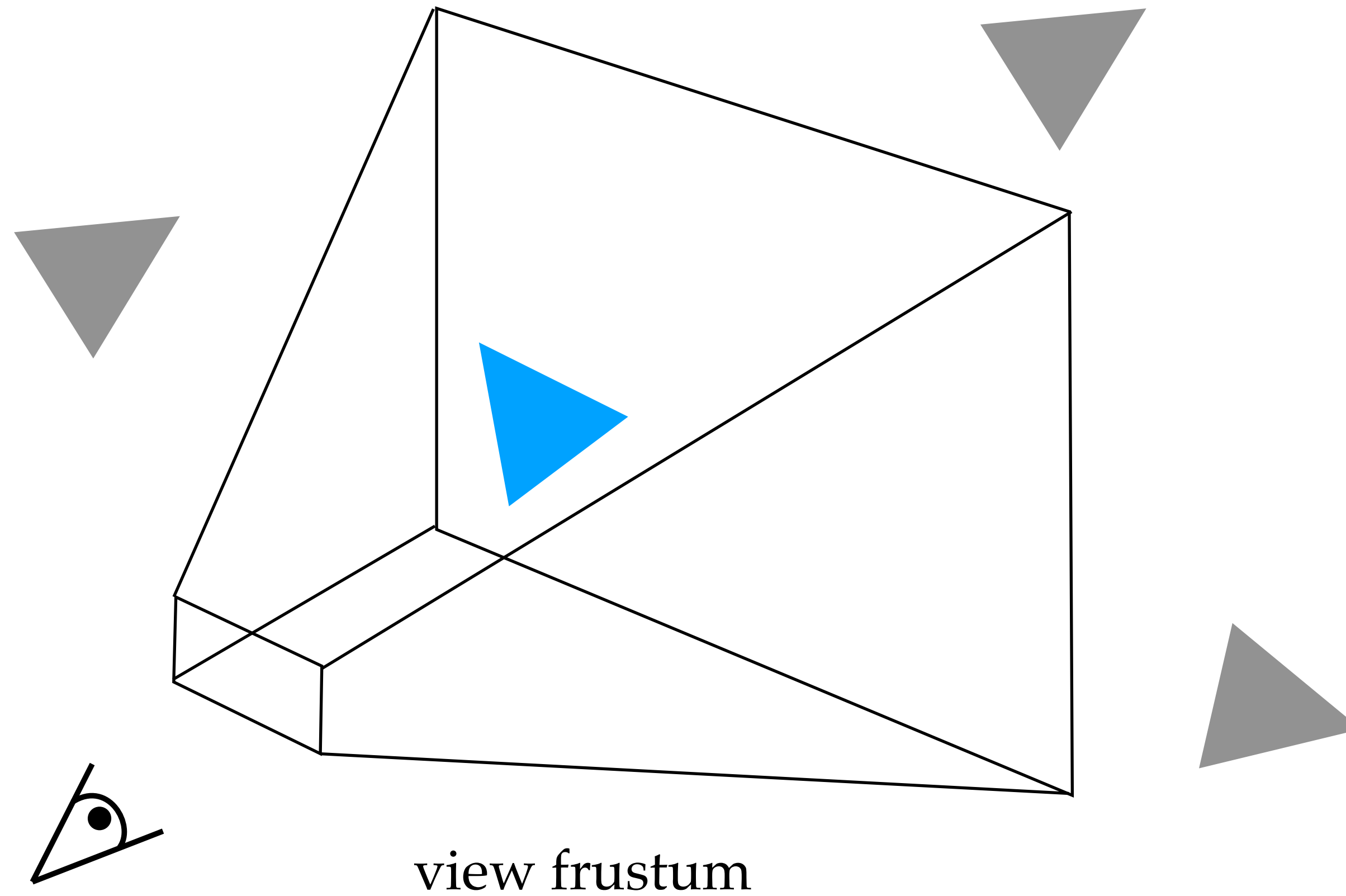
triangles outside of the “view frustum” can be discarded



Frustum culling

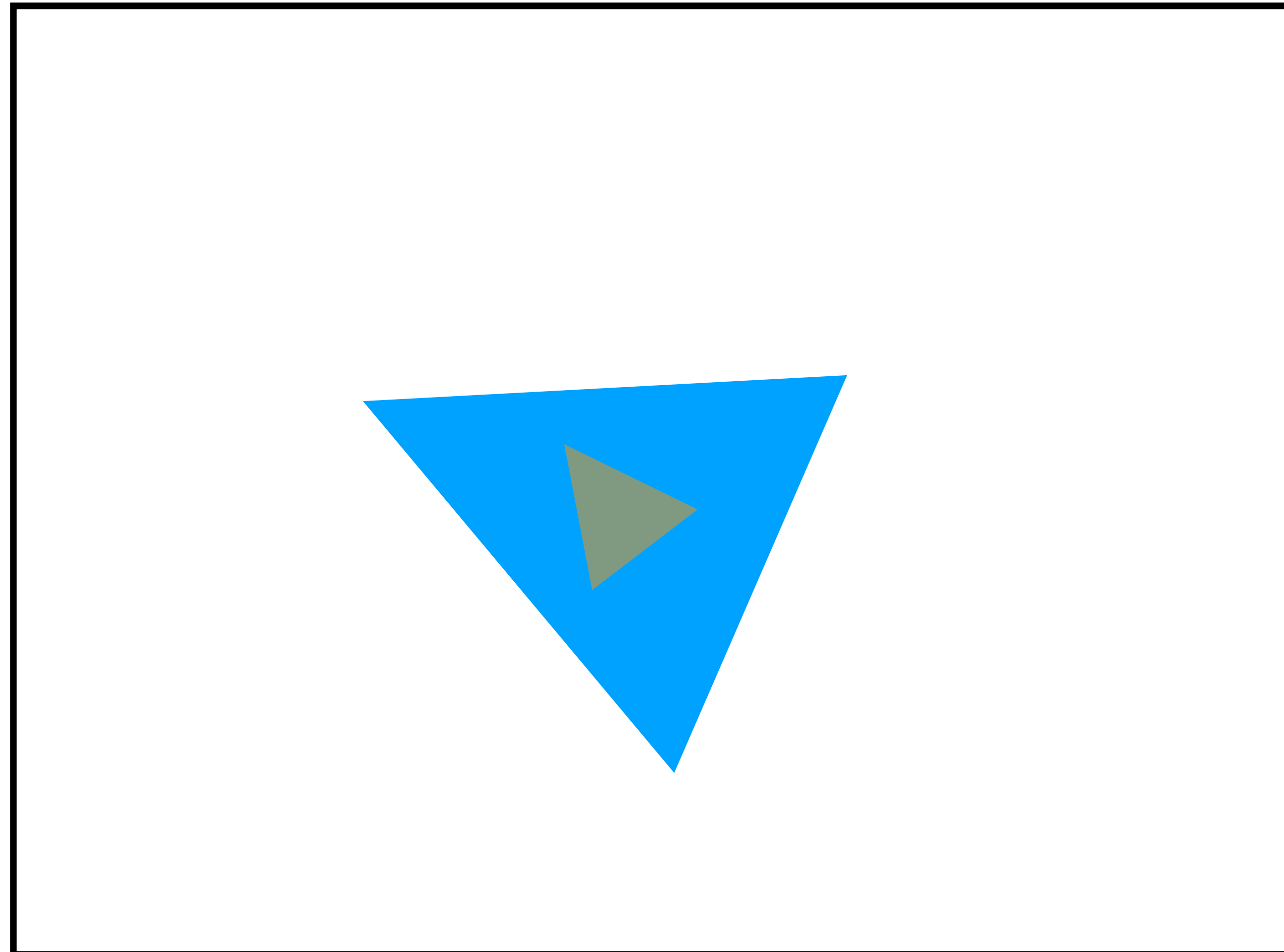
triangles outside of the “view frustum” can be discarded

many rasterizers also have a “far clipping plane” to discard triangles that are too far away



Occlusion culling

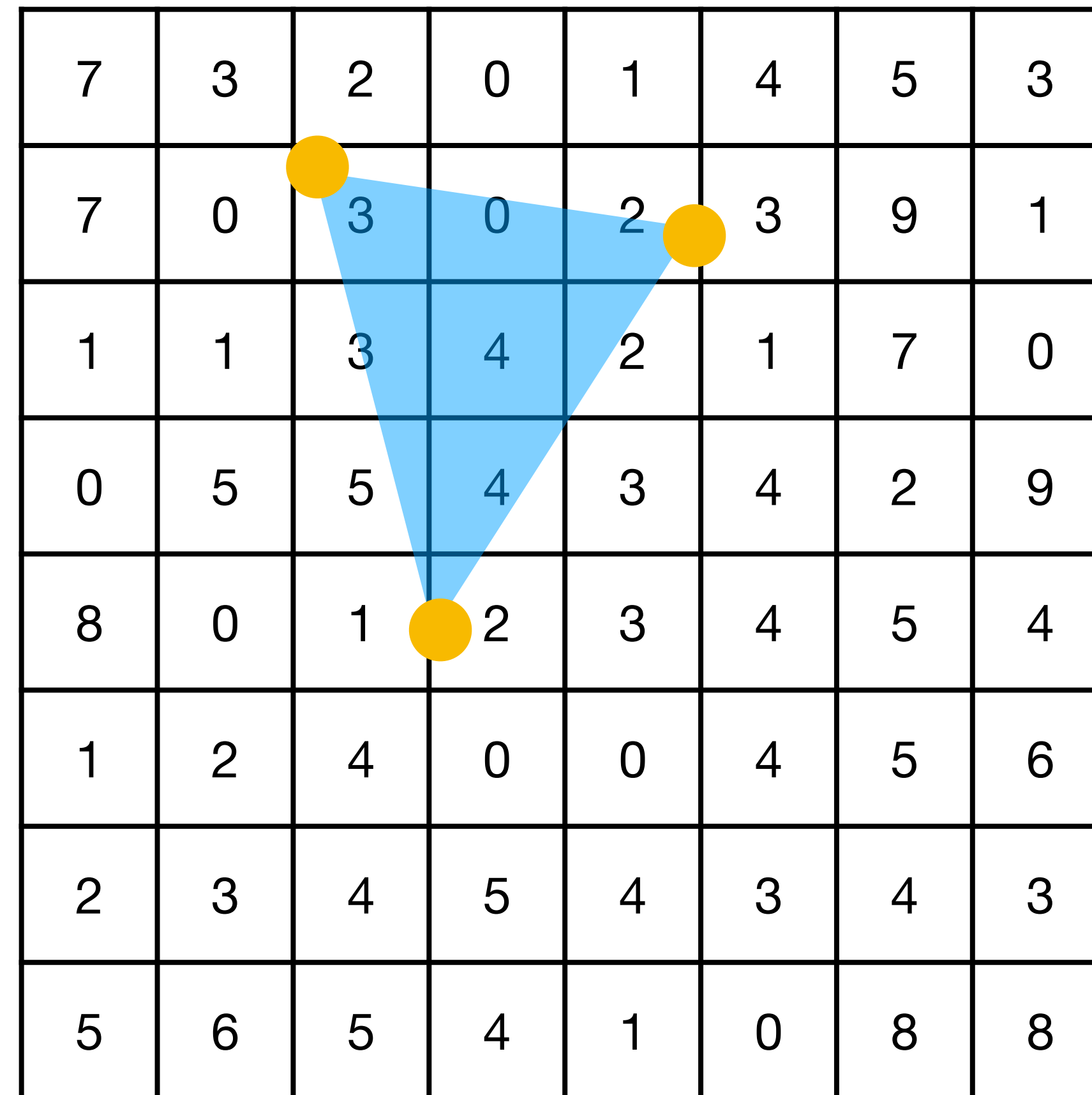
don't need to draw a triangle if it's fully blocked by others



Occlusion culling

Z buffer

- for each triangle, test with the current Z buffer
- find the minimum $|Z|$ among the three vertices (e.g., $\min |Z| = 6$)



Occlusion culling

- for each triangle, test with the current Z buffer
- find the minimum $|Z|$ among the three vertices (e.g., $\min |Z| = 6$)
- next, form the screen space bounding box

Z buffer

7	3	2	0	1	4	5	3
7	0	3	0	2	3	9	1
1	1	3	4	2	1	7	0
0	5	5	4	3	4	2	9
8	0	1	2	3	4	5	4
1	2	4	0	0	4	5	6
2	3	4	5	4	3	4	3
5	6	5	4	1	0	8	8

Occlusion culling

- for each triangle, test with the current Z buffer
- find the minimum $|Z|$ among the three vertices (e.g., $\min |Z| = 6$)
- next, form the screen space bounding box
- find the maximum $|Z|$ within the box (e.g., $\max |Z| = 5$)

Z buffer

7	3	2	0	1	4	5	3
7	0	3	0	2	3	9	1
1	1	3	4	2	1	7	0
0	5	5	4	3	4	2	9
8	0	1	2	3	4	5	4
1	2	4	0	0	4	5	6
2	3	4	5	4	3	4	3
5	6	5	4	1	0	8	8

Occlusion culling

- for each triangle, test with the current Z buffer
- find the minimum $|Z|$ among the three vertices (e.g., $\min |Z| = 6$)
- next, form the screen space bounding box
- find the maximum $|Z|$ within the box (e.g., $\max |Z| = 5$)
- skip the triangle if $\min |Z| > \max |Z|$

Z buffer

7	3	2	0	1	4	5	3
7	0	3	0	2	3	9	1
1	1	3	4	2	1	7	0
0	5	5	4	3	4	2	9
8	0	1	2	3	4	5	4
1	2	4	0	0	4	5	6
2	3	4	5	4	3	4	3
5	6	5	4	1	0	8	8

Occlusion culling

- for each triangle, test with the current Z buffer
- find the minimum $|Z|$ among the three vertices (e.g., $\min |Z| = 6$)
- next, form the screen space bounding box
slow if done naively!
- find the maximum $|Z|$ within the box (e.g., $\max |Z| = 5$)
- skip the triangle if $\min |Z| > \max |Z|$

Z buffer

7	3	2	0	1	4	5	3
7	0	3	0	2	3	9	1
1	1	3	4	2	1	7	0
0	5	5	4	3	4	2	9
8	0	1	2	3	4	5	4
1	2	4	0	0	4	5	6
2	3	4	5	4	3	4	3
5	6	5	4	1	0	8	8

Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

build a “pyramid” from the Z buffer

7	3	2	0	1	4	5	3
7	0	3	0	2	3	9	1
1	1	3	4	2	1	7	0
0	5	5	4	3	4	2	9
8	0	1	2	3	4	5	4
1	2	4	0	0	4	5	6
2	3	4	5	4	3	4	3
5	6	5	4	1	0	8	8

max
→

7	3	4	9
5	5	4	9
8	4	4	6
6	5	4	8

max
→

7	9
8	8

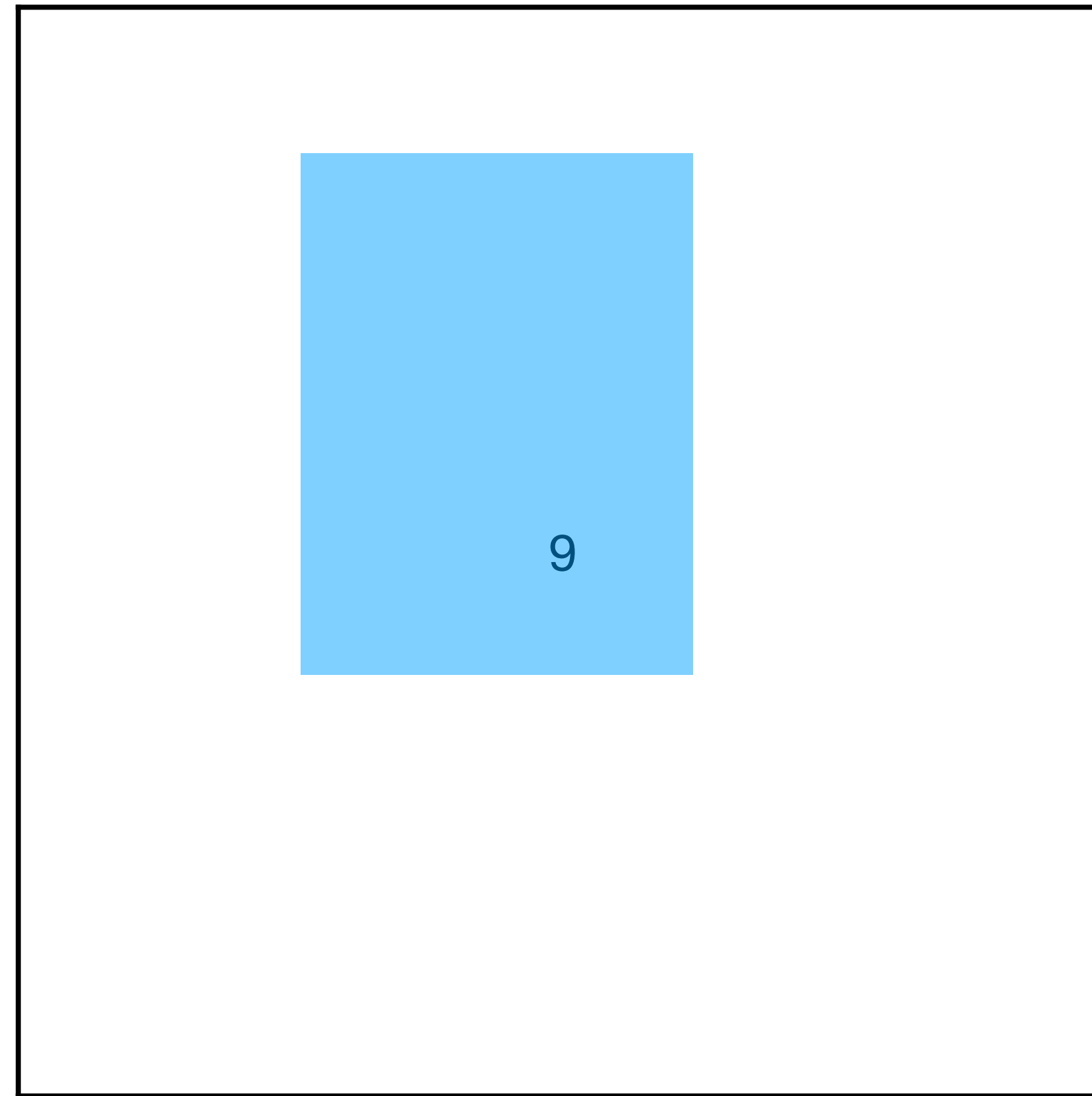
Hierarchical Z-Buffer Visibility

Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min Z = 6

- test the top level of the pyramid

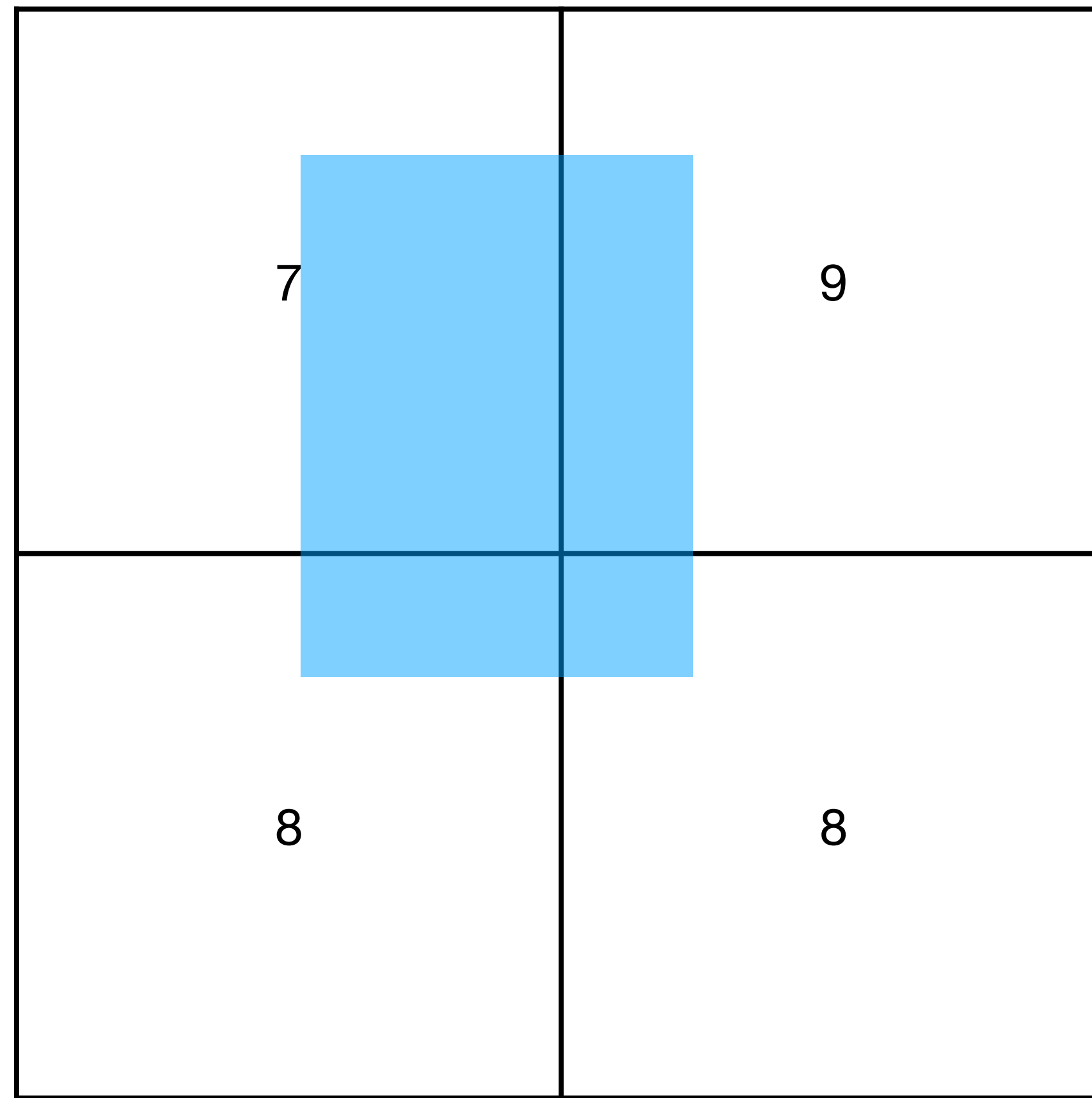


Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min Z = 6

- test the top level of the pyramid
- if not skipped, test the second level

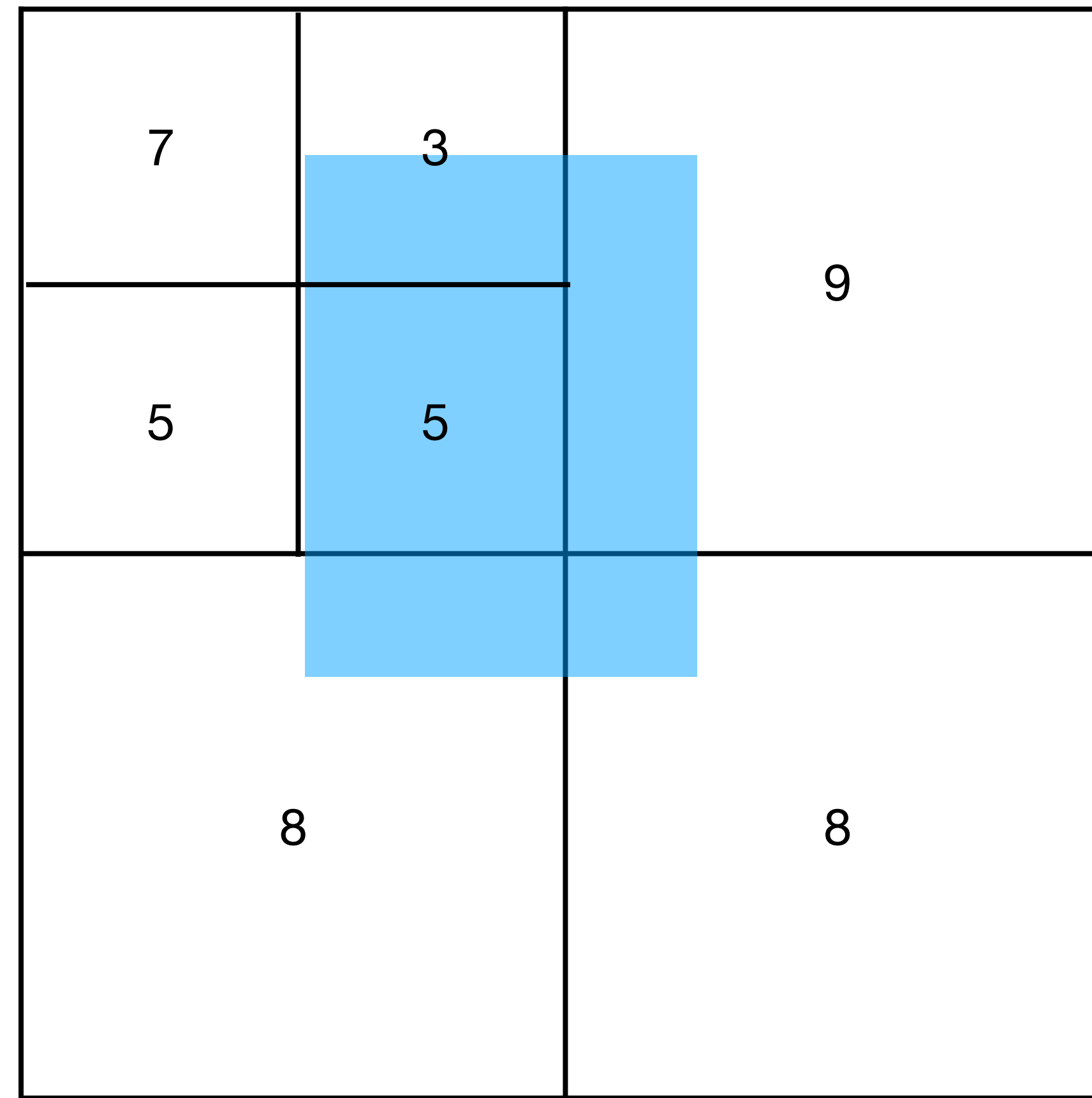


Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min Z = 6

- test the top level of the pyramid
- if not skipped, test the second level
- recurse into each cell

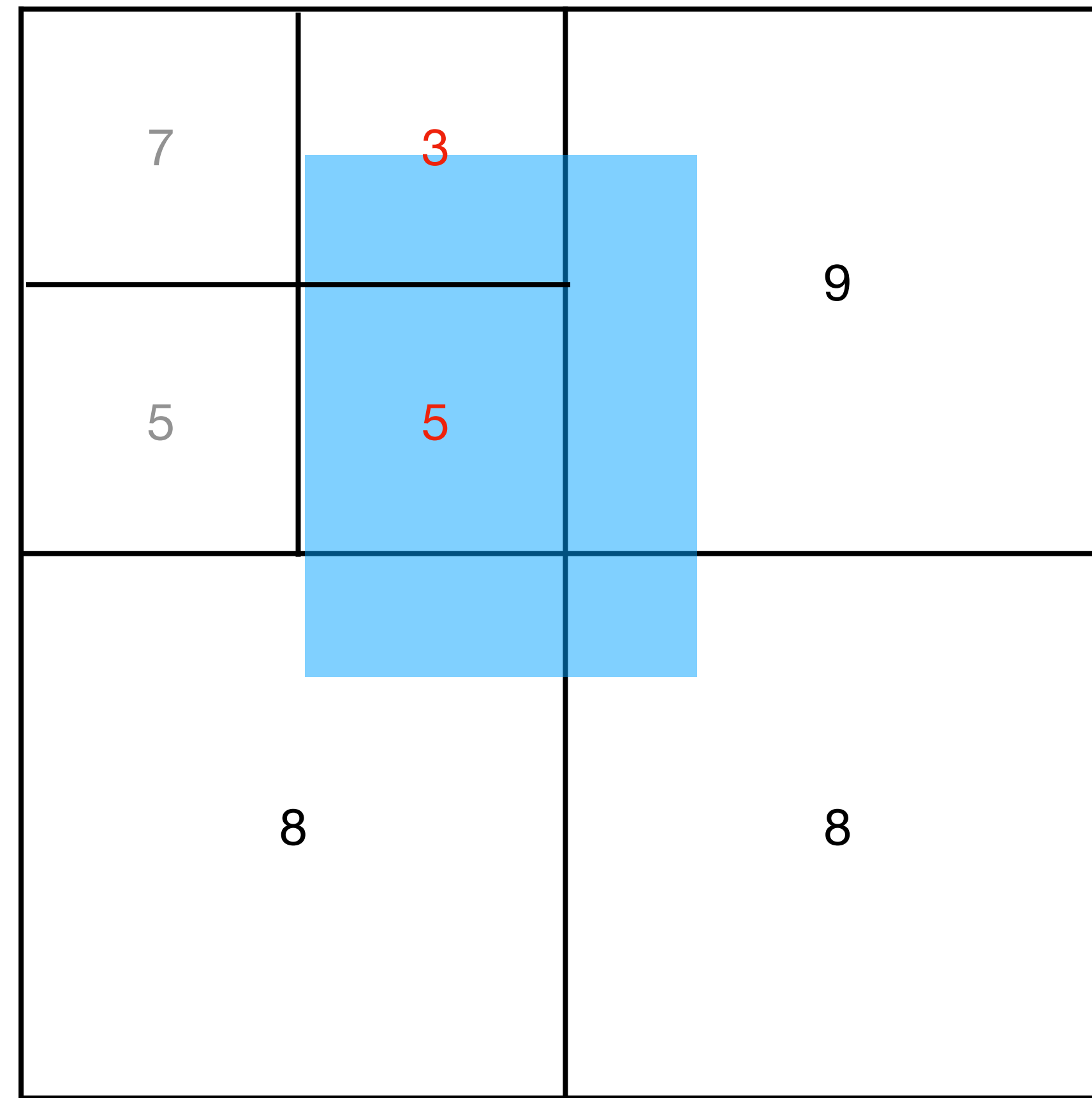


Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min $Z = 6$

- test the top level of the pyramid
- if not skipped, test the second level
- recurse into each cell
- stop recursing when $\max Z < \min Z$ (or no overlap with the bounding box)

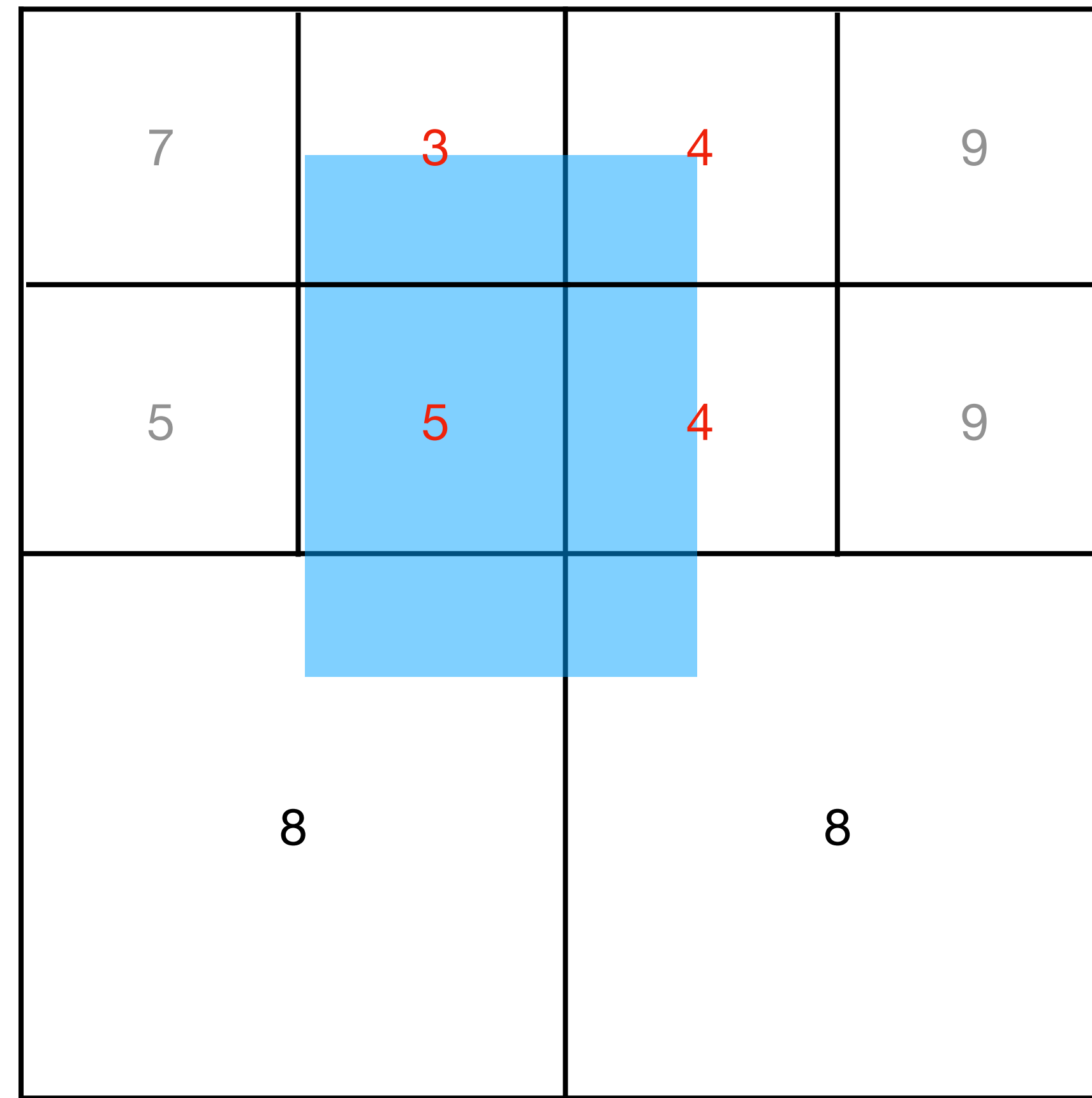


Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min $Z = 6$

- test the top level of the pyramid
- if not skipped, test the second level
- recurse into each cell
- stop recursing when $\max Z < \min Z$ (or no overlap with the bounding box)

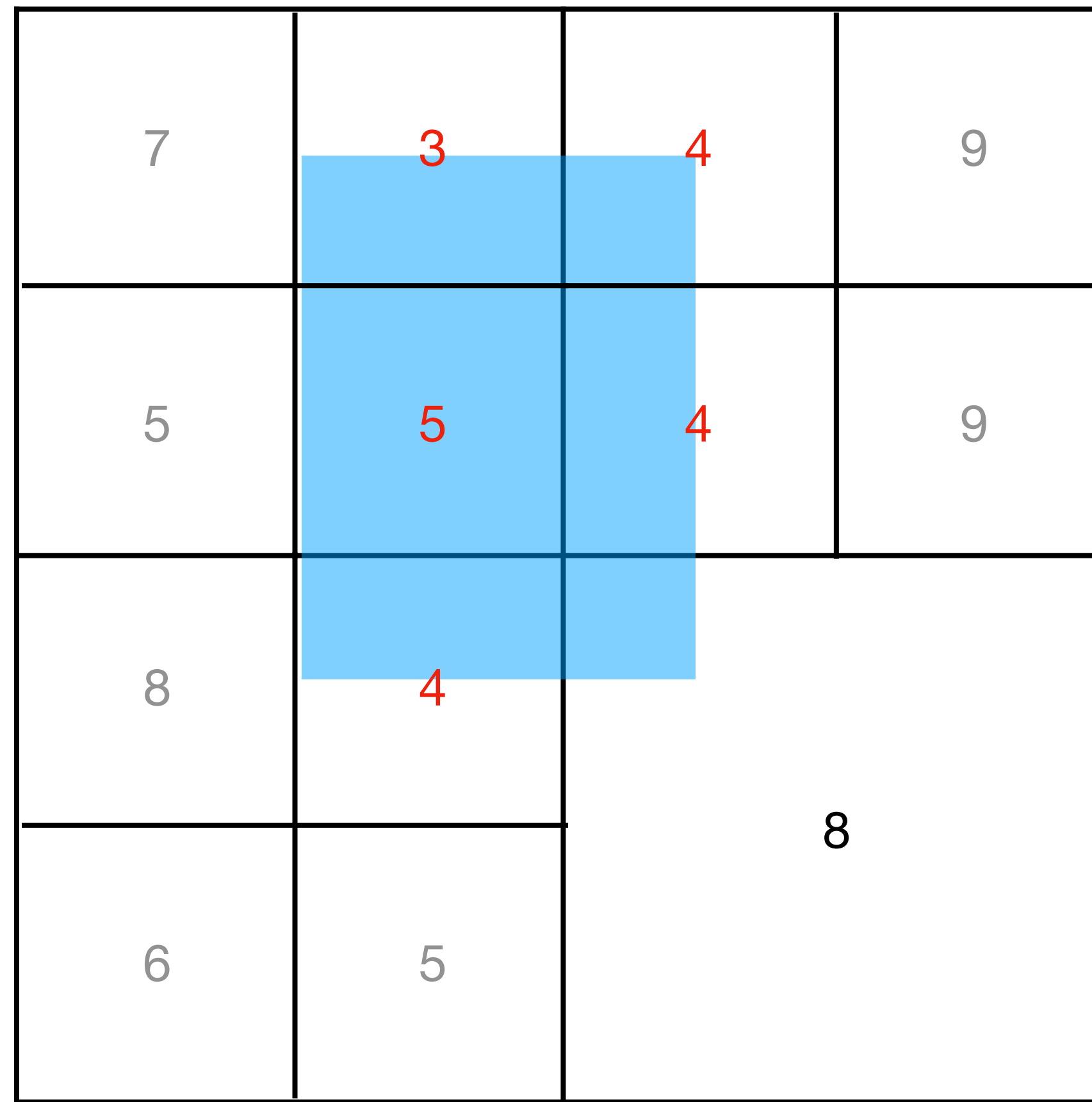


Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min $Z = 6$

- test the top level of the pyramid
- if not skipped, test the second level
- recurse into each cell
- stop recursing when $\max Z < \min Z$ (or no overlap with the bounding box)



Hierarchical Z-buffer for occlusion culling

[Greene et al. 1993]

min $Z = 6$

- test the top level of the pyramid
- if not skipped, test the second level
- recurse into each cell
- stop recursing when $\max Z < \min Z$ (or no overlap with the bounding box)

7	3	4	9
5	5	4	9
8	4	4	6
6	5	4	8

In practice: usually cull many triangles together

Triangle cluster culling

- Group triangles into clusters
 - Build bounding data for each cluster
- Cull clusters based on bounds
 - Frustum cull
 - Occlusion cull



SIGGRAPH 2021

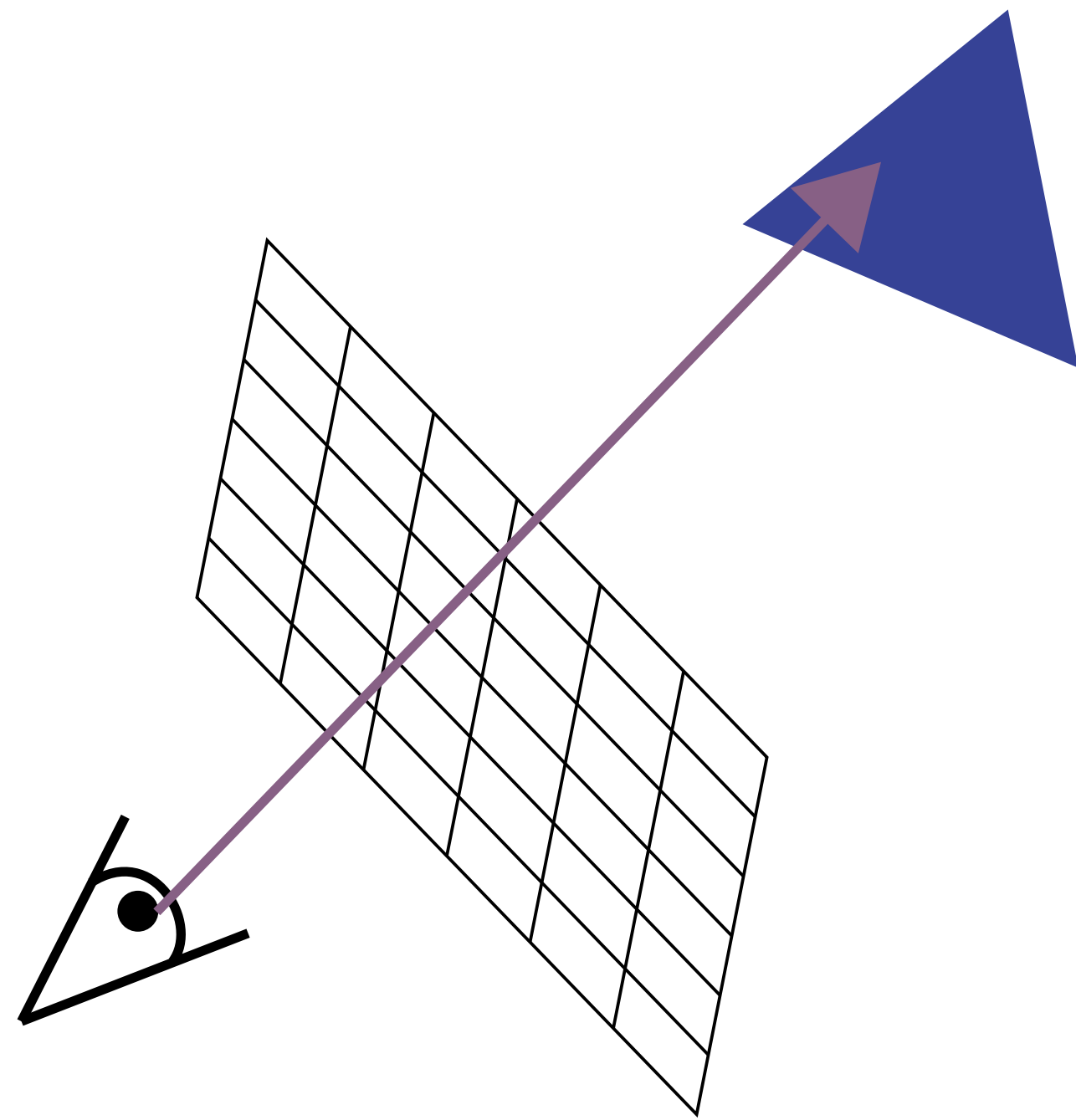
Hierarchical Z-Buffer is still used today!

Occlusion culling

- Occlusion cull against Hierarchical Z-Buffer (HZB)
- Calculate screen rect from bounds
- Test against lowest mip where screen rect $\leq 4 \times 4$ pixels



Next: ray tracing vs rasterization



v.s.

