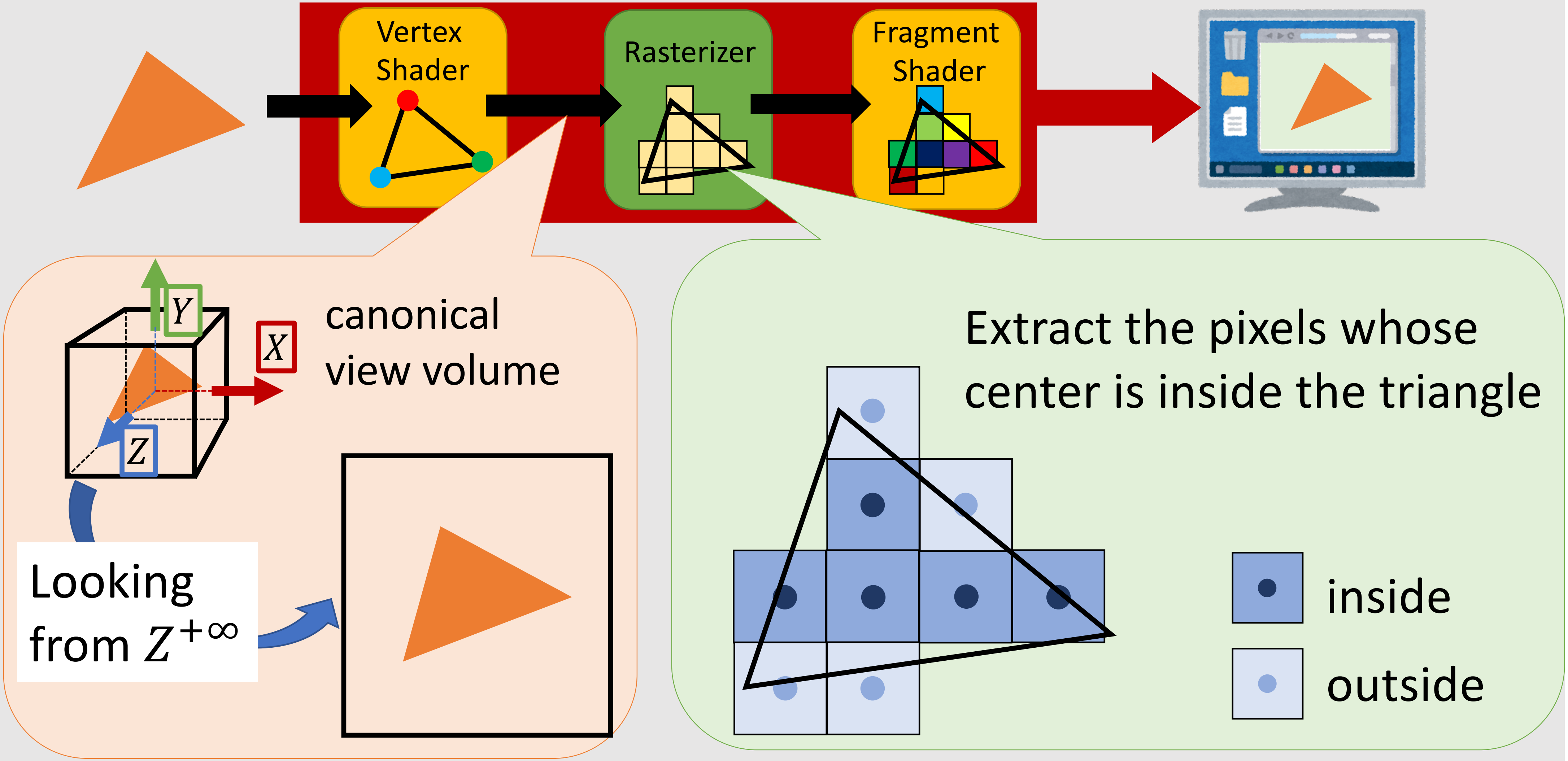
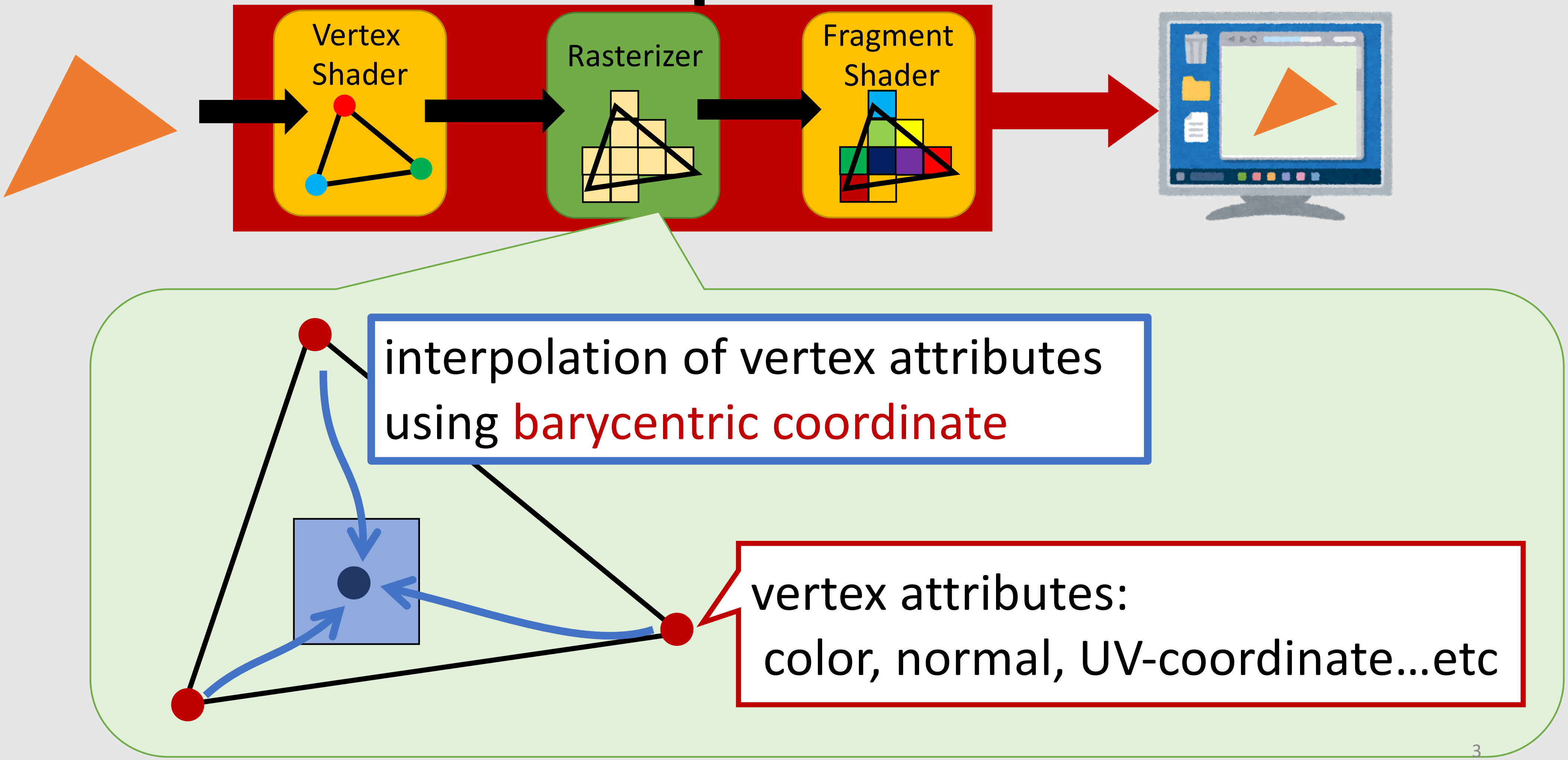
Rasterization

Rasterization



Rasterizer and Interpolation

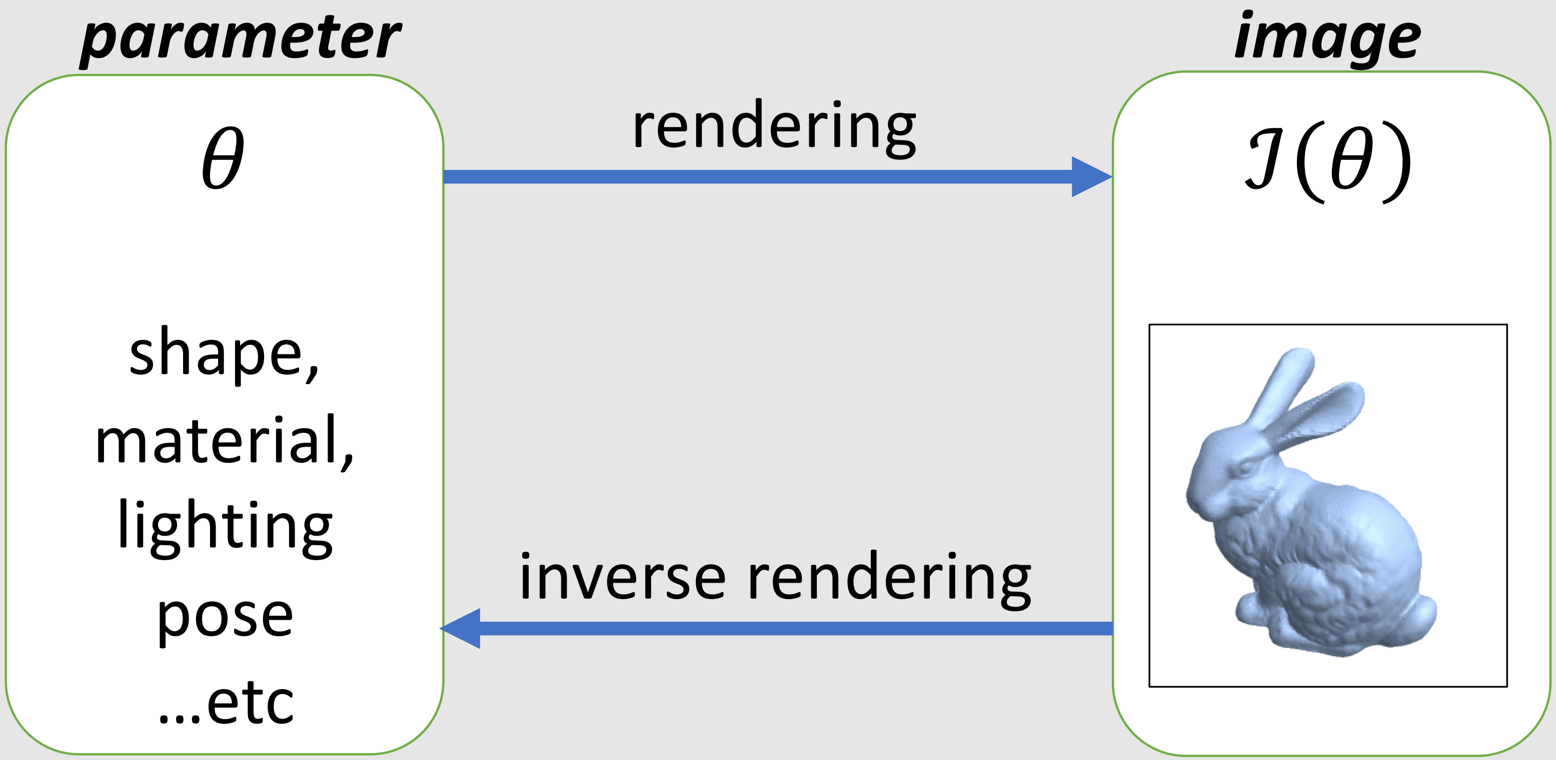


Why You Need to Understand Rasterization?

- Understand modern gaming engine archtecture

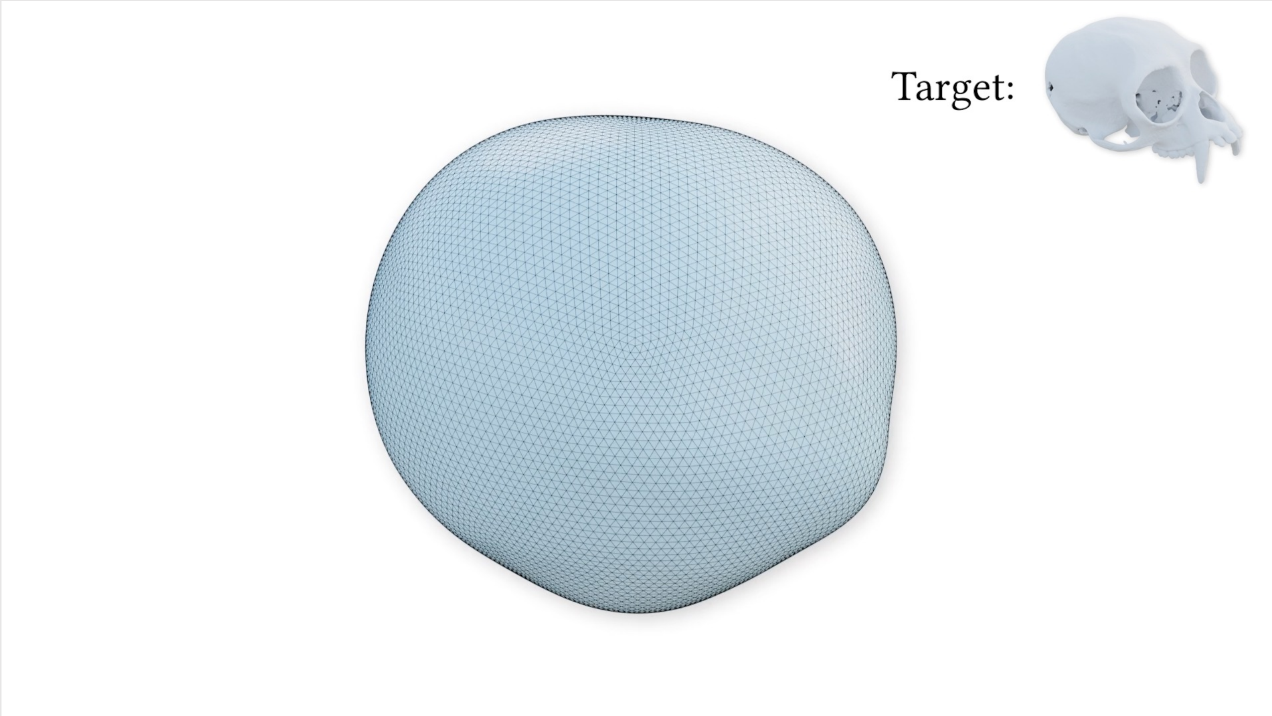
Why You Need to Understand Rasterization?

- Differentible rendering, inverse rendering



$$\frac{\partial \mathcal{J}(\theta)}{\partial \theta}$$

Example of Differentiable Rendering

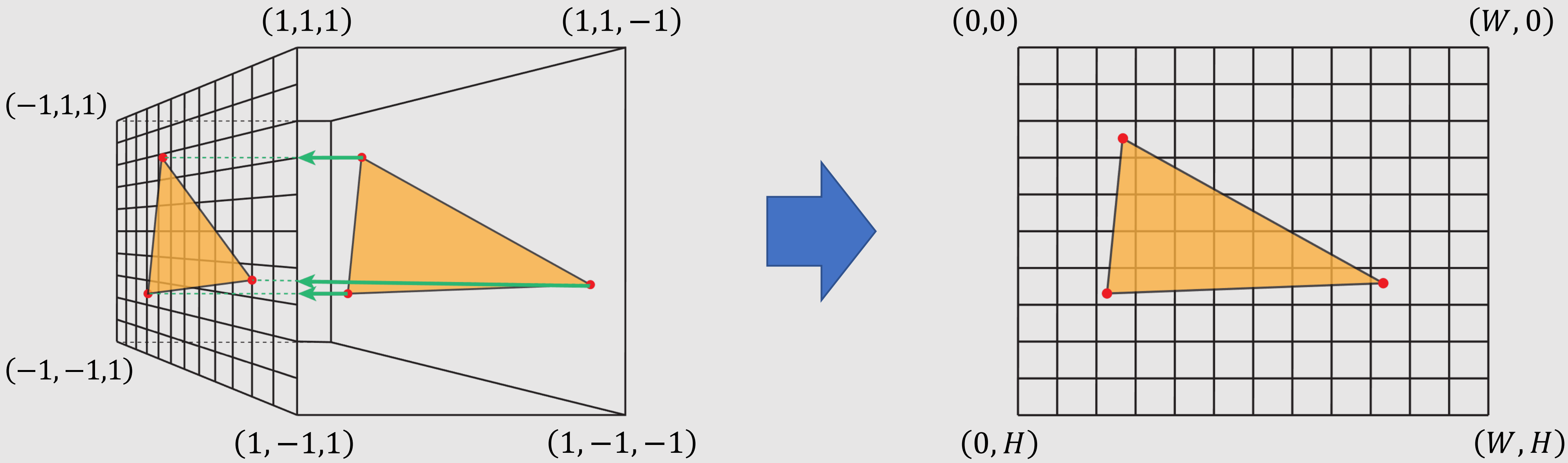


Large Steps in Inverse Rendering of Geometry,
Baptiste Nicolet Alec Jacobson Wenzel Jakob
In ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2021)

Rasterization: The First Step

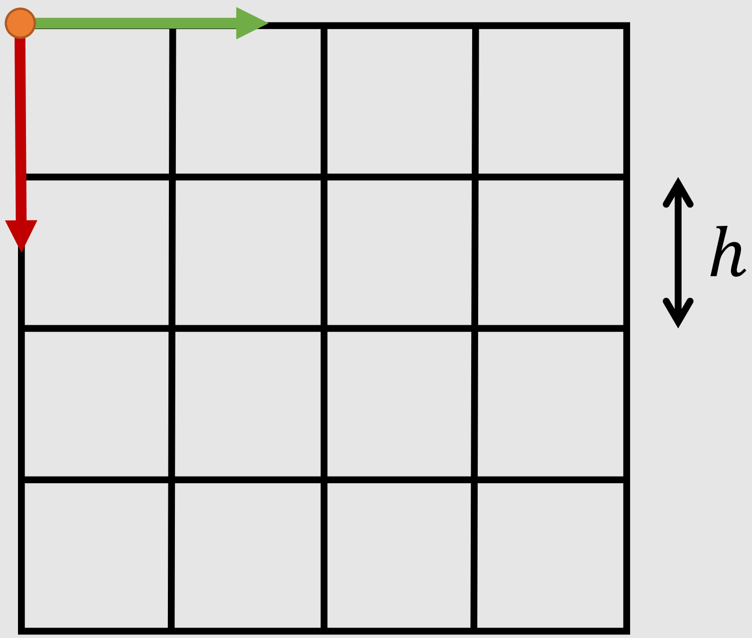
Orthongonal projection in cannonical view volume

Triangle in the image coordinate



Regular Grids

- Most common discretization for spatial values

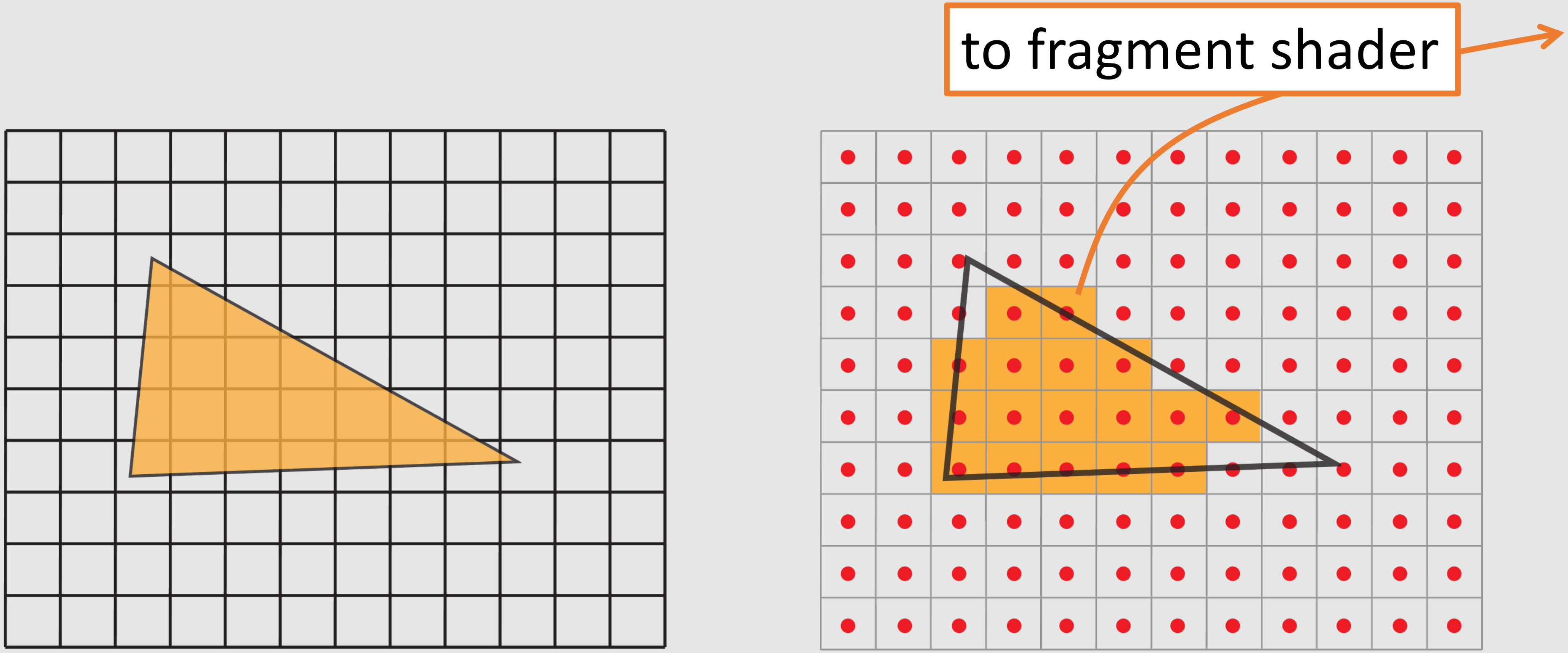


Let's find out the corresponding grid cell for (p_x, p_y)

Check it out!

Inside & Outside Test at the Center of Pixel

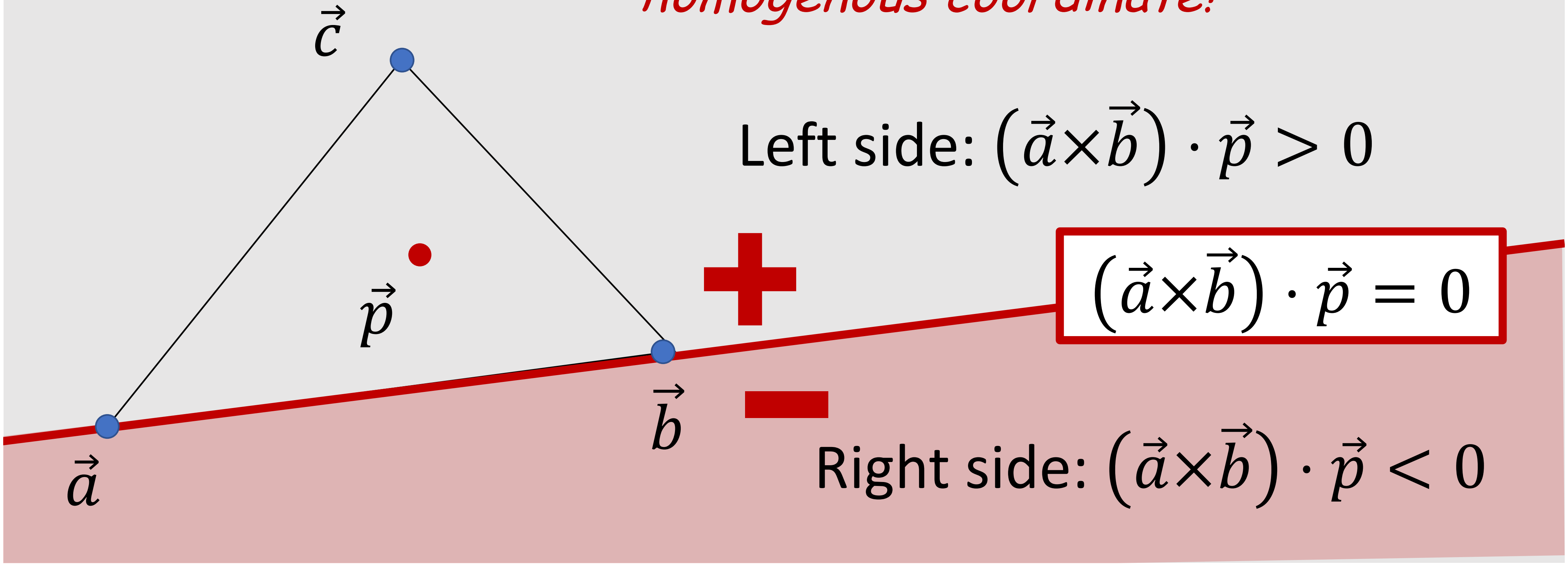
- Extract the pixels whose center is inside the triangle



Triangle Inside & Outside Test

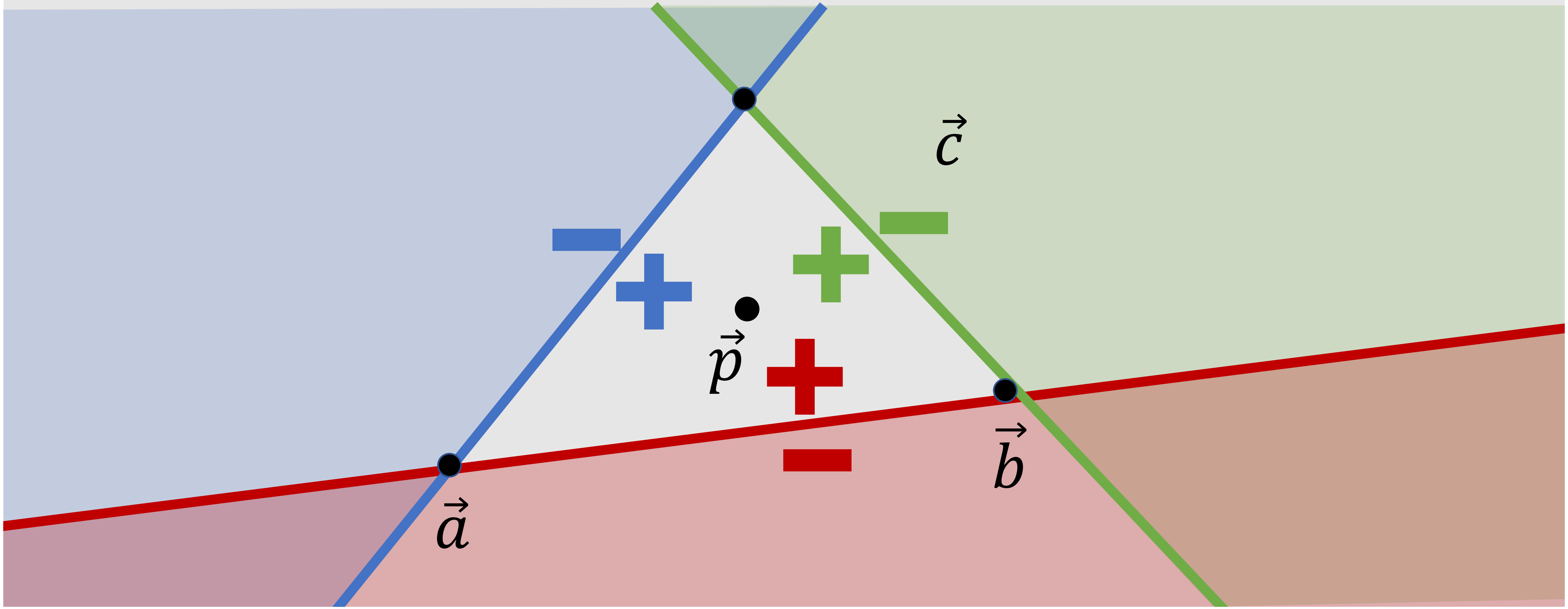
- Edge connecting $\vec{a} = \{a_x, a_y, 1\}^T$ and $\vec{b} = \{b_x, b_y, 1\}^T$

homogenous coordinate!

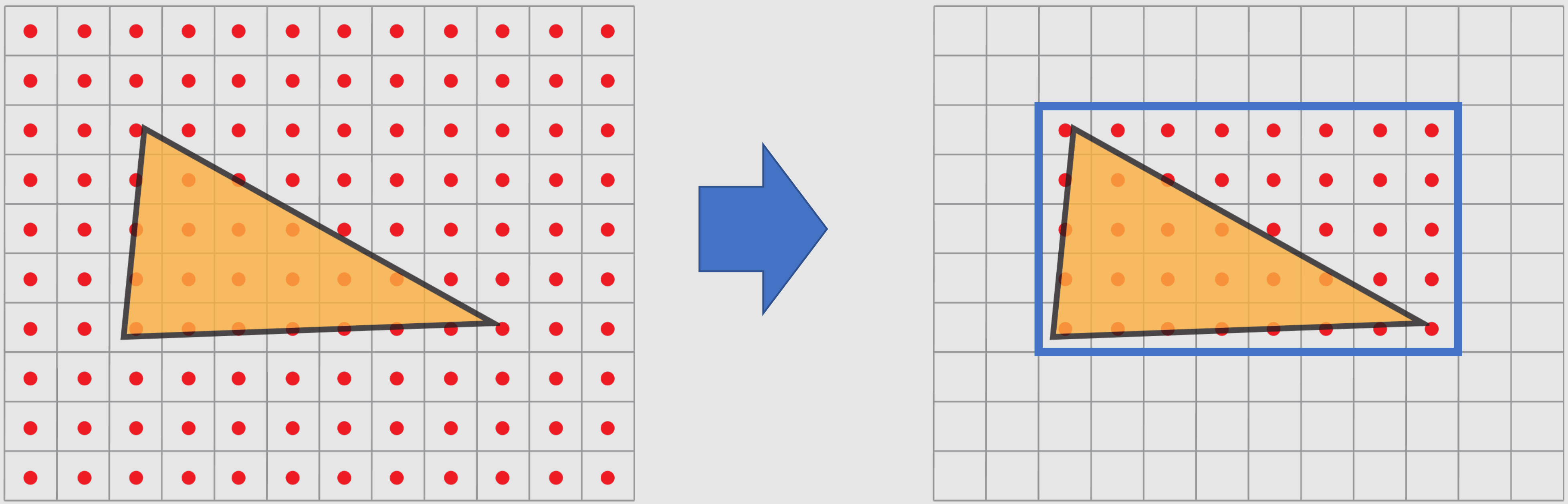


Triangle Inside & Outside Test

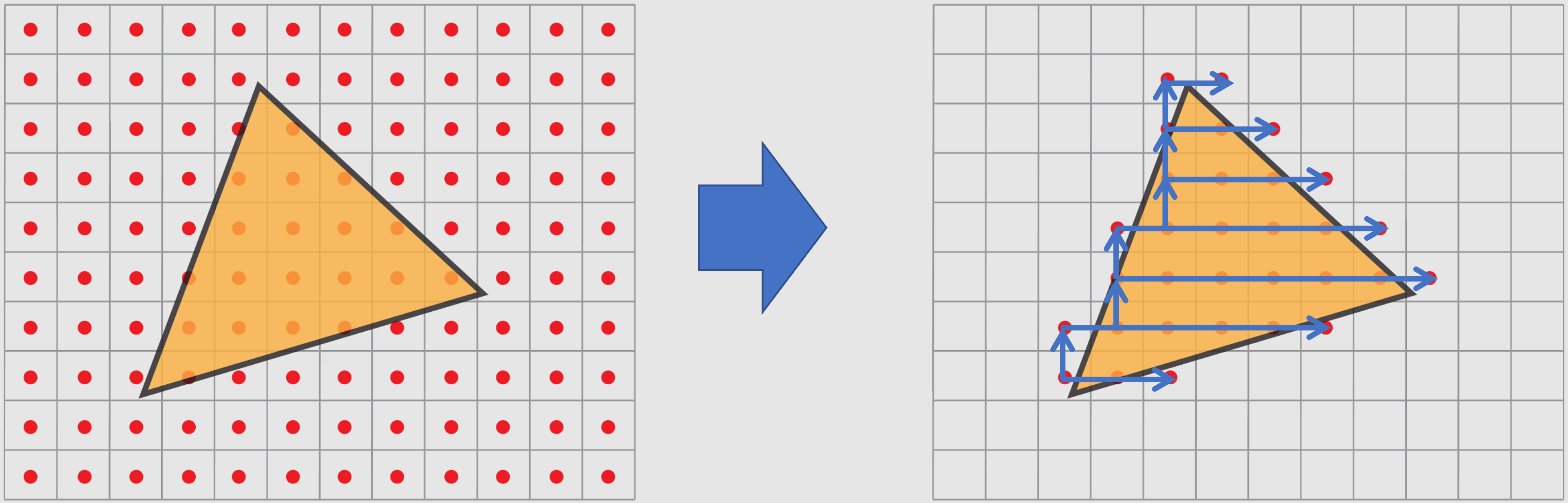
- Check if the signs are the same for all the three edges



Optimizing Test 1: Bounding Box



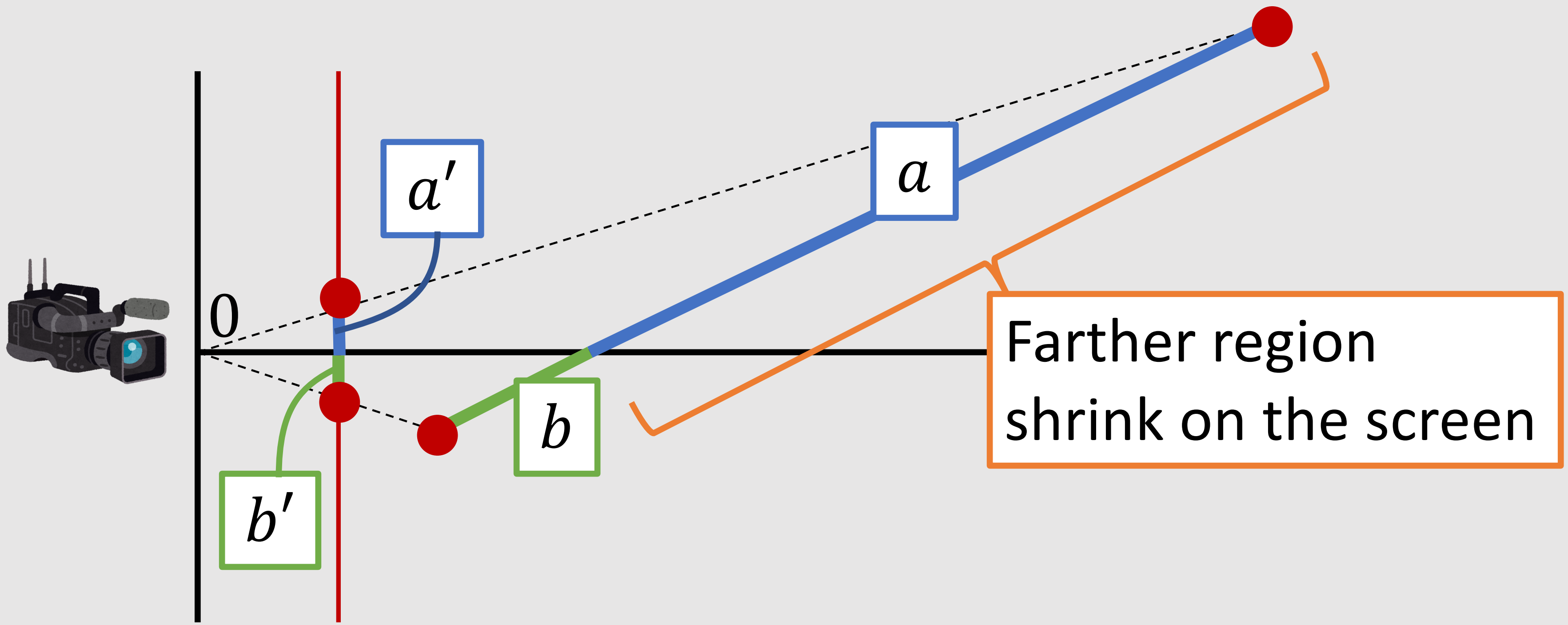
Optimizing Test 2: Pineda's Algorithm



Juan Pineda. 1988. A parallel algorithm for polygon rasterization. In Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH '88).

Distortion in Perspective Interpolation

$$a : b \neq a' : b'$$



Interpolation on Screen vs Object

top view

project then interpolate
"Gouraud" interpolation

interpolate then project
perspectively correct interpolation

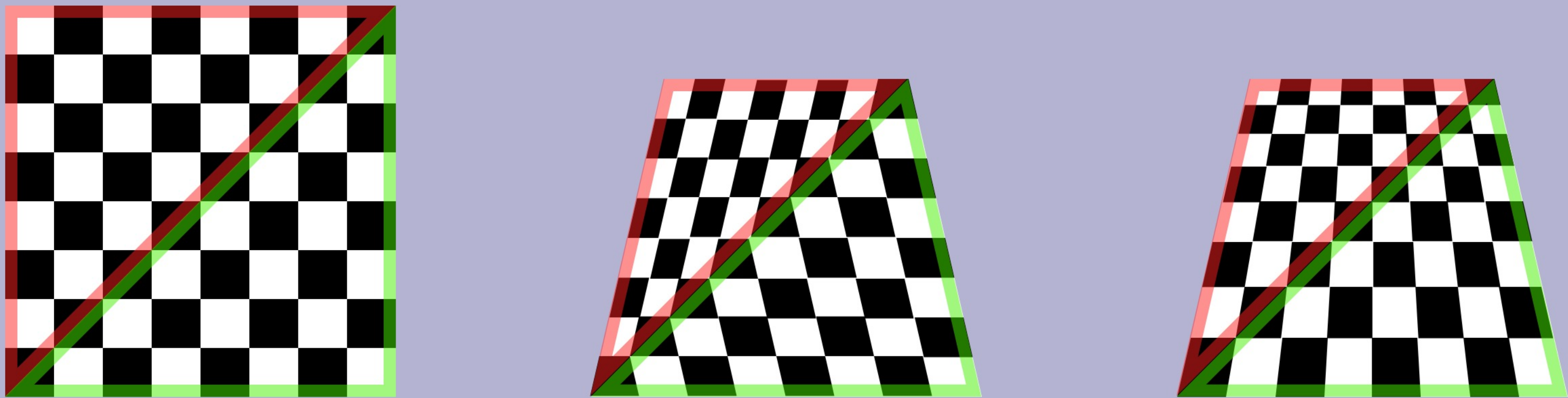
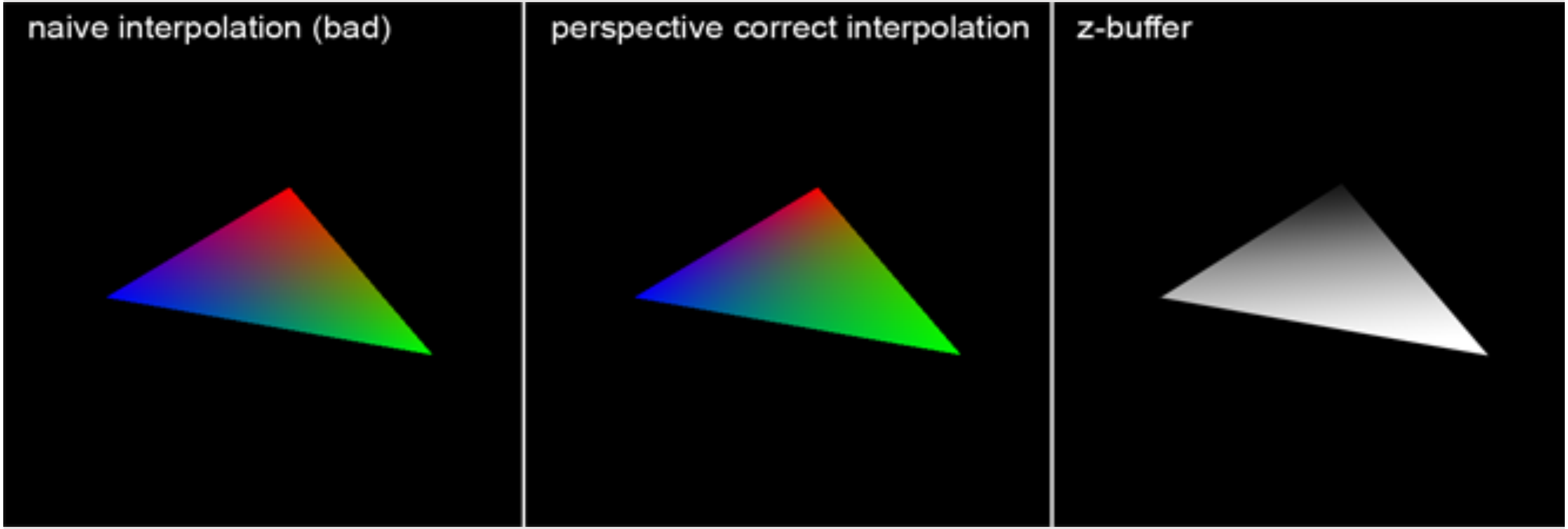
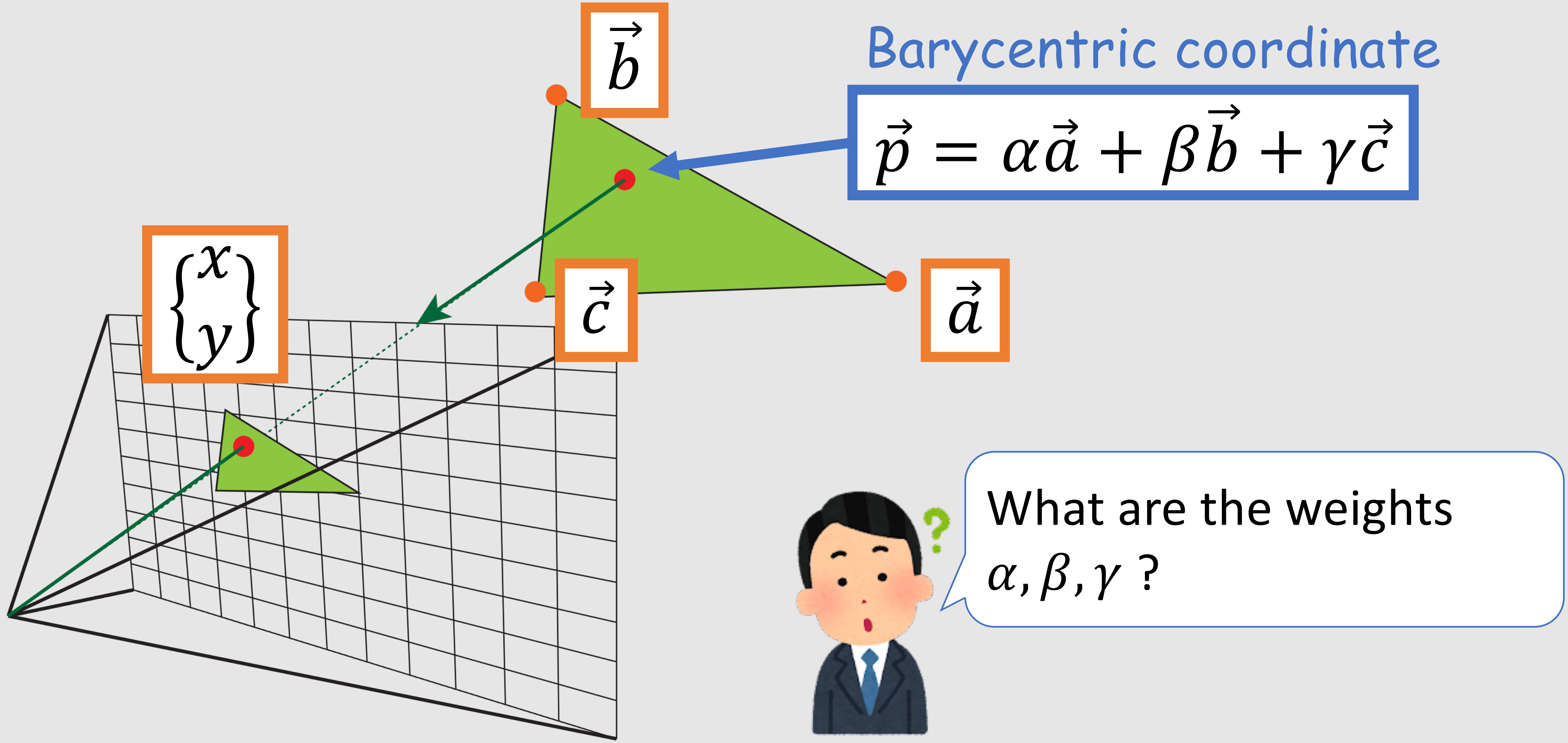


Image Credit: Darkness3560 @ Wikipedia

Perspectively Correct Interpolation

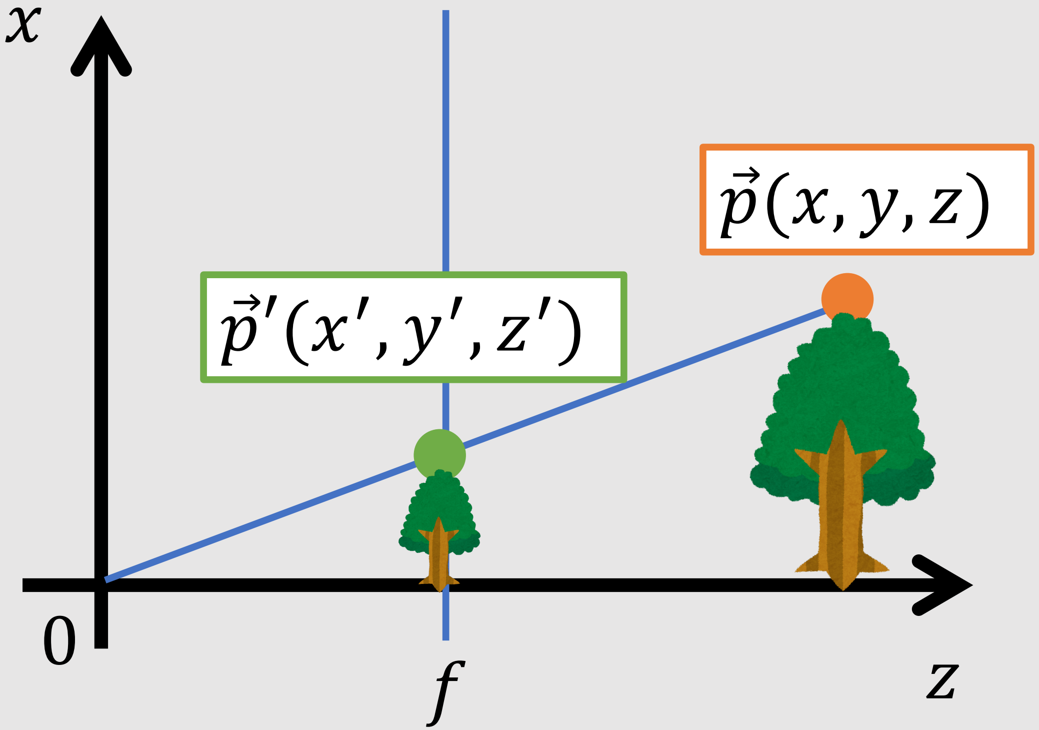


Perspectively Correct Interpolation



Simple Perspective

- Projecting $\vec{p}(x, y, z)$ on the image plane $z = f$ (f: focal length)



$$z' = f, \qquad \frac{x'}{x} = \frac{y'}{y} = \frac{z'}{z}$$

$$x' = \frac{fx}{z}, \qquad y' = \frac{fy}{z}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \propto \begin{pmatrix} x'' \\ y'' \\ w \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Baricentric Weights for Interpolation

- Distributive property of matrix

α, β, γ must satisfy
$\alpha + \beta + \gamma = 1$

$$\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} \propto \begin{Bmatrix} x' \\ y' \\ w \end{Bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p}$$

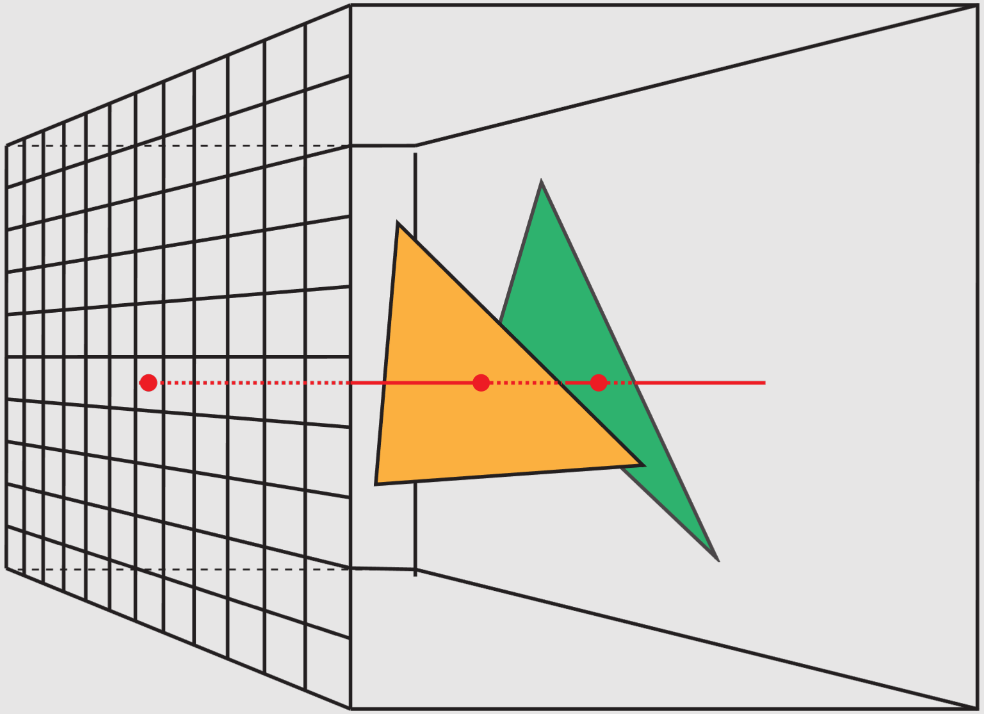
$$= H(\alpha\vec{a} + \beta\vec{b} + \gamma\vec{c})$$

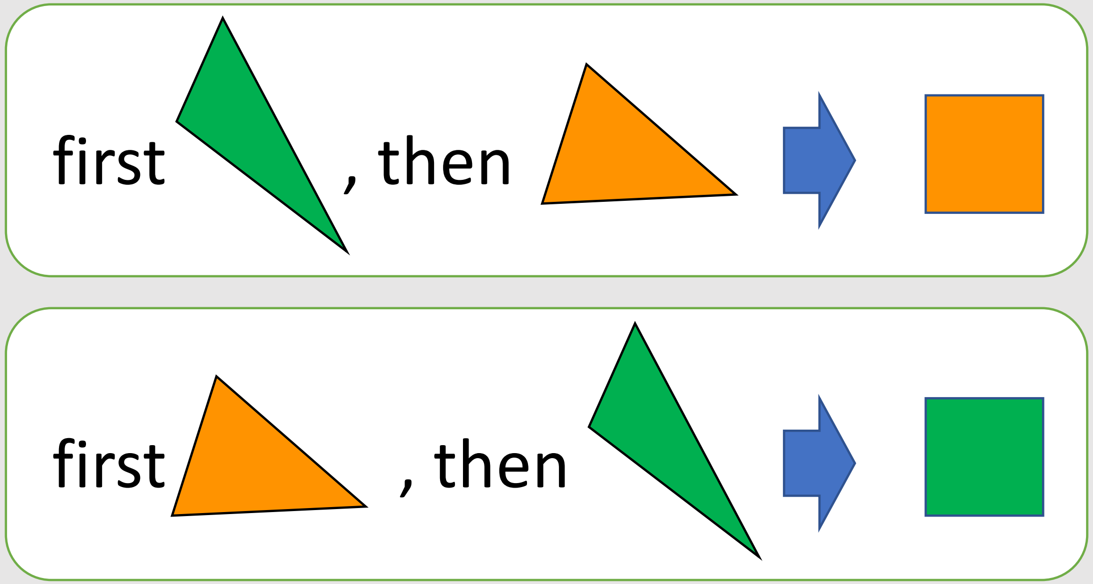
$$= (H\vec{a})\alpha + \left(H\vec{b}\right)\beta + (H\vec{c})\gamma$$

4 degrees of freedom,
4 constraints

How To Remove Order Dependency?

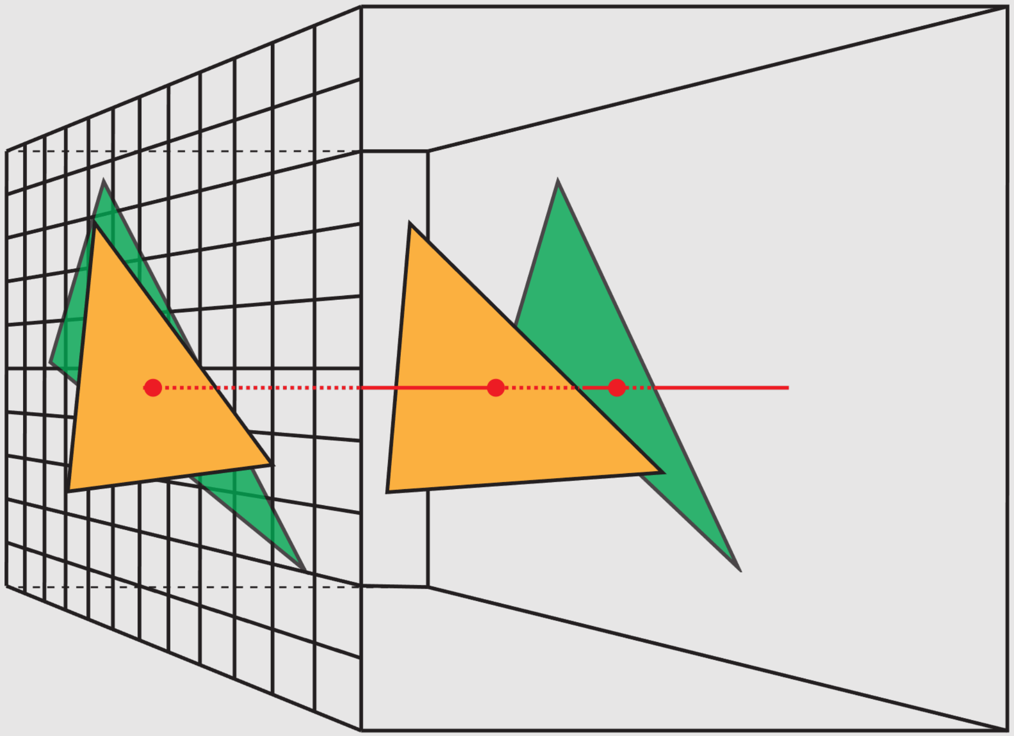
- Since shader is executed in ***parallel,*** difficult to handle ***occlusion***





Z-Buffer Algorithm

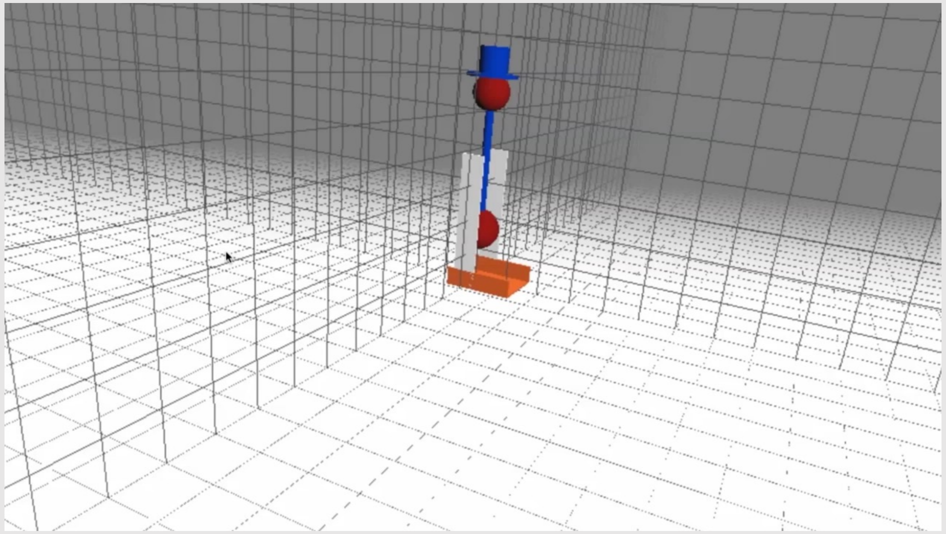
- Frame-buffer that keeps minimum depth for each pixel



```
for each triangle
  for each pixel (x,y)
    if (x,y) is inside triangle
      compute z
      if z < zbuffer[x,y]
        zbuffer[x,y]=z
        framebuffer[x,y]=shade()
```

Z-Fighting

- Flickering when different triangles has the same (or similar) depth



Z-Fighting - Interactive 3D Graphics
https://www.youtube.com/watch?v=CjckWVwd2ek

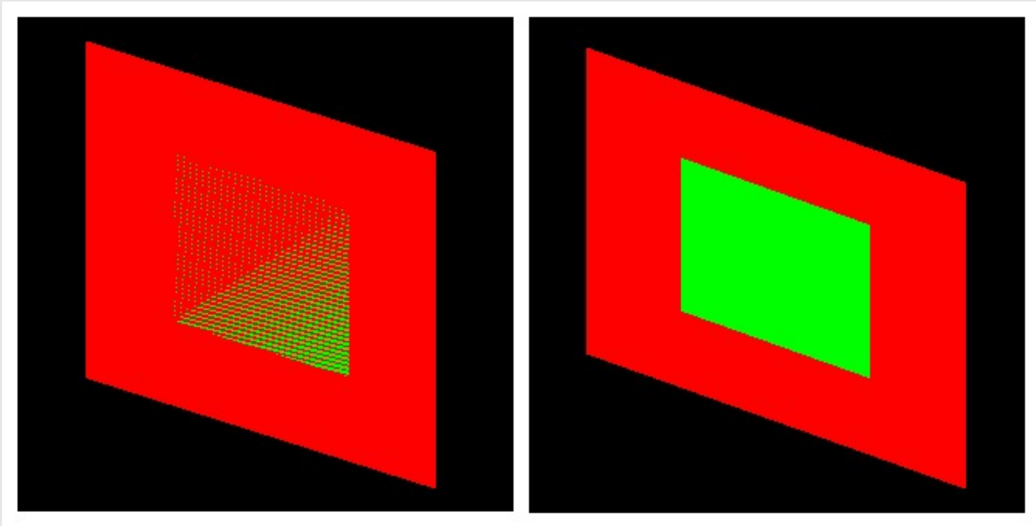
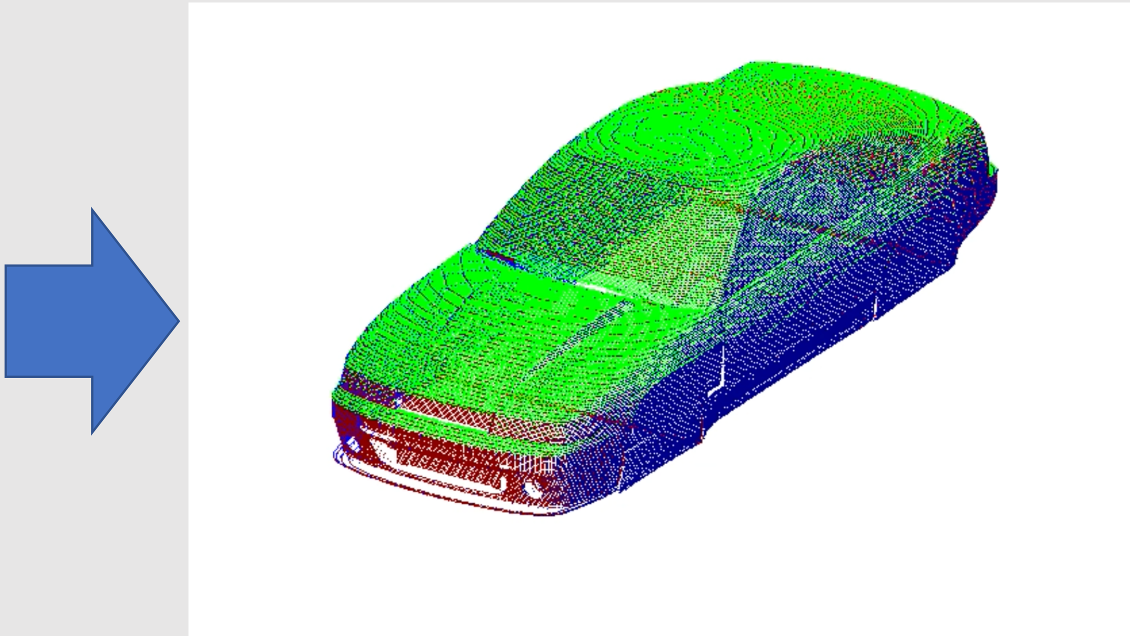


Image Credit: Wojciech Muła @ Wikipedia

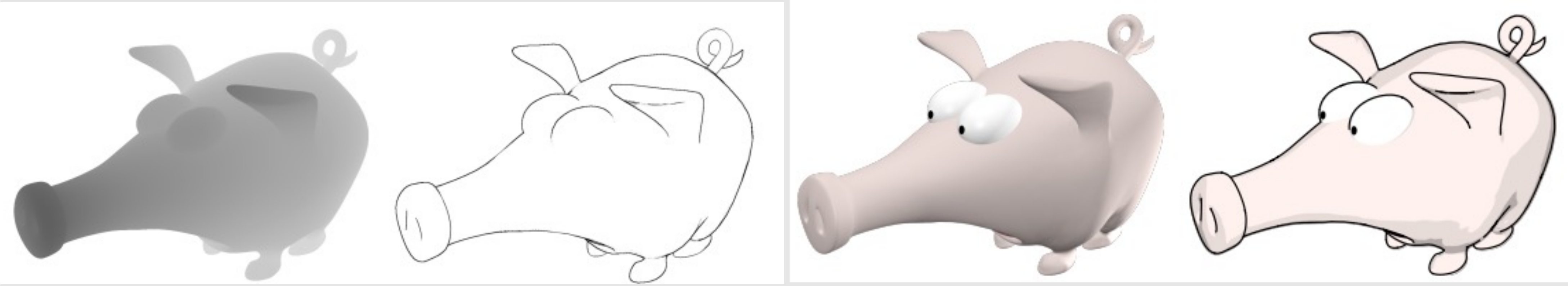
Biproduct of Z-buffer Method: Depth Image

- Depth Image can be used for various geometry processing



Depth Image Usage 1: Contour Drawing

- Non-photo realistic (NPR) rendering

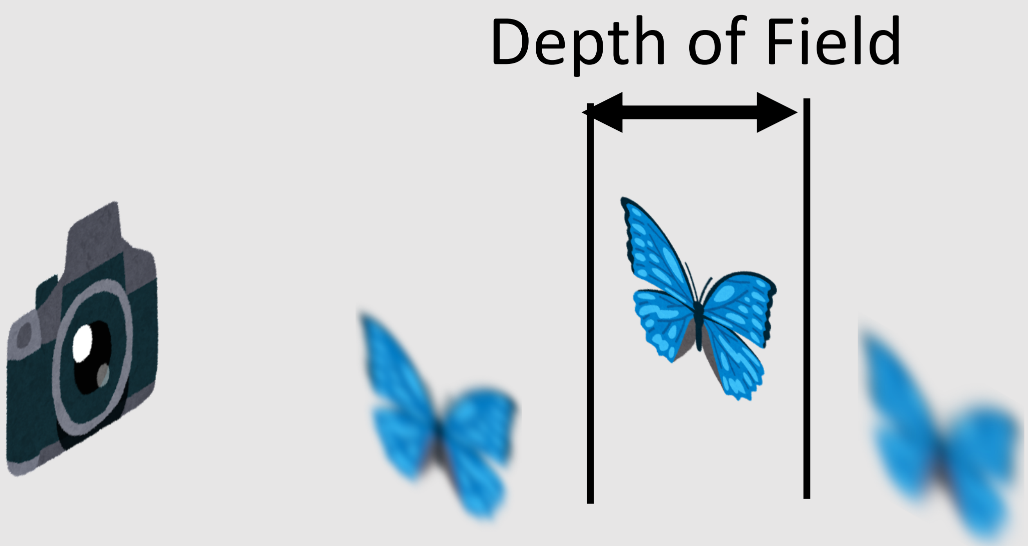


depth image | depth discontinuity | naïve diffuse shading | NPR rendering with contour

[1] Bénard, Pierre, and Aaron Hertzmann. "Line drawings from 3D models: A tutorial." Foundations and Trends® in Computer Graphics and Vision 11, no. 1-2 (2019): 1-159.

Depth Image Usage 2: DoF Effect

- Depth of field (DoF, 被写界深度)



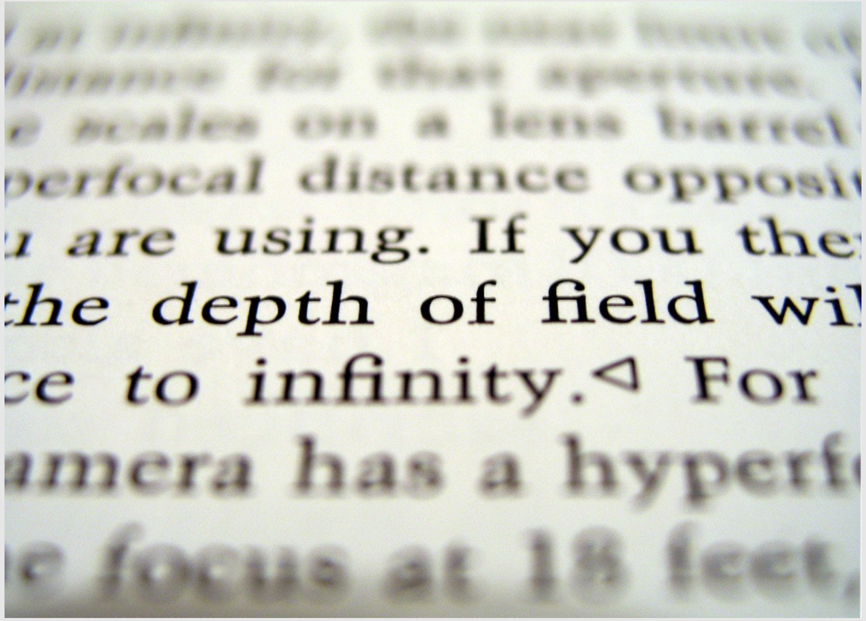


image from wikipedia

- Shallows depth of field → small range of focus, large appature
- Deep depth of field → pan focus, small appature

Depth Image Usage 3: Shadow Mapping

- Rendering image from light to find occlusion of light

Image with shadow

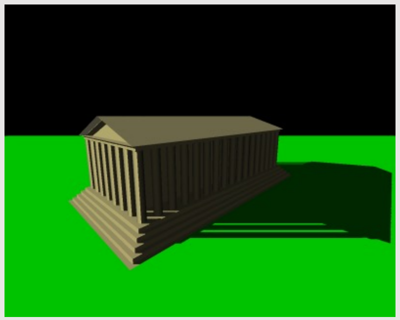
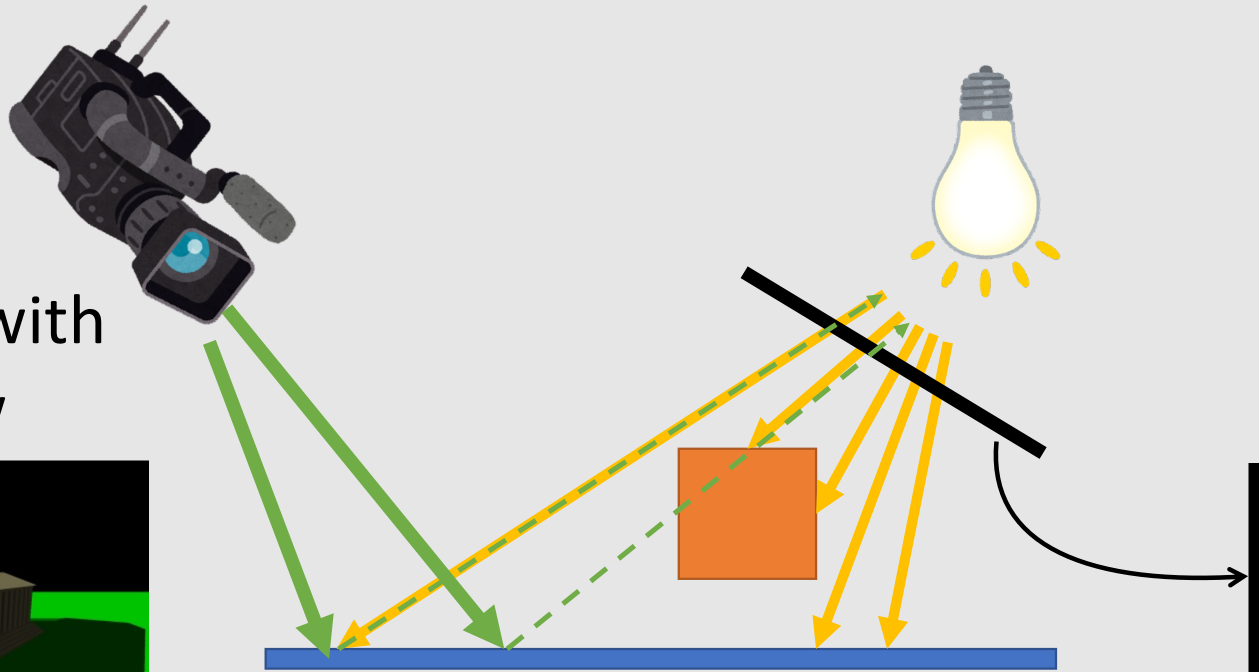


Image credit:
Praetor alpha @ Wikipedia



depth map

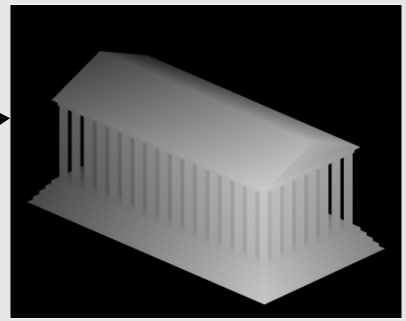
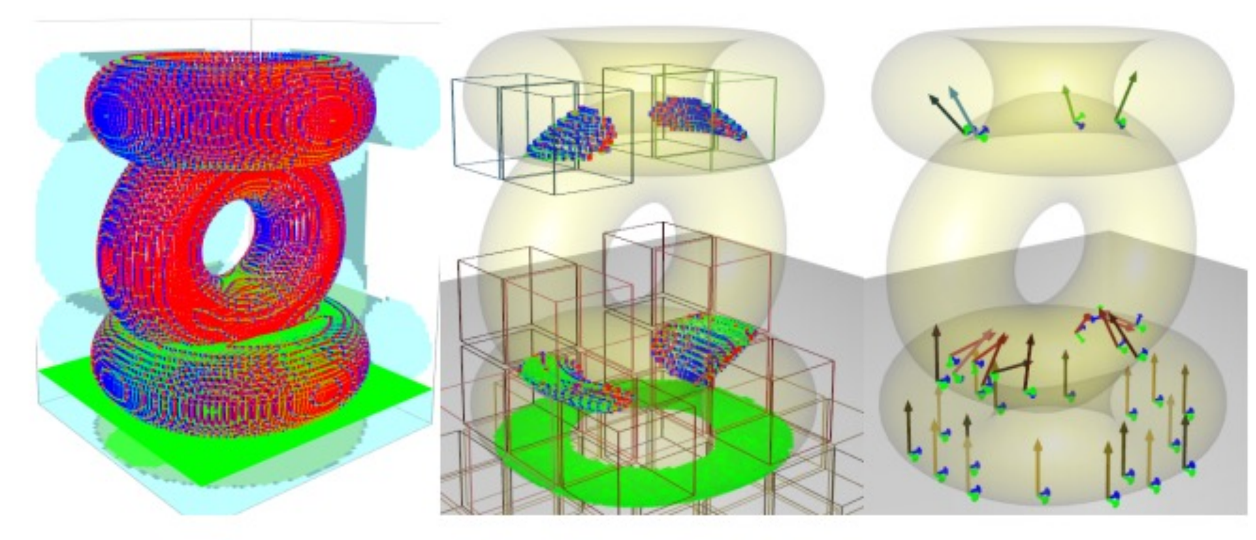
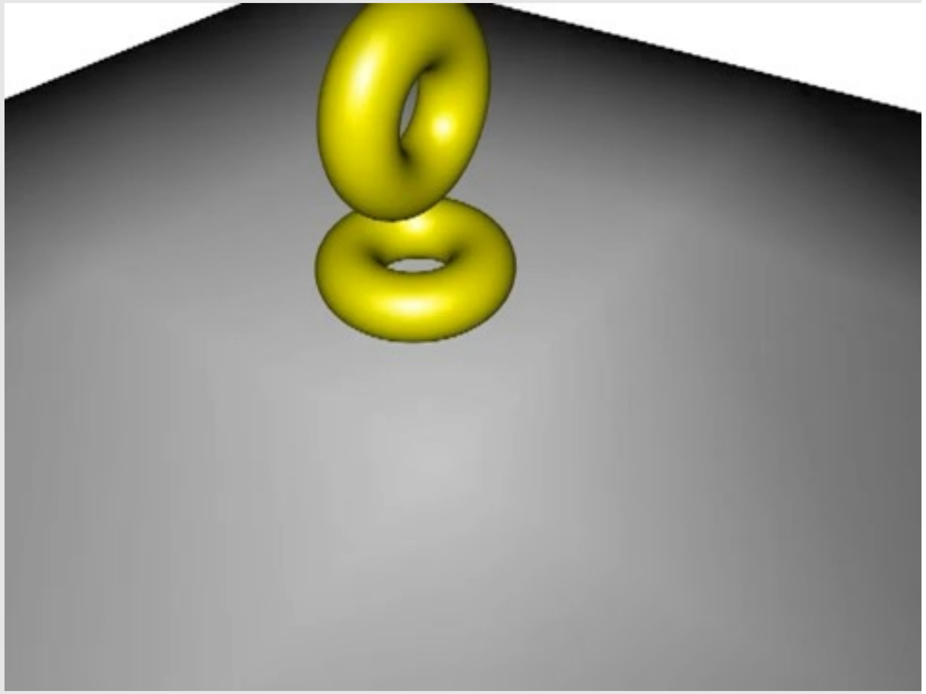


Image credit:
Praetor alpha @ Wikipedia

Depth Image Usage 4: Collision Detection

- Compute volume of intersection and its derivative

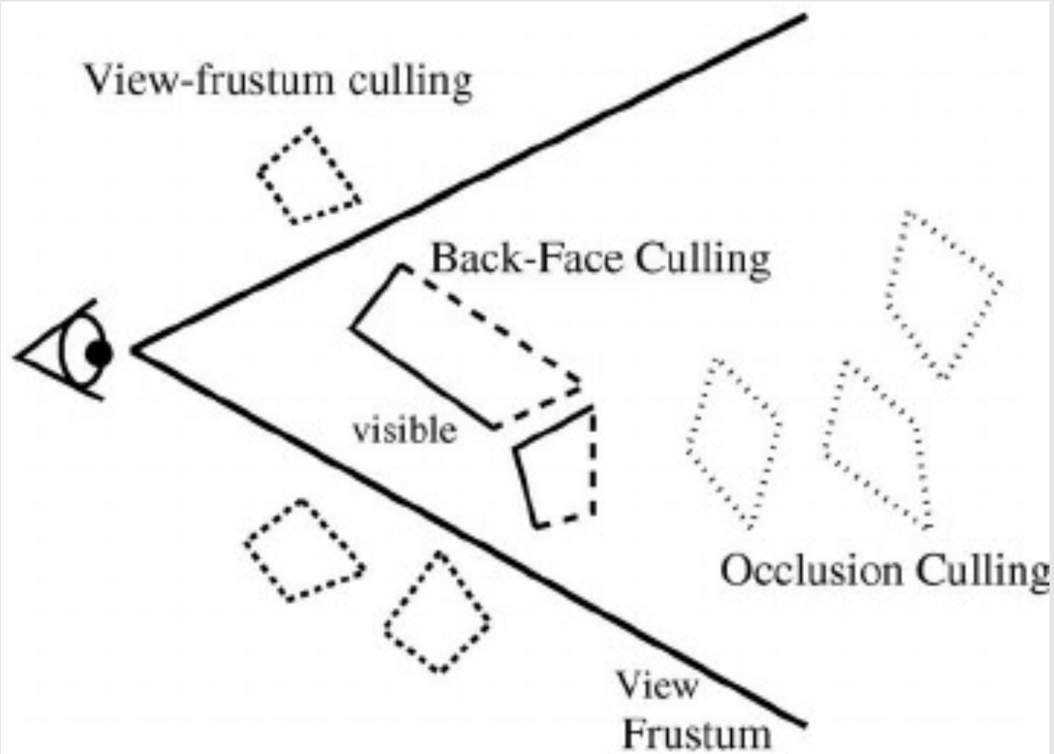




Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. 2010. Volume contact constraints at arbitrary resolution. ACM Trans. Graph. 29, 4, Article 82 (July 2010)

Acceleration Method 1: Culling

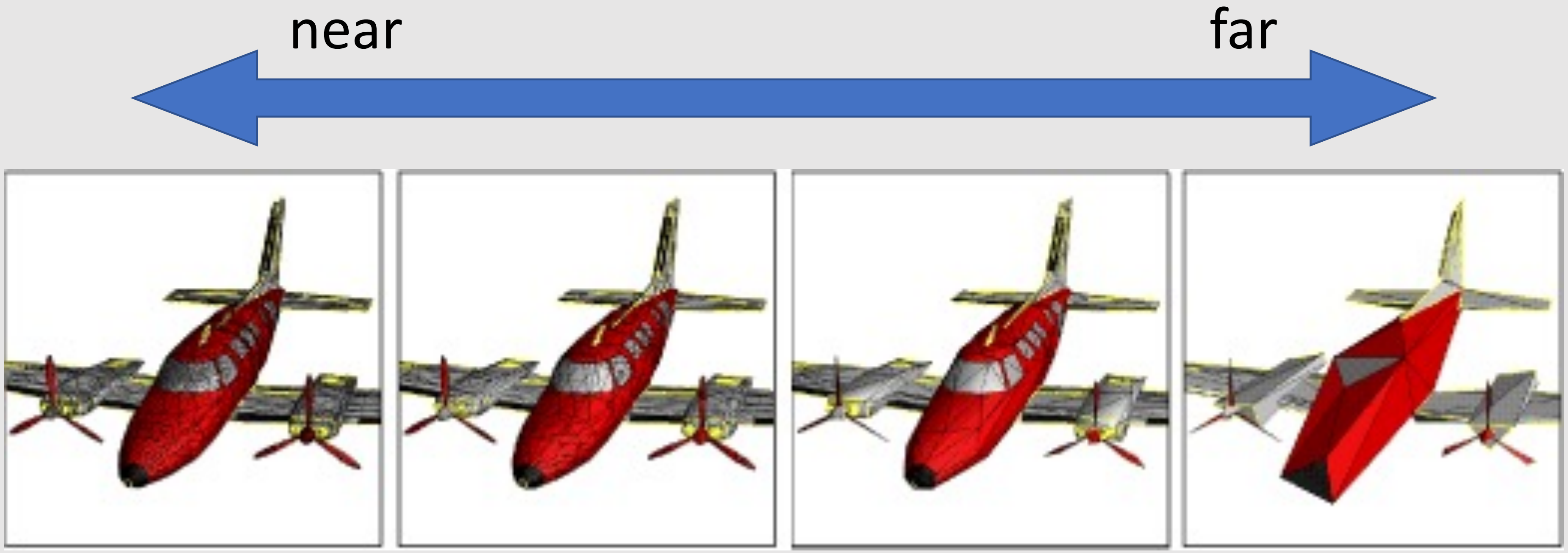
- Reduce number of triangle rasterized



Cohen-Or, Daniel & Chrysanthou, Yiorgos & Silva, Cláudio. (2001). A Survey of Visibility for Walkthrough Applications. Proceedings of SIGGRAPH.

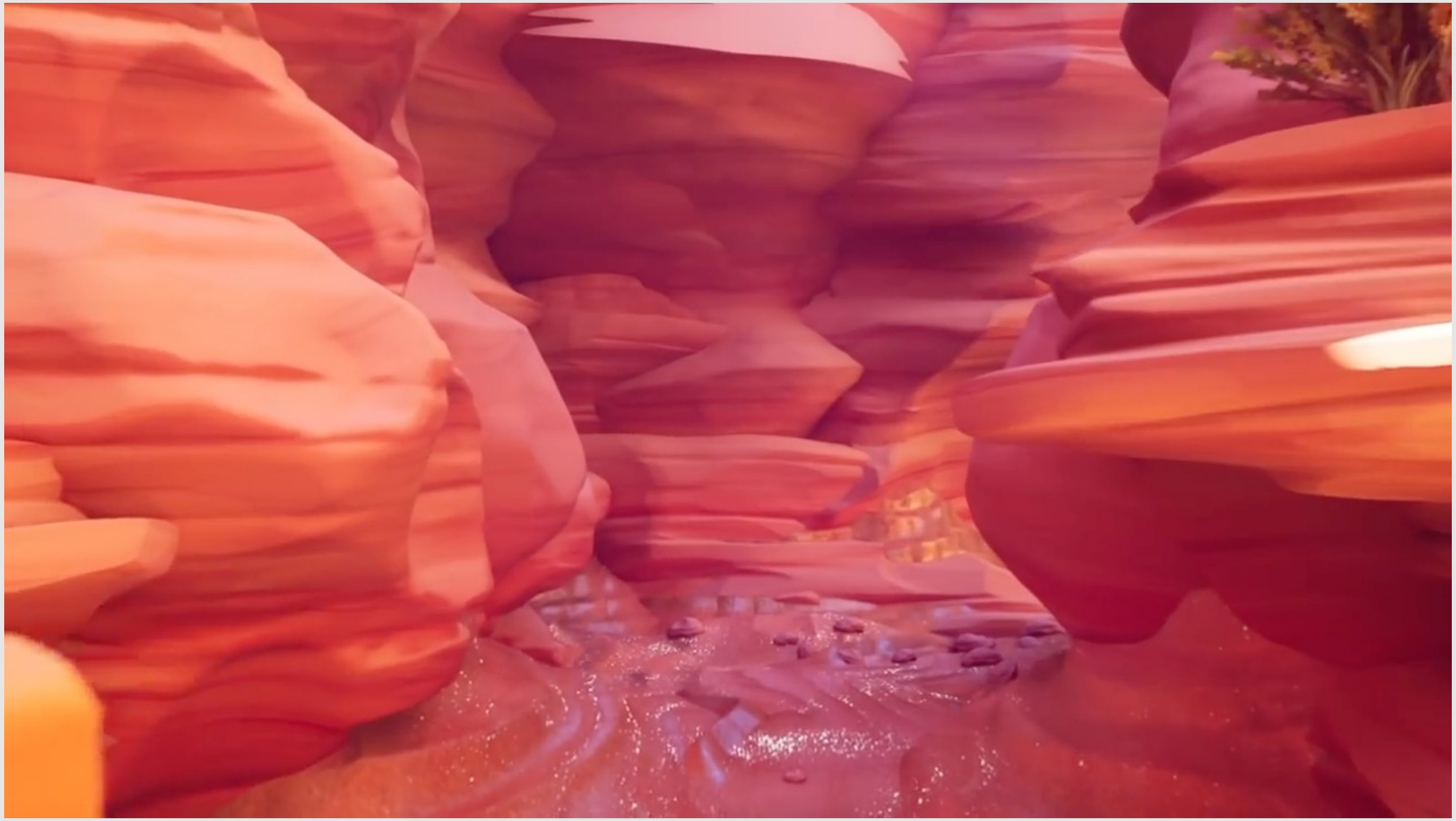
Acceleration Method 2: Level of Detail (LoD)

- Dynamically change the resolution of mesh



Hoppe, H. Progressive meshes. In Computer Graphics (SIGGRAPH'96 Proceedings).

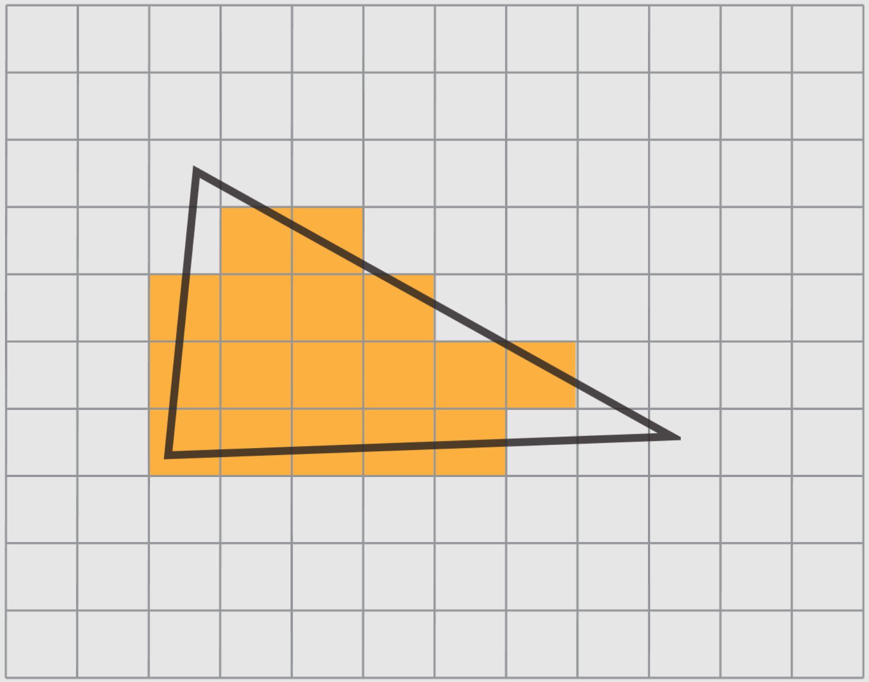
Nanaite in Unreal Engine 5

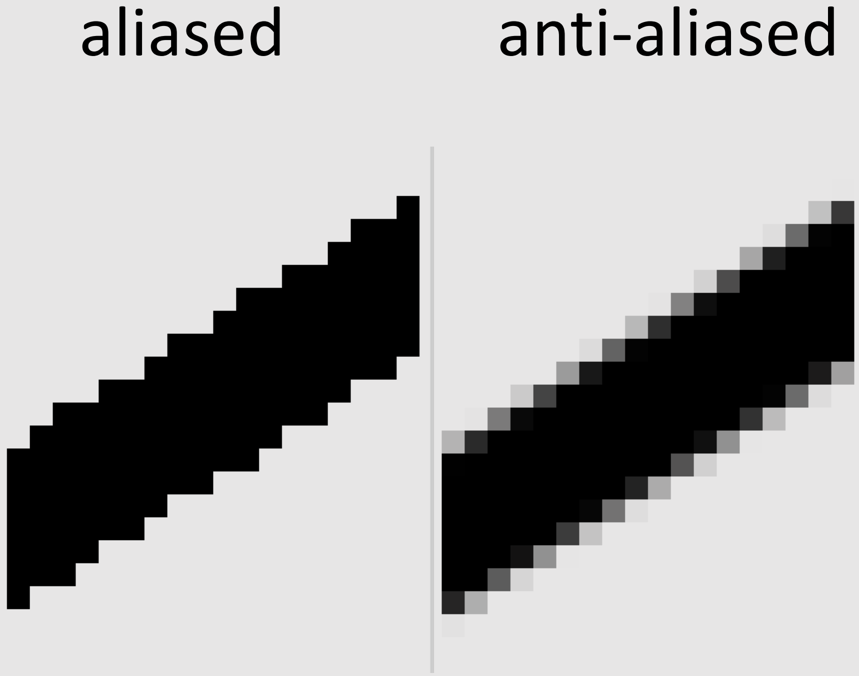


Nanite in UE5: The End of Polycounts? | Unreal Engine
https://www.youtube.com/watch?v=xUUSsXswyZM

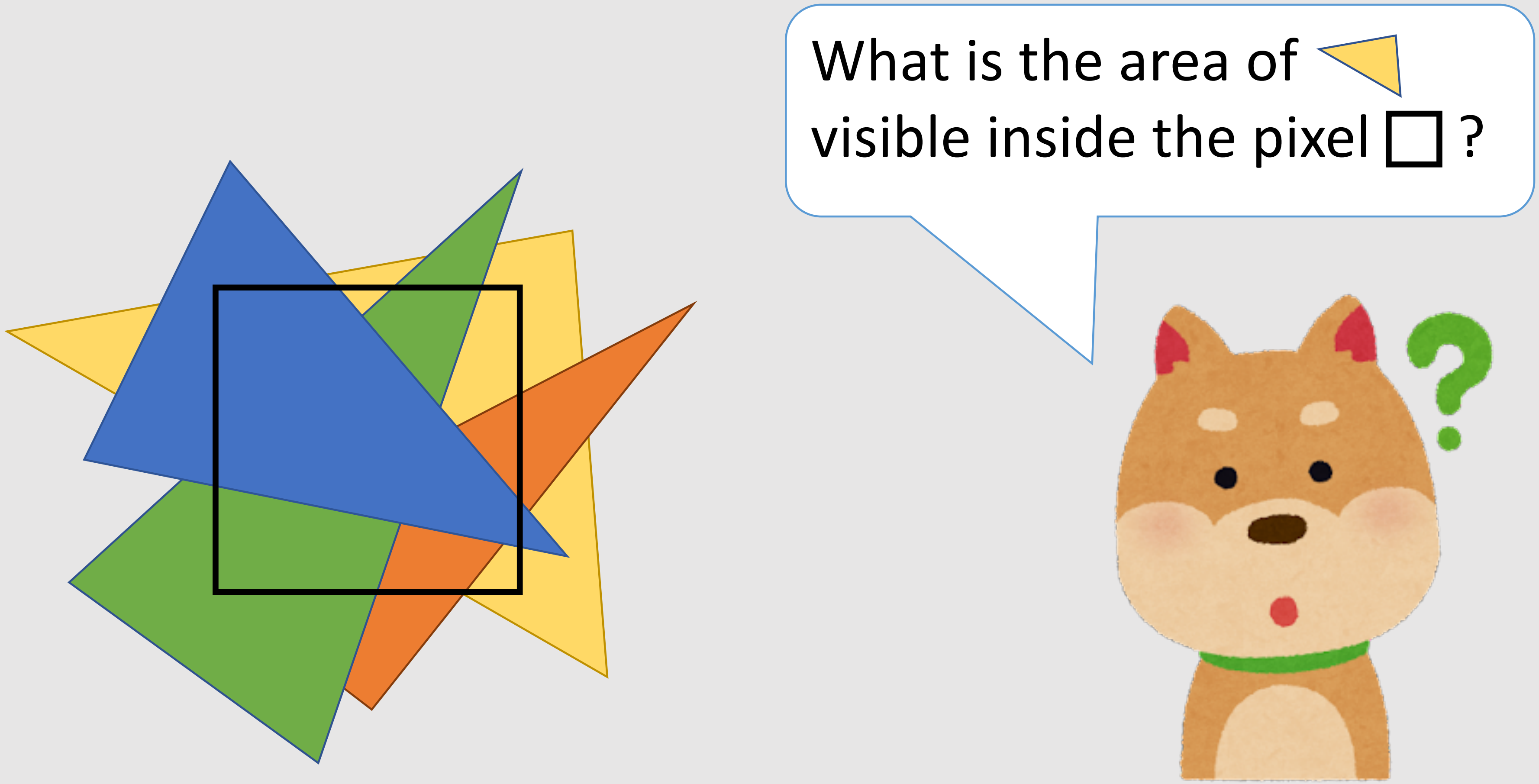
Sub-pixel Effects

Removing Jaggy Edge: Anti-Aliasing





How to Compute the “Coverage Ratio”?



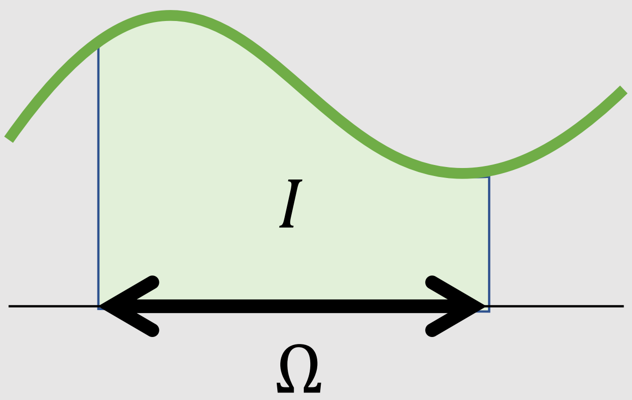
Monte Carlo Integration

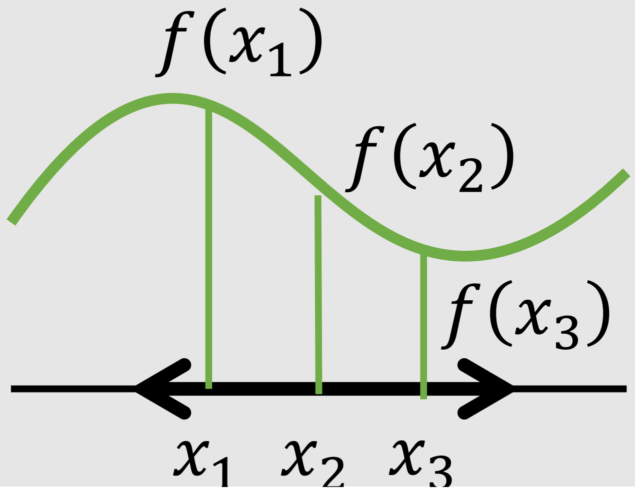
- Integration of a "difficult" function (i.e., we can only evaluate at discrete sample locations)

$$I = \int_{\Omega} f(x)dx$$

approximation

$$\frac{V}{N}\sum_{i=1}^{N} f(x_i) \qquad V = \int_{\Omega} dx \qquad x_1, \ldots, x_N \in \Omega$$

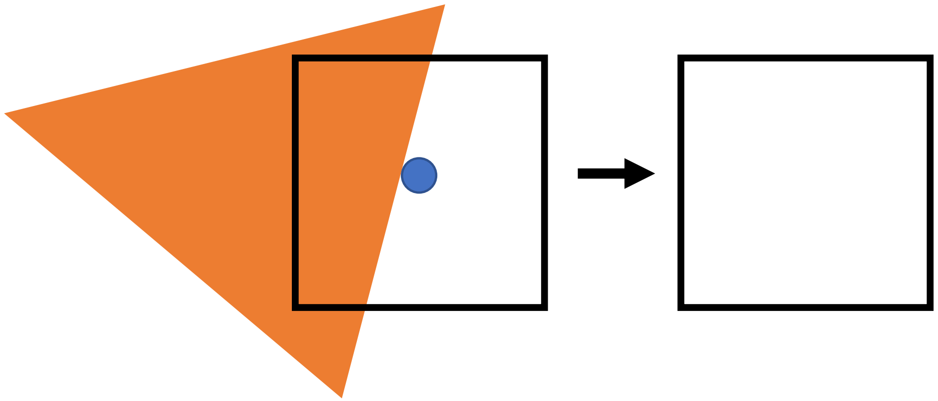




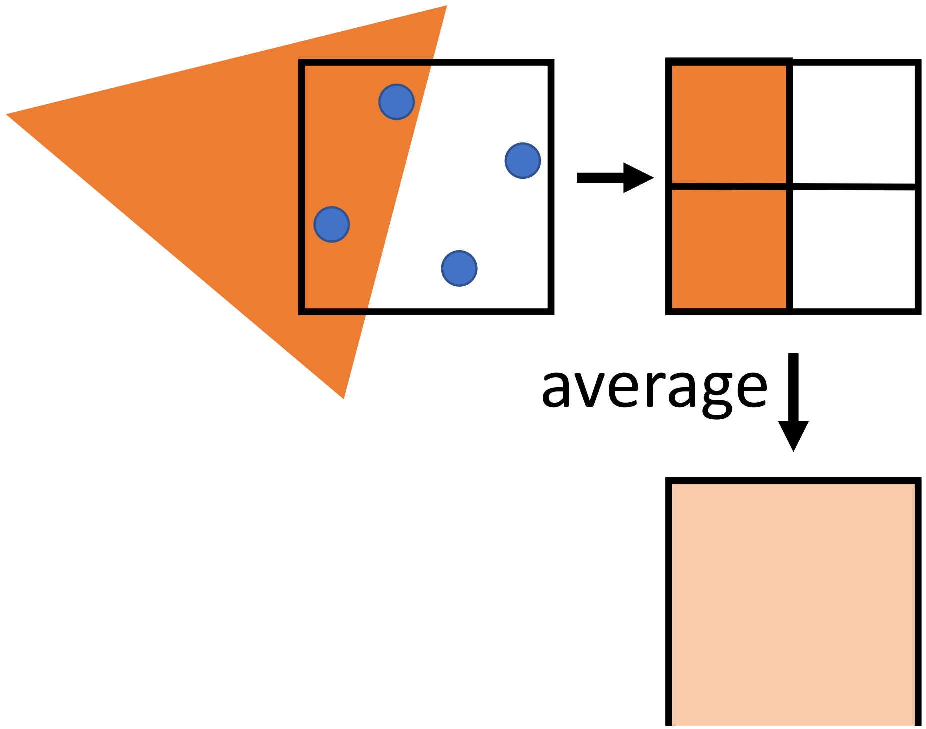
Basic Approach: Multiple Samples in a Pixel

- Finding coverage ratio approximately

Sample at center

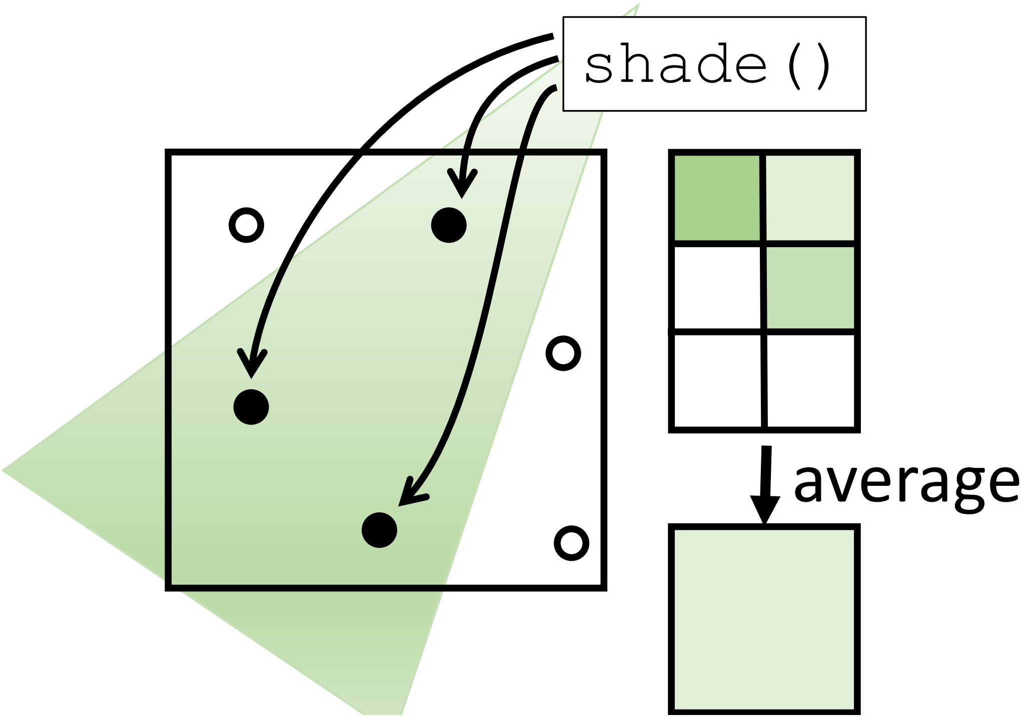


Multiple samples

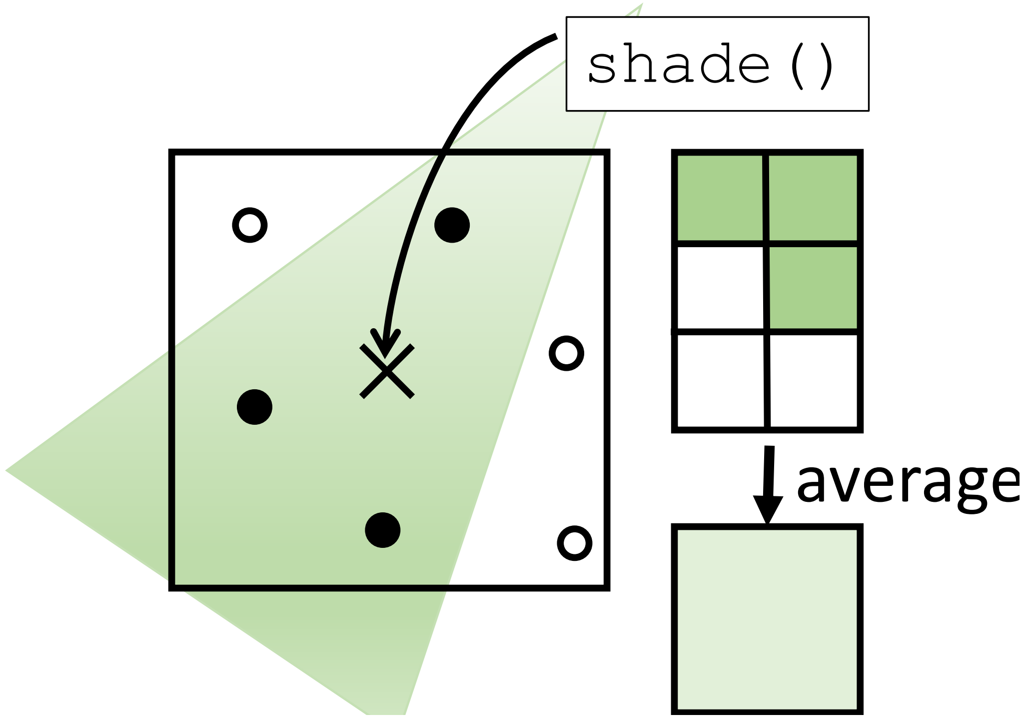


SuperSampling vs. MultiSampling

SuperSampling
fragment shader
for ***all*** samples



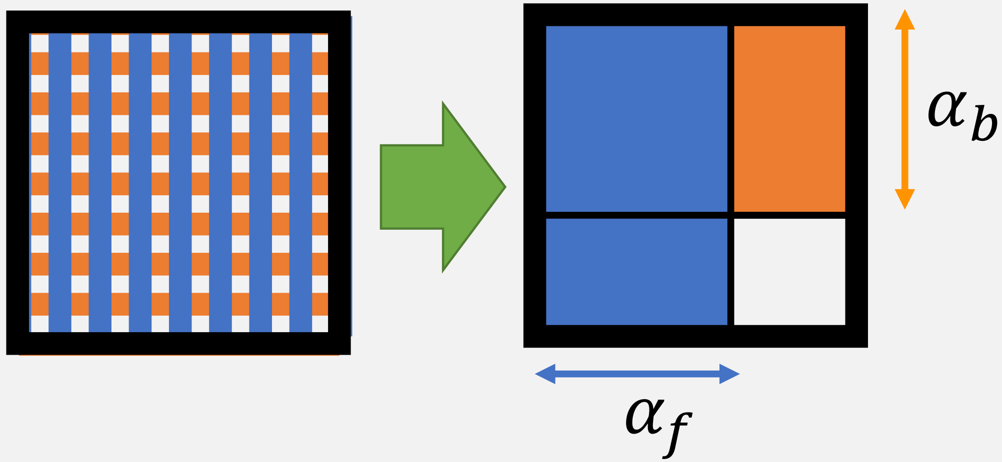
MultiSampling
fragment shader
for ***one*** sample



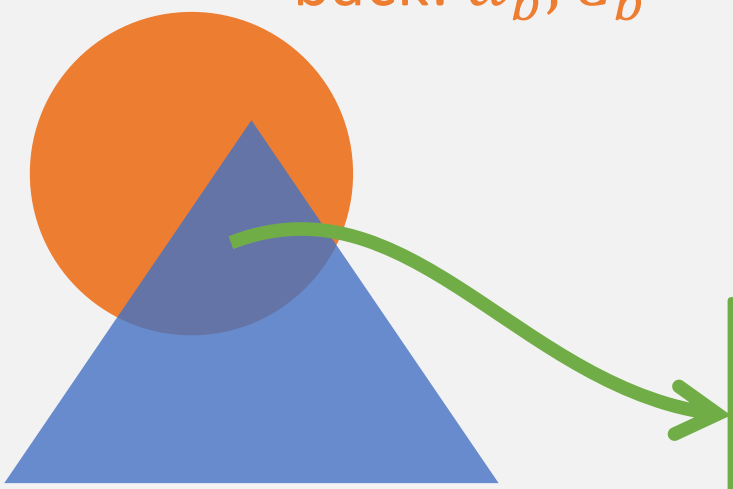
Transparency is Order Dependent

- Alpha value
  - 0 → completely transparent
  - 1 → opaque

inside one pixel



back: α_b, C_b



front: α_f, C_f

$$\alpha = \alpha_f + \alpha_b(1 - \alpha_f)$$
$$C\alpha = C_f\alpha_f + C_b\alpha_b(1 - \alpha_f)$$

Not symetric!

Painter's algorithm

- Sort geometry w.r.t. dpeth
- Draw from background

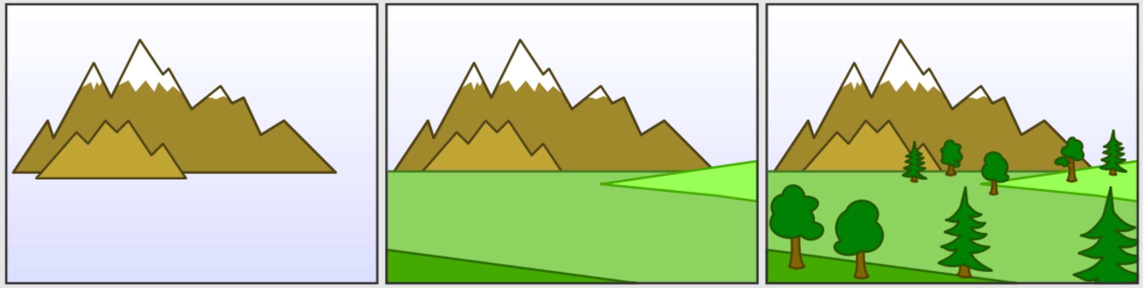


Image Credit: Zapyon @ Wikipedia

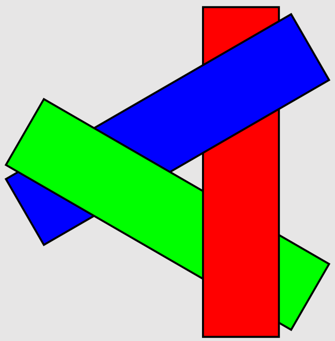


Image Credit: Wojciech Muła @ Wikipedia

- ☹ Cannot draw in parallel
- ☹ Cannot handle cyclical overlapping

Depth Peeling Technique [Evenritt et al]

- Use two depth buffer to render object fron to back

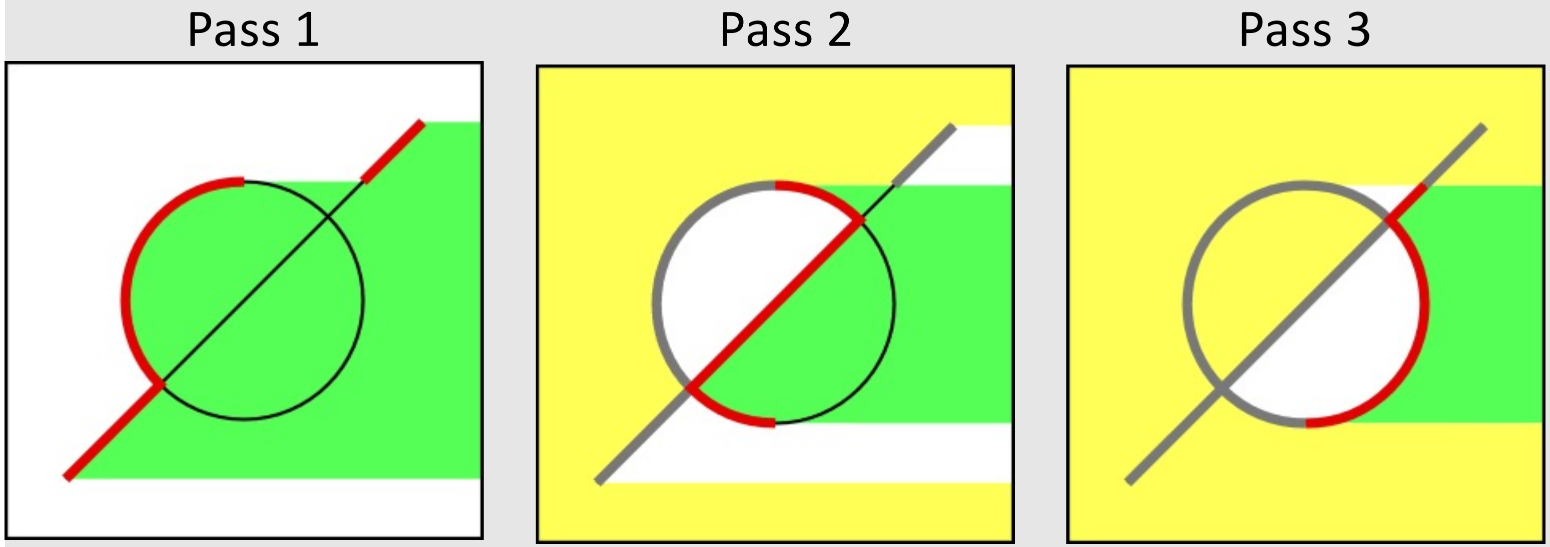


Image credit: 床井研究室，２００８年１１月２３日 Depth Pealing
https://marina.sys.wakayama-u.ac.jp/~tokoi/?date=20081123

[Evenritt et al]

Everitt, Cass (2001-05-15). "Interactive Order-Independent Transparency" (PDF). Nvidia