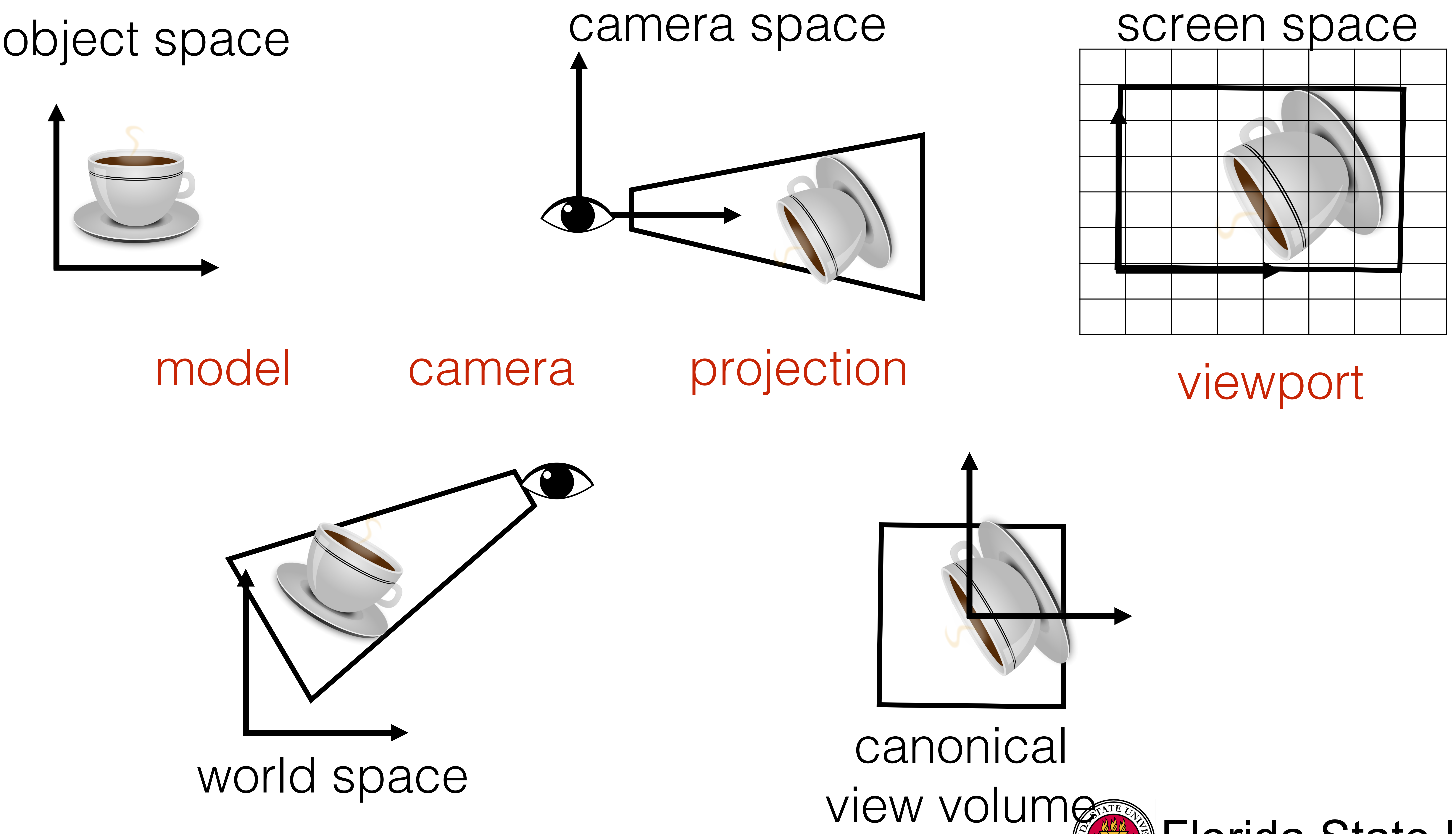
Rasterization

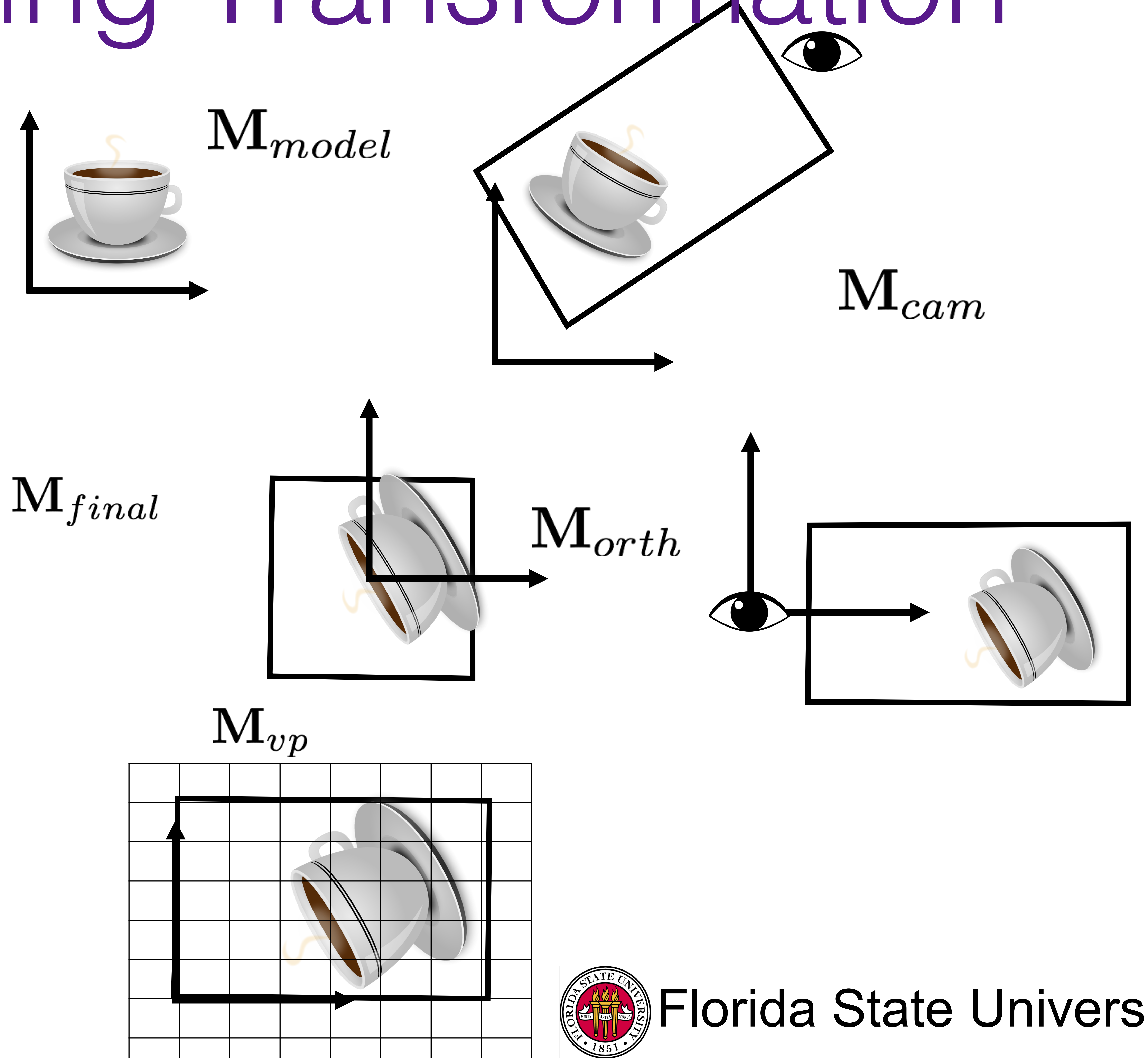
Acknowledgements: Daniele Panozzo

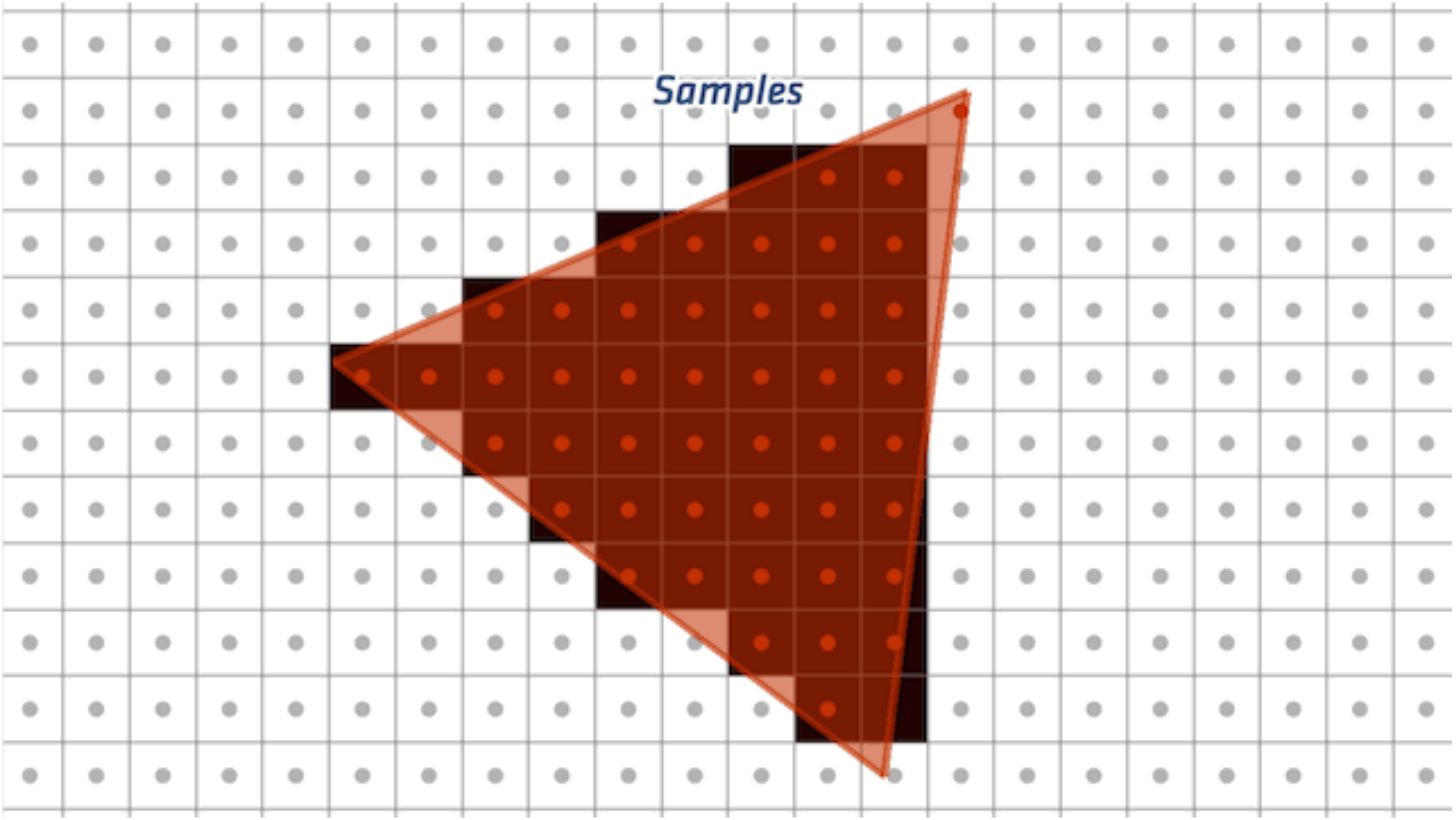
Recap: Viewing Transformation



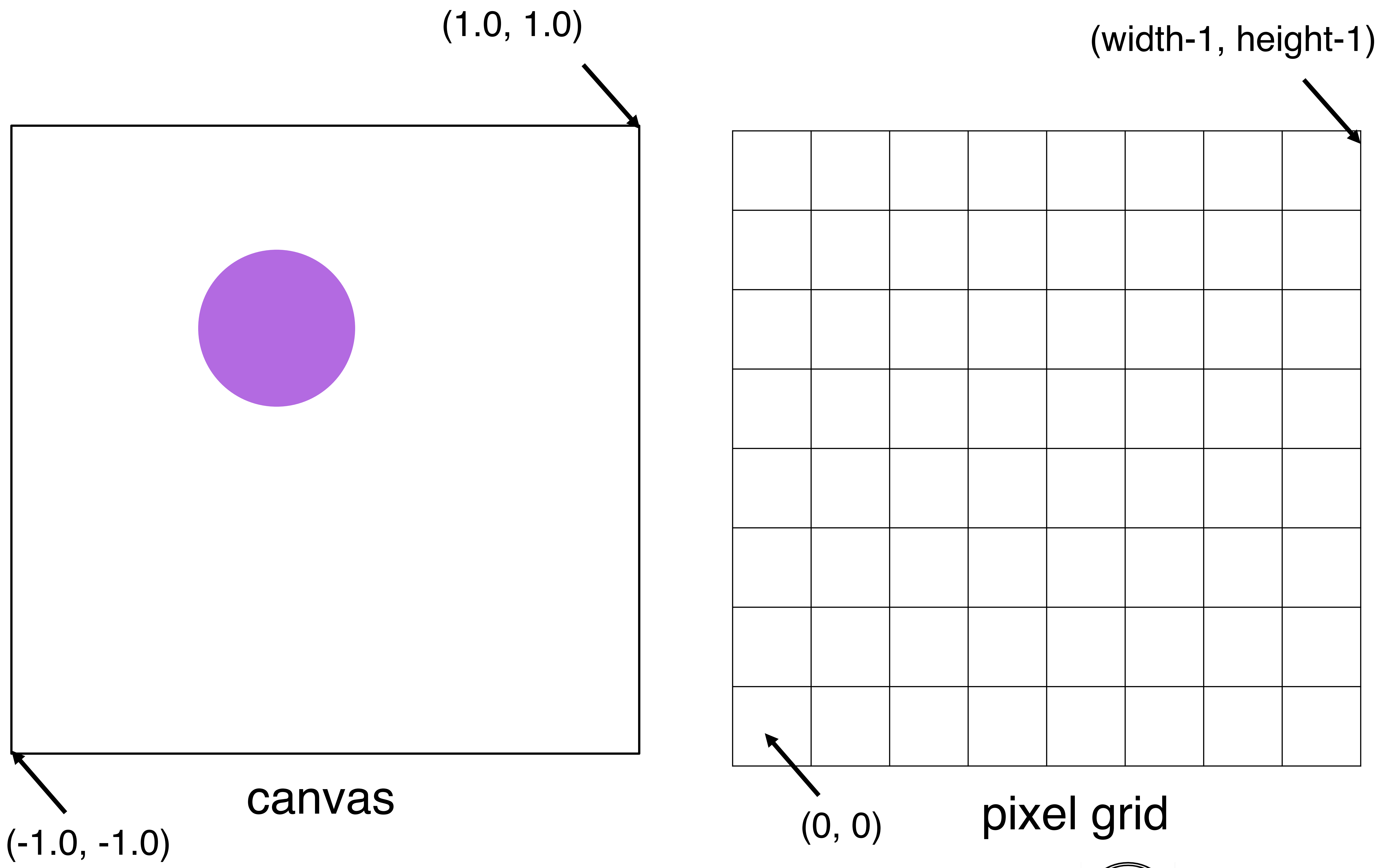
Recap: Viewing Transformation

- Construct Viewport Matrix $\mathbf{M}_{vp}$
- Construct Projection Matrix $\mathbf{M}_{orth}$
- Construct Camera Matrix $\mathbf{M}_{cam}$
- $\mathbf{M} = \mathbf{M}_{vp}\mathbf{M}_{orth}\mathbf{M}_{cam}$
- For each model
  - Construct Model Matrix $\mathbf{M}_{model}$
  - $\mathbf{M}_{final} = \mathbf{M}\mathbf{M}_{model}$
  - For every point **p** in each primitive of the model
    - $\mathbf{p}_{final} = \mathbf{M}_{final}\mathbf{p}$
  - Rasterize the model

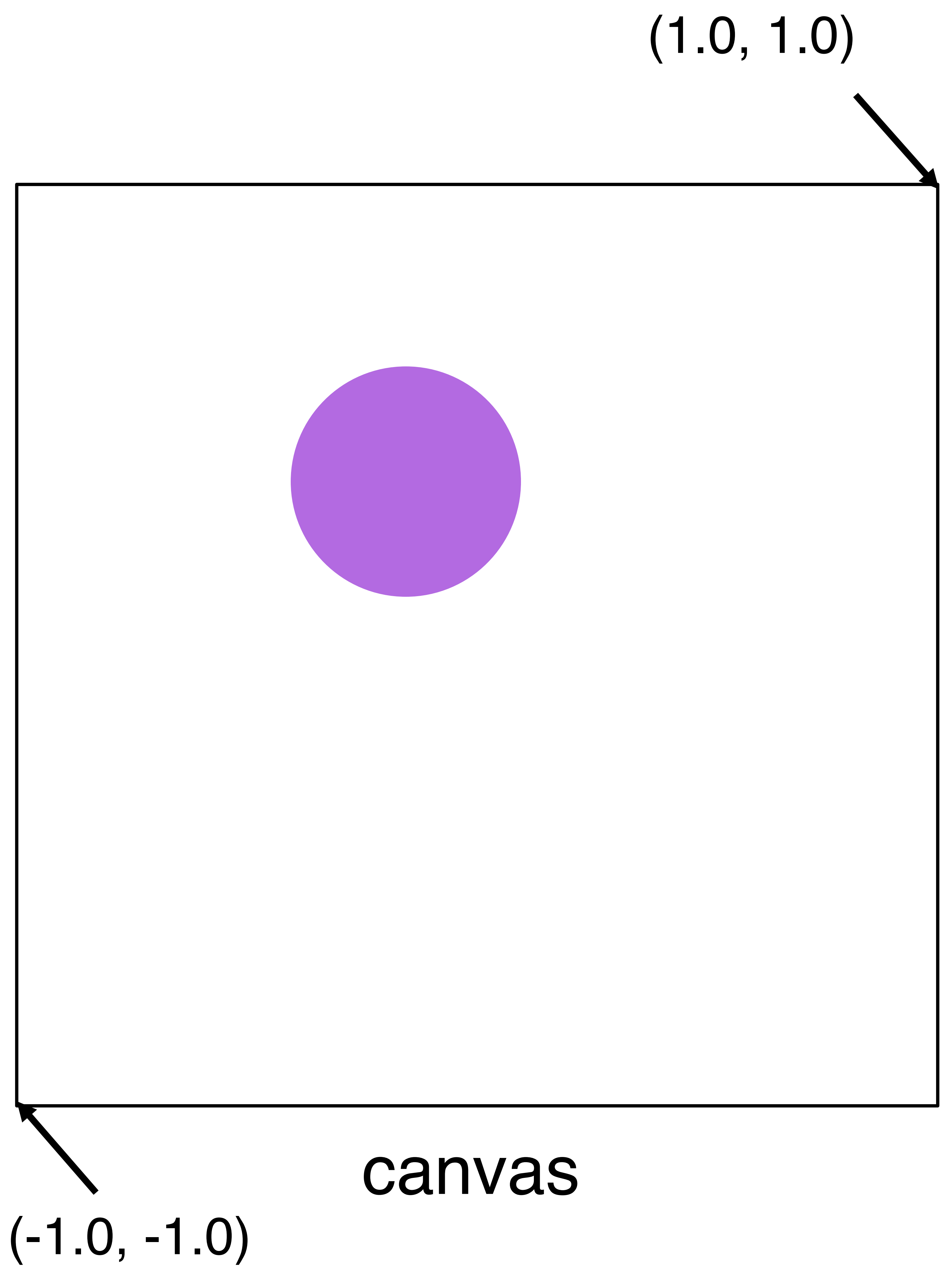


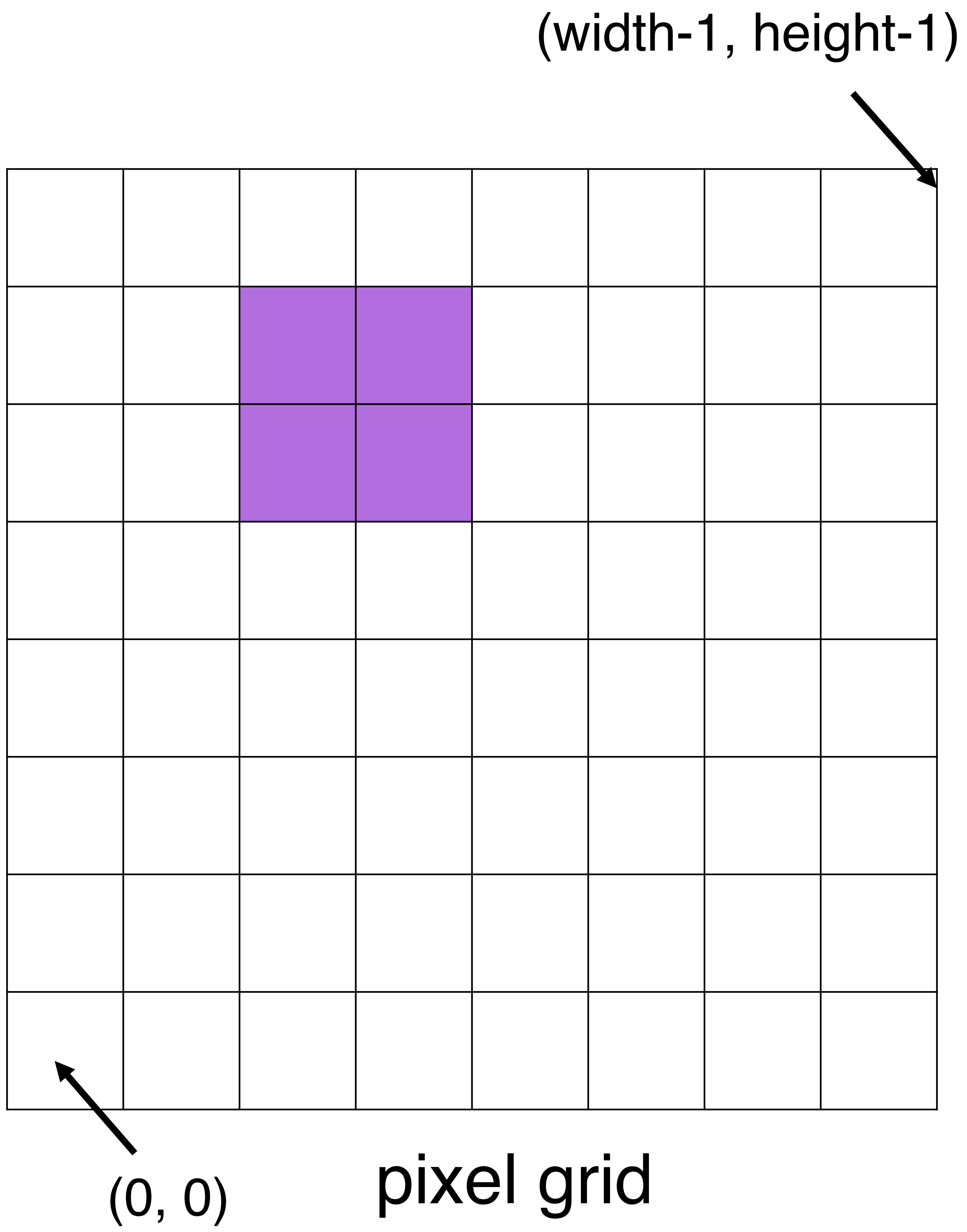

Samples

2D Canvas



2D Canvas





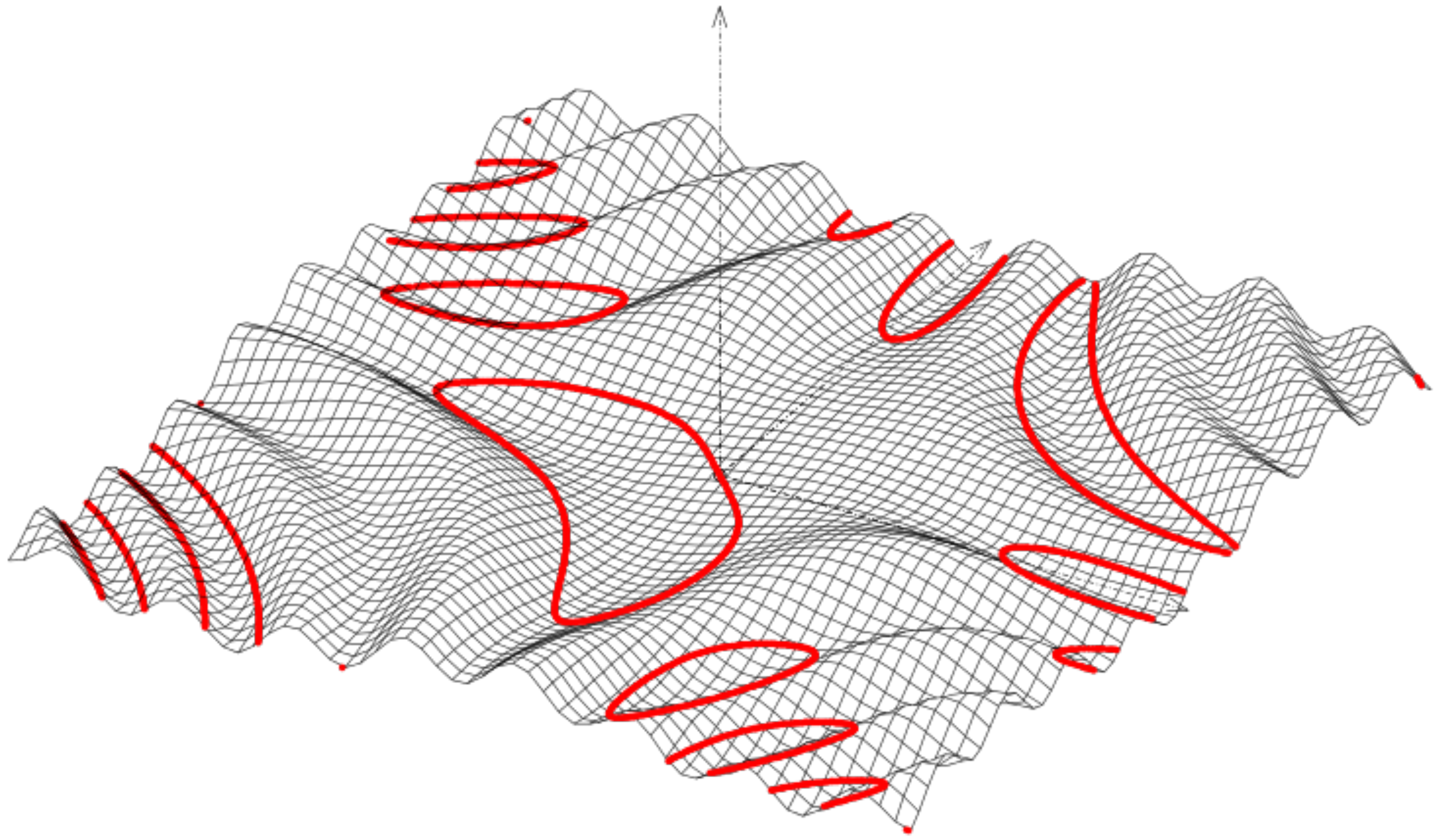
Implicit Geometry Representation

- Define a curve as zero set of 2D implicit function
  - $F(x,y) = 0 \rightarrow$ on curve
  - $F(x,y) < 0 \rightarrow$ inside curve
  - $F(x,y) > 0 \rightarrow$ outside curve
- Example: Circle with center (c_x, c_y) and radius r

$$F(x, y) = (x - c_x)^2 + (y - c_y)^2 - r^2$$

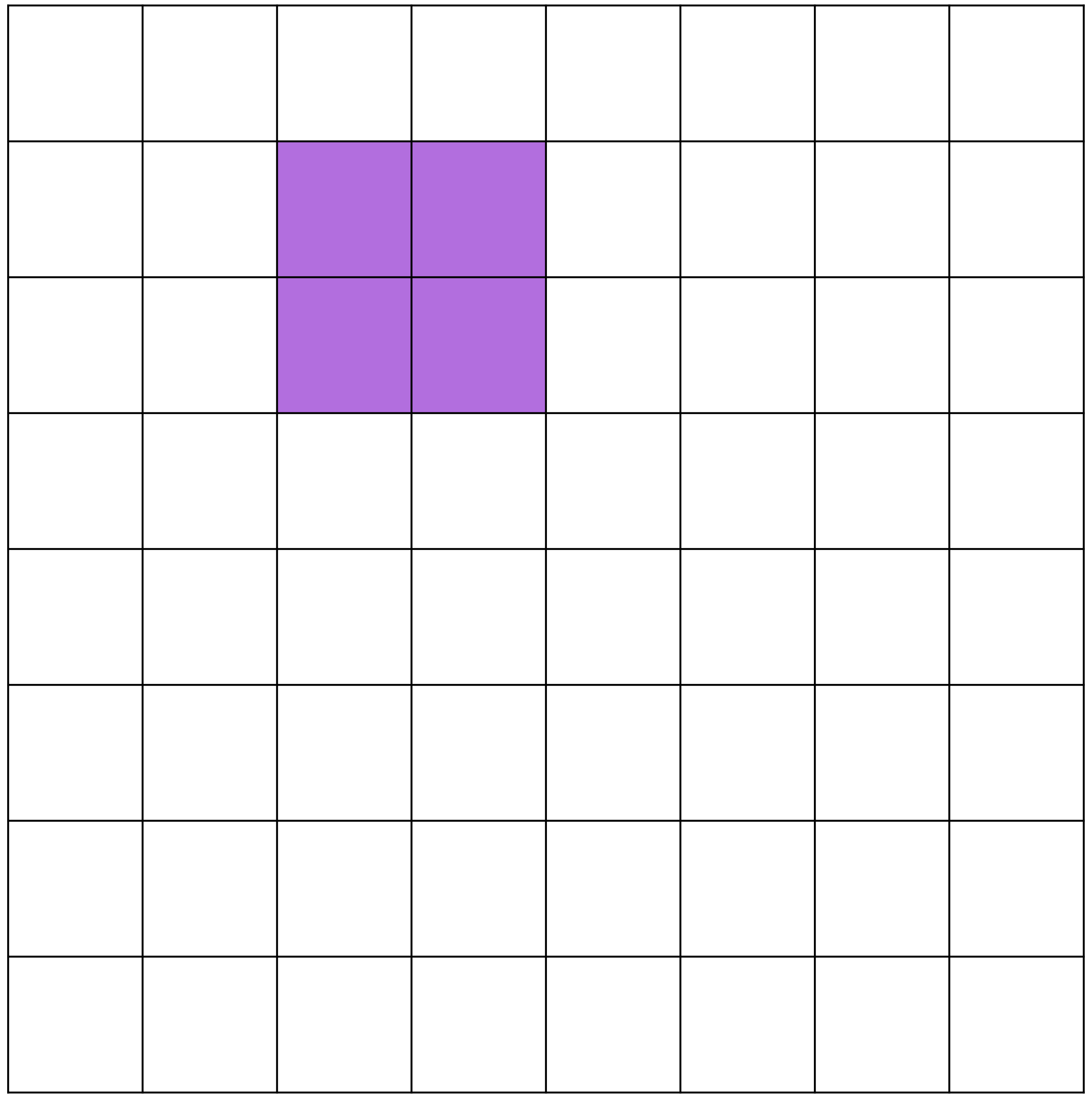
Implicit Geometry Representation

- Define a curve as zero set of 2D implicit function
  - $F(x,y) = 0 \rightarrow$ on curve
  - $F(x,y) < 0 \rightarrow$ inside curve
  - $F(x,y) > 0 \rightarrow$ outside curve



Implicit Rasterization

```
for all pixels (i,j)
        (x,y) = map_to_canvas (i,j)
    if F(x,y) < 0
        set_pixel (i,j, color)
```



Implicit Geometry Representation

- Example: Triangle

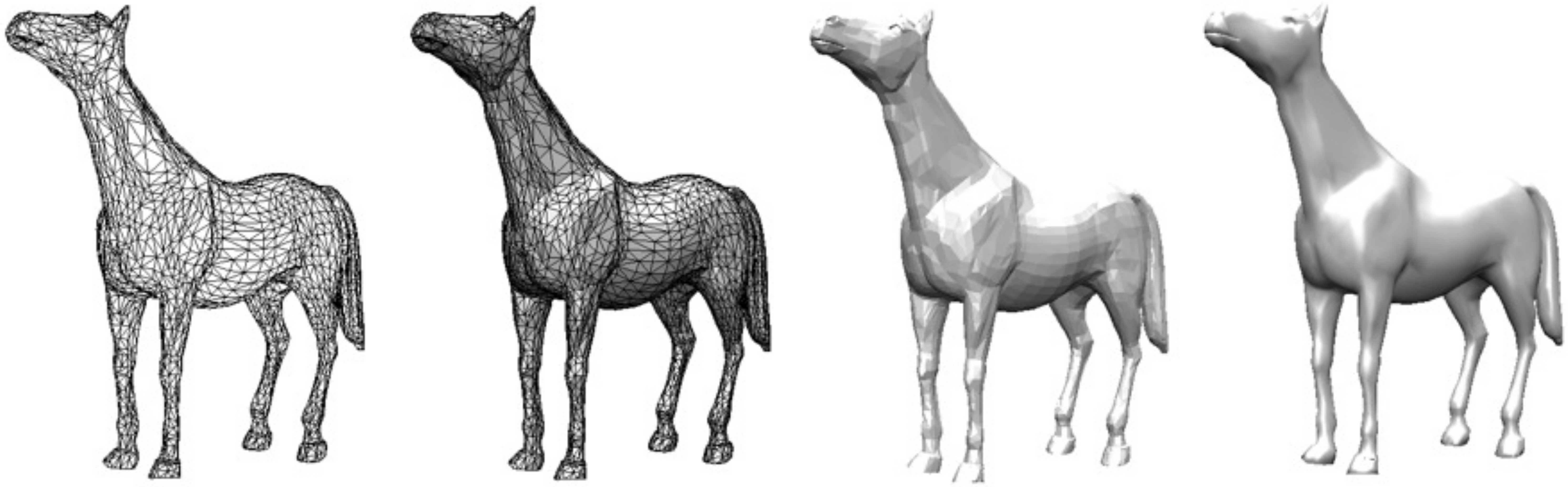
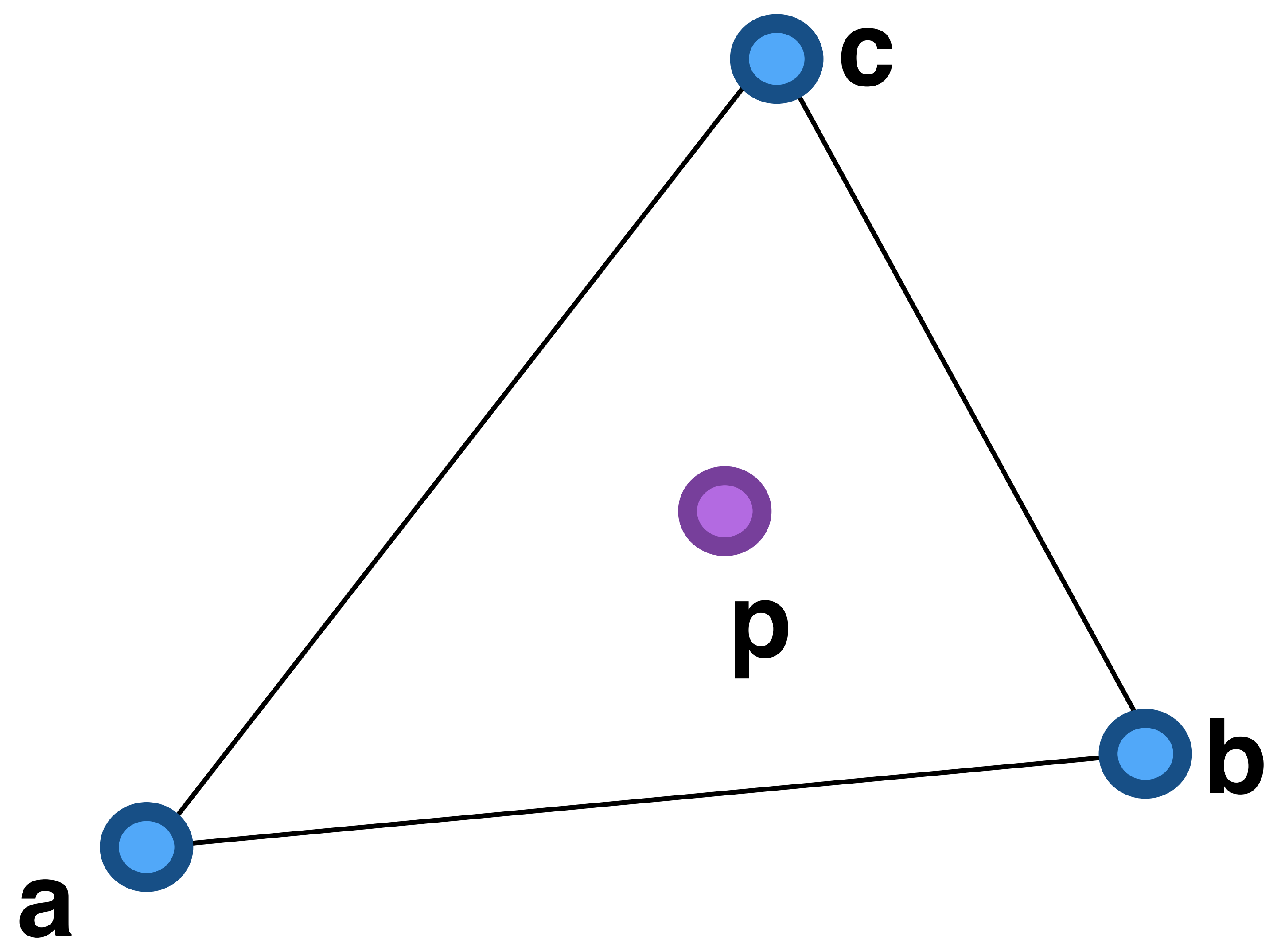


Image Copyright: Andrea Tagliasacchi

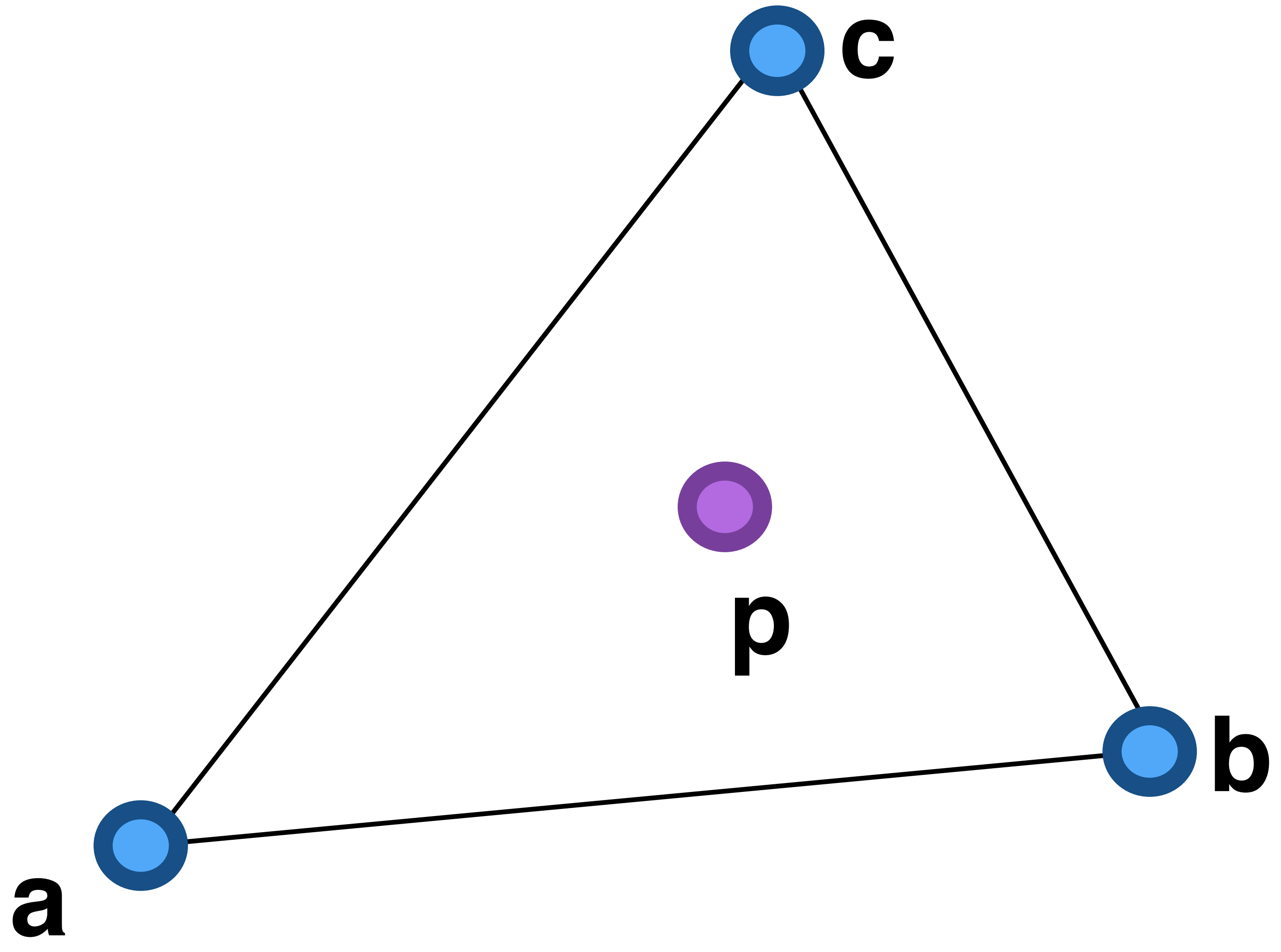
Barycentric Interpolation

- Barycentric coordinates:
  - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$



Barycentric Interpolation

- Barycentric coordinates:
  - **p** = α**a** + β**b** + γ**c**   with α + β + γ = 1
  - Unique for non-collinear **a**,**b**,**c**



$$\begin{bmatrix} \mathbf{a}_x & \mathbf{b}_x & \mathbf{c}_x \\ \mathbf{a}_y & \mathbf{b}_y & \mathbf{c}_y \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ 1 \end{bmatrix}$$

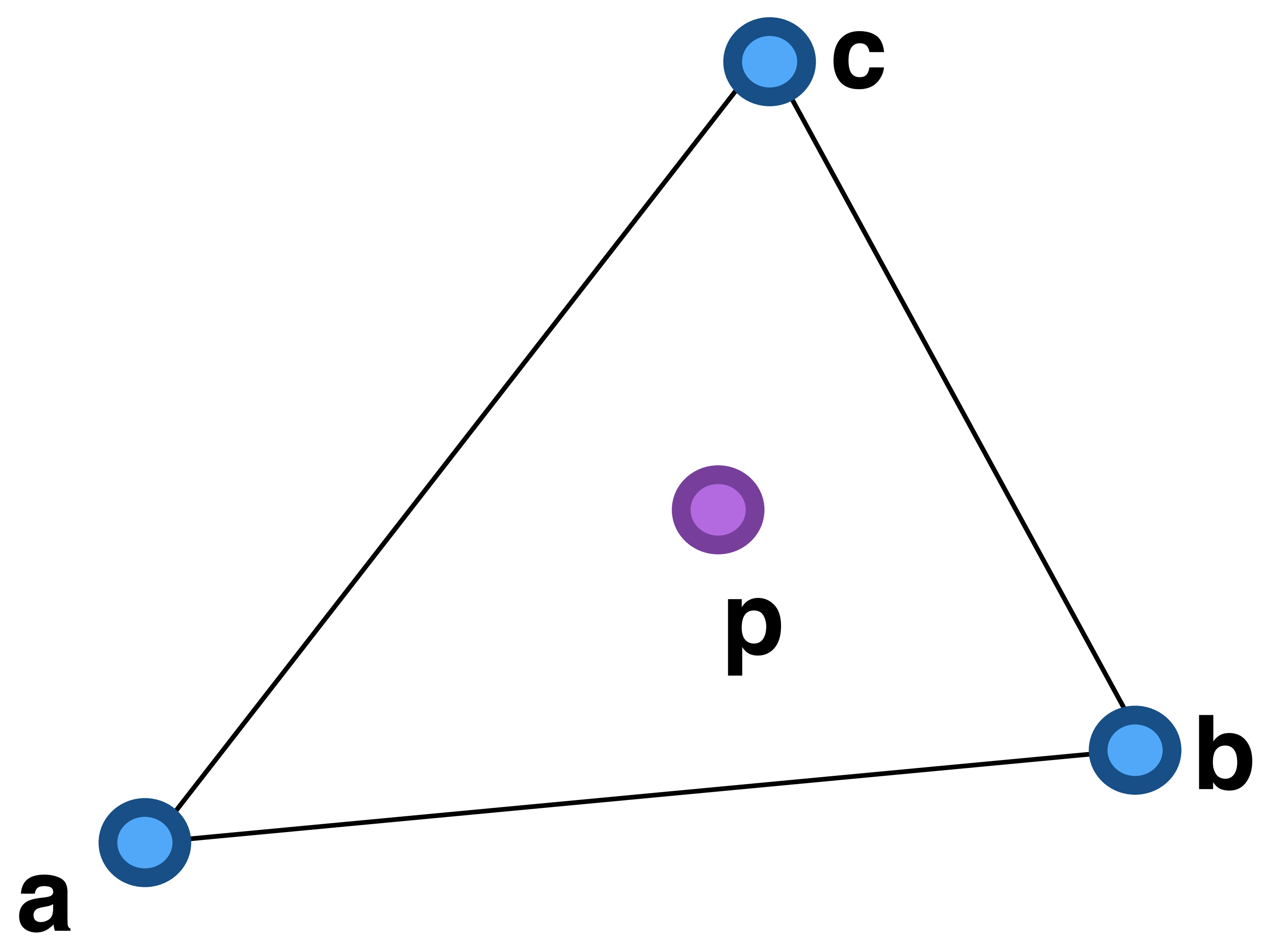
Barycentric Interpolation

- Barycentric coordinates:
  - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-collinear **a**,**b**,**c**
  - Ratio of triangle areas

$$\alpha(\mathbf{p}) = \frac{\text{area}(\mathbf{p}, \mathbf{b}, \mathbf{c})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$

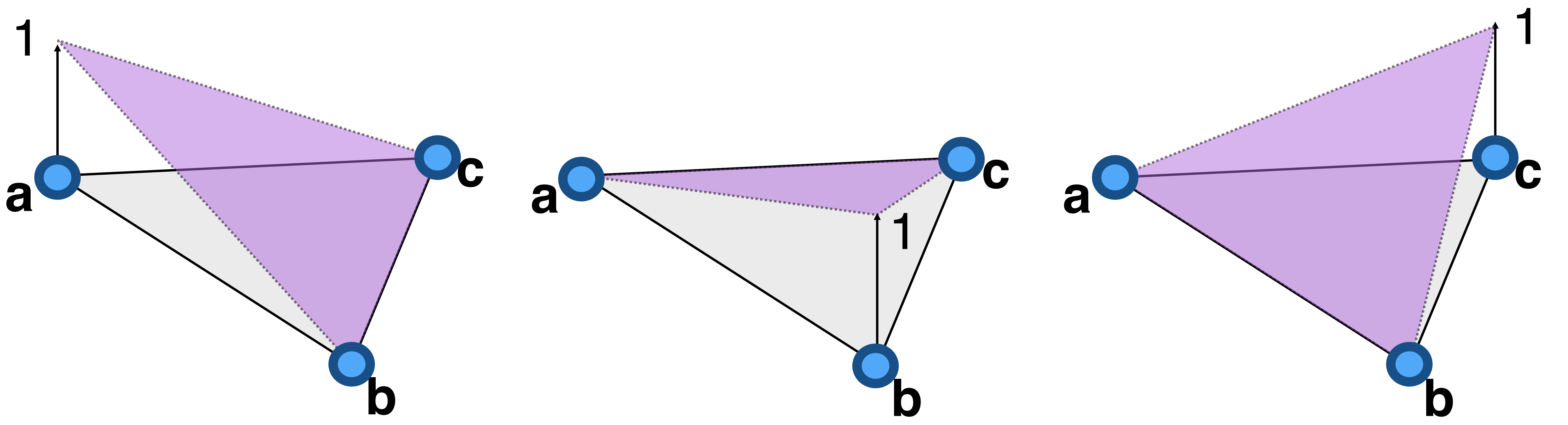
$$\beta(\mathbf{p}) = \frac{\text{area}(\mathbf{p}, \mathbf{c}, \mathbf{a})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$

$$\gamma(\mathbf{p}) = \frac{\text{area}(\mathbf{p}, \mathbf{a}, \mathbf{b})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$



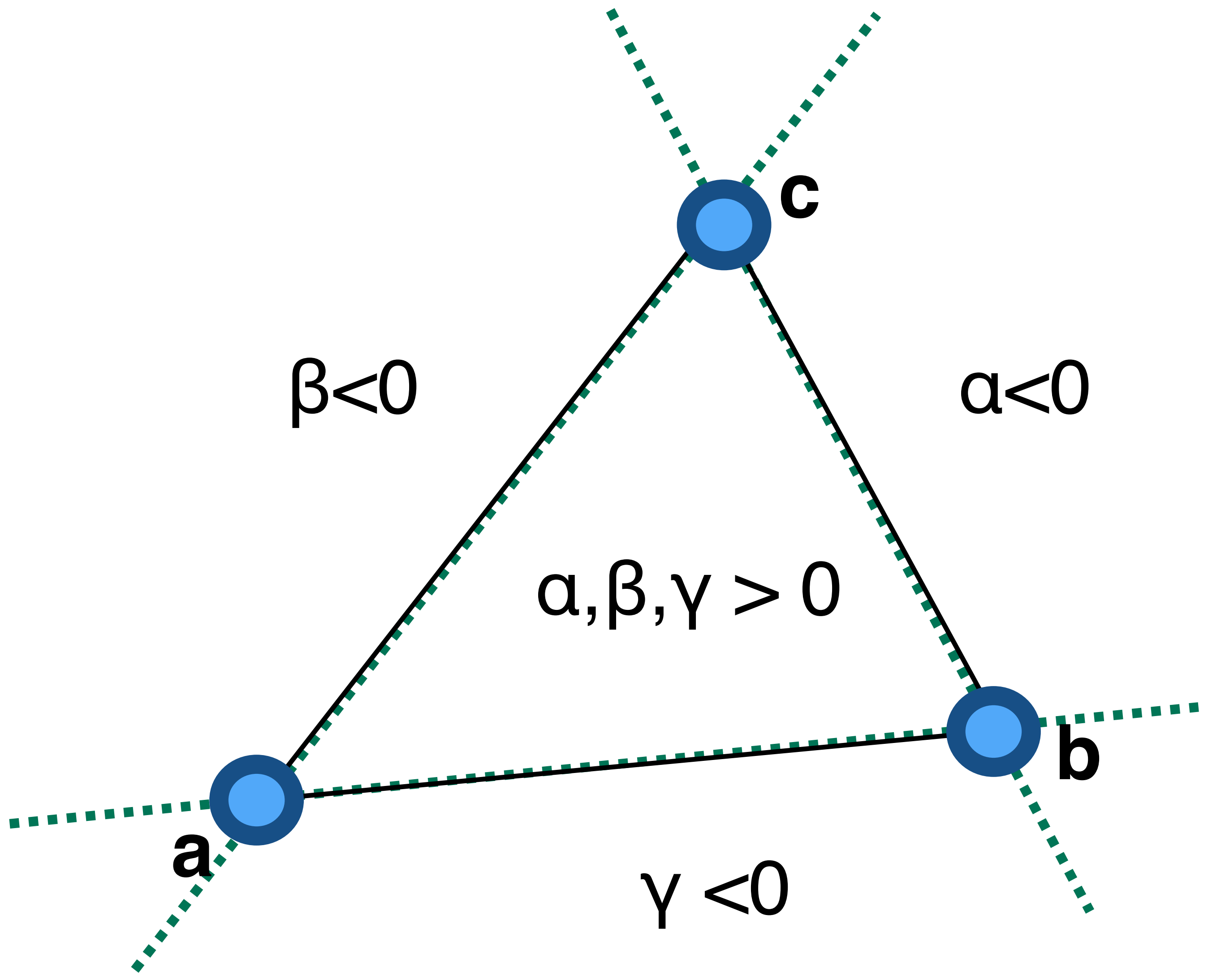
Barycentric Interpolation

- Barycentric coordinates:
  - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-collinear **a**,**b**,**c**
  - Ratio of triangle areas
  - $\alpha(\mathbf{p})$, $\beta(\mathbf{p})$, $\gamma(\mathbf{p})$ are linear functions



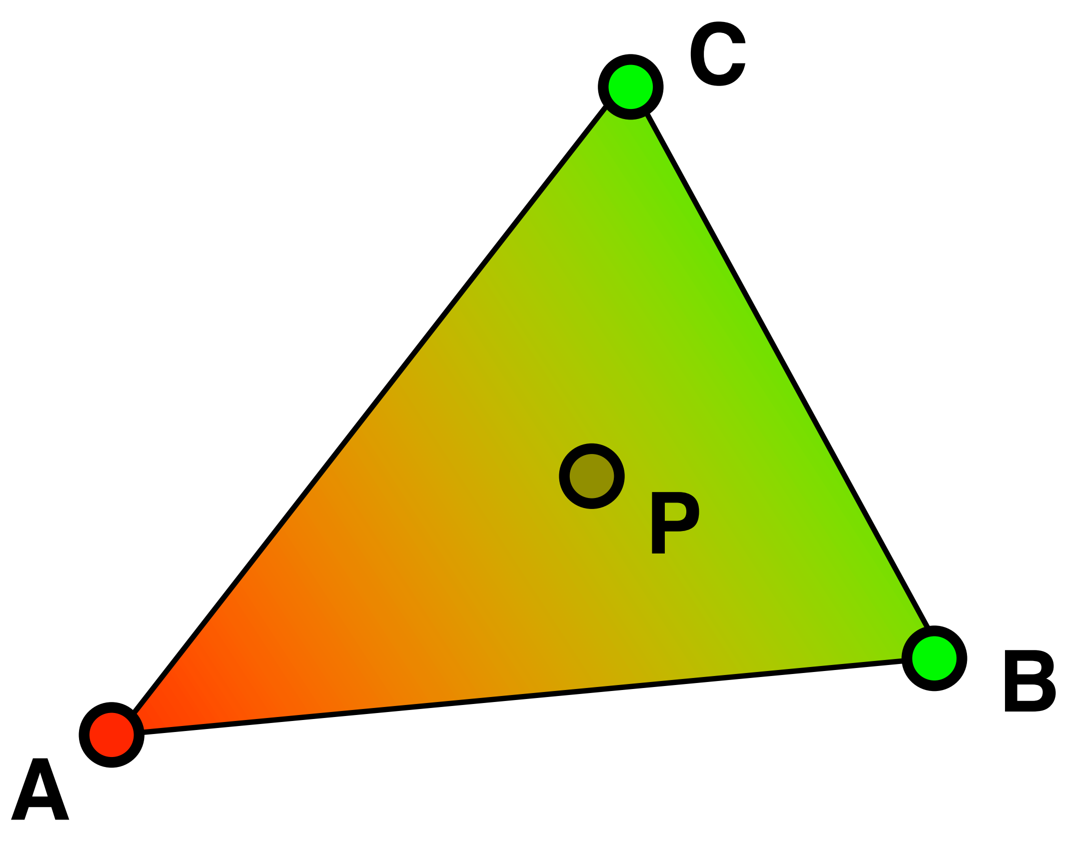
Barycentric Interpolation

- Barycentric coordinates:
  - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-collinear **a**,**b**,**c**
  - Ratio of triangle areas
  - $\alpha(\mathbf{p})$, $\beta(\mathbf{p})$, $\gamma(\mathbf{p})$ are linear functions
  - Gives inside/outside information



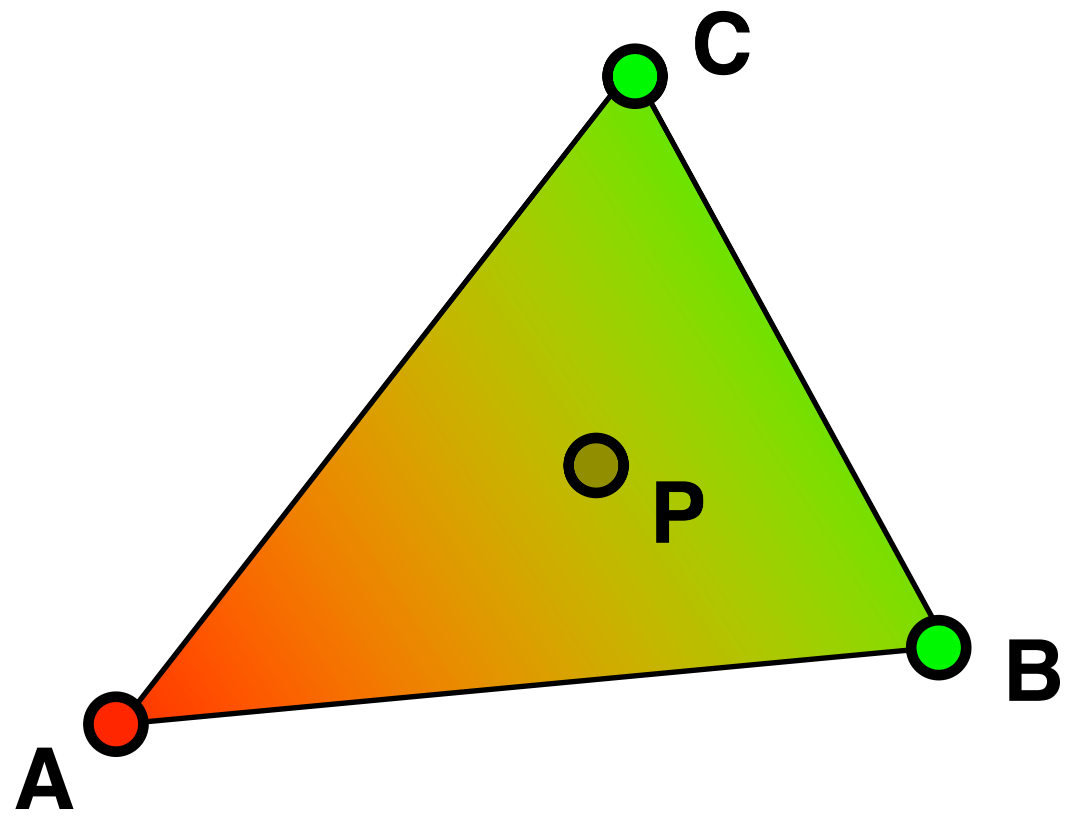
Barycentric Interpolation

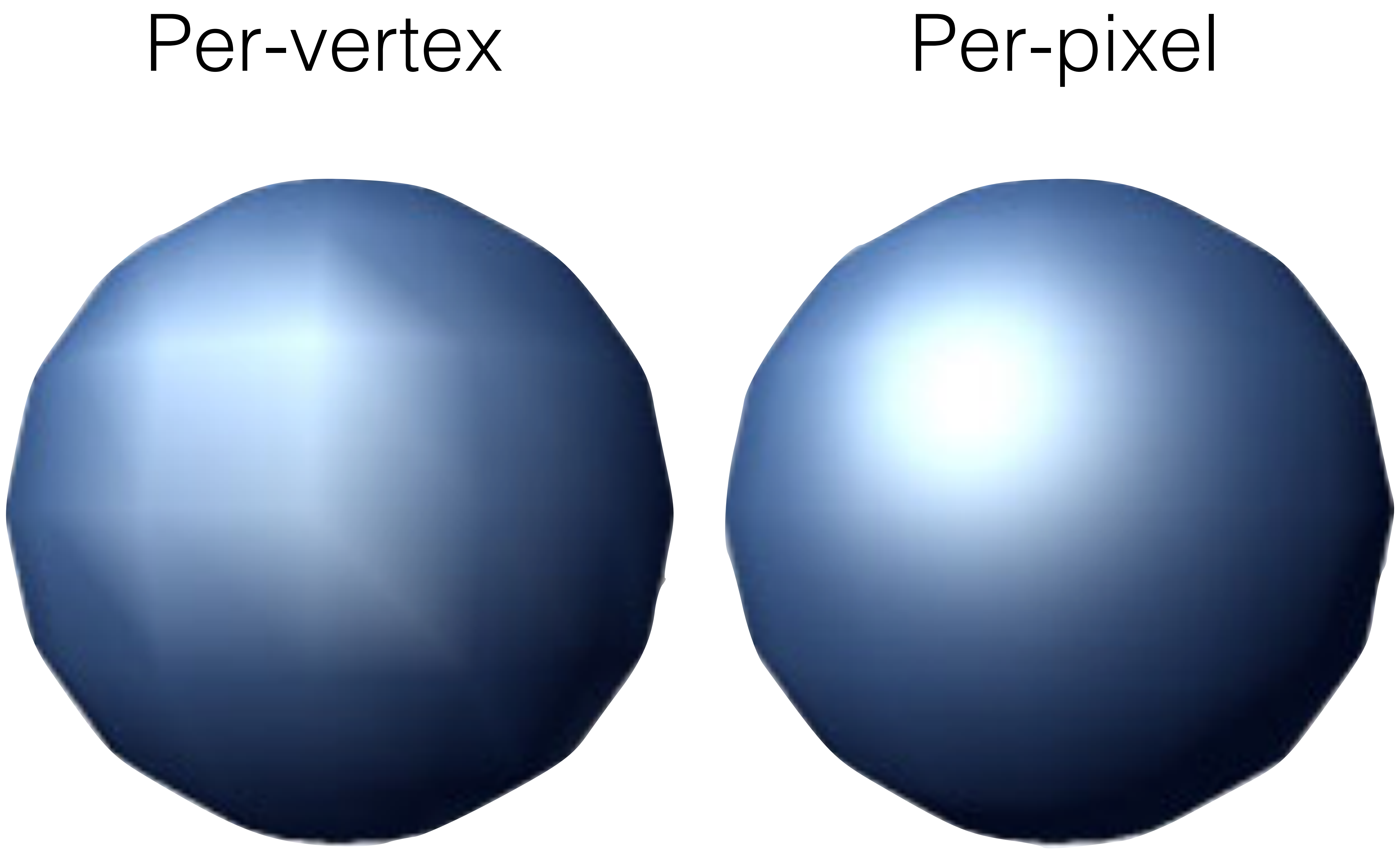
- Barycentric coordinates:
  - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-collinear **a**,**b**,**c**
  - Ratio of triangle areas
  - $\alpha(\mathbf{p})$, $\beta(\mathbf{p})$, $\gamma(\mathbf{p})$ are linear functions
  - Gives inside/outside information
  - Use barycentric coordinates to interpolate vertex normals (or other data, e.g. colors)



$$\mathbf{n}(\mathbf{P}) \quad = \quad \alpha \cdot \mathbf{n}(\mathbf{A}) + \beta \cdot \mathbf{n}(\mathbf{B}) + \gamma \cdot \mathbf{n}(\mathbf{C})$$

Color Interpolation





Evaluate color on vertices, then interpolates it

Interpolates positions and normals, then evaluate color on each pixel

http://www.cgchannel.com/2010/11/cg-science-for-artists-part-2-the-real-time-rendering-pipeline/

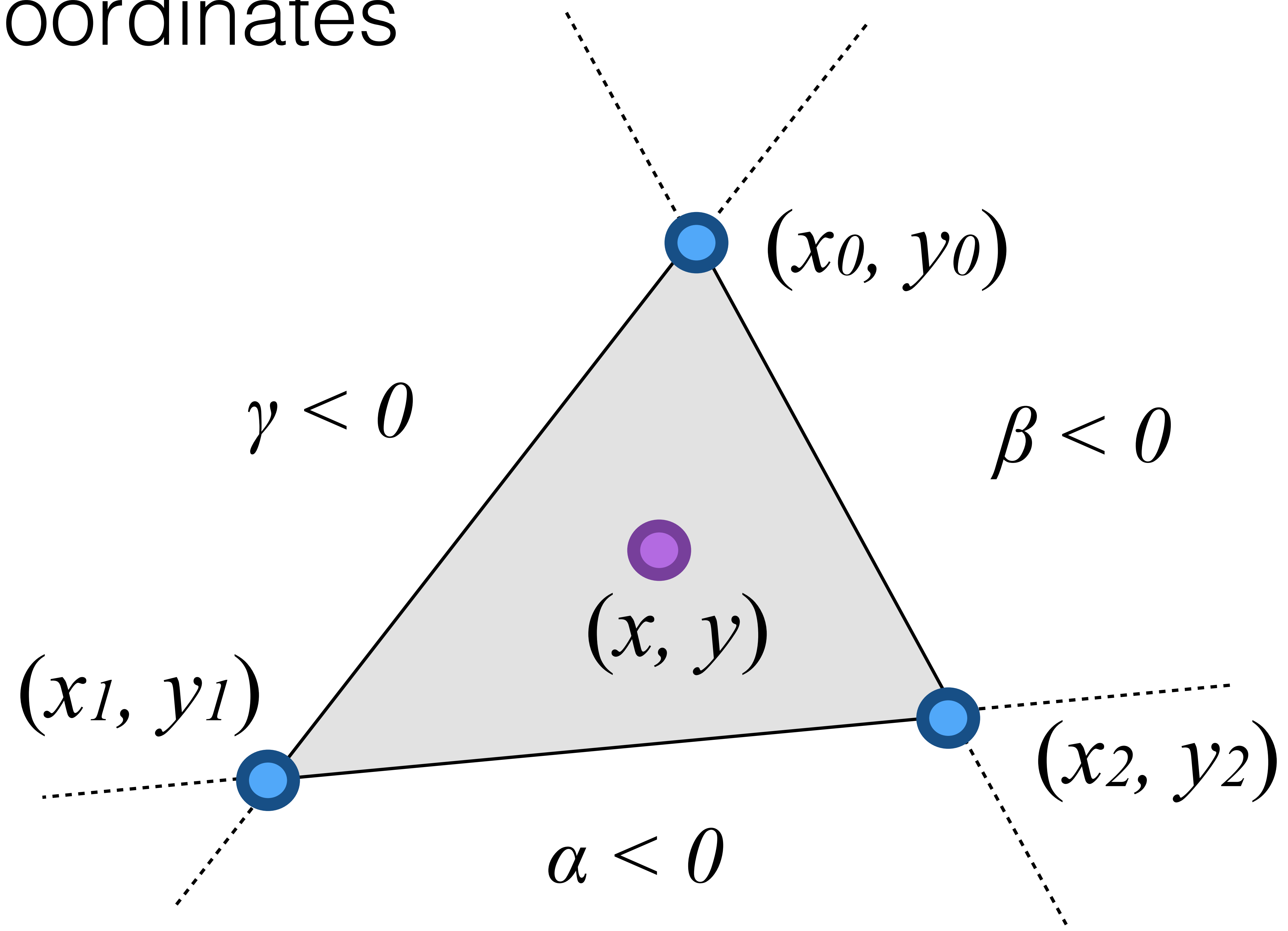
Triangle Rasterization

- Each triangle is represented as three 2D points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$
- Rasterization using barycentric coordinates

$$x = \alpha \cdot x_0 + \beta \cdot x_1 + \gamma \cdot x_2$$

$$y = \alpha \cdot y_0 + \beta \cdot y_1 + \gamma \cdot y_2$$

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



Triangle Rasterization

- Each triangle is represented as three 2D points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$
- Rasterization using barycentric coordinates

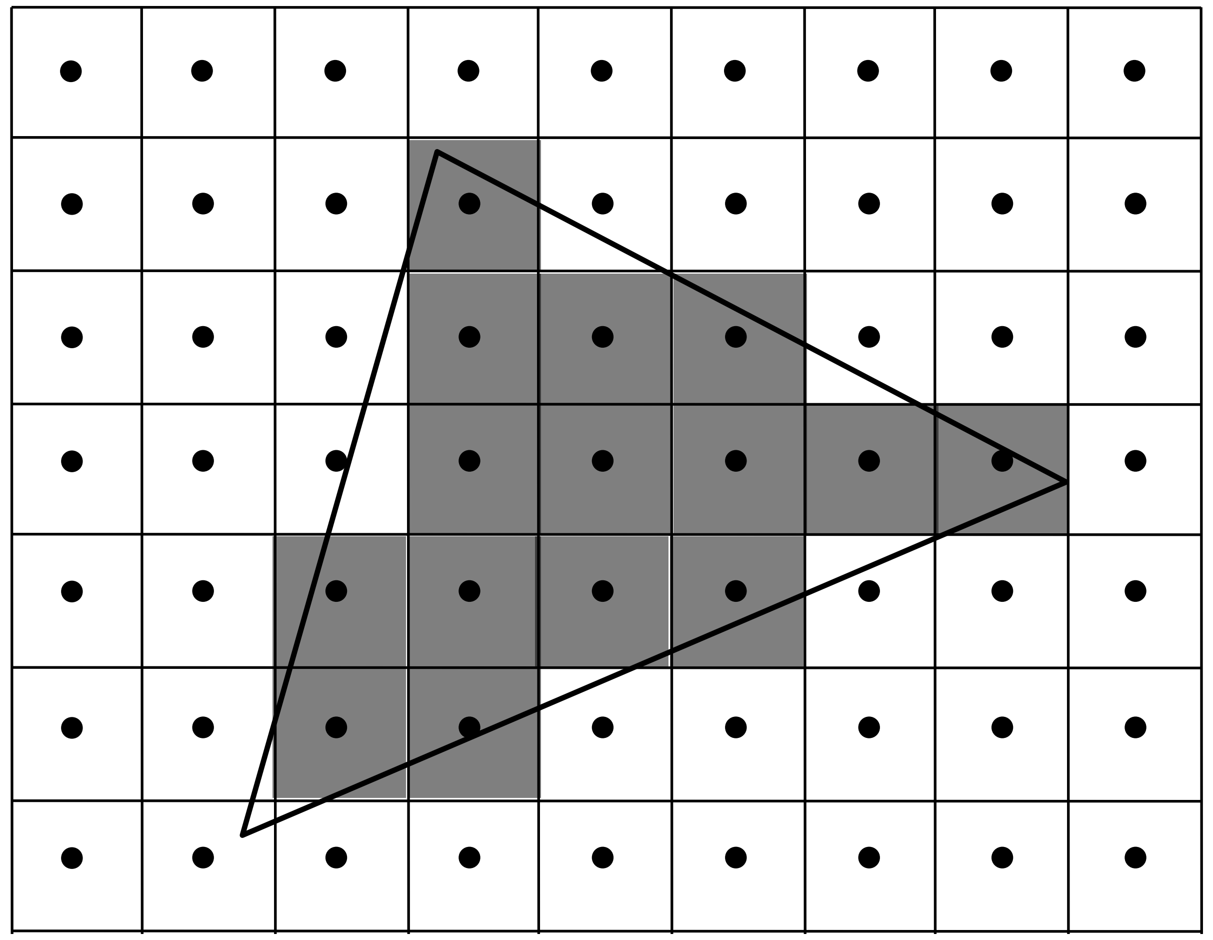
for all y **do**

  for all x **do**

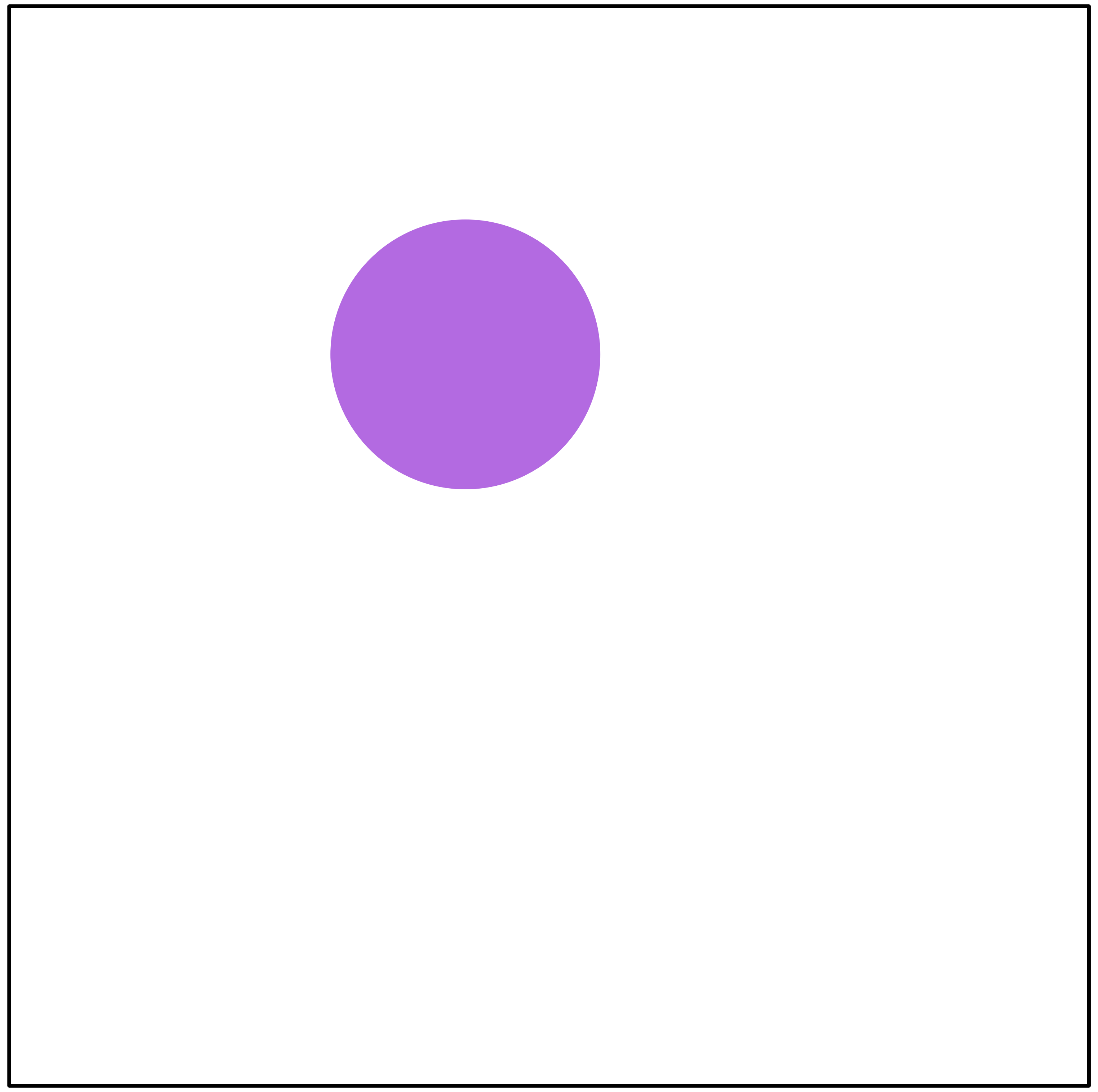
    compute (α, β, γ) for (x, y)

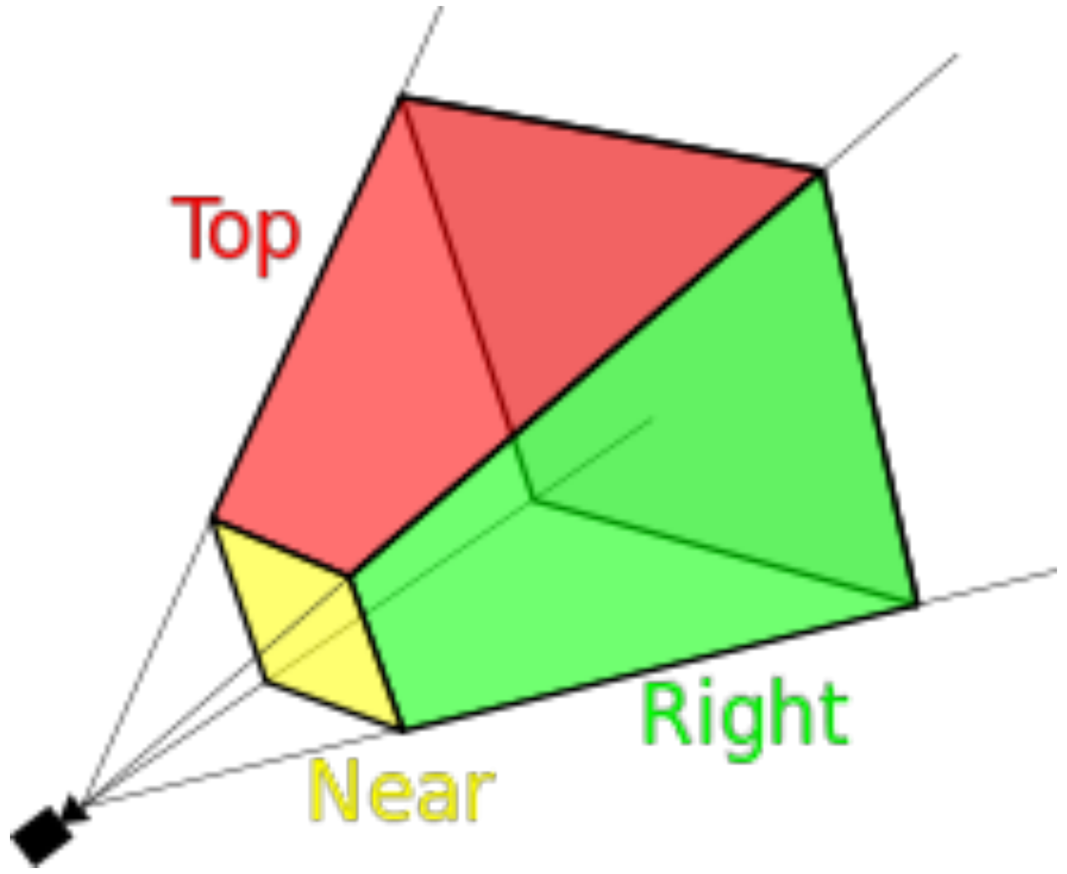
    if $(\alpha \in [0,1]$ and $\beta \in [0,1]$ and $\gamma \in [0,1]$

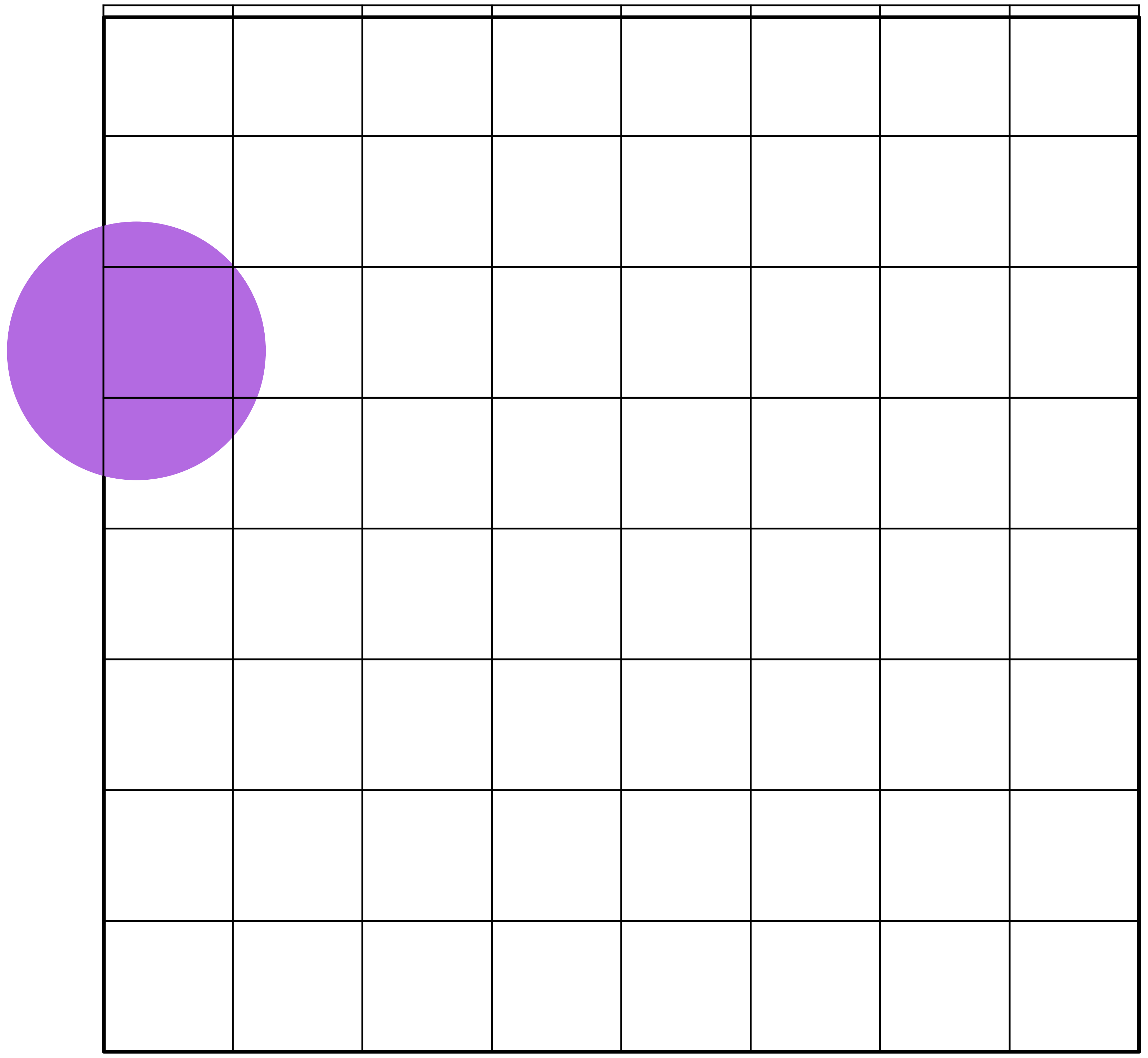
      set_pixel (x, y)



Clipping



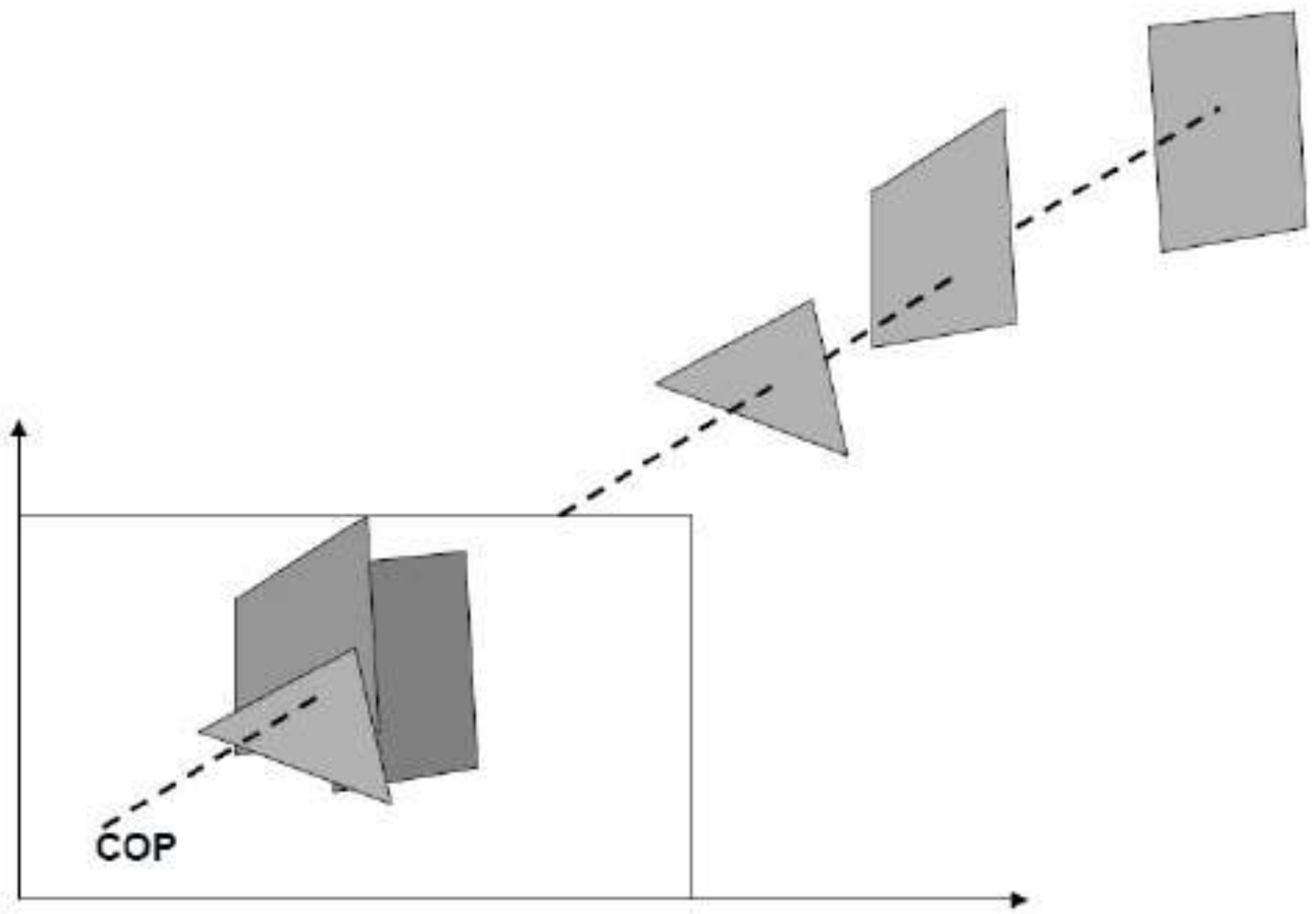




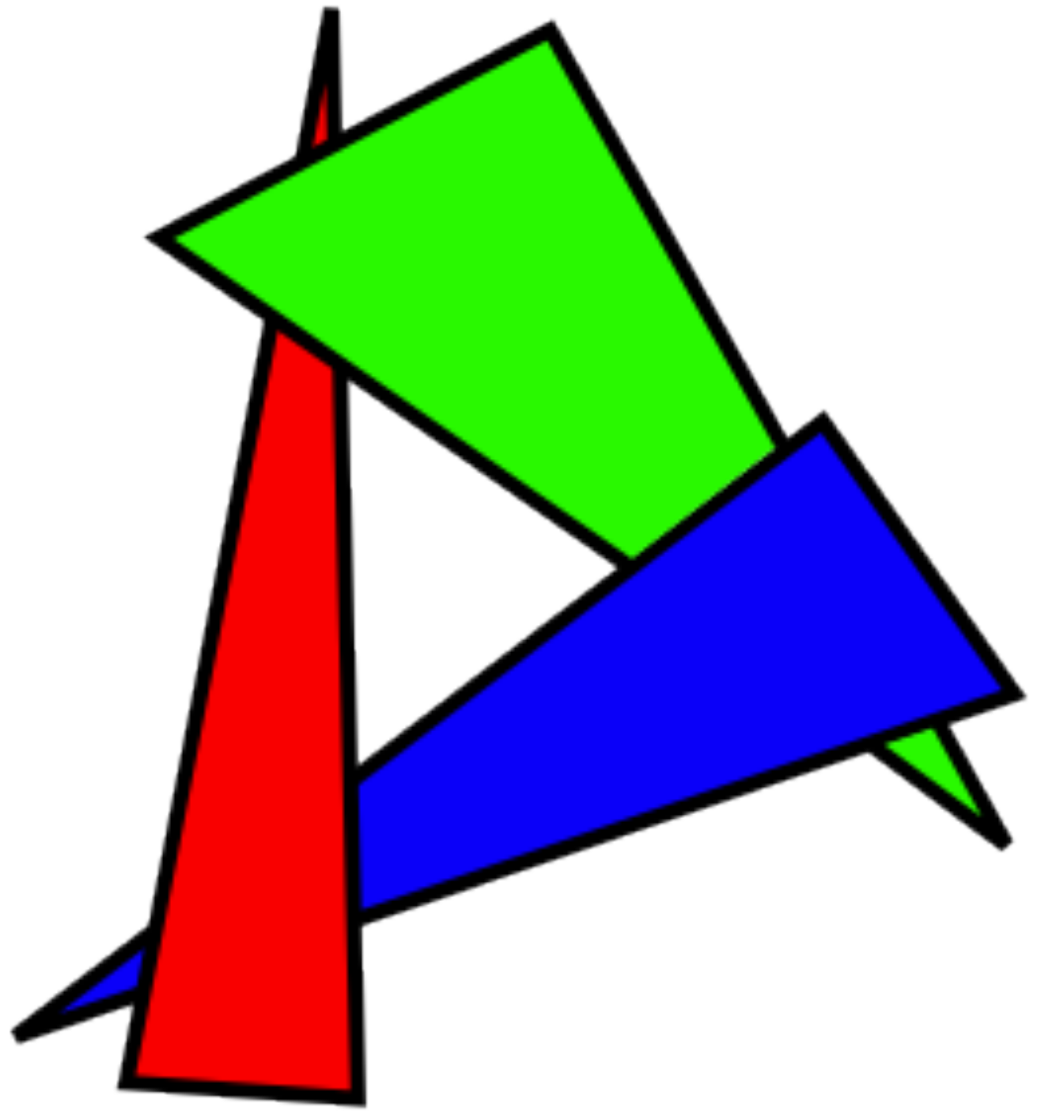
- Ok if you do it brute force
- Care is required if you are explicitly tracing the boundaries

Objects Depth Sorting

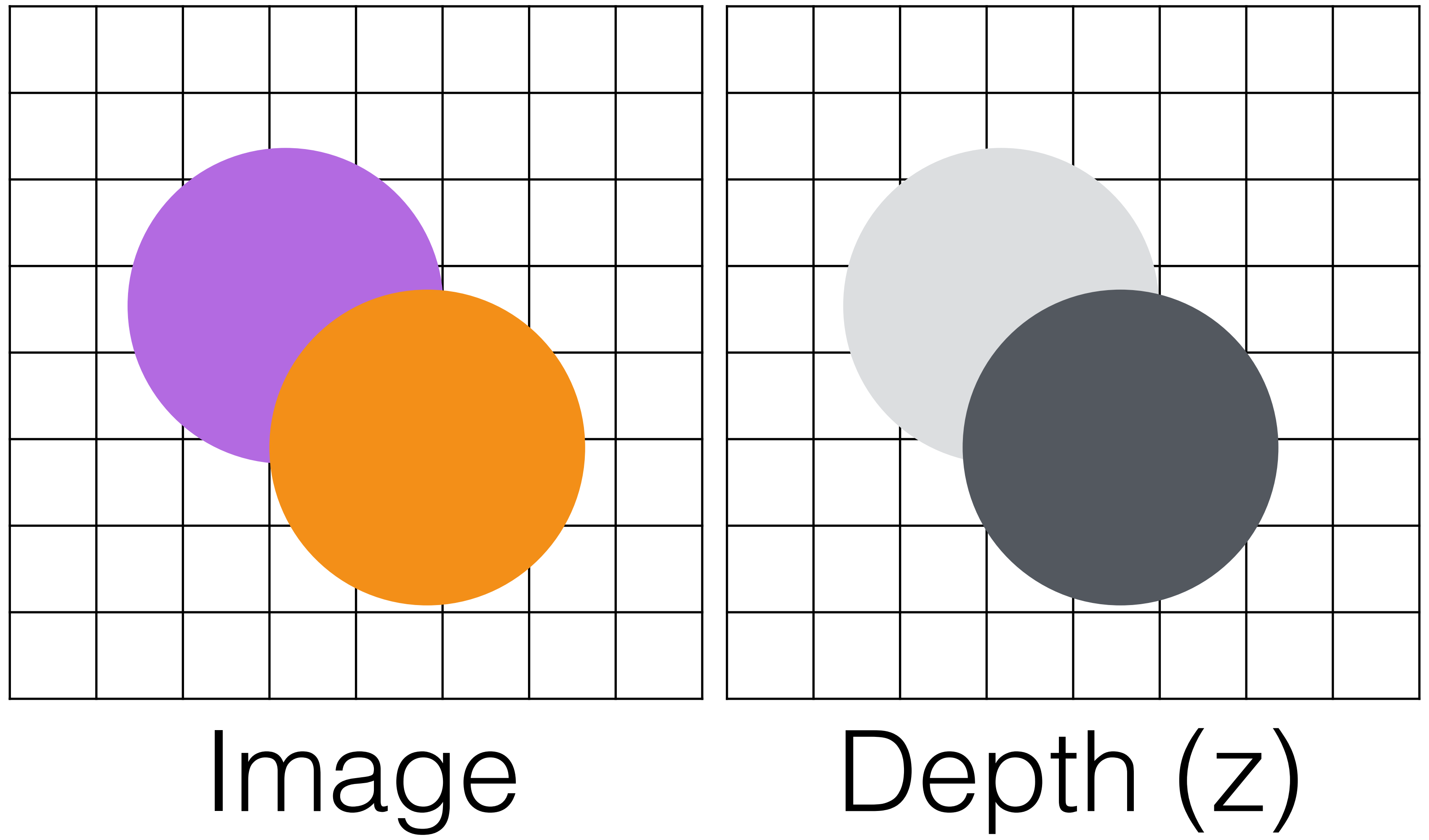
- To handle occlusion, you can sort all the objects in a scene by depth



- This is not always possible!



z-buffering

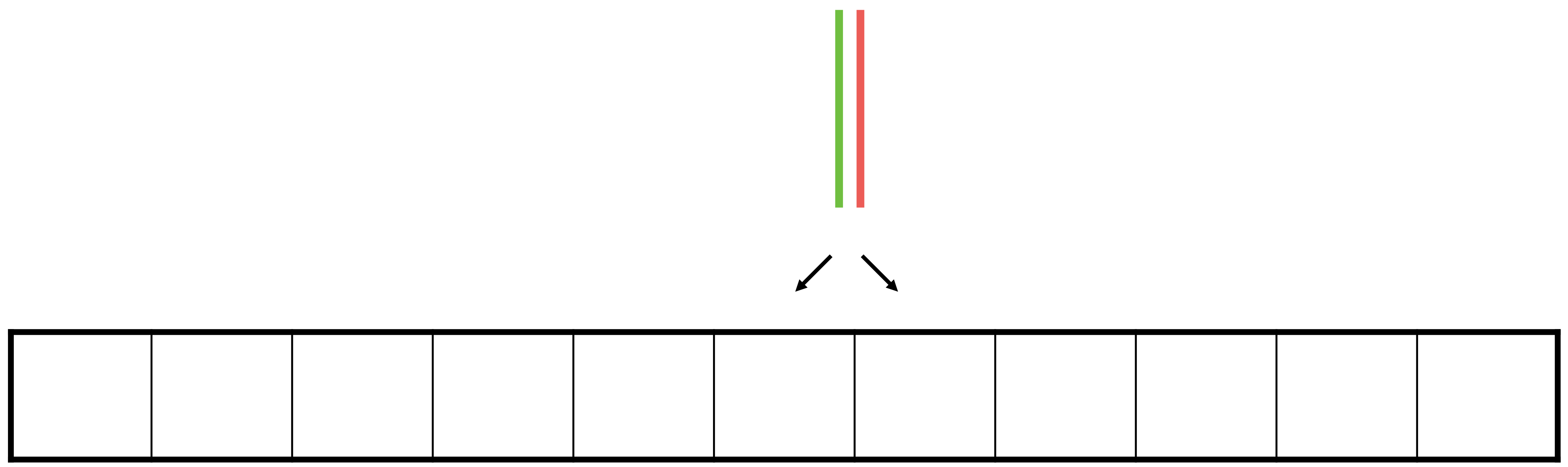


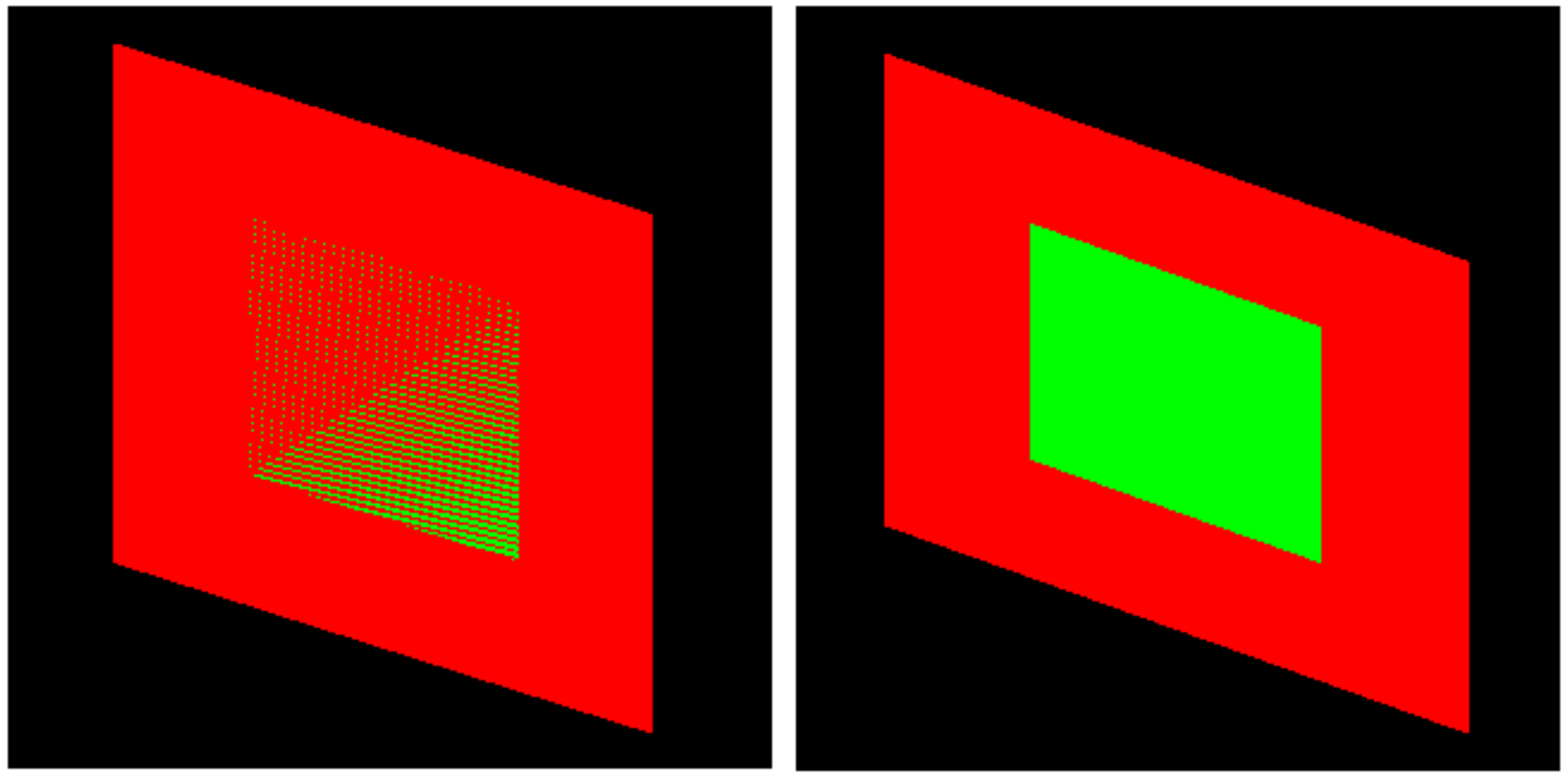
- You render the image both in the Image and in the depth buffer, where you store only the depth
- When a new fragment comes in, you draw it in the image only if it is closer
- This always work and it is cheap to evaluate! It is the default in all graphics hardware
- You still have to sort for transparency…

z-buffer quantization and “z-fighting”

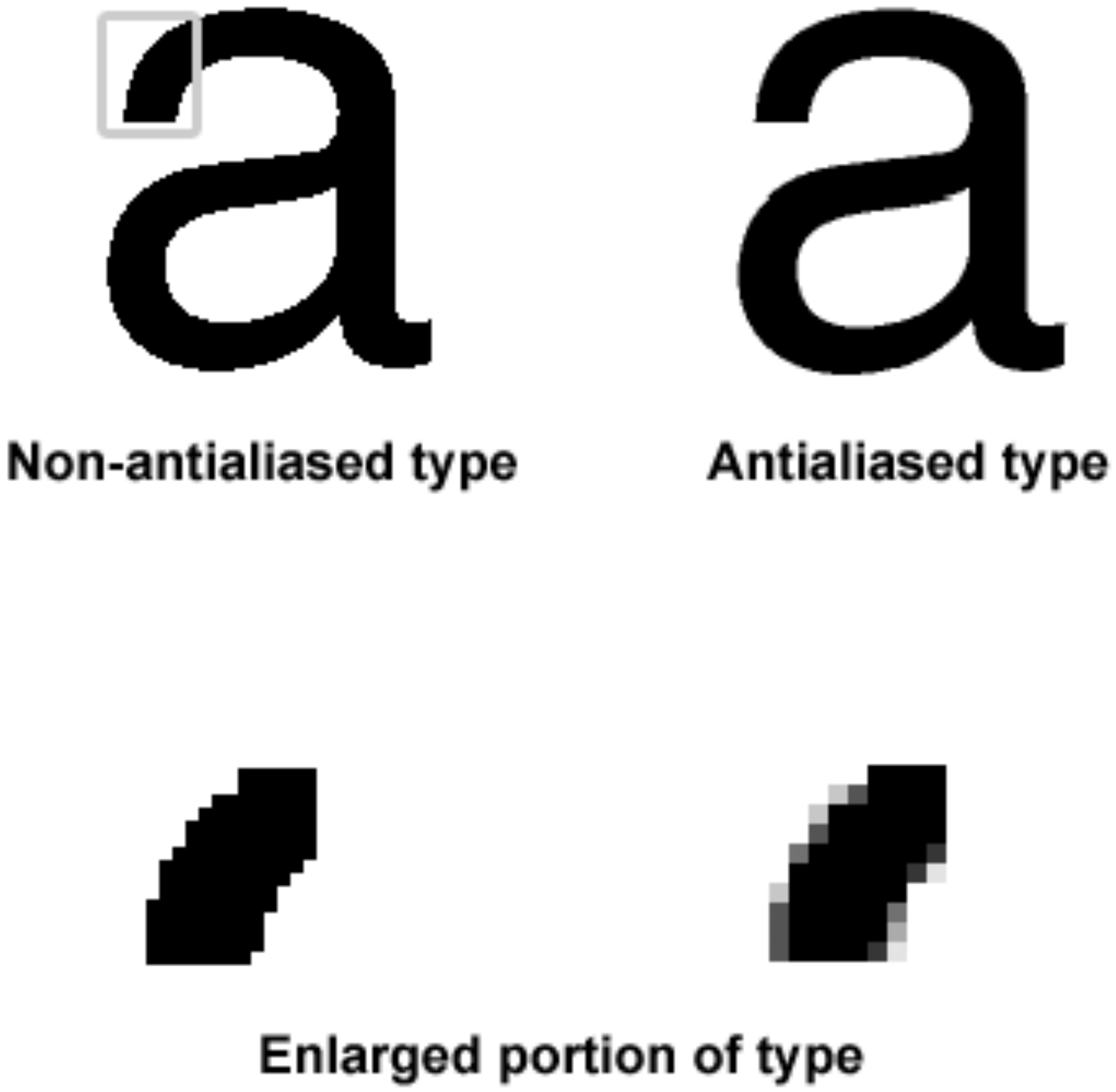
- The z-buffer is quantized (the number of bits is heavily dependent on the hardware platform)

- Two close object might be quantized differently, leading to strange artifacts, usually called “z-fighting”





Super Sampling Anti-Aliasing



- Render nxn pixels instead of one
- Assign the average to the pixel

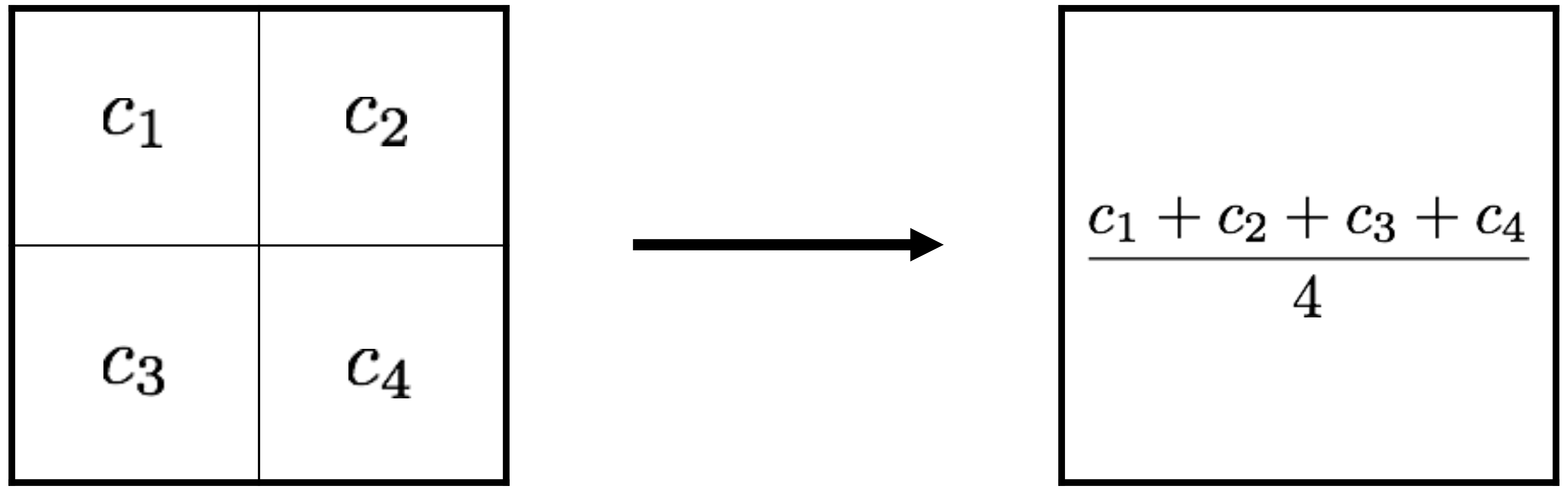
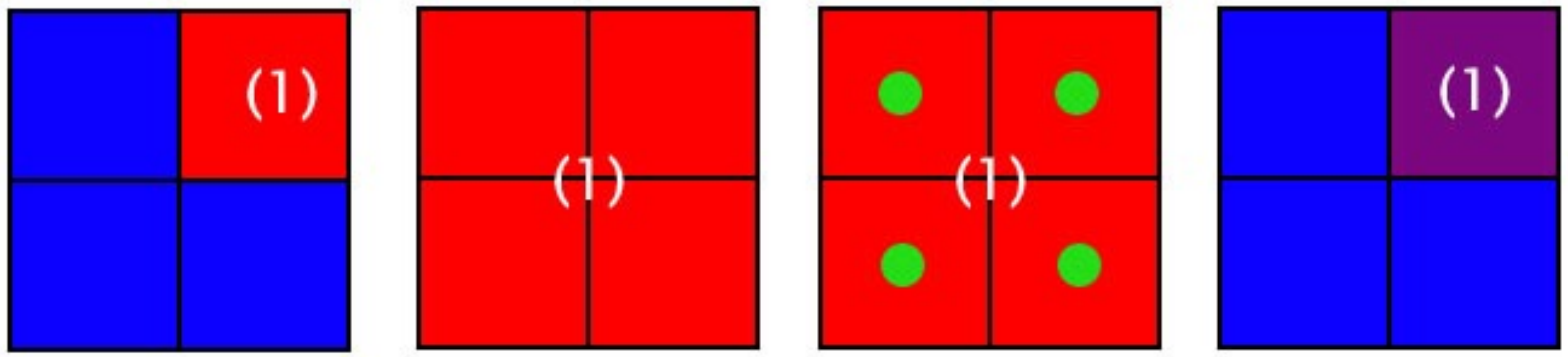


Image Copyright: Fritz Kessler

Many different names and variants

- SSAA (FSAA)
- MSAA
- CSAA
- EQAA
- FXAA
- TX AA



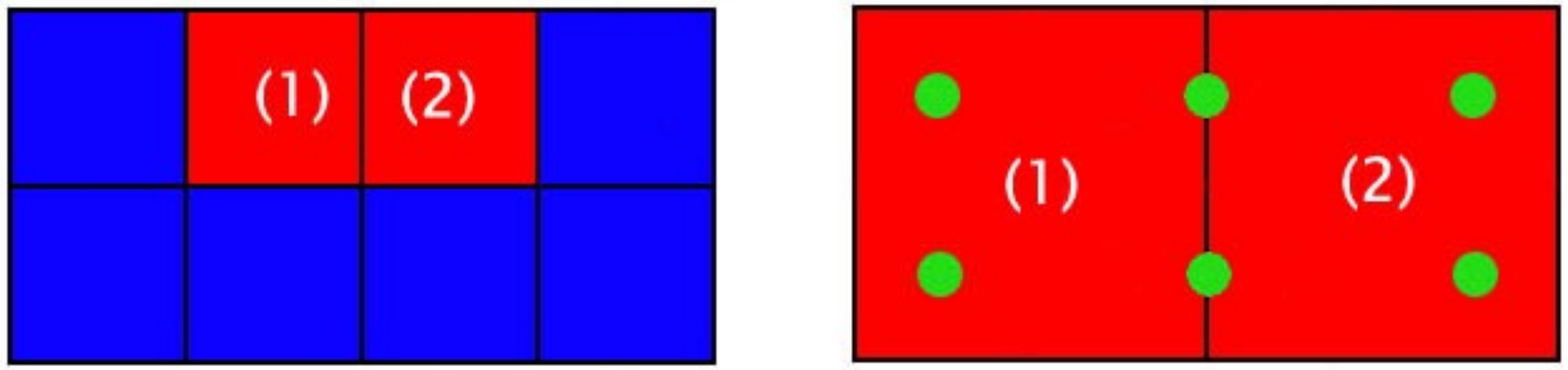
4 pixels

Pixel (1) gets supersampled to 4 times the resolution

Colour averages are taken from the 4 points in the pixel

When downsized again, (1) is a blended colour

MSAA



Rather than individually, pixels are sampled together. In this example, we've taken two. The pixels are not scaled up.

Assuming 4x AA, the two pixels share two samples in the middle, meaning only six samples instead of eight.

References

Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 8