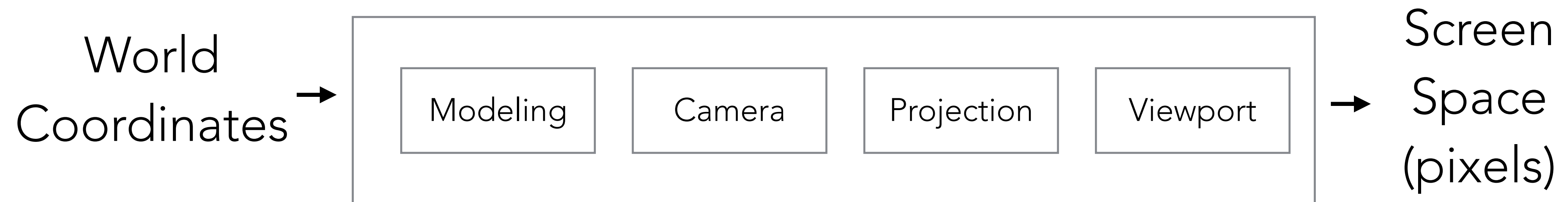
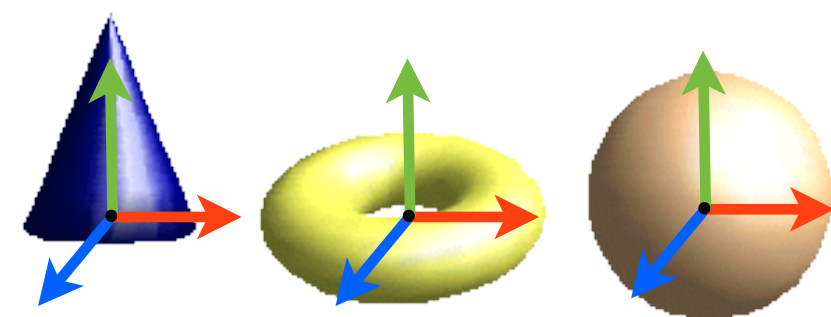


Viewing Transformations

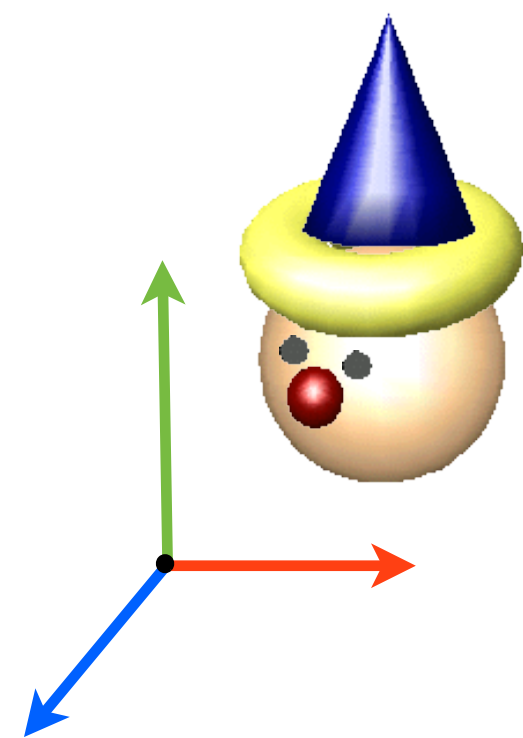
Viewing transformations



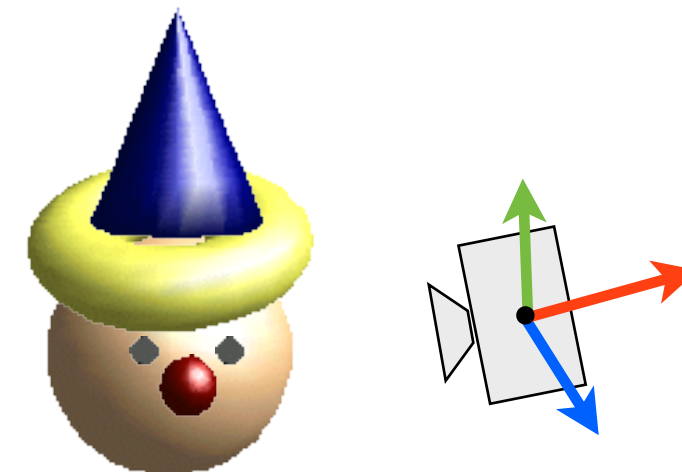
Coordinate Systems



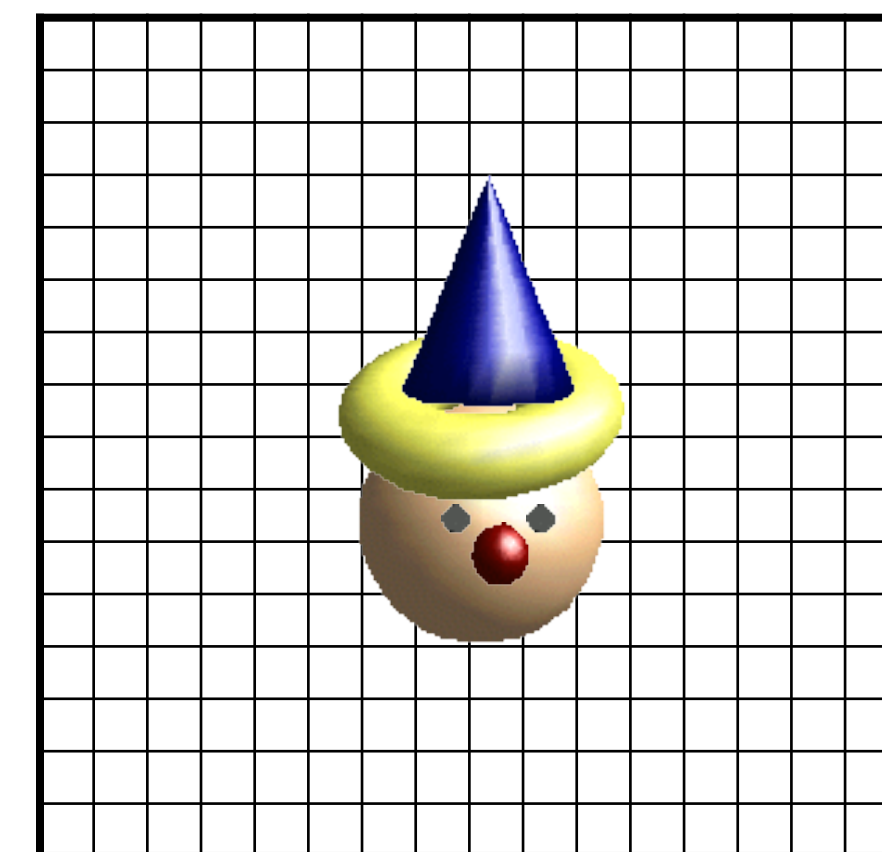
object
coordinates



world
coordinates



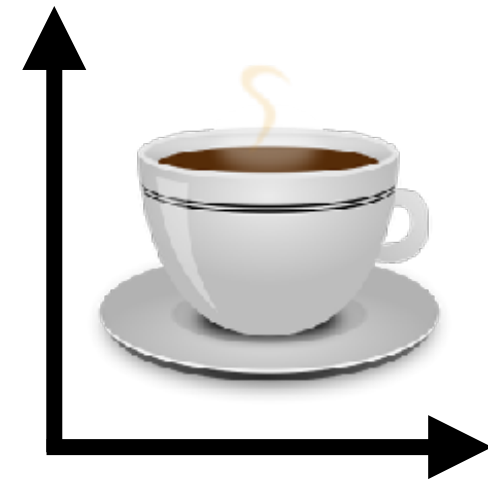
camera
coordinates



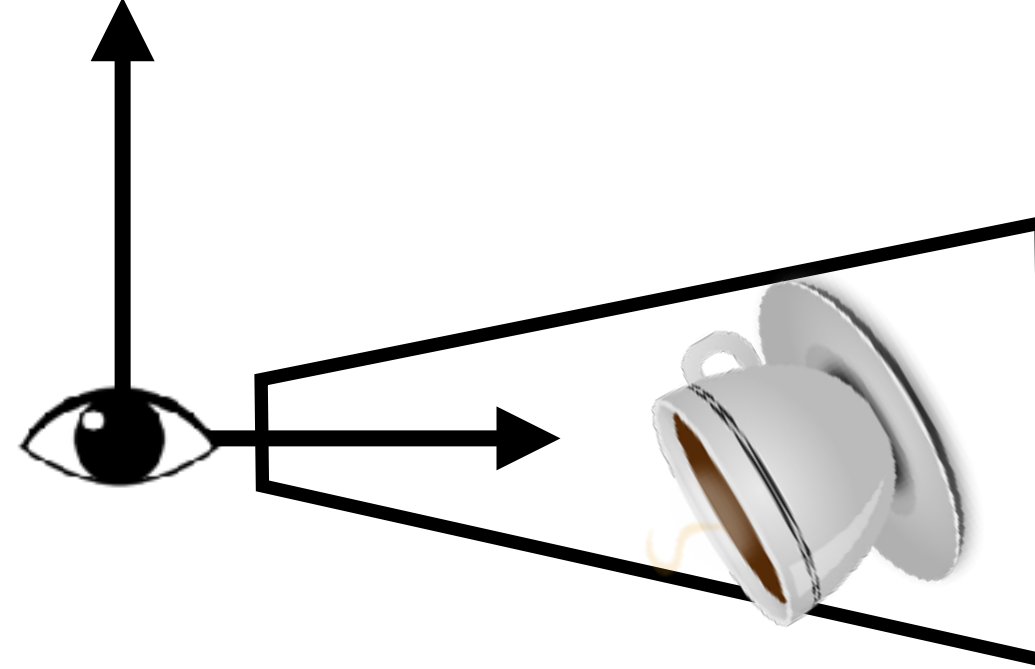
screen
coordinates

Viewing Transformation

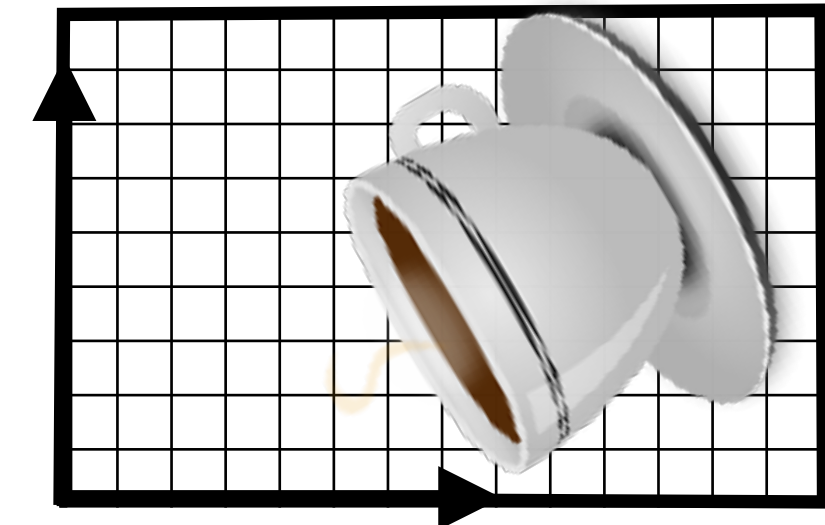
object space



camera space



screen space



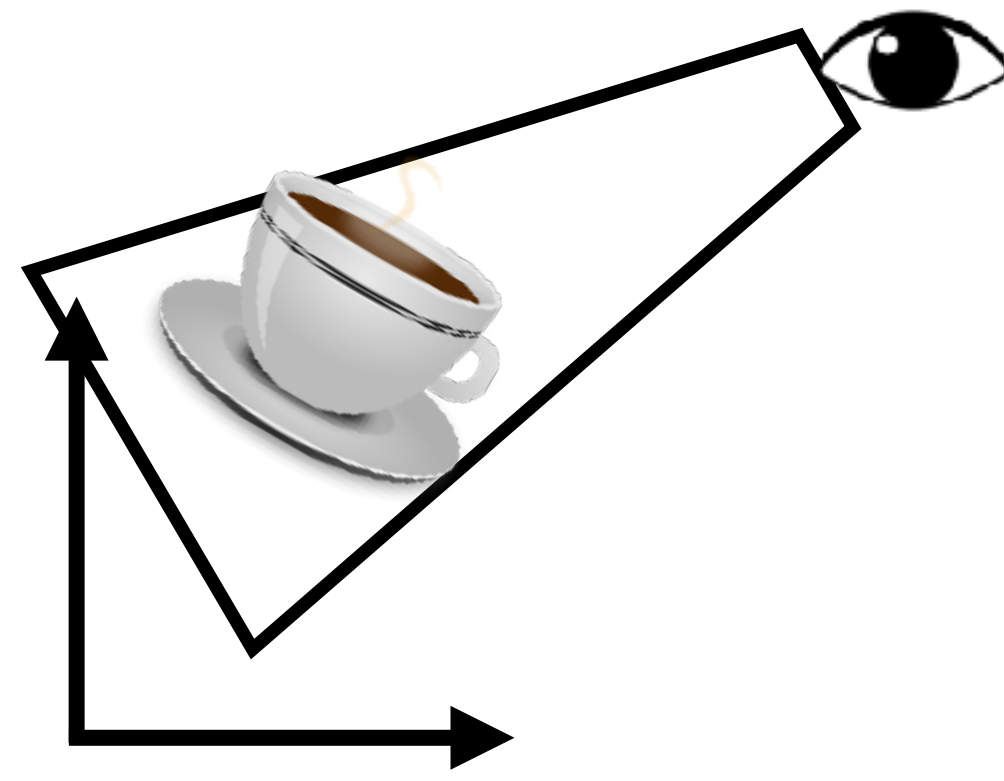
model

camera

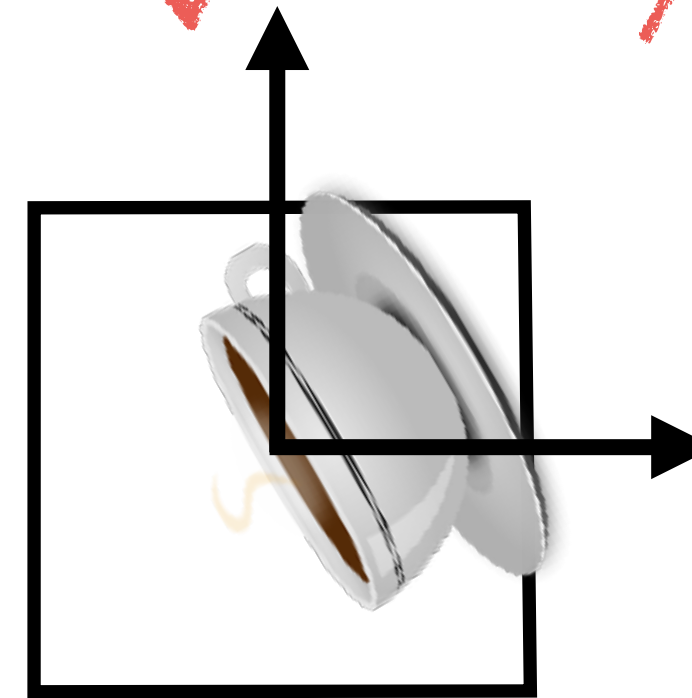
projection

viewport

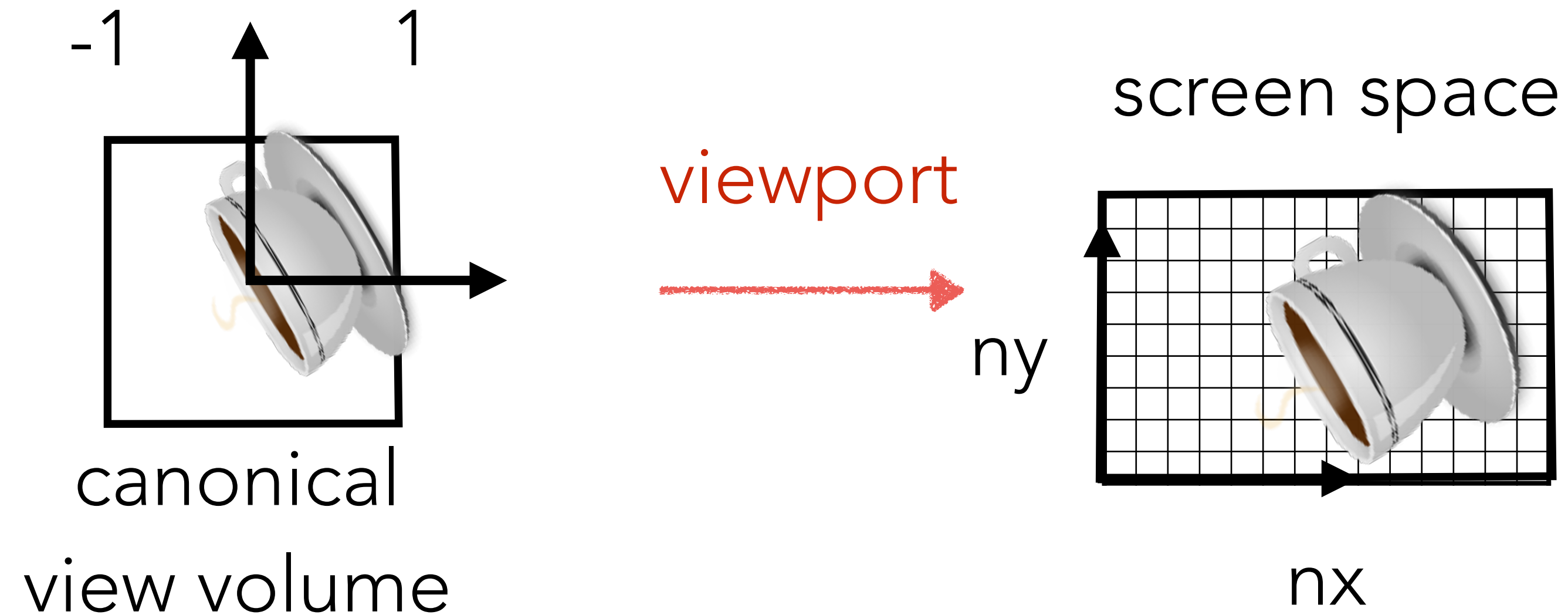
world space



canonical
view volume



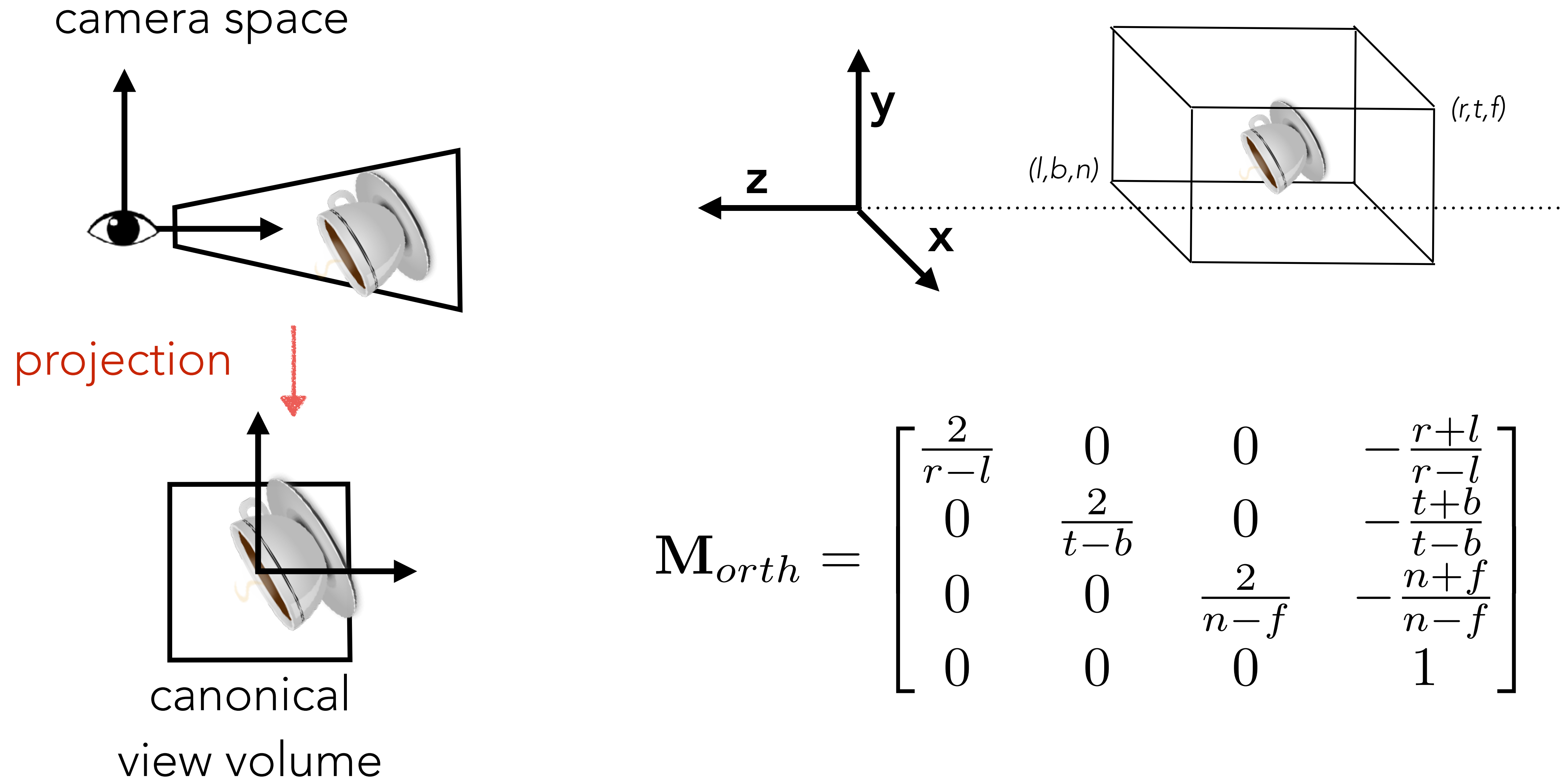
Viewport transformation



$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} nx/2 & 0 & \frac{nx-1}{2} \\ 0 & ny/2 & \frac{ny-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ 1 \end{bmatrix}$$

How does it look in 3D?

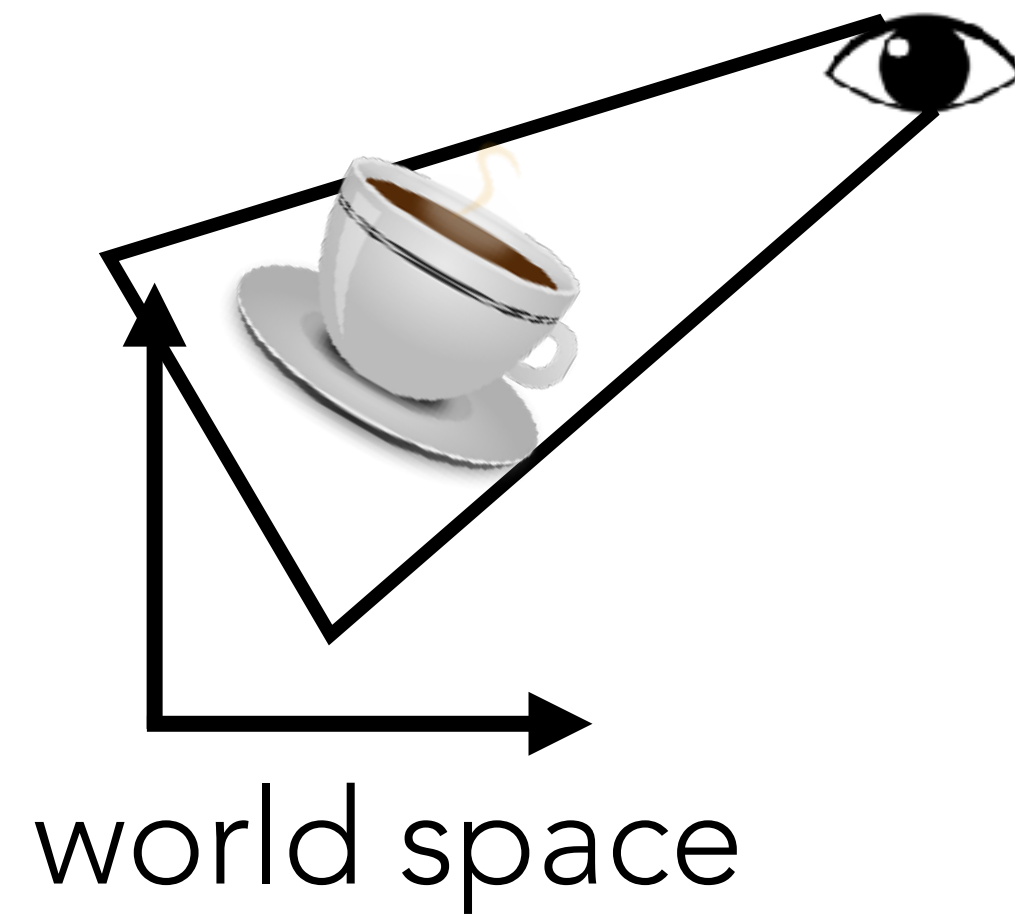
Orthographic Projection



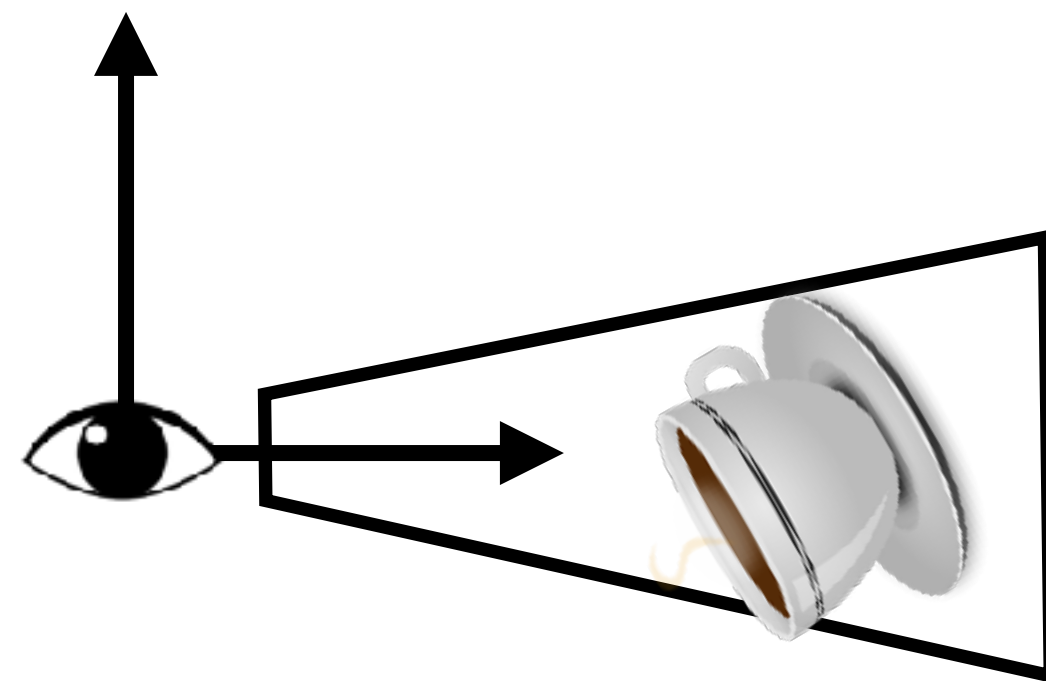
Camera Transformation

1. Construct the camera reference system given:

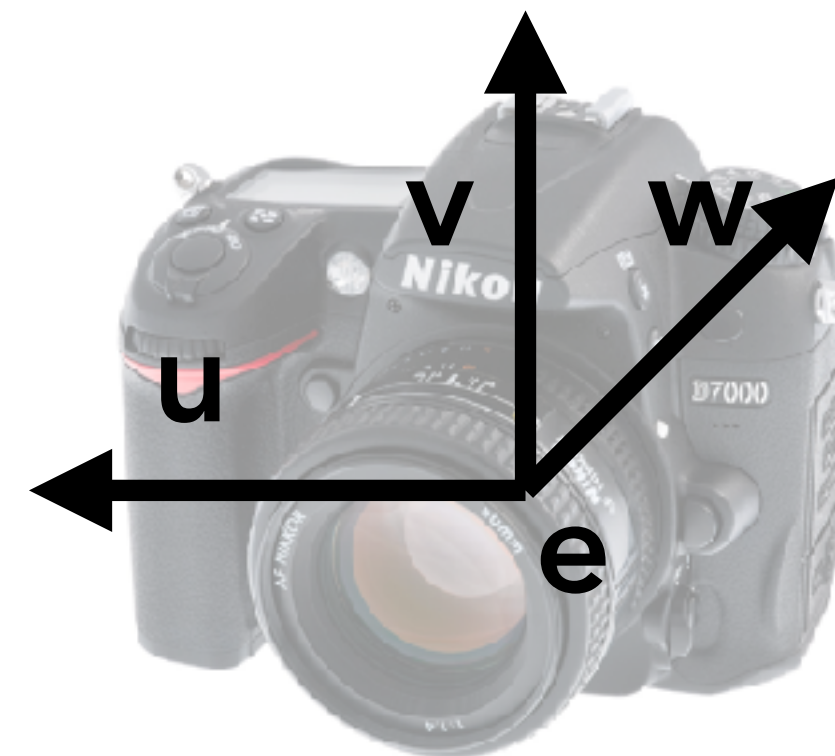
1. The eye position \mathbf{e}
2. The gaze direction \mathbf{g}
3. The view-up vector \mathbf{t}



camera

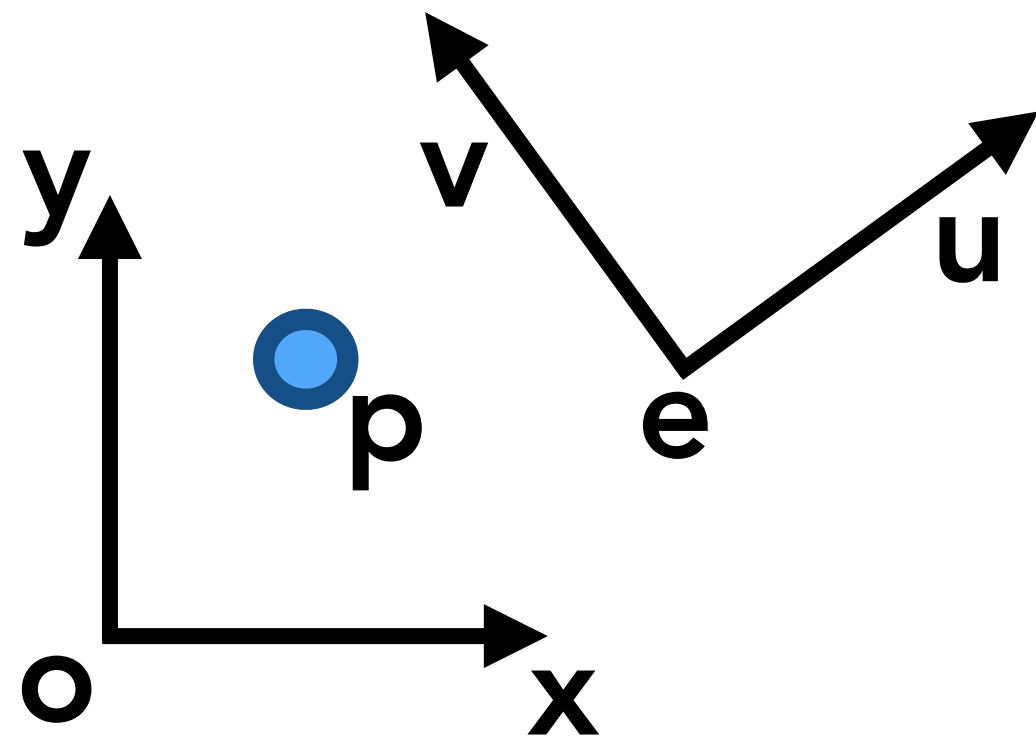


camera space



$$\mathbf{w} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}$$
$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$$
$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

Change of frame



$$\mathbf{p} = (p_x, p_y) = \mathbf{o} + p_x \mathbf{x} + p_y \mathbf{y}$$

$$\mathbf{p} = (p_u, p_v) = \mathbf{e} + p_u \mathbf{u} + p_v \mathbf{v}$$

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & v_x & e_x \\ u_y & v_y & e_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix}$$

$$\mathbf{p}_{xy} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{uv}$$

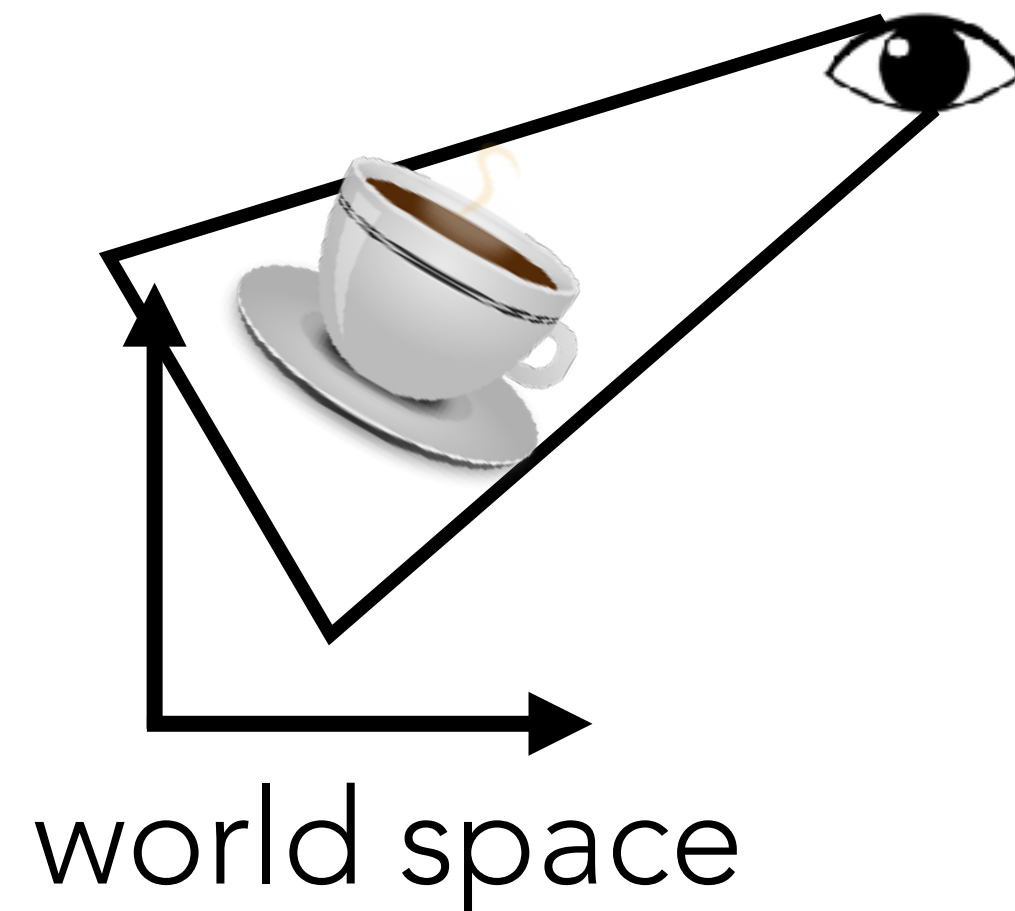
$$\mathbf{p}_{uv} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{e} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}_{xy}$$

Can you write it directly without the inverse?

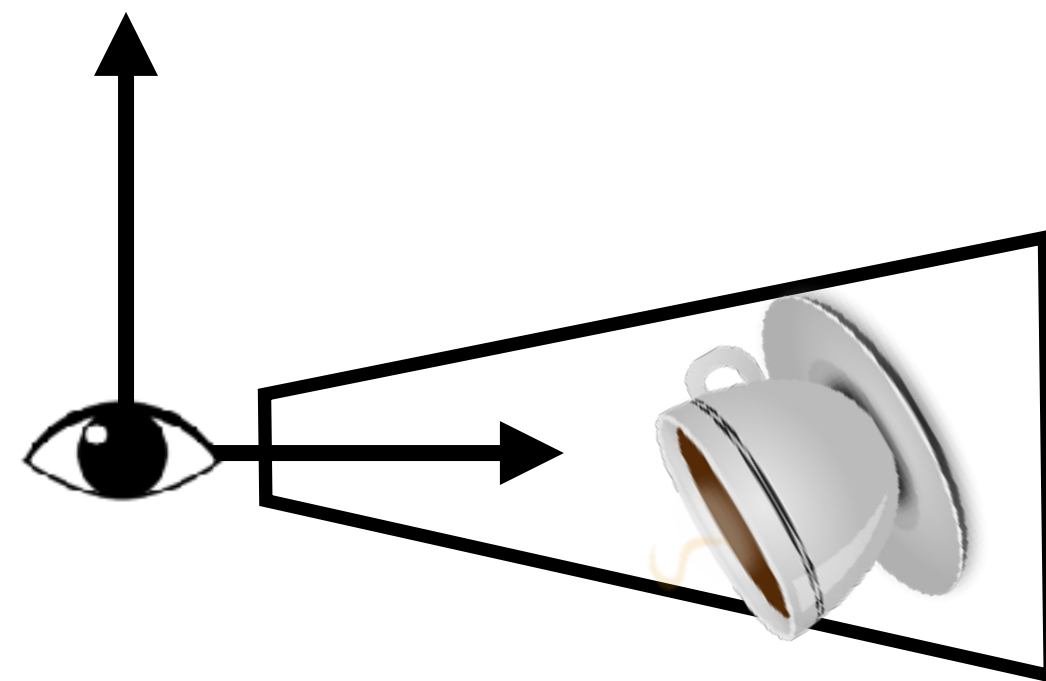
Camera Transformation

1. Construct the camera reference system given:

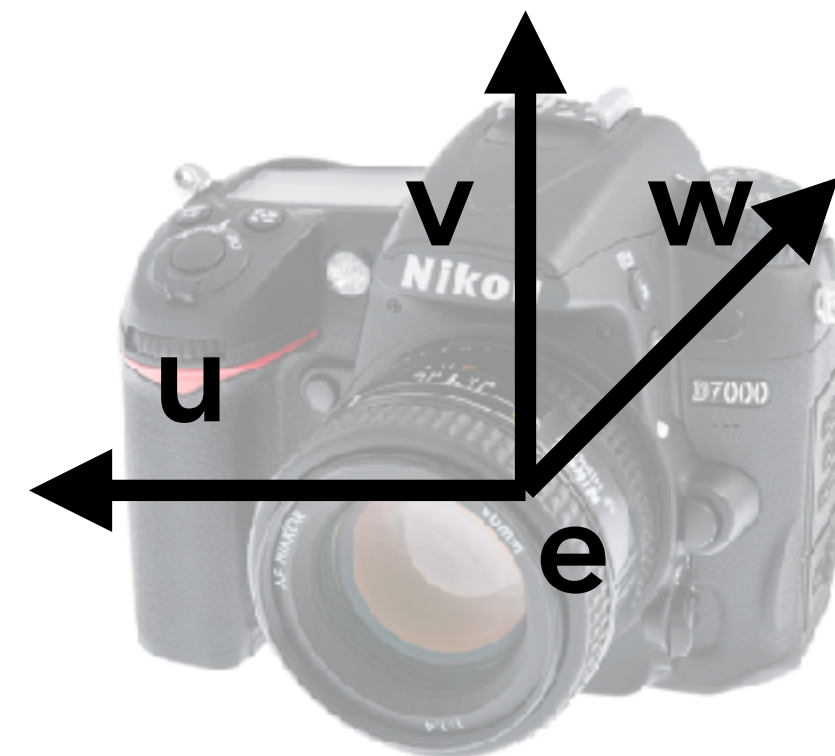
1. The eye position \mathbf{e}
2. The gaze direction \mathbf{g}
3. The view-up vector \mathbf{t}



camera



camera space



$$\mathbf{w} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}$$

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

2. Construct the unique transformations that converts world coordinates into camera coordinates

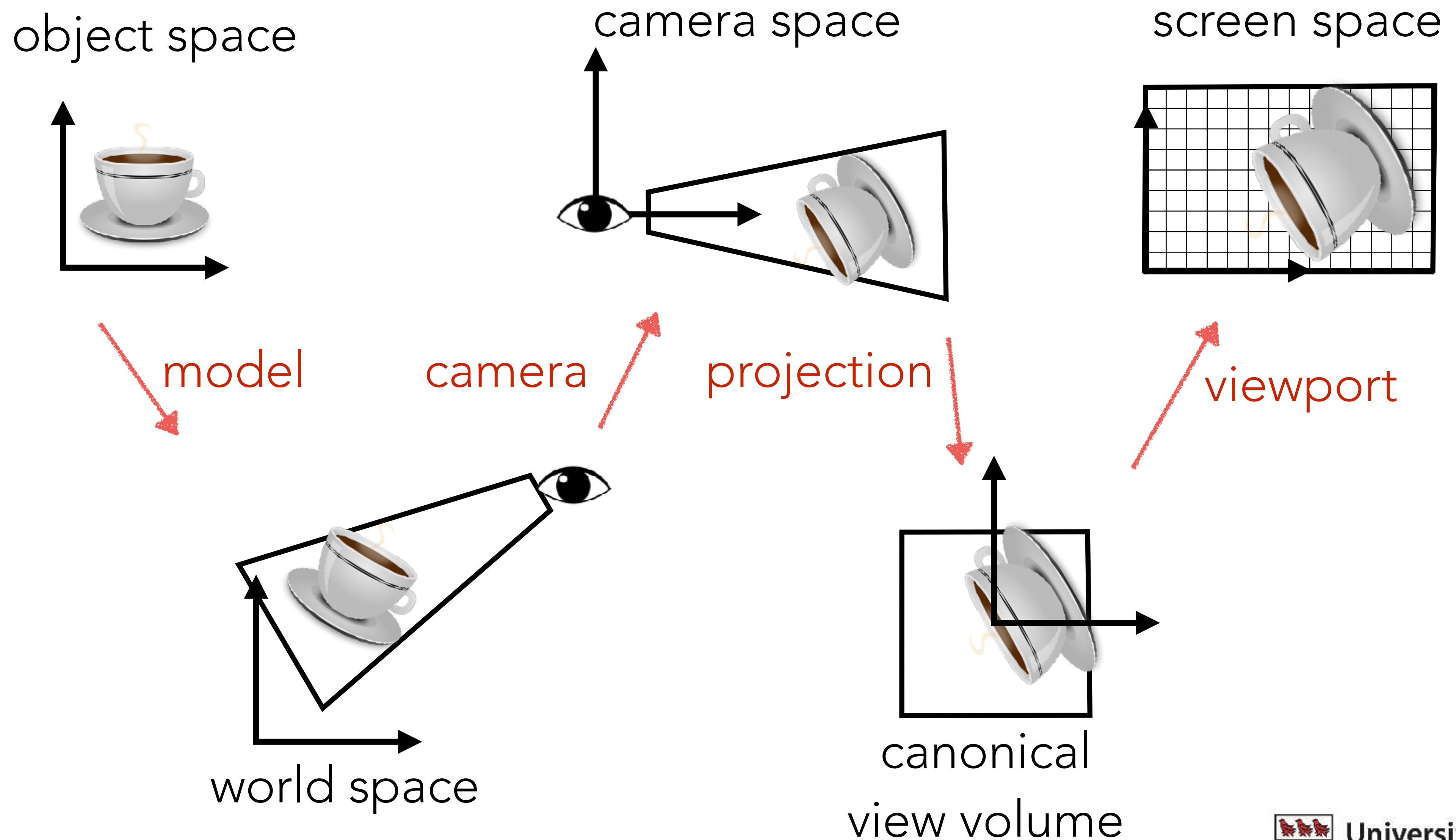
$$\mathbf{M}_{cam} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$



University
of Victoria

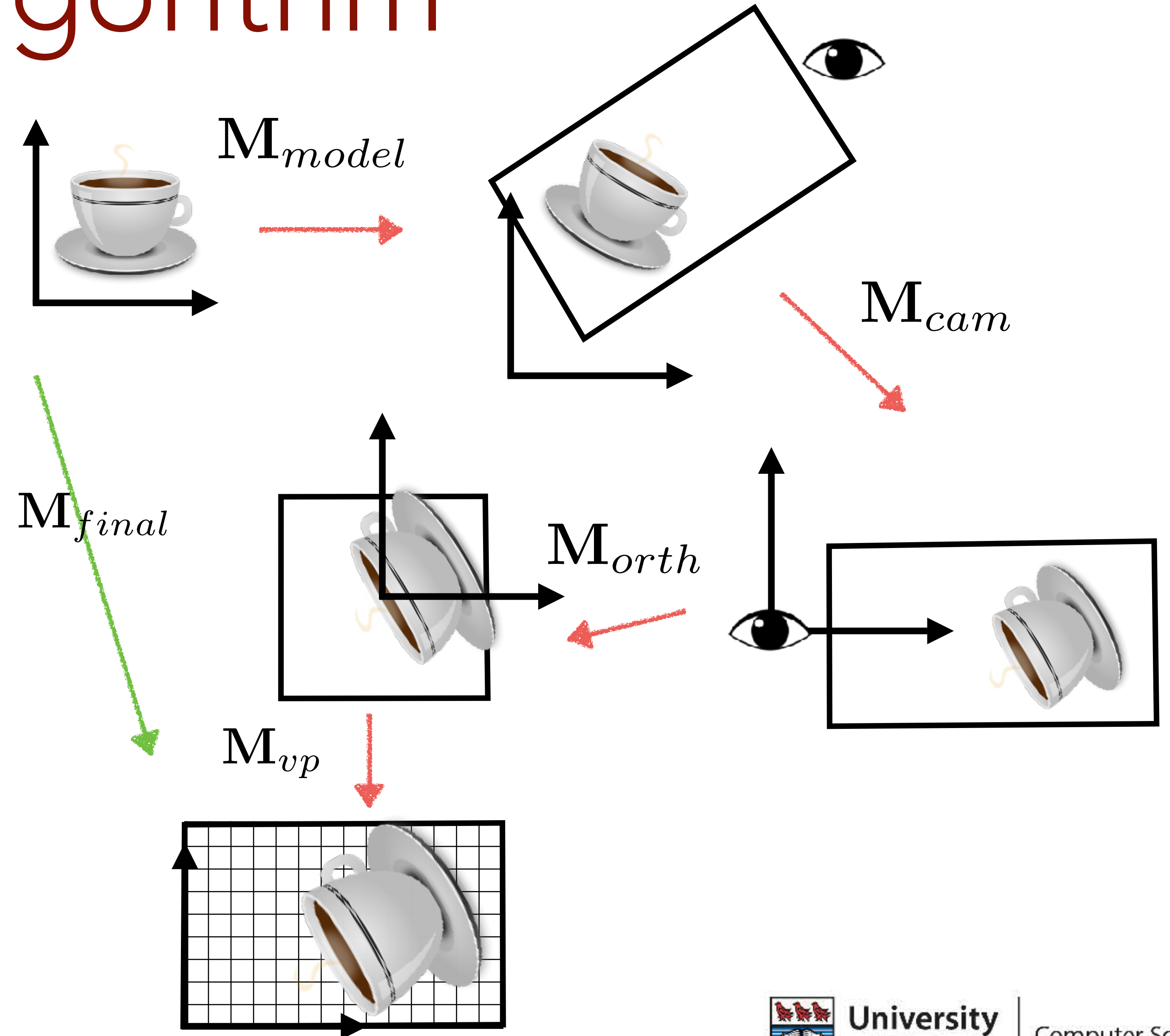
Computer Science

Viewing Transformation



Algorithm

- 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 \mathbf{p} in each primitive of the model
 - $\mathbf{p}_{final} = \mathbf{M}_{final}\mathbf{p}$
 - Rasterize the model

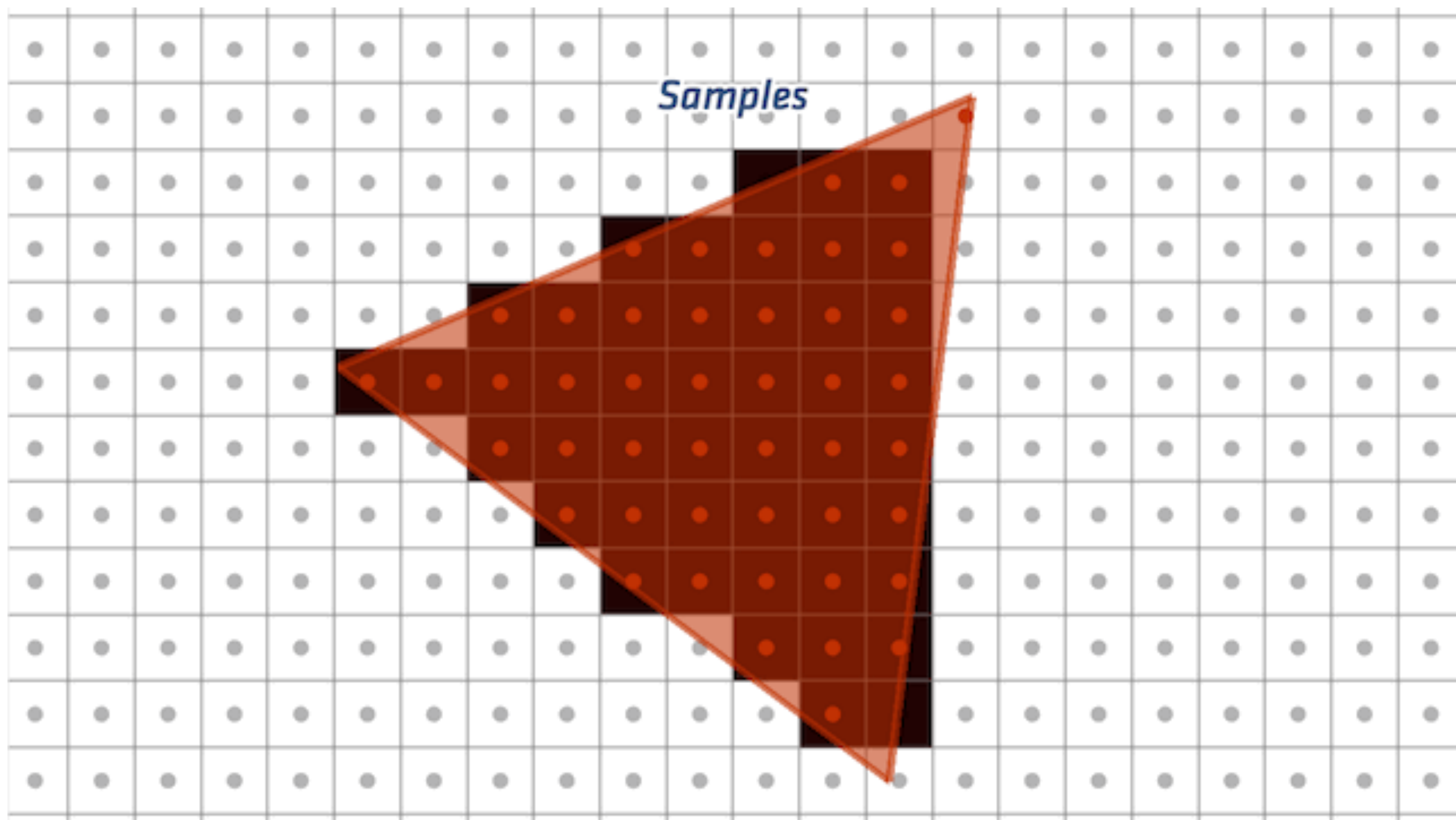


References

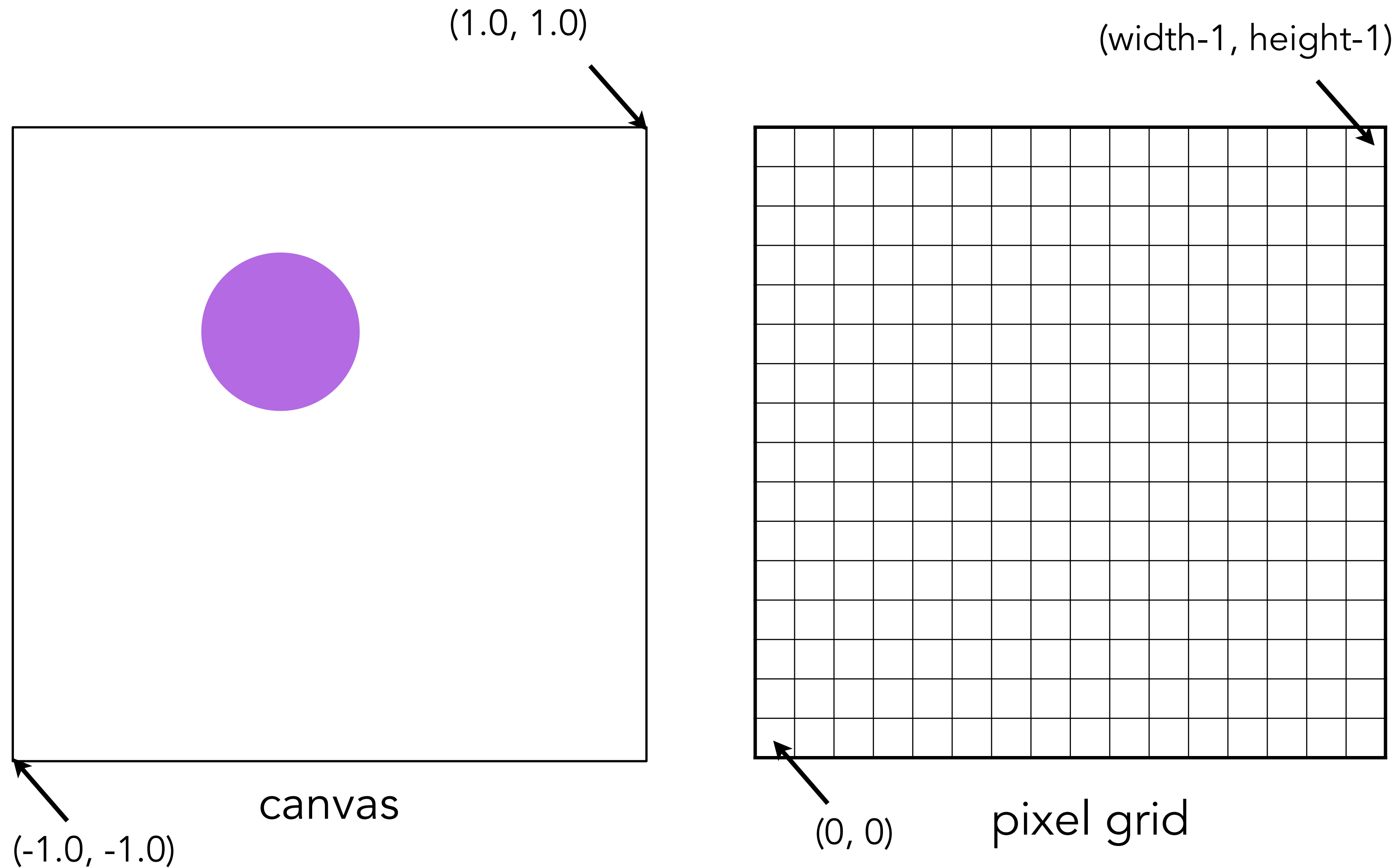
Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 7

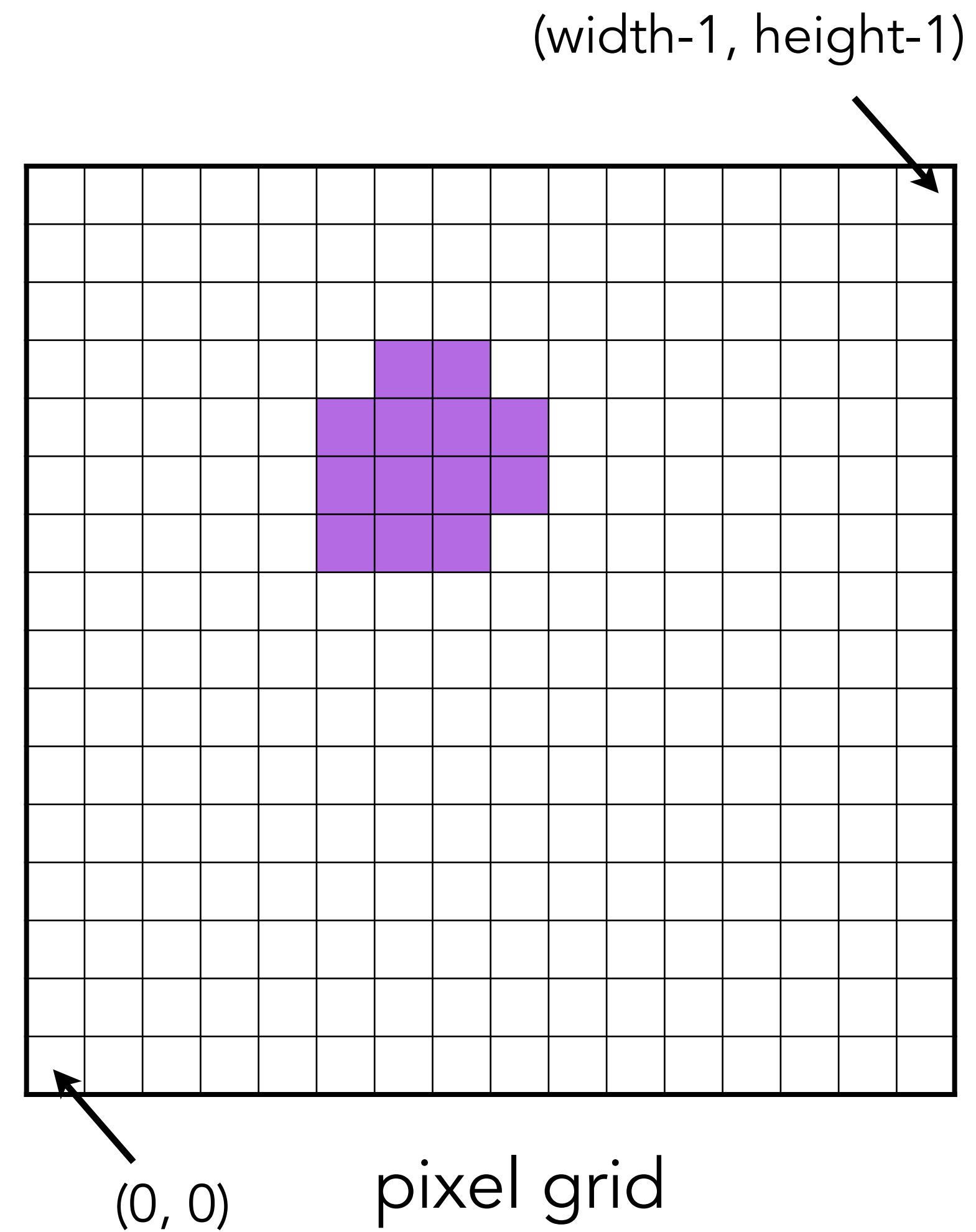
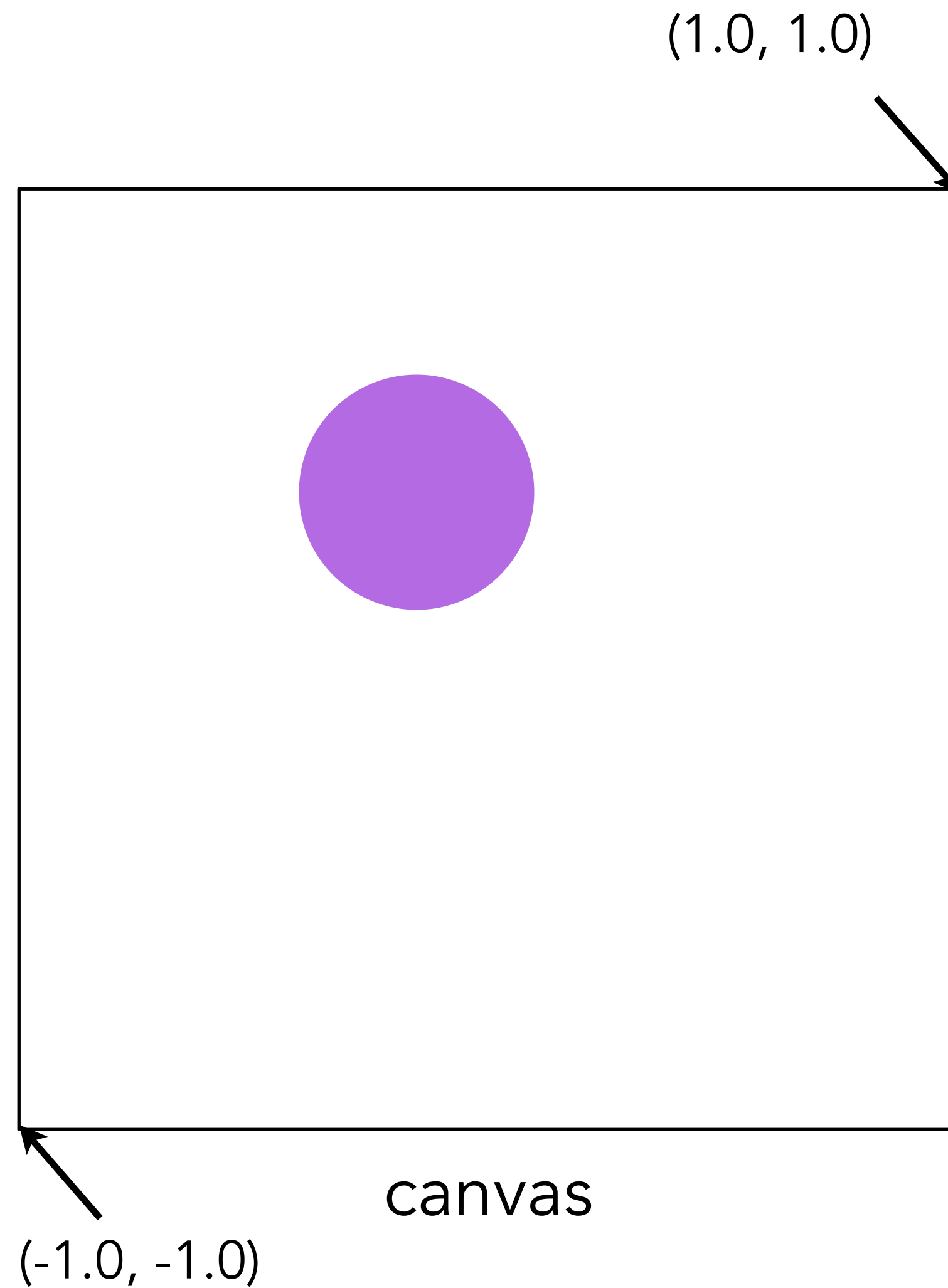
Rasterization



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

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



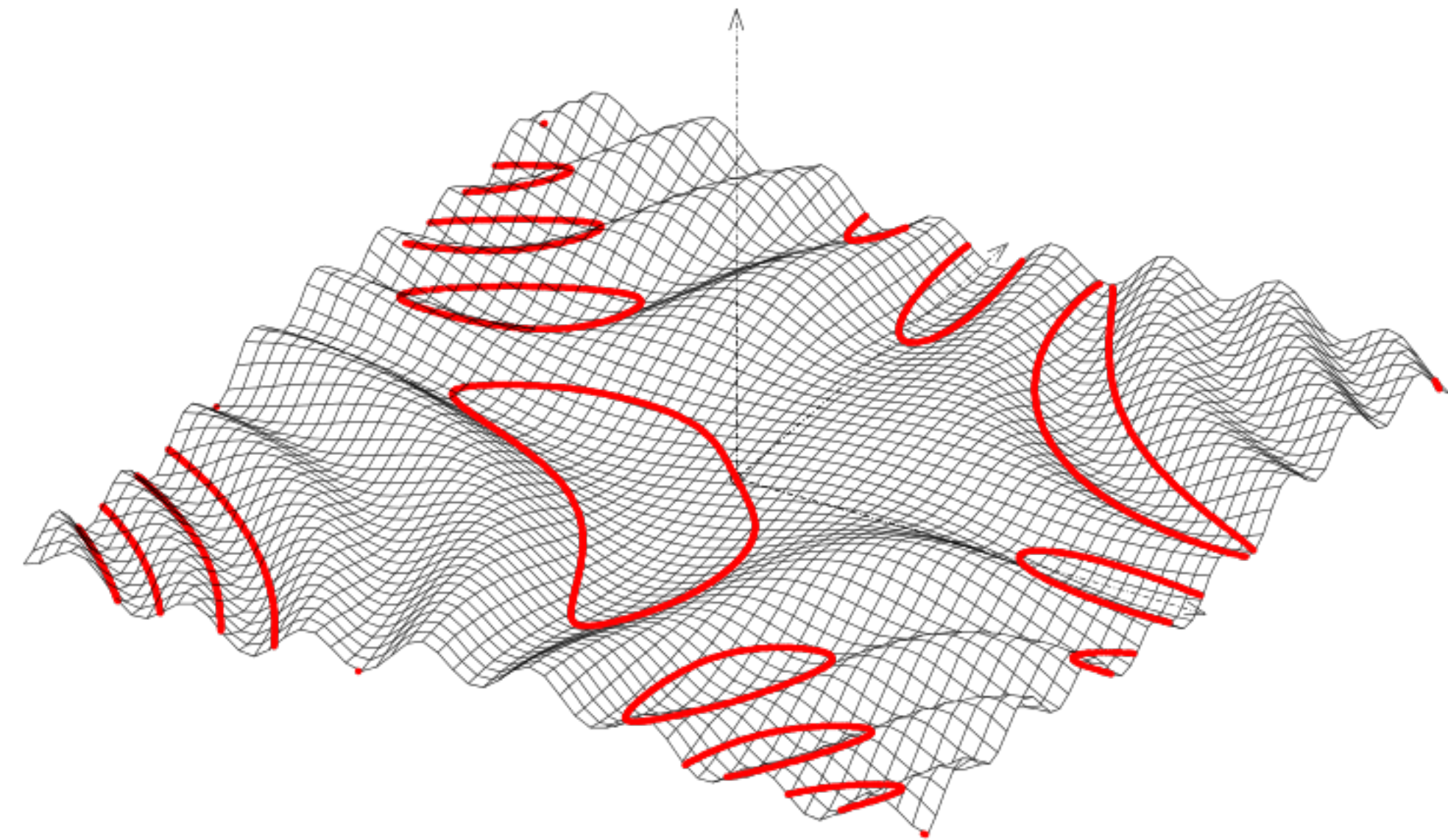
Implicit Geometry Representation

- 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



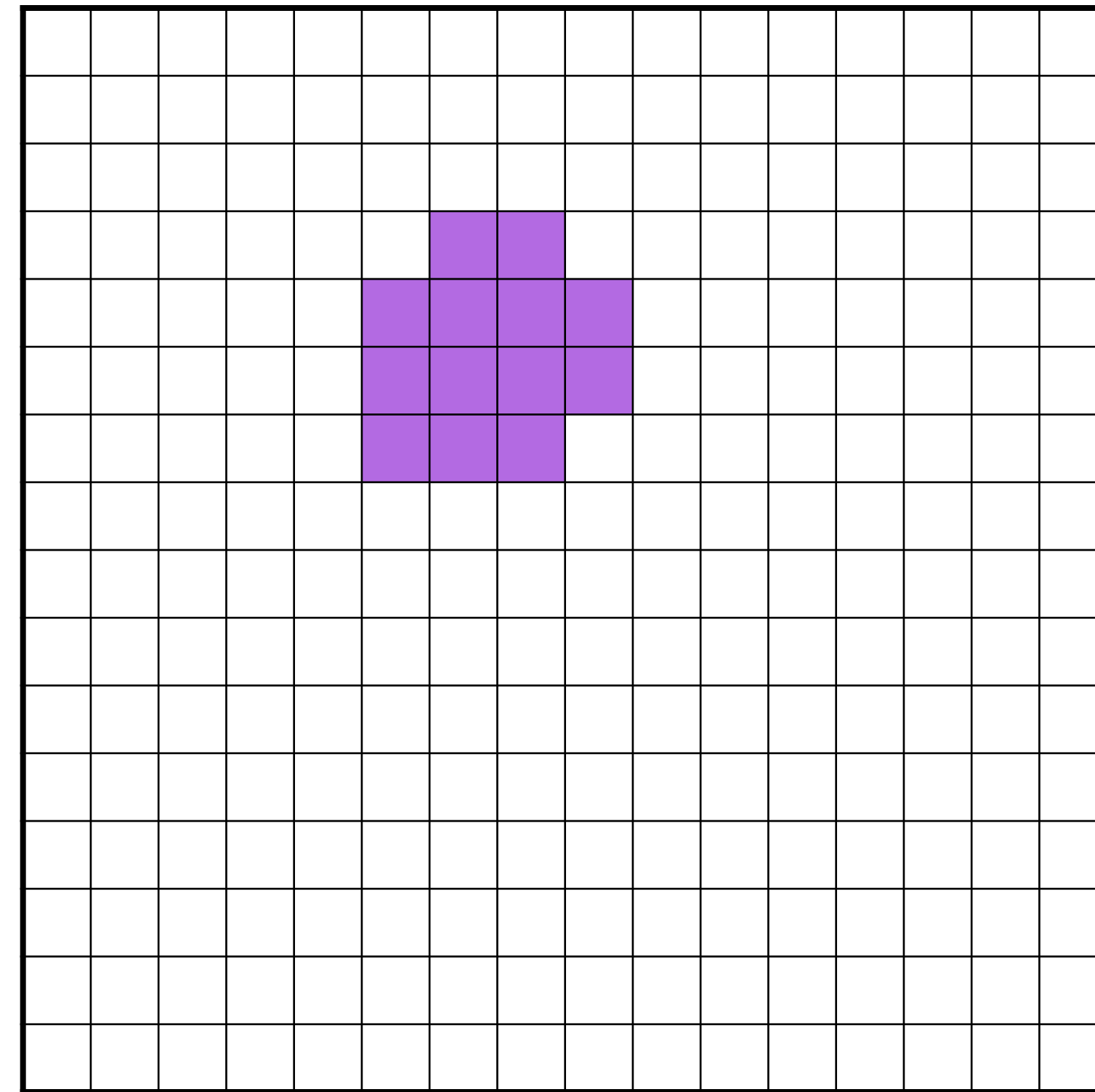
Implicit Rasterization

for all pixels (i,j)

$(x,y) = \text{map_to_canvas}(i,j)$

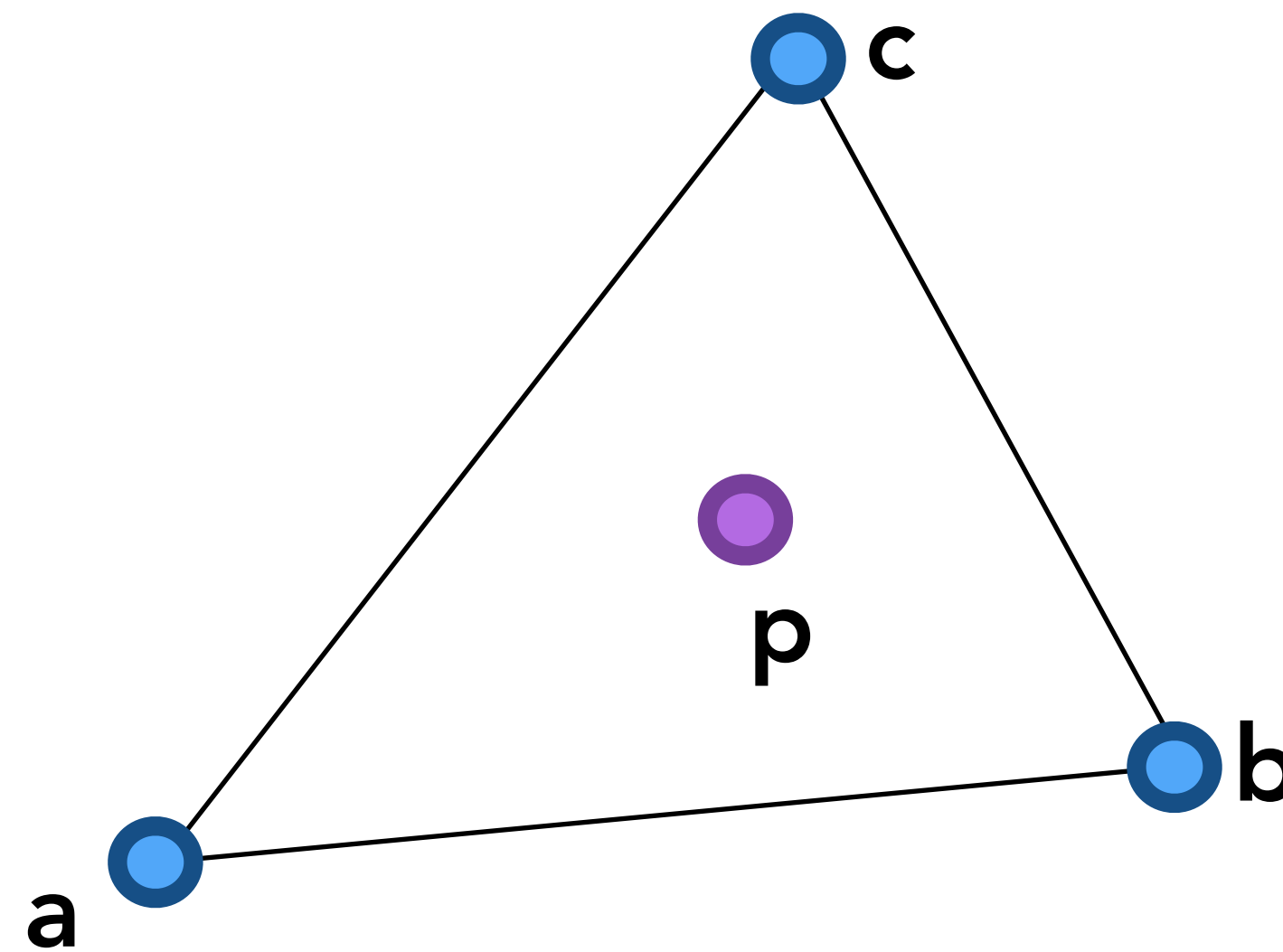
if $F(x,y) < 0$

set_pixel (i,j, color)



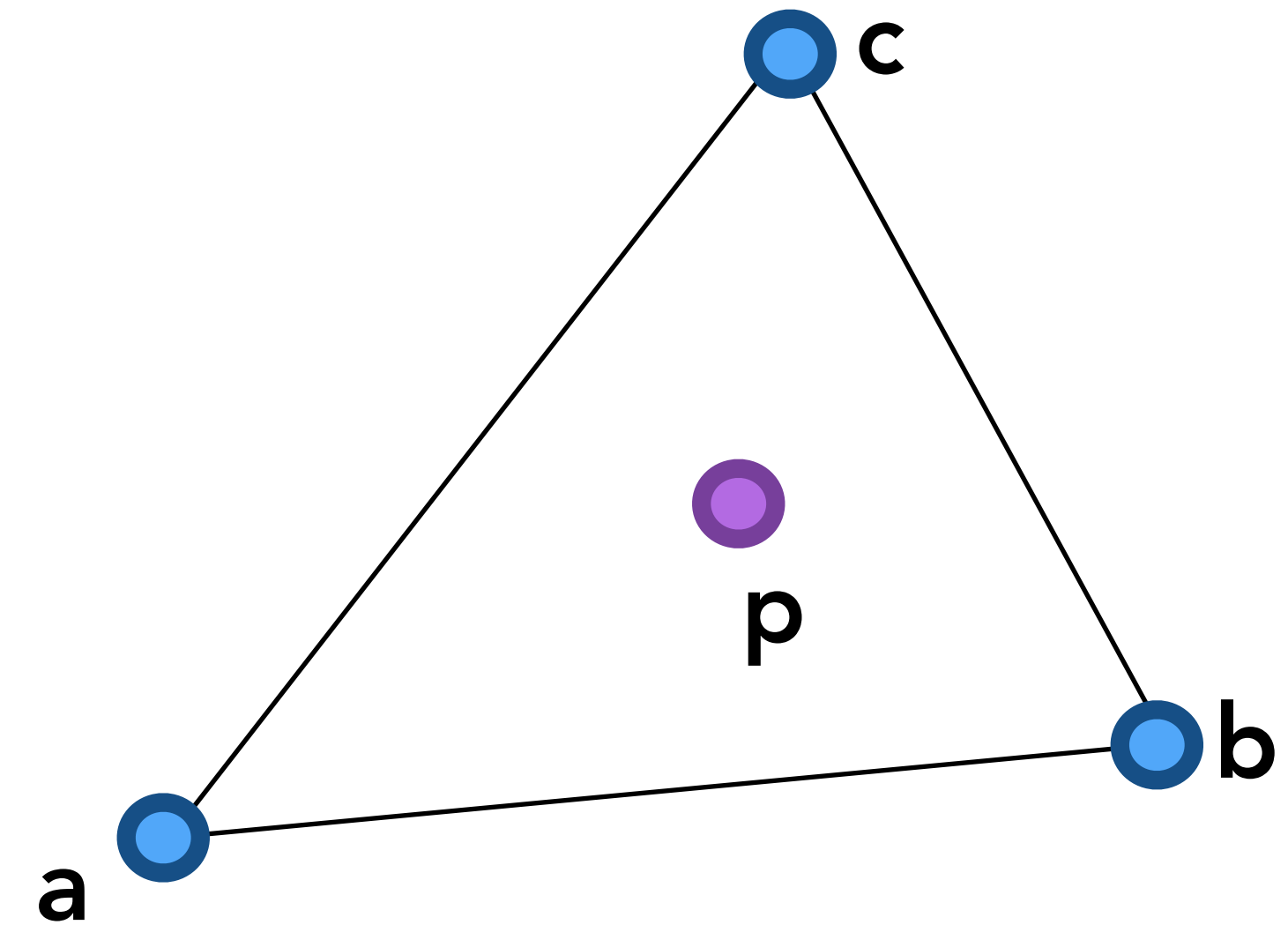
Barycentric Interpolation

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



Barycentric Interpolation

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



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

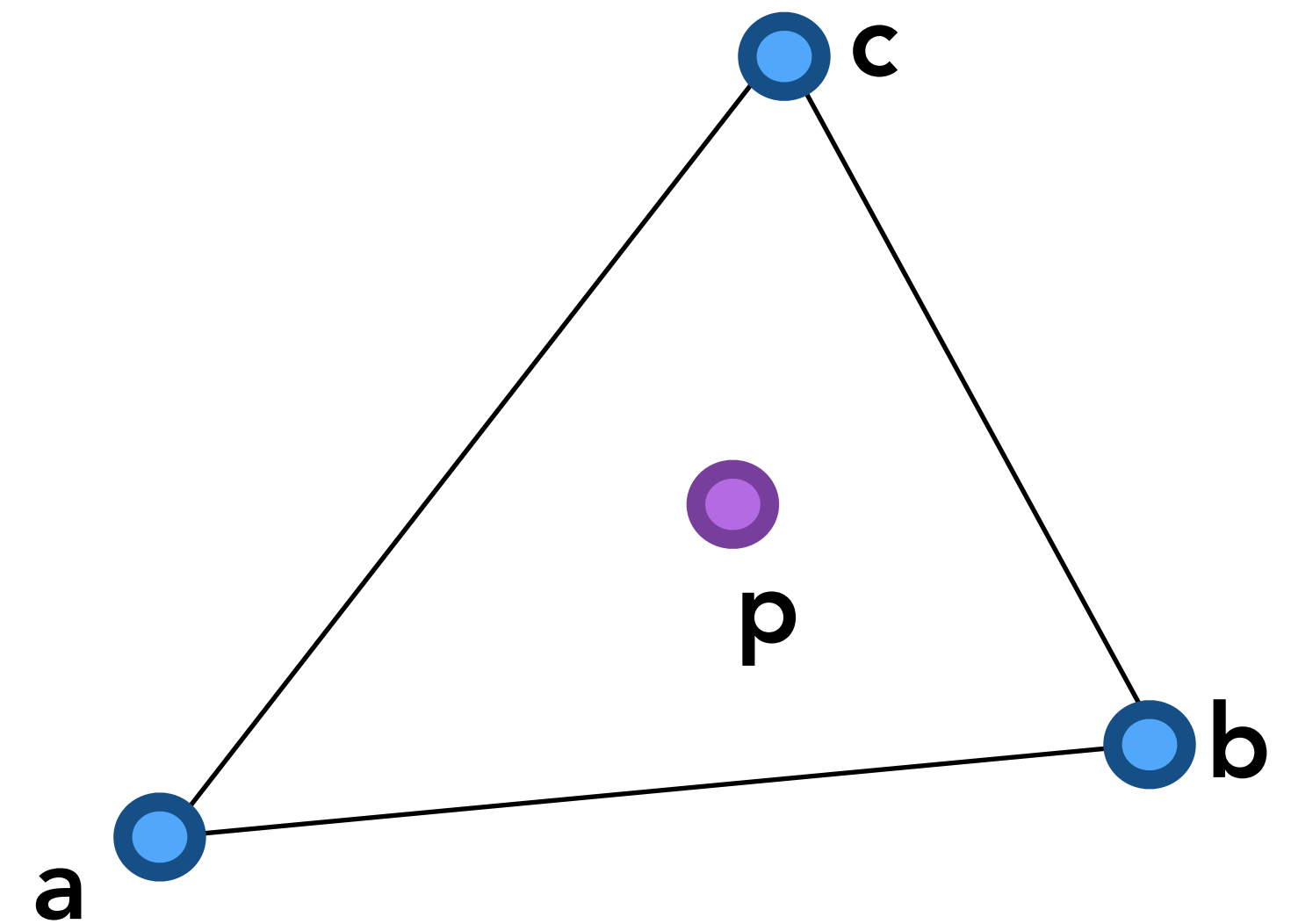
Barycentric Interpolation

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

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

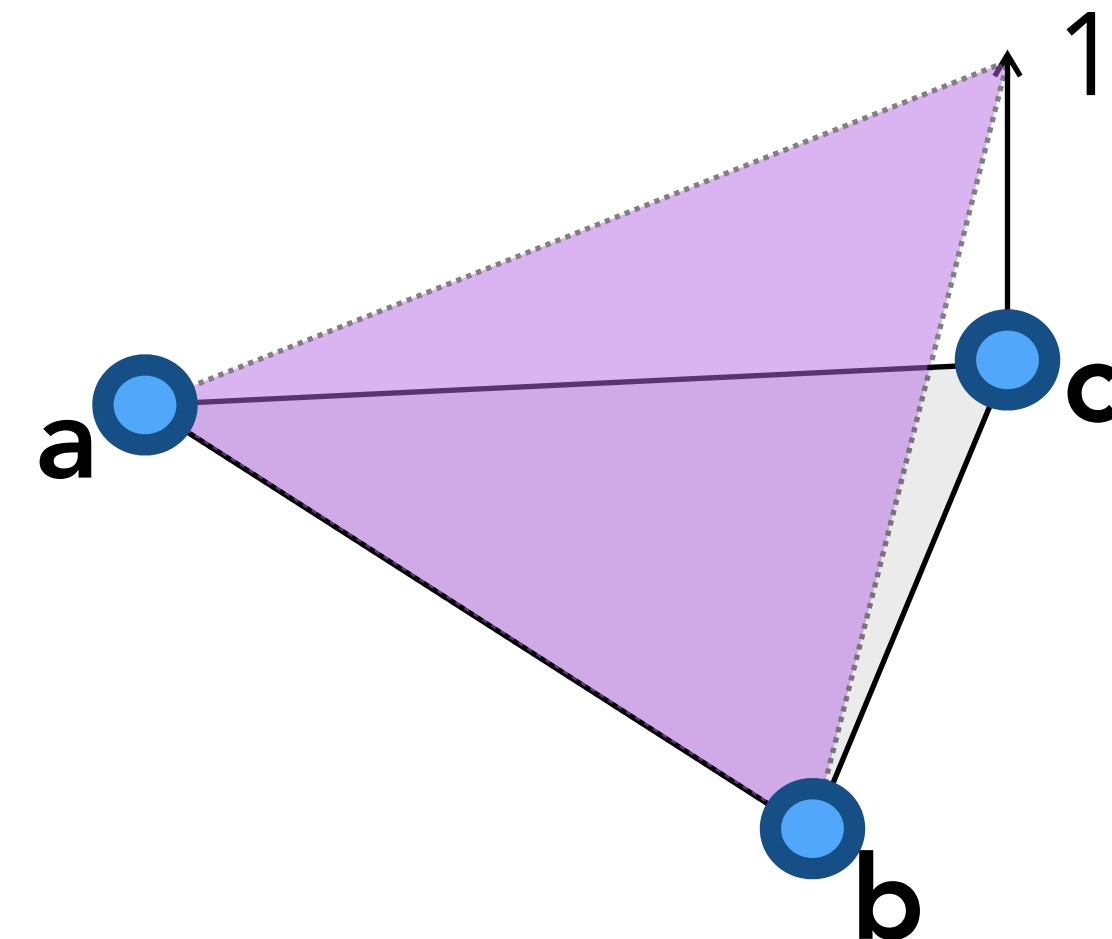
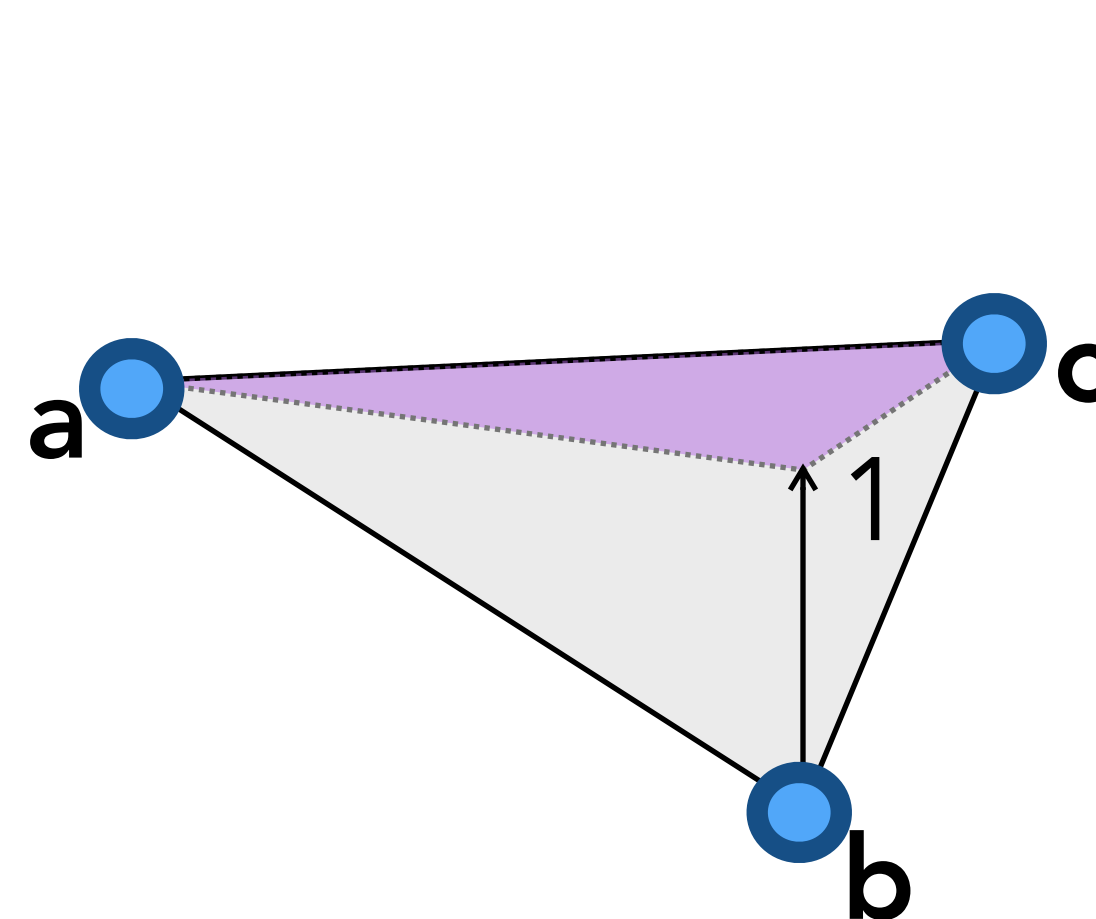
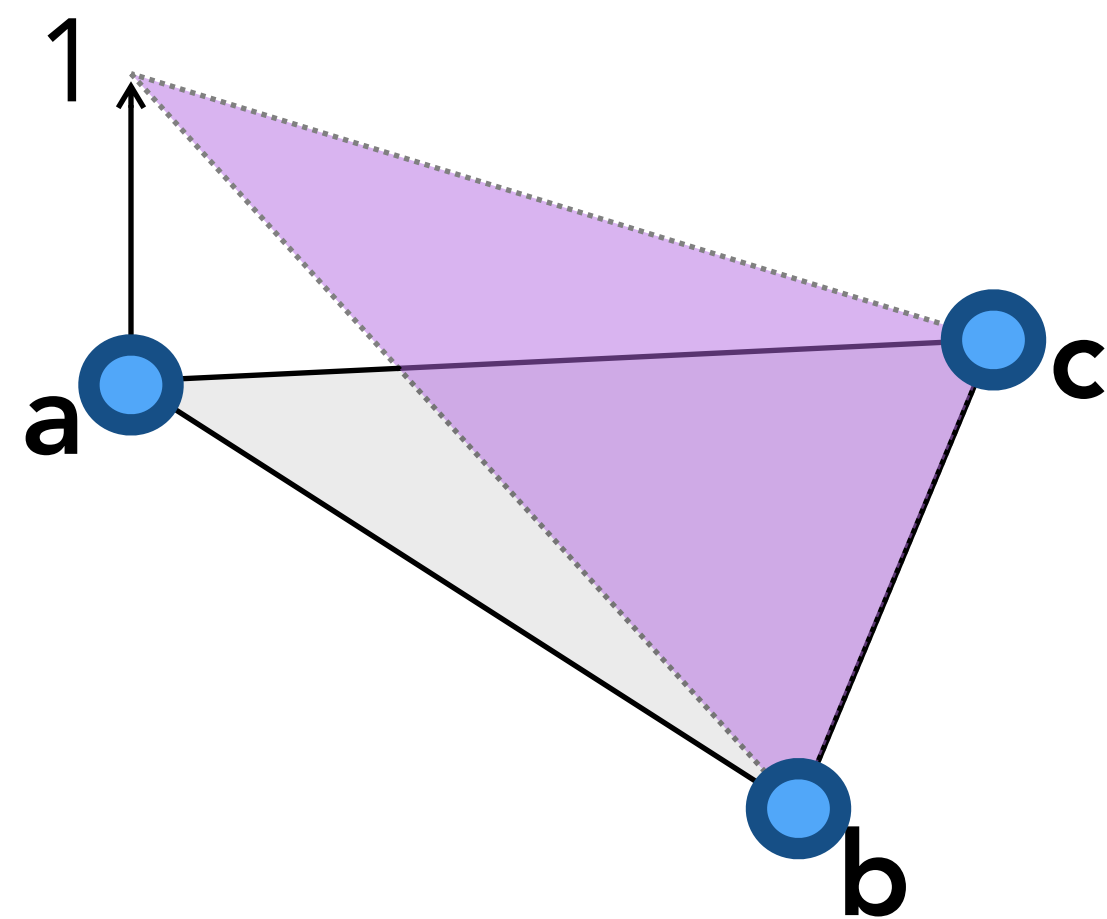
$$\beta(\mathbf{p}) = \frac{\text{area}(\mathbf{p}, \mathbf{c}, \mathbf{a})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$

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



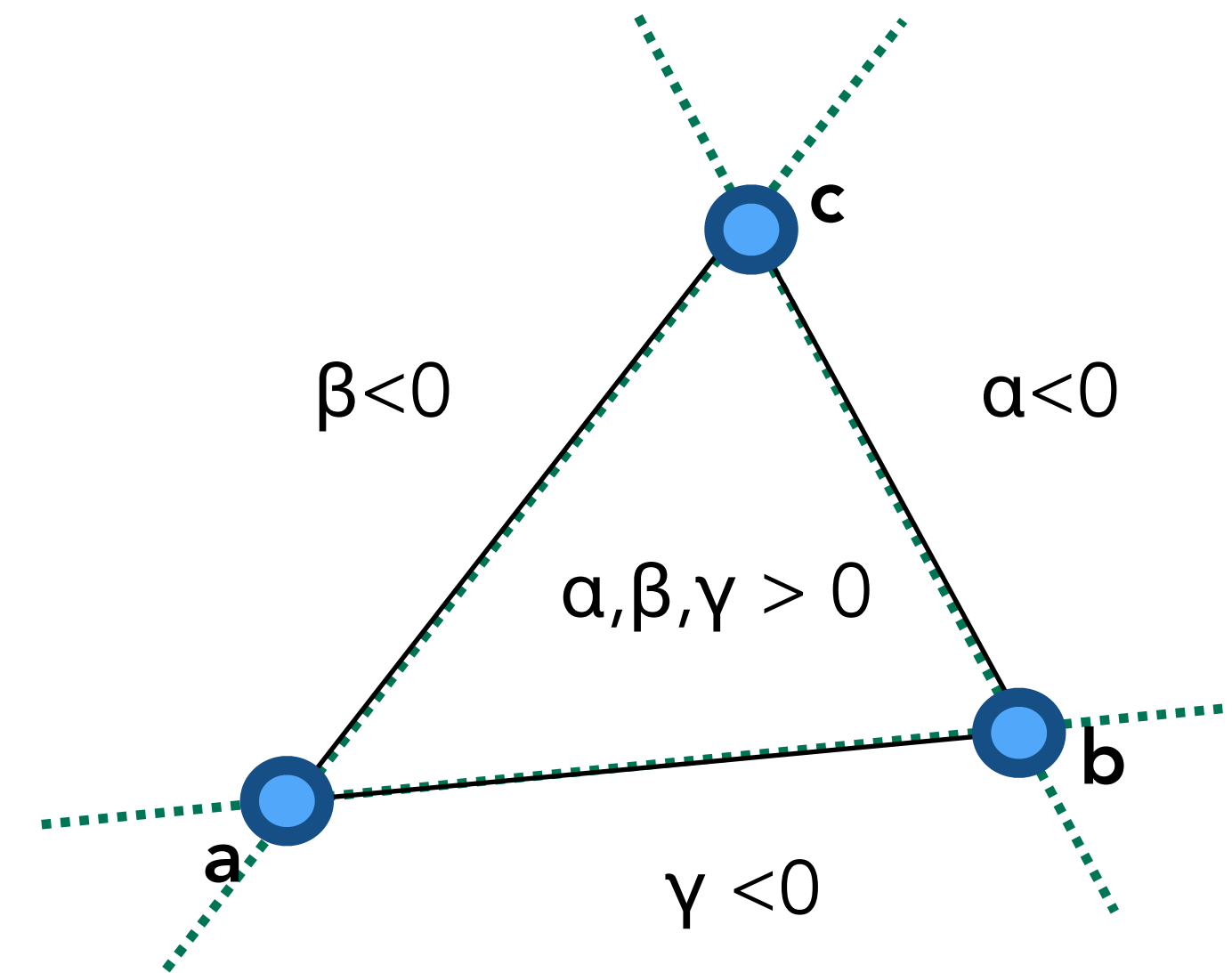
Barycentric Interpolation

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



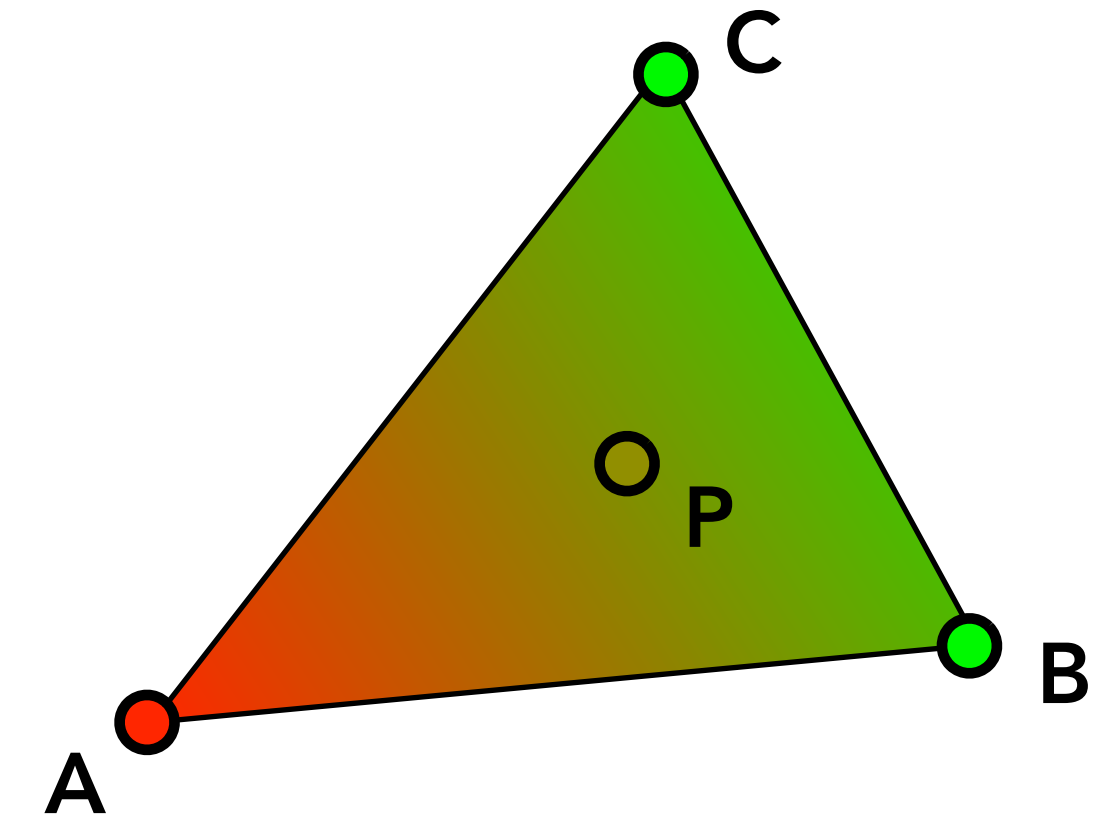
Barycentric Interpolation

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

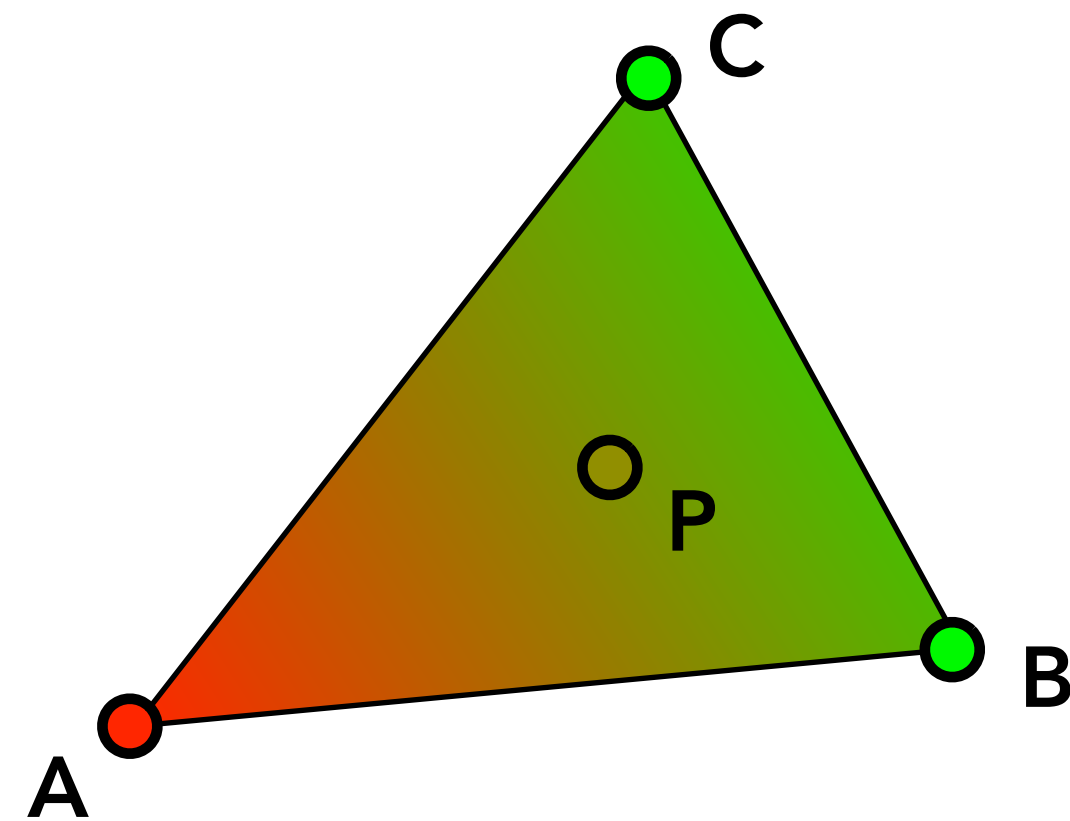


Barycentric Interpolation

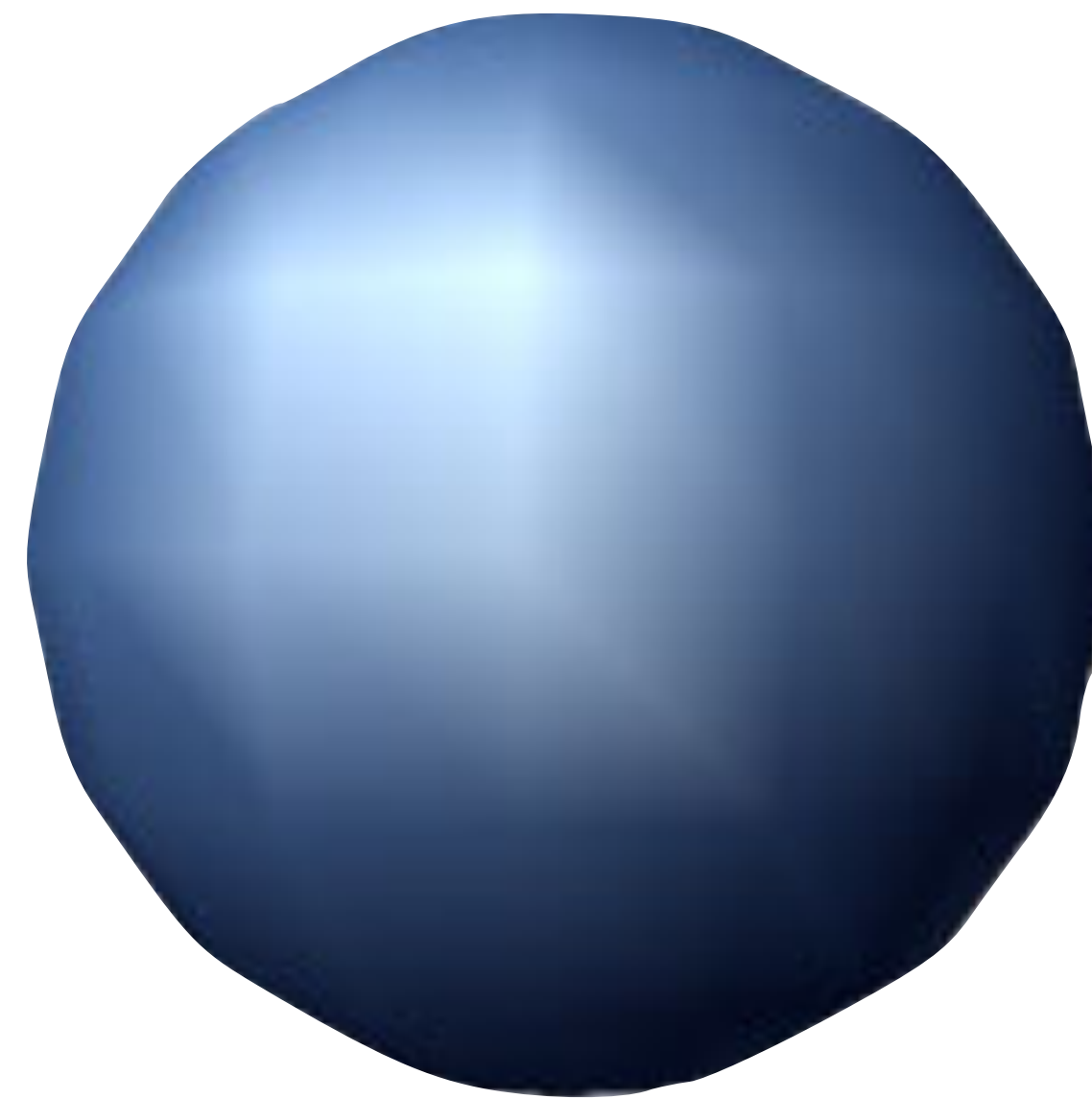
- Barycentric coordinates:
 - $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$ with $\alpha + \beta + \gamma = 1$
 - Unique for non-collinear $\mathbf{a}, \mathbf{b}, \mathbf{c}$
 - Ratio of triangle areas
 - $\alpha(\mathbf{p}), \beta(\mathbf{p}), \gamma(\mathbf{p})$ are linear functions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



Color Interpolation

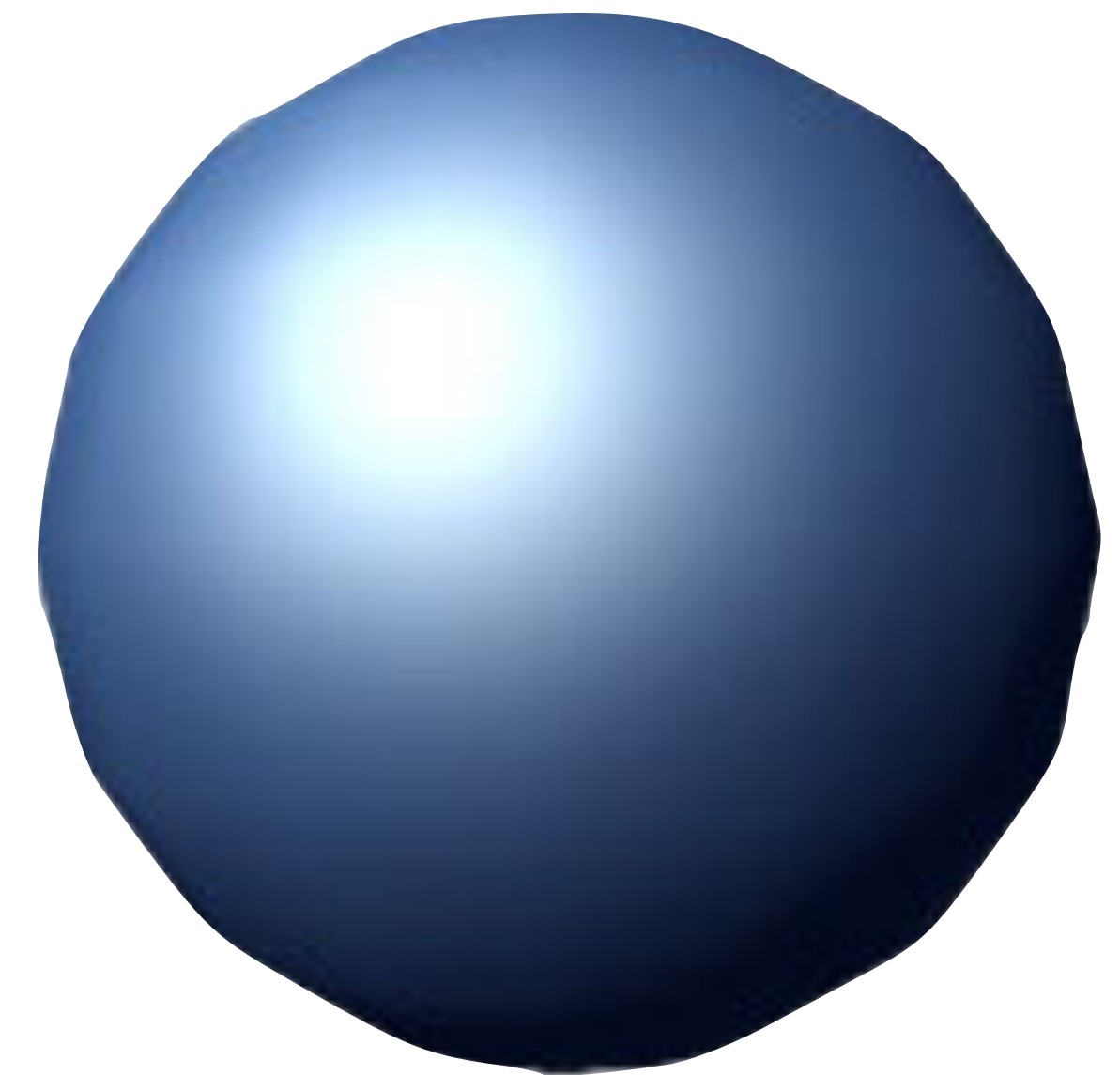


Per-vertex



Evaluate color on vertices,
then interpolates it

Per-pixel



Interpolates positions and normals,
then evaluate color on each pixel

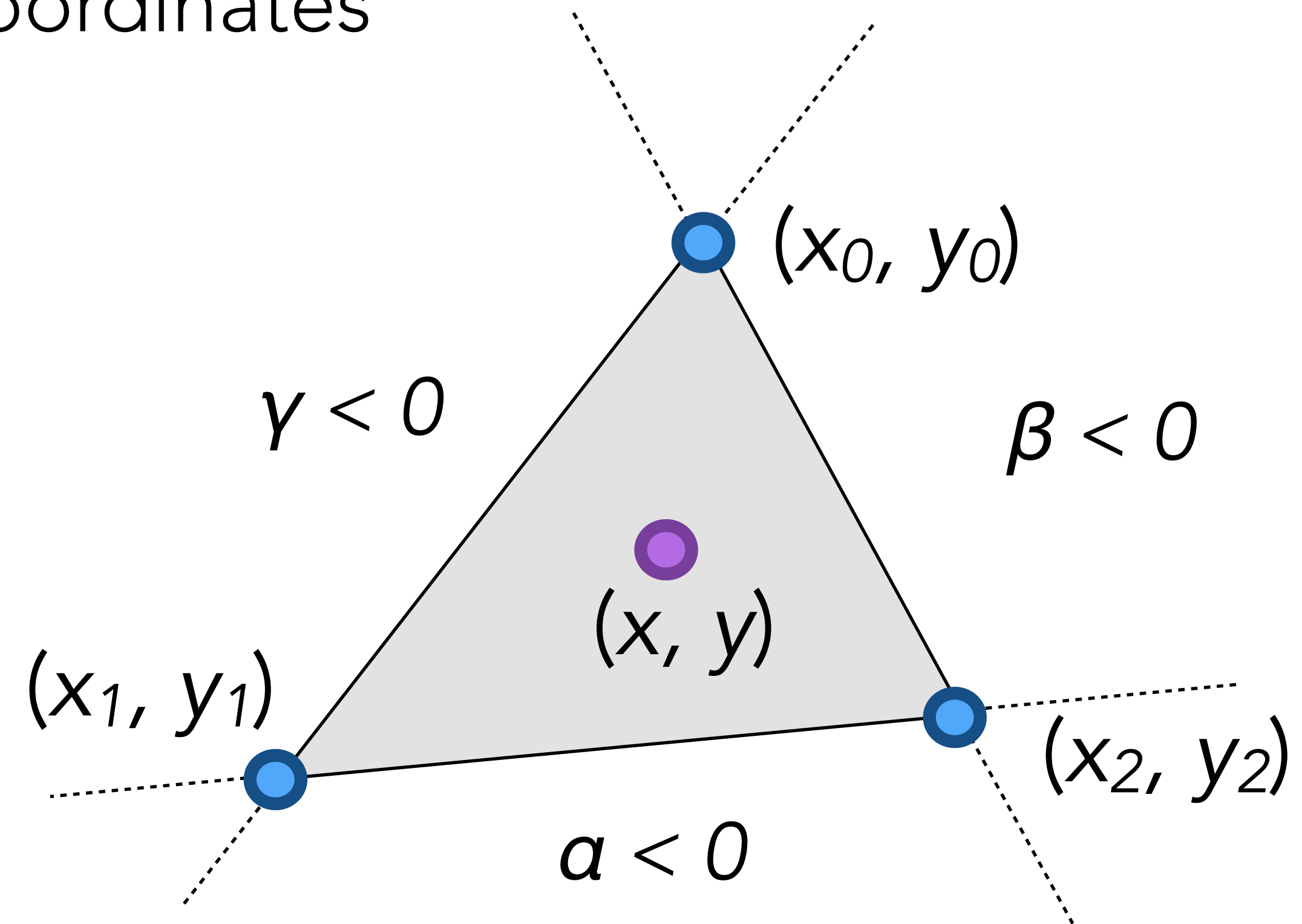
Triangle Rasterization

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

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

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

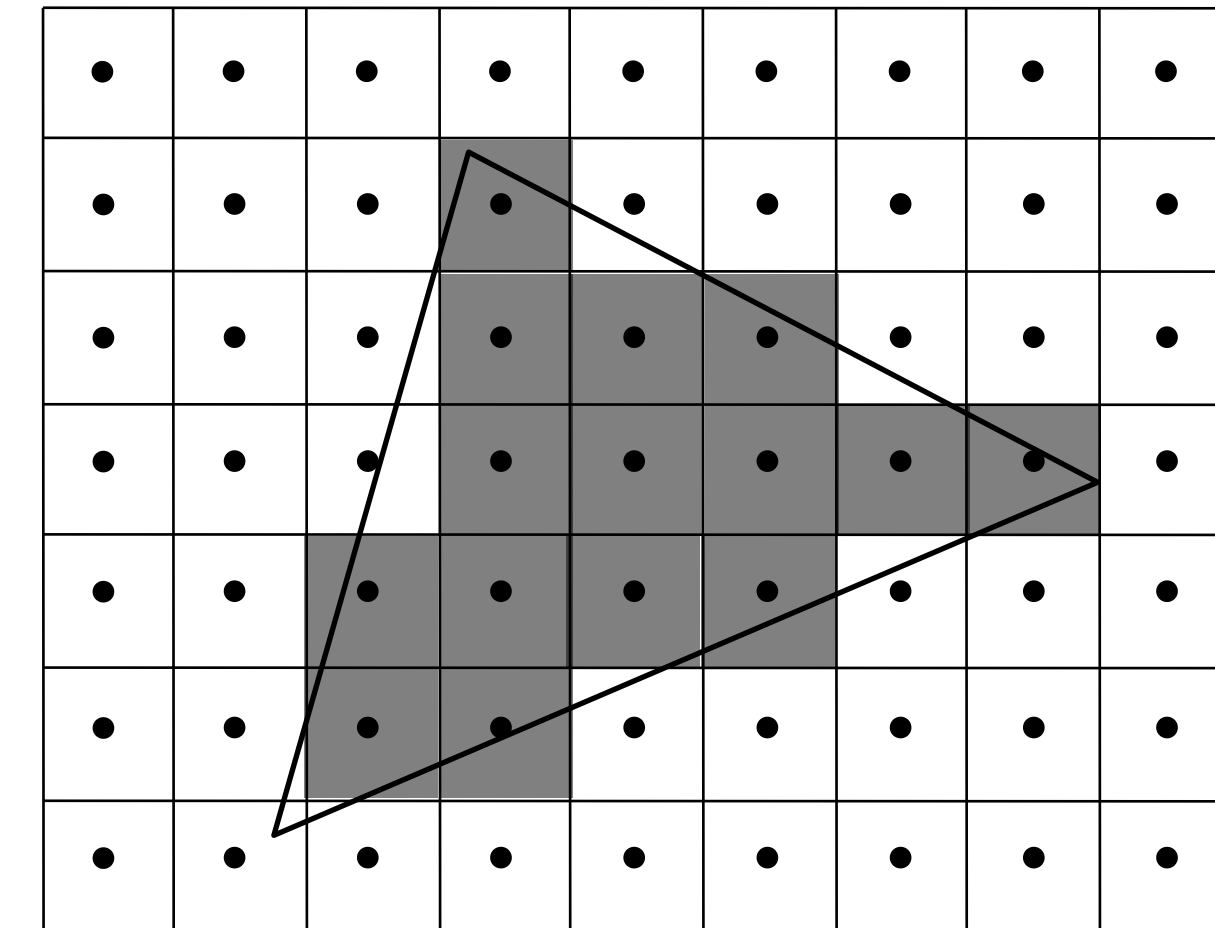
$$a + \beta + \gamma = 1$$



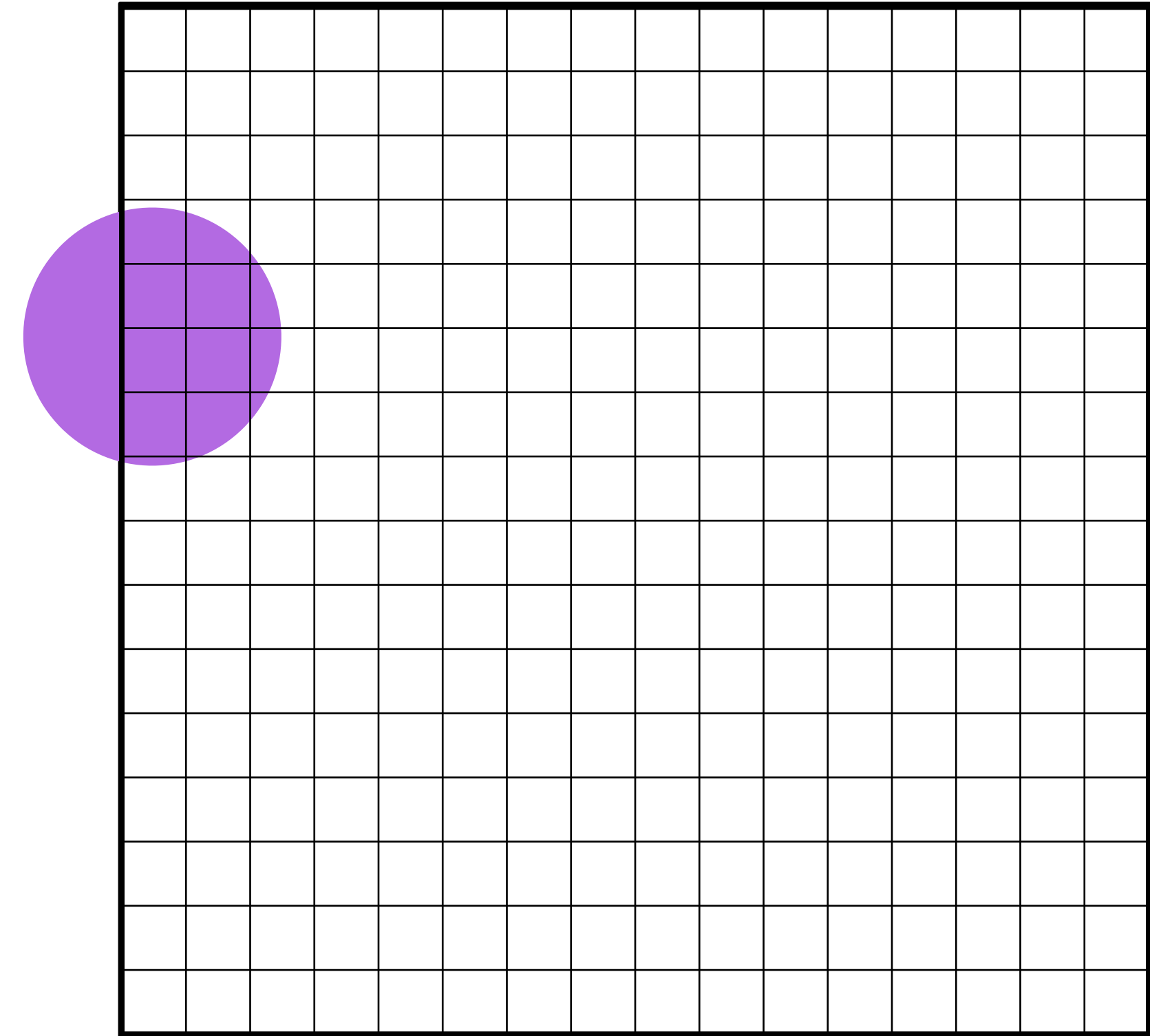
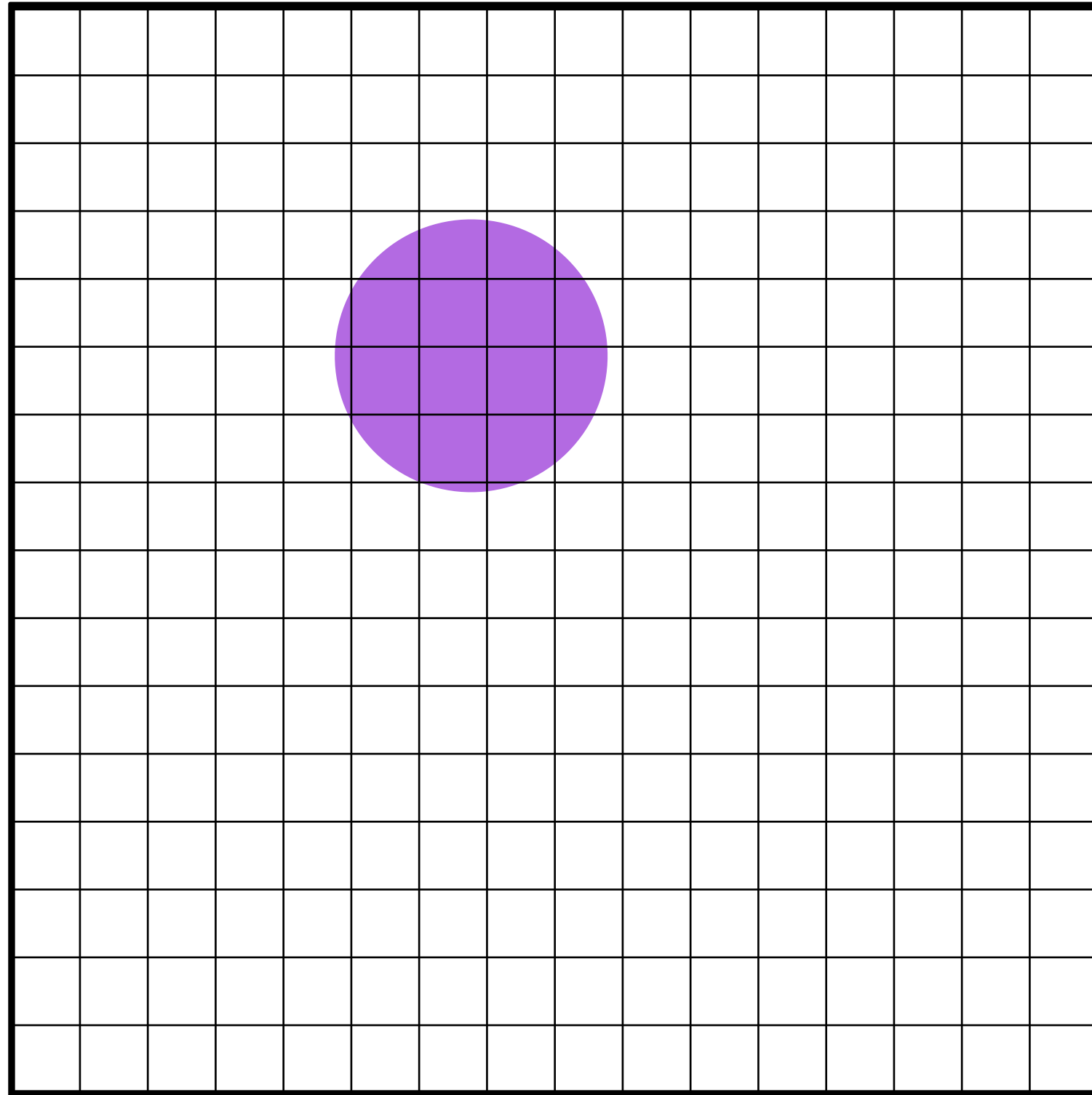
Triangle Rasterization

- 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  $(\alpha, \beta, \gamma)$  for  $(x, y)$ 
    if  $(\alpha \in [0, 1] \text{ and } \beta \in [0, 1] \text{ and } \gamma \in [0, 1])$ 
      set_pixel  $(x, y)$ 
```



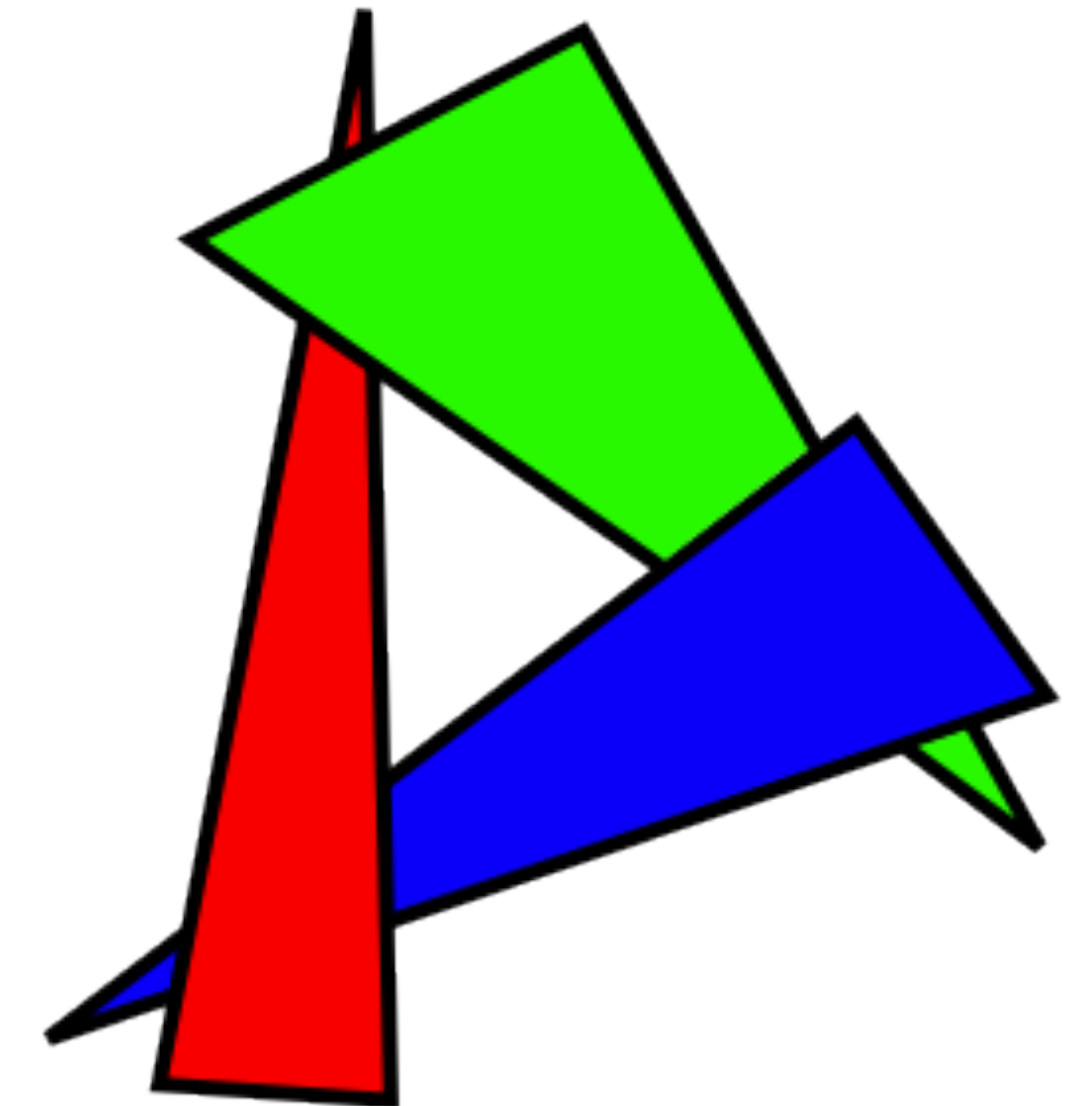
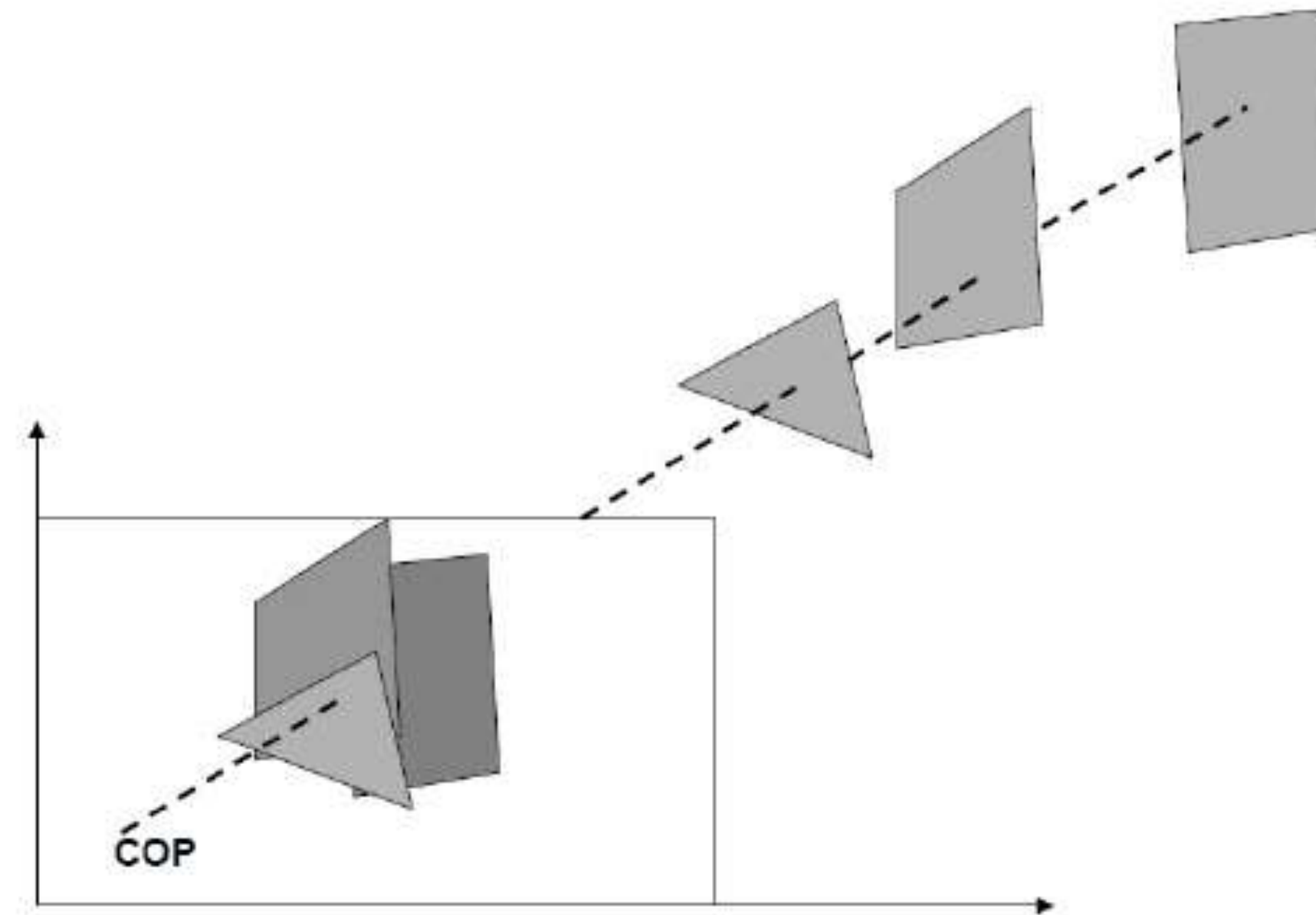
Clipping



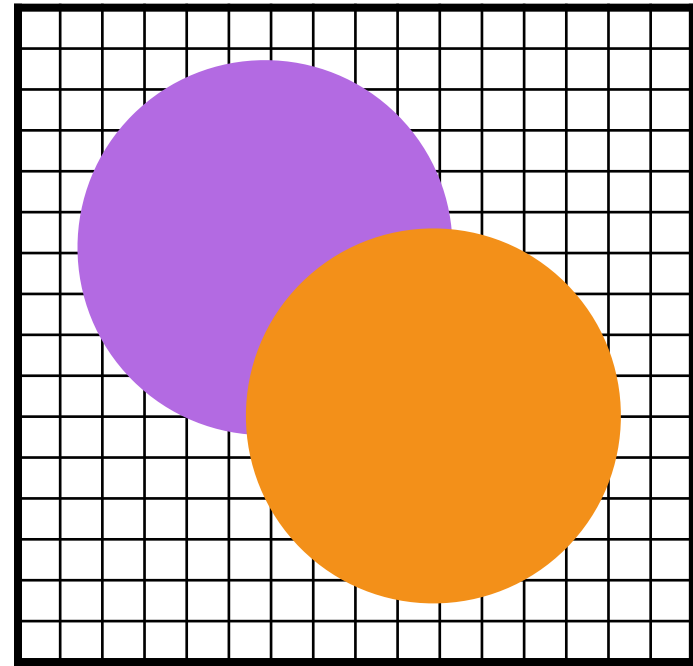
- Ok if you do it brute force
- Care is required if you are explicitly tracing the boundaries

Objects Depth Sorting

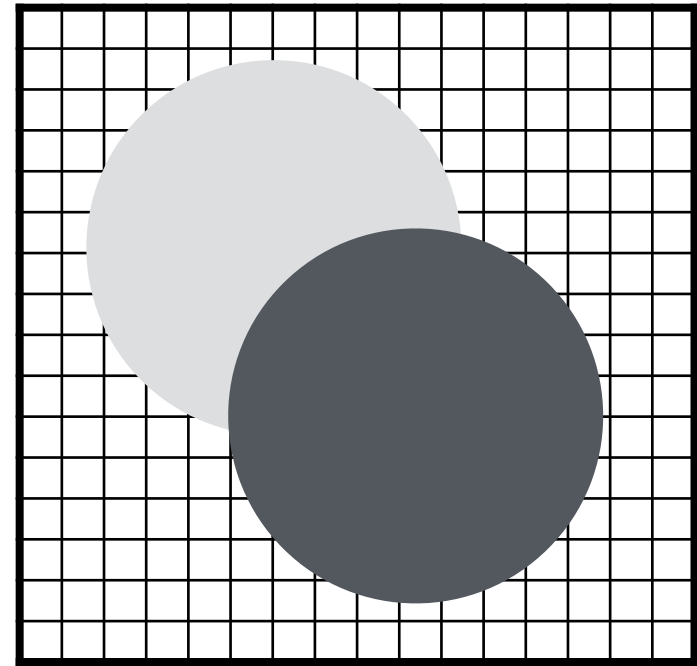
- To handle occlusion, you can sort all the objects in a scene by depth
- This is not always possible!



z-buffering



Image

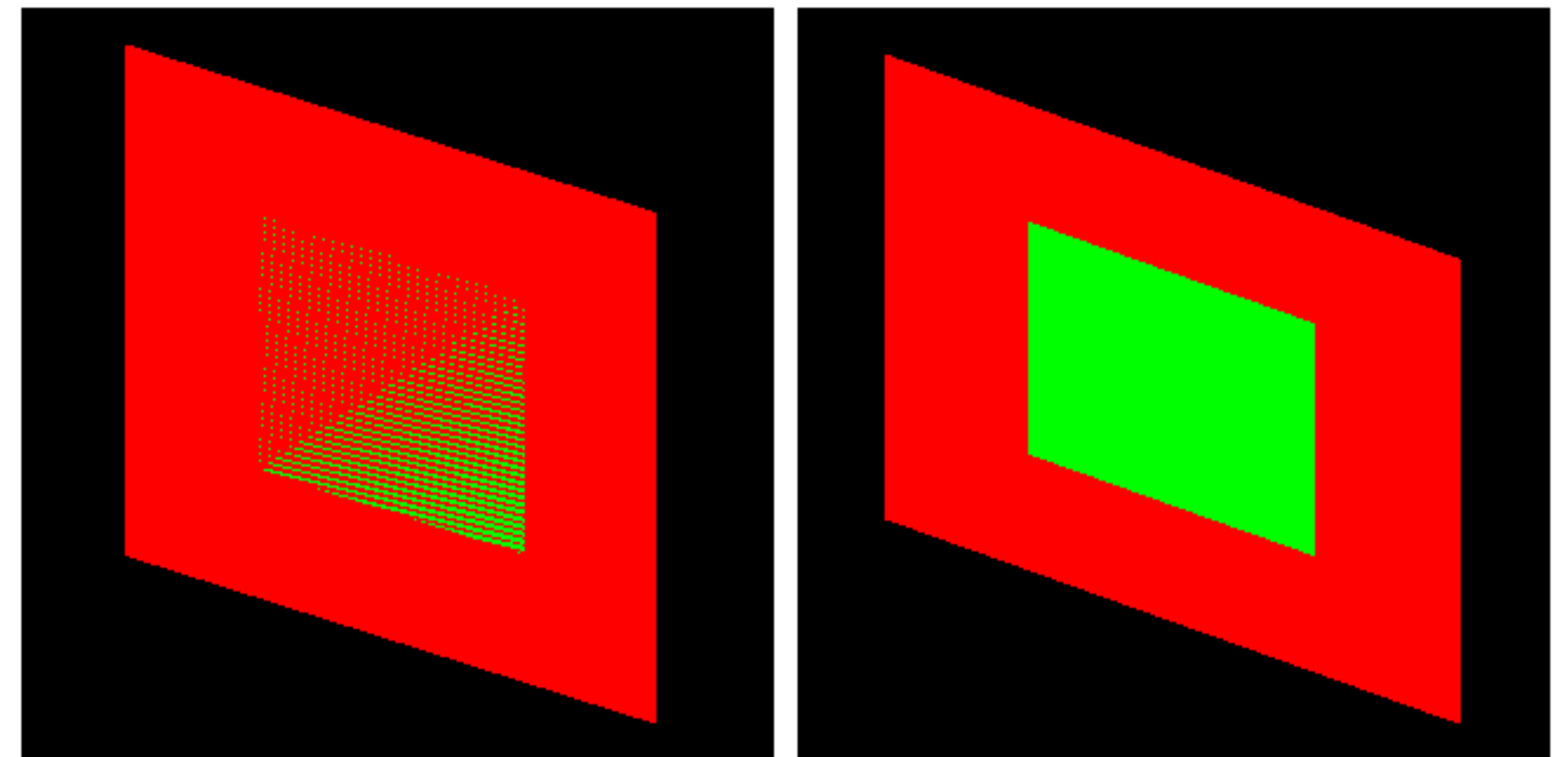
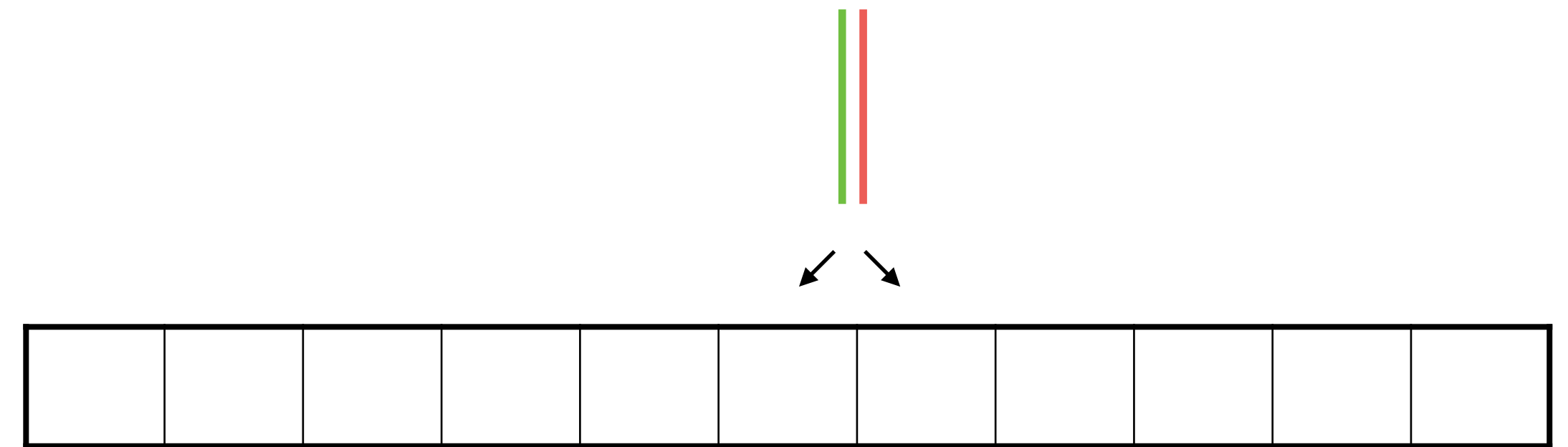


Depth (z)

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



Super Sampling Anti-Aliasing



Non-antialiased type

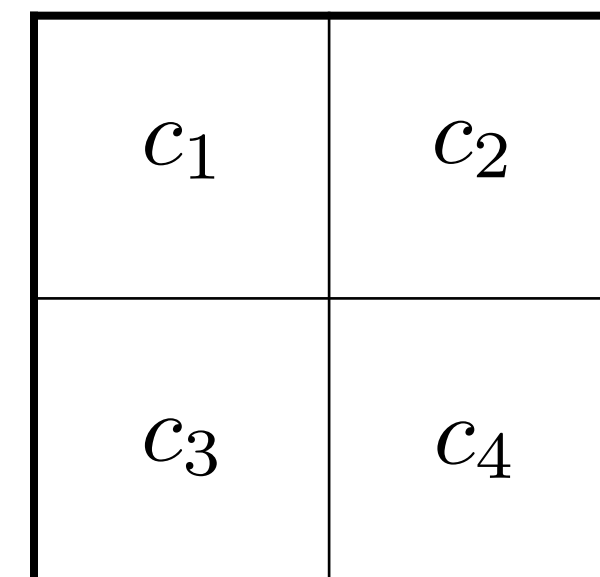


Antialiased type



Enlarged portion of type

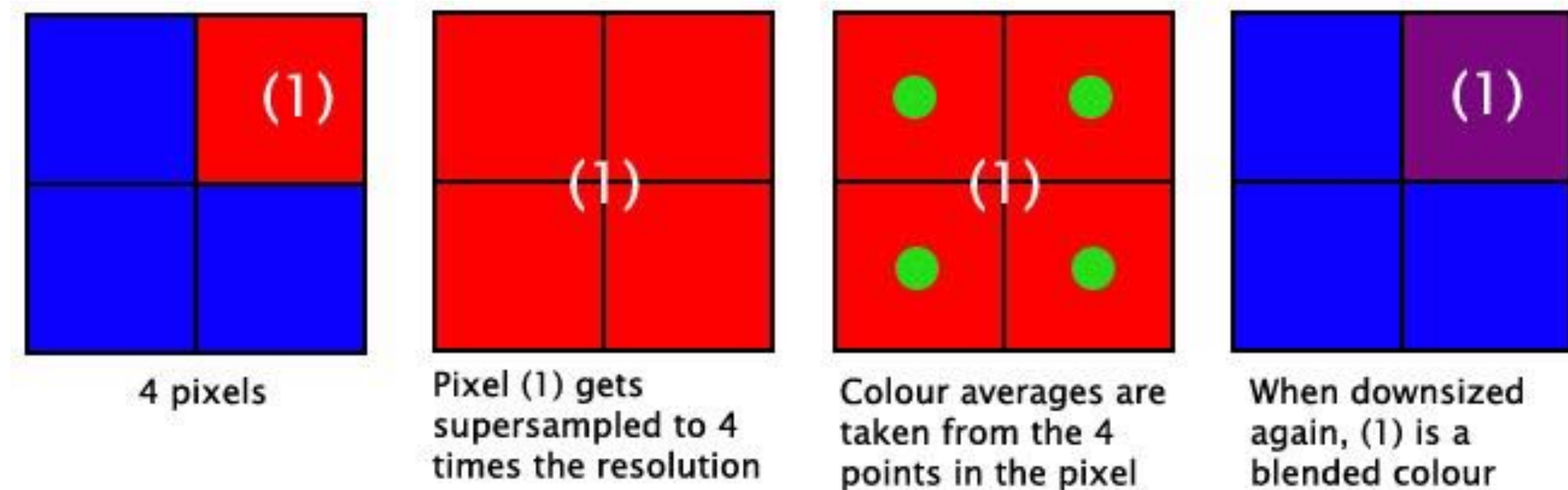
- Render $n \times n$ pixels instead of one
- Assign the average to the pixel



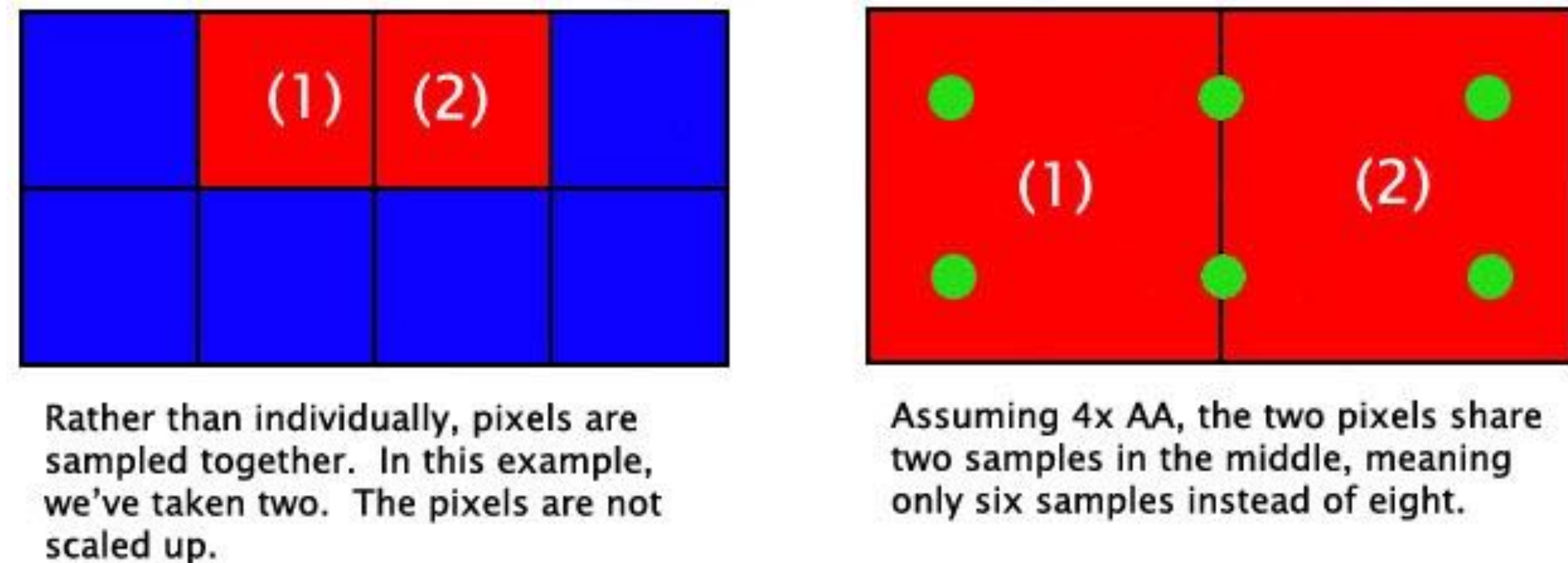
$$\frac{c_1 + c_2 + c_3 + c_4}{4}$$

Many different names and variants

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



MSAA



References

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

Chapter 8