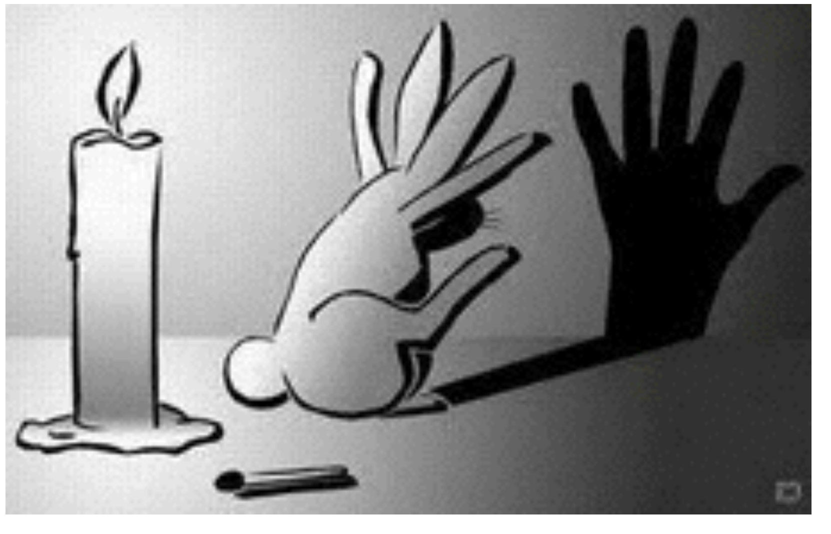
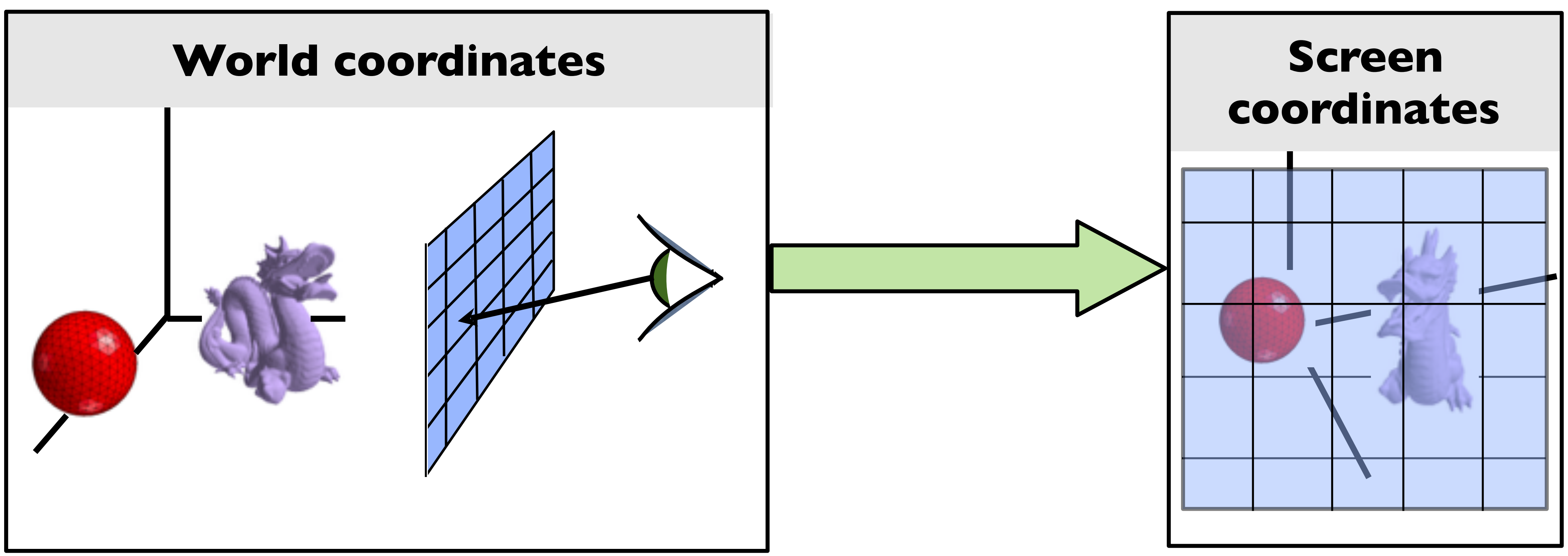
Perspective Transformations

Viewing Transformations

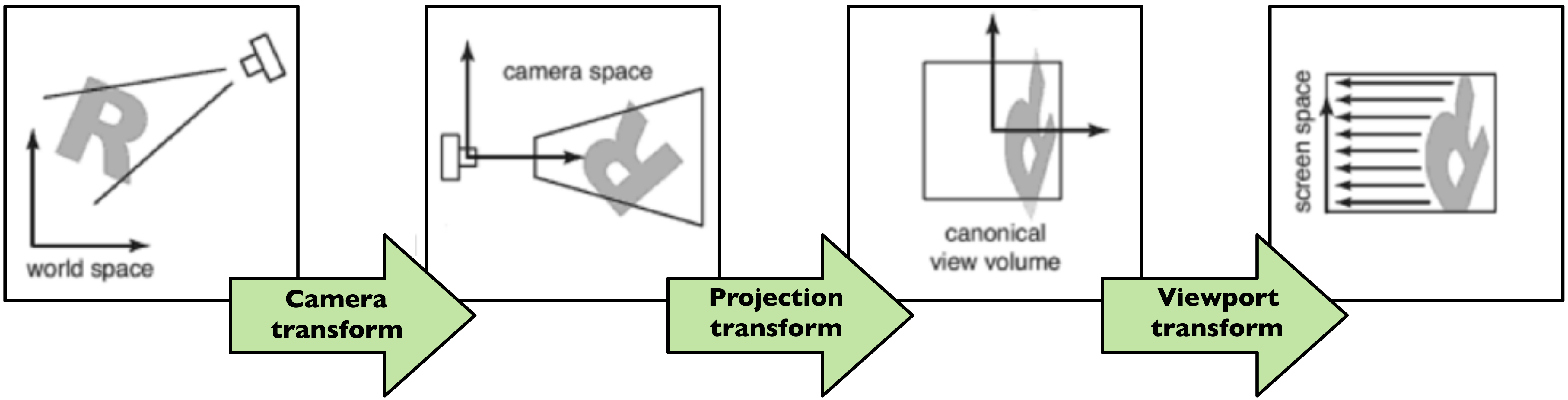


Viewing transformations

- Move objects from their 3D locations to their positions in a 2D view



Decomposition of viewing transforms



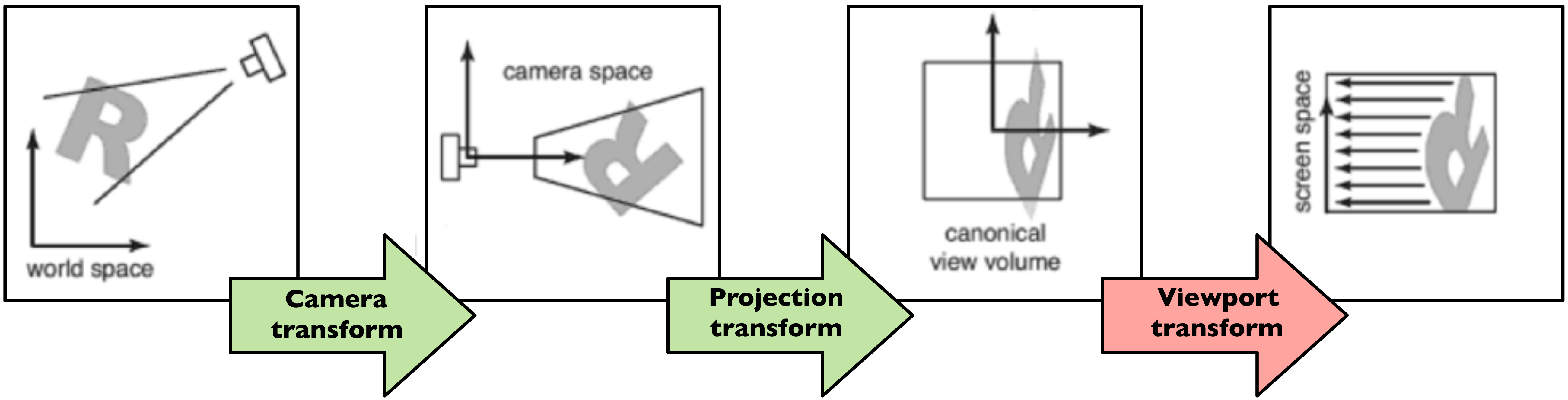
- rigid body transformation
- place camera at origin

- x, y, z in [-1,1]
- depends on type of projection

- map to pixel coordinates

Viewing transforms depend on: camera position and orientation, type of projection, field of view, image resolution

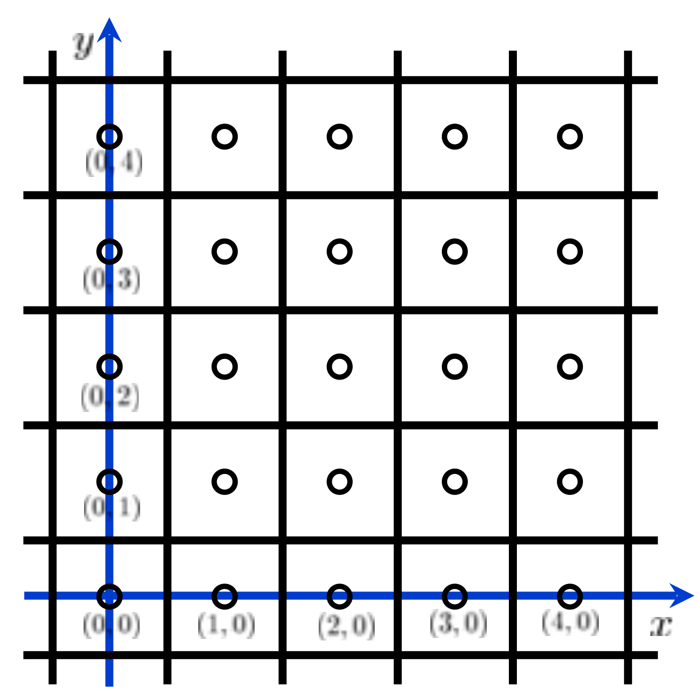
Viewport transform



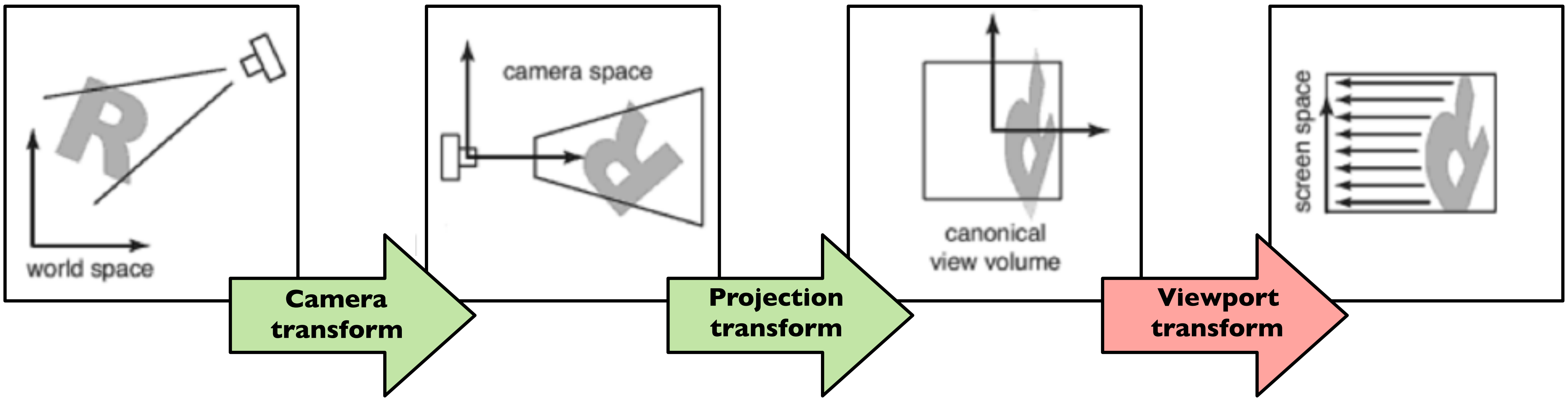
$$(x, y, z) \rightarrow (x', y', z')$$

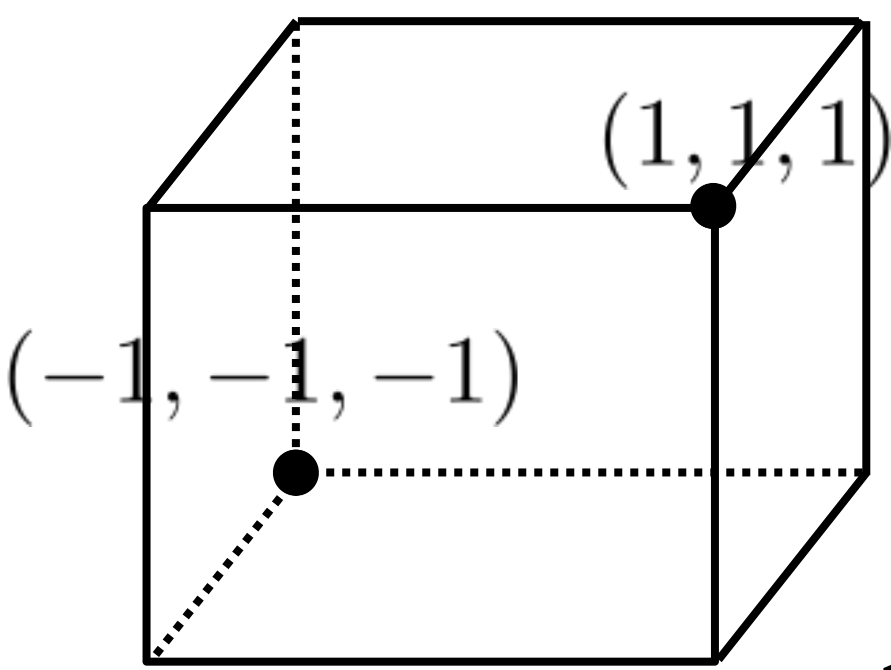
$$(x, y, z) \in [-1, 1]^3$$

$$x' \in [-.5, n_x - .5]$$
$$y' \in [-.5, n_y - .5]$$



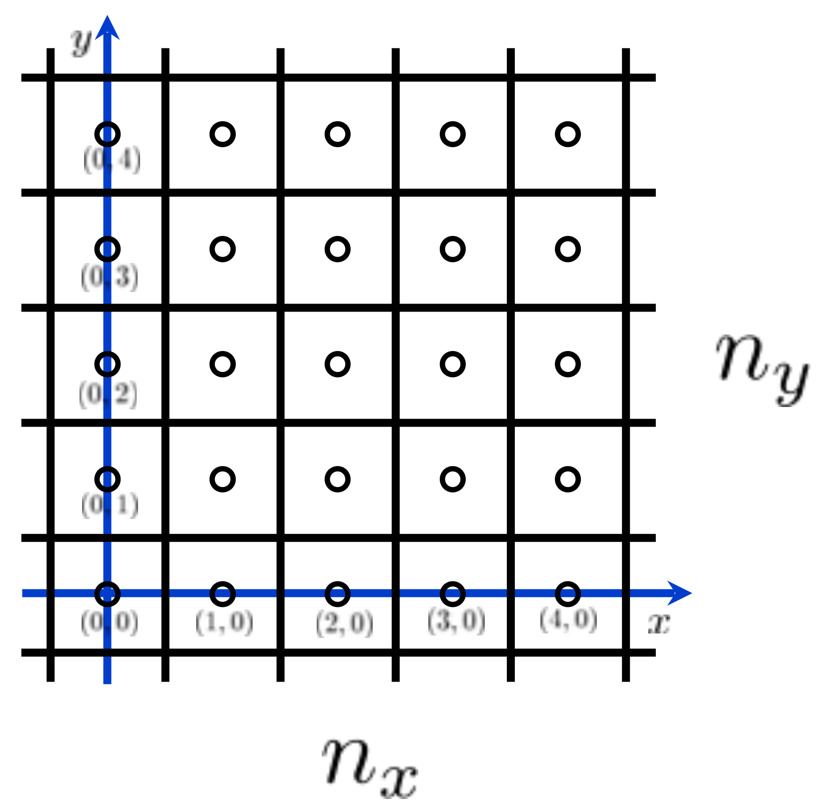
Viewport transform



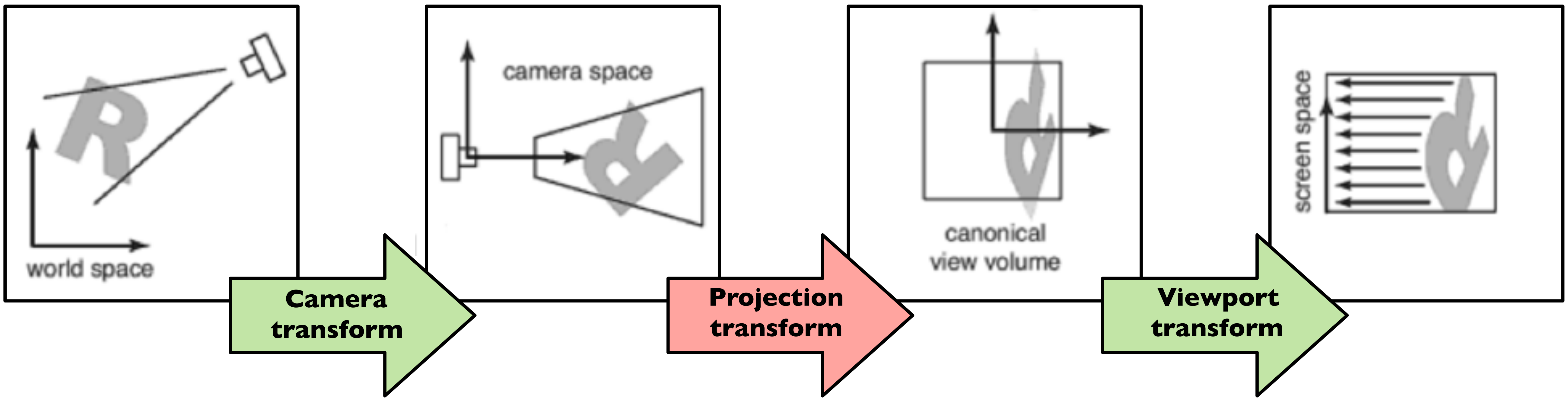


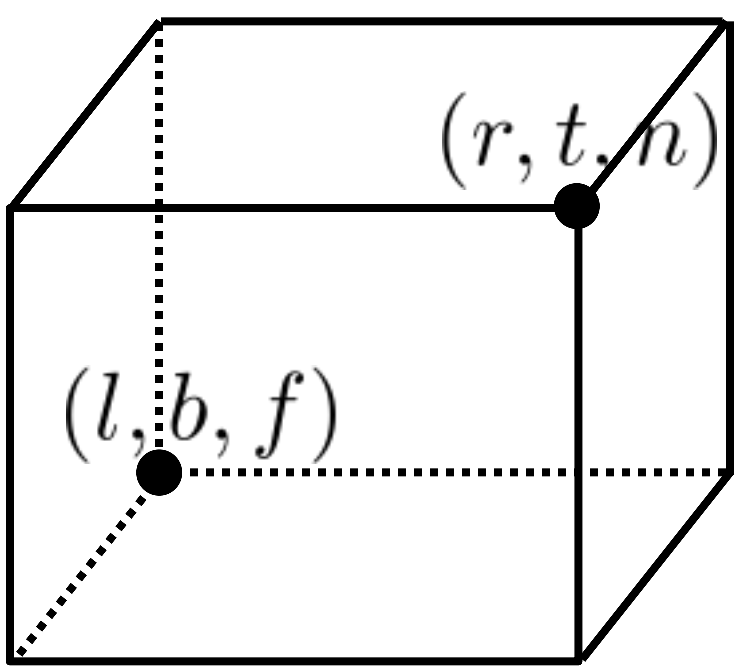
M_{vp}

<whiteboard>

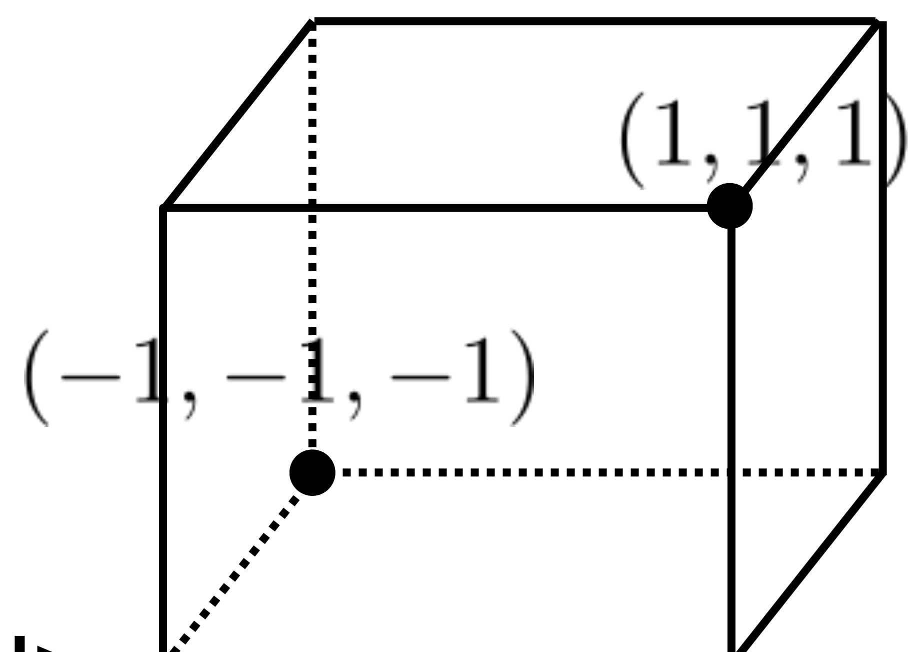


Orthographic Projection Transform



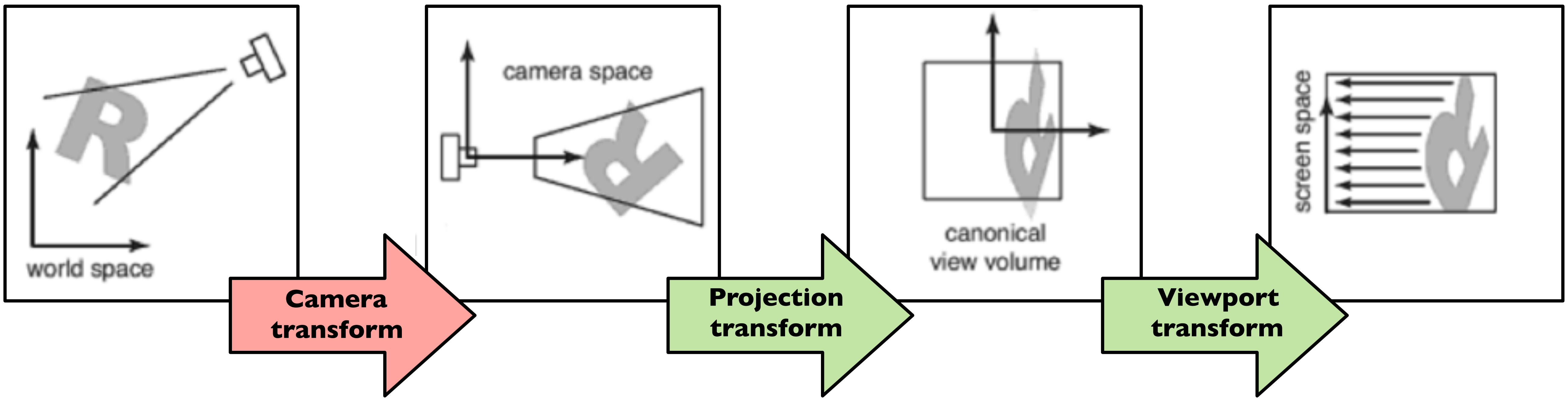


M_{orth}



<whiteboard>

Camera Transform



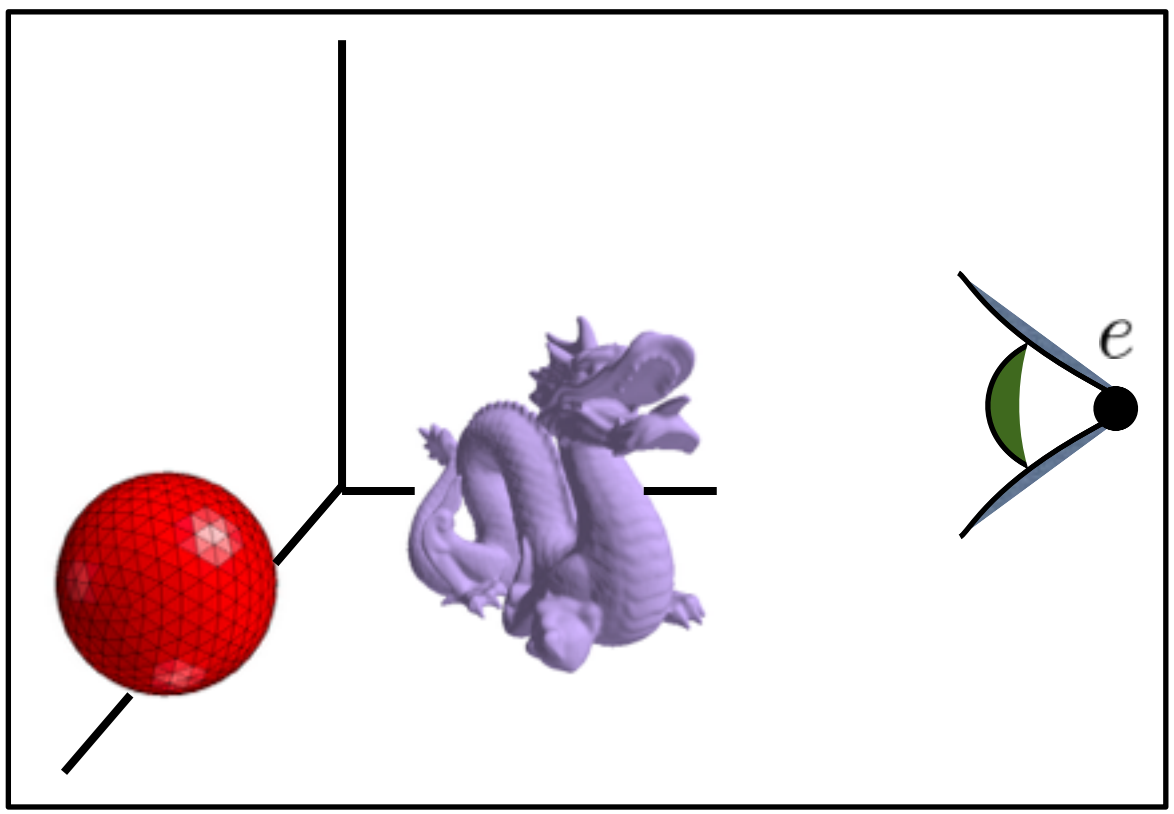
Camera Transform

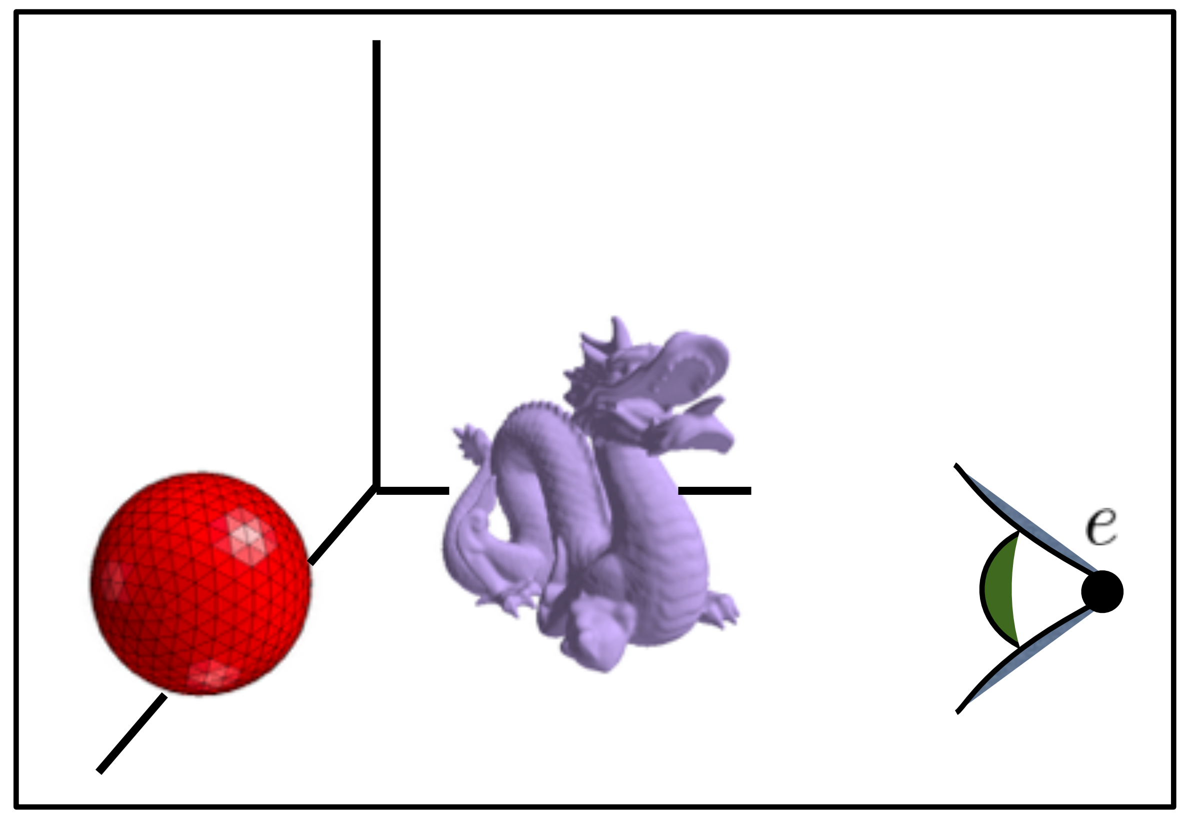
How do we specify the camera configuration?

Camera Transform

How do we specify the camera configuration?

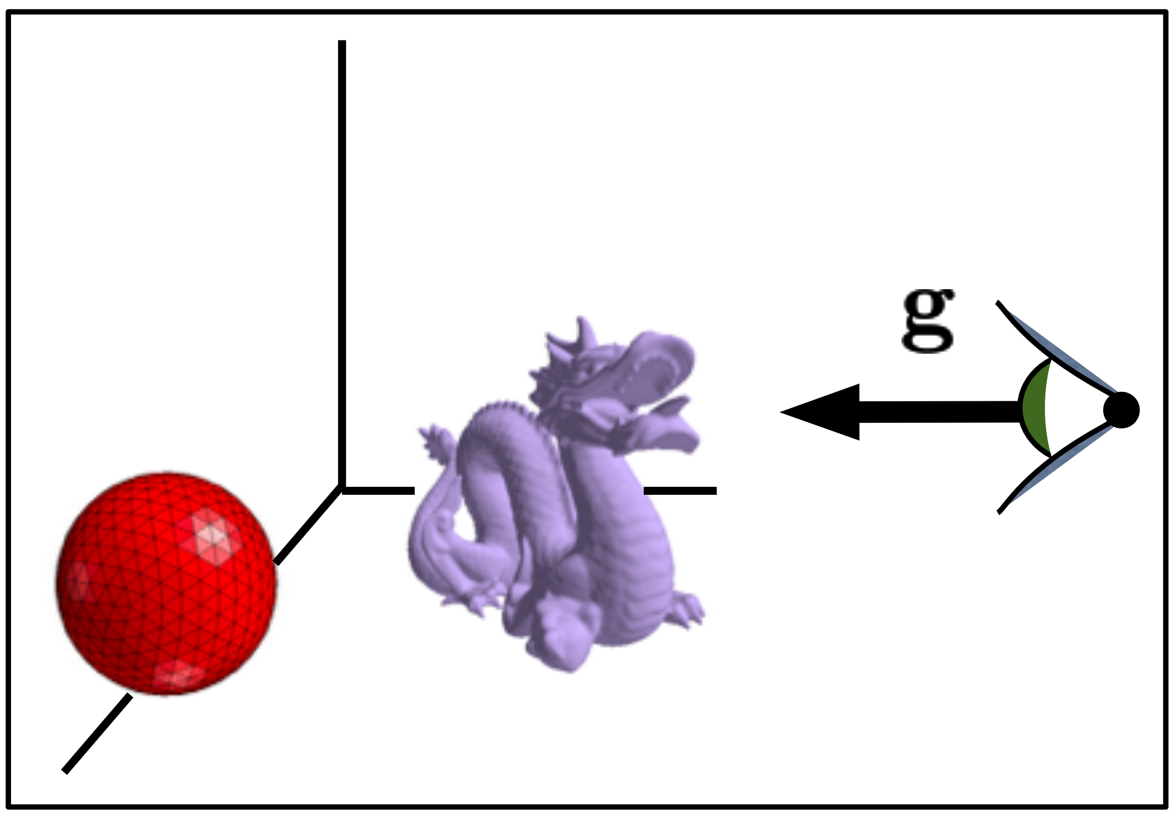
eye position

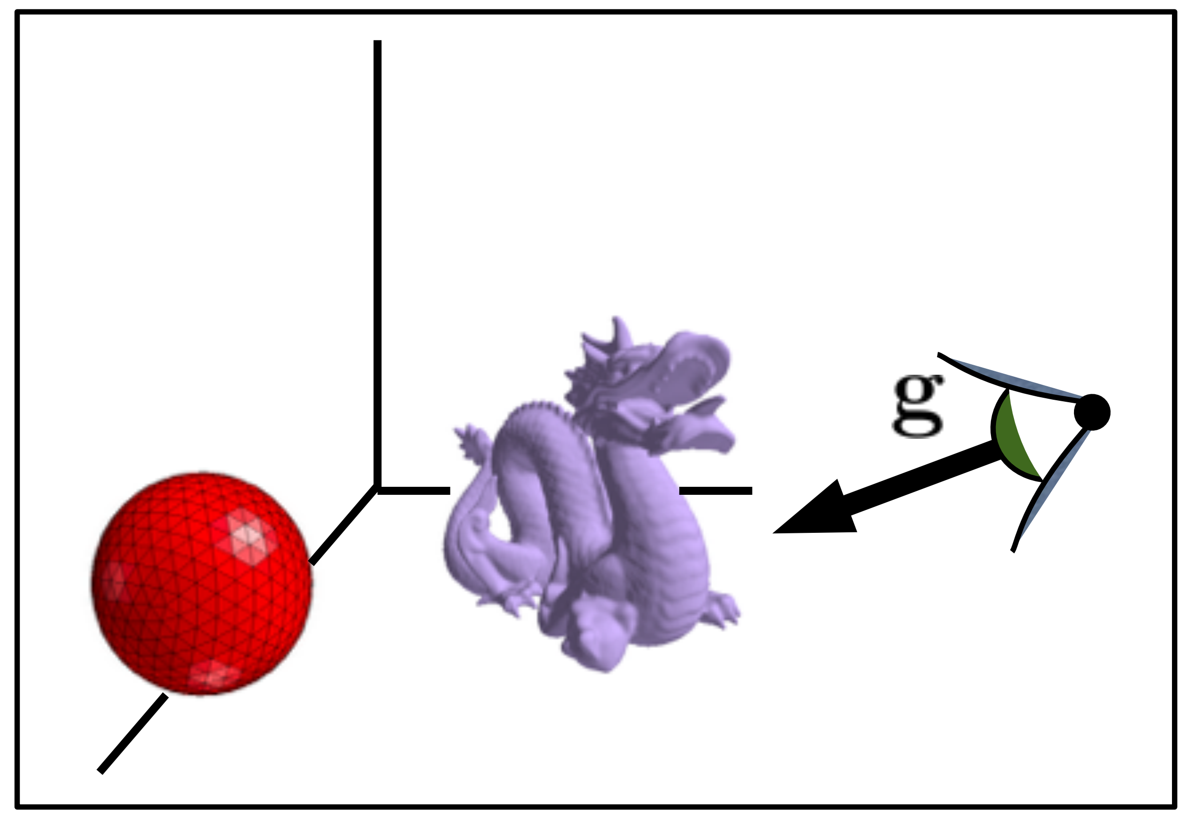




Camera Transform

How do we specify the camera configuration? **gaze direction**

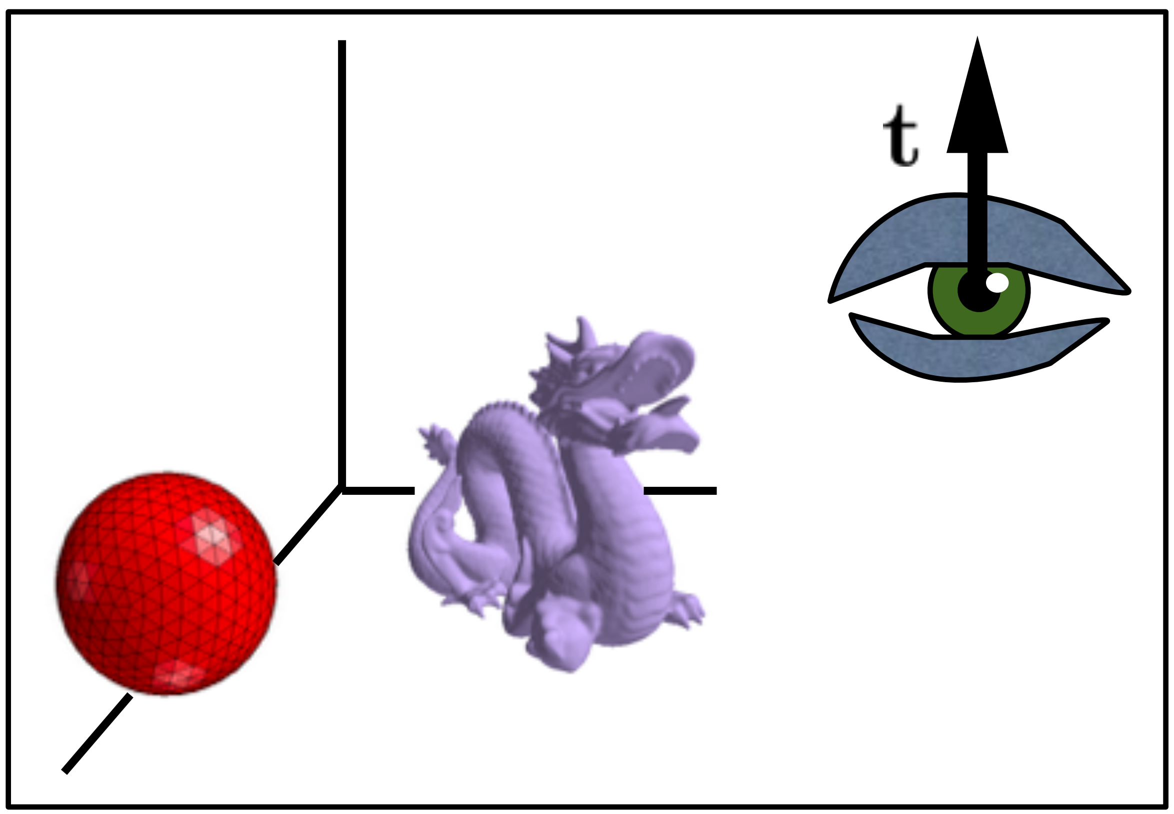


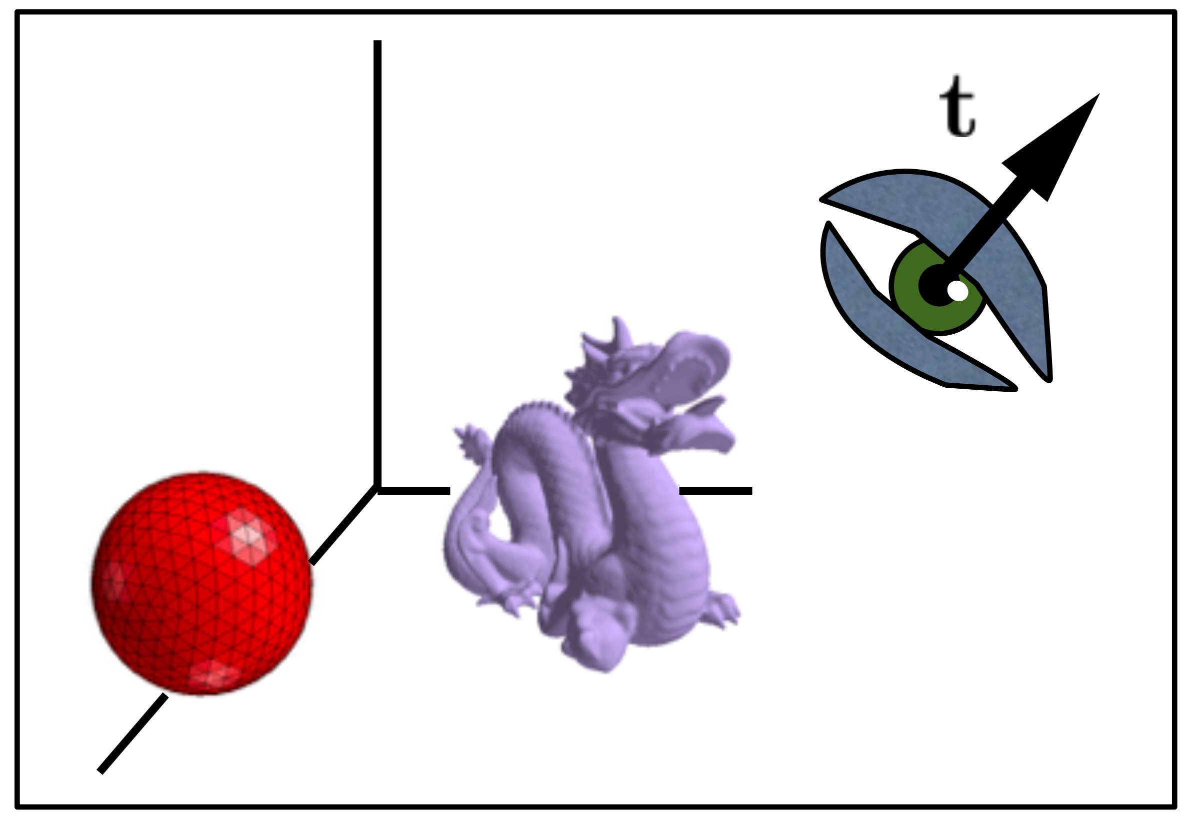


Camera Transform

How do we specify the camera configuration?

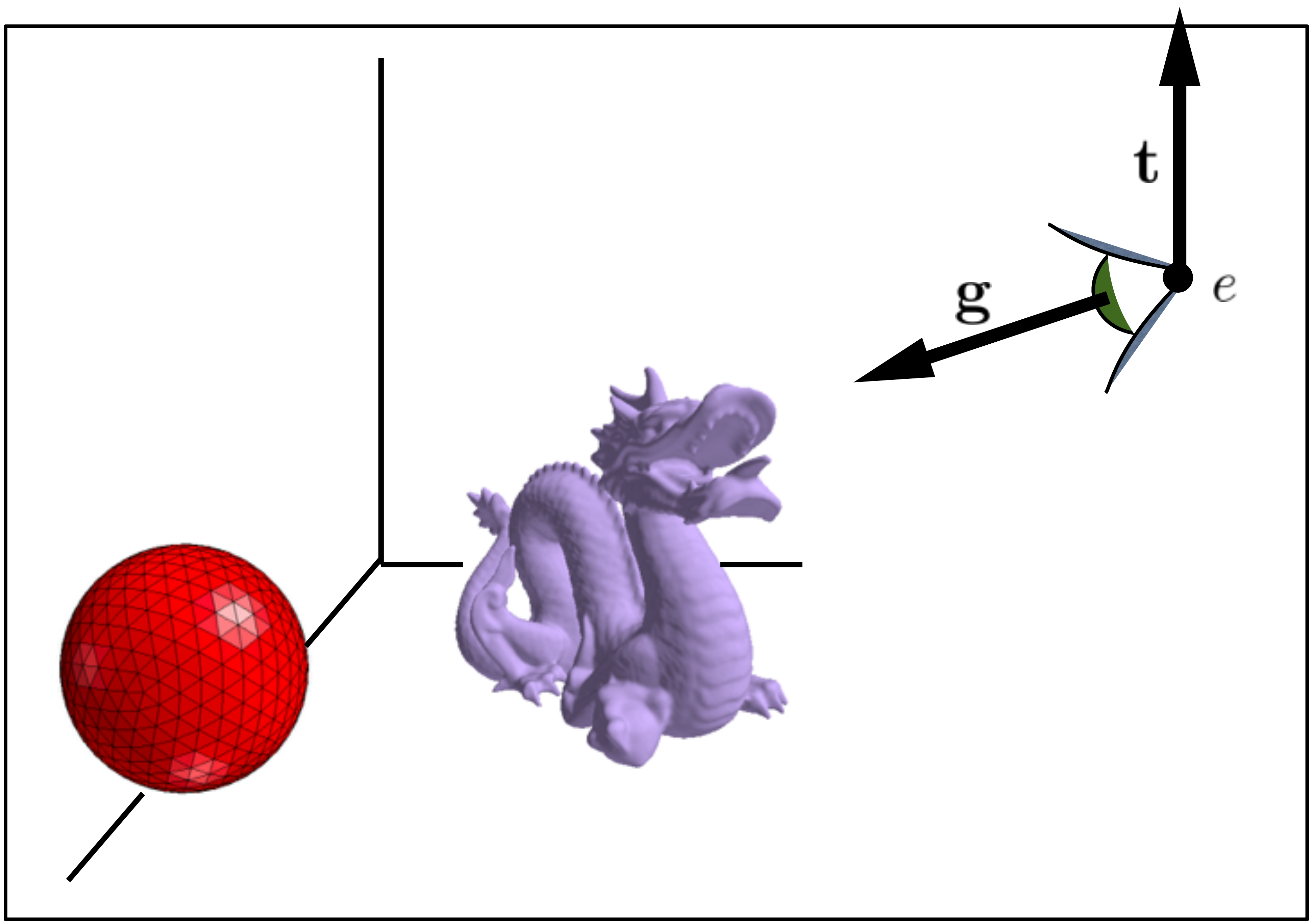
up vector



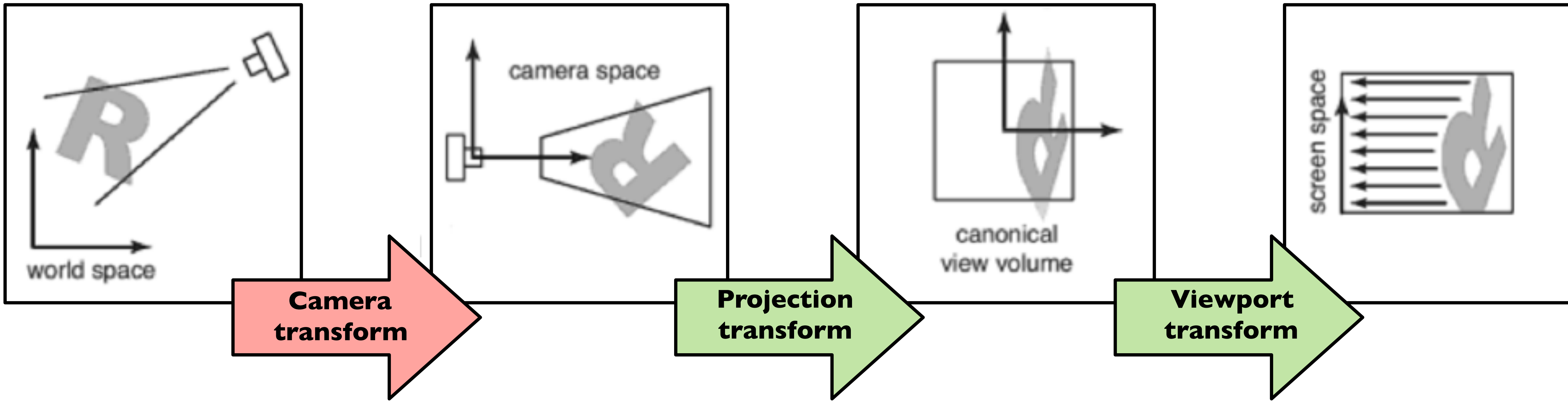


Camera Transform

How do we specify the camera configuration?



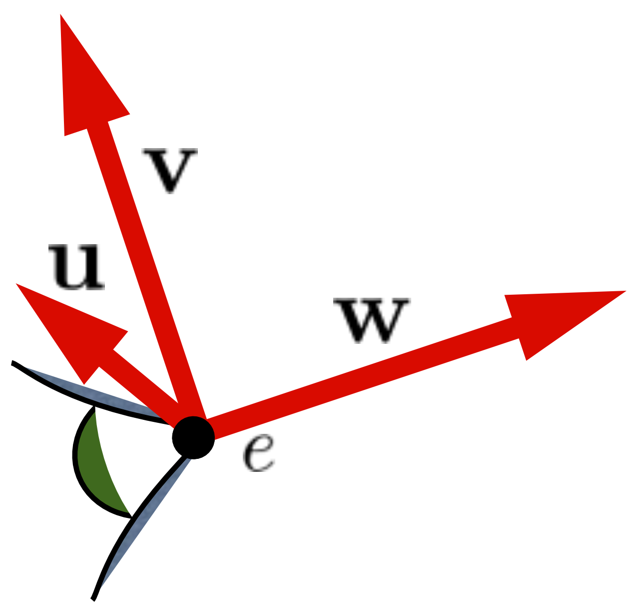
Camera Transform

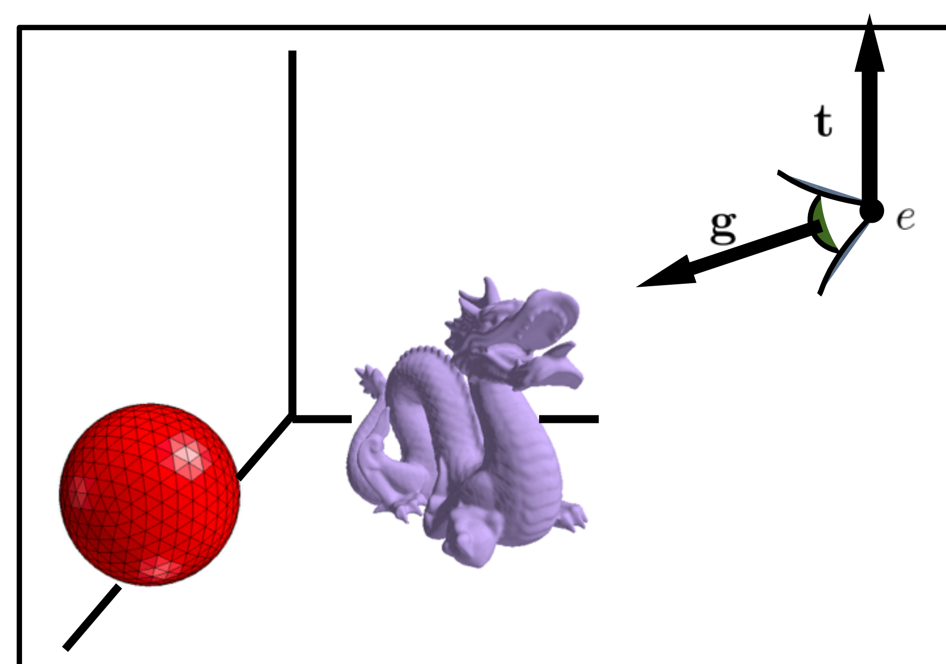


$$\mathbf{w} = -\frac{\mathbf{g}}{||\mathbf{g}||}$$

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

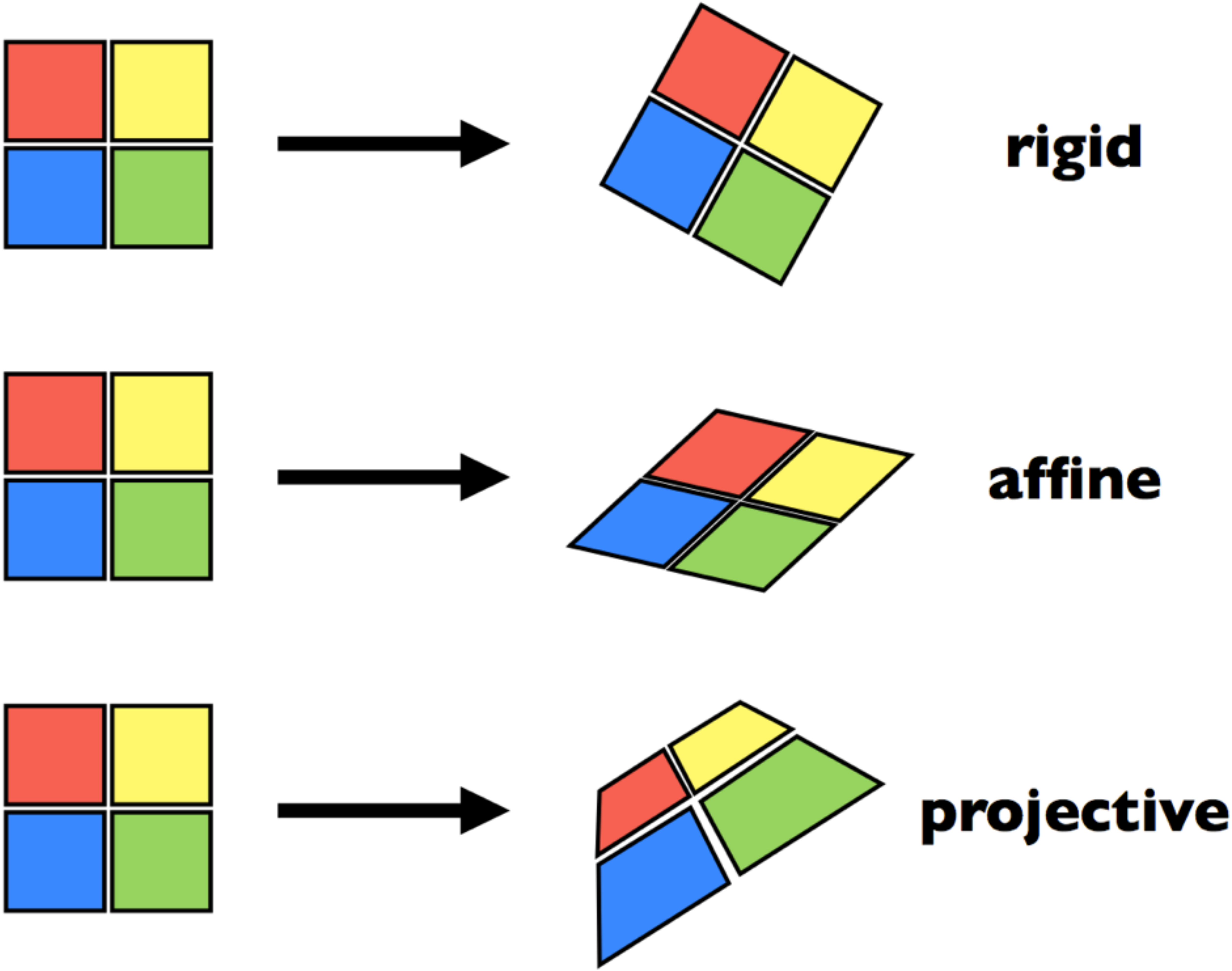
$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$



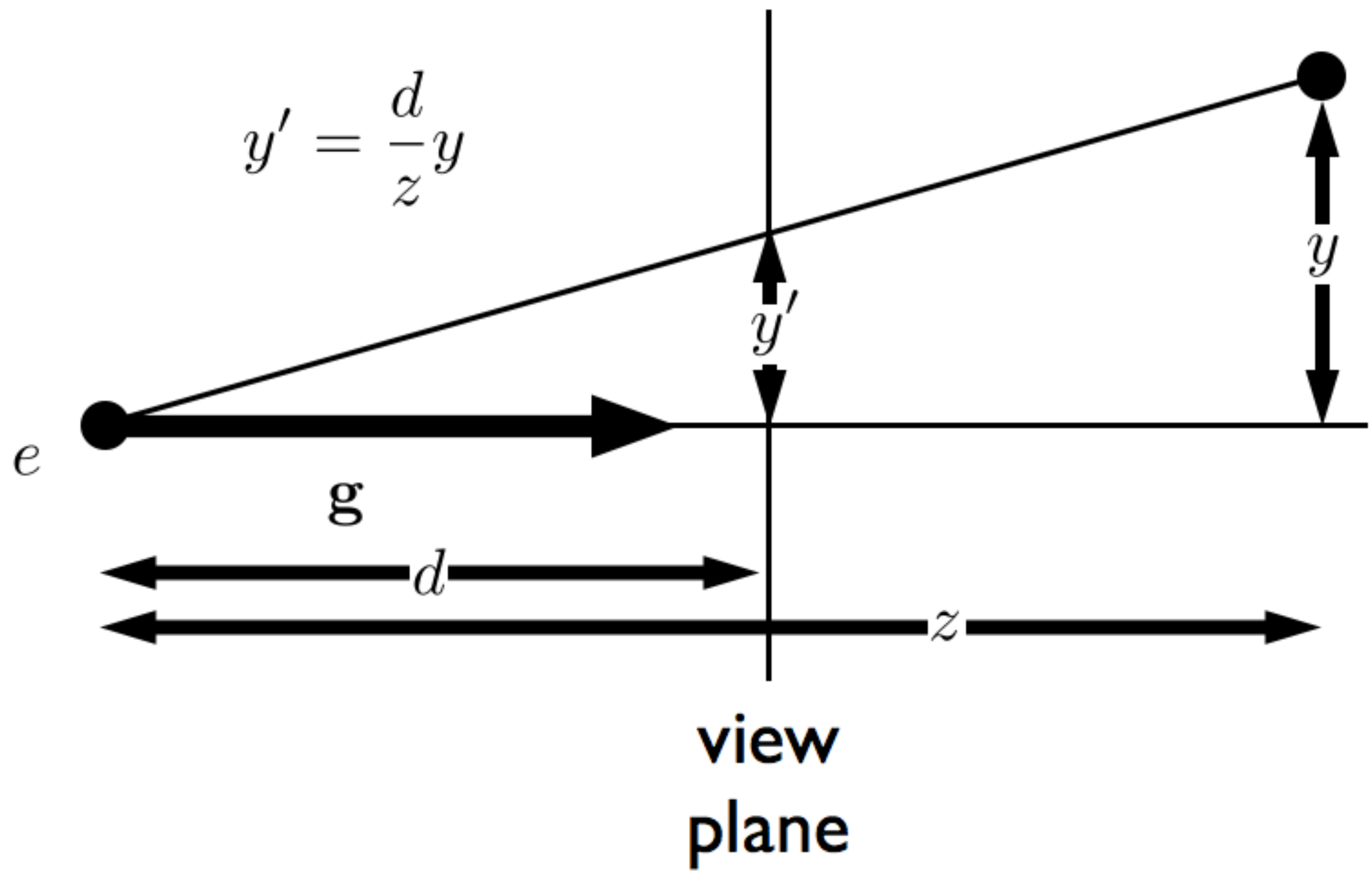


M_{cam} <whiteboard>

Perspective Viewing


rigid
affine
projective

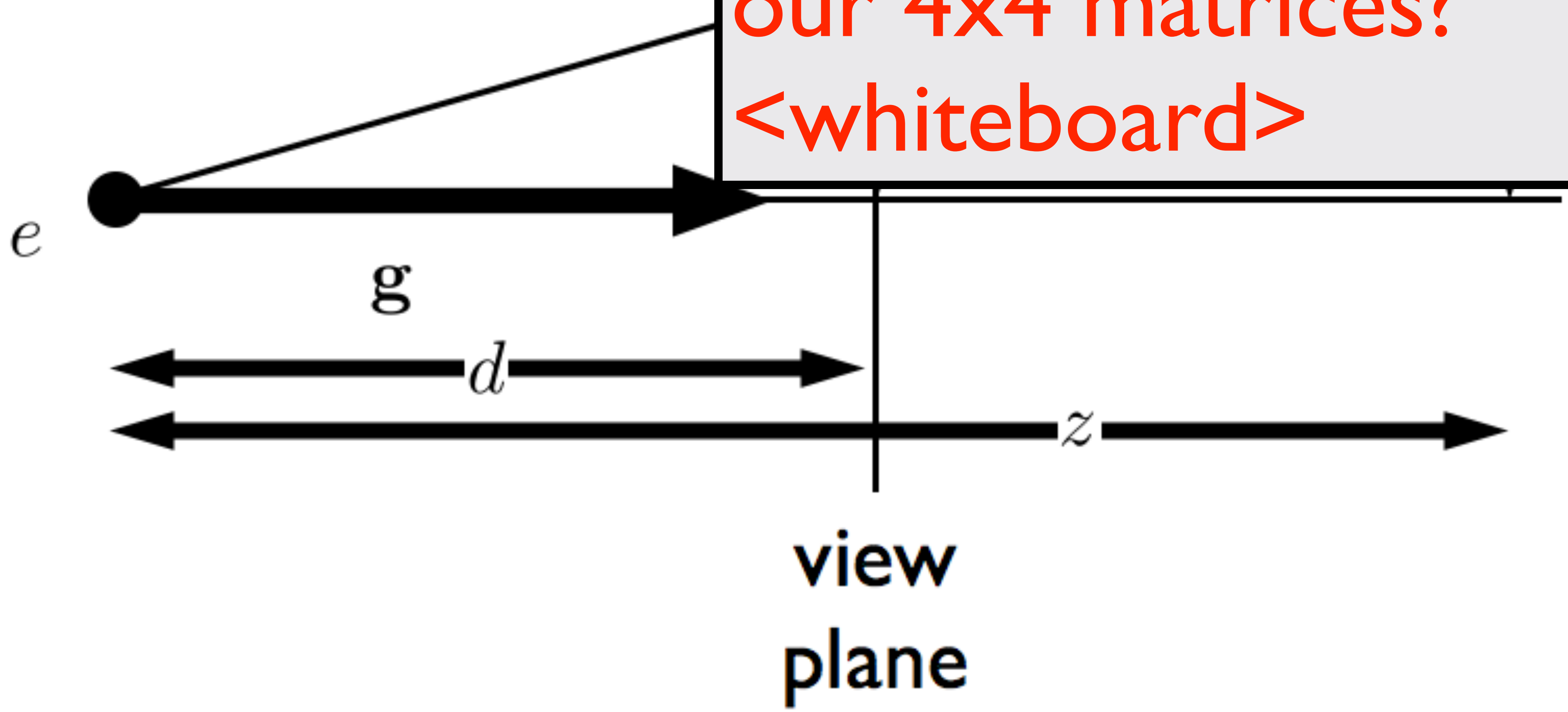
Projective Transformations



Projective Transformations

$$y' = \frac{d}{z} y$$

How can we represent this with our 4x4 matrices? <whiteboard>

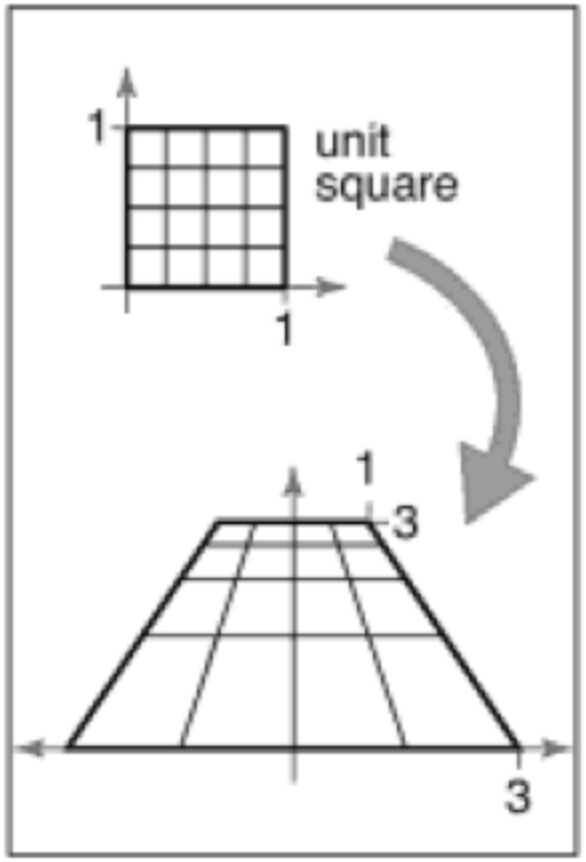


Projective Transformations

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ w \end{pmatrix} \rightarrow \begin{matrix} x = \dfrac{\tilde{x}}{w} \\ y = \dfrac{\tilde{y}}{w} \\ z = \dfrac{\tilde{z}}{w} \end{matrix}$$

Example:

$$M = \begin{pmatrix} 2 & 0 & -1 \\ 0 & 3 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$



[Shirley, Marschner]

<whiteboard>

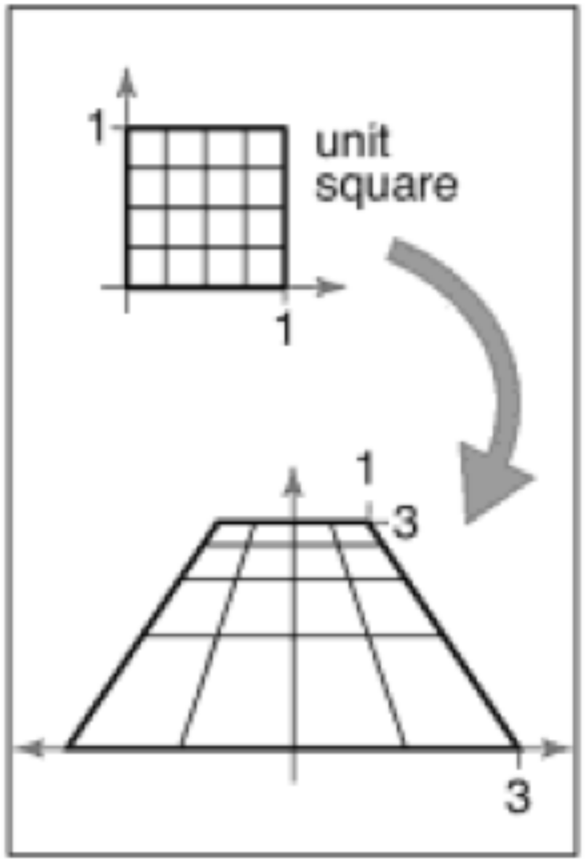
Projective Transformations

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ w \end{pmatrix} \rightarrow \begin{matrix} x = \dfrac{\tilde{x}}{w} \\ y = \dfrac{\tilde{y}}{w} \\ z = \dfrac{\tilde{z}}{w} \end{matrix}$$

We can now implement perspective projection!

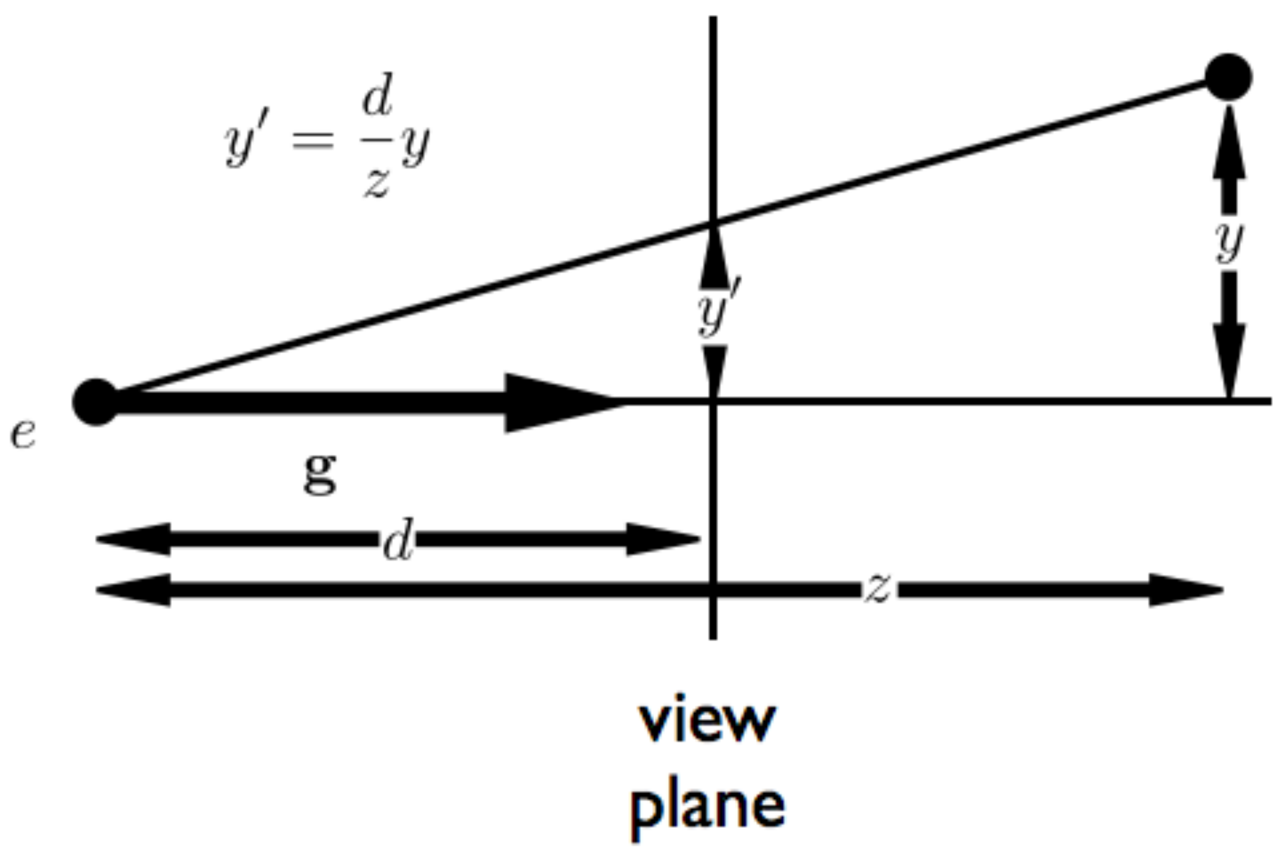
Example:

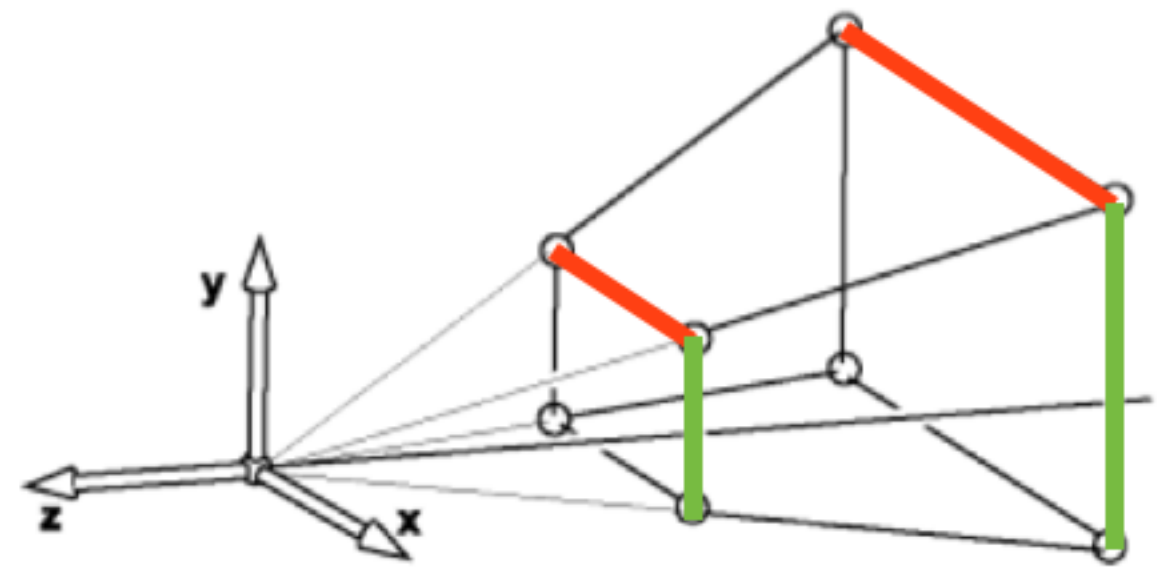
$$M = \begin{pmatrix} 2 & 0 & -1 \\ 0 & 3 & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$



[Shirley, Marschner]

Perspective Projection





both x and y get multiplied by d/z

[Shirley, Marschner]

Simple perspective projection

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \Rightarrow \begin{cases} x' = \frac{d}{z}x \\ y' = \frac{d}{z}y \\ z' = \frac{d}{z}z = d \end{cases}$$

This achieves a simple perspective projection onto the view plane z = d

but we've lost all information about z!

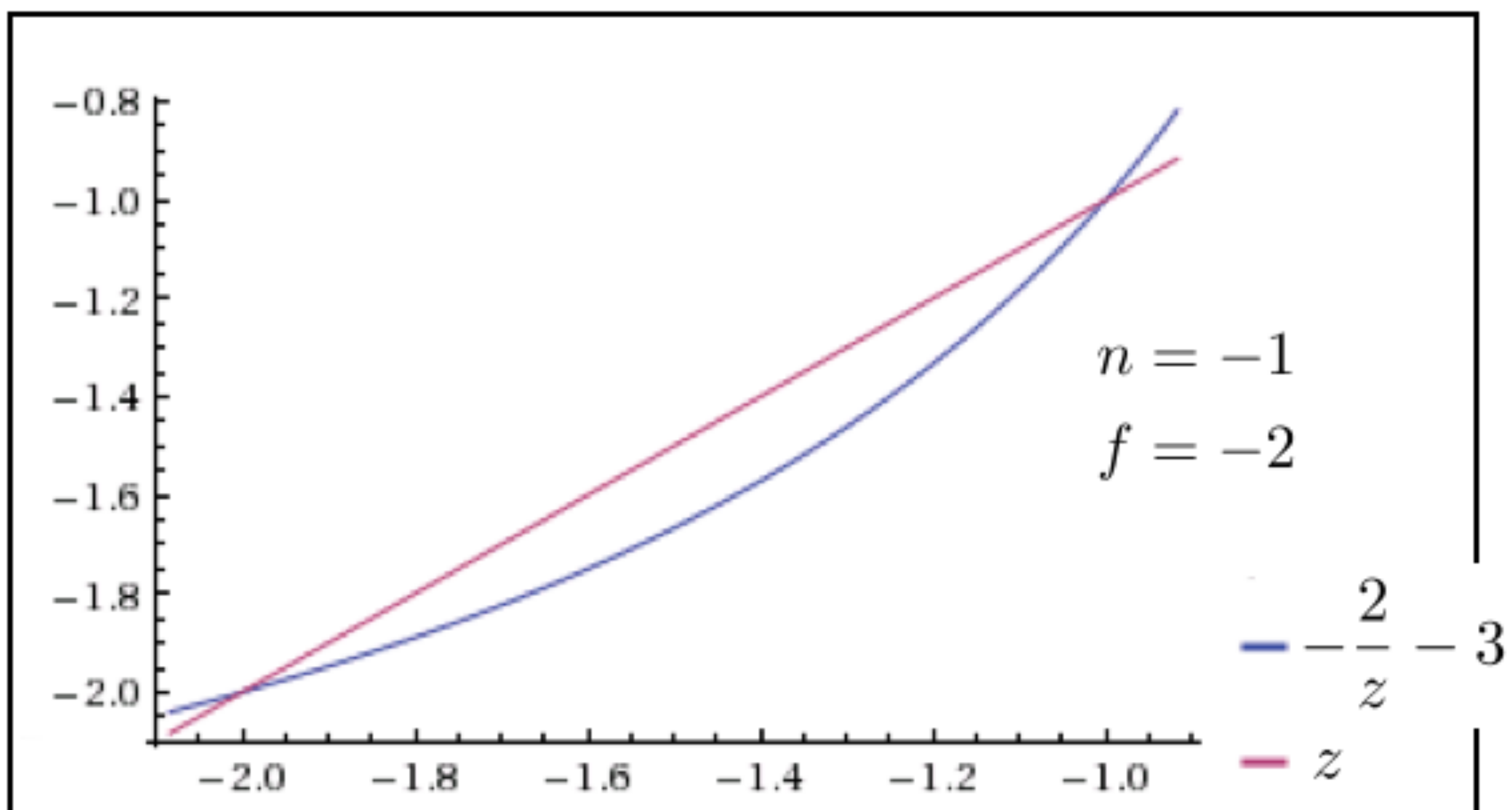
<whiteboard>

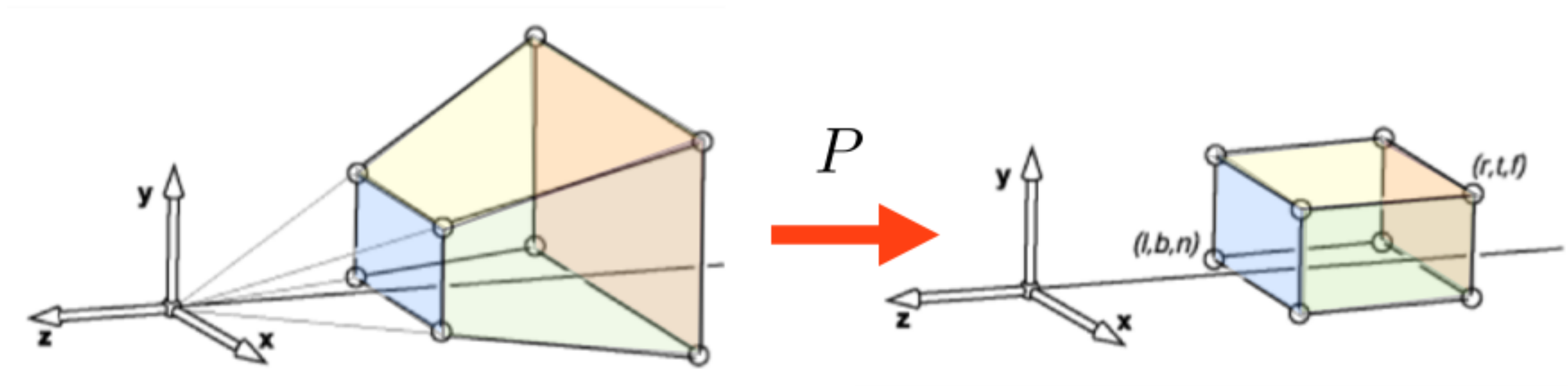
Perspective Projection

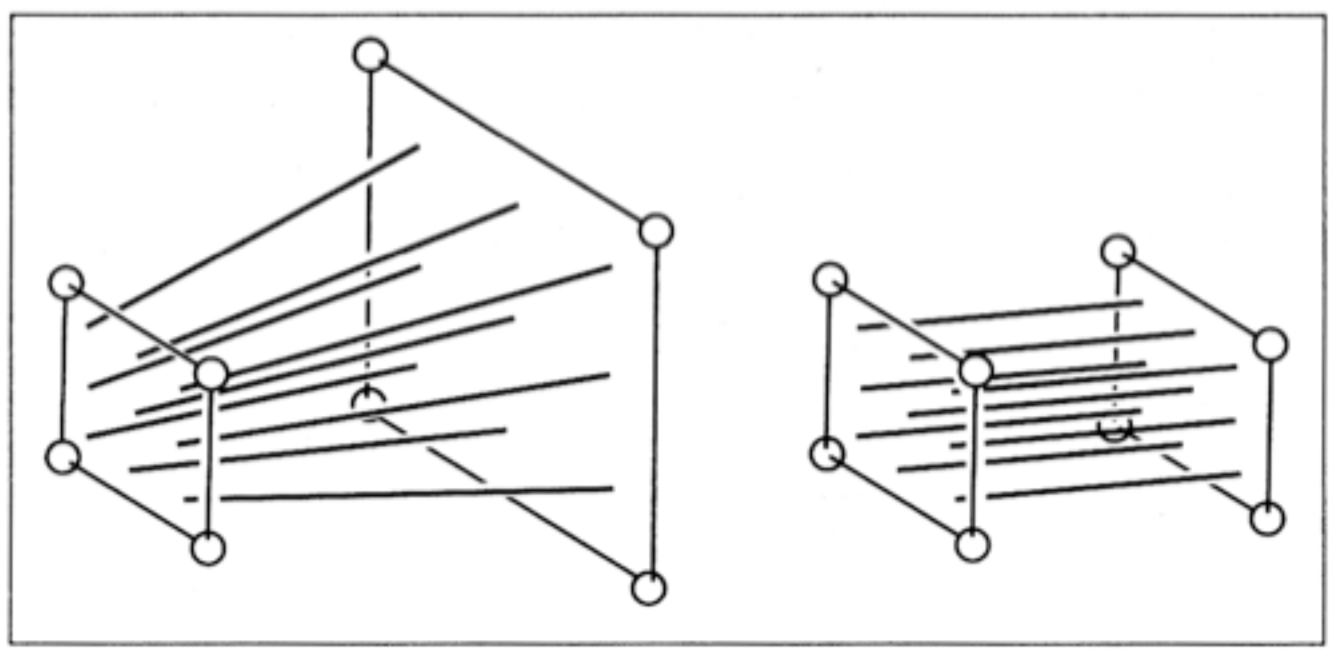
$$P = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$z' = (n+f) - \frac{nf}{z}$$

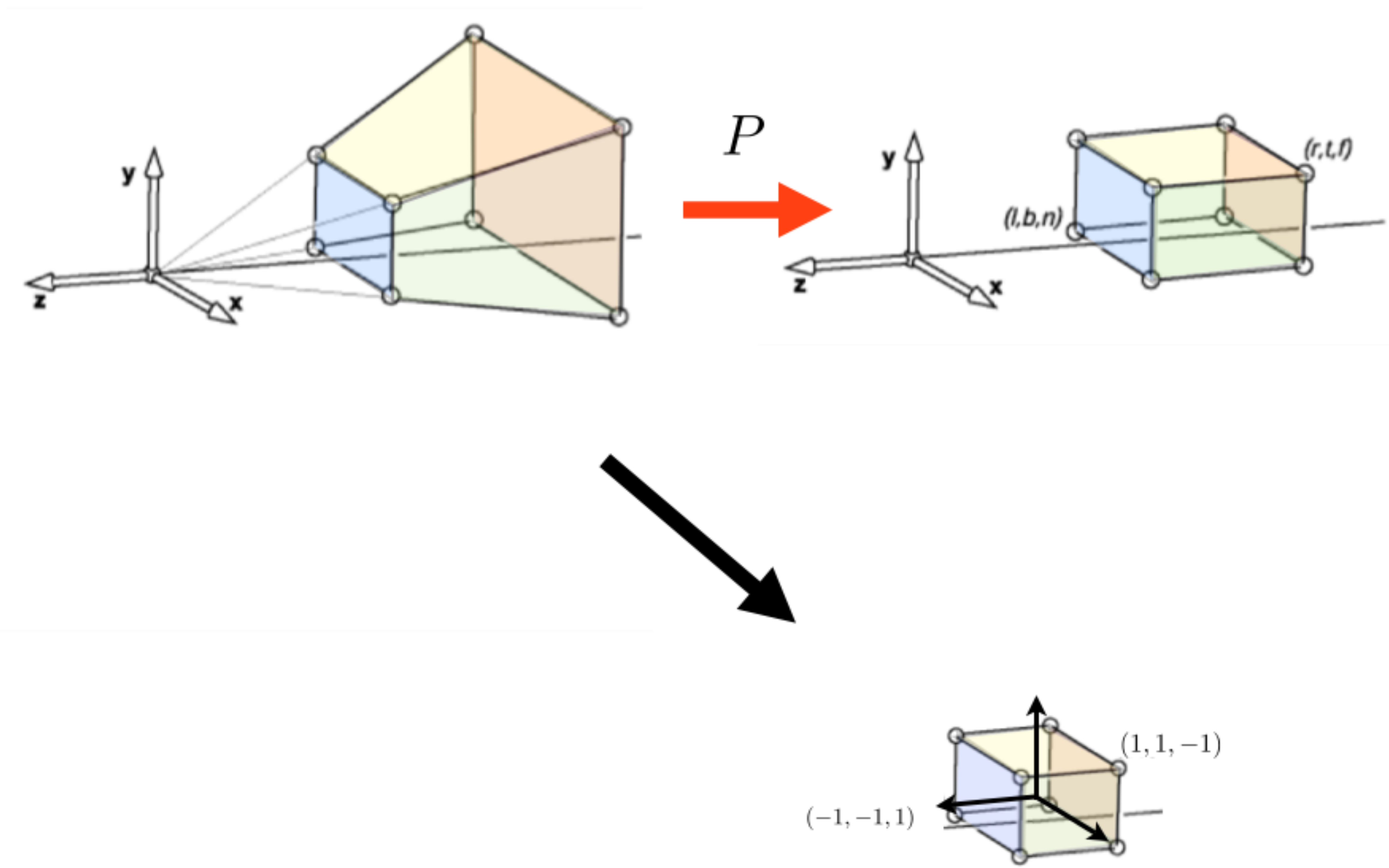
Example:



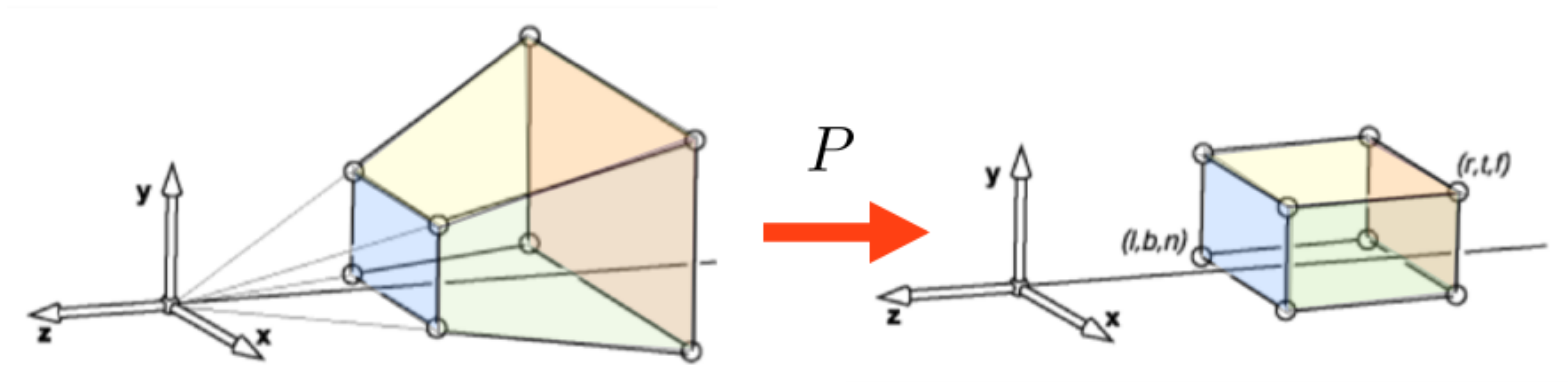


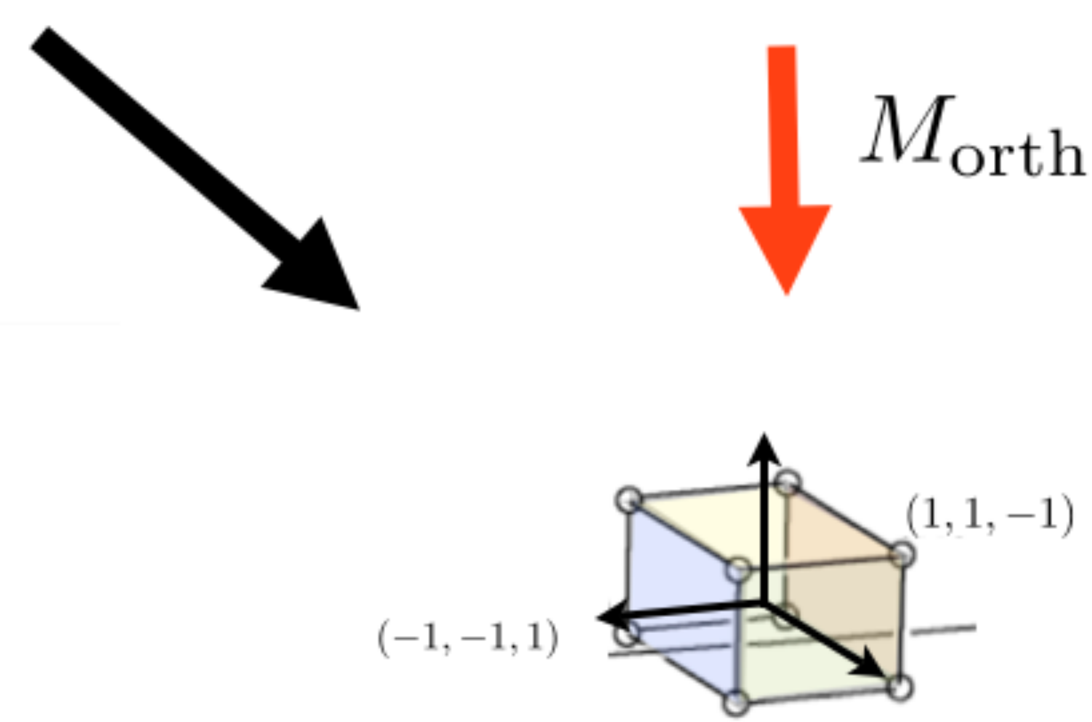


[Shirley, Marschner]



y
z
x
P
(r,t,f)
(l,b,n)
y
z
x
(1, 1, −1)
(−1, −1, 1)



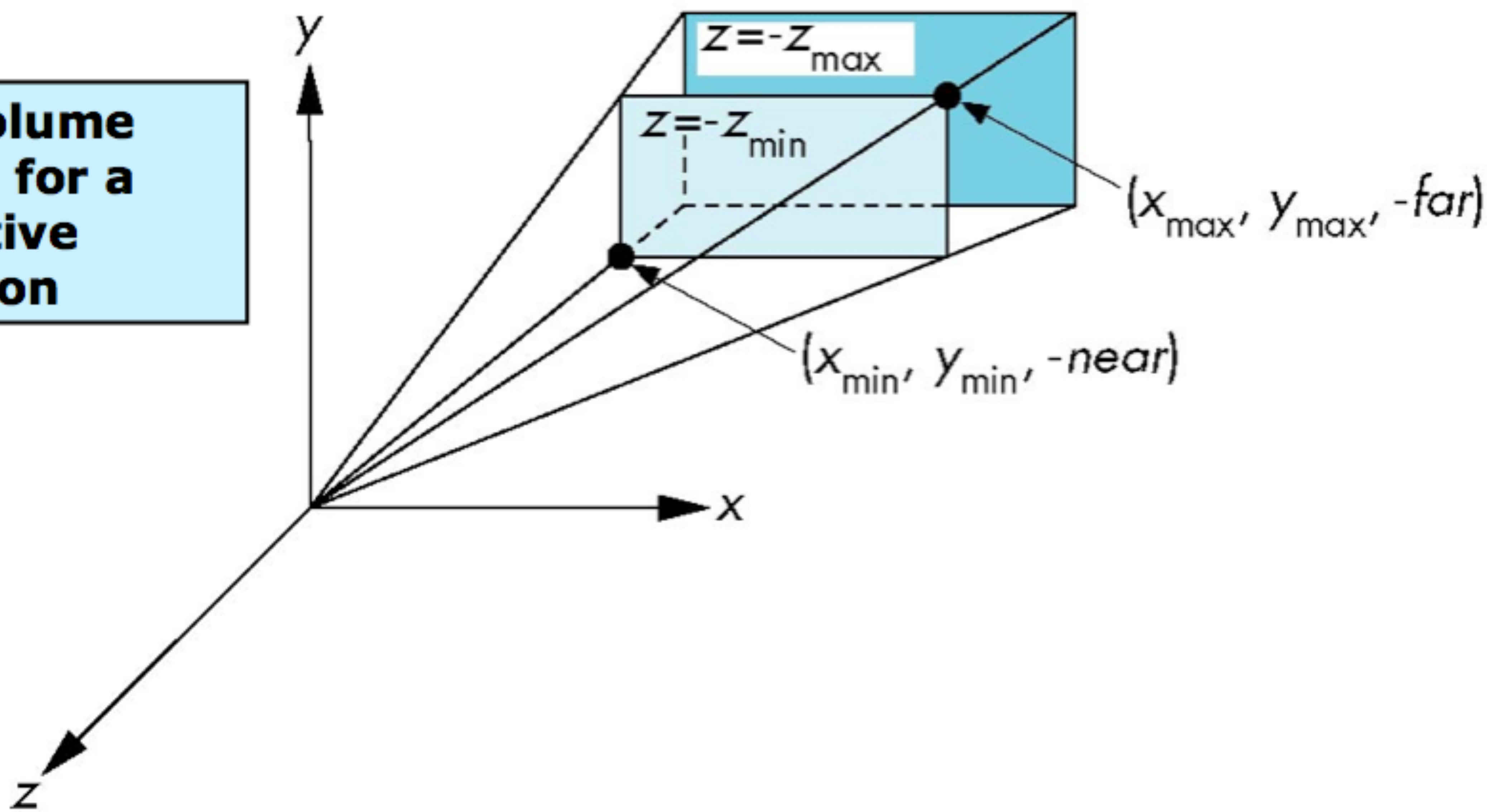


$$M_{\mathrm{per}} = M_{\mathrm{orth}} P$$

OpenGL Perspective Viewing

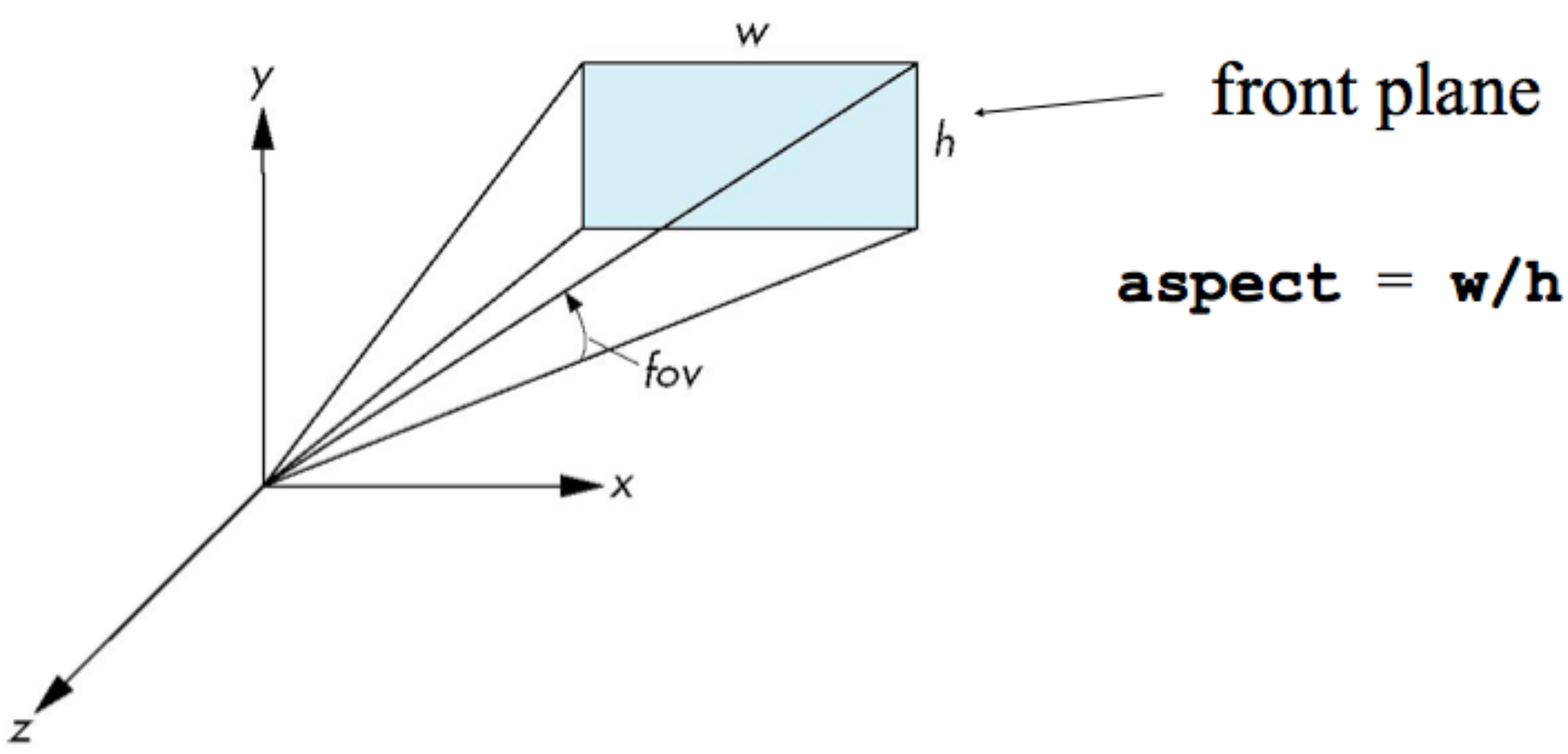
```
glFrustum(xmin,xmax,ymin,ymax,near,far)
```

Clipping volume (frustrum) for a perspective projection

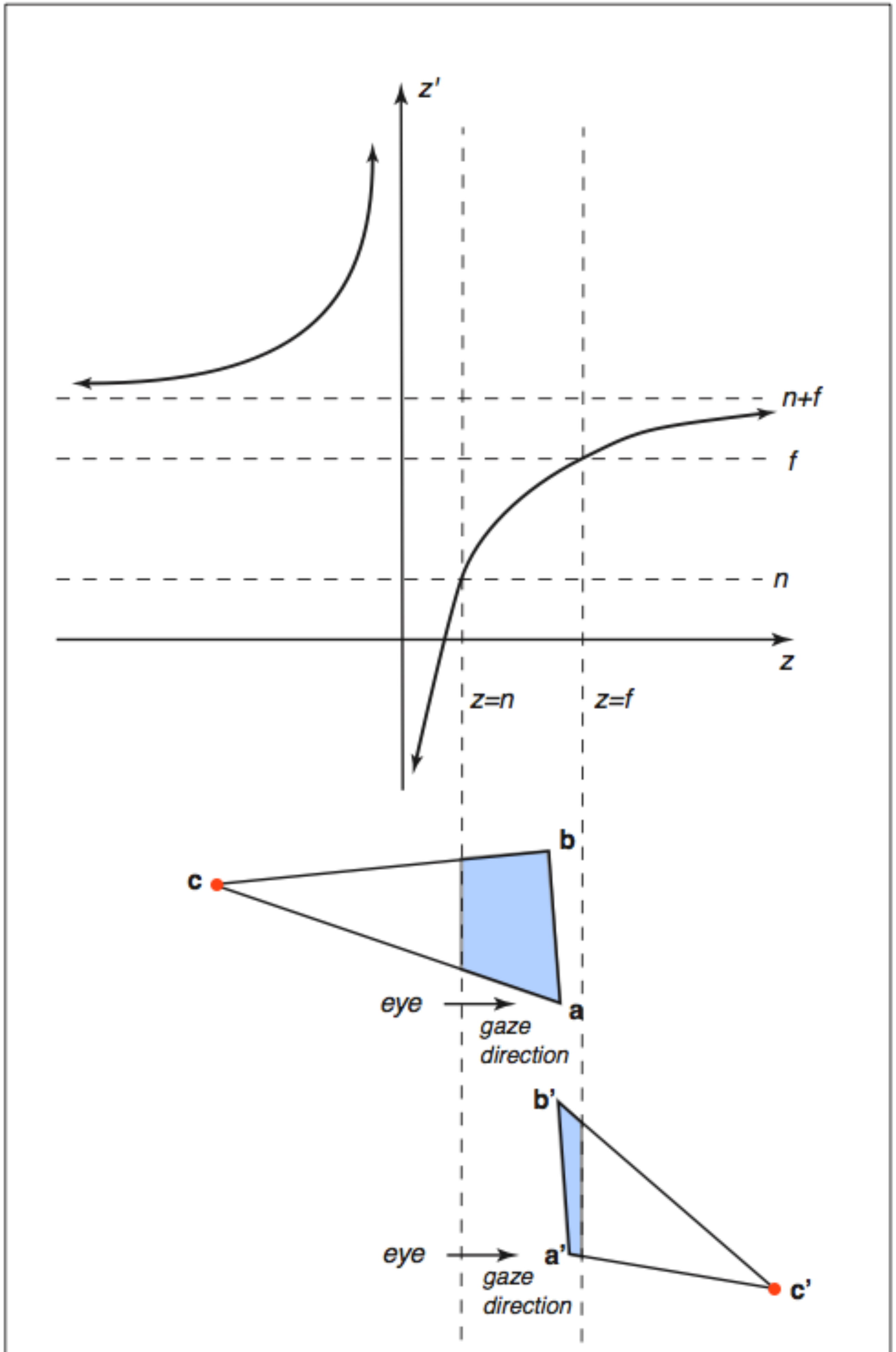


Using Field of View

With **`glFrustum`** it is often difficult to get the desired view **`gluPerpective(fovy, aspect, near, far)`** often provides a better interface



`aspect = w/h`

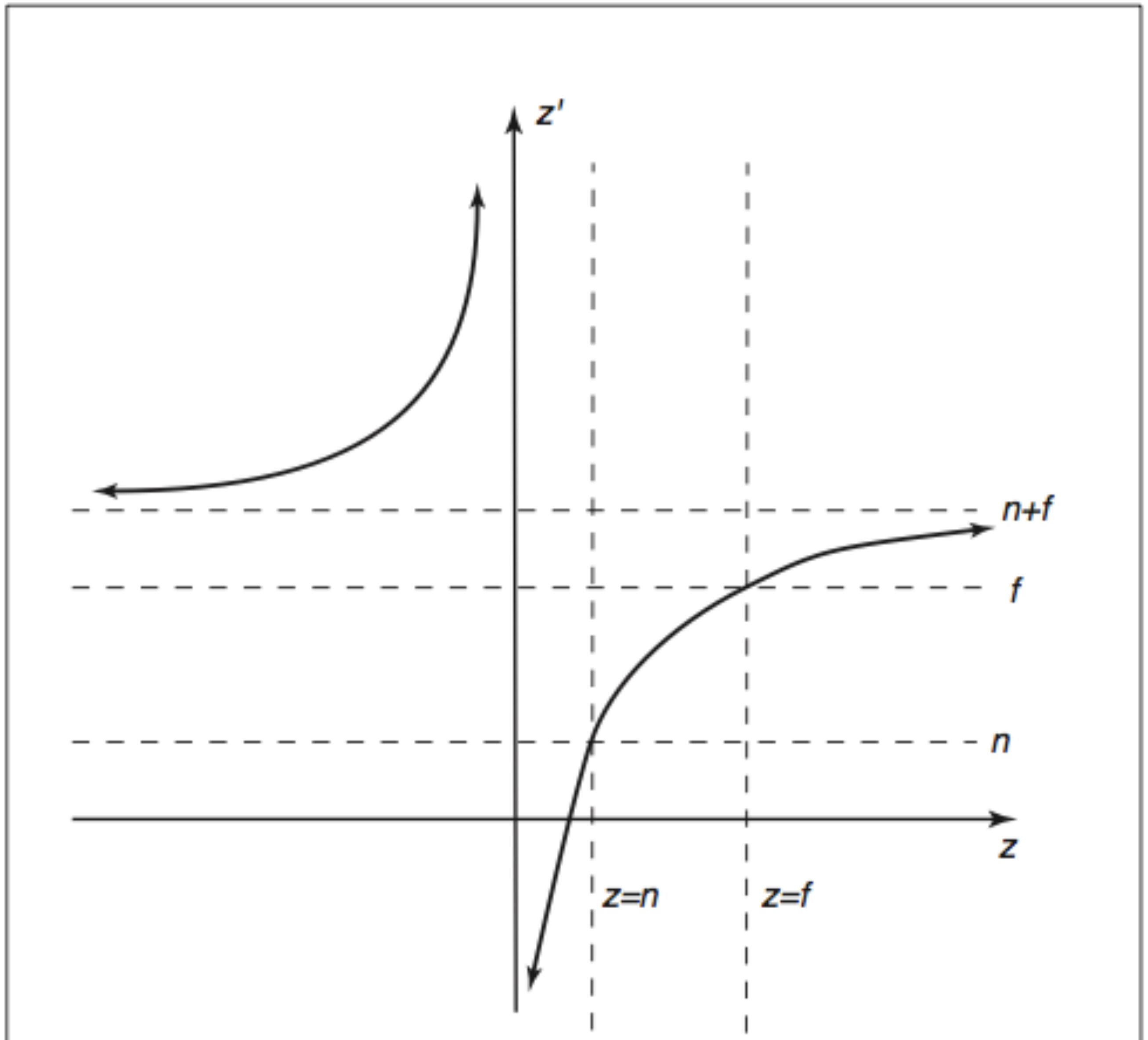


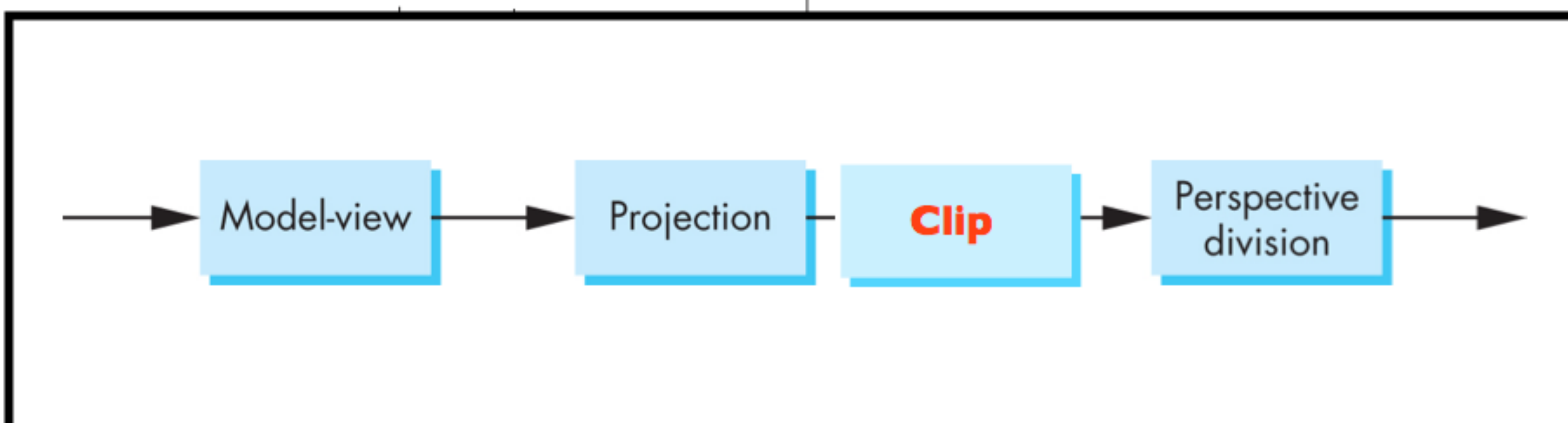
[Shirley, Marschner]

Clipping after the perspective transformation can cause problems

OpenGL clips **after** projection and **before** perspective division

$$-w \leq x \leq w$$
$$-w \leq y \leq w$$
$$-w \leq z \leq w$$





[Shirley, Marschner]