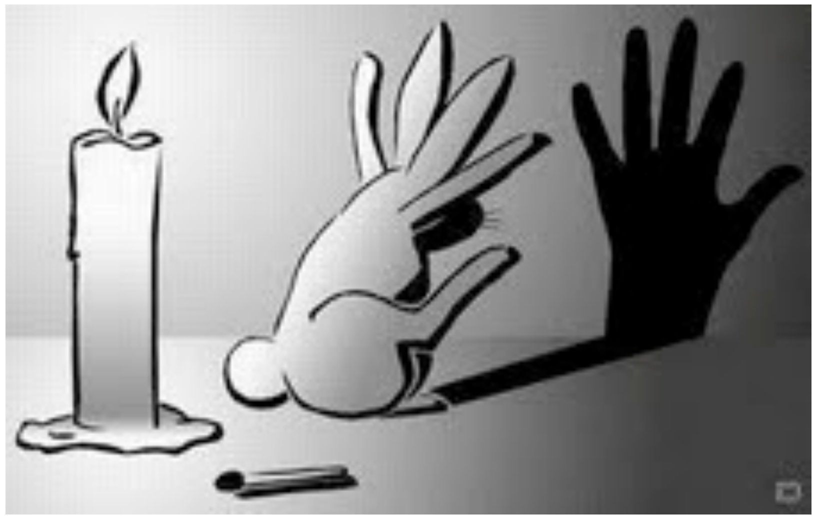
CS230 : Computer Graphics

Lecture 4: Viewing Transformations

Tamar Shinar
Computer Science & Engineering
UC Riverside

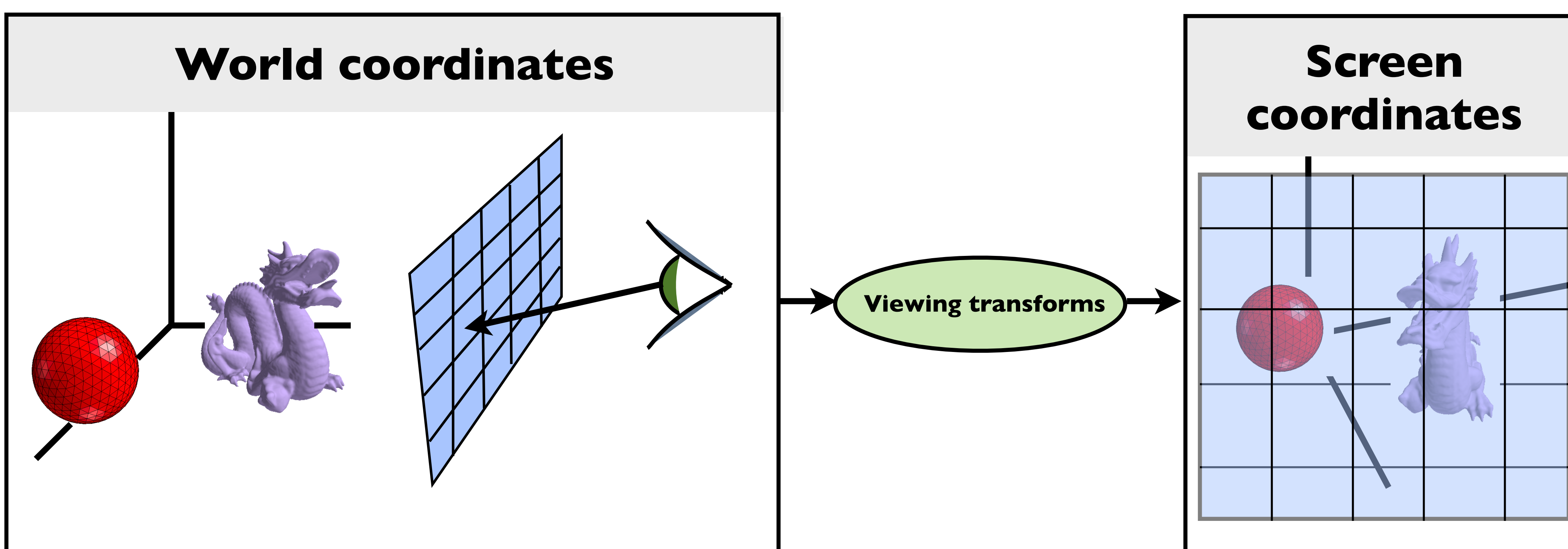
Viewing Transformations



Viewing transformations

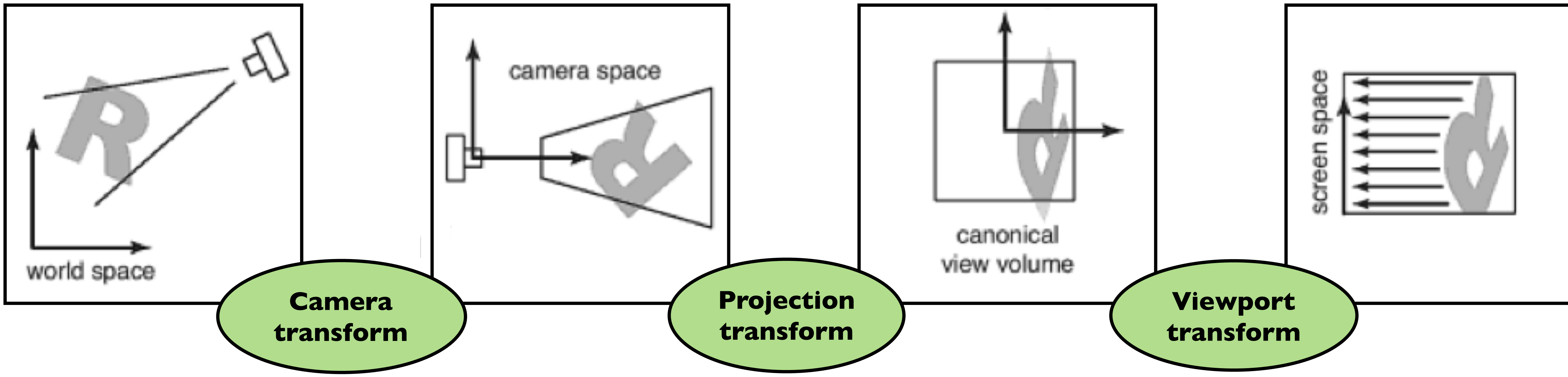
World space → **Viewing transforms** → **Image space**

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



The viewing transformation also project any pixels viewing ray back to the pixel’s position in **image space**

Decomposition of viewing transforms



- rigid body transformation
- place camera at origin

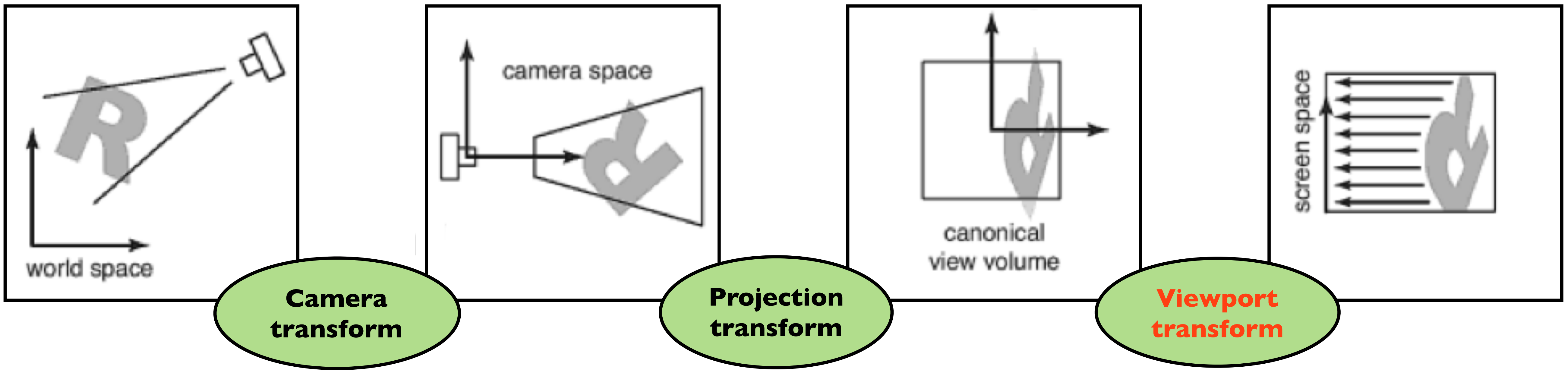
- x, y, z in [-1,1]
- depends on type of projection

- map to pixel coordinates

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

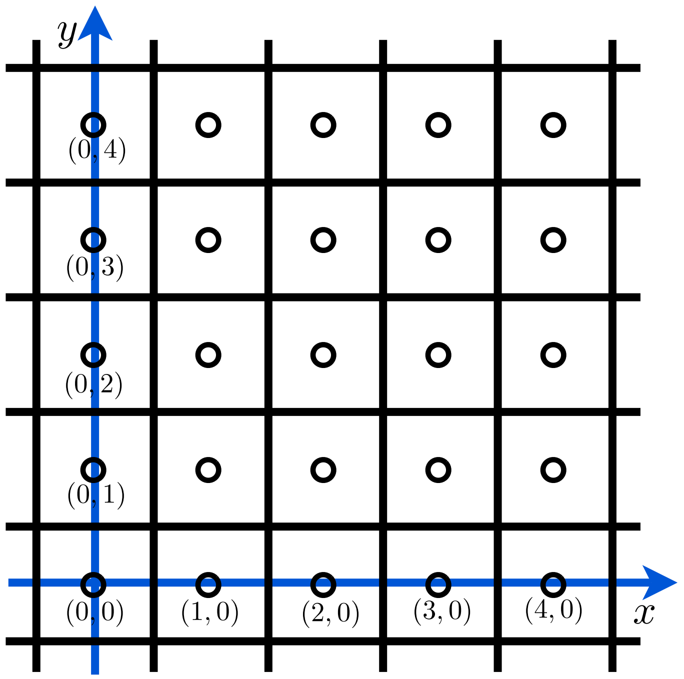
there are several names for these spaces: “camera space” = “eye space”, “canonical view volume” = “clip space”= “normalized device coordinates”, “screen space=pixel coordinates” and for the transforms: “camera transformation” = “viewing transformation”

Viewport transform

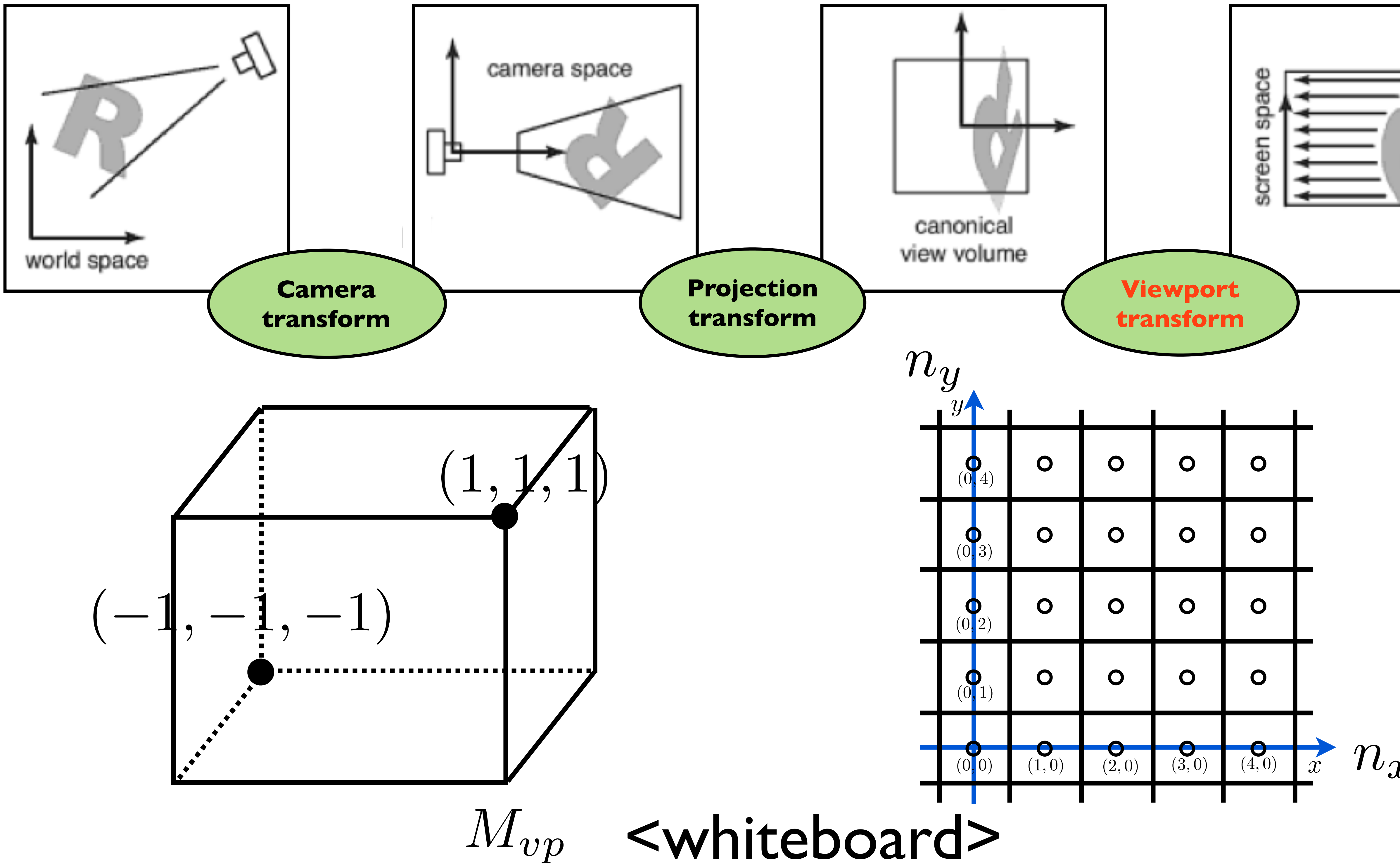


$$(x, y, z) \to (x', y', z')$$

$$(x, y, z) \in [-1, 1]^3 \qquad \begin{aligned} x' &\in [-.5, n_x - .5] \\ y' &\in [-.5, n_y - .5] \end{aligned}$$

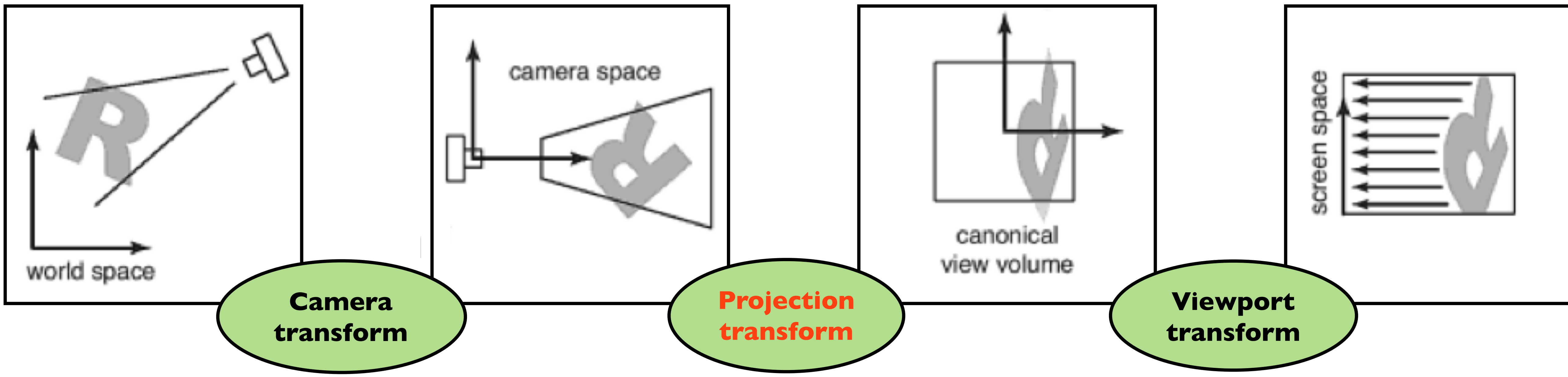


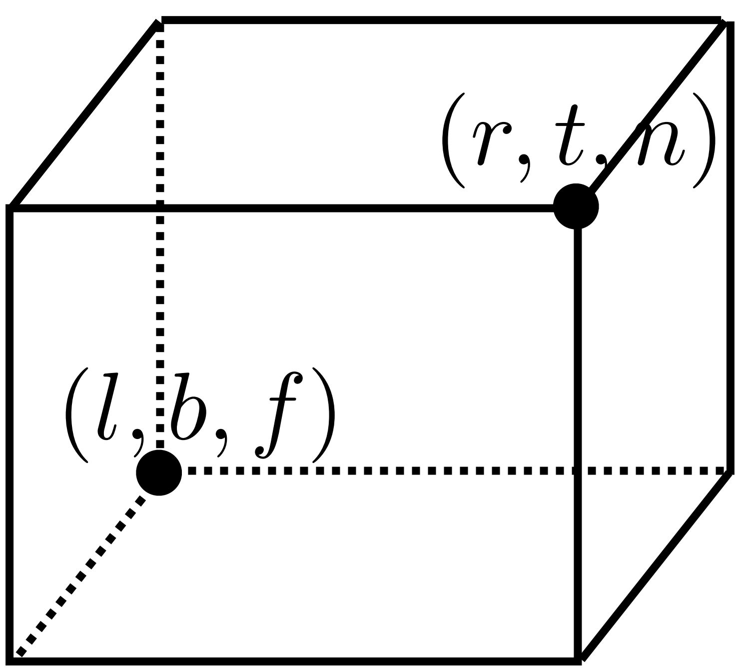
Viewport transform

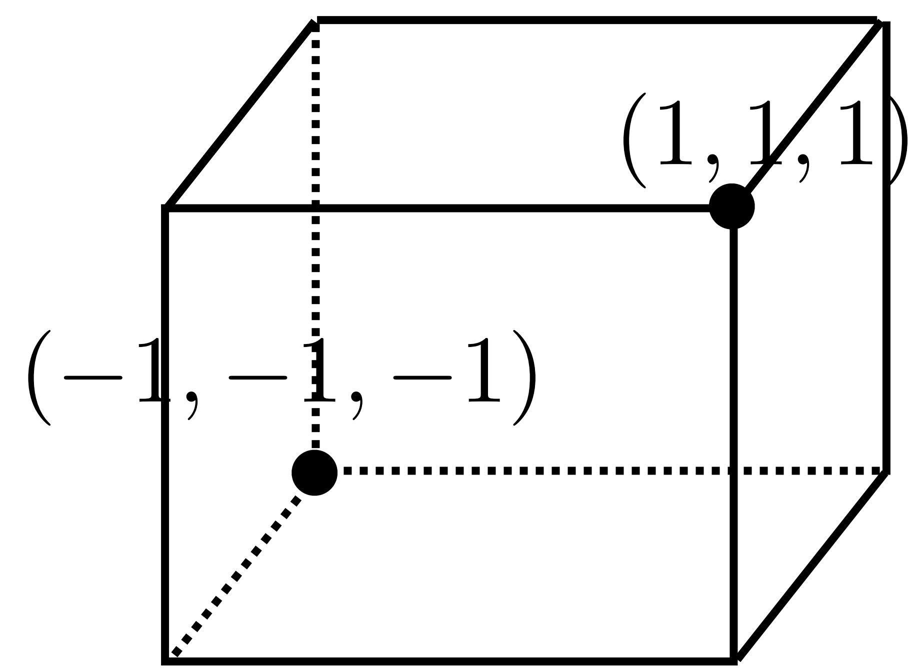


<whiteboard>

Orthographic Projection Transform



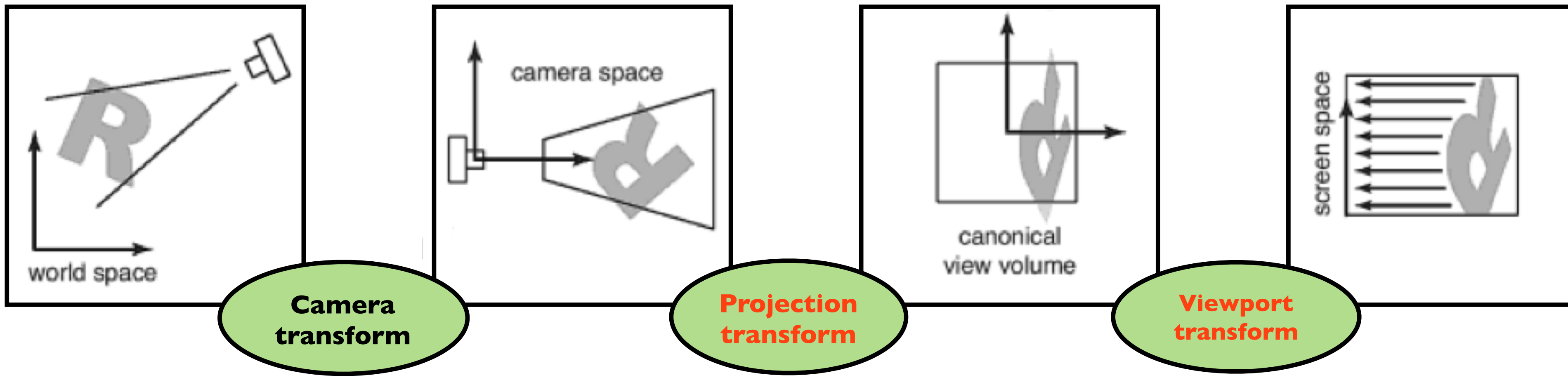




n_x

M_{orth} <whiteboard>

Line drawing algorithm

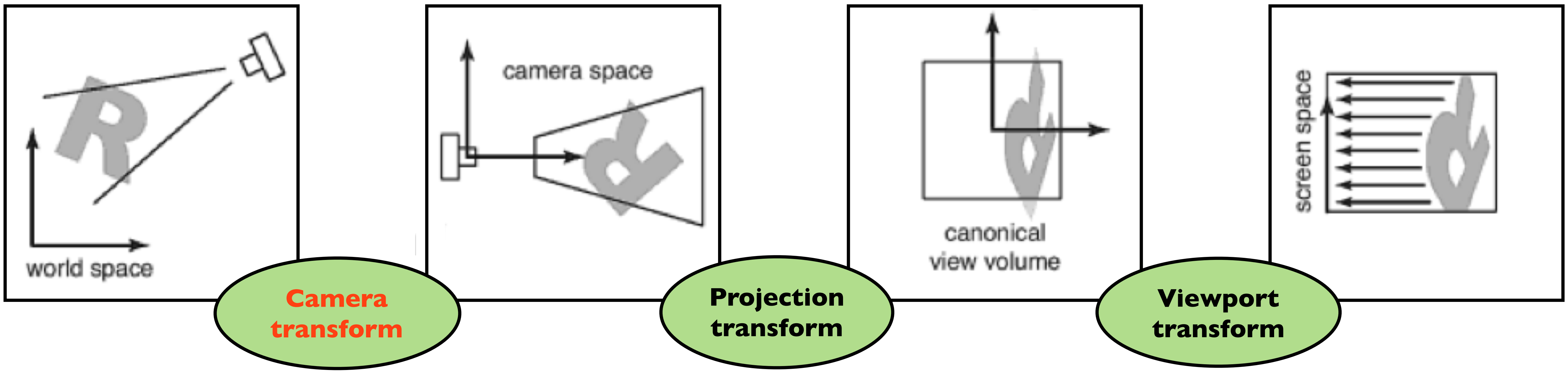


construct M_{vp}
construct M_{orth}
$M = M_{vp} M_{orth}$
for each line segment (a_i, b_i) do
$\mathbf{p} = M\mathbf{a}_i$
$\mathbf{q} = M\mathbf{b}_i$
drawline (x_p, y_p, x_q, y_q)

draw lines specified in <u>camera</u> space

Shirley, Marschner 7.1

Camera Transform



Camera Transform

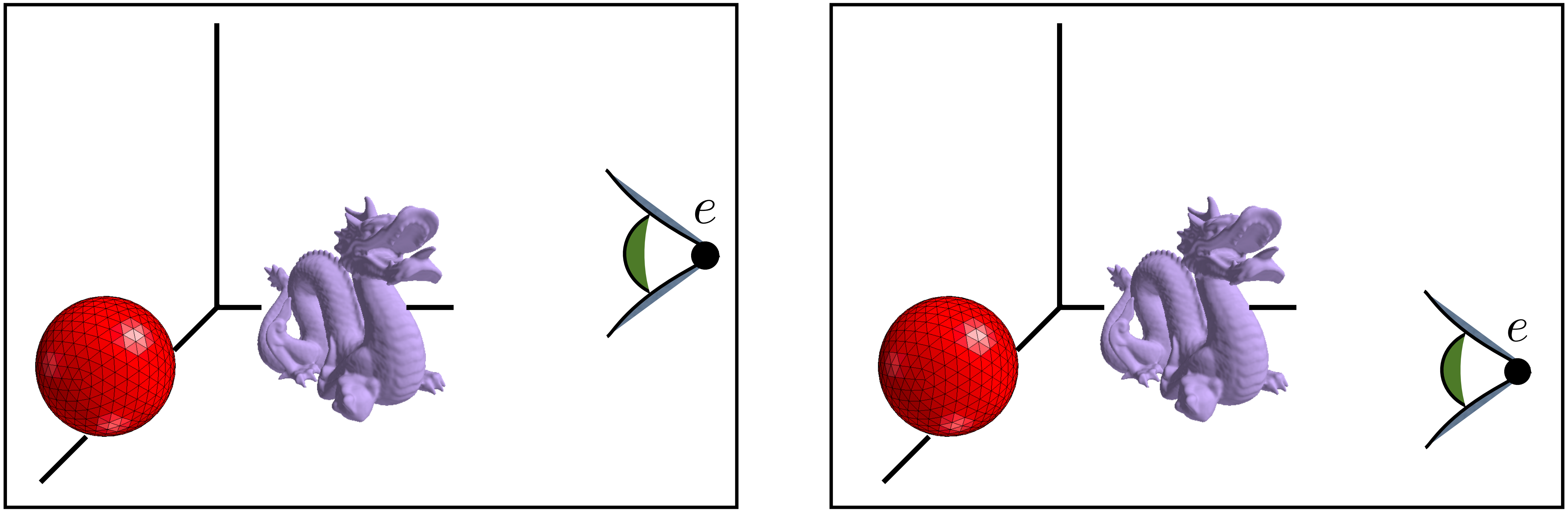
How do we specify the camera configuration?

(orthogonal case)

Camera Transform

How do we specify the camera configuration?

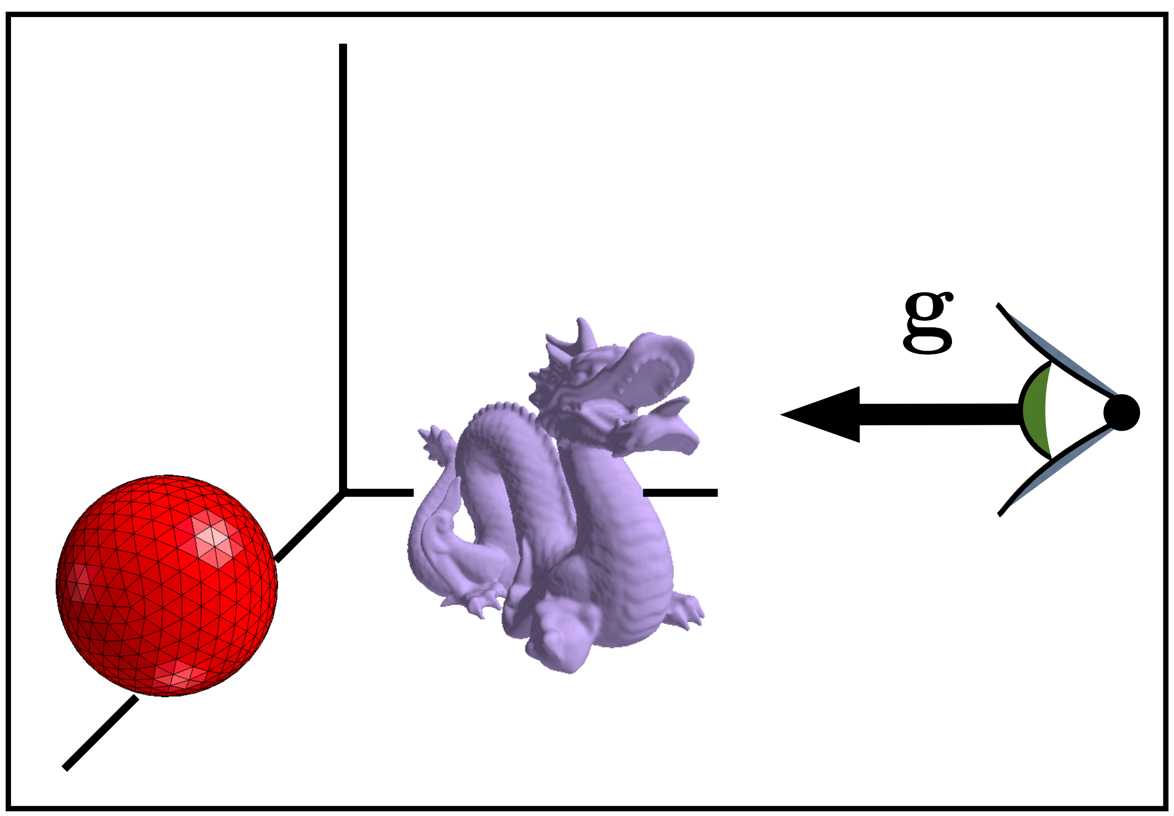
eye position

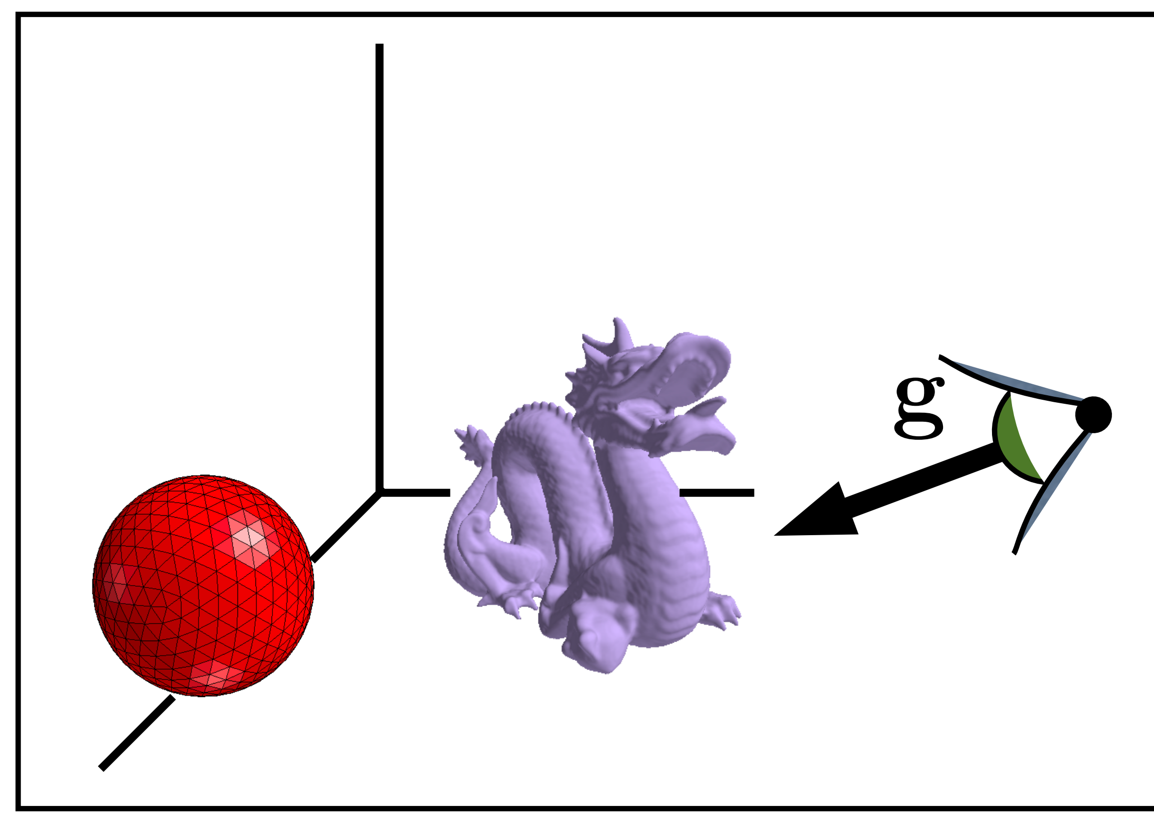


Camera Transform

How do we specify the camera configuration?

gaze direction

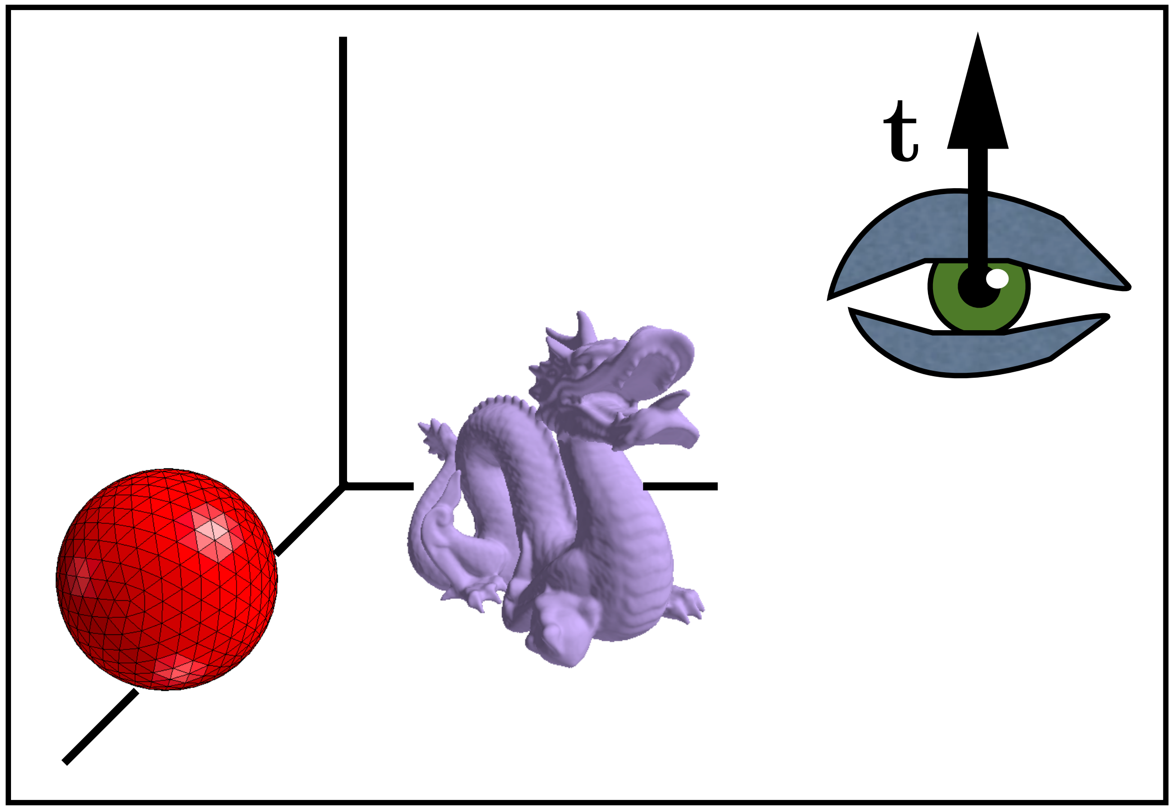


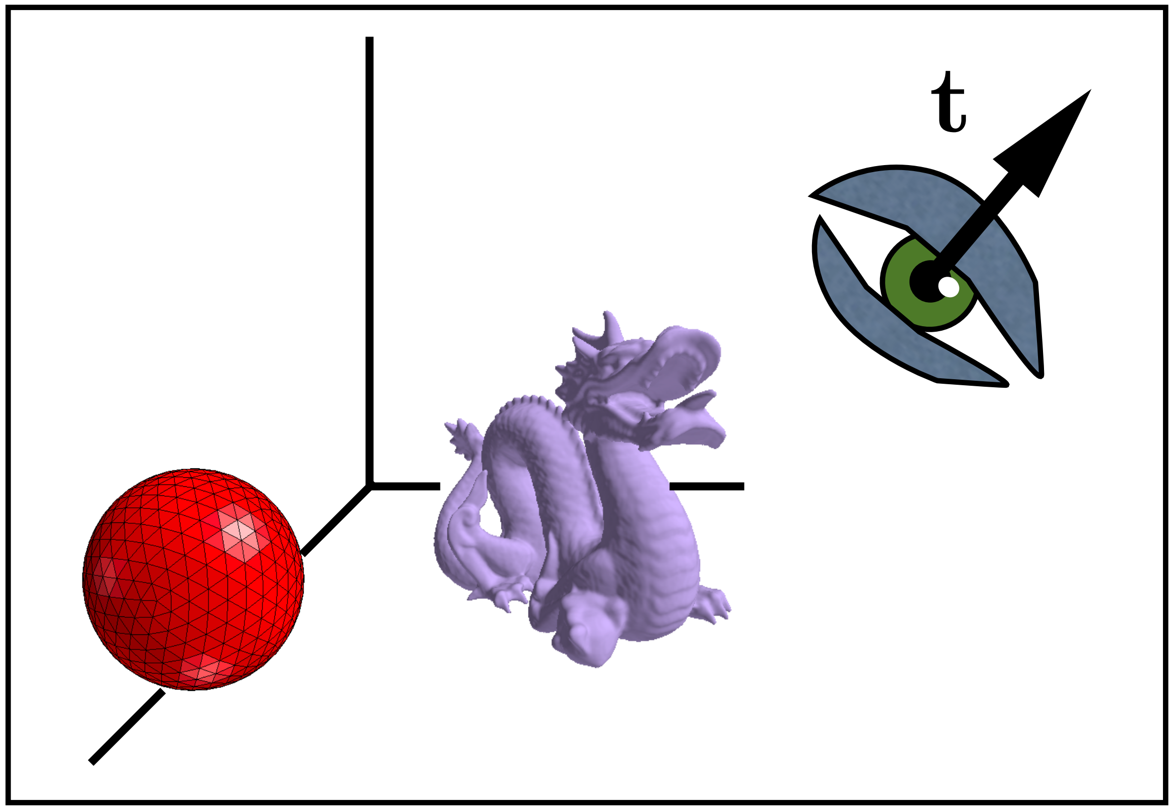


Camera Transform

How do we specify the camera configuration?

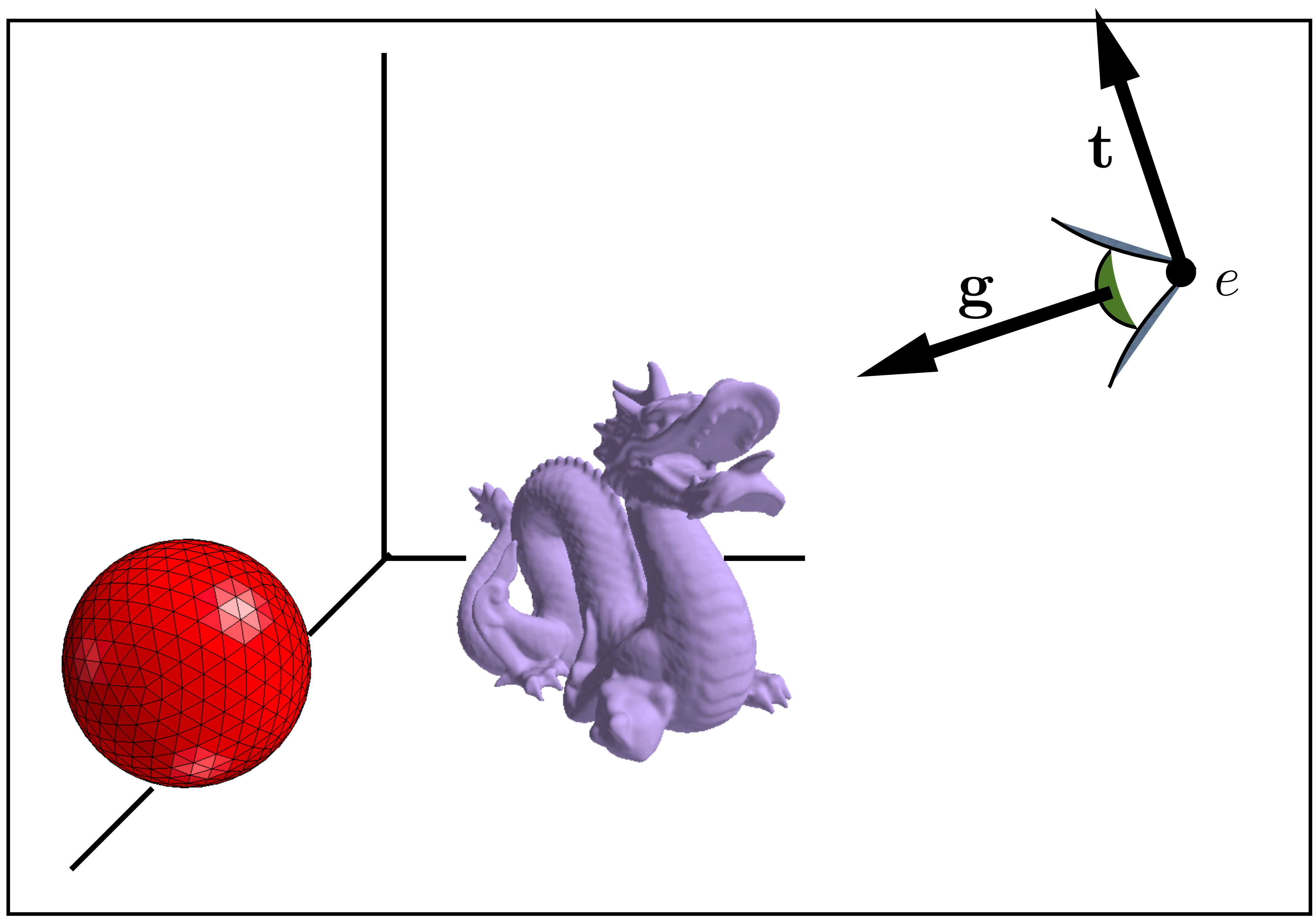
up vector



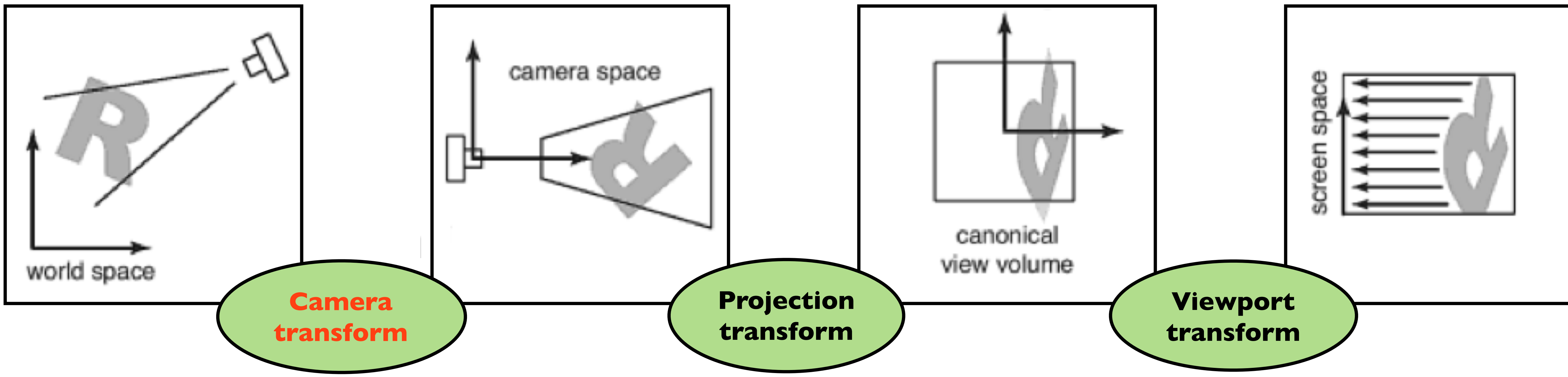


Camera Transform

How do we specify the camera configuration?



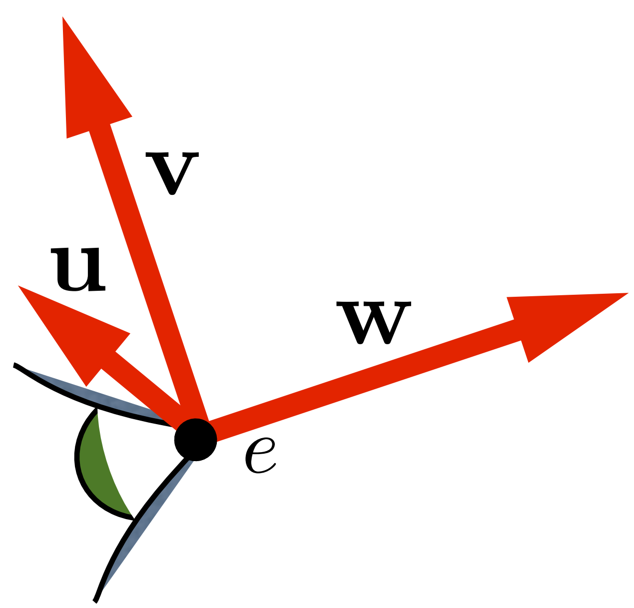
Camera Transform

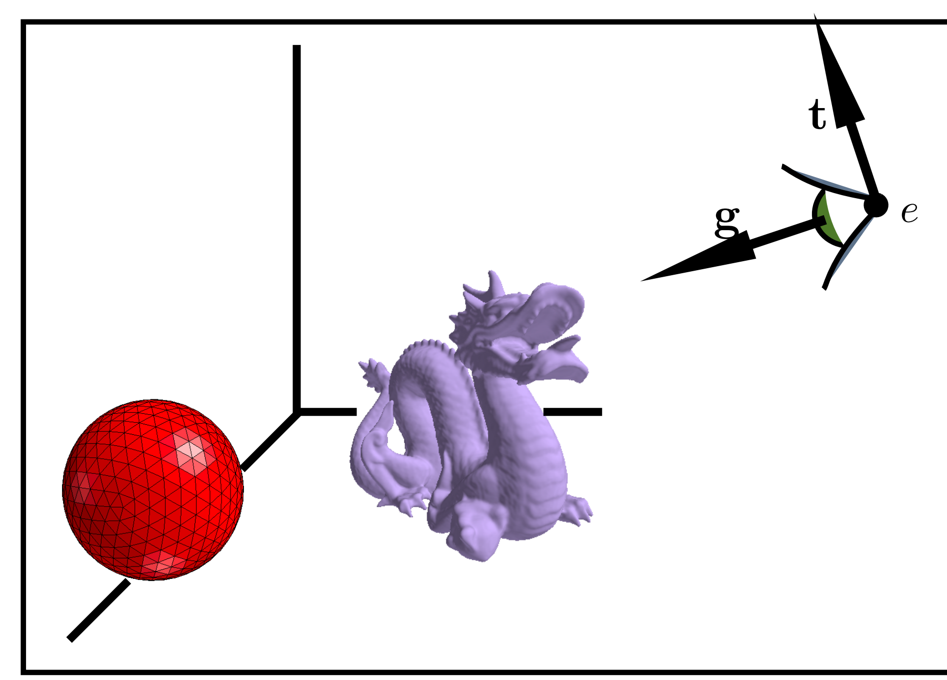


$$\mathbf{u} = \mathbf{v} \times \mathbf{w}$$

$$\mathbf{v} = \frac{\mathbf{t}}{|\mathbf{t}|}$$

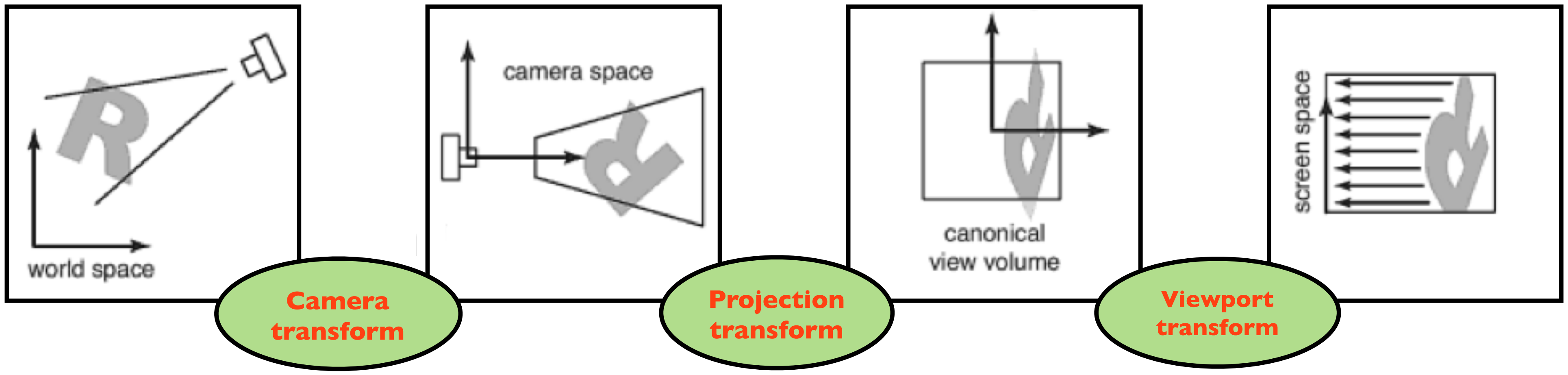
$$\mathbf{w} = -\frac{\mathbf{g}}{|\mathbf{g}|}$$





M_{cam} <whiteboard>

Line drawing algorithm

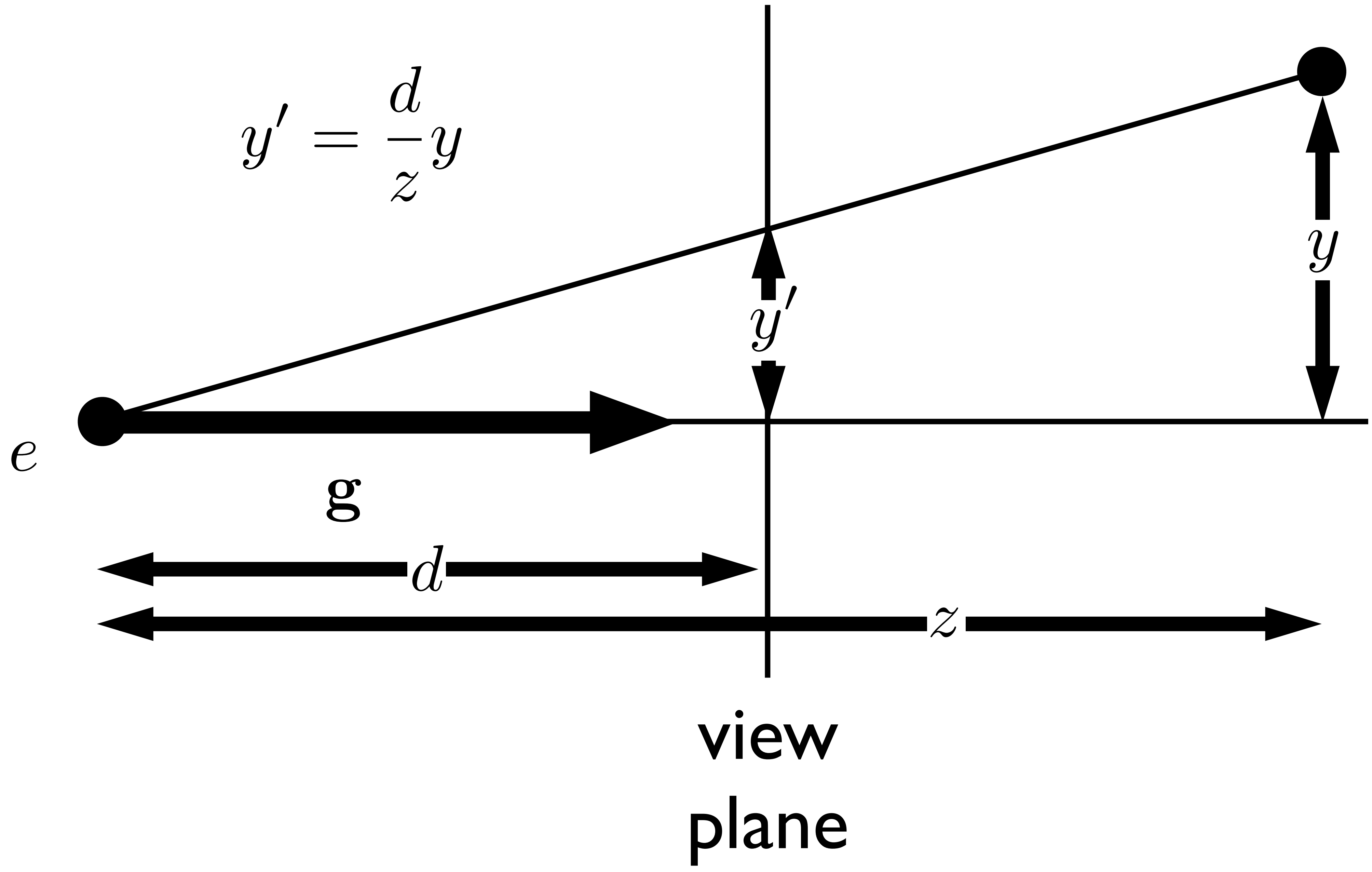


construct M_{vp} M_{cam}
construct M_{orth}
$M = M_{vp} M_{orth} M_{cam}$
for each line segment (a_i, b_i) do
$\mathbf{p} = M\mathbf{a}_i$
$\mathbf{q} = M\mathbf{b}_i$
drawline (x_p, y_p, x_q, y_q)

Shirley, Marschner 7.1

draw lines specified in <u>world</u> space

Projective Transformations



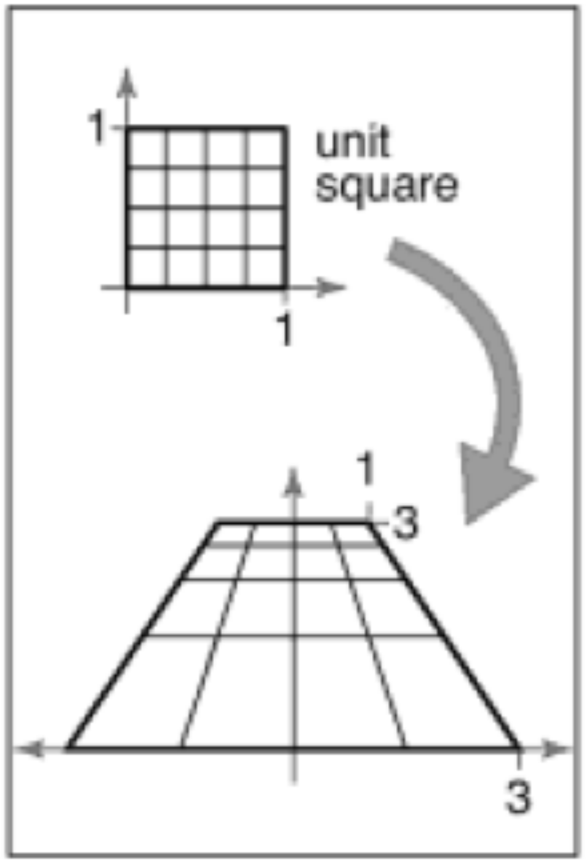
note that the height, **y'**, in **camera space** is proportion to y and inversely proportion to z. We want to be able to specify such a transformation with our **4x4 matrix machinery**

Projective Transformations

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ w \end{pmatrix} \rightarrow \quad \begin{aligned} x &= \frac{\tilde{x}}{w} \\ y &= \frac{\tilde{y}}{w} \\ z &= \frac{\tilde{z}}{w} \end{aligned}$$

Example:

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



Shirley, Marschner

Note: this makes our homogeneous representation for unique only **up to a constant**

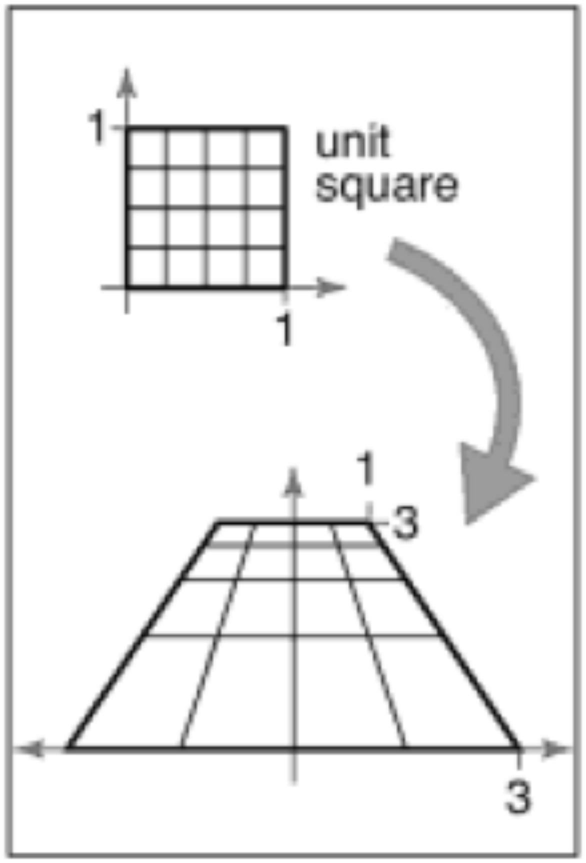
Projective Transformations

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

We can now implement perspective projection!

Example:

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

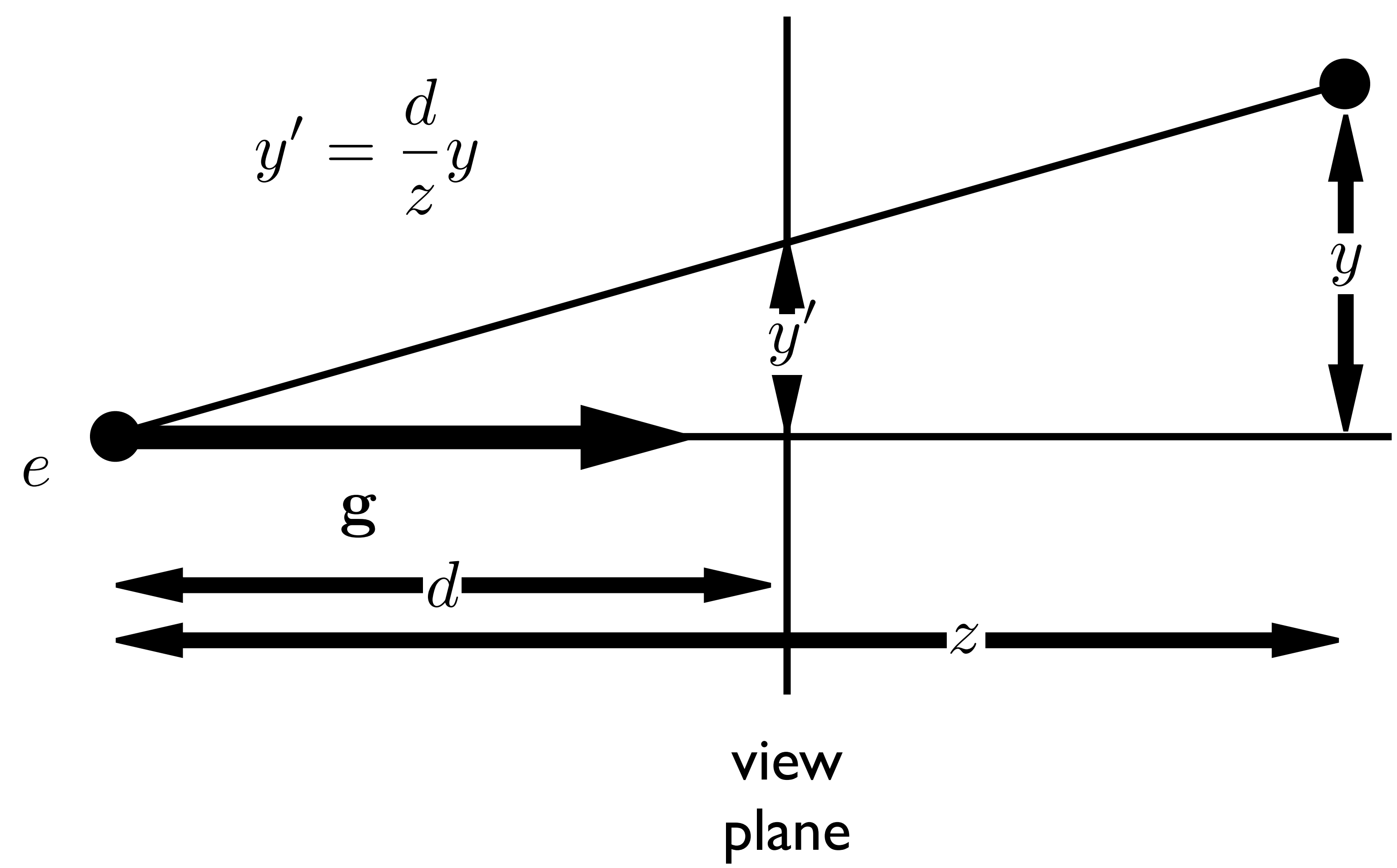


Shirley, Marschner

Perspective Projection

$$\begin{pmatrix} dy \\ z \end{pmatrix} = \begin{pmatrix} d & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} y \\ z \\ 1 \end{pmatrix}$$

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

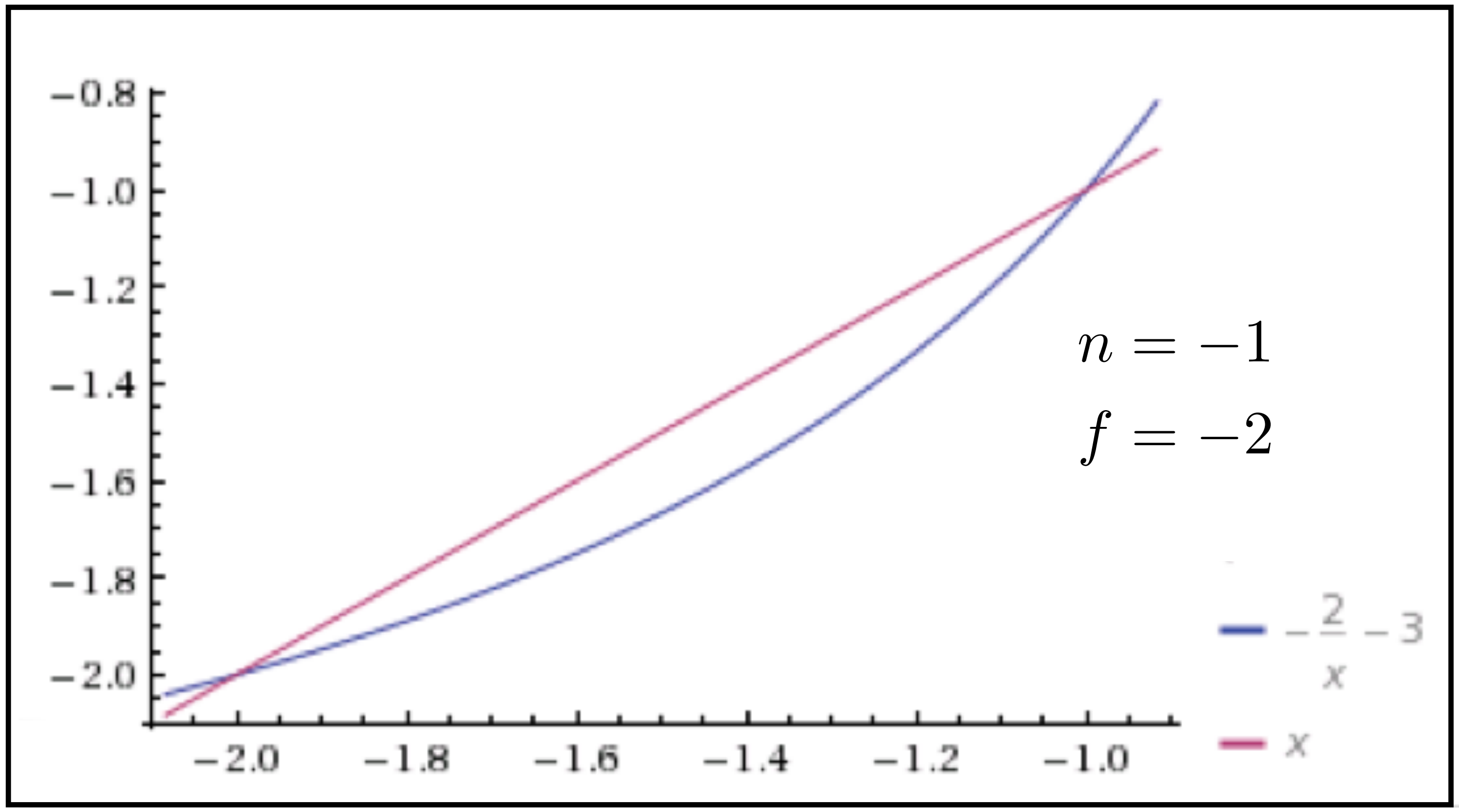


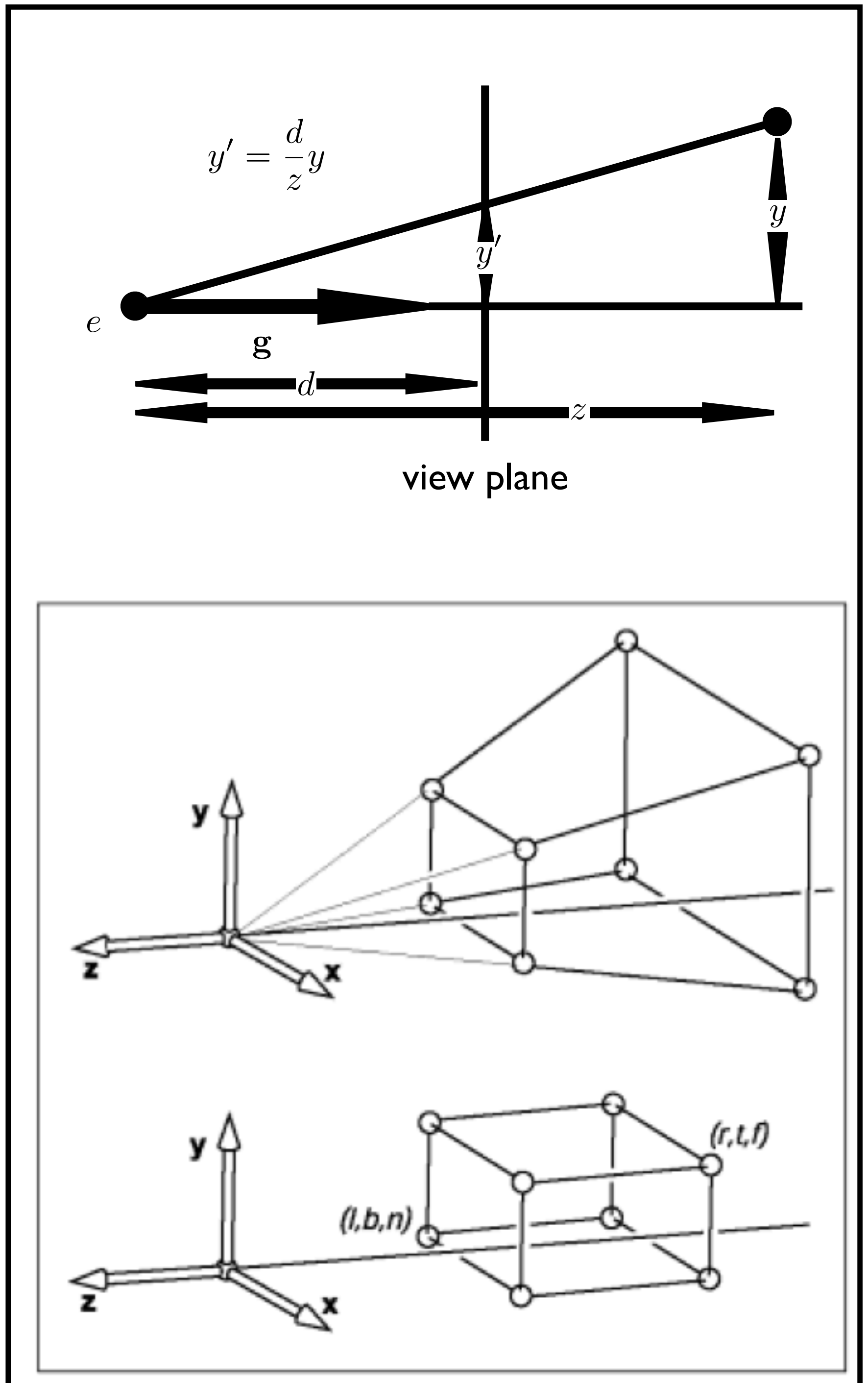
note that the height, **y’**, in **camera space** is proportion to y and inversely proportion to z. We want to be able to specify such a transformation with our **4x4 matrix machinery**

Perspective Projection

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

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

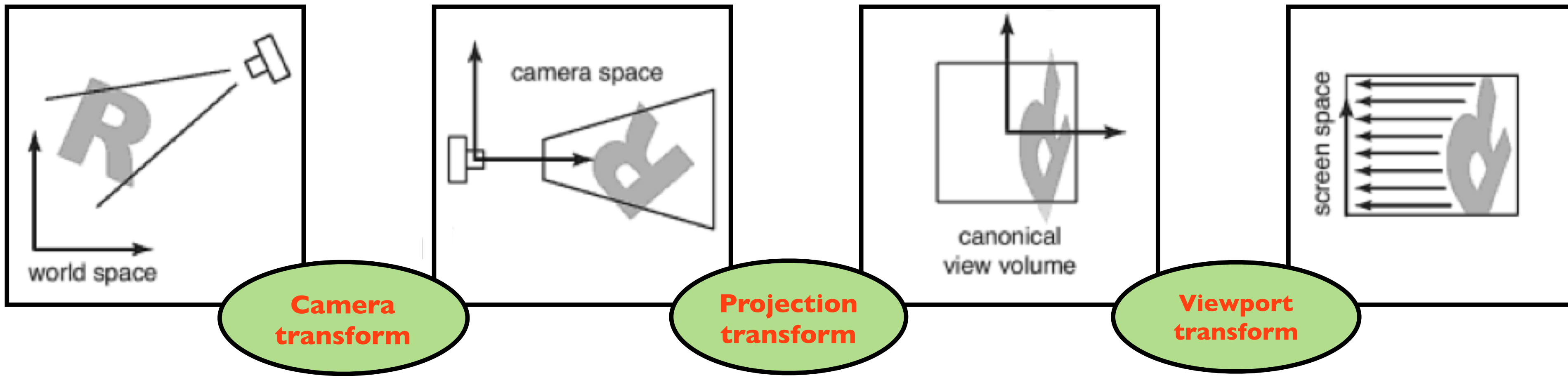




Shirley, Marschner

This does not preserve **z** completely, but it preserves **z = n, f** and is **monotone** (preserves ordering) with respect to z

Line drawing algorithm



construct M_{vp} M_{cam}

construct M_{per}

$M = M_{vp} M_{per} M_{cam}$

for each line segment (a_i, b_i) do

$\mathbf{p} = M\mathbf{a}_i$

$\mathbf{q} = M\mathbf{b}_i$

drawline $(x_p/w_p, y_p/w_p, x_q/w_p, y_q/w_p)$

Shirley, Marschner 7.1

draw lines specified in <u>world</u> space