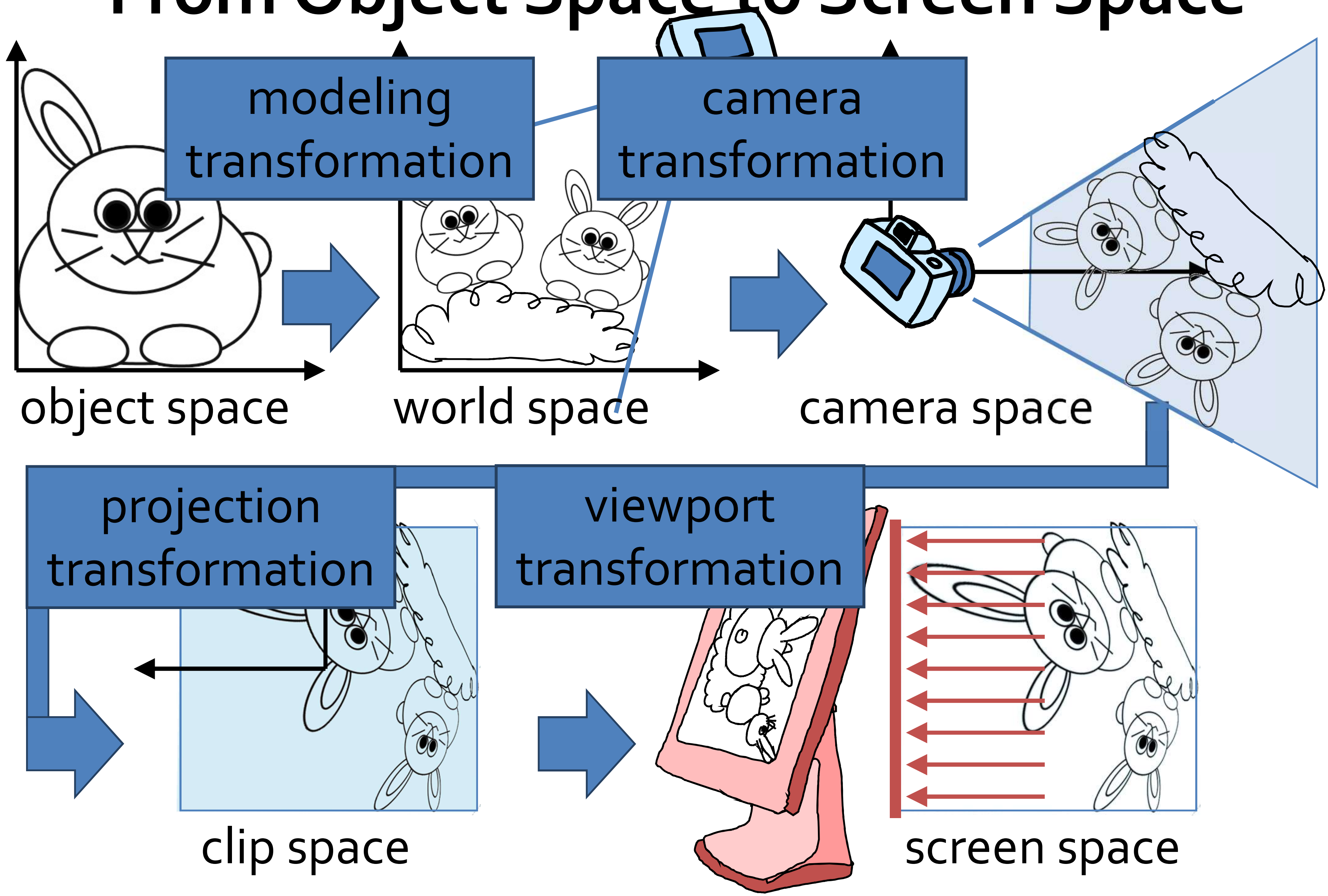
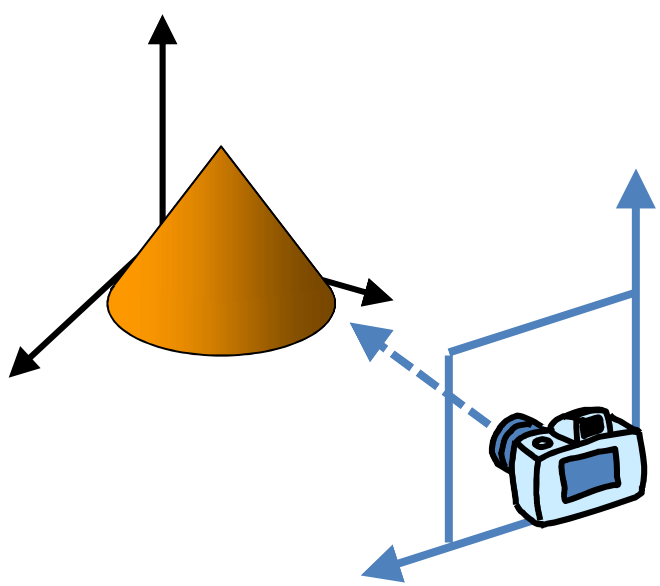
Viewing

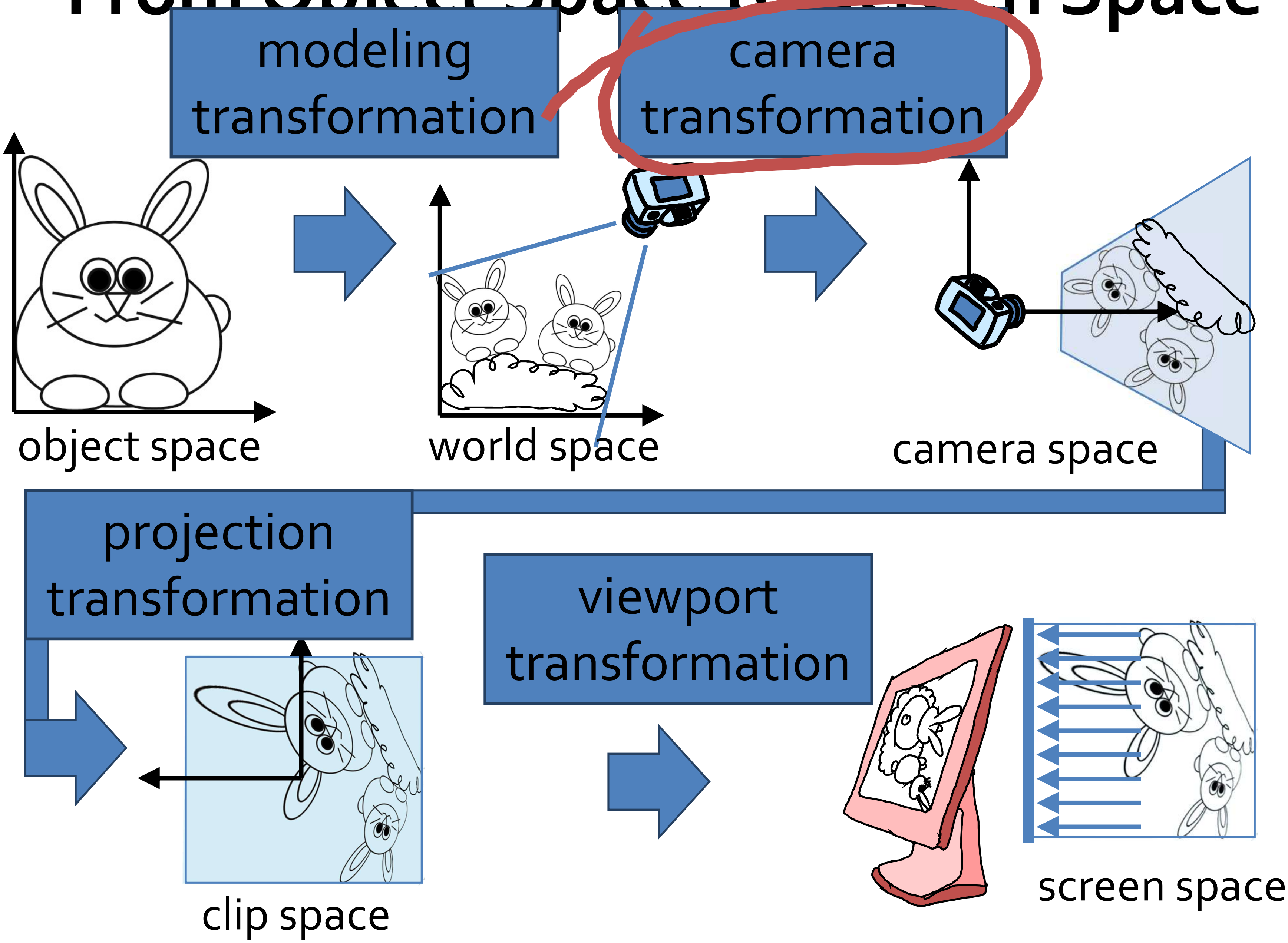
From Object Space to Screen Space



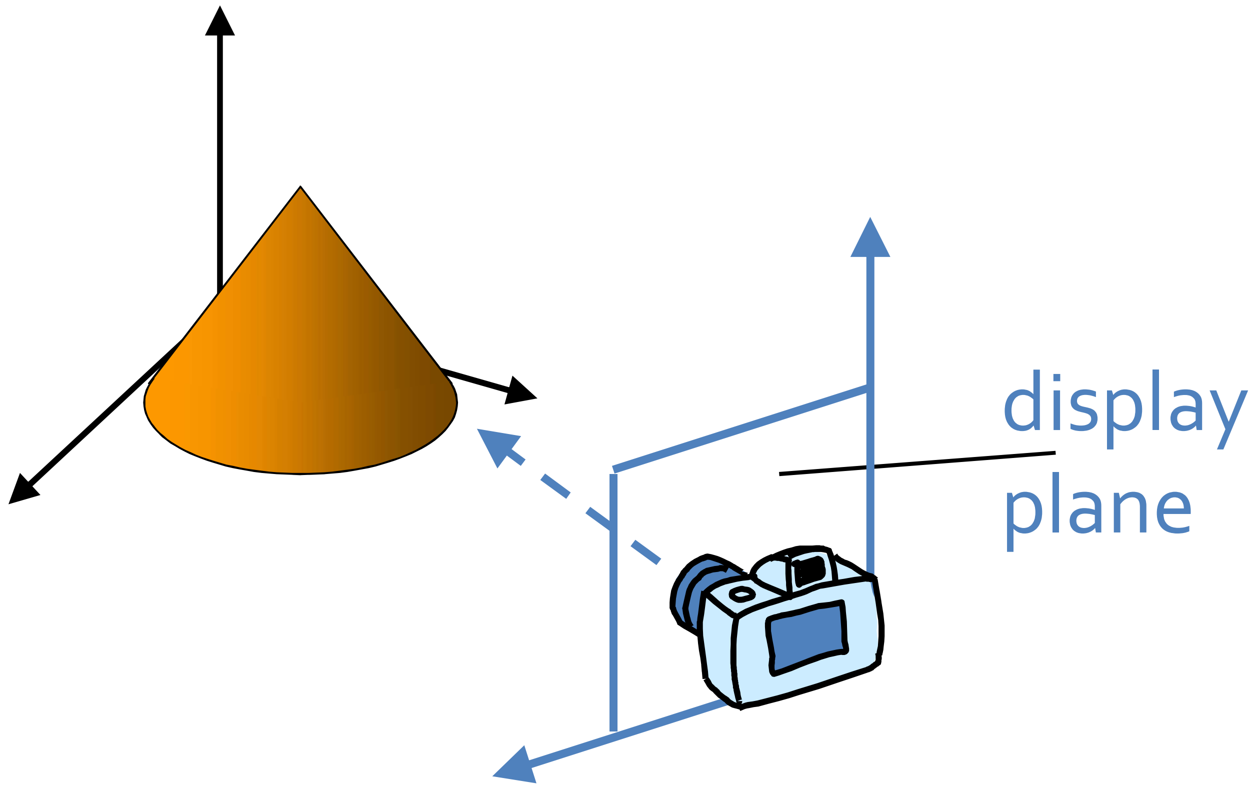
Camera Transformation



From Object Space to Screen Space



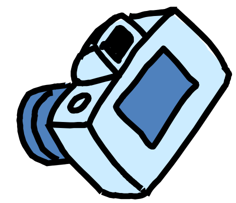
Viewing: Projection Plane

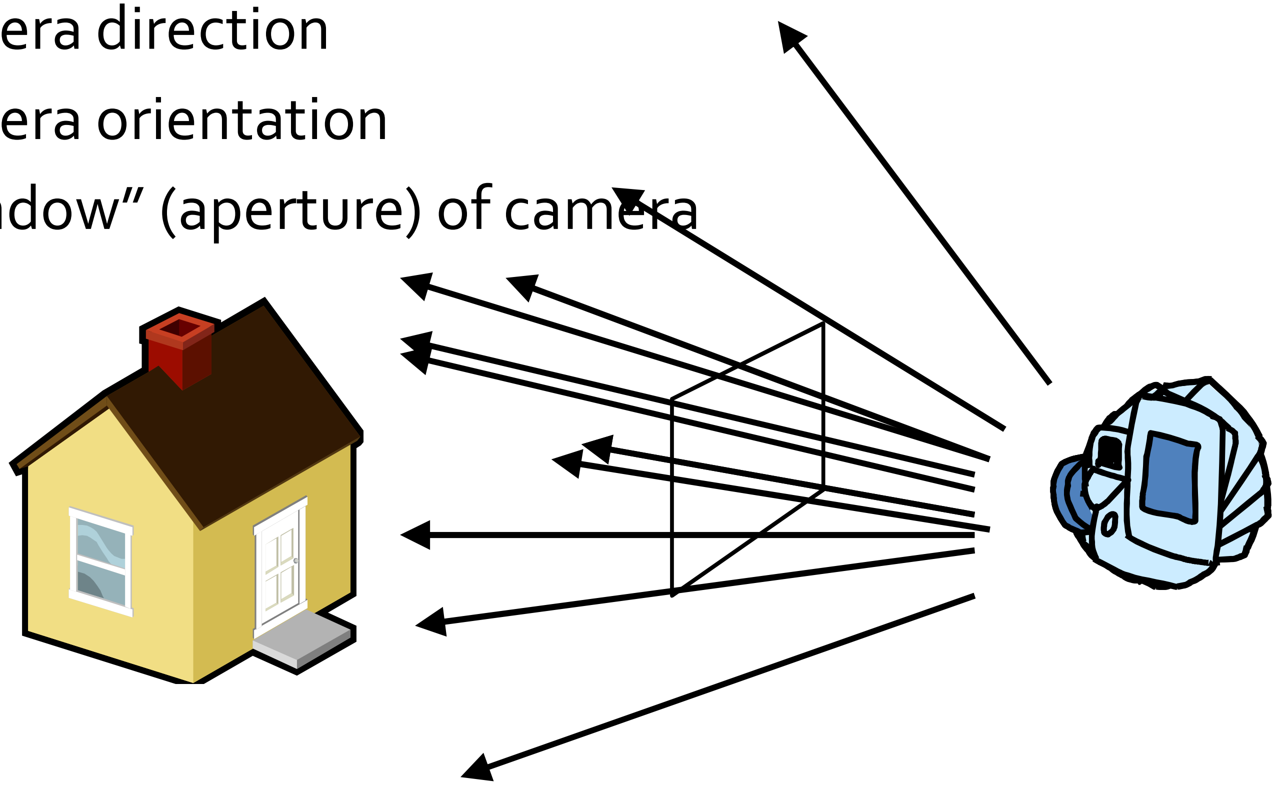


coordinate reference for obtaining a selected view of a 3D scene

Viewing: Camera Definition

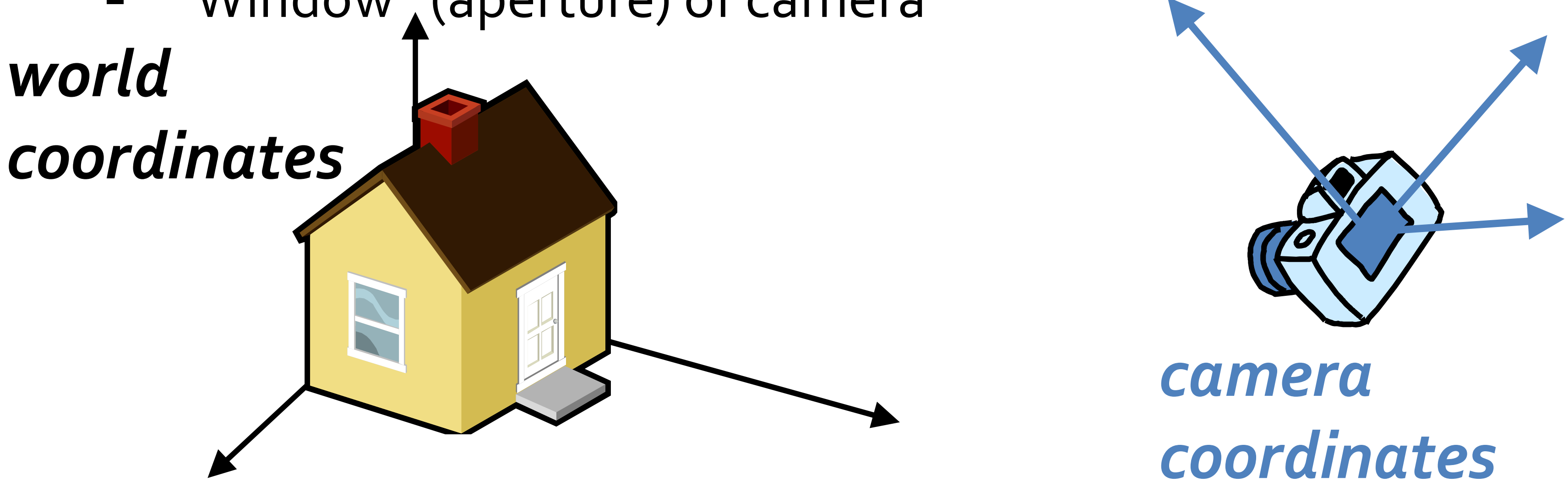
- Similar to taking a photograph
- Involves selection of
  - Camera position
  - Camera direction
  - Camera orientation
  - “Window” (aperture) of camera





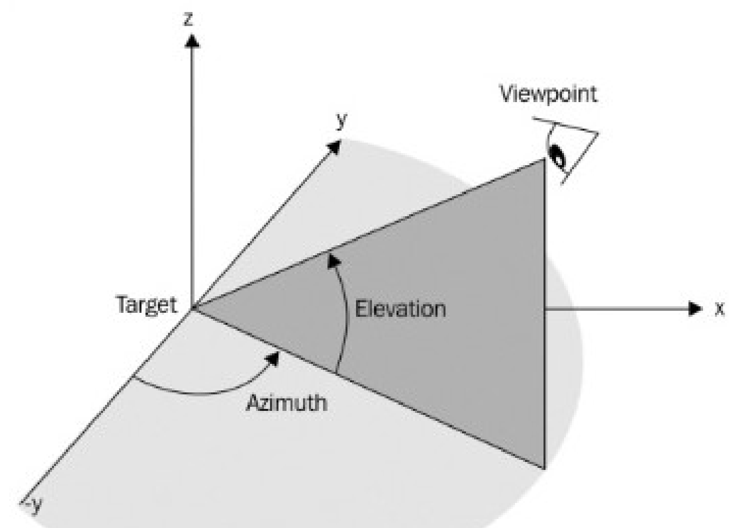
Viewing: Camera Definition

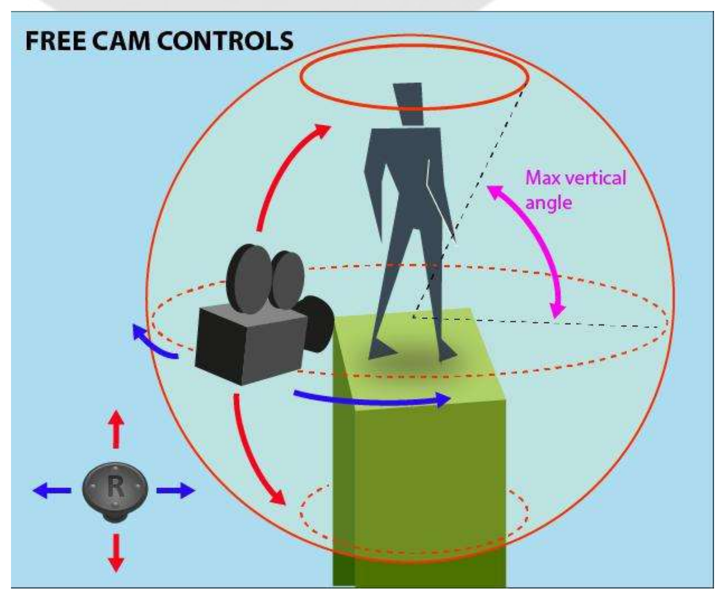
- Similar to taking a photograph
- Involves selection of
  - Camera position
  - Camera direction
  - Camera orientation
  - “Window” (aperture) of camera



Orbiting Camera

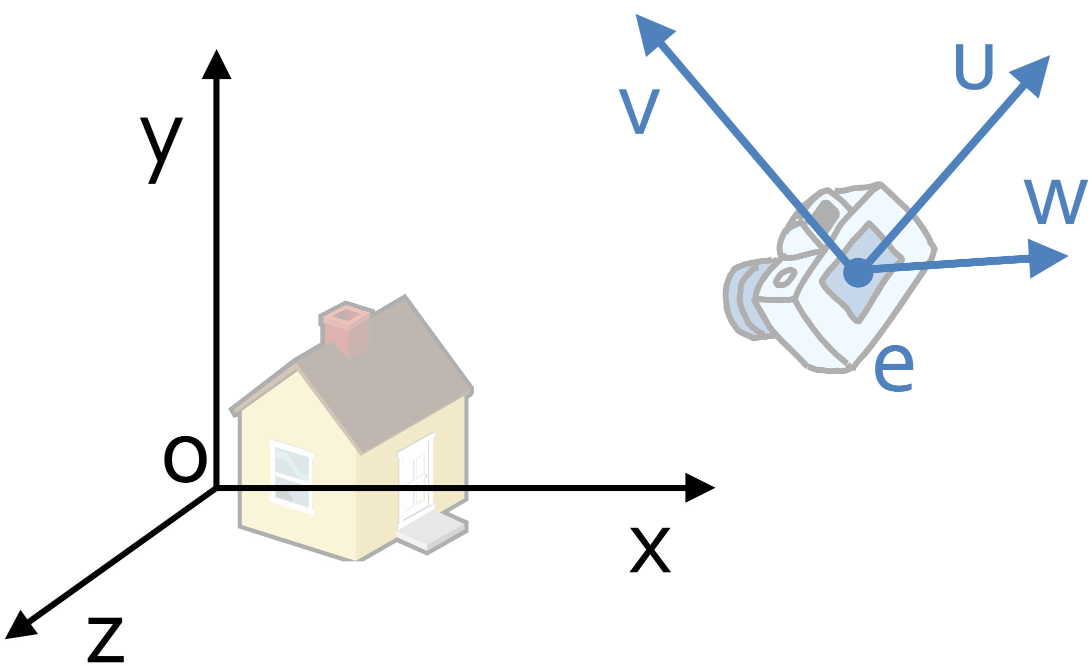
- Often defined by
  - Target
  - Distance to target
  - Azimuth
  - Elevation





Viewing: Camera Transformation (1)

- View reference point
  - Origin of camera coordinate system
  - Camera position or look-at point



right-handed camera-coordinate system, with axes u, v, w, relative to world-coordinate scene

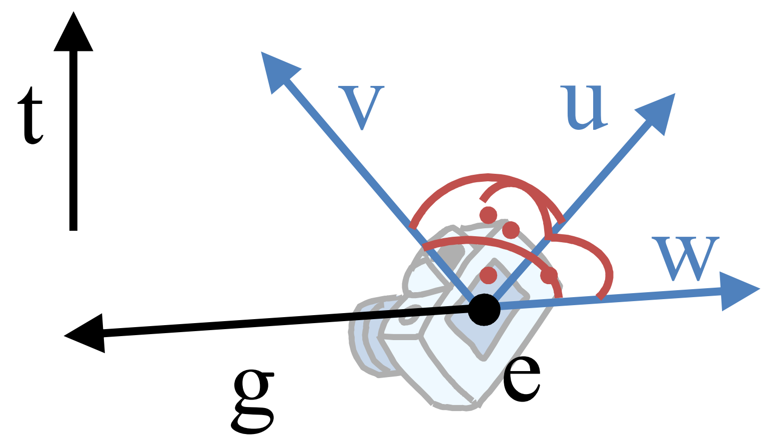
Viewing: Camera Transformation (2)

- e … eye position
- g … gaze direction
  (positive w-axis points to the viewer)
- t … view-up vector

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

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

$$v = w \times u$$

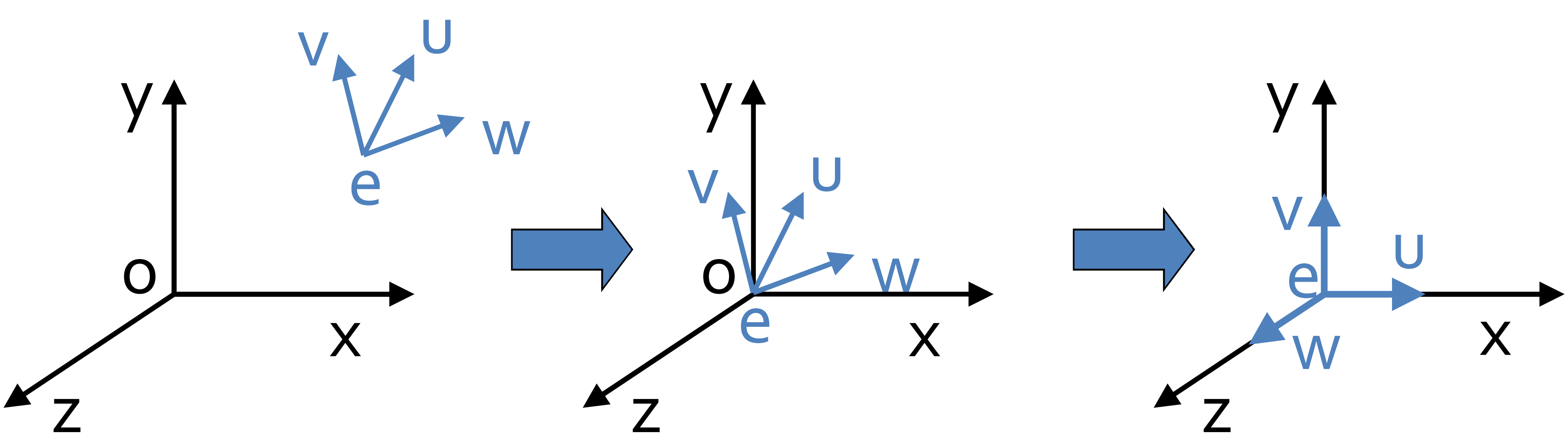


Viewing: Camera Transformation (4)

$$\mathbf{M}_{\text{cam}} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

alternative calculation of $\mathbf{M}_{\mathbf{cam}}$ for aligning viewing system with world-coordinate axes using axis vectors

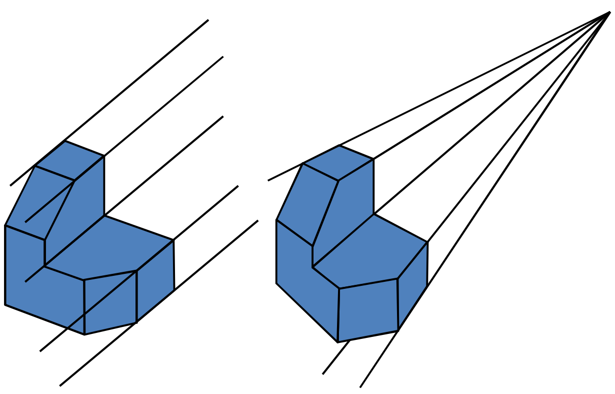
Viewing: Camera Transformation (3)

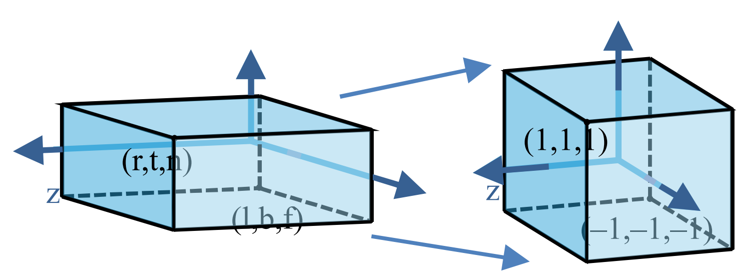


$$\mathbf{M_{cam} = R_z \cdot R_y \cdot R_x \cdot T}$$

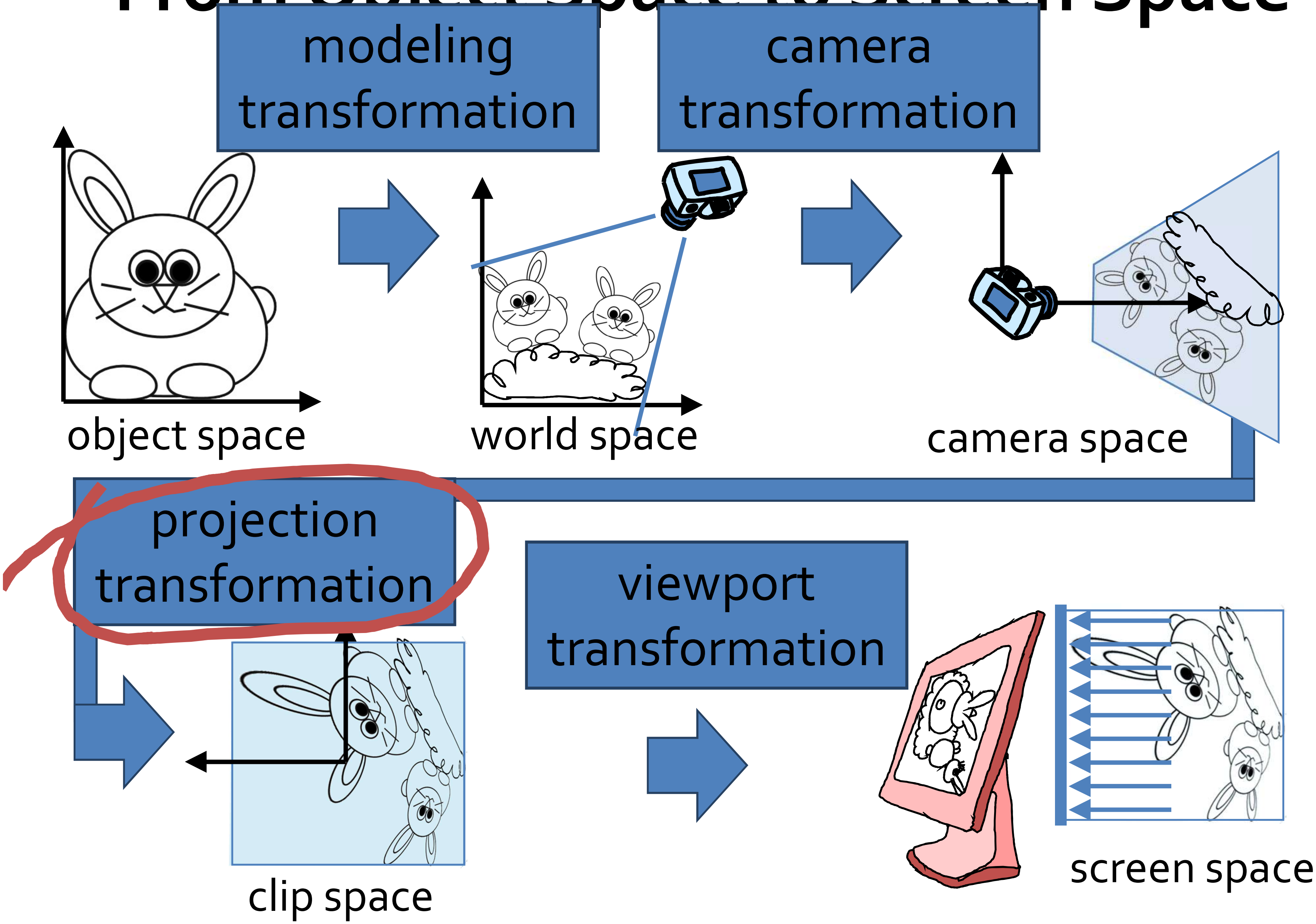
aligning viewing system with world-coordinate axes using translate-rotate transformations

Projection Transformation

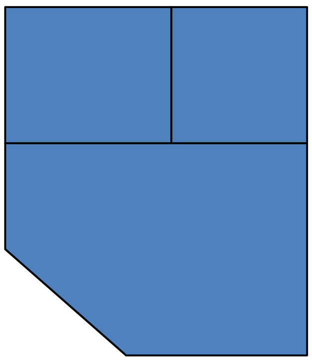




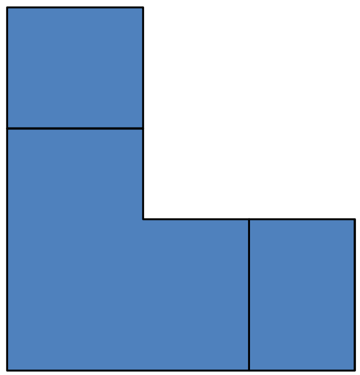
From Object Space to Screen Space



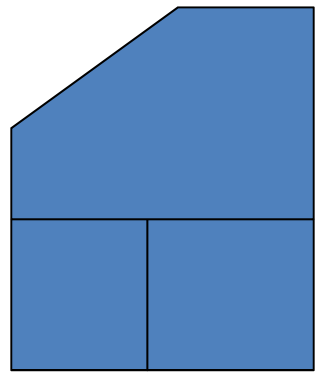
Parallel Projection
(Orthographic Projection)



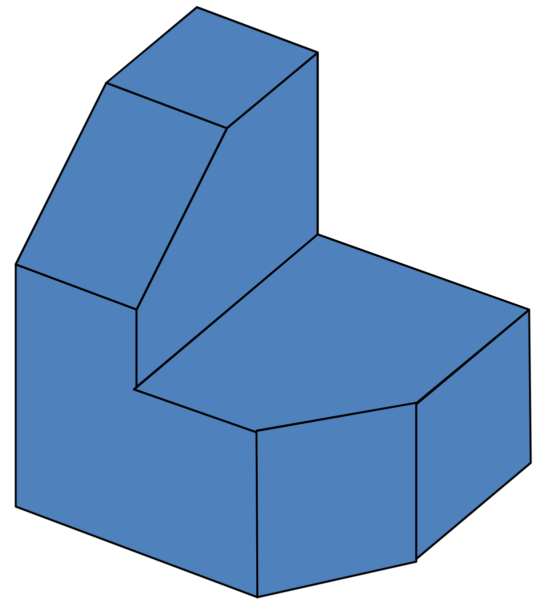
top



side

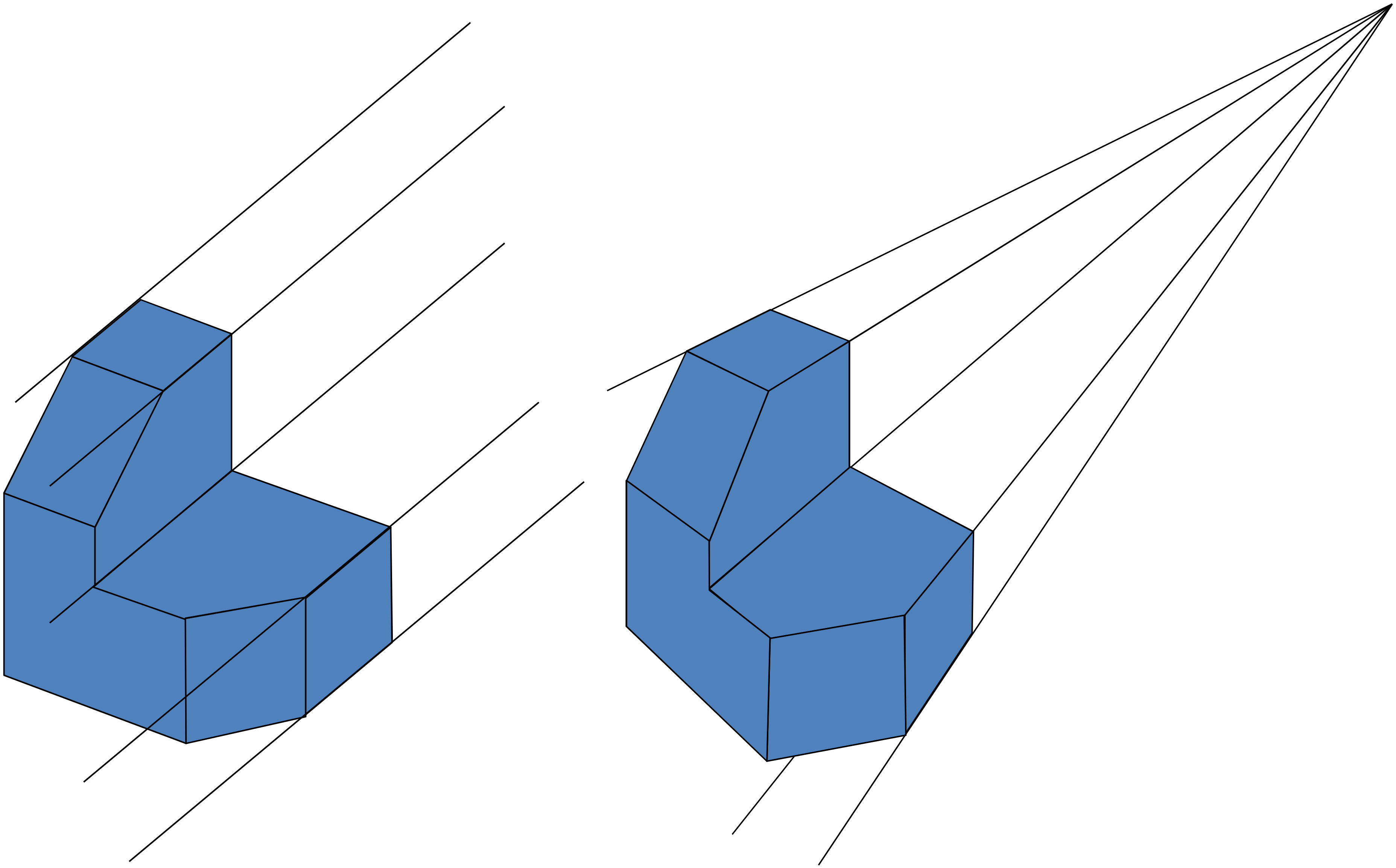


front

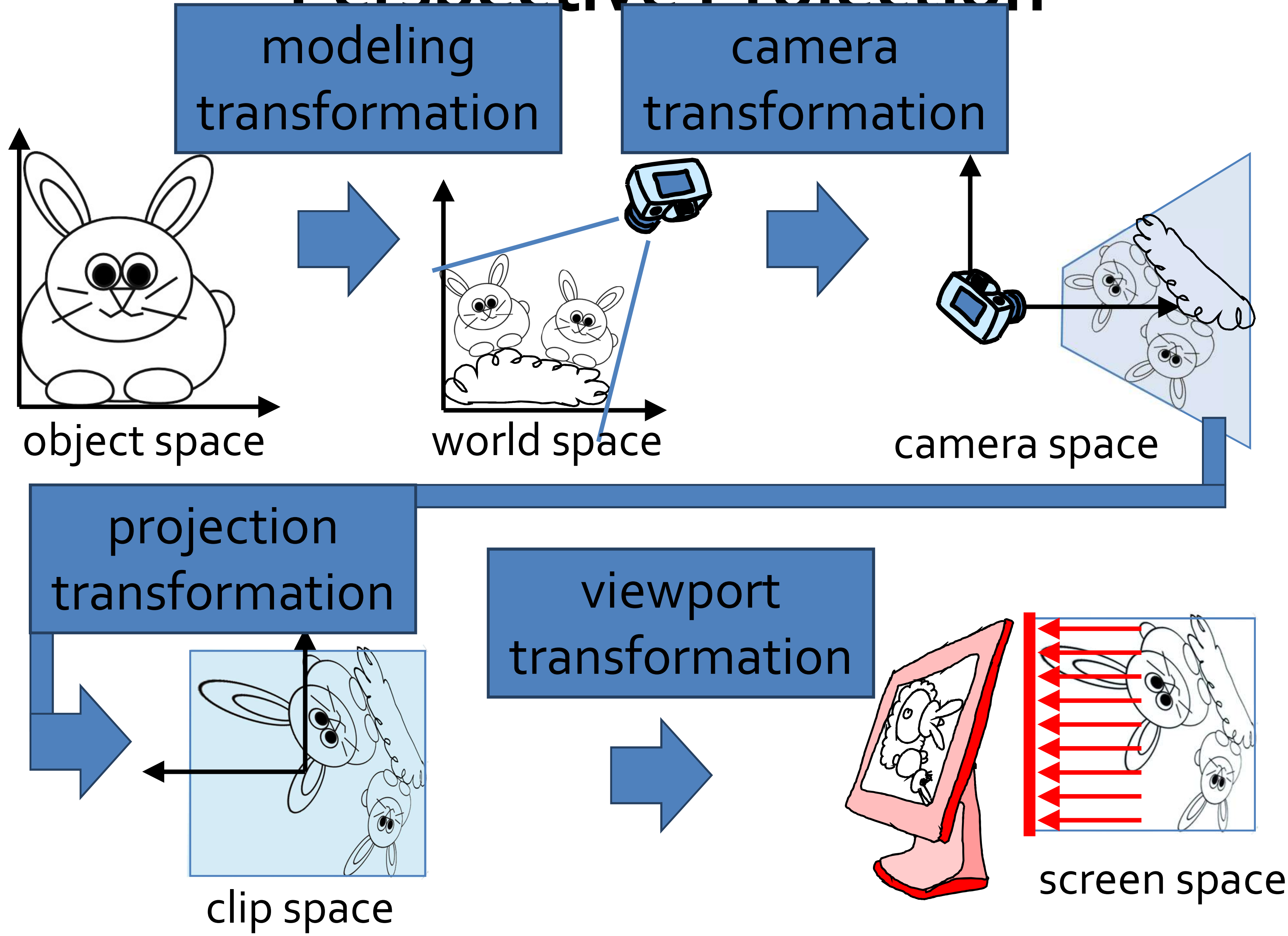


3 parallel-projection views of an object, showing relative proportions from different viewing positions

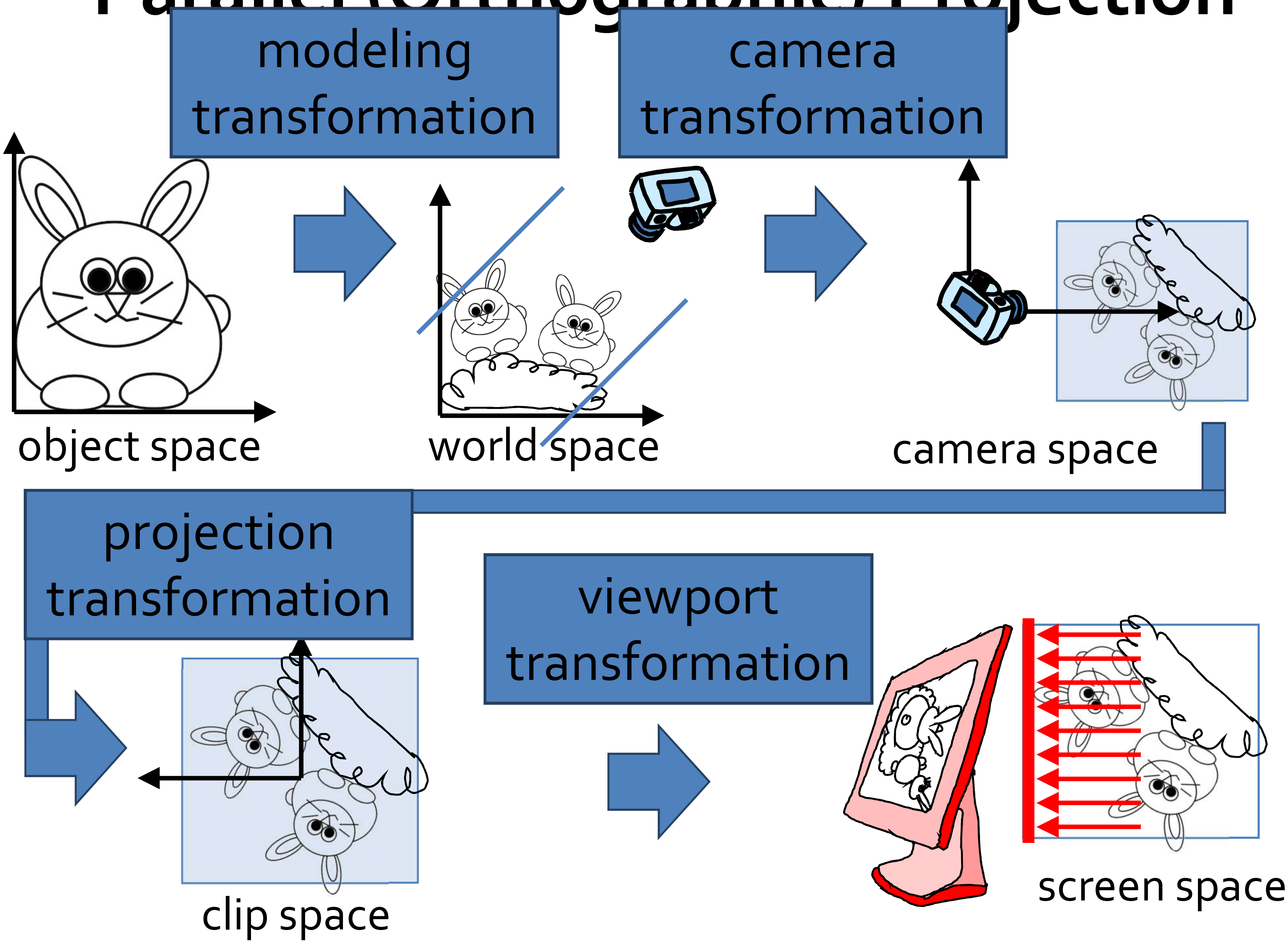
Parallel vs. Perspective Projection



Perspective Projection



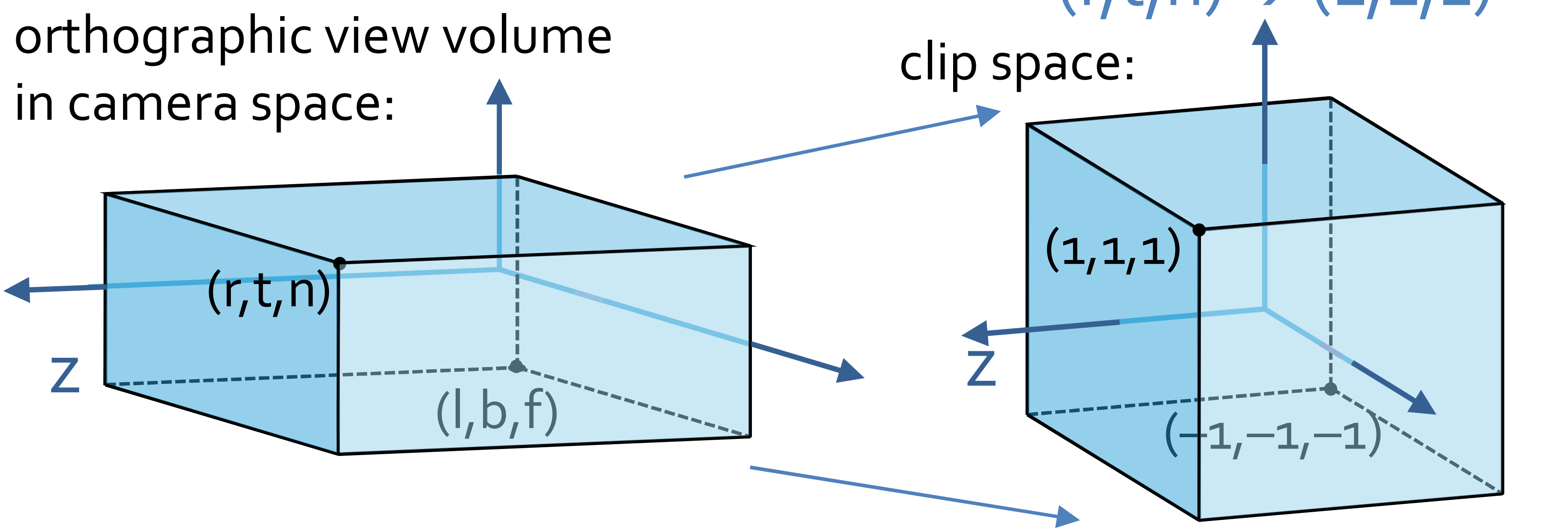
Parallel (Orthographic) Projection



Projection Transformation (Orthographic)

- Assumption: scene in box $[l,r]\times[b,t]\times[f,n]$
- Orthographic camera looking in $-z$ direction
- Transformation to clip space

(l,b,f) → (−1,−1,−1)
(r,t,n) → (1,1,1)



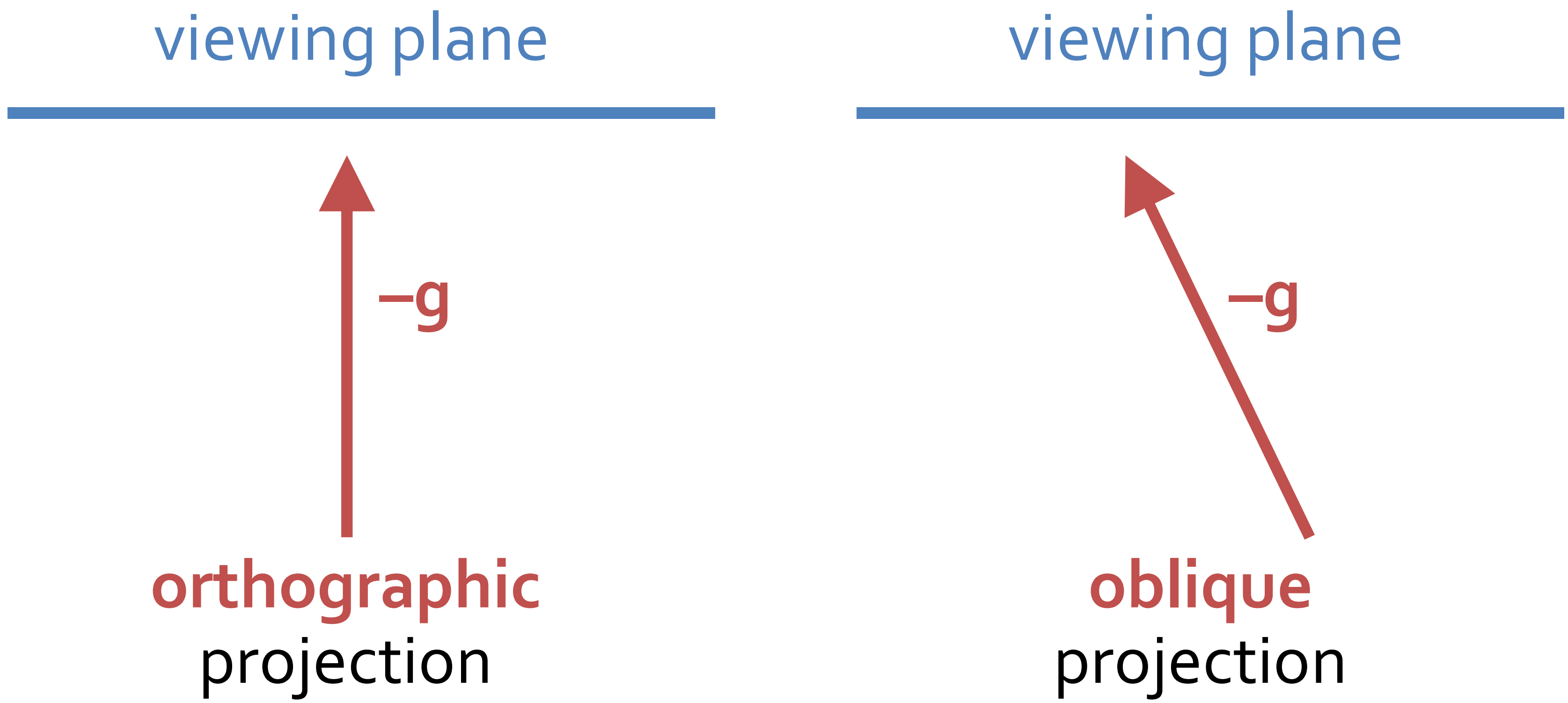
Projection Transformation (Orthographic)

(l,b,f) → (−1,−1,−1)

(r,t,n) → (1,1,1)

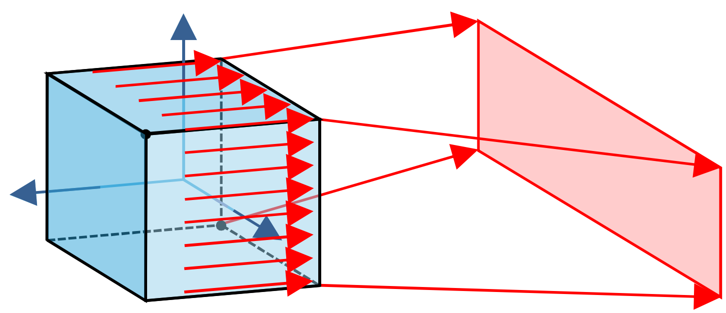
$$M_{orth} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parallel Projection (1)

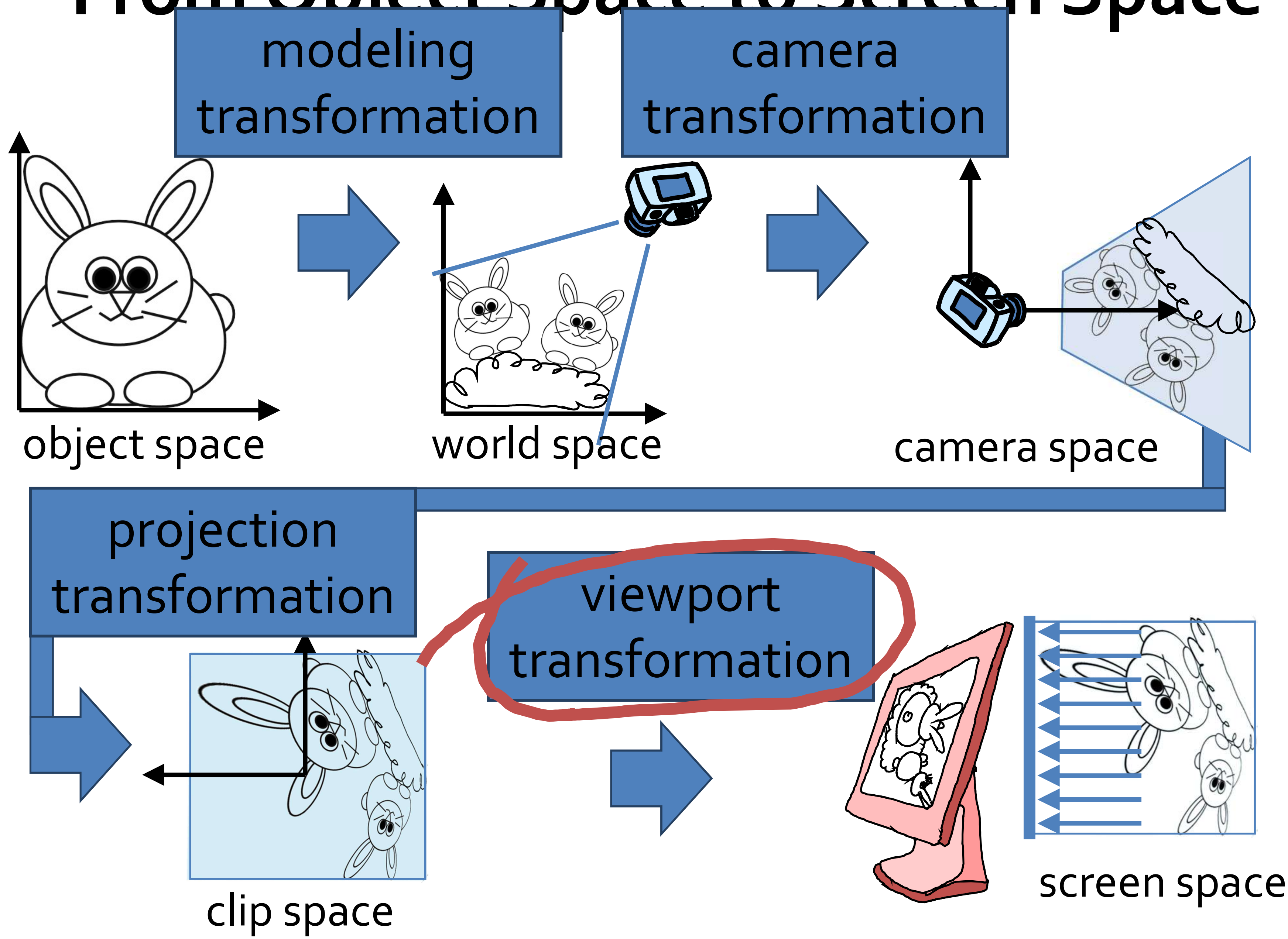


orientation of the projection vector −g

Viewport Transformation

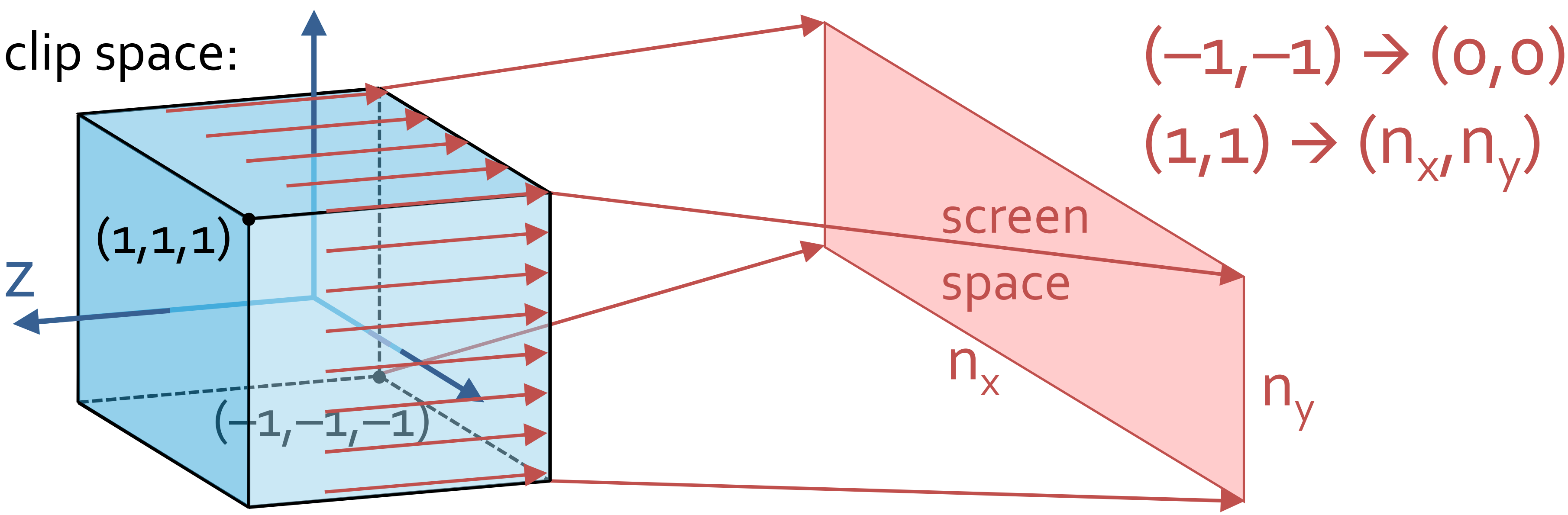


From Object Space to Screen Space



Viewport Transformation (1)

- Assumption: scene is in clip space !
- Clip space = [(–1, –1, –1), (1, 1, 1)]
- Orthographic camera looking in –z direction
- Screen resolution $n_x \times n_y$ pixels



Viewport Transformation (2)

can be done with the matrix

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ 1 \end{bmatrix} = \begin{bmatrix} n_x/2 & 0 & n_x/2 \\ 0 & n_y/2 & n_y/2 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$(-1,-1) \rightarrow (0,0)$

$(1,1) \rightarrow (n_x, n_y)$

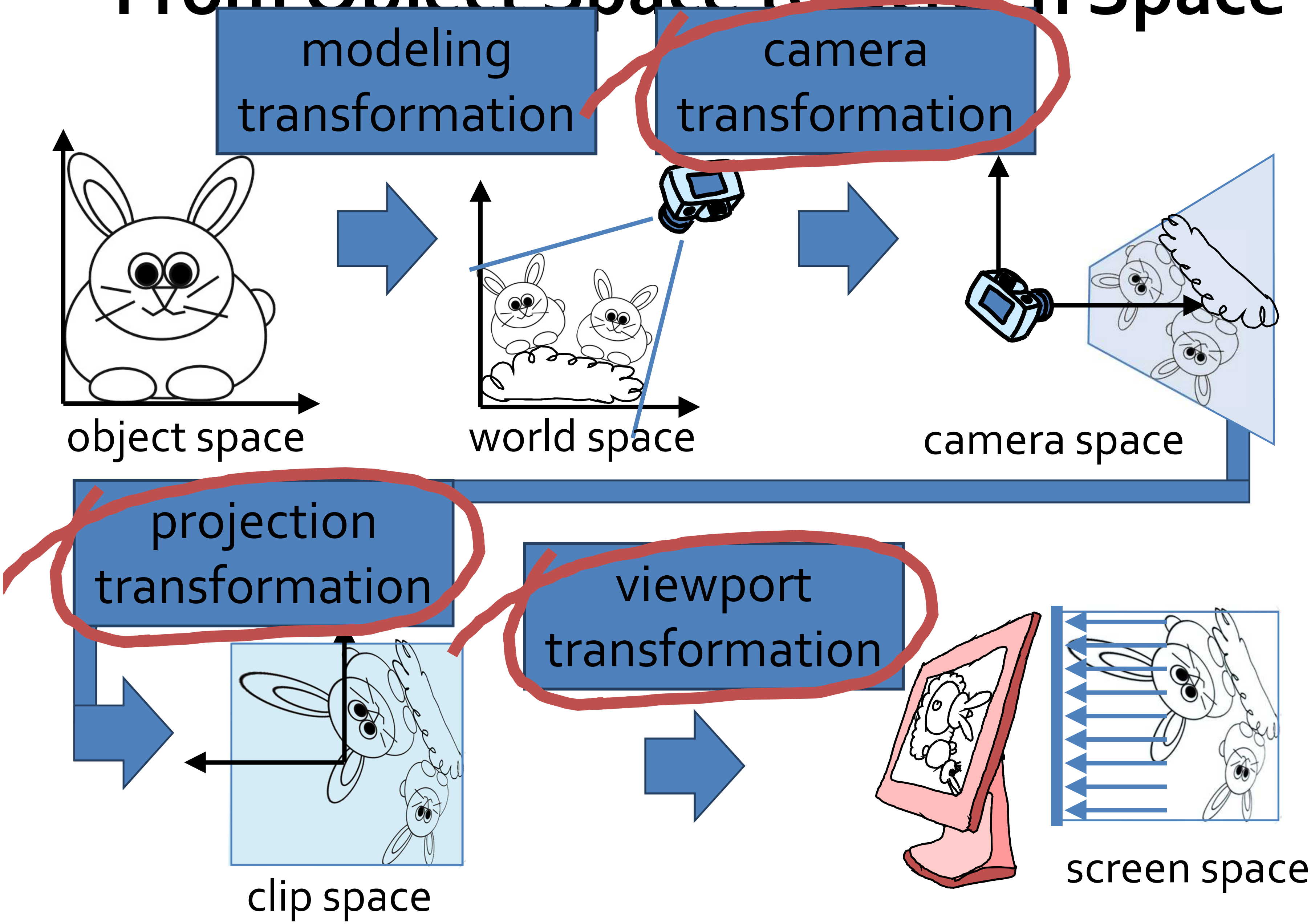
this ignores the z-coordinate, but...

Viewport Transformation (3)

- … we will need z later to remove hidden parts of the image, so we add a row and column to keep z

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} n_x/2 & 0 & 0 & n_x/2 \\ 0 & n_y/2 & 0 & n_y/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{M_{vp}}$$

From Object Space to Screen Space



Viewing: Camera + Projection + Viewport

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = \left(M_{vp} \cdot M_{orth} \cdot M_{cam}\right) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

pixels on the screen

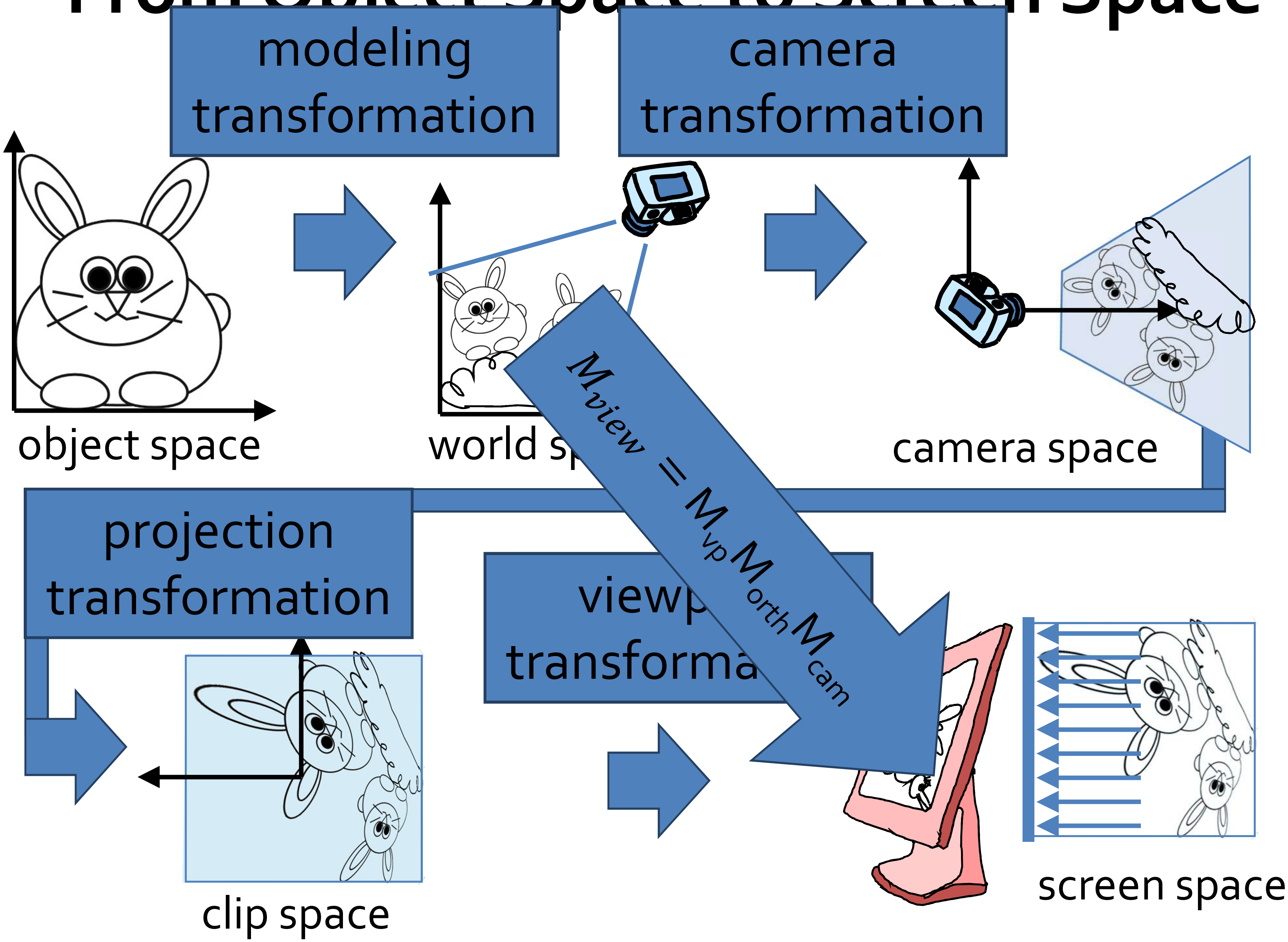
viewport transformation

projection transformation

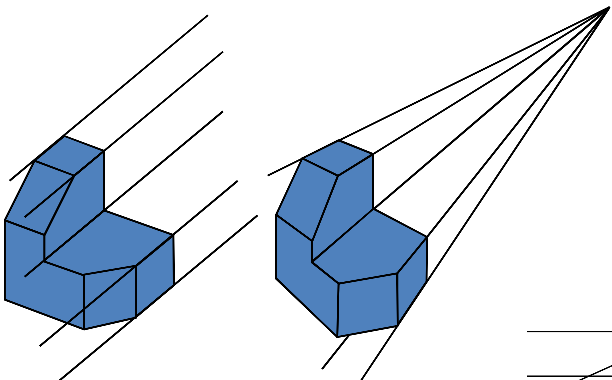
camera transformation

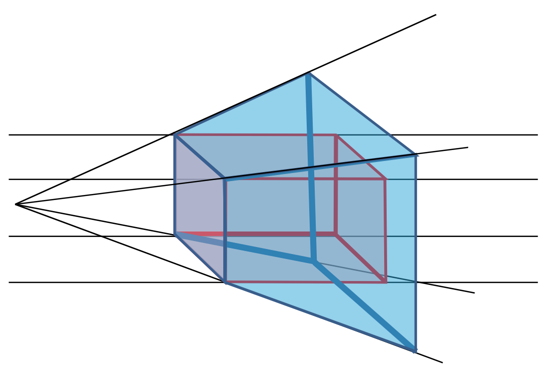
world coordinates

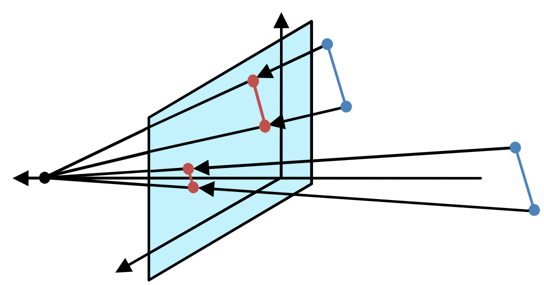
From Object Space to Screen Space



Perspective Projection

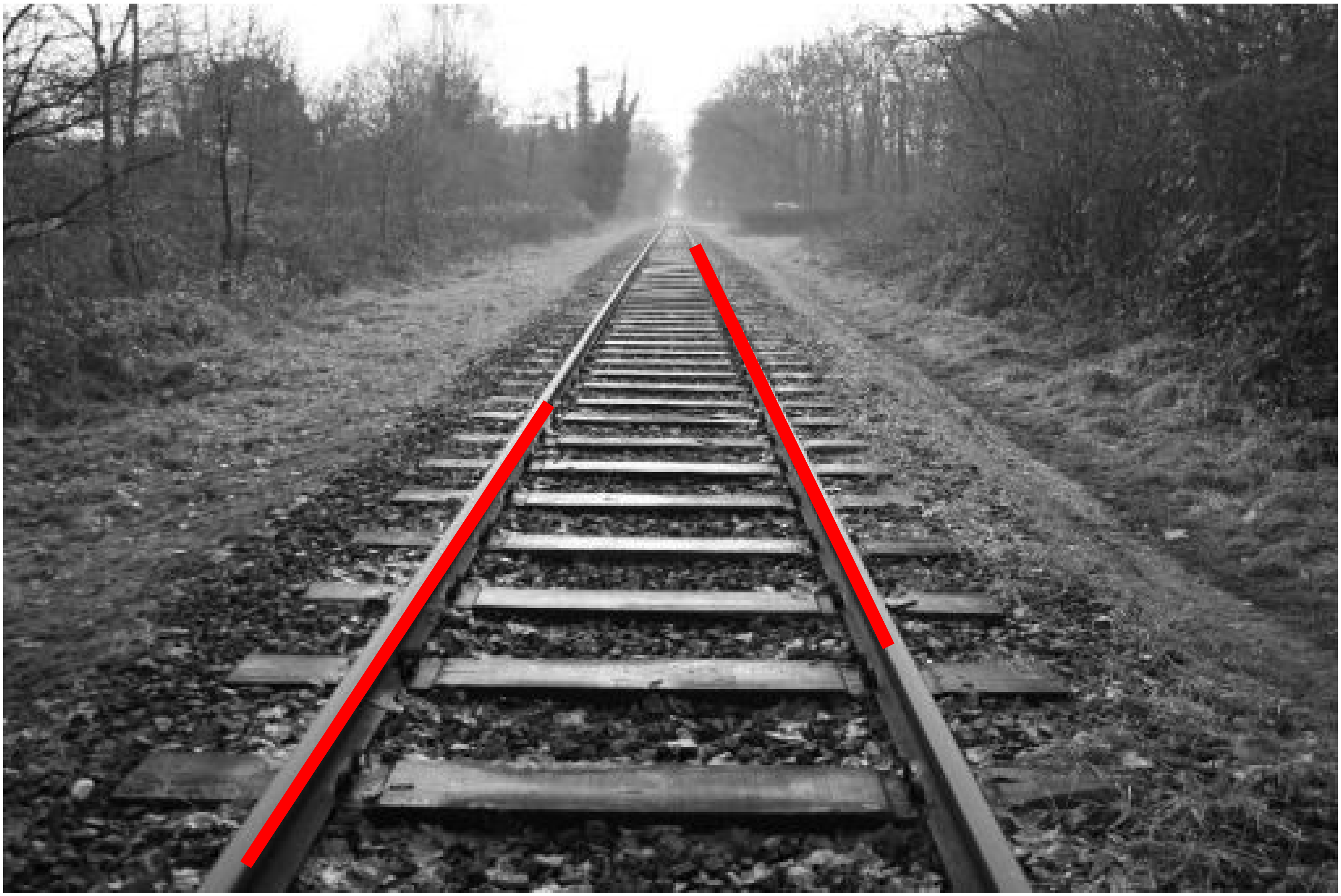




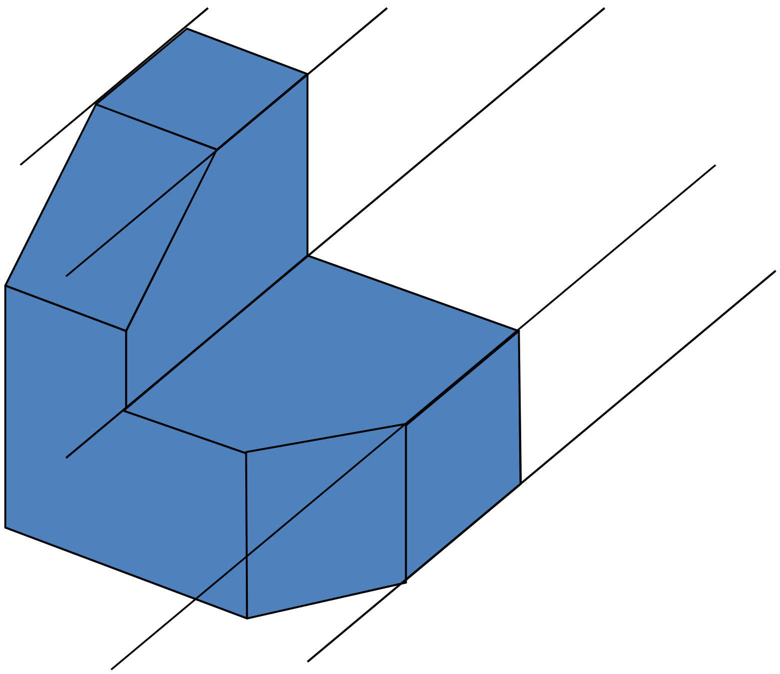


Perspective Projection

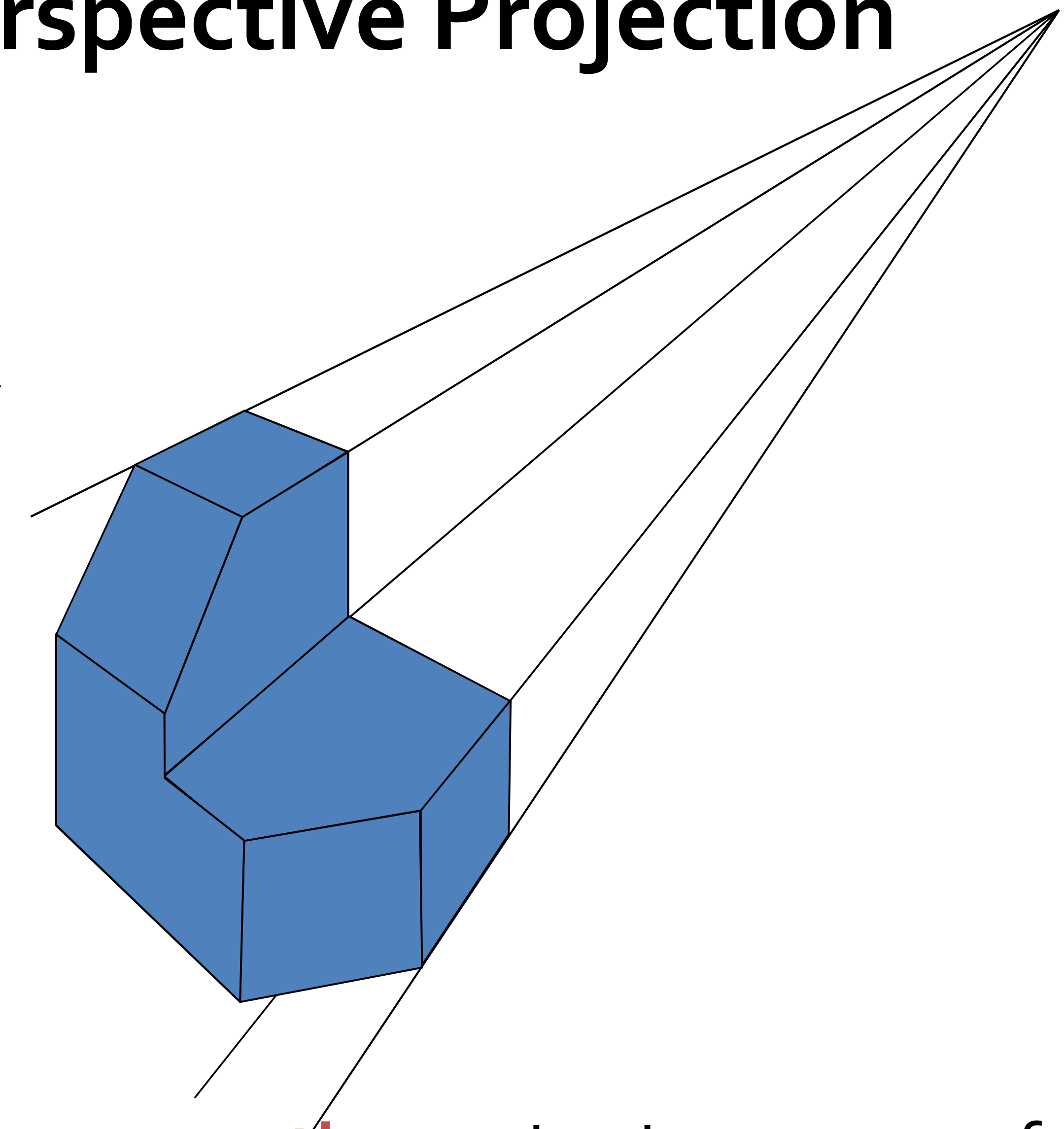
Perspective Projection



Parallel vs. Perspective Projection



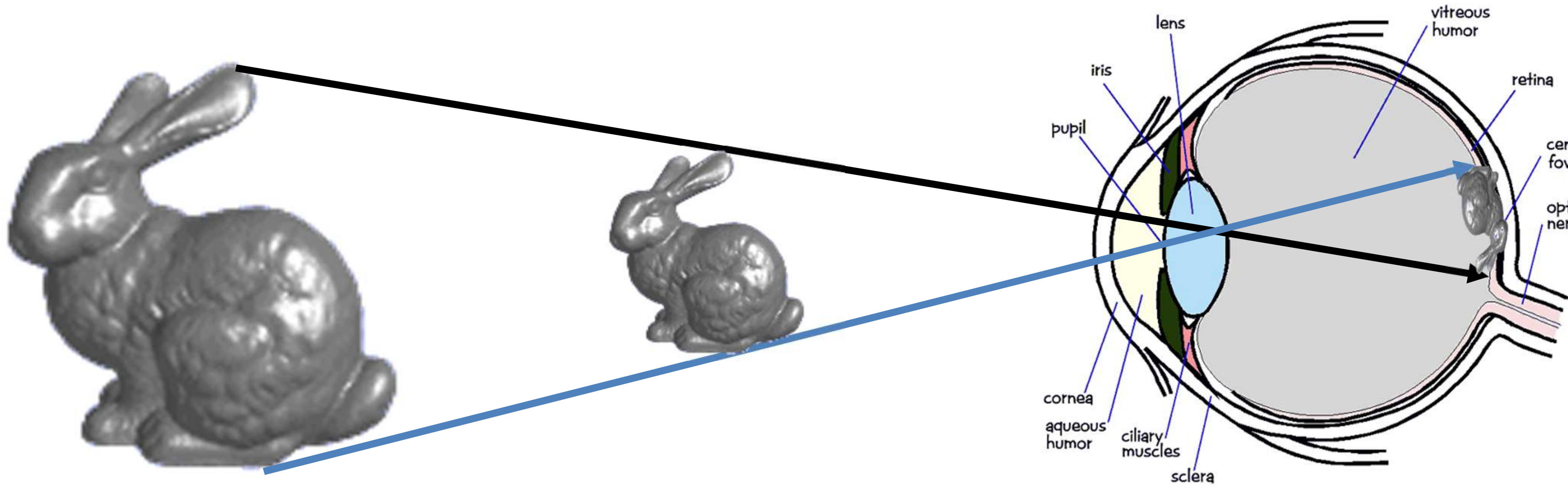
parallel projection:
preserves relative
proportions & parallel
features
(affine transform.)



perspective projection: center of
projection, realistic views

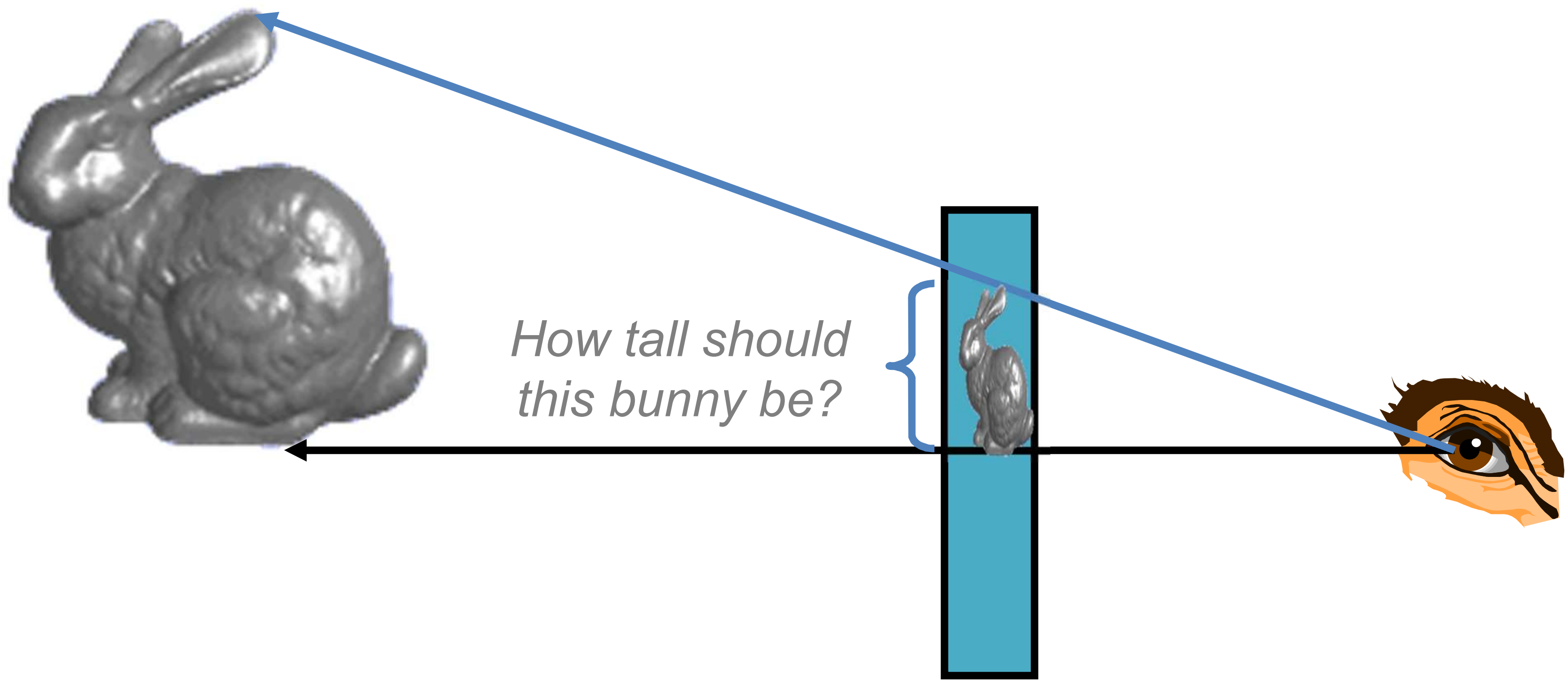
Perspective Projection

- In the real world, objects exhibit *perspective foreshortening*: distant objects appear smaller
- The basic situation:



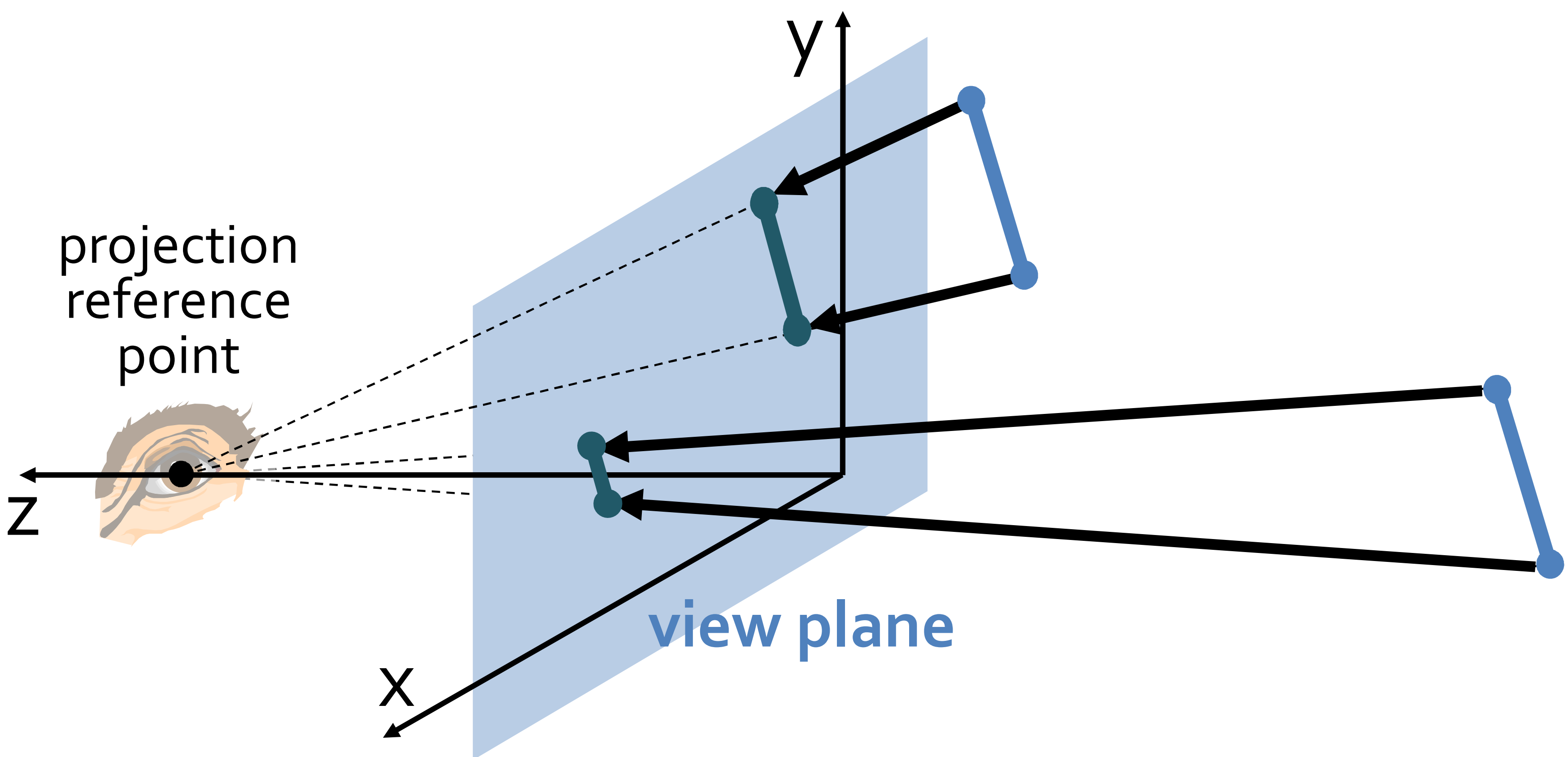
Perspective Projection

- When we do 3-D graphics, we think of the screen as a 2-D window onto the 3-D world:



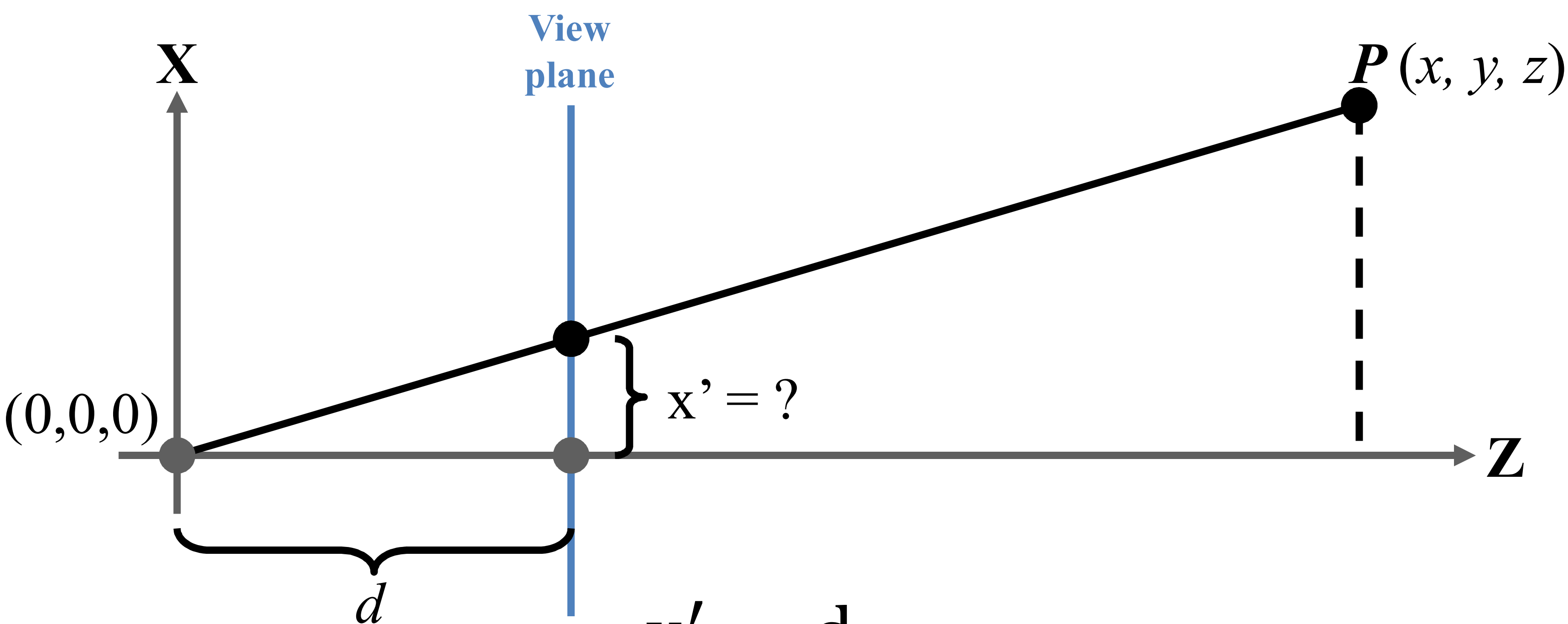
Perspective Projection

- Equal-sized objects at different distances from view plane



Perspective Projection

- The geometry of the situation is that of similar triangles. View from above:

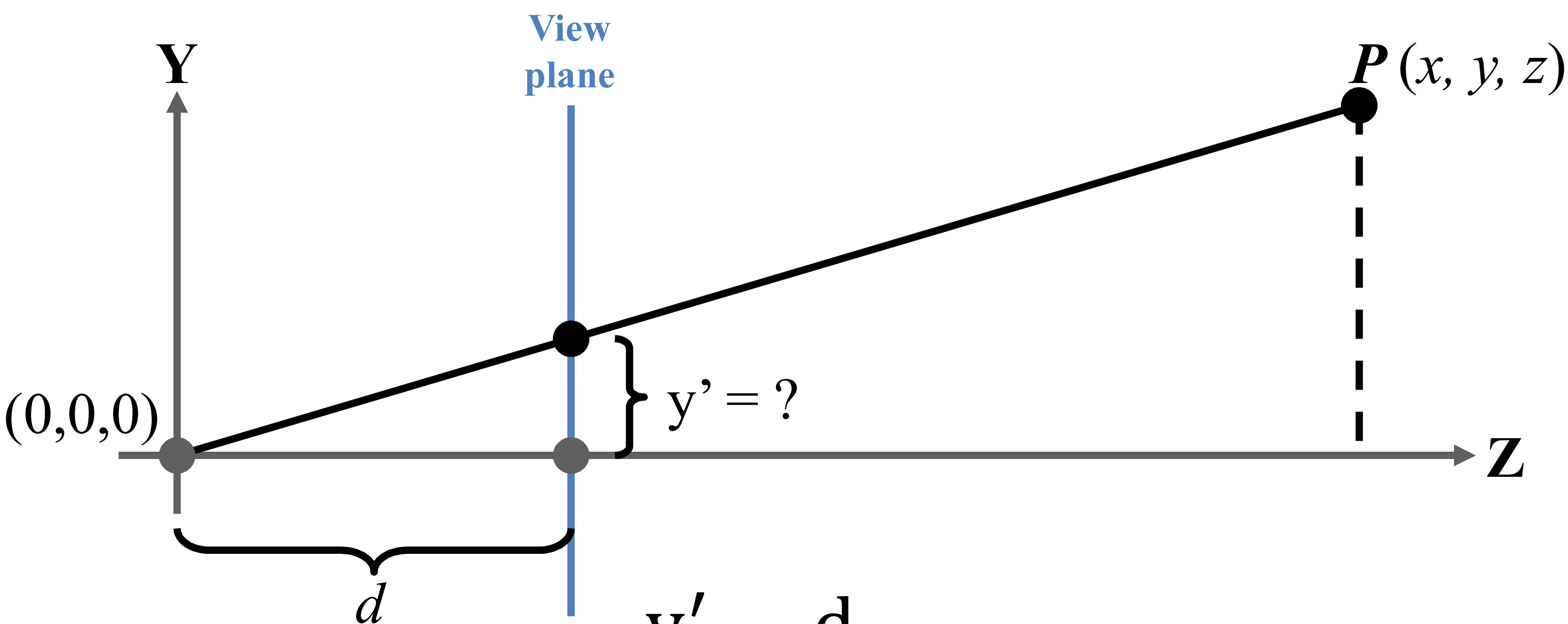


- What is x′?

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

Perspective Projection

- The geometry of the situation is that of similar triangles. View from side:



- What is y'?

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

Perspective Projection

- Desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

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

$$x' = \frac{d \cdot x}{z} = \frac{x}{z/d} \qquad y' = \frac{d \cdot y}{z} = \frac{y}{z/d} \qquad z' = d$$

- *What could a matrix look like to do this?*

A Perspective Projection Matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

A Perspective Projection Matrix

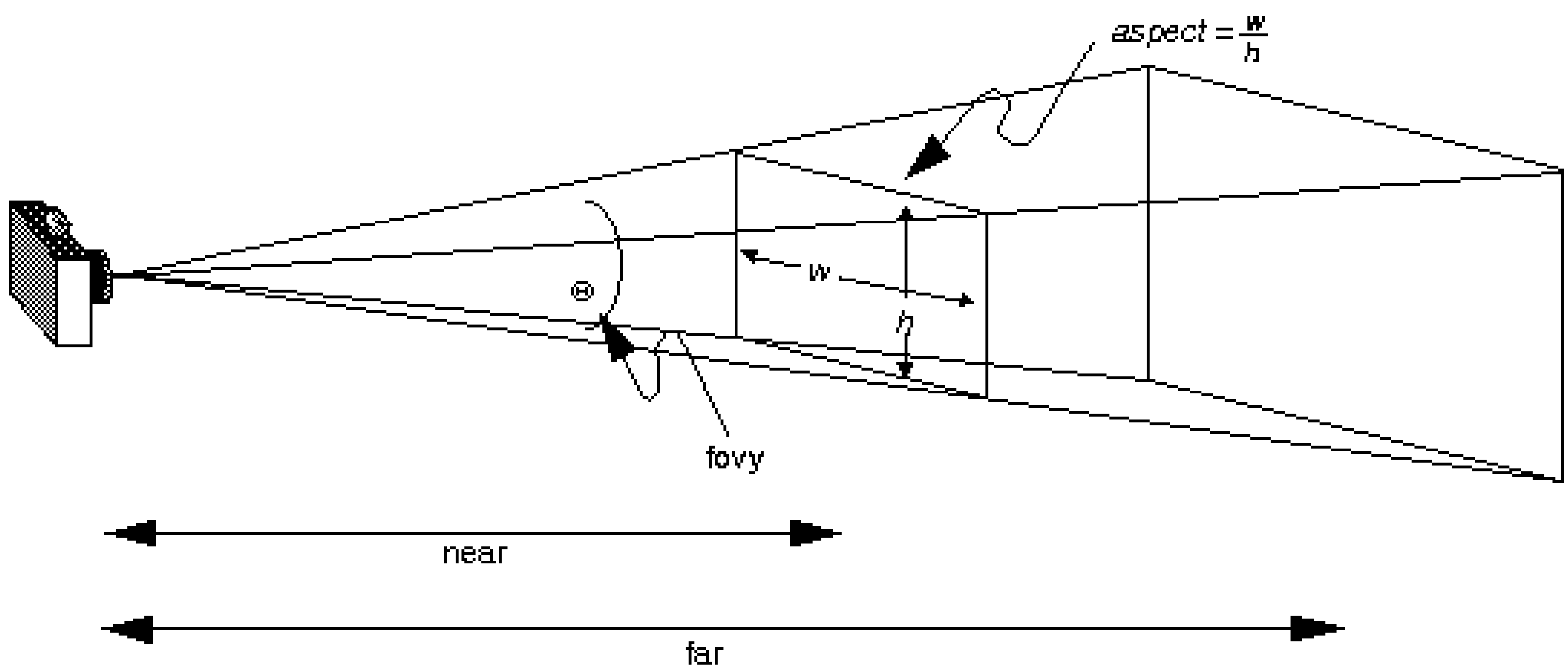
- Example:

$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Or, in 3-D coordinates:
- Problem with z?

$$\left(\frac{x}{z/d}, \quad \frac{y}{z/d}, \quad d \right)$$

Perspective Projections



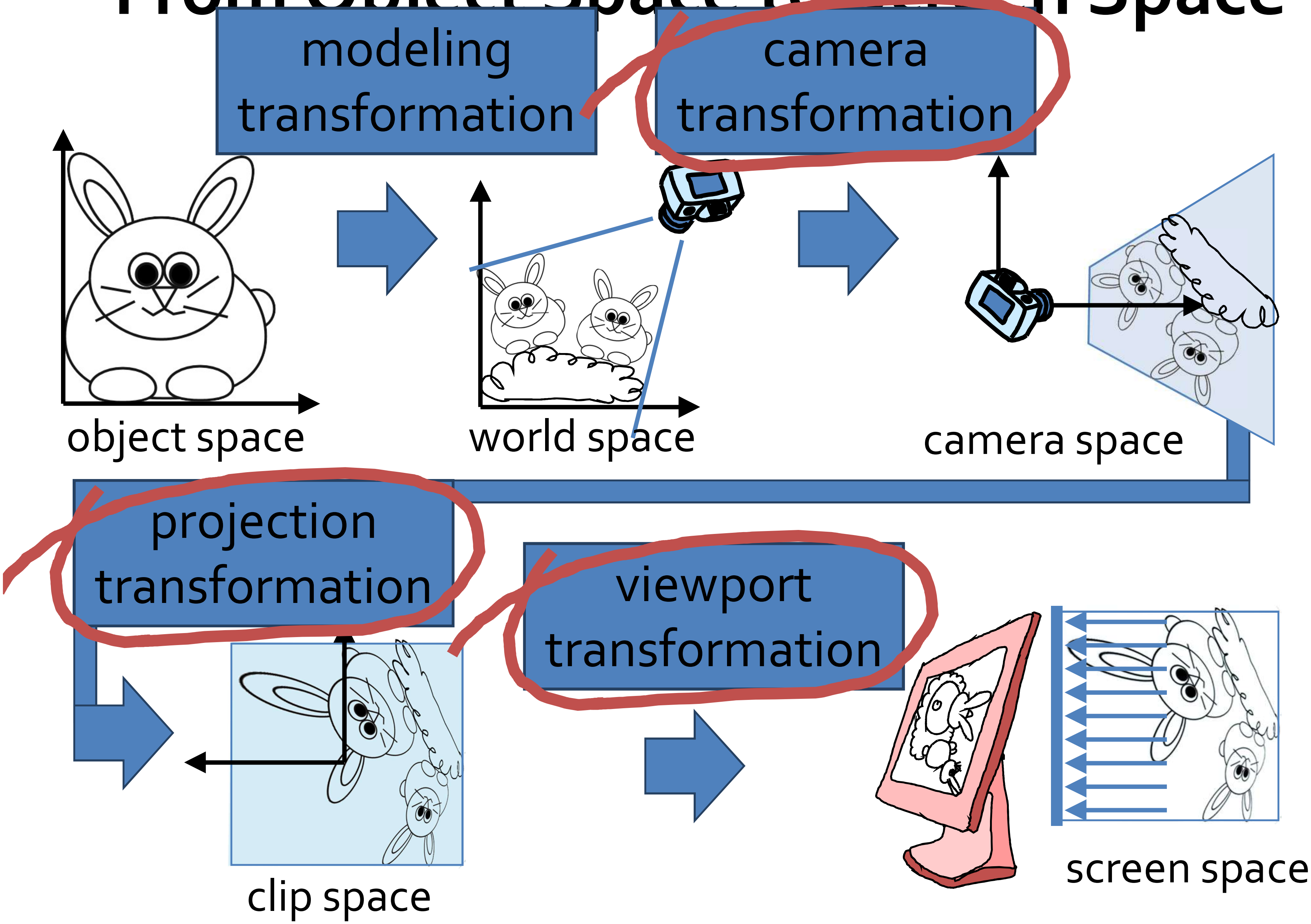
A Perspective Projection Matrix

- OpenGL's `gluPerspective()` command generates a slightly more complicated matrix:

$$\begin{bmatrix} \dfrac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \left(\dfrac{Z_{far}+Z_{near}}{Z_{near}-Z_{far}}\right) & \left(\dfrac{2\times Z_{far}\times Z_{near}}{Z_{near}-Z_{far}}\right) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\text{where} \quad f = \cot\left(\frac{fov_y}{2}\right)$$

From Object Space to Screen Space



Viewing: Camera + Projection + Viewport

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = \left(M_{vp} \cdot P \cdot M_{cam} \right) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

pixels on the screen

viewport transformation

perspective projection

camera transformation

world coordinates