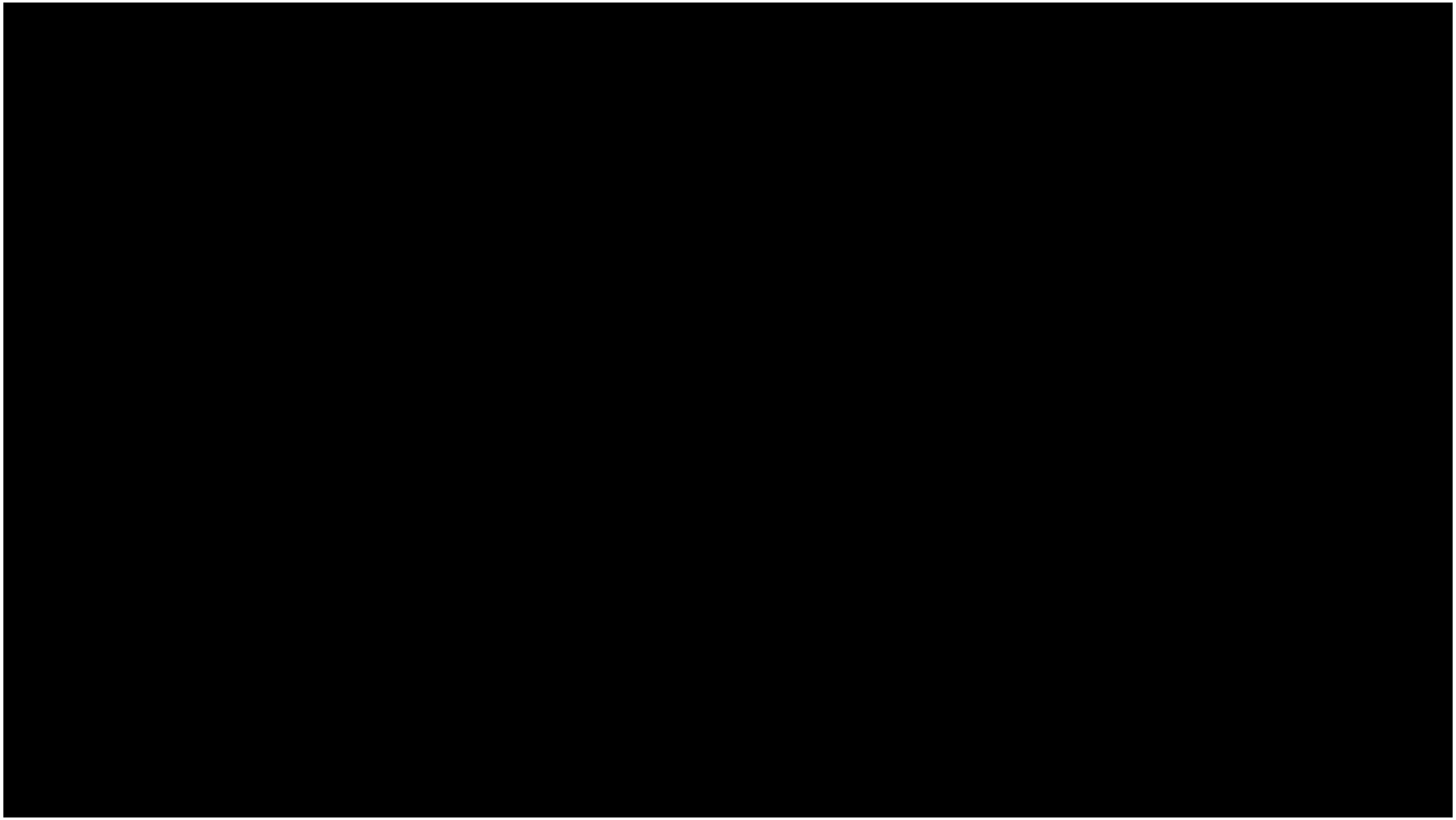
Topic 6:

3D Transformations

- Homogeneous 3D transformations
- Scene Hierarchies
- Change of basis and rotations in 3D

Showtime:



Logitics

- Assignment 1 Due Tomorrow
- Assignment 2 available today/tomorrow
- For assignment questions use the bulletin board or email:
  - csc418tas@cs.toronto.edu
- “When will your slides be online ?”
  - Today ☺

Representing 2D transforms as a 3x3 matrix

Translate a point $[x\ y]^T$ by $[t_x\ t_y]^T$:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Rotate a point $[x\ y]^T$ by an angle t :

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos t & -\sin t & 0 \\ \sin t & \cos t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Scale a point $[x\ y]^T$ by a factor $[s_x\ s_y]^T$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Representing 3D transforms as a 4x4 matrix

Translate a point $[x\ y\ z]^T$ by $[t_x\ t_y\ t_z]^T$:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotate a point $[x\ y\ z]^T$ by an angle t around z axis:

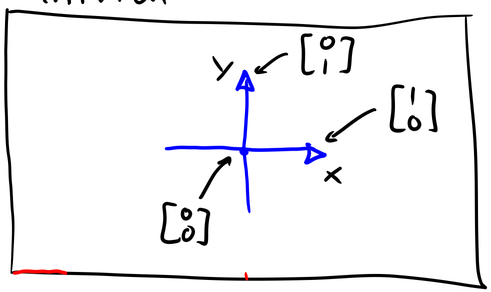
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{pmatrix} \cos t & -\sin t & 0 & 0 \\ \sin t & \cos t & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scale a point $[x\ y\ z]^T$ by a factor $[s_x\ s_y\ s_z]^T$

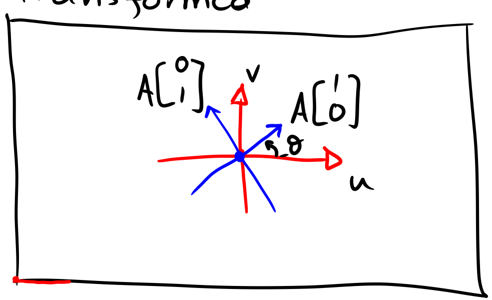
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Elementary Rotations in 3D

Initial



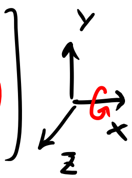
Transformed



$$\left[\begin{array}{c|c} A_{2\times 2} & \begin{matrix}0\\0\end{matrix} \\ \hline 0\ 0 & 1\end{array}\right] \qquad A = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$

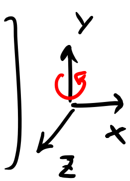
Rotation by θ about x axis

$$H_x^\theta = \left[\begin{array}{c|c} A_x & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\,0\,0 & 1\end{array}\right] \qquad A_x = \begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & \cos\theta\end{bmatrix}$$



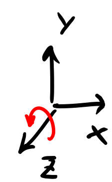
Rotation by θ about y axis:

$$H_y^\theta = \left[\begin{array}{c|c} A_y & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\,0\,0 & 1\end{array}\right] \qquad A_y = \begin{bmatrix}\cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta\end{bmatrix}$$



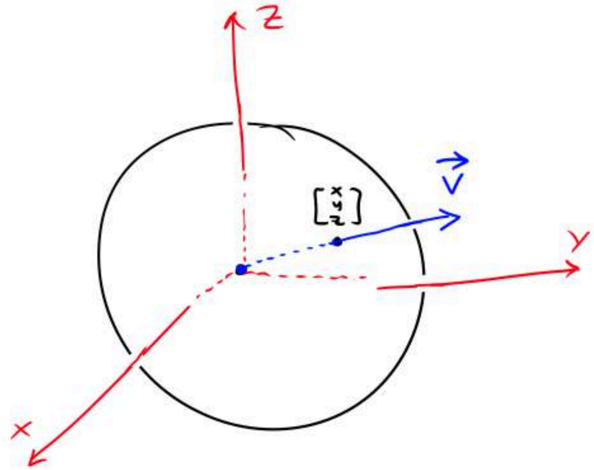
Rotation by θ about z axis:

$$H_z^\theta = \left[\begin{array}{c|c} A_z & \begin{matrix}0\\0\\0\end{matrix} \\ \hline 0\,0\,0 & 1\end{array}\right] \qquad A_z = \begin{bmatrix}\cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix}$$



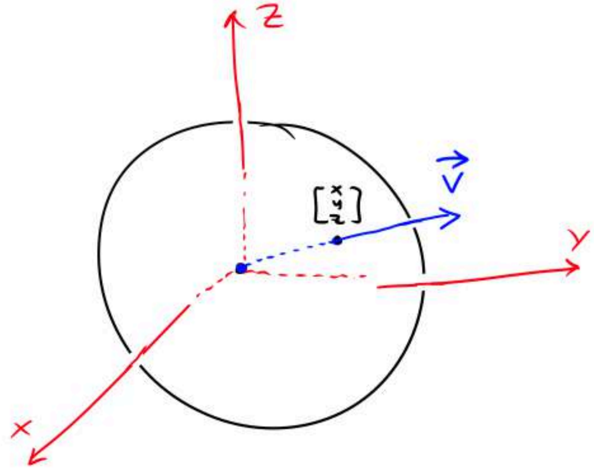
Rotation About Arbitrary Vector?

Question: How do we define A when it is a rotation about an arbitrary vector $\vec{v}$?



Rotation About Arbitrary Vector?

Question: How do we define A when it is a rotation about an arbitrary vector $\vec{v}$?



Ans: Express it as a composition of the three elementary matrices A_x, A_y, A_z

Rotation by ϑ about x axis

$$H_x^{\vartheta} = \left[\begin{array}{c|c} A_x & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 000 & 1 \end{array}\right] \quad A_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\vartheta & -\sin\vartheta \\ 0 & \sin\vartheta & \cos\vartheta \end{bmatrix}$$

Rotation by ϑ about y axis:

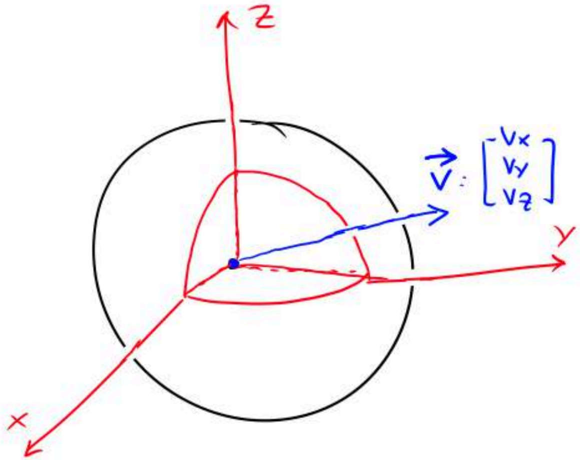
$$H_y^{\vartheta} = \left[\begin{array}{c|c} A_y & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 000 & 1 \end{array}\right] \quad A_y = \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\vartheta & 0 & \cos\vartheta \end{bmatrix}$$

Rotation by ϑ about z axis:

$$H_z^{\vartheta} = \left[\begin{array}{c|c} A_z & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 000 & 1 \end{array}\right] \quad A_z = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation About Arbitrary Vector: Construction

Question: How do we define A when it is a rotation of ϕ about an arbitrary vector $\vec{v}$?

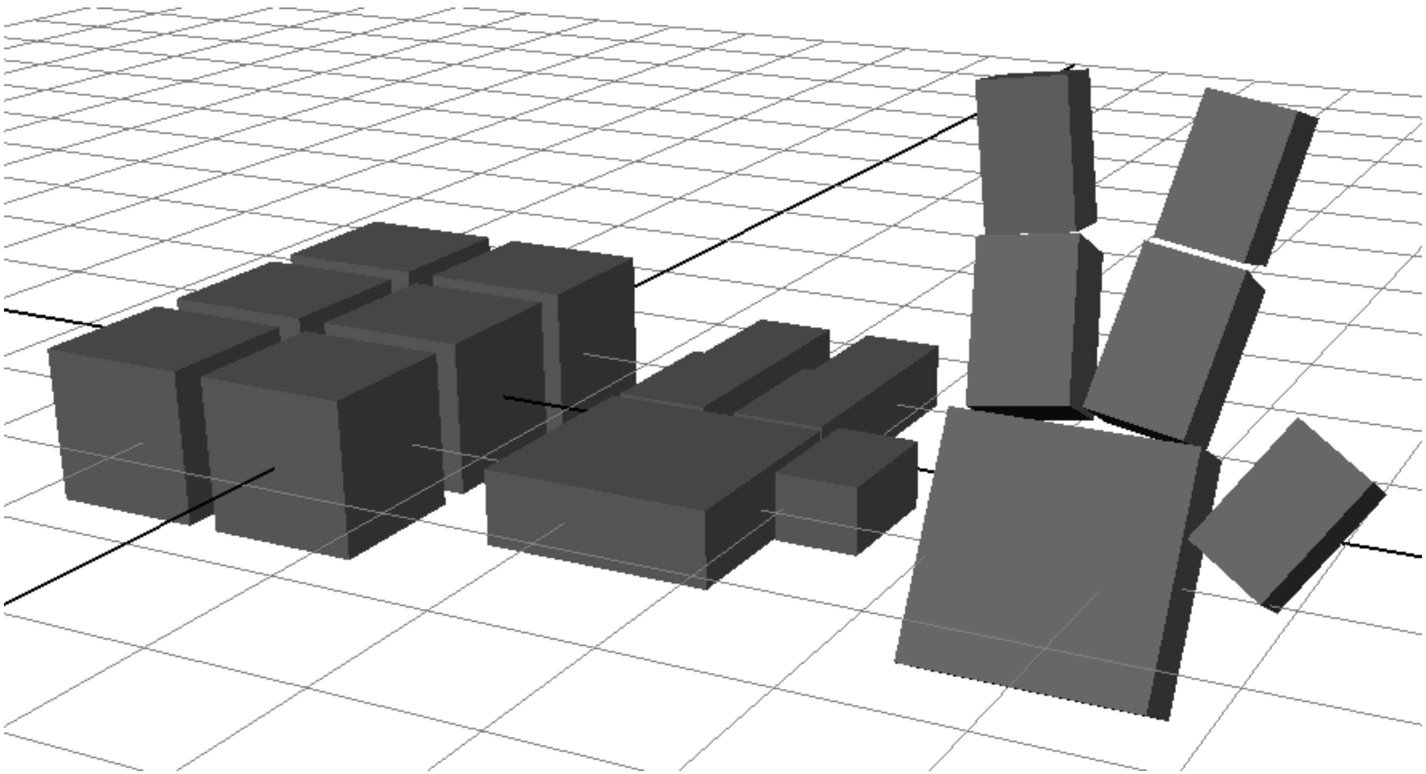


Ans: Express it as a composition of the three elementary matrices A_x, A_y, A_z

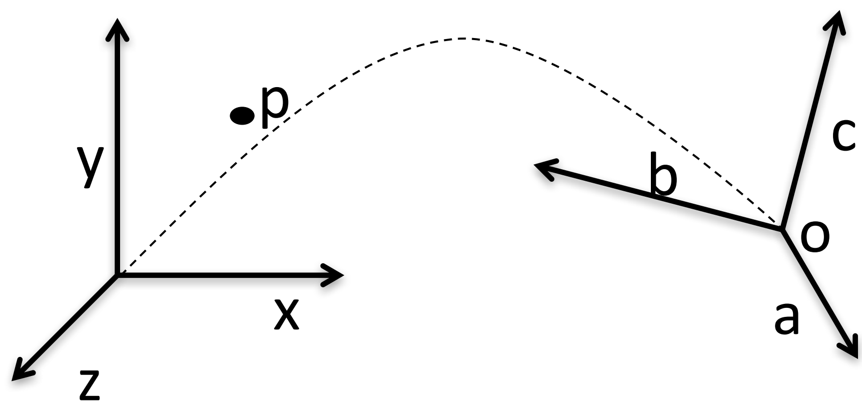
Basic idea: Since we know how to do rotations about z, we will do the following:

* Align $\vec{v}$ with the $\vec{z}$ axis "temporarily" (using axis-aligned rotations)
* Rotate about $\vec{v}$ using A_z
* Undo the temporary alignment

Scene Hierarchies



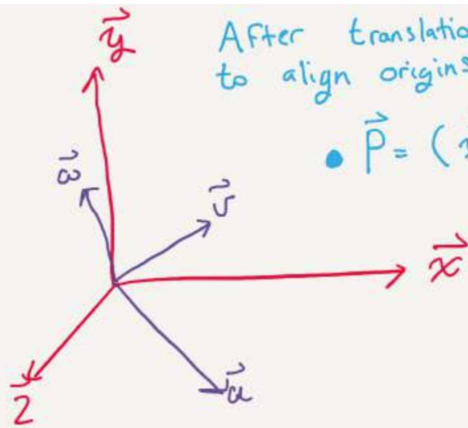
Change of reference frame/basis matrix



$$p = a p_x' + b p_y' + c p_z' + o$$

$$p = \begin{pmatrix} a & b & c & o \\ 0 & 0 & 0 & 1 \end{pmatrix} p'$$

$$p' = \begin{pmatrix} a & b & c & o \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} p$$



After translation to align origins

- $\vec{P} = (\underbrace{x, y, z}_{\text{scalars}})$

A) In Euclidean Basis

$$\vec{P} = x \underbrace{\begin{bmatrix}1\\0\\0\end{bmatrix}}_{\vec{e}_1} + y \underbrace{\begin{bmatrix}0\\1\\0\end{bmatrix}}_{\vec{e}_2} + z \underbrace{\begin{bmatrix}0\\0\\1\end{bmatrix}}_{\vec{e}_3}$$

But in $\vec{u}\vec{v}\vec{w}$ space

$$\vec{P} = a\vec{\omega} + b\vec{u} + c\vec{v}$$

This Defines a System of Equations

B) $a\vec{\omega} + b\vec{u} + c\vec{v} = x\vec{e}_1 + y\vec{e}_2 + z\vec{e}_3$

Exploit orthogonality:

$a = x\vec{\omega}^T\vec{e}_1 + y\,\vec{\omega}^T\vec{e}_2 + z\vec{\omega}^T\vec{e}_3$

$b = \ldots$

$c = \ldots$

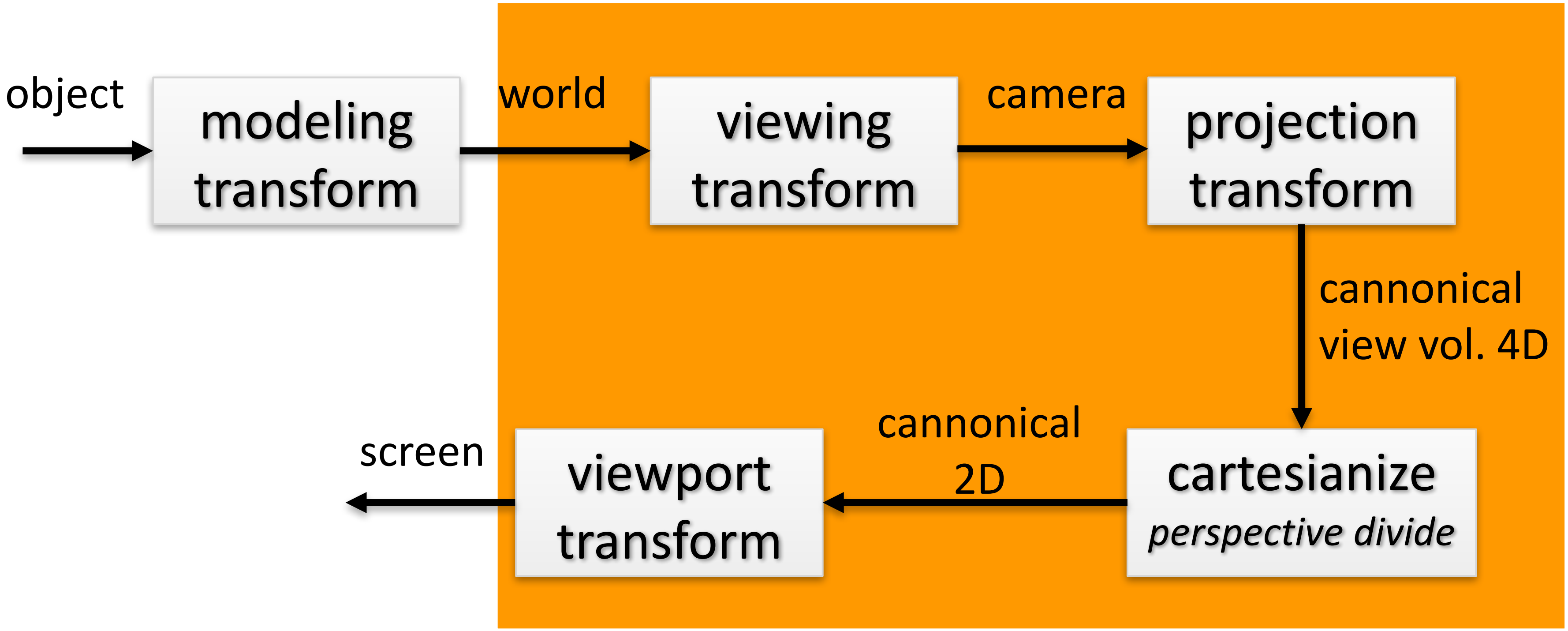
$\vec{P}_{in}$ $\vec{e}_1, \vec{e}_2, \vec{e}_3$

$$\Rightarrow \underbrace{\begin{bmatrix}a\\b\\c\end{bmatrix}}_{\vec{P}_{in}\ \vec{\omega},\vec{u},\vec{v}} = \begin{bmatrix}\vec{\omega}^T\vec{e}_1 & \vec{\omega}^T\vec{e}_2 & \vec{\omega}^T\vec{e}_3\\ \vec{u}^T\vec{e}_1 & \vec{u}^T\vec{e}_2 & \vec{u}^T\vec{e}_3\\ \vec{v}^T\vec{e}_1 & \vec{v}^T\vec{e}_2 & \vec{v}^T\vec{e}_3\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}$$

$\rightarrow \cos(\theta^{\vec{v}}_{\vec{e}_3})$

$\boxed{\text{Transformation Between Frames}}$

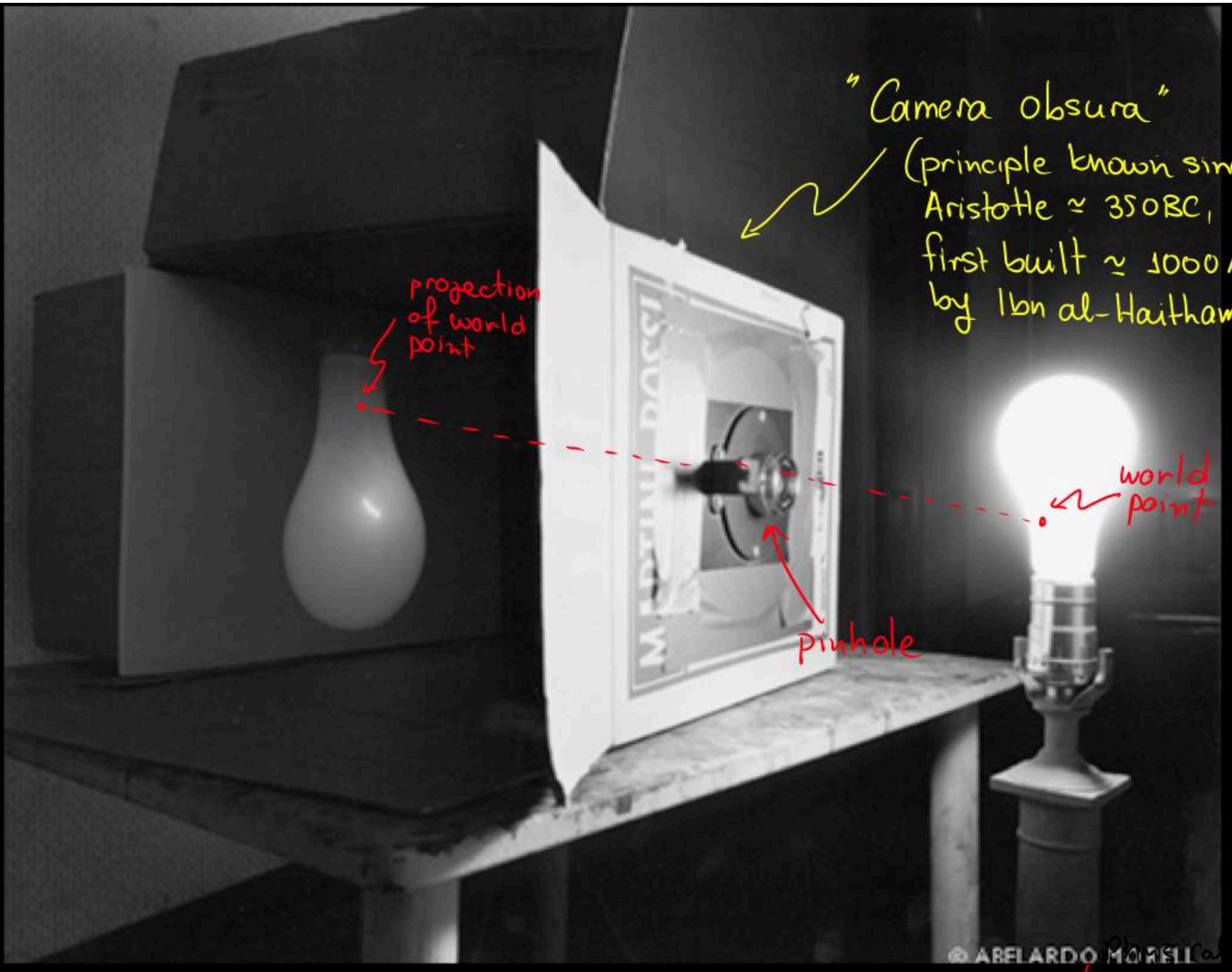
Viewing Pipeline



Topic 7:

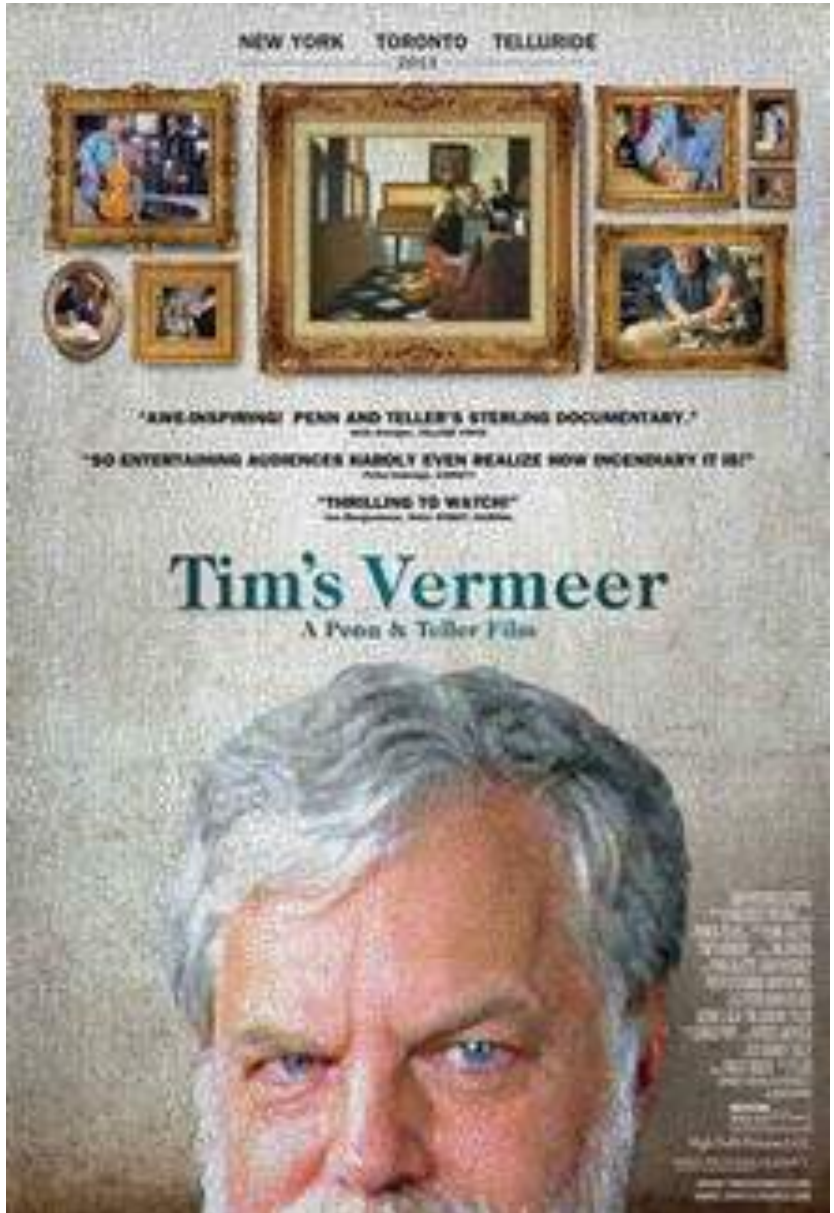
3D Viewing

- Camera Model
- Orthographic projection
- The world-to-camera transformation
- Perspective projection
- The transformation chain for 3D viewing

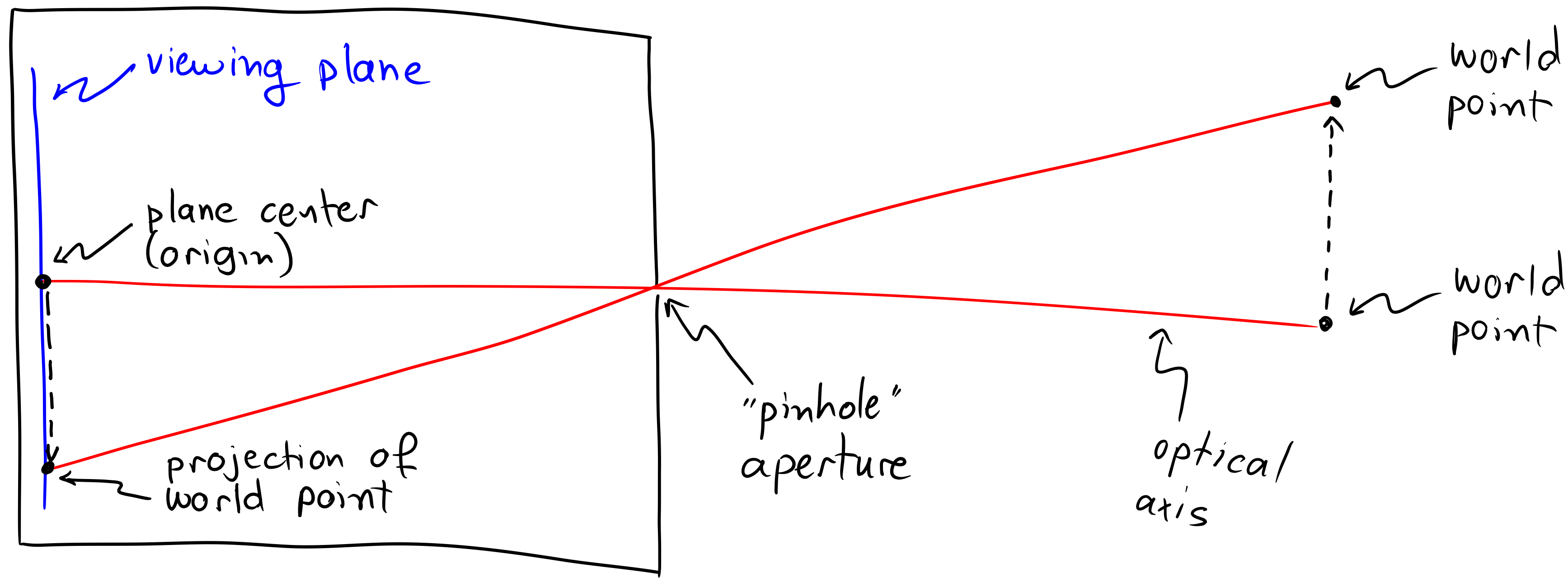


Light Bulb, 1991

www.lensculture.com/morell.html

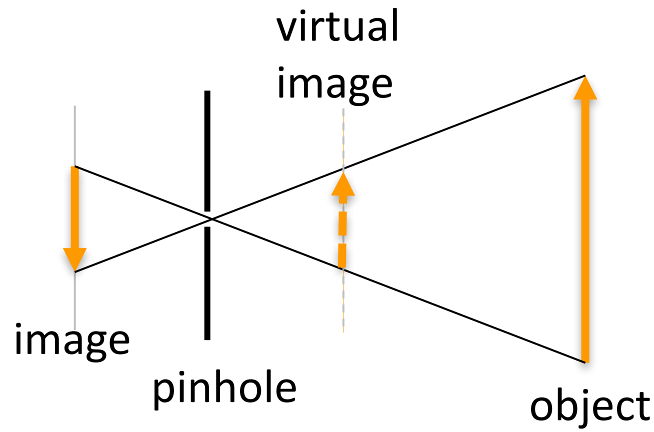

NEW YORK TORONTO TELLURIDE
"AWE-INSPIRING! PENN AND TELLER'S STERLING DOCUMENTARY."
"SO ENTERTAINING AUDIENCES HARDLY EVEN REALIZE HOW INCENDIARY IT IS!"
"THRILLING TO WATCH!"
Tim's Vermeer
A Penn & Teller Film

The Pinhole Camera

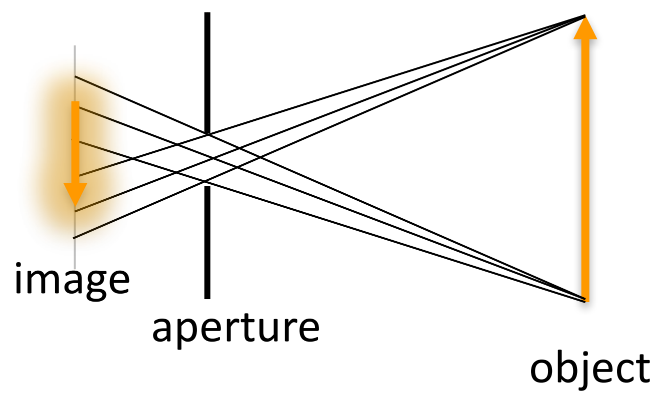


Camera model

Ideal pinhole camera

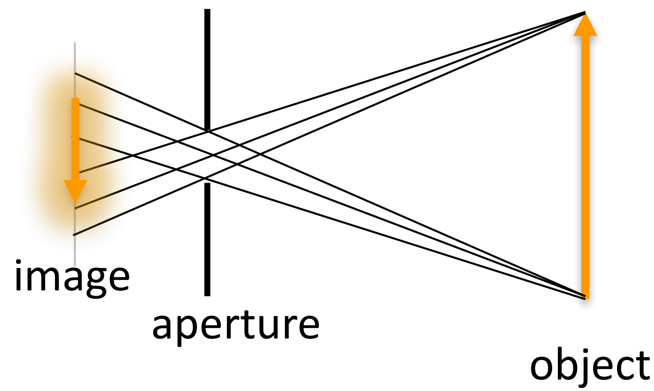


Real pinhole camera

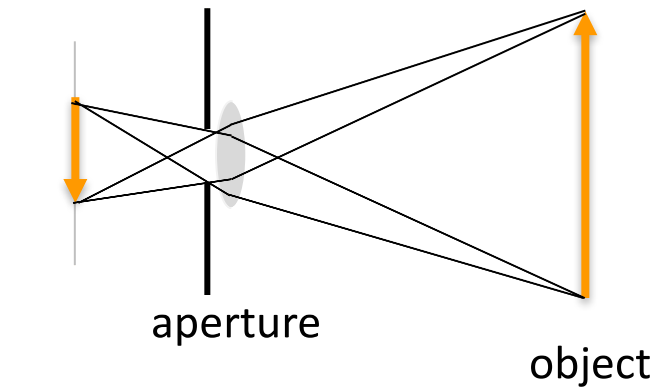


Camera model

Real pinhole camera

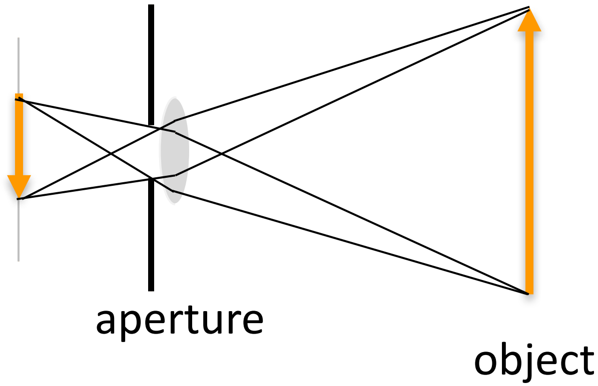


Camera with a lens

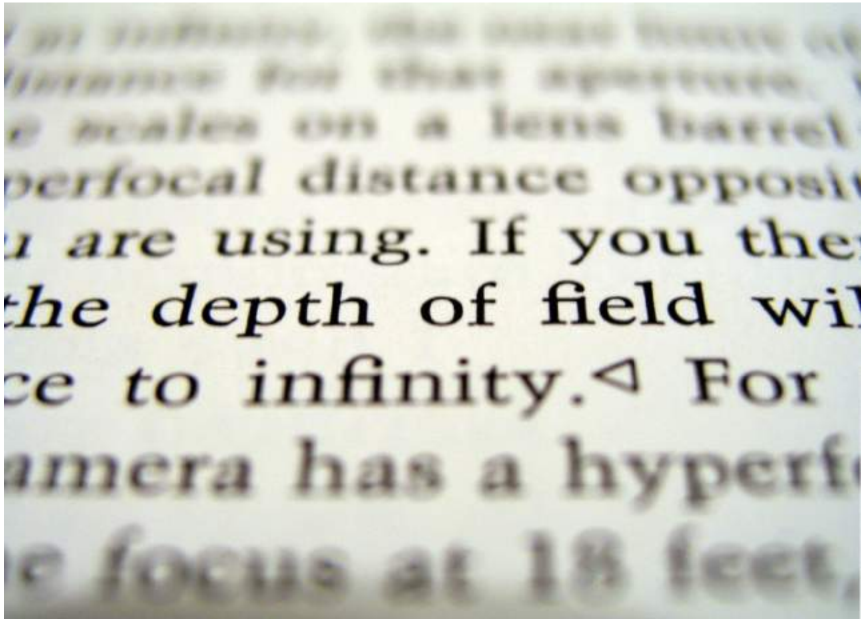


Camera model

Camera with a lens

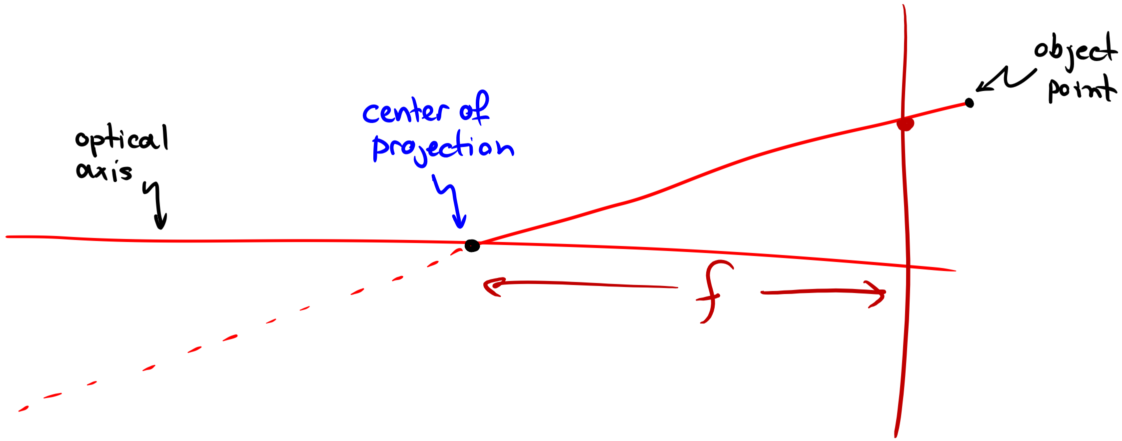


Depth of Field



The Pinhole Camera: Basic Geometry in 2D

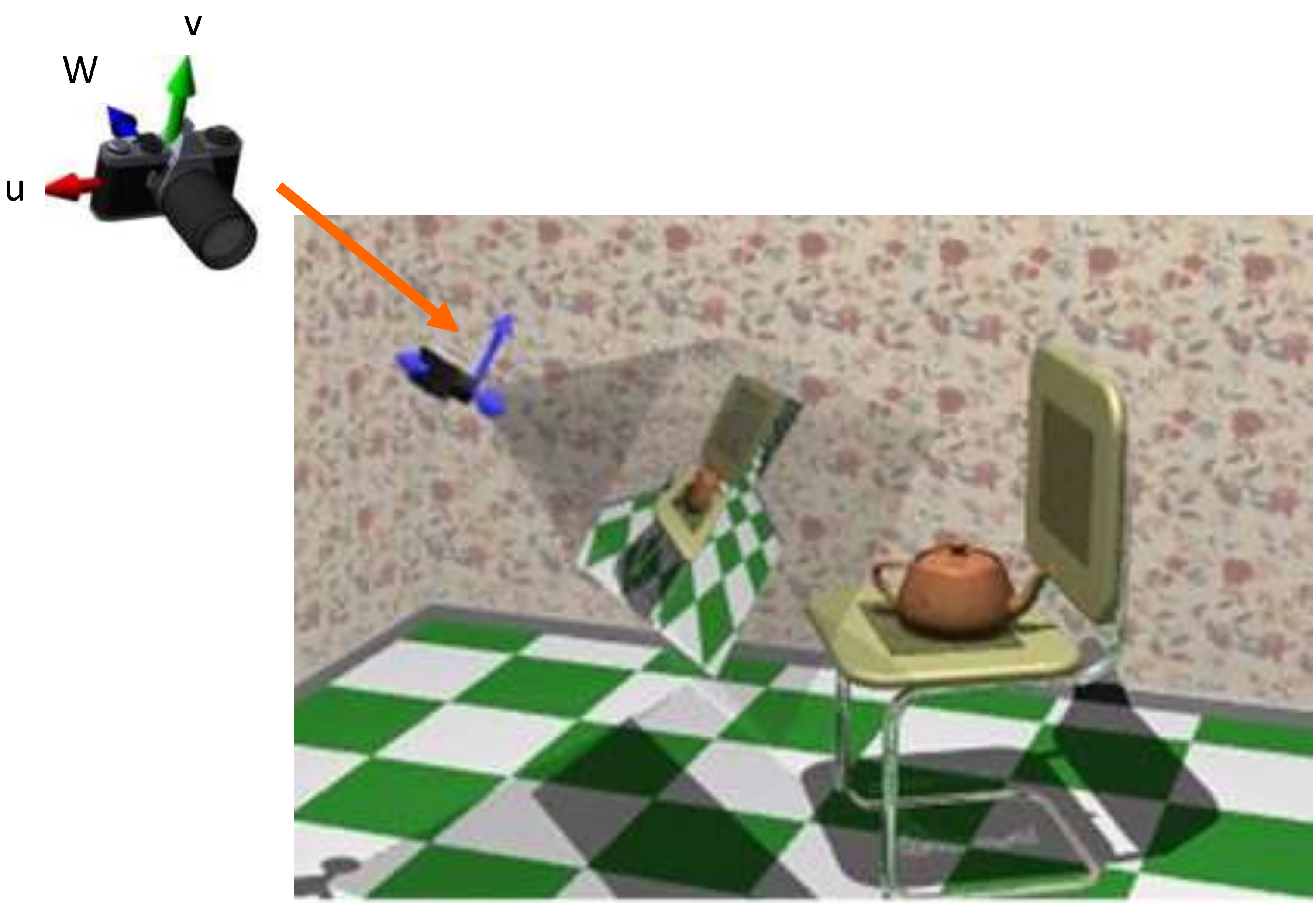
We will only consider the idealized pinhole model here (a.k.a. perspective projection)



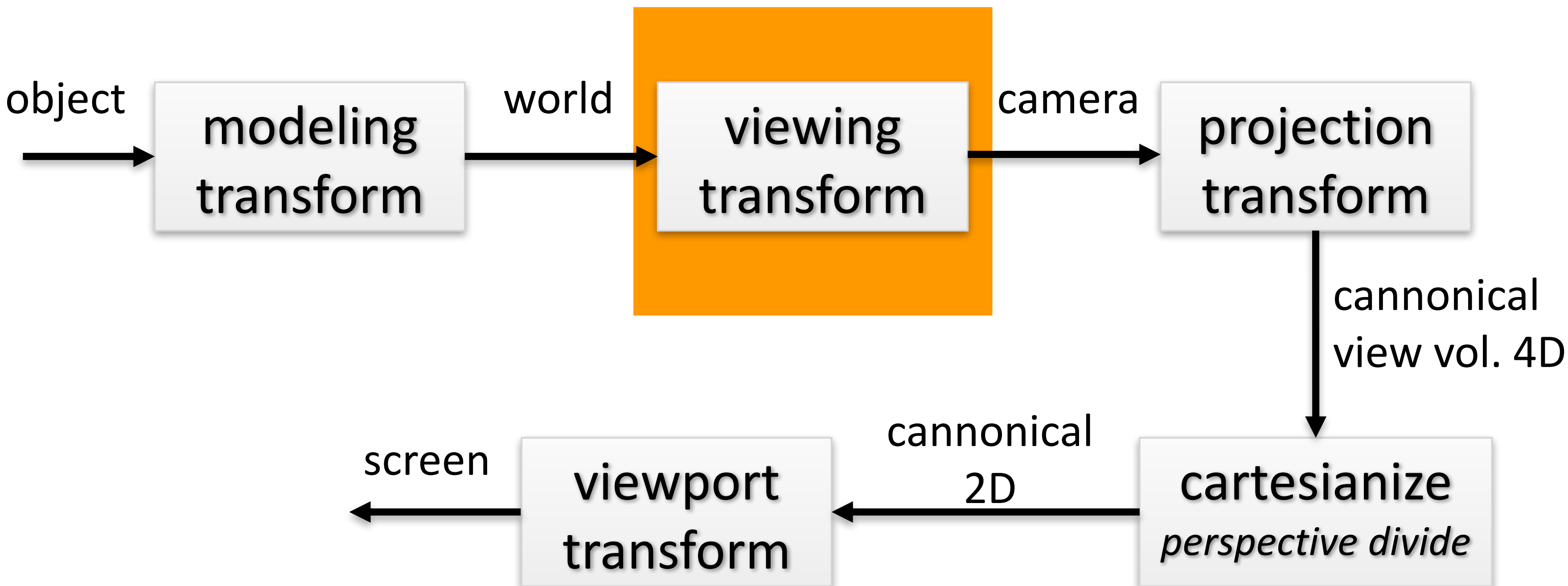
Simplification #1: Take plane-to-pinhole distance = f

Simplification #2: "Undo" image reversal by placing viewing plane in front of pinhole

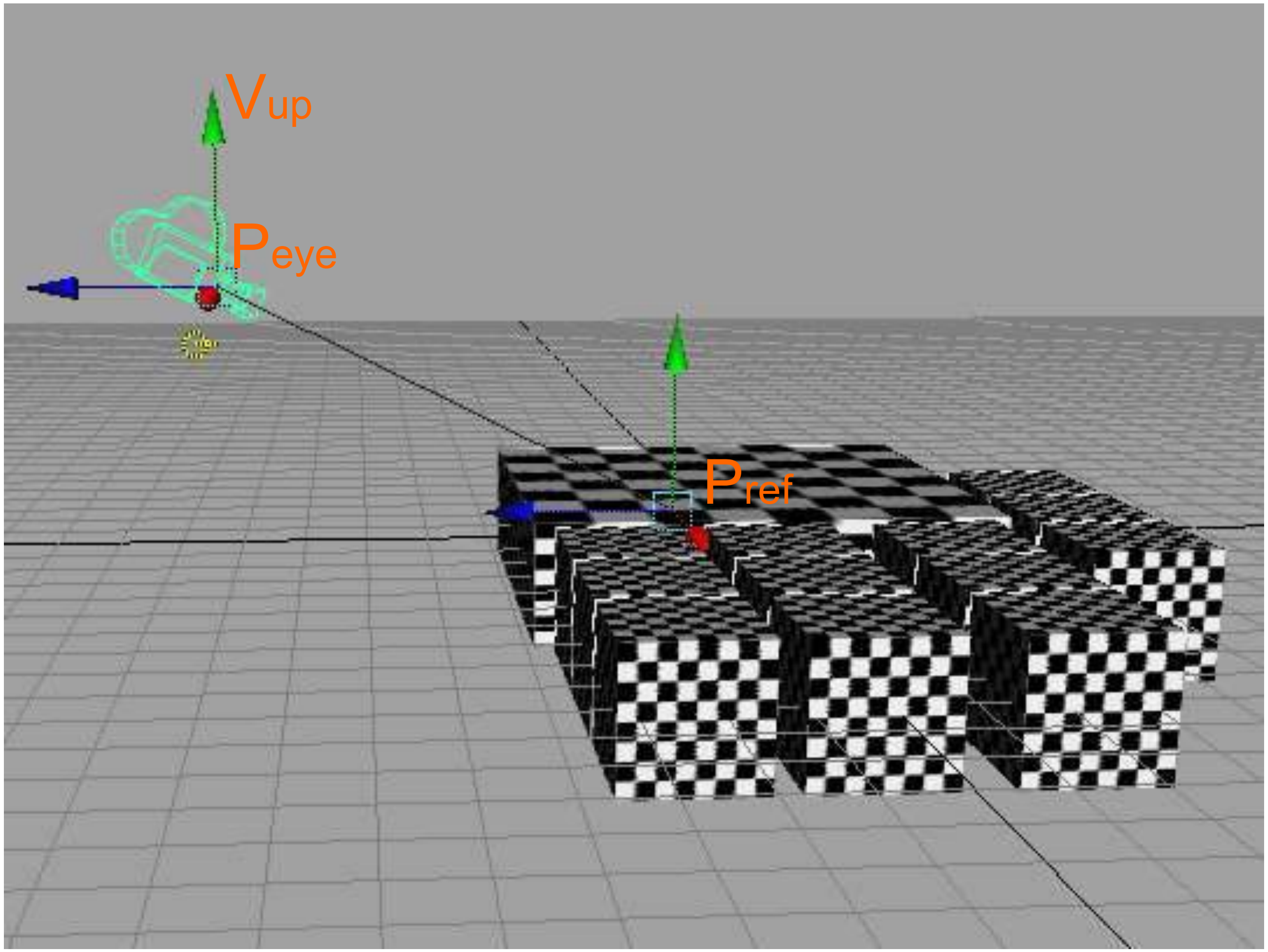
Camera model



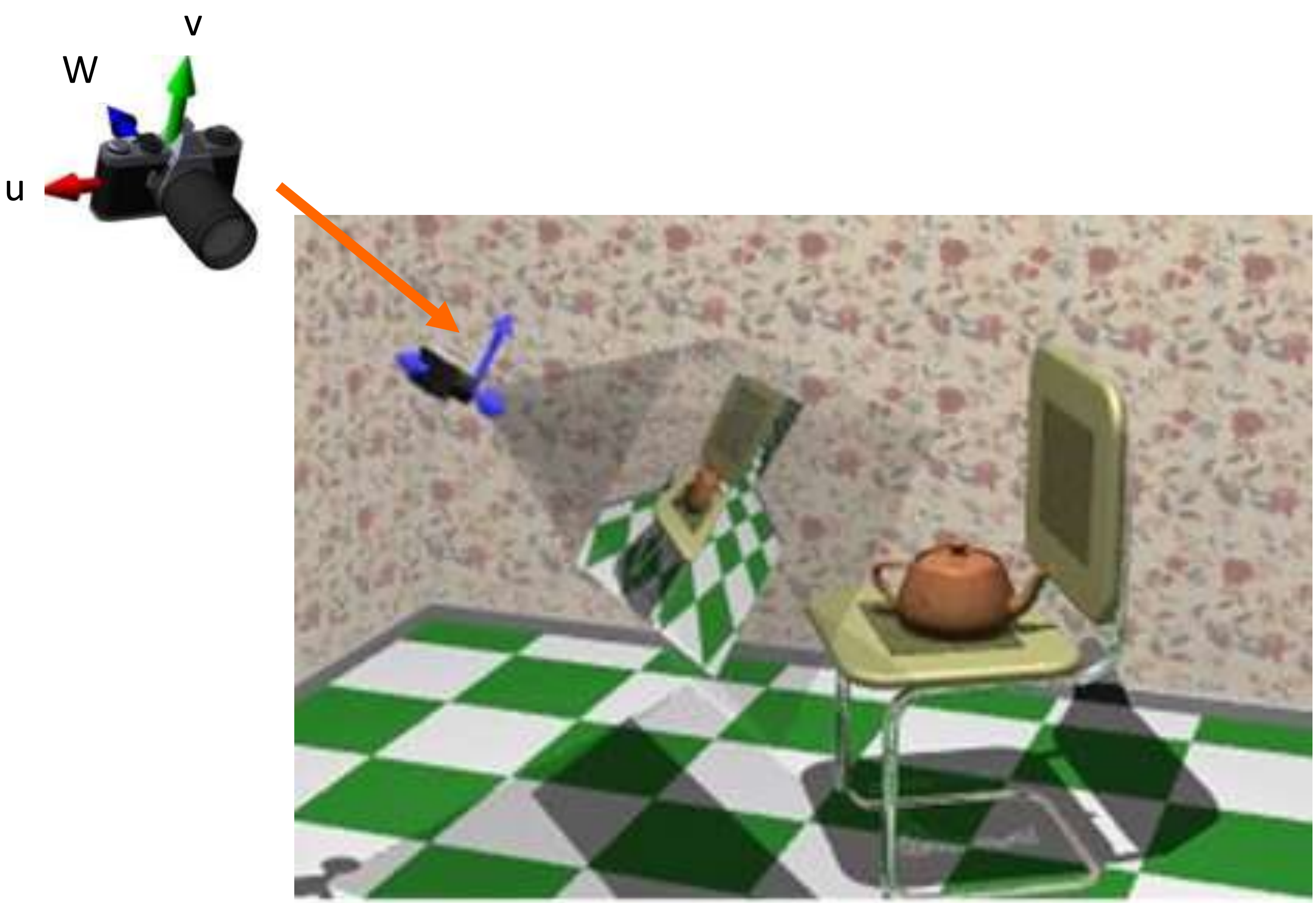
Viewing Pipeline



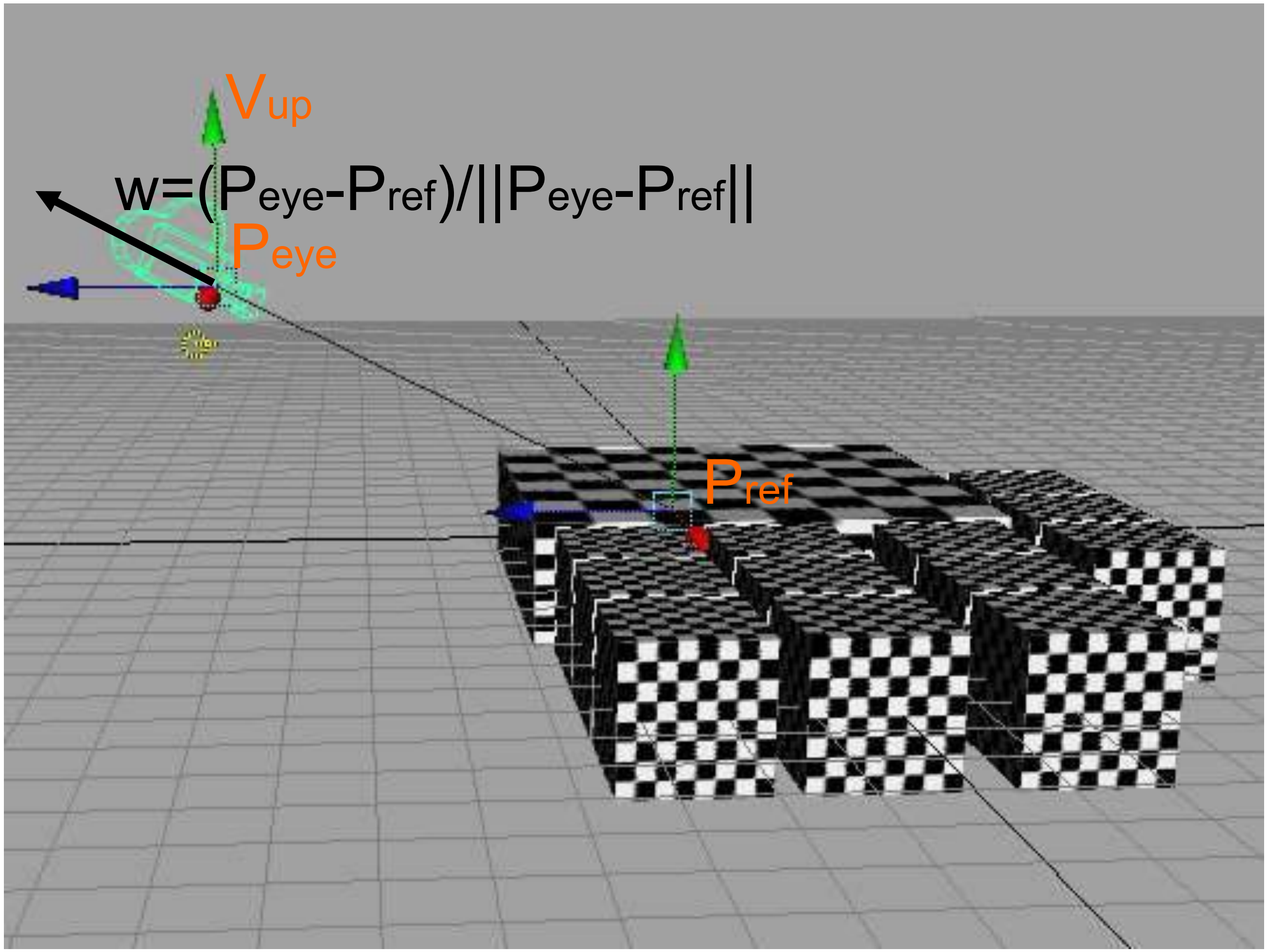
Viewing Transform



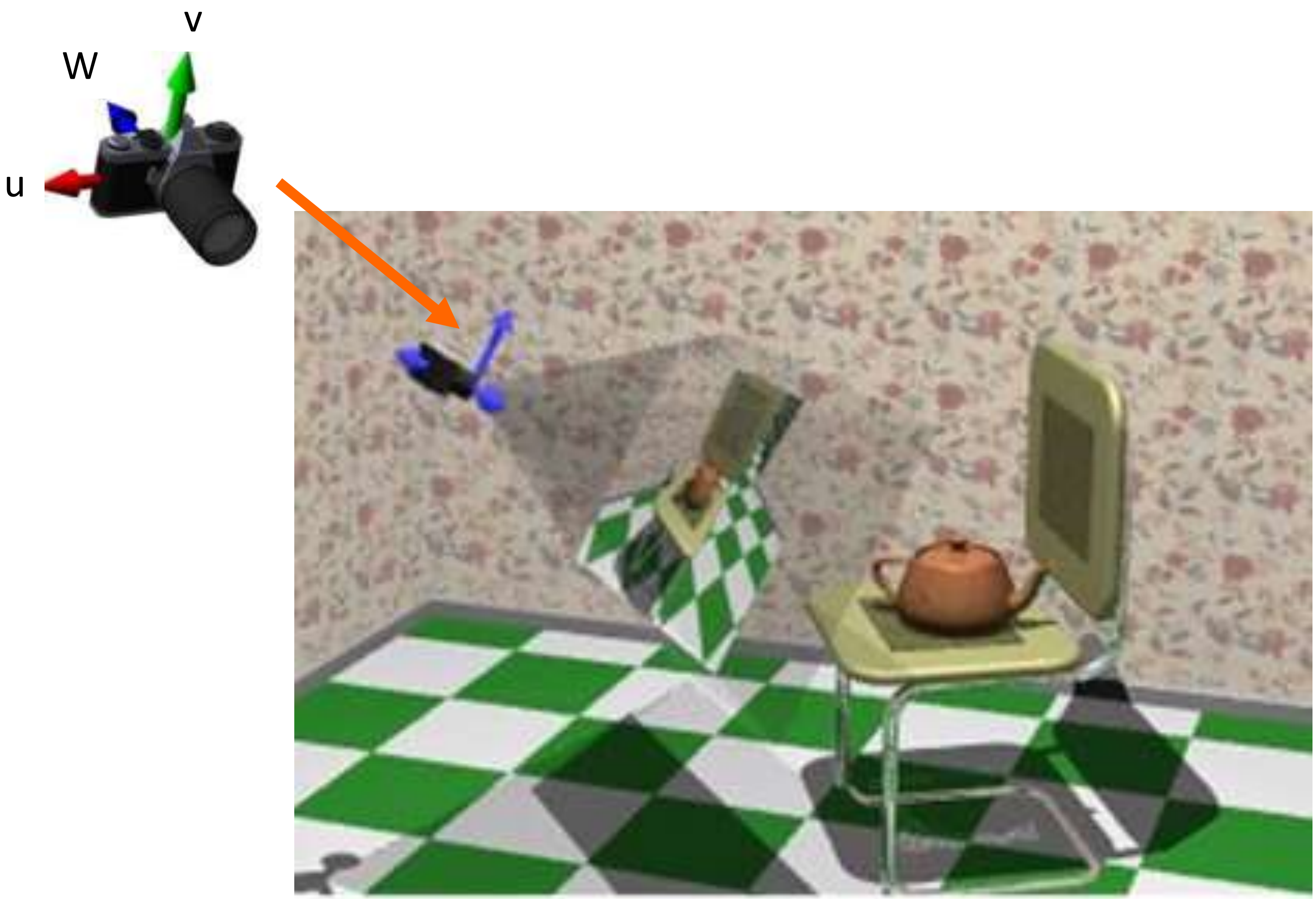
Camera model



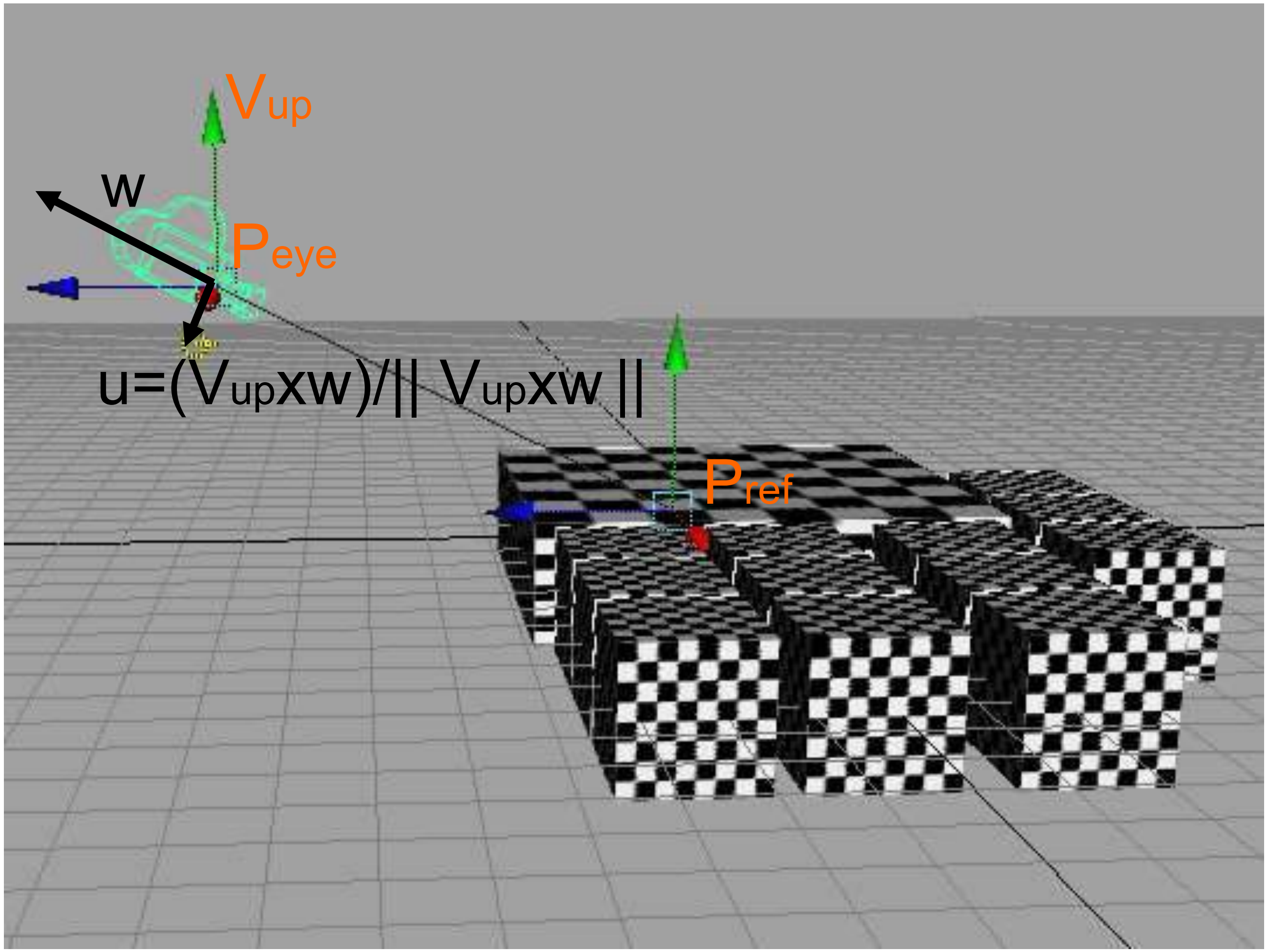
Viewing Transform



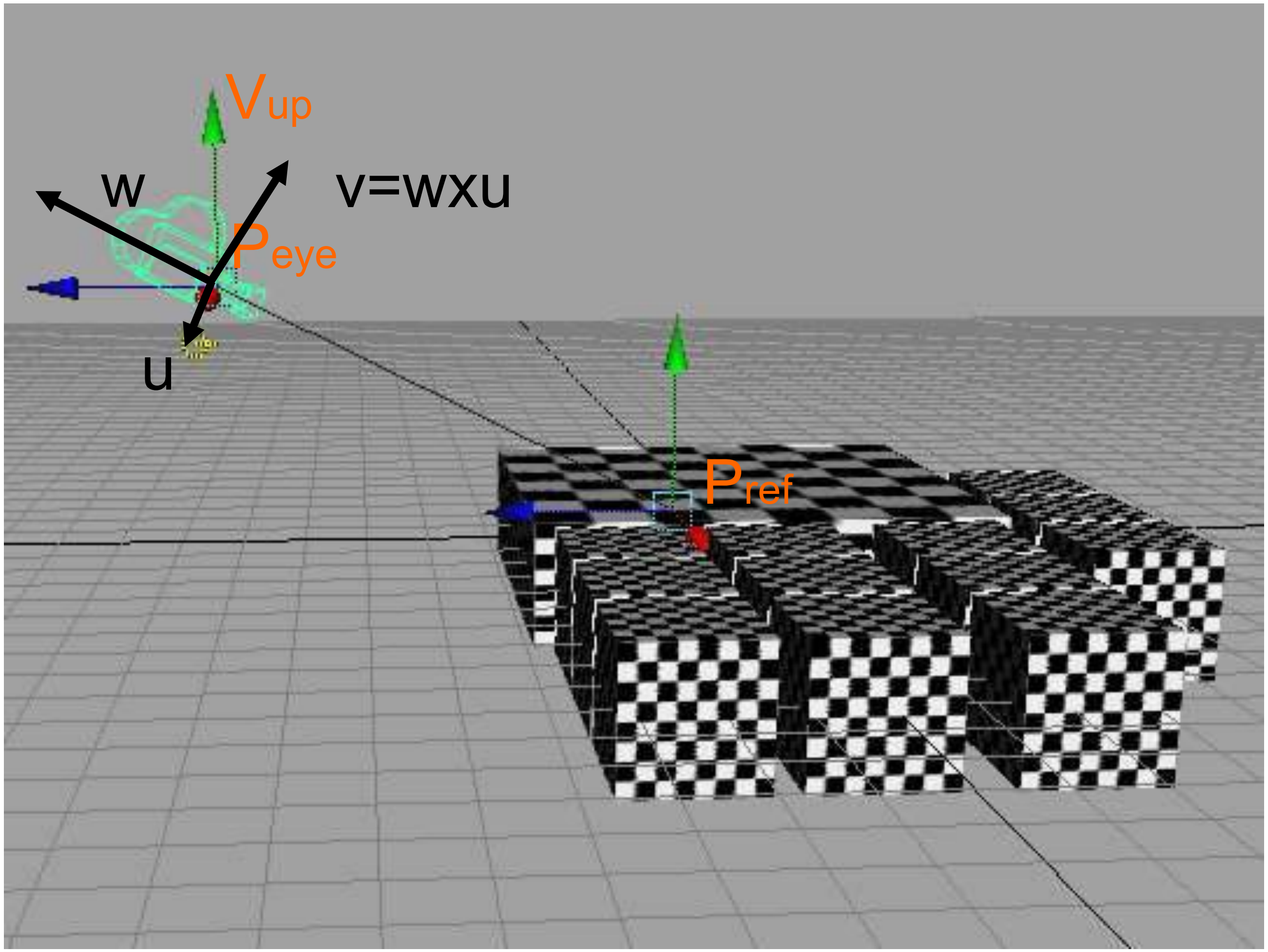
Camera model

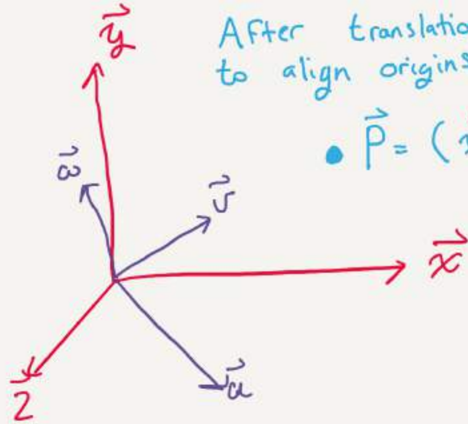


Viewing Transform



Viewing Transform





After translation to align origins

• $\vec{P} = (\overbrace{x, y, z}^{\text{scalars}})$

A) In Euclidean Basis

$$\vec{P} = x \underbrace{\begin{bmatrix}1\\0\\0\end{bmatrix}}_{\vec{e}_1} + y \underbrace{\begin{bmatrix}0\\1\\0\end{bmatrix}}_{\vec{e}_2} + z \underbrace{\begin{bmatrix}0\\0\\1\end{bmatrix}}_{\vec{e}_3}$$

But in $\vec{u}\vec{v}\vec{w}$ space

$$\vec{P} = a\vec{w} + b\vec{u} + c\vec{v}$$

This Defines a System of Equations

B) $a\vec{w} + b\vec{u} + c\vec{v} = x\vec{e}_1 + y\vec{e}_2 + z\vec{e}_3$

Exploit orthogonality:

$a = x\vec{w}^T\vec{e}_1 + y\vec{w}^T\vec{e}_2 + z\vec{w}^T\vec{e}_3$

$b = \dots$

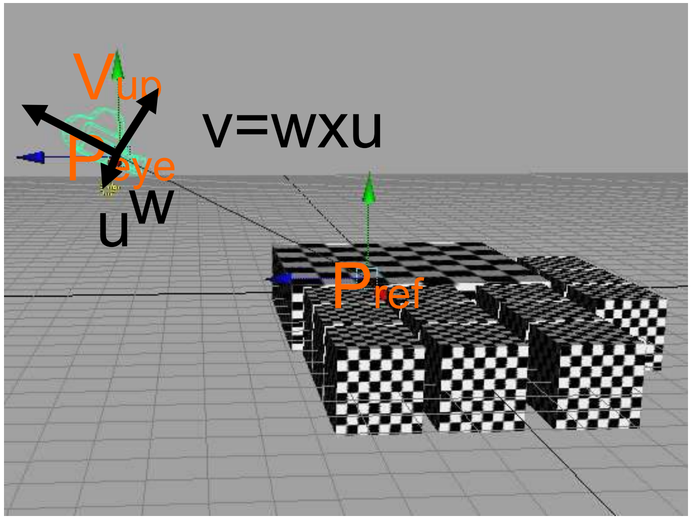
$c = \dots$

$$\Rightarrow \underbrace{\begin{bmatrix}a\\b\\c\end{bmatrix}}_{\vec{P} \text{ in } \vec{w},\vec{u},\vec{v}} = \begin{bmatrix}\vec{w}^T\vec{e}_1 & \vec{w}^T\vec{e}_2 & \vec{w}^T\vec{e}_3\\ \vec{u}^T\vec{e}_1 & \vec{u}^T\vec{e}_2 & \vec{u}^T\vec{e}_3\\ \vec{v}^T\vec{e}_1 & \vec{v}^T\vec{e}_2 & \vec{v}^T\vec{e}_3\end{bmatrix} \overbrace{\begin{bmatrix}x\\y\\z\end{bmatrix}}^{\vec{P} \text{ in } \vec{e}_1,\vec{e}_2,\vec{e}_3}$$

$\vec{v}^T\vec{e}_3 \rightarrow \cos(\theta^{\vec{v}}_{\vec{e}_3})$

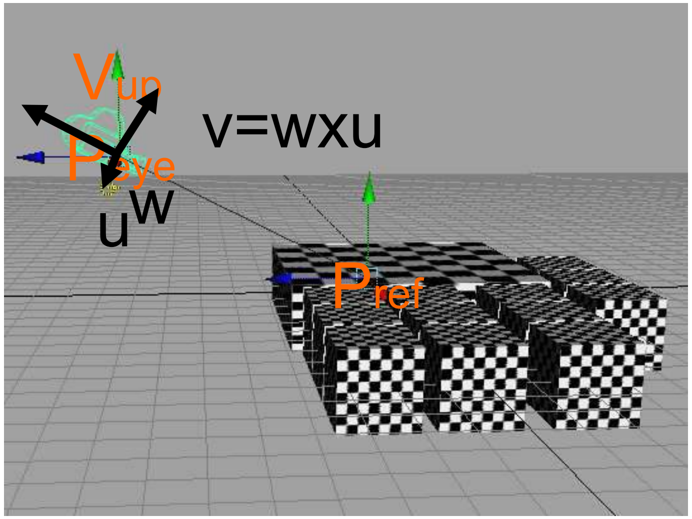
$\boxed{\text{Transformation Between Frames}}$

Change-of-basis Matrix



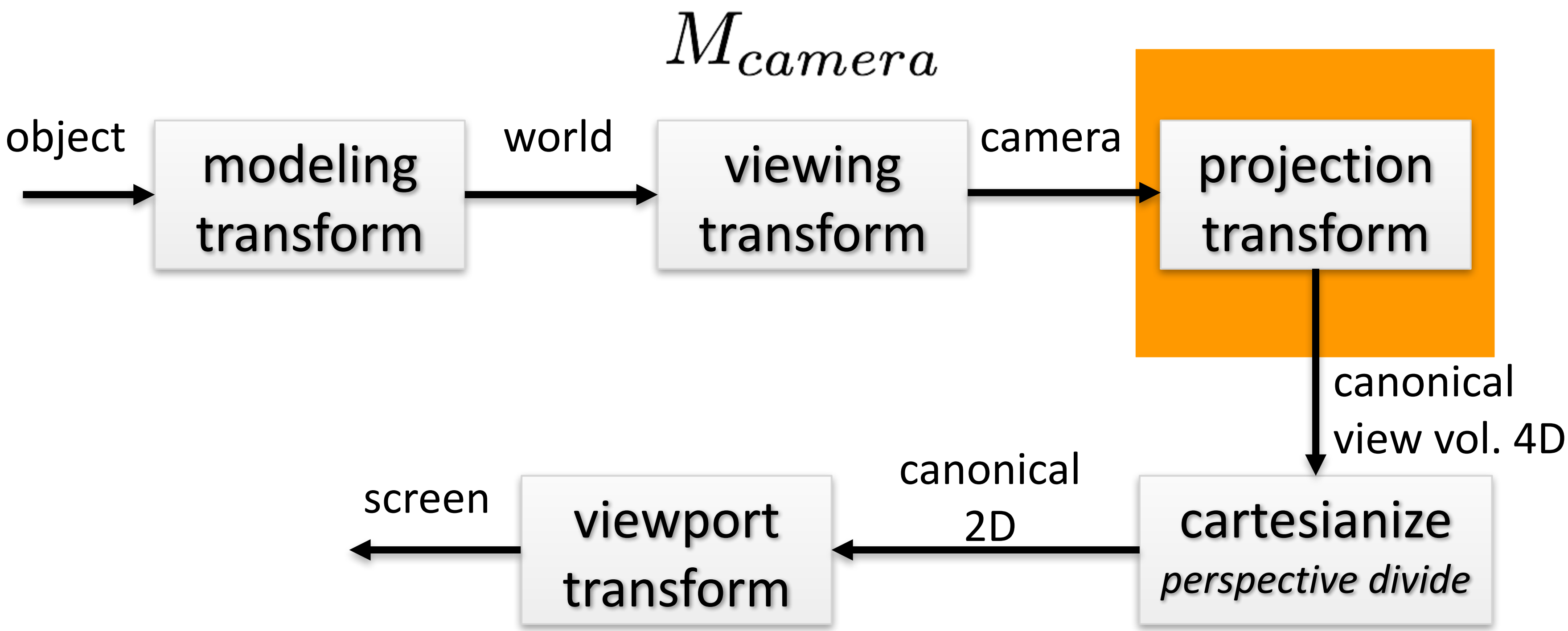
$$M_{camera} = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Change-of-basis Matrix



$$M_{camera} = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewing Pipeline



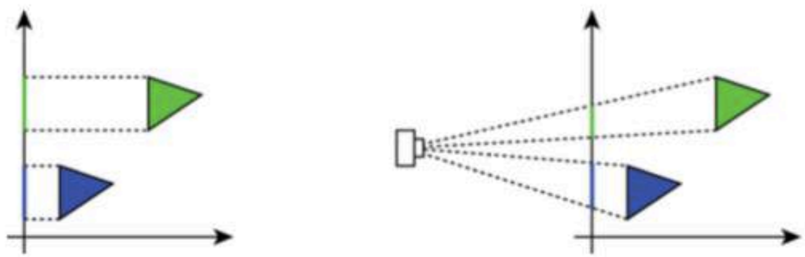
Camera model

What is the difference between these images?


frankfurt airport tunnel (wikipedia.com)
object-to-camera
distance affects an
object's projection
In real photos
parallel lines
do converge
in the distance!

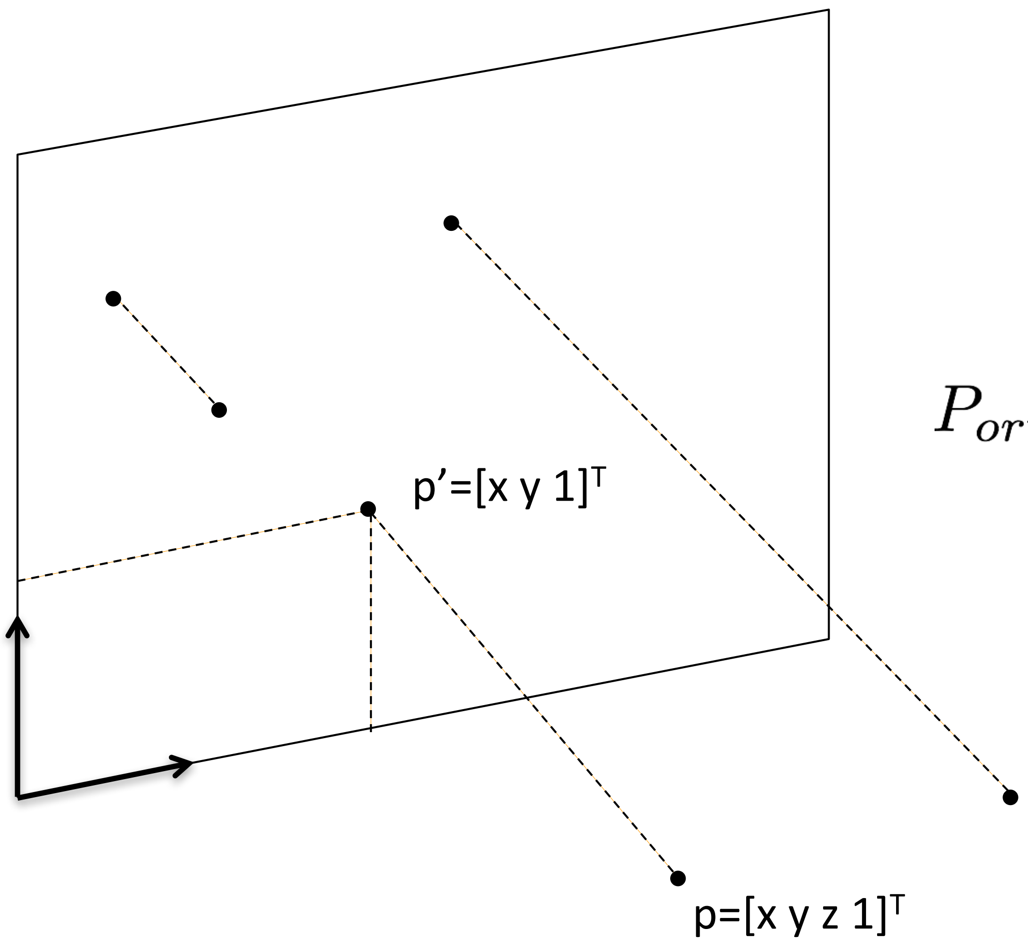
Parallel vs Perspective projection

- Parallel: usage in mechanical and architectural drawings
- Perspective projection: more natural and realistic



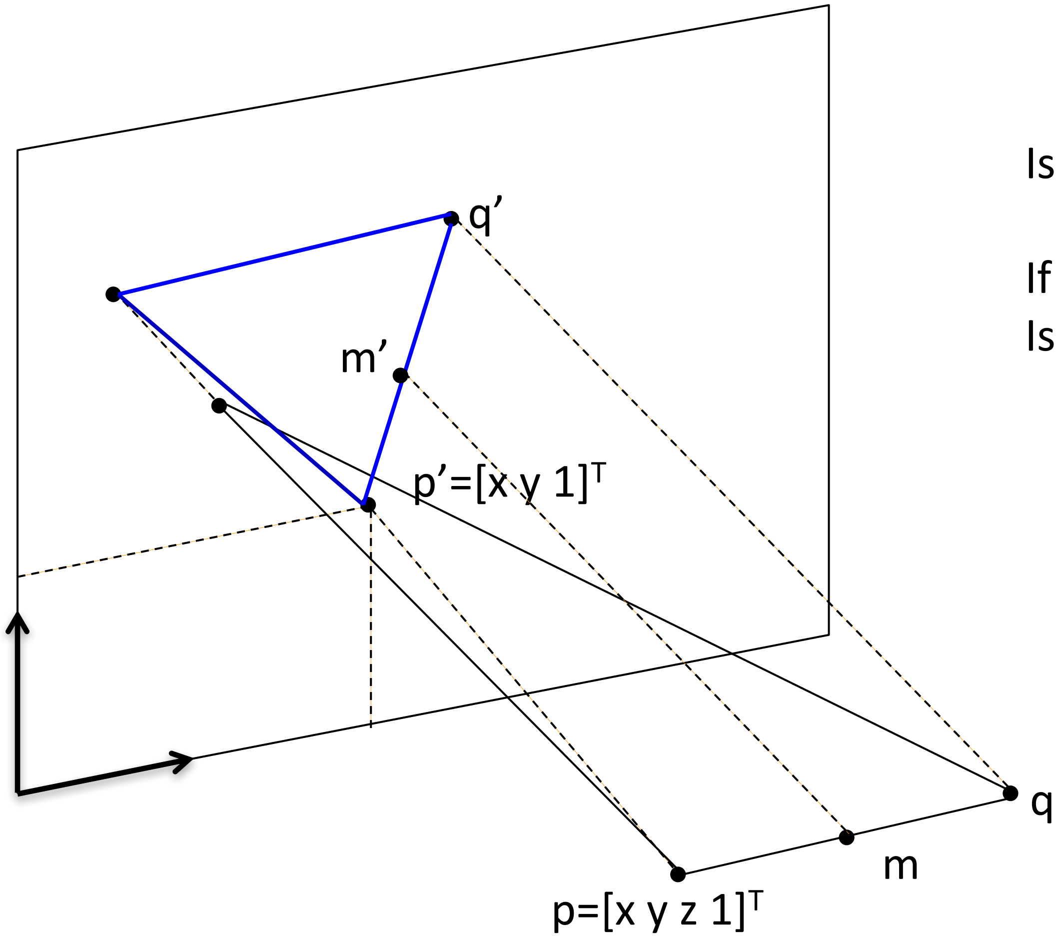
- How to get 3D objects perspectively correct on 2D screen?
- Note: usually your **API** takes care of most of this, but it's good to know what's going on behind those function calls (esp. when debugging your code)

Orthographic projection



$$P_{ortho} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Orthographic projection

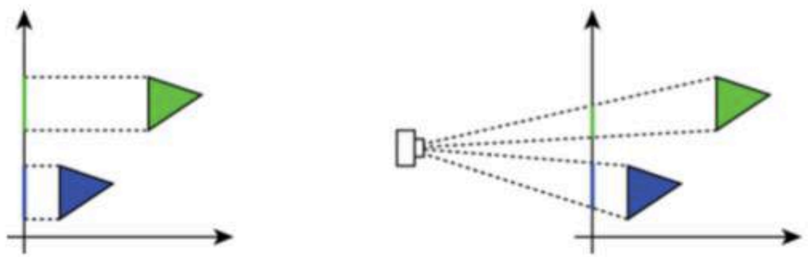


Is |p-q| = |p'-q'| ?

If m= (p+q)/2,
Is m' = (p'+q')/2?

Parallel vs Perspective projection

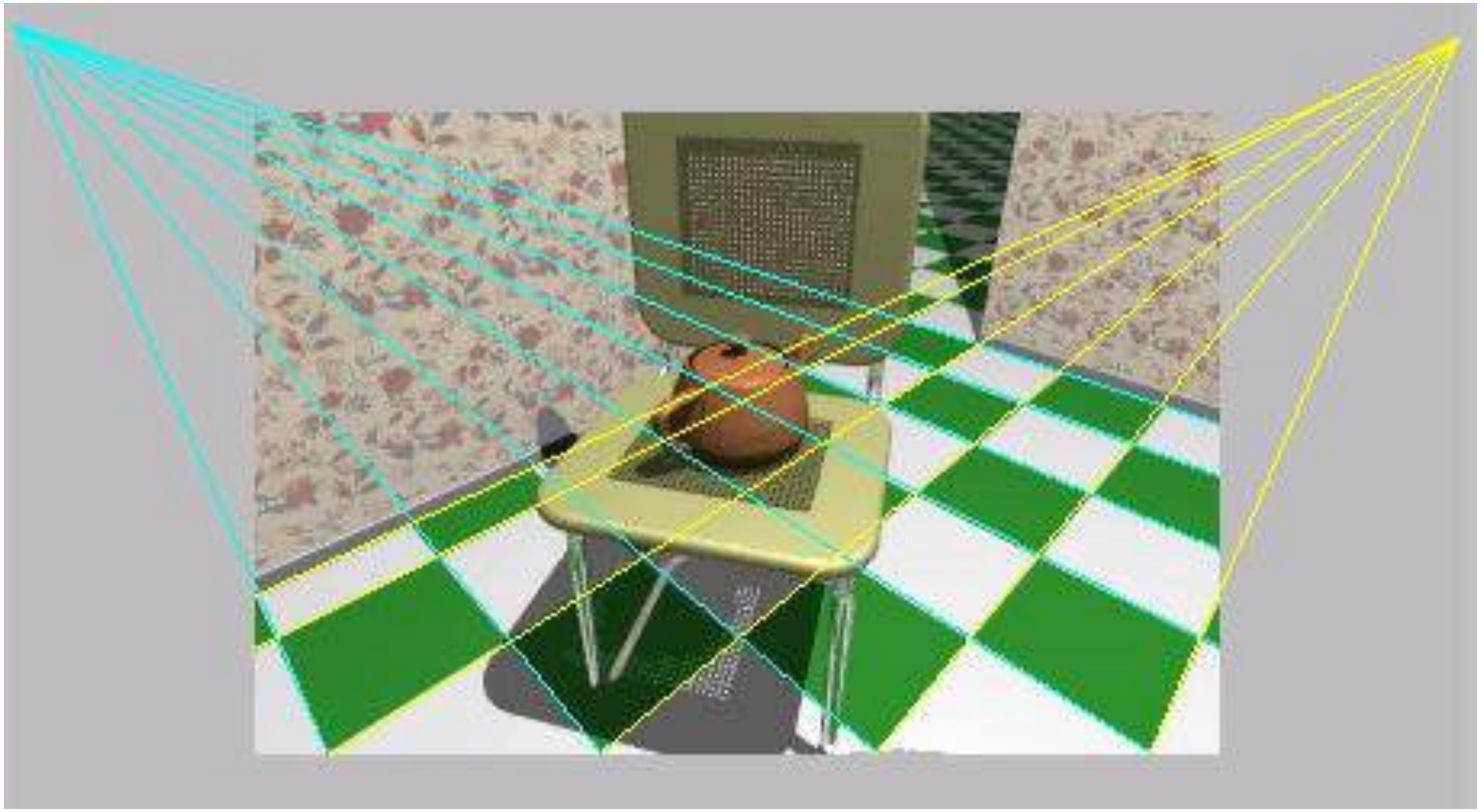
- Parallel: usage in mechanical and architectural drawings
- Perspective projection: more natural and realistic



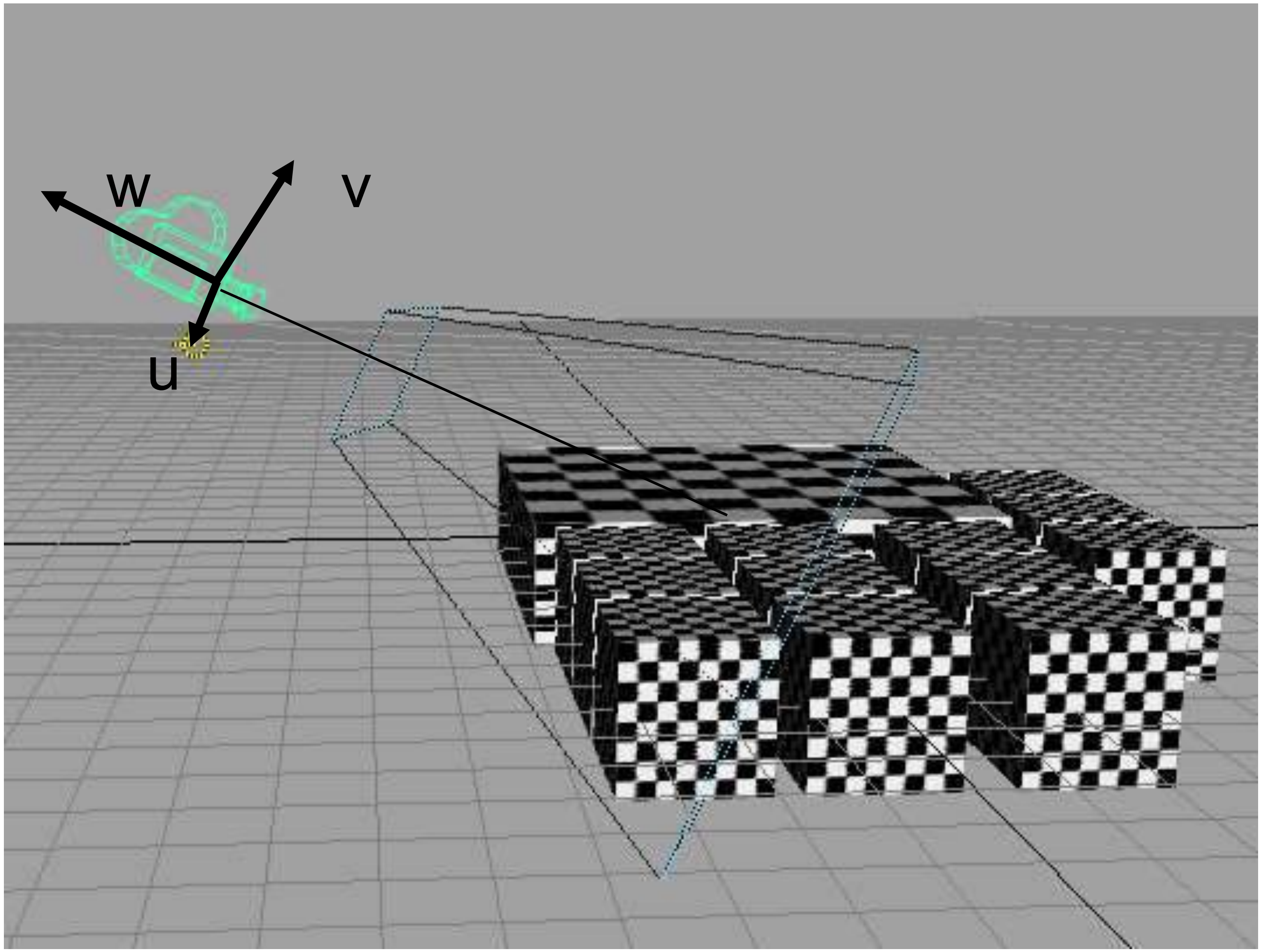
- How to get 3D objects perspectively correct on 2D screen?
- Note: usually your **API** takes care of most of this, but it's good to know what's going on behind those function calls (esp. when debugging your code)

Camera model

Perspective Projection



Perspective projection

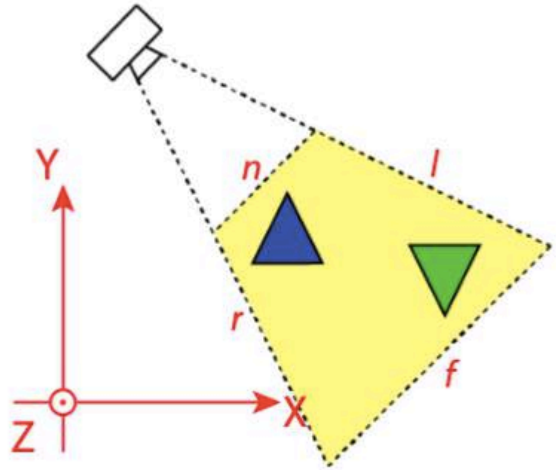


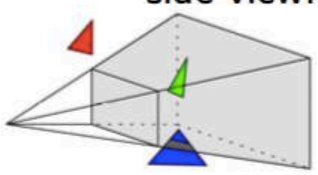
View frustum

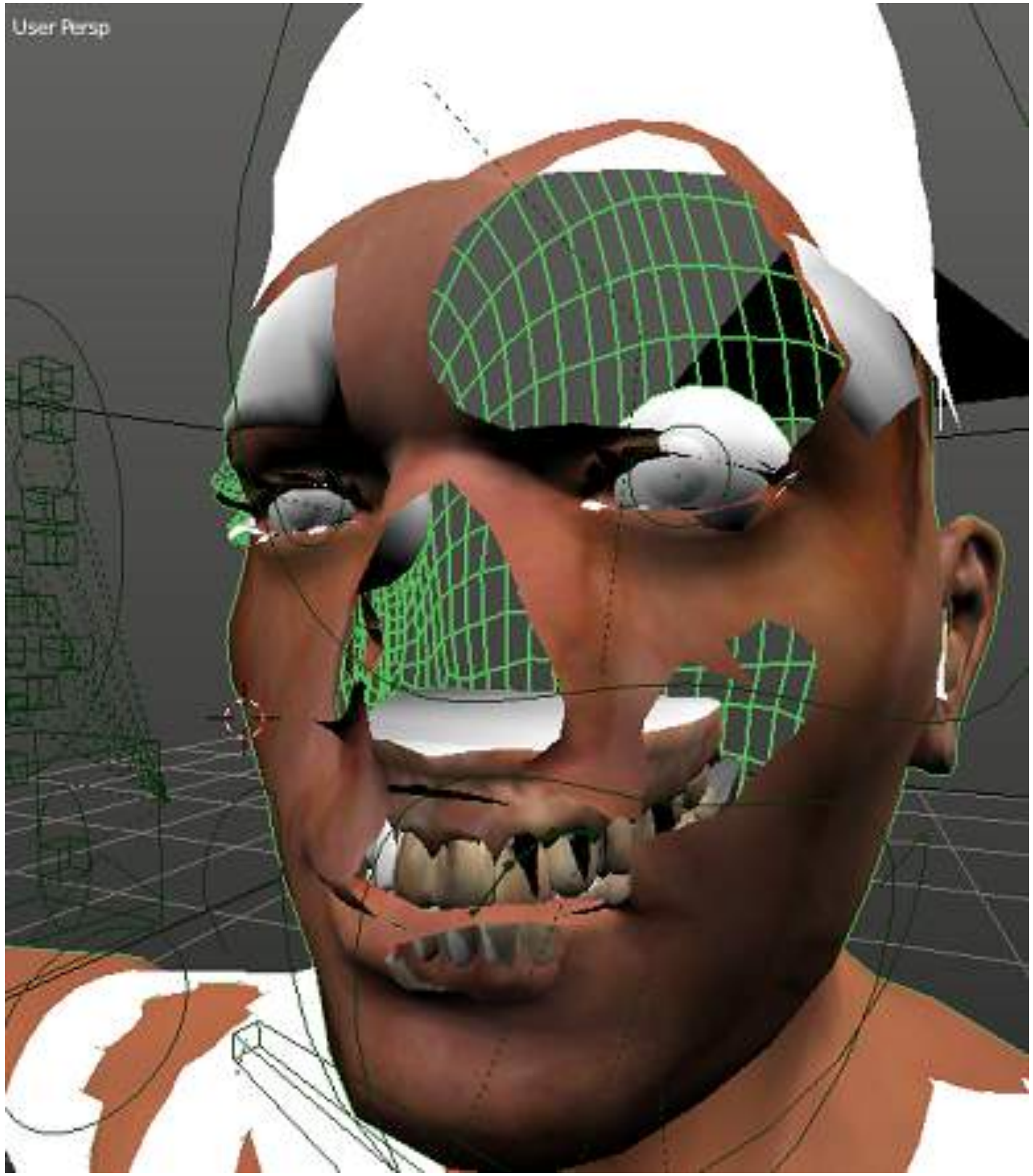
The view frustum (aka view volume) specifies everything that the camera can see. It's defined by

- the left plane l
- the right plane r
- the top plane t
- the bottom plane b
- the near plane n
- the far plane f

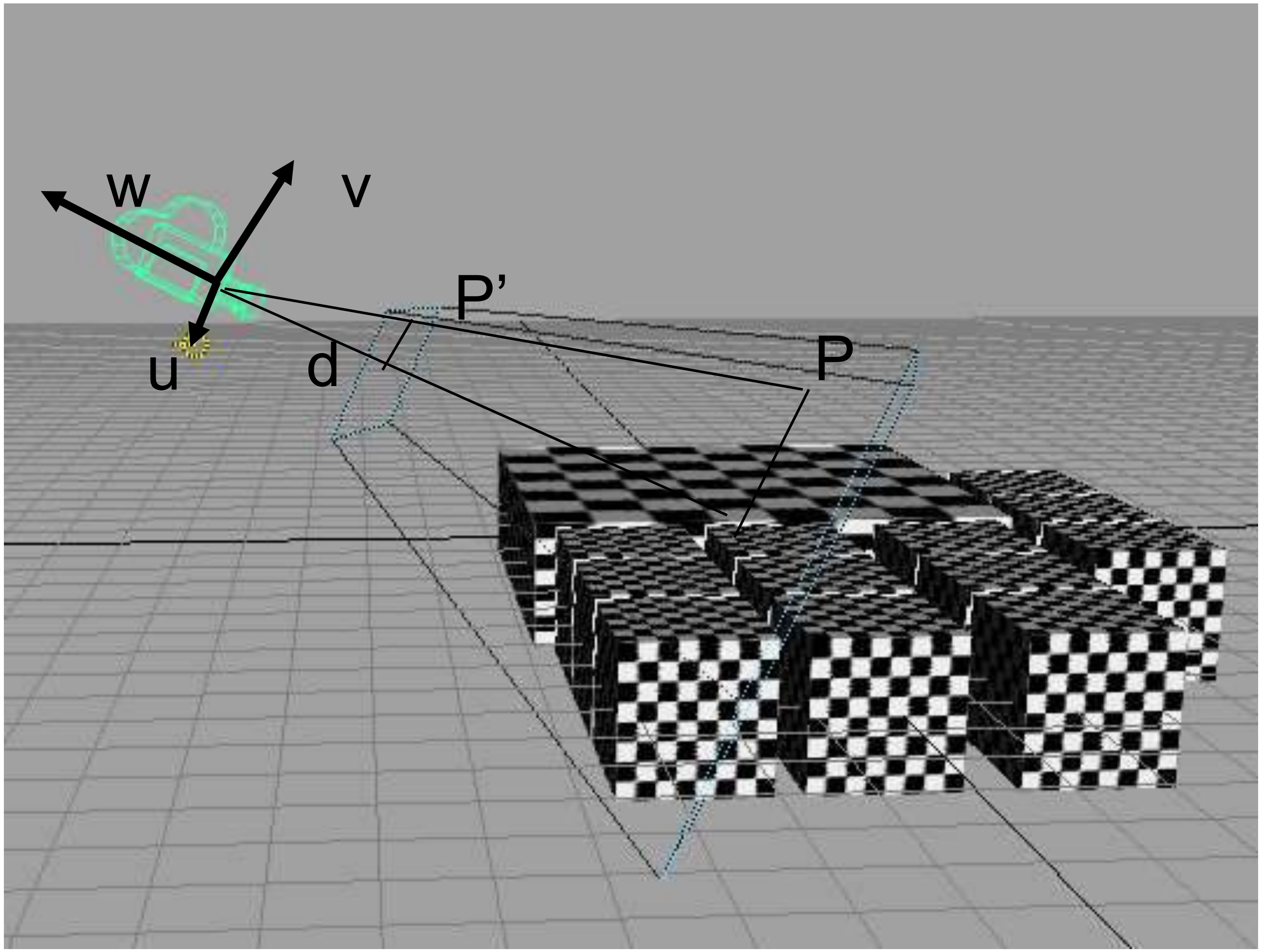
For now, we assume *wireframe models* that are *completely within* the view frustum



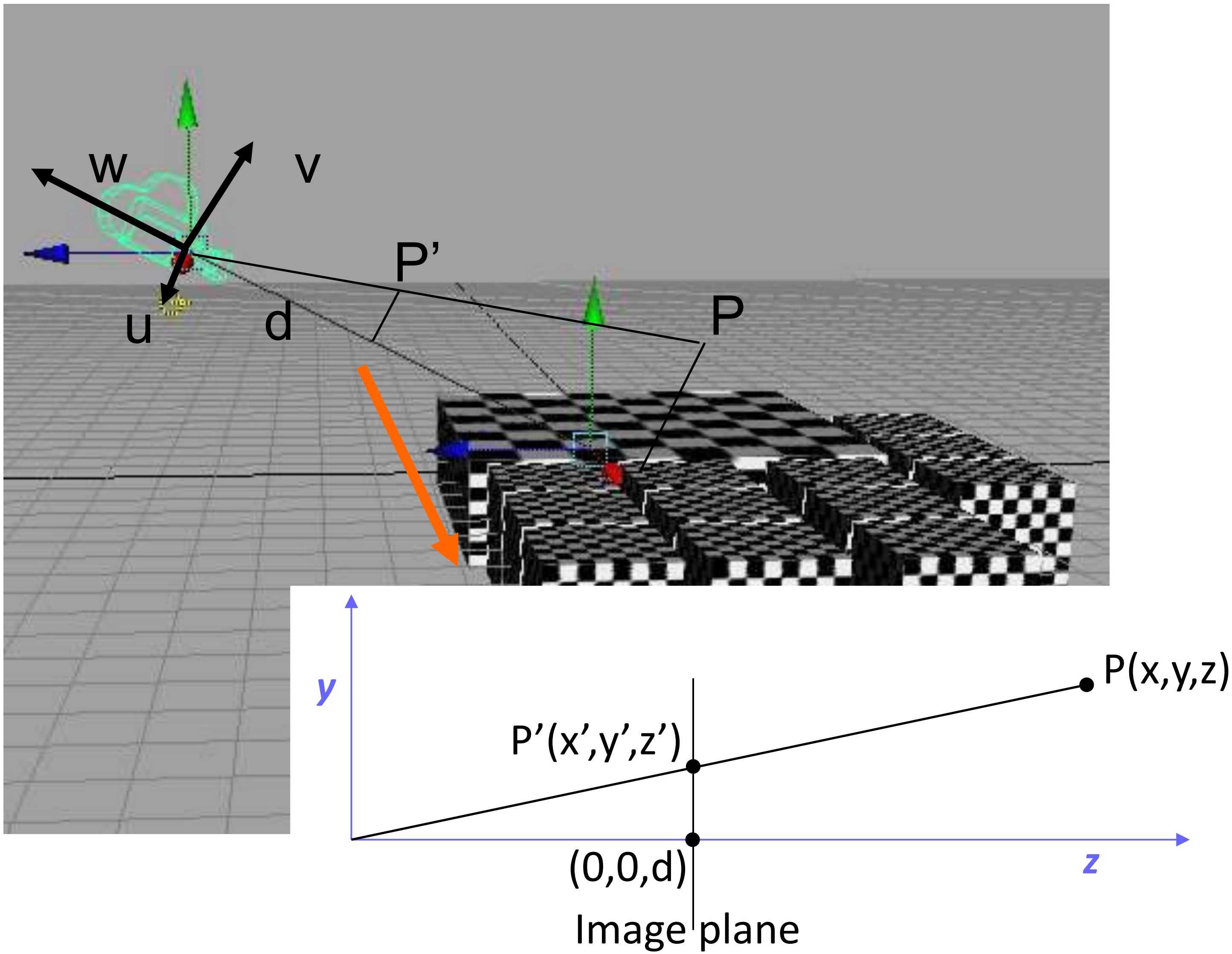



User Persp

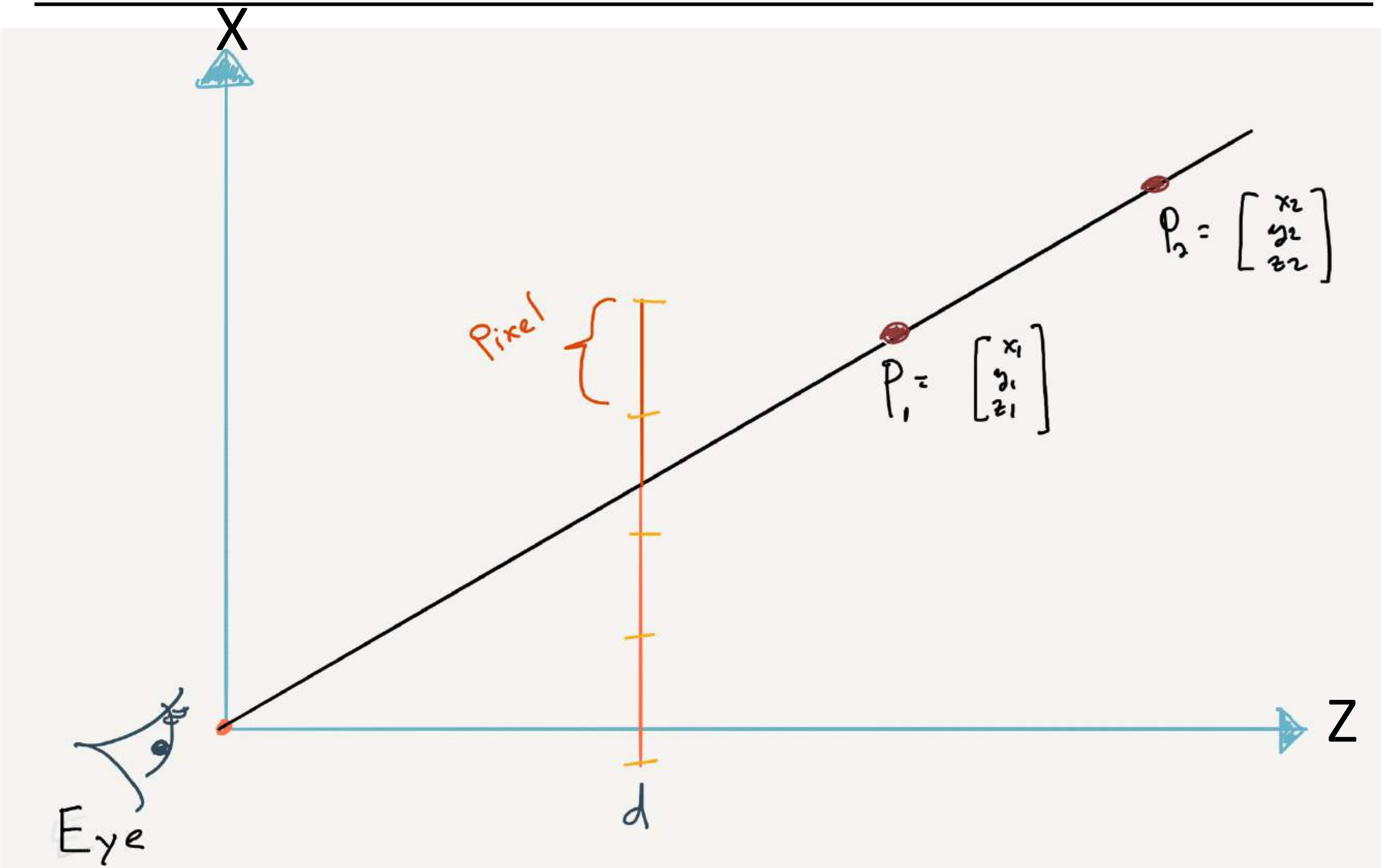
Perspective projection



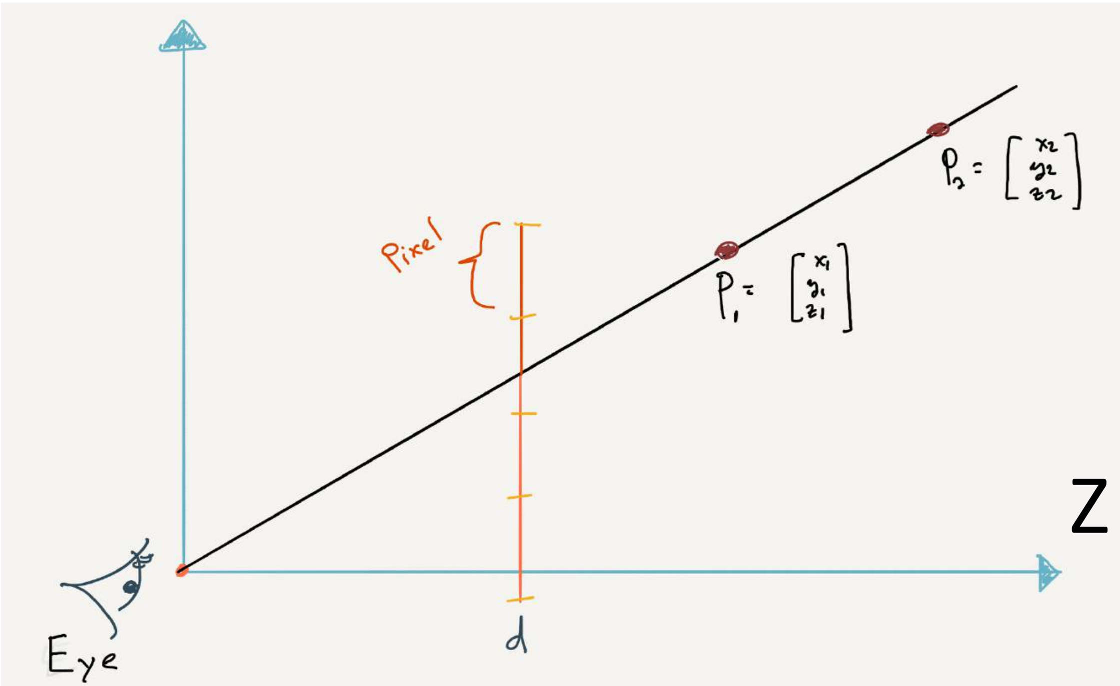
Simple Perspective



Simple Perspective



X



Z

$$P_{persp} = \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \\ 1 \end{pmatrix} \qquad w = \frac{z}{d}$$

Using Homogenous Coordinates

With homogeneous coordinates, the vector

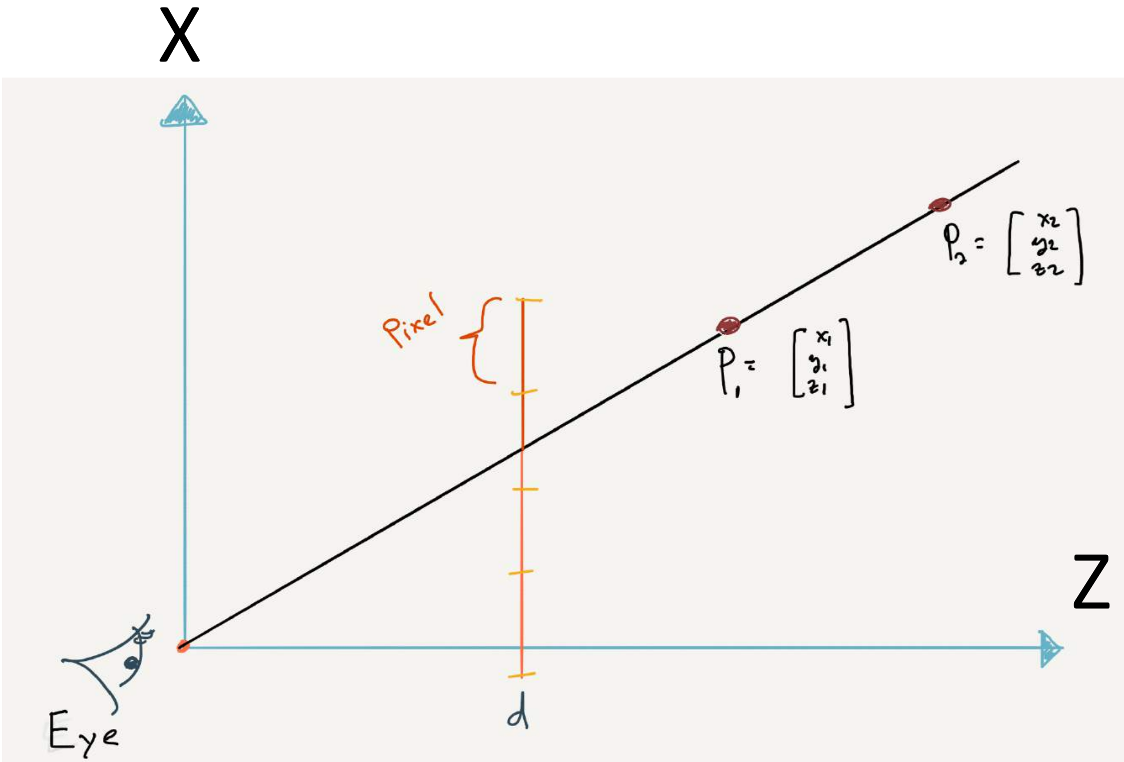
$$(x, y, z, 1) \quad \text{represents the point} \quad (x, y, z).$$

Now we extend this in a way that the homogeneous vector

$$(x, y, z, w) \quad \text{represents the point} \quad (x/w, y/w, z/w).$$

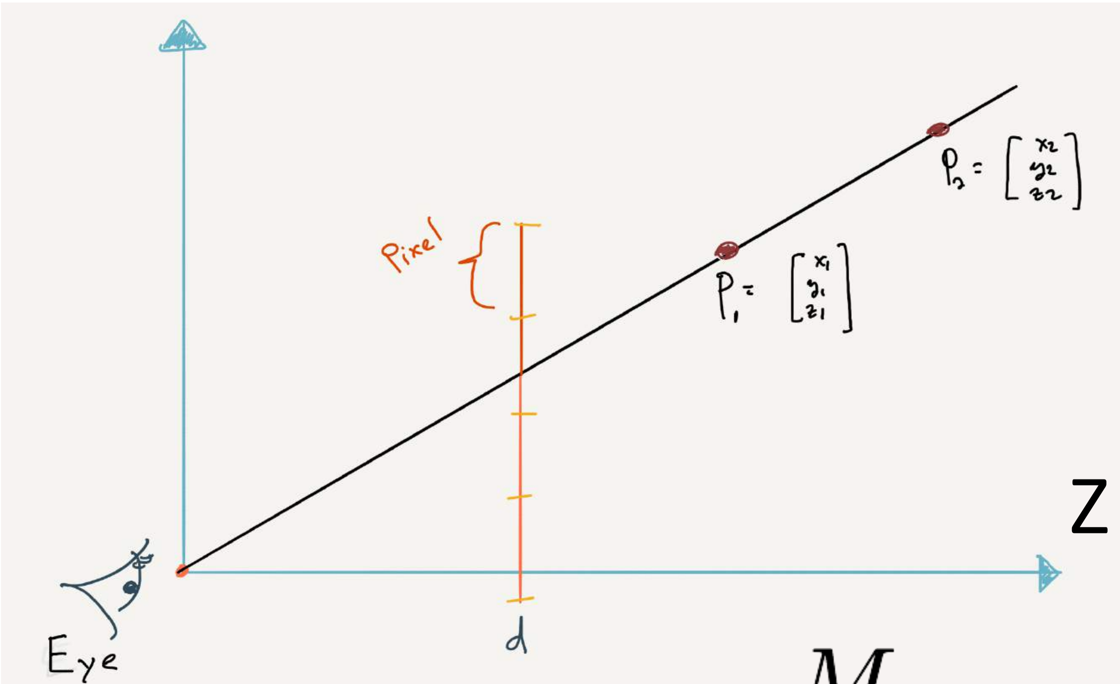
And matrix transformation becomes:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ e & f & g & h \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



$$P_{persp} = \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \\ 1 \end{pmatrix} \qquad w = \frac{z}{d}$$

X



$$P_{persp} = \overbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix}}^{M_{persp}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Homogenous Coordinate after
Orthographic Projection

$$M_{ortho}$$

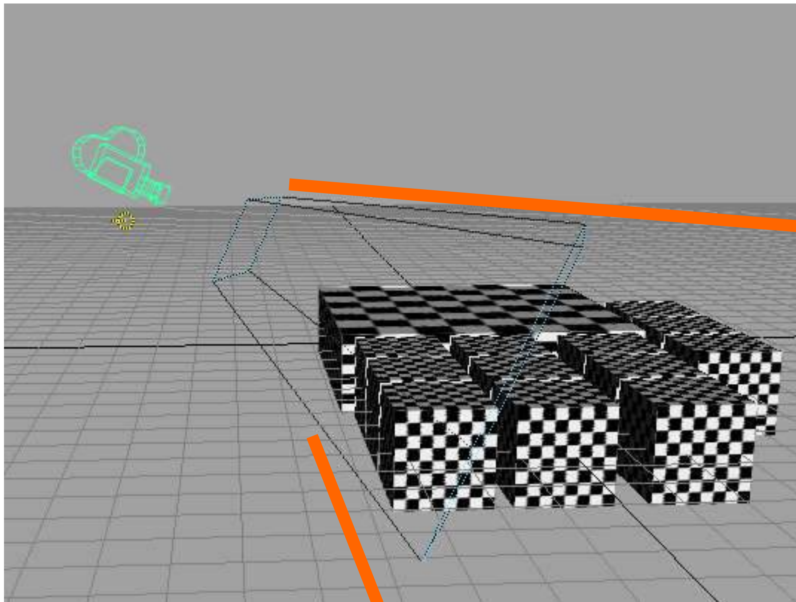
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

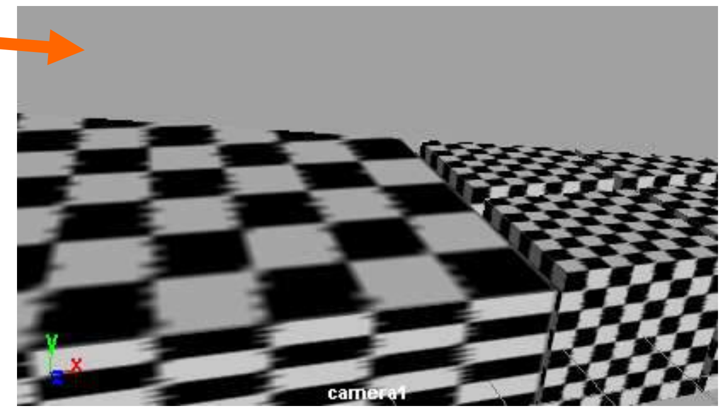
Homogenous Coordinate after
Perspective Projection

$$M_{persp}$$

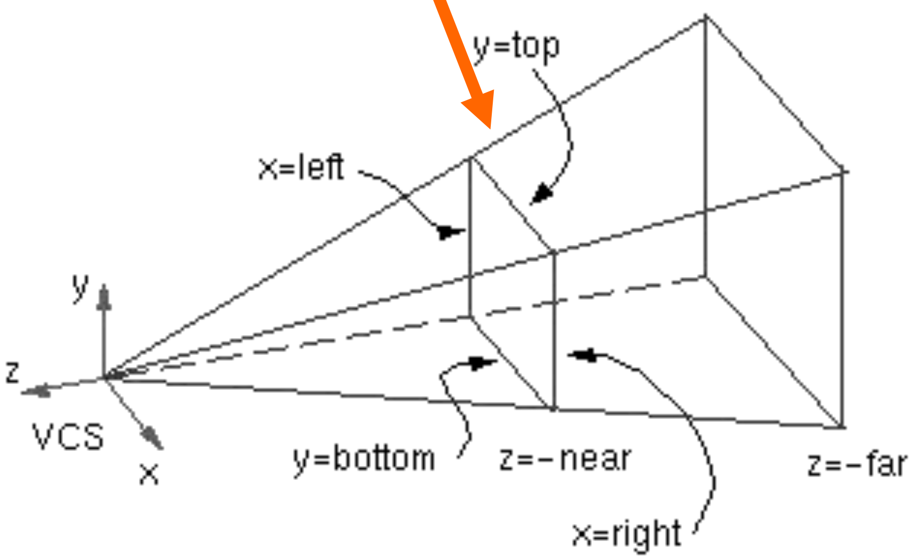
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{pmatrix}$$

Viewing volumes





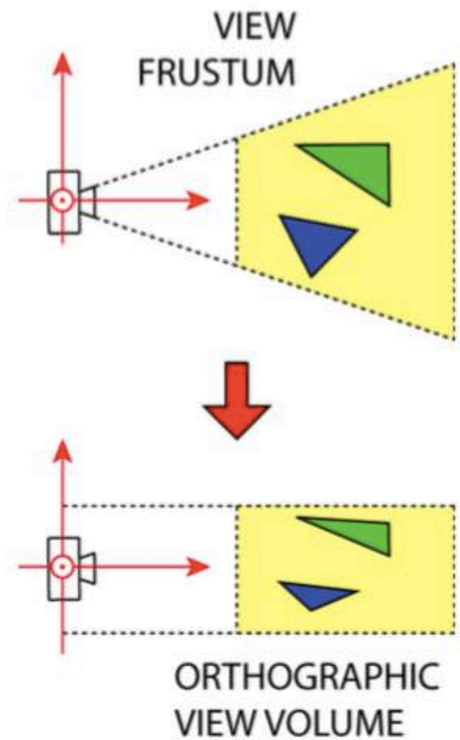
Projected image



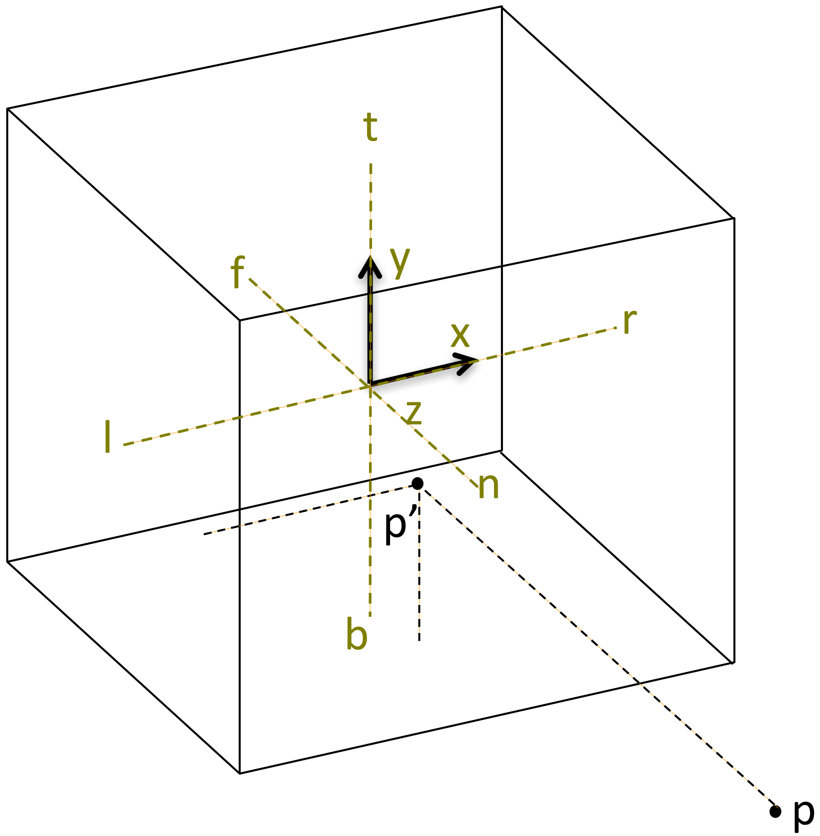
Transforming the View Frustim

We have to transform the view frustum into the orthographic view volume. The transformation needs to

- Map lines through the origin to lines parallel to the z axis
- Map points on the viewing plane to themselves.
- Map points on the far plane to (other) points on the far plane.
- Preserve the near-to-far order of points on a line.



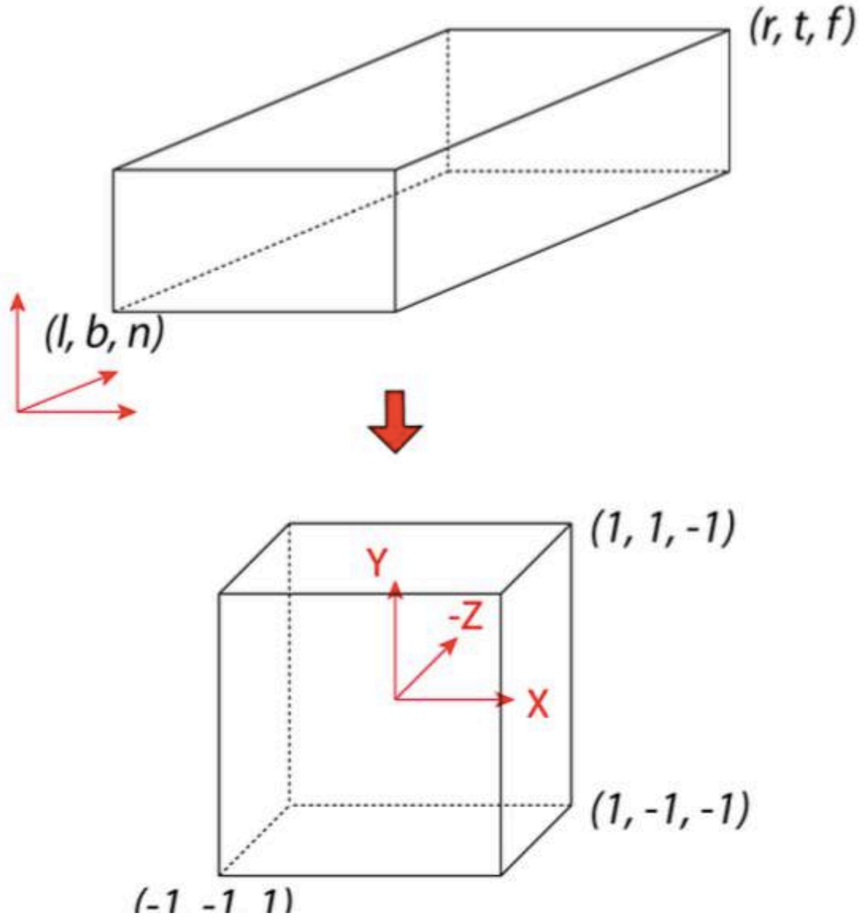
Cannonical view volume



Map 3D to a cube centered at the origin of side length 2!

The orthographic view volume

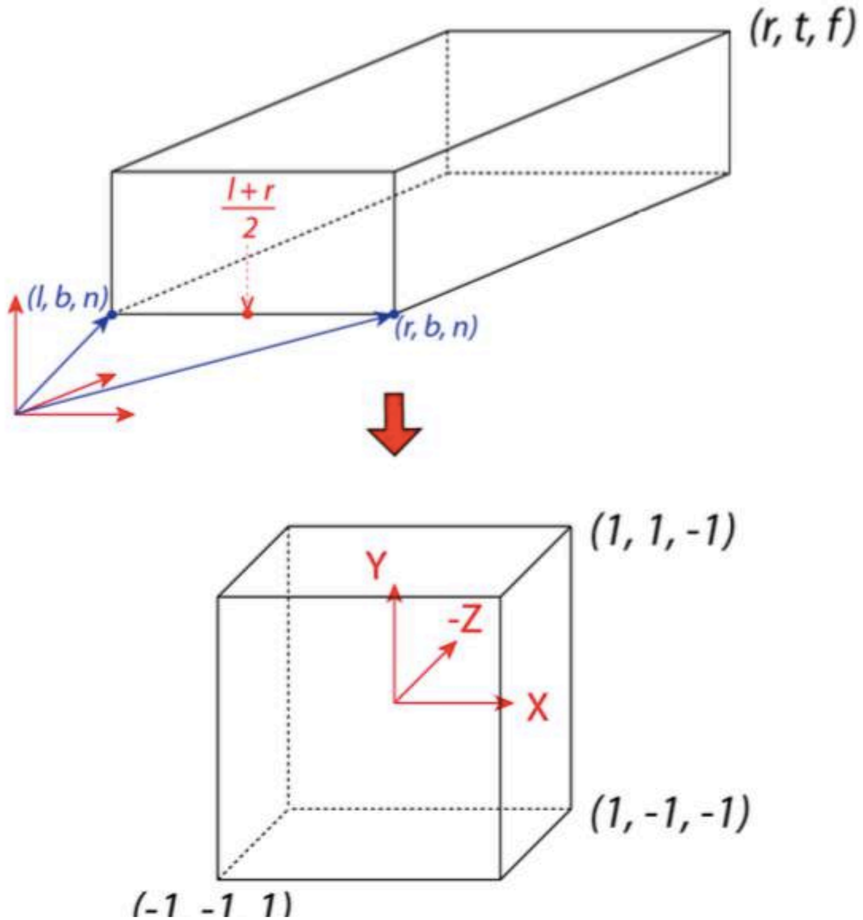
...how do we get the data from the axis-aligned box $[l, r] \times [b, t] \times [n, f]$ to a $2 \times 2 \times 2$ box around the origin?



The orthographic view volume

First we need to move the center to the origin:

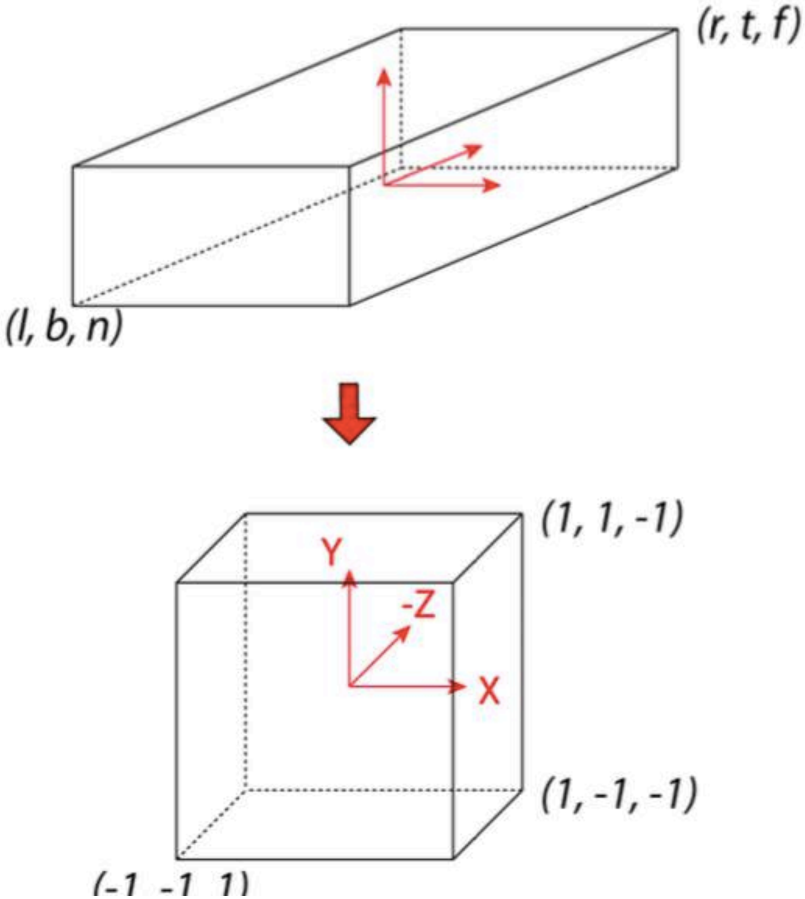
$$\begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



The orthographic view volume

Then we have to scale everything to $[-1, 1]$:

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Note: we divide by the length, e.g. $\frac{1}{\frac{r-l}{2}}$ for the $x-$coordinate value

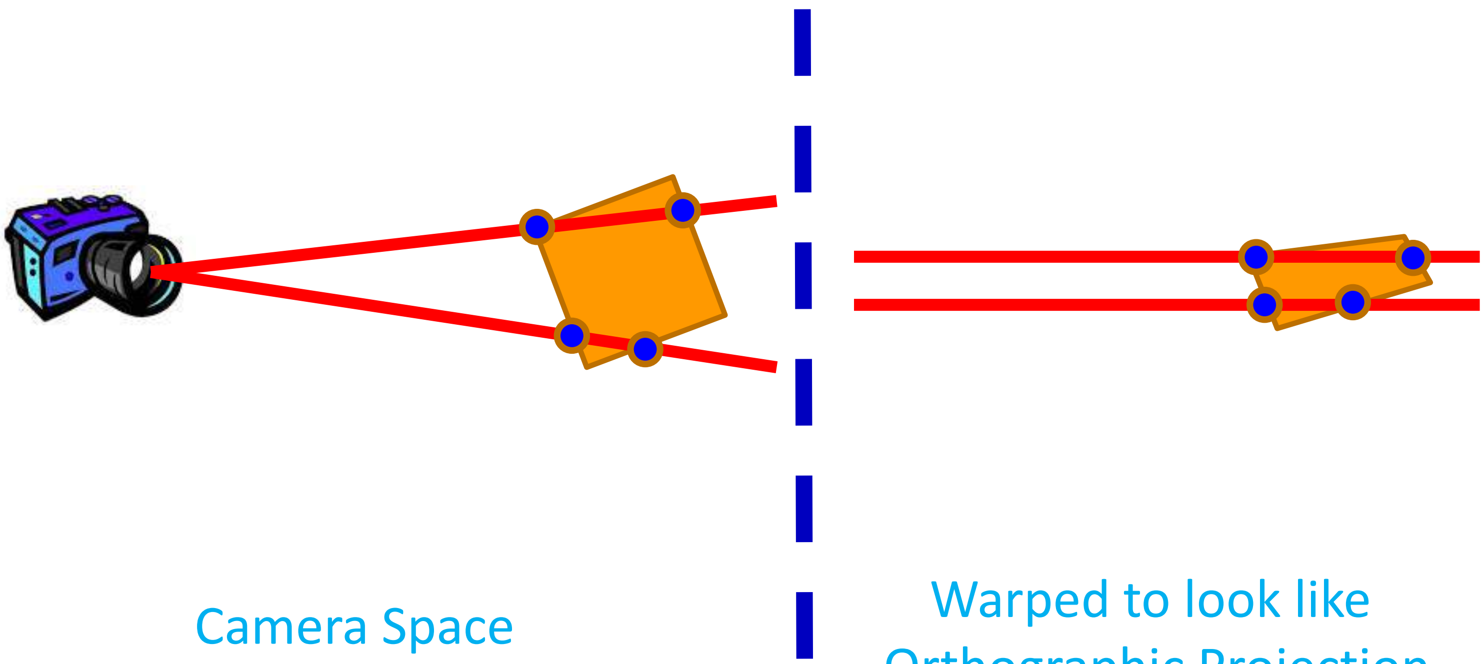
The orthographic view volume

Since these are just matrix multiplications (associative!), we can combine them into one matrix:

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

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

Homogeneous Coords and Perspective



Camera Space

Warped to look like
Orthographic Projection

Perspective Matrix

The following matrix will do the trick:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Remember that

- we are looking in negative Z-direction
- n, f denote the near and far plane of the view frustum
- n serves as projection plane

Let's verify that ...

Perspective Matrix

$$M_{persp}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n} - f \\ \frac{z}{n} \end{pmatrix} \xrightarrow{\text{homogenize}} \begin{pmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n + f - \frac{fn}{z} \\ 1 \end{pmatrix}$$

Indeed, that gives the correct values for x_s and y_s.

But what about z? Remember our requirements for z:

- stays the same for all points on the near and fare planes
- does not change the order along the $Z-$axis for all other points

Verify

We have $z_s = n + f - \frac{fn}{z}$ and need to prove that ...

- points on the near plane are mapped to themselves, i.e. if $z = n$, then $z_s = n$:

  $$z_s = n + f - \frac{fn}{n} = n + f - f = n$$

  and obviously $x_s = \frac{nx}{n} = x$ and $y_s = \frac{ny}{n} = y$.

- points on the far plane stay on the far plane, i.e. if $z = f$, then $z_s = f$:

  $$z_s = n + f - \frac{fn}{f} = n + f - n = f$$

and ...

Verify

We have $z_s = n + f - \frac{fn}{z}$ and need to prove that . . .

- $z-$values for points within the view frustum stay within the view frustum,
  i.e. if $z > n$ then $z_s > n$:

  $$z_s = n + f - \frac{fn}{z} > n + f - \frac{fn}{n} = n$$

  and if $z < f$ then $z_s < f$:

  $$z_s = n + f - \frac{fn}{z} < n + f - \frac{fn}{f} = f$$

and . . .

Verify

We have $z_s = n + f - \frac{fn}{z}$ and need to prove that ...

- the order along the Z-axis is preserved, i.e. if $0 > n \geq z_1 > z_2 \geq f$ then $z_{1s} > z_{2s}$:

  With $z_{1s} = n + f - \frac{fn}{z_1}$ and $z_{2s} = n + f - \frac{fn}{z_2}$ we get:

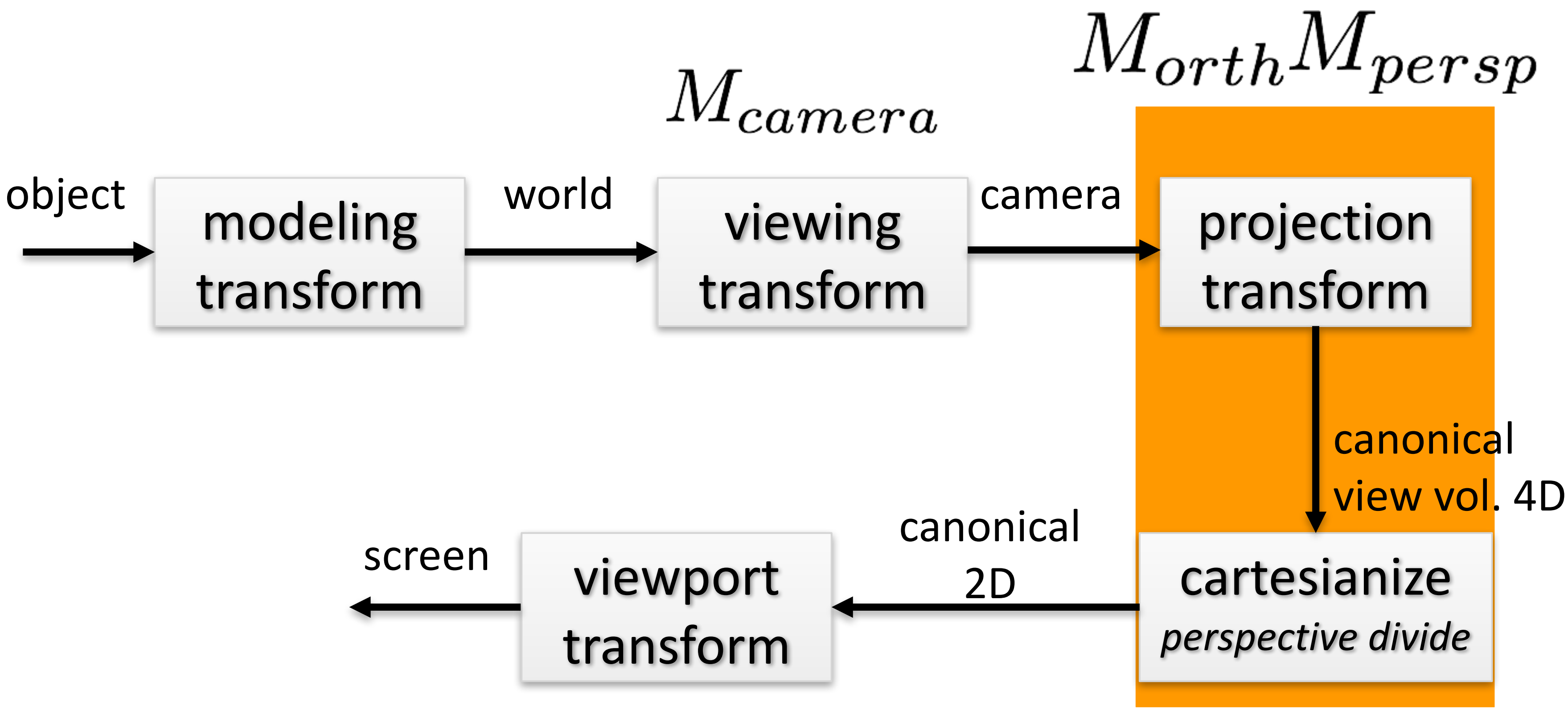
  $$z_{1s} - z_{2s} = \frac{fn}{z_2} - \frac{fn}{z_1} = \frac{(z_1 - z_2)fn}{z_1 z_2}.$$

  Because of $f, z_1, z_2, n < 0$ we have $\frac{fn}{z_1 z_2} > 0$, and because of $z_1 > z_2$, we have $z_1 - z_2 > 0$, so

  $$z_{1s} - z_{2s} > 0 \text{ or}$$

  $$z_{1s} > z_{2s}$$

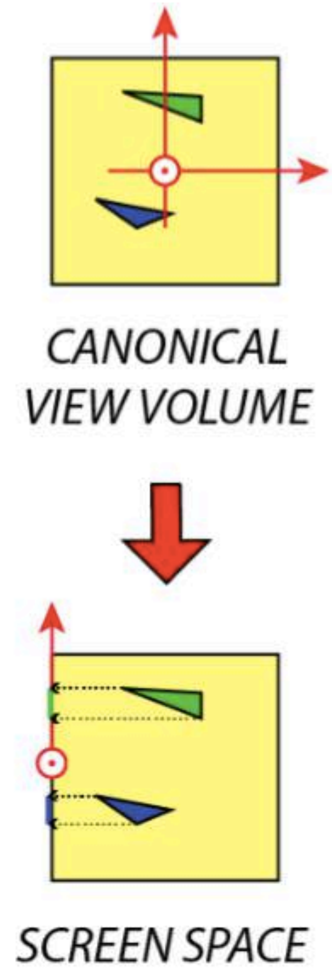
Viewing Pipeline



$$\begin{pmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n+f-\frac{fn}{z} \\ 1 \end{pmatrix}$$

Viewport Transform

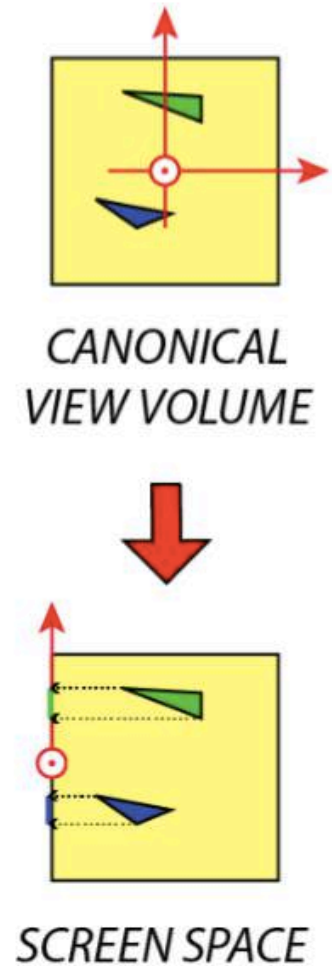
Now all that's left is a parallel projection along the Z-axis (every easy) and . . .



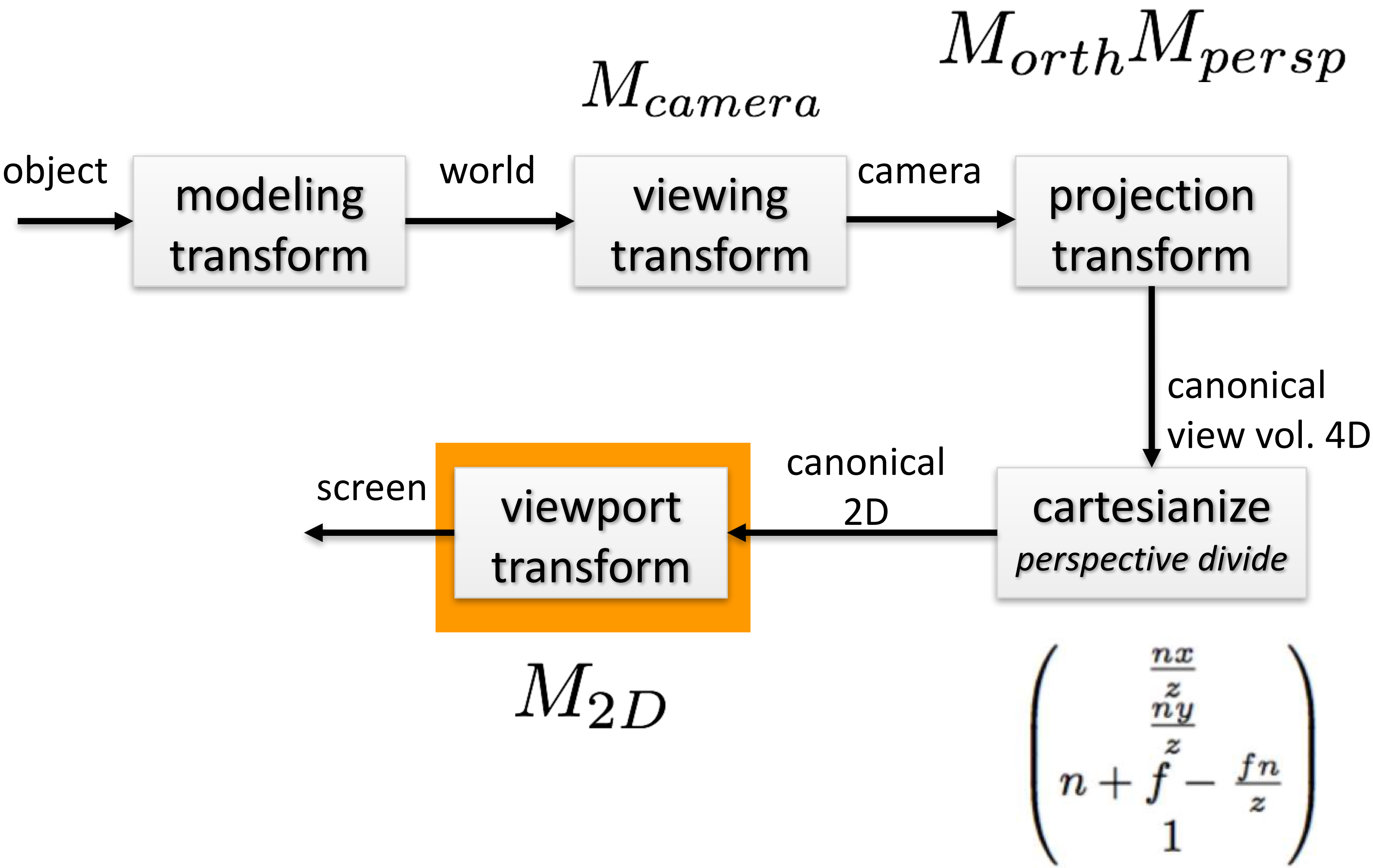
Viewport Transform

Now all that's left is a parallel projection along the Z-axis (every easy) and . . .

$$M_{2D} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Viewing Pipeline



Viewport Transform

…a windowing transformation in order to display the square $[-1, 1]^2$ onto an $n_x \times n_y$ image.

Again, these are just some (simple) matrix multiplications