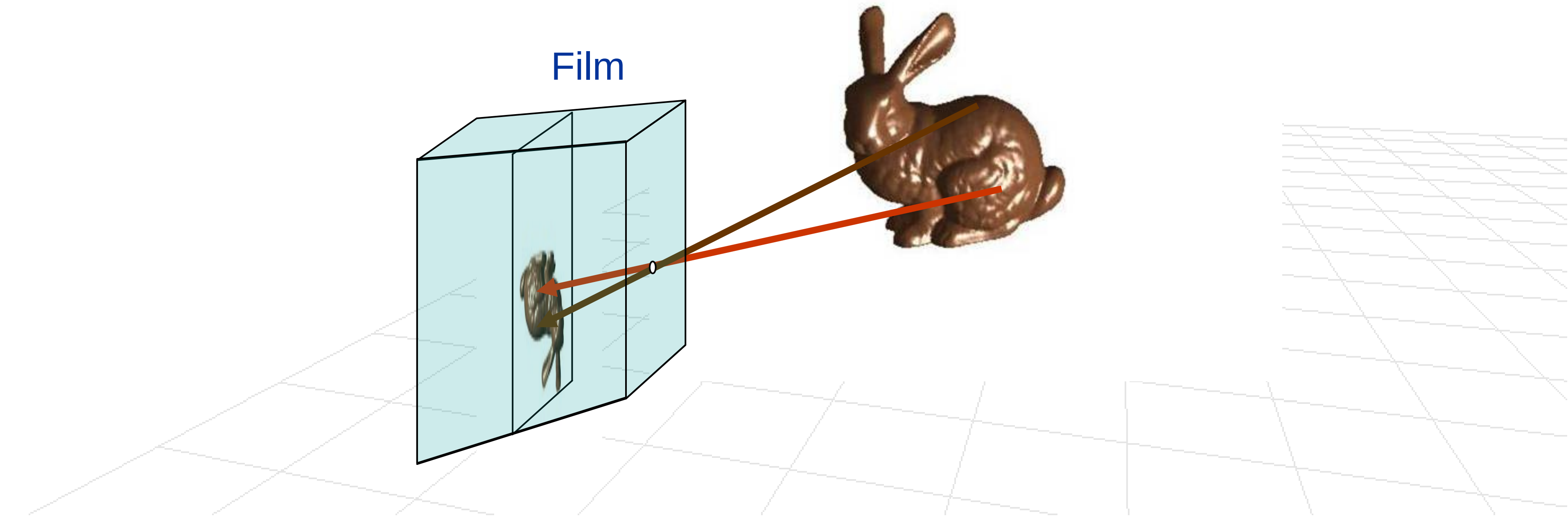
Announcements

- Assignment 1
  - theory (**due Today**)
  - programming (**due next Friday**)
    - You should **draw polygons** (not line strips) for the parts of the penguin
    - Issues with OpenGL should be resolved soon
- **Practice midterms** are now on-line
  - Solutions will not be made available, but you can ask TA or myself questions during office hours (and tutorial)
  - We will have extra office hours before the exam

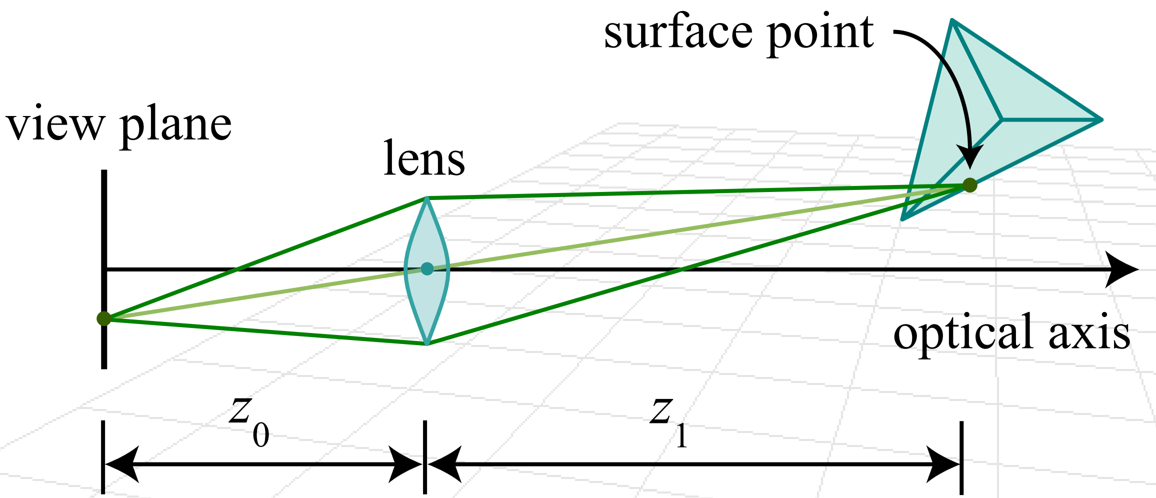
Last class review …

- Camera models
  - **Pinhole camera**



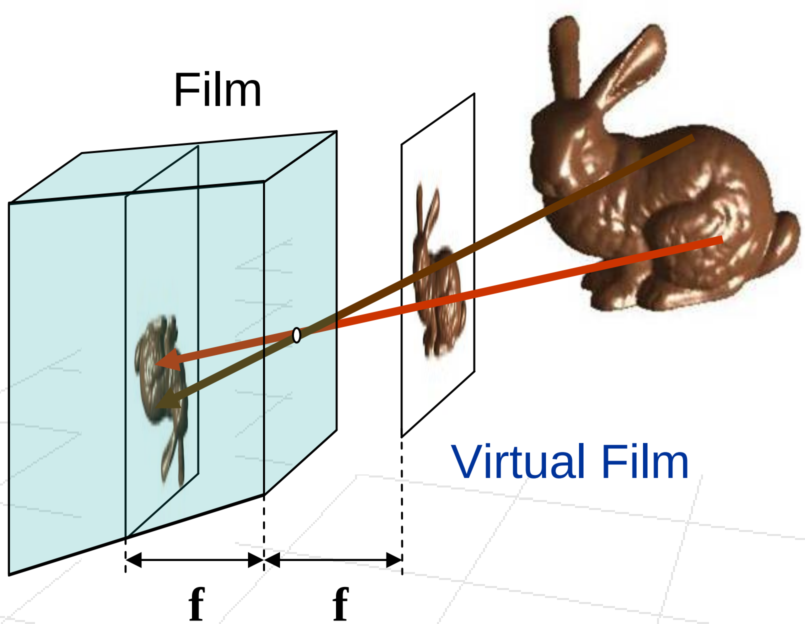
Last class review …

- Camera models
  - Pinhole camera
  - **Thin Lens Model** – lens is used to focus the light



Last class review …

- Camera models
  - Pinhole camera
  - Thin Lens Model – lens is used to focus the light
  - Relationship between thin lens model and pinhole camera
  - **Conceptual pinhole camera**



Last class review …

- Camera models
  - Pinhole camera
  - Thin Lens Model – lens is used to focus the light
  - Relationship between thin lens model and pinhole camera
  - Conceptual pinhole camera
  - **Perspective Projection**

$$\mathbf{y}^* = \frac{\mathbf{f}}{\mathbf{p}_z}\mathbf{p}_y \qquad \mathbf{x}^* = \frac{\mathbf{f}}{\mathbf{p}_z}\mathbf{p}_x$$

Lets step back again …

- What do we need to render a scene
  - **Scene with 3D objects**
  - Position and orientation of camera in the world coordinates
  - Transformation of objects from world to camera coordinates
  - **Project the objects onto film**
  - Visibility (with respect to the view volume)
  - No need to render everything, only things we can see

Camera Models
Part 2

Computer Graphics, CSCD18

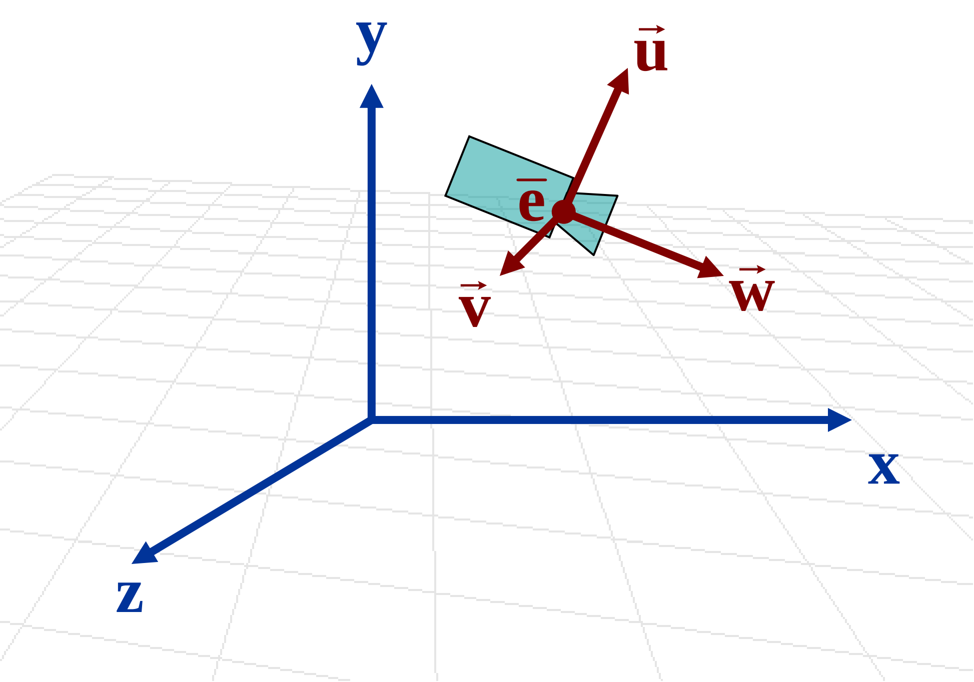
Fall 2008

Instructor: Leonid Sigal

Position and Orientation of Camera

- How can we specify a camera coordinate frame
  - We need an origin (at the pinhole) – lets call it $\bar{\mathbf{e}}$, and 3 unit vectors to define the camera coordinate frame $\vec{\mathbf{u}}, \vec{\mathbf{v}}, \vec{\mathbf{w}}$
- In general,
  - Camera can be anywhere in the world
  - Can move as a function of time

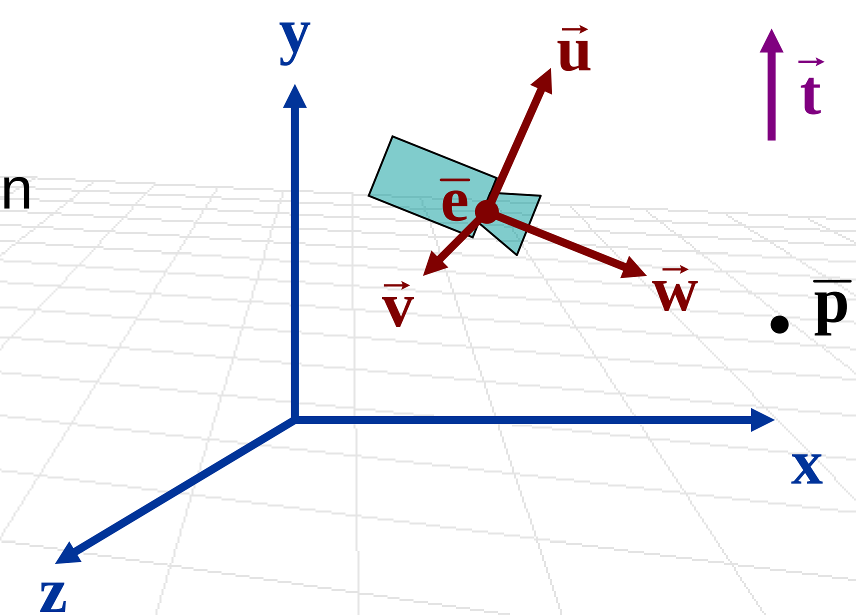
Bullet Time effect – Movie "The Matrix"



Position and Orientation of Camera

- How can we specify a camera coordinate frame
  - We need an origin (at the pinhole) – lets call it $\bar{\mathbf{e}}$, and 3 unit vectors to define the camera coordinate frame $\vec{\mathbf{u}}, \vec{\mathbf{v}}, \vec{\mathbf{w}}$
- How can we intuitively specify $\vec{\mathbf{u}}, \vec{\mathbf{v}}, \vec{\mathbf{w}}$
  - Let's pick a **point in the scene where we want to look** - $\bar{\mathbf{p}}$

$$\vec{\mathbf{w}} = \frac{\bar{\mathbf{p}} - \bar{\mathbf{e}}}{\left\|\bar{\mathbf{p}} - \bar{\mathbf{e}}\right\|}$$



  - Designate **up direction** $\vec{\mathbf{t}}$, then

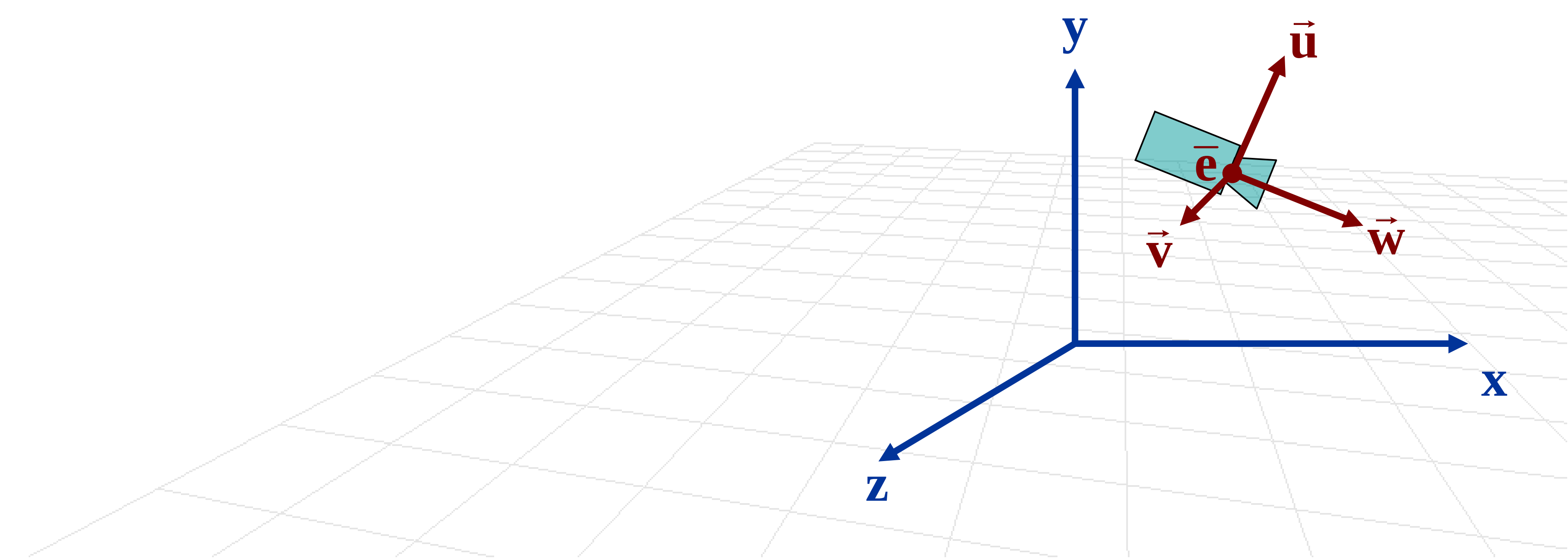
$$\vec{\mathbf{u}} = \frac{\vec{\mathbf{t}} \times \vec{\mathbf{w}}}{\left\|\vec{\mathbf{t}} \times \vec{\mathbf{w}}\right\|}$$

  - $\vec{\mathbf{v}}$ must be perpendicular to

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

Position and Orientation of Camera

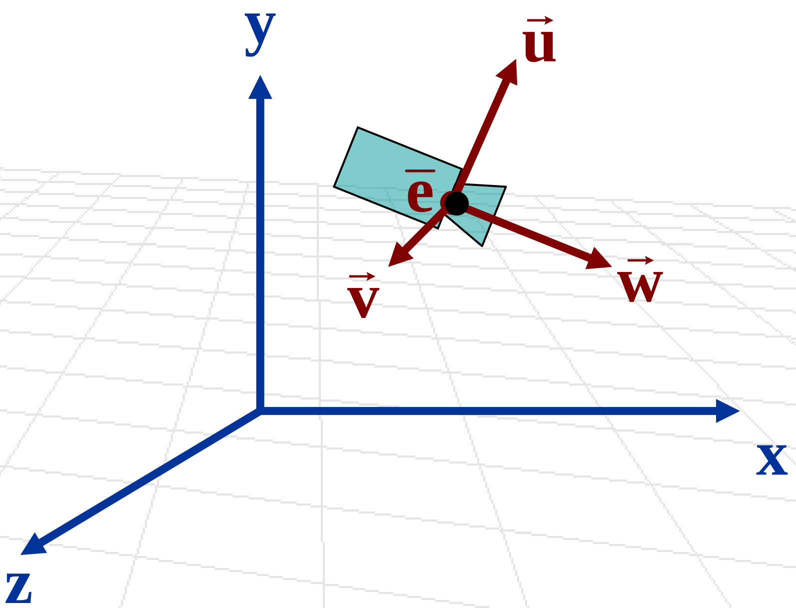
- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

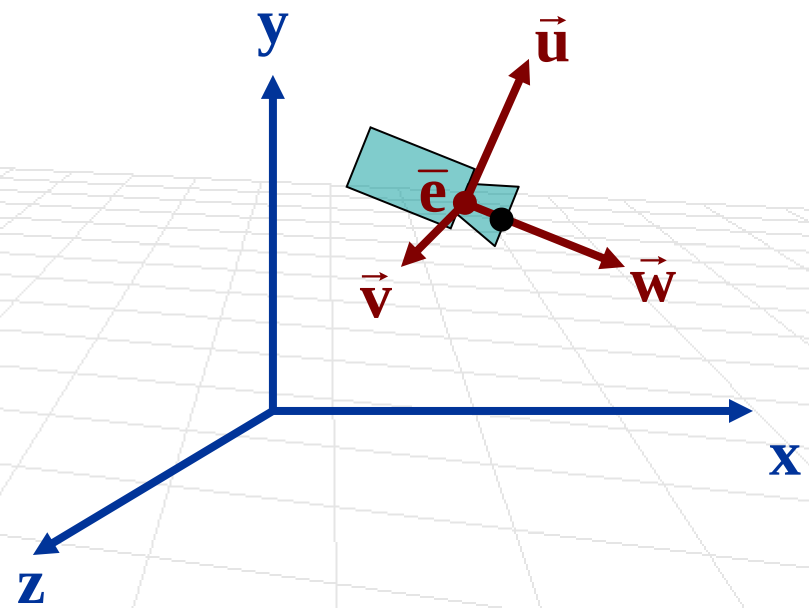
Camera Coordinates	World Coordinates
$(0,0,0)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

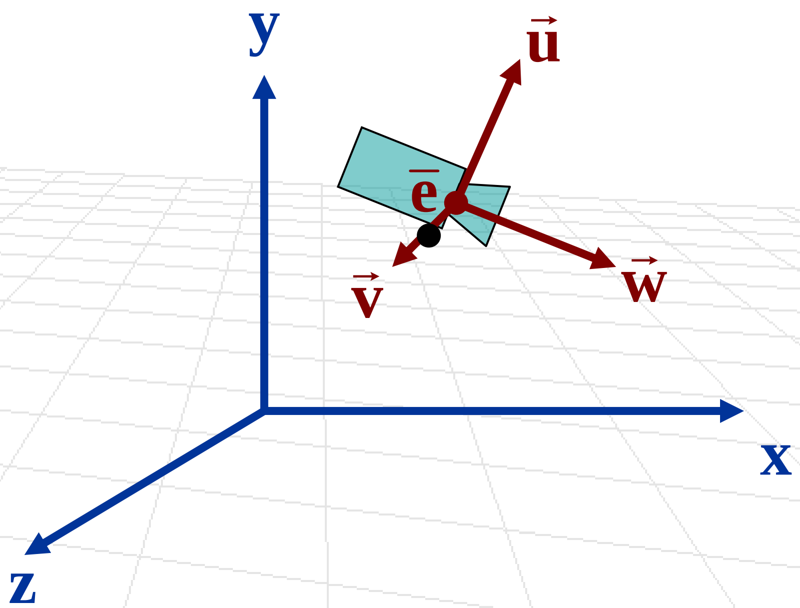
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{\mathbf{e}}$
$(0,0,\mathbf{f})$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

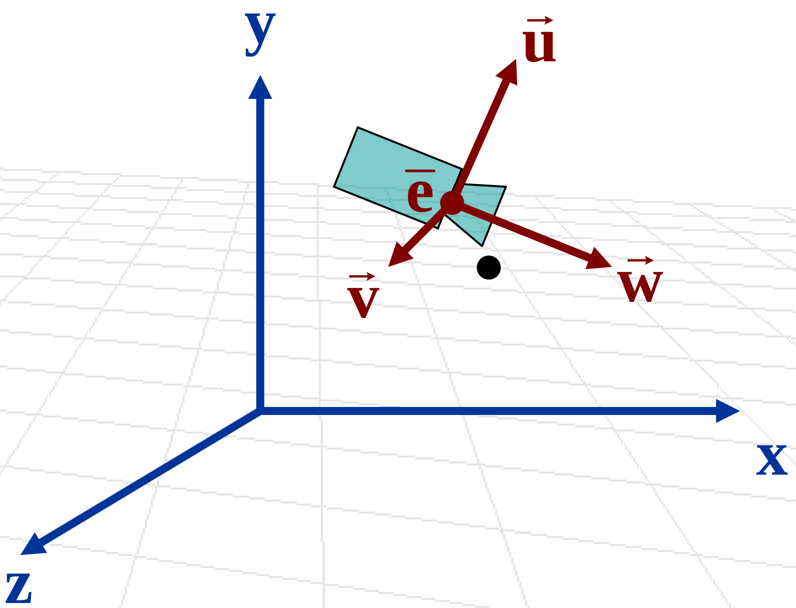
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{\mathbf{e}}$
$(0,0,\mathbf{f})$	$\bar{\mathbf{e}}+\mathbf{f}\vec{\mathbf{w}}$
$(0,1,0)$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

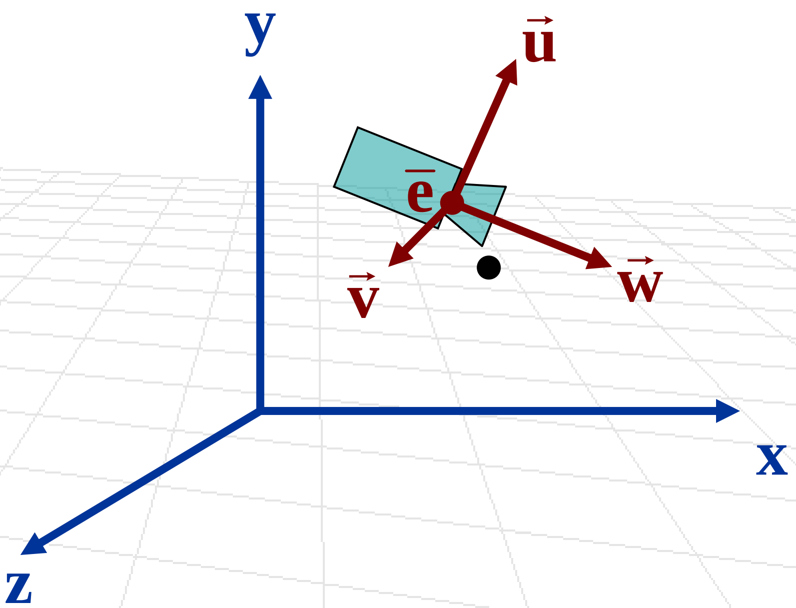
Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{\mathbf{e}}$
$(0,0,\mathbf{f})$	$\bar{\mathbf{e}}+\mathbf{f}\vec{\mathbf{w}}$
$(0,1,0)$	$\bar{\mathbf{e}}+\vec{\mathbf{v}}$
$(0,1,\mathbf{f})$	



Camera to World Transformation

- Now that we have a camera defined in world coordinate frame, how do we take a point in the camera coordinate frame and map to the world coordinate frame?
- Let's try some points

Camera Coordinates	World Coordinates
$(0,0,0)$	$\bar{\mathbf{e}}$
$(0,0,\mathbf{f})$	$\bar{\mathbf{e}} + \mathbf{f}\vec{\mathbf{w}}$
$(0,1,0)$	$\bar{\mathbf{e}} + \vec{\mathbf{v}}$
$(0,1,\mathbf{f})$	$\bar{\mathbf{e}} + \vec{\mathbf{v}} + \mathbf{f}\vec{\mathbf{w}}$



Camera to World Transformation

- It's relatively easy to show that any **point in camera coordinate frame** can be expressed **in world coordinate frame** using the following homogenized transformation:

$$\bar{\mathbf{p}}^{\mathrm{w}} = \mathbf{M}_{\mathrm{cw}} \bar{\mathbf{p}}^{\mathrm{c}}$$

$$\mathbf{M}_{\mathrm{cw}} = \begin{bmatrix} [\vec{\mathbf{u}}, \vec{\mathbf{v}}, \vec{\mathbf{w}}] & \bar{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix}$$

- See lecture notes for details

Camera to World Transformation

- It's relatively easy to show that any **point in camera coordinate frame** can be expressed **in world coordinate frame** using the following homogenized transformation:

$$\overline{\mathbf{p}}^{w} = \mathbf{M}_{cw}\overline{\mathbf{p}}^{c}$$

- Actually, what we need is the inverse:

$$\overline{\mathbf{p}}^{c} = \mathbf{M}_{wc}\overline{\mathbf{p}}^{w}$$

Inverting the Camera to World Transformation

We have:

$$\overline{\mathbf{p}}^{\mathbf{w}} = \mathbf{M}_{\mathbf{cw}}\overline{\mathbf{p}}^{\mathbf{c}} \qquad \mathbf{M}_{\mathbf{cw}} = \begin{bmatrix} \mathbf{A} & \overline{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{\mathbf{u}} & \vec{\mathbf{v}} & \vec{\mathbf{w}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

We want:

$$\overline{\mathbf{p}}^{\mathbf{c}} = \mathbf{M}_{\mathbf{wc}}\overline{\mathbf{p}}^{\mathbf{w}}$$

Inverting the Camera to World Transformation

We have:

$$\bar{\mathbf{p}}^{w} = \mathbf{M}_{cw}\bar{\mathbf{p}}^{c}$$

$$\mathbf{M}_{cw} = \begin{bmatrix} \mathbf{A} & \bar{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{\mathbf{u}} & \vec{\mathbf{v}} & \vec{\mathbf{w}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\bar{\mathbf{p}}^{w} = \mathbf{A}\bar{\mathbf{p}}^{c} + \bar{\mathbf{e}}$$

$$\bar{\mathbf{p}}^{c} = \mathbf{A}^{-1}\left(\bar{\mathbf{p}}^{w} - \bar{\mathbf{e}}\right)$$

We want:

$$\bar{\mathbf{p}}^{c} = \mathbf{M}_{wc}\bar{\mathbf{p}}^{w}$$

Inverting the Camera to World Transformation

We have:

$$\overline{\mathbf{p}}^{\mathbf{w}} = \mathbf{M}_{\mathbf{cw}}\overline{\mathbf{p}}^{\mathbf{c}} \qquad \mathbf{M}_{\mathbf{cw}} = \begin{bmatrix} \mathbf{A} & \overline{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{\mathbf{u}} & \vec{\mathbf{v}} & \vec{\mathbf{w}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\overline{\mathbf{p}}^{\mathbf{w}} = \mathbf{A}\overline{\mathbf{p}}^{\mathbf{c}} + \overline{\mathbf{e}}$$

$$\overline{\mathbf{p}}^{\mathbf{c}} = \mathbf{A}^{-1}\left(\overline{\mathbf{p}}^{\mathbf{w}} - \overline{\mathbf{e}}\right)$$

Since $\mathbf{A}$ is orthonormal (easy to check), the inverse of $\mathbf{A}$ is simply a transpose

$$\overline{\mathbf{p}}^{\mathbf{c}} = \mathbf{A}^{\mathbf{T}}\left(\overline{\mathbf{p}}^{\mathbf{w}} - \overline{\mathbf{e}}\right)$$

$$\overline{\mathbf{p}}^{\mathbf{c}} = \mathbf{A}^{\mathbf{T}}\overline{\mathbf{p}}^{\mathbf{w}} - \mathbf{A}^{\mathbf{T}}\overline{\mathbf{e}}$$

We want:

$$\overline{\mathbf{p}}^{\mathbf{c}} = \mathbf{M}_{\mathbf{wc}}\overline{\mathbf{p}}^{\mathbf{w}}$$

Inverting the Camera to World Transformation

We have:

$$\overline{\mathbf{p}}^{w} = \mathbf{M}_{cw}\overline{\mathbf{p}}^{c} \qquad \mathbf{M}_{cw} = \begin{bmatrix} \mathbf{A} & \overline{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \vec{\mathbf{u}} & \vec{\mathbf{v}} & \vec{\mathbf{w}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix}$$

$$\overline{\mathbf{p}}^{w} = \mathbf{A}\overline{\mathbf{p}}^{c} + \overline{\mathbf{e}}$$

$$\overline{\mathbf{p}}^{c} = \mathbf{A}^{-1}\left(\overline{\mathbf{p}}^{w} - \overline{\mathbf{e}}\right)$$

Since $\mathbf{A}$ is orthonormal (easy to check), the inverse of $\mathbf{A}$ is simply a transpose

$$\overline{\mathbf{p}}^{c} = \mathbf{A}^{T}\left(\overline{\mathbf{p}}^{w} - \overline{\mathbf{e}}\right)$$

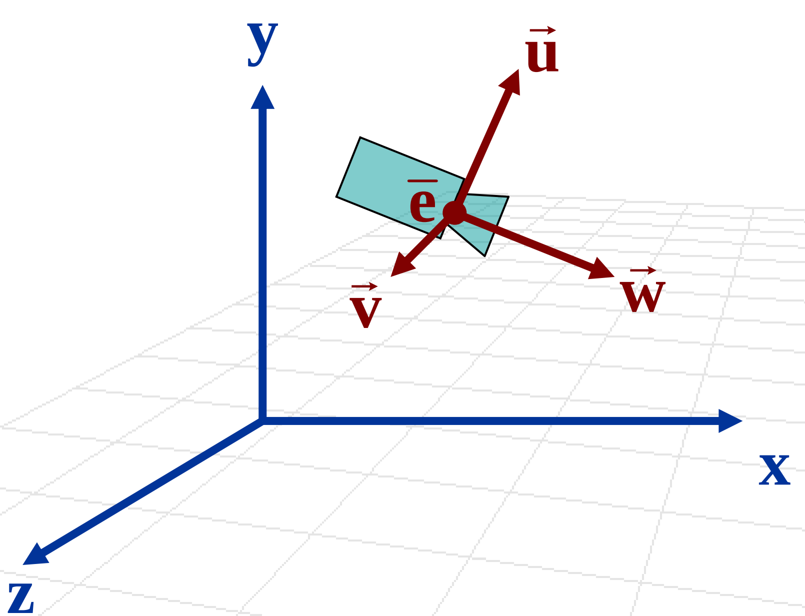
$$\overline{\mathbf{p}}^{c} = \mathbf{A}^{T}\overline{\mathbf{p}}^{w} - \mathbf{A}^{T}\overline{\mathbf{e}}$$

We want:

$$\overline{\mathbf{p}}^{c} = \mathbf{M}_{wc}\overline{\mathbf{p}}^{w} \qquad \mathbf{M}_{wc} = \begin{bmatrix} \mathbf{A}^{T} & -\mathbf{A}^{T}\overline{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix} \qquad \mathbf{A}^{T} = \begin{bmatrix} \leftarrow & \vec{\mathbf{u}} & \rightarrow \\ \leftarrow & \vec{\mathbf{v}} & \rightarrow \\ \leftarrow & \vec{\mathbf{w}} & \rightarrow \end{bmatrix}$$

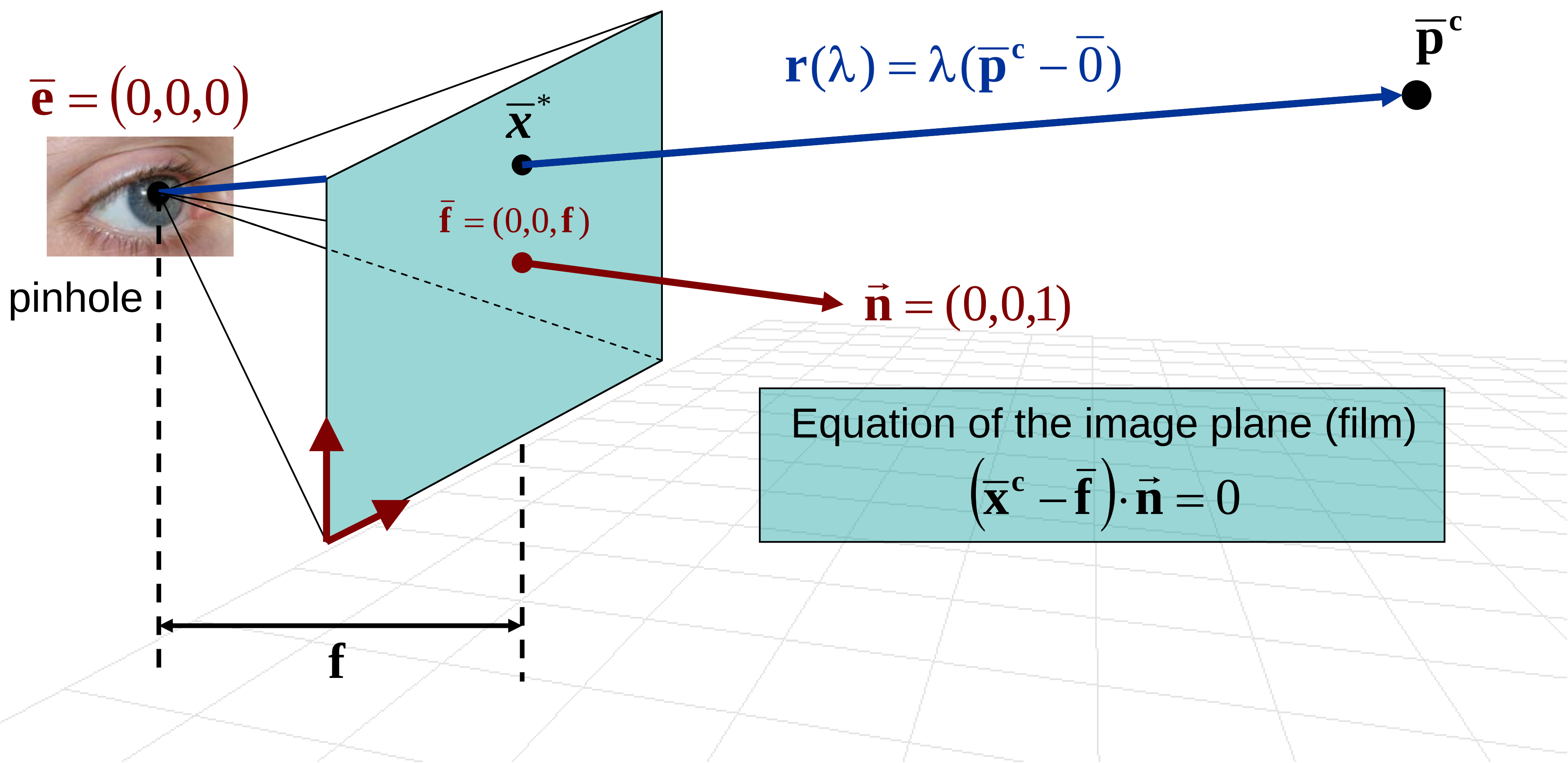
Perspective Projection (Again)

- Earlier we derive perspective projection using similar triangles
- Now, we will go through an exercise of doing it algebraically (it's a good exercise)



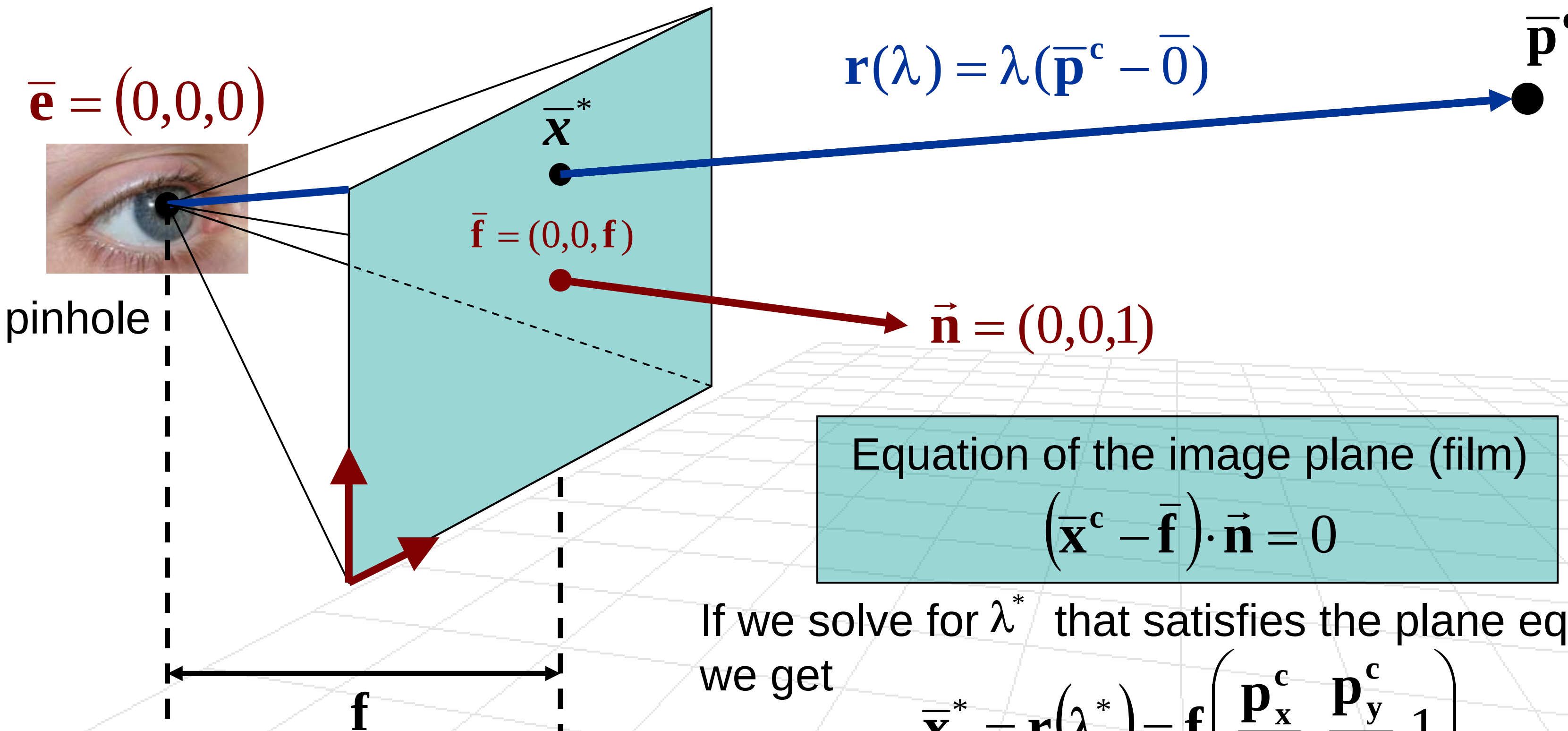
Perspective Projection

Lets consider everything in the camera coordinate frame



Perspective Projection

Lets consider everything in the camera coordinate frame



Equation of the image plane (film)

$$\left(\bar{\mathbf{x}}^c - \bar{\mathbf{f}}\right) \cdot \vec{\mathbf{n}} = 0$$

If we solve for λ^* that satisfies the plane equation we get

$$\bar{\mathbf{x}}^* = \mathbf{r}\left(\lambda^*\right) = \mathbf{f}\left(\frac{\mathbf{p}_x^c}{\mathbf{p}_z^c}, \frac{\mathbf{p}_y^c}{\mathbf{p}_z^c}, 1\right)$$

Perspective Projection

- The mapping from a point $\overline{\mathbf{p}}^{\mathbf{c}}$ in camera coordinates to point $\left(\mathbf{x}^*, \mathbf{y}^*, 1\right)$ in the image plane, is what we will call the **perspective projection**

$$\overline{\mathbf{x}}^* = \mathbf{r}\left(\lambda^*\right) = \mathbf{f}\left(\frac{\mathbf{p}_{\mathbf{x}}^{\mathbf{c}}}{\mathbf{p}_{\mathbf{z}}^{\mathbf{c}}}, \frac{\mathbf{p}_{\mathbf{y}}^{\mathbf{c}}}{\mathbf{p}_{\mathbf{z}}^{\mathbf{c}}}, 1\right)$$

Just a scaling factor, we can ignore

Homogeneous Perspective

- The mapping of point $\overline{\mathbf{p}}^{c} = \left(\mathbf{p}_{x}^{c}, \mathbf{p}_{y}^{c}, \mathbf{p}_{z}^{c}\right)$ to $\overline{\mathbf{x}}^{*} = \left(\mathbf{x}^{*}, \mathbf{y}^{*}, 1\right)$ is the form of scaling transformation, but since it depends on the depth of the point $\mathbf{p}_{z}^{c}$, it is not linear (remember the tapering example from last week)

- It would be very useful if we can express this non-linear transformation as a linear transformation (matrix). Why?

Homogeneous Perspective

- The mapping of point $\overline{\mathbf{p}}^{\mathbf{c}} = \left(\mathbf{p}_{\mathbf{x}}^{\mathbf{c}}, \mathbf{p}_{\mathbf{y}}^{\mathbf{c}}, \mathbf{p}_{\mathbf{z}}^{\mathbf{c}}\right)$ to $\overline{\mathbf{x}}^{*} = \left(\mathbf{x}^{*}, \mathbf{y}^{*}, 1\right)$ is the form of scaling transformation, but since it depends on the depth of the point $\mathbf{p}_{\mathbf{z}}^{\mathbf{c}}$, it is not linear (remember the tapering example from last week)

- It would be very useful if we can express this non-linear transformation as a linear transformation (matrix). Why?

$$\overline{\mathbf{x}}^{*} = \mathbf{M}_{\mathbf{p}} \mathbf{M}_{\mathbf{wc}} \overline{\mathbf{p}}^{\mathbf{w}}$$

Homogeneous Perspective

- We can express it a a linear transformation in homogeneous coordinates (this is one of the benefits of using homogeneous coordinates!)
- Here's the transformation that does what we want:

$$\mathbf{M_p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix}$$

- Let's prove this is true

Homogeneous Perspective

Claim:

$$\begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{x}^* \\ \mathbf{y}^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{p}_x^c / \mathbf{p}_z^c \\ \mathbf{p}_y^c / \mathbf{p}_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \mathbf{M}_p \overline{\mathbf{p}}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ 1 \end{pmatrix}$$

Homogeneous Perspective

Claim:

$$\begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{x}^* \\ \mathbf{y}^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{p}_x^c / \mathbf{p}_z^c \\ \mathbf{p}_y^c / \mathbf{p}_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \mathbf{M}_p \overline{\mathbf{p}}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ 1 \end{pmatrix}$$

Proof:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ \mathbf{p}_z^c / \mathbf{f} \end{bmatrix}$$

Point in homogeneous coordinates can be scaled arbitrarily

Homogeneous Perspective

Claim:

$$\begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{x}^* \\ \mathbf{y}^* \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{f}\begin{pmatrix} \mathbf{p}_x^c / \mathbf{p}_z^c \\ \mathbf{p}_y^c / \mathbf{p}_z^c \\ 1 \end{pmatrix} \\ 1 \end{bmatrix} = \mathbf{M}_p \overline{\mathbf{p}}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ 1 \end{pmatrix}$$

Proof:

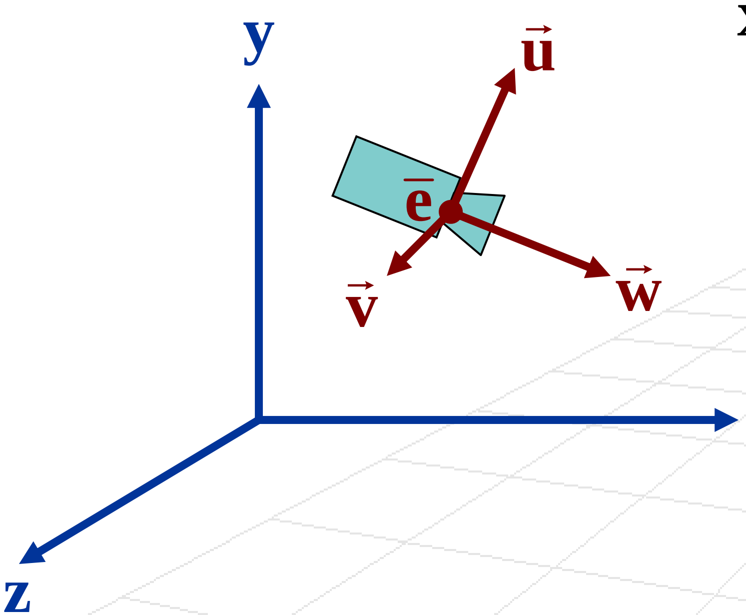
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x^c \\ \mathbf{p}_y^c \\ \mathbf{p}_z^c \\ \mathbf{p}_z^c / \mathbf{f} \end{bmatrix} = \cancel{\mathbf{p}_z^c / \mathbf{f}} \begin{bmatrix} \mathbf{f}\mathbf{p}_x^c / \mathbf{p}_z^c \\ \mathbf{f}\mathbf{p}_y^c / \mathbf{p}_z^c \\ \mathbf{f} \\ 1 \end{bmatrix}$$

Point in homogeneous coordinates can be scaled arbitrarily

Putting together a camera model

Projecting a world point to image (film) plane

$$\overline{\mathbf{x}}^* = \mathbf{M}_p \mathbf{M}_{wc} \overline{\mathbf{p}}^w$$

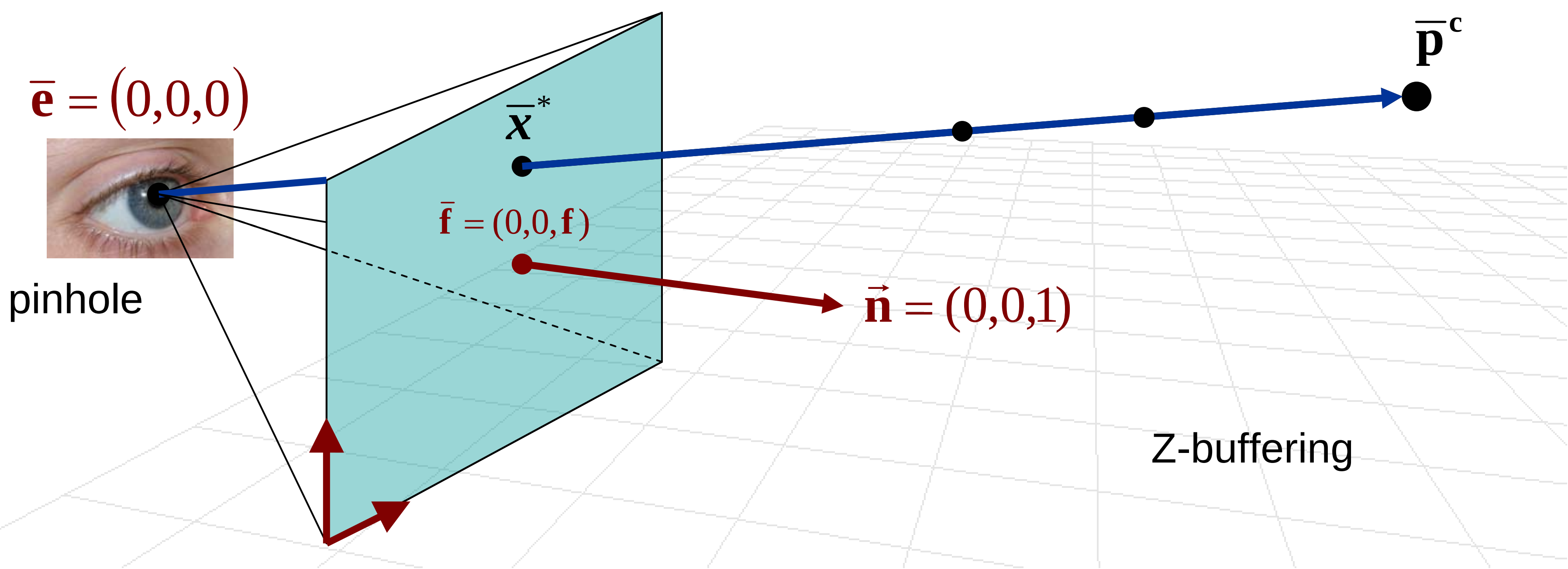


$$\overline{\mathbf{x}}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{A}^T & -\mathbf{A}^T \overline{\mathbf{e}} \\ [0,0,0] & 1 \end{bmatrix} \overline{\mathbf{p}}^w$$

where $\mathbf{A}^T = \begin{bmatrix} \leftarrow & \vec{\mathbf{u}} & \rightarrow \\ \leftarrow & \vec{\mathbf{v}} & \rightarrow \\ \leftarrow & \vec{\mathbf{w}} & \rightarrow \end{bmatrix}$

Pseudodepth

- We would like to change the projection transform so that z-component of the projection gives us useful information (not just a constant **f**)
- We want it to encode something about depth of a point. Why?



Pseudodepth

- Standard homogeneous perspective projection

$$\mathbf{M_p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix}$$

- Pseudodepth projection matrix

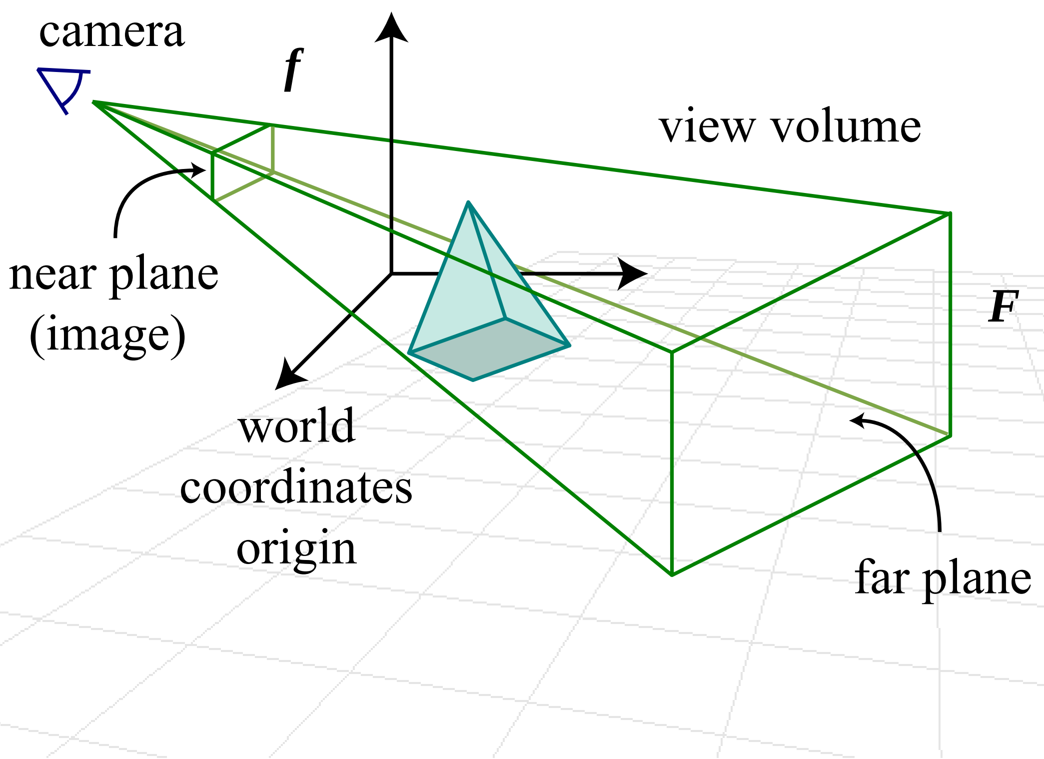
$$\mathbf{M_p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mathbf{a} & \mathbf{b} \\ 0 & 0 & 1/\mathbf{f} & 0 \end{bmatrix}$$

$$\mathbf{z}^* = \frac{\mathbf{f}}{\mathbf{p}_\mathbf{z}^\mathbf{c}}\left(\mathbf{a}\mathbf{p}_\mathbf{z}^\mathbf{c} + \mathbf{b}\right)$$

Pseudodepth

How do we pick **a** and **b**?

$$\mathbf{z}^* = \frac{\mathbf{f}}{\mathbf{p}_z^c}\left(\mathbf{a}\mathbf{p}_z^c + \mathbf{b}\right)$$

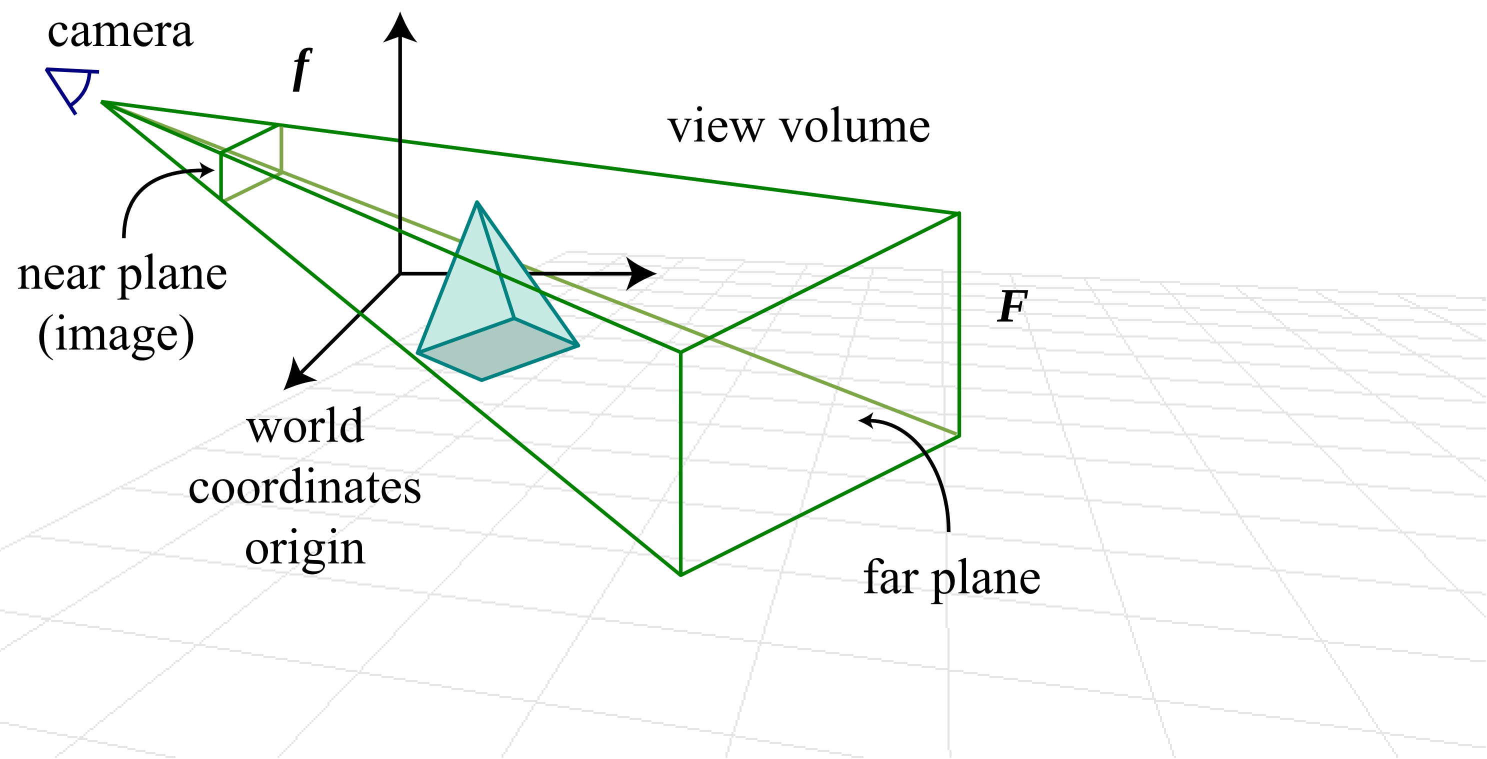


Pseudodepth

How do we pick **a** and **b**?

$$z^* = \frac{f}{p_z^c}\left(a p_z^c + b\right)$$

$$z^* = \begin{cases} -1 & \textbf{when} \quad p_z^c = f \\ 1 & \textbf{when} \quad p_z^c = F \end{cases}$$



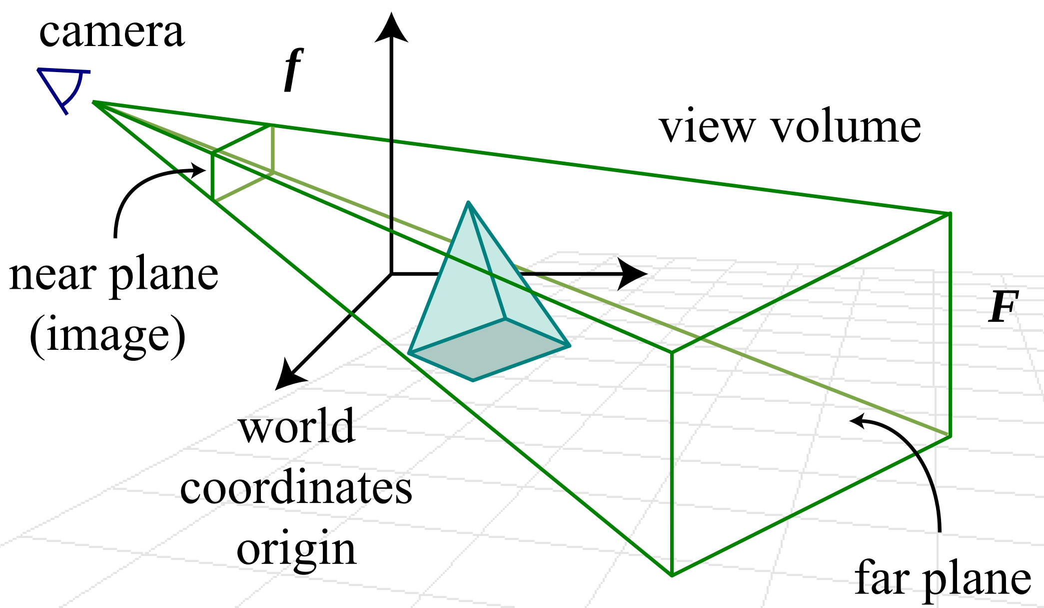
Pseudodepth

How do we pick **a** and **b**?

$$\mathbf{z}^* = \frac{\mathbf{f}}{\mathbf{p}_z^c}\left(\mathbf{a}\mathbf{p}_z^c + \mathbf{b}\right)$$

$$-1 = \mathbf{af} + \mathbf{b}$$

$$1 = \mathbf{af} + \mathbf{b}\frac{\mathbf{f}}{\mathbf{F}}$$



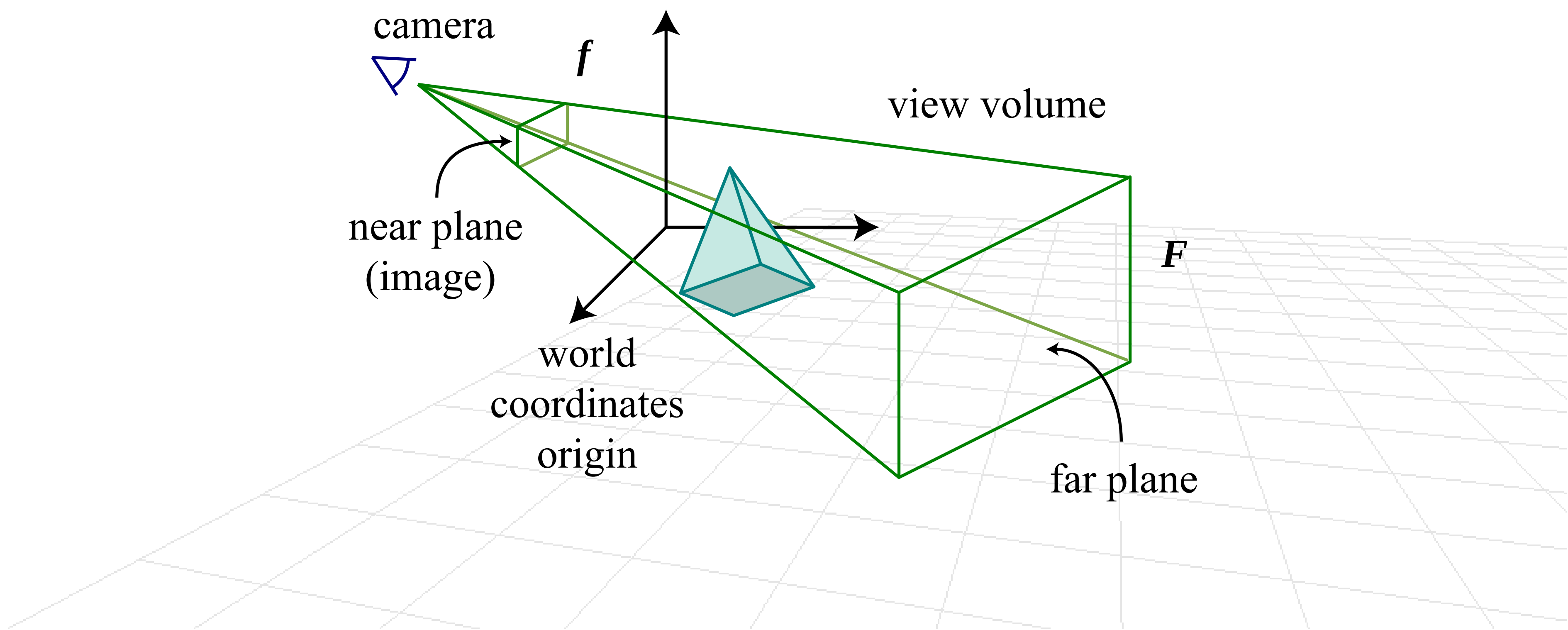
Pseudodepth

How do we pick **a** and **b**?

$$\mathbf{z}^* = \frac{\mathbf{f}}{\mathbf{p}_z^c}\left(\mathbf{a}\mathbf{p}_z^c + \mathbf{b}\right)$$

$$\mathbf{b} = \frac{2\mathbf{F}}{\mathbf{f} - \mathbf{F}}$$

$$\mathbf{a} = -\frac{1}{\mathbf{f}}\left(\frac{\mathbf{f} + \mathbf{F}}{\mathbf{f} - \mathbf{F}}\right)$$



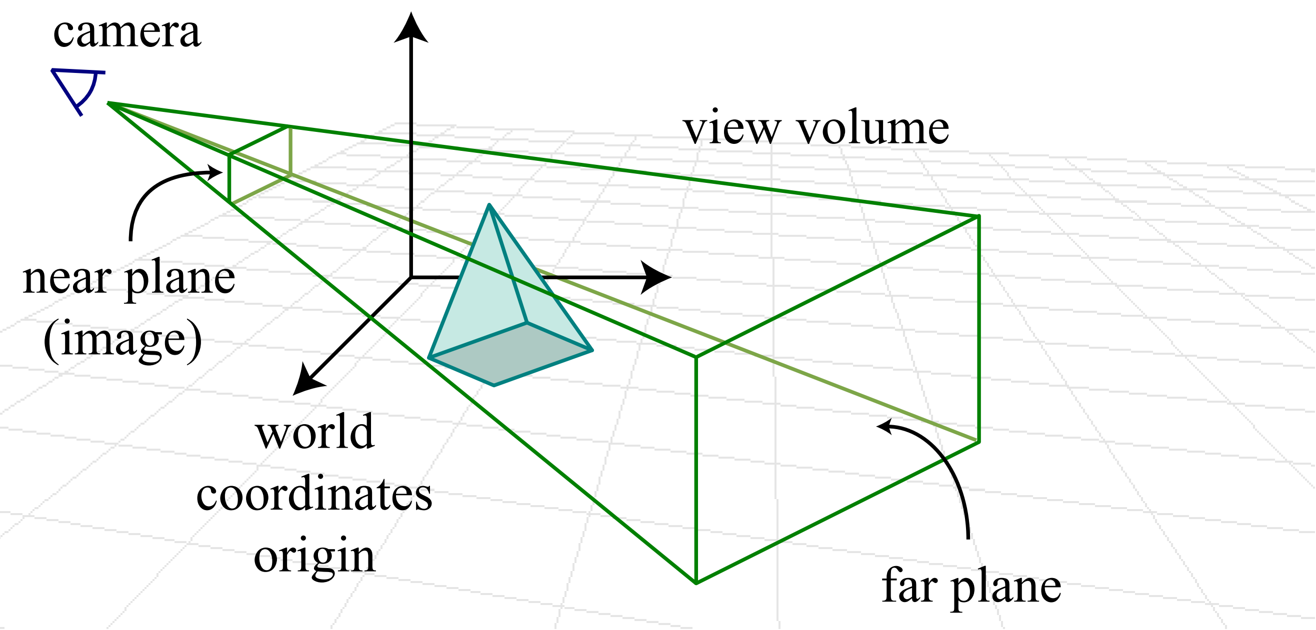
Pseudodepth

Standard homogeneous perspective with pseudodepth

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{2\boldsymbol{F}}{\boldsymbol{f}-\boldsymbol{F}} & -\dfrac{1}{\boldsymbol{f}}\left(\dfrac{\boldsymbol{f}+\boldsymbol{F}}{\boldsymbol{f}-\boldsymbol{F}}\right) \\ 0 & 0 & 1/\boldsymbol{f} & 0 \end{bmatrix}$$

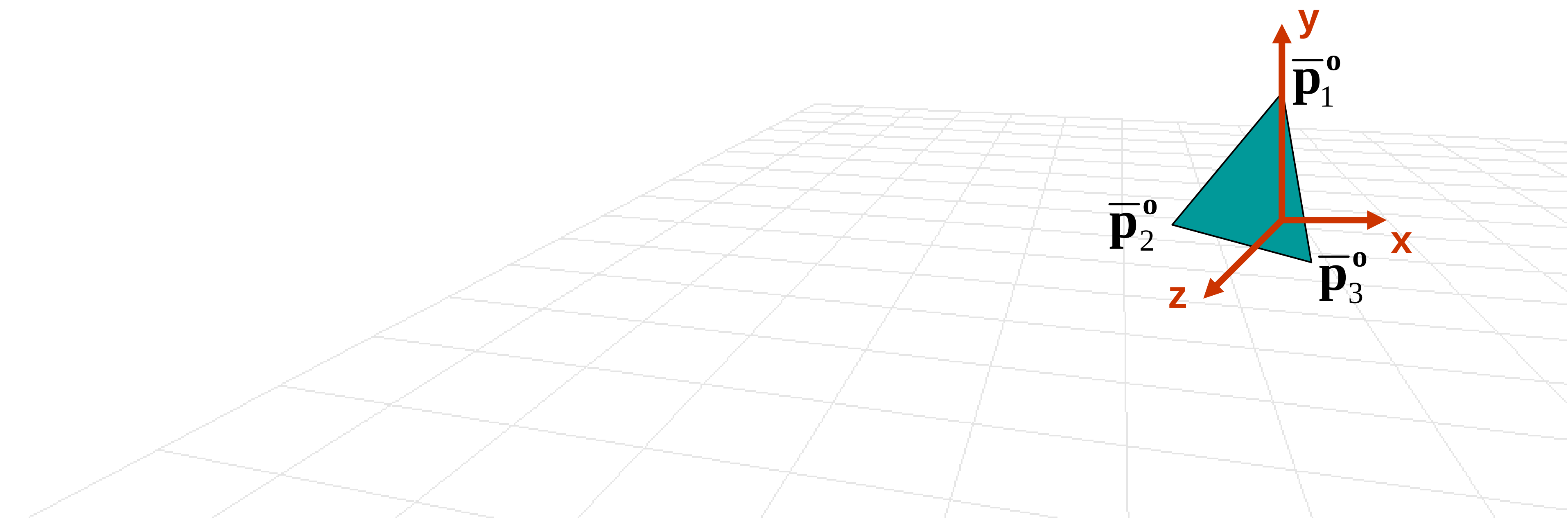
Near and Far Planes

- Anything closer than **near plane** is considered to be behind the camera and does not need to be rendered
- Anything further away from the camera than **far plane** is too far to be visible, so it is not rendered
- **Practical issue:** far plane too far away will lead to imprecision in the computed pseudodeph and hence rendering



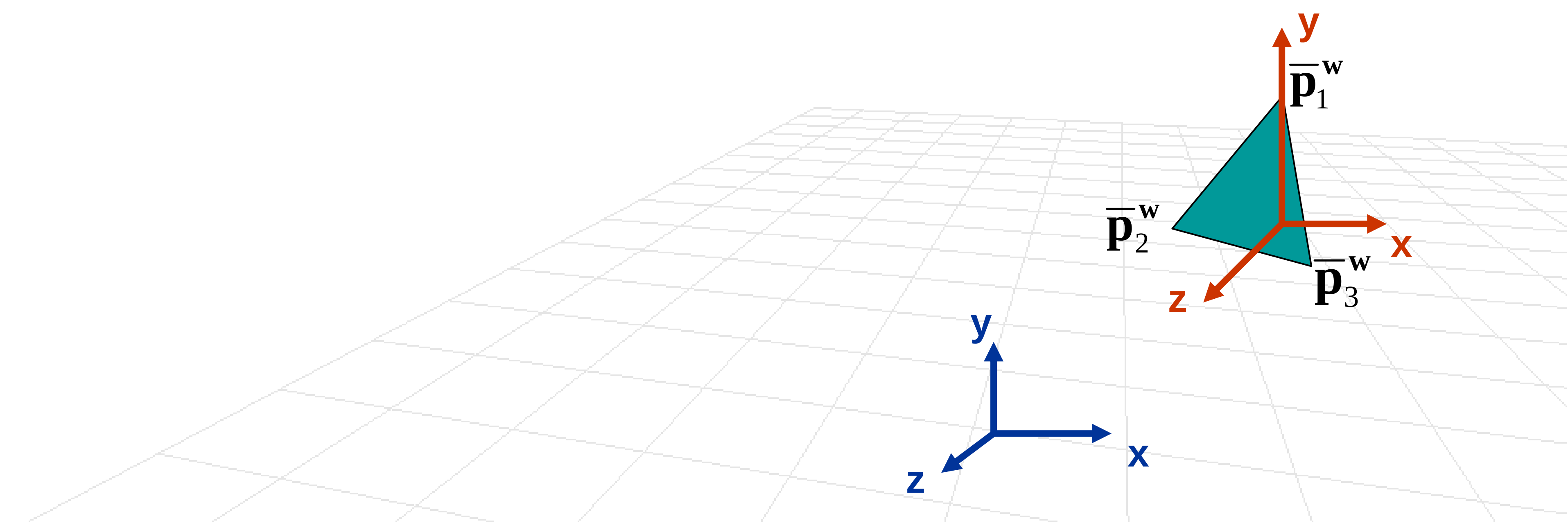
Projecting Triangle

- Lets review steps in the rendering hierarchy
  - Triangle is given in the object-based coordinate frame as three vertices



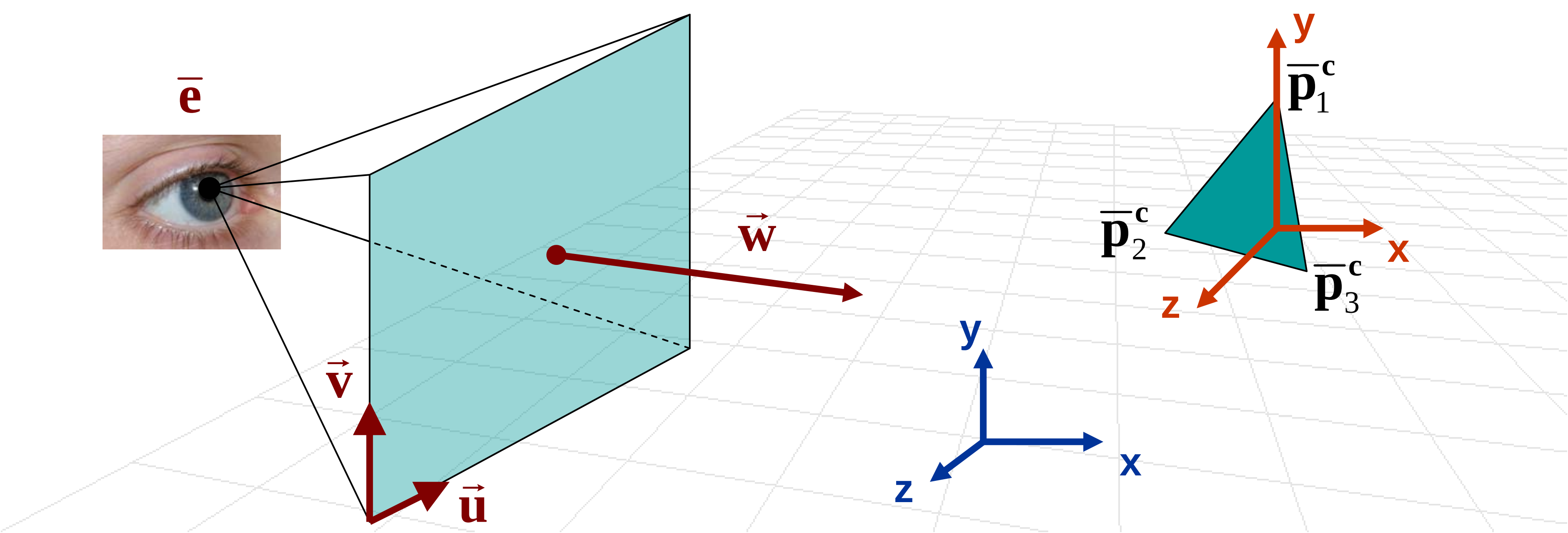
Projecting Triangle

- Lets review steps in the rendering hierarchy
  - Triangle is given in the object-based coordinate frame as three vertices
  - Transform to world coordinated $\overline{\mathbf{p}}_{\mathbf{i}}^{\mathbf{w}} = \mathbf{M}_{\mathbf{ow}} \overline{\mathbf{p}}_{\mathbf{i}}^{\mathbf{o}}$



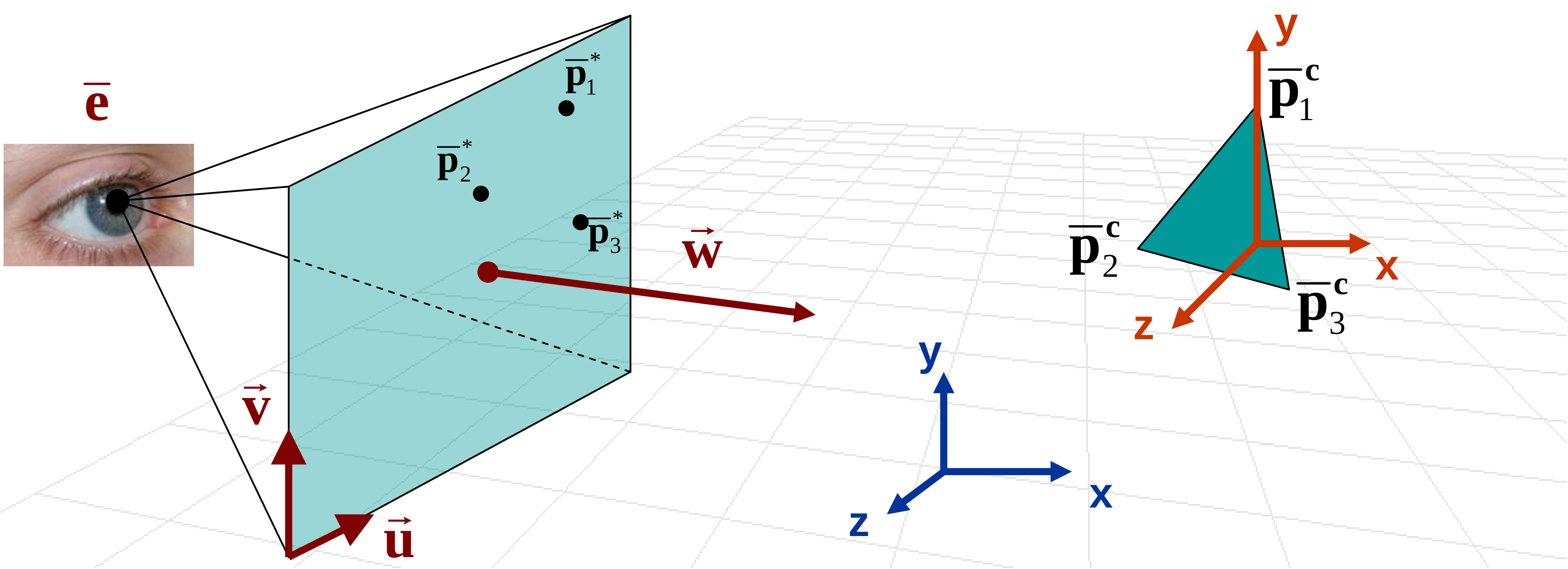
Projecting Triangle

- Lets review steps in the rendering hierarchy
  - Triangle is given in the object-based coordinate frame as three vertices
  - Transform to world coordinated $\bar{\mathbf{p}}_i^w = \mathbf{M}_{ow}\bar{\mathbf{p}}_i^o$
  - Transform from world to camera coordinates $\bar{\mathbf{p}}_i^c = \mathbf{M}_{wc}\bar{\mathbf{p}}_i^w$



Projecting Triangle

- Lets review steps in the rendering hierarchy
  - Triangle is given in the object-based coordinate frame as three vertices
  - Transform to world coordinated $\overline{\mathbf{p}}_i^w = \mathbf{M}_{ow}\overline{\mathbf{p}}_i^o$
  - Transform from world to camera coordinates $\overline{\mathbf{p}}_i^c = \mathbf{M}_{wc}\overline{\mathbf{p}}_i^w$
  - Apply homogeneous perspective $\overline{\mathbf{p}}_i^* = \mathbf{M}_p\overline{\mathbf{p}}_i^c$
    - Divide by last component



Projecting Triangle

- Lets review steps in the rendering hierarchy
  - Triangle is given in the object-based coordinate frame as three vertices
  - Transform to world coordinated $\bar{\mathbf{p}}_i^w = \mathbf{M}_{ow}\bar{\mathbf{p}}_i^o$
  - Transform from world to camera coordinates $\bar{\mathbf{p}}_i^c = \mathbf{M}_{wc}\bar{\mathbf{p}}_i^w$
  - Apply homogeneous perspective $\bar{\mathbf{p}}_i^* = \mathbf{M}_p\bar{\mathbf{p}}_i^c$
    - Divide by last component