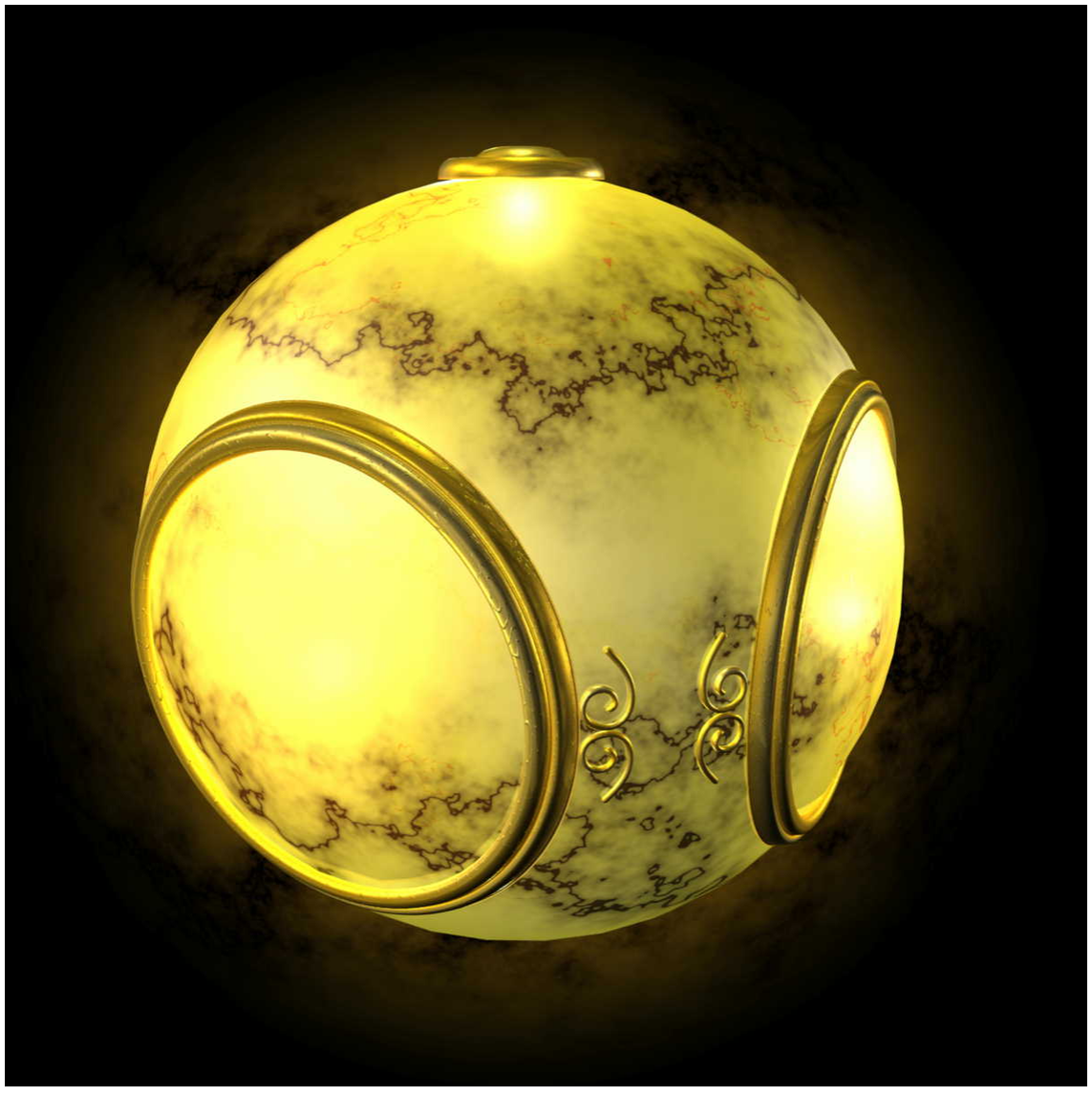
Viewing

CS452/CS552/EE465/EE505: Computer Graphics

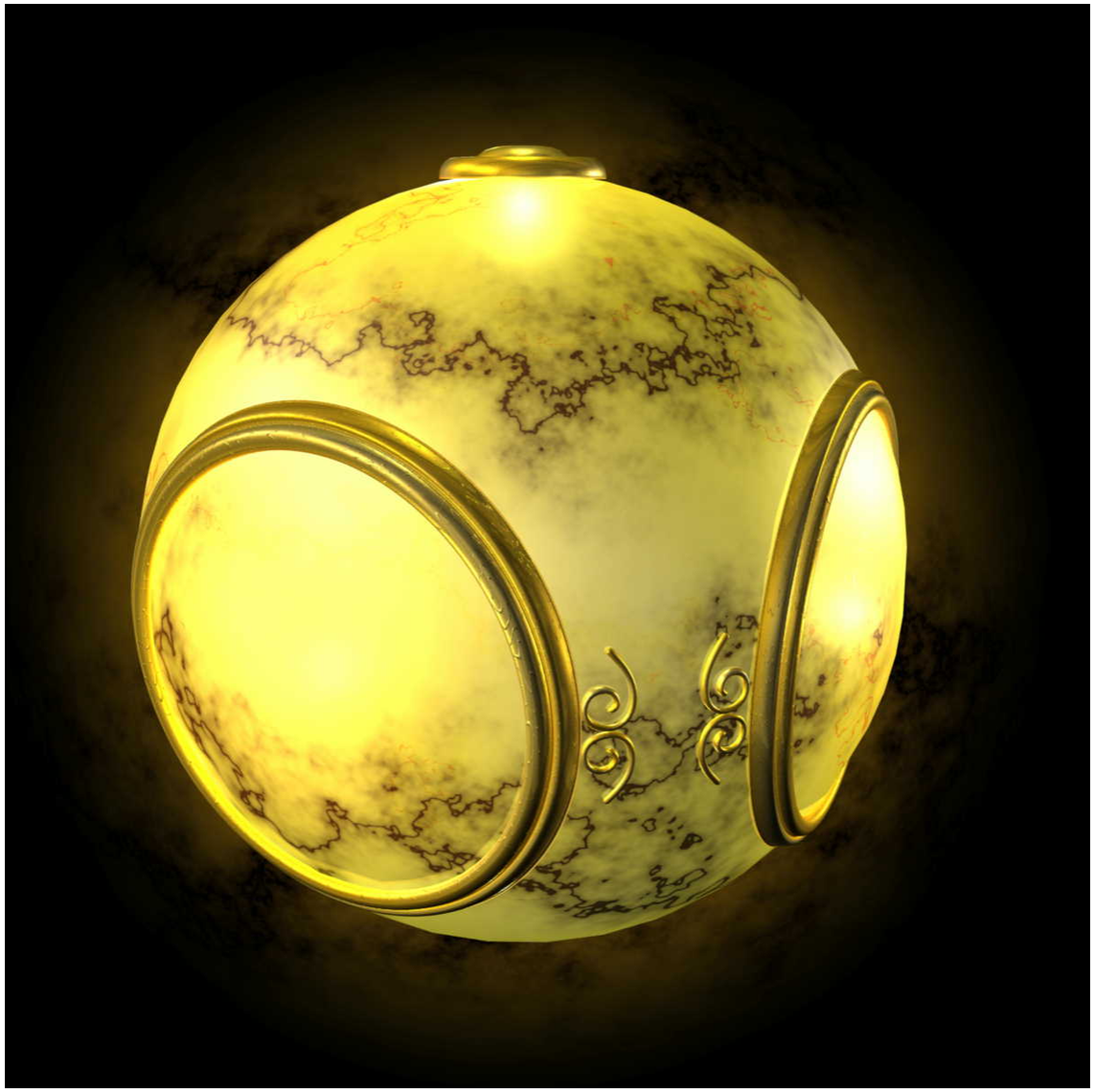
Chapter 5 of Angel & Shreiner

Where we are heading...

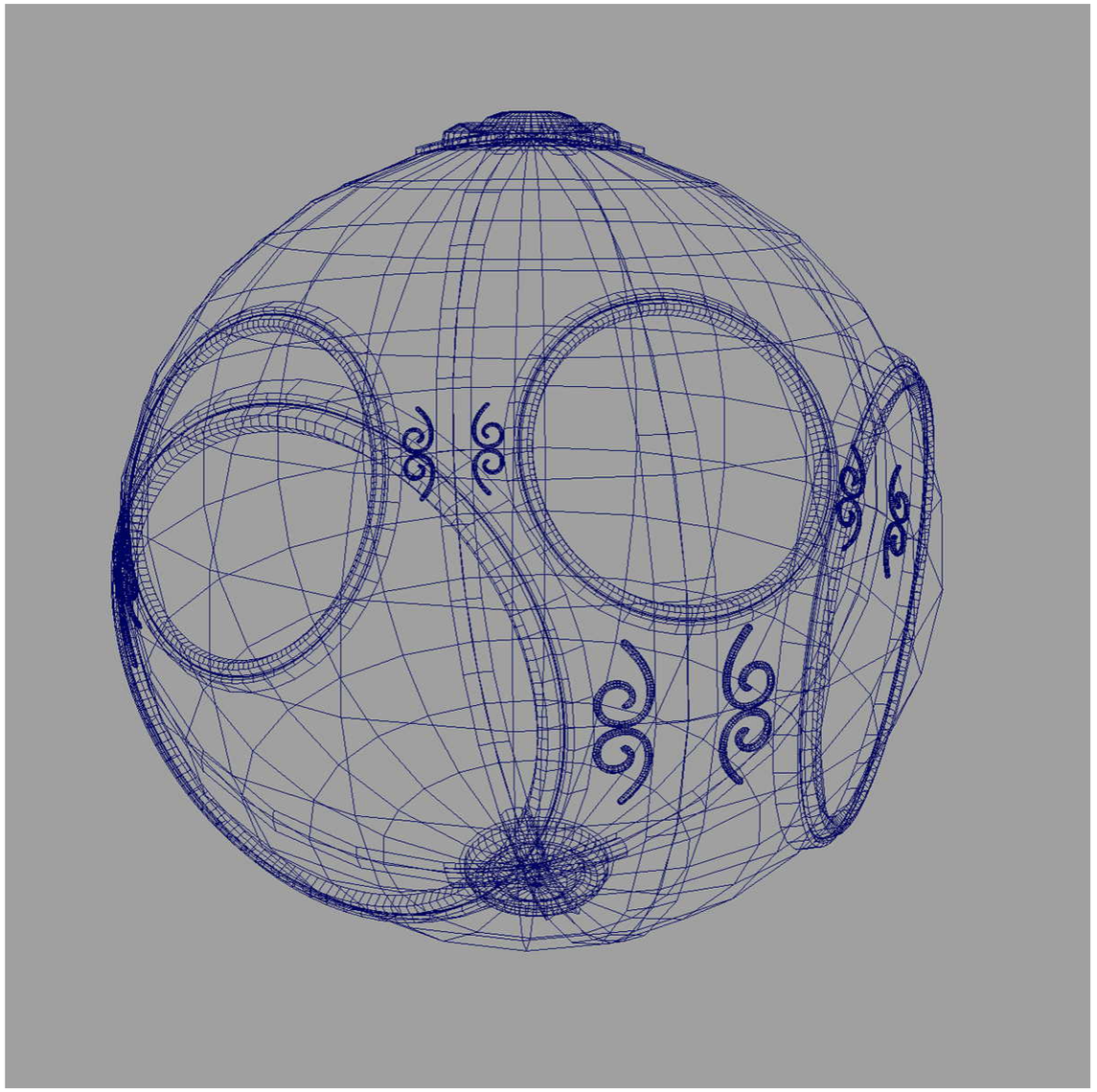


When I think of computer graphics, I think of this.

Where we are heading...

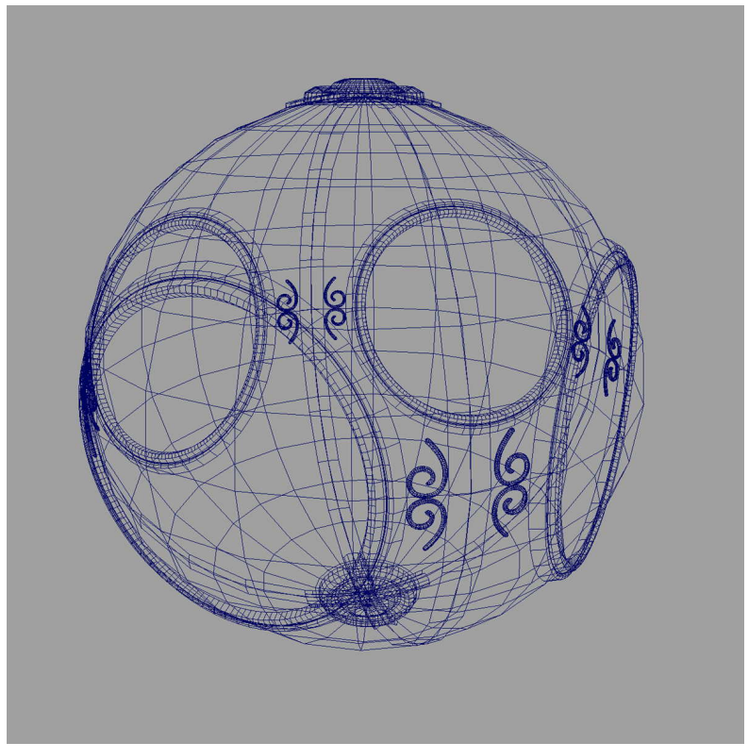


When I think of computer graphics, I think of this.

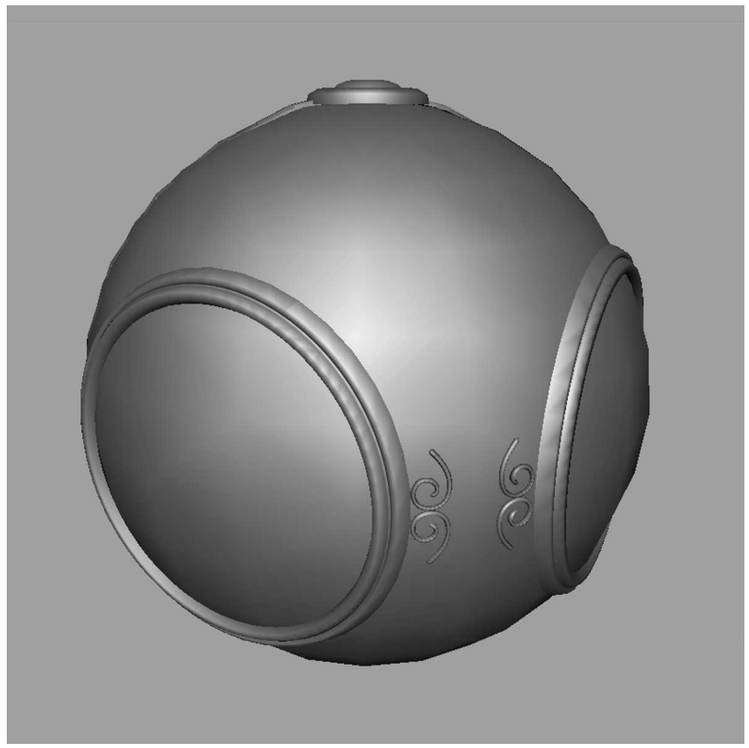


Up until now, here is where we have gotten to (3D shapes).

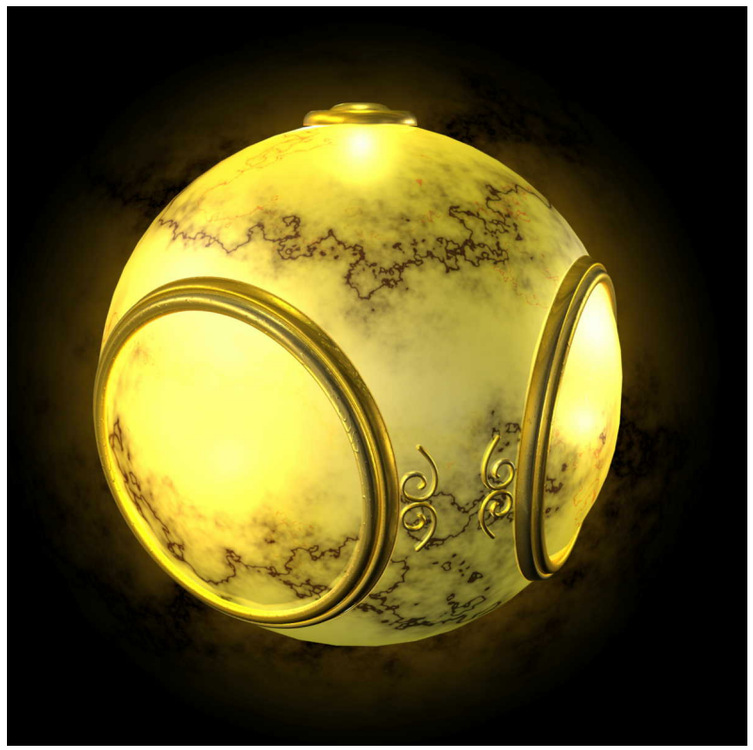
Where we are heading...



Camera **Viewing**
3D Shape



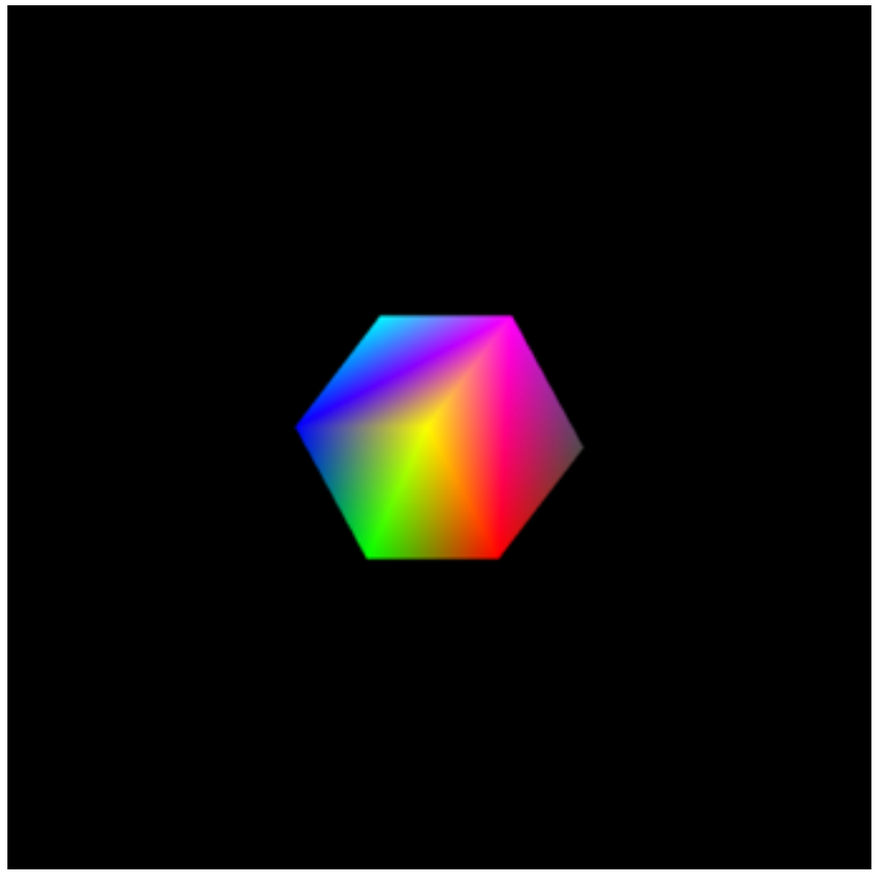
Lighting



Texture

Back in Transformations in 3D...

We transformed our object,
but we did not change our camera

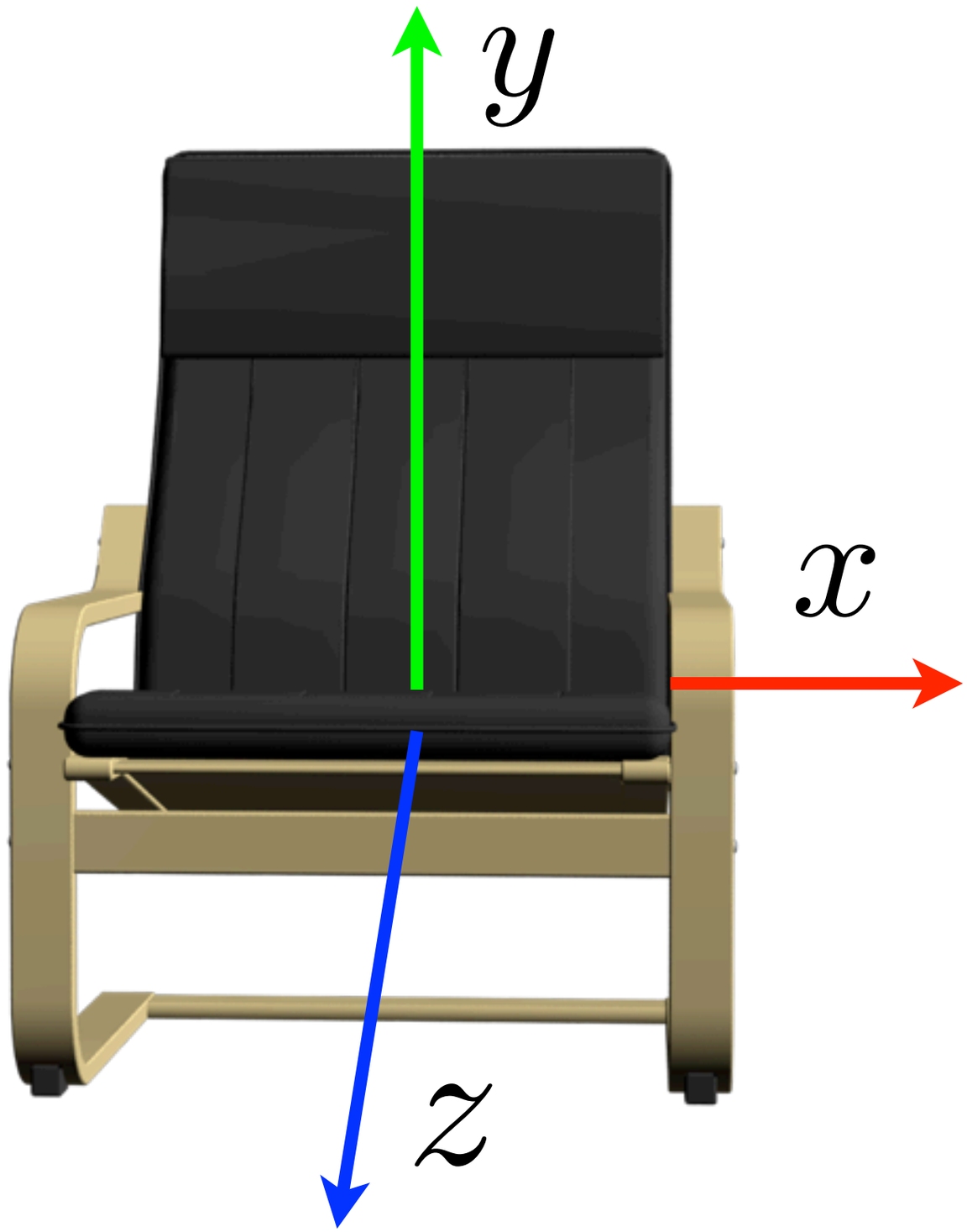


$$\mathbf{v}_f = \mathbf{M}_y \mathbf{M}_x \mathbf{v}_i$$

```
gl_Position=M_y * M_x * vertexPosition;
```

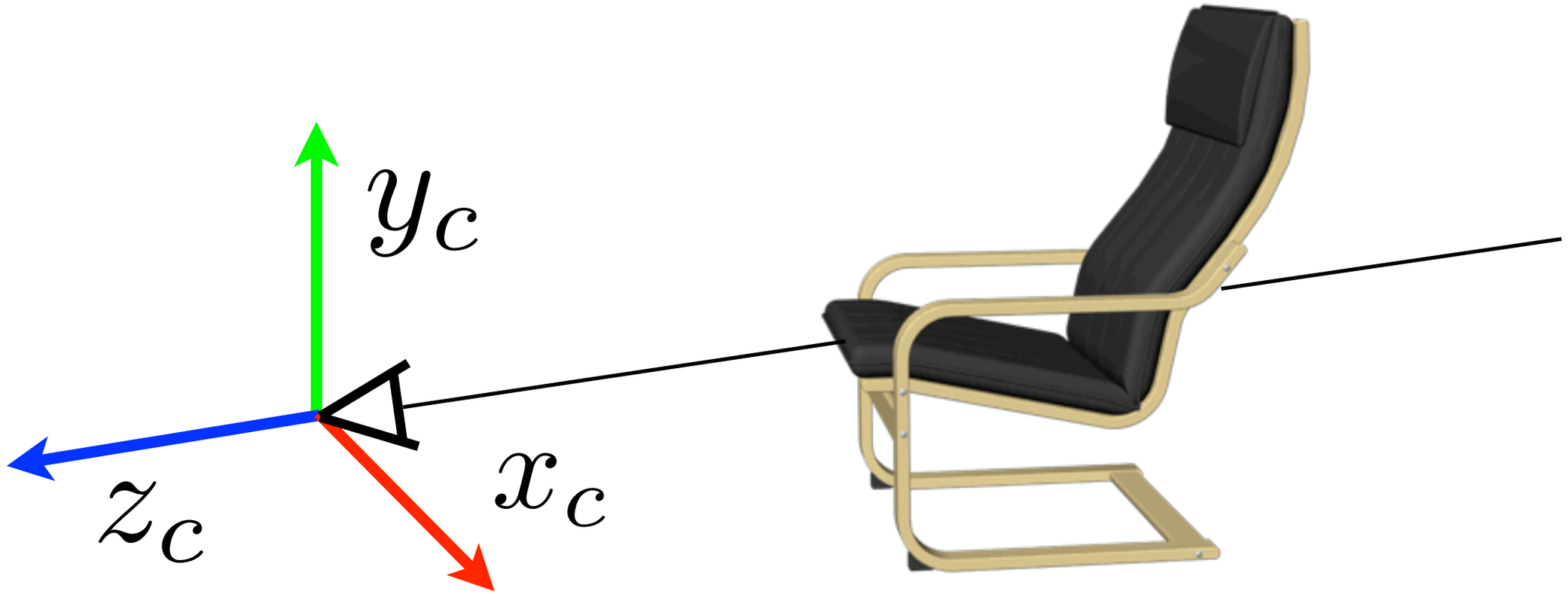
We can think of three coordinate systems

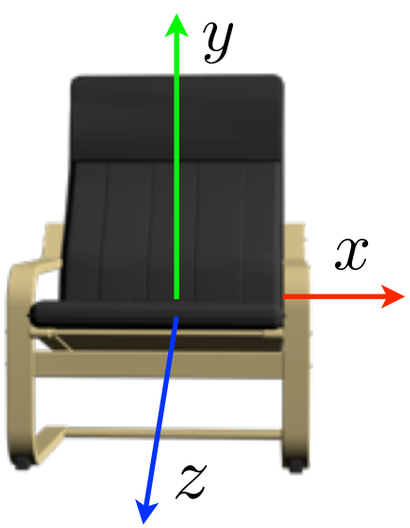
Object Coordinate System



We can think of three coordinate systems

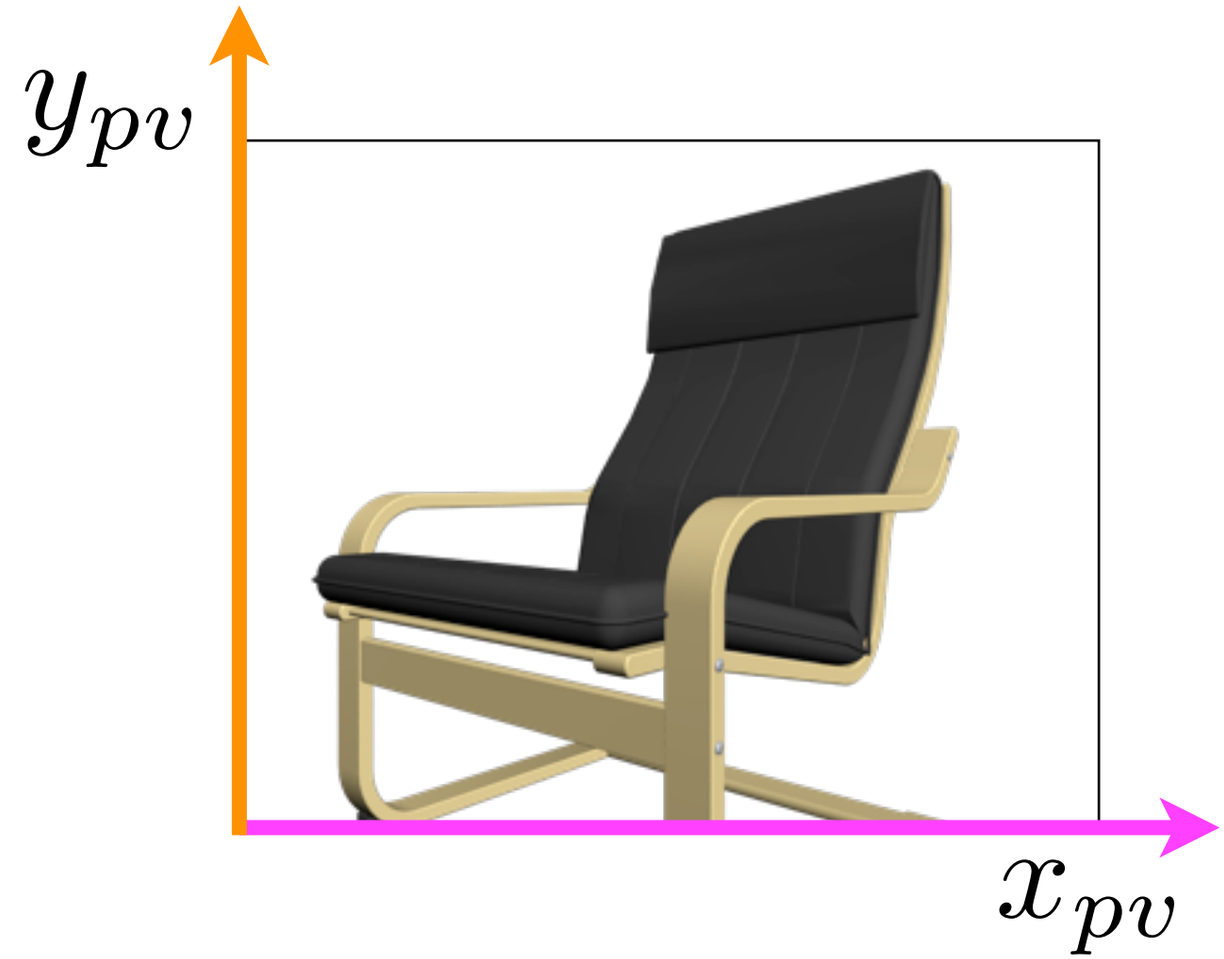
Camera Coordinate System

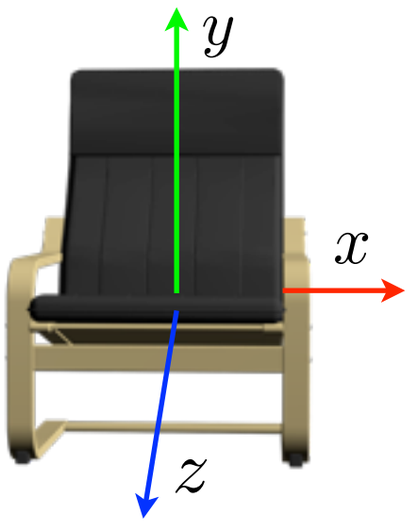


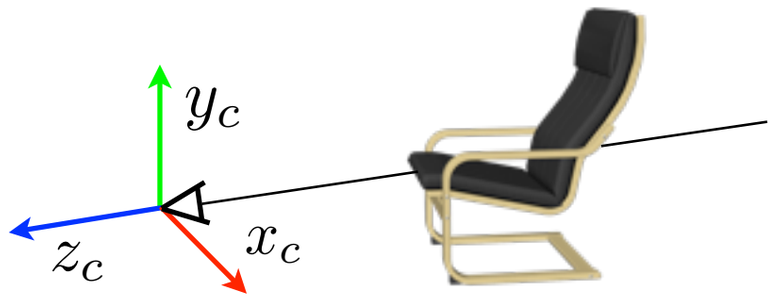


We can think of three coordinate systems

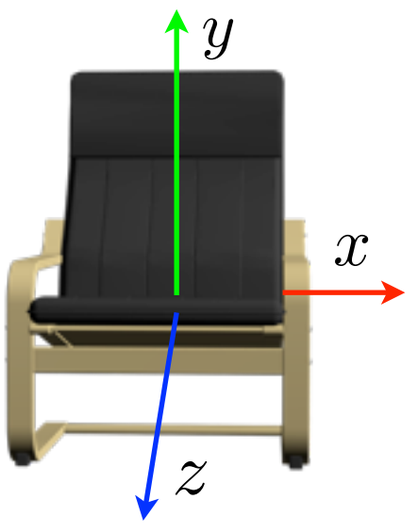
Image (or Clip) Coordinate System



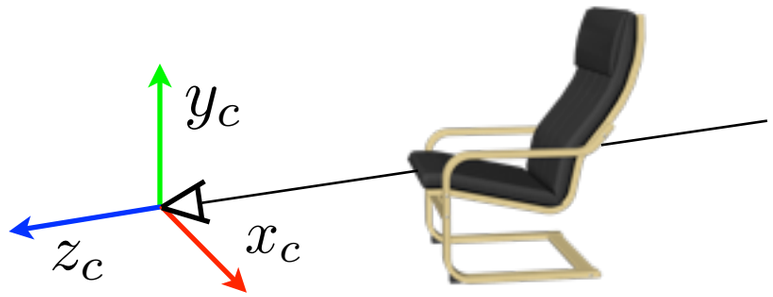




We can think of three coordinate systems



Object Coordinates



Camera Coordinates

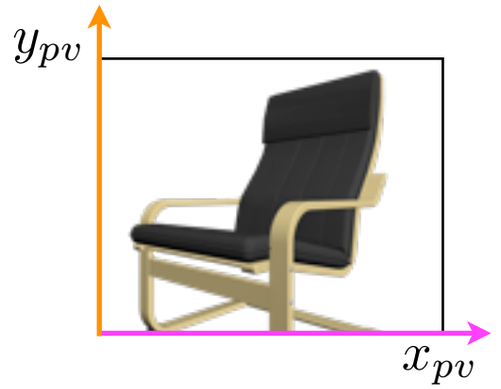
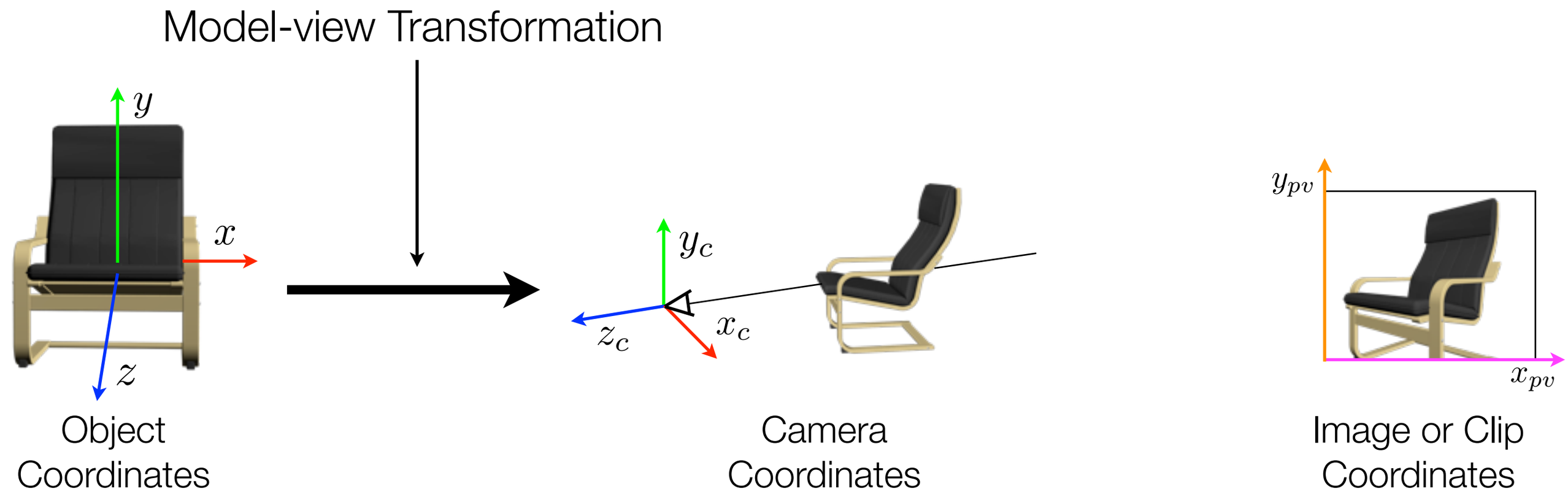


Image or Clip Coordinates

Back in Transformations in 3D...

The transformation applied to the input coordinates is called **model-view transformation**



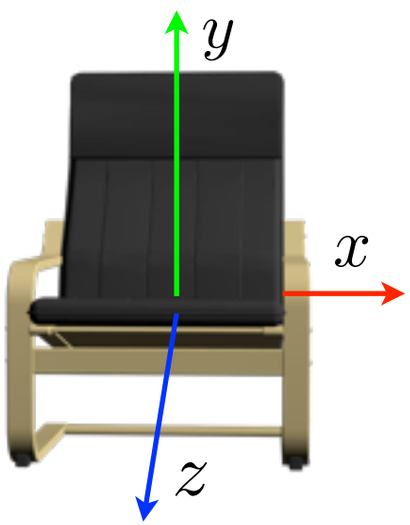
Back in Transformations in 3D...

and the matrix applied to the input coordinates
is called **model-view matrix**

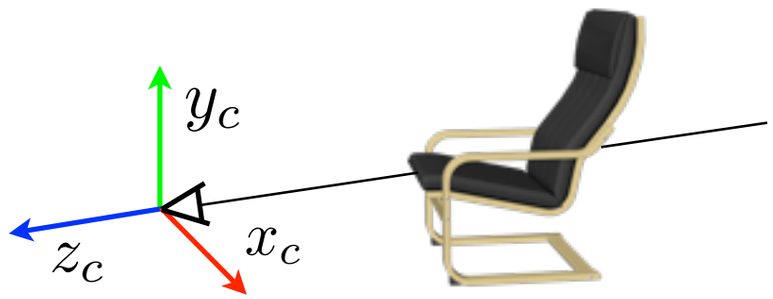
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Model-view Matrix

Model-view Transformation



Object
Coordinates



Camera
Coordinates

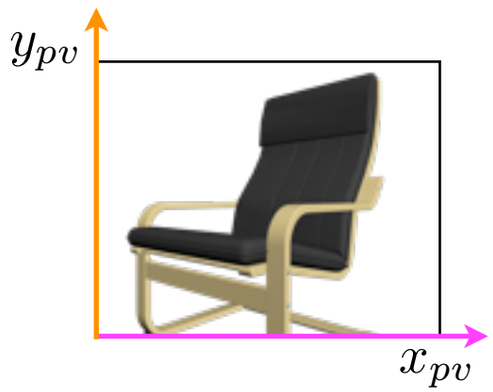
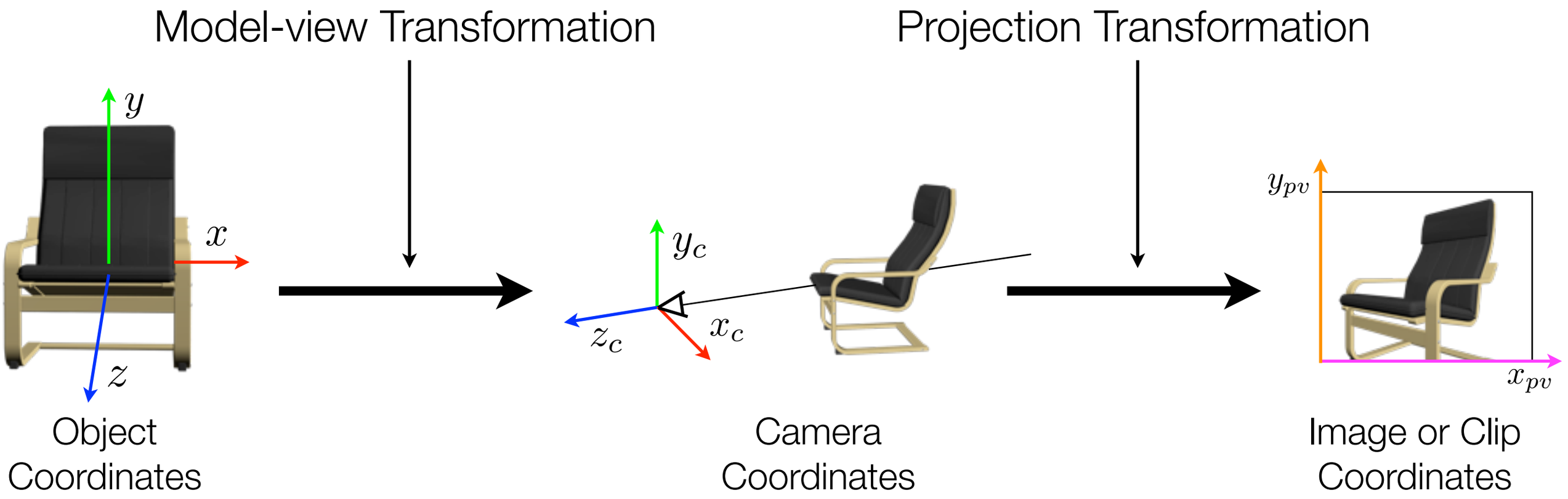


Image or Clip
Coordinates

In this lecture...

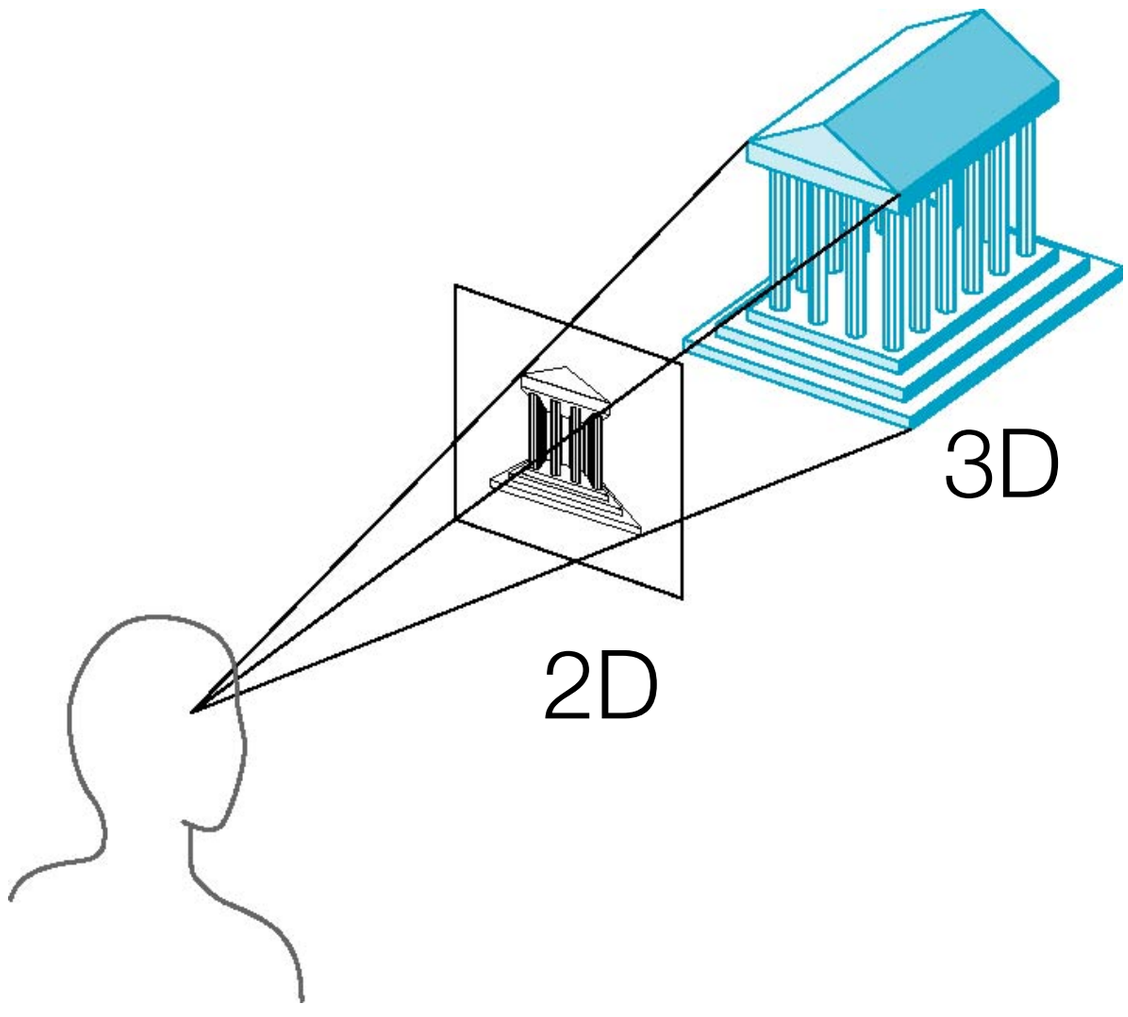
We will look at the **projection transformation**
and the projection matrix



What is a projection?

What is a projection?

Taking 3D information and collapsing it to 2D using an eye or camera.



Two Types of Projections

Type 1: Parallel

Projectors (lines of projection) are parallel, i.e., they meet at infinity. The lines specify a **direction of projection** (DOP).

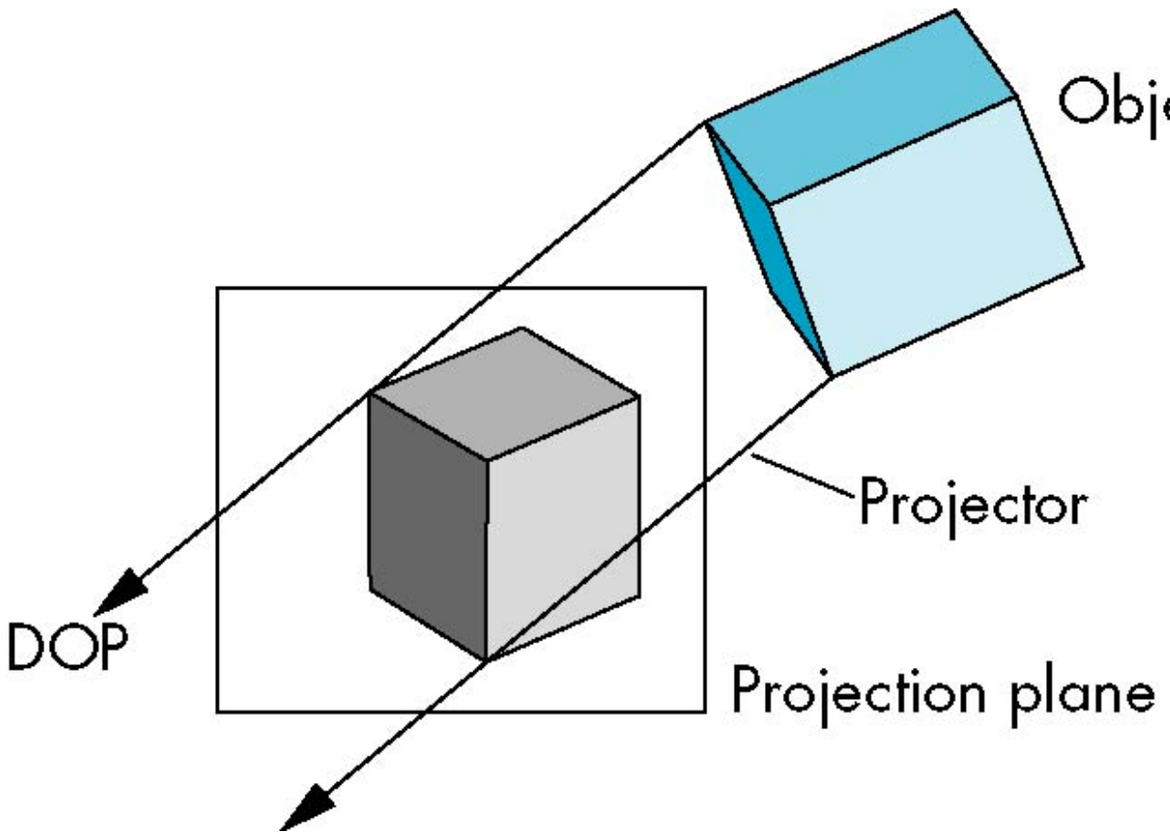


Image from Angel & Shreiner: Interactive Computer Graphics 7E.

Two Types of Projections

Type 1: Parallel -- Orthographic

Projectors orthogonal to projection plane.

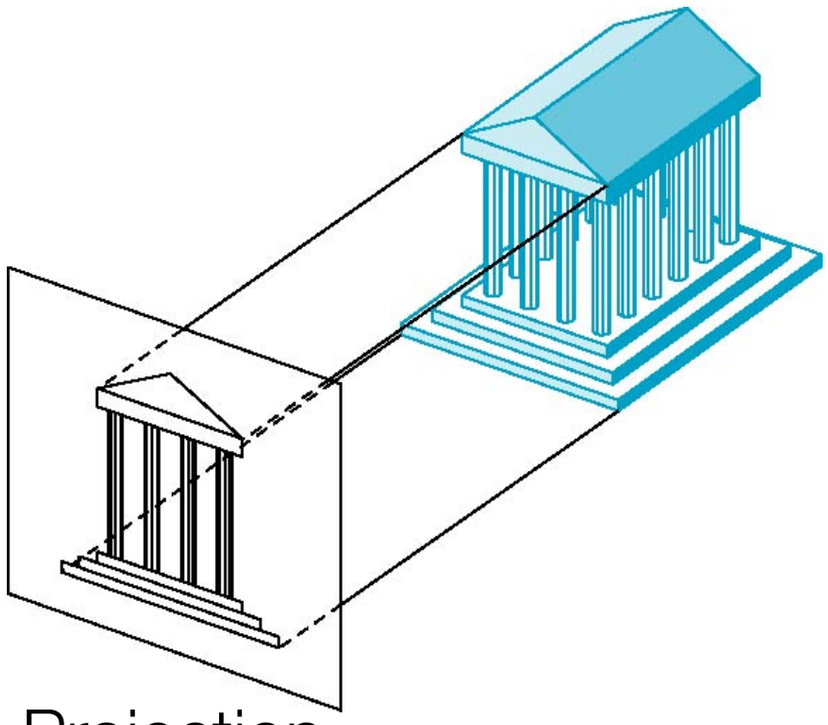
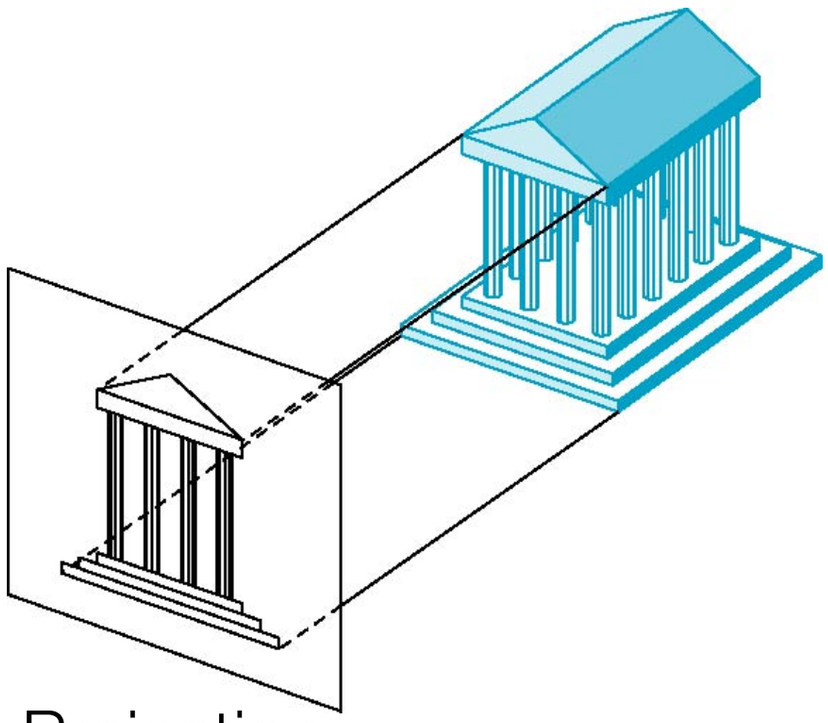


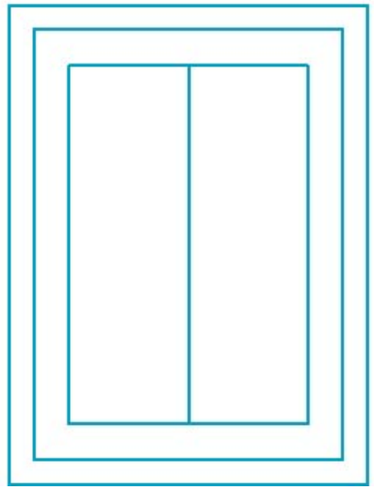
Image from Angel & Shreiner: Interactive Computer Graphics 7E.

Two Types of Projections

Type 1: Parallel -- Orthographic

Multiview orthographic projections are each parallel to principal face of the object.





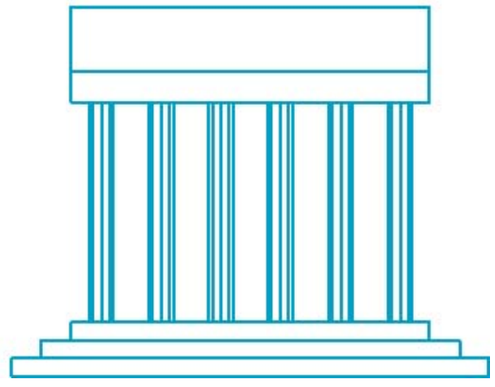
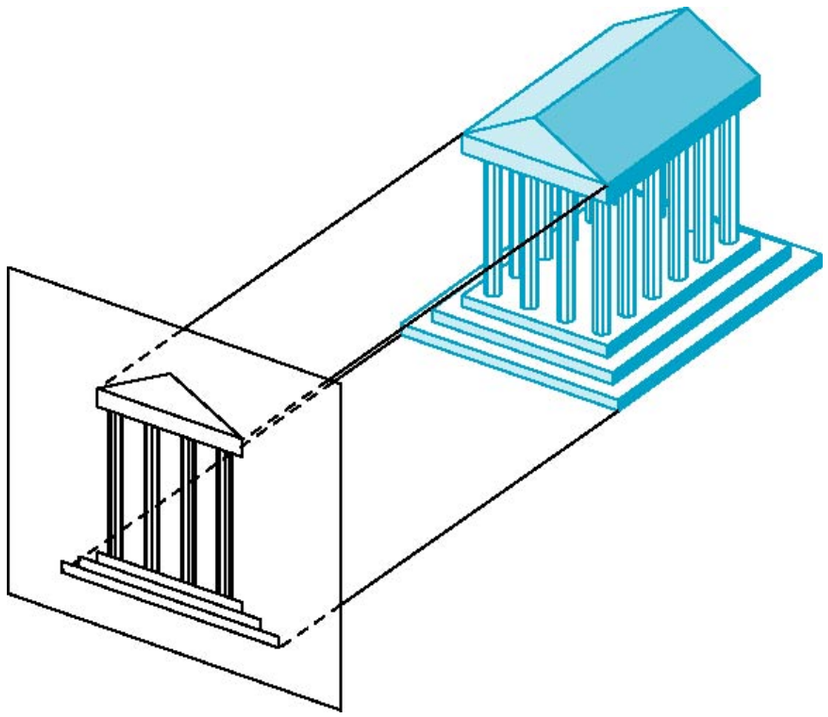


Image from Angel & Shreiner: Interactive Computer Graphics 7E.

Two Types of Projections

Type 1: Parallel -- Orthographic

Axonometric projections show more than one principal face of the object.



Axonometric

Image from Angel & Shreiner: Interactive Computer Graphics 7E.

Two Types of Projections

Type 1: Parallel -- Oblique

Projectors make a non right-angle with projection plane.

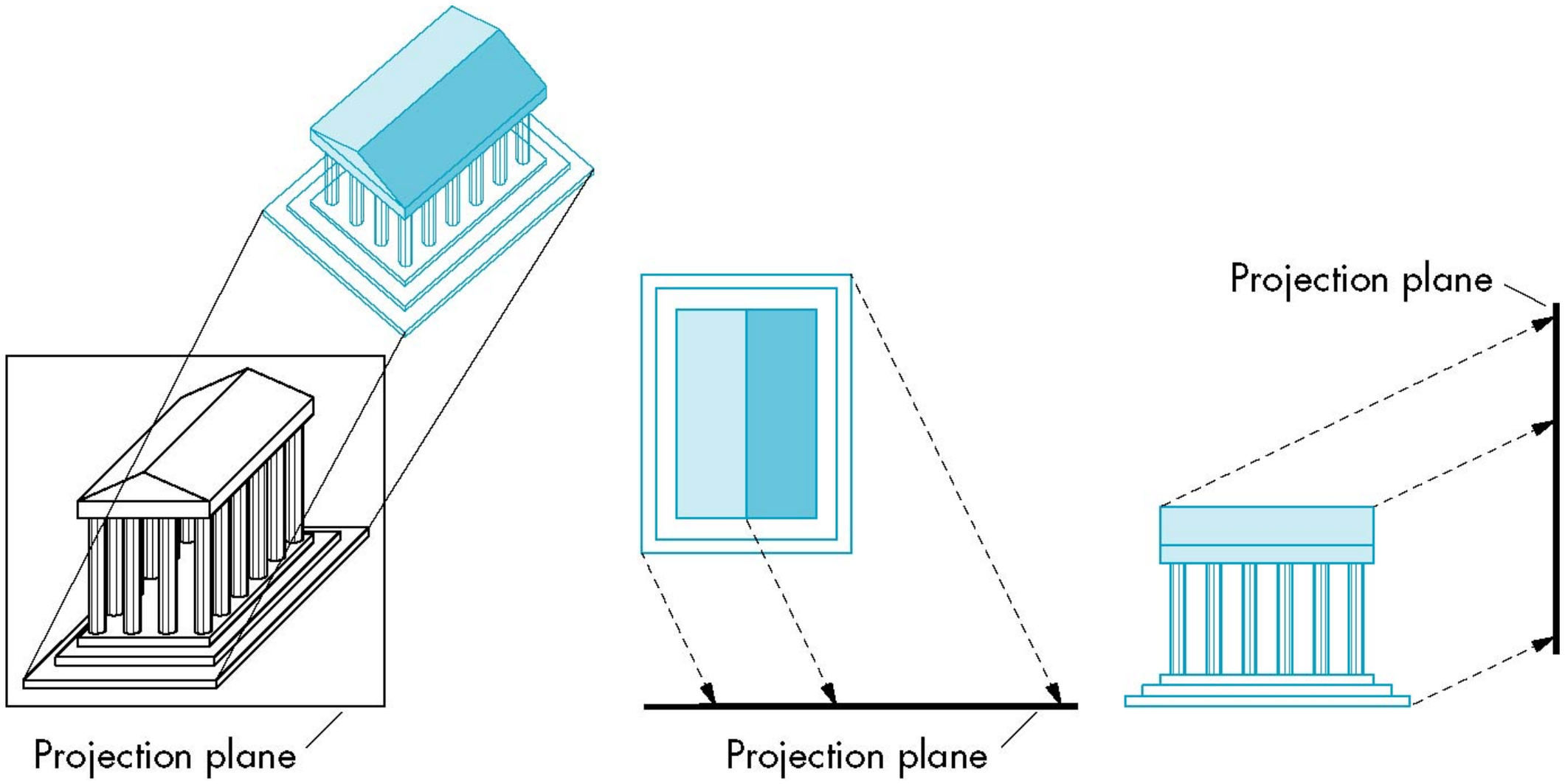


Image from Angel & Shreiner: Interactive Computer Graphics 7E.

Two Types of Projections

Type 2: Perspective

Projectors (lines of projection) meet at a point called **center of projection** (COP).

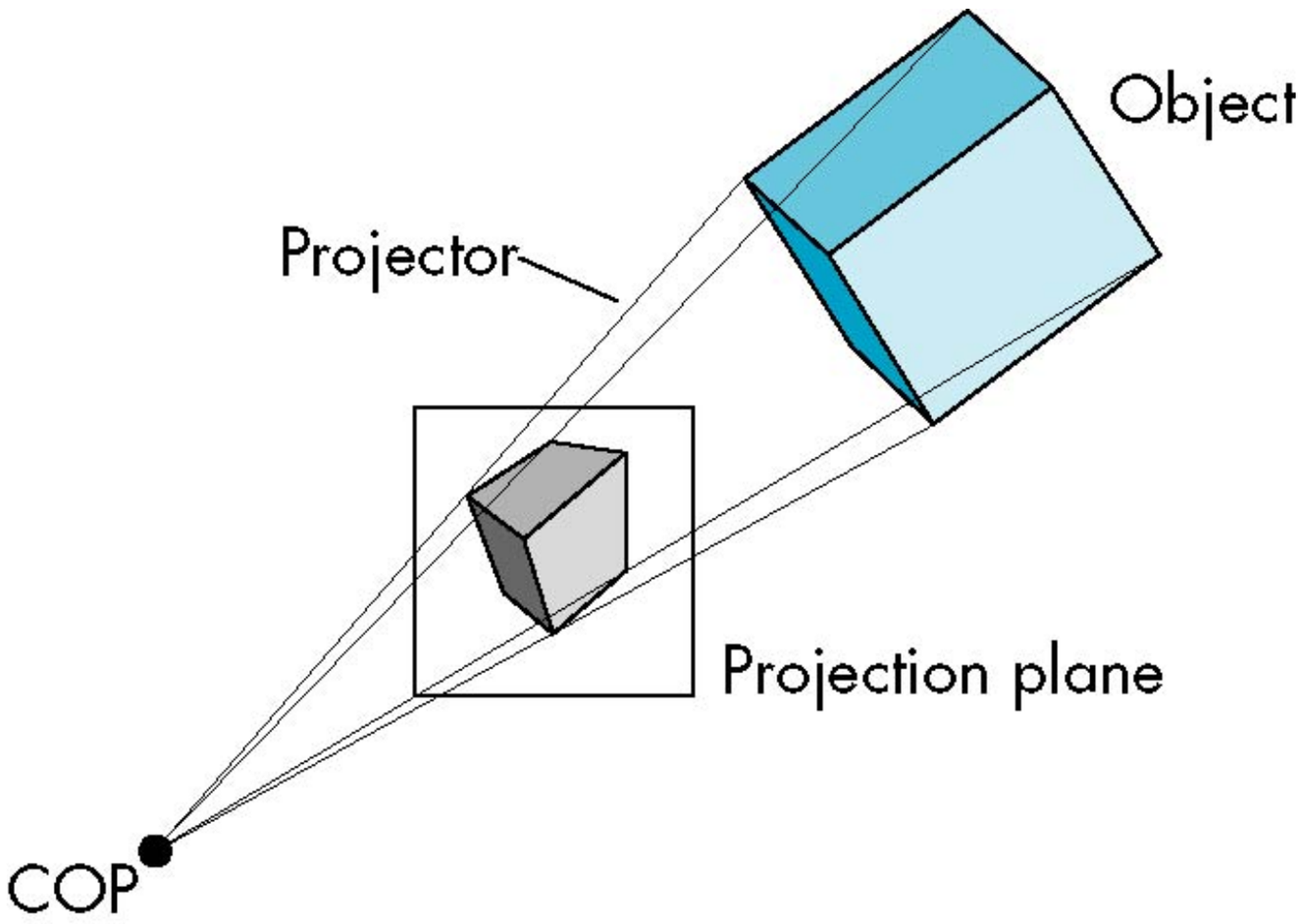


Image from Angel & Shreiner: Interactive Computer Graphics 7E.

When we looked at imaging...

We looked at **perspective projection** (type 2)

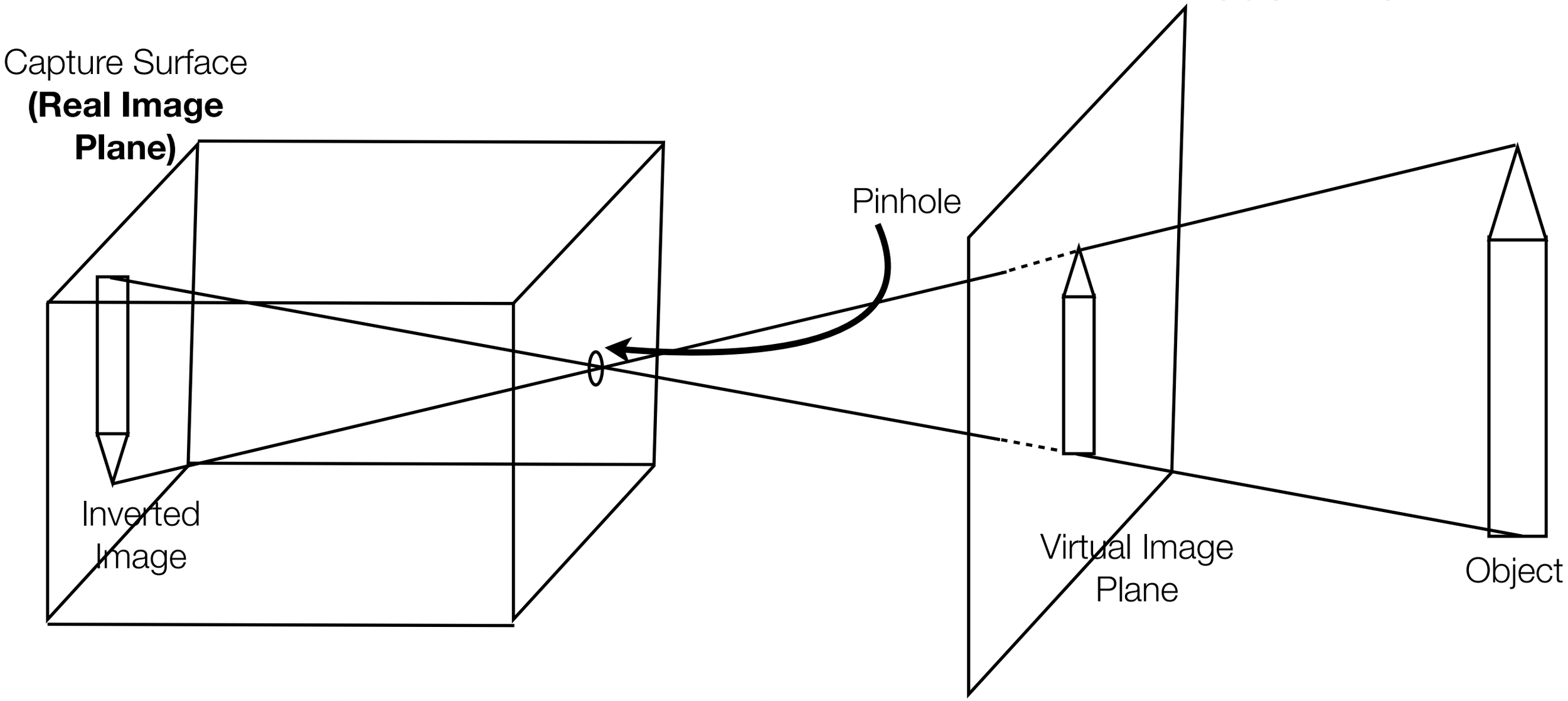


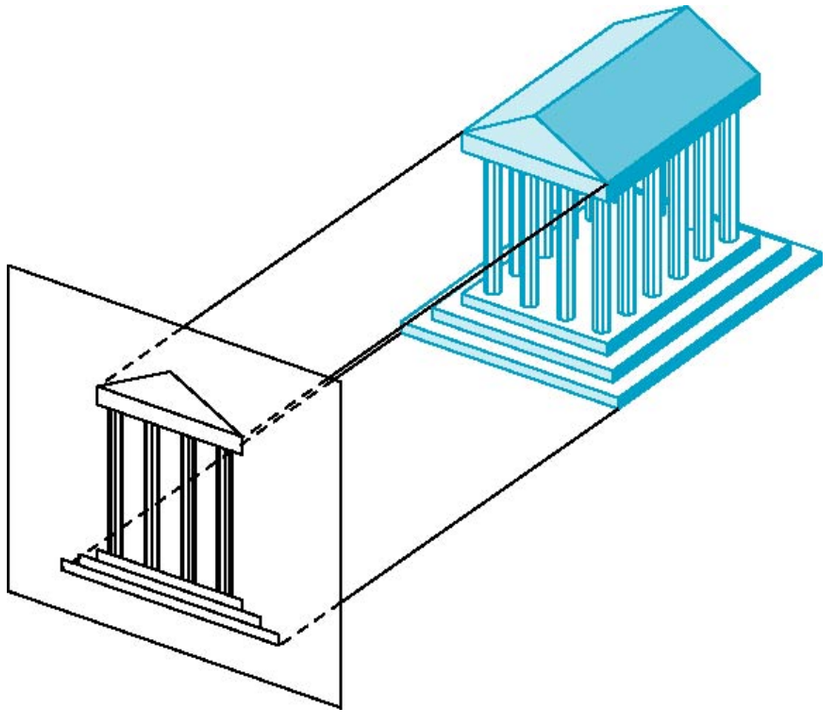
Image is actually formed on the **real image plane**, but is inverted.

To make life simpler, in computer graphics, we introduce a **virtual image plane** that shows an erect image.

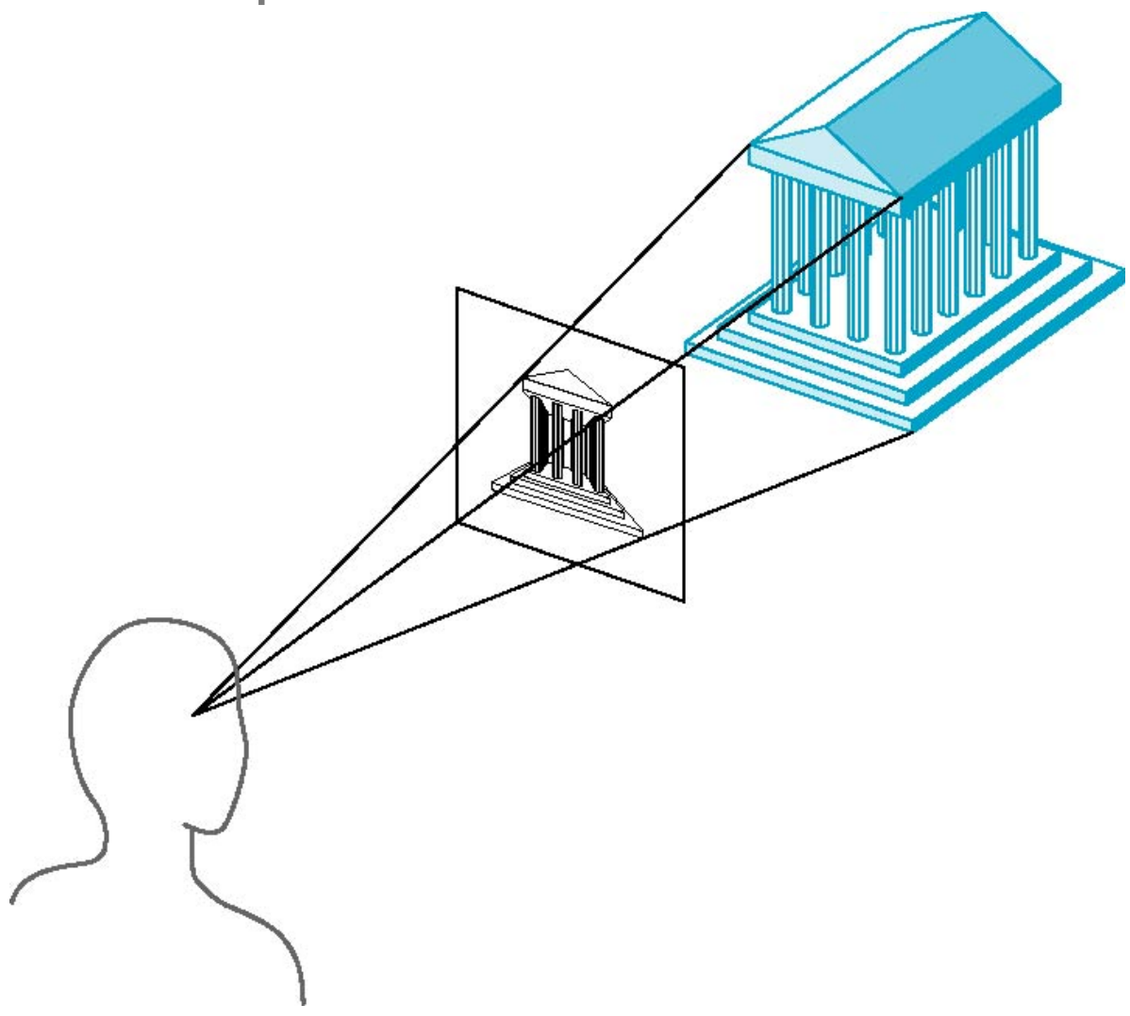
Image is NOT formed on virtual image plane.

Two Types of Projections

As the object moves away from the camera,
its size on the projection plane...



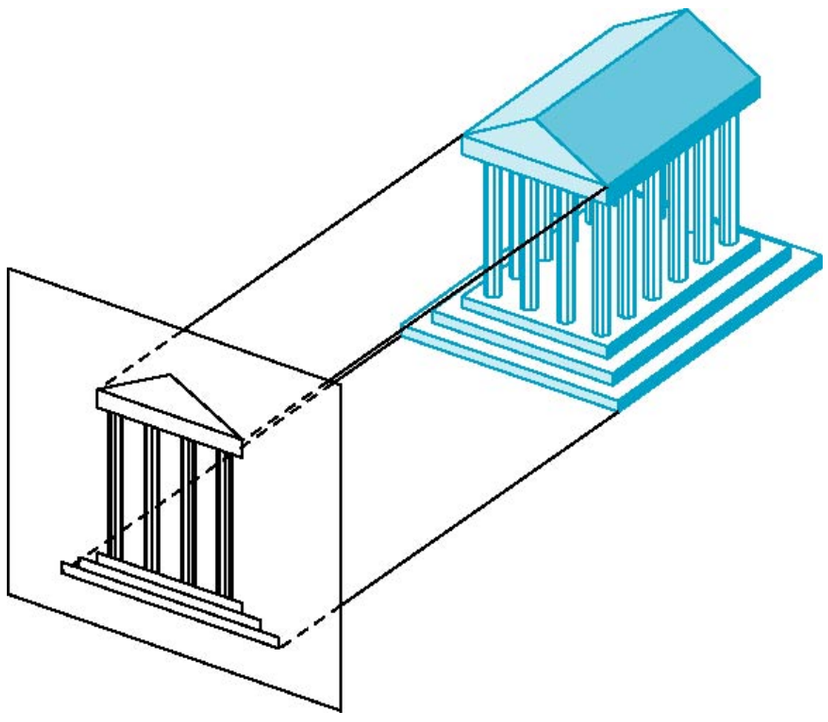
Remains the **same**
in **parallel** projection



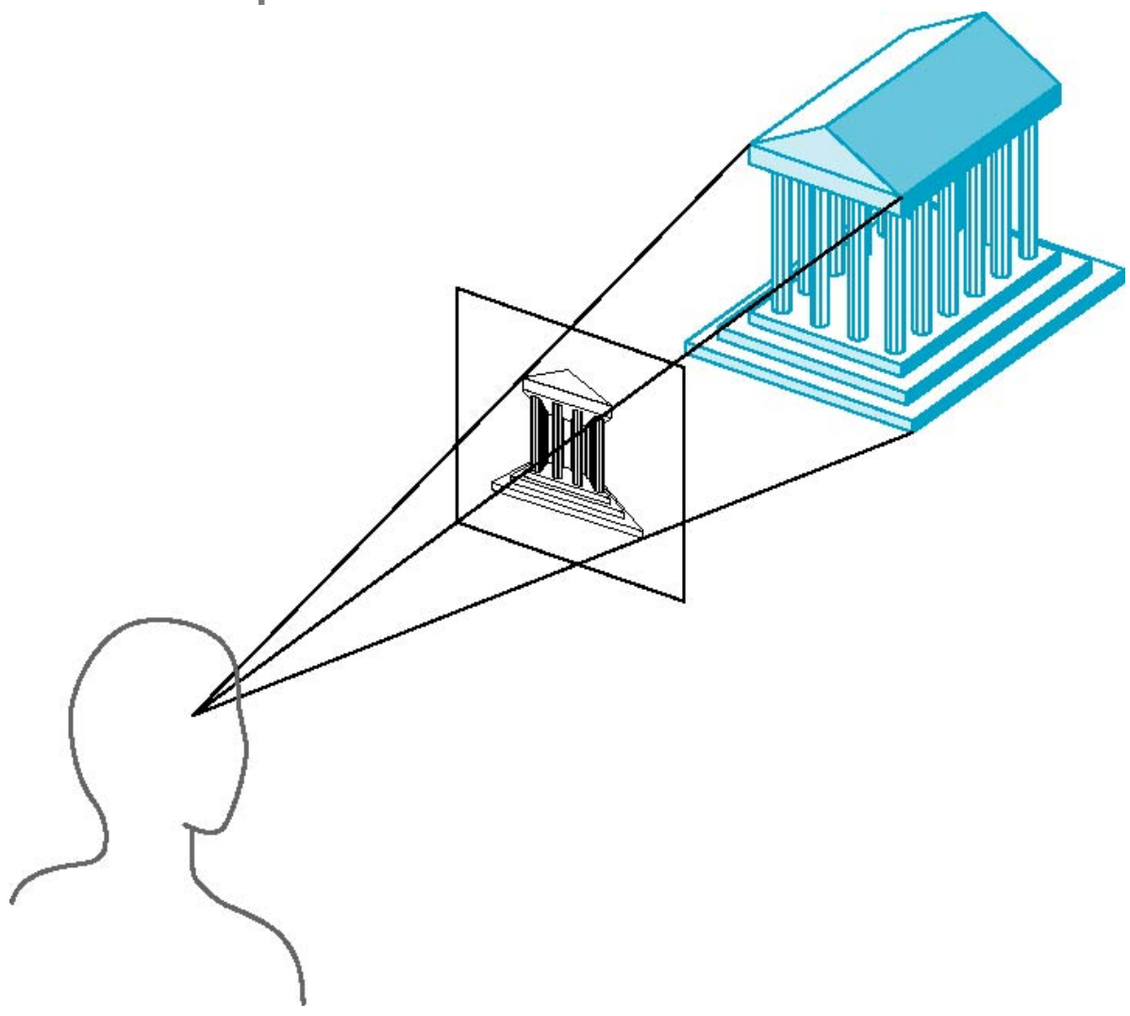
Decreases
in **perspective** projection

Two Types of Projections

As the object moves away from the camera,
its size on the projection plane...



Remains the **same**
in **parallel** projection



Decreases
in **perspective** projection

Real cameras and eyes perform
perspective projection

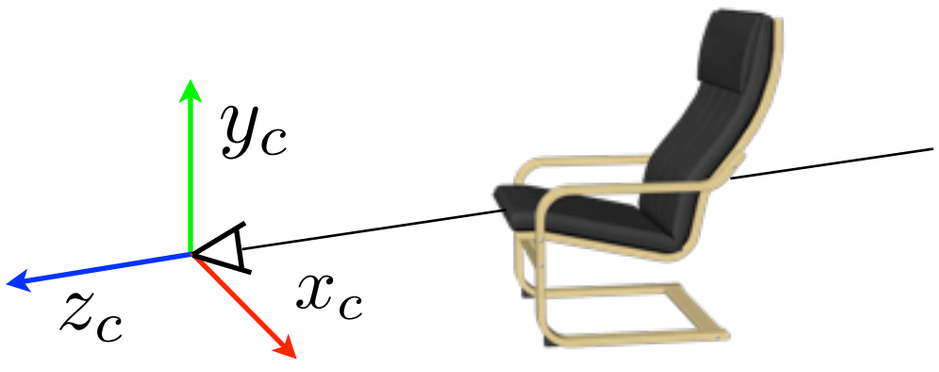
Before we model projections in WebGL, let us talk a bit about positioning the camera.

Positioning the Camera

Positioning the Camera

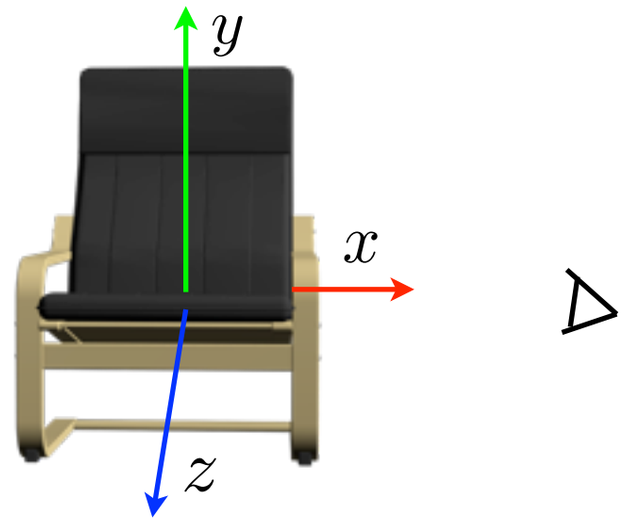
There are two ways to position a camera and an object

Camera Coordinates



Keep the **camera at the origin** and **move the object** using the **modelview** transformation.

Object Coordinates



Keep the **object at the origin** and **move the camera** using the **inverse of the modelview** transformation.

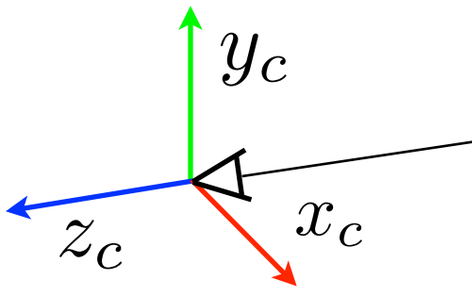
Positioning the Camera

There are two ways to position a camera and an object

Camera Coordinates

Object Coordinates

Both methods provide exactly the same image!

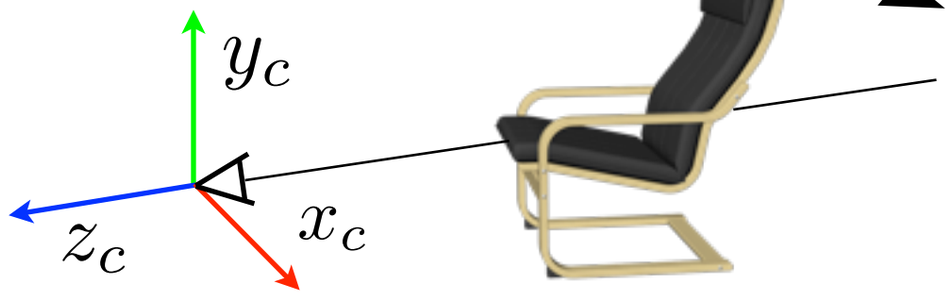


Keep the **came** **t at the origin** and **move the** **amera** using the **modelview** t **of the modelview** transformation.

Positioning the Camera

There are two ways to position a camera and an object

Camera Coordinates



Keep the **camera at the origin** and **move the object** using the **modelview** transformation.

WebGL uses this one.

In WebGL,

the camera is at the origin (0,0,0),

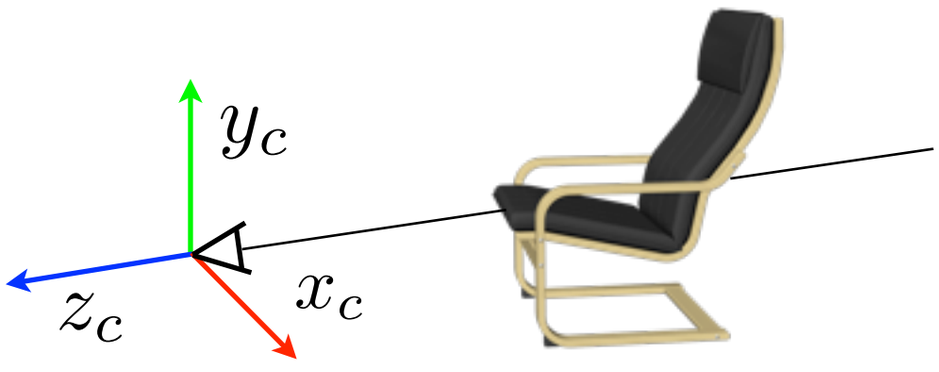
the x- and y-axes of the camera are aligned with the image,

the z-axis of the camera points out of the image.

Positioning the Camera

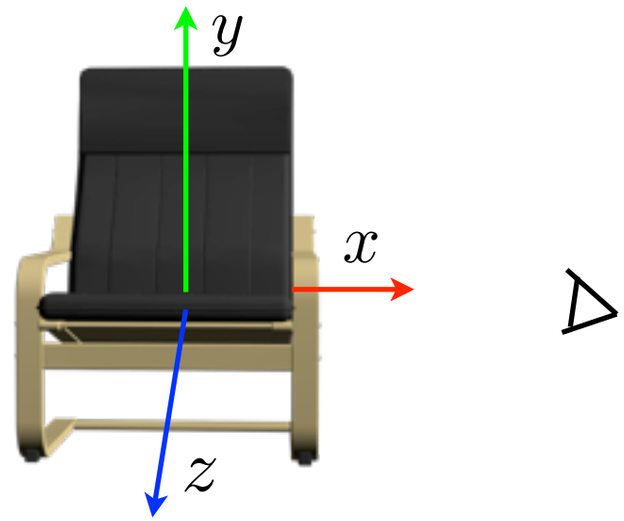
There are two ways to position a camera and an object

Camera Coordinates



Keep the **camera at the origin** and **move the object** using the **modelview** transformation.

Object Coordinates

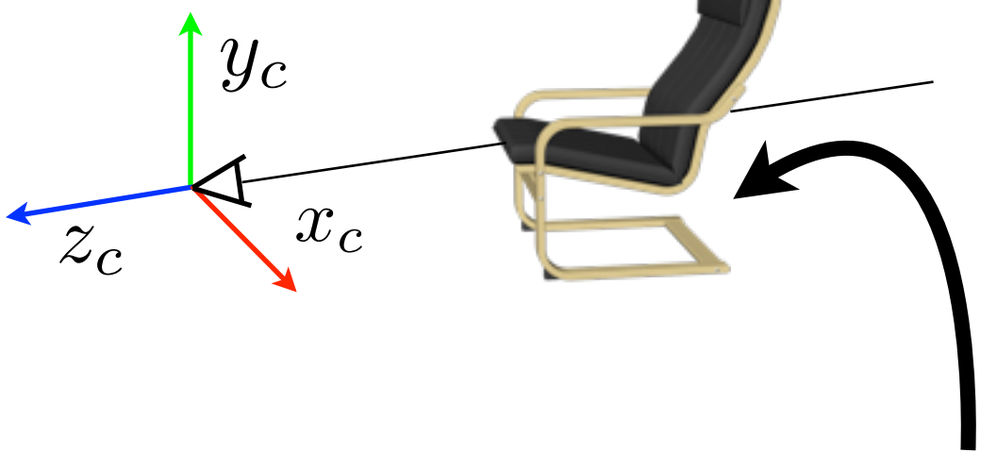


Keep the **object at the origin** and **move the camera** using the **inverse of the modelview** transformation.

Positioning the Camera

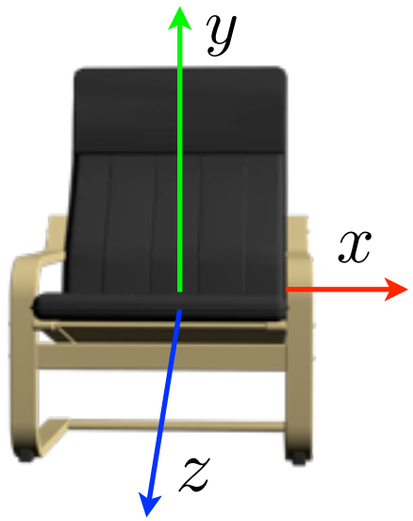
Similarly, **you** have two options in WebGL

Camera Coordinates



You can use the previous lecture to specify a **modelview transformation** for the **object**, and never move the camera.

Object Coordinates



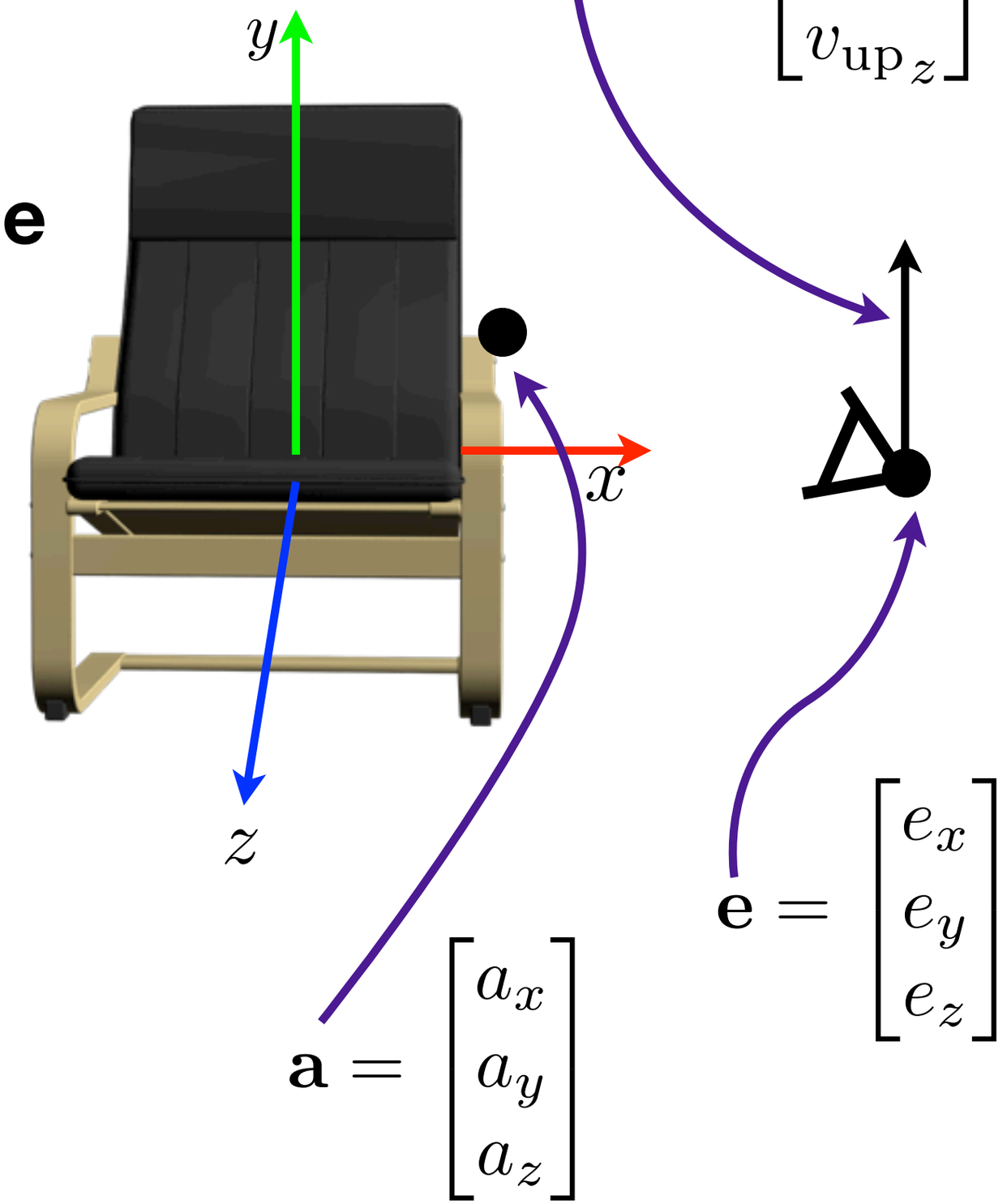
Or you can use the **look-at** approach to specify an **inverse modelview transformation** for the **camera**.

and then invert it to get the **modelview transformation** for the **object**.

Look-At Approach

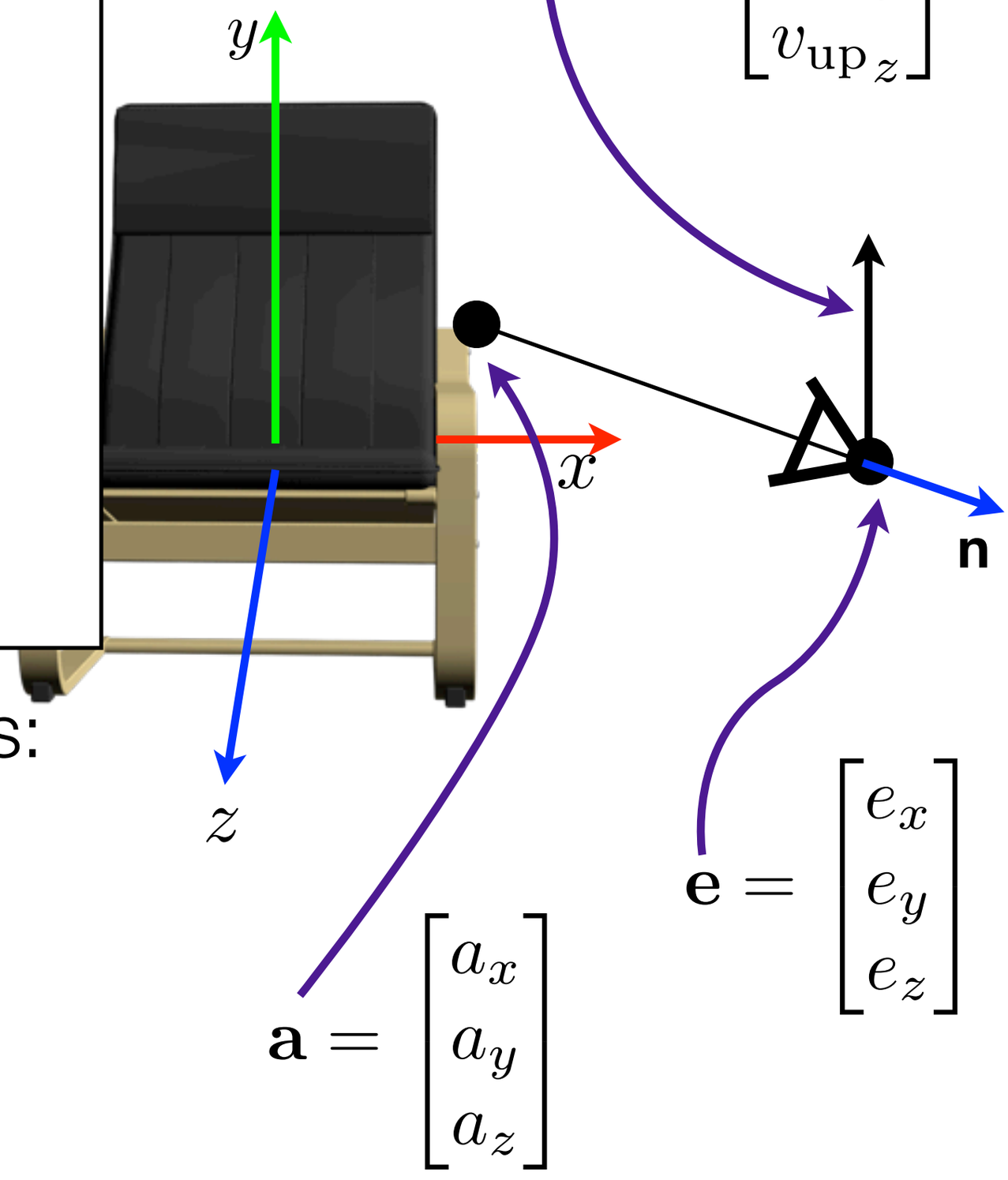
Specify three items:

- View Reference Point or Eye **e**
- Look At Point **a**
- Up Vector $\mathbf{v}_{\text{up}}$ (Unit vector)



Look-At Approach

$$\mathbf{d} = \mathbf{e} - \mathbf{a}, \quad \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|} \leftarrow \text{Vector norm}$$



Compute in Object Coordinates:

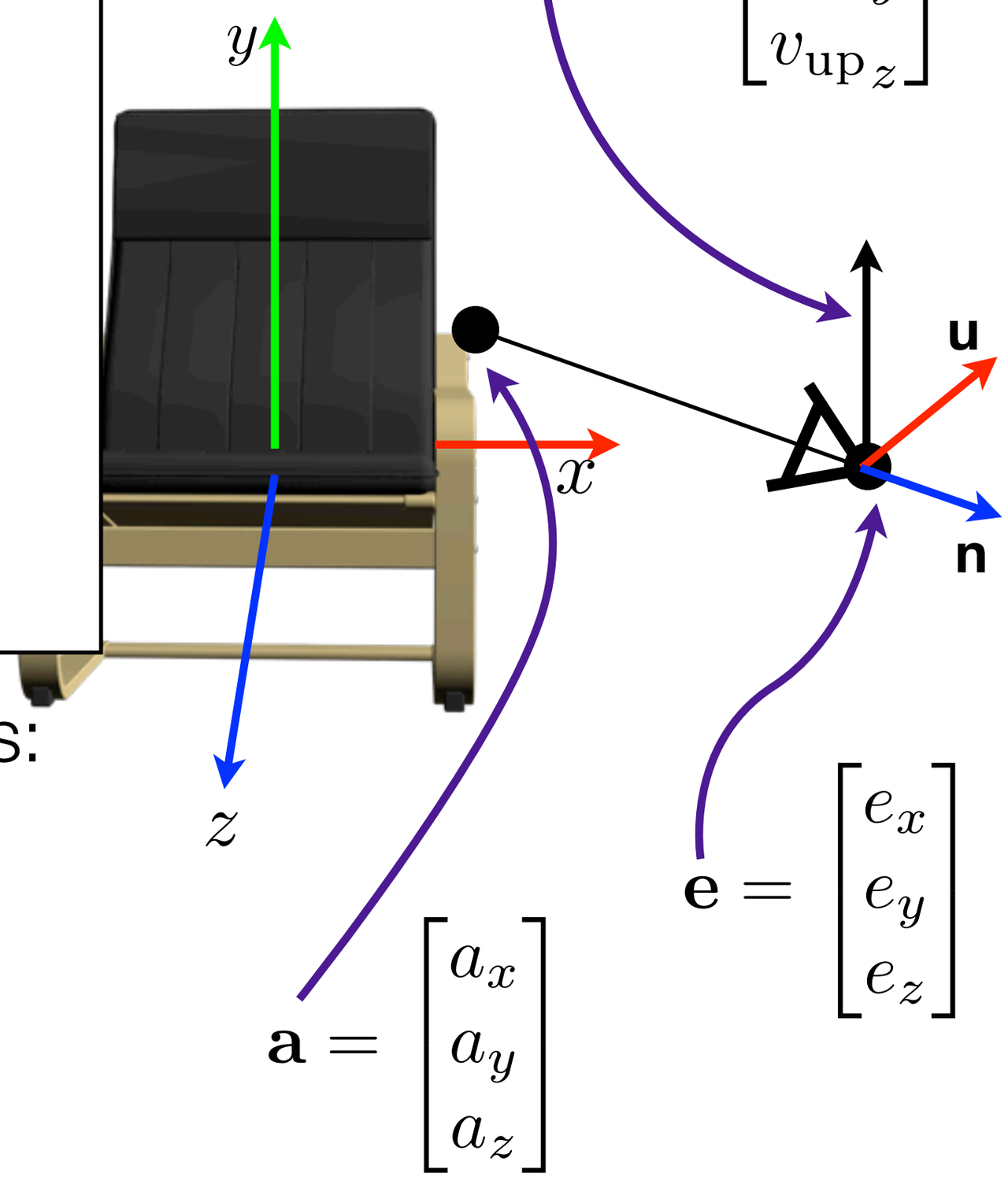
View plane normal, **n**

Look-At Approach

$$\mathbf{d} = \mathbf{e} - \mathbf{a}, \quad \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$ ← Vector norm

$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n}, \quad \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

Cross product



Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**

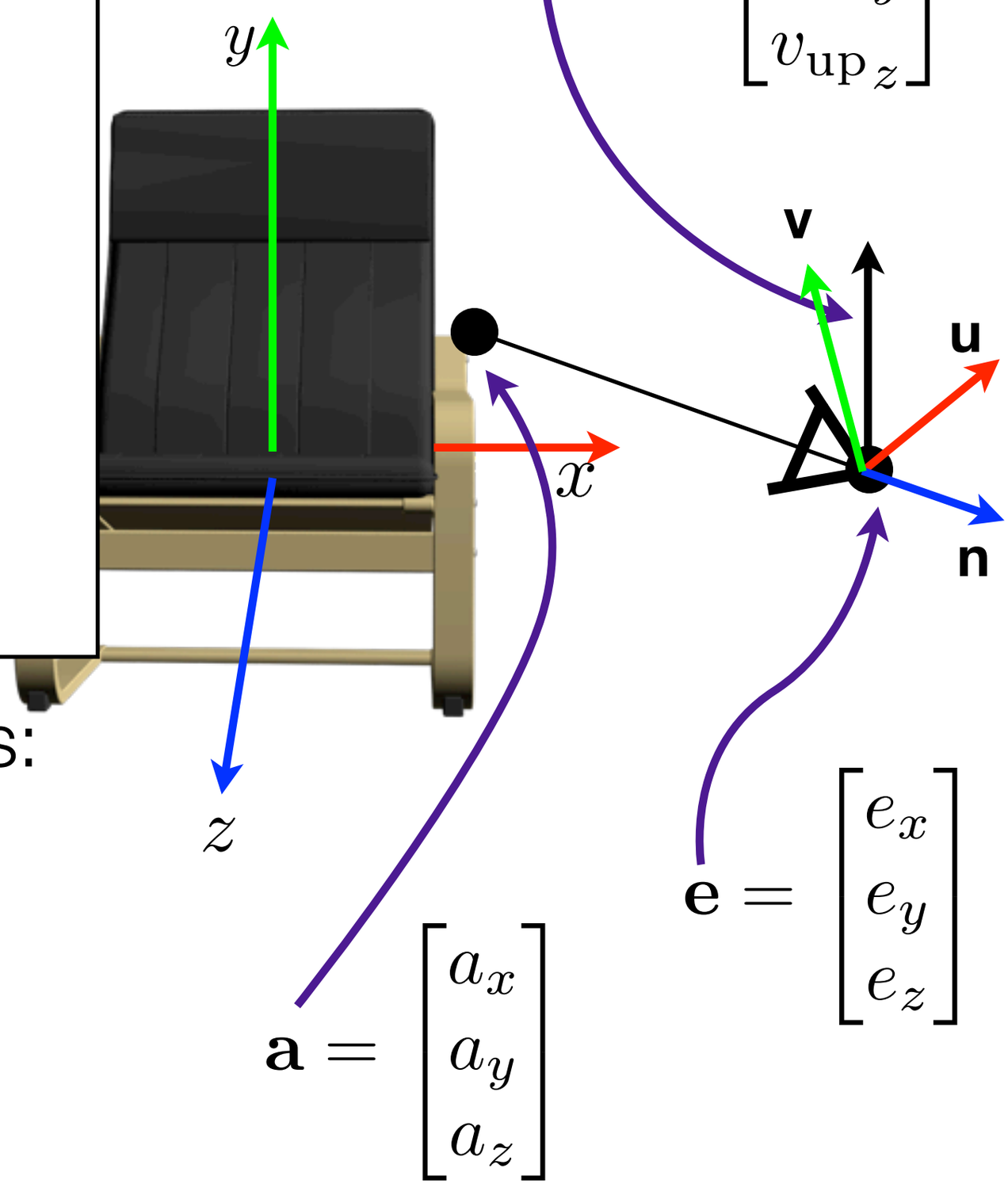
Look-At Approach

$$\mathbf{d} = \mathbf{e} - \mathbf{a}, \quad \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$ ← Vector norm

$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n}, \quad \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

Cross product

$$\mathbf{l} = \mathbf{n} \times \mathbf{u}, \quad \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$



Compute in Object Coordinates:

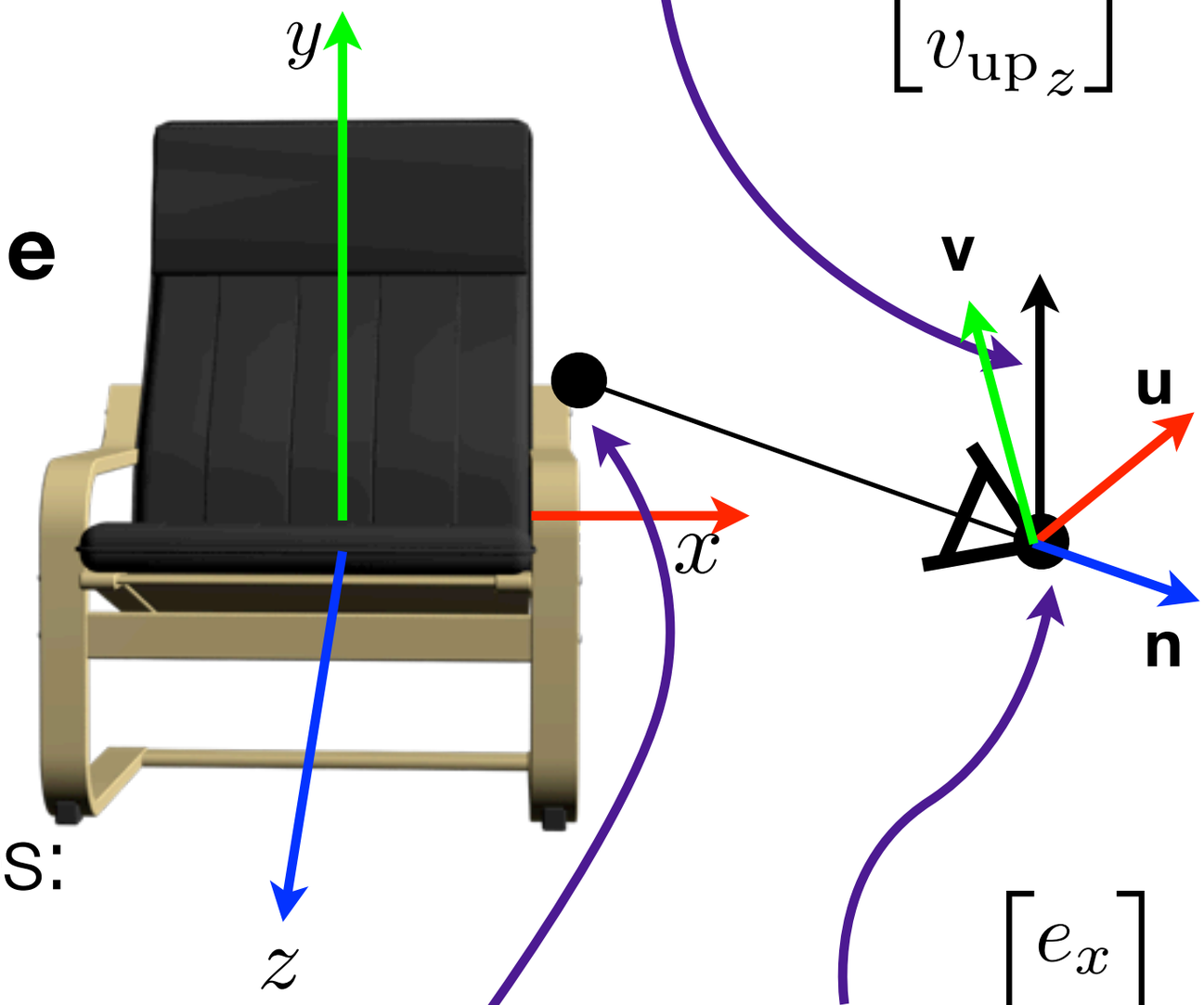
- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

Look-At Approach

Specify three items:

- View Reference Point or Eye **e**
- Look At Point **a**
- Up Vector $\mathbf{v}_{\text{up}}$ (Unit vector)

$$\mathbf{v}_{\text{up}} = \begin{bmatrix} v_{\text{up}_x} \\ v_{\text{up}_y} \\ v_{\text{up}_z} \end{bmatrix}$$



Compute in Object Coordinates:

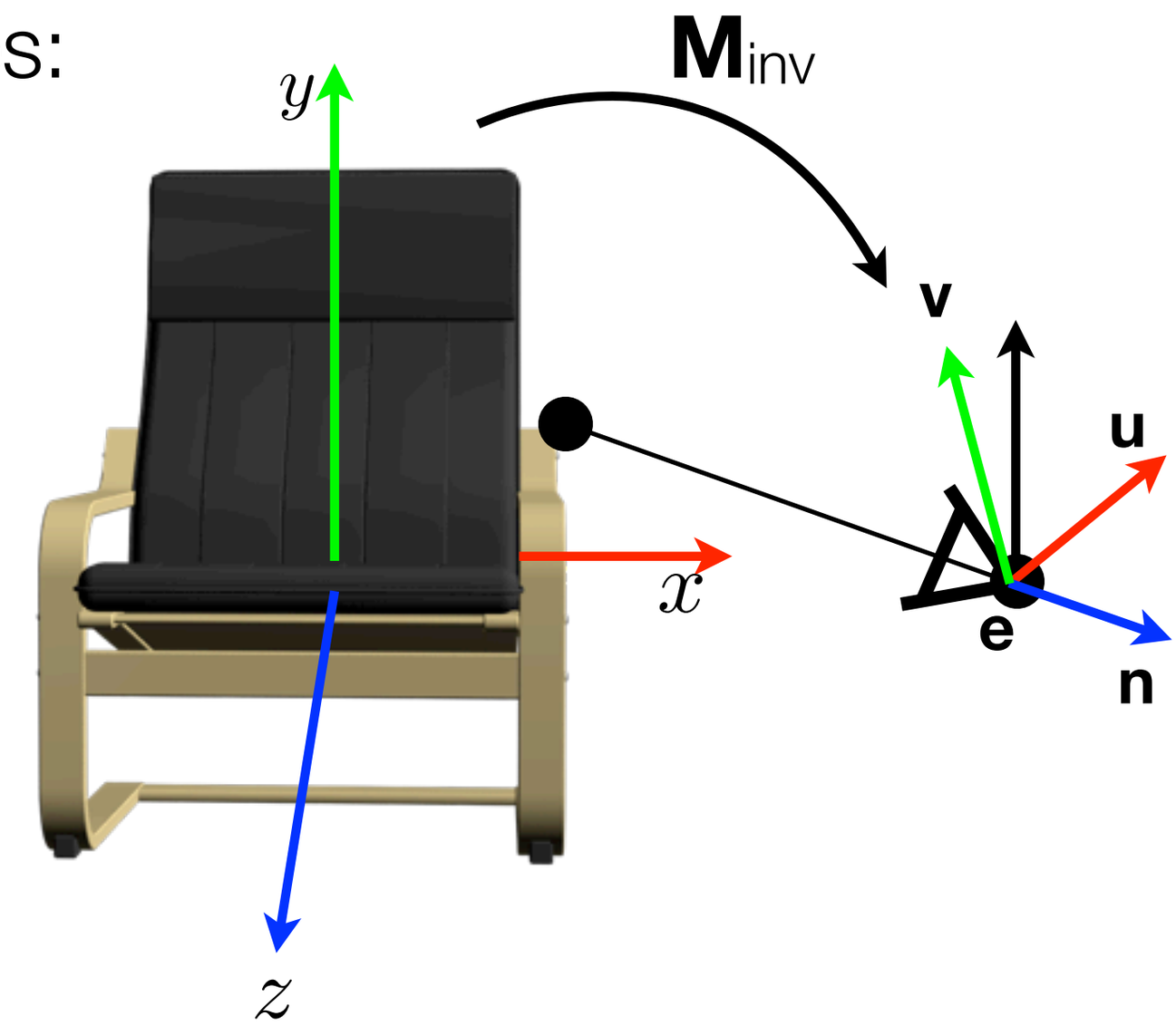
- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

You can use these to set up an inverse model-view transformation $\mathbf{M}_{inv}$ for the camera...

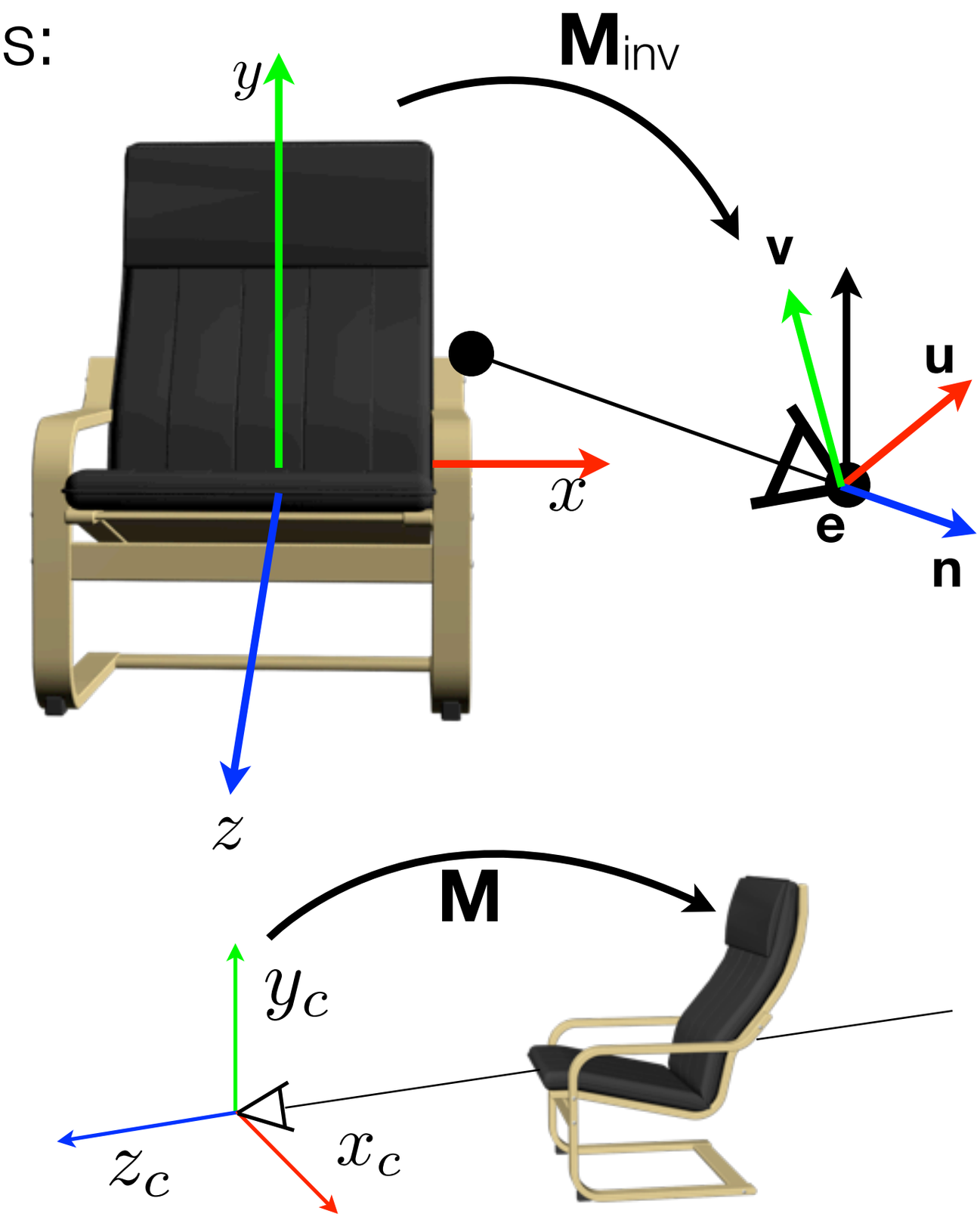


Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

You can use these to set up an inverse model-view transformation $\mathbf{M}_{inv}$ for the camera and invert it to get a model-view transformation for the object **M**.

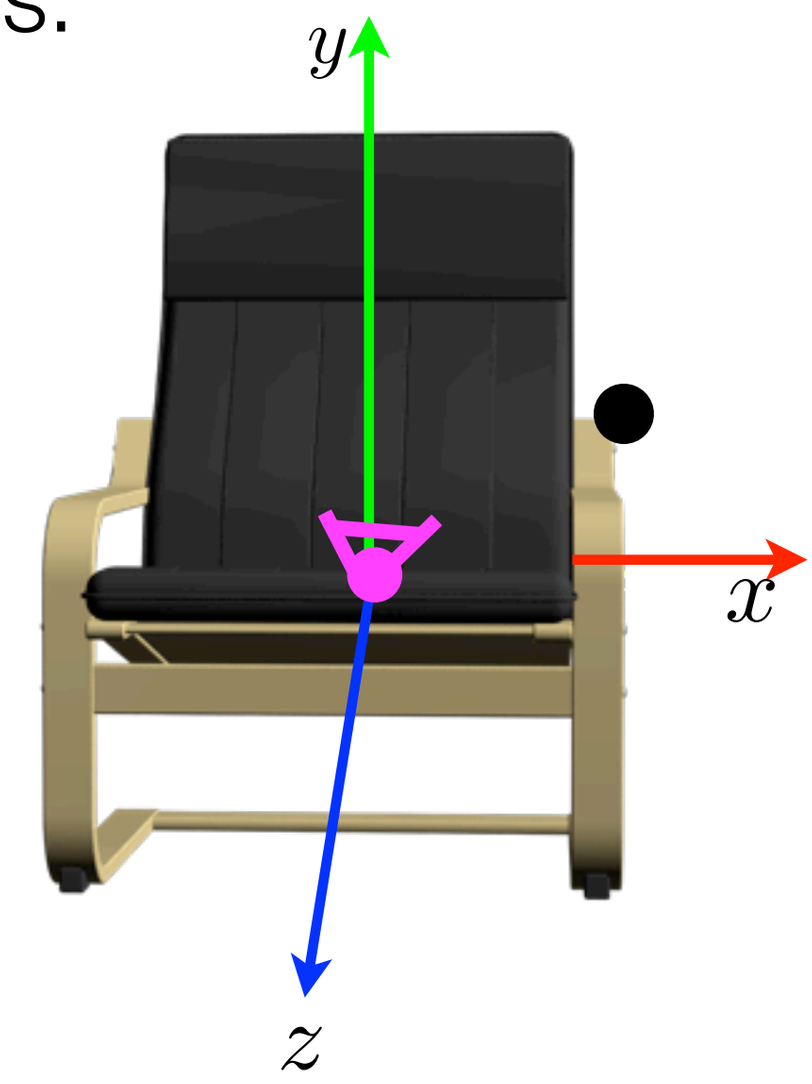


Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

To set up the Camera in Object Coordinates, i.e., to get $\mathbf{M}_{inv}$:



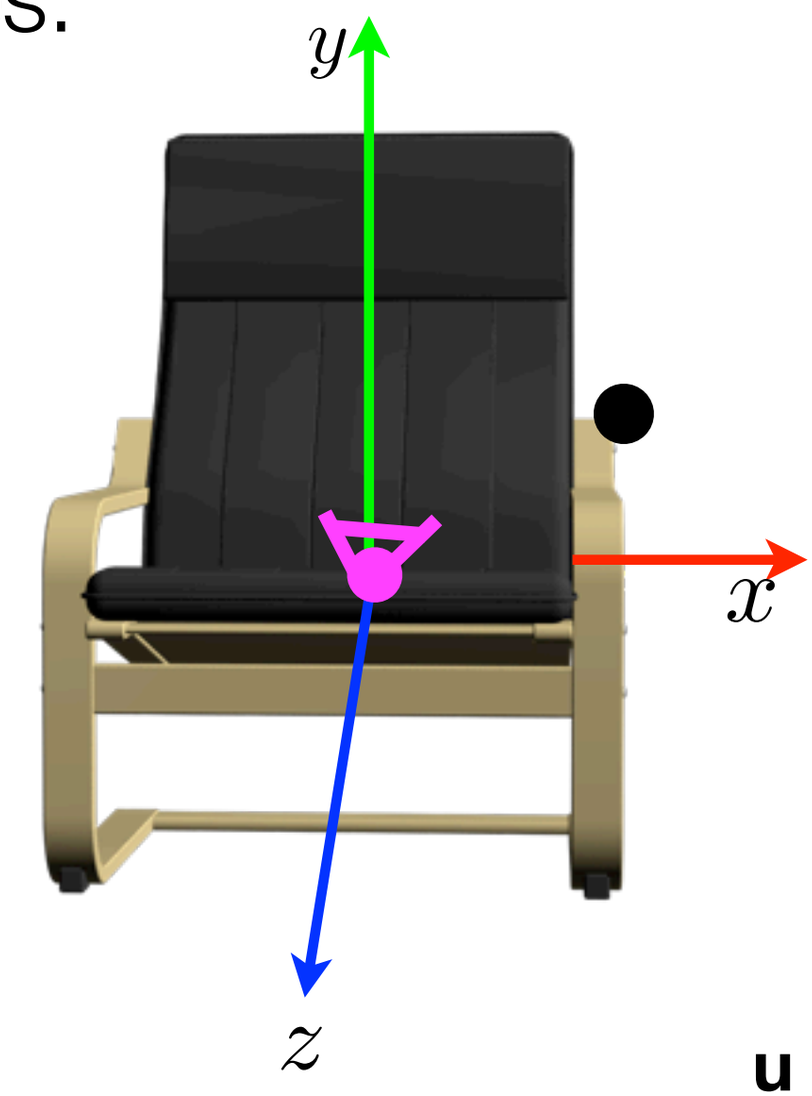
Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

To set up the Camera in Object Coordinates, i.e., to get $\mathbf{M}_{inv}$:

- First rotate the camera



$$\begin{matrix} \mathbf{u} & \mathbf{v} & \mathbf{n} & \\ \downarrow & \downarrow & \downarrow & \\ \end{matrix}$$
$$\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

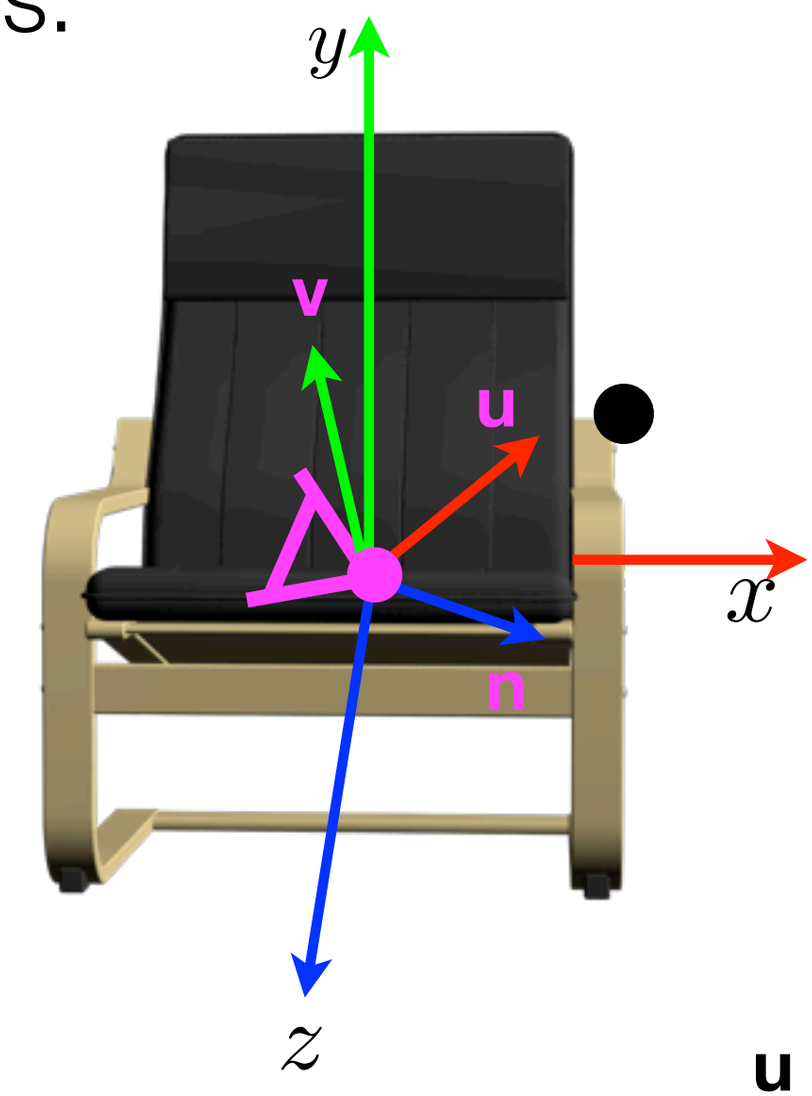
Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

To set up the Camera in Object Coordinates, i.e., to get $\mathbf{M}_{inv}$:

- First rotate the camera



$$\begin{array}{cccc} \mathbf{u} & \mathbf{v} & \mathbf{n} & \\ \downarrow & \downarrow & \downarrow & \\ \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

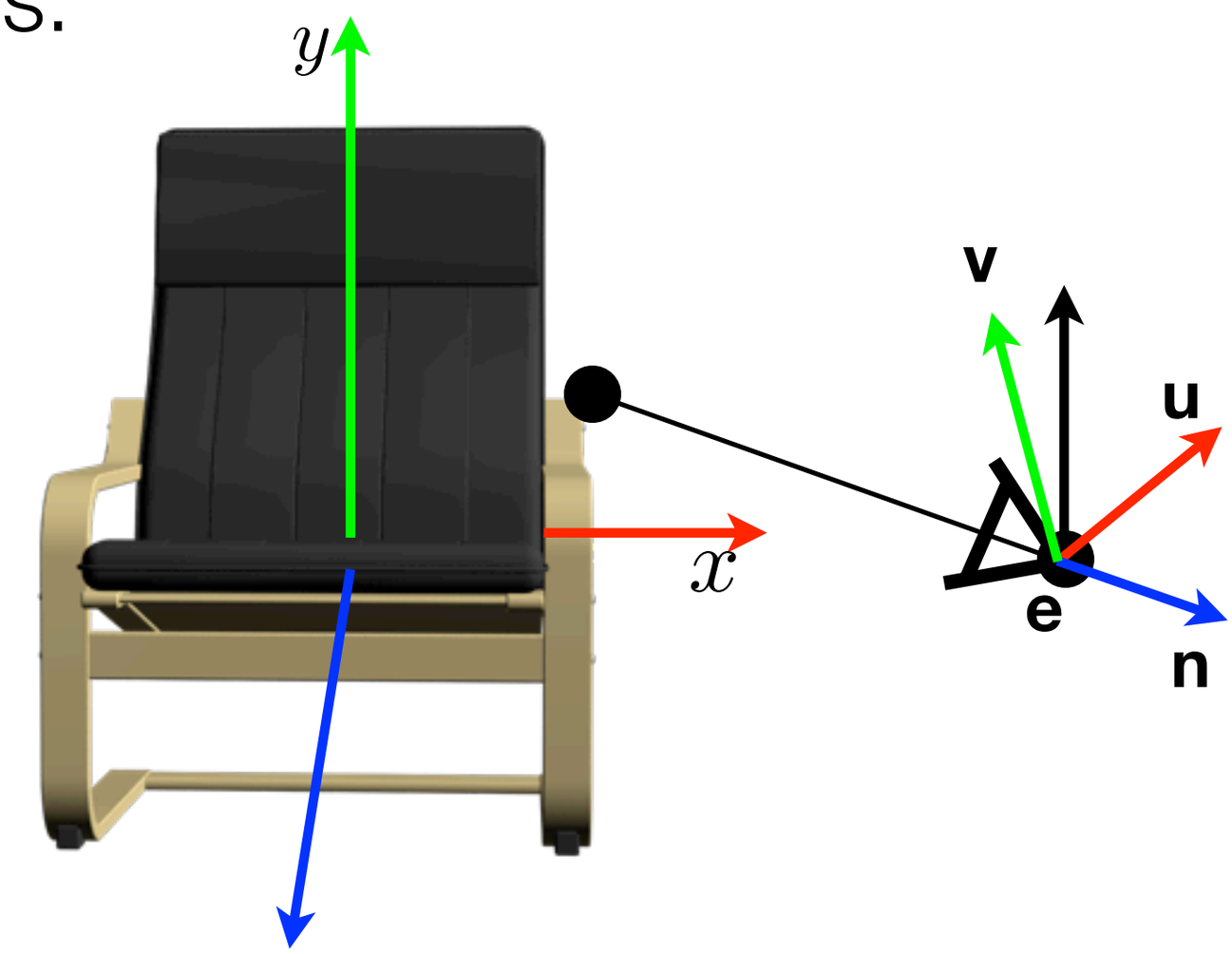
Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

To set up the Camera in Object Coordinates, i.e., to get $\mathbf{M}_{inv}$:

- First rotate the camera
- Then translate the rotated camera



$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e **u** **v** **n**

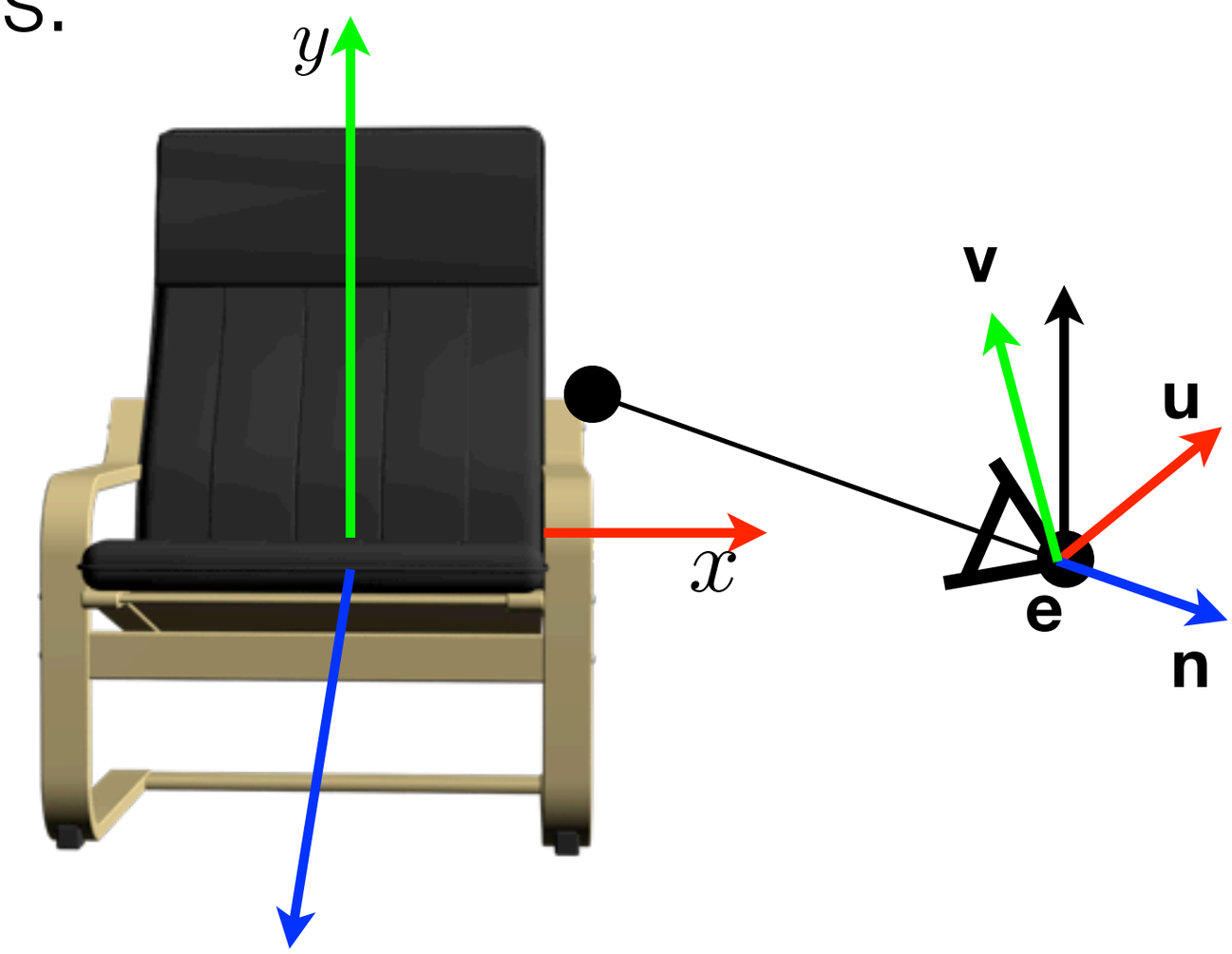
Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

To set up the Camera in Object Coordinates, i.e., to get $\mathbf{M}_{inv}$:

- First rotate the camera
- Then translate the rotated camera



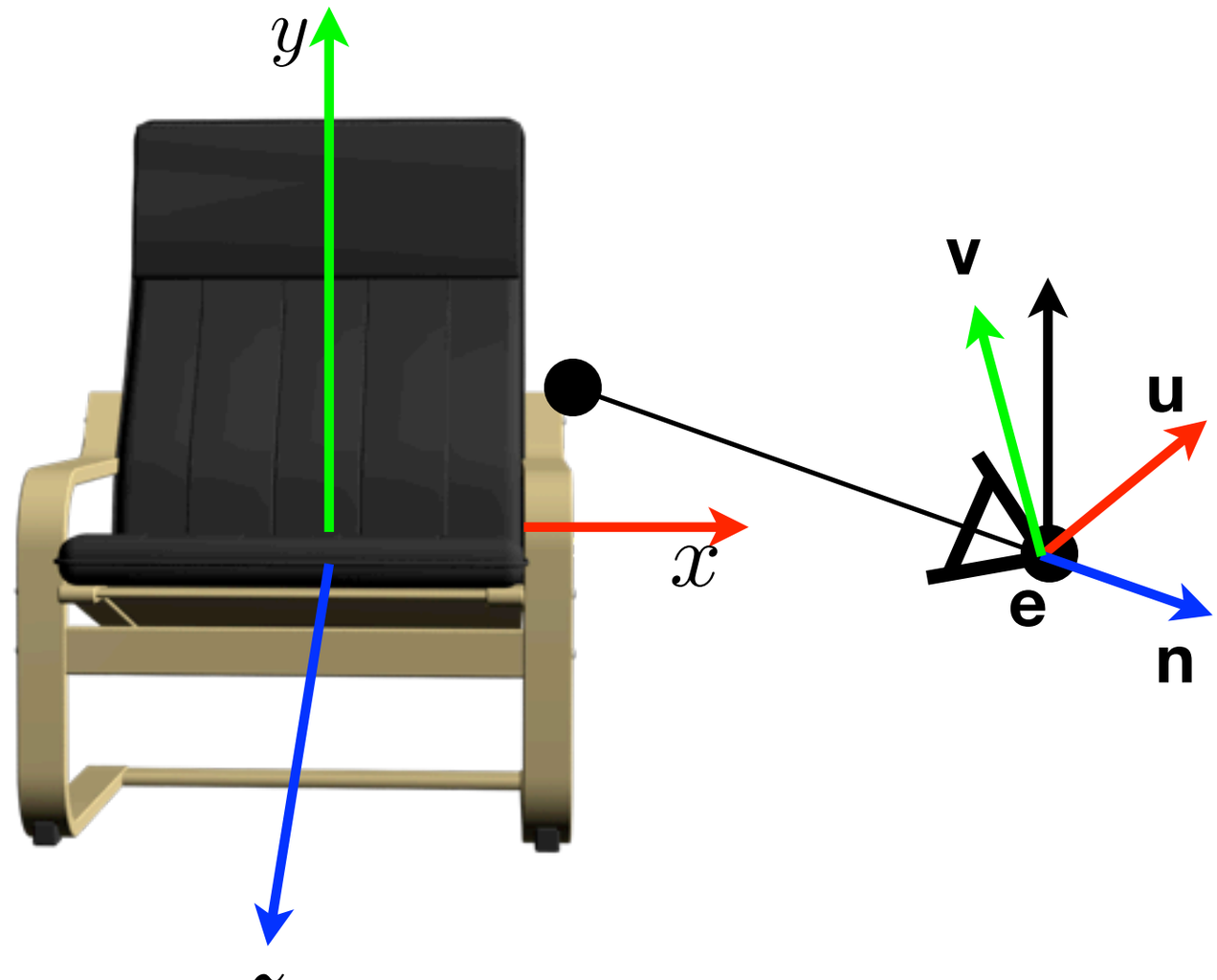
$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e **u** **v** **n**

Inverse Model-view Transformation for Camera

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{inv})^{-1}$



Inverse Model-view Transformation for Camera

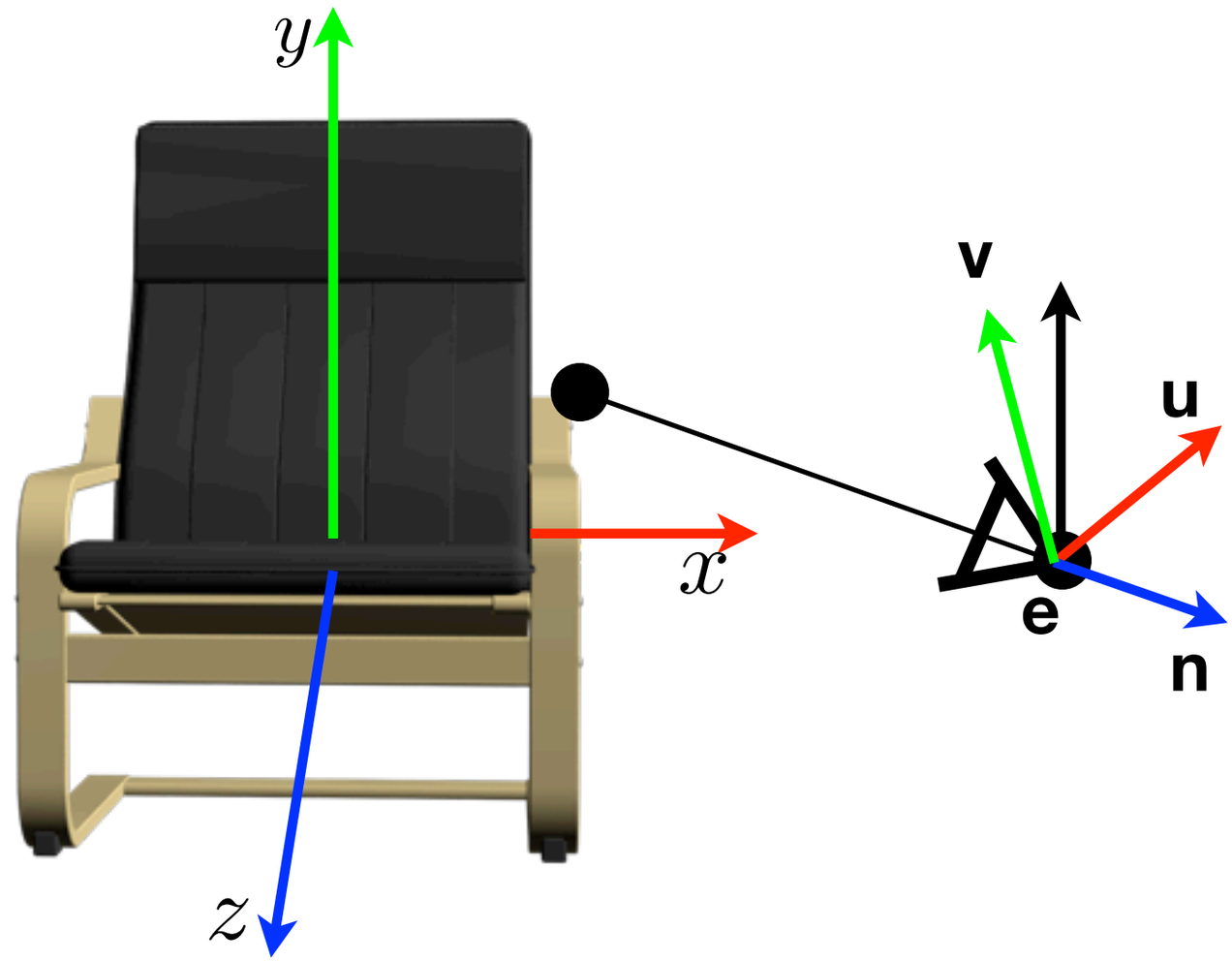
$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e **u** **v** **n**

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$



$$\mathbf{M}_{\text{camtrans}} \qquad \mathbf{M}_{\text{camrot}}$$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

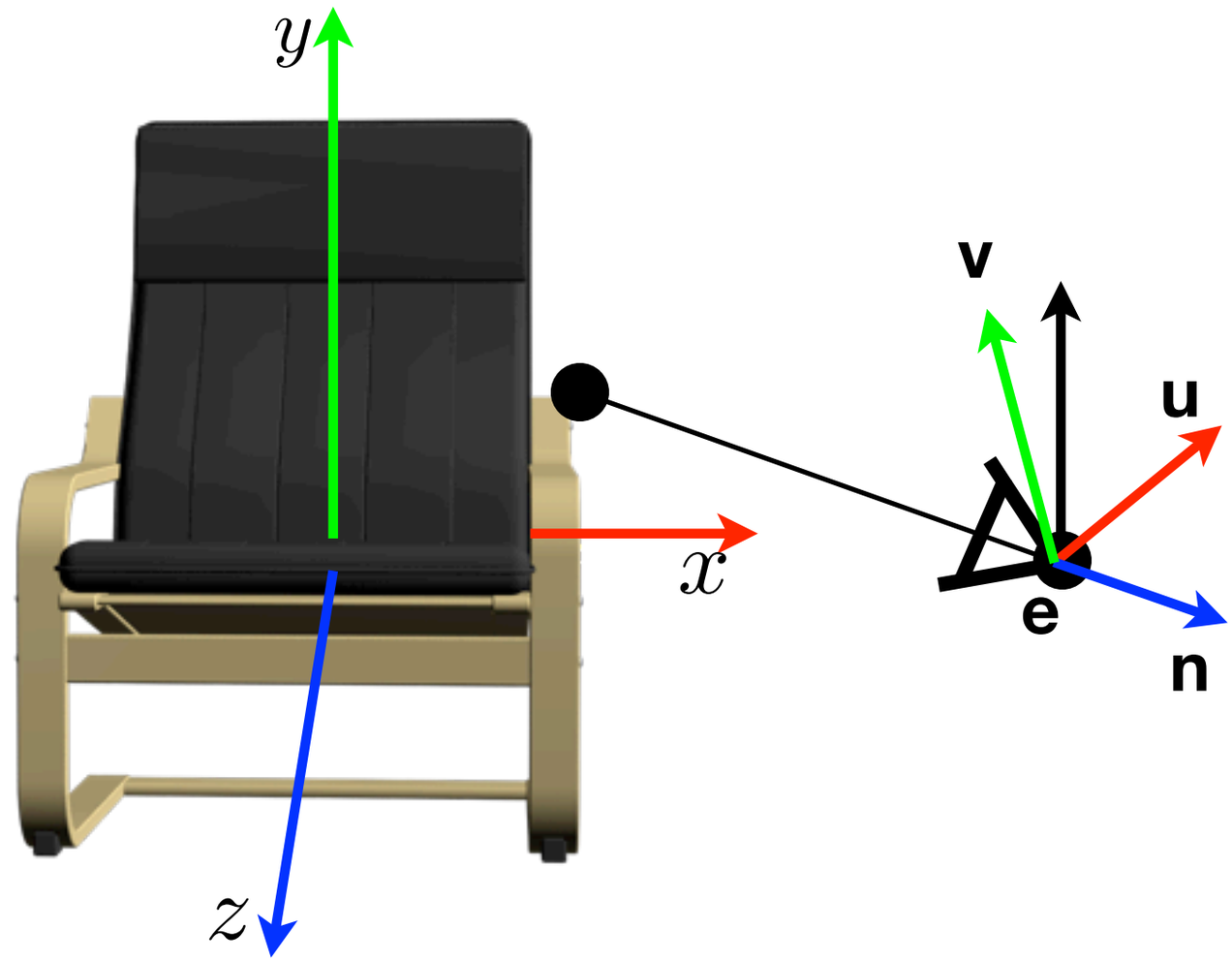
Inverse Model-view Transformation for Camera

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$



$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}_{\text{camtrans}} \qquad \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}_{\text{camrot}}$$

Inverse Model-view Transformation for Camera

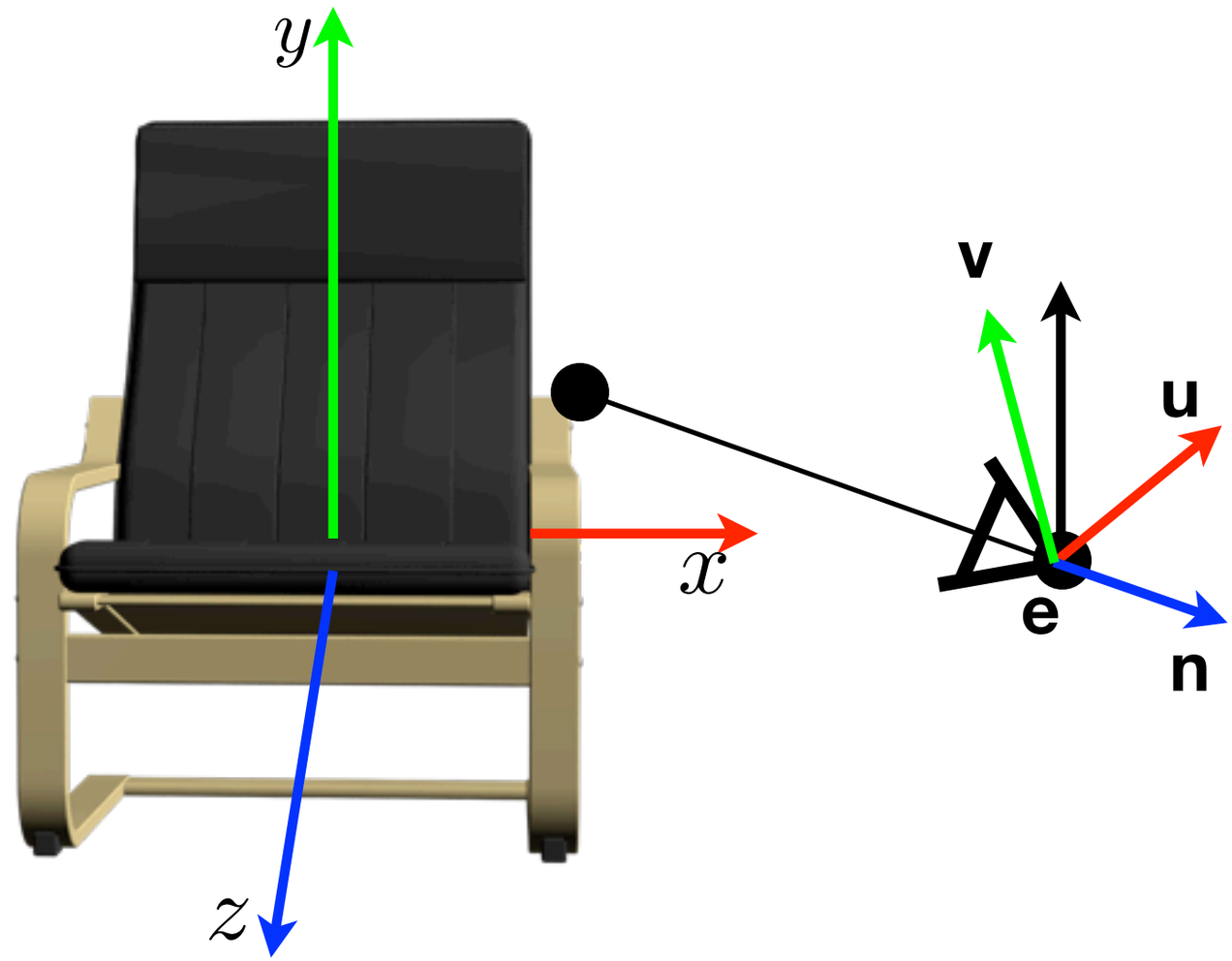
Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$

$$= (\mathbf{M}_{\text{camrot}})^{-1}(\mathbf{M}_{\text{camtrans}})^{-1}$$



$$\mathbf{M}_{\text{camtrans}} \qquad \mathbf{M}_{\text{camrot}}$$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Model-view Transformation for Camera

Look-At Approach

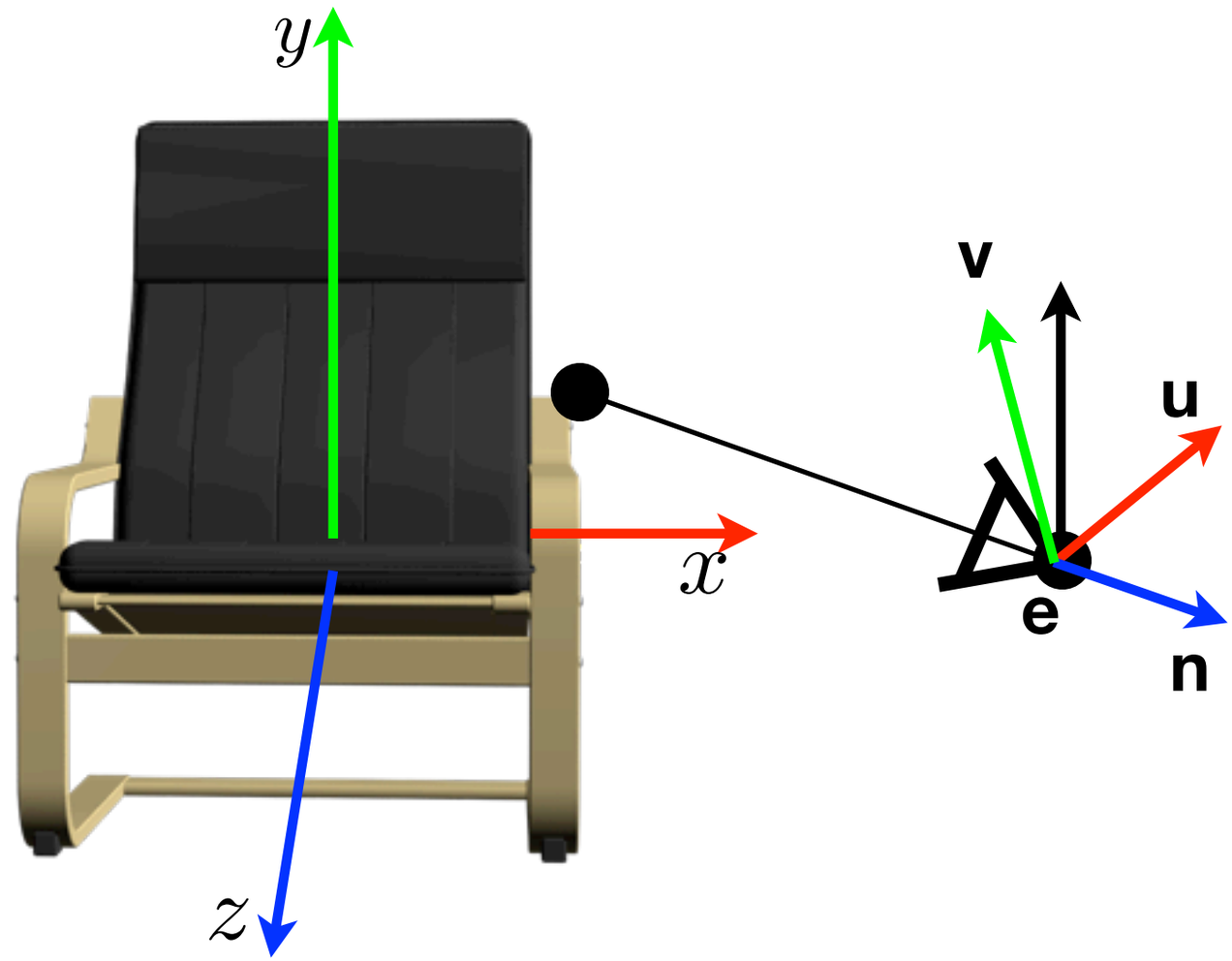
Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$

$$= (\mathbf{M}_{\text{camrot}})^{-1}(\mathbf{M}_{\text{camtrans}})^{-1}$$

$$= \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$



$\mathbf{M}_{\text{camtrans}}$ $\mathbf{M}_{\text{camrot}}$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Model-view Transformation for Camera

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

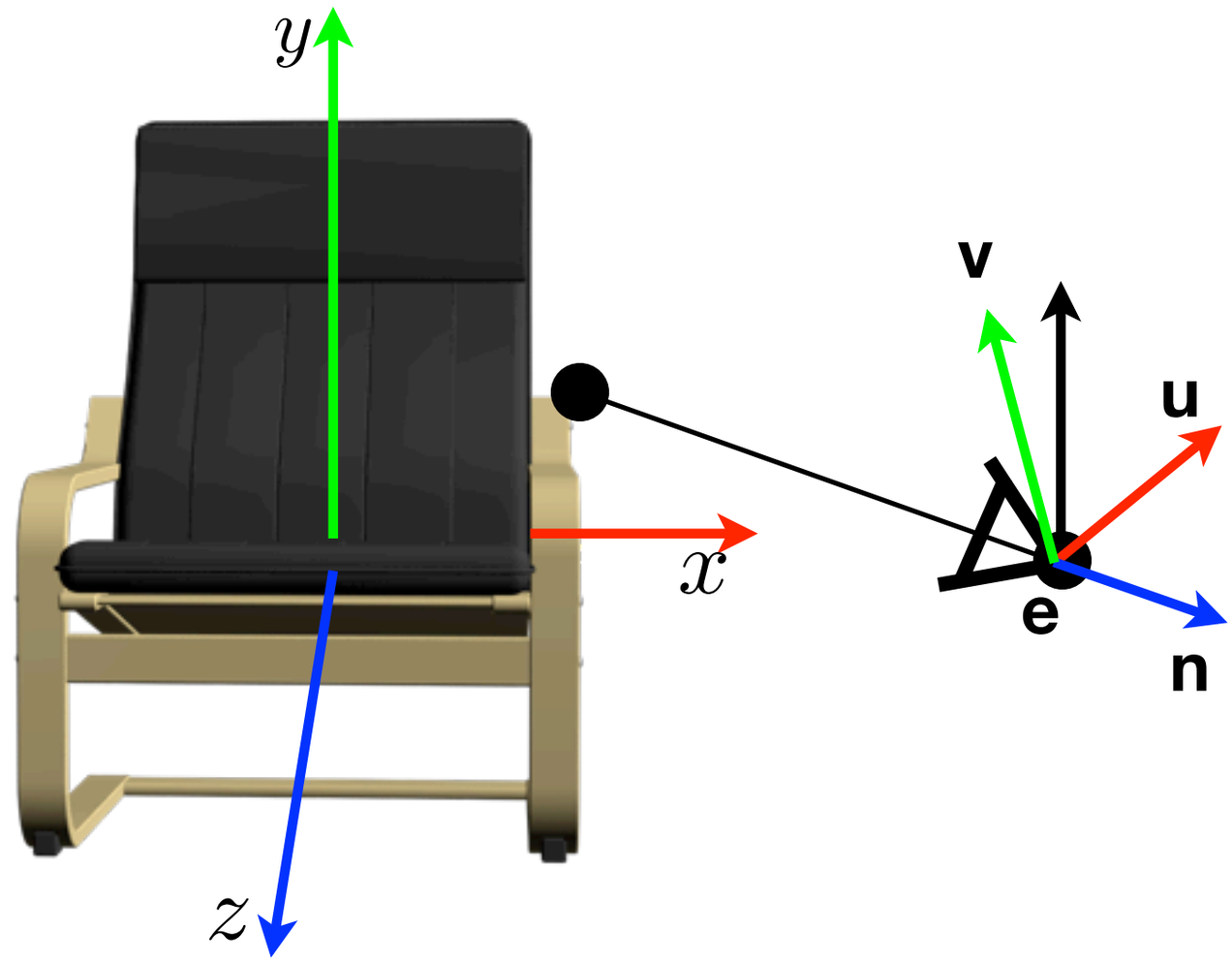
$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$

$$= (\mathbf{M}_{\text{camrot}})^{-1}(\mathbf{M}_{\text{camtrans}})^{-1}$$

$$= \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$



$\mathbf{M}_{\text{camtrans}}$ $\mathbf{M}_{\text{camrot}}$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Model-view Transformation for Camera

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{inv})^{-1}$

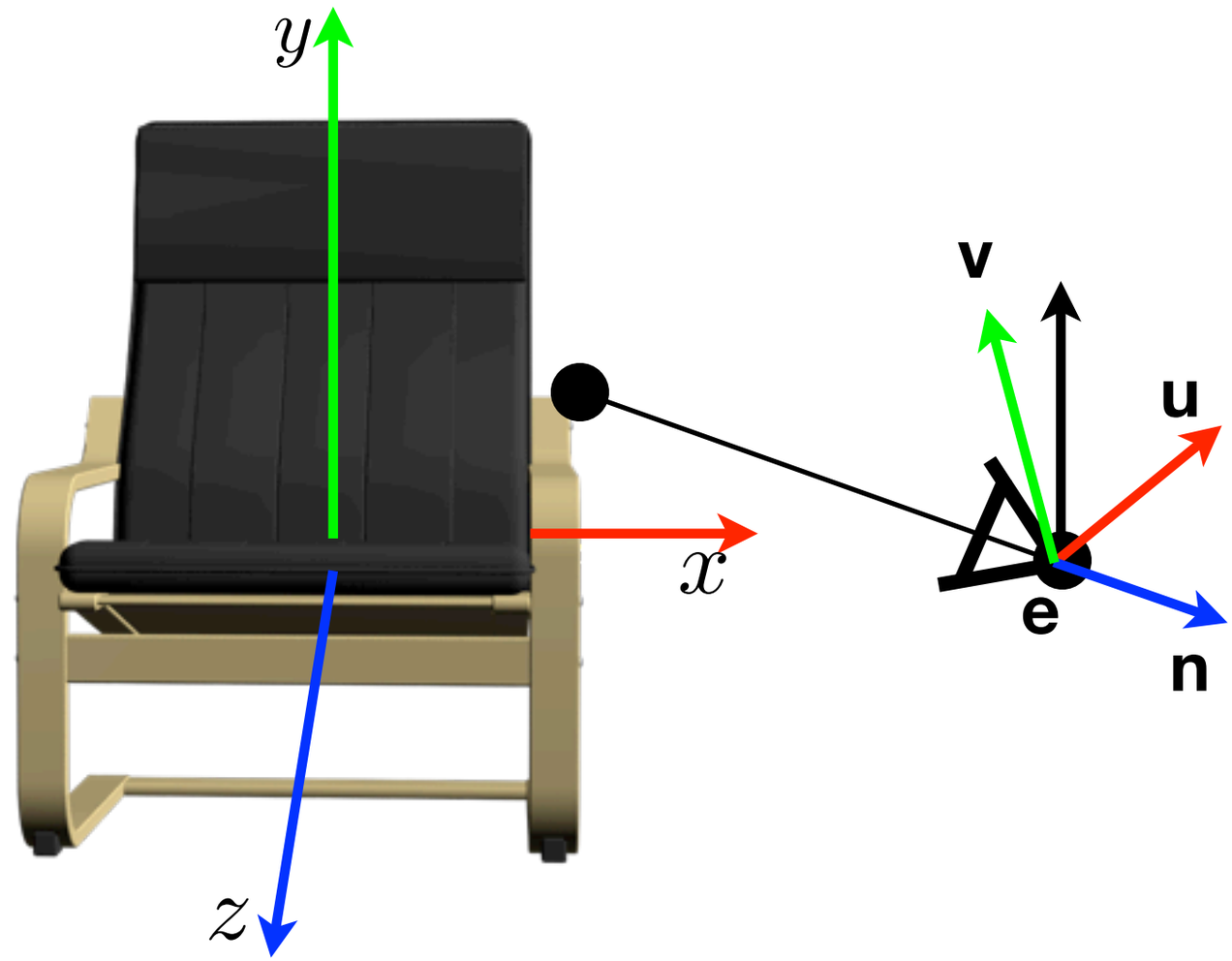
$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$

$$= (\mathbf{M}_{\text{camrot}})^{-1}(\mathbf{M}_{\text{camtrans}})^{-1}$$

$$= \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$\mathbf{M}_{\text{camtrans}}$ $\mathbf{M}_{\text{camrot}}$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Model-view Transformation for Camera

Look-At Approach

Model-view transformation for object $\mathbf{M} = (\mathbf{M}_{\text{inv}})^{-1}$

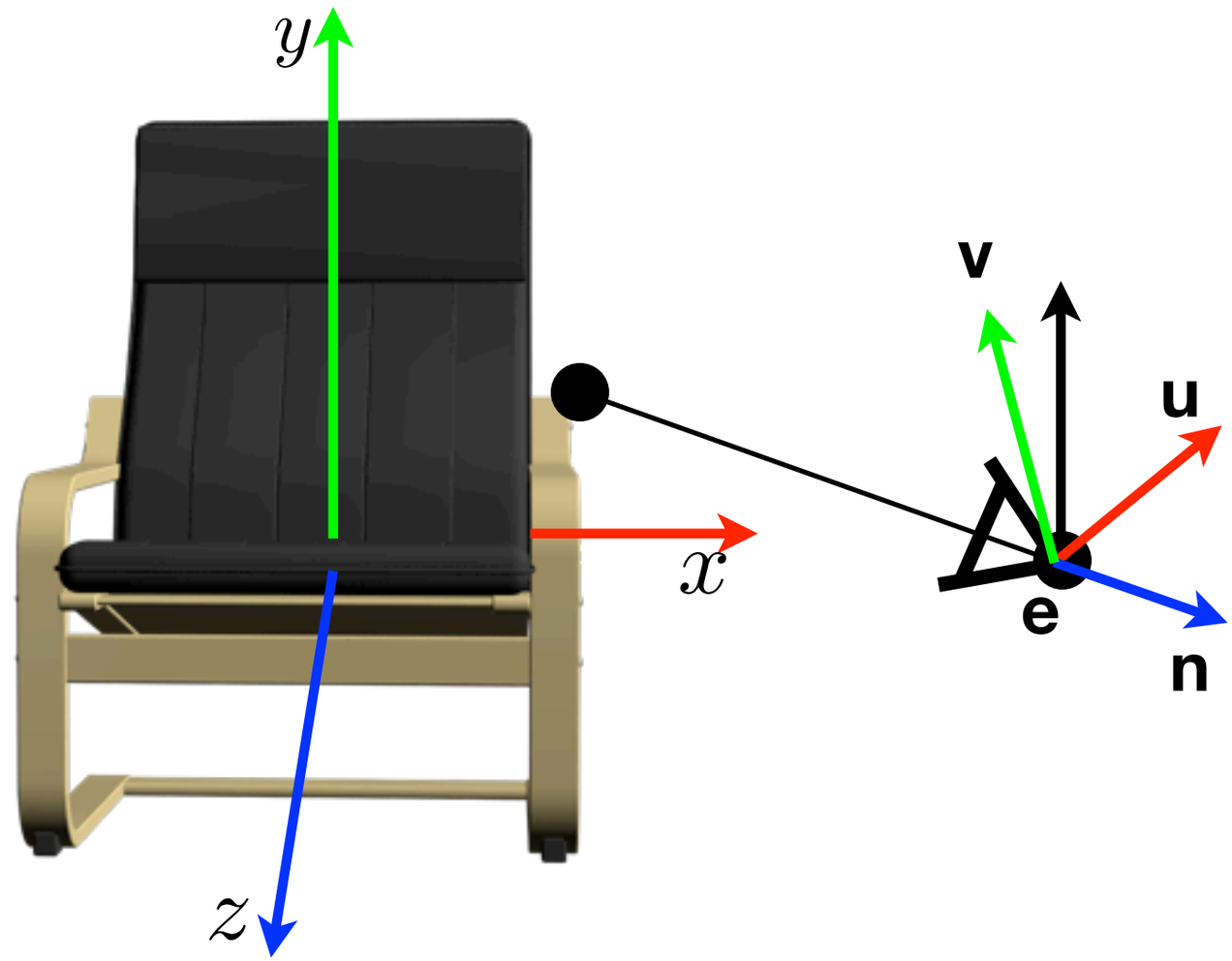
$$\mathbf{M}_{\text{inv}} = \mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}}$$

$$\mathbf{M} = (\mathbf{M}_{\text{camtrans}}\mathbf{M}_{\text{camrot}})^{-1}$$

$$= (\mathbf{M}_{\text{camrot}})^{-1}(\mathbf{M}_{\text{camtrans}})^{-1}$$

$$= \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$\mathbf{M}_{\text{camtrans}}$ $\mathbf{M}_{\text{camrot}}$

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse Model-view Transformation for Camera

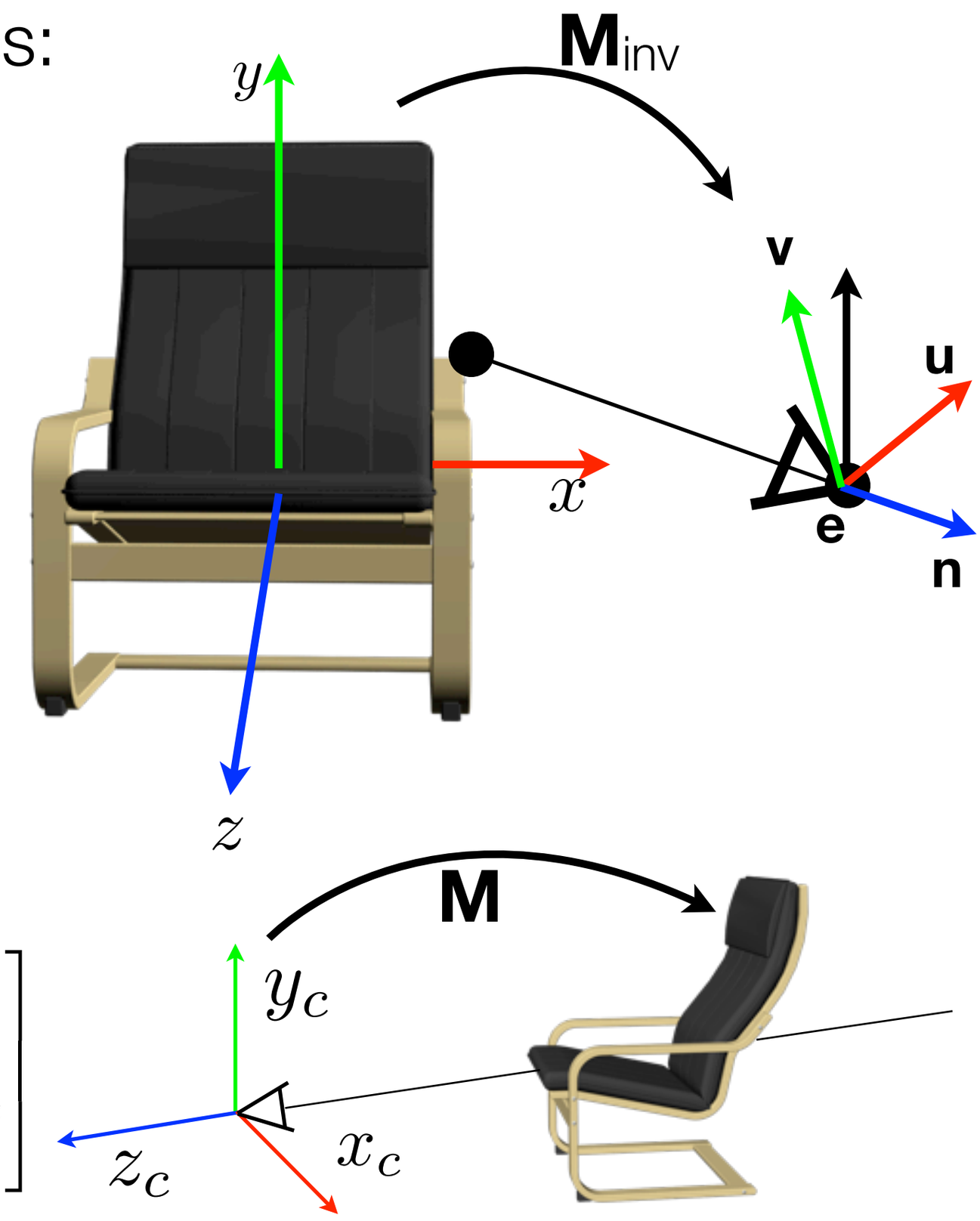
Look-At Approach

Compute in Object Coordinates:

- View plane normal, **n**
- View plane horizontal axis, **u**
- View plane vertical axis, **v**

On performing the matrix multiplication on the previous slide, we get the following result for **M**.

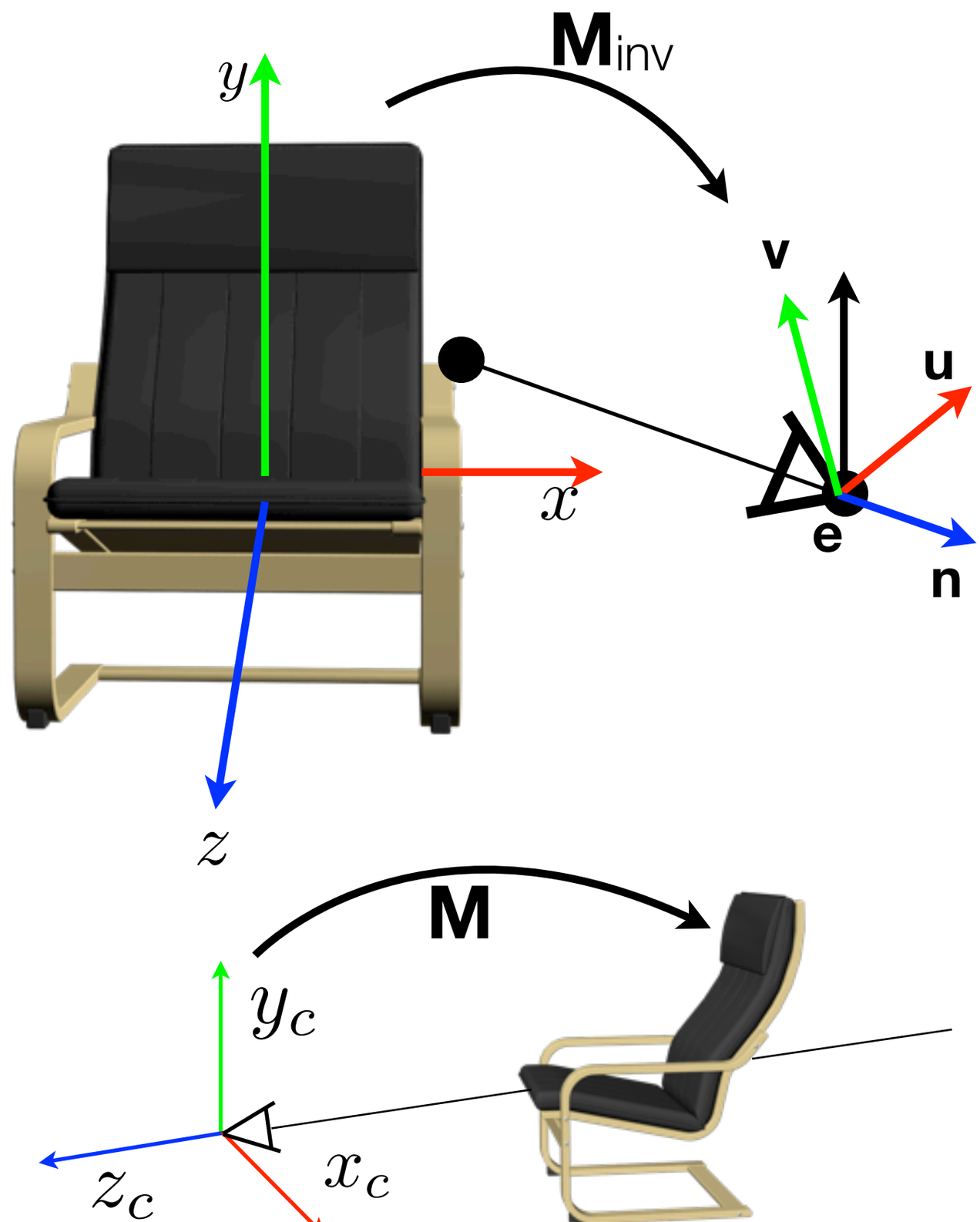
$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Look-At Approach

What about $\mathbf{M}_{\text{inv}}$?
What is its expression?

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

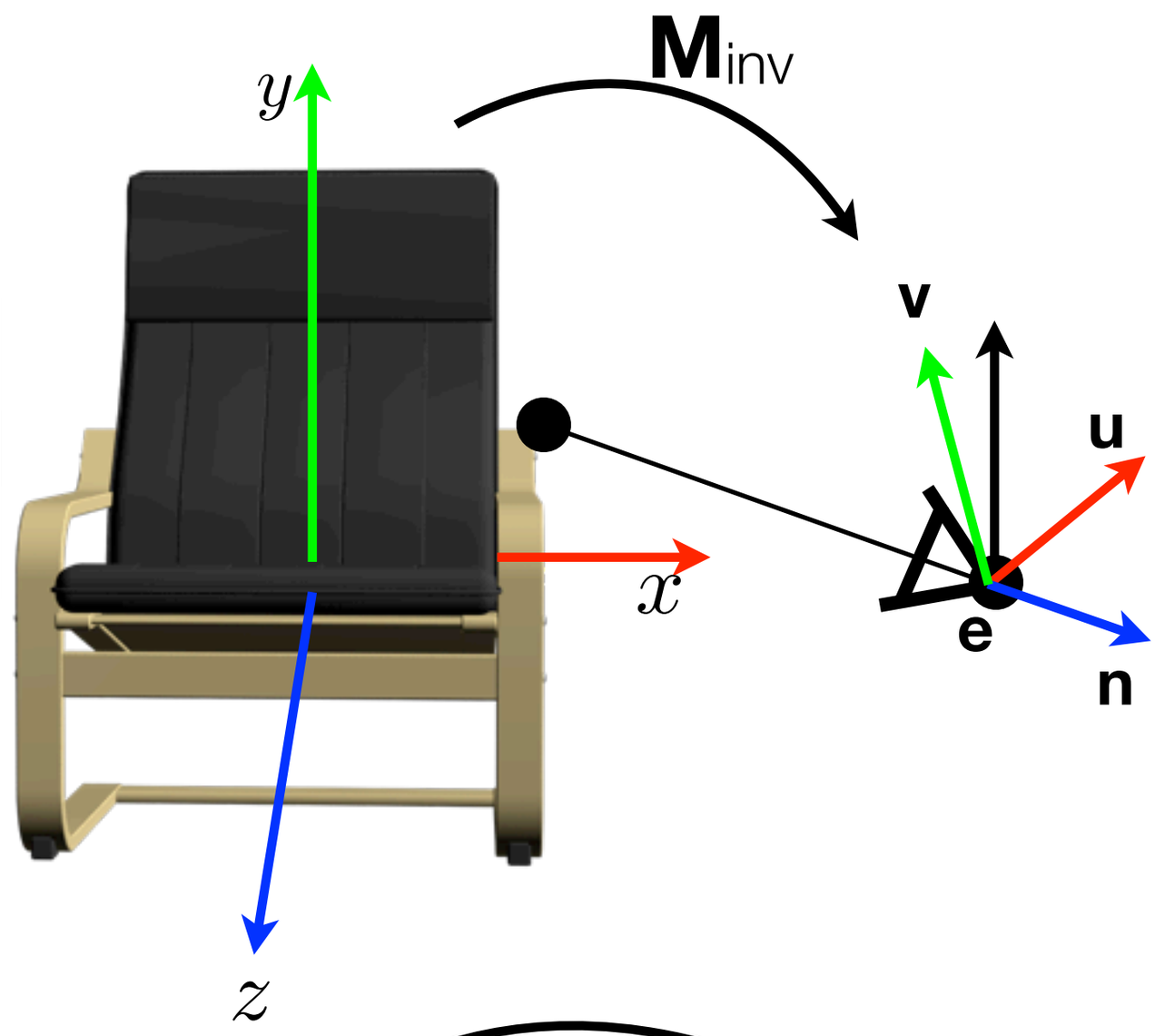


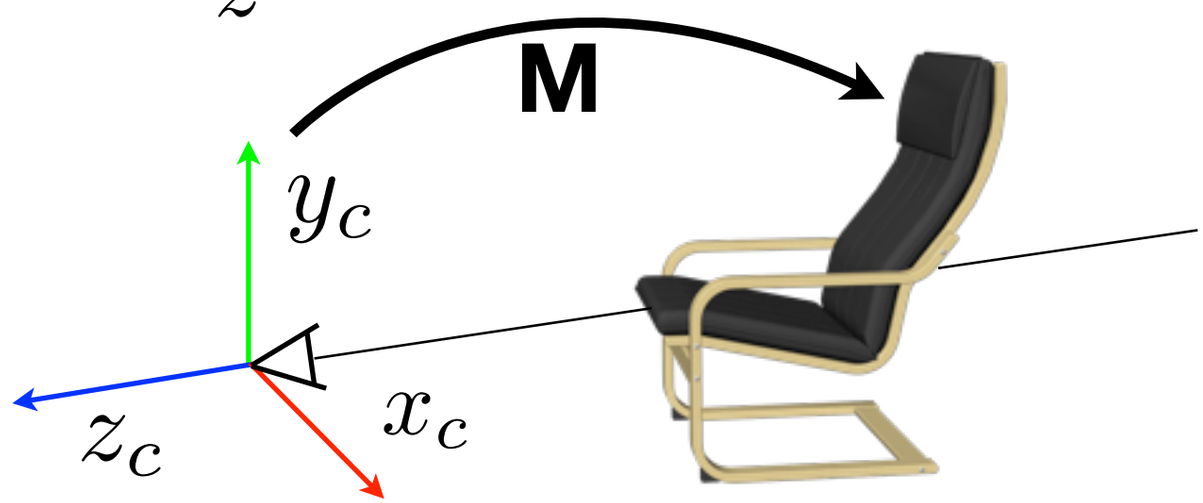
Look-At Approach

What about $\mathbf{M}_{\text{inv}}$?
What is its expression?

$$\mathbf{M}_{\text{inv}} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} u_x & v_x & n_x & e_x \\ u_y & v_y & n_y & e_y \\ u_z & v_z & n_z & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





Now let us get into the equations for projections themselves!

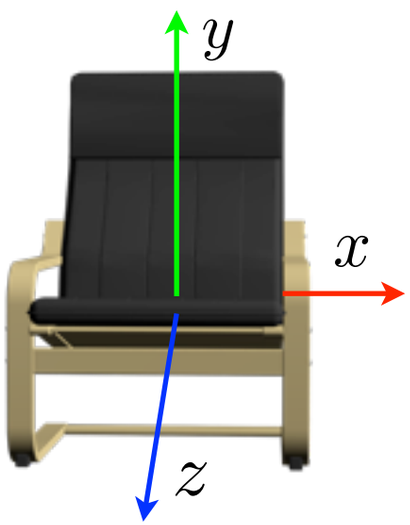
Projection Transformation

Model-view Matrix

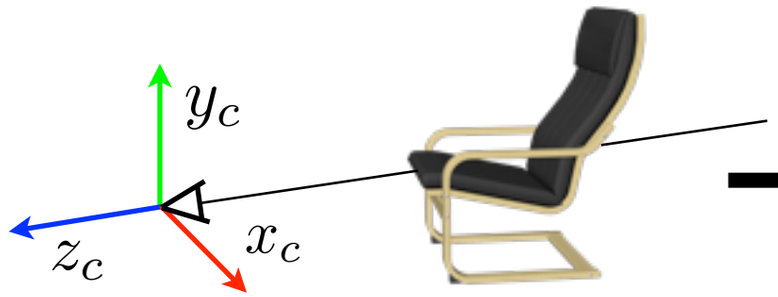
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Model-view Transformation

Projection Transformation



Object Coordinates



Camera Coordinates

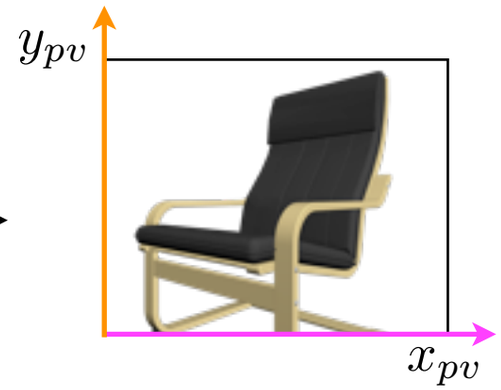


Image or Clip Coordinates

Projection Transformation

Model-view Matrix

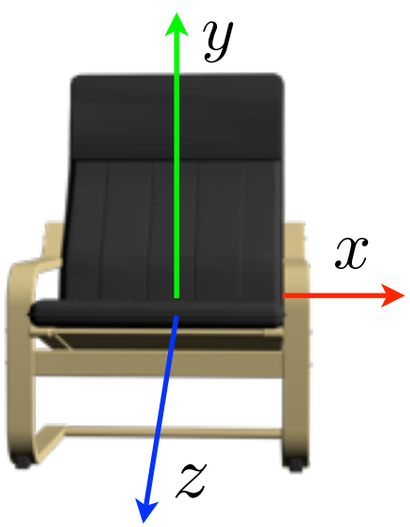
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Model-view Transformation

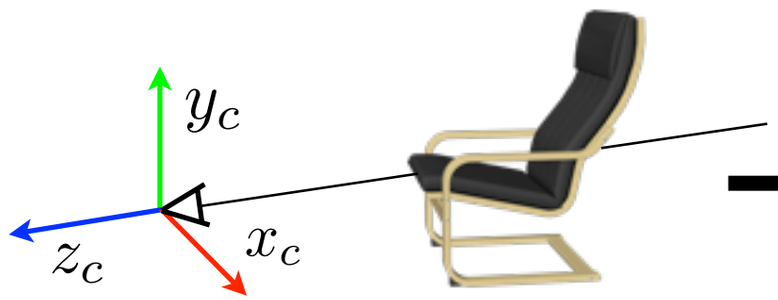
Projection

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Projection Transformation



Object Coordinates



Camera Coordinates

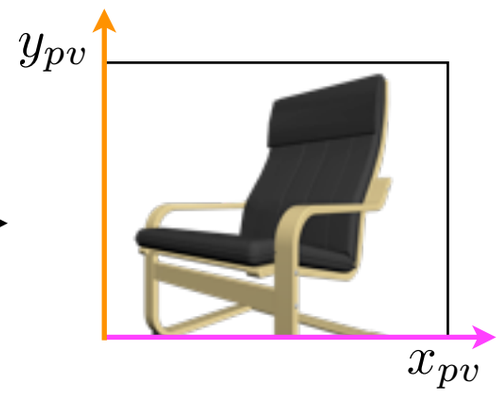


Image or Clip Coordinates

Projection Transformation

Model-view Matrix

Projection

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Used in display → x_{pv}, y_{pv}

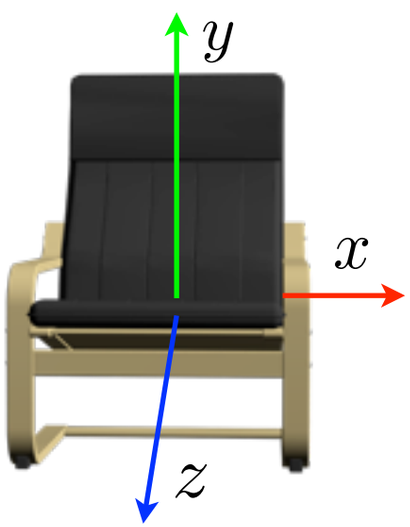
Used in depth test → z_{pv}

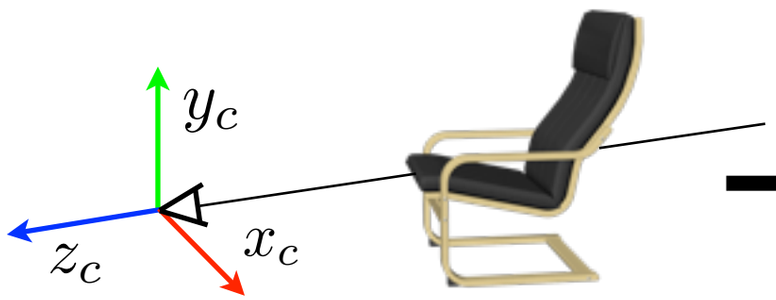
Used in perspective projection → w_{pv}

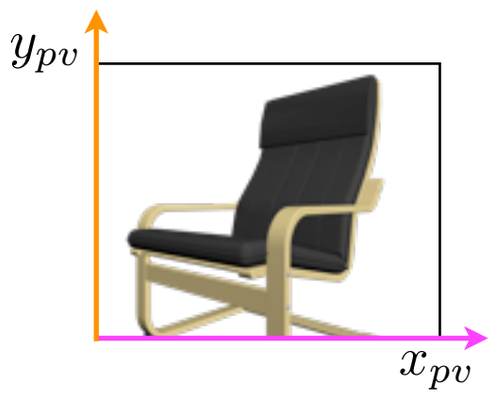
$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Model-view Transformation

Projection Transformation







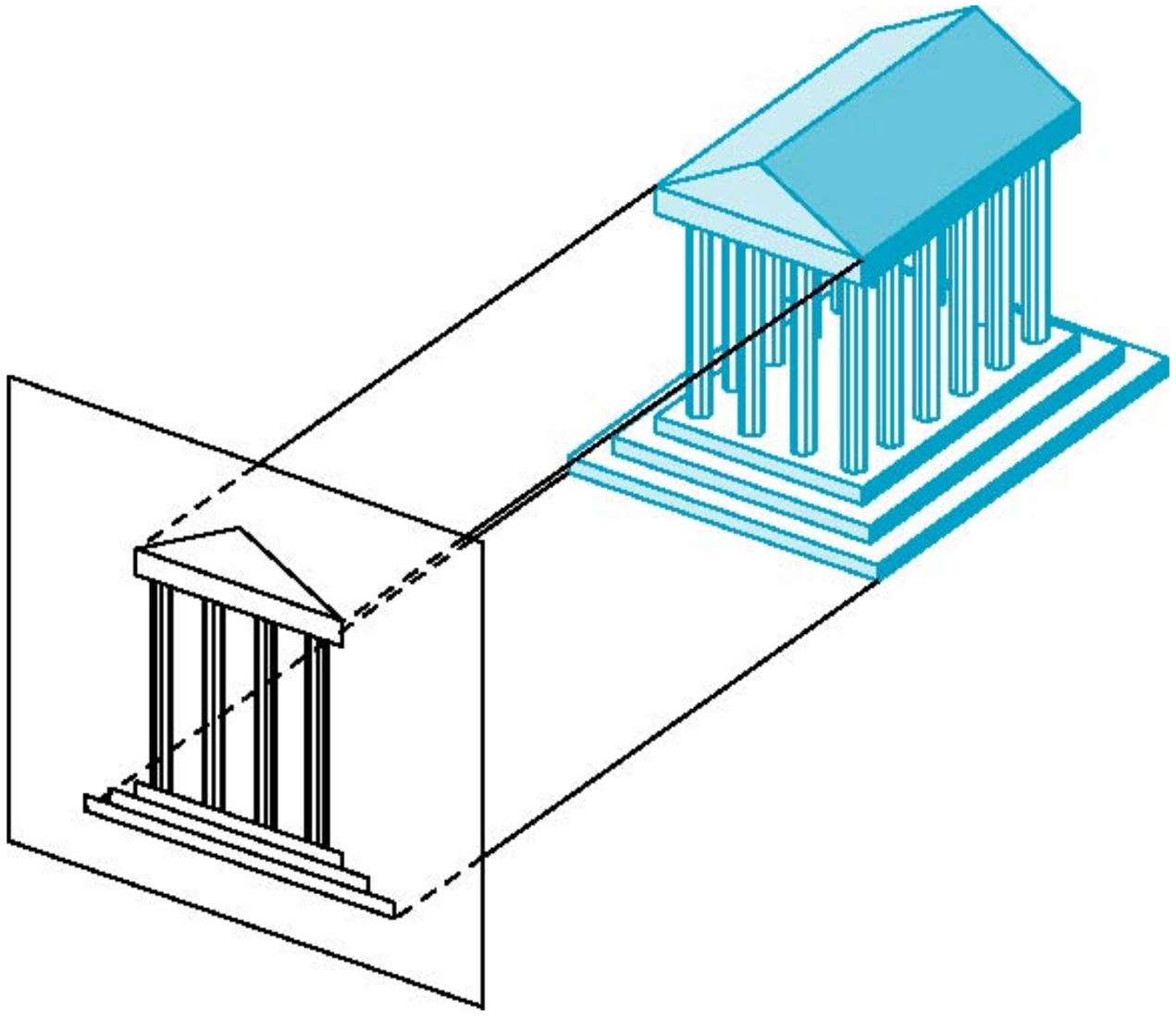
Object Coordinates

Camera Coordinates

Image or Clip Coordinates

We will start with parallel projections, in particular orthographic ones.

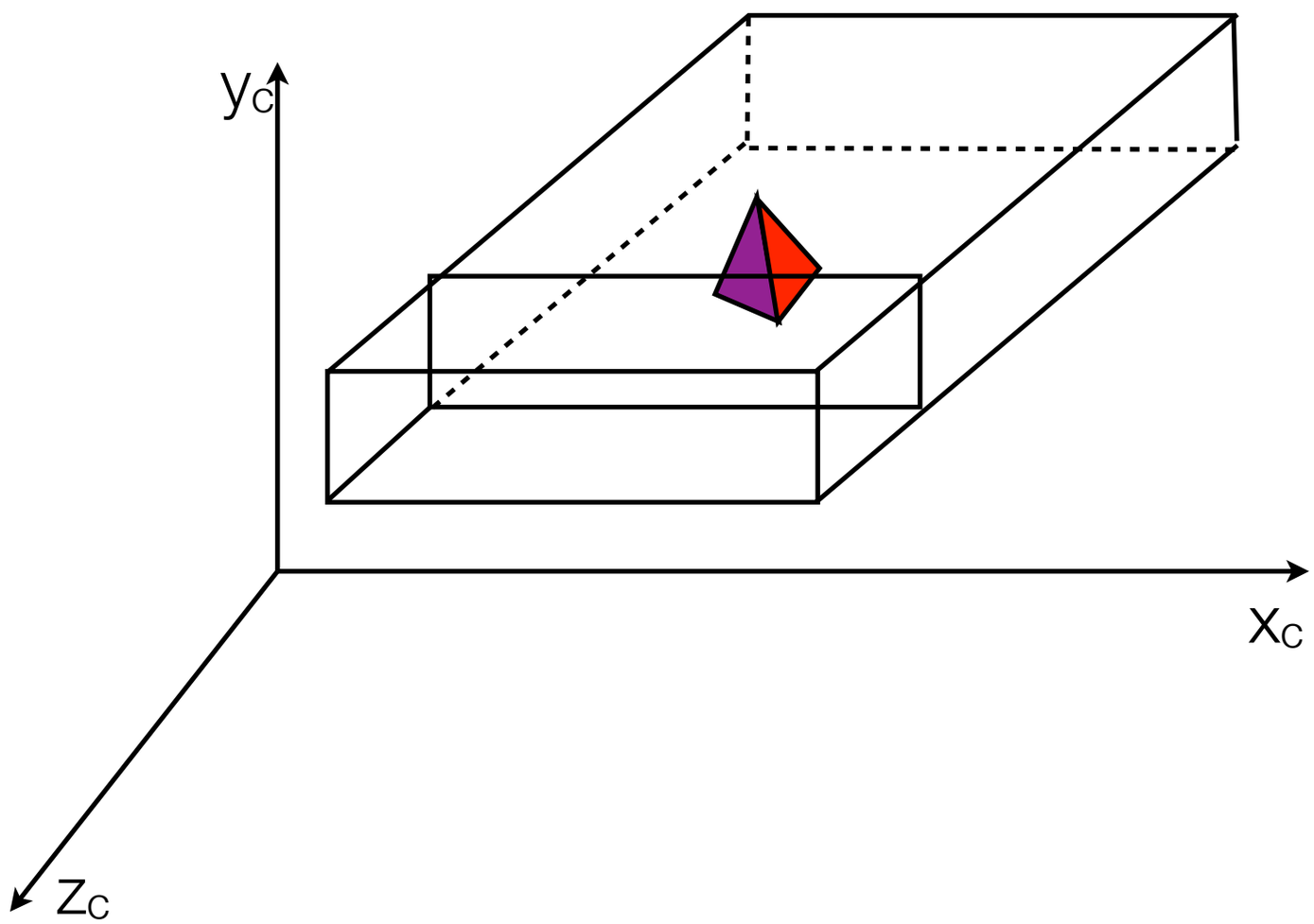
Parallel (Orthographic) Projections



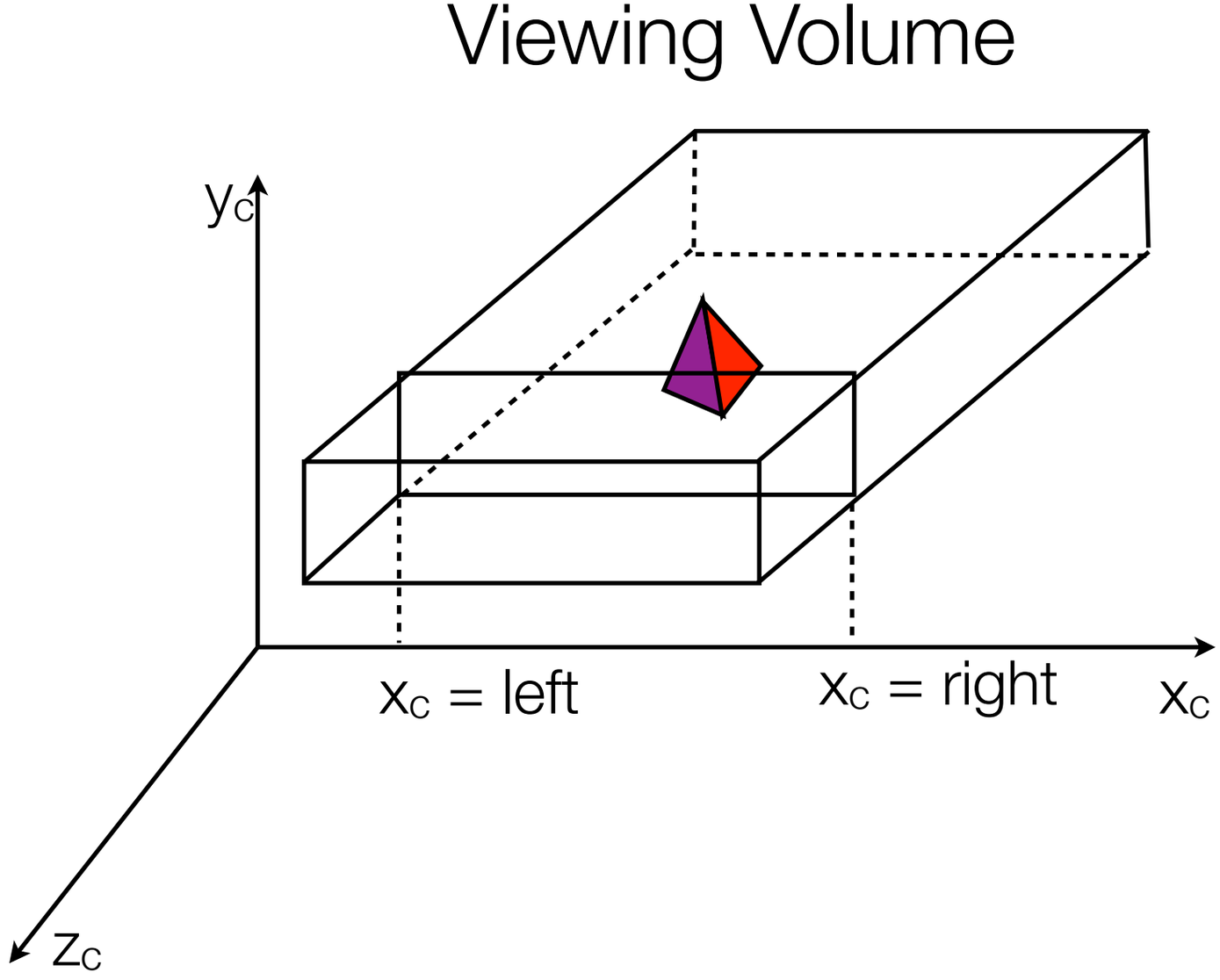
Parallel (Orthographic) Projections

We consider a **viewing volume** (a cuboid)

Viewing Volume

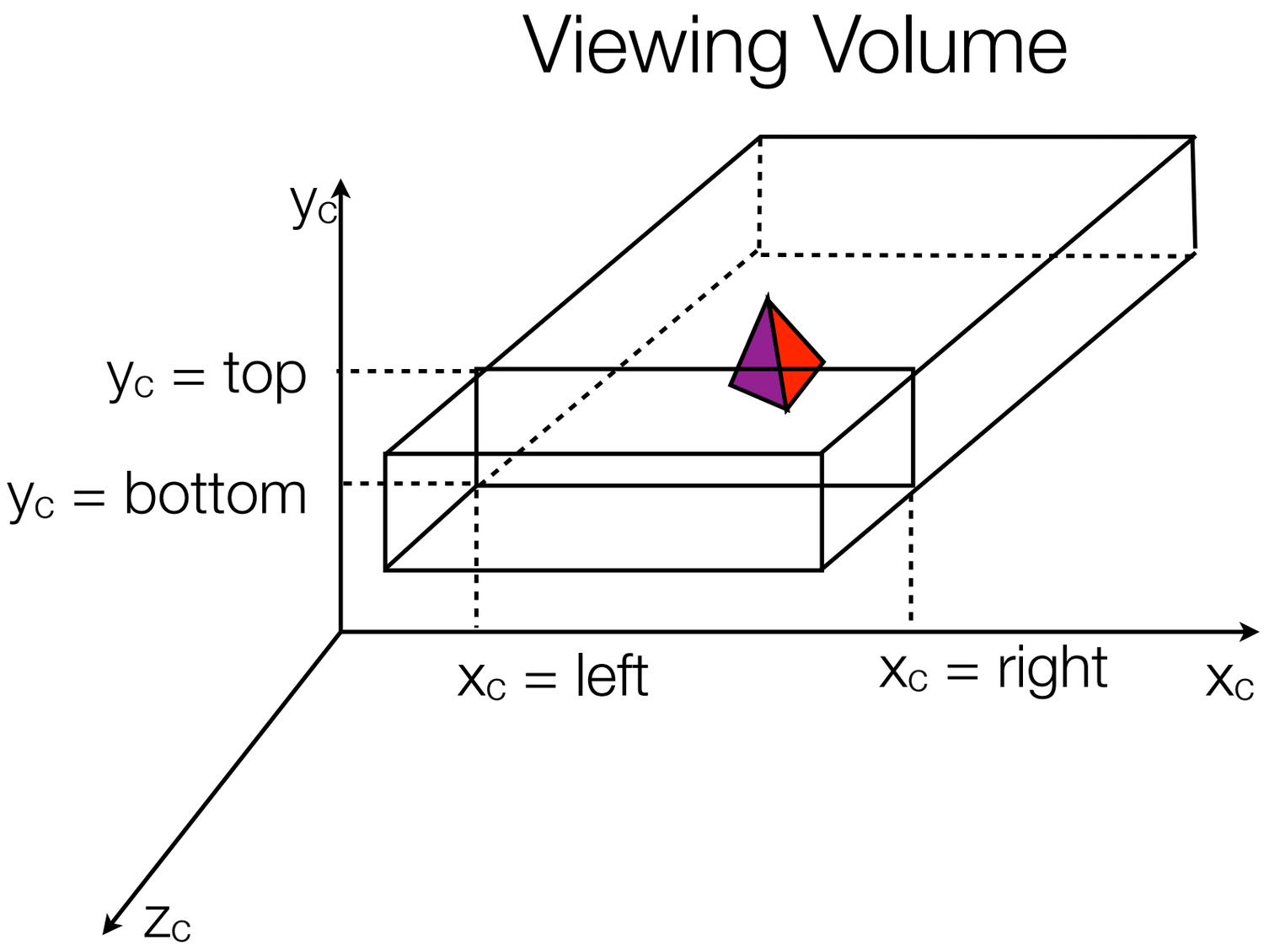


Parallel (Orthographic) Projections



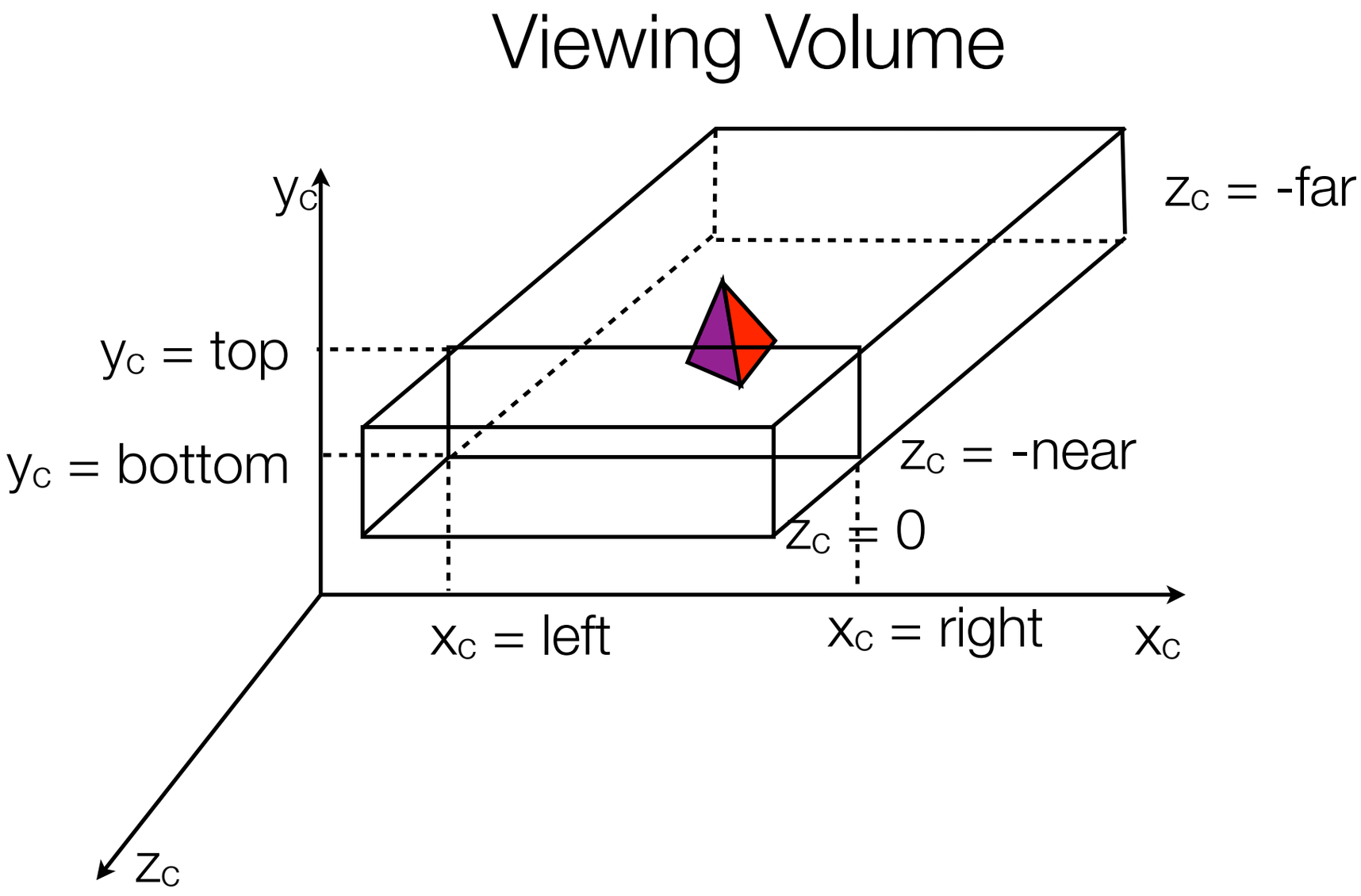
Left plane, right plane,

Parallel (Orthographic) Projections



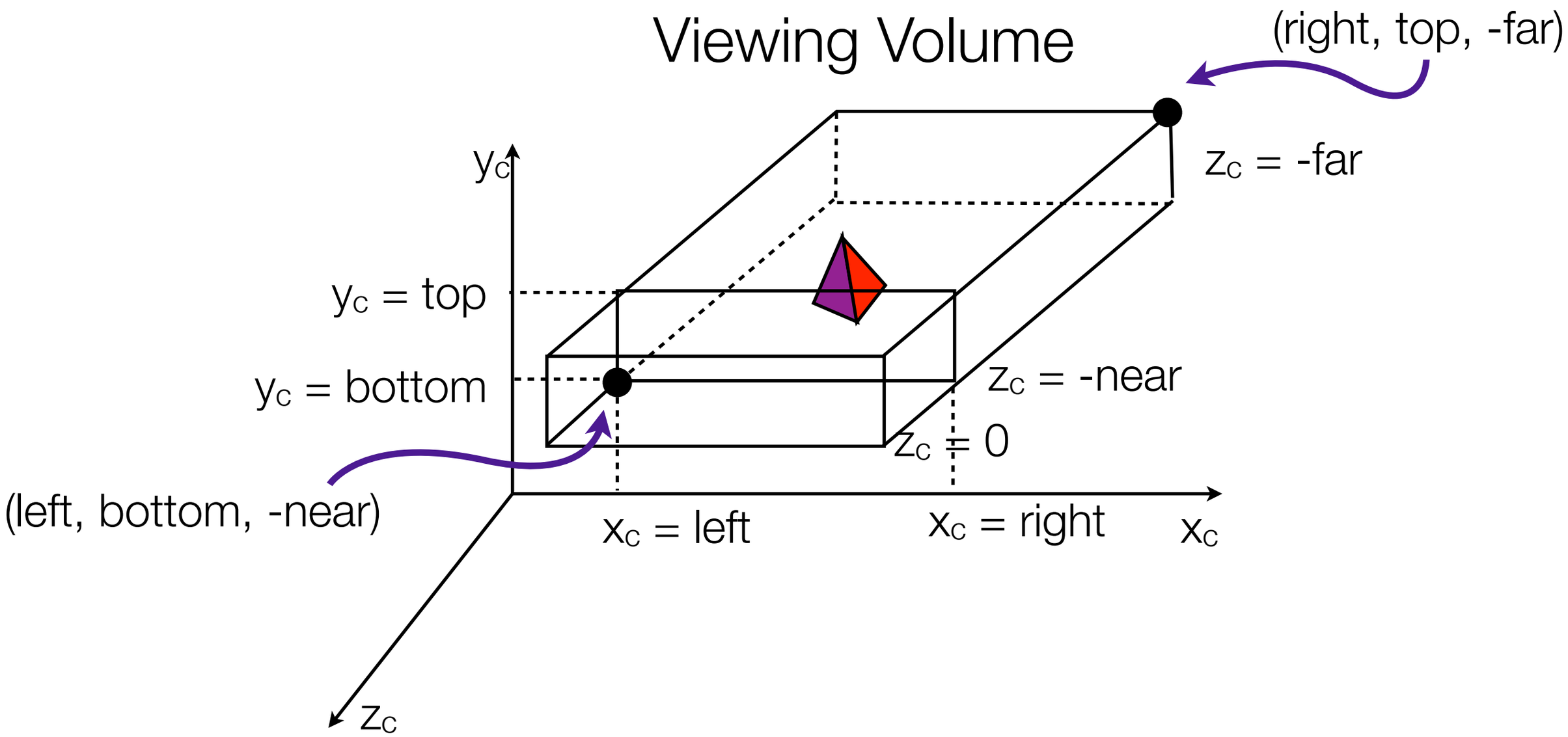
Left plane, right plane, top plane, bottom plane,

Parallel (Orthographic) Projections

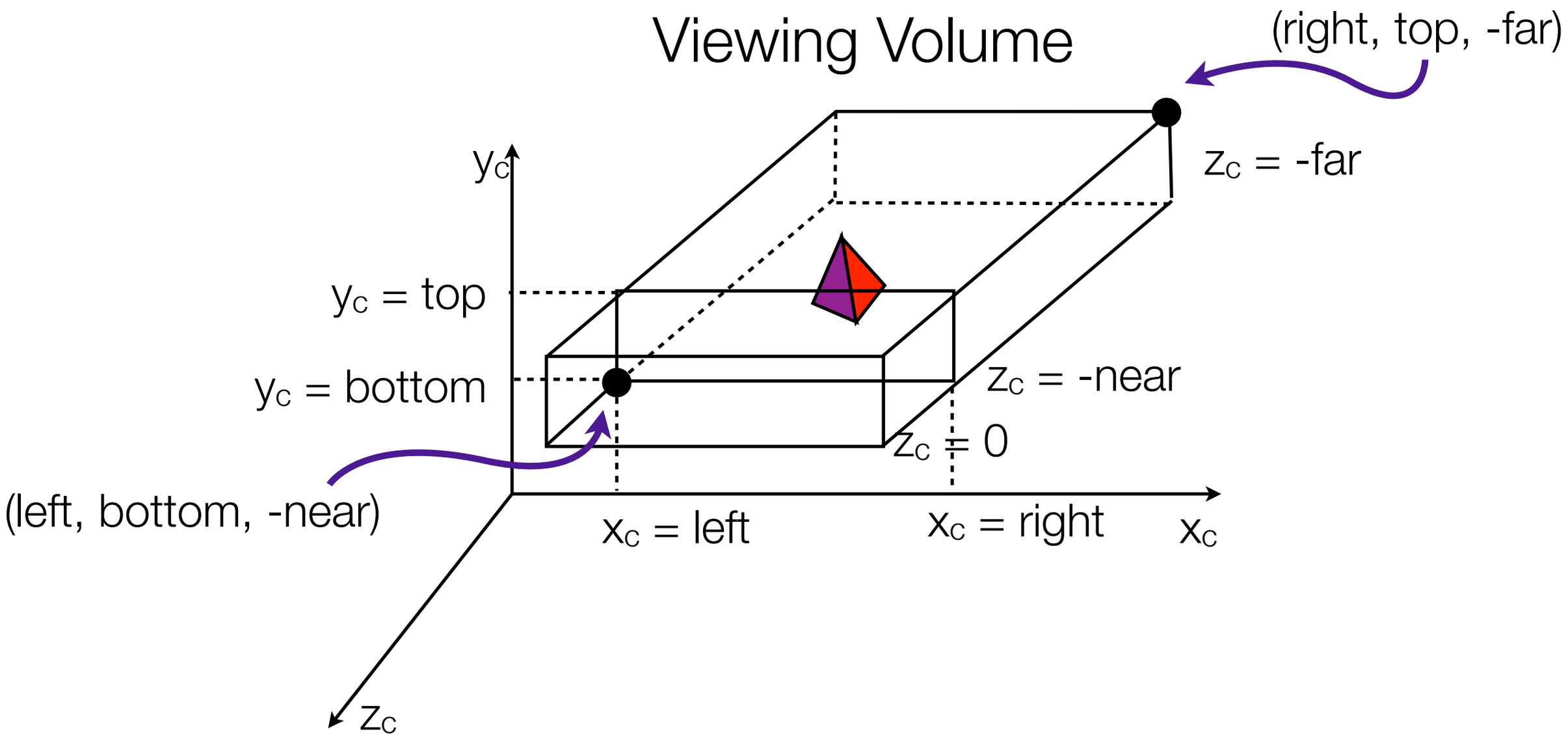


Left plane, right plane, top plane, bottom plane, near plane, far plane

Parallel (Orthographic) Projections

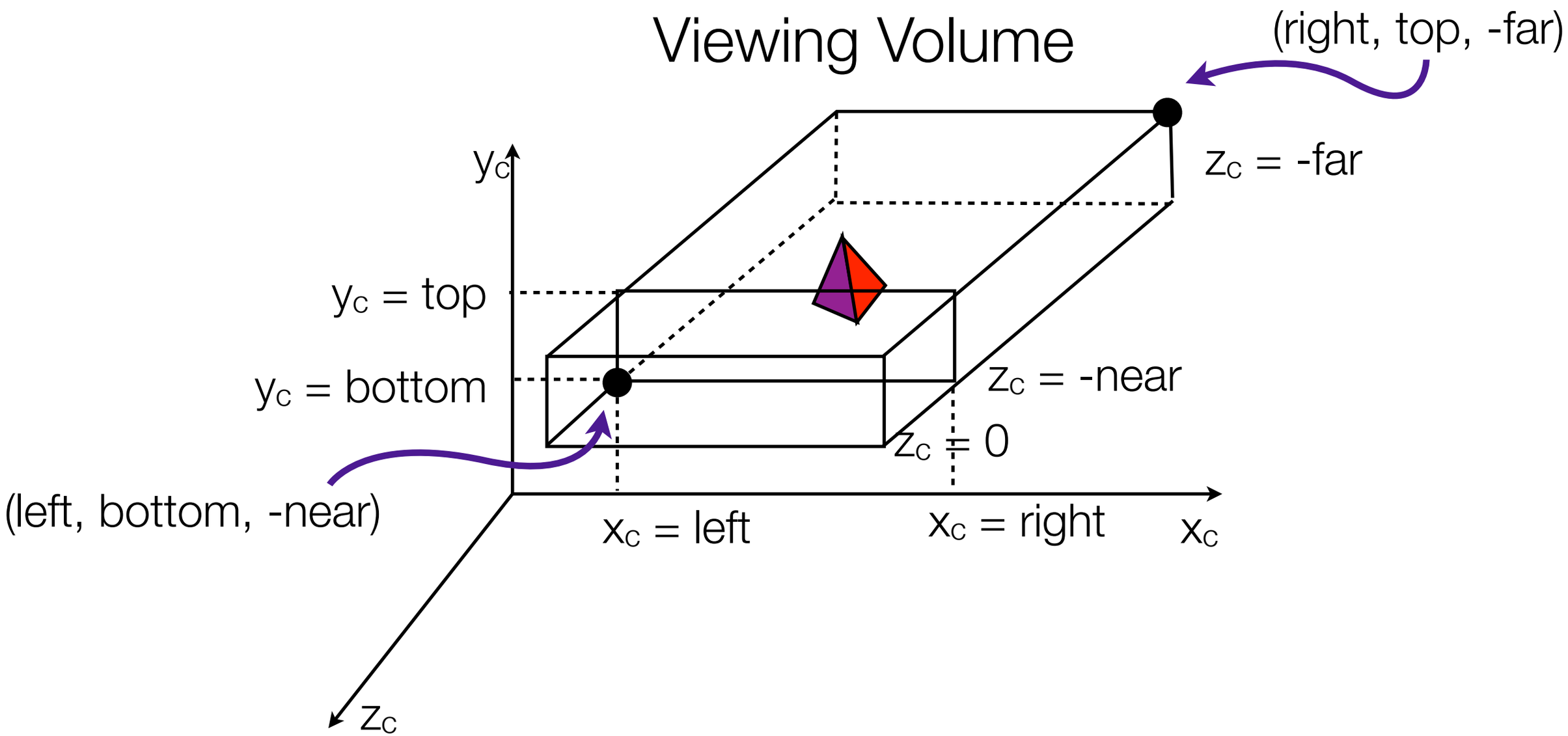


Parallel (Orthographic) Projections



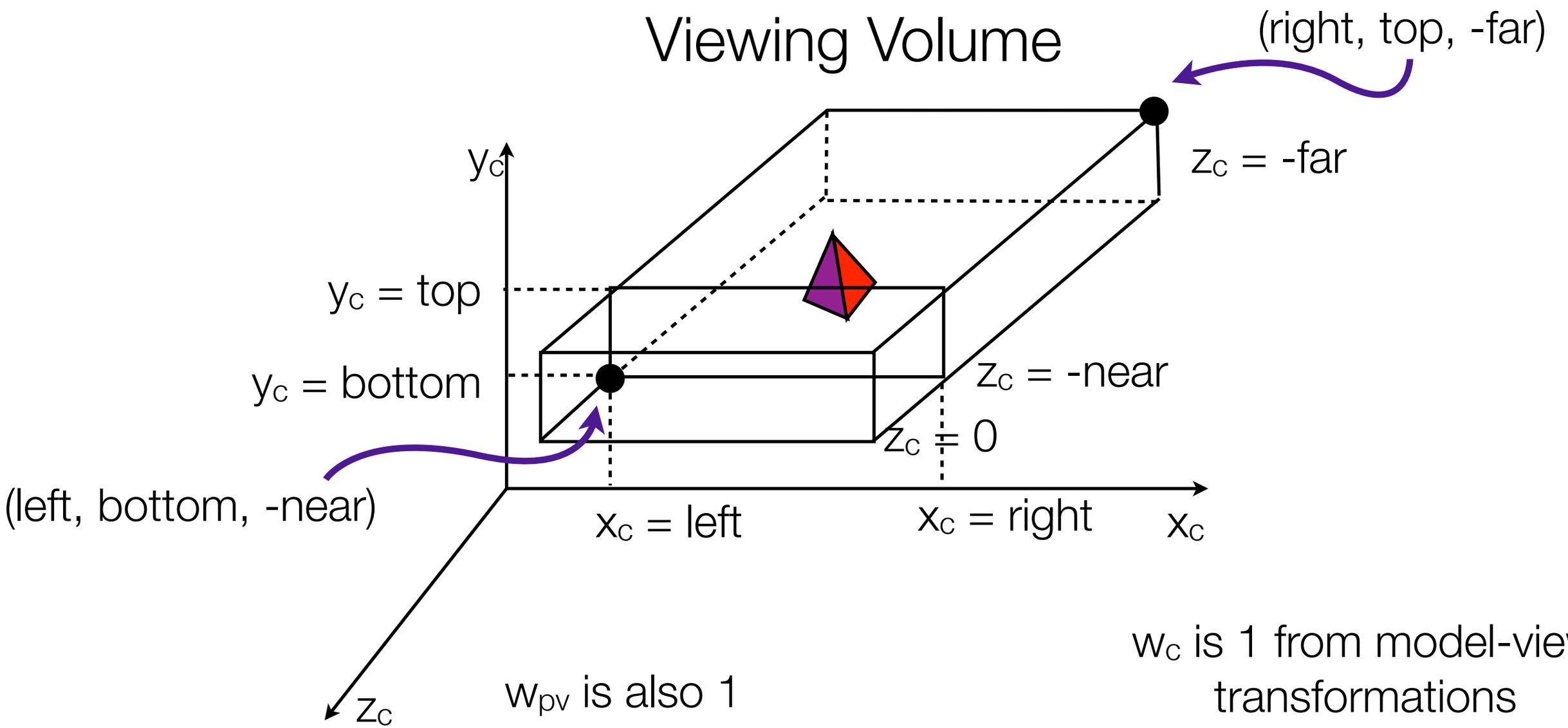
$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parallel (Orthographic) Projections



$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Parallel (Orthographic) Projections

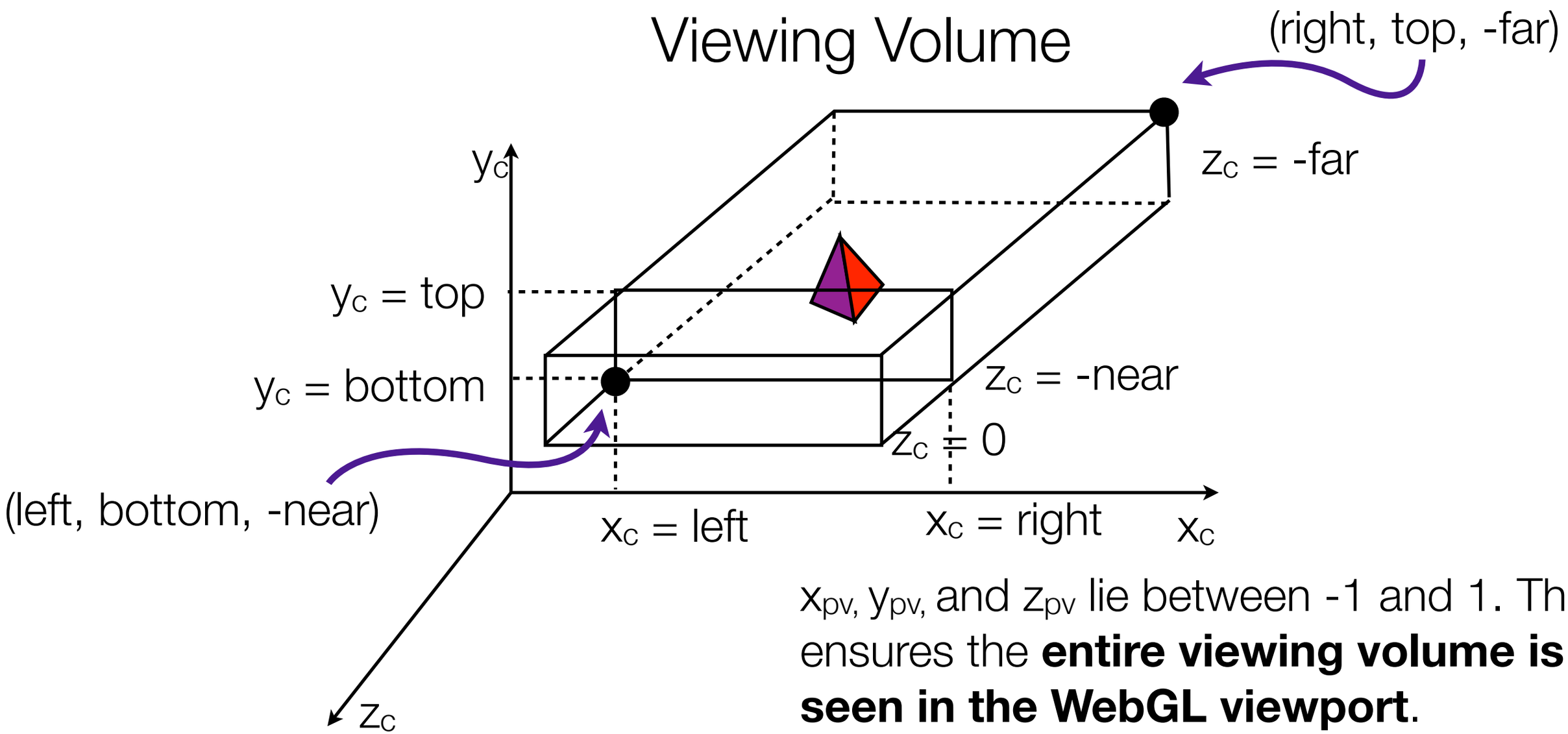


w_{pv} is also 1

w_c is 1 from model-view transformations

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

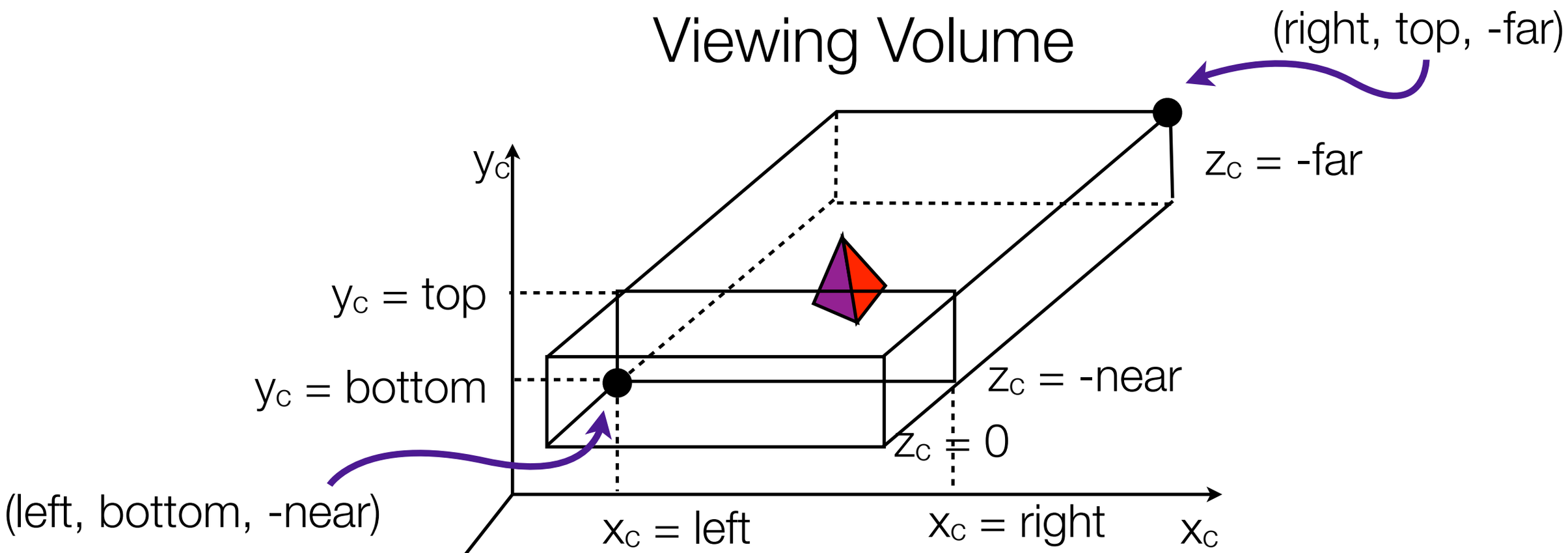
Parallel (Orthographic) Projections



x_{pv}, y_{pv}, and z_{pv} lie between -1 and 1. This ensures the **entire viewing volume is seen in the WebGL viewport**.

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Parallel (Orthographic) Projections

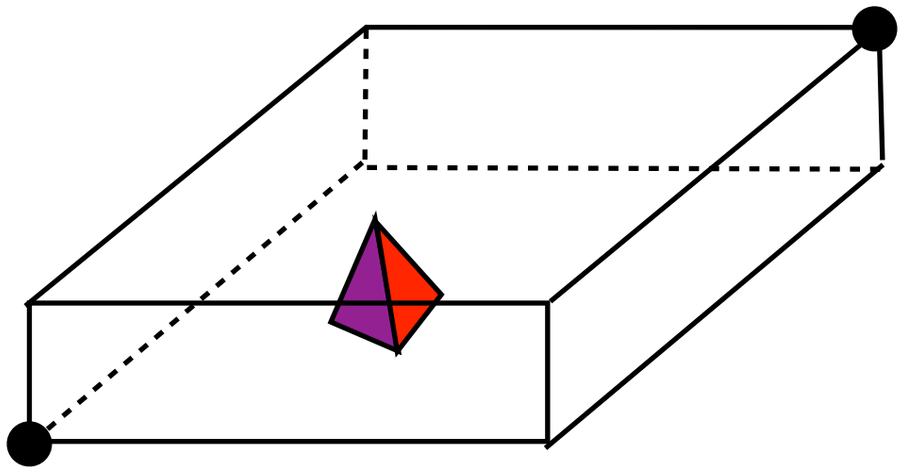


x_{pv}, y_{pv}, and z_{pv} lie between -1 and 1. This makes the orthogonal projection matrix a **normalizing transformation**.

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

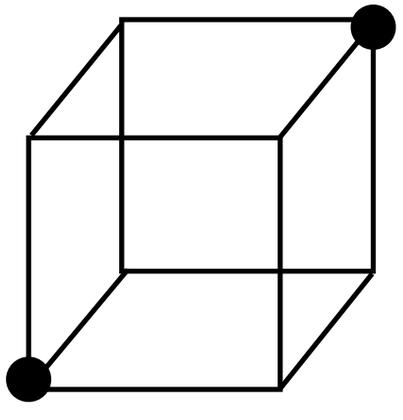
Understanding the Parallel Projection Effect in WebGL

(right, top, -far)



(left, bottom, -near)

Camera Coordinates



Clip Coordinates

Understanding the Parallel Projection Effect in WebGL

left can be +ve or -ve

right can be +ve or -ve

top can be +ve or -ve

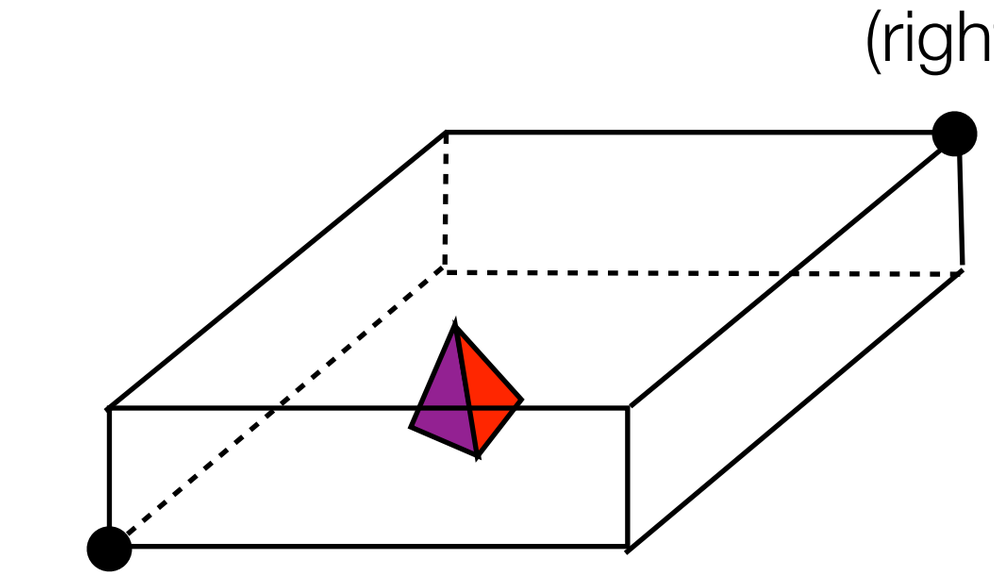
bottom can be +ve or -ve

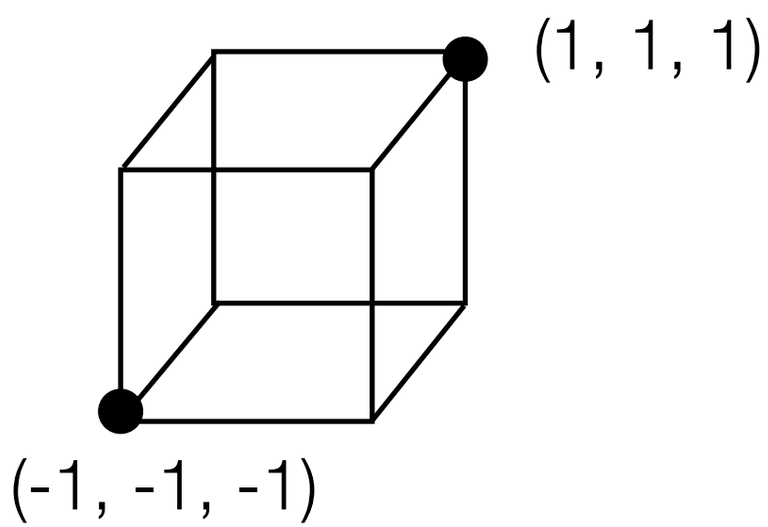
(-near) must be -ve

near must be +ve

(-far) must be -ve

far must be +ve



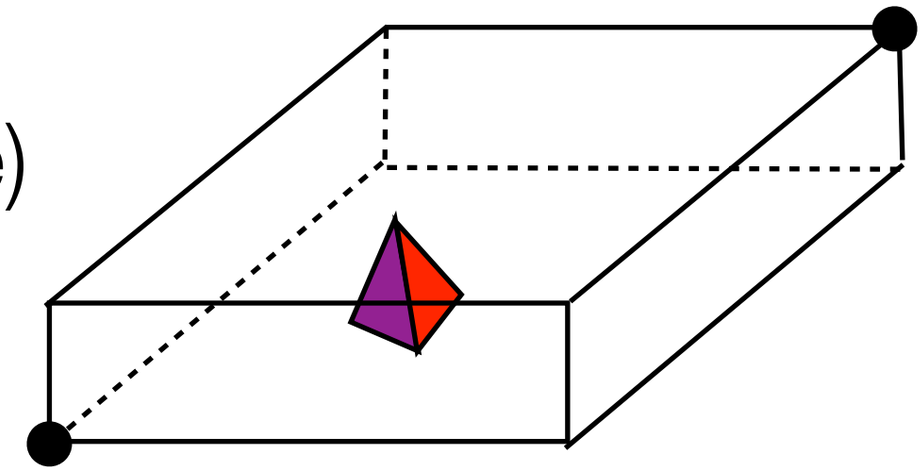


Camera Coordinates

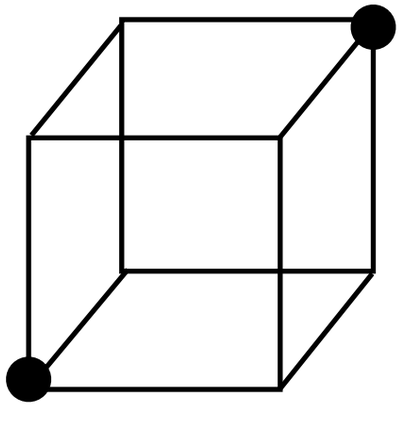
Clip Coordinates

Understanding the Parallel Projection Effect in WebGL

Right-handed
coordinate system
(positive z out of slide)
(negative z into slide)



Left-handed
coordinate system
(positive z into slide)
(negative z out of slide)

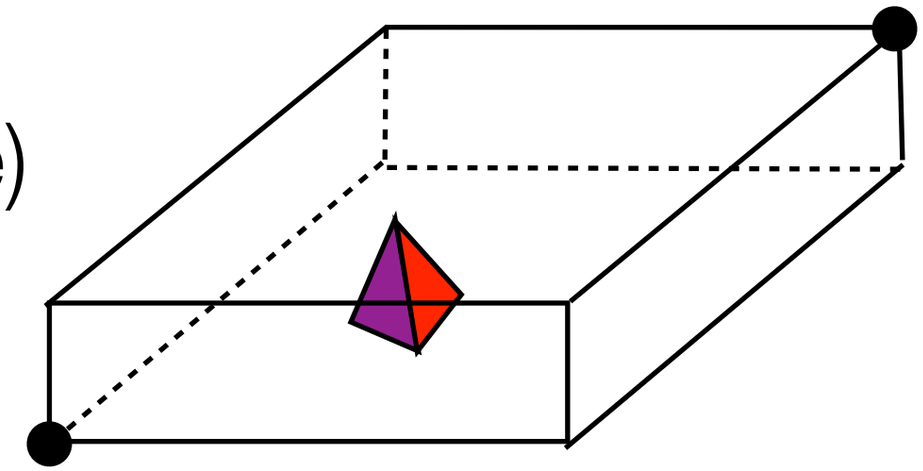


Camera
Coordinates

Clip
Coordinates

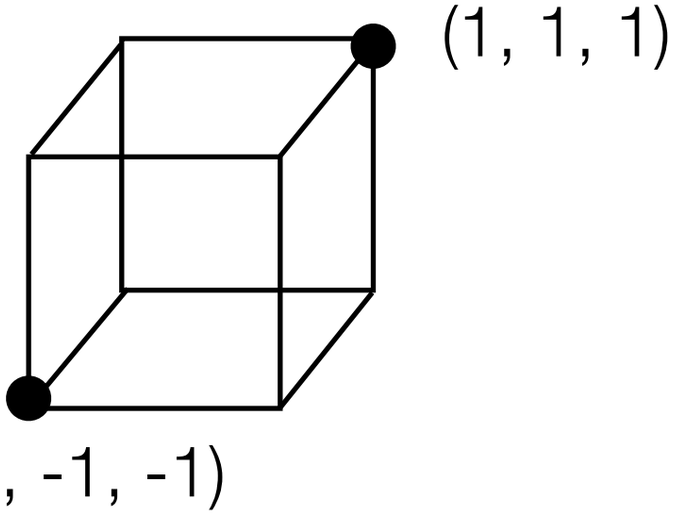
Understanding the Parallel Projection Effect in WebGL

Right-handed
coordinate system
(positive z out of slide)
(negative z into slide)



Camera
Coordinates

Left-handed
coordinate system
(positive z into slide)
(negative z out of slide)



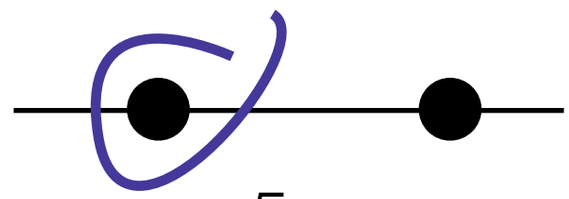
Clip
Coordinates

z coordinate
is used for ordering
rendered pixels.

Understanding the Parallel Projection Effect in WebGL

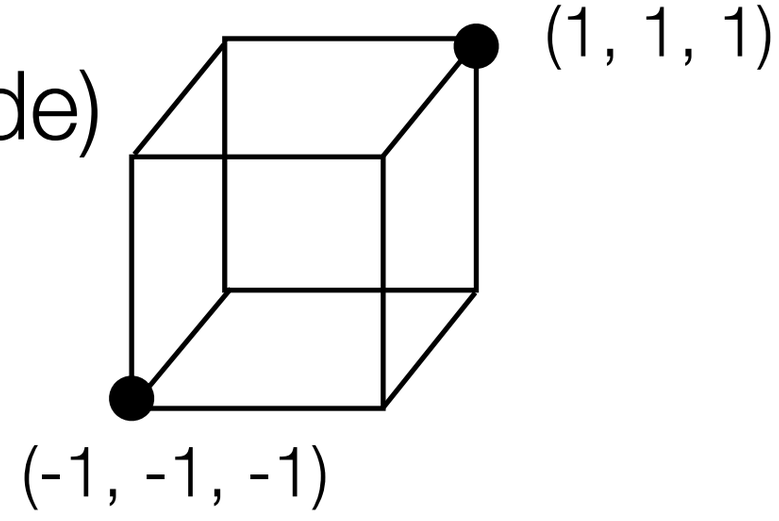
z_{pv} gets stored in **depth buffer**.

If two pixels have same x_{pv} and y_{pv},
the pixel with lower z_{pv} gets rendered.



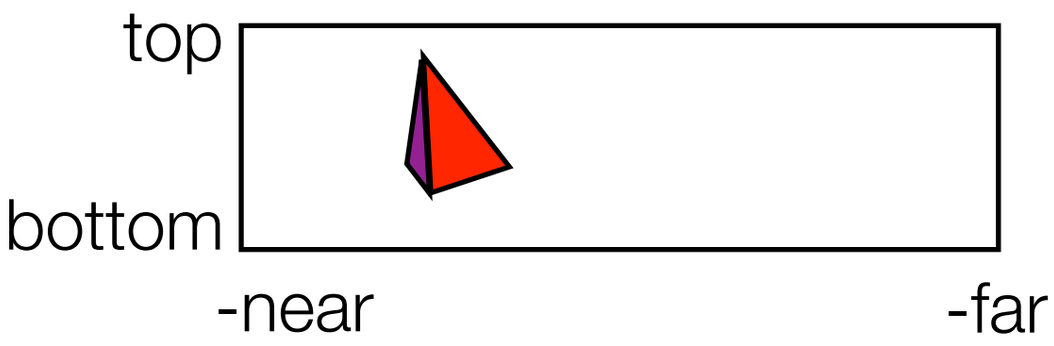
coordinate system
(positive z into slide)
(negative z out of slide)

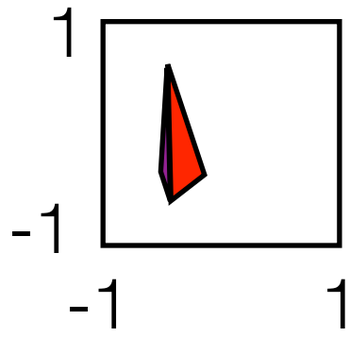
z coordinate
is used for ordering
rendered pixels.



Clip
Coordinates

Side View of Parallel Projection

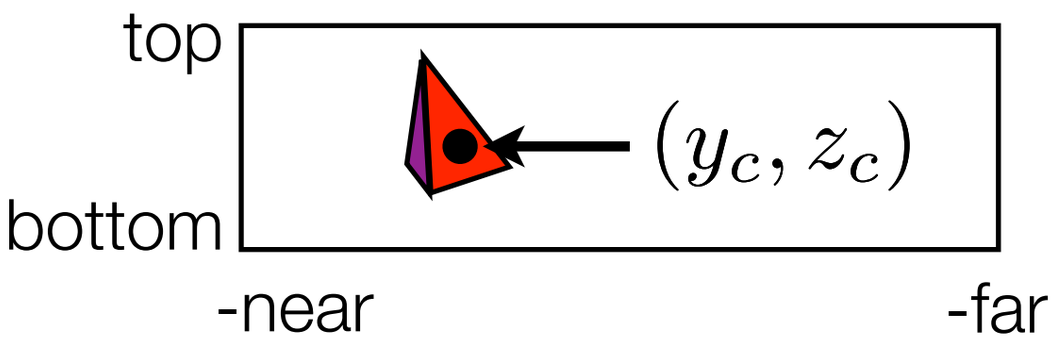


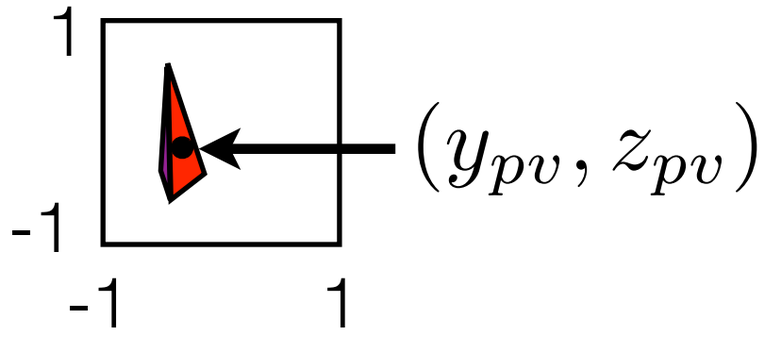


Camera Coordinates

Clip Coordinates

Side View of Parallel Projection

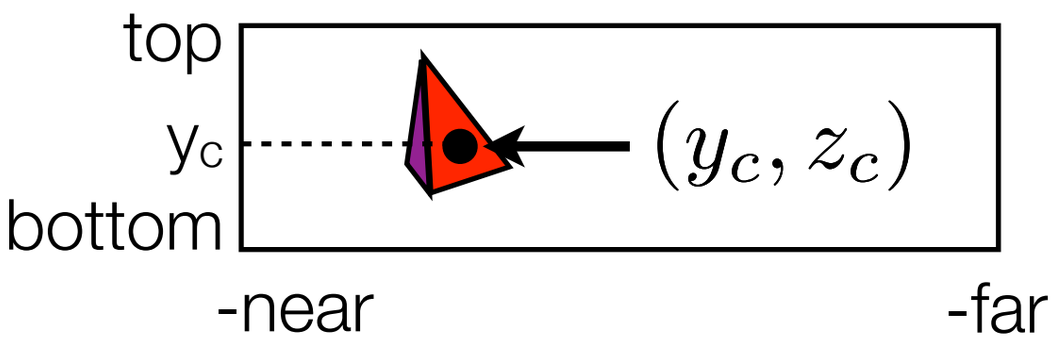


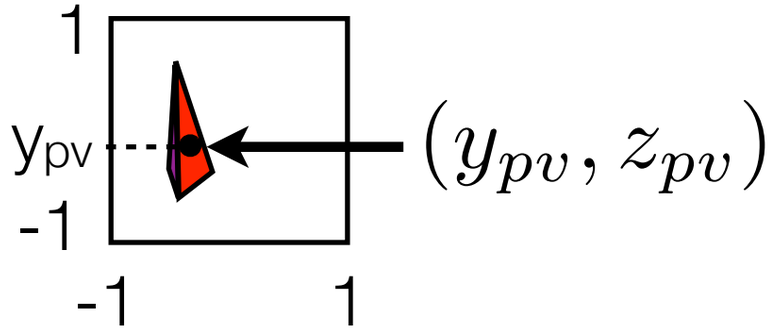


Camera Coordinates

Clip Coordinates

Side View of Parallel Projection

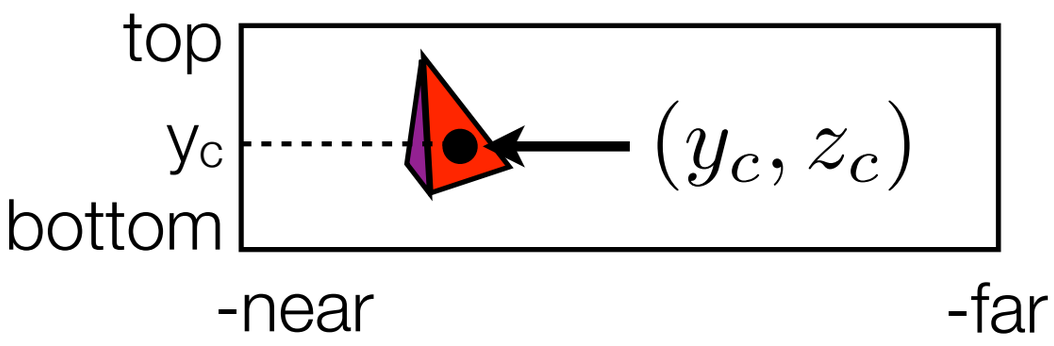


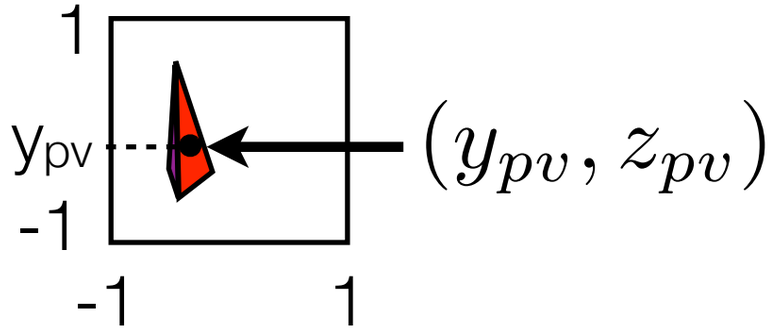


Camera Coordinates

Clip Coordinates

Side View of Parallel Projection





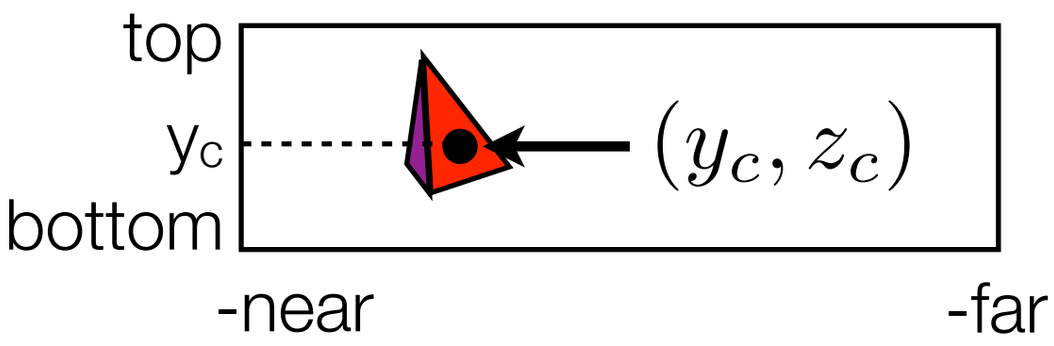
Camera Coordinates

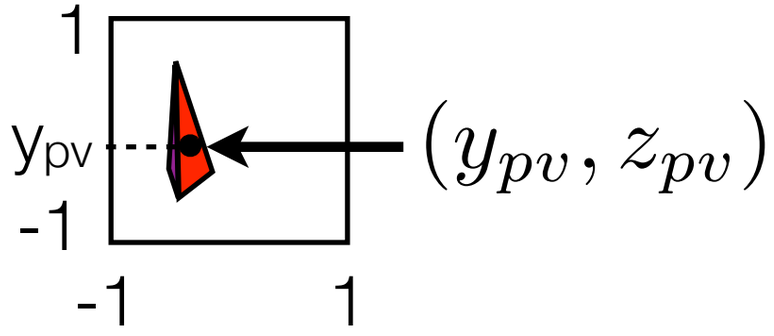
Clip Coordinates

$$\frac{y_{pv} - (-1)}{1 - (-1)} = \frac{y_c - \text{bottom}}{\text{top} - \text{bottom}}$$

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Side View of Parallel Projection





Camera Coordinates

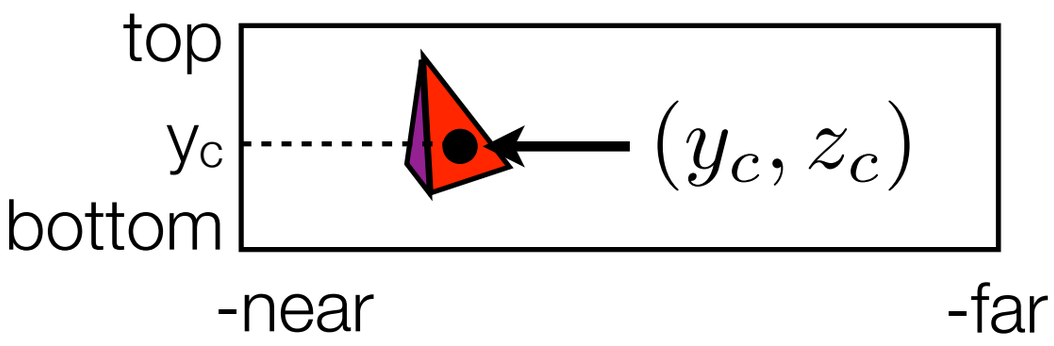
Clip Coordinates

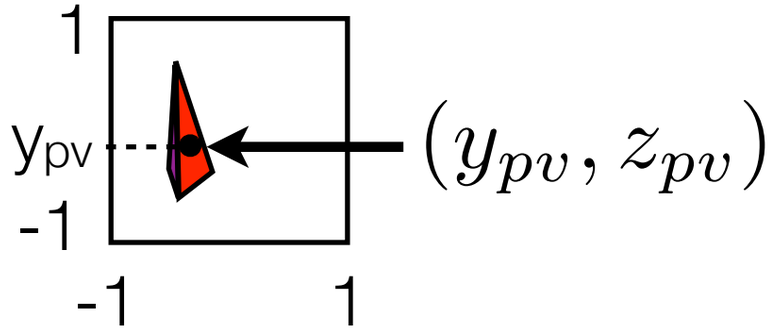
$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

$$\frac{y_{pv} - (-1)}{1 - (-1)} = \frac{y_c - \text{bottom}}{\text{top} - \text{bottom}}$$

$$\frac{y_{pv} + 1}{2} = \frac{y_c}{\text{top} - \text{bottom}} - \frac{\text{bottom}}{\text{top} - \text{bottom}}$$

Side View of Parallel Projection





Camera Coordinates

Clip Coordinates

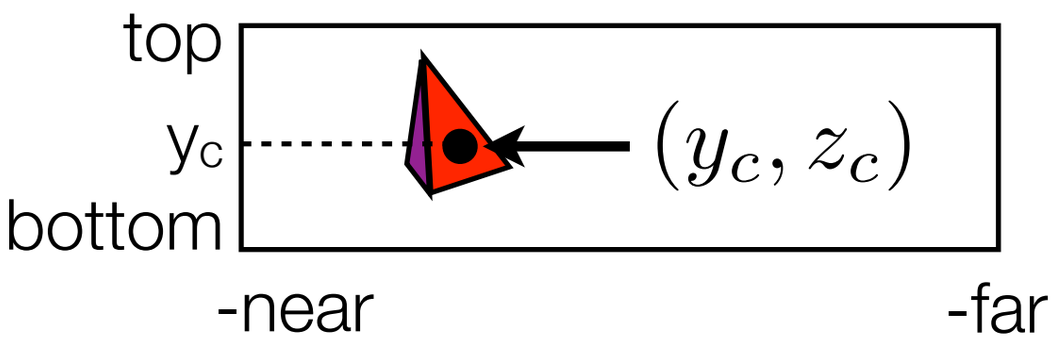
$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

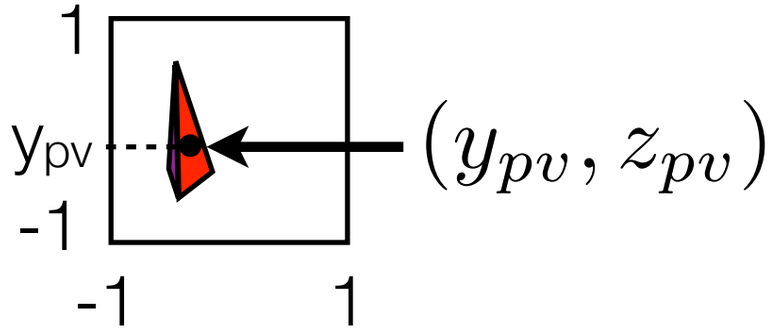
$$\frac{y_{pv} - (-1)}{1 - (-1)} = \frac{y_c - \text{bottom}}{\text{top} - \text{bottom}}$$

$$\frac{y_{pv} + 1}{2} = \frac{y_c}{\text{top} - \text{bottom}} - \frac{\text{bottom}}{\text{top} - \text{bottom}}$$

$$y_{pv} = 2\frac{y_c}{\text{top} - \text{bottom}} - 2\frac{\text{bottom}}{\text{top} - \text{bottom}} - 1$$

Side View of Parallel Projection





Camera Coordinates

Clip Coordinates

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

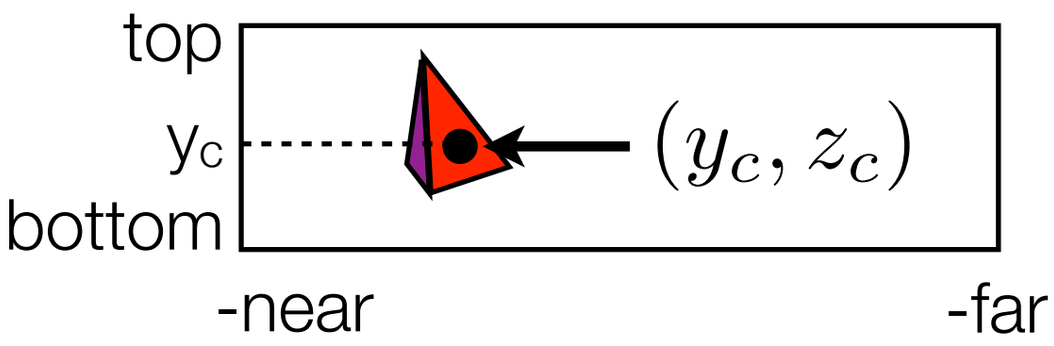
$$\frac{y_{pv} - (-1)}{1 - (-1)} = \frac{y_c - \text{bottom}}{\text{top} - \text{bottom}}$$

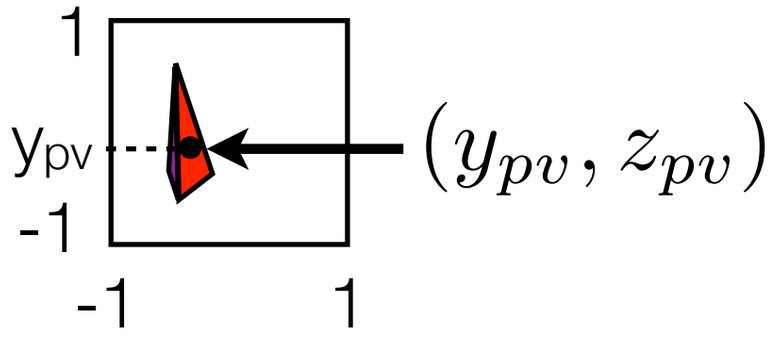
$$\frac{y_{pv} + 1}{2} = \frac{y_c}{\text{top} - \text{bottom}} - \frac{\text{bottom}}{\text{top} - \text{bottom}}$$

$$y_{pv} = 2\frac{y_c}{\text{top} - \text{bottom}} - 2\frac{\text{bottom}}{\text{top} - \text{bottom}} - 1$$

$$= \frac{2}{\text{top} - \text{bottom}} y_c - \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}}$$

Side View of Parallel Projection





Camera Coordinates

Clip Coordinates

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

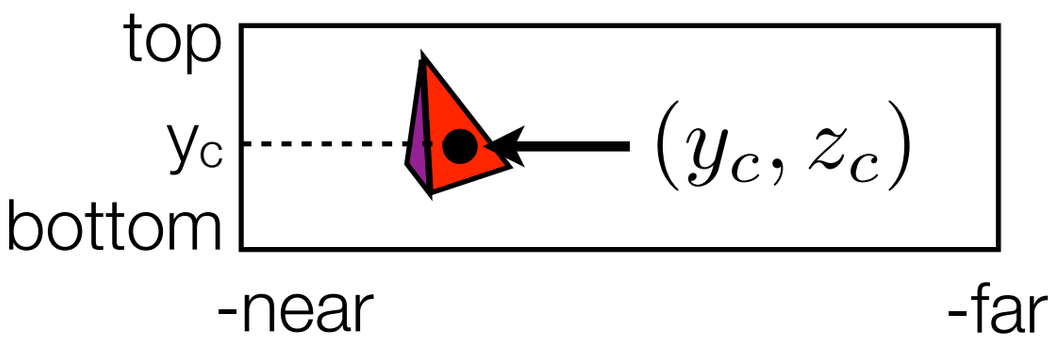
$$\frac{y_{pv} - (-1)}{1 - (-1)} = \frac{y_c - \text{bottom}}{\text{top} - \text{bottom}}$$

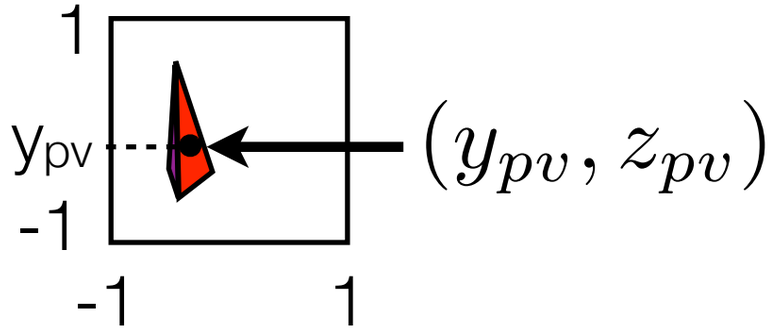
$$\frac{y_{pv} + 1}{2} = \frac{y_c}{\text{top} - \text{bottom}} - \frac{\text{bottom}}{\text{top} - \text{bottom}}$$

$$y_{pv} = 2\frac{y_c}{\text{top} - \text{bottom}} - 2\frac{\text{bottom}}{\text{top} - \text{bottom}} - 1$$

$$= \frac{2}{\text{top} - \text{bottom}} y_c - \frac{\text{top} + \text{bottom}}{\text{top} - \text{bottom}}$$

Side View of Parallel Projection

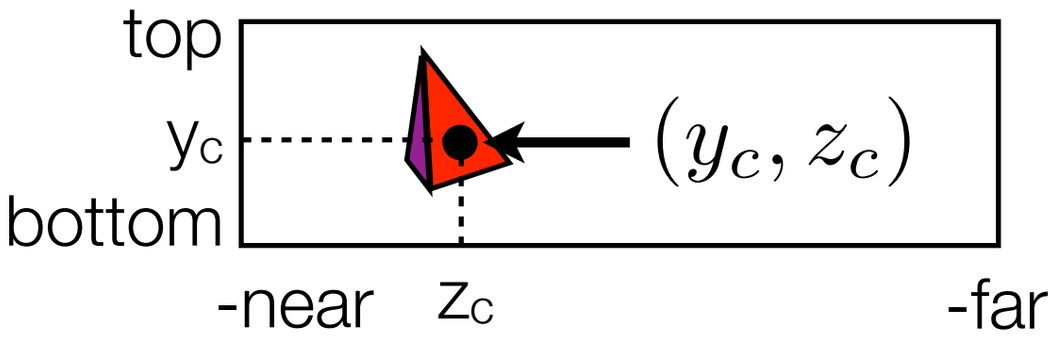


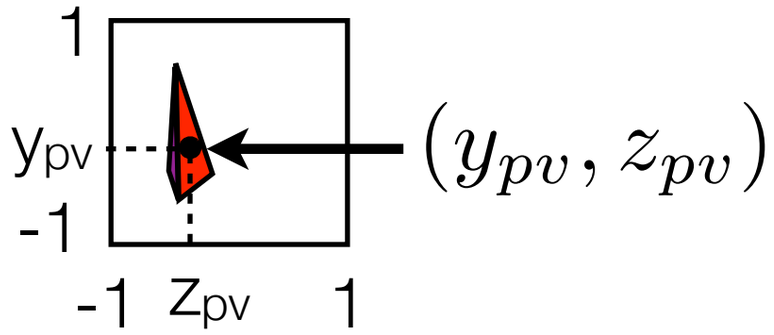


Camera Coordinates

Clip Coordinates

Side View of Parallel Projection





Camera Coordinates

Clip Coordinates

$$\frac{z_{pv} - (-1)}{1 - (-1)} = \frac{z_c - (-\text{near})}{(-\text{far}) - (-\text{near})}$$

$$\frac{z_{pv} + 1}{2} = -\frac{z_c - \text{near}}{\text{far} - \text{near}}$$

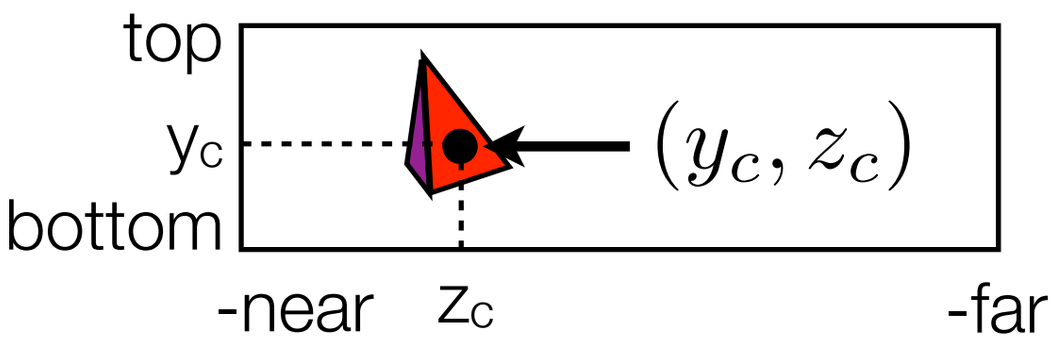
$$\frac{z_{pv} + 1}{2} = -\frac{z_c}{\text{far} - \text{near}} - \frac{\text{near}}{\text{far} - \text{near}}$$

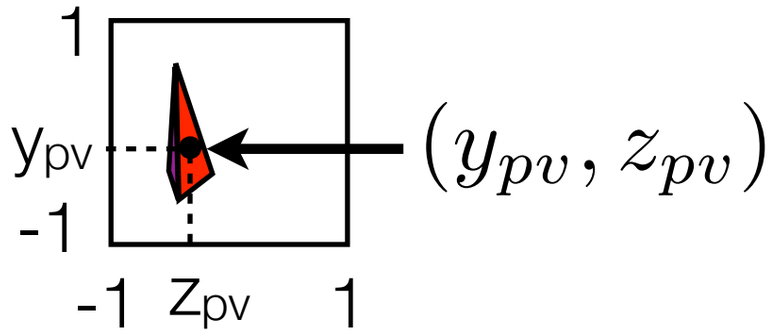
$$z_{pv} = -2\frac{z_c}{\text{far} - \text{near}} - 2\frac{\text{near}}{\text{far} - \text{near}} - 1$$

$$\boxed{= -\frac{2}{\text{far} - \text{near}} z_c - \frac{\text{far} + \text{near}}{\text{far} - \text{near}}}$$

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Side View of Parallel Projection





Camera Coordinates

Clip Coordinates

$$\frac{z_{pv} - (-1)}{1 - (-1)} = \frac{z_c - (-\text{near})}{(-\text{far}) - (-\text{near})}$$

$$\frac{z_{pv} + 1}{2} = -\frac{z_c - \text{near}}{\text{far} - \text{near}}$$

$$\frac{z_{pv} + 1}{2} = -\frac{z_c}{\text{far} - \text{near}} - \frac{\text{near}}{\text{far} - \text{near}}$$

$$z_{pv} = -2\frac{z_c}{\text{far} - \text{near}} - 2\frac{\text{near}}{\text{far} - \text{near}} - 1$$

$$= -\frac{2}{\text{far} - \text{near}} z_c - \frac{\text{far} + \text{near}}{\text{far} - \text{near}}$$

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

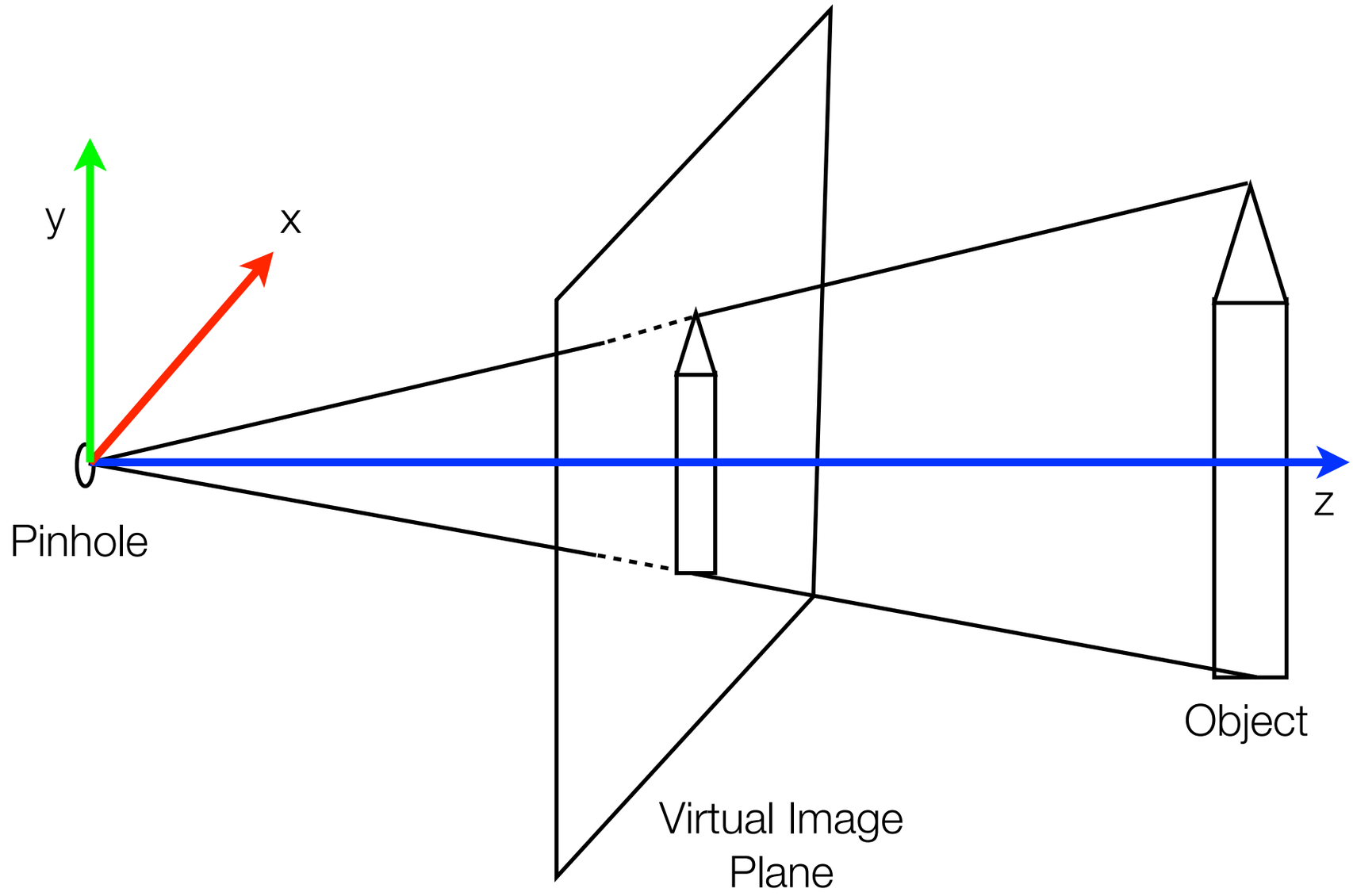
Now let us look at perspective projections.

Perspective projections mimic a real camera.

So let us start with projection equations for a real camera!

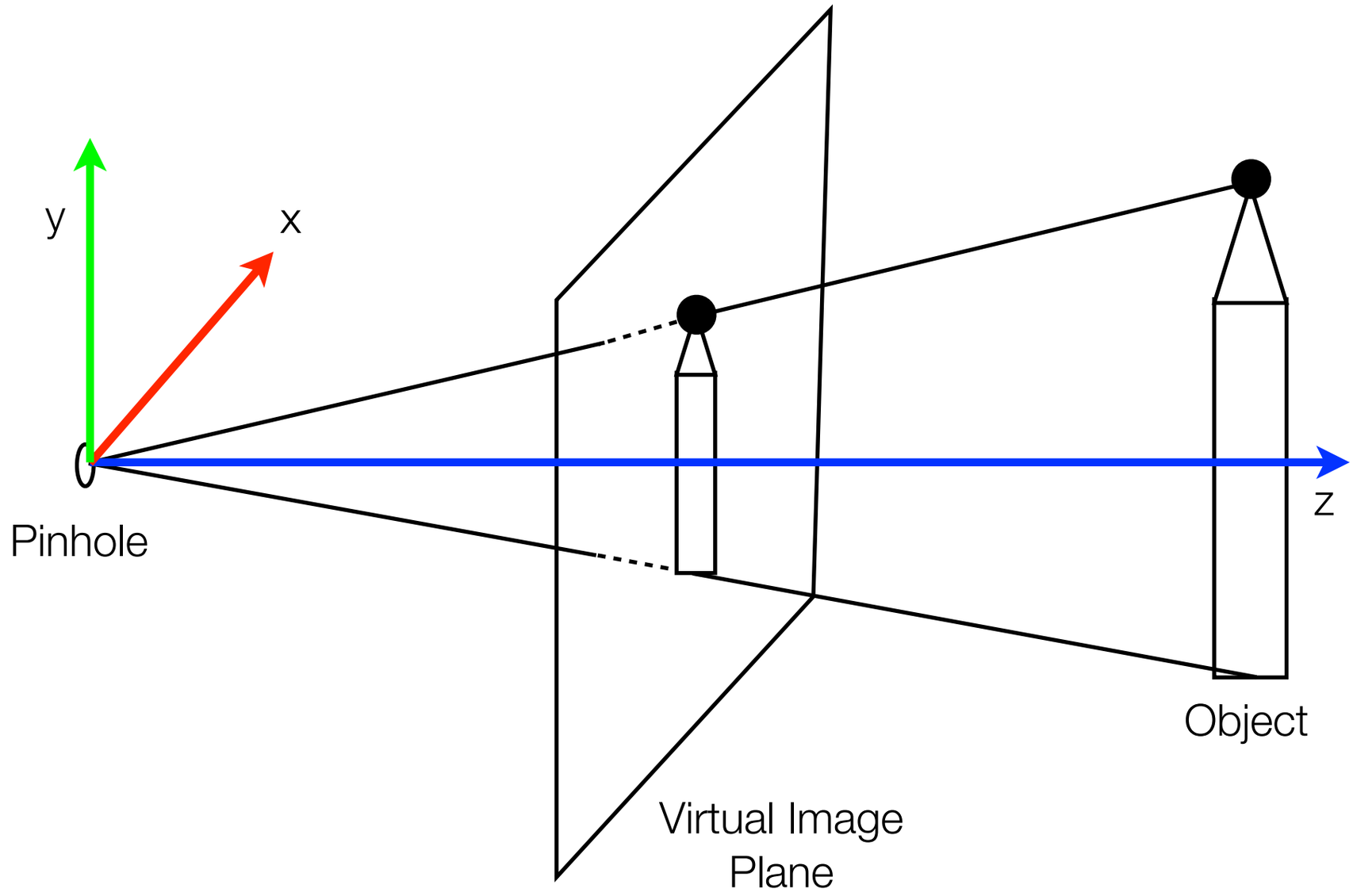
Perspective Projections

Back when we were looking at image formation, we saw how to project a 3D object in 2D.



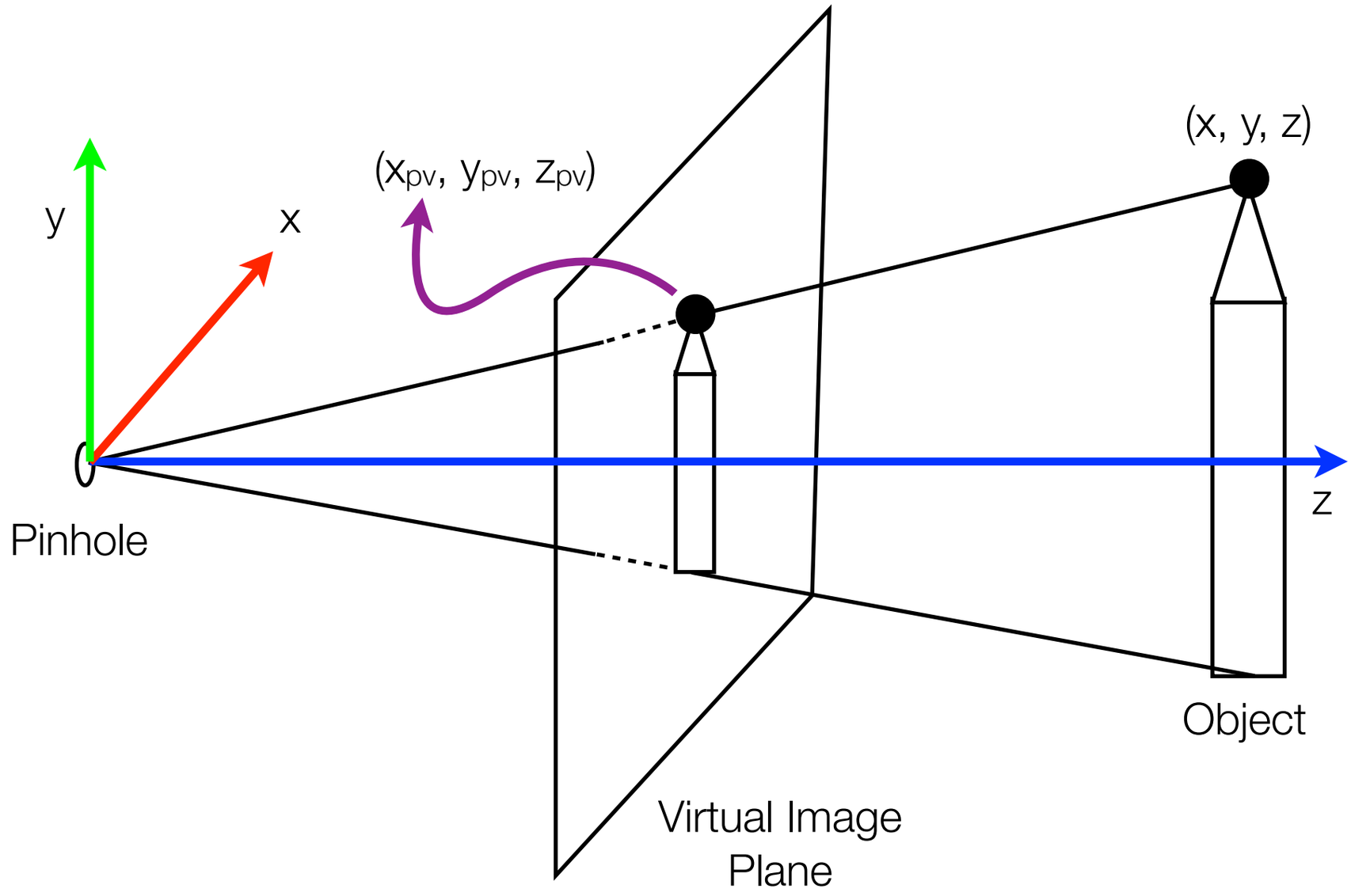
Perspective Projections

We saw how to project a 3D point in 2D



Perspective Projections

We saw how to project a 3D point in 2D



Perspective Projections

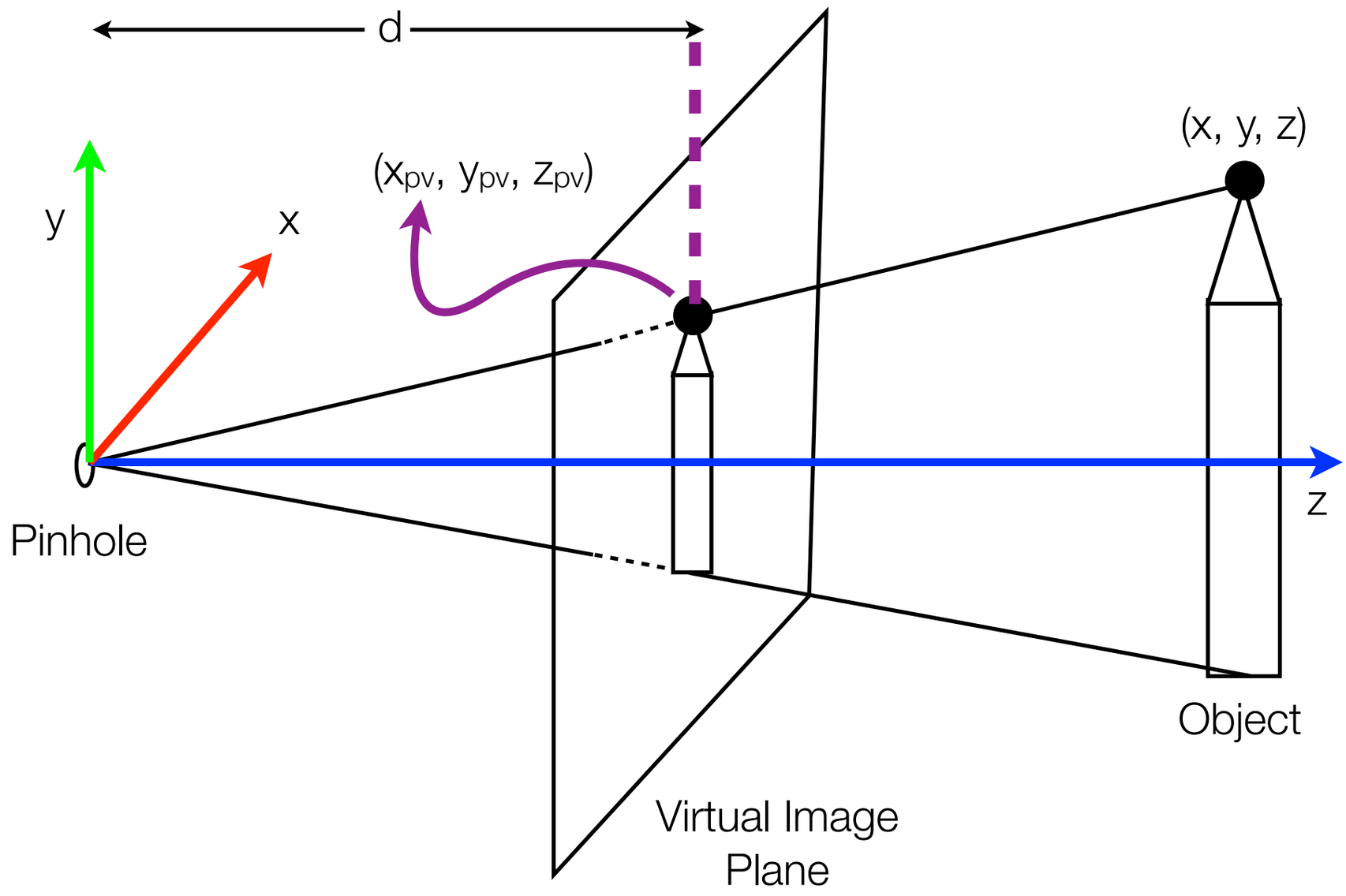
We saw how to project a 3D point in 2D

Projection Equations

$$x_{pv} = \frac{xd}{z}$$

$$y_{pv} = \frac{yd}{z}$$

$$z_{pv} = d$$



Perspective Projections

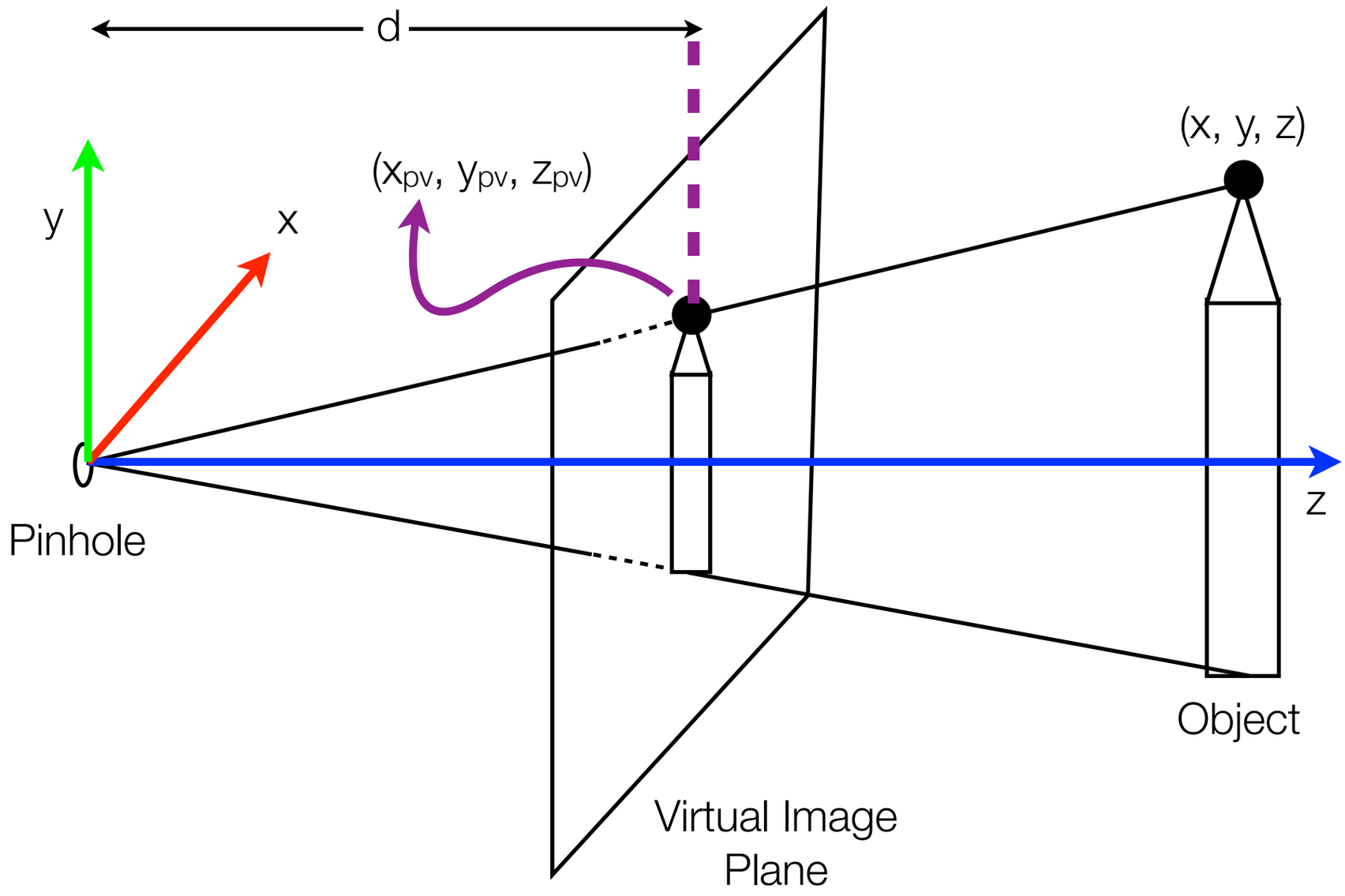
We will make a few changes...

Projection Equations

$$x_{pv} = \frac{xd}{z}$$

$$y_{pv} = \frac{yd}{z}$$

$$z_{pv} = d$$



Perspective Projections

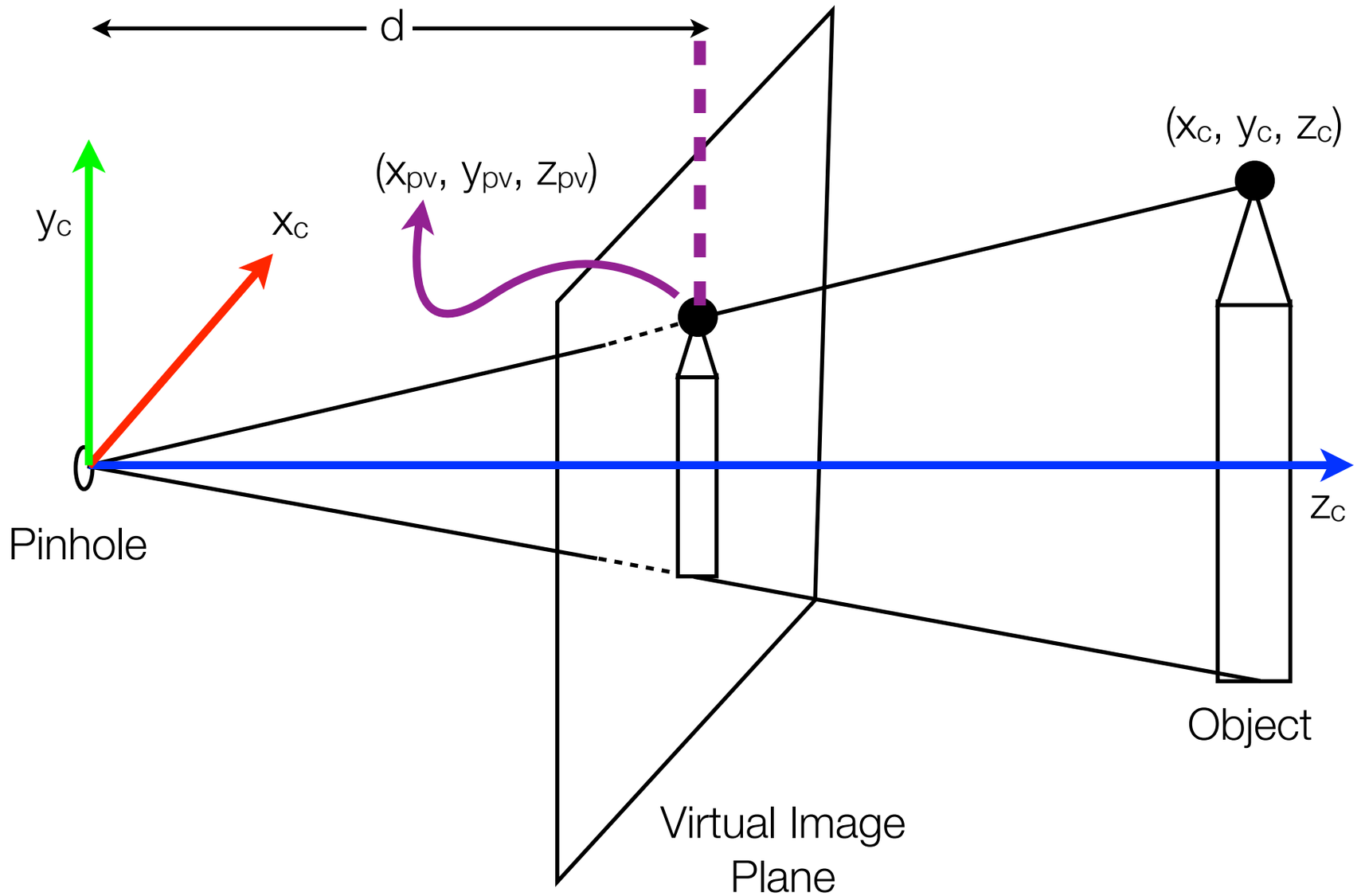
First, we will specifically label the camera coordinates

Projection Equations

$$x_{pv} = \frac{x_c d}{z_c}$$

$$y_{pv} = \frac{y_c d}{z_c}$$

$$z_{pv} = d$$



Perspective Projections

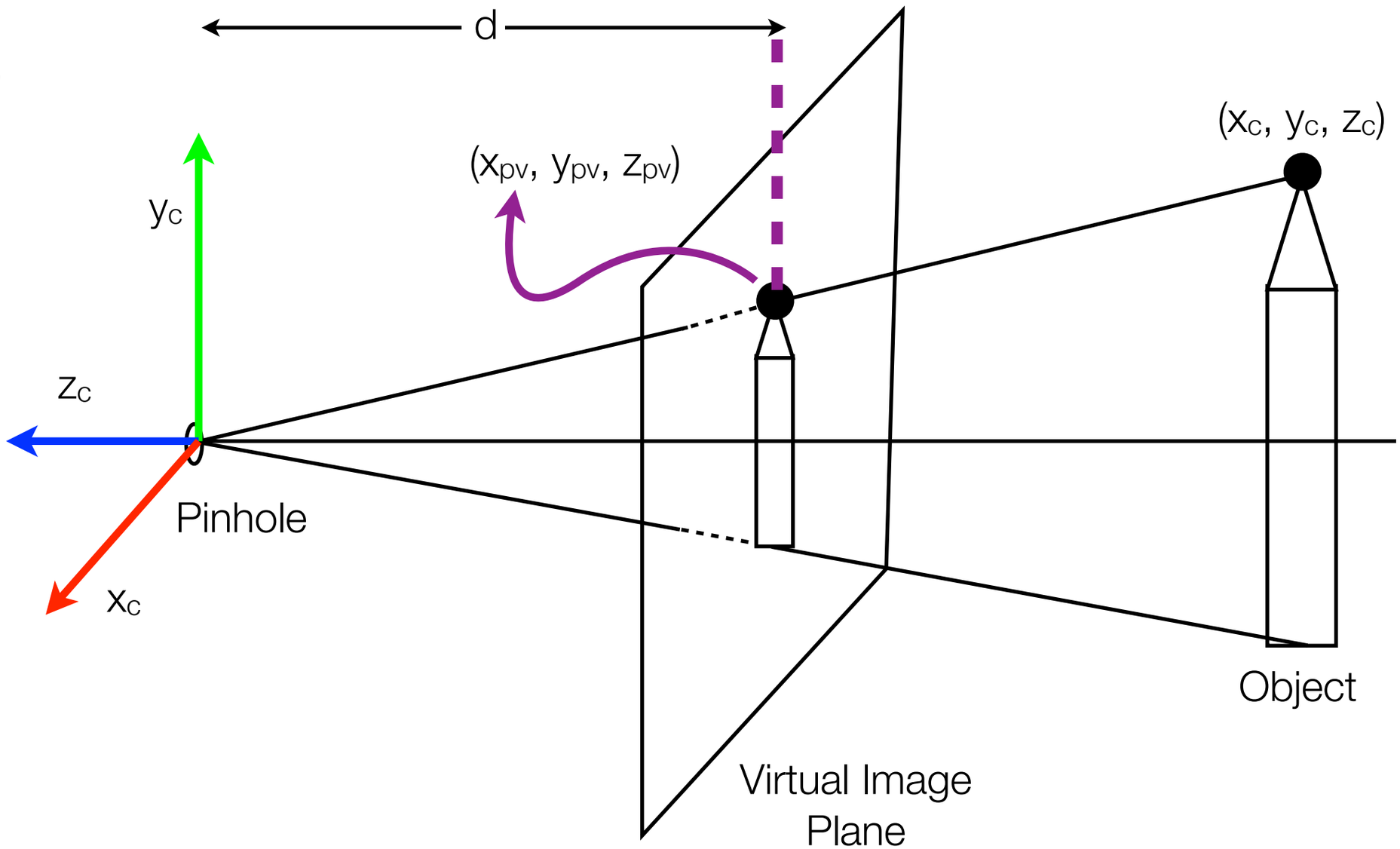
Then, we will turn our coordinate system so that z_c faces away from the image as in WebGL: Equations get negated

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$



Perspective Projections

Equations are non-linear in x_c, y_c, and z_c because of division by z_c.

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Perspective Projections

So on their own, they cannot be expressed as
[output vector] = [Matrix] [input vector]

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What are the individual equations?

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What are the individual equations?

$$x_{pv} = dx_c$$

$$y_{pv} = dy_c$$

$$z_{pv} = dz_c$$

$$w_{pv} = -z_c$$

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What are the individual equations?

$$x_{pv} = dx_c$$

$$y_{pv} = dy_c$$

$$z_{pv} = dz_c$$

$$w_{pv} = -z_c$$

Note something interesting!
w_{pv} **is no longer 1**!

Perspective Projections

Pr on:

x_p

y_p

z_p

Perspective Projections
cause your w coordinate to not equal 1!

$$x_{pv} = dx_c$$

$$y_{pv} = dy_c$$

$$z_{pv} = dz_c$$

$$w_{pv} = -z_c$$

Note something interesting!
w_{pv} **is no longer 1**!

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What are the individual equations?

$$x_{pv} = dx_c$$

$$y_{pv} = dy_c$$

$$z_{pv} = dz_c$$

$$w_{pv} = -z_c$$

Note something interesting!
w_{pv} **is no longer 1**!

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What are the individual equations?

$$x_{pv} = dx_c$$

$$y_{pv} = dy_c$$

$$z_{pv} = dz_c$$

$$w_{pv} = -z_c$$

Note something interesting!
w_{pv} **is no longer 1**!

We are going to do this new wonky thing of **dividing everything** by w_{pv}!

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$x_{pv} = \frac{-dx_c}{z_c}$$

After dividing by $w_{pv} = -z_c$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d,\ w_{pv} = 1$$

Perspective Projections

We will look at a different way to write the equations

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$x_{pv} = \frac{-dx_c}{z_c}$$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d,\ w_{pv} = 1$$

After dividing by $w_{pv} = -z_c$

Does not this look just like the equations on the left?

To correctly perform perspective projections...

Divide x_{pv}, y_{pv}, and z_{pv} by w_{pv}.

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$x_{pv} = \frac{-dx_c}{z_c}$$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d, \; w_{pv} = 1$$

After dividing by $w_{pv} = -z_c$

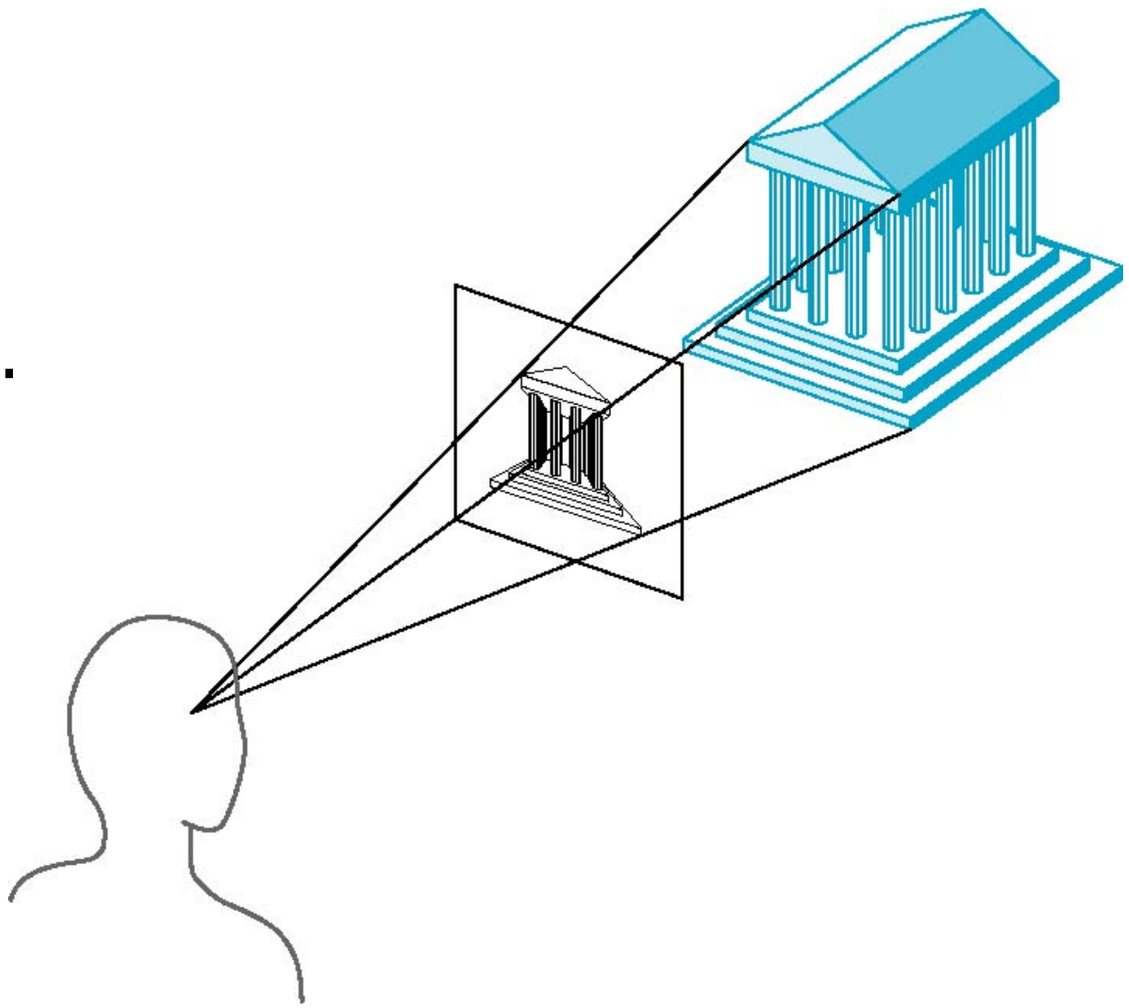
Does not this look just like the equations on the left?

To correctly perform perspective projections...

Divide x_{pv}, y_{pv}, and z_{pv} by w_{pv}.

Division by w_{pv} represents shrinking of an object as it moves away from a camera.

The farther an object, the larger is w_{pv}, and the smaller its image.



$$z_{pv} = -d,\ w_{pv} = 1$$

To correctly perform perspective projections...

Divide x_{pv}, y_{pv}, and z_{pv} by w_{pv}.

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$x_{pv} = \frac{-dx_c}{z_c}$$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d, \; w_{pv} = 1$$

After dividing by $w_{pv} = -z_c$

Does not this look just like the equations on the left?

This is all very well if x_{pv}, y_{pv}, and z_{pv} can fit in the viewport

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

$$x_{pv} = \frac{-dx_c}{z_c}$$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d,\ w_{pv} = 1$$

But if they cannot (which happens more often than not) we need to add some more parameters to the projection matrix

Projection Equations

$$x_{pv} = -\frac{x_c d}{z_c}$$

$$y_{pv} = -\frac{y_c d}{z_c}$$

$$z_{pv} = -d$$

Let us look at the following matrix-vector multiplication:

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

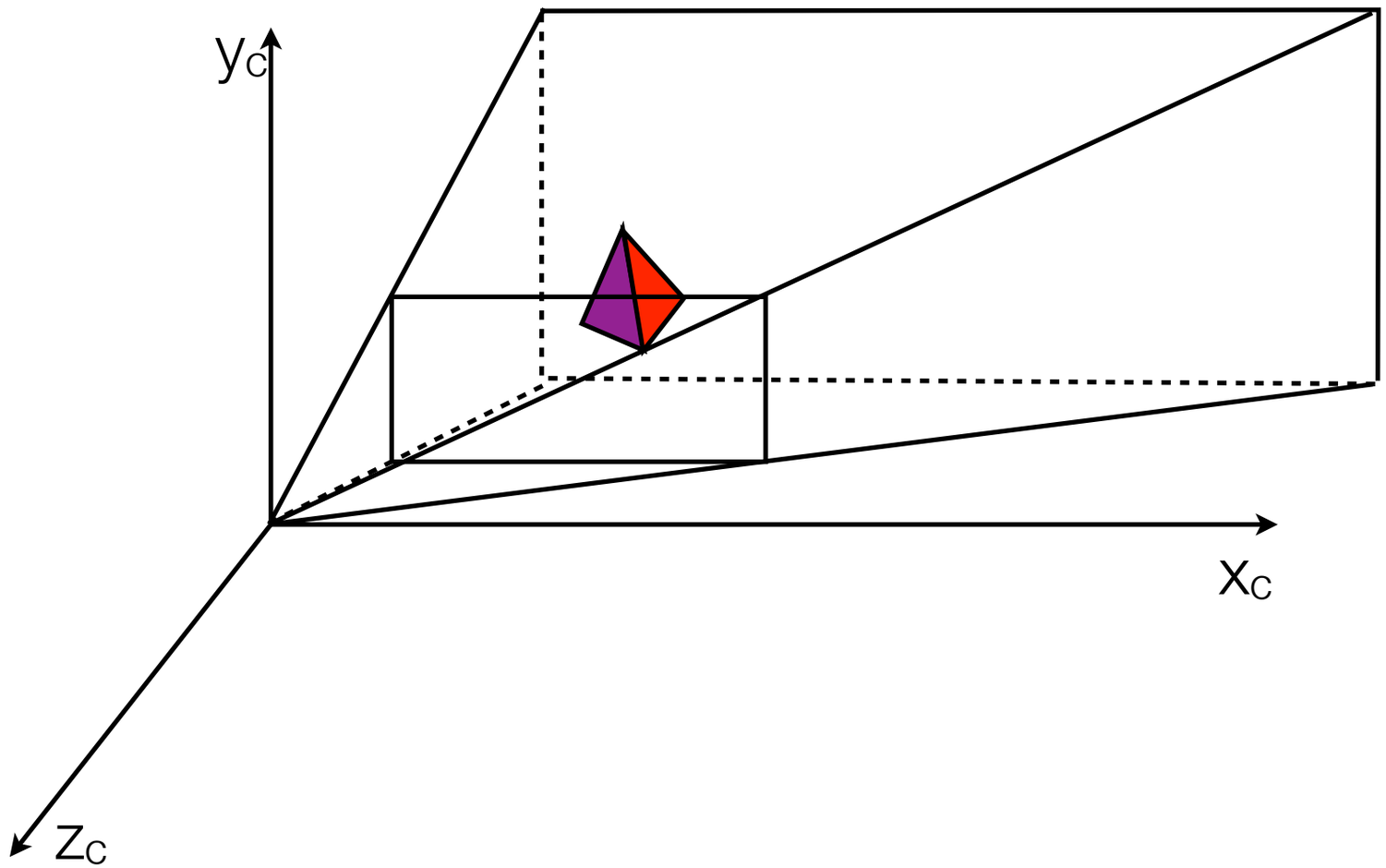
$$x_{pv} = \frac{-dx_c}{z_c}$$

$$y_{pv} = \frac{-dy_c}{z_c}$$

$$z_{pv} = -d,\ w_{pv} = 1$$

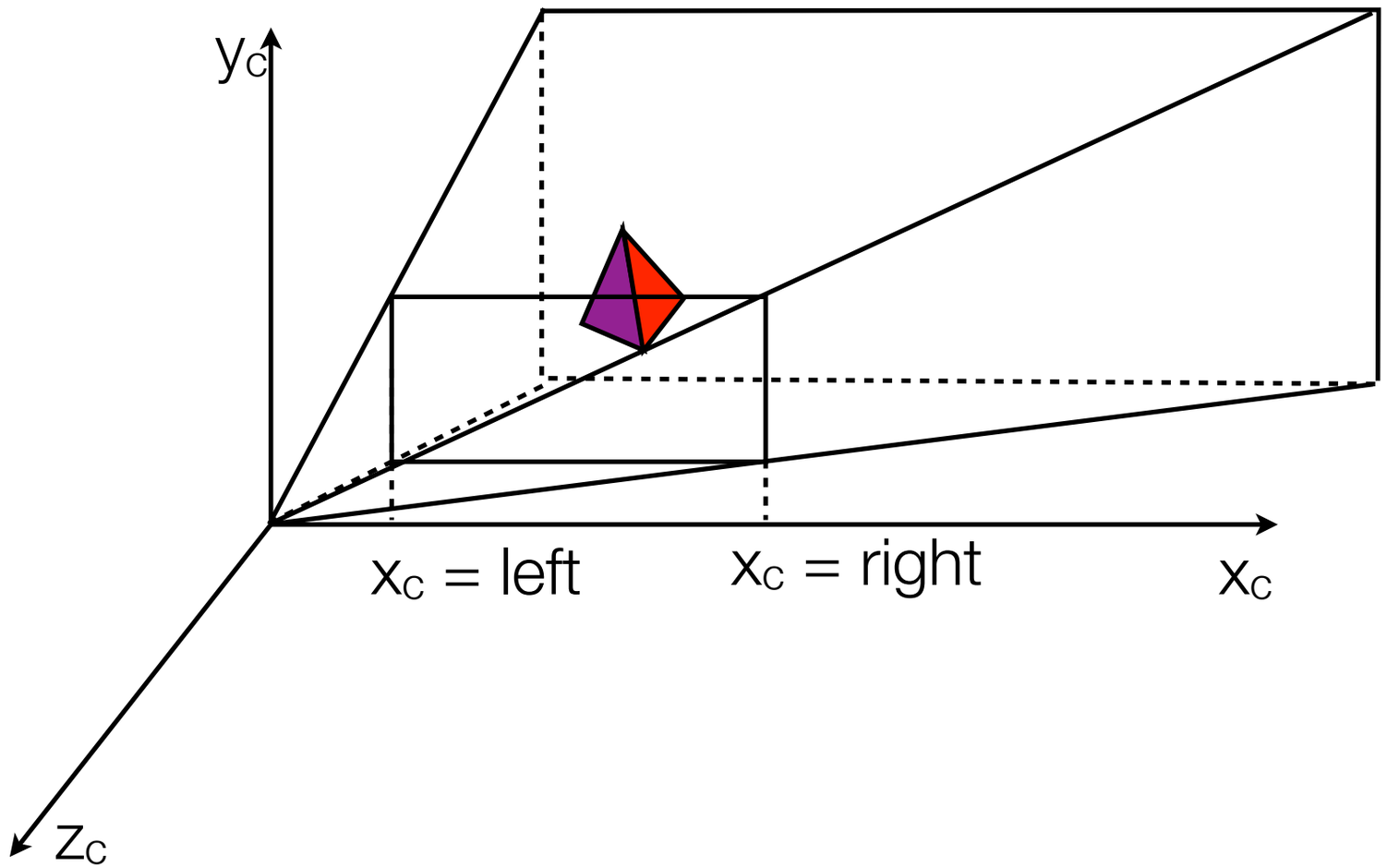
Perspective Projections

Our **viewing volume** is now a frustum



Perspective Projections

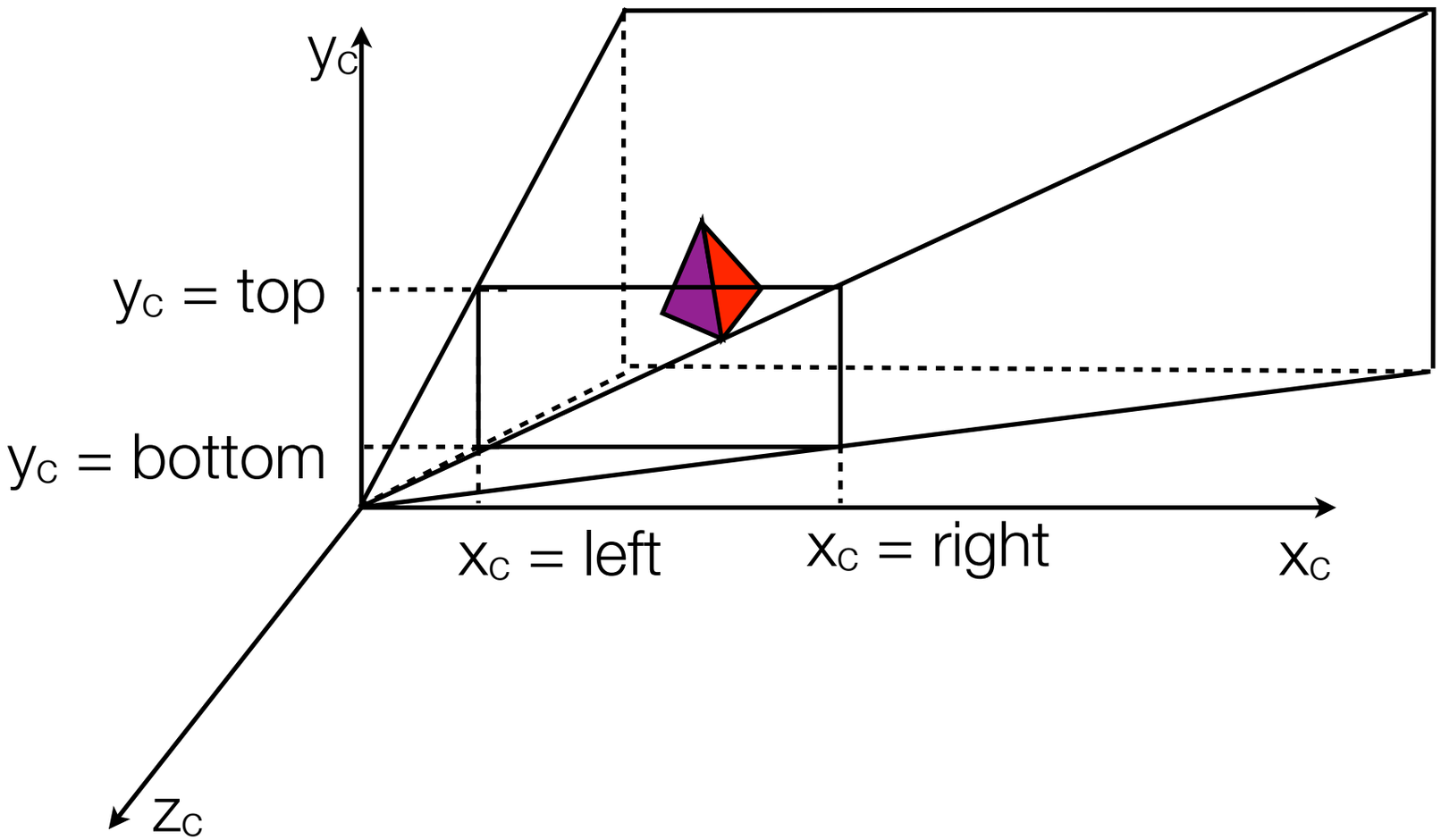
Our **viewing volume** is now a frustum



Left plane, right plane,

Perspective Projections

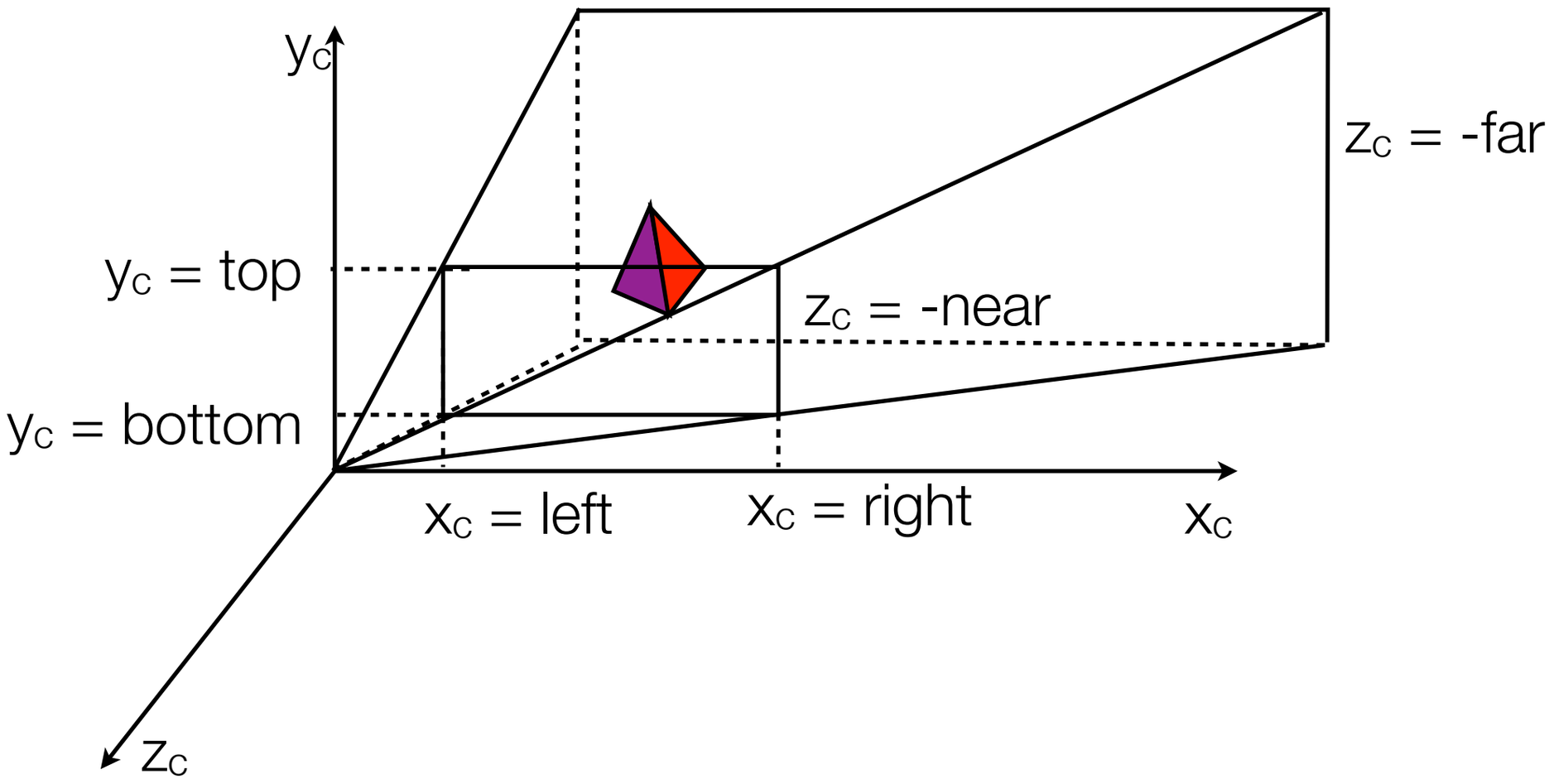
Our **viewing volume** is now a frustum



Left plane, right plane, top plane, bottom plane, near plane, far plane

Perspective Projections

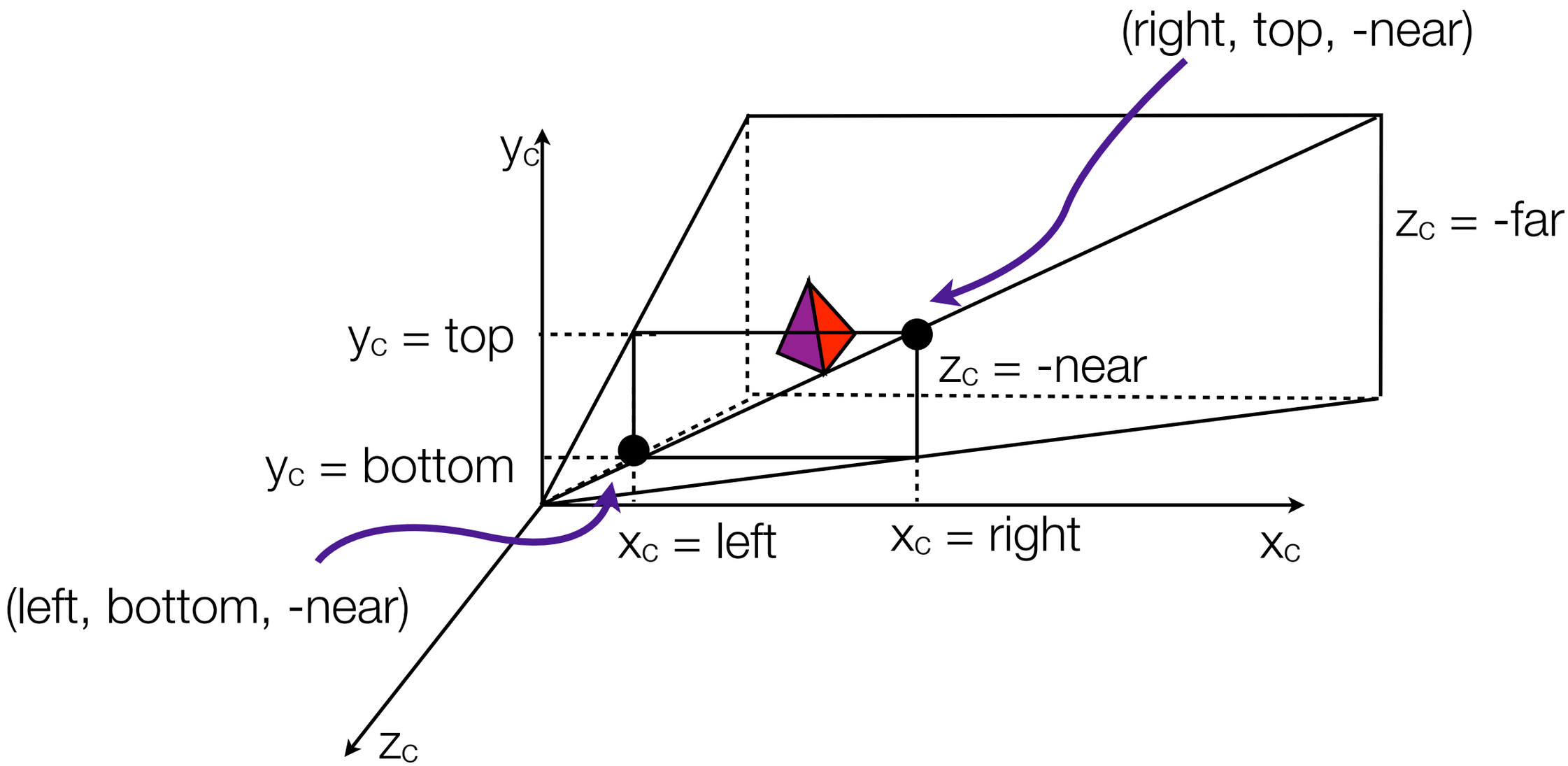
Our **viewing volume** is now a frustum



Left plane, right plane, top plane, bottom plane, near plane, far plane

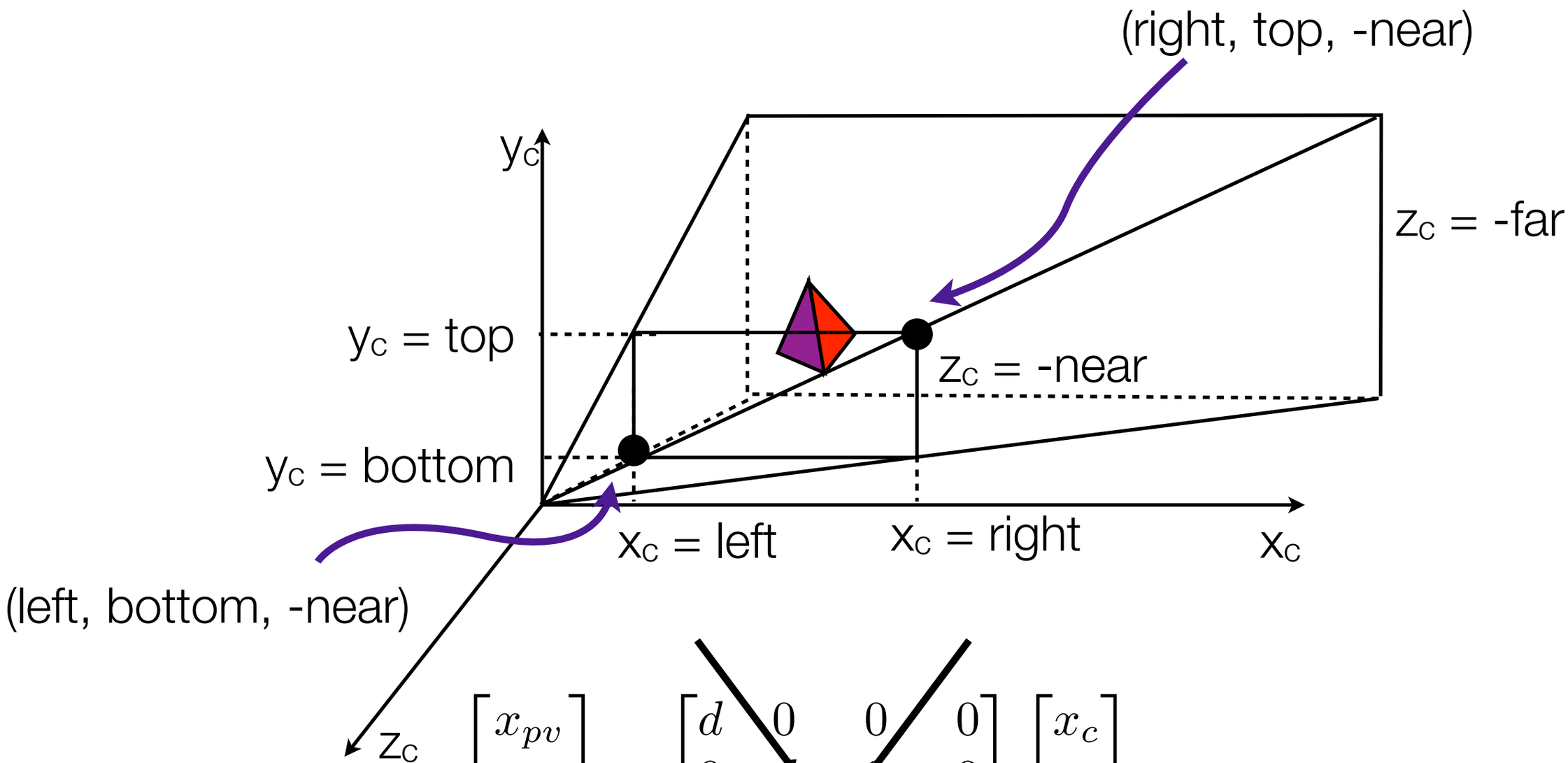
Perspective Projections

Our **viewing volume** is now a frustum



Perspective Projections

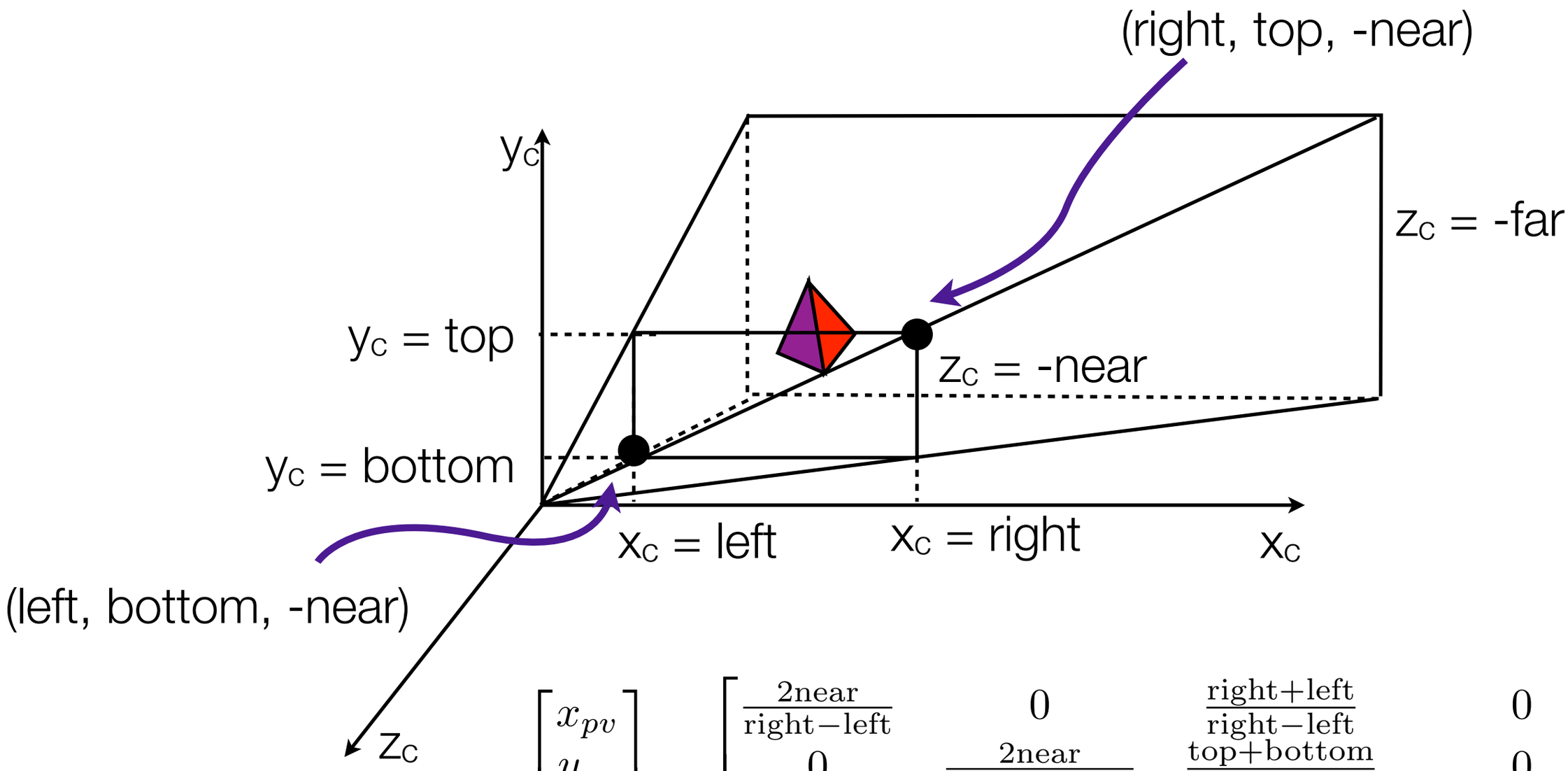
Our **viewing volume** is now a frustum



$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

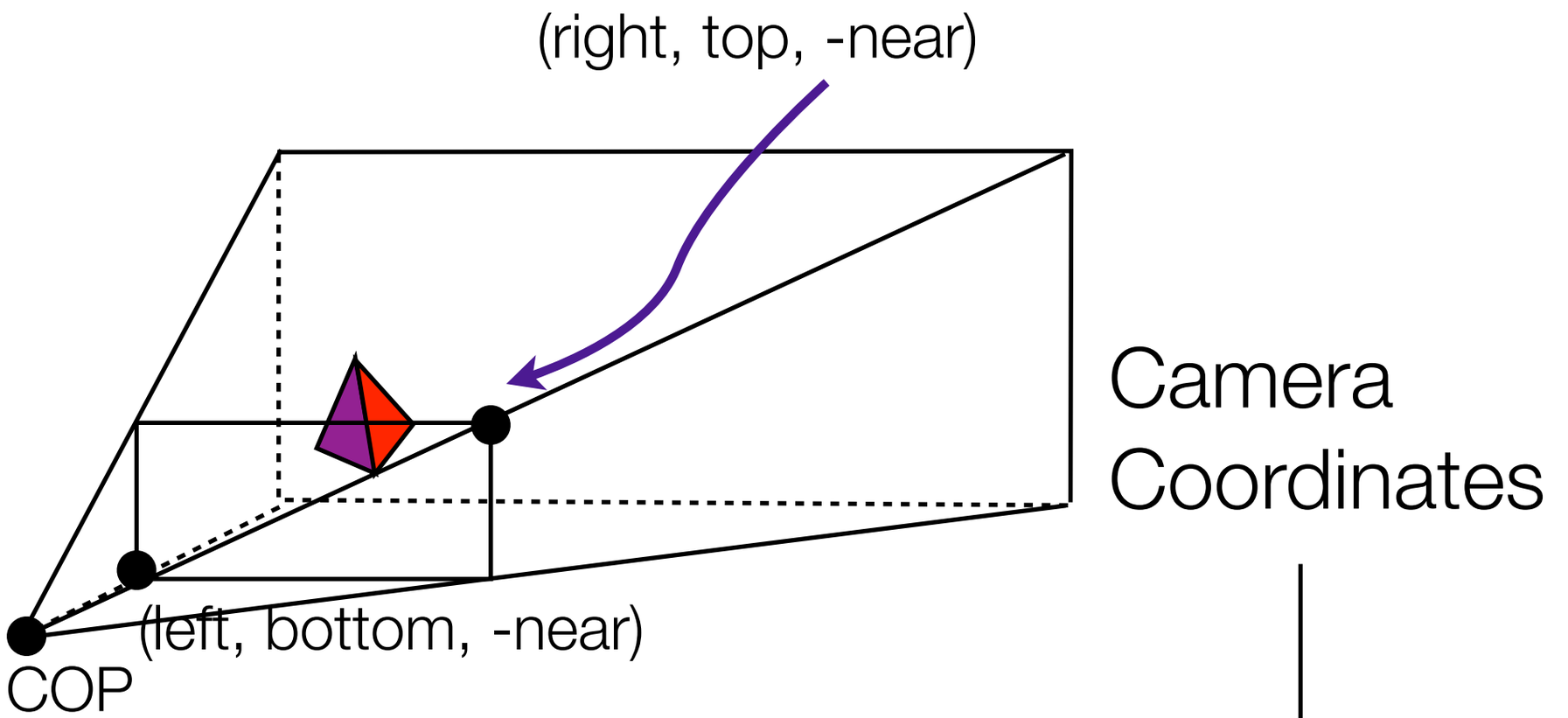
Perspective Projections

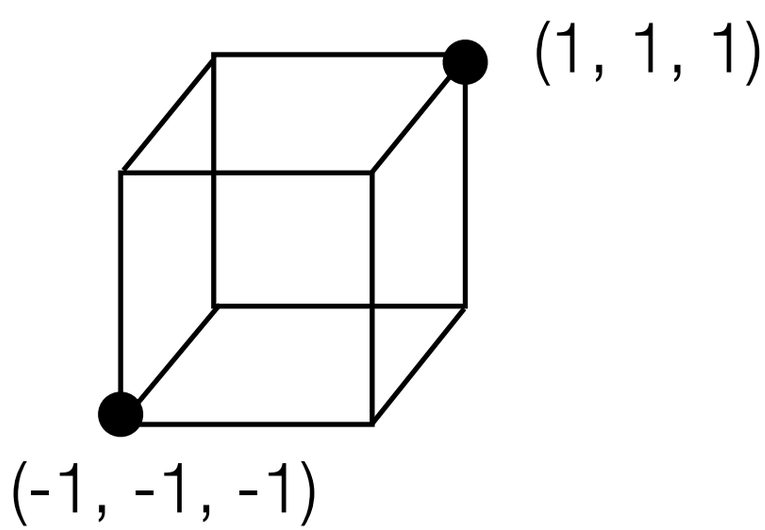
Our **viewing volume** is now a frustum



$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2\text{near}}{\text{right}-\text{left}} & 0 & \frac{\text{right}+\text{left}}{\text{right}-\text{left}} & 0 \\ 0 & \frac{2\text{near}}{\text{top}-\text{bottom}} & \frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Understanding the Perspective Projection Effect in WebGL

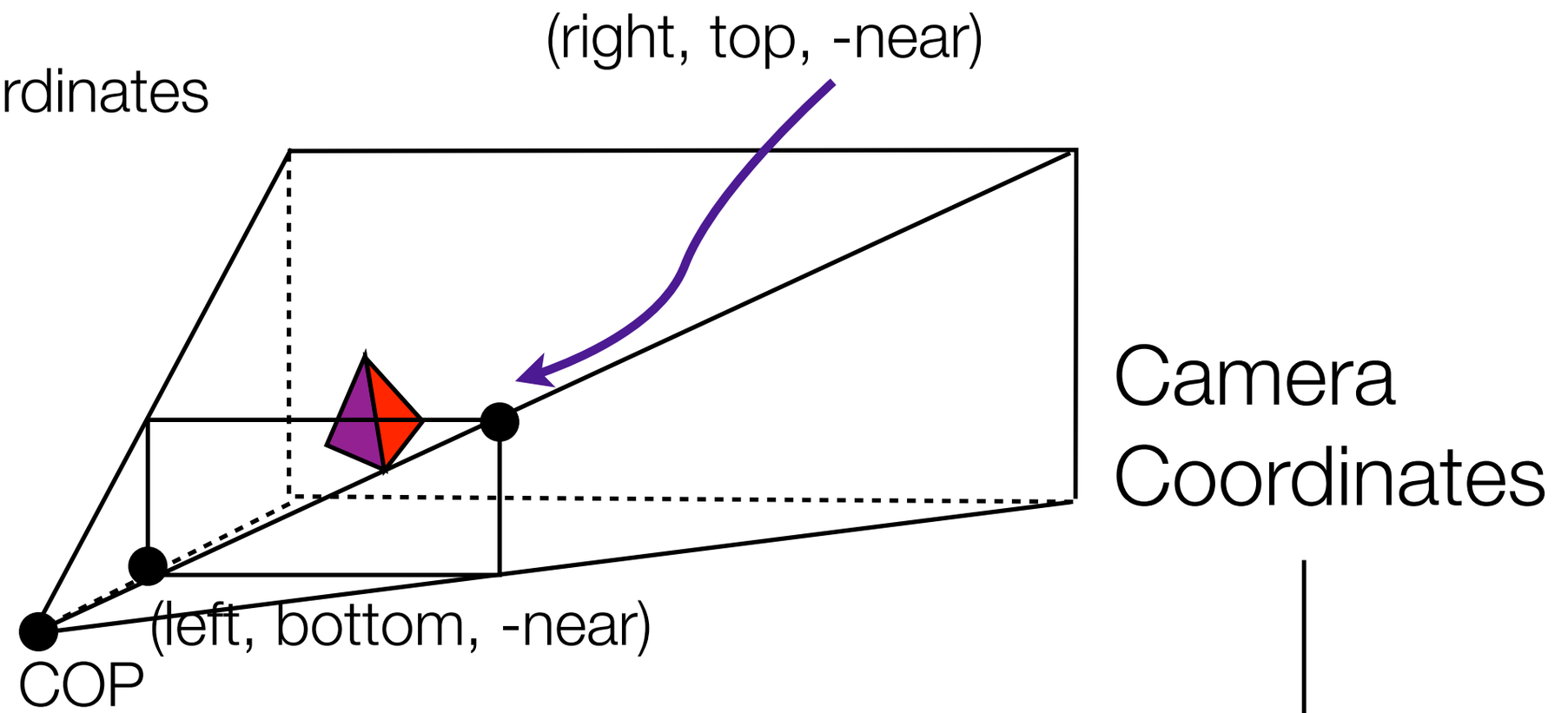




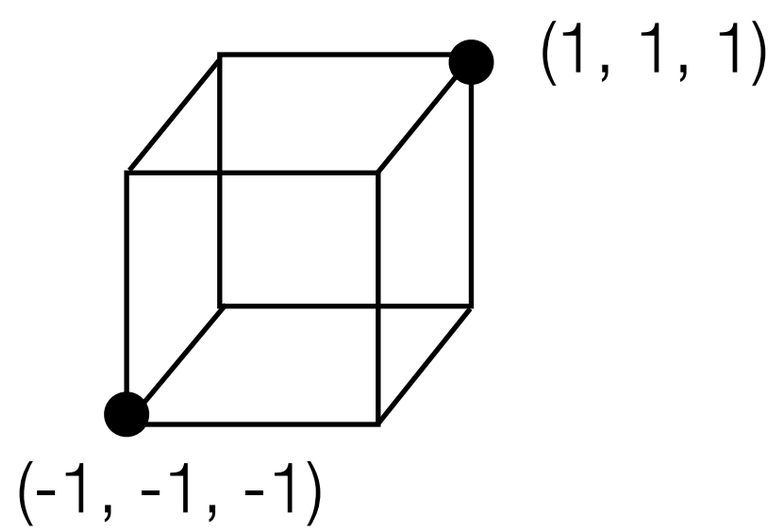
Clip Coordinates

Understanding the Perspective Projection Effect in WebGL

Note:
Frustum in camera coordinates
is transformed into
box in clip coordinates



Camera Coordinates



Clip Coordinates

Understanding the Perspective Projection Effect in WebGL

left can be +ve or -ve

right can be +ve or -ve

top can be +ve or -ve

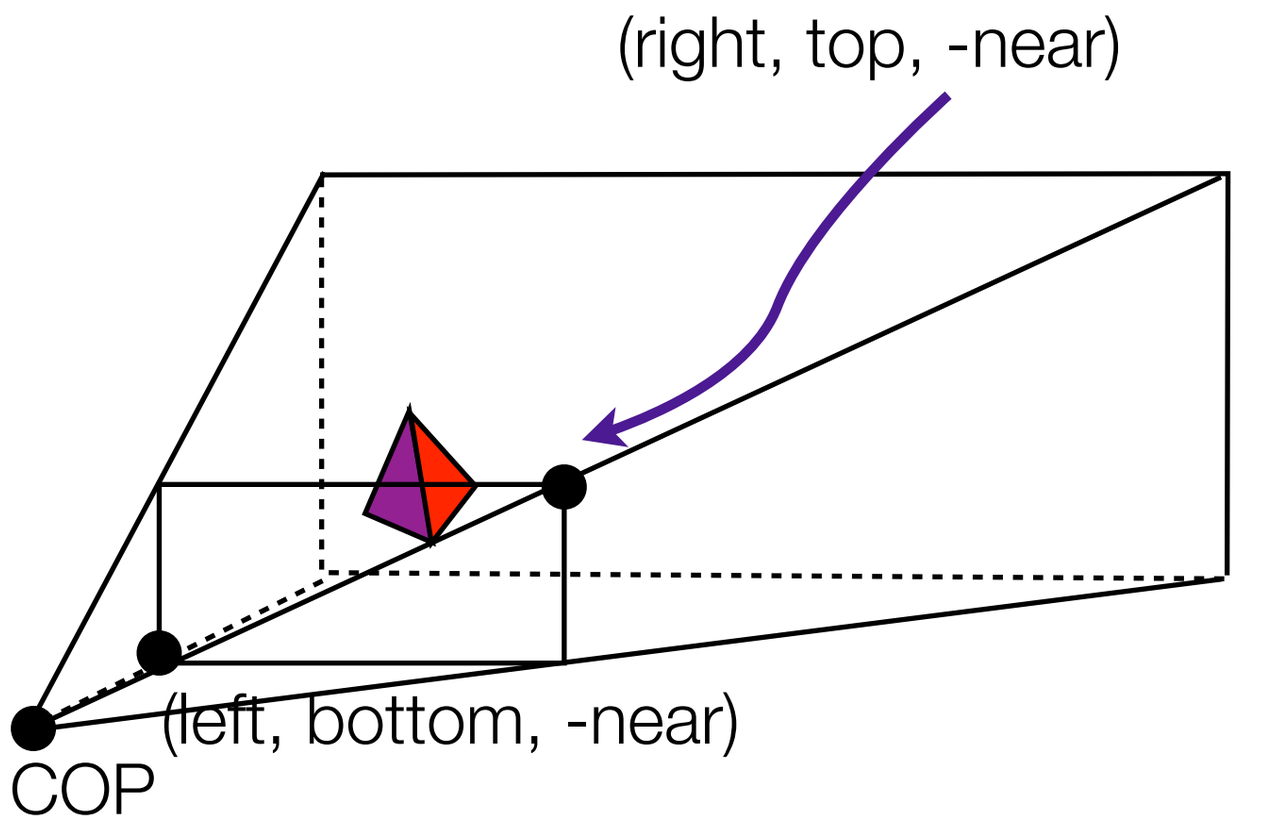
bottom can be +ve or -ve

(-near) must be -ve

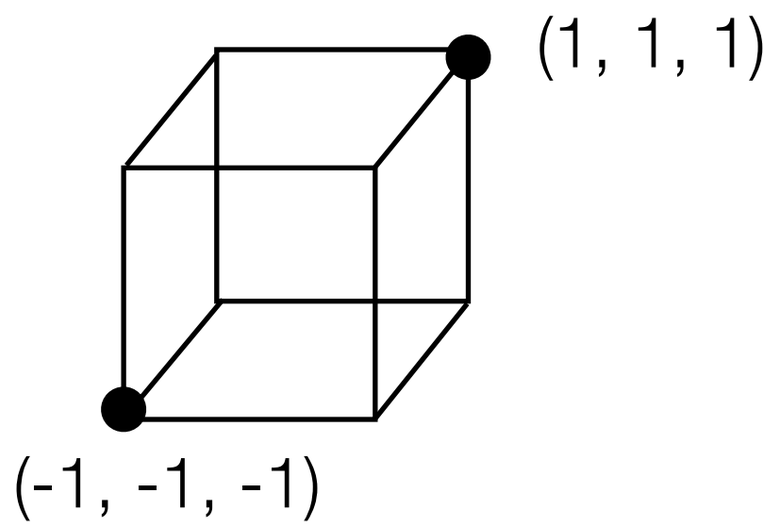
near must be +ve

(-far) must be -ve

far must be +ve



Camera Coordinates



Clip Coordinates

Understanding the Perspective Projection Effect in WebGL

left can be +ve or -ve

right can be +ve or -ve

top can be +ve or -ve

bottom can be +ve or -

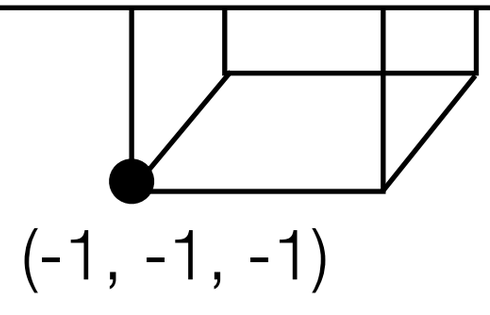
(-near) must be -ve

near must be +ve

(-far) must be -ve

far must be +ve

In perspective projection
ensuring that (-near) and (-far)
are both negative
assures that the center of projection
lies ahead of (-near)
and encompasses the whole scene.



Clip
Coordinates

Understanding the Perspective Projection Effect in WebGL

left can be +ve or -ve

right can be +ve or -ve

top can be +ve or -ve

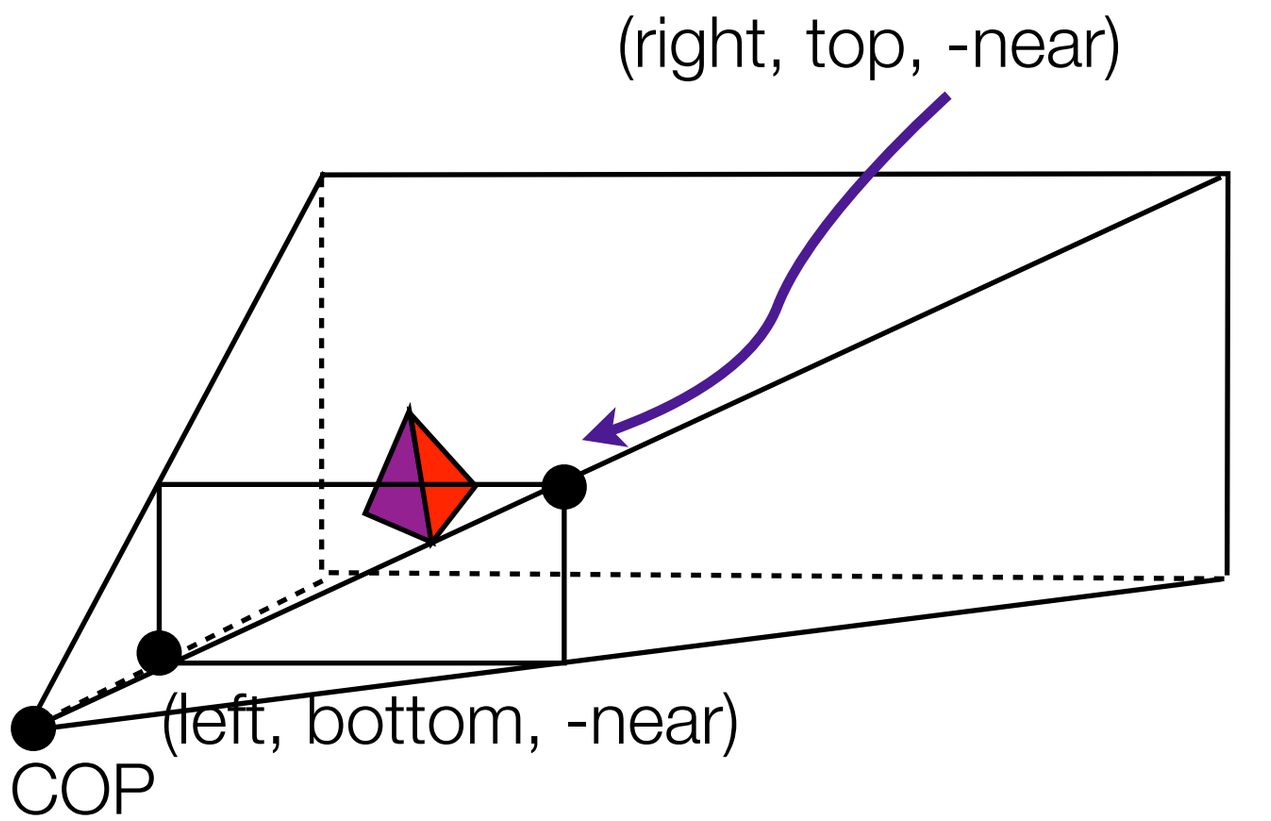
bottom can be +ve or -ve

(-near) must be -ve

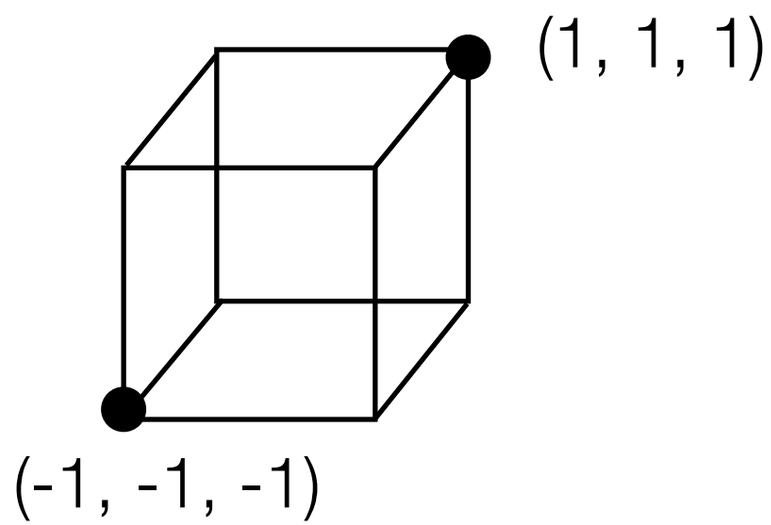
near must be +ve

(-far) must be -ve

far must be +ve

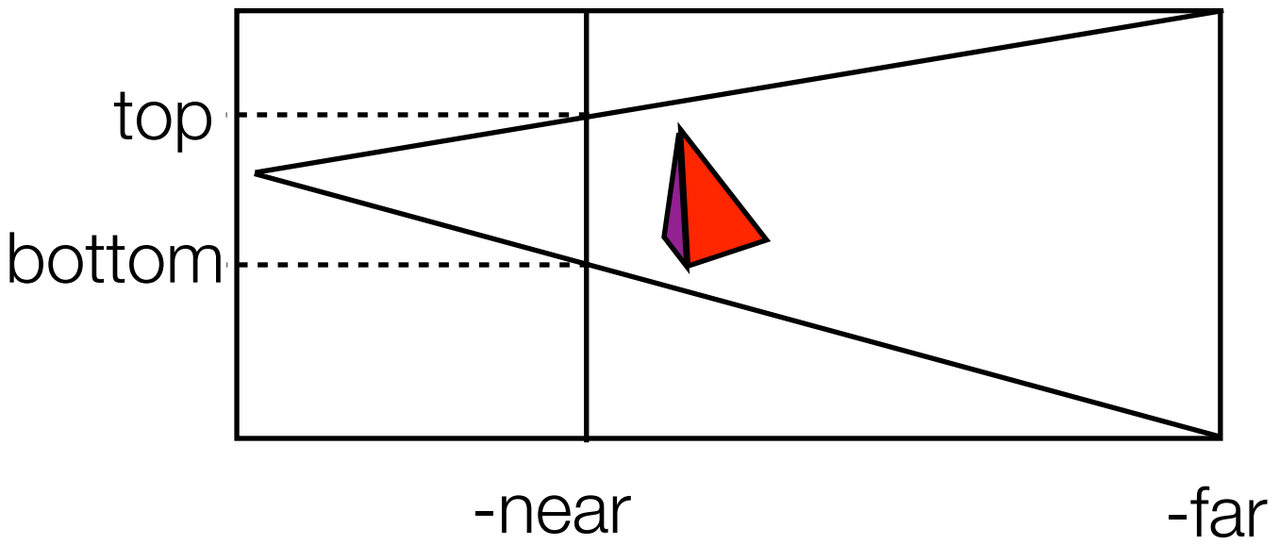


Camera Coordinates

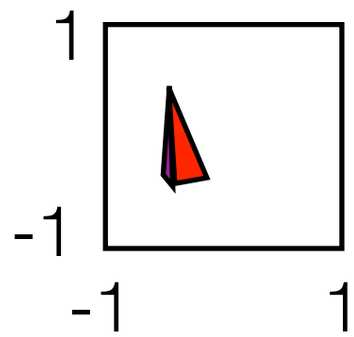


Clip Coordinates

Side View of Perspective Projection

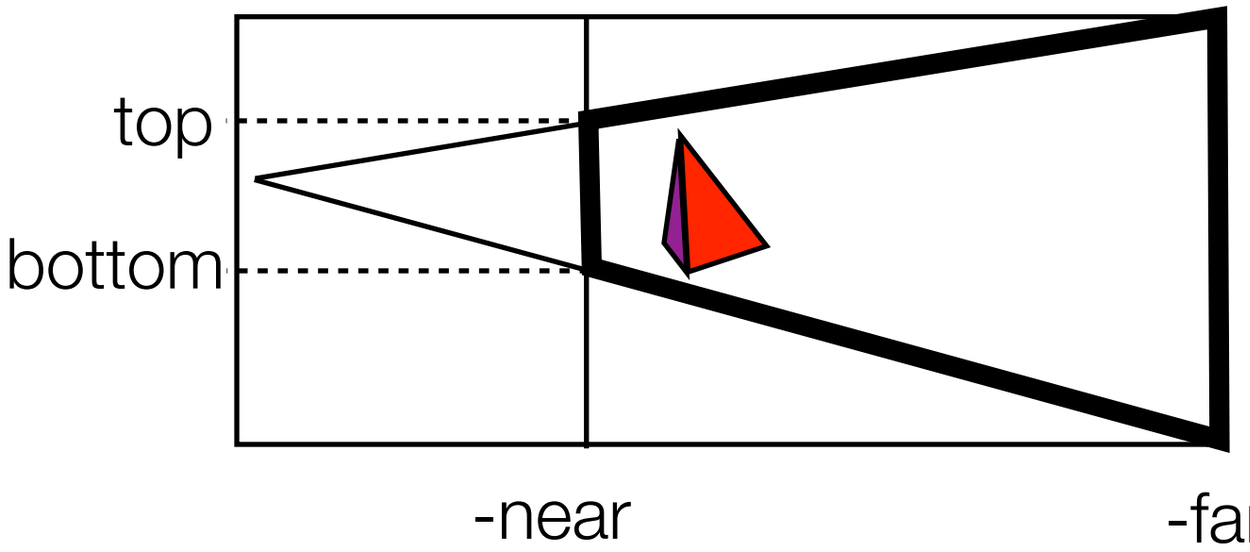


Camera Coordinates

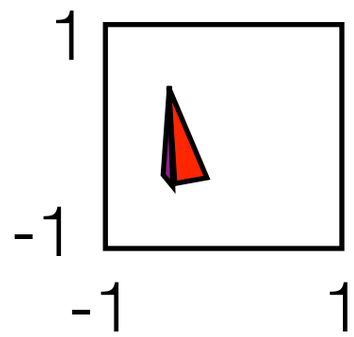


Clip Coordinates

Side View of Perspective Projection



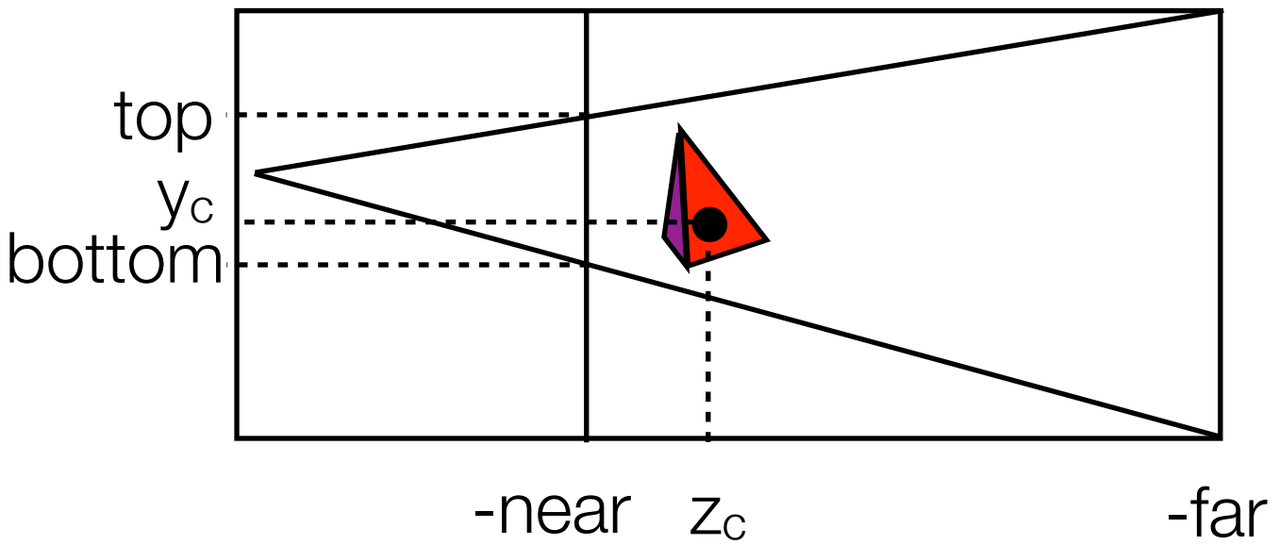
Camera Coordinates



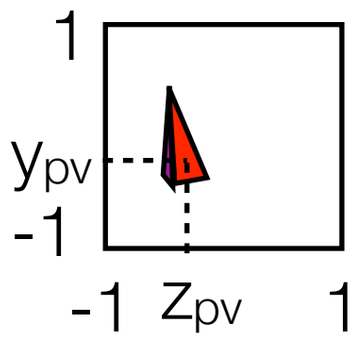
Clip Coordinates

Everything in highlighted view volume is considered for rendering.

Side View of Perspective Projection

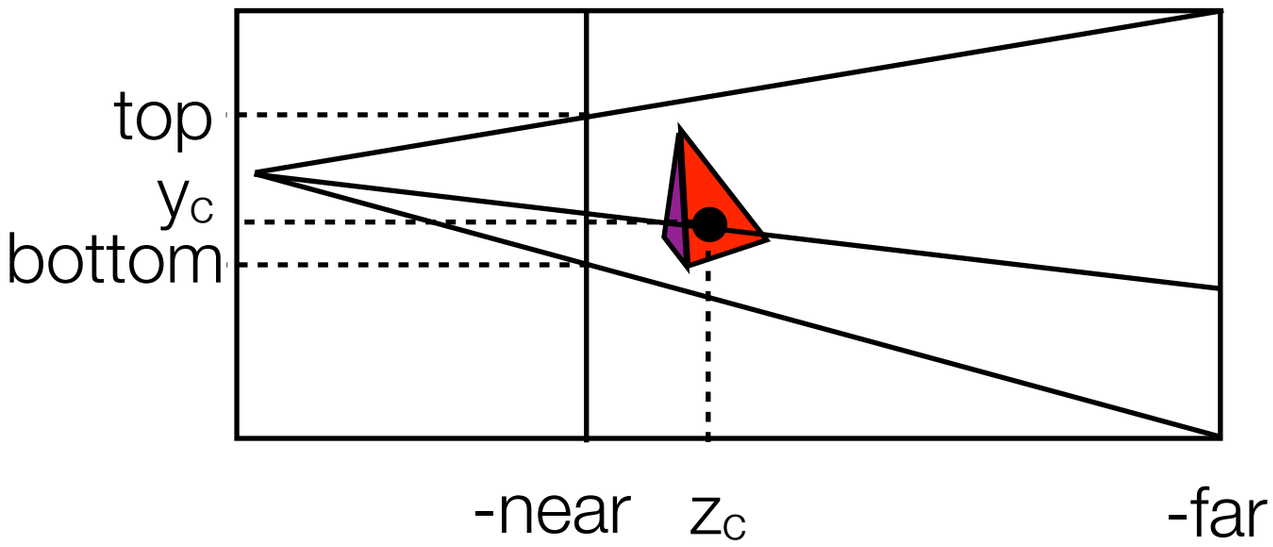


Camera Coordinates

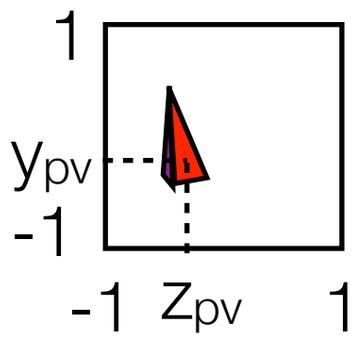


Clip Coordinates

Side View of Perspective Projection



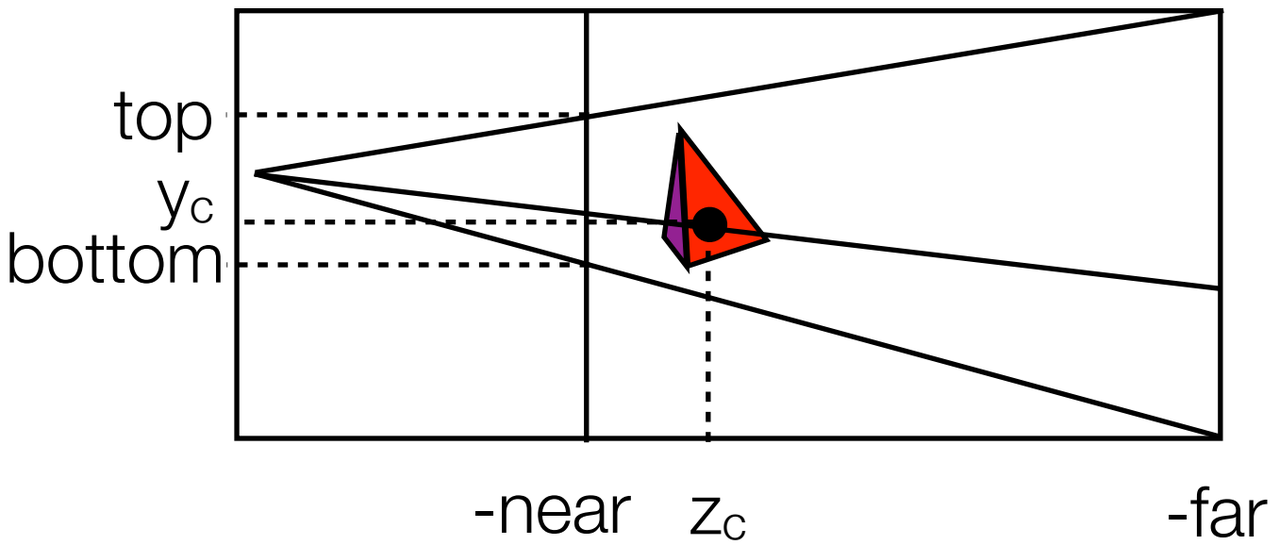
Camera Coordinates



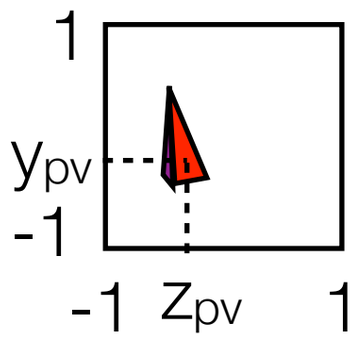
Clip Coordinates

Algebra is more challenging.

Side View of Perspective Projection



Camera Coordinates



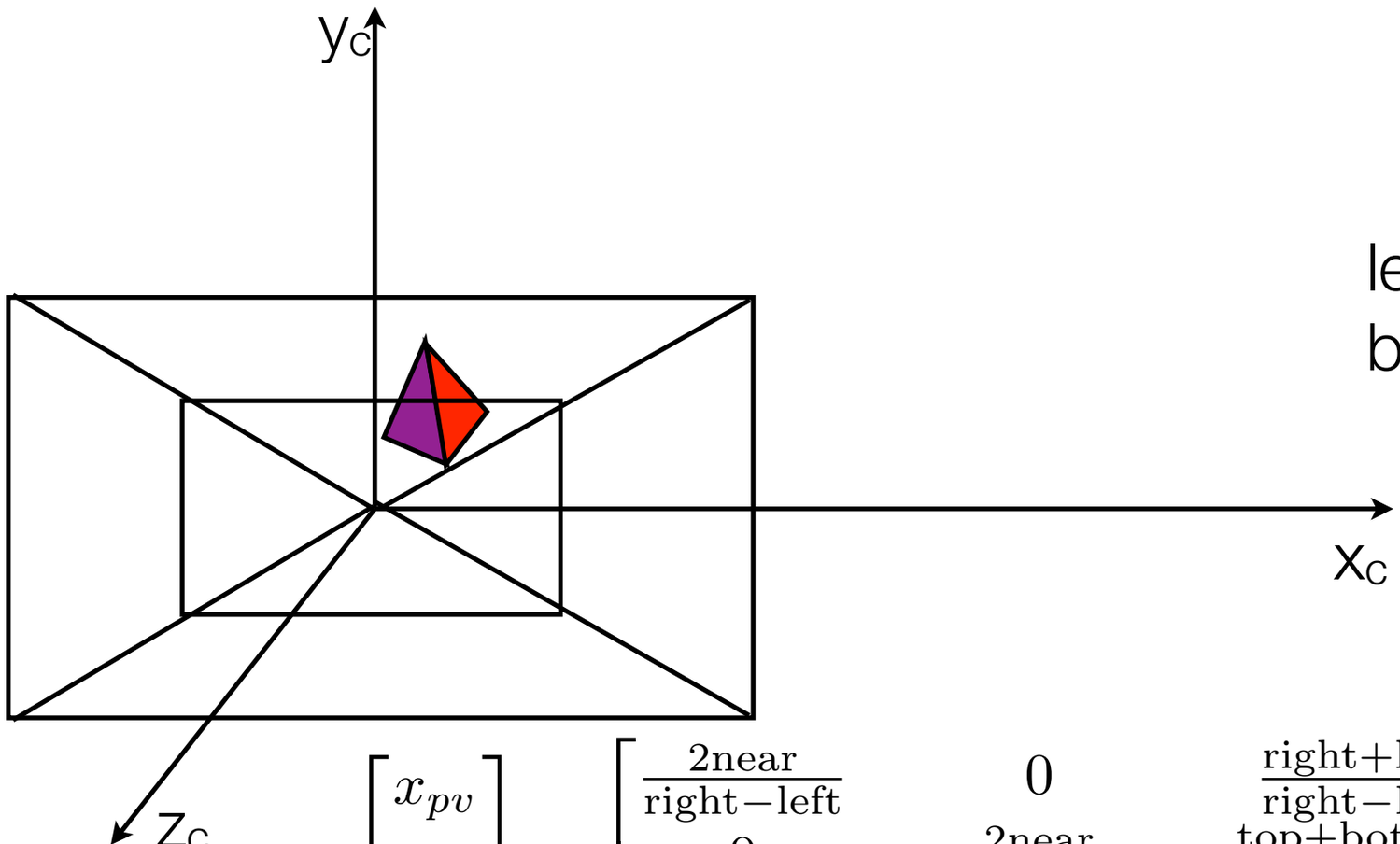
Clip Coordinates

Algebra is more challenging.

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2\text{near}}{\text{right}-\text{left}} & 0 & \frac{\text{right}+\text{left}}{\text{right}-\text{left}} & 0 \\ 0 & \frac{2\text{near}}{\text{top}-\text{bottom}} & \frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

Viewing volume symmetric about z axis



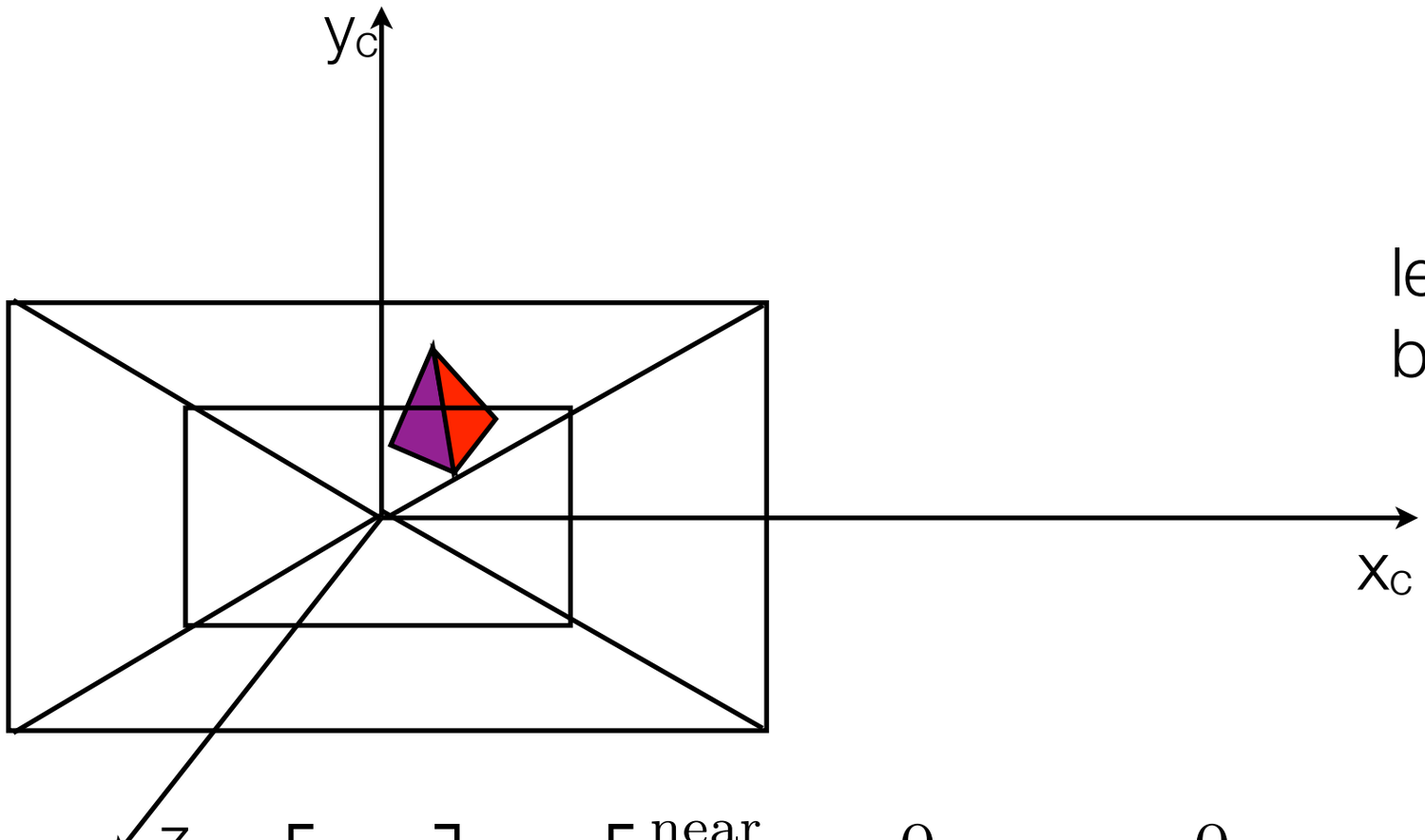
left = -right

bottom = -top

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{2\text{near}}{\text{right}-\text{left}} & 0 & \frac{\text{right}+\text{left}}{\text{right}-\text{left}} & 0 \\ 0 & \frac{2\text{near}}{\text{top}-\text{bottom}} & \frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

Viewing volume symmetric about z axis



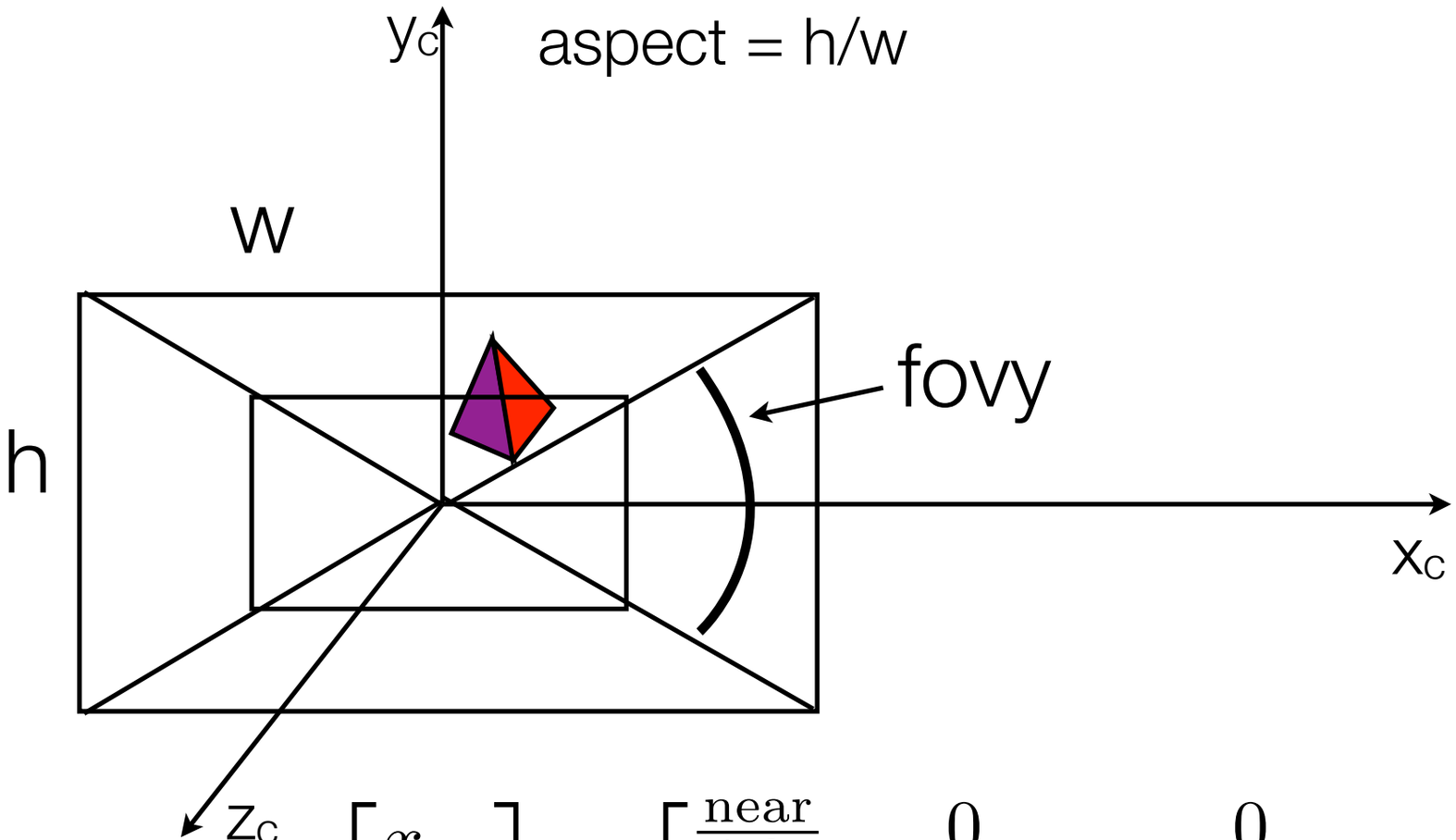
left = -right

bottom = -top

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{\text{near}}{\text{right}} & 0 & 0 & 0 \\ 0 & \frac{\text{near}}{\text{top}} & 0 & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

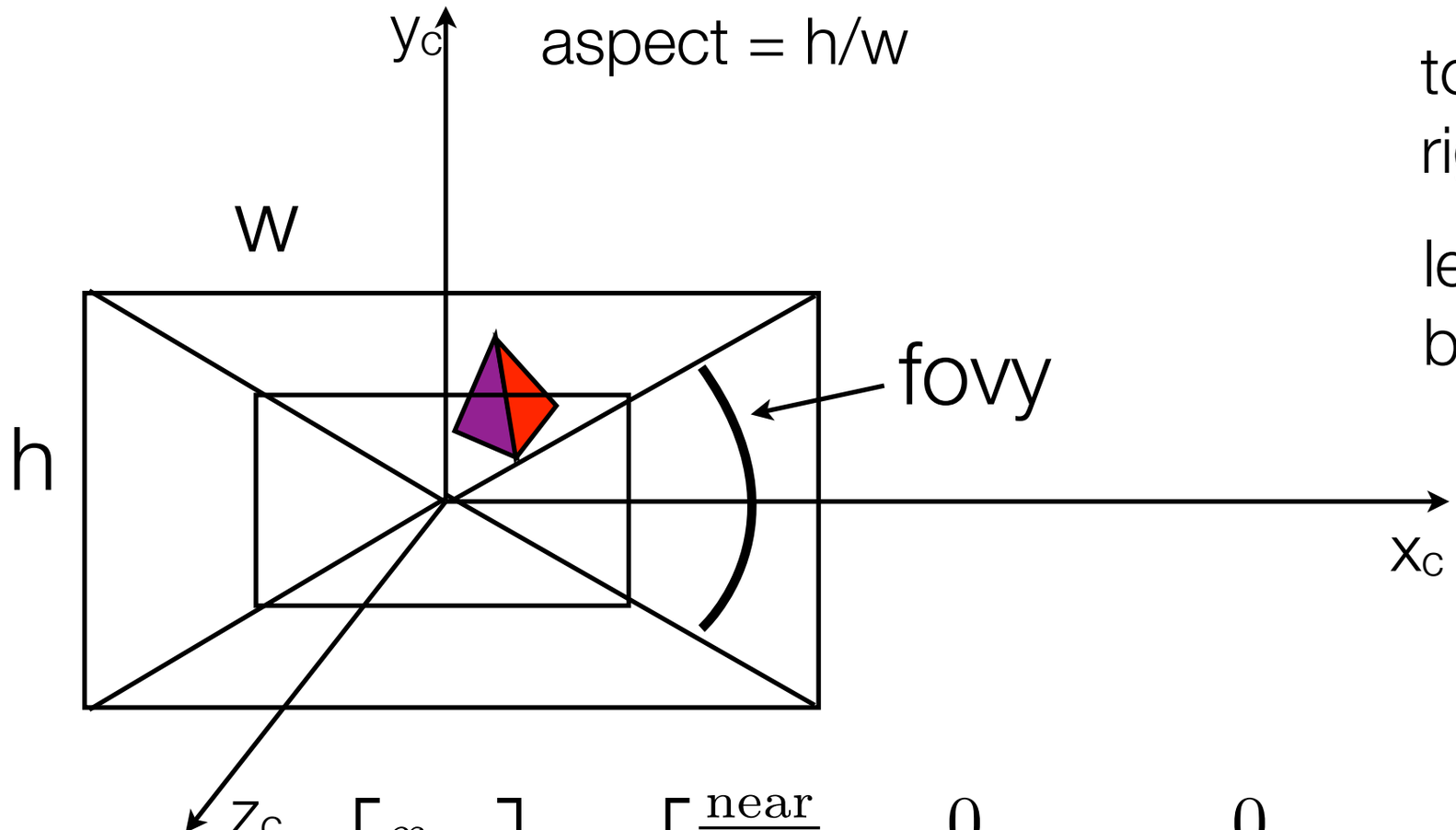
Sometimes you will have the aspect ratio (aspect) field of view angle for the y axis (fovy)



$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{\text{near}}{\text{right}} & 0 & 0 & 0 \\ 0 & \frac{\text{near}}{\text{top}} & 0 & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Perspective Projection

You can get top and right from near, far, fovy, and aspect



top = near * tan(fovy)
right = top * aspect

left = -right
bottom = -top

$$\begin{bmatrix} x_{pv} \\ y_{pv} \\ z_{pv} \\ w_{pv} \end{bmatrix} = \begin{bmatrix} \frac{\text{near}}{\text{right}} & 0 & 0 & 0 \\ 0 & \frac{\text{near}}{\text{top}} & 0 & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

What to do in WebGL?

What to do in WebGL?

Set up projection matrix **P** for whichever projection you are doing (parallel or perspective).

What to do in WebGL?

Set up projection matrix **P** for whichever projection you are doing (parallel or perspective).

Multiply projection matrix **P to left of** model-view matrix **M**.

```
vf = P * M * vi;
```

What to do in WebGL?

Set up projection matrix **P** for whichever projection you are doing (parallel or perspective).

Multiply projection matrix **P to left of** model-view matrix **M**.

```
vf = P * M * vi;
```

Note: You do not need to do the ‘divide by w’ thing. WebGL automatically does that for you.

Important Note for WebGL

We deal with right-handed coordinate systems
in computer graphics,
where the negative z axis points **away** from us.

However, WebGL uses a left-handed coordinate system
for its **clip coordinates**.

In particular, z=+1 is further away from us than z=-1,
or the negative z axis points **toward** us.

Important Note for WebGL

Another way to think about this is that
WebGL requires farther objects to have a higher value
of z_{pv} to do the depth test correctly.

Important Note for WebGL

Another way to think about this is that
WebGL requires farther objects to have a higher value
of z_{pv} to do the depth test correctly.

The perspective projection matrices handle this
by introducing a negation for z_{pv}.

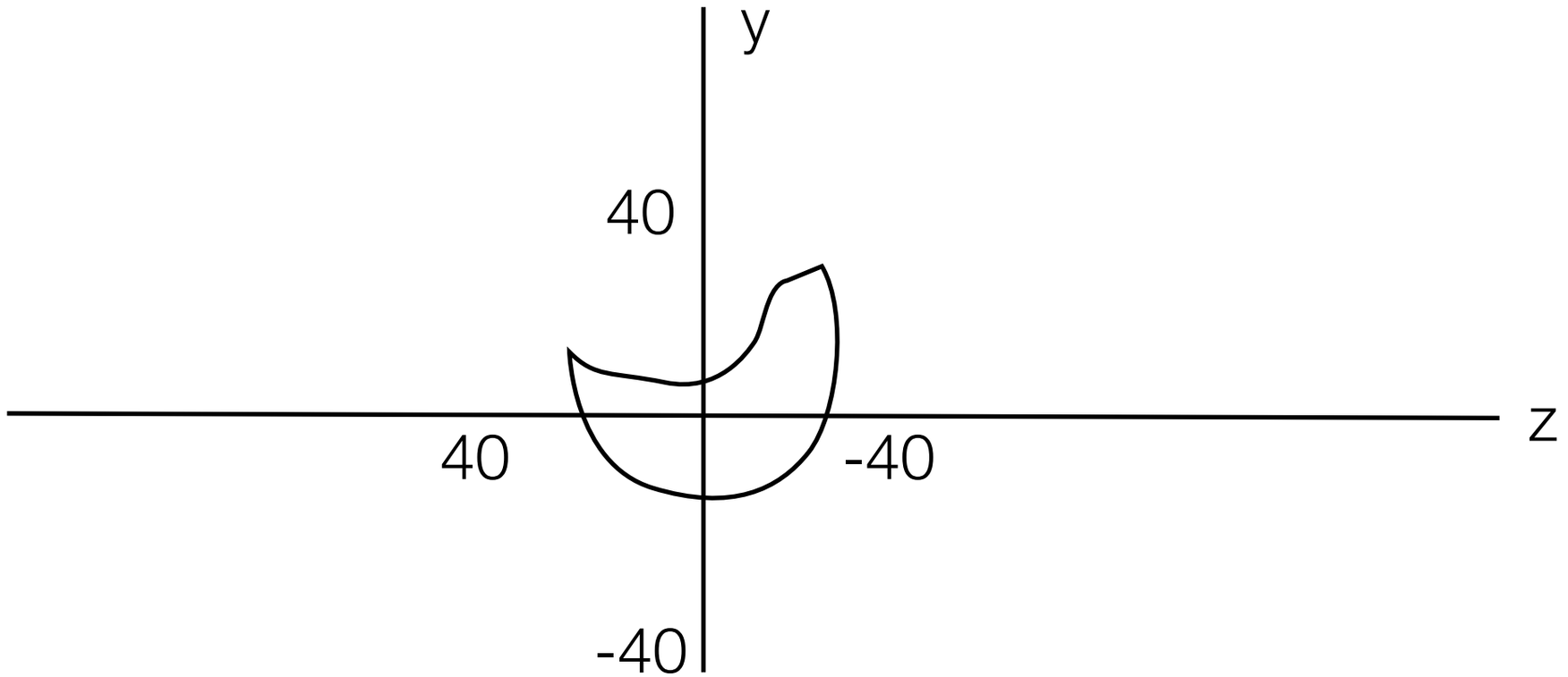
Simpler Version of Lab 4

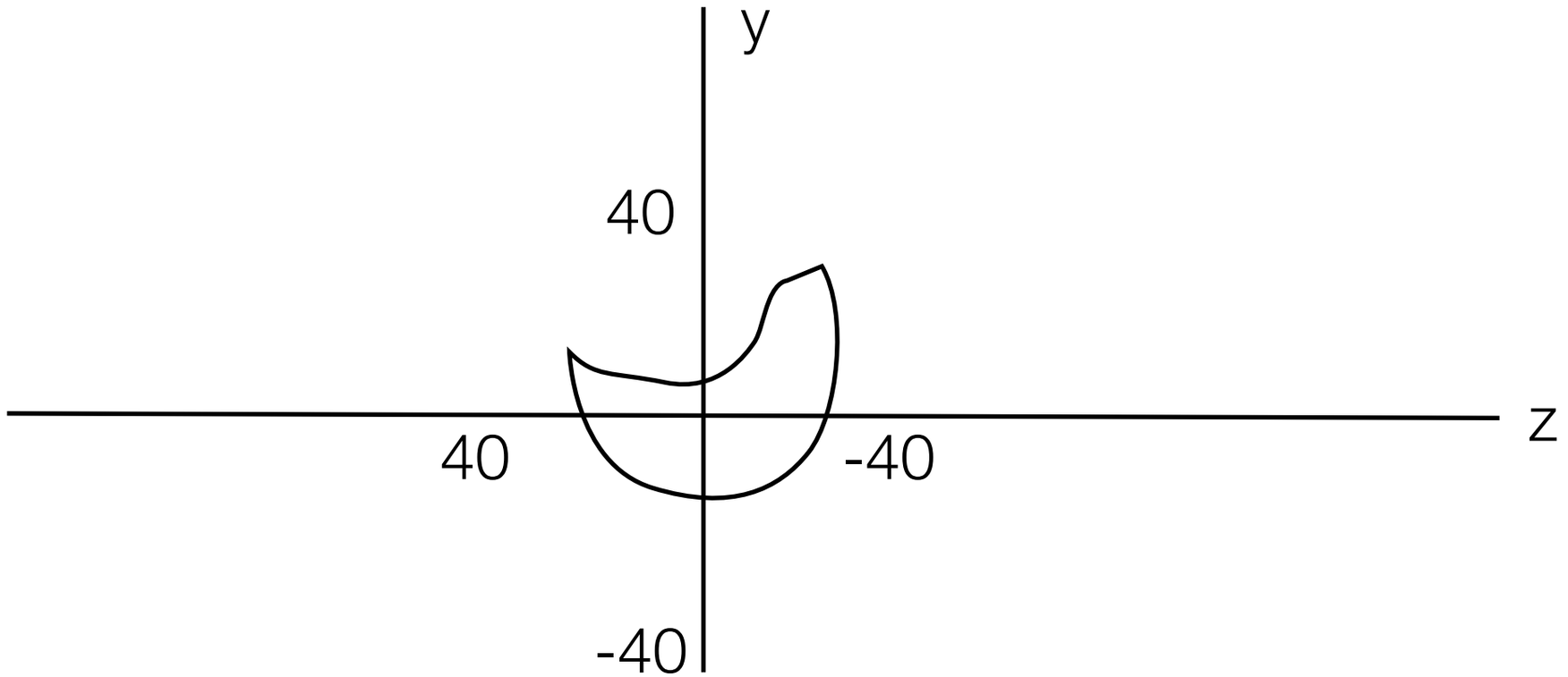
Ball Chair

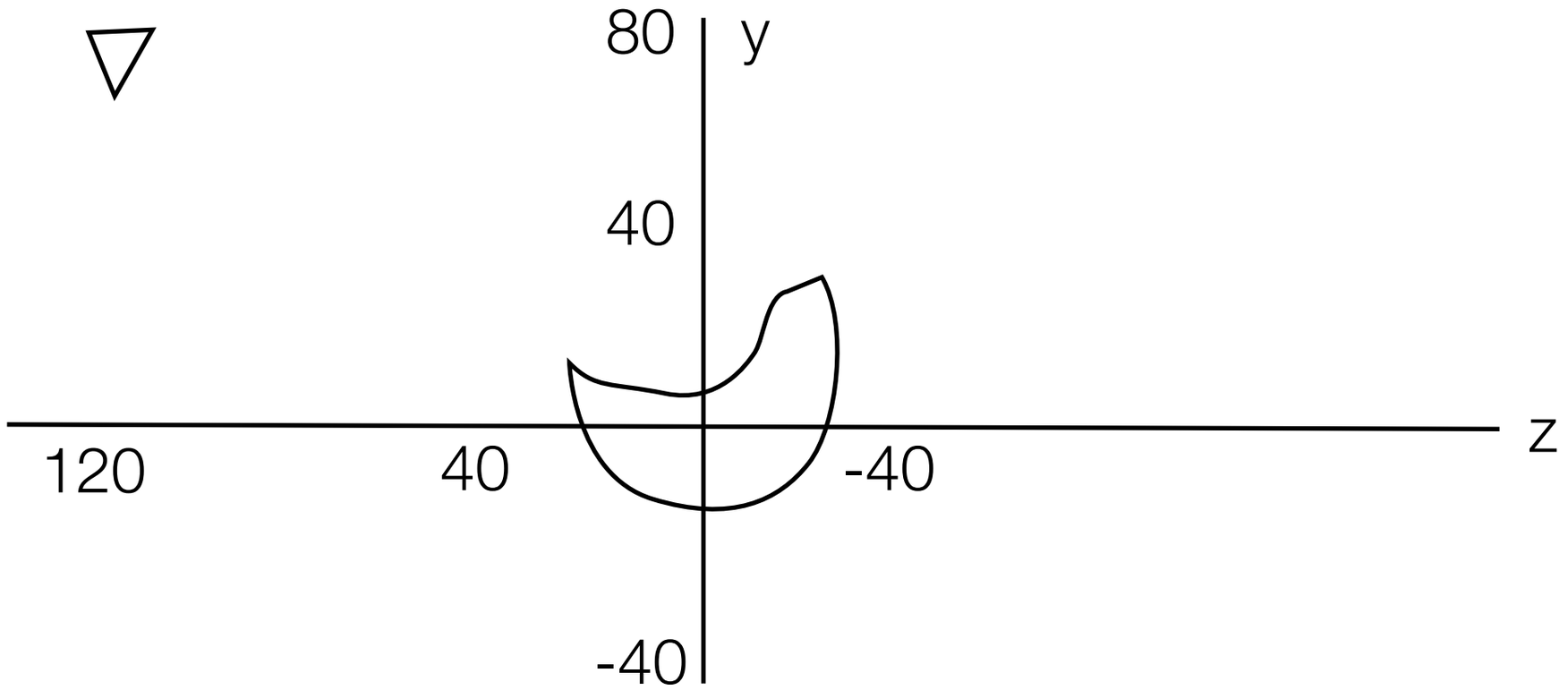
In Lab 4, you only need view and shading, no texture.

Simpler Version of Lab 4

Side view of coordinate system

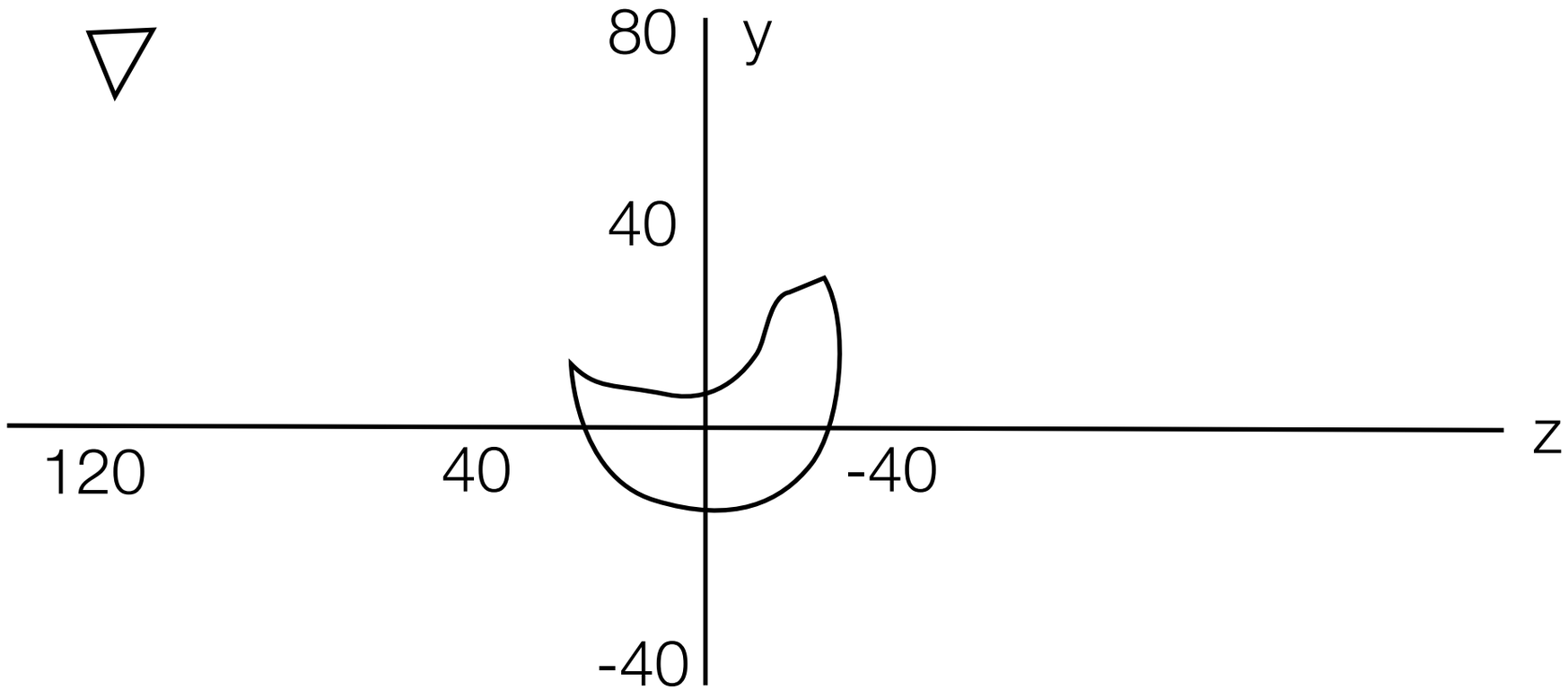



y
40
z
40
-40
-40



$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

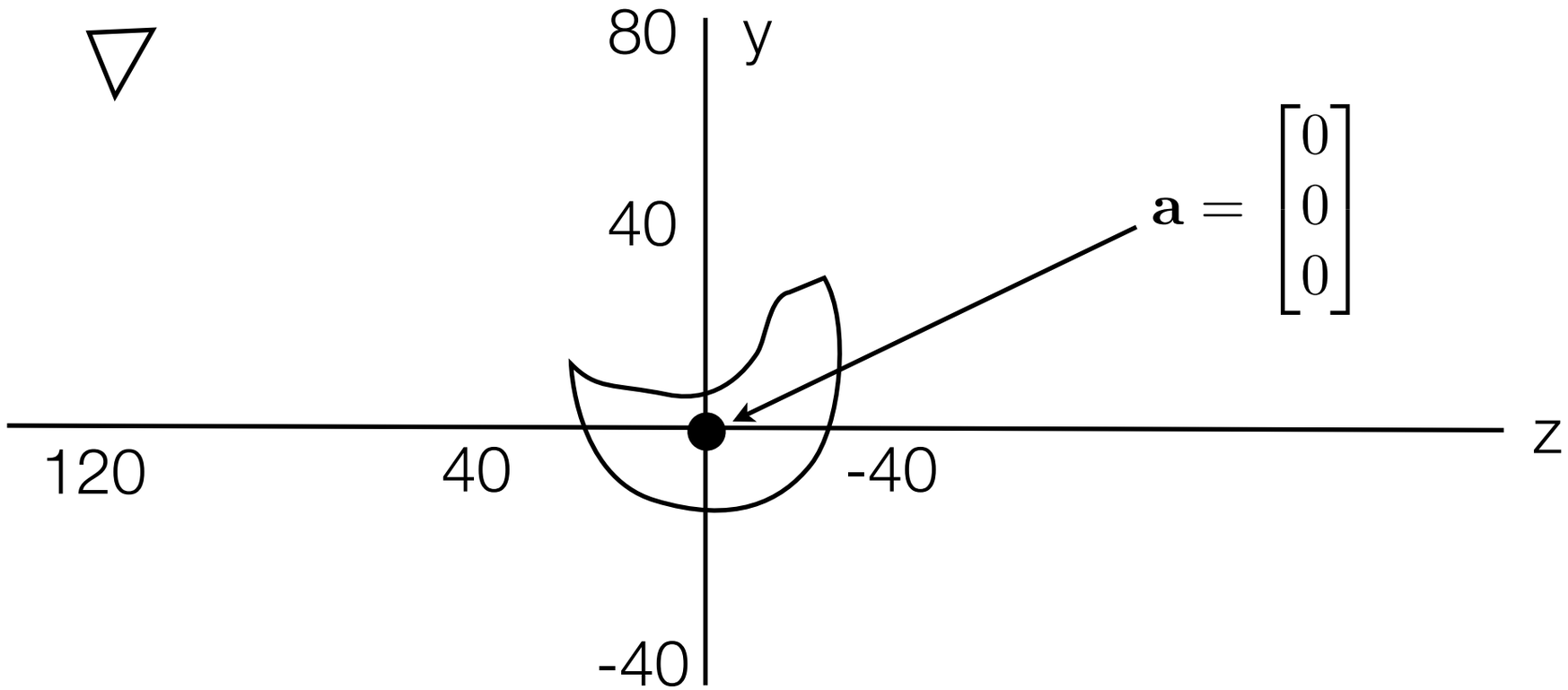
$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



‘**a**’ (at point) can be anything within the vicinity of the chair

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

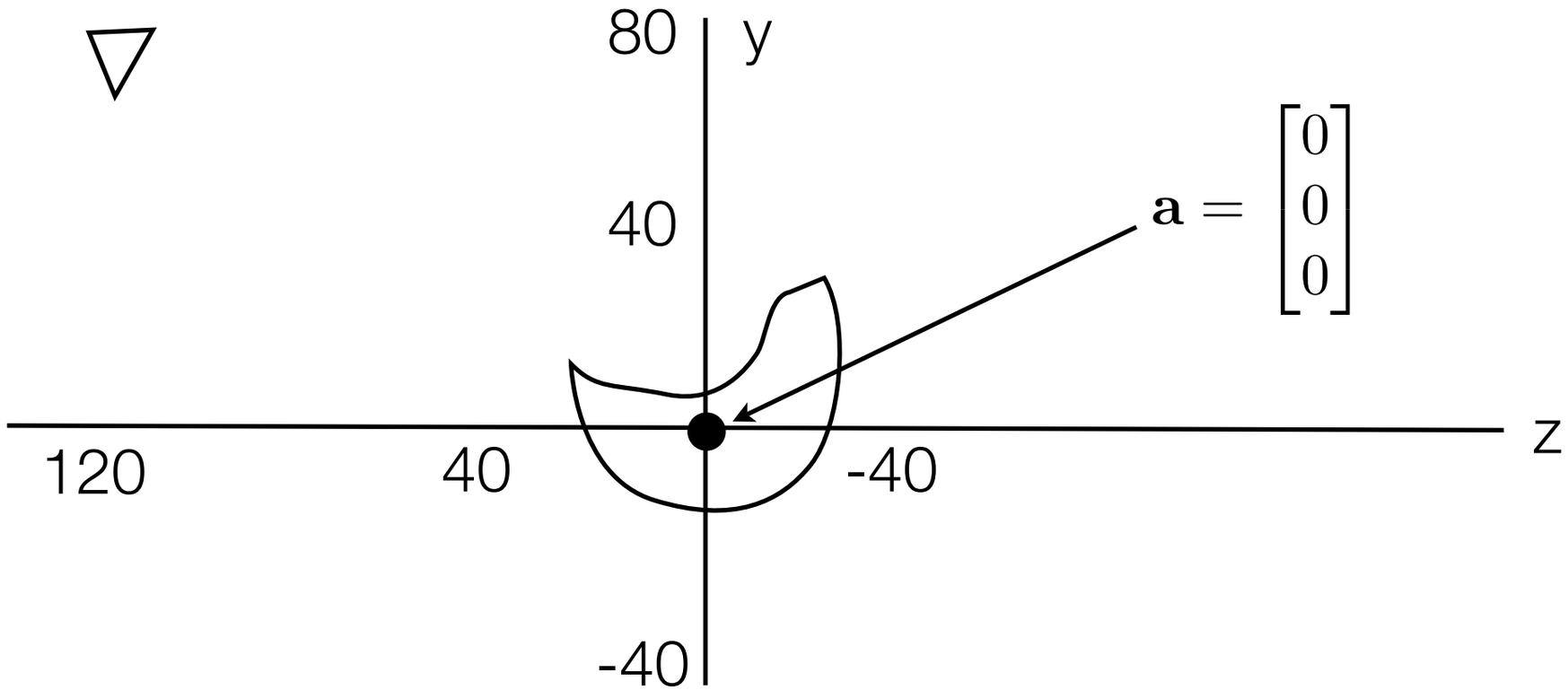
$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



‘**a**’ (at point) can be anything within the vicinity of the chair

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$

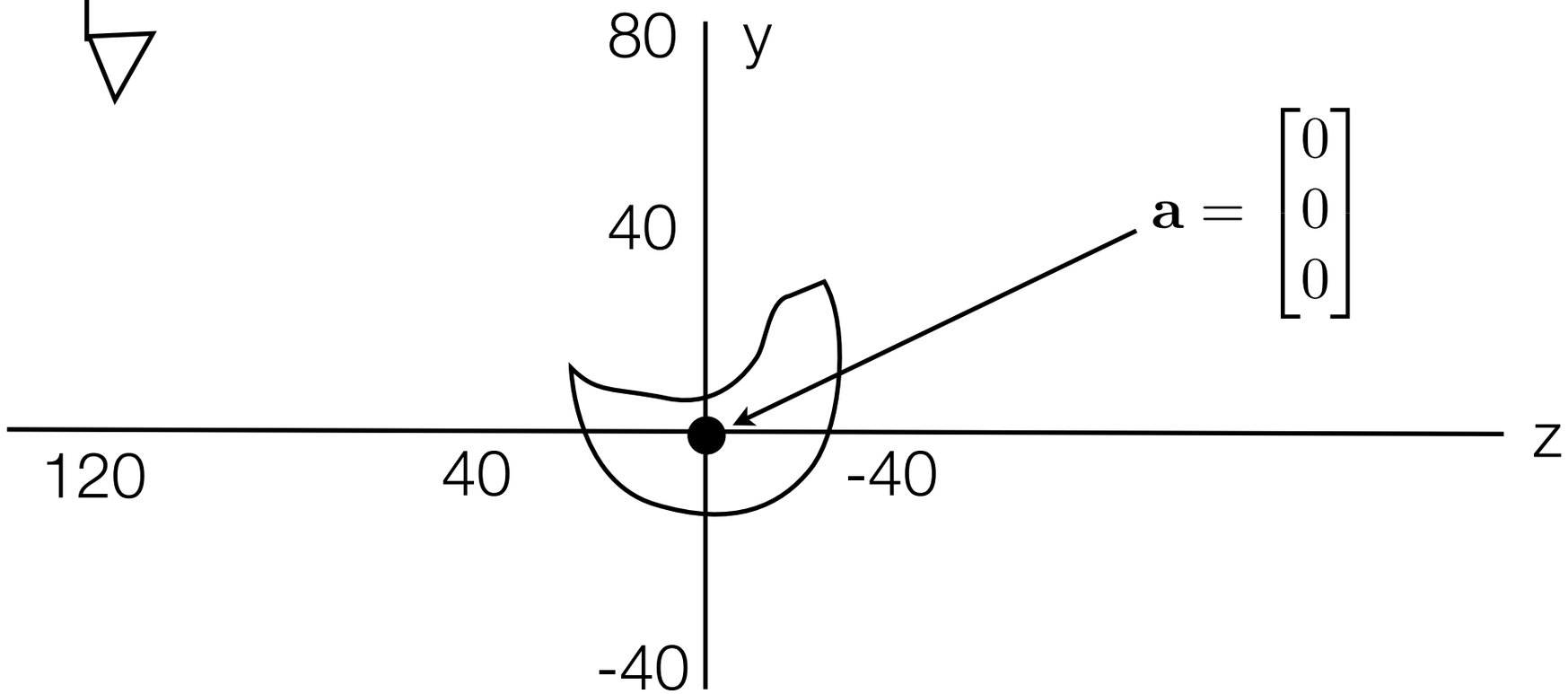


‘$\mathbf{v}_{\text{up}}$’ (up vector)
along positive y axis.

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_{\text{up}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



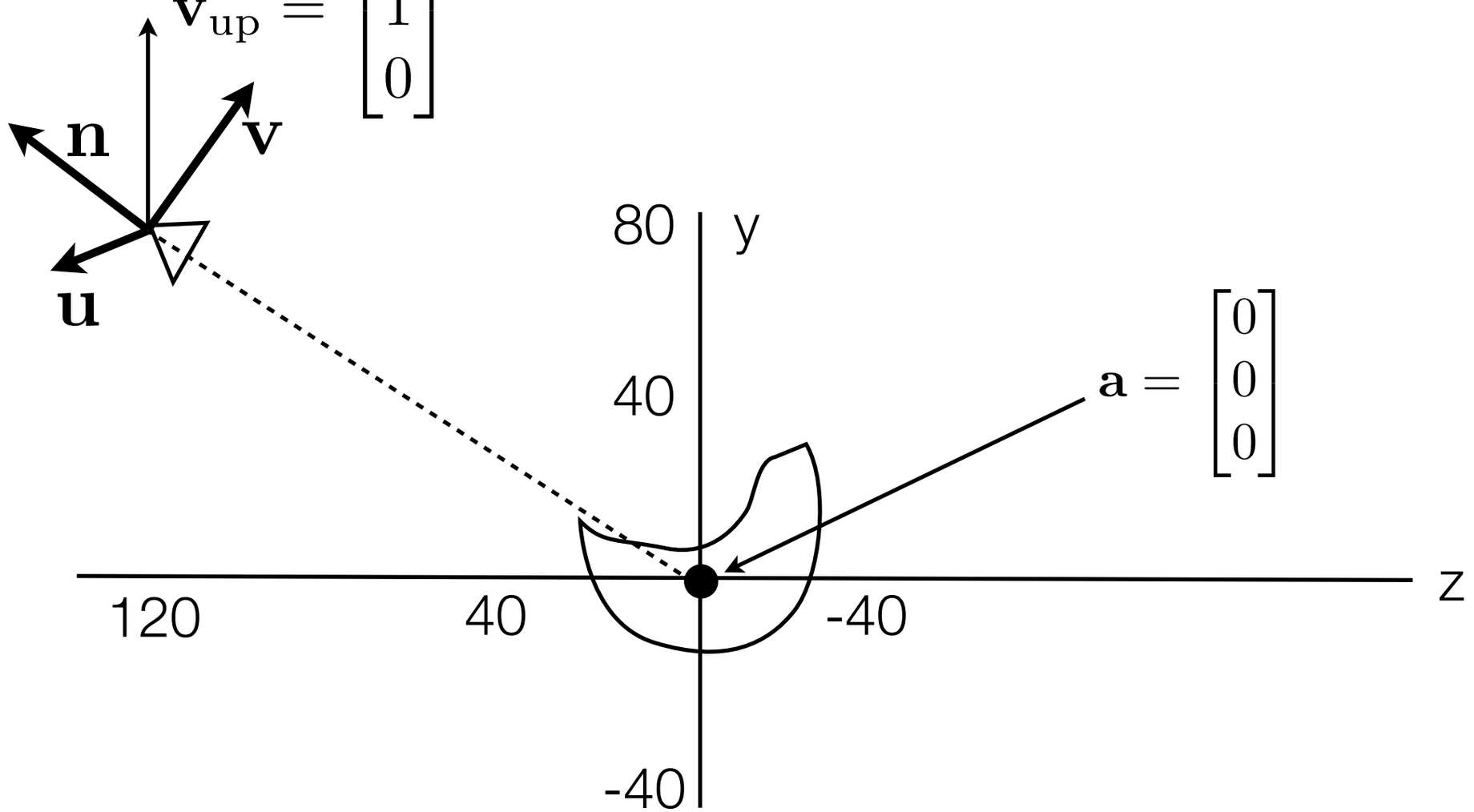
$$\mathbf{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

‘$\mathbf{v}_{\text{up}}$’ (up vector)
along positive y axis.

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_{\text{up}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



$$\mathbf{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{d} = \mathbf{e} - \mathbf{a},\ \ \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

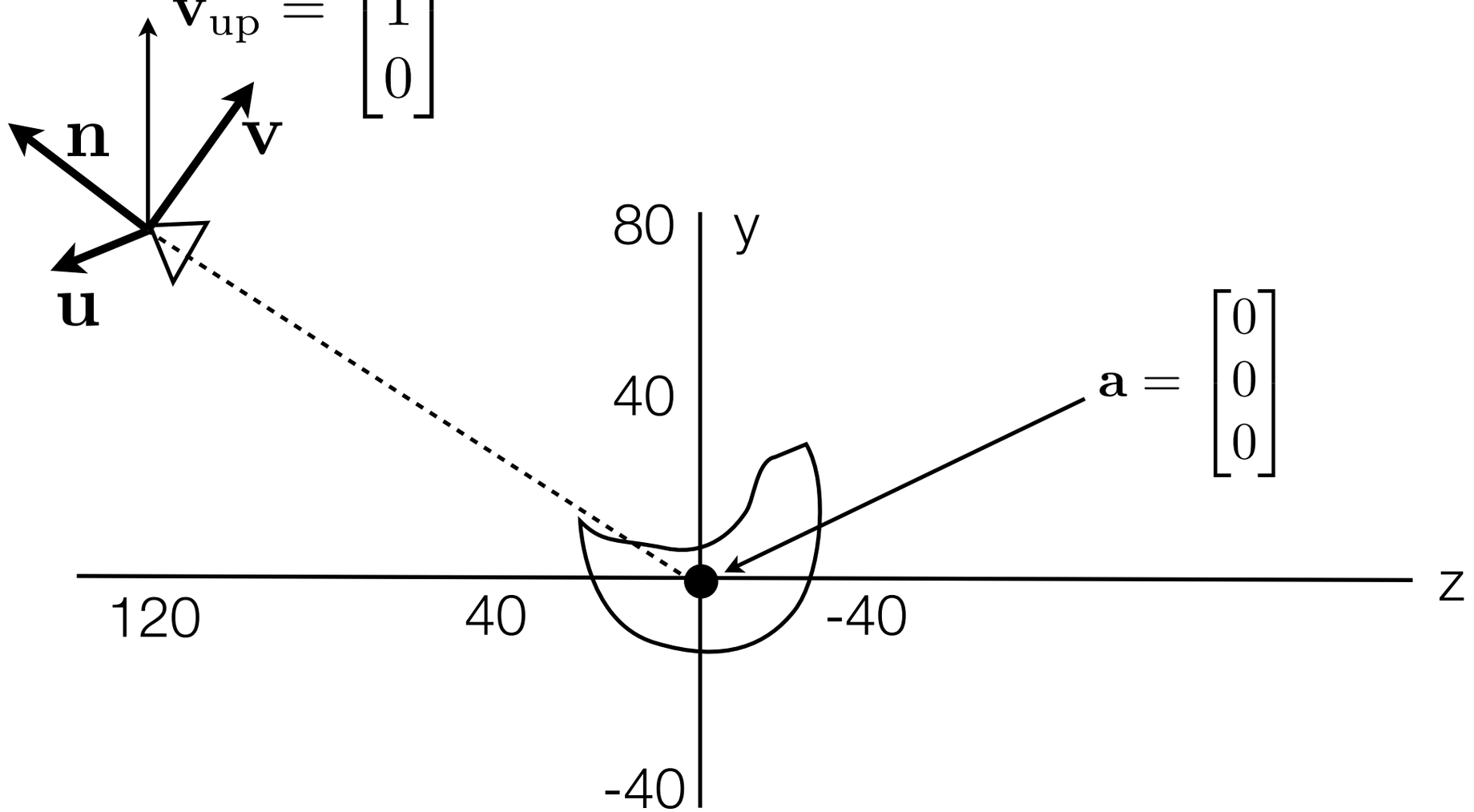
$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n},\ \ \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u},\ \ \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_{\text{up}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



$$\mathbf{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Note:
Lab requires a different **e**!

$$\mathbf{d} = \mathbf{e} - \mathbf{a}, \;\; \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

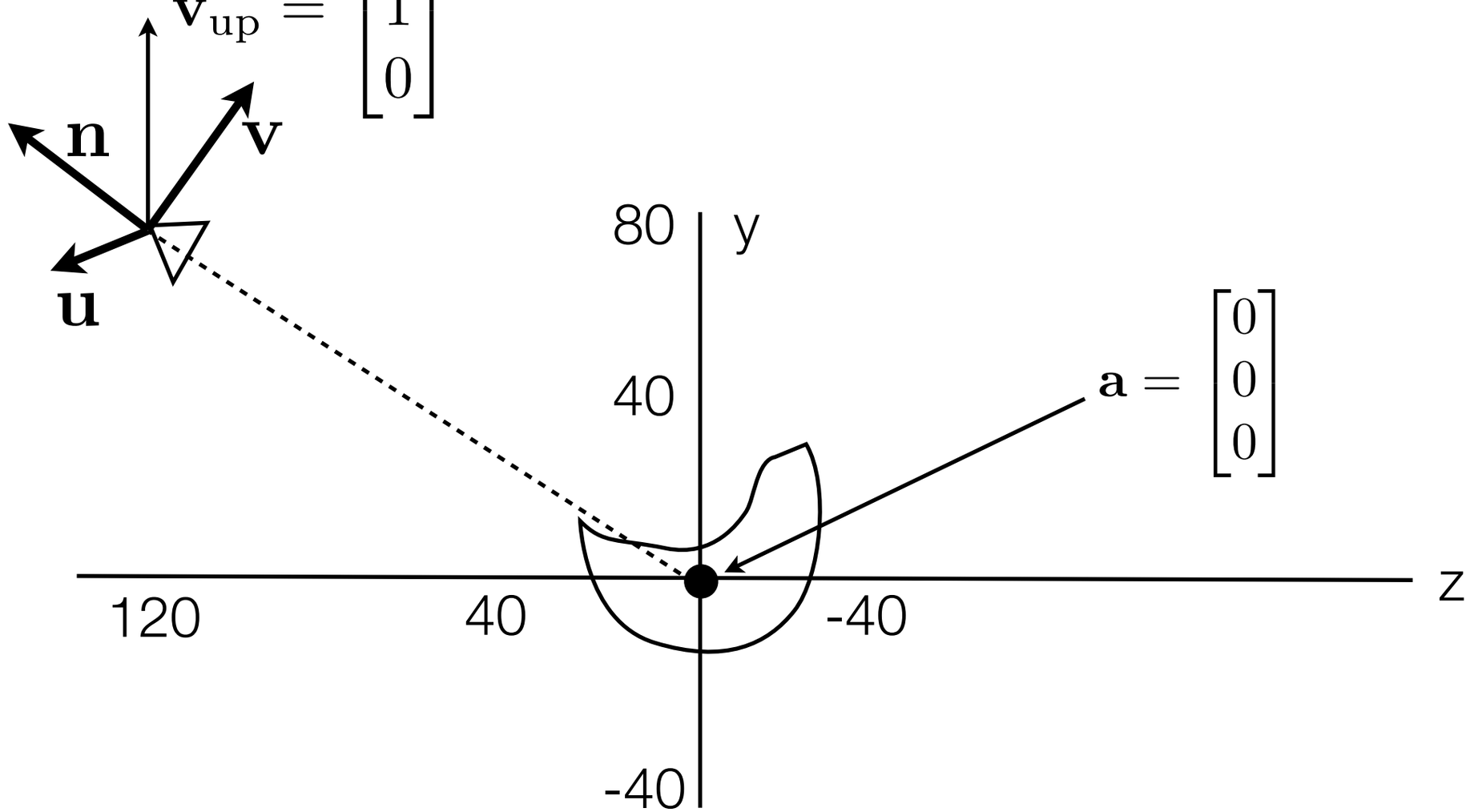
$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n}, \;\; \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u}, \;\; \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v}_{\text{up}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} 0 \\ 80 \\ 120 \end{bmatrix}$$



$$\mathbf{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

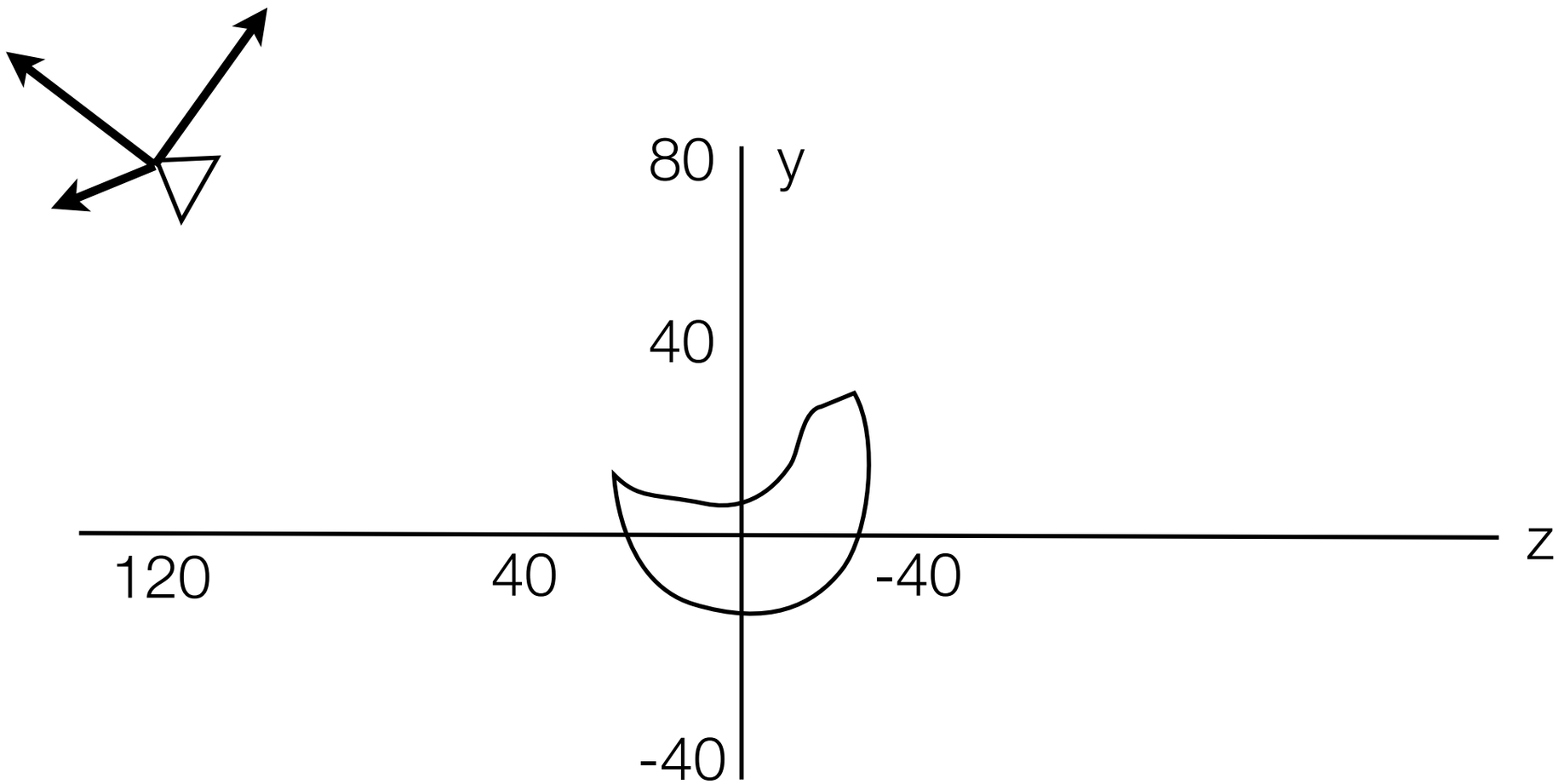
$$\mathbf{d} = \mathbf{e} - \mathbf{a},\ \ \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n},\ \ \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u},\ \ \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In WebGL, the coordinate system changes to be centered at the camera.



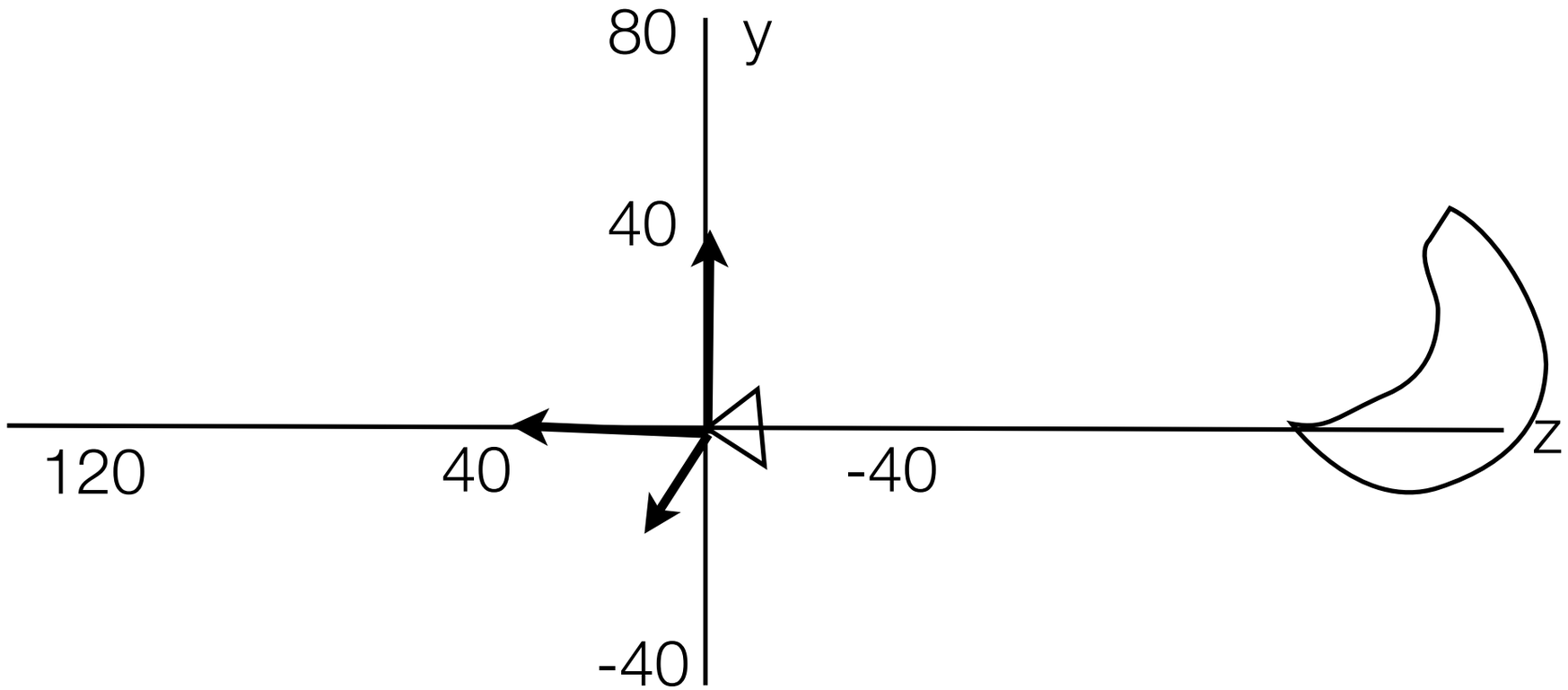
$$\mathbf{d} = \mathbf{e} - \mathbf{a},\ \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n},\ \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u},\ \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The object is transformed further out along the negative z-axis.



$$\mathbf{d} = \mathbf{e} - \mathbf{a},\ \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

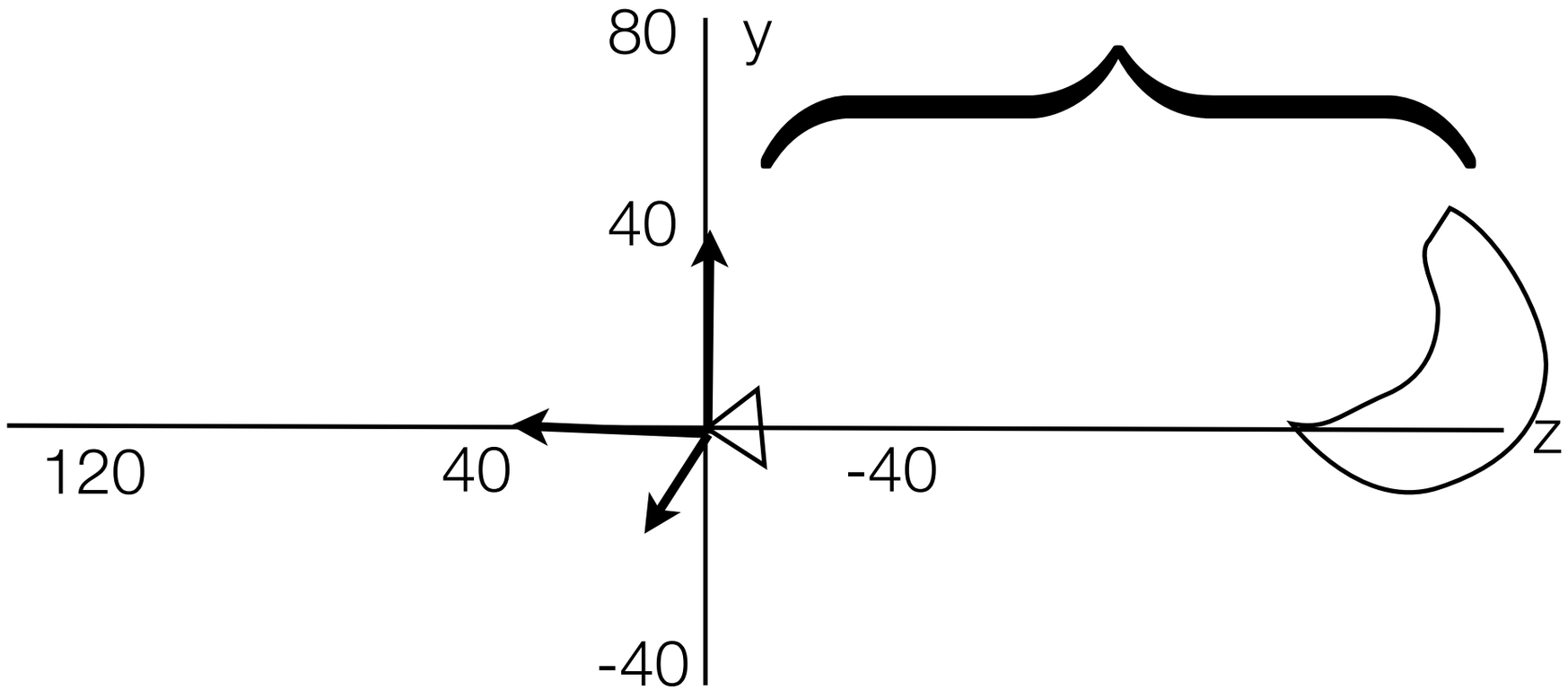
$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n},\ \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u},\ \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The object is transformed further out along the negative z-axis.

$$\|\mathbf{e}-\mathbf{a}\| = \sqrt{(0-0)^2+(120-0)^2+(80-0)^2} = 144.22$$



$$\mathbf{d} = \mathbf{e}-\mathbf{a},\ \mathbf{n} = \frac{\mathbf{d}}{\|\mathbf{d}\|}$$

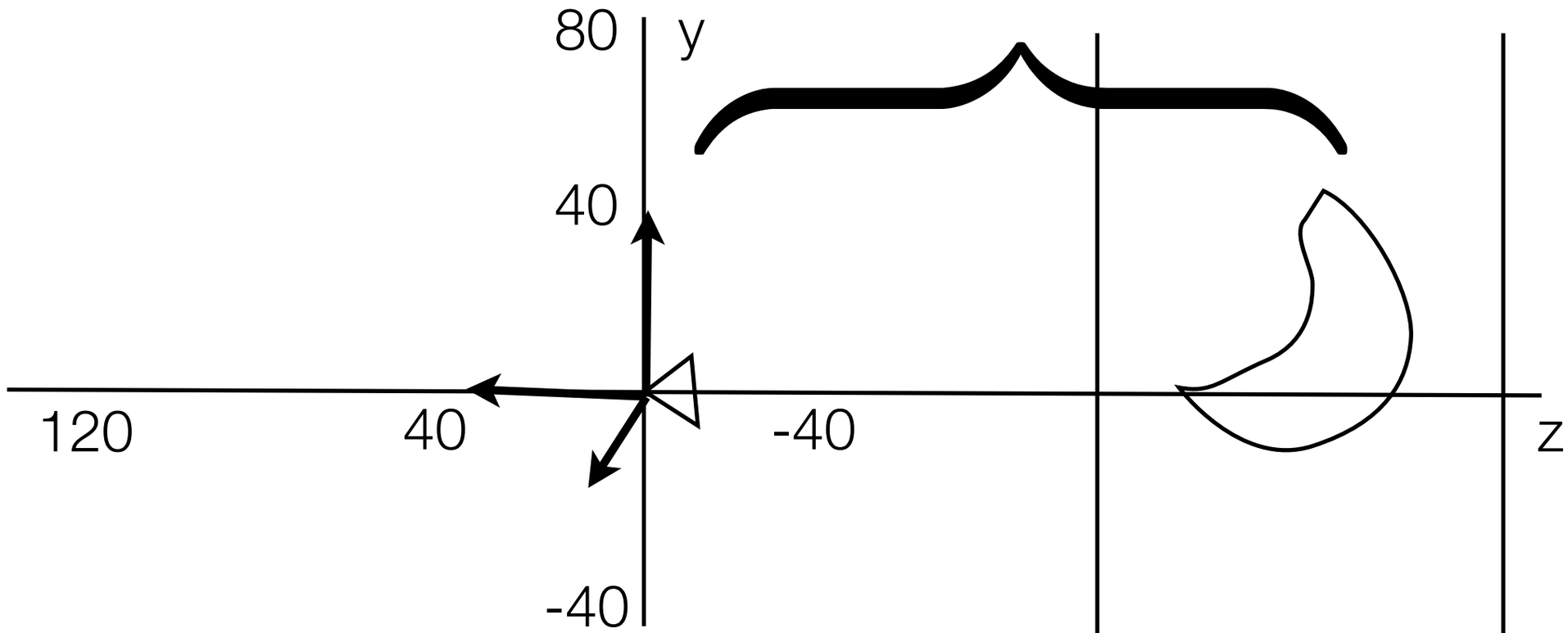
$$\mathbf{k} = \mathbf{v}_{\text{up}} \times \mathbf{n},\ \mathbf{u} = \frac{\mathbf{k}}{\|\mathbf{k}\|}$$

$$\mathbf{l} = \mathbf{n} \times \mathbf{u},\ \mathbf{v} = \frac{\mathbf{l}}{\|\mathbf{l}\|}$$

$$\mathbf{M} = \begin{bmatrix} u_x & u_y & u_z & -e_x u_x - e_y u_y - e_z u_z \\ v_x & v_y & v_z & -e_x v_x - e_y v_y - e_z v_z \\ n_x & n_y & n_z & -e_x n_x - e_y n_y - e_z n_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The object is transformed further out along the negative z-axis.

$$\|\mathbf{e}-\mathbf{a}\| = \sqrt{(0-0)^2+(120-0)^2+(80-0)^2} = 144.22$$



-near = -144.22+ about 50
near = 144.22 - about 50

-far = -144.22 - about 50
far = 144.22 + about 50

$$\mathbf{P}_{\text{persp}} = \begin{bmatrix} \frac{2\text{near}}{\text{right}-\text{left}} & 0 & \frac{\text{right}+\text{left}}{\text{right}-\text{left}} & 0 \\ 0 & \frac{2\text{near}}{\text{top}-\text{bottom}} & \frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} & 0 \\ 0 & 0 & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} & -\frac{2\ \text{far}\ \text{near}}{\text{far}-\text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\mathbf{P}_{\text{orth}} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{left}+\text{right}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & -\frac{2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$