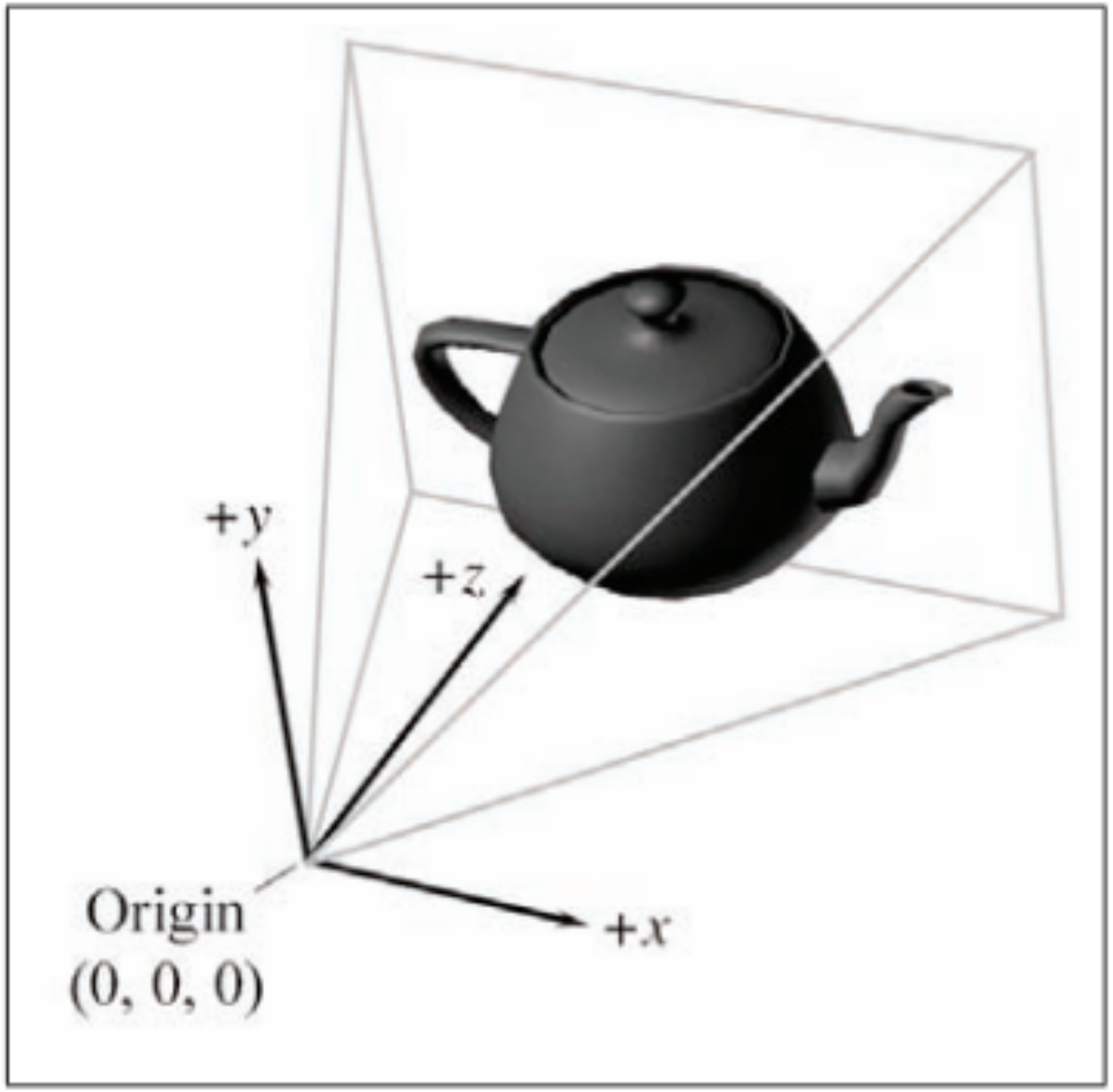
3D Rendering Geometry (continued)

CS 355: Interactive Graphics and Image Processing

Let’s revisit the camera space…

Camera Space



World to Camera

Let's make it easier

- You need to know
  - Position of camera in world coordinates

$$\mathbf{c} = (c_x, c_y, c_z)$$

  - Orientation of camera as given by a set of basic vectors in world coordinates

$$\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$$

Camera's x Camera's y Camera's z

Specifying the Camera

"Look from" point	$\mathbf{p}_{\text{from}}$
"Look at" point	$\mathbf{p}_{\text{at}}$
"Up" vector	$\mathbf{v}_{\text{up}}$

Roughly!

Building Coordinate System

Optical axis (Z) first: $$\mathbf{e}_3 = \frac{\mathbf{p}_{\text{at}} - \mathbf{p}_{\text{from}}}{\|\mathbf{p}_{\text{at}} - \mathbf{p}_{\text{from}}\|}$$

Then side (X): $$\mathbf{e}_1 = \frac{\mathbf{e}_3 \times \mathbf{v}_{\text{up}}}{\|\mathbf{e}_3 \times \mathbf{v}_{\text{up}}\|}$$

Then straighten “up” (Y): $$\mathbf{e}_2 = \frac{\mathbf{e}_1 \times \mathbf{e}_3}{\|\mathbf{e}_1 \times \mathbf{e}_3\|}$$

“Gram - Schmidt” orthogonalization

World to Camera

- Two steps:
  - **Translate**
    everything to be relative to the camera position
  - **Rotate**
    into the camera's viewing orientation

$$\begin{bmatrix} 1 & 0 & 0 & \boxed{\begin{matrix} -c_x \\ -c_y \\ -c_z \end{matrix}} \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} & 0 \\ e_{21} & e_{22} & e_{23} & 0 \\ e_{31} & e_{32} & e_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let’s revisit the pipeline…

Pipeline So Far

Idea: let's cull as much as we can before dividing

World-to-camera transformation

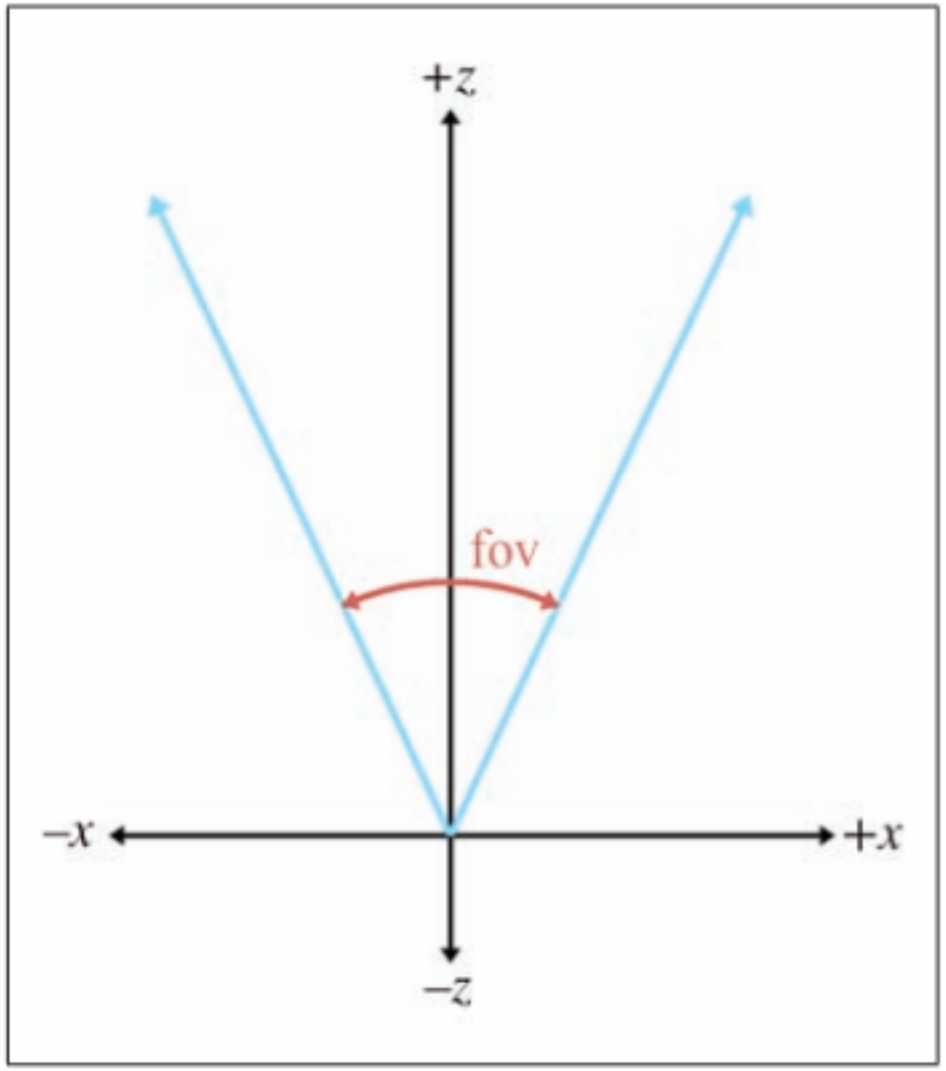
$$\begin{bmatrix} \boxed{\begin{matrix} x \\ y \end{matrix}} \\ f \\ 1 \end{bmatrix} \sim \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ Z_c/f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} & 0 \\ e_{21} & e_{22} & e_{23} & 0 \\ e_{31} & e_{32} & e_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boxed{\begin{matrix} X_w \\ Y_w \\ Z_w \end{matrix}} \\ 1 \end{bmatrix}$$

Normalize Project Rotate Translate

Big problem: lots of time spent on stuff you can't see!

Field of View

- All cameras have a limited field of view
- Field of view depends on the focal length
    - Zoomed in - smaller
    - Zoomed out - larger



Near and Far Planes

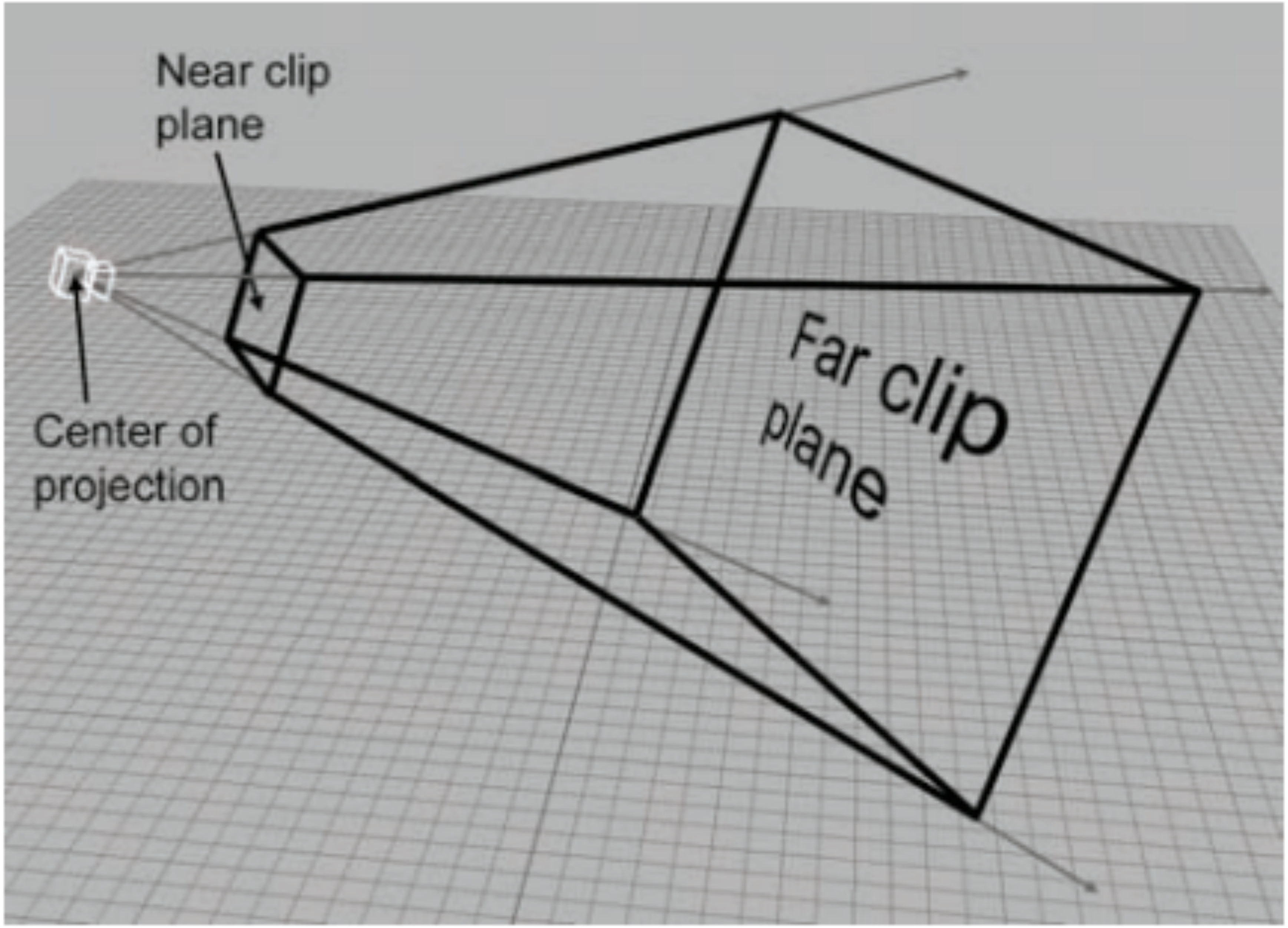
- We don’t want to render things behind us, or perhaps even just barely in front of us

“Near plane”

- We don’t care about things too far away to see well

“Far plane”

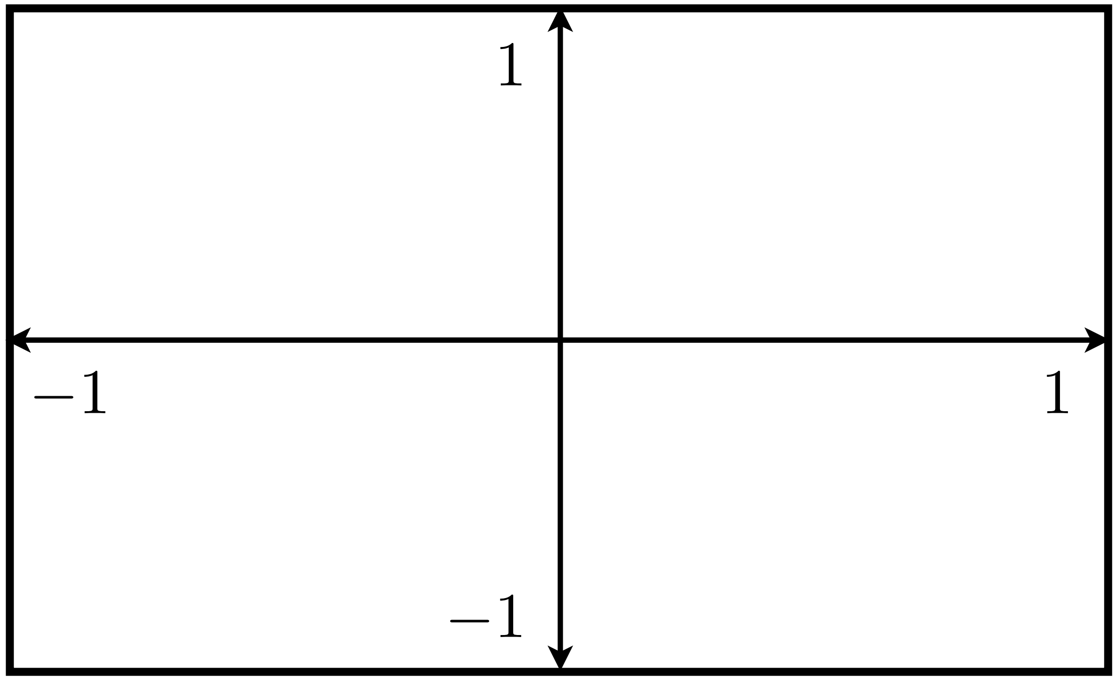
View Frustum



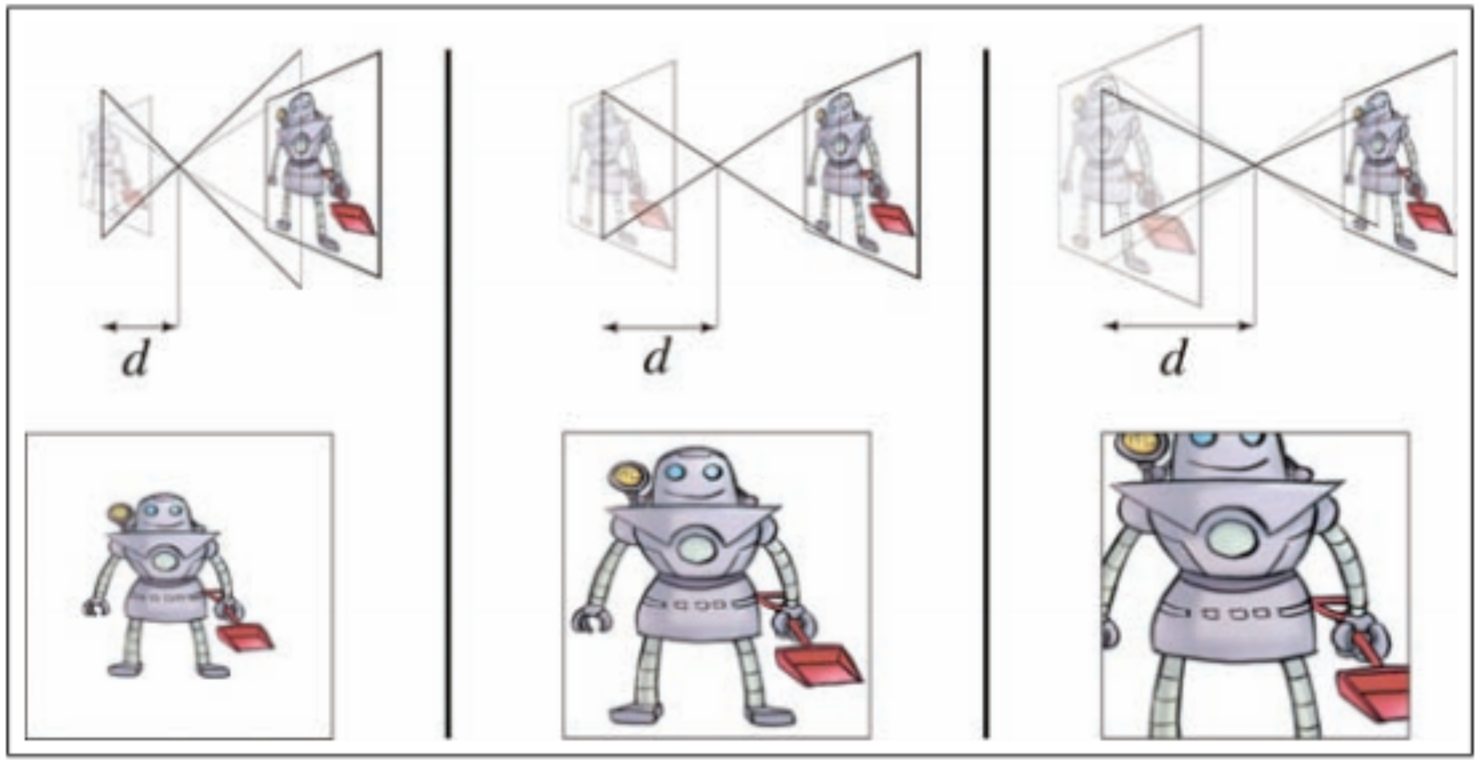
Can we clip things outside the view frustum without doing a divide?

Canonical View

- To simplify, let's assume we map to [-1,1] in both *x* and *y* directions
- Also map [near, far] depth range to [-1,1]
- Maps frustum to $[-1,1]^3$ cube



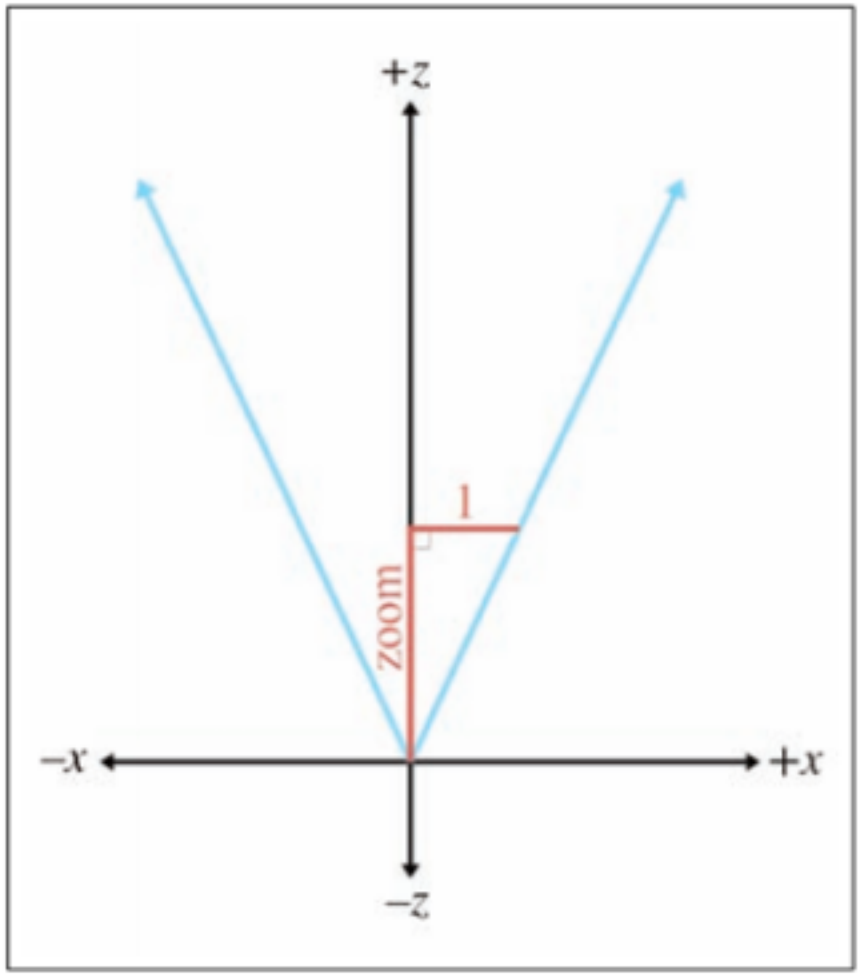
Changing Focal Length



Changing focal length changes overall zoom,
but also affects the shape of the view frustum

Zoom

- Mapping to a canonical window loses
  - Real horizontal width
  - Real vertical width
- We'll need to fold this into our projection matrix
- Think in terms of different "zoom" levels for x and y



$$\text{zoom} = \frac{1}{\tan(\text{fov}/2)}$$

The Clip Matrix

- Let’s build a new projection matrix that
  - Scales visible x to [-1,1]
  - Scales visible y to [-1,1]
  - Scales near to far z to [-1,1]

The Clip Matrix

- Let's build a new projection matrix that
  - Scales visible x to [-1,1]
  - Scales visible y to [-1,1]
  - Scales near to far z to [-1,1]

$$\begin{bmatrix} \mathrm{zoom}_x & 0 & 0 & 0 \\ 0 & \mathrm{zoom}_y & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{-2nf}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The Clip Matrix

$$\begin{bmatrix} \boxed{\begin{matrix} x/w \\ y/w \\ z/w \end{matrix}} \\ 1 \end{bmatrix} \sim \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \mathrm{zoom}_x & 0 & 0 & 0 \\ 0 & \mathrm{zoom}_y & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{-2nf}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

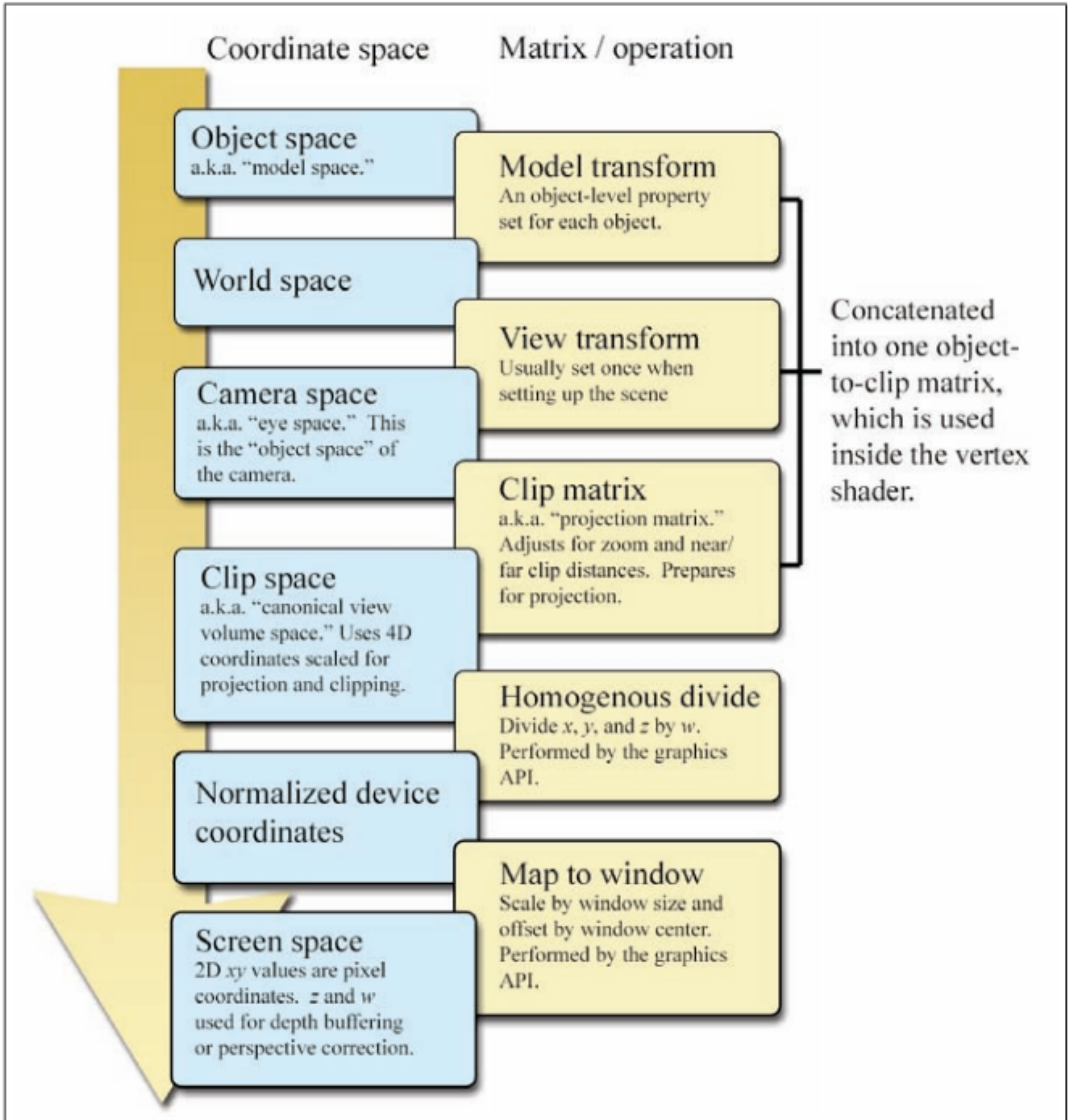
All of these are in the range [-1,1]

Clipping Tests

Left	$x < -w$
Right	$x > w$
Bottom	$y < -w$
Top	$y > w$
Near	$z < -w$
Far	$z > w$

Culling / Clipping

- If an entire primitive (line, polygon) fails the same clipping test, it is outside the field of view—if so, throw out
- If part fails and part passes, clip to the portion in view (create partial primitive) and process from there
- Clipping against multiple planes may not leave anything left — if so, throw out


Coordinate space
Matrix / operation
Object space
a.k.a. "model space."
Model transform
An object-level property set for each object.
World space
View transform
Usually set once when setting up the scene
Camera space
a.k.a. "eye space." This is the "object space" of the camera.
Clip matrix
a.k.a. "projection matrix." Adjusts for zoom and near/far clip distances. Prepares for projection.
Concatenated into one object-to-clip matrix, which is used inside the vertex shader.
Clip space
a.k.a. "canonical view volume space." Uses 4D coordinates scaled for projection and clipping.
Homogenous divide
Divide x, y, and z by w. Performed by the graphics API.
Normalized device coordinates
Map to window
Scale by window size and offset by window center. Performed by the graphics API.
Screen space
2D xy values are pixel coordinates. z and w used for depth buffering or perspective correction.

To Screen Space

- Map [-1,1] x [-1,1] to screen
  - Scale x by half the width
  - Invert y and scale by half the height
  - Translate origin from center to upper left corner

$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ 1 \end{bmatrix} = \begin{bmatrix} \text{width}/2 & 0 & \text{width}/2 \\ 0 & -\text{height}/2 & \text{height}/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$

Rendering Geometry

- ✓ Transform from object to world coordinates
- ✓ Transform from world to camera coordinates
- ✓ Clipping: near plane, far plane, field of view
- ✓ Perspective projection
- ✓ View transformation

Coming up...

- Points, lines, and polygons
- Visibility testing