3D Transforms and Graphics Pipeline

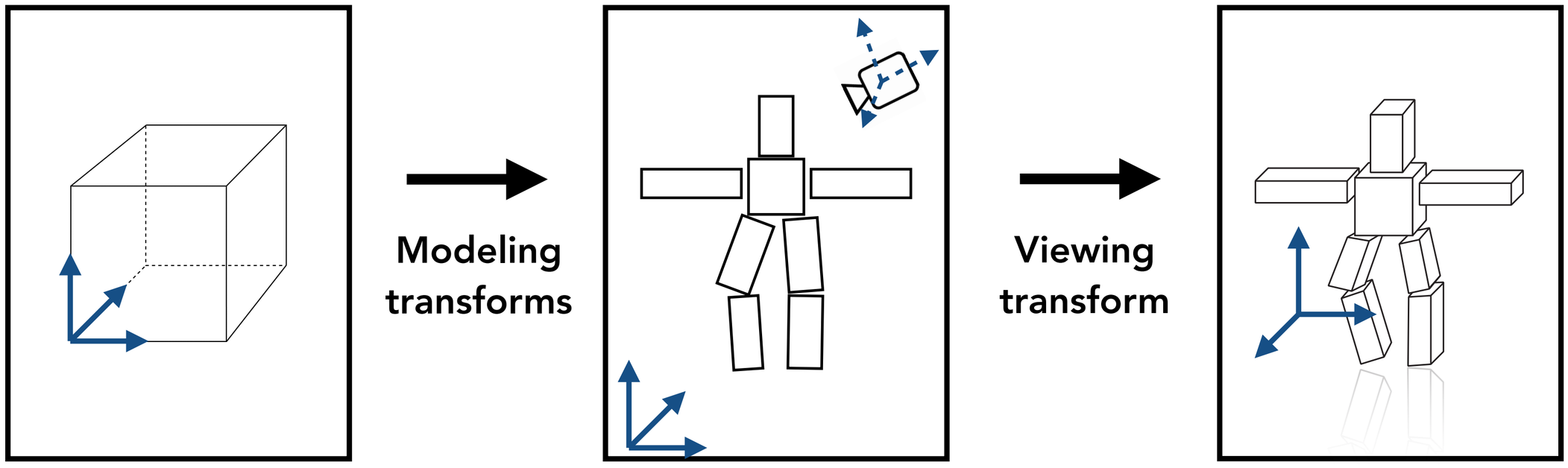
Computer Graphics and Imaging
UC Berkeley CS184
Summer 2020

Announcements

- First Discord project party was last night
- Please stop by if you need help!
  - Wednesday 3-5pm
  - Friday 3-5pm
- Can get direct help from staff using “queue” or chat with students in the text channels

3D Viewing Transforms

Full transform “stack”



Which transform do I modify to...

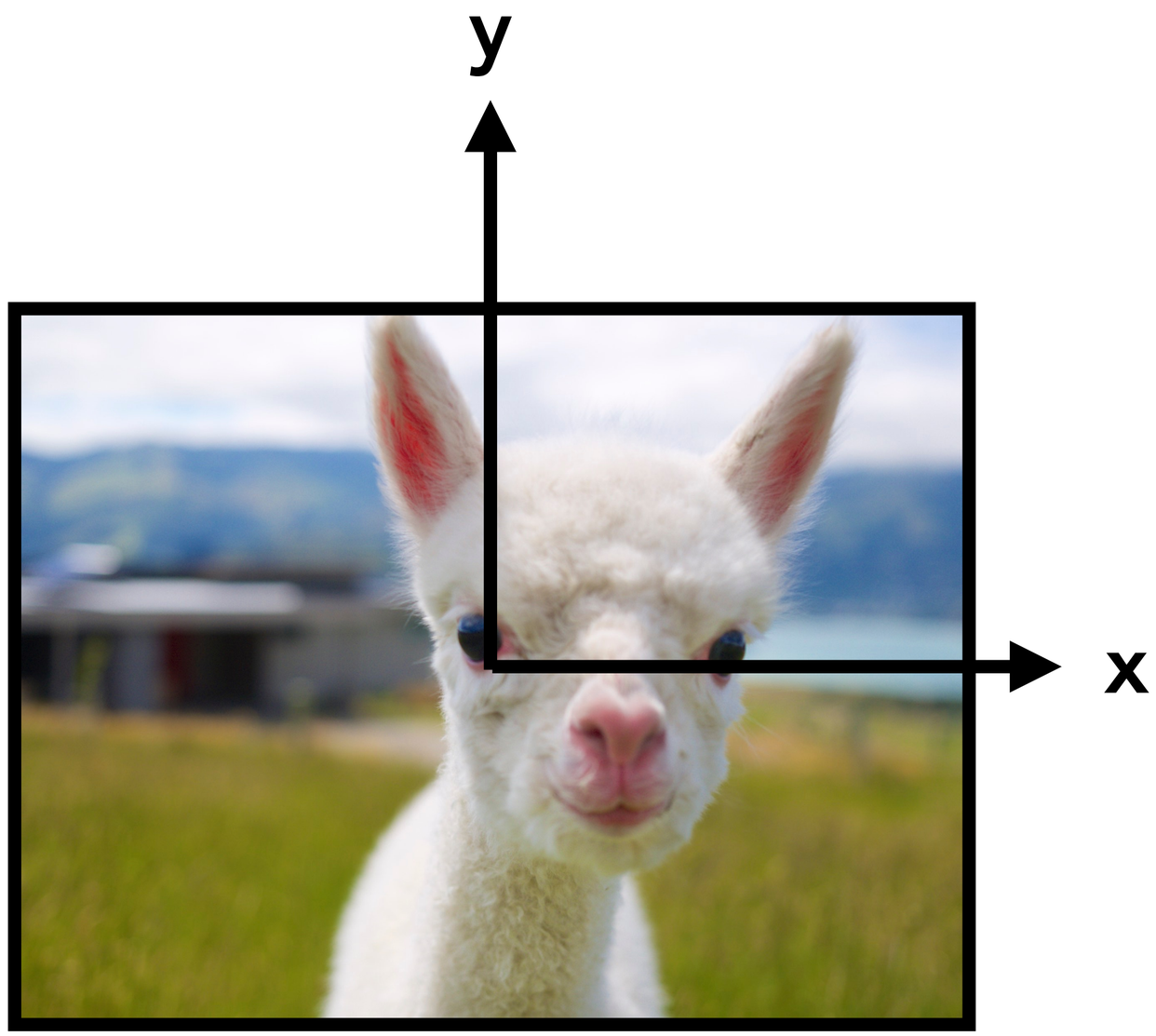
- Move camera closer to object?
- Change output rendering resolution?
- Move robot relative to other objects in scene?
- Change camera's field of view?

"Standard" Camera Space

We will use this convention for "standard" camera coordinates:

- camera located at the origin
- looking down *negative* z-axis
- vertical vector is y-axis
- (x-axis) orthogonal to y & z

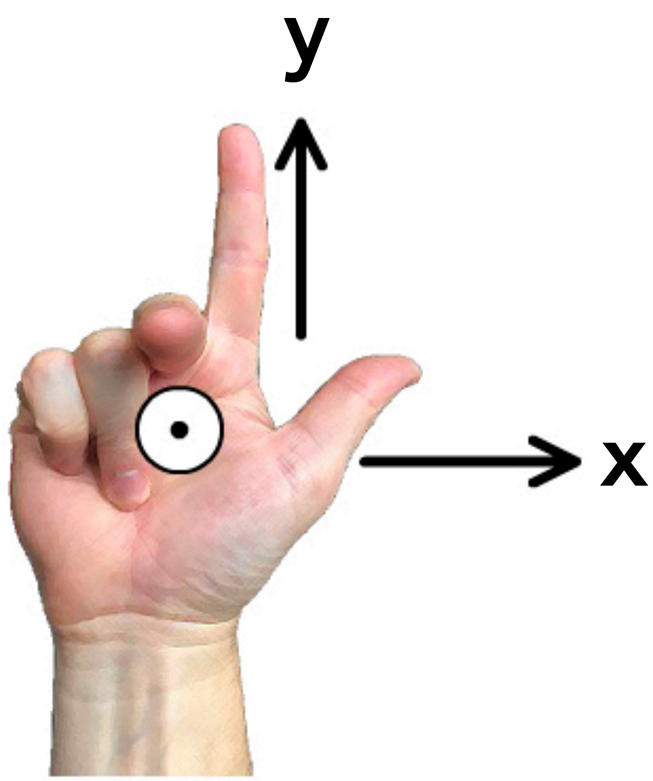
"Standard" Camera Coordinates

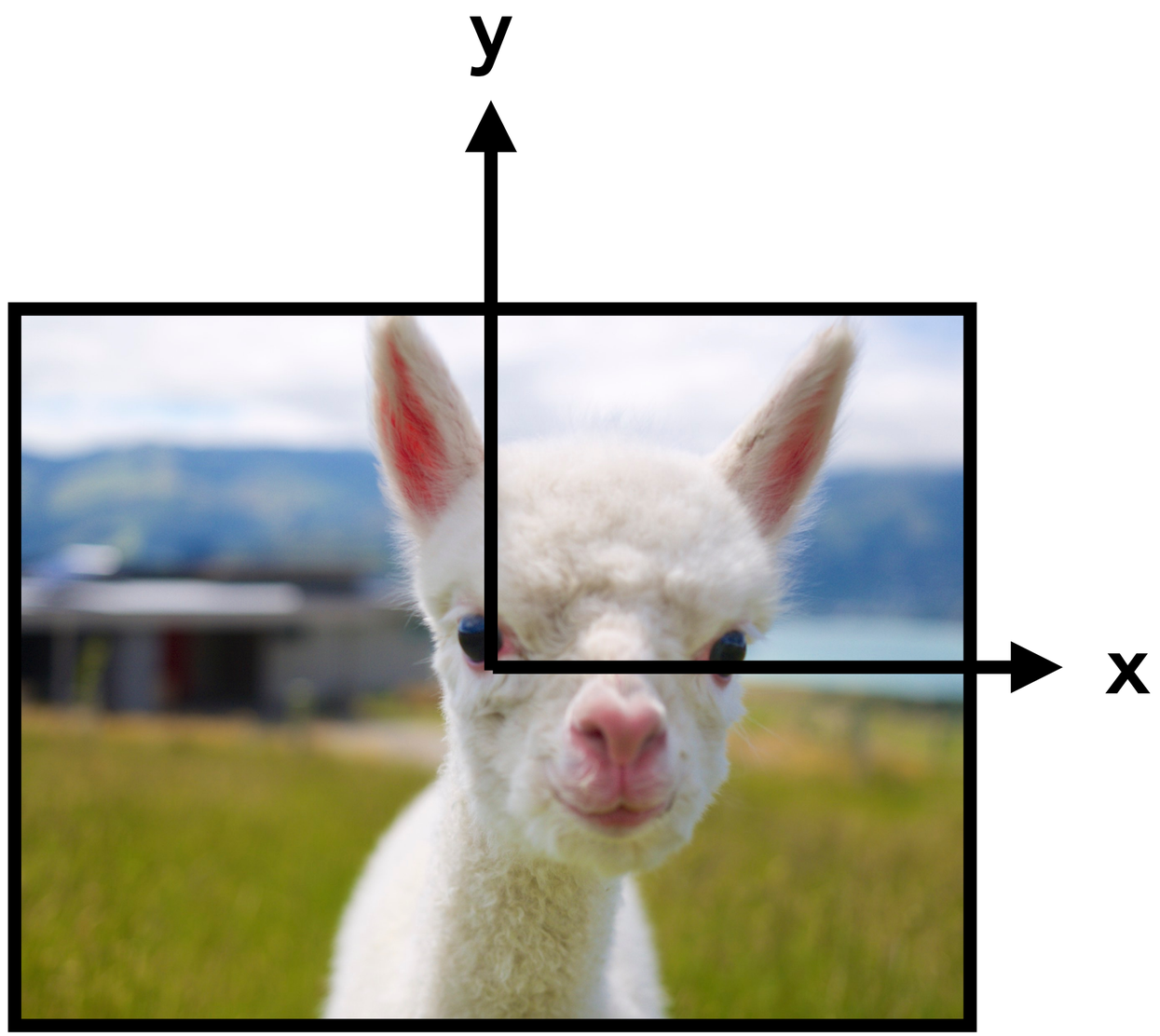


Resulting image
(z-axis pointing away from scene)

"Standard" Camera Coordinates

Right Hand Rule



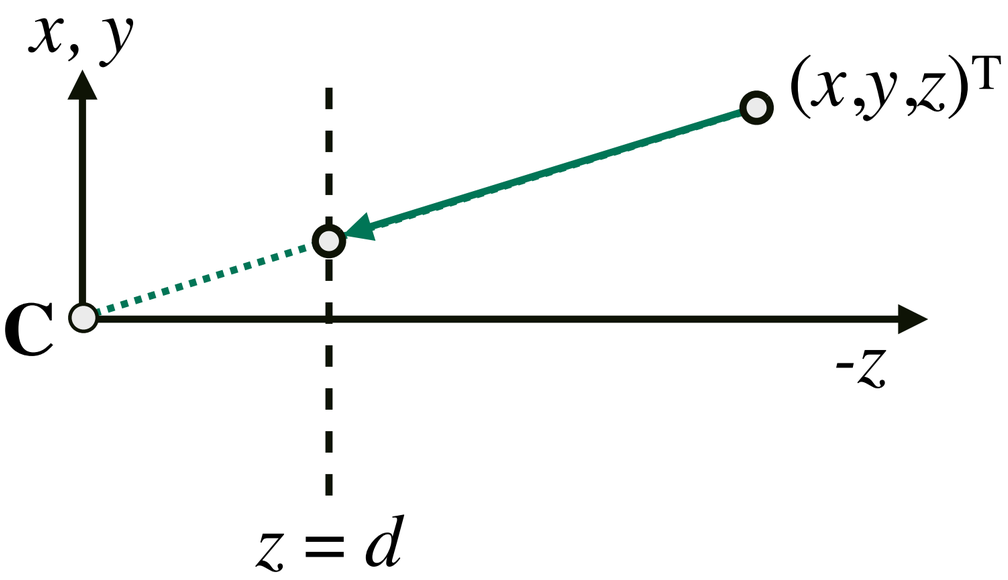


Resulting image
(z-axis pointing away from scene)

Projective Transforms

Standard perspective projection

- Center of projection: $(0, 0, 0)^{\mathrm{T}}$
- Image plane at $z = d$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \cdot d/z \\ y \cdot d/z \\ d \end{pmatrix}$$

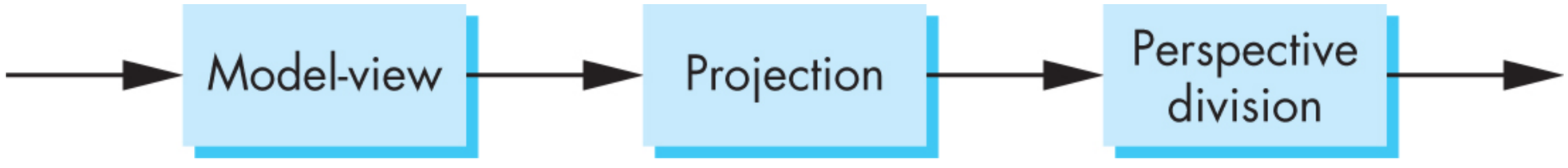
Homogenous Coordinates (3D)

$$\mathbf{p} = \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} \longleftrightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

$$\mathbf{q} = \mathbf{M} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \longleftrightarrow \begin{pmatrix} xd/z \\ yd/z \\ d \\ 1 \end{pmatrix}$$

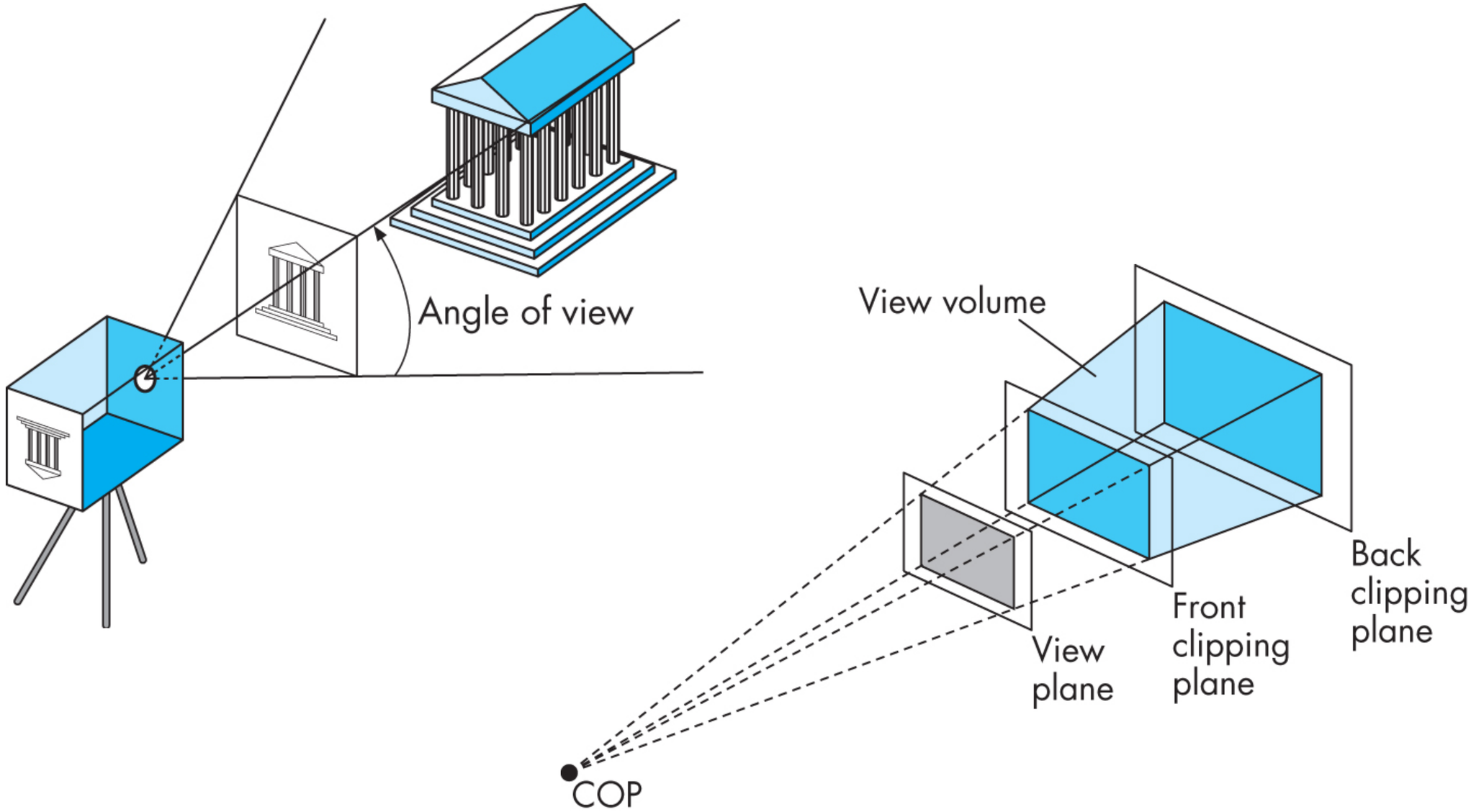
Note non-zero term in final row.
First time we have seen this.



Perspective Transform Matrix

$$\mathbf{P} = \begin{bmatrix} \frac{near}{right} & 0 & 0 & 0 \\ 0 & \frac{near}{top} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{-2far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Specifying Perspective Projection

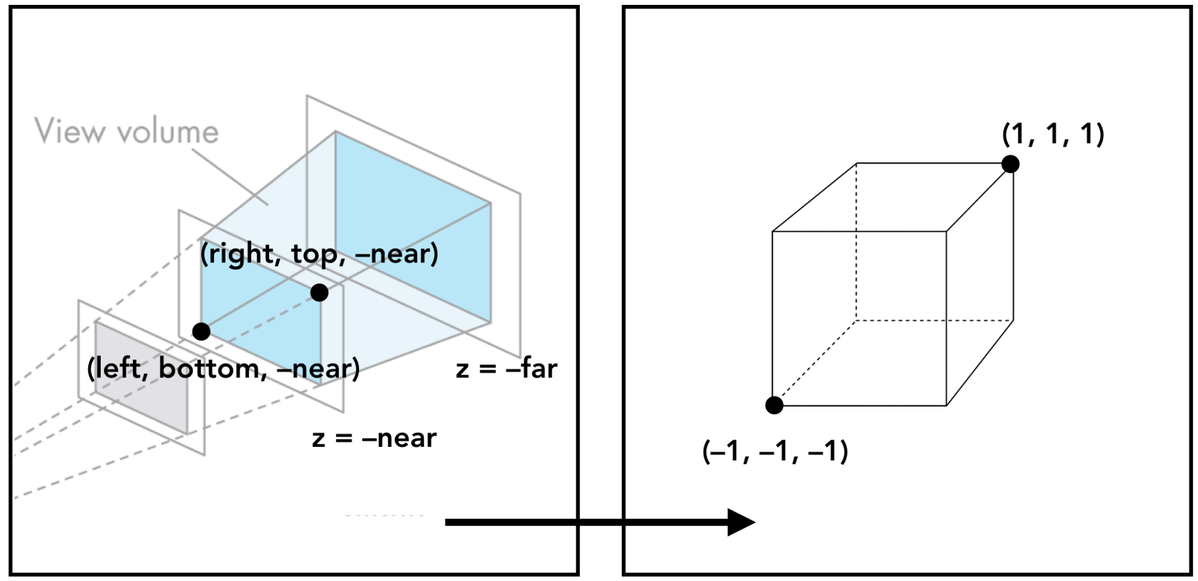


From Angel and Shreiner, Interactive Computer Graphics

Perspective Projection Transform

Notes:

- Need not be symmetric about z-axis, but for simplicity here we assume so
- This transform will preserve depth information (ordering) in NDC

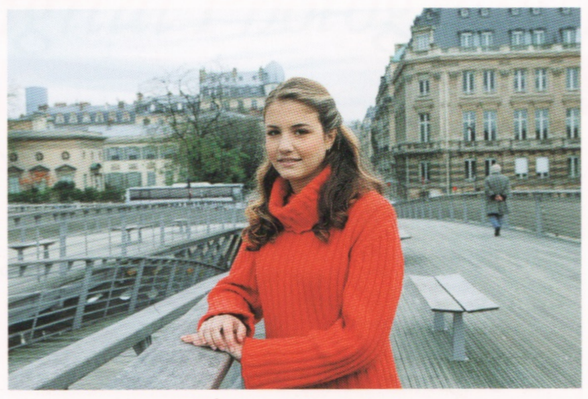


Which picture has the largest field of view?

16mm

24mm

50mm



200mm

135mm

And which picture has photographer standing farthest away?

From Canon EF Lens Work III

Graphics Pipeline

What is the ordering of these operations?

A. Z-buffer visibility test

B. Evaluate shading function

C. Apply perspective transform

D. Rasterization (point-in-triangle test)

What is the ordering of these operations?

A. Z-buffer visibility test

B. Evaluate shading function

C. Apply perspective transform

D. Rasterization (point-in-triangle test)

Answer: CDBA for pixel shading, or BCDA for vertex shading

Caveat: modern GPUs also allow for an “early depth test” mode that runs a z-buffer test *before* the fragment shader…

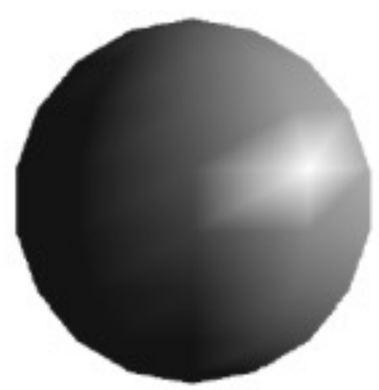
Shading Frequency: Triangle, Vertex or Pixel

Shade each triangle (flat shading)

- Triangle face is flat — one normal vector
- Not good for smooth surfaces

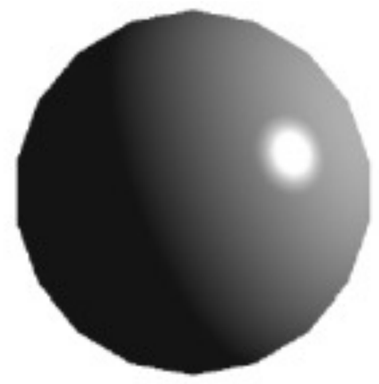
Shade each vertex (“Gouraud” shading)

- Interpolate colors from vertices across triangle
- Each vertex has a normal vector

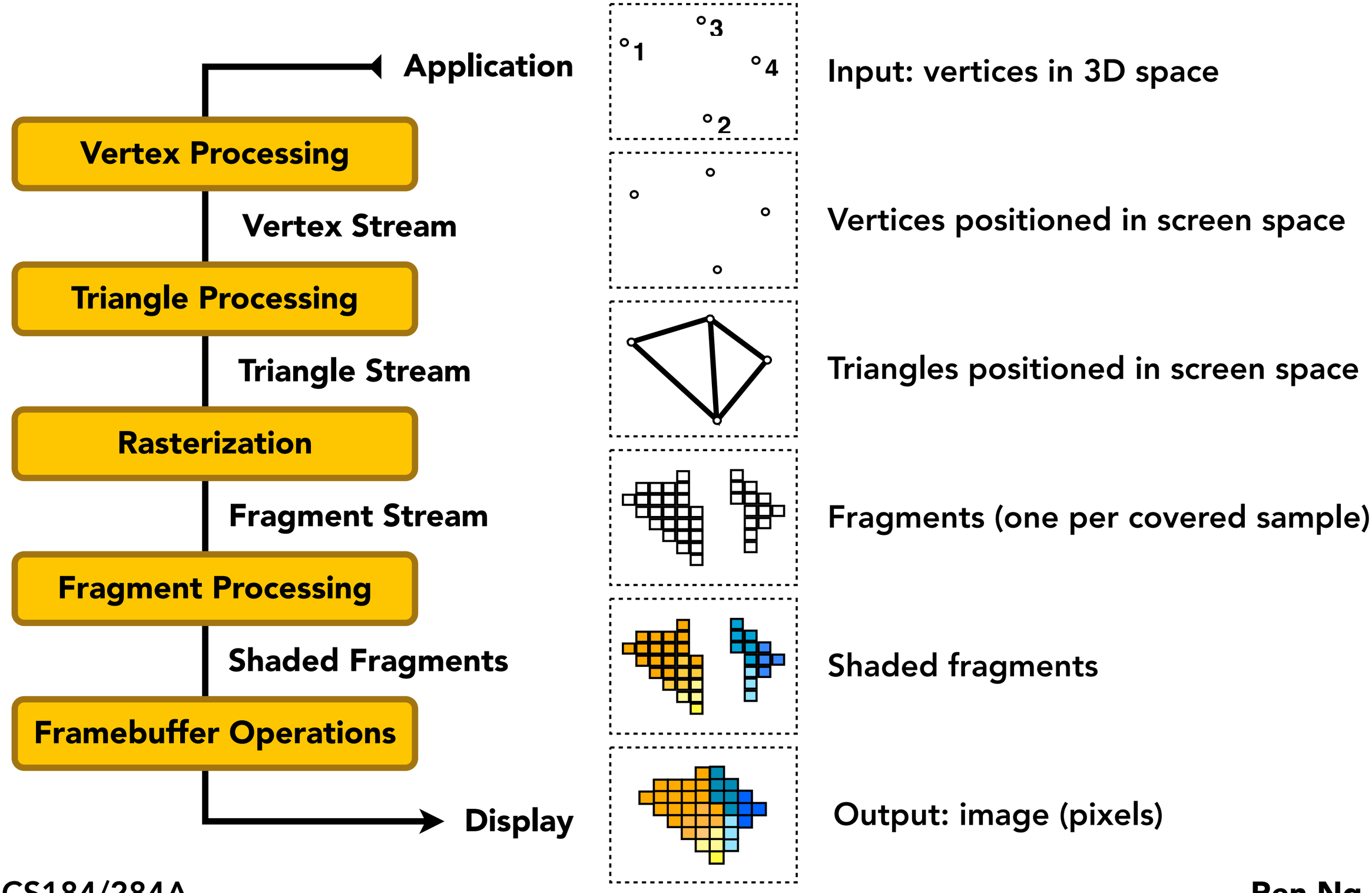


Shade each pixel (“Phong” shading)

- Interpolate normal vectors across each triangle
- Compute full shading model at each pixel



Rasterization Pipeline



Demo time