

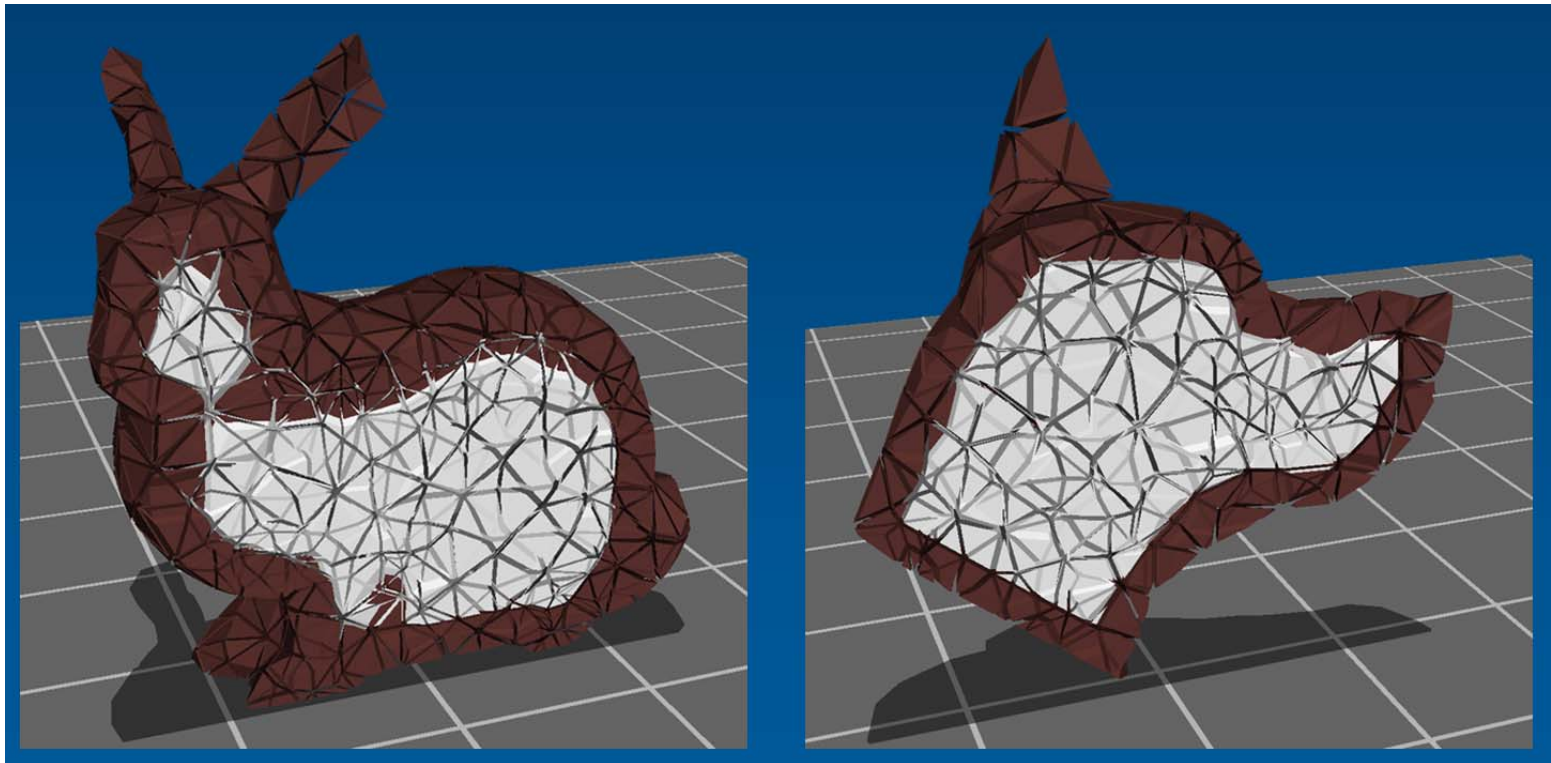
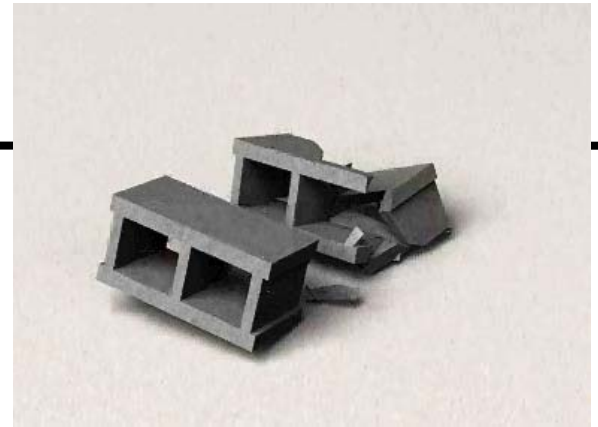
The Graphics Pipeline: Projective Transformations

Today

- Review & Schedule
- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations

Last Week:

- Animation & Quaternions
- Finite Element Simulations
 - collisions, fracture, & deformation



Schedule

- Final Project
 - Post your ideas on the web page
 - Meet with staff to talk about project ideas
 - *sign up for an appointment on Friday*
 - Proposal due on Monday October 27th
- Friday October 24th: Assignment 5 due
- Office Hours this week:
 - Tuesday after class (Rob – student center)
 - Wednesday 7-9 (Patrick – student center)
 - Thursday after class (Fredo – student center)
 - Friday 3-5, student center (Barb – student center)

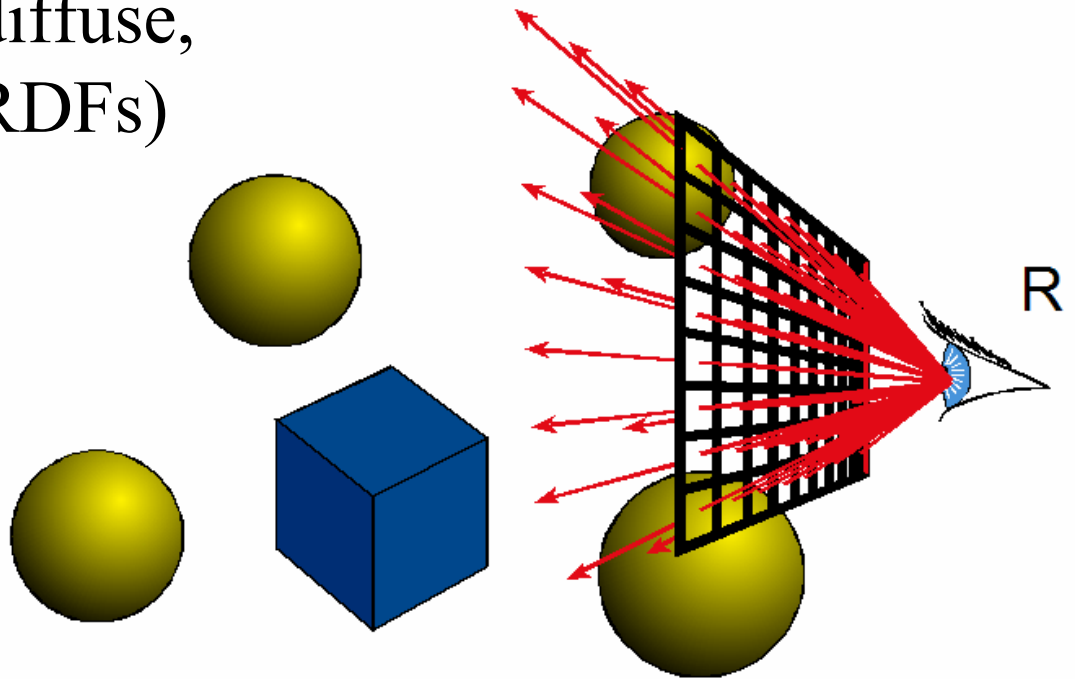
Questions?

Today

- Review & Schedule
- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations

What have we done so far?

- Ray Casting / Tracing
 - ray/primitive intersections
 - transformations
 - local shading (diffuse, ambient, $\rightarrow$ BRDFs)
 - global effects (shadows, transparency, caustics, ...)



Ray Casting / Tracing

for every pixel, construct a ray from the eye

for every object in the scene *Grid Acceleration*

intersect ray with object

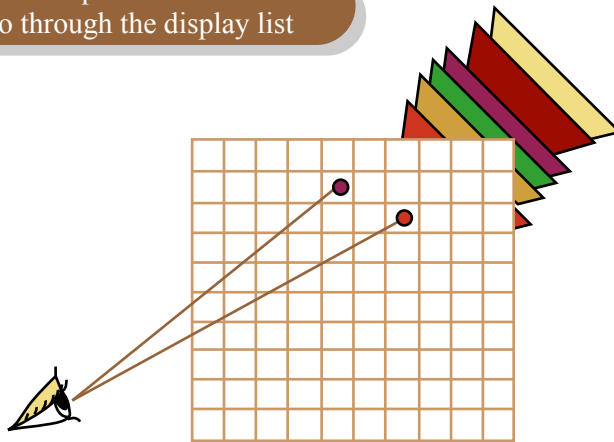
find closest intersection with the ray

compute normal at point of intersection

compute color for pixel (shoot secondary rays)

"Inverse-Mapping" Approach

For each pixel on the screen
go through the display list

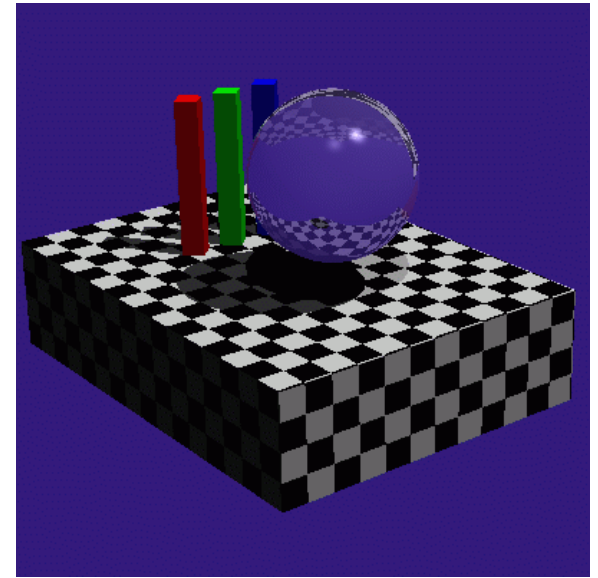
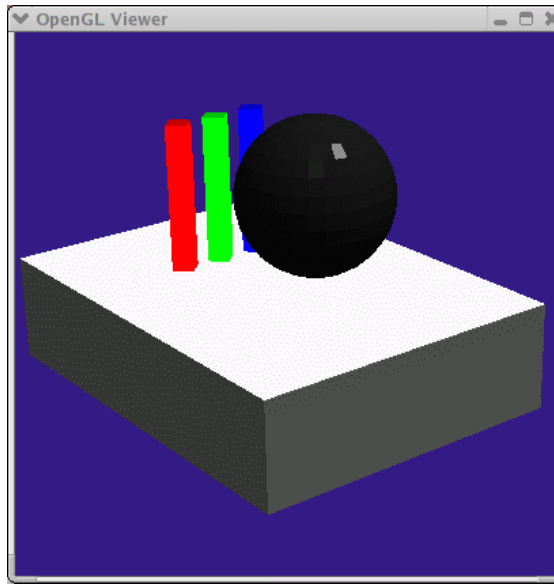
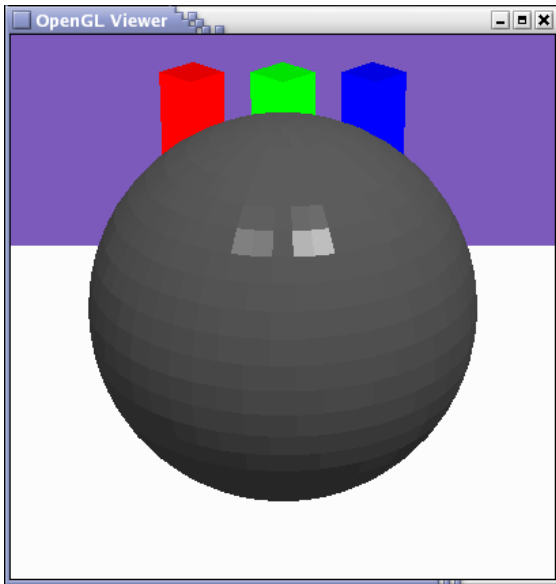


Ray Casting / Tracing

- Advantages?
 - Smooth variation of normal, silhouettes
 - Generality: can render anything that can be intersected with a ray
 - Atomic operation, allows recursion
- Disadvantages?
 - Time complexity (N objects, R pixels)
 - Usually too slow for interactive applications
 - Hard to implement in hardware (lacks computation coherence, must fit entire scene in memory)

Can we render things interactively?

- Of course! games, 3D modeling packages, architectural walkthroughs, assignment 5, etc.



How do we render interactively?

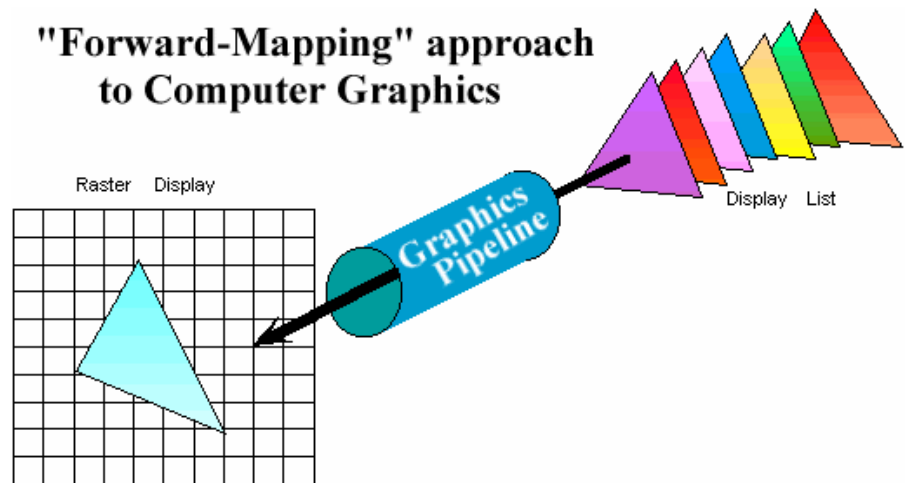
- Use the graphics hardware (the graphics pipeline), via OpenGL, MesaGL, or DirectX
- Most global effects available in ray tracing will be sacrificed, but some can be approximated.

Image removed due to copyright considerations

Scan Conversion – Graphics Pipeline

- Primitives are processed one at a time
- Early stages involve analytic processing
- Sampling occurs late in the pipeline
- Minimal state required

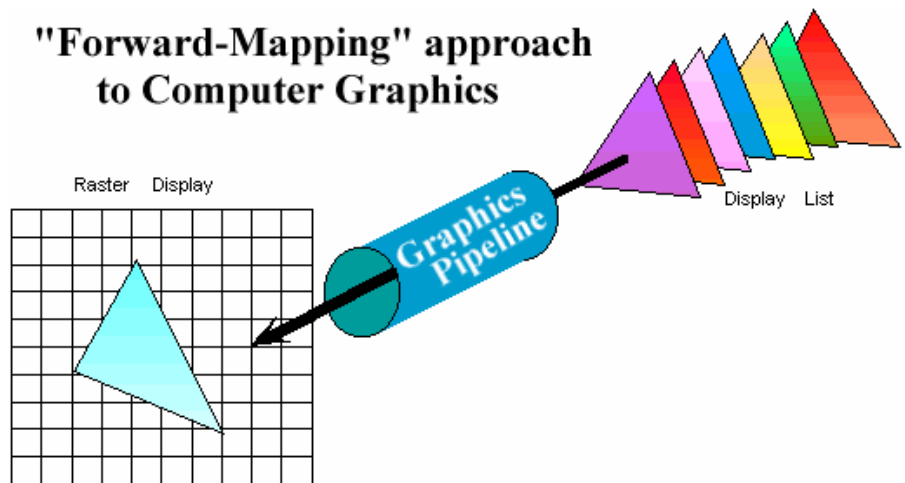
```
glBegin(GL_TRIANGLES)  
glNormal3f(...)  
glVertex3f(...)  
glVertex3f(...)  
glVertex3f(...)  
glEnd();
```



Scan Conversion – Graphics Pipeline

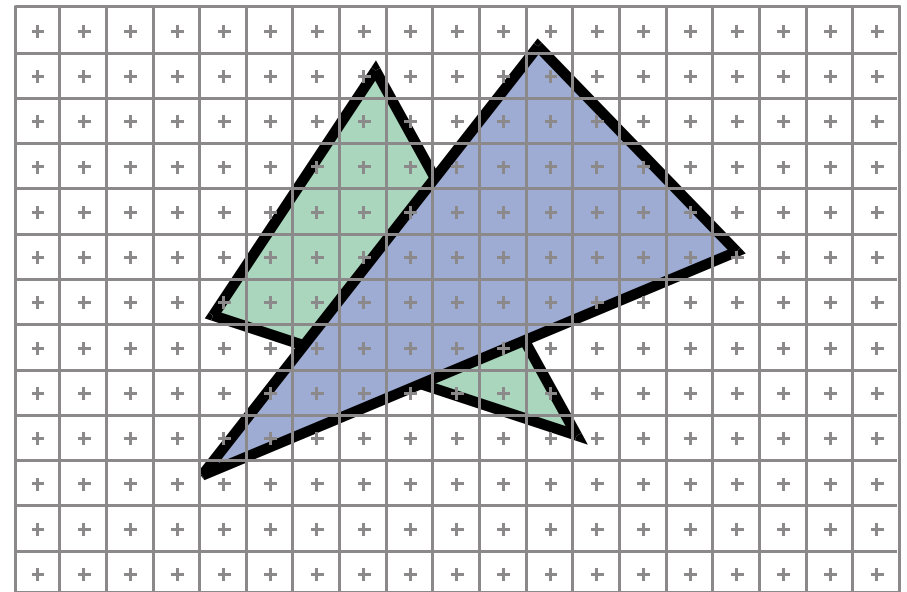
for every object in the scene
 shade the vertices
 scan convert the object to the framebuffer
 interpolate the color computed for each vertex
 remember the closest value per pixel

```
glBegin(GL_TRIANGLES)  
glNormal3f(...)  
glVertex3f(...)  
glVertex3f(...)  
glVertex3f(...)  
glEnd();
```



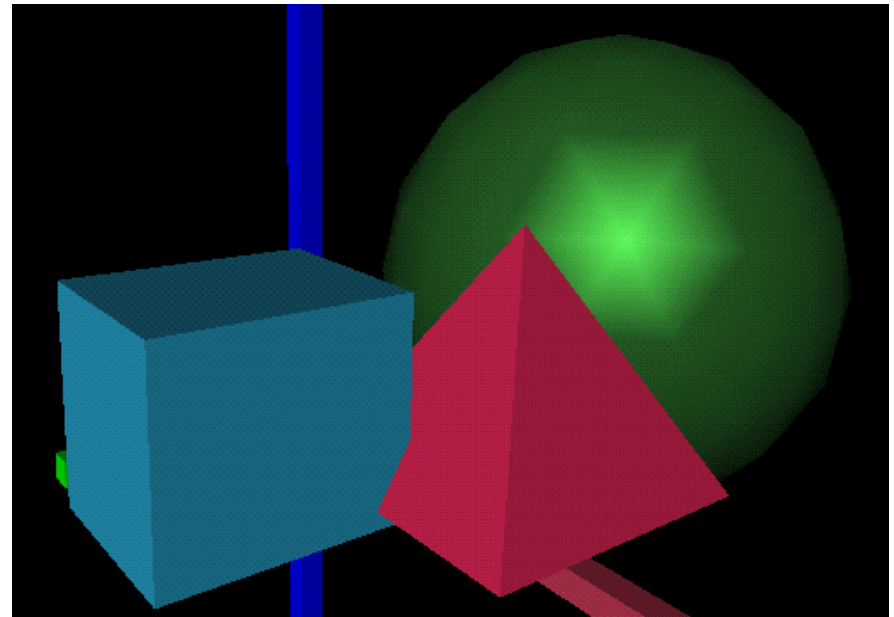
Scan Conversion

- Given the primitive's vertices & the illumination at each vertex:
- Figure out which pixels to "turn on" to render the primitive
- Interpolate the illumination values to "fill in" the primitive



Limitations of Scan Conversion

- Restricted to scan-convertible primitives
 - Object polygonization
- Faceting, shading artifacts
- Effective resolution is hardware dependent
- No handling of shadows, reflection, transparency
- Problem of overdraw (high depth complexity)
- What if there are more triangles than pixels?

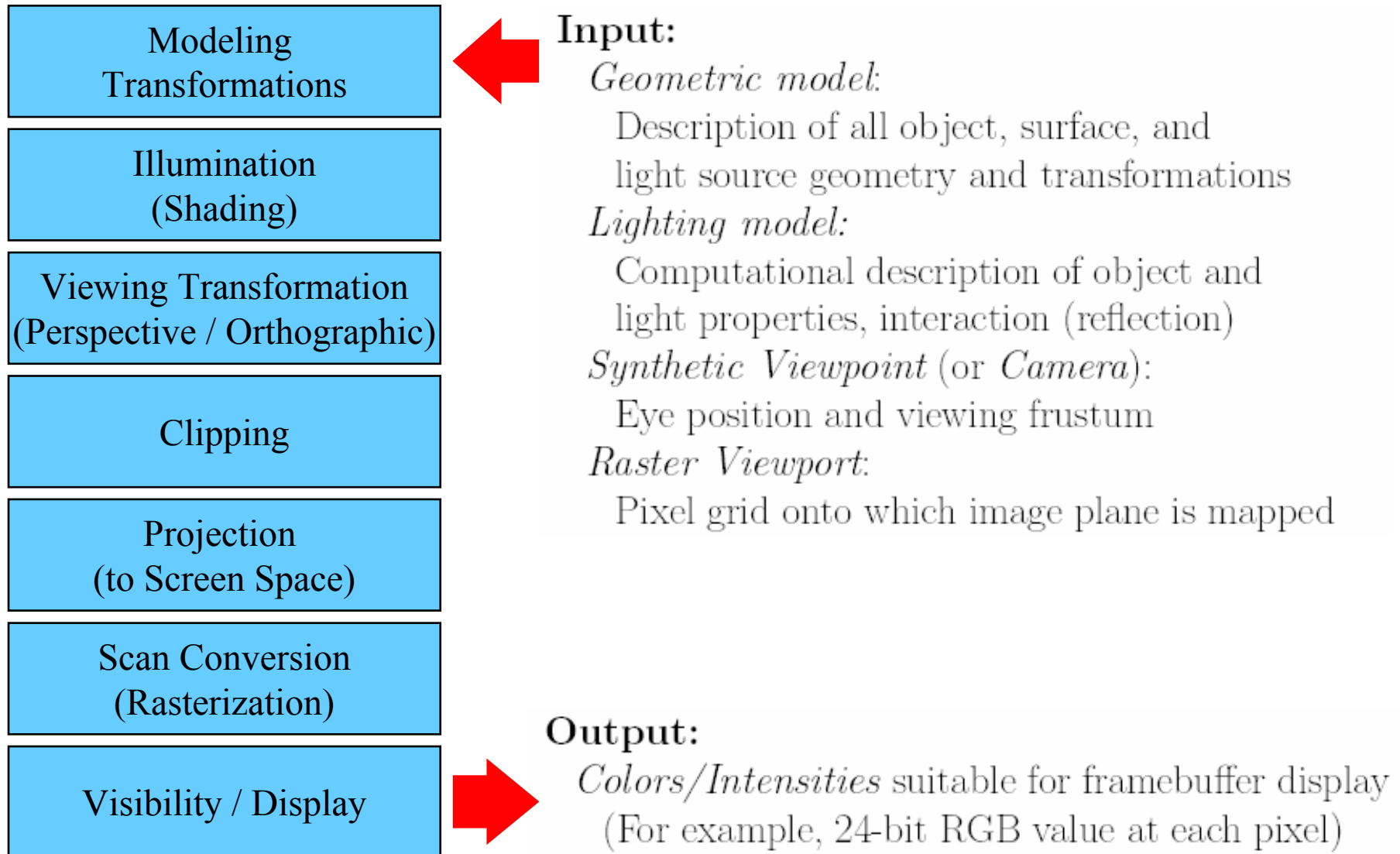


Questions?

Today

- Review & Schedule
- Ray Casting / Tracing vs. Scan Conversion
- **The Graphics Pipeline**
- Projective Transformations

The Graphics Pipeline



The Graphics Pipeline

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

Clipping

Projection
(to Screen Space)

Scan Conversion
(Rasterization)

Visibility / Display

- Primitives are processed in a series of stages
- Each stage forwards its result on to the next stage
- The pipeline can be drawn and implemented in different ways
- Some stages may be in hardware, others in software
- Optimizations & additional programmability are available at some stages

Modeling Transformations

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

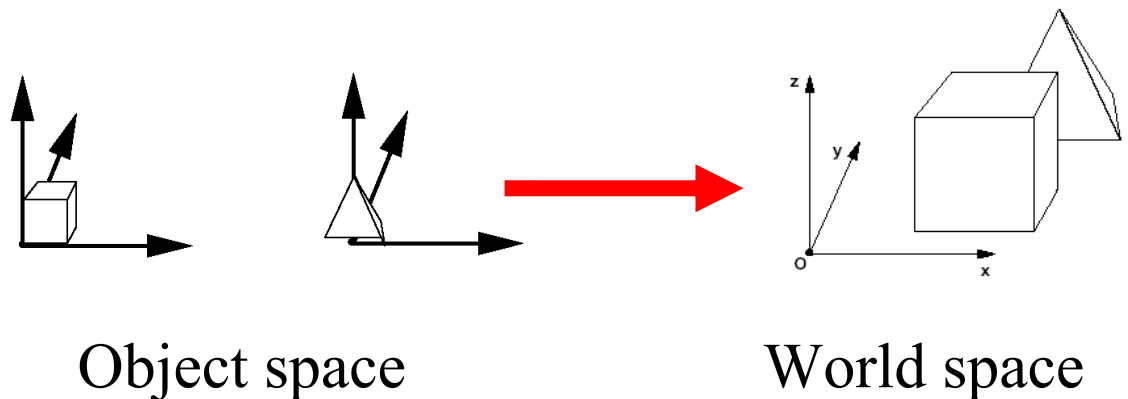
Clipping

Projection
(to Screen Space)

Scan Conversion
(Rasterization)

Visibility / Display

- 3D models defined in their own coordinate system (object space)
- Modeling transforms orient the models within a common coordinate frame (world space)



Illumination (Shading) (Lighting)

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

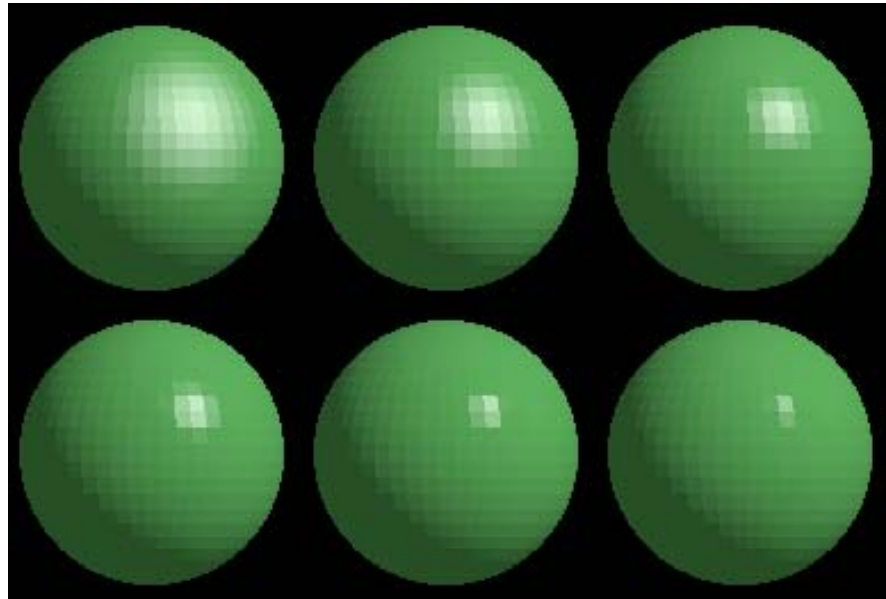
Clipping

Projection
(to Screen Space)

Scan Conversion
(Rasterization)

Visibility / Display

- Vertices lit (shaded) according to material properties, surface properties (normal) and light sources
- Local lighting model (Diffuse, Ambient, Phong, etc.)



Viewing Transformation

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

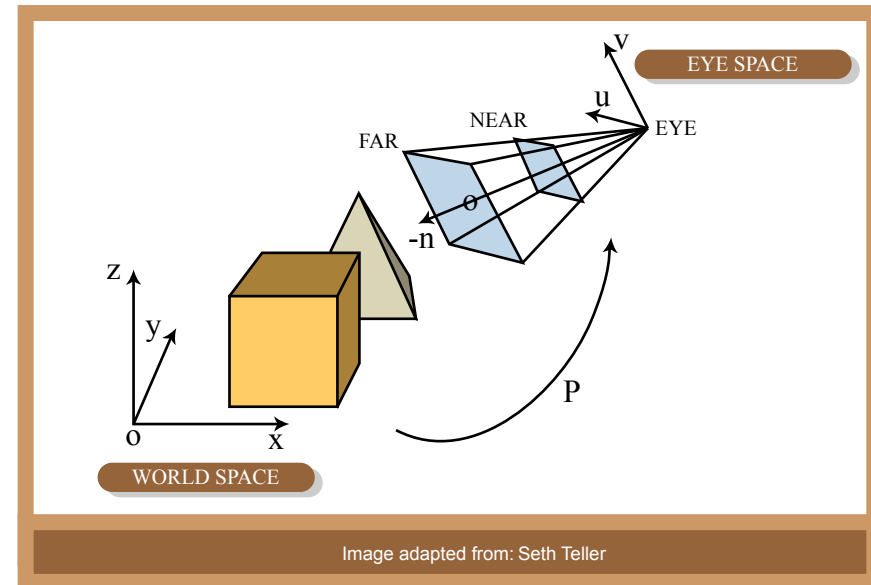
Clipping

Projection
(to Screen Space)

Scan Conversion
(Rasterization)

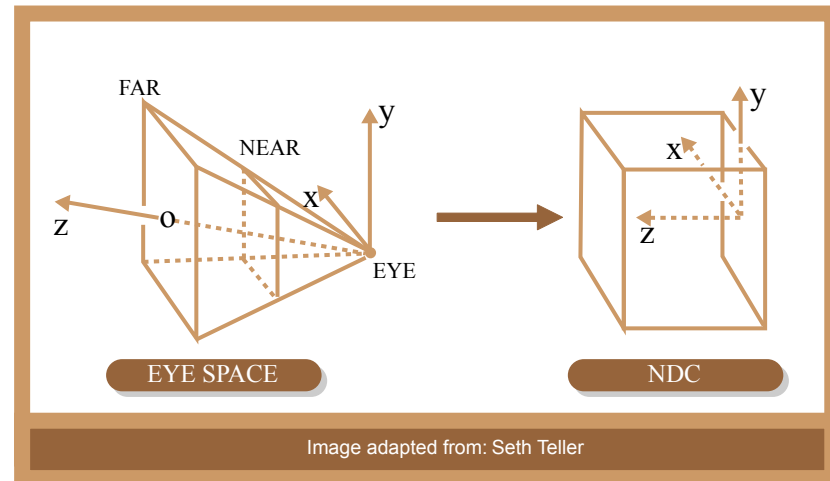
Visibility / Display

- Maps world space to eye space
- Viewing position is transformed to origin & direction is oriented along some axis (usually z)

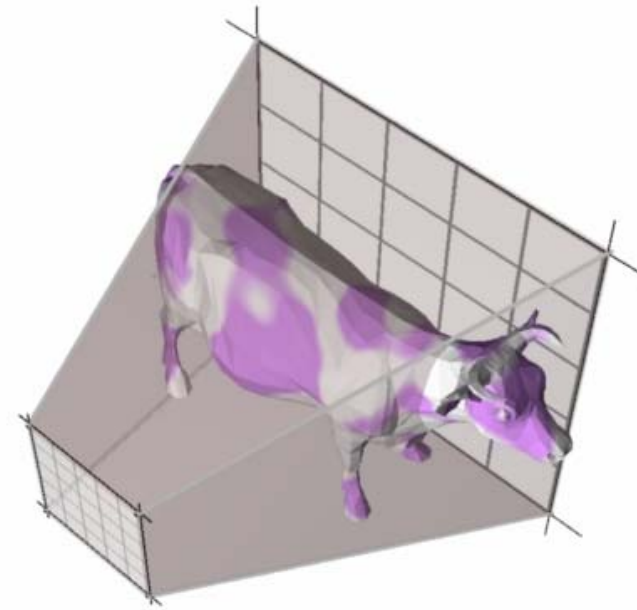


Clipping

- Transform to Normalized Device Coordinates (NDC)



- Portions of the object outside the view volume (view frustum) are removed



Projection

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

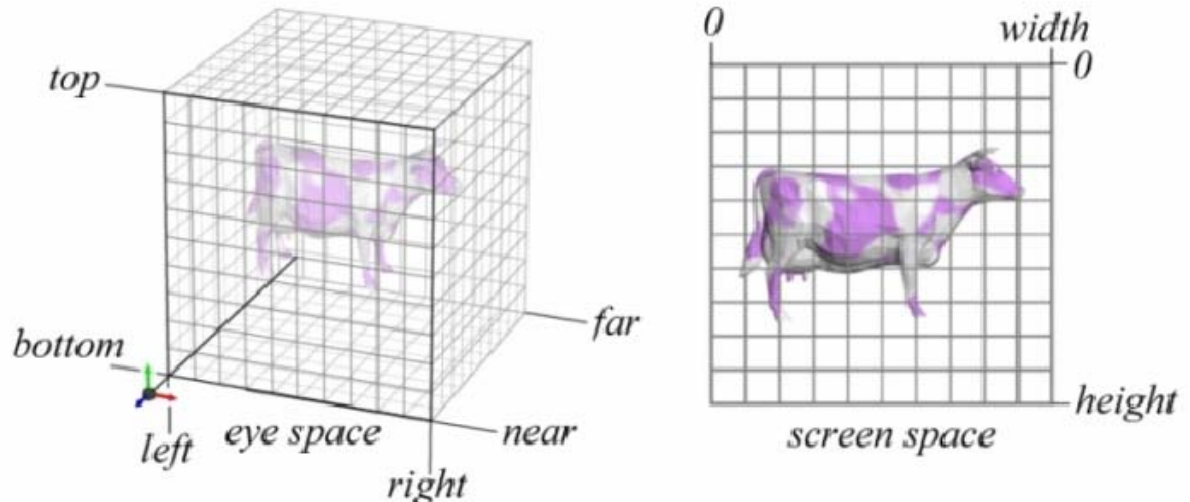
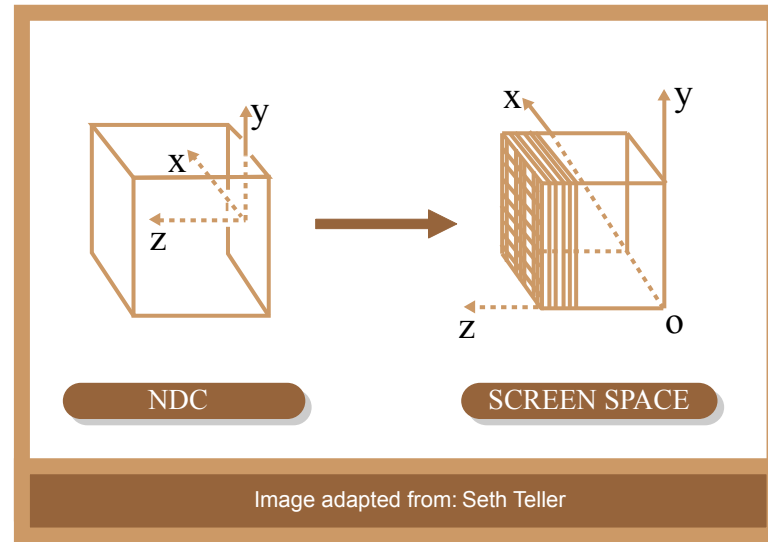
Clipping

**Projection
(to Screen Space)**

Scan Conversion
(Rasterization)

Visibility / Display

- The objects are projected to the 2D image plane (screen space)



Scan Conversion (Rasterization)

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

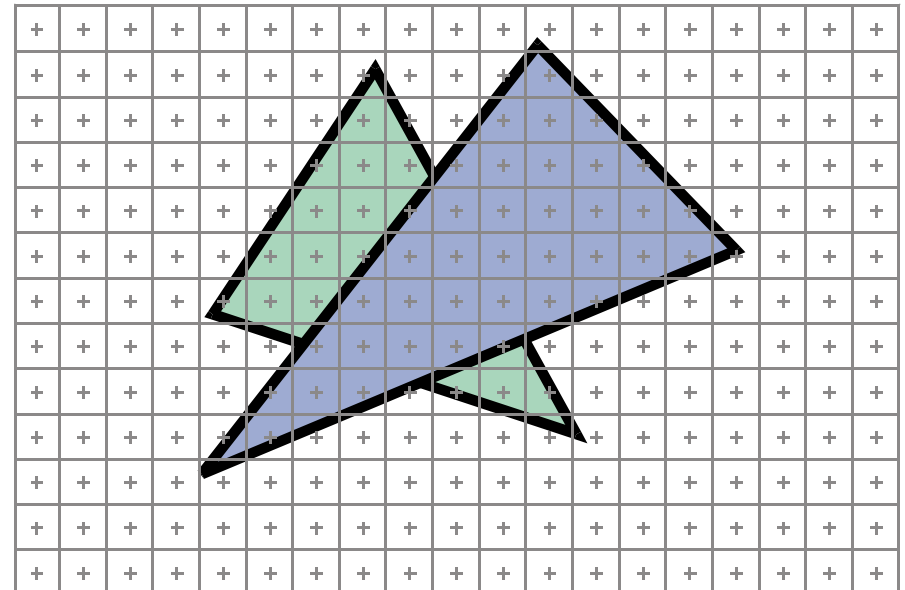
Clipping

Projection
(to Screen Space)

Scan Conversion
(Rasterization)

Visibility / Display

- Rasterizes objects into pixels
- Interpolate values as we go (color, depth, etc.)



Visibility / Display

Modeling
Transformations

Illumination
(Shading)

Viewing Transformation
(Perspective / Orthographic)

Clipping

Projection
(to Screen Space)

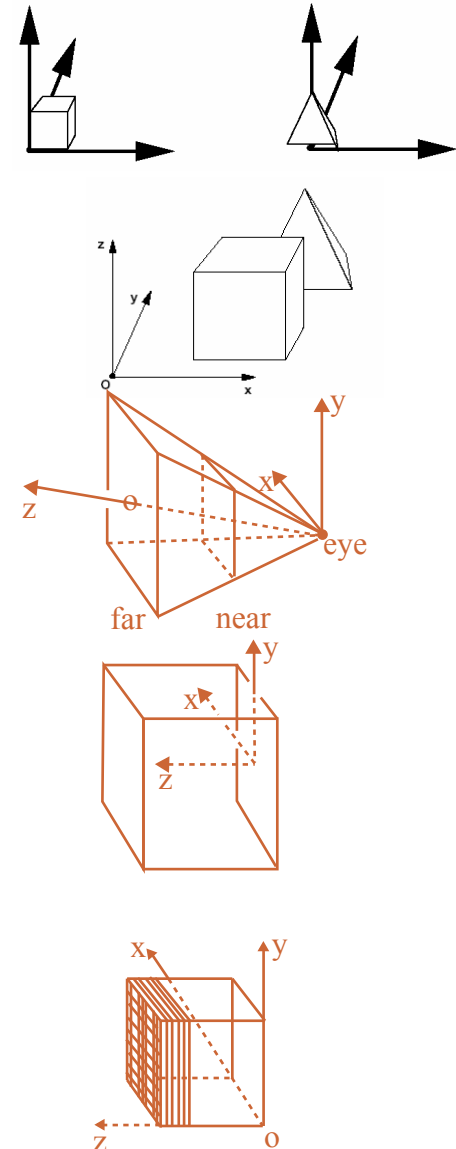
Scan Conversion
(Rasterization)

Visibility / Display

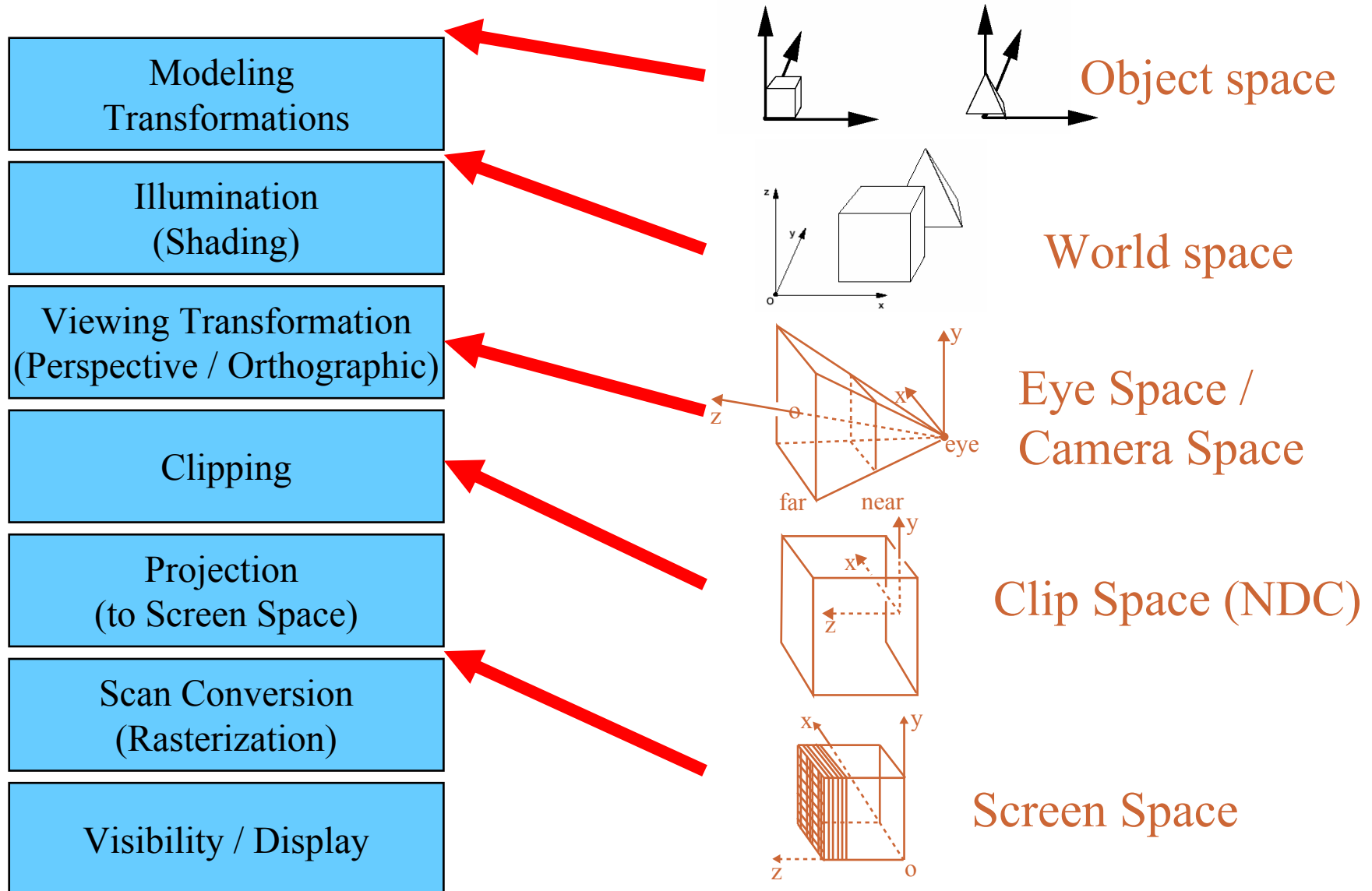
- Each pixel remembers the closest object (depth buffer)
- Almost every step in the graphics pipeline involves a change of coordinate system. Transformations are central to understanding 3D computer graphics.

Common Coordinate Systems

- Object space
 - local to each object
- World space
 - common to all objects
- Eye space / Camera space
 - derived from view frustum
- Clip space / Normalized Device Coordinates (NDC)
 - $[-1,-1,-1] \rightarrow [1,1,1]$
- Screen space
 - indexed according to hardware attributes



Coordinate Systems in the Pipeline

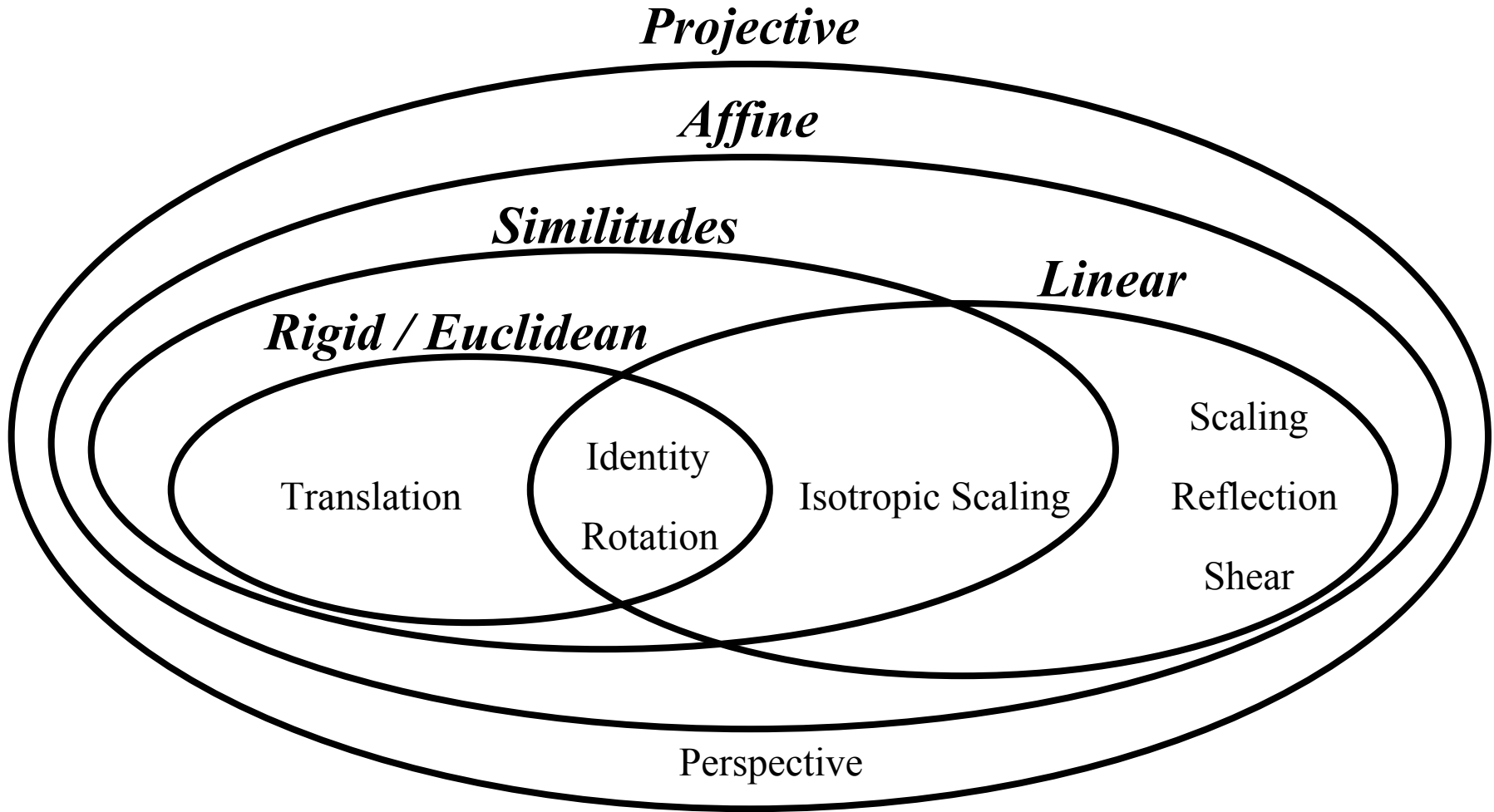


Questions?

Today

- Review & Schedule
- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations
 - Transformations & Homogeneous Coordinates
 - Orthographic & Perspective Projections
 - Canonical View Volume

Remember Transformations?



Homogeneous Coordinates

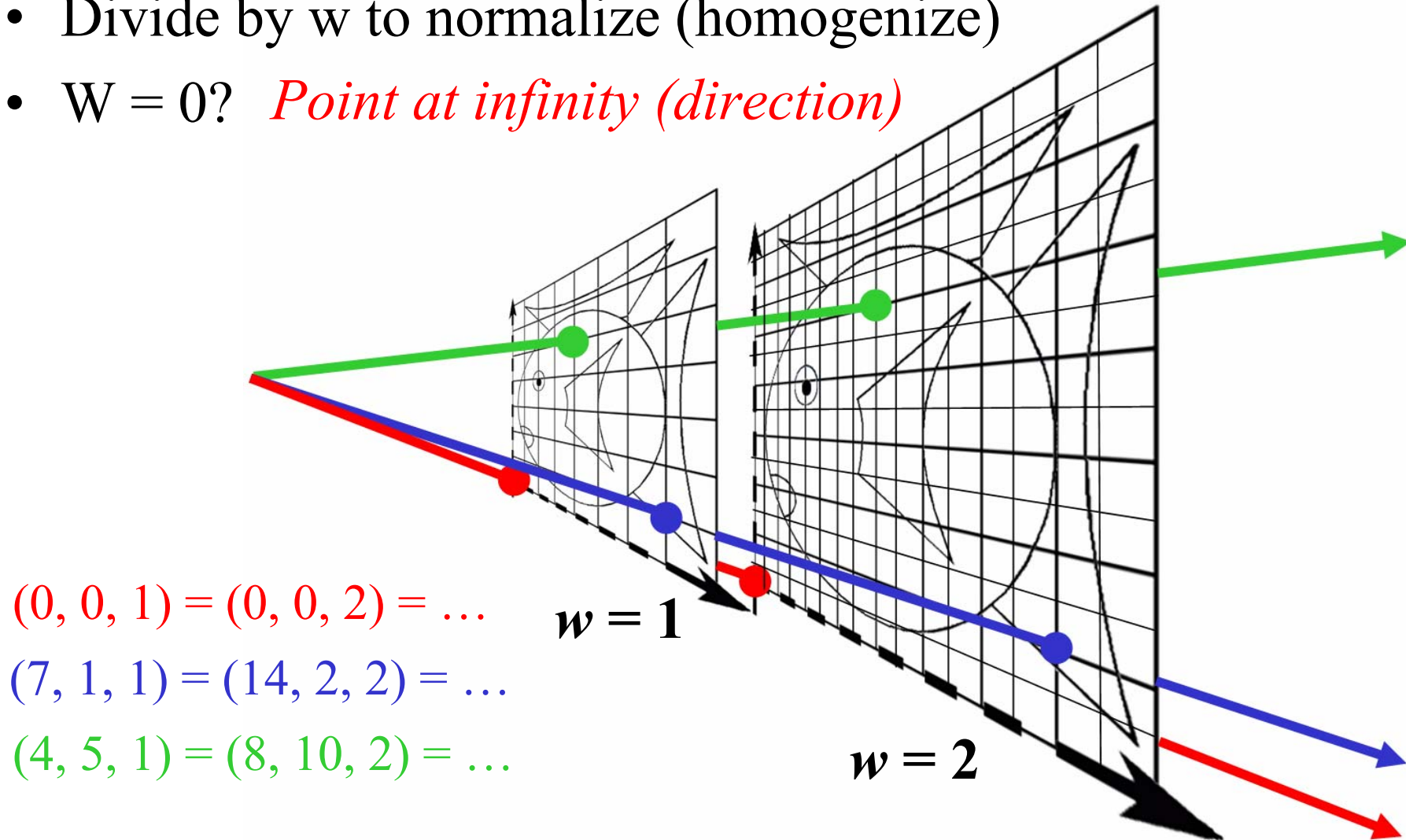
- Most of the time $w = 1$, and we can ignore it

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- If we multiply a homogeneous coordinate by an *affine matrix*, w is unchanged

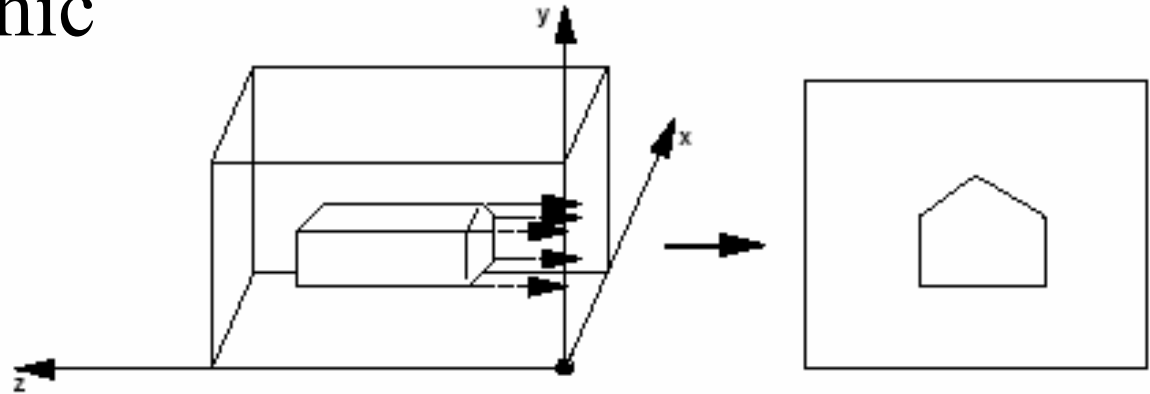
Homogeneous Visualization

- Divide by w to normalize (homogenize)
- $W = 0$? *Point at infinity (direction)*

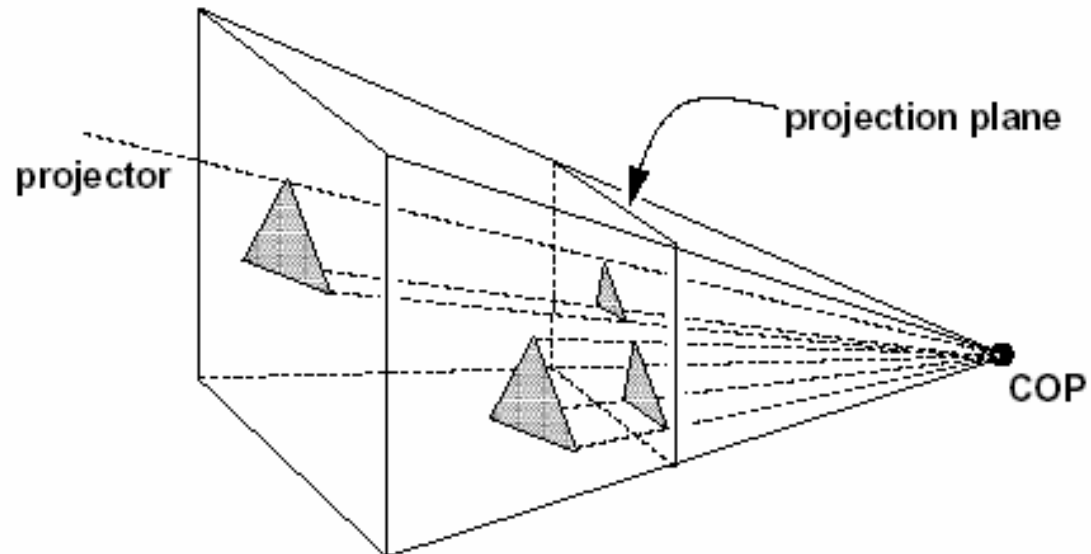


Orthographic vs. Perspective

- Orthographic

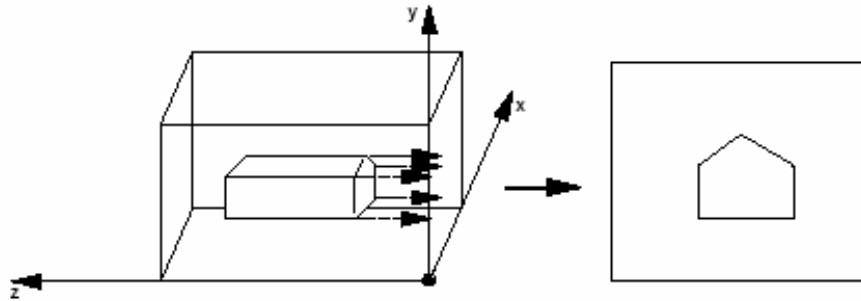


- Perspective



Simple Orthographic Projection

- Project all points along the z axis to the $z = 0$ plane

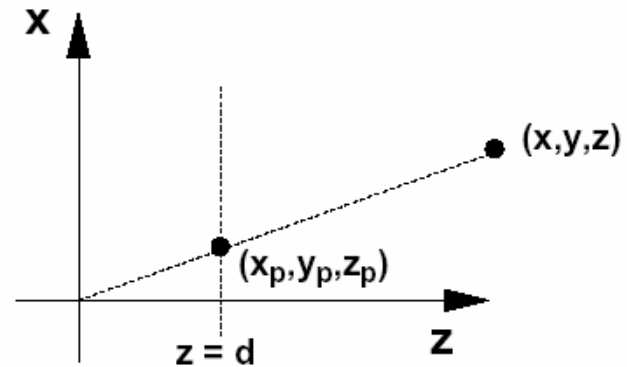


$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Simple Perspective Projection

- Project all points along the z axis to the $z = d$ plane, eyepoint at the origin:

$$\begin{aligned}x_p &= \frac{d \cdot x}{z} = \frac{x}{z/d} \\y_p &= \frac{d \cdot y}{z} = \frac{y}{z/d} \\z_p &= d\end{aligned}$$



homogenize

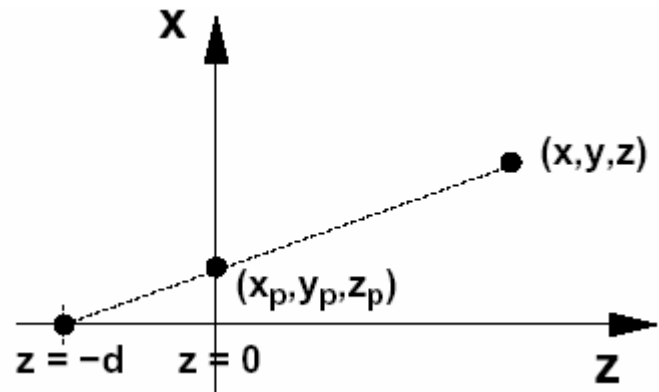
$$\begin{pmatrix} x * d / z \\ y * d / z \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z / d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Alternate Perspective Projection

- Project all points along the z axis to the $z = 0$ plane, eyepoint at the $(0,0,-d)$:

$$x_p = \frac{d \cdot x}{z + d} = \frac{x}{(z/d) + 1}$$

$$y_p = \frac{d \cdot y}{z + d} = \frac{y}{(z/d) + 1}$$



homogenize

$$\begin{pmatrix} x * d / (z + d) \\ y * d / (z + d) \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ (z + d) / d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

In the limit, as $d \rightarrow \infty$

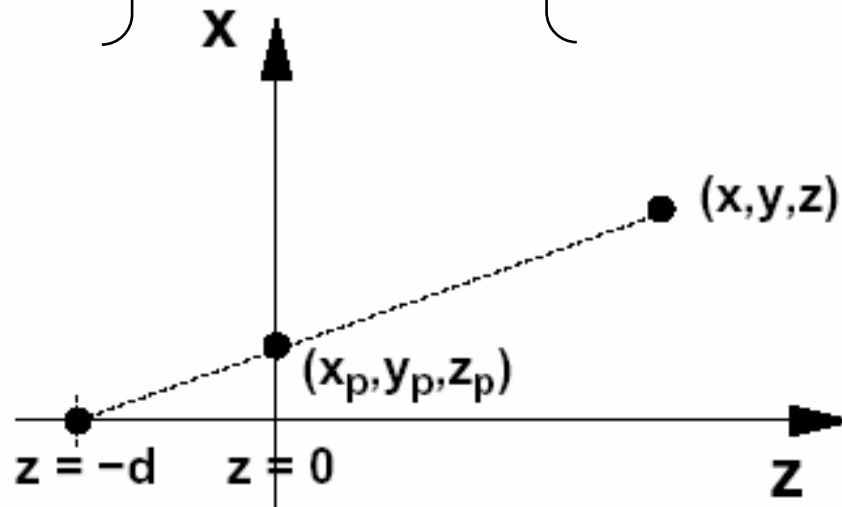
this perspective
projection matrix...

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

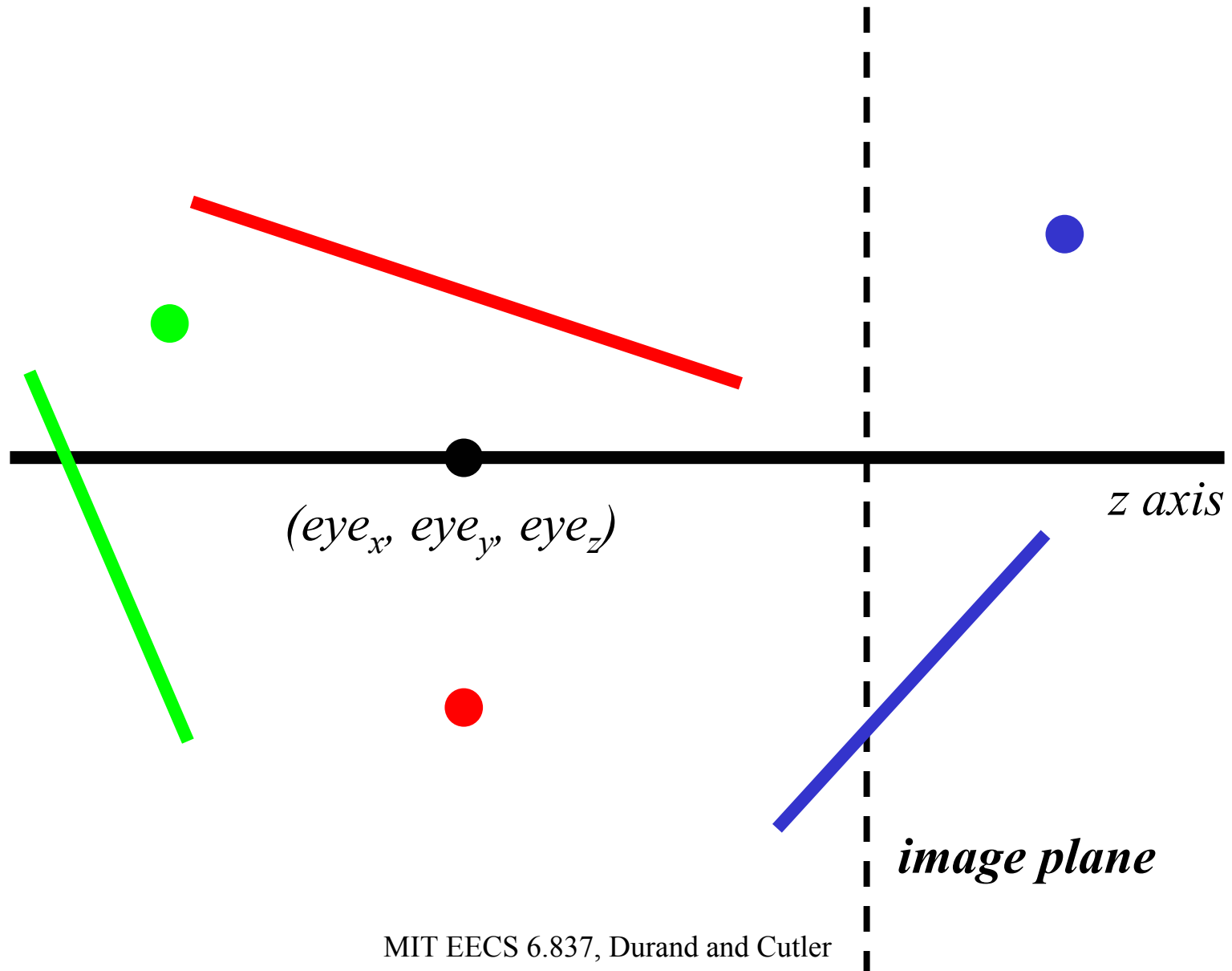
$\rightarrow$

...is simply an
orthographic projection

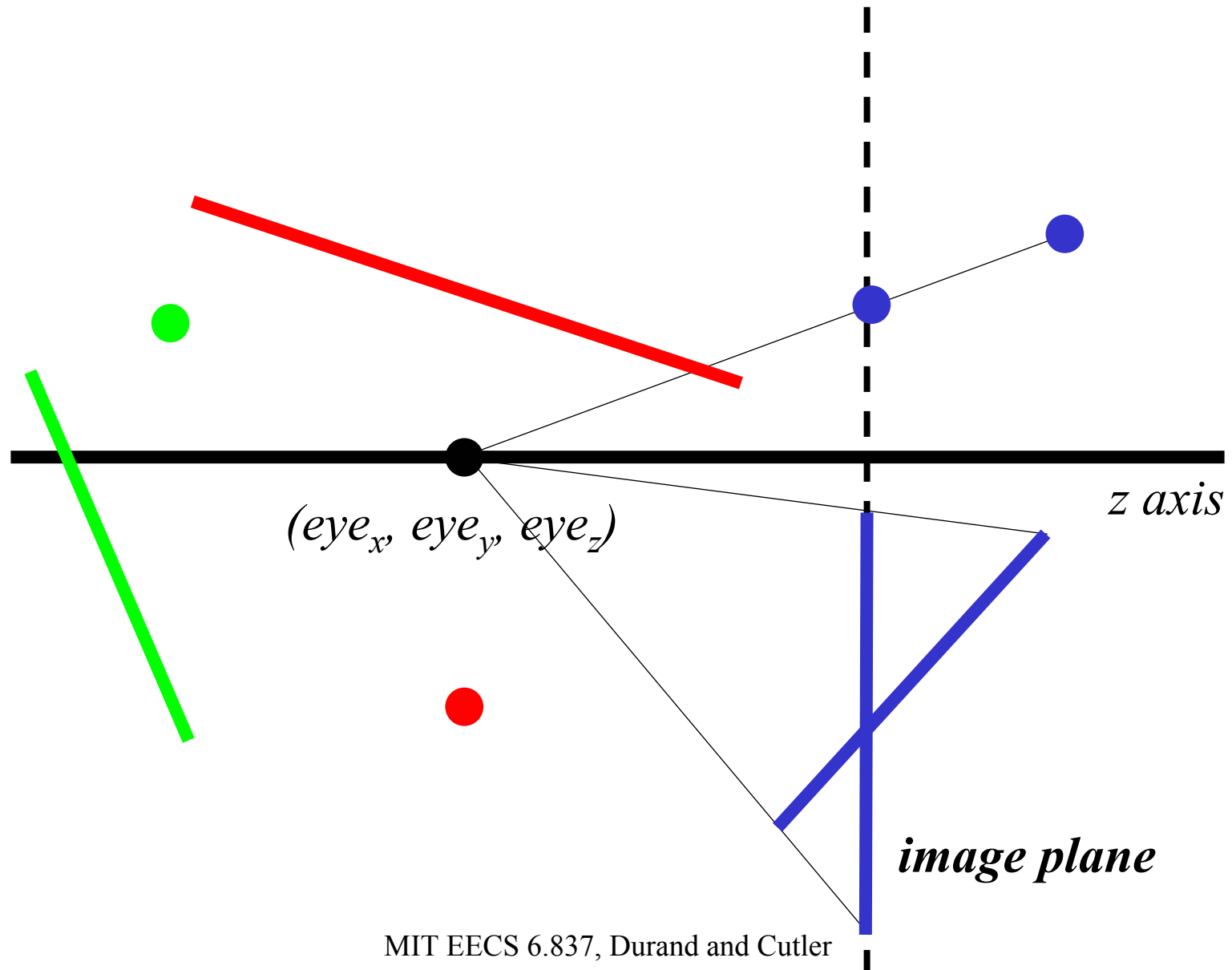
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



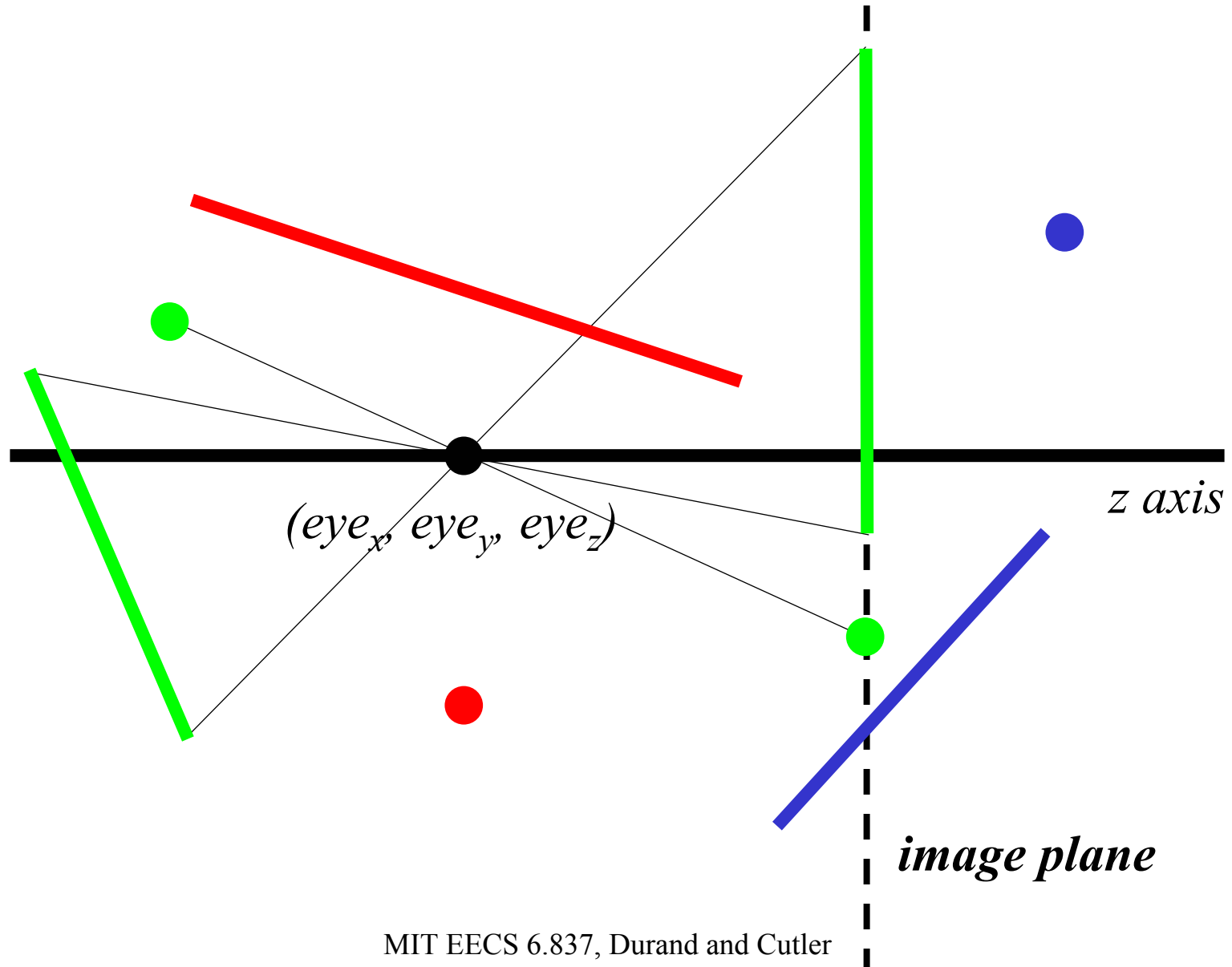
What if the p_z is $\leq eye_z$?



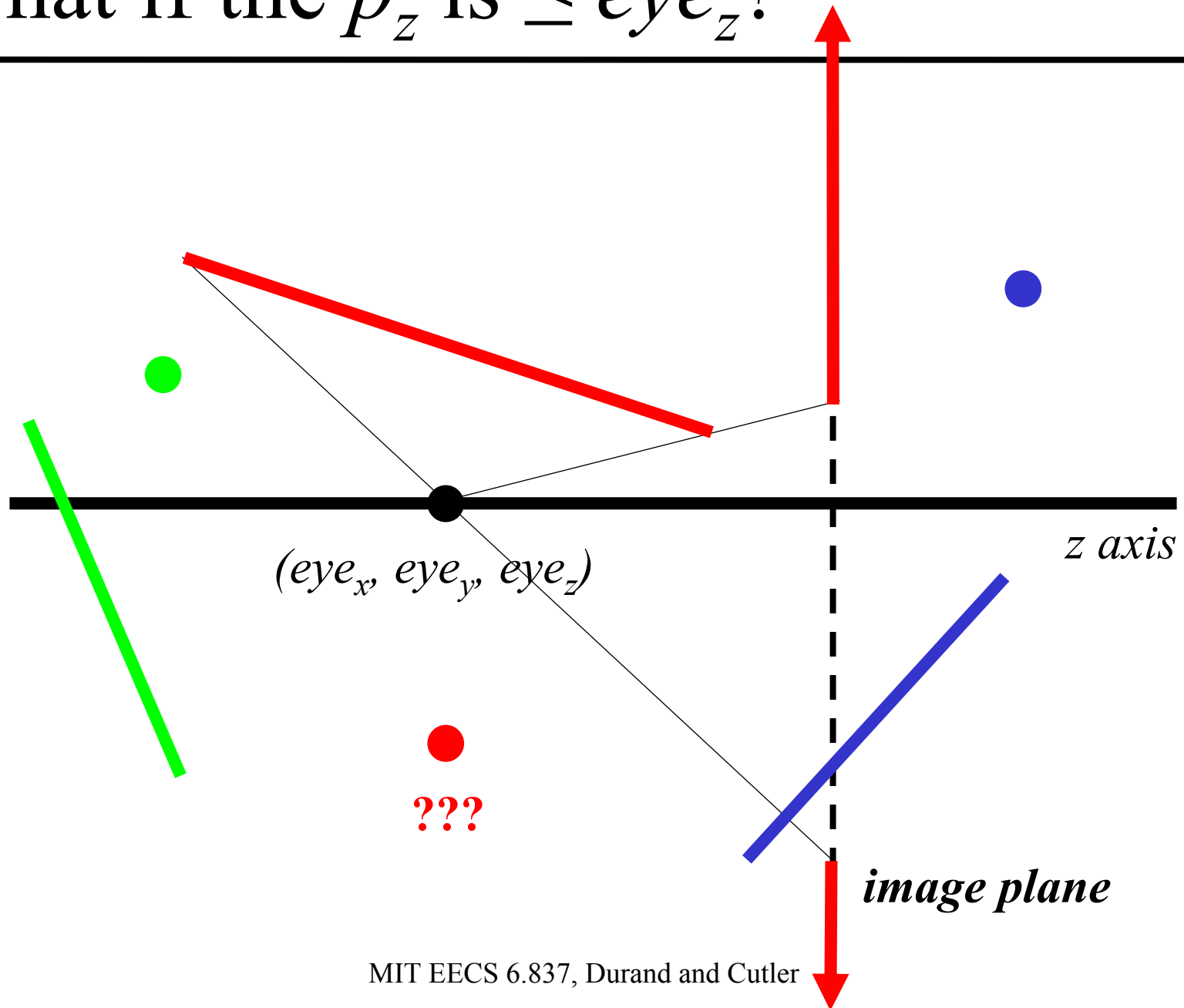
What if the p_z is $\leq eye_z$?



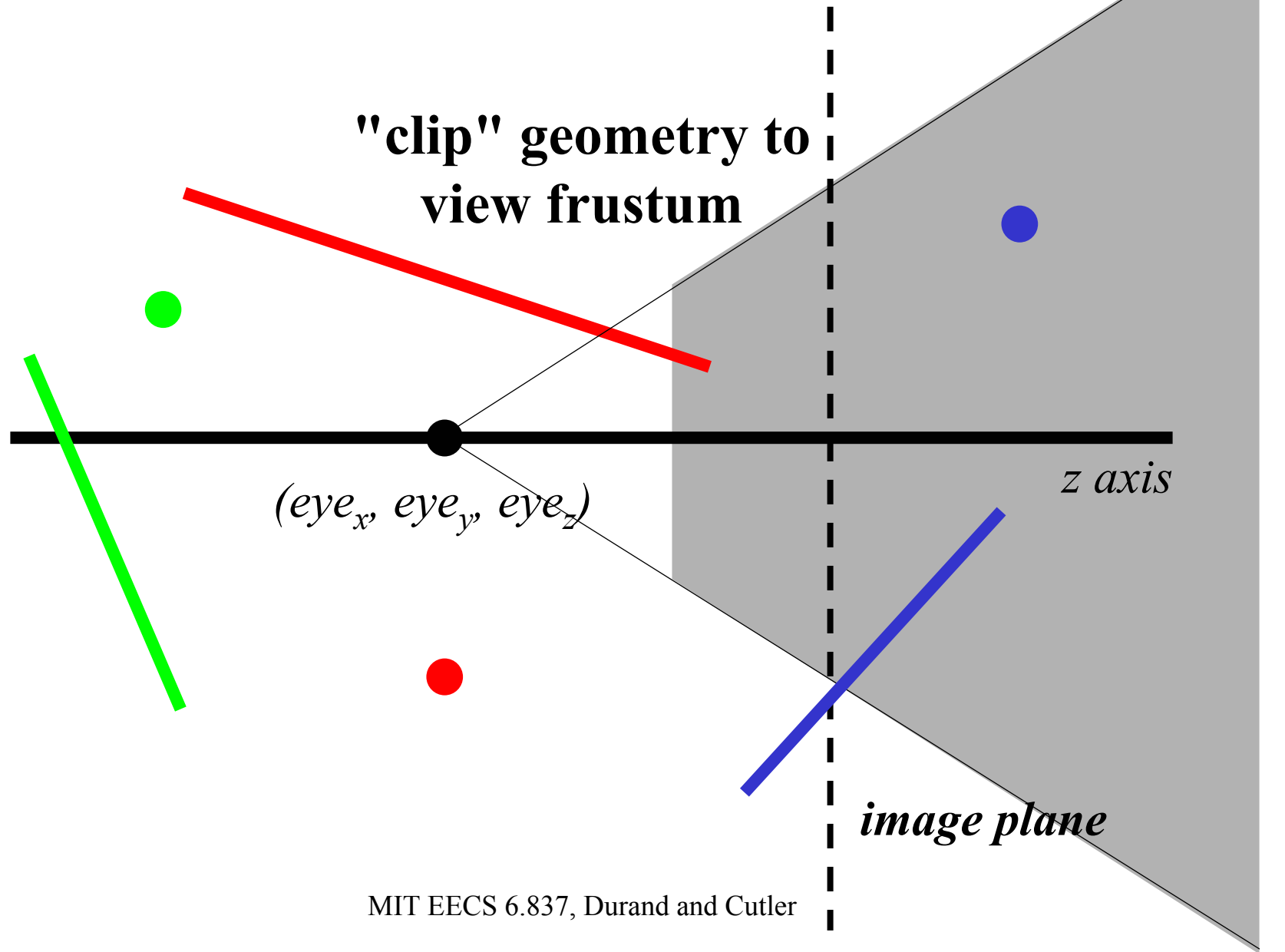
What if the p_z is $\leq eye_z$?



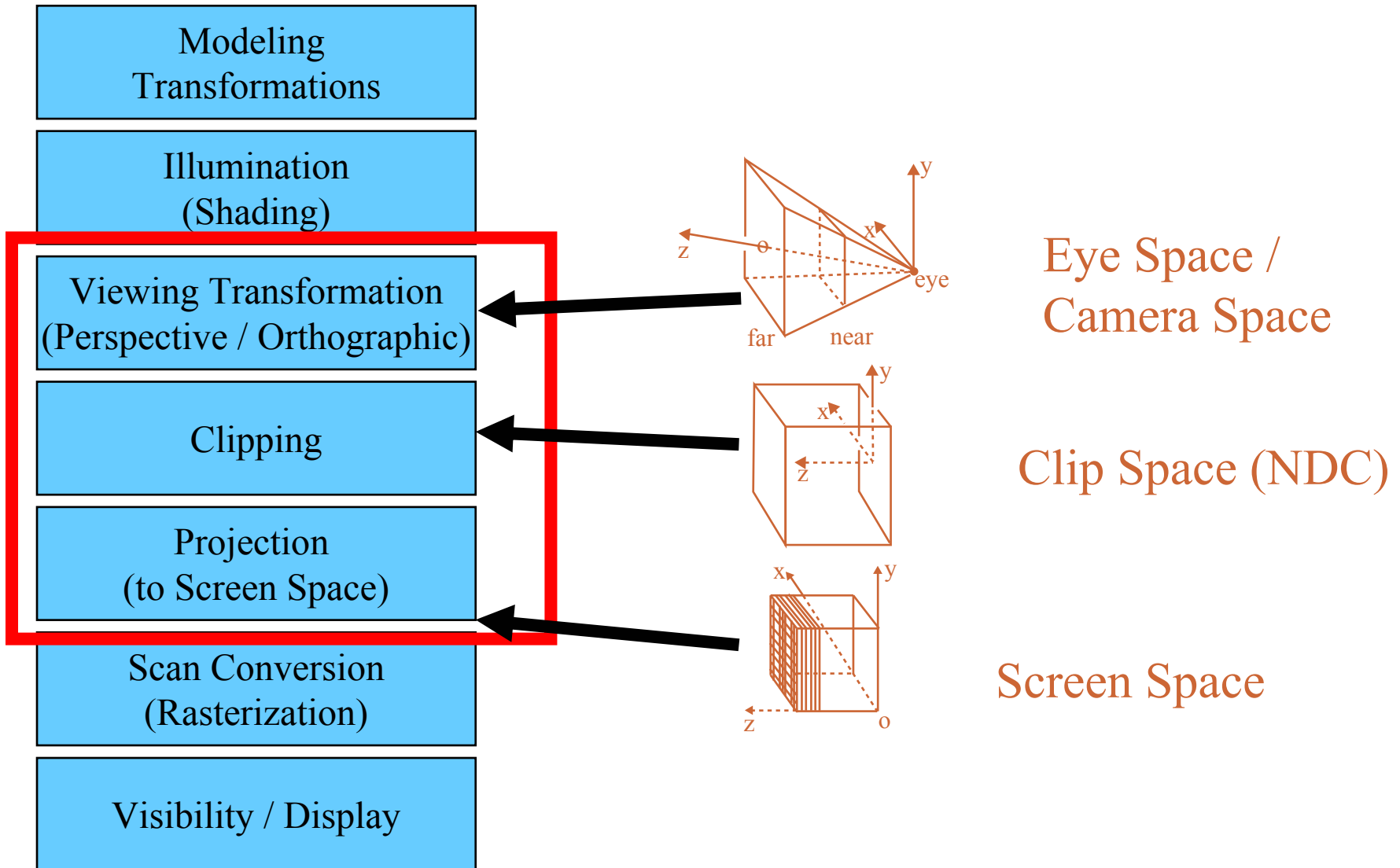
What if the p_z is $\leq eye_z$?



What if the p_z is $\leq eye_z$?

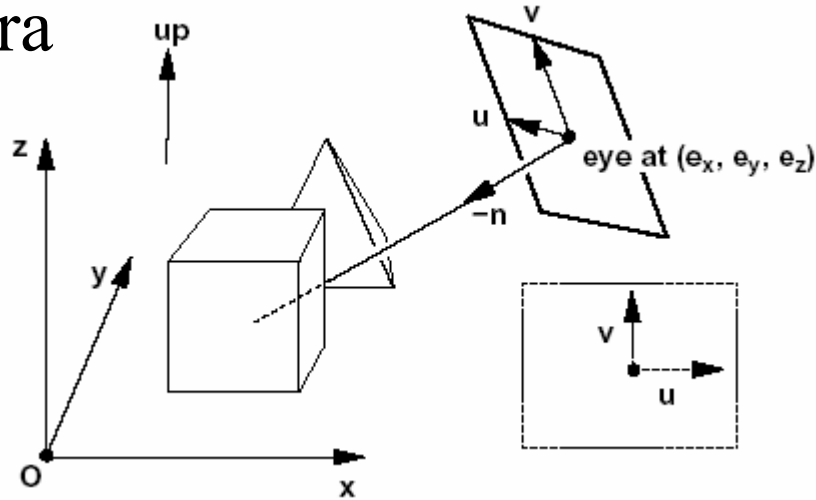


Where are projections in the pipeline?



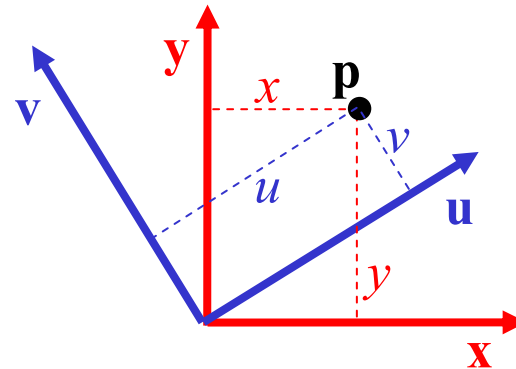
World Space $\rightarrow$ Eye Space

Positioning the camera



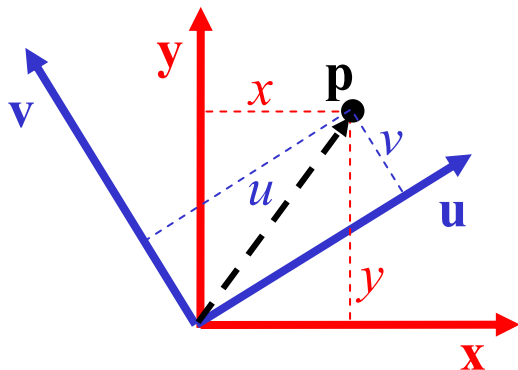
Translation + Change of orthonormal basis (Lecture 4)

- Given: coordinate frames **xyz** & **uvn**, and point $\mathbf{p} = (x, y, z)$
- Find: $\mathbf{p} = (u, v, n)$



Change of Orthonormal Basis

$$\begin{bmatrix} u \\ v \\ n \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



where:

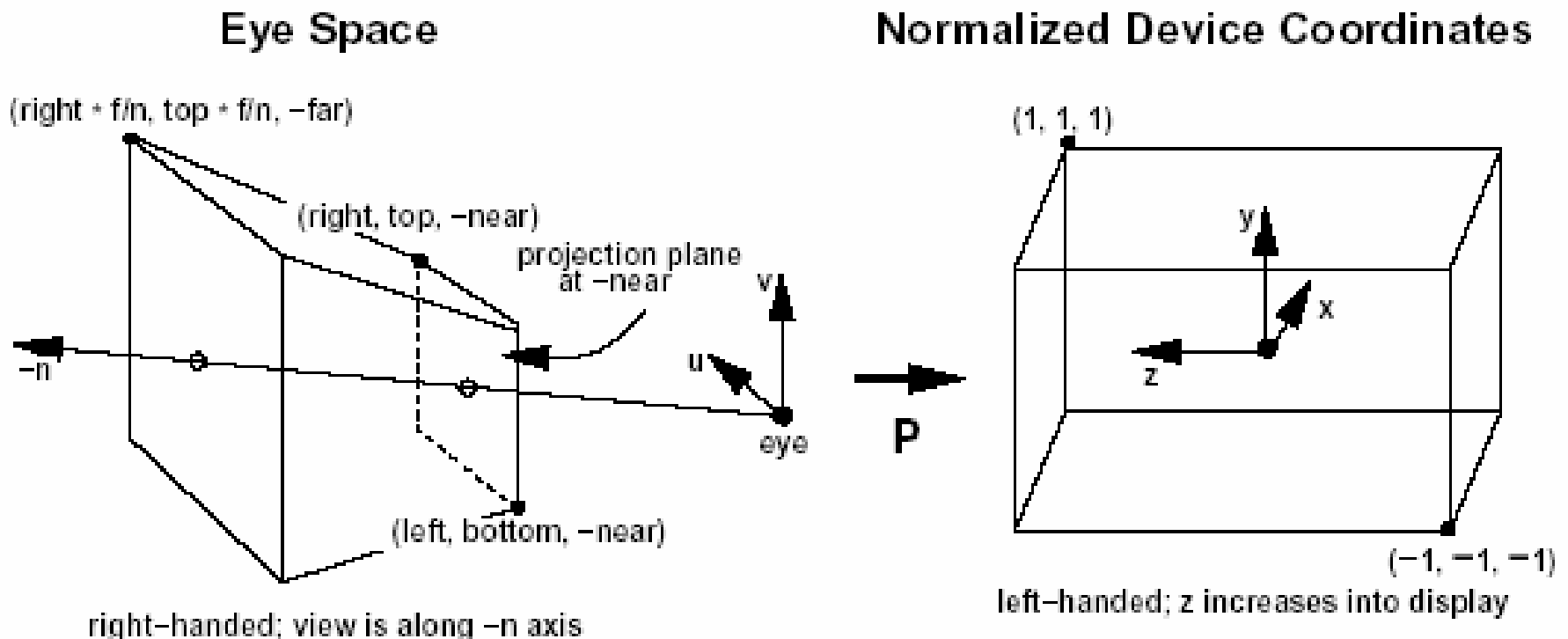
$$u_x = \mathbf{x} \cdot \mathbf{u}$$

$$u_y = \mathbf{y} \cdot \mathbf{u}$$

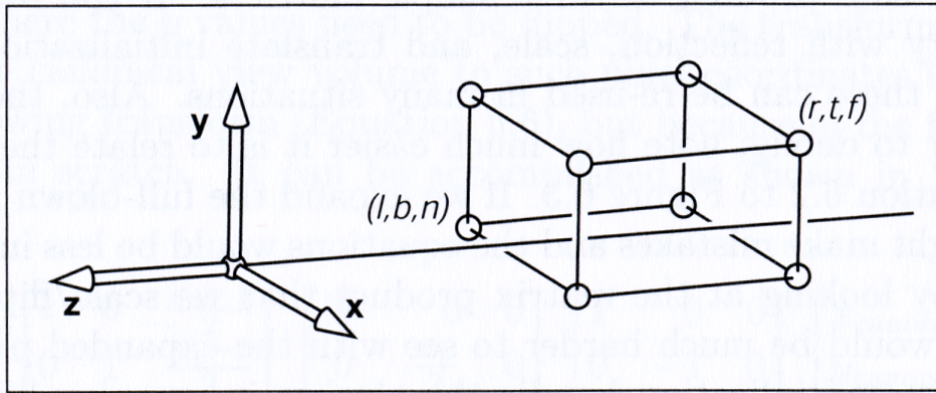
etc.

Normalized Device Coordinates

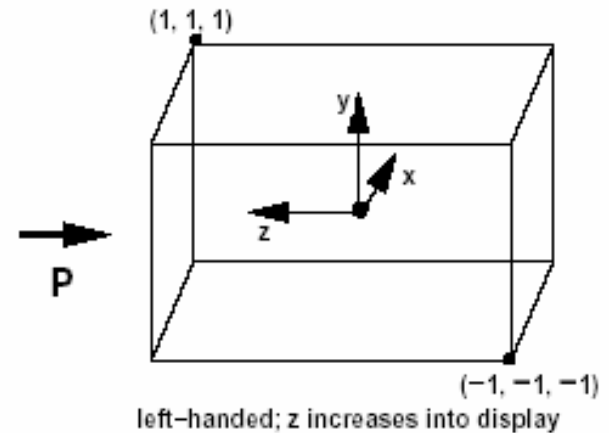
- Clipping is more efficient in a rectangular, axis-aligned volume: $(-1, -1, -1) \rightarrow (1, 1, 1)$ *OR* $(0, 0, 0) \rightarrow (1, 1, 1)$



Canonical Orthographic Projection



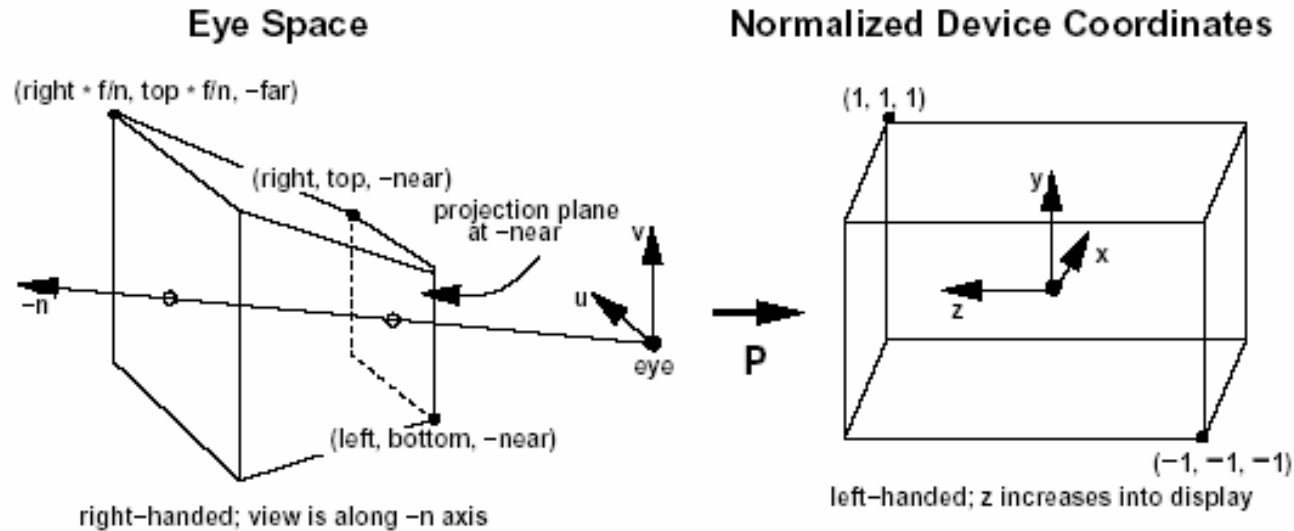
Normalized Device Coordinates



Orthographic

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & \frac{-(\text{right} + \text{left})}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{bottom} - \text{top}} & 0 & \frac{-(\text{bottom} + \text{top})}{\text{bottom} - \text{top}} \\ 0 & 0 & \frac{2}{\text{far} - \text{near}} & \frac{-(\text{far} + \text{near})}{\text{far} - \text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Canonical Perspective Projection



Perspective

$$\begin{bmatrix} x'w \\ y'w \\ z'w \\ w \end{bmatrix} = \begin{bmatrix} \frac{2 \cdot \text{near}}{\text{right} - \text{left}} & 0 & \frac{-(\text{right} + \text{left})}{\text{right} - \text{left}} & 0 \\ 0 & \frac{2 \cdot \text{near}}{\text{bottom} - \text{top}} & \frac{-(\text{bottom} + \text{top})}{\text{bottom} - \text{top}} & 0 \\ 0 & 0 & \frac{\text{far} + \text{near}}{\text{far} - \text{near}} & \frac{-2 \cdot \text{near} \cdot \text{far}}{\text{far} - \text{near}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Questions?

Next Time: Line Rasterization