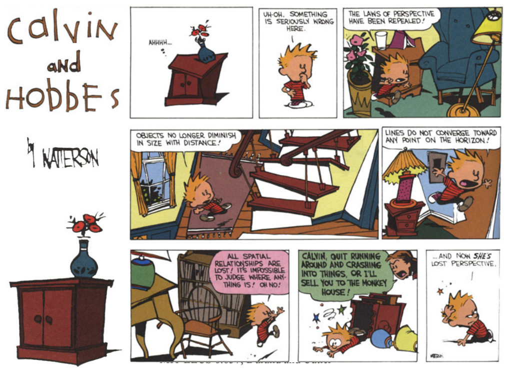
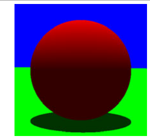
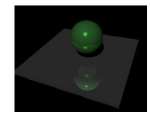
Graphics Pipeline: Projective Transformations

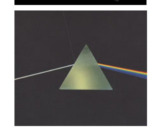


Last Time

- Shadows
  - cast ray to light
  - stop after first intersection
- Reflection & Refraction
  - compute direction of recursive ray
- Recursive Ray Tracing
  - maximum number of bounces OR
  - contribution < error threshhold
- Epsilon…

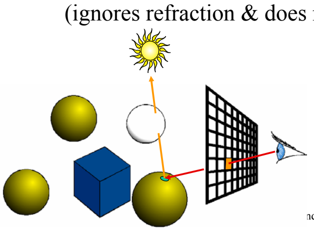




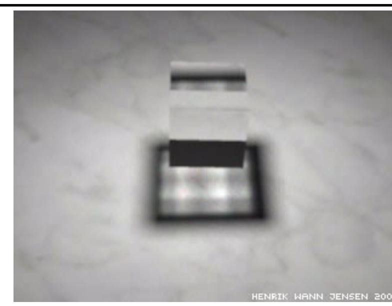


Does Ray Tracing Simulate Physics?

- Ray Tracing is full of dirty tricks
- For example, shadows of transparent objects:
  - opaque?
  - multiply by transparency color?
    (ignores refraction & does not produce caustics)



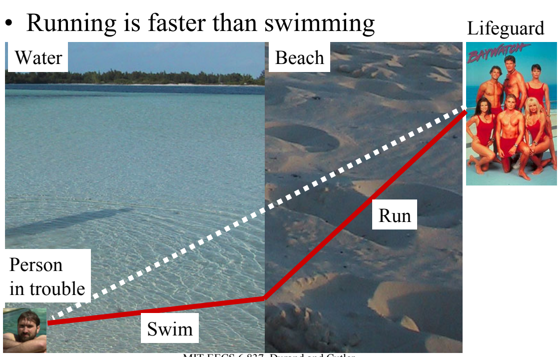
Correct Transparent Shadow



Animation by Henrik Wann Jensen
Using advanced refraction technique
(refraction for illumination is usually not handled that well)

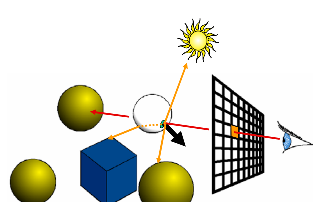
Refraction and the Lifeguard Problem

- Running is faster than swimming



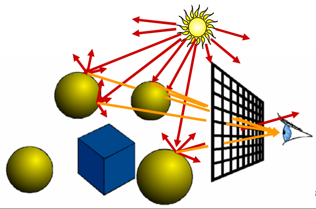
Does Ray Tracing Simulate Physics?

- Photons go from the light to the eye, not the other way
- What we do is *backward ray tracing*



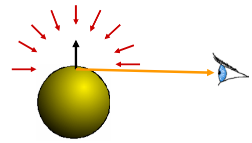
Forward Ray Tracing

- Start from the light source
  - But low probability to reach the eye
- What can we do about it?
  - Always send a ray to the eye…. still not efficient



The Rendering Equation

- Clean mathematical framework for light-transport simulation



- At each point, outgoing light in one direction is the integral of incoming light in all directions multiplied by reflectance property
- *We'll see this later…*

Questions?

Today

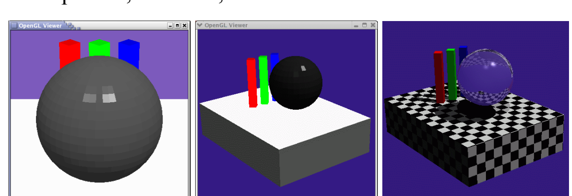
- Ray Casting / Tracing vs. Scan Conversion
  - advantages & disadvantages
  - when is each appropriate?
- The Graphics Pipeline
- Projective Transformations
- Introduction to Clipping

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)

How Do We Render Interactively?

- Use graphics hardware (the graphics pipeline), via OpenGL, MesaGL, or DirectX



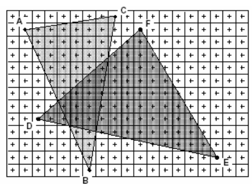
assignment 3 *assignment 4…*

- Most global effects available in ray tracing will be sacrificed, but some can be approximated

Scan Conversion

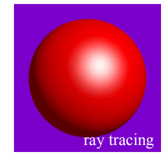
- Given a primitive's vertices & the illumination at each vertex:
- Figure out which pixels to "turn on" to render the primitive
- Interpolate the illumination values to "fill in" the primitive
- At each pixel, keep track of the closest primitive (z buffer)

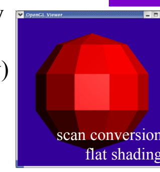
```
glBegin(GL_TRIANGLES)
glNormal3f(...)
glVertex3f(...)
glVertex3f(...)
glVertex3f(...)
glEnd();
```

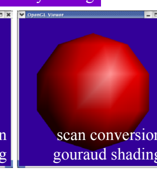


Limitations of Scan Conversion

- 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 many more triangles than pixels?







Ray Casting vs. Rendering Pipeline

Ray Casting

```
For each pixel
    For each object
```

Send pixels to the scene

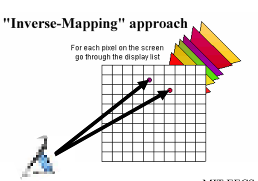
Discretize first

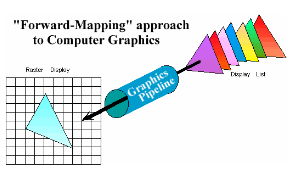
Rendering Pipeline

```
For each triangle
    For each pixel
```

Project scene to the pixels

Discretize last





Ray Casting vs. Rendering Pipeline

Ray Casting

```
For each pixel
    For each object
```

- Whole scene must be in memory
- Depth complexity: no computation for hidden parts
- Atomic computation
- More general, more flexible
  - Primitives, lighting effects, adaptive antialiasing

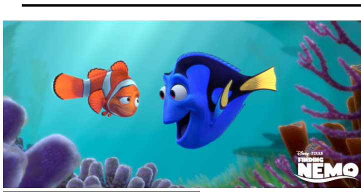
Rendering Pipeline

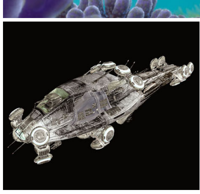
```
For each triangle
    For each pixel
```

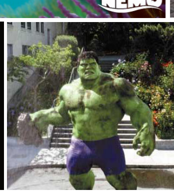
- Primitives processed one at a time
- Coherence: geometric transforms for vertices only
- Early stages involve analytic processing
- Computation increases with depth of the pipeline
  - Good bandwidth/computation ratio
- Sampling occurs late in the pipeline
- Minimal state required

Movies

both pipeline and ray tracing

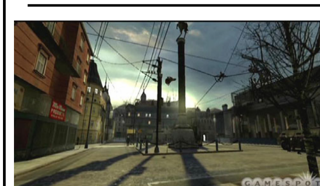


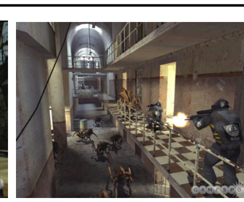


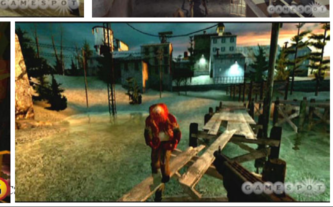


Games

pipeline

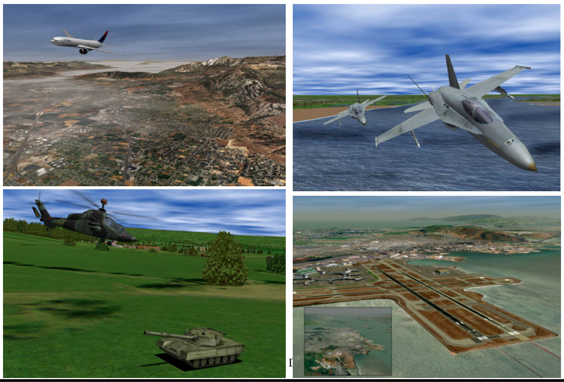






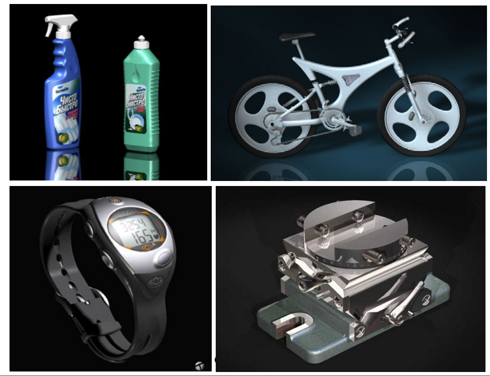
Simulation

pipeline (painter for a long time)



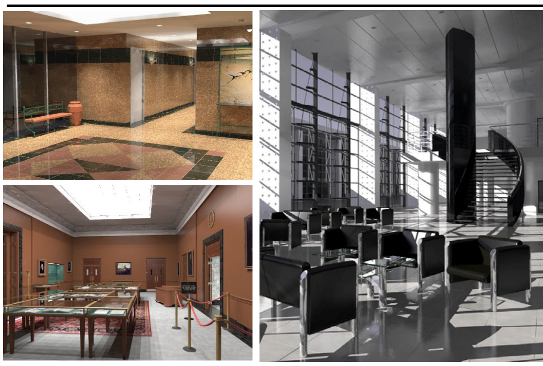
CAD-CAM & Design

pipeline during design, anything for final image



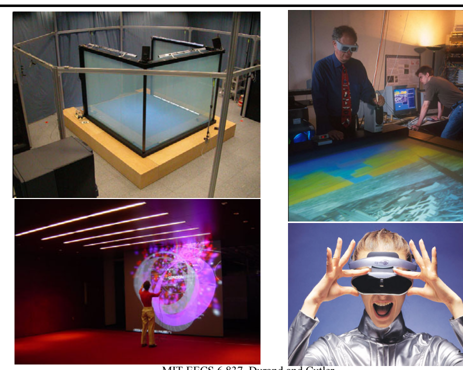
Architecture

ray-tracing, pipeline with preprocessing for complex lighting



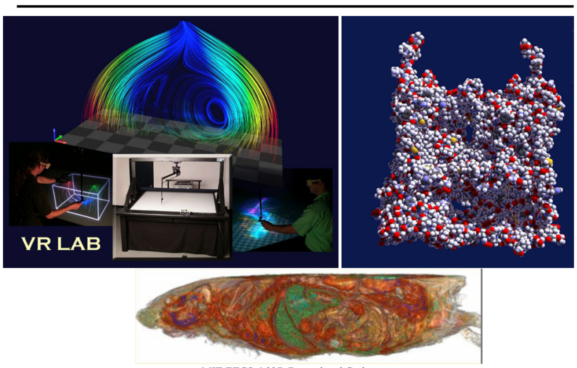
Virtual Reality

pipeline



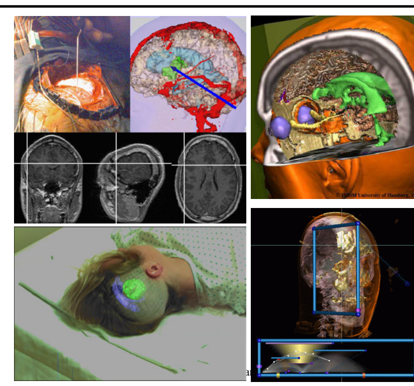
Visualization

mostly pipeline, ray-tracing for high-quality eye candy, interactive ray-tracing is starting



Medical Imaging

same as visualization



Questions?

Today

- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations
- Introduction to Clipping

The Graphics Pipeline

- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

Input:
Geometric model:
Description of all object, surface, and light source geometry and transformations
Lighting model:
Computational description of object and light properties, interaction (reflection)
Synthetic Viewpoint (or *Camera*):
Eye position and viewing frustum
Raster Viewport:
Pixel grid onto which image plane is mapped

Output:
Colors/Intensities suitable for framebuffer display
(For example, 24-bit RGB value at each pixel)

The Graphics Pipeline

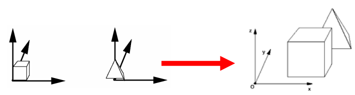
- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

- 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)

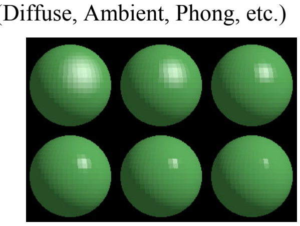


Object space — World space

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.)

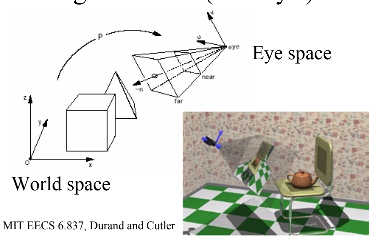


Viewing Transformation

- Modeling Transformations
- Illumination (Shading)
- **Viewing Transformation (Perspective / Orthographic)**
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

• Maps world space to eye space

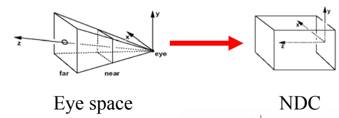
• Viewing position is transformed to origin & direction is oriented along some axis (usually *z*)



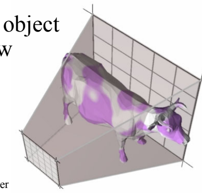
Clipping

- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- **Clipping**
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

• Transform to Normalized Device Coordinates (NDC)



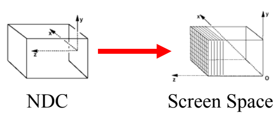
• Portions of the object outside the view volume (view frustum) are removed

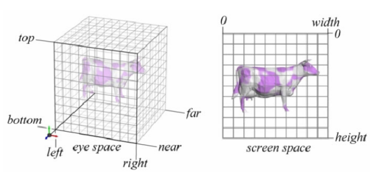


Projection

- 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 place (screen space)



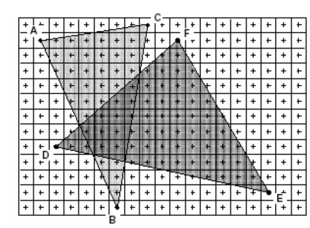


Scan Conversion (Rasterization)

- 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

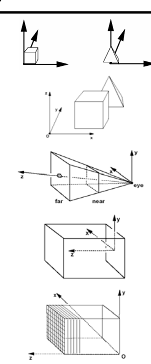
- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- **Visibility / Display**

• Each pixel remembers the closest object (depth buffer)

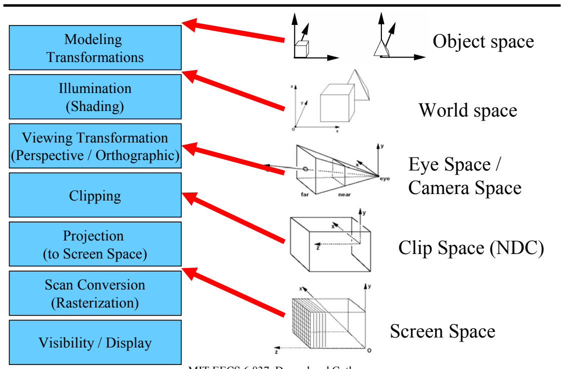
• Almost every step in the graphics pipeline involves a change of coordinate system. Transformations are central to understanding 3D computer graphics.

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] → [1,1,1]
- Screen space
  - indexed according to hardware attributes



Coordinate Systems in the Pipeline

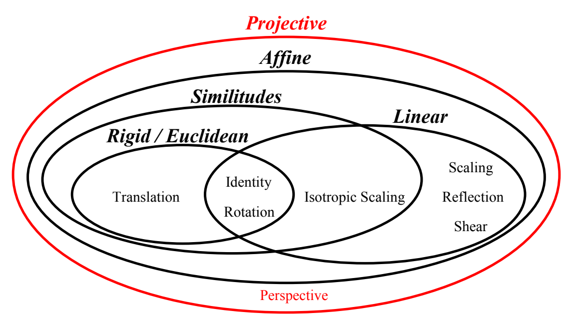


Questions?

Today

- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations
  - Transformations & Homogeneous Coordinates
  - Orthographic & Perspective Projections
  - Coordinate Systems & Projections in the Pipeline
  - Canonical View Volume
- Introduction to Clipping

Remember Transformations?



Homogeneous Coordinates

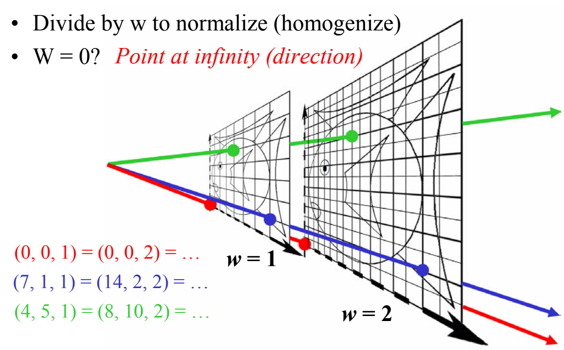
- Most of the time w = 1, and we can ignore it

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

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

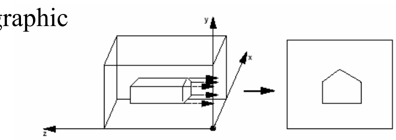
Homogeneous Visualization

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

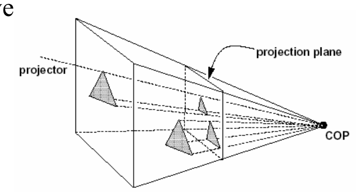


Orthographic vs. Perspective

- Orthographic

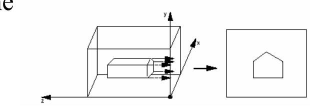


- Perspective



Simple Orthographic Projection

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

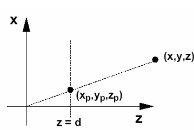


$$\begin{pmatrix} x \\ y \\ \color{red}{0} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \color{red}{0} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Simple Perspective Projection

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

$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}$$
$$y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$
$$z_p = d$$



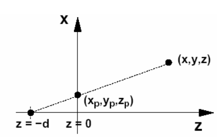
homogenize

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

Alternate Perspective Projection

- Project all points 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

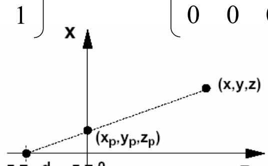
$$\begin{pmatrix} \color{red}{x * d / (z + d)} \\ \color{red}{y * d / (z + d)} \\ \color{red}{0} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \color{red}{0} \\ \color{red}{(z + d)/ d} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \color{red}{0} & 0 \\ 0 & 0 & \color{red}{1/d}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

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

this perspective projection matrix...

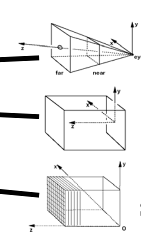
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

...is simply an orthographic projection



Where are projections in the pipeline?

- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display



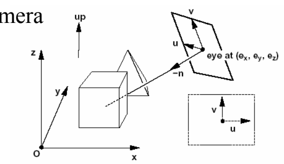
Eye Space / Camera Space

Clip Space (NDC)

Screen Space

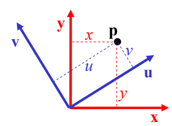
World Space → Eye Space

Positioning the camera



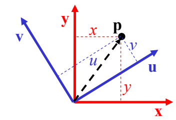
Translation + Change of orthonormal basis

- Given: coordinate frames **xyz** & **uvn,** and point **p** = (x,y,z)
- Find: **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:

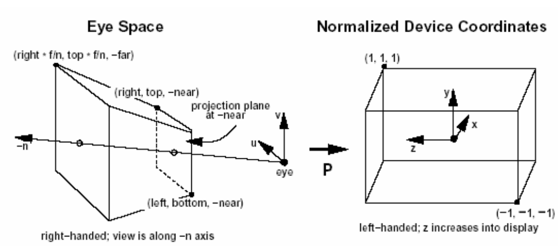
$u_x = \mathbf{x} \cdot \mathbf{u}$

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

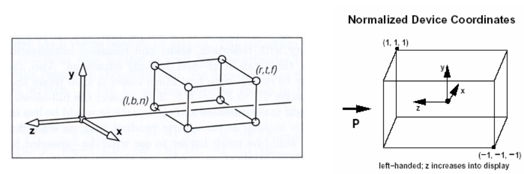
etc.

Normalized Device Coordinates

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



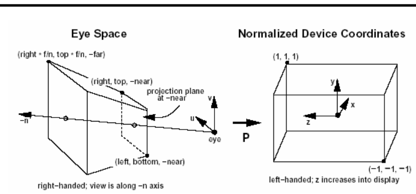
Canonical Orthographic Projection



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

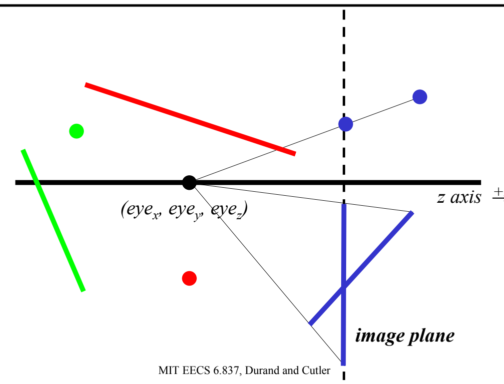
$$\begin{bmatrix} x'w \\ y'w \\ z'w \\ w \end{bmatrix} = \begin{bmatrix} \frac{2 \cdot \text{near}}{\text{right}-\text{left}} & 0 & \frac{-(\text{right}+\text{left})}{\text{right}-\text{left}} & 0 \\ 0 & \frac{2 \cdot \text{near}}{\text{bottom}-\text{top}} & \frac{-(\text{bottom}+\text{top})}{\text{bottom}-\text{top}}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

Questions?

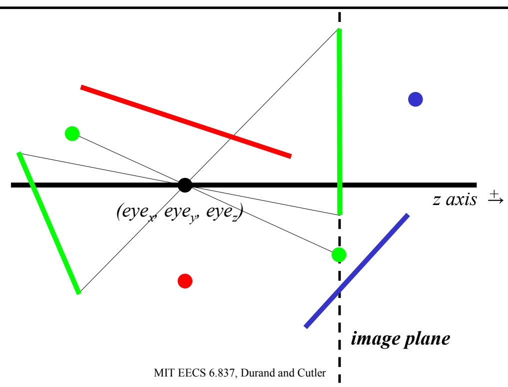
Today

- Ray Casting / Tracing vs. Scan Conversion
- The Graphics Pipeline
- Projective Transformations
- Introduction to Clipping
  - Projecting to the Image Plane
  - Why Clip?
  - Clipping Strategies

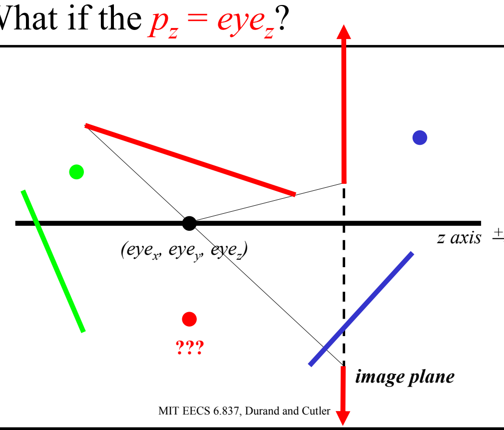
What if the p_z is > eye_z?



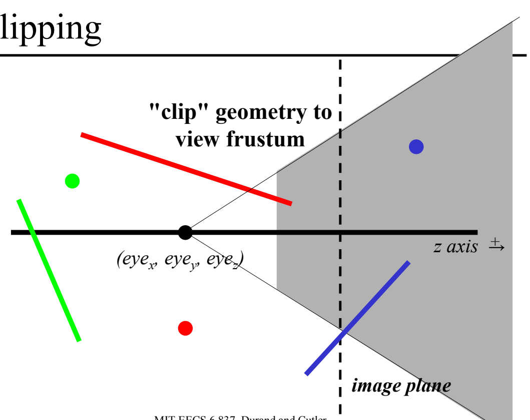
What if the p_z is < eye_z?



What if the p_z = eye_z?

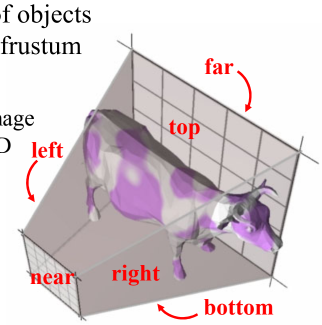


Clipping



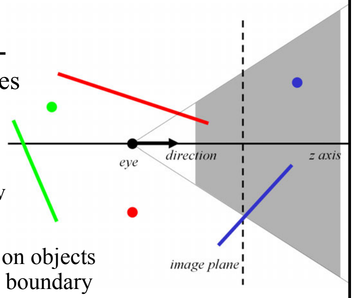
Clipping

- Eliminate portions of objects outside the viewing frustum
- View Frustum
  - boundaries of the image plane projected in 3D
  - a near & far clipping plane
- User may define additional clipping planes



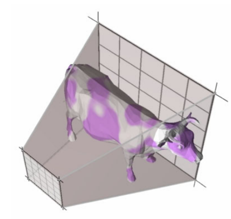
Why Clip?

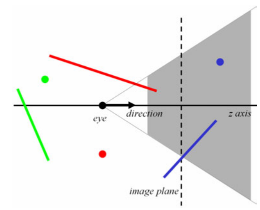
- Avoid degeneracies
  - Don't draw stuff behind the eye
  - Avoid division by 0 and overflow
- Efficiency
  - Don't waste time on objects outside the image boundary
- Other graphics applications (often non-convex)
  - Hidden-surface removal, Shadows, Picking, Binning, CSG (Boolean) operations (2D & 3D)



Clipping Strategies

- Don't clip (and hope for the best)
- Clip on-the-fly during rasterization
- Analytical clipping: alter input geometry





Next Time: Clipping & Line Rasterization