Coordinate systems & Viewing, Rendering Polygonal Objects

Recap

- 3D Transformations
- Nested Coordinate Systems
- "Virtual Cameras"

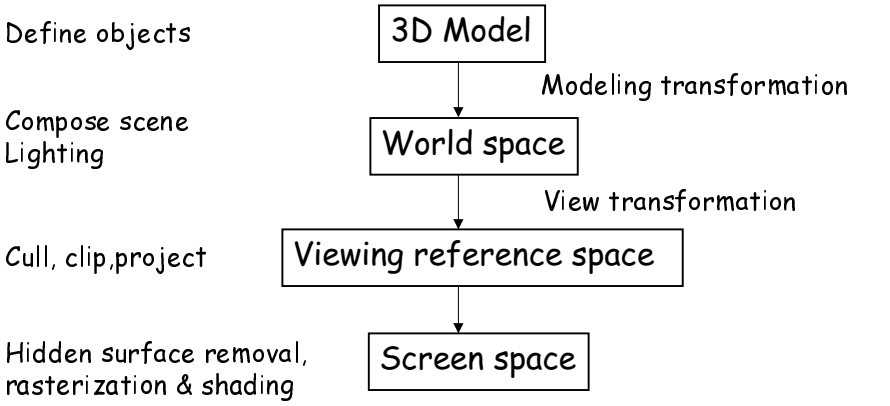
Recap: Transforms and OpenGL

- Transforms go onto matrix "stacks"
- All vertices xformed by top matrices

Mentioned two stacks. Why?

- Separate modeling & viewing xforms

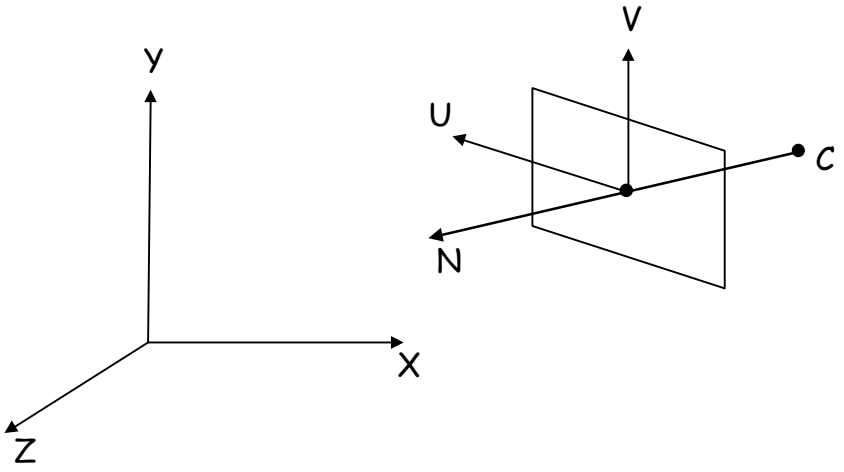
3D viewing process



Camera Coordinate System (aka Eye aka View)

- What do we need to define what is seen?

Specifying a view



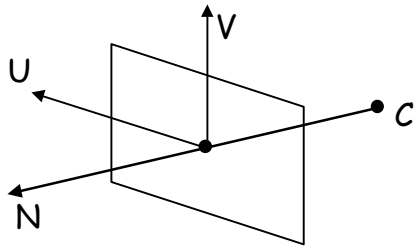
Translate C to origin

$$\begin{pmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T(-C)}$$

Rotate UVN->XYZ

We want to take u into (1, 0, 0)
v into (0, 1, 0)
n into (0, 0, 1)

First derive n, u, and v from user input:



Rotate UVN

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{R_{UVN}}$$

Rendering Polygonal Meshes

- Optimize for polygon
  - Work:
    - Per polygon
    - Per pixel

World->ViewSpace

- Per vertex
  - Transformations
  - Clipping

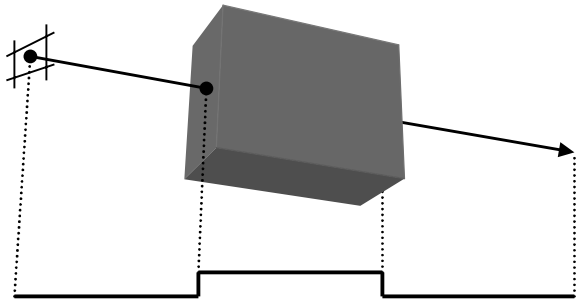
ViewSpace->Screen

- Lots per vertex, incremental per pixel
  - Shading
  - Hidden surface removal
- Polygon order independent

Rendering CSG

- Three ways of rendering CSG reps
  - Ray Tracing
  - Volumetric (Voxel rep)
  - Variation of z-buffer

Basic Idea: Ray/primitive Classification



Combine Ray Classification

Modify z-buffer for CSG