CS 559 Lecture 18
Viewing

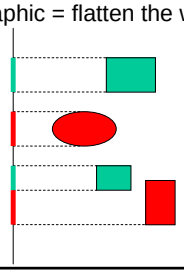
Mike Gleicher
October 2007
Notes – not for display

Viewing Transformations

- How do we transform the 3D world to the 2D window?
- Concepts:
  - World Coordinates
  - View (or window) VOLUME
    - Need 3rd dimension later to get occlusions right
  - Viewing Coordinates
    - 3D Viewing coordinates
- Separate Issues
  - Visibility (what's in front)
  - Clipping (what is outside of the view volume)

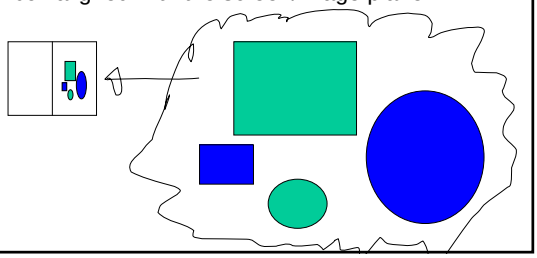
Orthographic Projection

- Projection = transformation that reduces dimension
- Orthographic = flatten the world onto the film plane



View Volumes / Transformations

- Viewing transformation puts the world into the viewing volume
- A box aligned with the screen/image plane



Canonical View Volume

- -1 to 1 (zero centered)
- XY is screen (y-up)
- Z is towards viewer (right handed coordinates)
  - Negative Z is into screen
- For this reason, some people like left-handed

2 Views of Viewing Transform

- Put world into viewing volume
- Position camera in world (view volume into world)

- Clip stuff that is outside of the volume
- Somehow get closer stuff to show up instead of farther things (if we want solid objects)

Orthographic Projection

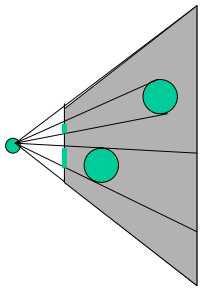
- Rotate / Translate / Scale View volume
  - Can map any volume to view volume
- Sometimes pick skews
- Things far away are just as big
  - No perspective
- Easy – and we can make measurements
  - Useful for technical drawings
  - Looks weird for real stuff
    - Far away objects too big

Perspective Projection

- Farther objects get smaller
- Eye (or focal) point
- Image plane
- View frustum (truncated pyramid)
- Two ways to look at it:
  - Project world onto image plane
  - Transform world into rectangular view volume (that is then orthographically projected)

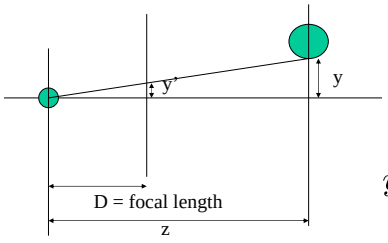
Perspective

- Eye point
- Film plane
- Frustum
- Simplification
  - Film plane centered with respect to eye
  - Site down negative Z axis
    - Can transform world to fit



Basic Perspective

- Similar Triangles
- Warning = using d for focal length (like book)
  - F will be "far plane"



$$\frac{y}{z} = \frac{y'}{d}$$

$$y' = \frac{d}{z} y$$

Use Homogeneous coordinates!

- Use divide by w to get perspective divide
- Issues with simple version:
  - Font / back of viewing volume
  - Need to keep some of Z in Z (not flatten)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ z/z = 1 \\ 1 \end{bmatrix}$$

The real perspective matrix

- N = near distance, F = far distance
- Z = n put on front plane, z=f put on far plane

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & -n & 0 \end{bmatrix}$$

Shirley's Perspective Matrix

- After we do the divide, we get an unusual thing for z – it does preserve the order and keeps n&f

$$\mathbf{Px} = \mathbf{P}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\frac{n}{z} \\ y\frac{n}{z} \\ n + f - \frac{fn}{z} \\ 1 \end{bmatrix}$$

Camera Model

- The "window coordinate" system is all the we really know
- In a sense, it is the camera coordinate system
- Easiest to think about it as a camera taking a picture of the work
- Transform world coordinates into camera coordinates
  - Or, think about it the other way…

How to describe cameras?

- Rotate and translate (and scale) the world to be in view
- The camera is a physical object (that can be rotated and translated in the world)
- Easier ways to specify cameras
  - Lookfrom/at/vup