Viewing/Projections I

Week 3, Fri Jan 25

Review: Camera Motion

- rotate/translate/scale difficult to control
- arbitrary viewing position
  - eye point, gaze/lookat direction, up vector
Review: World to View Coordinates

- translate **eye** to origin
- rotate **view** vector (**lookat** – **eye**) to **w** axis
- rotate around **w** to bring **up** into **vw**-plane

\[
M_{w2v} = \begin{bmatrix}
    u_x & u_y & u_z & -u \cdot e \\
    v_x & v_y & v_z & -v \cdot e \\
    w_x & w_y & w_z & -w \cdot e \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]
Projections I
Pinhole Camera

- ingredients
  - box, film, hole punch
- result
  - picture

www.kodak.com
www.pinhole.org
www.debevec.org/Pinhole
Pinhole Camera

- theoretical perfect pinhole
- light shining through tiny hole into dark space yields upside-down picture
Pinhole Camera

- non-zero sized hole
- blur: rays hit multiple points on film plane
Real Cameras

- pinhole camera has small aperture (lens opening)
  - minimize blur

- problem: hard to get enough light to expose the film

- solution: lens
  - permits larger apertures
  - permits changing distance to film plane without actually moving it
    - cost: limited depth of field where image is in focus

Graphics Cameras

- real pinhole camera: image inverted

- computer graphics camera: convenient equivalent
General Projection

• image plane need not be perpendicular to view plane
Perspective Projection

- our camera must model perspective
Perspective Projection

- our camera must model perspective
Projective Transformations

• planar geometric projections
  • planar: onto a plane
  • geometric: using straight lines
  • projections: 3D -> 2D
• aka projective mappings

• counterexamples?
Projective Transformations

- properties
  - lines mapped to lines and triangles to triangles
  - parallel lines do NOT remain parallel
    - e.g. rails vanishing at infinity

- affine combinations are NOT preserved
  - e.g. center of a line does not map to center of projected line (perspective foreshortening)
Perspective Projection

- project all geometry
  - through common center of projection (eye point)
  - onto an image plane
Perspective Projection

how tall should this bunny be?

center of projection (eye point)

projection plane

how tall should this bunny be?
Basic Perspective Projection

similar triangles

\[ \frac{y'}{d} = \frac{y}{z} \rightarrow y' = \frac{y \cdot d}{z} \]

\[ \frac{x'}{d} = \frac{x}{z} \rightarrow x' = \frac{x \cdot d}{z} \]

- nonuniform foreshortening
- not affine

\[ z' = d \]
Perspective Projection

• desired result for a point \([x, y, z, 1]^{T}\) projected onto the view plane:

\[
\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}
\]

\[
x' = \frac{x \cdot d}{z}, \quad y' = \frac{y \cdot d}{z}, \quad z' = d
\]

• what could a matrix look like to do this?
Simple Perspective Projection Matrix

\[
\begin{bmatrix}
  x & z/d \\
  y & z/d \\
  d & d
\end{bmatrix}
\]
Simple Perspective Projection Matrix

\[
\begin{bmatrix}
\frac{x}{z/d} \\
\frac{y}{z/d} \\
\frac{z/d}{d}
\end{bmatrix}
\] is homogenized version of

\[
\begin{bmatrix}
x \\
y \\
z \\
z/d
\end{bmatrix}
\]

where w = z/d
Simple Perspective Projection Matrix

\[
\begin{bmatrix}
\frac{x}{z/d} \\
\frac{y}{z/d} \\
\frac{z}{d}
\end{bmatrix}
\]

is homogenized version of

where \( w = \frac{z}{d} \)

\[
\begin{bmatrix}
\frac{x}{z/d} \\
\frac{y}{z/d} \\
\frac{z}{d}
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1/d & 0 & 0
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
z/d
\end{bmatrix}
\]
Perspective Projection

- expressible with 4x4 homogeneous matrix
  - use previously untouched bottom row
- perspective projection is irreversible
  - many 3D points can be mapped to same (x, y, d) on the projection plane
  - no way to retrieve the unique z values
Moving COP to Infinity

• as COP moves away, lines approach parallel
• when COP at infinity, orthographic view
Orthographic Camera Projection

- camera’s back plane parallel to lens
- infinite focal length
- no perspective convergence
- just throw away z values

\[
\begin{bmatrix}
    x_p \\
    y_p \\
    z_p \\
    1
\end{bmatrix} =
\begin{bmatrix}
    1 & 0 & 0 & 0 & x \\
    0 & 1 & 0 & 0 & y \\
    0 & 0 & 0 & 0 & z \\
    0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]
Perspective to Orthographic

• transformation of space
  • center of projection moves to infinity
  • view volume transformed
    • from frustum (truncated pyramid) to parallelepiped (box)
View Volumes

- specifies field-of-view, used for clipping
- restricts domain of $z$ stored for visibility test
Canonical View Volumes

- standardized viewing volume representation

**Perspective**

- Front plane
- Back plane
- y or x = +/- z

**Orthographic**

- Front plane
- Back plane
- y or x = 1
- z = -1

Orthogonal parallel
Why Canonical View Volumes?

• permits standardization
  • clipping
    • easier to determine if an arbitrary point is enclosed in volume with canonical view volume vs. clipping to six arbitrary planes
• rendering
  • projection and rasterization algorithms can be reused
Normalized Device Coordinates

• convention
  • viewing frustum mapped to specific parallelepiped
    • Normalized Device Coordinates (NDC)
    • same as clipping coords
  • only objects inside the parallelepiped get rendered
  • which parallelepiped?
    • depends on rendering system
Normalized Device Coordinates

left/right $x = +/- 1$, top/bottom $y = +/- 1$, near/far $z = +/- 1$

Camera coordinates

NDC

Frustum
Understanding Z

- z axis flip changes coord system handedness
- RHS before projection (eye/view coords)
- LHS after projection (clip, norm device coords)
Understanding Z

near, far always positive in OpenGL calls

```c
glOrtho(left, right, bot, top, near, far);
glFrustum(left, right, bot, top, near, far);
glPerspective(fovy, aspect, near, far);
```

perspective view volume

orthographic view volume
Understanding Z

• why near and far plane?
  • near plane:
    • avoid singularity (division by zero, or very small numbers)
  • far plane:
    • store depth in fixed-point representation (integer), thus have to have fixed range of values (0…1)
    • avoid/reduce numerical precision artifacts for distant objects
Orthographic Derivation

- scale, translate, reflect for new coord sys
Orthographic Derivation

- scale, translate, reflect for new coord sys

\[
y' = a \cdot y + b
\]

\[
y = top \rightarrow y' = 1
\]

\[
y = bot \rightarrow y' = -1
\]
Orthographic Derivation

- scale, translate, reflect for new coord sys

\[ y' = a \cdot y + b \]

\[ y = top \rightarrow y' = 1 \]
\[ 1 = a \cdot top + b \]
\[ y = bot \rightarrow y' = -1 \]
\[ -1 = a \cdot bot + b \]

\[ b = 1 - a \cdot top, b = -1 - a \cdot bot \]

\[ 1 - a \cdot top = -1 - a \cdot bot \]
\[ 1 - (-1) = -a \cdot bot - (-a \cdot top) \]

\[ 2 = a(-bot + top) \]
\[ a = \frac{2}{top - bot} \]

\[ 1 = \frac{2}{top - bot} \cdot top + b \]

\[ b = 1 - \frac{2 \cdot top}{top - bot} \]

\[ b = \frac{(top - bot) - 2 \cdot top}{top - bot} \]

\[ b = \frac{-top - bot}{top - bot} \]
Orthographic Derivation

- scale, translate, reflect for new coord sys

\[ y' = a \cdot y + b \]

\[ y = top \rightarrow y' = 1 \]
\[ y = bot \rightarrow y' = -1 \]

\[ a = \frac{2}{top - bot} \]
\[ b = -\frac{top + bot}{top - bot} \]

same idea for right/left, far/near
Orthographic Derivation

- scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\
0 & \frac{2}{\text{top} - \text{bot}} & 0 & -\frac{\text{top} + \text{bot}}{\text{top} - \text{bot}} \\
0 & 0 & -\frac{2}{\text{far} - \text{near}} & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\
0 & 0 & 0 & 1
\end{bmatrix} P$$
Orthographic Derivation

- **scale**, translate, reflect for new coord sys

\[
P' = \begin{bmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\
0 & \frac{2}{\text{top} - \text{bot}} & 0 & -\frac{\text{top} + \text{bot}}{\text{top} - \text{bot}} \\
0 & 0 & \frac{-2}{\text{far} - \text{near}} & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\
0 & 0 & 0 & 1
\end{bmatrix} P
\]
Orthographic Derivation

- scale, **translate**, reflect for new coord sys

\[
P' = \begin{bmatrix}
\frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\
0 & \frac{2}{top - bot} & 0 & -\frac{top + bot}{top - bot} \\
0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Orthographic Derivation

- scale, translate, \textcolor{red}{reflect} for new coord sys

\[
P' = \begin{bmatrix}
\frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\
0 & \frac{2}{\text{top} - \text{bot}} & 0 & -\frac{\text{top} + \text{bot}}{\text{top} - \text{bot}} \\
0 & 0 & -\frac{2}{\text{far} - \text{near}} & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\
0 & 0 & 0 & 1
\end{bmatrix} P
\]
Orthographic OpenGL

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left, right, bot, top, near, far);
```