University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2005

Tamara Munzner

Projections

Week 4, Mon Jan 24

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2005
News

make sure you’ve signed up for demo slot
Review: Display Lists

- reuse block of OpenGL code
  - efficiency
  - multiple instances of same object
  - static objects redrawn often
  - exploit hierarchical structure when possible
- set up list once with glNewList/glEndList
  - call multiple times
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

\[
M_{w2v} = \begin{bmatrix}
    u_x & u_y & u_z & -u \cdot e \\
    v_x & v_y & v_z & -v \cdot e \\
    w_x & w_y & w_z & -w \cdot e \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]
University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2005

Tamara Munzner

Projections
Pinhole Camera

- ingredients
  - box
  - film
  - hole punch
- results
- pictures!

www.kodak.com

www.debevec.org/Pinhole

www.pinhole.org
Pinhole Camera

- theoretical perfect pinhole

![Diagram of a pinhole camera with a pinhole, film plane, and one ray of projection.]
Pinhole Camera

non-zero sized hole

film plane

pinhole

multiple rays
of projection
Pinhole Camera

field of view and focal length
Pinhole Camera

Field of view and focal length
Real Cameras

- pinhole camera has small **aperture** (lens opening)
  - hard to get enough light to expose the film

  ![Real pinhole camera diagram](image)

- lens permits larger apertures
- lens permits changing distance to film plane without actually moving the film plane

  ![Camera diagram](image)

**price to pay:** limited depth of field
Graphics Cameras

- real pinhole camera: image inverted

- computer graphics camera: convenient equivalent
image plane need not be perpendicular to view plane

image plane

eye point

image plane

eye point
Perspective Projection

Our camera must model perspective.
Perspective Projection

Our camera must model perspective.
our camera must model perspective
Perspective Projections

classified by vanishing points

one-point perspective

two-point perspective

three-point 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?
Basic Perspective Projection

similar triangles

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

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

but \( z' = d \)

- nonuniform foreshortening
- not affine
desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

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

$$x' = \frac{x \cdot d}{z} = \frac{x}{z/d}, \quad y' = \frac{y \cdot d}{z} = \frac{y}{z/d}, \quad z = d$$

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

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

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

is homogenized version of

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

where \( w = z/d \)
Perspective Projection Matrix

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

is homogenized version of

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

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

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

\[
\begin{bmatrix}
  x_p \\
  y_p \\
  z_p \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]
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
View Volume

- 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

Frustum

NDC

$x = 1$

$x = -1$

$z = -1$

$z = 1$
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

\[
\text{glOrtho}(\text{left}, \text{right}, \text{bot}, \text{top}, \text{near}, \text{far});
\]

\[
\text{glFrustum}(\text{left}, \text{right}, \text{bot}, \text{top}, \text{near}, \text{far});
\]

\[
\text{glPerspective}(\text{fovy}, \text{aspect}, \text{near}, \text{far});
\]
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

n scale, translate, reflect for new coord sys

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

\[ y = \text{top} \rightarrow y' = 1 \]
\[ y = \text{bot} \rightarrow y' = -1 \]
Orthographic Derivation

scale, translate, reflect for new coord sys

\[ y' = a \cdot y + b \]
\[ y = \text{top} \rightarrow y' = 1 \quad 1 = a \cdot \text{top} + b \]
\[ y = \text{bot} \rightarrow y' = -1 \quad -1 = a \cdot \text{bot} + b \]

\[ b = 1 - a \cdot \text{top}, \quad b = -1 - a \cdot \text{bot} \]
\[ 1 - a \cdot \text{top} = -1 - a \cdot \text{bot} \]
\[ 1 - (-1) = -a \cdot \text{bot} - (-a \cdot \text{top}) \]
\[ 2 = a(-\text{bot} + \text{top}) \]
\[ a = \frac{2}{\text{top} - \text{bot}} \]
\[ 1 = \frac{2}{\text{top} - \text{bot}} \quad \text{top} + b \]
\[ b = 1 - \frac{2 \cdot \text{top}}{\text{top} - \text{bot}} \]
\[ b = \frac{(\text{top} - \text{bot}) - 2 \cdot \text{top}}{\text{top} - \text{bot}} \]
\[ b = \frac{-\text{top} - \text{bot}}{\text{top} - \text{bot}} \]
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

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

scale, translate, reflect for new coord sys
Orthographic Derivation

n 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

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

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