Review: Graphics Cameras
- real pinhole camera: image inverted
- computer graphics camera: convenient equivalent

Review: Basic Perspective Projection
- similar triangles
  \[
  x' = \frac{x}{d} \quad \Rightarrow \quad \frac{x'}{y'} = \frac{x}{y} = \frac{d}{z}
  \]

Review: Orthographic Cameras
- center of projection at infinity
- no perspective convergence
- just throw away z values
  \[
  \begin{bmatrix}
  x \\
  y \\
  z/d \\
  1
  \end{bmatrix}
  \]

Orthographic Derivation
- scale, translate, reflect for new coord sys
  - scale, translate, reflect for new coord sys
    \[
    y' = a \cdot y + b \quad y = \text{top} \Rightarrow y' = 1 \\
    y = \text{bot} \Rightarrow y' = -1
    \]

Orthographic Derivation
- scale, translate, reflect for new coord sys
  \[
  b = \frac{1}{d} \cdot \text{top} - \frac{1}{d} \cdot \text{bot} \\
  a = \frac{2}{d} \cdot \text{bot}
  \]

Orthographic Derivation
- scale, translate, reflect for new coord sys
  \[
  b = \frac{2}{d} \cdot \text{top} \quad b = \frac{2}{d} \cdot \text{bot}
  \]

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

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bottom,top,near,far);

Demo

• Brown applets: viewing techniques
  • parallel/orthographic cameras
  • projection cameras
  • http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/viewing_techniques.html

Asymmetric Frusta

• our formulation allows asymmetry
  • why bother?
  • binocular stereo

Simpler Formulation

• left, right, bottom, top, near, far
  • nonintuitive
  • often overkill
  • look through window center
    • symmetric frustum
  • constraints
    • left = -right, bottom = -top

Field-of-View Formulation

• FOV in one direction + aspect ratio (w/h)
  • determines FOV in other direction
  • also set near, far (reasonably intuitive)

Perspective OpenGL

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(left,right,bottom,top,near,far);

Demo: Frustum vs. FOV

• Nate Robins tutorial (take 2):
  • http://www.xmission.com/~nate/tutors.html

Asymmetric Frusta

• our formulation allows asymmetry
  • why bother? binocular stereo
  • view vector not perpendicular to view plane

Perspective Normalization

• perspective viewing frustum transformed to cube
  • orthographic rendering of cube produces same image as perspective rendering of original

Predistortion

• Tuebingen applets from Frank Hanisch
  • http://www.gis.uni-tuebingen.de/projects/gris/dev/index.html#Transformationen

Demo

• Nate Robins tutorial (take 2):
  • http://www.xmission.com/~nate/tutors.html

Projective Rendering Pipeline

object O2W world WCS W2V viewing V2C
  modeling transformation viewing transformation
  projection transformation clipping C2N

OCS - object/model coordinate system
WCS - world coordinate system
VC2 - viewing/camera/eye coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device/display/screen coordinate system

Projective Rendering Pipeline

object O2W world WCS W2V viewing V2C
  modeling transformation viewing transformation
  projection transformation clipping C2N

OCS - object/model coordinate system
WCS - world coordinate system
VC2 - viewing/camera/eye coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device/display/screen coordinate system

Separate Warp From Homogenization

• warp requires only standard matrix multiply
  • distort such that orthographic projection of distorted objects is desired persp projection
  • w is changed
  • clip after warp, before divide
  • division by w: homogenization

Perspective Normalization

• perspective viewing frustum transformed to cube
  • orthographic rendering of cube produces same image as perspective rendering of original

Separate Warp From Homogenization

• warp requires only standard matrix multiply
  • distort such that orthographic projection of distorted objects is desired persp projection
  • w is changed
  • clip after warp, before divide
  • division by w: homogenization

Projective Rendering Pipeline

object O2W world WCS W2V viewing V2C
  modeling transformation viewing transformation
  projection transformation clipping C2N

OCS - object/model coordinate system
WCS - world coordinate system
VC2 - viewing/camera/eye coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device/display/screen coordinate system

Predistortion

• Tuebingen applets from Frank Hanisch
  • http://www.gis.uni-tuebingen.de/projects/gris/dev/index.html#Transformationen

Asymmetric Frusta

• our formulation allows asymmetry
  • why bother?

Field-of-View Formulation

• FOV in one direction + aspect ratio (w/h)
  • determines FOV in other direction
  • also set near, far (reasonably intuitive)
Coordinate Systems

- viewing (4-space, W=1)
- clipping (4-space parallelepiped, with COP moved backwards to infinity)
- normalized device (3-space parallelepiped)

Projection Matrix

- divide by w
- scale & translate

Perspective Rendering

Projective Rendering Pipeline

- OCS - object/model coordinate system
- WCS - world coordinate system
- VCS - viewing/camera/eye coordinate system
- C2N - clipping coordinate system
- NDCS - normalized device coordinate system
- DCS - device/display/screen coordinate system

NDC to Device Transformation

- map from NDC to pixel coordinates on display
- NDC range is x = -1...1, y = -1...1, z = ... 500

Perspective To NDCS Derivation

- simple example earlier:

Proj. Example

- viewing/camera/eye
- clipping
- perspective divide

Perspective Example

- tracks in VCS:
  - left x=1, y=1
  - right x=1, y=1
- view volume:
  - left x=1, right =1
  - top y=1, near =1, far =4

Perspective Derivation

- complete: shear, scale, projection-normalization

Brown applets: viewing techniques

- parallel/orthographic cameras
- projection cameras

http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/viewing_techniques.html