Perspective Projection (cont.)

Wolfgang Heidrich

Summer Internship

**Wanted:**
- Undergraduate student for summer research on graphics (mostly 2D imaging, digital photography)
- NSERC undergraduate research fellowship

**Prerequisites:**
- Strong programming skills, ideally C++
- Not afraid to learn new math & algorithms

**Interested?**
- Talk to me after lecture, or send email…
**Course News**

**Assignment 1**
- Due February 2

**Homework 1**
- Discussed in labs this week

**Homework 2**
- Exercise problems for perspective
- Discussed in labs next week

**Quiz 1**
- One week from today (Wed, Jan 28)

---

**Course News (cont.)**

**Reading list**
- Previously published chapters numbers were from an old book version...

**Reading for Quiz (new book version):**
- Math prereq: Chapter 2.1-2.4, 4
- Intro: Chapter 1
- Affine transformations: Ch. 6 (was: Ch. 5, old book)
- Perspective: Ch 7 (was: Ch. 6, old book)
  - *Also reading for this week*...
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scanning Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing

Projective Rendering Pipeline

object
OCS → O2W
world
WCS → W2V
viewing
VCS
proportion
V2C

modeling transformation
W2V
viewing transformation
projection transformation
C2N
perspective divide
N2D
viewport transformation

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

In computer graphics:

- Image plane is conceptually in front of the center of projection
- Perspective transformations belong to a class of operations that are called projective transformations
- Linear and affine transformations also belong to this class
- All projective transformations can be expressed as 4x4 matrix operations

Perspective Projection

Synopsis:

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

**Example:**
- Assume image plane at $z=-1$
- A point $[x, y, z, 1]^T$ projects to $[-x/z, -y/z, -z/z, 1]^T = [x, y, z, -z]^T$

**Perspective Projection**

**Analysis:**
- This is a special case of a general family of transformations called *projective transformations*.
- These can be expressed as 4x4 homogeneous matrices!
  - **E.g. in the example:**
    $$
    T \begin{pmatrix}
    x \\ y \\ z \\ 1
    \end{pmatrix} = \begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & -1 & 0
    \end{pmatrix} \cdot \begin{pmatrix}
    x \\ y \\ z \\ 1
    \end{pmatrix} = \begin{pmatrix}
    -x/z \\ -y/z \\ -z \\ 1
    \end{pmatrix}
    $$

Wolfgang Heidrich
Demos

*Tuebingen applets from Frank Hanisch*

(this is the English version)

---

**Projective Rendering Pipeline**

<table>
<thead>
<tr>
<th>Object</th>
<th>World</th>
<th>Viewing</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCS</td>
<td>WCS</td>
<td>VCS</td>
</tr>
<tr>
<td>modeling transformation</td>
<td>viewing transformation</td>
<td>projection transformation</td>
</tr>
</tbody>
</table>

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

- C2N: perspective divide
- N2D: viewport transformation
- Clipping DCS

Wolfgang Heidrich
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Geomery Processing

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing

Projective Transformations

*Transformation of space:*
- Center of projection moves to infinity
- Viewing frustum is transformed into a parallelepiped
**Projective Transformations**

**Convention:**
- Viewing frustum is mapped to a specific parallelepiped
  - *Normalized Device Coordinates (NDC)*
- Only objects inside the parallelepiped get rendered
- Which parallelepiped is used depends on the rendering system

**OpenGL:**
- Left and right image boundary are mapped to $x=-1$ and $x=+1$
- Top and bottom are mapped to $y=-1$ and $y=+1$
- Near and far plane are mapped to $z$ and $z$
Projective Transformations

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

Projective Transformations

Determining the matrix representation

- Need to observe 5 points in general position, e.g.
  - \([\text{left},0,0,1]^T \rightarrow [1,0,0,1]^T\)
  - \([0,\text{top},0,1]^T \rightarrow [0,1,0,1]^T\)
  - \([0,0,-f,1]^T \rightarrow [0,0,1,1]^T\)
  - \([0,0,-n,1]^T \rightarrow [0,0,0,1]^T\)
  - \([\text{left}*f/n,\text{top}*f/n,-f,1]^T \rightarrow [1,1,1,1]^T\)
- Solve resulting equation system to obtain matrix
**Perspective Derivation**

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix}
=egin{bmatrix}
  E & 0 & A & 0 \\
  0 & F & B & 0 \\
  0 & 0 & C & D \\
  0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]

- \(x' = Ex + Az\)
- \(x = \text{left} \rightarrow x'/w' = 1\)
- \(x = \text{right} \rightarrow x'/w' = -1\)
- \(y' = Fy + Bz\)
- \(y = \text{top} \rightarrow y'/w' = 1\)
- \(y = \text{bottom} \rightarrow y'/w' = -1\)
- \(z' = Cz + D\)
- \(z = \text{near} \rightarrow z'/w' = 1\)
- \(z = \text{far} \rightarrow z'/w' = -1\)

\[
y' = Fy + Bz, \quad \frac{y'}{w'} = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{-z}, \quad 1 = F \frac{y}{-z} + B \frac{z}{-z}, \quad 1 = F \frac{y}{-z} - B, \quad 1 = F \frac{\text{top}}{-(-\text{near})} - B, \quad 1 = F \frac{\text{top}}{\text{near}} - B
\]

**Similarly for other 5 planes**

**6 planes, 6 unknowns**

\[
\begin{bmatrix}
  2n & 0 & r + l & 0 \\
  r - l & 2n & t + b & 0 \\
  0 & t - b & -(f + n) & -2fn \\
  0 & 0 & f - n & f - n \\
  0 & 0 & -1 & 0
\end{bmatrix}
\]
**Perspective Example**

\[
\begin{bmatrix}
\frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\
0 & \frac{2n}{t-b} & \frac{t-b}{t-b} & 0 \\
0 & 0 & \frac{-2}{f+n} & \frac{-2}{f+n} \\
0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -\frac{5}{3} & -\frac{8}{3} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

**Projective Transformations**

**Asymmetric Viewing Frusta**

Frustum

- left = -1, right = 1
- bot = -1, top = 1
- near = 1, far = 4
**Projective Transformations**

*Alternative specification of symmetric frusta*

- Field-of-view (fov) $\alpha$
- Fov/2
- Field-of-view in y-direction (fovy) + aspect ratio

![Frustum Diagram]

**Perspective Matrices in OpenGL**

*Perspective Matrices:*

- `glFrustum( left, right, bottom, top, near, far )`
  - Specifies perspective transform (near, far are always positive)

*Convenience Function:*

- `gluPerspective( fovy, aspect, near, far )`
  - Another way to do perspective
Projective Transformations

**Properties:**

- All transformations that can be expressed as homogeneous 4x4 matrices (in 3D)
- 16 matrix entries, but multiples of the same matrix all describe the same transformation
  - 15 degrees of freedom
  - The mapping of 5 points uniquely determines the transformation

---

Projective Transformations

**Properties**

- Lines are 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)*
Orthographic Camera Projection

- Camera's back plane parallel to lens
- Infinite focal length
- No perspective convergence

- Just throw away z values

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

Projection Taxonomy

- planar projections
- perspective: 1, 2, 3-point
- parallel
- oblique
- orthographic
- cabinet
- cavalier

axonometric:
- isometric
- dimetric
- trimetric

http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20

Wolfgang Heidrich
Perspective Projections
classified by vanishing points

one-point perspective

two-point perspective

three-point perspective

Axonometric Projections

- projectors perpendicular to image plane

3 Equal axes 2 Equal axes 0 Equal axes
3 Equal angles 2 Equal angles 0 Equal angles

A. ISOMETRIC  B. DIMETRIC  C. TRIMETRIC

http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20
**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
Projective Rendering Pipeline

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

O2W: modeling transformation
W2V: viewing transformation
V2C: projection transformation

C2N: perspective divide
N2D: viewport transformation
clipping CCS
normalized device NDCS
device DCS

Window-To-Viewport Transformation

Generate pixel coordinates
- Map $x$, $y$ from range $-1\ldots 1$ (normalized device coordinates) to pixel coordinates on the screen
- Map $z$ from $-1\ldots 1$ to $0\ldots 1$ (used later for visibility)
- Involves 2D scaling and translation
Coming Up:

**Friday:**
- Transformations of planes and normals

**Friday/Next Week**
- Lighting/shading

*Don’t forget the quiz...!*