Foundations of Computer Graphics
Online Lecture 5: Viewing
Orthographic Projection
Ravi Ramamoorthi

Motivation
- We have seen transforms (between coord systems)
- But all that is in 3D
- We still need to make a 2D picture
- Project 3D to 2D. How do we do this?
- This lecture is about viewing transformations

Demo (Projection Tutorial)
- Nate Robbins OpenGL tutors
- Projection tutorial
- Download others

What we’ve seen so far
- Transforms (translation, rotation, scale) as 4x4 homogeneous matrices
- Last row always 0 0 0 1. Last w component always 1
- For viewing (perspective), we will use that last row and w component no longer 1 (must divide by it)

Outline
- Orthographic projection (simpler)
- Perspective projection, basic idea
- Derivation of gluPerspective (handout: glFrustum)
- Brief discussion of nonlinear mapping in z

Projections
- To lower dimensional space (here 3D -> 2D)
- Preserve straight lines
- Trivial example: Drop one coordinate (Orthographic)
**Orthographic Projection**

- Characteristic: Parallel lines remain parallel
- Simplest form: project onto x-y plane, drop z coordinate
- Useful for technical drawings etc.

**In general**

- We have a cuboid that we want to map to the normalized or square cube from [-1, +1] in all axes
- We have parameters of cuboid (l, r; t, b; n, f)

**Orthographic Perspective**

In general

- We have a cuboid that we want to map to the normalized or square cube from [-1, +1] in all axes
- We have parameters of cuboid (l, r; t, b; n, f)

**Orthographic Matrix**

- First center cuboid by translating
- Then scale into unit cube

**Transformation Matrix**

- Looking down –z, t and n are negative (n > f)
- OpenGL convention: positive n, f, negate internally

**Transformation Matrix**

\[
M = \begin{pmatrix}
\frac{2}{r-l} & 0 & 0 & 0 \\
0 & \frac{2}{t-b} & 0 & 0 \\
0 & 0 & \frac{2}{f-n} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 & -\frac{r+l}{2} \\
0 & 1 & 0 & -\frac{t+b}{2} \\
0 & 0 & 1 & -\frac{f+n}{2} \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Foundations of Computer Graphics

Online Lecture 5: Viewing

Perspective Projection

Ravi Ramamoorthi

Outline

- Orthographic projection (simpler)
- Perspective projection, basic idea
- Derivation of gluPerspective (handout: glFrustum)
- Brief discussion of nonlinear mapping in z

Perspective Projection

- Most common computer graphics, art, visual system
- Further objects are smaller (size, inverse distance)
- Parallel lines not parallel; converge to single point

Overhead View of Our Screen

Looks like we’ve got some nice similar triangles here?

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

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

In Matrices

- Note negation of z coord (focal plane \(-d\))
- (Only) last row affected (no longer 0 0 0 1)
- w coord will no longer = 1. Must divide at end

\[
P = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{d} & 0
\end{pmatrix}
\]
Foundations of Computer Graphics

Online Lecture 5: Viewing

Derivation of gluPerspective

Ravi Ramamoorthi

Outline

- Orthographic projection (simpler)
- Perspective projection, basic idea
- Derivation of gluPerspective (handout: glFrustum)
- Brief discussion of nonlinear mapping in z

Remember projection tutorial

Viewing Frustum

```latex
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{d} & 0 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix} = \begin{bmatrix}
- \frac{d^* x}{z} \\
- \frac{d^* y}{z} \\
- z \\
- d \\
1 \\
\end{bmatrix}
```
### Screen (Projection Plane)

Field of view (fovy)

Aspect ratio = width / height

### gluPerspective

- gluPerspective(fovy, aspect, zNear > 0, zFar > 0)
- Fovy, aspect control fov in x, y directions
- zNear, zFar control viewing frustum

### In Matrices

- Simplest form:
  \[
  P = \begin{pmatrix}
  \frac{1}{\text{aspect}} & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & \frac{1}{d} & 0 \\
  \end{pmatrix}
  \]
- Aspect ratio taken into account
- Homogeneous, simpler to multiply through by d
- Must map z vals based on near, far planes (not yet)

### Overhead View of Our Screen

\[ \theta = ? \quad d = ? \]

\[ \theta = \frac{\text{fovy}}{2} \quad d = \cot \theta \]

### In Matrices

\[
P = \begin{pmatrix}
\frac{1}{\text{aspect}} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & \frac{1}{d} & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
d \\
0 \\
0 \\
0 \\
\end{pmatrix}
\]

- A and B selected to map n and f to -1, +1 respectively
Z mapping derivation

\[
\begin{pmatrix}
A & B \\
-1 & 0
\end{pmatrix}
\begin{pmatrix} z \\ 1 \end{pmatrix} = ?
\]

Simultaneous equations?

\[-A + \frac{B}{n} = -1 \]
\[-A + \frac{B}{f} = +1 \]

\[
\begin{pmatrix}
A & B \\
-1 & 0
\end{pmatrix}
\begin{pmatrix} z \\ 1 \end{pmatrix} = ?
\]

Simultaneous equations?

\[-A + \frac{B}{n} = -1 \]
\[-A + \frac{B}{f} = +1 \]

\[
Az + B - z
\]
\[= -A - \frac{B}{z} \]

Outline

- Orthographic projection (simpler)
- Perspective projection, basic idea
- Derivation of gluPerspective (handout: glFrustum)
- Brief discussion of nonlinear mapping in z

Mapping of Z is nonlinear

\[
\begin{pmatrix}
Az + B \\
- z
\end{pmatrix} = -A - \frac{B}{z}
\]

- Many mappings proposed: all have nonlinearities
- Advantage: handles range of depths (10cm – 100m)
- Disadvantage: depth resolution not uniform
  - More close to near plane, less further away
- Common mistake: set near = 0, far = infty. Don’t do this. Can’t set near = 0; lose depth resolution.
Summary: The Whole Viewing Pipeline

Model coordinates → Model transformation

World coordinates → Camera Transformation (gluLookAt)

Eye coordinates → Perspective Transformation (gluPerspective)

Screen coordinates → Viewport transformation

Window coordinates → Raster transformation

Device coordinates

Slide courtesy Greg Humphreys