Computer Graphics

Lecture 17

Projection Transformations

Perspective Projection
Announcements

• HW2 is out - a couple more problems may be added no later than Wednesday night.
Viewing Transformations: Overview

A standard sequence of transforms to go from object (model) space to screen (image) space.
Last time: Orthographic Camera

• Rays were already parallel to the z axis, so we only had to fiddle with scales.

• Introduced near and far clipping planes
  • Excuse: throw away stuff behind the camera and too far away
  • Real reason: limit the range of possible depths (we'll need this later)
Orthographic Projection

- The result of our hard work:

\[
M_{\text{orth}} = \begin{bmatrix}
\frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\
0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\
0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Camera Coordinates

Normalized Device Coordinates
Perspective Projection

- In a perspective camera, we have to warp space in a more dramatic way.
Perspective Projection

• In a perspective camera, we have to warp space in a more dramatic way.

• Demo: https://www.cs.cornell.edu/courses/cs4620/2019fa/demos/view_explore/view_explore.html
Perspective Projection

- In a perspective camera, we have to warp space in a more dramatic way.
- Demo: https://www.cs.cornell.edu/courses/cs4620/2019fa/demos/view_explore/view_explore.html
- Recall: linear transformations preserve parallelism.

We don't have the tools for the job!
Perspective Projection

Exercise:
Find $y_s$, the y coordinate of the point where $(x, y, z)$ projects onto the viewport.
Homogeneous coordinates revisited

• Perspective requires division
  – that is not part of affine transformations
  – in affine, parallel lines stay parallel
    • therefore not vanishing point
    • therefore no rays converging on viewpoint

• “True” purpose of homogeneous coords: projection
Homogeneous coordinates revisited

- Introduced $w = 1$ coordinate as a placeholder
  \[
  \begin{bmatrix}
  x \\
  y \\
  z \\
  1
  \end{bmatrix} \rightarrow
  \begin{bmatrix}
  x \\
  y \\
  z \\
  1
  \end{bmatrix}
  \]
  - used as a convenience for unifying translation with linear

- Can also allow arbitrary $w$
  \[
  \begin{bmatrix}
  x \\
  y \\
  z \\
  1
  \end{bmatrix} \sim
  \begin{bmatrix}
  wx \\
  wy \\
  wz \\
  w
  \end{bmatrix}
  \]
Implications of $w$

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix} \sim \begin{bmatrix}
  wx \\
  wy \\
  wz \\
  w
\end{bmatrix}
\]

- All scalar multiples of a 4-vector are equivalent
- When $w$ is not zero, can divide by $w$
  - therefore these points represent “normal” affine points
- When $w$ is zero, it’s a point at infinity, a.k.a. a direction
  - this is the point where parallel lines intersect
  - can also think of it as the vanishing point
- Digression on projective space
Perspective projection

To implement perspective, just move $z$ to $w$:

$$
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
-dx/z \\
-dy/z \\
1
\end{bmatrix} \sim \begin{bmatrix}
dx \\
dy \\
-z
\end{bmatrix} = \begin{bmatrix}
d & 0 & 0 & 0 \\
0 & d & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
$$
What can projective transformations do?

• Map a quadrilateral to another quadrilateral.

• https://iis.uibk.ac.at/public/piater/courses/demos/homography/homography.xhtml
What can projective transformations do?

• Map a quadrilateral to another quadrilateral.

• [https://iis.uibk.ac.at/public/piater/courses/demos/homography/homography.xhtml](https://iis.uibk.ac.at/public/piater/courses/demos/homography/homography.xhtml)

• Aside: line segments still map to line segments, so we can still do wireframe rendering.
View volume: perspective
View volume: perspective (clipped)
Carrying depth through perspective

- Perspective has a varying denominator—can’t preserve depth!
- Compromise: preserve depth on near and far planes

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{bmatrix}
\sim
\begin{bmatrix}
  \tilde{x} \\
  \tilde{y} \\
  \tilde{z} \\
  -z
\end{bmatrix}
= \begin{bmatrix}
  d & 0 & 0 & 0 \\
  0 & d & 0 & 0 \\
  0 & 0 & a & b \\
  0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

- that is, choose \( a \) and \( b \) so that \( z'(n) = n \) and \( z'(f) = f \).

\[
\tilde{z}(z) = az + b
\]

\[
\frac{\tilde{z}}{-\tilde{z}} = \frac{az + b}{-\tilde{z}}
\]

want \( z'(n) = n \) and \( z'(f) = f \)

result: \( a = -(n + f) \) and \( b = nf \) (try it)
Carrying depth through perspective

\[
\begin{bmatrix}
    x' \\
    y' \\
    z' \\
    1
\end{bmatrix} \sim \begin{bmatrix}
    \tilde{x} \\
    \tilde{y} \\
    \tilde{z} \\
    -z
\end{bmatrix} = \begin{bmatrix}
    d & 0 & 0 & 0 \\
    0 & d & 0 & 0 \\
    0 & 0 & a & b \\
    0 & 0 & -1 & 0
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\]

\[a = -(n + f) \text{ and } b = nf\]

Example: 
n=1, f=10

\[
\text{Input interpretation:}
\]

\[
\text{Plot: } 1 + 10 - \frac{10}{z} \quad z = 1 \text{ to } 10
\]
Carrying depth through perspective

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix}
\sim
\begin{bmatrix}
\tilde{x} \\
\tilde{y} \\
\tilde{z} \\
-\tilde{z}
\end{bmatrix}
= \begin{bmatrix}
d & 0 & 0 & 0 \\
0 & d & 0 & 0 \\
0 & 0 & a & b \\
0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

\[a = -(n + f)\text{ and } b = nf\]

Example:
\[n=1, f=10\]
Carrying depth through perspective

- Perspective has a varying denominator—can’t preserve depth!
- Compromise: preserve depth on near and far planes

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{bmatrix}
\sim
\begin{bmatrix}
  \tilde{x} \\
  \tilde{y} \\
  \tilde{z} \\
  -z
\end{bmatrix}
= \begin{bmatrix}
  d & 0 & 0 & 0 \\
  0 & d & 0 & 0 \\
  0 & 0 & a & b \\
  0 & 0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

- that is, choose a and b so that \( z'(n) = n \) and \( z'(f) = f \).
Official perspective matrix

- Use near plane distance as the projection distance
  - i.e., \( d = -n \)

- Scale by \(-1\) to have fewer minus signs
  - scaling the matrix does not change the projective transformation

\[
P = \begin{bmatrix}
n & 0 & 0 & 0 \\
0 & n & 0 & 0 \\
0 & 0 & n + f & -fn \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Perspective projection matrix

- Product of perspective matrix with orth. projection matrix

\[
\mathbf{M}_{\text{per}} = \mathbf{M}_{\text{orth}} \mathbf{P}
\]

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

\[
\begin{bmatrix}
\frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\
0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\
0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
Perspective transformation chain

- Transform into world coords (modeling transform, $M_m$)
- Transform into eye coords (camera xf., $M_{\text{cam}} = F_c^{-1}$)
- Perspective matrix, $P$
- Orthographic projection, $M_{\text{orth}}$
- Viewport transform, $M_{\text{vp}}$

\[
\mathbf{p}_s = M_{\text{vp}} M_{\text{orth}} P M_{\text{cam}} M_m \mathbf{p}_o
\]

\[
\begin{bmatrix}
\frac{x_s}{2} \\
\frac{y_s}{2} \\
\frac{z_c}{2}
\end{bmatrix} = \begin{bmatrix}
\frac{n_x - 1}{2} \\
\frac{n_y - 1}{2}
\end{bmatrix} \begin{bmatrix}
\frac{\frac{2}{r-l}}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\
0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b}
\end{bmatrix} \begin{bmatrix}
\frac{2}{n-f} & 0 & 0 & \frac{n+f}{n-f} \\
0 & \frac{2}{n-f} & 0 & \frac{n+f}{n-f}
\end{bmatrix} \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & n & 0 \\
0 & 0 & n+f & -fn
\end{bmatrix} \begin{bmatrix}
x_o \\
y_o \\
z_o
\end{bmatrix}
\]

© 2014 Steve Marschner • 24