CS4620/5620: Lecture 8

Viewing

Announcements

• This week
  • Viewing Transforms (orthographic)
  • Open GL Introduction (W/F and office hours after class in practicum slot. Not required)
  • Next week (perspective and the graphics pipeline)

• PA 1: will be released Wed
  – Modeling and Transforms, OpenGL
  – PA 2: Shading and Texturing
Plane projection in drawing:
hardware pipeline rendering

Viewing (forward)

• Forward approach
  – start from a point in 3D
  – compute its projection into the image

• Central tool is matrix transformations
  – combines seamlessly with coordinate transformations used to position camera and model
  – ultimate goal: single matrix operation to map any 3D point to its correct screen location
Mathematics of projection

- Always work in eye coords
  - assume eye point at 0 and plane perpendicular to z

- Orthographic case
  - a simple projection: just toss out z \((distance)\)

- Perspective case: scale diminishes with z \((distance)\)
  - increases with \(d\)

Pipeline of transformations

- Standard sequence of transforms

Orthographic transformation chain

- Start with coordinates in object’s local coordinates
- Transform into world coords (modeling transform, $M_m$)
- Transform into eye coords (camera xfm., $M_{cam}$)
- Orthographic projection, $M_{orth}$
- Viewport transform, $M_{vp}$

$$p_s = M_{vp}M_{orth}M_{cam}M_mp_o$$
**Viewing transformation**

The camera matrix rewrites all coordinates in eye space.

**Camera and modeling matrices**

- Mvp and Morth are in eye coordinates
  - before we do anything we need to transform into that space
- Transform from world (canonical) to eye space is traditionally called the *viewing matrix*
  - it is the canonical-to-frame matrix for the camera frame
  - that is, $F_c^{-1}$
- Remember that geometry would originally have been in the object’s local coordinates; transform into world coordinates is called the *modeling matrix*, $M_m$
- Note some systems (e.g. OpenGL) combine the two into a *modelview matrix* and just skip world coordinates
Orthographic transformation chain

- Start with coordinates in object’s local coordinates
- Transform into world coords (modeling transform, $M_m$)
- Transform into eye coords (camera xf., $M_{cam} = F_c^{-1}$)
- Orthographic projection, $M_{orth}$
- Viewport transform, $M_{vp}$

$$p_s = M_{vp}M_{orth}M_{cam}M_mp_o$$

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} n_r/2 & 0 & 0 & n_u/2 \\ 0 & n_u/2 & 0 & n_u/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2/(r-l) & 0 & 0 & -(r+l)/(r-l) \\ 0 & 2/(t-b) & 0 & -(t+b)/(t-b) \\ 0 & 0 & 2/(n-f) & -(n+f)/(n-f) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ e \end{bmatrix}^{-1} M_m \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$
Windowing transforms

- This transformation is worth generalizing
  – take one axis-aligned rectangle or box to another
- a useful, if mundane, piece of a transformation chain

\[
\begin{bmatrix}
1 & 0 & x' \\
0 & 1 & y' \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\frac{x'_h - x'_l}{x_h - x_l} & 0 & 0 \\
0 & \frac{y'_h - y'_l}{y_h - y_l} & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & -x_l \\
0 & 1 & -y_l \\
0 & 0 & 1
\end{bmatrix}
\]
Pipeline of transformations

- Standard sequence of transforms

View volume
orthographic vs. perspective
Orthographic projection

• View volume with bounds
  – minus-z view direction

  – specify view by left, right, top, bottom

Orthographic projection

• View volume with bounds
  – minus-z view direction

  – specify view by left, right, top, bottom
  – also near, far
**Clipping planes**

- In object-order systems we always use at least two *clipping planes* that further constrain the view volume
  - near plane: parallel to view plane; things between it and the viewpoint will not be rendered
  - far plane: also parallel; things behind it will not be rendered

- These planes are:
  - partly to remove unnecessary stuff (e.g. behind the camera)
  - but really to constrain the range of depths
    (we'll see why later)

**Canonical view volume**

- Special case: the *canonical view volume* (cube of size 2)
- *Normalized device coordinates*

\[
NDCS = \{(1, 1, 1), (-1, -1, -1)\}
\]
Parallel projection: orthographic

to implement orthographic, just toss out $z$:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Canonical view volume

- Special case: the canonical view volume (cube of size 2)
- Normalized device coordinates

$$(1, 1, 1)$$

$$(1, -1, 1)$$

$$(1, 1, -1)$$

$$(1, -1, -1)$$

NDCS
Orthographic projection

- View volume with bounds
  - minus-z view direction
  - specify view by left, right, top, bottom
  - also near, far

\[
\begin{bmatrix}
\frac{x'_h - x'_i}{x_h - x_i} & 0 & 0 & \frac{x'_f x_h - x'_f x_i}{x_h - x_i} \\
0 & \frac{y'_h - y'_i}{y_h - y_i} & 0 & \frac{y'_f y_h - y'_f y_i}{y_h - y_i} \\
0 & 0 & \frac{z'_h - z'_i}{z_h - z_i} & \frac{z'_f z_h - z'_f z_i}{z_h - z_i} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

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

• Standard sequence of transforms

Viewing a cube of size 2

• Pixel centers at integer values
Viewing a cube of size 2

- To draw in image, need coordinates in pixel units, though
- Pixel centers at integer values

Viewport transformation

\[
\begin{bmatrix}
x_{\text{screen}} \\ y_{\text{screen}} \\ 1
\end{bmatrix} = \begin{bmatrix}
\frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x_{\text{canonical}} \\ y_{\text{canonical}} \\ 1
\end{bmatrix}
\]
Viewport transformation

- In 3D, carry along \( z \) for the ride
  - one extra row and column

\[
\begin{bmatrix}
\frac{n_x}{2} & 0 & 0 & \frac{n_x - 1}{2} \\
0 & \frac{n_y}{2} & 0 & \frac{n_y - 1}{2} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Orthographic transformation chain

- Start with coordinates in object’s local coordinates
- Transform into world coords (modeling transform, \( M_m \))
- Transform into eye coords (camera xf., \( M_{\text{cam}} = F_c^{-1} \))
- Orthographic projection, \( M_{\text{orth}} \)
- Viewport transform, \( M_{\text{vp}} \)

\[
p_s = M_{\text{vp}} M_{\text{orth}} M_{\text{cam}} M_m p_o
\]
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, $M_{\text{vp}}$

\[
\mathbf{p}_s = M_{\text{vp}}M_{\text{orth}}PM_{\text{cam}}M_{\text{m}}\mathbf{p}_o
\]

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

- Next time OpenGL Introduction
- PA 1 release on Wed