Announcements

- Grading slots on next Thursday
  - Please sign up as a group
- If you don’t have a group yet for PA 1 send mail immediately to cs4620-staff-I
- Debugging your program
  - See things everywhere
  - Use white
  - Use encoded information: ray direction (-1, 1) -> (0, 1)
  - point of intersection
  - normal

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 xf, $M_{cam}$)
- Orthographic projection, $M_{orth}$
- Viewport transform, $M_{vp}$

$$\mathbf{p}_w = M_{vp} M_{orth} M_{cam} M_m \mathbf{p}_o$$

Viewing a cube of size 2

- Start by looking at a restricted case: the canonical view volume

Viewing a cube of size 2

- To draw in image, need coordinates in pixel units, though
- Exactly the opposite of mapping $(ij)$ to $(u,v)$ in ray generation
- Pixel centers at integer values now
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

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
Orthographic projection

- We can implement this by mapping the view volume to the canonical view volume.
- This is just a 3D windowing transformation!

$$\begin{bmatrix}
\frac{1}{x_{min} - x_{max}} & 0 & 0 & 0 \\
0 & \frac{1}{y_{min} - y_{max}} & 0 & 0 \\
0 & 0 & \frac{1}{z_{min} - z_{max}} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

$$M_{orth} = \begin{bmatrix}
\frac{2}{w_{x}} & 0 & 0 & -\frac{w_{x}}{2} \\
0 & \frac{2}{w_{y}} & 0 & -\frac{w_{y}}{2} \\
0 & 0 & \frac{2}{w_{z}} & -\frac{w_{z}}{2} \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Orthographic transformation chain

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

$$p_s = M_{vp}M_{orth}M_{cam}M_m p_o$$

Viewing transformation

the camera matrix rewrites all coordinates in eye space

Camera and modeling matrices

- We worked out all the preceding transforms starting from 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

Canonical to Frame Matrix

$$M_{cam} = \begin{bmatrix}
u & v & w & e \\
0 & 0 & 0 & 1
\end{bmatrix}^{-1} = \begin{bmatrix}x_o & y_o & z_o & 0 \\
x_v & y_v & z_v & 0 \\
x_e & y_e & z_e & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}^{-1} = \begin{bmatrix}1 & 0 & 0 & -x_o \\
0 & 1 & 0 & -y_o \\
0 & 0 & 1 & -z_o \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Orthographic transformation chain

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

$$p_s = M_{vp}M_{orth}M_{cam}M_m p_o$$

$$\begin{bmatrix}x_e \\
y_e \\
z_e \end{bmatrix} = \begin{bmatrix}x_v & 0 & 0 & -x_o \\
0 & y_v & 0 & -y_o \\
0 & 0 & z_v & -z_o \end{bmatrix} \begin{bmatrix}x_o & 0 & 0 & -x_v \\
0 & y_o & 0 & -y_v \\
0 & 0 & z_o & -z_v \end{bmatrix} \begin{bmatrix}u & v & w & e \\
0 & 0 & 0 & 1
\end{bmatrix}^{-1} M_m \begin{bmatrix}x_e \\
y_e \\
z_e \end{bmatrix}$$
Perspective transformation chain

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

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_o}{w} & 0 & 0 & x_o \\ 0 & \frac{y_o}{w} & 0 & y_o \\ 0 & 0 & 1 & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{0}{n} & 0 & 0 & 0 \\ 0 & \frac{n}{n} & 0 & n \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Perspective projection

$$\frac{y'}{d} = \frac{y}{-z}$$
$$y' = -dz/z$$

Homogeneous coordinates revisited

- Perspective requires division
  - that is not part of affine transformations
  - in affine, parallel lines stay parallel
  - therefore no vanishing point
  - therefore no rays converging on viewpoint
- “True” purpose of homogeneous coords: projection

Homogeneous coordinates revisited

- Introduced $w = 1$ coordinate as a placeholder
  - used as a convenience for unifying translation with linear
- Can also allow arbitrary $w$, and make $w$ the denominator

What does $w$ do?

- Linear transforms
  $$x' = ax + by + cz$$
- Affine transforms
  $$x' = ax + by + cz + d$$
- Projective transforms
  $$x' = ax + by + cz + d$$
  $$ex + fy + gz + h$$
  - denominator the same for $y'$ and $z'$
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

Perspective projection

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

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
-dx/z \\
-dy/z \\
1
\end{bmatrix} \sim \begin{bmatrix}
dx \\
dy \\
-dz
\end{bmatrix} = \begin{bmatrix}
d & 0 & 0 & 0 \\
0 & d & 0 & 0 \\
0 & 0 & -1 & 0
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]