Today: Matrix reloaded 2
(aka transformations strike back)
(or Viewing Transformations)

Welcome
Some notes for today’s lecture

- High difficulty level today!
- Will be closely following book chapter 7
  (read it thoroughly to grab the concepts)
- Most of the lecture will be in symbols
Today

• Redo part of last lecture

• Rotation co-ordinate transformation revisited

• Viewing transformation (getting world space $\rightarrow$ screen space)
  – viewport transformation
  – orthographic transformation
  – camera transformation
  – projection transformation
  – “graphics pipeline”: putting everything together
Redo part of last lecture
From the previous lecture

- Point, or active transformations
  - translation
  - projection \{ \text{redo} \}
  - reflection
  - scaling
  - shearing
  - rotation
  - linear transformation properties
  - combining transformations (and transformation back!)

- Co-ordinate, or passive transformations
Translation: a matrix operation

- We translate a point \( P (x, y, z) \) by \((a_x, a_y, a_z)\)
  
i.e., \(x' = x + a_x\), \(y' = y + a_y\), \(z' = z + a_z\)

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix}
= \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & 1 & 0 & a_y \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}; \quad \vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \\
1 \end{bmatrix}
\]
How to think about an extended vector for 2D

- Note: A “real” vector $\vec{v}$, by construction, satisfies $\vec{v} \cdot \hat{f} = 0$

  e.g., the (2+1)D representation of a real vector in 2D is $
  \begin{bmatrix}
  v_x \\
  v_y \\
  0
  \end{bmatrix}$
Projecting and reflecting vectors
Projecting and reflecting vectors

These rules always hold!
Projecting a vector: a matrix operation in (3+1)D

\[ \vec{w}_p = \vec{v} - (\vec{v} \cdot \hat{n}) \hat{n} ; \text{ for } d = 3, \quad \vec{w}_p = \begin{bmatrix} \vec{w}_p \cdot \hat{x} \\ \vec{w}_p \cdot \hat{y} \\ \vec{w}_p \cdot \hat{z} \\ \vec{w}_p \cdot \hat{f} \end{bmatrix} = \begin{bmatrix} \vec{v} \cdot \hat{x} - (\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{x}) \\ \vec{v} \cdot \hat{y} - (\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{y}) \\ \vec{v} \cdot \hat{z} - (\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{z}) \\ \vec{v} \cdot \hat{f} \end{bmatrix} \]

\[ \vec{w}_p = M_p \vec{v} ; \quad M_p = \begin{bmatrix} 1 - n_x^2 & -n_x n_y & -n_x n_z & 0 \\ -n_x n_y & 1 - n_y^2 & -n_y n_z & 0 \\ -n_x n_z & -n_y n_z & 1 - n_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Reflecting a vector: a matrix operation in \((3+1)\)D

\[ \vec{w}_r = \vec{v} - 2(\vec{v} \cdot \hat{n})\hat{n}; \text{ for } d = 3, \vec{w}_r = \begin{bmatrix} \vec{v} \cdot \hat{x} - 2(\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{x}) \\ \vec{v} \cdot \hat{y} - 2(\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{y}) \\ \vec{v} \cdot \hat{z} - 2(\vec{v} \cdot \hat{n})(\hat{n} \cdot \hat{z}) \\ \vec{v} \cdot \hat{f} \end{bmatrix} \]

\[ \vec{w}_r = M_r \vec{v}; \quad M_r = \begin{bmatrix} 1 - 2n_x^2 & -2n_x n_y & -2n_x n_z & 0 \\ -2n_x n_y & 1 - 2n_y^2 & -2n_y n_z & 0 \\ -2n_x n_z & -2n_y n_z & 1 - 2n_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
Projecting and reflecting points
Projecting and reflecting points (on an object)

- Can we use vector projection/reflection formulae for points as well?
  yes, provided care is taken
Projecting and reflecting points (on an object)

- Can we use vector projection/reflection formulae for points as well? 
  yes, provided care is taken

- Why?
  because specifying a vector (arrow) does not specify its starting point;
  although point P \((x, y, z)\) is reached by vector \( \begin{bmatrix} x \\ y \\ z \end{bmatrix} \) from the origin,
  the origin may “move” upon projection/reflection
Projecting points (on an object)

• Specifying a vector (arrow) does not specify its starting point; although point P \((x, y, z)\) is reached by vector \[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\] from the origin, the origin may “move” upon projection/reflection.

\[
\vec{v} - (\vec{v} \cdot \hat{n}) \hat{n} = \text{projected vector}
\]
Projecting and reflecting points (on an object)

- Specifying a vector (arrow) does not specify its starting point; although point $P (x, y, z)$ is reached by vector $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ from the origin, the origin may "move" upon projection/reflection.

- The case for reflection is similar (not shown further).
Rotation co-ordinate transformation revisited
Active rotation in (2+1)D revisited

- Active:
  \[
  \begin{bmatrix}
  x' \\
  y' \\
  1
  \end{bmatrix}
  =
  \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y \\
  1
  \end{bmatrix}
  \]

- Now consider the passive (co-ordinate) rotation
Active rotation in (2+1)D revisited

- Active:
  \[
  \begin{bmatrix}
  x' \\
  y' \\
  1
  \end{bmatrix} =
  \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y \\
  1
  \end{bmatrix}
  \]

- Now consider the passive (co-ordinate) rotation \(Q\).
  \[
  \begin{bmatrix}
  x' \\
  y' \\
  1
  \end{bmatrix} =
  M_{ro}
  \begin{bmatrix}
  x \\
  y \\
  1
  \end{bmatrix};
  M_{ro} =?
  \]
Active rotation in (2+1)D revisited

- Active: \[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Now consider the passive (co-ordinate) rotation \(Q\).

- Question: \[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = M_{ro} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}; M_{ro} = ?
\]

Answer: \[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} = \begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]
Passive rotation in (2+1)D

\[
\begin{bmatrix}
  x' \\
y' \\
1
\end{bmatrix}
= \begin{bmatrix}
  \cos \theta & \sin \theta & 0 \\
  -\sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
y \\
1
\end{bmatrix}
\]

\[
\cos \theta = \hat{x} \cdot \hat{x}' = \hat{y} \cdot \hat{y}', \quad \sin \theta = \hat{y} \cdot \hat{x}' = -\hat{x} \cdot \hat{y}'
\]

then

\[
\begin{bmatrix}
  x' \\
y' \\
1
\end{bmatrix}
= \begin{bmatrix}
  \hat{x}' \cdot \hat{x} & \hat{x}' \cdot \hat{y} & 0 \\
  \hat{y}' \cdot \hat{x} & \hat{y}' \cdot \hat{y} & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
y \\
1
\end{bmatrix}
= \begin{bmatrix}
  x_{x'} & y_{x'} & 0 \\
  x_{y'} & y_{y'} & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
y \\
1
\end{bmatrix}
\]
Passive rotation in (3+1)D

\[
\begin{bmatrix}
    u \\
v \\
w \\
1
\end{bmatrix} =
\begin{bmatrix}
    \hat{u} \cdot \hat{x} & \hat{u} \cdot \hat{y} & \hat{u} \cdot \hat{z} & 0 \\
    \hat{v} \cdot \hat{x} & \hat{v} \cdot \hat{y} & \hat{v} \cdot \hat{z} & 0 \\
    \hat{w} \cdot \hat{x} & \hat{w} \cdot \hat{y} & \hat{w} \cdot \hat{z} & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x_u & y_u & z_u & 0 \\
x_v & y_v & z_v & 0 \\
x_w & y_w & z_w & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
Viewing transformation
What is viewing transformation?
What is viewing transformation?

- Will achieve these (passive!) transformations by concatenating matrices
What is viewing transformation?

- Will achieve these (passive!) transformations by concatenating matrices (and we need to do that in the reverse order of transformations)
Viewport transformation
Viewport transformation

\[ M_{vp} \]
Viewport transformation

- Canonical view space: \((x, y, z) \in [-1, 1]^3\)
- Screen space: \(n_x \times n_y\) pixels
- \(M_{vp}\): transform \([-1, 1]^2 \rightarrow [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]\)
  (only for \(x\) and \(y\), don’t care about \(z\))
Viewport transformation: $M_{vp}$

- $M_{vp}$: transform $[-1, 1]^2 \rightarrow [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]$
  (only for $x$ and $y$, don’t care about $z$)

- Concatenate scaling (diagonal matrix) after translation:

\[
\begin{bmatrix}
\frac{n_x}{2} & 0 & 0 & 0 \\
0 & \frac{n_y}{2} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & \frac{n_x - 1}{2} \\
0 & 1 & 0 & \frac{n_y - 1}{2} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Viewport transformation: $M_{vp}$

- $M_{vp}$: transform $[-1, 1]^2 \rightarrow [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]$ (only for $x$ and $y$, don’t care about $z$)

- Concatenate scaling (diagonal matrix) after translation:

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

- After concatenation, we obtain $M_{vp} =$

\[
\begin{pmatrix}
\frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\
0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Orthographic projection transformation
Orthographic transformation

- $M_{\text{ortho}}$: transform $[l, r] \times [b, t] \times [n, f] \rightarrow [-1, 1]^3$
Orthographic transformation: $M_{\text{ortho}}$

- $M_{\text{ortho}}$: transform $[l, r] \times [b, t] \times [n, f] \rightarrow [-1, 1]^3$

- Concatenate scaling (diagonal matrix) after translation:

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

- After concatenation we obtain $M_{\text{ortho}} =$

$$
\begin{bmatrix}
\frac{2}{r-l} & 0 & 0 & -\frac{r+l}{2} \\
0 & \frac{2}{t-b} & 0 & -\frac{t+b}{2} \\
0 & 0 & \frac{2}{n-f} & -\frac{n+f}{2} \\
0 & 0 & 0 & 1
\end{bmatrix}
$$
Camera transformation
The camera...

- Camera specifications
  - position $\vec{e}$
  - gaze direction $\hat{g}$; $\hat{w} = -\hat{g}$
  - view up vector $\hat{t}$: any vector that symmetrically bisects the viewer’s head into left and right, and points “to the sky”
  - finally, $\hat{u} = (\hat{t} \times \hat{w})/||\hat{t} \times \hat{w}||$, and $\hat{v} = \hat{w} \times \hat{u}$
    $(\hat{u}, \hat{v}, \hat{w})$ forms a right-handed co-ordinate system
World co-ordinates to camera co-ordinates

- $M_{\text{cam}}$: transform $(x, y, z) \rightarrow (u, v, w)$
World co-ordinates to camera co-ordinates

- $M_{\text{cam}}$: transform $(x, y, z) \rightarrow (u, v, w)$

- Concatenate rotation of axes after translation:

\[
\begin{bmatrix}
  x_u & y_u & z_u & 0 \\
  x_v & y_v & z_v & 0 \\
  x_w & y_w & z_w & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 & -x_e \\
  0 & 1 & 0 & -y_e \\
  0 & 0 & 1 & -z_e \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

- After concatenation we obtain $M_{\text{cam}}$
Finally we can put things together

- The “graphics pipeline”:
  
  world space $\rightarrow$ camera space $\rightarrow$ canonical view space $\rightarrow$ screen space

\[
\begin{align*}
M_{\text{cam}} \downarrow & \quad M_{\text{ortho}} \downarrow & \quad M_{\text{vp}} \\
M = M_{\text{vp}} M_{\text{ortho}} M_{\text{cam}}
\end{align*}
\]
Finally we can put things together

- The “graphics pipeline”:
  world space $\rightarrow$ camera space $\rightarrow$ canonical view space $\rightarrow$ screen space

$$M = M_{vp} M_{ortho} M_{cam}; \text{ no, we are not done yet!}$$
Because we have missed the perspective effect
A simpler perspective case

\[ y_s = \frac{yd}{z} \]

(this is the principle we need to implement in 3D)
Camera field of view

![Diagram of camera field of view](image)
Camera field of view
Projection transformation
Projection transformation

- After projection, we want:

\[
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]

\[
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
= \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix}
\]
Projection transformation

- After projection, we want:
  - projection $P:\begin{bmatrix}x \\ y \\ z \\ 1\end{bmatrix}=\begin{bmatrix}nx \\ ny \\ z \\ ? \end{bmatrix}$
  - constraints:
    $z=\{n, f\} \xrightarrow{P} \{n, f\}$
Projection transformation

- After projection, we want:
  \[
  \text{projection } P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ ? \\ 1 \end{bmatrix}
  \]
  - constraints:
    \[
    z = \{n, f\} \xrightarrow{P} \{n, f\}
    \]
- No unique choice for \( P \)
Fixing $P$ for projection

- We want $P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ z \\ ny \\ ? \end{bmatrix}$, with $z = \{n, f\} \xrightarrow{P} \{n, f\}$
- Cannot be achieved by a simple matrix multiplication by $P$
Fixing $P$ for projection: follow the book

- We want $P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ \frac{fn}{z} \end{bmatrix}$, with $z = \{n, f\} \rightarrow \{n, f\}$

- Cannot be achieved by a simple matrix multiplication by $P$

- Choose $P = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$;

$$P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ (n+f)z - fn \\ z \end{bmatrix} \sim \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n+f - \frac{fn}{z} \\ 1 \end{bmatrix}$$
Finally, the graphics pipeline

- World space $\rightarrow$ camera space $\rightarrow$ canonical view space $\rightarrow$ screen space

\[ M = M_{vp} M_{ortho} P M_{cam} \]
Summary

• Redo part of the previous lecture

• Rotation co-ordinate transformation revisited

• Viewing transformation (getting world space \(\rightarrow\) screen space)
  – viewport transformation
  – orthographic transplantation
  – camera transformation
  – projection transformation
  – “graphics pipeline”: putting everything together
Summary

- Redo part of the previous lecture
- Rotation co-ordinate transformation revisited
- Viewing transformation (getting world space $\rightarrow$ screen space)
  - viewport transformation
  - orthographic transformation
  - camera transformation
  - projection transformation
  - “graphics pipeline”: putting everything together
- Good luck with the rest of the second half of the course!
Finally, references...

- Book chapter 6.5: Co-ordinate transformations
- Book chapter 7: Viewing transformations