To Do

- Start doing assignment 1
  - Time is short, but needs only little code [Due Thu Feb 11]
  - Ask questions or clear misunderstandings by next lecture

- Specifics of HW 1
  - Last lecture covered basic material on transformations in 2D
  - Likely need this lecture to understand full 3D transformations
  - Last lecture had full derivation of 3D rotations. You only need final formula (actually not even that, setrot function available)
  - gluLookAt derivation this lecture helps clarifying some ideas
  - Read and post on newsgroup re questions

Outline

- Translation: Homogeneous Coordinates
- Transforming Normals
- Rotations revisited: coordinate frames
- gluLookAt (quickly)

Exposition is slightly different than in the textbook

Homogeneous Coordinates

- Add a fourth homogeneous coordinate (w=1)
- 4x4 matrices very common in graphics, hardware
- Last row always 0 0 0 1 (until next lecture)

\[
\begin{pmatrix}
 x' \\
 y' \\
 z' \\
 w'
\end{pmatrix} = \begin{pmatrix}
 1 & 0 & 0 & 5 \\
 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
 x \\
 y \\
 z \\
 1
\end{pmatrix} = \begin{pmatrix}
 x + 5 \\
 y \\
 z \\
 1
\end{pmatrix}
\]

Translation

- E.g. move x by +5 units, leave y, z unchanged
- We need appropriate matrix. What is it?

\[
\begin{pmatrix}
 x' \\
 y' \\
 z'
\end{pmatrix} = \begin{pmatrix}
 ? \\
 x \\
 y \\
 z
\end{pmatrix} = \begin{pmatrix}
 x + 5 \\
 y \\
 z
\end{pmatrix}
\]

transformation_game.jar

Representation of Points (4-Vectors)

Homogeneous coordinates

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

- Divide by 4th coord (w) to get (inhomogeneous) point
- Multiplication by w > 0, no effect
- Assume w ≥ 0. For w > 0, normal finite point. For w = 0, point at infinity (used for vectors to stop translation)
Advantages of Homogeneous Coords

- Unified framework for translation, viewing, rot...
- Can concatenate any set of transforms to 4x4 matrix
- No division (as for perspective viewing) till end
- Simpler formulas, no special cases
- Standard in graphics software, hardware

General Translation Matrix

\[
T = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
I_3 & T \\
0 & 1
\end{bmatrix}
\]

\[
P' = TP = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} = \begin{bmatrix}
x + T_x \\
y + T_y \\
z + T_z \\
1
\end{bmatrix} = P + T
\]

Combining Translations, Rotations

- Order matters!! TR is not the same as RT (demo)
- General form for rigid body transforms
- We show rotation first, then translation (commonly used to position objects) on next slide. Slide after that works it out the other way

\[
P' = (TR)P = MP = RP + T
\]

\[
M = \begin{bmatrix}
1 & 0 & 0 & T_x \\
0 & 1 & 0 & T_y \\
0 & 0 & 1 & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
R_x & R_y & R_z & 0 \\
R_y & R_x & R_z & 0 \\
R_z & R_y & R_x & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
R_x & R_y & R_z & T_x \\
R_y & R_x & R_z & T_y \\
R_z & R_y & R_x & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
R & T \\
0 & 1
\end{bmatrix}
\]

Outline

- Translation: Homogeneous Coordinates
- Transforming Normals
- Rotations revisited: coordinate frames
- gluLookAt (quickly)

Exposition is slightly different than in the textbook
**Normals**
- Important for many tasks in graphics like lighting
- Do not transform like points e.g. shear
- Algebra tricks to derive correct transform

**Finding Normal Transformation**
\[ t \rightarrow M t \quad n \rightarrow Q n \quad Q = ? \]
\[ n^T t = 0 \]
\[ n^T Q^T M t = 0 \quad \Rightarrow \quad Q^T M = I \]
\[ Q = (M^{-1})^T \]

**Outline**
- Translation: Homogeneous Coordinates
- Transforming Normals
- Rotations revisited: coordinate frames
- `gluLookAt` (quickly)

Section 6.5 of textbook

**Coordinate Frames**
- All of discussion in terms of operating on points
- But can also change coordinate system
- Example, motion means either point moves backward, or coordinate system moves forward

**Coordinate Frames: In general**
- Can differ both origin and orientation (e.g. 2 people)
- One good example: World, camera coord frames (H1)

**Coordinate Frames: Rotations**
\[ R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \]
\[ \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]
Geometric Interpretation 3D Rotations

- Rows of matrix are 3 unit vectors of new coord frame
- Can construct rotation matrix from 3 orthonormal vectors

\[
R_{uvw} = \begin{pmatrix}
  x_u & y_u & z_u \\
  x_v & y_v & z_v \\
  x_w & y_w & z_w
\end{pmatrix} \quad u = x_uX + y_uY + z_uZ
\]

Axis-Angle formula (summary)

\[
(b \rightarrow a)_{ROT} = (I_{3 \times 3} \cos \theta - a a^T \cos \theta)b + (A^* \sin \theta)b
\]

\[
(b \rightarrow a)_{ROT} = (aa^T)b
\]

\[
R(a, \theta) = I_{3 \times 3} \cos \theta + aa^T(1 - \cos \theta) + A^* \sin \theta
\]

Outline

- Translation: Homogeneous Coordinates
- Transforming Normals
- Rotations revisited: coordinate frames
- \textit{gluLookAt} (quickly)

Case Study: Derive \textit{gluLookAt}

Defines camera, fundamental to how we view images
- \textit{gluLookAt}(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)
- Camera is at eye, looking at center, with the up direction being up
- May be important for HW1
- Combines many concepts discussed in lecture
- Core function in OpenGL for later assignments

Steps

- \textit{gluLookAt}(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)
- Camera is at eye, looking at center, with the up direction being up

- First, create a coordinate frame for the camera
- Define a rotation matrix
- Apply appropriate translation for camera (eye) location

Constructing a coordinate frame?

We want to associate \( w \) with \( a \), and \( v \) with \( b \)
- But \( a \) and \( b \) are neither orthogonal nor unit norm
- And we also need to find \( u \)

\[
w = \frac{a}{\|a\|}
\]

\[
u = \frac{b \times w}{\|b \times w\|}
\]

\[
v = w \times u
\]

from lecture 2
Constructing a coordinate frame

- We want to position camera at origin, looking down –Z dirn
- Hence, vector \( \mathbf{a} \) is given by \( \text{eye} - \text{center} \)
- The vector \( \mathbf{b} \) is simply the up vector

\[ \mathbf{w} = \frac{\mathbf{a}}{||\mathbf{a}||}, \quad \mathbf{u} = \frac{\mathbf{b} \times \mathbf{w}}{||\mathbf{b} \times \mathbf{w}||}, \quad \mathbf{v} = \mathbf{w} \times \mathbf{u} \]

Geometric Interpretation 3D Rotations

- Rows of matrix are 3 unit vectors of new coord frame
- Can construct rotation matrix from 3 orthonormal vectors

\[
R_{uvw} = \begin{pmatrix}
x_u & y_u & z_u \\
x_v & y_v & z_v \\
x_w & y_w & z_w
\end{pmatrix}, \quad \mathbf{u} = x_u X + y_u Y + z_u Z
\]

Steps

- gluLookAt(\text{eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz})
- Camera is at eye, looking at center, with the up direction being up
- First, create a coordinate frame for the camera
- Define a rotation matrix
- Apply appropriate translation for camera (eye) location

Translation

- gluLookAt(\text{eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz})
- Camera is at eye, looking at center, with the up direction being up

- Cannot apply translation after rotation
- The translation must come first (to bring camera to origin) before the rotation is applied

Combining Translations, Rotations

\[
P' = (RT)P = MP = R(P + T) = RP + RT
\]

\[
M = \begin{pmatrix}
R_0 & R_1 & R_2 & 0 & 1 & 0 & 0 \\
R_3 & R_4 & R_5 & 0 & 0 & 1 & 0 \\
R_6 & R_7 & R_8 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
R_0 & R_1 & R_2 & 0 & T_x & T_y & T_z \\
R_3 & R_4 & R_5 & 0 & 0 & 1 & 0 \\
R_6 & R_7 & R_8 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]
### gluLookAt final form

\[
\begin{pmatrix}
  x & y & z & 0 \\
  x & y & z & 0 \\
  x & y & z & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  1 & 0 & 0 & -e_x \\
  0 & 1 & 0 & -e_y \\
  0 & 0 & 1 & -e_z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x & y & z & -x e_x - y e_y - z e_z \\
  x & y & z & -x e_x - y e_y - z e_z \\
  x & y & z & -x e_x - y e_y - z e_z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]