Computer Graphics (Fall 2008)
COMS 4160, Lecture 4: Transformations 2
http://www.cs.columbia.edu/~cs4160

To Do
- Start doing assignment 1
  - Time is short, but needs only little code [Due Thu Sep 25, 11:59pm]
  - Ask questions or clear misunderstandings by next lecture
- Specifics of HW 1
  - Last lecture covered basic material on transformations in 2D. You likely need this lecture though to understand full 3D transformations
  - Last lecture had some complicated stuff on 3D rotations. You only need final formula (actually not even that, setrot function available)
  - gluLookAt derivation this lecture should help clarifying some ideas
- Read bulletin board and webpage!!

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

Exposition is slightly different than in the textbook

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

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

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)

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

Representation of Points (4-Vectors)
- Divide by 4th coord (w) to get (inhomogeneous) point
- Multiplication by w > 0, no effect
- Assume w \geq 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, rotation...
- Can concatenate any set of transforms to a 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

**General Form for Rigid Body Transforms**

\[ \begin{bmatrix} R_1 & R_2 & R_3 & 0 \\ 0 & R_1 & R_2 & 0 \\ 0 & 0 & R_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & R_3 & T_x \\ 0 & R_1 & R_2 & T_y \\ 0 & 0 & R_1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' + T_z \\ 1 \end{bmatrix} \]

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

Outline
- Translation: Homogeneous Coordinates
- Transforming Normals
- Rotations revisited: coordinate frames
- \texttt{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
\[ R = \begin{bmatrix} \cos \theta & -\sin \theta & u \\ \sin \theta & \cos \theta & v \\ 0 & 0 & 1 \end{bmatrix} \]
\[ \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \]
Geometric Interpretation 3D Rotations

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

\[
R_{uvw} = \begin{pmatrix}
    x_u & y_u & z_u \\
x_v & y_v & z_v \\
x_w & y_w & z_w
\end{pmatrix}
\]

\[u = x_u X + y_u Y + z_u Z\]

Axis-Angle formula (summary)

\[
(b \setminus a)_{\text{ROT}} = (I_{3x3} \cos \theta - aa^T \cos \theta)b + (A^+ \sin \theta)b
\]

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

\[
R(a, \theta) = I_{3x3} \cos \theta + aa^T (1 - \cos \theta) + A^+ \sin \theta
\]

Case Study: Derive \text{gluLookAt}

Defines camera, fundamental to how we view images

- \text{gluLookAt}(eyes, eye\_x, eye\_y, eye\_z, center\_x, center\_y, center\_z, up\_x, up\_y, up\_z)
- Camera is at eye, looking at center, with the up direction being up
- May be important for HW1
- Combines many concepts discussed in lecture so far
- Core function in OpenGL for later assignments

Outline

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

Not fully covered in textbooks. However, look at sections 6.5 and 7.2.1

We've already covered the key ideas, so we go over it quickly showing how things fit together

Steps

- \text{gluLookAt}(eyes, eye\_x, eye\_y, eye\_z, center\_x, center\_y, center\_z, up\_x, up\_y, up\_z)
- 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

\[ w = a \]
\[ u = b \times w \]
\[ v = w \times u \]

- We want to position camera at origin, looking down \(-Z\) direction.
- Hence, vector \(w\) is given by \(\text{eye} - \text{center}\).
- The vector \(u\) is simply the \(\text{up}\) vector.

Steps

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

Geometric Interpretation 3D Rotations

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

\[ R_{uvw} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix} \]

\[ u = x_u X + y_u Y + z_u Z \]

Steps

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

Translation

- `gluLookAt(\text{eyex}, \text{eyey}, \text{eyez}, \text{centerx}, \text{centery}, \text{centerz}, \text{upx}, \text{upy}, \text{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

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

\[ M = \begin{bmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

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

\[ \begin{bmatrix} R_{33} & R_{32} & R_{31} & 0 \\ R_{23} & R_{22} & R_{21} & 0 \\ R_{13} & R_{12} & R_{11} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
**gluLookAt final form**

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