To Do

- Start doing HW 1
  - Time is short, but needs only little code [Due Mon Jan 30]
  - 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
  - gluLookAt derivation this lecture helps clarifying some ideas
- Read and post on Piazza re questions
- Any remaining issues with edX edge graders, submission of homeworks?

Outline

- Translation: Homogeneous Coordinates
- Combining Transforms: Scene Graphs
- Transforming Normals
- Rotations revisited: coordinate frames
- gluLookAt (quickly)

Translation

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

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

Transformations game demo

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)

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

Representation of Points (4-Vectors)

Homogeneous coordinates

- Divide by 4th coord (w) to get \( P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \) (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, 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

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

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

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
- Demos with applet, homework 1

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

\[
M = \begin{pmatrix}
R_{11} & R_{12} & R_{13} & 0 \\
R_{21} & R_{22} & R_{23} & 0 \\
R_{31} & R_{32} & R_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
R & T \\
0 & 1
\end{pmatrix}
\]

Outline

- Translation: Homogeneous Coordinates
- Combining Transforms: Scene Graphs
- Transforming Normals
- Rotations revisited: coordinate frames
- gluLookAt (quickly)

Slides for this part courtesy Prof. O’Brien
Hierarchical Scene Graph

- `Hierarchical Scene Description`
- `Hierarchical Scene Composition`
- `Draw scene with pre-and-post-order traversal`
- `Apply node, draw children, undo node if applicable`
- `Nodes can carry out any function`
- `Geometry, transforms, groups, color, …`
- `Requires stack to “undo” post children`
- `Transform stacks in OpenGL`
- `Caching and instancing possible`
- `Instances make it a DAG, not strictly a tree`

Drawing a Scene Graph

Example Scene-Graphs

- What is the "Right" Hierarchy for this 18-Wheeler?
- What is the "Right" Hierarchy for this 18-Wheeler?

Outline

- Translation: Homogeneous Coordinates
- Combining Transforms: Scene Graphs
- Transforming Normals
- Rotations revisited: coordinate frames
- `gluLookAt` (quickly)

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

\[
t \rightarrow Mt \\
n \rightarrow Qn \\
Q = ?
\]

\[
n^T t = 0 \\
n^T Q^T Mt = 0 \implies Q^T M = I
\]

\[
Q = (M^{-1})^T
\]
Outline

- Translation: Homogeneous Coordinates
- Combining Transforms: Scene Graphs
- Transforming Normals
- Rotations revisited: coordinate frames
- gluLookAt (quickly)

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

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

\[
R_{uvw} = \begin{bmatrix}
  x_u & y_u & z_u \\
  x_v & y_v & z_v \\
  x_w & y_w & z_w \\
\end{bmatrix} \quad u = x_u X + y_u Y + z_u Z
\]
**Axis-Angle formula (summary)**

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

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

**Case Study: Derive gluLookAt**

Defines camera, fundamental to how we view images
- `gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)`
- Camera is at eye, looking at center, with the up direction being up

![Eye](image)

- May be important for HW1
- Combines many concepts discussed in lecture
- Core function in OpenGL for later assignments

**Steps**

- `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

**Outline**

- Translation: Homogeneous Coordinates
- Combining Transforms: Scene Graphs
- Transforming Normals
- Rotations revisited: coordinate frames
- `gluLookAt (quickly)`

**Constructing a coordinate frame**

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

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

![Eye](image)
Steps

- `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

Geometric Interpretation 3D Rotations

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

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

\[
\begin{align*}
    u &= x_u X + y_u Y + z_u Z \\
    v &= x_v X + y_v Y + z_v Z \\
    w &= x_w X + y_w Y + z_w Z \\
\end{align*}
\]

Steps

- `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

Translation

- `gluLookAt(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

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

\[
M = \begin{bmatrix}
    R_{11} & R_{12} & R_{13} & 0 \\
    R_{21} & R_{22} & R_{23} & 0 \\
    R_{31} & R_{32} & R_{33} & 0 \\
    0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & T_x \\
    0 & 1 & 0 & T_y \\
    0 & 0 & 1 & T_z \\
\end{bmatrix}
\begin{bmatrix}
    R_{3x3} & R_{3x3T_{3x1}} \\
\end{bmatrix}
\]

gluLookAt final form

\[
\begin{bmatrix}
    x_u & y_u & z_u & 0 \\
    x_v & y_v & z_v & 0 \\
    x_w & y_w & z_w & 0 \\
    0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & -e_x \\
    0 & 1 & 0 & -e_y \\
    0 & 0 & 1 & -e_z \\
    0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    -x_u e_x - y_u e_y - z_u e_z \\
    -x_v e_x - y_v e_y - z_v e_z \\
    -x_w e_x - y_w e_y - z_w e_z \\
\end{bmatrix}
\]

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