More on Transformations

COS 426
Agenda

Grab-bag of topics related to transformations:

• General rotations
  ◦ Euler angles
  ◦ Rodrigues’s rotation formula

• Maintaining camera transformations
  ◦ First-person
  ◦ Trackball

• How to transform normals
3D Coordinate Systems

- Right-handed vs. left-handed
3D Coordinate Systems

• **Right-handed** vs. **left-handed**

• Right-hand rule for rotations: positive rotation = counterclockwise rotation about axis
General Rotations

• Recall: set of rotations in 3-D is 3-dimensional
  ○ Rotation group SO(3)
  ○ Non-commutative
  ○ Corresponds to orthonormal $3 \times 3$ matrices with determinant $= +1$

• Need 3 parameters to represent a general rotation (Euler’s rotation theorem)
Euler Angles

• Specify rotation by giving angles of rotation about 3 coordinate axes

• 12 possible conventions for order of axes, but one standard is Z-X-Z
Euler Angles

- Another popular convention: X-Y-Z
- Can be interpreted as yaw, pitch, roll of airplane
Rodrigues’s Formula

- Even more useful: rotate by an arbitrary angle (1 number) about an arbitrary axis (3 numbers, but only 2 degrees of freedom since unit-length)
Rodrigues’s Formula

• An arbitrary point \( p \) may be decomposed into its components along and perpendicular to \( a \)

\[
p = a \left( p \cdot a \right) + \left[ p - a \left( p \cdot a \right) \right]
\]
Rodrigues’s Formula

- Rotating component along \(\mathbf{a}\) leaves it unchanged
- Rotating component perpendicular to \(\mathbf{a}\) (call it \(\mathbf{p}_\perp\)) moves it to \(\mathbf{p}_\perp \cos \theta + (\mathbf{a} \times \mathbf{p}_\perp) \sin \theta\)
Rodrigues’s Formula

- Putting it all together:

\[ R_\mathbf{p} = a (\mathbf{p} \cdot a) + \mathbf{p}_\perp \cos \theta + (a \times \mathbf{p}_\perp) \sin \theta \]

\[ = aa^T \mathbf{p} + (\mathbf{p} - aa^T \mathbf{p}) \cos \theta + (a \times \mathbf{p}) \sin \theta \]

- So,

\[ R = aa^T + (I - aa^T) \cos \theta + [a]_x \sin \theta \]

where \([a]_x\) is the “cross product matrix”

\[ [a]_x = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \]
Rotating One Direction into Another

- Given two directions $d_1$, $d_2$ (unit length), how to find transformation that rotates $d_1$ into $d_2$?
  - There are many such rotations!
  - Choose rotation with minimum angle

- Axis = $d_1 \times d_2$

- Angle = $\cos^{-1}(d_1 \cdot d_2)$

- More stable numerically: $\tan^{-12}(|d_1 \times d_2|, d_1 \cdot d_2)$
Agenda

Grab-bag of topics related to transformations:

• General rotations
  ◦ Euler angles
  ◦ Rodrigues’s rotation formula

• Maintaining camera transformations
  ◦ First-person
  ◦ Trackball

• How to transform normals
Camera Coordinates

Canonical camera coordinate system

- Convention is right-handed (looking down –z axis)
- Convenient for projection, clipping, etc.

- Camera right vector maps to X axis
- Camera up vector maps to Y axis
- Camera back vector maps to Z axis (pointing out of page)
Viewing Transformation

- Mapping from world to camera coordinates
  - Eye position maps to origin
  - Right vector maps to +X axis
  - Up vector maps to +Y axis
  - Back vector maps to +Z axis
Finding the viewing transformation

• We have the camera (in world coordinates)
• We want $T$ taking objects from world to camera

\[ p^c = T \ p^w \]

• Trick: find $T^{-1}$ taking objects in camera to world

\[ p^w = T^{-1} p^c \]

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} =
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]
Finding the Viewing Transformation

- Trick: map from camera coordinates to world
  - Origin maps to eye position
  - Z axis maps to Back vector
  - Y axis maps to Up vector
  - X axis maps to Right vector

\[
\begin{bmatrix}
  x' \\
y' \\
z' \\
w'
\end{bmatrix} =
\begin{bmatrix}
  R_x & U_x & B_x & E_x \\
  R_y & U_y & B_y & E_y \\
  R_z & U_z & B_z & E_z \\
  R_w & U_w & B_w & E_w
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

- This matrix is $T^{-1}$ so we invert it to get $T$ … easy!
Maintaining Viewing Transformation

For first-person camera control, need 2 operations:

- Turn: rotate($\theta$, 0, 1, 0) in local coordinates
- Advance: translate(0, 0, $-v^*\Delta t$) in local coordinates

- Key: transformations act on local, not global coords
- To accomplish: right-multiply by translation, rotation

\[ M_{\text{new}} \leftarrow M_{\text{old}} T_{-v^*\Delta t, z} R_{\theta, y} \]
Maintaining Viewing Transformation

Object manipulation: “trackball” or “arcball” interface

• Map mouse positions to surface of a sphere

• Compute rotation axis, angle

• Apply rotation to global coords: left-multiply

\[
\mathbf{M}_{\text{new}} \leftarrow R_{\theta,a} \mathbf{M}_{\text{old}}
\]
Agenda

Grab-bag of topics related to transformations:

• General rotations
  ◦ Euler angles
  ◦ Rodrigues’s rotation formula

• Maintaining camera transformations
  ◦ First-person
  ◦ Trackball

• How to transform normals
Transforming Normals

Normals do not transform the same way as points!

- Not affected by translation
- Not affected by shear perpendicular to the normal
Transforming Normals

• Key insight: normal remains perpendicular to surface tangent

• Let $t$ be a tangent vector and $n$ be the normal

$$t \cdot n = 0 \quad \text{or} \quad t^T n = 0$$

• If matrix $M$ represents an affine transformation, it transforms $t$ as

$$t \rightarrow M_L t$$

where $M_L$ is the linear part (upper-left $3\times3$) of $M$
Transforming Normals

• So, after transformation, want

\[(M_L t)^T n_{\text{transformed}} = 0\]

• But we know that

\[t^T n = 0\]

\[t^T M_L (M_L^T)^{-1} n = 0\]

\[(M_L t)^T (M_L^T)^{-1} n = 0\]

• So,

\[n_{\text{transformed}} = (M_L^T)^{-1} n\]
Transforming Normals

• Conclusion: normals transformed by inverse transpose of linear part of transformation

• Note that for rotations, inverse = transpose, so inverse transpose = identity
  ◦ normals just rotated
COS 426 Midterm exam

• Thursday, 3/16

• Regular time/place: 3:00-4:20, CS105

• Covers color, image processing, shape representations, but not transformations
  ◦ Also responsible for knowing all required parts of first two programming assignments

• Closed book, no electronics, one page of notes / formulas