Lecture 22 of 41

Animation 2 of 3: Rotations, Quaternions
Dynamics & Kinematics

William H. Hsu
Department of Computing and Information Sciences, KSU

Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:
Today: Chapter 17, esp. §17.1 – 17.2, Eberly
Next class: Chapter 10, 13, §17.3 – 17.5, Eberly

Animation 2 of 3: Rotations, Quaternions
Dynamics & Kinematics

Lecture Outline
• Reading for Last Class: §11.1 – 11.6 Eberly 2e (736), Flash handout
• Reading for Today: §17.1 – 17.2, Eberly 2e
• Reading for Next Class: Chapter 10, 13, §17.3 – 17.5, Eberly 2e
• Previously: Evaluators, Piecewise Polynomial Curves, Bicubic Surfaces
• Last Time: Maya & Animation Preliminaries – Ross Tutorials
  • Maya interface: navigation, menus, tools, primitives
  • Ross tutorials (http://bit.ly/dFpTwq)
• Today: Rotations in Animation
  • Flight dynamics: roll, pitch, yaw
  • Matrix, angles (fixed, Euler, axis), quaternions, exponential maps
  • Dynamics: forward (trajectories, simulation), inverse ( # & ballistics)
  • Kinematics: forward, inverse
• Next Time: Videos Part 4 – Modeling & Simulation

Where We Are

Left to Right

References: Maya Character Rigging

Aaron Ross
Founder, Digital Arts Guild
http://dr-yo.com
http://www.youtube.com/user/DigitalArtsGuild

Jim Lammers
President
Trinity Animation
http://www.trinity3d.com

Larry Neuberger
Associate Professor, Alfred State SUNY College of Technology
Online Instructor, Art Institute of Pittsburgh
http://poorhousefx.com

Acknowledgements:
CGA Rotations, Dynamics & Kinematics

Rick Parent
Professor
Department of Computer Science and Engineering
Ohio State University
http://www.cse.ohio-state.edu/~parent/

David C. Brogan
Visiting Assistant Professor, Computer Science Department, University of Virginia
http://www.cs.virginia.edu/~dbrogan/

Steve Rotenberg
University of California – San Diego
http://graphics.ucsd.edu

Spaces & Transformations

Left-handed v. right handed
Homogeneous coordinates
4x4 transformation matrix (TM)
Concatenating TMs
Basic transformations (TMs)
Display pipeline
Rotations [1]: Orientation

- We have defined "orientation" to mean an object’s instantaneous rotational configuration.
- Think of it as the rotational equivalent of position.

Rotations [2]: Representing Position

- Cartesian coordinates (x, y, z) are an easy and natural means of representing a position in 3D space.
- There are many other alternatives such as polar notation (r, θ, φ) and you can invent others if you want to.

Rotations [3]: Euler’s Theorem

- Euler’s Theorem: Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis.
- Not to be confused with Euler angles, Euler’s formula, Euler integration, Newton-Euler dynamics, inviscid Euler equations, Euler characteristic...
- Leonard Euler (1707-1783)

Rotations [4]: Euler Angles

- This means that we can represent an orientation with 3 numbers.
- A sequence of rotations around principal axes is called an Euler Angle Sequence.
- Assuming we limit ourselves to 3 rotations without successive rotations about the same axis, we could use any of the following 12 sequences:
  - XYZ, XZY, XYX, XZX
  - YXZ, YXZ, YXY, YZY
  - ZXY, ZYX, ZXZ, ZYX

Representing Orientations

Example: fixed angles - rotate around global axes

\[ \mathbf{P} = R_z(\gamma)R_y(\beta)R_x(\alpha)\mathbf{P} \]

Orientation: \((\alpha, \beta, \gamma)\)
Working with Fixed Angles & Rotation Matrices (RMs)

- Orthonormalizing a RM
- Extracting fixed angles from an orientation
- Extracting fixed angles from a RM
- Making a RM from fixed angles
- Making a RM from transformed unit coordinate system (TUCS)

Transformations in Pipeline

- object → world: often rigid transforms
- world → eye: rigid transforms
- perspective matrix: uses 4th component of homo. coords
- perspective divide
- image → screen: 2D map to screen coordinates
- Clipping: procedure that considers view frustum

Error Considerations

Accumulated round-off error - transform data:
- transform world data by delta RM
- update RM by delta RM; apply to object data
- update angle; form RM; apply to object data
- orthonormalization
- rotation matrix: orthogonal, unit-length columns
- iterate update by taking cross product of 2 vectors
- scale to unit length
- considerations of scale
- miles-to-inches can exceed single precision arithmetic

Six Ways to Represent Orientations

Rotation matrix
- Fixed angles: rotate about global coordinate system
- Euler angles: rotate about local coordinate system
- Axis-angle: arbitrary axis and angle
- Quaternions: mathematically handy axis-angle 4-tuple
- Exponential map: 3-tuple version of quaternions

Representing 3 Rotational Degrees of Freedom (DOFs)

- 3x3 Matrix (9 DOFs)
- Rows of matrix define orthogonal axes
- Euler Angles (3 DOFs)
- Rot x + Rot y + Rot z
- Axis-angle (4 DOFs)
- Axis of rotation + Rotation amount
- Quaternion (4 DOFs)
- 4 dimensional complex numbers

Method 1 – Transformation Matrix [1]

4 × 4 Homogeneous TMs

\[
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
\end{bmatrix}
\]
Method 1 – Transformation Matrix [2]:
Translation

\[
\begin{bmatrix}
a & b & c & t_x \\
e & f & g & t_y \\
i & j & k & t_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Method 1 – Transformation Matrix [3]:
Rotation about x, y, z

Rotation about x axis (Roll)

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) & 0 \\
0 & \sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Rotation about y axis (Pitch)

\[
\begin{bmatrix}
\cos(\phi) & 0 & \sin(\phi) & 0 \\
0 & 1 & 0 & 0 \\
-\sin(\phi) & 0 & \cos(\phi) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Rotation about z axis (Yaw)

\[
\begin{bmatrix}
\cos(\gamma) & -\sin(\gamma) & 0 & 0 \\
\sin(\gamma) & \cos(\gamma) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Method 2 – Fixed Angles [1]

\[P' = R_z(\gamma)R_y(\beta)R_x(\alpha)P\]

Fixed order: e.g., x, y, z; also could be y, x, z

Method 2 – Fixed Angles [2]:
Gimbal Lock

Fixed angle: e.g., x, y, z

\[(0 \ 0 \ 0) \quad (0 \ 90 \ 0)\]

Method 2 – Fixed Angles [3]:
Order of Rotations

Fixed order of rotations: x, y, z

What do these epsilon rotations do?

\[(0 \pm \epsilon \ 90 \ 0) \quad (0 \ 90 \pm \epsilon \ 0) \quad (0 \ 90 \ 0 \pm \epsilon)\]

Method 2 – Fixed Angles [4]:
Interpolating Fixed Angles

Interpolating FA representations does not produce intuitive rotation because of gimbal lock

\[(0 \ 90 \ 0) \quad (90 \ 0 \ 90)\]
Method 2 – Fixed Angles [5]: Gimbal Lock Illustrated

- Gimbal Lock: Term Derived from Mechanical Problem in Gimbal
- Gimbal: Mechanism That Supports Compass, Gyroscope

Method 3 – Euler Angles [1]

\[(\alpha \ \beta \ \gamma)\]
Prescribed order: e.g., x, y, z or x, y, z
Rotate around (rotated) local axes

Note: fixed angles are same as Euler angles in reverse order and vice versa

\[(\alpha \ \beta \ \gamma) \rightarrow P' = R_z(\gamma)R_y(\beta)R_x(\alpha)P\]

Method 3 – Euler Angles [2]

\[(\phi, \theta, \psi) = P_nP_r\]

- Rotate \(\phi\), degrees about x-axis
- Rotate \(\theta\), degrees about y-axis
- Rotate \(\psi\), degrees about z-axis

Axis order is not defined
- All are legal
- Pick one

Method 4 – Axis-Angle [1]

Given
- \(r\) – vector in space to rotate
- \(n\) – unit-length axis in space about which to rotate
- \(\theta\) – amount about \(n\) to rotate

Solve
- \(r'\) – rotated vector

Method 4 – Axis-Angle [3]: Axis-Angle to Series of Rotations

- Concatenate the following:
  - Rotate A around z to x-y plane
  - Rotate A' around y to x-axis
  - Rotate theta around x
  - Undo rotation around y-axis
  - Undo rotation around z-axis
Method 4 – Axis-Angle [4]: Axis-Angle to Rotation Matrix

\[ \hat{A} = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 \\ a_3 & a_4 & a_1 & a_2 \\ a_2 & a_1 & a_4 & a_3 \\ a_1 & a_2 & a_3 & a_4 \end{bmatrix} \]

\[ \hat{A} = \begin{bmatrix} 0 & -a_z & a_y & a_x \\ a_z & 0 & -a_x & a_y \\ -a_y & a_x & 0 & -a_z \\ -a_x & -a_y & a_z & 0 \end{bmatrix} \]

\[ \text{Rot}_{\theta} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \hat{A} + \cos(\theta)(I - \hat{A}) + \sin(\theta)\hat{A} \]

Method 5 – Quaternions [1]

\[ \text{Rot}_{\theta} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = [\cos(\theta), \sin(\theta)v] \]

Same as axis-angle, but different form
Still rotate about given axis
Mathematically convenient form

\[ \begin{bmatrix} x \\ y \\ z \end{bmatrix} \]

Note: In this form v is a scaled version of the given axis of rotation, A

Method 5 – Quaternions [2]: Arithmetic

Addition

\[ [s_1 + s_2, v_1 + v_2] = [s_1, v_1] + [s_2, v_2] \]

Multiplication

\[ q_1q_2 = [s_1s_2 - v_1\cdot v_2, s_1v_2 + s_2v_1 + v_1\times v_2] \]

Inner Product

\[ q_1 \cdot q_2 = s_1s_2 + v_1 \cdot v_2 \]

Length

\[ |q| = \sqrt{q \cdot q} \]

Method 5 – Quaternions [3]: Inverse & Normalization

Inverse

\[ q^{-1} = \frac{1}{|q|^2} [s, -v] \]

\[ qq^{-1}q = [1, 0, 0, 0] \]

\[ (pq)^{-1} = q^{-1}p^{-1} \]

Unit quaternion

\[ \hat{q} = \frac{q}{|q|} \]

Method 5 – Quaternions [4]: Representation

Vector

\[ [0, v] \]

Transform

\[ v' = \text{Rot}_q(v) = qvq^{-1} \]

Method 5 – Quaternions [5]: Geometric Operations

\[ \text{Rot}_q(v) = \text{Rot}_q(v) \]

\[ \text{Rot}_q(v) = \text{Rot}_q(v) \]

\[ v' = \text{Rot}_q(v) = qvq^{-1} \]

\[ v' = \text{Rot}_q(v) = q^{-1}(qvq^{-1})q = v \]
Method 5 – Quaternions [6]:
Unit Quaternion Conversions

\[
\text{Rot}(\theta, x, y, z) = \begin{bmatrix}
1 - 2y^2 - 2z^2 & 2x\theta - 2xz & 2x\theta - 2xy \\
2x\theta - 2xz & 1 - 2x^2 - 2z^2 & 2y\theta - 2xz \\
2x\theta - 2xy & 2y\theta - 2xz & 1 - 2x^2 - 2y^2
\end{bmatrix}
\]

Axis-Angle

\[
\begin{align*}
\theta &= 2\cos^{-1}(x) \\
(x, y, z) &= \frac{\mathbf{v}}{|\mathbf{v}|}
\end{align*}
\]

Method 5 – Quaternions [7]:
Properties

Method 6 – Exponential Maps

We can formulate an exponential map from \( R^n \) to \( V^n \) as follows:

\[
\exp(\mathbf{w}) = [\cos(\|\mathbf{w}\|)] \quad \text{and for } \mathbf{w} \neq \mathbf{0}
\]

\[
\mathbf{q} = \begin{bmatrix}
\sin(\|\mathbf{w}\|) \\
\sin(\|\mathbf{w}\|) \\
\cos(\|\mathbf{w}\|)
\end{bmatrix}
\]


Quaternions [1]:
Matrix to Quaternion

- Matrix to quaternion is not too bad, I just don’t have room for it here
- It involves a few “if” statements, a square root, three divisions, and some other stuff
- See Sam Buss’s book (p. 305) for the algorithm

Quaternions [2]:
Axis-Angle to Quaternion

- A quaternion can represent a rotation by an angle \( \theta \) around a unit axis \( \mathbf{a} \):

\[
\mathbf{q} = \begin{bmatrix}
\cos(\theta/2) \\
\sin(\theta/2) \\
\sin(\theta/2) \\
\sin(\theta/2)
\end{bmatrix}
\]

- If \( \mathbf{a} \) is a unit length, then \( \mathbf{q} \) will be also

Dynamics & Kinematics

- Dynamics: Study of Motion & Changes in Motion
  - Forward: model forces over time to find state, e.g.,
    - Given: initial position \( p_0 \), velocity \( v_0 \), gravitational constants
    - Calculate: position \( p_t \) at time \( t \)
  - Inverse: given state and constraints, calculate forces, e.g.,
    - Given: desired position \( p_d \) at time \( t \), gravitational constants
    - Calculate: position \( p_0 \), velocity \( v_0 \) needed
  - For non-particle objects: rigid-body dynamics (http://bit.ly/6d5w)

- Kinematics: Study of Motion without Regard to Causative Forces
  - Modeling systems – e.g., articulated figure
  - Forward: from angles to position (http://bit.ly/32tS)
  - Inverse: finding angles given desired position (http://bit.ly/0y7b)
Summary

- Reading for Next Class: §Chapter 10, 13, §17.3 – 17.5, Eberly 2nd Edition
- Maya interface: navigation, menus, tools, primitives
- GUI, viewports, transforms, nodes, attributes, deformers, scenes
- Object modeling and rigging: driven keys, blend shape

Today: Rotations in Animation
- Flight dynamics: roll, pitch, yaw
- Matrix, angles (fixed, Euler, axis), quaternions, exponential maps
- Dynamics: forward (trajectories, simulation), inverse (e.g., ballistics)
- Kinematics: forward, inverse

Previous Videos (#3): Morphing & Other Special Effects (SFX)
- Next Set of Videos (#4): Modeling & Simulation
- Next Class: Animation for Simulation, Visualization

Terminology

- Maya Software for 3-D Modeling & Animation
  - Shelves and hotkeys, viewports
  - Channel box, deformers – controlling complex vertex meshes
- Rigging Character Models: Defining Components of Articulated Figure
  - Joints – axis of rotation, angular degree(s) of freedom (DOFs)
  - Bones – attached to joints, rotate about joint axis
- Dynamics (Motion under Forces) vs. Kinematics (Articulated Motion)
  - Roll (Rotation about x), Pitch (Rotation about y), Yaw (Rotation about z)
- Today: Six Degrees of Rotation
  - Matrix – what we studied before: 4 × 4 Homogeneous TMs
  - Euler angles – rotate around local axes (themselves rotated)
  - Axis-angle – rotate around arbitrary axis
  - Quaternions – different representation of arbitrary rotation
  - Exponential maps – 3-D representation related to quaternions