Lecture 22 of 41

Animation 2 of 3: Rotations, Quaternions Dynamics & Kinematics

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
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 *2e* – see **http://bit.ly/ieUq45**
Next class: Chapter 10, 13, §17.3 – 17.5, Eberly *2e*
Ross's Maya tutorials: **http://bit.ly/dFpTwq**
PolyFacecom's Maya character modeling tutorials: **http://bit.ly/h6tzrd**
Wikipedia, *Flight Dynamics*: **http://bit.ly/gVaQCX**

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)**
  - **Preview of character models: PolyFacecom (http://bit.ly/h6tzrd)**
- **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**
- **Next Time: Videos Part 4 – Modeling & Simulation**

Where We Are

21	Lab 4a: Animation Basics	Flash animation handout
22	Animation 2: Rotations; Dynamics, Kinematics	Chapter 17, esp. §17.1 – 17.2
23	Demos 4: Modeling & Simulation; Rotations	Chapter 10[1], 13[2], §17.3 – 17.5
24	Collisions 1: axes, OBBs, Lab 4b	§2.4.3, 8.1, GL handout
25	Spatial Sorting: Binary Space Partitioning	Chapter 6, esp. §6.1
26	Demos 5: More CGA; Picking; HW/Exam	Chapter 7[2]; § 8.4
27	Lab 5a: Interaction Handling	§ 8.3 – 8.4; 4.2, 5.0, 5.6, 9.1
28	Collisions 2: Dynamic, Particle Systems	§ 9.1, particle system handout
	Exam 2 review; Hour Exam 2 (evening)	Chapters 5 – 6, 7[2] – 8, 12, 17
29	Lab 5b: Particle Systems	Particle system handout
30	Animation 3: Control & IK	§ 5.3, CGA handout
31	Ray Tracing 1: intersections, ray trees	Chapter 14
32	Lab 6a: Ray Tracing Basics with POV-Ray	RT handout
33	Ray Tracing 2: advanced topic survey	Chapter 15, RT handout
34	Visualization 1: Data (Quantities & Evidence)	Tufte handout (1)
35	Lab 6b: More Ray Tracing	RT handout
36	Visualization 2: Objects	Tufte handout (2 & 4)
37	Color Basics; Term Project Prep	Color handout
38	Lab 7: Fractals & Terrain Generation	Fractals/Terrain handout
39	Visualization 3: Processes; Final Review 1	Tufte handout (3)
40	Project presentations 1; Final Review 2	–
41	Project presentations 2	–
	Final Exam	Ch. 1 – 8, 10 – 15, 17, 20

Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review; and the green-shaded entry, that of the term project.

Green, blue and red letters denote exam review, exam, and exam solution review dates.

References: Maya Character Rigging

Aaron Ross
Founder, Digital Arts Guild
http://dr-yo.com
http://bit.ly/fzxN74
http://www.youtube.com/user/DigitalArtsGuild

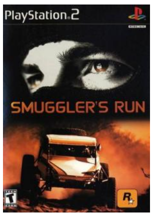
Jim Lammers
President
Trinity Animation
http://www.trinity3d.com
http://bit.ly/i6yfyV

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/

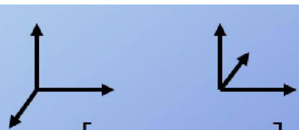
David C. Brogan
Visiting Assistant Professor, Computer Science Department, University of Virginia
http://www.cs.virginia.edu/~dbrogan/
Susquehanna International Group (SIG)
http://www.sig.com



Steve Rotenberg
Visiting Lecturer
Graphics Lab
University of California – San Diego
CEO/Chief Scientist, PixelActive
http://graphics.ucsd.edu

Spaces & Transformations

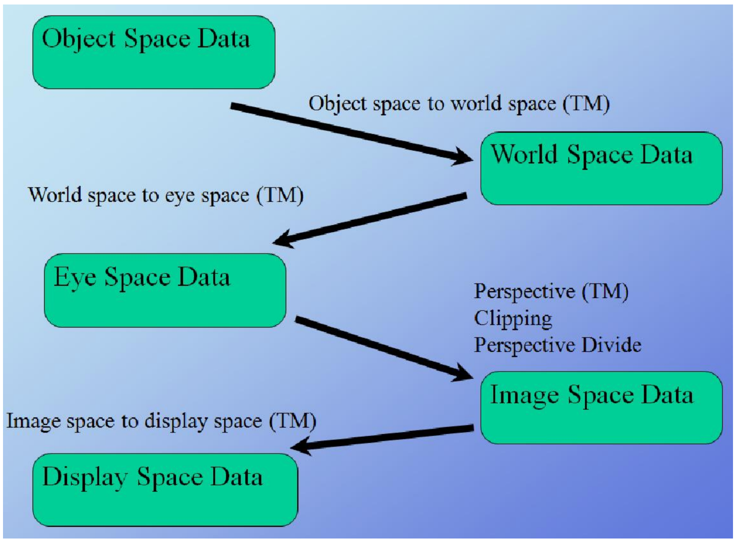
Left-handed v. right handed
Homogeneous coordinates:
4x4 transformation matrix (TM)
Concatenating TMs
Basic transformations (TMs)
Display pipeline



$$[x \quad y \quad z \quad 1]$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Display Pipeline



Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Rotations [1]: Orientation

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

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005, http://bit.ly/f0ViAN

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

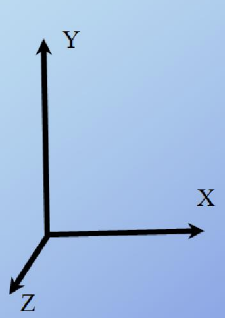
- This means that we can represent an orientation with 3 numbers
- A sequence of rotations around principal axes is called an *Euler Angle Sequence*
- Assuming we limit ourselves to 3 rotations without successive rotations about the same axis, we could use any of the following 12 sequences:

XYZ	XZY	XYX	XZX
YXZ	YZX	YXY	YZY
ZXY	ZYX	ZXZ	ZYZ

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005, http://bit.ly/f0ViAN

Representing Orientations

Example: fixed angles - rotate around global axes

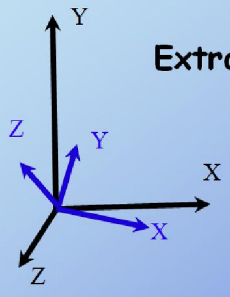


Orientation: $(\alpha \quad \beta \quad \gamma)$

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

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

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 exeed single precision arithmetic

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

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

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

CIS 536/636
Introduction to Computer Graphics
Lecture 22 of 41
Computing & Information Sciences
Kansas State University

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

University of Virginia

Adapted from slides ♥ 2000 – 2004 D. Brogan, University of Virginia
CS 445/645, Introduction to Computer Graphics, http://bit.ly/h9AHRg

Computer Science at the University of Virginia

Method 1 – Transformation Matrix [1]
4 × 4 Homogeneous TMs

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 1 – Transformation Matrix [2]: Translation

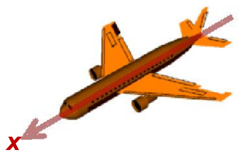
$$\begin{bmatrix} a & b & c & t_x \\ e & f & g & t_y \\ i & j & k & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

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

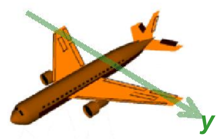
Rotation about *x* axis (Roll)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rotation about *y* axis (Pitch)

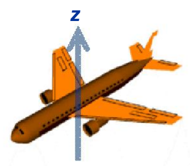
$$\begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



)06 Wikipedia,
ıht Dynamics
'/bit.ly/gVaQCX

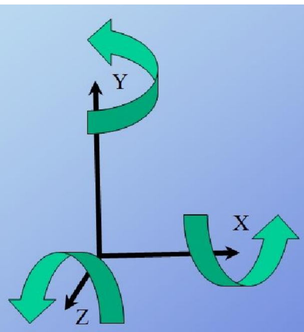
Rotation about *z* axis (Yaw)

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



Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 2 – Fixed Angles [1]



$$(\alpha \quad \beta \quad \gamma) \longrightarrow P' = R_z(\gamma)R_y(\beta)R_x(\alpha)P$$

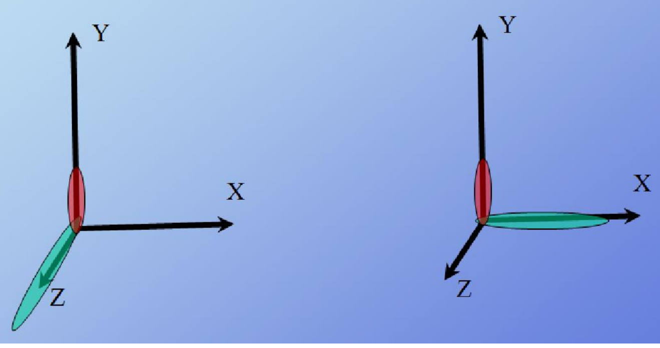
Fixed order: e.g., x, y, z; also could be x, y, x
Global axes

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 2 – Fixed Angles [2]: Gimbal Lock

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

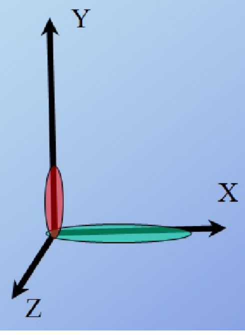
$(0 \quad 0 \quad 0)$ $(0 \quad 90 \quad 0)$



Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

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

$$\begin{pmatrix}0 & 90 & 0\end{pmatrix}$$



Fixed order of rotations: x, y, z

What do these epsilon rotations do?

$$\begin{pmatrix}0\pm\varepsilon & 90 & 0\end{pmatrix}$$

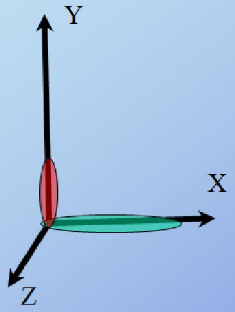
$$\begin{pmatrix}0 & 90\pm\varepsilon & 0\end{pmatrix}$$

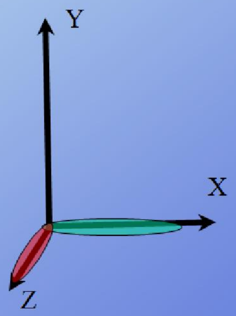
$$\begin{pmatrix}0 & 90 & 0\pm\varepsilon\end{pmatrix}$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

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

$$\begin{pmatrix}0 & 90 & 0\end{pmatrix} \qquad \begin{pmatrix}90 & 0 & 90\end{pmatrix}$$





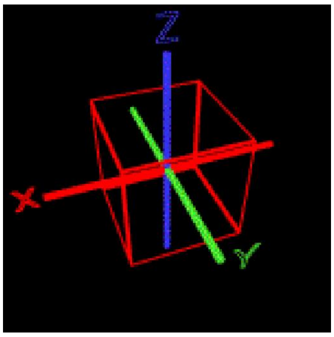
Interpolating FA representations does not produce intuitive rotation because of gimbal lock

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

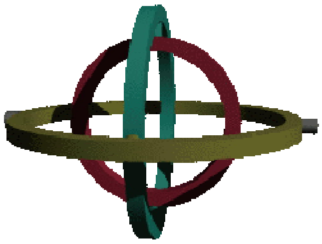
25

Method 2 – Fixed Angles [5]: Gimbal Lock Illustrated

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



Anticz.com © 2001 M. Brown
http://bit.ly/6NIXVr



Gimbal Lock © 2006 Wikipedia
(Rendered using POV-Ray)
http://bit.ly/hR88V2

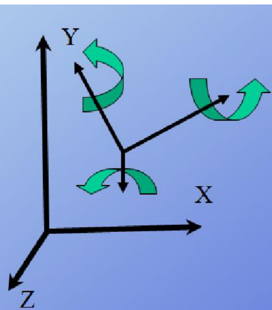
Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Computer Animation, http://bit.ly/ggbjuv

Computer Science at the University of Virginia

26

Method 3 – Euler Angles [1]

$$(\alpha \quad \beta \quad \gamma)$$



Prescribed order: e.g., x, y, z or x, y, x
Rotate around (rotated) local axes

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

$$(\alpha \quad \beta \quad \gamma) \longrightarrow P' = R_x(\alpha)R_y(\beta)R_z(\gamma)P$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 3 – Euler Angles [2]

$(\theta_x, \theta_y, \theta_z) = R_z R_y R_x$

- Rotate θ_x degrees about x-axis
- Rotate θ_y degrees about y-axis
- Rotate θ_z degrees about z-axis

Axis order is not defined

- (y, z, x), (x, z, y), (z, y, x)… are all legal
- Pick one

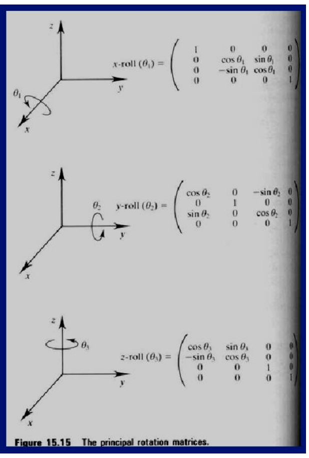


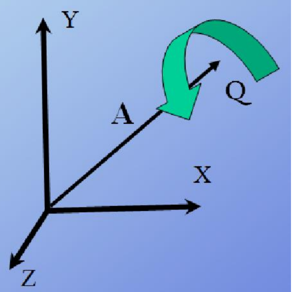
Figure 15.15 The principal rotation matrices.

Adapted from slides ♥ 2000 – 2004 D. Brogan, University of Virginia
CS 445/645, Introduction to Computer Graphics, http://bit.ly/h9AHRg

Computer Science at the University of Virginia

Method 4 – Axis-Angle [1]

$$[\theta \quad A]$$
$$[\theta \quad (x \quad y \quad z)]$$



Rotate about given axis
Euler's Rotation Theorem
OpenGL
Fairly easy to interpolate between orientations
Difficult to concatenate rotations

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

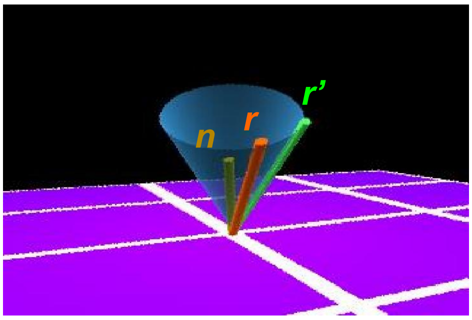
Method 4 – Axis-Angle [2]

Given

- *r* – vector in space to rotate
- *n* – unit-length axis in space about which to rotate
- ❑ –amount about *n* to rotate

Solve

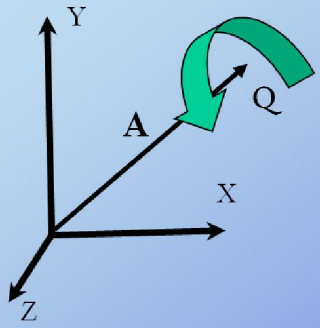
- *r’* – rotated vector



Adapted from slides ♥ 2000 – 2004 D. Brogan, University of Virginia
CS 445/645, Introduction to Computer Graphics, http://bit.ly/h9AHRg

Computer Science at the University of Virginia

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



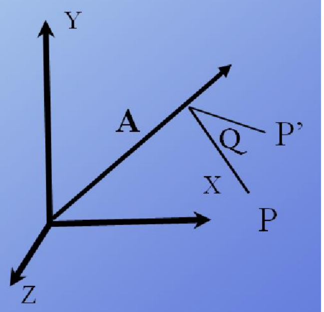
Concatenate the following:
Rotate A around z to x-z plane
Rotate A’ around y to x-axis
Rotate theta around x
Undo rotation around y-axis
Undo rotation around z-axis

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 4 – Axis-Angle [4]: Axis-Angle to Rotation Matrix

$$\hat{A} = \begin{bmatrix} a_x a_x & a_x a_y & a_x a_z \\ a_y a_x & a_y a_y & a_y a_z \\ a_z a_x & a_z a_y & a_z az \end{bmatrix}$$

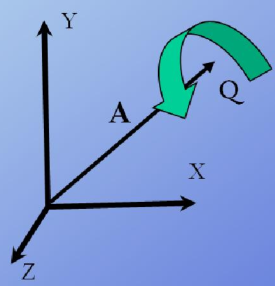
$$A^* = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$



$$Rot_{(\theta \ (x \ \ y \ \ z))} = \hat{A} + \cos(\theta)(I - \hat{A}) + \sin(\theta) A^*$$

Method 5 – Quaternions [1]

$$Rot_{(\theta \ \ A)} = \left[\cos\left(\frac{\theta}{2}\right) \quad \sin\left(\frac{\theta}{2}\right) * A \right]$$



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

$$\begin{bmatrix} s & v \end{bmatrix}$$
$$q$$

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

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 5 – Quaternions [2]: Arithmetic

Addition	$[s_1 + s_2 \quad v_1 + v_2] = [s_1 \quad v_1] + [s_2 \quad v_2]$
Multiplication	$q_1 q_2 = [s_1 s_2 - v_1 \cdot v_2 \quad s_2 v_1 + s_1 v_2 + v_1 \times v_2]$
Inner Product	$q_1 \cdot q_2 = s_1 s_2 + v_1 \cdot v_2$
Length	$\lVert q \rVert = \sqrt{q \cdot q}$

Method 5 – Quaternions [3]: Inverse & Normalization

Inverse	$q^{-1} = \frac{1}{\lVert q \rVert^2} [s \quad -v]$
	$qq^{-1} = q^{-1}q = [1 \quad (0 \quad 0 \quad 0)]$
	$(pq)^{-1} = q^{-1} p^{-1}$
Unit quaternion	$\hat{q} = \frac{q}{\lVert q \rVert}$

Method 5 – Quaternions [4]: Representation

Vector $[0 \quad v]$

Transform $v' = Rot_q(v) = qvq^{-1}$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 5 – Quaternions [5]: Geometric Operations

$$Rot_q(v) = Rot_{-q}(v)$$

$$Rot_q(v) = Rot_{kq}(v)$$

$$v'' = Rot_q(Rot_p(v)) = Rot_{qp}(v)$$

$$v'' = Rot_{q^{-1}}(Rot_q(v)) = q^{-1}(qvq^{-1})q = v$$

Adapted from slides ♥ 2007 – 2011 R. Parent, Ohio State University
CSE 682 (Computer Animation), http://bit.ly/feUESy
CSE 683/684A (Computer Animation Algorithms & Techniques), http://bit.ly/f8Myky

Method 5 – Quaternions [6]: Unit Quaternion Conversions

$$Rot_{[s\ \ x\ \ y\ \ z]} = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz-2sy \\ 2xy-2sz & 1-2x^2-2z^2 & 2yz-2sx \\ 2xz-2sy & 2yz-2sx & 1-2x^2-2y^2 \end{bmatrix}$$

Axis-Angle
$$\begin{cases} \theta = 2\cos^{-1}(s) \\ (x, y, z) = v / \|v\| \end{cases}$$

Method 5 – Quaternions [7]: Properties

Avoid gimbal lock

Easy to rotate a point

Easy to convert to a rotation matrix

Easy to concatenate – quaternion multiply

Easy to interpolate – interpolate 4-tuples

How about smoothly (in both space and time) interpolate?

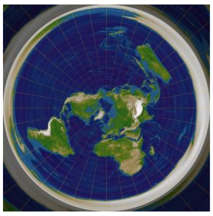
Method 6 – Exponential Maps

We can formulate an exponential map from R^3 to S^3 as follows:

$$e^{[0,0,0]^T} = [0,0,0,1]^T \qquad \text{and for } \mathbf{v} \neq \mathbf{0} \qquad e^{\mathbf{v}} = \sum_{m=0}^{\infty} (\tfrac{1}{2}\tilde{\mathbf{v}})^m = [\sin(\tfrac{1}{2}\theta)\hat{\mathbf{v}}, \ \cos(\tfrac{1}{2}\theta)]^T ,$$

$$\mathbf{q} = e^{\mathbf{v}} = \left[\sin(\tfrac{1}{2}\theta)\frac{\mathbf{v}}{\theta}, \ \cos(\tfrac{1}{2}\theta)\right]^T = \left[\frac{\sin(\tfrac{1}{2}\theta)}{\theta}\mathbf{v}, \ \cos(\tfrac{1}{2}\theta)\right]^T$$

Original paper, *Journal of Graphics Tools*: Grassia (1998), http://bit.ly/gwHQnt



Wikipedia: *Exponential Map*,

$$\exp(\tilde{\omega}) = \exp\left(\begin{bmatrix} 0 & -z\theta & y\theta \\ z\theta & 0 & -x\theta \\ -y\theta & x\theta & 0 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 2(x^2-1)s^2+1 & 2xys^2-2zcs & 2xzs^2+2ycs \\ 2xys^2+2zcs & 2(y^2-1)s^2+1 & 2yzs^2-2xcs \\ 2xzs^2-2ycs & 2yzs^2+2xcs & 2(z^2-1)s^2+1 \end{bmatrix},$$

Wikipedia: *Rotation Matrix*, http://bit.ly/edluTR

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

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005, http://bit.ly/f0ViAN

Quaternions [2]: Axis-Angle to Quaternion

- A quaternion can represent a rotation by an angle θ around a unit axis **a**:

$$\mathbf{q} = \left[\cos\frac{\theta}{2} \quad a_x \sin\frac{\theta}{2} \quad a_y \sin\frac{\theta}{2} \quad a_z \sin\frac{\theta}{2} \right]$$

or

$$\mathbf{q} = \left\langle \cos\frac{\theta}{2}, \mathbf{a} \sin\frac{\theta}{2} \right\rangle$$

- If **a** is unit length, then **q** will be also

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005, http://bit.ly/f0ViAN

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_t at time t, gravitational constants
    - Calculate: position p_0, velocity v_0 needed
  - Wikipedia: http://bit.ly/hH43dX (see also: "Analytical dynamics")
  - For non-particle objects: rigid-body dynamics (http://bit.ly/dLvejg)
- Kinematics: Study of Motion without Regard to Causative Forces
  - Modeling systems – *e.g.*, articulated figure
  - Forward: from angles to position (http://bit.ly/eh2d1c)
  - Inverse: finding angles given desired position (http://bit.ly/hsyTb0)
  - Wikipedia: http://bit.ly/hr8r2u

Forward Kinematics
© 2009 Wikipedia

Summary

- Reading for Next Class: §Chapter 10, 13, §17.3 – 17.5, Eberly *2e*
- Last Time: Maya & CGA, Ross Tutorials (http://bit.ly/dFpTwq)
  - 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
- Lab 4: Unreal Wiki Tutorial, Modeling/Rigging (http://bit.ly/dLRkXN)

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
  - Fixed angles – global basis
  - 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