GDC 09

learn
network
inspire

www.GDConf.com

Game Developers Conference®

March 23-27, 2009 | Moscone Center, San Francisco

Orientation Representation

Jim Van Verth
NVIDIA Corporation
(jim@essentialmath.com)

Topics Covered

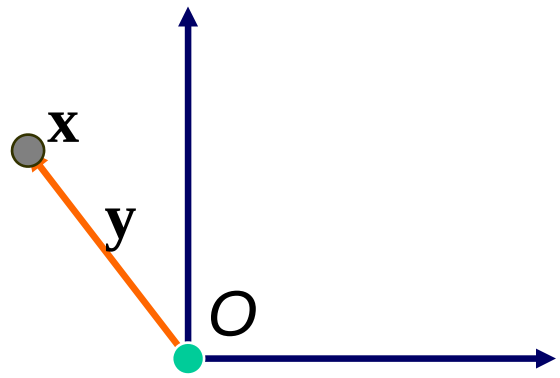
» What is orientation?
» Various orientation representations
» Why quaternions rock

Orientation vs. Rotation

- Orientation is described relative to some reference frame
- A rotation changes object from one orientation to another
- Can represent orientation as a rotation from the reference frame

Orientation vs. Rotation

- Analogy: think position and translation
- Reference is origin
- Can represent position **x** as translation **y** from origin



Ideal Orientation Format

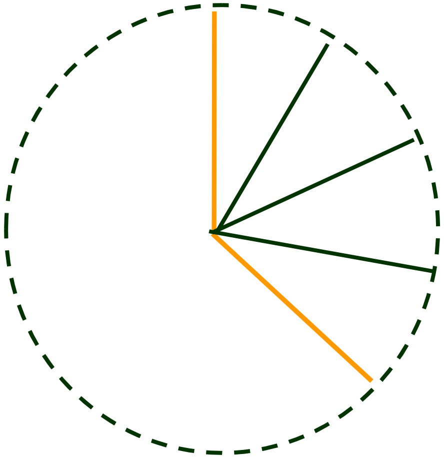
- Represent 3 degrees of freedom with minimum number of values
- Allow concatenations of rotations
- Math should be simple and efficient
  - concatenation
  - rotation
  - interpolation

Interpolating Orientation

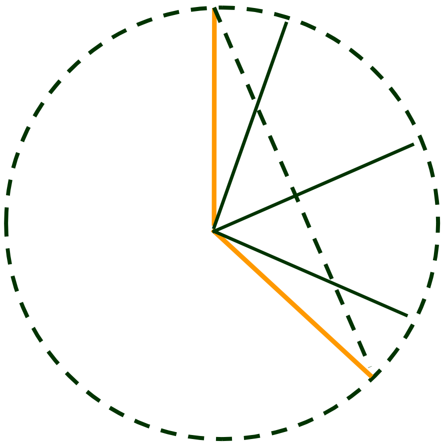
- Not as simple, but more important
- E.g. camera control
  - Store orientations for camera, interpolate
- E.g. character animation
  - Body location stored as point
  - Joints stored as rotations
- Need way to interpolate between orientations

Interpolating Orientations

- Want: interpolated orientations generate equal intervals of angle as t increases



Linear Interpolation (Lerp)



- Just like position

  $(1\text{-}t)\,\mathbf{p} + t\,\mathbf{q}$

- Problem

  Covers more arc in the middle

  I.e. rotates slower on the edges, faster in the middle

Spherical Linear Interpolation

» The solution!

» AKA slerp

» Interpolating from $\mathbf{p}$ to $\mathbf{q}$ by a factor of t

$$\mathbf{slerp}(\mathbf{p},\mathbf{q},t) = \mathbf{p}^{1-t}\mathbf{q}^{t}$$

$$= \mathbf{p}(\mathbf{p}^{-1}\mathbf{q})^{t}$$

» Problem: taking an orientation to a power is often not an easy – or cheap – operation

Orientation Formats

- Matrices
- Euler angles
- Axis-Angle
- Quaternions

Matrices as Orientation

- Matrices just fine, right?
- No...
  - 9 values to interpolate
  - don’t interpolate well

Interpolating Matrices

» Say we interpolate halfway between each element

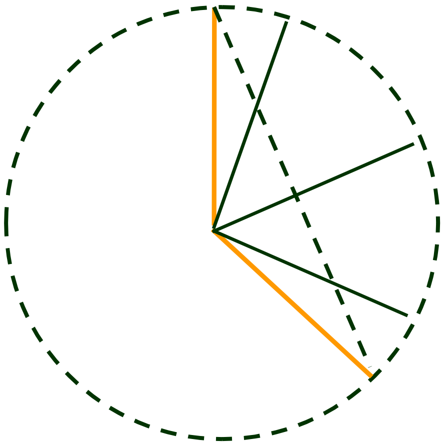
$$0.5\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}+0.5\begin{pmatrix}0&-1&0\\1&0&0\\0&0&1\end{pmatrix}=\begin{pmatrix}0.5&-0.5&0\\0.5&0.5&0\\0&0&1\end{pmatrix}$$

» Result isn't a rotation matrix!

» Need Gram-Schmidt orthonormalization

Interpolating Matrices

- Look at lerp diagram again
- Orange vectors are basis vectors



- Get shorter in the middle!

Interpolating Matrices

- Solution: do slerp?
- Taking a matrix to a power is not cheap
- Can do it by extracting axis-angle, interpolating, and converting back
- There are better ways

Why Not Euler Angles?

- Three angles
  - Heading, pitch, roll
- However
  - Dependant on coordinate system
  - No easy concatenation of rotations
  - Still has interpolation problems
  - Can lead to gimbal lock

Euler Angles vs. Fixed Angles

- One point of clarification
- Euler angle - rotates around local axes
- Fixed angle - rotates around world axes
- Rotations are reversed
  - *x-y-z* Euler angles == *z-y-x* fixed angles

Euler Angle Interpolation

- Example:
  - Halfway between (0, 90, 0) & (90, 45, 90)
  - Lerp directly, get (45, 67.5, 45)
  - Desired result is (90, 22.5, 90)
- Can use Hermite curves to interpolate
  - Assumes you have correct tangents
- AFAIK, slerp not even possible

Euler Angle Concatenation

- Can't just add or multiply components
- Best way:
  - Convert to matrices
  - Multiply matrices
  - Extract euler angles from resulting matrix
- Not cheap

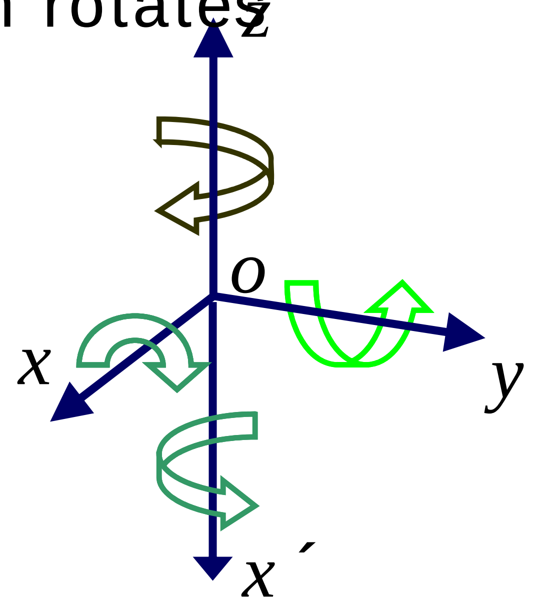
Gimbal Lock

- Euler/fixed angles not well-formed
- Different values can give same rotation
- Example with z-y-x fixed angles:

  (90, 90, 90) = (0, 90, 0)
- Why? Rotation of 90° around y aligns x and z axes
- Rotation around z cancels x rotation

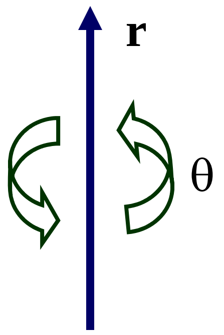
Gimbal Lock

- Loss of one degree of freedom
- Alignment of axes (e.g. rotate x into $-z$)
  - Any value of x rotation rotates cw around z axis



Axis and Angle

- Specify vector, rotate ccw around it
- Used to represent arbitrary rotation
    - orientation = rotation from reference
- Can interpolate, messy to concatenate



Axis and Angle

» Matrix conversion

$$\mathbf{R} = \begin{pmatrix} tx^2 + c & txy - sz & txz + sy \\ txy + sz & ty^2 + c & tyz - sx \\ txz - sy & tyz + sx & tz^2 + c \end{pmatrix}$$

where

$$c = \cos(\theta)$$

$$s = \sin(\theta)$$

$$t = 1 - \cos(\theta)$$

Quaternion

- Pre-cooked axis-angle format
- 4 data members
- Well-formed
- (Reasonably) simple math
  - concatenation
  - interpolation
  - rotation

What is a Quaternion?

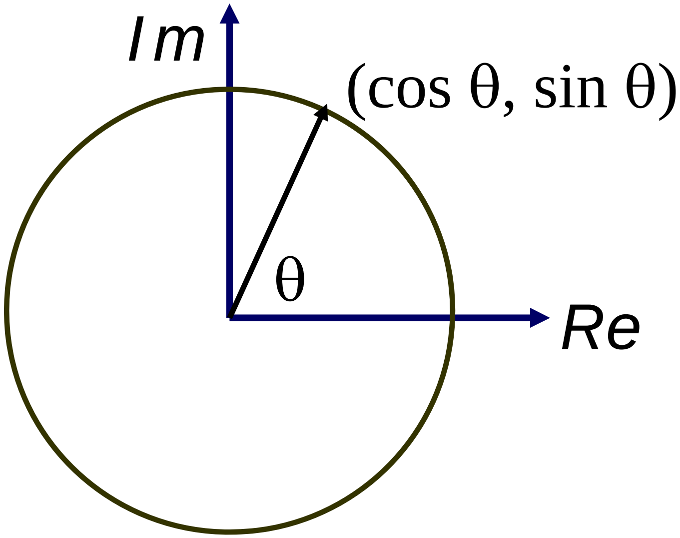
- Look at complex numbers first

$$x = a + bi$$

- If normalized ($a^2 + b^2 = 1$), can use these to represent 2D rotation

Reign on, Complex Plane

» Unit circle on complex plane



» Get

$$a + ib = \cos\theta + i\sin\theta$$
$$= e^{i\theta}$$

Digression

- You may seen this:

$$0 = e^{\pi i} + 1$$

- Falls out from

$$
\begin{aligned}
0 &= e^{\pi \mathbf{i}} + 1 \\
&= \cos\pi + \mathbf{i}\sin\pi + 1 \\
&= -1 + \mathbf{i}(0) + 1 \\
&= 0
\end{aligned}
$$

What is a Quaternion?

- Created as extension to complex numbers

$$a + b\mathbf{i}$$

becomes

$$w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

- Can rep as coordinates

$$(w, x, y, z)$$

- Or scalar/vector pair

$$(w, \mathbf{v})$$

What is Rotation Quaternion?

- Normalize quat is rotation representation
  - also avoids f.p. drift
- To normalize, multiply by

$$1/\sqrt{x^2+y^2+z^2+w^2}$$

Why *4* values?

- One way to think of it:
- 2D rotation ->
  - One degree of freedom
- Normalized complex number ->
  - One degree of freedom
- 3D rotation ->
  - Three degrees of freedom
- Normalized quaternion ->
  - Three degrees of freedom

What is Rotation Quaternion?

- Normalized quat (w, x, y, z)
- w represents angle of rotation θ

  $w = \cos(\theta/2)$
- x, y, z from <u>normalized</u> rotation axis $\hat{\mathbf{r}}$

  $(x\ y\ z) = \mathbf{v} = \sin(\theta/2)\cdot\hat{\mathbf{r}}$
- Often write as $(w, \mathbf{v})$
- In other words, modified axis-angle

Creating Quaternion

» So for example, if want to rotate 90° around z-axis:

$$w = \cos(45^\circ) = \sqrt{2}/2$$

$$x = 0 \cdot \sin(45^\circ) = 0$$

$$y = 0 \cdot \sin(45^\circ) = 0$$

$$z = 1 \cdot \sin(45^\circ) = \sqrt{2}/2$$

$$\mathbf{q} = (\sqrt{2}/2, 0, 0, \sqrt{2}/2)$$

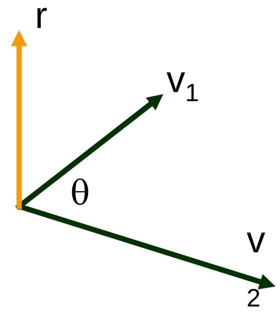
Creating Quaternion

- Another example
  - Have vector $\mathbf{v}_1$, want to rotate to $\mathbf{v}_2$
  - Need rotation vector $\hat{\mathbf{r}}$, angle θ

$$\theta = \mathrm{acos}(\hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2)$$

$$\mathbf{r} = \mathbf{v}_1 \times \mathbf{v}_2$$

  - Plug into previous formula



Creating Quaternion

- From Game Gems 1 (Stan Melax)
- Use trig identities to avoid arccos

  Normalize $\mathbf{v}_1$, $\mathbf{v}_2$

$$\mathbf{r} = \hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_2 \qquad s = \sqrt{2(1 + \hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2)}$$

  Build quat

$$\mathbf{q} = \left(\frac{s}{2}, \frac{\mathbf{r}}{s}\right)$$

  More stable when $\mathbf{v}_1$, $\mathbf{v}_2$ near parallel

Multiplication

» Provides concatenation of rotations

» Take $\mathbf{q}_0 = (w_0, \mathbf{v}_0) \quad \mathbf{q}_1 = (w_1, \mathbf{v}_1)$

» If w_0, w_1 are zero:

» Non-commutative:

$$\mathbf{q}_0\mathbf{q}_1 \neq \mathbf{q}_1\mathbf{q}_0$$

Identity and Inverse

- Identity quaternion is $(1, 0, 0, 0)$
  - applies no rotation
  - remains at reference orientation
- $\mathbf{q}^{-1}$ is inverse
  - $\mathbf{q} \cdot \mathbf{q}^{-1}$ gives identity quaternion
- Inverse is same axis but opposite angle

Computing Inverse

» $(w, \mathbf{v})^{-1} = (\cos(\theta/2), \sin(\theta/2) \cdot \hat{\mathbf{r}})$

$$(w, \mathbf{v})^{-1} = (\cos(-\theta/2), \sin(-\theta/2) \cdot \hat{\mathbf{r}})$$

$$(w, \mathbf{v})^{-1} = (\cos(\theta/2), -\sin(\theta/2) \cdot \hat{\mathbf{r}})$$

$$(w, \mathbf{v})^{-1} = (w, -\mathbf{v})$$

» Only true if $\mathbf{q}$ is normalized

i.e. $\mathbf{r}$ is a unit vector

» Otherwise scale by

$$1/(w^2 + x^2 + y^2 + z^2)$$

Vector Rotation

- Have vector $\mathbf{p}$, quaternion $\mathbf{q}$
- Treat $\mathbf{p}$ as quaternion $(0, \mathbf{p})$
- Rotation of $\mathbf{p}$ by $\mathbf{q}$ is $\mathbf{q}\,\mathbf{p}\,\mathbf{q}^{-1}$
- Vector $\mathbf{p}$ and quat $(w, \mathbf{v})$ boils down to

assumes $\mathbf{q}$ is normalized

Vector Rotation (cont'd)

- Why does $\mathbf{q}\ \mathbf{p}\ \mathbf{q}^{-1}$ work?
- One way to think of it:
  - first multiply rotates halfway and into 4th dimension
  - second multiply rotates rest of the way, back into 3rd
- See references for more details

Vector Rotation (cont'd)

- Can concatenate rotation

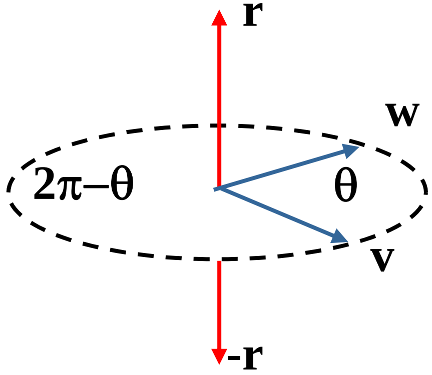
$$\mathbf{q}_1 \cdot (\mathbf{q}_0 \cdot \mathbf{p} \cdot \mathbf{q}_0^{-1}) \cdot \mathbf{q}_1^{-1} = (\mathbf{q}_1 \cdot \mathbf{q}_0) \cdot \mathbf{p} \cdot (\mathbf{q}_1 \cdot \mathbf{q}_0)^{-1}$$

- Note multiplication order: right-to-left

Demo

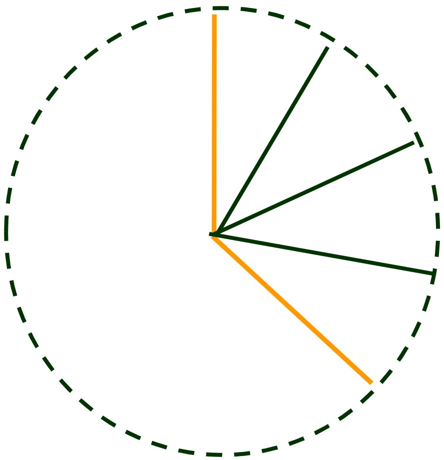
Vector Rotation (cont'd)

- $\mathbf{q}$ and $-\mathbf{q}$ rotate vector to same place
- But not quite the same rotation
- $-\mathbf{q}$ has axis $-\mathbf{r}$, with angle $2\pi-\theta$
- Causes problems with interpolation



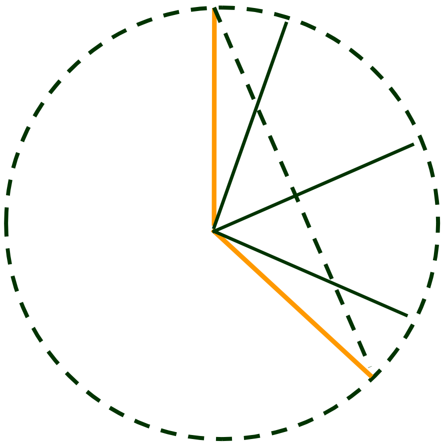
Quaternion Interpolation

- Recall: Want equal intervals of angle



Linear Interpolation

- Familiar formula
  - $(1\text{-}t)\,\mathbf{p} + t\,\mathbf{q}$
- Familiar problems
  - Cuts across sphere
  - Moves faster in the middle
  - Resulting quaternions aren't normalized



Spherical Linear Interpolation

- There *is* a (somewhat) nice formula for slerp:

where $\cos \alpha = \mathbf{p} \bullet \mathbf{q}$

And $\mathbf{p}$, $\mathbf{q}$ *unit* quaternions

Faster Slerp

- Lerp is pretty close to slerp
- Just varies in speed at middle
- Idea: can correct using simple spline to modify t (adjust speed)
- From Jon Blow's column, *Game Developer*, March 2002
- Near lerp speed w/slerp precision

Faster Slerp

```
float f = 1.0f - 0.7878088f*cosAlpha;
float k = 0.5069269f;
f *= f;
k *= f;
float b = 2*k;
float c = -3*k;
float d = 1 + k;
t = t*(b*t + c) + d;
```

Faster Slerp

- Alternative technique presented by Thomas Busser in Feb 2004 *Game Developer*
- Approximate slerp with spline function
- Very precise – but necessary? Not sure

Which One?

- Technique used depends on data
- Lerp generally good enough for motion capture (lots of samples)
    - Need to normalize afterwards
- Slerp only needed if data is sparse
    - Blow's method for simple interpolation
    - (Also need to normalize)

- These days, Blow says just use lerp. YMMV.

One Caveat

- Negative of normalized quat rotates vector to same place as original
  - (–axis, 2π–angle)
- If dot product of two interpolating quats is < 0, takes long route around sphere
- Solution, negate one quat, then interpolate
- Preprocess to save time

Operation Wrap-Up

- Multiply to concatenate rotations
- Addition only for interpolation (don't forget to normalize)
- Be careful with scale
  - Quick rotation assumes unit quat
  - Don't do $(0.5 \cdot \mathbf{q}) \cdot \mathbf{p}$
  - Use lerp or slerp with identity quaternion

Quaternion to Matrix

» Normalized quat converts to 3x3 matrix

$$\mathbf{M}_q = \begin{pmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy \\ 2xy-2wz & 1-2x^2-2z^2 & 2yz+2wx \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 \end{pmatrix}$$

Quats and Transforms

- Can store transform in familiar form
  - Vector $\mathbf{t}$ for translation (just add)
  - Quat $\mathbf{r}$ for orientation (just multiply)
  - Scalar s for uniform scale (just scale)
- Have point $\mathbf{p}$, transformed point is

$$\mathbf{p}' = \mathbf{r}(s\mathbf{p})\mathbf{r}^{-1} + \mathbf{t}$$

Quats and Transforms (cont'd)

- Concatenation of transforms in this form

$$s' = s_1 s_2$$

$$\mathbf{r}' = \mathbf{r}_2 \mathbf{r}_1$$

$$\mathbf{t}' = \mathbf{t}_2 + \mathbf{r}_2 (s_2 \mathbf{t}_1) \mathbf{r}_2^{-1}$$

- Tricky part is to remember rotation and scale affect translations

Summary

- Talked about orientation
- Formats good for internal storage
  - Matrices
  - Quaternions
- Formats good for UI
  - Euler angles
  - Axis-angle
- Quaternions funky, but generally good

References

- Shoemake, Ken, "Animation Rotation with Quaternion Curves," *SIGGRAPH '85*, pp. 245-254.
- Shoemake, Ken, "Quaternion Calculus for Animation," SIGGRAPH Course Notes, *Math for SIGGRAPH*, 1989.
- Hanson, Andrew J., *Visualizing Quaternions*, Morgan Kaufman, 2006.
- Van Verth, James M. and Lars M. Bishop, *Essential Mathematics for Games and Interactive Applications,* 2nd Edition, Morgan Kaufman, 2008.

References

- Blow, Jonathan, “Hacking Quaternions,” *Game Developer*, March 2002.
- Busser, Thomas, “PolySlerp: A fast and accurate polynomial approximation of spherical linear interpolation (Slerp),” *Game Developer*, February 2004.
- Van Verth, Jim, “Vector Units and Quaternions,” *GDC 2002. http://www.essentialmath.com*