Quaternions

"Quaternions came from Hamilton after his really good work had been done; and, though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Clark Maxwell."

Lord Kelvin, 1892.

Complex numbers on steroid

How did you feel the first time you heard about complex numbers and square roots of -1?

Did you ever wonder if your math teacher pulled them out of his hat? Was it April fool's day?

Why are math teachers allowed to invent imaginary roots for -1?

Can we do it too? And why stop at one imaginary root?

In this lecture, I will give you not one, not two, but three square roots of -1

- And if it's not enough, I can come up with four more!

Rotation interpolation: Flawed Solution

Interpolate each entry independently

Example: M_0 is identity and M_1 is 90^o around x-axis

$$\text{Interpolate}\left(\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&0&1\\0&-1&0\end{bmatrix}\right)=\begin{bmatrix}1&0&0\\0&0.5&0.5\\0&-0.5&0.5\end{bmatrix}$$

Is the result a rotation matrix?

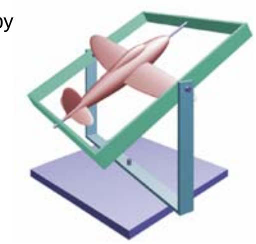
No, it does not preserve rigidity (angles and lengths)

Euler Angles

An Euler angle is a rotation about a single axis.

Any orientation can be described by composing three rotations, one around each coordinate axis.

Roll, pitch and yaw (perfect for flight simulation)



http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

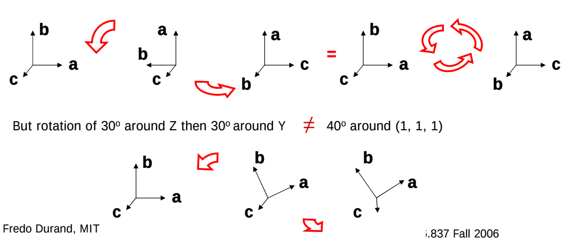
Interpolating Euler Angles

Natural orientation representation: interpolate independently 3 angles for 3 degrees of freedom

However, leads to unnatural interpolation:

rotation of 90^o around Z, then 90^o around Y = 120^o around (1, 1, 1)

But rotation of 30^o around Z then 30^o around Y ≠ 40^o around (1, 1, 1)



Euler ambiguity

$\theta_y, \theta_p, \theta_r$ represents the same rotation as

$\theta'_y = \theta_y \pm \pi$

$\theta'_p = -\theta_p \pm \pi$

$\theta'_r = \theta_r \pm \pi$

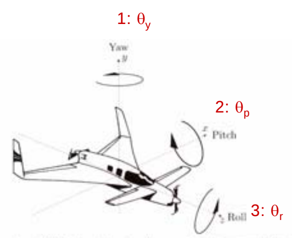
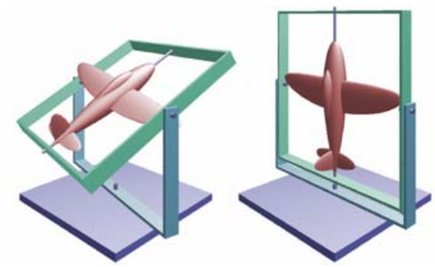


Figure XII.1. Yaw, pitch, and roll represent rotations around the y-a

From Buss page 297

Gimbal Lock

Two or more axis align resulting in a loss of rotation degrees of freedom.

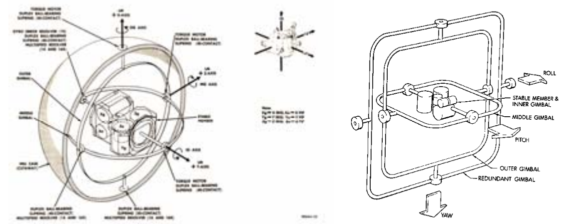


http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

Euler Angles in the Real World

Apollo inertial measurement unit

To “prevent” lock, they added a fourth Gimbal!



http://www.hq.nasa.gov/office/pao/History/alsj/gimbals.html

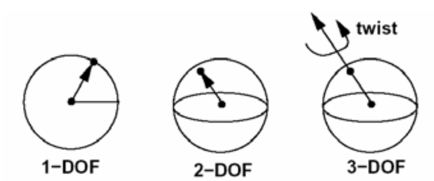
Solution: Quaternion Interpolation

Interpolate orientation on the unit sphere in 4D

- Logical and easy, isn't it?

By analogy:

1-, 2-, 3-DOF rotations as constrained points on 1, 2, 3-spheres in 2D, 3D and 4D



Euler vs. Quaternion: the revenge

Euler is good

- For user interface
  - But often needs to be added a fourth angle to prevent Gimbal lock

Quaternions are good

- For interpolation
- For math nerds
- For quantum physics

Questions?

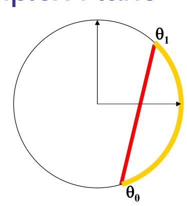
1D Sphere and Complex Plane

1 angle

- But messy to handle because modulo 2π

Solution:

- Represent rotation by point on circle
- Use interpolation in 2D plane
- Project back to circle



And we can say that the 2D plane is the complex plane

- Orientation = complex argument of the number
- Not strictly necessary, but extends more easily to 3D rotations

- Interestingly, composition of rotation ⇔ complex multiplication
  - Trivial with exponential notation $re^{i\theta}$

Velocity Issue: *lerp* vs. *slerp*

Linear Interpolation (*lerp*) interpolates the straight line between the two orientations

$$\text{lerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \mathbf{q}_0(1-t) + \mathbf{q}_1 t$$

→ *lerp* motion does not have uniform velocity:

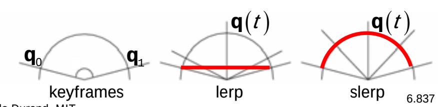
Spherical Linear Interpolation (*slerp*) interpolates along the arc lines by adding a sine term:

$$\text{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$

where ω is the angle between q_0 and q_1

Note that we still interpolate in 2D plane, but with non-uniform speed

Silly to make things so complex in 2D, but will be critical in 3D



interpolate along arc line rather than secant

Slerp derivation

Frame c_0, v

v $=[c_1-(c_1.c_0)c_0]/\sin\omega$

$=[c_1-\cos\omega\, c_0]/\sin\omega$

P(t) in frame c_0, v:

($\cos\omega t$, $\sin\omega t$)

That is

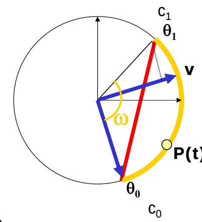
$P(t)=c_0 \cos\omega t+v \sin\omega t$

$=c_0 \cos\omega t +\sin\omega t\, [c_1-\cos\omega\, c_0]/\sin\omega$

$=c_0\, [\cos\omega t -\sin\omega t \cos\omega/\sin\omega]+ c_1\sin\omega t /\sin\omega$

$=c_0\, [\cos\omega t \sin\omega -\sin\omega t \cos\omega]/\sin\omega+ c_1\sin\omega t /\sin\omega$

$=c_0 \cos(\omega-\omega t) /\sin\omega+ c_1\sin\omega t /\sin\omega$



[remember your triginometry!]

Side note: Trigonometry formulae

Best way to remember trigonometry: complex numbers:

$e^{i(\alpha+\beta)} = e^{i\alpha}e^{i\beta}$

$(\cos\alpha+\beta, \sin\alpha+\beta)=(\cos\alpha, \sin\alpha)(\cos\beta, \sin\beta)$

now recall complex number multiplication

$(\cos\alpha+\beta, \sin\alpha+\beta)$

$=(\cos\alpha\cos\beta - \sin\alpha\sin\beta,\quad \cos\alpha\sin\beta+\cos\beta\sin\alpha)$

Other way: matrices

$$\begin{pmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix}$$

Simpler expression of slerp

Directly interpolate in exponential form

$e^{i(\beta-\alpha)t}e^{i\alpha}$

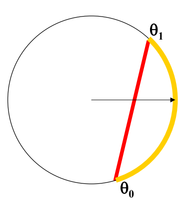
That is

$\text{slerp}(c_0, c_1, t)=(c_1 c_0^{-1})^t c_0$

Recap

rotation in 2D embedded on unit circle

slerp for uniform speed



2-Angle Orientation

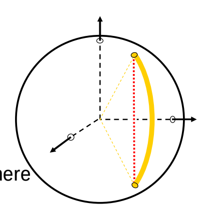
2 angles

- Messy because modulo 2π and pole

Embed 2-sphere in 3D

Use linear interpolation in 3D space

Orientation = projection onto the sphere



Use slerp for velocity correction

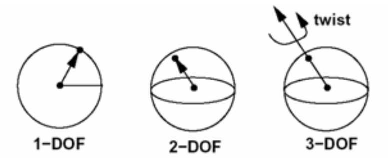
- Note that velocity is still a 1D problem along the great circle

3 Angles – Quaternions!

Use the same principle
- interpolate in higher-dimensional space
- Project back to unit sphere

Probably need the 3-sphere embedded in 4D

More complex, harder to visualize
- And some subtle differences



Question?

Quaternions: hypercomplex numbers

Due to Hamilton (1843)

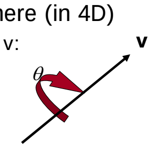
Can be defined like complex numbers but with 4 coordinates
- d+ai+bj+ck
- One real part (d), three imaginary ones.

Based on three different roots of -1:
- $i^2 = j^2 = k^2 = -1$

Quaternions: rotation representation

Quaternions are unit vectors on 3-sphere (in 4D)
- Right-hand rotation of q radians about v: q = (cos(θ/2); **v** sin(θ/2)),
  - Note how the 3 imaginary coordinates are noted as a vector

  → also often noted (s, **v**) or (d, **u**)



Notes
- We use *unit* quaternions to represent rotations
- q & -q represent the same orientation

Quaternion to matrix

Quaternion (d, a, b, c)

$$\begin{pmatrix} 1-2b^2-2c^2 & 2ab-2dc & 2ac+2db \\ 2ab+2dc & 1-2a^2-2c^2 & 2yz-2wx \\ 2ca-2db & 2bc+2da & 1-2a^2-2b^2 \end{pmatrix}$$

Demo

http://www.gamedev.net/reference/programming/features/qpowers/page7.asp

What do we need?

What do we want to do with rotations?

➔What we need to learn to do with quaternions

- Compose
- Interpolate
  - Multiply by a scalar
  - Add
- Invert
- Apply to a vector
- Convert back and forth
  - Axis-angle to quaternion
  - Quaternion to matrix
  - Also Matrix, Euler

Quaternions: teaser

Algebraic formulation will allow us to add, multiply, invert, interpolate

We'll make sure it's all consistent with geometric formulation

Rotations will be unit quaternions

Multiplication will be composition of rotations

Conjugation will be the inverse

We'll be able to interpolate in the 4D space and project back to the unit sphere of quaternions

Some differences with 2D

Question?

Complex numbers/Quaternions

Complex

- c=a+bi

Addition: componentwise

Multiplication by scalar:

s(a+ib)

= sa+sbi

Quaternions

- q=d+ai+bj+ck
- (note the different convention)

Addition: componentwise

Mulitplication by scalar:

s(d+ia+bj+ck)

= sd+sai+sbj+sck

Multiplication

Basic rule:

- $i^2=-1$

Consequence (by linearity)

(a+bi)(a'+b'i)

=(aa'-bb')+i(ab'+a'b)

Basic rules:

- $i^2 = j^2 = k^2 = -1$
- ij = k = -ji
- jk = i = -kj
- ki = j = -ik

Consequence (by linearity)

(d+ia+jb+kc)(d'+ia'+jb'+kc')

= (dd'-aa'-bb'-cc')
+ (da'+ad'+bc'-cb') i
+ (db'+d'b+ca'-ac') j
+ (dc'+cd'+ab'-ba') k

Crucial note:
multiplication is **not** commutative

- for example, jk=— kj

Geometric formulation of multiplication

q=(d, a, b, c)= (d; **u**) = (cos θ/2; sinθ/2 **v**)

Study the previous formula

(d+ia+jb+kc)(d'+ia'+jb'+kc')

= (dd' -aa'-bb'-cc')+
(da'+ad' + bc'-cb') i + (db'+d'b + ca'-ac') j + (dc'+cd' + ab'-ba') k

The first part contains a dot product, the second part has the same "crossing pattern" as cross products

(d; **u**) (d';**u'**)= (dd'-**u.u'**, d**u'**+d'**u** + **u**× **u'**)

Verifications

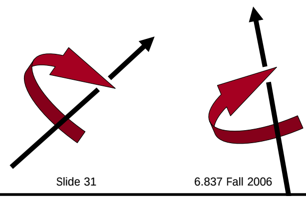
- Check the formula
- We have scalars on the left, vectors on the right

Multiplication = composition

Fundamental property of quaternions:

If quaternions q_1 & q_2 represent rotations R_1 and R_2, then the multiplication q_1q_2 represents the composition R_1R_2

- And remember that quaternion multiplication is non commutative



Multiplication = composition

(d; **u**) (d';**u'**)= (dd'-**u.u'**, d**u'**+d'**u** + **u**× **u'**)

Sanity check:
a radians around **v** times *b* radians around **v**

- (cos(a/2); **v** sin(a/2)) (cos(b/2); **v** sin(b/2))
- (cos(a/2)cos(b/2) - sin(a/2)**v** · sin(b/2)**v** ;
  cos(b/2)sin(a/2)**v** + cos(a/2)sin(b/2)**v** + **v**×**v**)
- (cos(a/2)cos(b/2) - sin(a/2)sin(b/2) ;
  v [cos(b/2)sin(a/2) + cos(a/2)sin(b/2)])
  because sin(x+y) = sin x cos y + cos x sin y
  cos(x+y) = cos x cos y – sin x sin y
- (cos((a+b)/2) ; **v** sin((a+b)/2))
  (a+b) radians around **v**

What we know about quaternions

2 constructions:

- (cos θ/2; sin θ/2 **v**) represents rotation of θ around **v**
- Extension of complex numbers:
  d+ia+jb+kc with 3 roots of -1

Addition, multiplication by scalar

Multiplication:
(d; **u**) (d';**u'**)= (dd'-**u.u'**; d**u'**+d'**u** + **u**× **u'**)

- Corresponds to rotation composition
- Non commutative, same as rotations

What do we need?

What do we want to do with rotations?
➔What we need to learn to do with quaternions

- Compose
- Interpolate
  - Multiply by a scalar
  - Add
- Invert
- Apply to a vector
- Convert back and forth
  - Axis-angle to quaternion
  - Quaternion to matrix
  - Also Matrix, Euler

Question?

Various properties

Are addition/multiplcation commutative, associative, distributive?

Quaternion addition is commutative and associative

- q+s=s+q
- q+s+r=(q+s)+r=q+(s+r)

Quaternion multiplication is associative

- qsr=(qs)r=q(sr)

Quaternion multiplication is not commutative

- qs≠ sq

Left and right distribution hold

- q(r+s)=qr+qs
- (r+s)q=rq+sq

Difference complex/quaternions

1/ Quaternions have non-commutative multiplication

2/ q and –q represent the same rotation

- Consider (cos $\theta/2$; sin$\theta/2$ **v**)
- Same as (cos -$\theta/2$; (sin-$\theta/2$) -**v**)
- This means a rotation of -θ around -**v**

3/ Quaternions look a lot scarier!

Norm & unit complex/quaternions

Complex numbers

Norm/magnitude of c=a+ib :

$$\sqrt{a^2+b^2}$$

Quaternions

Norm/magnitude of q=d+ia+jb+kc:

$$\sqrt{d^2+a^2+b^2+c^2}$$

Different formulation:

$$||(d;\mathbf{u})|| = \sqrt{d^2+||\mathbf{u}||^2}$$

Verify that rotations are unit quaternions

- rotation of angle θ around unit axis **v**

$||(\cos\theta/2; \sin\theta/2\ \mathbf{v})||^2$
$= \cos^2\theta/2 + \sin^2\theta/2\ ||\mathbf{u}||^2$
$= \cos^2\theta/2 + \sin^2\theta/2$
$= 1$

Identity quaternion

How do we represent the identity?

For complex numbers/2D rotation:

(1, 0)

For quaternions:

No preferred axis ➔ suggests null vector part

We need Iq=q

- Recall: (d; **u**) (d';**u'**)= (dd' - **u.u'**, d**u'** + d'**u** + **u**× **u'**)
- We want (d; **u**)(d_I; **0**)=(d; **u**)
- ➔ the identity quaternion is (1; **0**)

Inverse and conjugate

Complex numbers

Conjugate (a+ib)* =a-ib

Inverse: cc'=1 ➔ c'=c* / $||c||^2$

For rotations:

Only unit complex numbers

c* is the inverse rotation of c

Trivial given the polar notation

Quaternions

Conjugate: (d; **u**)* = (d; -**u**)

Do we have the same formula for the inverse?

That is, how do we get $q^{-1}q$=(1; **0**)

Inverse and conjugate

Recall: (d; **u**) (d';**u'**)= (dd'-**u.u'**; d**u'**+d'**u** + **u**× **u'**)

Consider a unit quaternion (d; **u**) and its conjugate (d; -**u**)

(d; u)*(d; u) =(d; -**u**)(d; **u**)
=(dd+**u.u**; d**u** – d**u** – **u**× **u**)
=(1; **0**) [Since (d; **u**) has unit length, dd+**u.u**=1]

We have shown that the conjugate of a unit quaternion is its inverse

- Note that the left and right inverses are the same

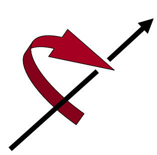
Inverse & conjugate: geometry

(cos $\theta/2$; sin$\theta/2$ **v**) represents a rotation of angle θ around **v**

Inverse rotation:

- Angle -θ around **v**
- Angle θ around –**v**

In both cases, leads to (cos $\theta/2$; - sin$\theta/2$ **v**)



Duh!

Inverse & conjugates: matrices

What is the inverse of a rotation matrix?

The conjugate/transpose matrix!

- For a rotation (or any orthonormal matrix) $M^TM=I$
- Formally, to get the conjugate of a complex-valued matrix, take the transpose and the conjugate of each coefficient.

The notion of conjugation is the related between matrices & quaternions

- Ain't that cool?

Norm and conjugate

Complex numbers	Quaternions
$\|\|c\|\|^2 = cc^*$	$\|\|q\|\|^2 = qq^* = q^*q$

What we know about quaternions

2 constructions:

- $(\cos \theta/2; \sin \theta/2\ \mathbf{v})$ represents rotation of θ around $\mathbf{v}$
- Extension of complex numbers: d+ia+jb+kc with 3 roots of -1

Addition, multiplication by scalar

Multiplication:
$(d;\ \mathbf{u})\ (d';\mathbf{u'}) = (dd'-\mathbf{u}.\mathbf{u'};\ d\mathbf{u'}+d'\mathbf{u} + \mathbf{u}\times \mathbf{u'})$

- Corresponds to rotation composition
- Non commutative, same as rotations

Identity quaternion is $(1;\ \mathbf{0})$

The conjugate $(d; -\mathbf{u})$ is the inverse rotation

- $q^*q=(1;\ \mathbf{0})$ for unit quaternions

What do we need?

What do we want to do with rotations?

➔What we need to learn to do with quaternions

- Compose
- Interpolate
  - Multiply by a scalar
  - Add
- Invert
- Apply to a vector
- Convert back and forth
  - Axis-angle to quaternion
  - Quaternion to matrix
  - Also Matrix, Euler

Questions?

Rotating a point: 2D/complex case

We can represent a 2D vector (x, y) with a complex $re^{i\alpha}$

Rotation of angle θ is the complex $e^{i\theta}$

Rotation application = complex multiplication

Warning: This is not so simple for quaternions & 3D rotations

- For one thing, quaternions are 4D and vectors are 3D

Rotating a point

The vector **p** is represented by the quaternion (0, **p**)
To rotate 3D point/vector **p** by rotation/quaternion q, compute:

$$q\ (0; p)\ q^{-1}$$

Rotating a point

$q\ (0; p)\ q^{-1}$ & $(d; \mathbf{u})\ (d'; \mathbf{u'}) = (dd' - \mathbf{u}.\mathbf{u'},\ d\mathbf{u'} + d'\mathbf{u} + \mathbf{u} \times \mathbf{u'})$

Example: $p = (x,y,z)$
$q = (\cos(\theta/2), 0, 0, \sin(\theta/2)) = (c, 0, 0, s)$
$q^{-1} = (\cos(\theta/2), 0, 0, -\sin(\theta/2)) = (c, 0, 0, -s)$

$$\begin{aligned}
q\ (0; p)\ q^{-1} &= (c, 0, 0, s)\ (0, x, y, z)\ (c, 0, 0, -s) \\
&= (c*0 - zs\ ;\ c\mathbf{p} + 0(0,0,s) + (0,0,s) \times \mathbf{p})\ (c,0,0,-s) \\
&= (-zs\ ;\ c\mathbf{p} + (-sy, sx, 0))\ (c,0,0,-s) \\
&= (-zsc - (c\mathbf{p} + (-sy,sx,0)) \cdot (0,0,-s)\ ;\ -zs(0,0,-s) \\
&\qquad + c(c\mathbf{p} + (-sy,sx,0)) + (c\mathbf{p} + (-sy,sx,0)) \times (0,0,-s)) \\
&= (0;\ (0,0,zs^2) + c^2\mathbf{p} + (-csy,csx,0) + \\
&\qquad (-csy,csx,0) + (s^2x,s^2y,0)) \\
&= (0,\ c^2x - 2csy - s^2x,\ c^2y + 2csx - s^2y,\ zs^2 + sc^2) \\
&= (0,\ x\cos(q/2) - y\sin(q/2),\ x\sin(q/2) + y\cos(q/2),\ z)
\end{aligned}$$

Can we prove it more formally?

You need to decompose **p** along a component $\mathbf{p_1}$ along **v** and one $\mathbf{p_2}$ orthogonal to **v**

Use linearity.

Prove that $(c, s\mathbf{v})(0; \mathbf{p}_1)(c, -s\mathbf{v}) = (0, \mathbf{p}_1)$

Prove that $(c, s\mathbf{v})(0; \mathbf{p}_2)(c, -s\mathbf{v})$ is the appropriate rotation

- Express in the frame $\mathbf{v}$, $\mathbf{p}_2$, $\mathbf{v} \times \mathbf{p}_2$

What we know about quaternions

2 constructions:

- $(\cos\theta/2; \sin\theta/2\ \mathbf{v})$ represents rotation of θ around **v**
- Extension of complex numbers: d+ia+jb+kc with 3 roots of -1

Addition, multiplication by scalar

Multiplication: $(d; \mathbf{u})\ (d'; \mathbf{u'}) = (dd' - \mathbf{u}.\mathbf{u'};\ d\mathbf{u'} + d'\mathbf{u} + \mathbf{u} \times \mathbf{u'})$

- Corresponds to rotation composition
- Non commutative, same as rotations

Identity quaternion is (1; **0**)

The conjugate (d; -**u**) is the inverse rotation

- q*q=(1; **0**) for unit quaternions

To apply a rotation to **p**, compute $q\ (0; \mathbf{p})\ q^{-1}$

What do we need?

What do we want to do with rotations?
➔What we need to learn to do with quaternions

- Compose
- Interpolate
  - Multiply by a scalar
  - Add
- Invert
- Apply to a vector

- Convert back and forth
  - Axis-angle to quaternion
  - Quaternion to matrix
  - Also Matrix, Euler

Question?

Interpolation: two rotations

Given two rotations v, θ and v' θ', perform linear interpolation:

Convert to quaternions q & q'

- Choose between q & -q: minimize angle with q'

Use slerp to interpolate in 4D quaternion space

- You get a rotation for each time step

Fredo Durand, MIT — Slide 55 — 6.837 Fall 2006

Exponential form of quaternions

We've seen addition & multiplication

What about exponential? Log?

- They are really useful for complex numbers
- So are they for quaternions

Can be derived with usual series expression of exponential

- exp(d, m**u**) =exp(d) (cos m; **u** sin m)
  where **u** is a unit vector

Exponential form of quaternions:
every quaternion q can be written as

q= R. exp ((0; **u**)θ)

where R is real, u is unit length and θ is real

- similar to $c=r\,e^{i}\theta$ for complex

log(q)=[log R, uθ]

Fredo Durand, MIT — Slide 56 — 6.837 Fall 2006

slerp in quaternion exponential

In complex: $slerp(c_0, c_1, t)=(c_1 c_0^{-1})^t c_0$

Same in quaternion: $slerp(q_0, q_1, t)=(q_1\, q_0^{-1})^t q_0$

With the power t easy to compute in exponential form

Fredo Durand, MIT — Slide 57 — 6.837 Fall 2006

Interpolation: two rotations

Given two rotations v, θ and v' θ', perform linear interpolation:

Convert to quaternions q & q'

- Choose between q & -q: minimize angle with q'
- Use exponential form

Use slerp to interpolate in 4D quaternion space

- $slerp(q_0, q_1, t)=(q_1\, q_0^{-1})^t q_0$

You get a rotation for each time step

Fredo Durand, MIT — Slide 58 — 6.837 Fall 2006

What we know about quaternions

2 constructions:

- (cos θ/2; sin θ/2 **v**) represents rotation of θ around **v**
- Extension of complex numbers: d+ia+jb+kc with 3 roots of -1

Addition, multiplication by scalar

Multiplication: (d; **u**) (d';**u'**)= (dd'-**u.u'**; d**u'**+d'**u** + **u**× **u'**)

- Corresponds to rotation composition
- Non commutative, same as rotations

Identity quaternion is (1; **0**)

The conjugate (d; -**u**) is the inverse rotation

- q*q=(1; **0**) for unit quaternions

To apply a rotation to **p**, compute q (0; **p**) q^{-1}

Exponential form

- and application to slerp

Fredo Durand, MIT — Slide 59 — 6.837 Fall 2006

What do we need?

What do we want to do with rotations?

➔What we need to learn to do with quaternions

- ☑ Compose
- ☑ Interpolate & slerp
  - ☑ Multiply by a scalar
  - ☑ Add
- ☑ Invert
- ☑ Apply to a vector
- Convert back and forth
  - ☑ Axis-angle to quaternion
  - ☑ Quaternion to matrix
  - Also Matrix, Euler

Fredo Durand, MIT — Slide 60 — 6.837 Fall 2006

Questions

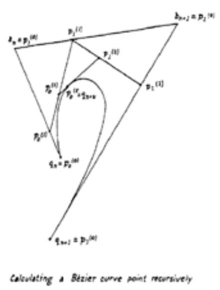
Problem for splines

Slerp only works to interpolate between TWO positions

For splines, we need to interpolate more, typically 4

De Casteljau to the rescue!

De Casteljau construction only interpolates between pairs of points.



Quaternion Interpolation

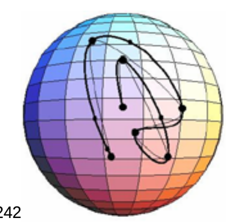
Higher-order interpolations must stay on sphere

See Shoemake, SIGGRAPH '85 for:

- Matrix equivalent of composition
- Details of higher-order interpolation
- More of underlying theory

Problems

- No favored direction (e.g. up for camera)
- Needs more key points to specify multiple rotations



http://portal.acm.org/citation.cfm?id=325242

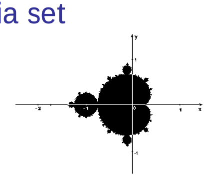
Extensions

Better interpolation

- E.g. minimize acceleration, velocity constraint
- http://www.gg.caltech.edu/STC/rr_sig97.html
- http://portal.acm.org/citation.cfm?id=218486&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864
- http://portal.acm.org/citation.cfm?id=134086&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864

From Kim et al. 1995

Fun: Mandelbrot & Julia set



For each pixel at coordinate c_0

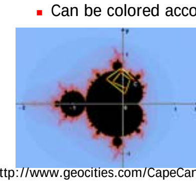
series: $c_{n+1}=c_n^2+c$

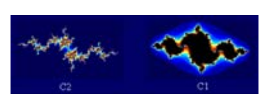
where c is a constant

- same for the whole image for Julia sets
  - One image per c_0
- $c=c_0$ for Mandelbrot set

Set = set of pixels where the series does not diverges

- Can be colored according to how fast it diverges





Julia sets for two different c_0

http://www.geocities.com/CapeCanaveral/2854/mandelbrot.html

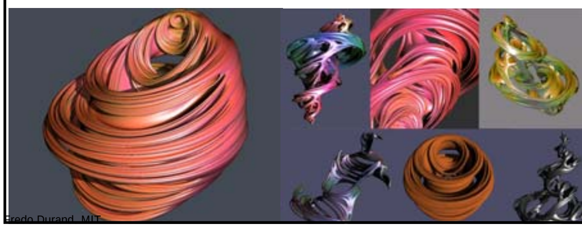
Demo

Fredo Durand, MIT Slide 67 6.837 Fall 2006

Fun: Julia sets in quaternion space

Same kind of series, but quaternions instead of complex
a slice can be visualized in 3D
the full 4D thing leads to nice animations

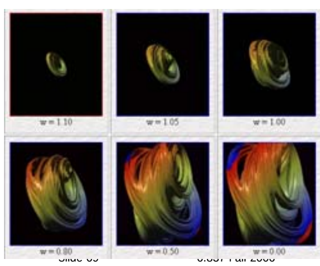
http://www.devmaster.net/forums/showthread.php?t=4448



Fredo Durand, MIT

More

http://local.wasp.uwa.edu.au/~pbourke/fractals/quatjulia/
http://www.geocities.com/CapeCanaveral/2854/
http://www.bluestarfolly.com/art/quaternion.html
http://commons.wikimedia.org/wiki/Julia_set
http://graphics.ucsd.edu/courses/rendering/2005/jkelly/index.html
http://www.mysticfractal.com/



Fredo Durand, MIT

Questions?

Julia Sets in Quaternion space

- http://aleph0.clarku.edu/~djoyce/julia/explorer.html
- Pascal Massimino http://skal.planet-d.net/
- http://www.chaospro.de/gallery/gallery.php?cat=Anim

Fredo Durand, MIT Slide 70 6.837 Fall 2006

Key references

http://portal.acm.org/citation.cfm?id=325242
http://www.gg.caltech.edu/STC/rr_sig97.html
for pdf:
http://portal.acm.org/citation.cfm?id=258870&coll=portal&dl=ACM&CFID=1729050&CFTOKEN=74418864
http://portal.acm.org/citation.cfm?id=134086&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864
http://portal.acm.org/citation.cfm?id=218486&dl=ACM&coll=portal&CFID=1729050&CFTOKEN=74418864

Fredo Durand, MIT Slide 71 6.837 Fall 2006

Links

http://www.euclideanspace.com/maths/geometry/rotations/conversions/eulerToQuaternion/index.htm
http://history.hyperjeff.net/hypercomplex
http://math.hyperjeff.net/hypercomplex/
http://www.gamedev.net/reference/programming/features/qpowers/default.asp
http://www.ogre3d.org/wiki/index.php/Quaternion_and_Rotation_Primer
http://www.sjbrown.co.uk/?article=quaternions
http://www.isner.com/tutorials/quatSpells/quaternion_spells_14.htm
http://www.geometrictools.com/Documentation/KeyframeAnimation.pdf
http://en.wikipedia.org/wiki/Quaternion
http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/
http://local.wasp.uwa.edu.au/~pbourke/fractals/quatjulia/
http://portal.acm.org/citation.cfm?id=325242
http://books.elsevier.com/companions/0120884003/vq/Quaternion-Maps/index.html
http://www.maths.tcd.ie/pub/HistMath/People/Hamilton/Letters/BroomeBridge.html
http://www.akpeters.com/product.asp?ProdCode=1349
http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation
ftp://ftp.cis.upenn.edu/pub/graphics/shoemake/quatut.ps.Z
http://books.elsevier.com/companions/0120884003/vq/index.html
http://www.unpronounceable.com/julia/
http://number-none.com/product/Understanding%20Slerp,%20Then%20Not%20Using%20It/
http://www.gamedev.net/reference/programming/features/qpowers/page7.asp
http://graphics.stanford.edu/courses/cs348c-95-fall/software/quatdemo/

Fredo Durand, MIT Slide 72 6.837 Fall 2006