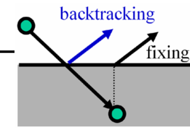
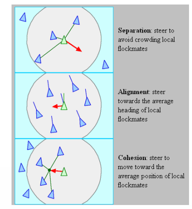
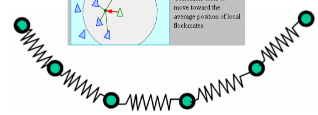
Animation III

Last Time:



- Collision detection & Collision response
- Particle interaction
  - flocking
- Spring-Mass systems
  - chains
  - meshes





Final Projects I

- Do the user interface (if any) last
- Avoid adding lots of new scene file parsing
- Don't expect to understand all the details from some research paper before you start coding
- It's ok to do a simplified implementation (just discuss the limitations in your report)
- Normal lab hours next week
- Additional office hours (just send email!)

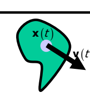
Final Projects II

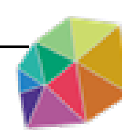
- CSG folks
  → get it to work in your raytracer first
  → pre-visualization second
- Rigid Body folks
  → take baby steps from PS #9
- Distributed Ray Tracing folks
  → looks good
- Have Fun!

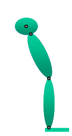
Questions?

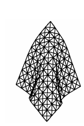
Today

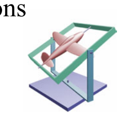
- More Dynamics
  - Rigid Body
  - Fracture
  - Deformation
- Forward & Inverse Kinematics
- Interpolation of Rotations
  - Euler Angles
  - Quaternions

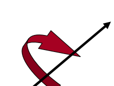






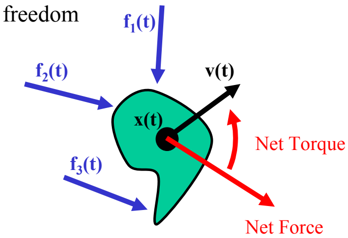






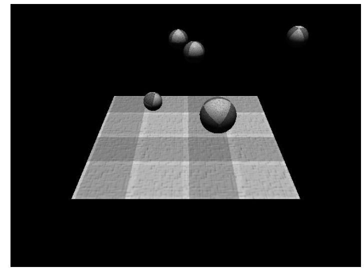
Rigid Body Dynamics

- Could use particles for all points on the object
  - But rigid body does not deform
  - Few degrees of freedom
- Use only one particle at the center of mass
- Compute Net Force & Net Torque



Rigid Body Dynamics

- Physics
  - Velocity
  - Acceleration
  - Angular Momentum
- Collisions
- Friction

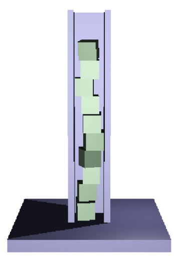


from: Darren Lewis
http://www.stanford.edu/~dalewis/cs448a/rigidbody.html

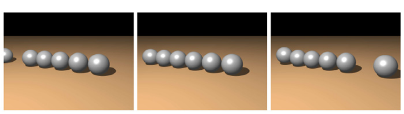
Collisions

Victor J. Milenkovic & Harald Schmidl
Optimization-Based Animation
SIGGRAPH 2001

- We know how to simulate bouncing really well
- But resting collisions are hard to manage



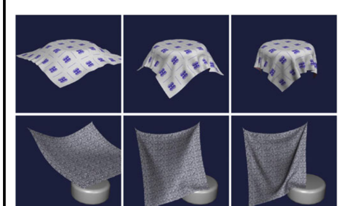
Guendelman, Bridson & Fedkiw
Nonconvex Rigid Bodies with Stacking
SIGGRAPH 2003

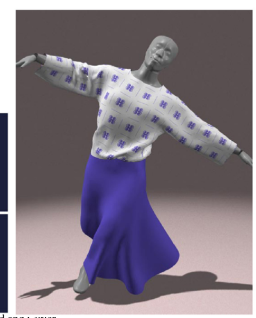


Cloth

David Baraff & Andrew Witkin
Large Steps in Cloth Simulation
SIGGRAPH 1998

- Dynamic motion driven by animation

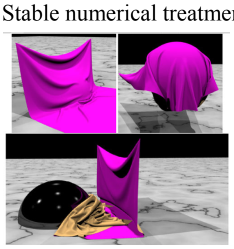




Collisions

Robert Bridson, Ronald Fedkiw & John Anderson
Robust Treatment of Collisions, Contact and Friction for Cloth Animation
SIGGRAPH 2002

- Cloth has many points of contact
- Efficient collision detection
- Stable numerical treatment



Finite Element Method

- To solve the continuous problem (deformation of all points of the object)
  - Discretize the problem
  - Express the interrelationship
  - Solve a big linear system
- More principled than Mass-Spring

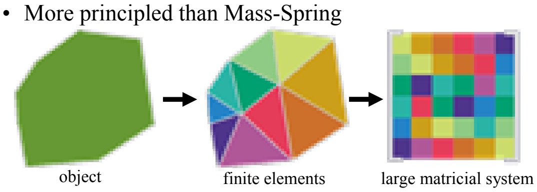
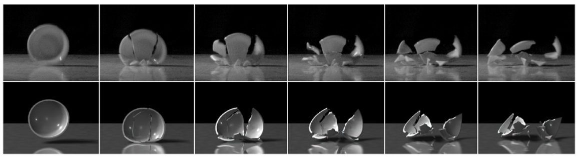


Diagram from Debunne et al. 2001

Fracture

James O'Brien & Jessica Hodgins
Graphical Modeling and Animation of Brittle Fracture
SIGGRAPH 1999

- Fracture threshhold
- Remeshing
  – need connectivity info!
- Material properties
- Parameter tuning



"Half-Edge" Connectivity Data Structure

- For efficiently finding adjacent elements
- Each oriented half edge points to:
  – the oppositely-oriented half edge
  – the next vertex
  – the next half edge
  – the polygonal face
- Many variations…

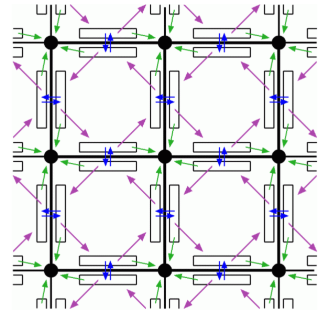
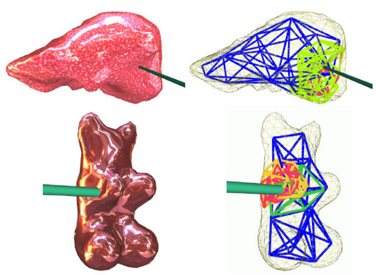


Diagram from Justin Legakis

Level of Detail

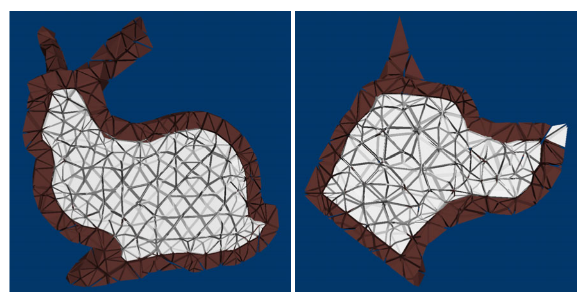
Gilles Debunne , Mathieu Desbrun, Marie-Paule Cani, & Alan H. Barr
Dynamic Real-Time Deformations using Space & Time Adaptive Sampling
SIGGRAPH 2001

- Interactive shape deformation
- Use high-resolution model only in areas of extreme deformation



Multiple Materials

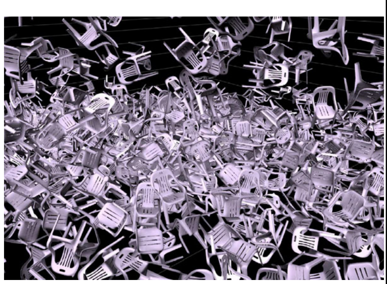
Mueller, Dorsey, McMillan, Jagnow, & Cutler
Stable Real-Time Deformations
Symposium on Computer Animation 2002



Reduced Deformation

Doug L. James & Dinesh K. Pai
BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models
SIGGRAPH 2004

- Collisions are expensive
- Deformation is expensive
- This is a lot of geometry!
- Simplify the simulation model



Fluid Simulation

- Discretize volume of fluid
  – Exchanges and velocity at voxel boundary
- Write Navier Stokes equations
  – Incompressible, etc.
- Numerical integration
  – Finite elements, finite differences
- Challenges:
  – Robust integration, stability
  – Speed
  – Realistic surface

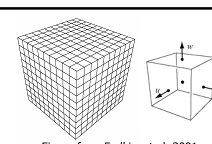


Figure from Fedkiw et al. 2001

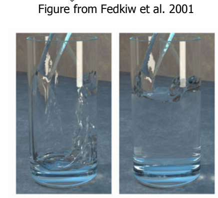
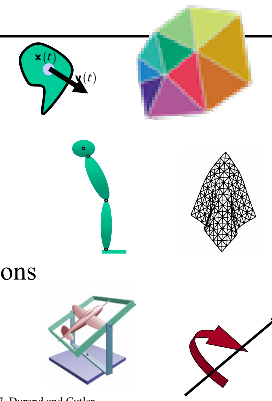


Figure 1: Water being poured into a glass (55x120x55 grid cells).
Figure from Enright et al. 2002

Questions?

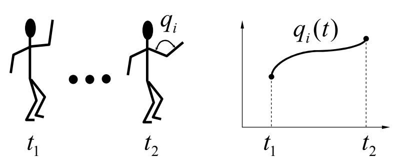
Today

- More Dynamics
  - Rigid Body
  - Fracture
  - Deformation
- Forward & Inverse Kinematics
- Interpolation of Rotations
  - Euler Angles
  - Quaternions



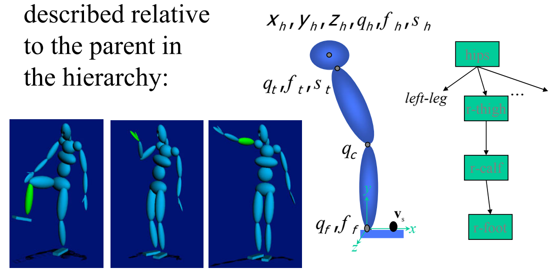
Articulated Models

- Articulated models:
  - rigid parts
  - connected by joints
- They can be animated by specifying the joint angles as functions of time.



Skeleton Hierarchy

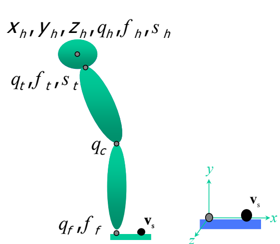
- Each bone transformation described relative to the parent in the hierarchy:



1 DOF: knee 2 DOF: wrist 3 DOF: arm

Forward Kinematics

- Given skeleton parameters p, and the position of the effecter in local coordinates V_s, what is the position of the effector in the world coordinates V_w?



$V_w = T(x_h,y_h,z_h)R(q_h,f_h,s_h)T_hR(q_t,f_t,s_t)T_tR(Q_c)T_cR(q_f,f_f)V_s$

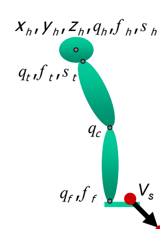
$V_w = S(p)V_s$

Inverse Kinematics (IK)

- Given the position of the effecter in local coordinates V_s and the *desired position* V_w in world coordinates, what are the skeleton parameters p?
- Much harder requires solving the inverse of the non-linear function:

  find p s.t. $S(p)V_s = V_w$

- Underdetermined problem with many solutions



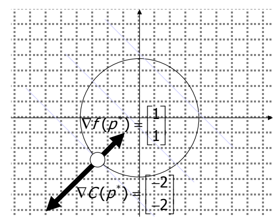
Real IK Problem

- Find a "natural" skeleton configuration for a given collection of pose constraints
- A *scalar objective function g(p)* measures the quality of a pose, g(p) is minimum for most natural poses
  - Example g(p): deviation from natural pose, joint stiffness, power consumption, etc…
- A *vector constraint function C(p) = 0* collects all pose constraints:

Optimization Example

$$p^* = \underset{p}{\arg\min} \quad p_1 + p_2$$

$$\text{s.t.} \quad p_1^2 + p_2^2 = 2$$



Numerical solution using gradient methods:

- Guess initial solution x_0
- Take a step in direction dk:
  - Steepest descent
  - Newton's method
  - Quasi-Newton methods
- Iterate

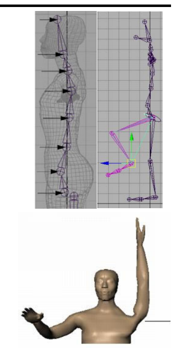
Learn more in 6.839!

Kinematics vs. Dynamics

- Kinematics
  - Describes the positions of body parts as a function of skeleton parameters.
- Dynamics
  - Describes the positions of body parts as a function of applied forces.

How Do They Animate Movies?

- Keyframing mostly
- Articulated figures, inverse kinematics
- Skinning
  - Complex deformable skin, muscle, skin motion
- Hierarchical controls
  - Smile control, eye blinking, etc.
  - Keyframes for these higher-level controls
- A huge time is spent building the 3D models, its skeleton and its controls
- Physical simulation for secondary motion
  - Hair, cloths, water
  - Particle systems for "fuzzy" objects

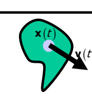


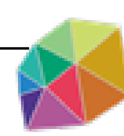
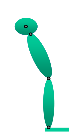
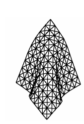
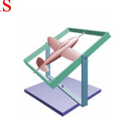
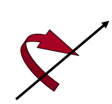
Images from the Maya tutorial

Questions?

Today

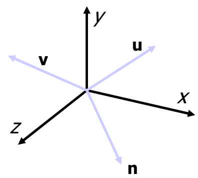
- More Dynamics
  - Rigid Body
  - Fracture
  - Deformation
- Forward & Inverse Kinematics
- Interpolation of Rotations
  - Euler Angles
  - Quaternions



Interpolating Orientations in 3-D

- Rotation matrices
- Given rotation matrices M_i and time t_i, find $M(t)$ such that $M(t_i)=M_i$



$$M = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix}$$

Flawed Solution

- Interpolate each entry independently
- Example: M_0 is identity and M_1 is 90° around x-axis

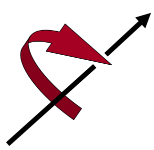
$$\text{Interpolate}\left(\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&0&1\\0&-1&0\end{bmatrix}\right)=\begin{bmatrix}1&0&0\\0&0.5&0.5\\0&-0.5&0.5\end{bmatrix}$$

- Is the result a rotation matrix?

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

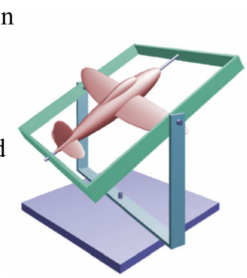
3D Rotations

- How many degrees of freedom for 3D orientations?
- 3 degrees of freedom:
  - direction of rotation and angle
  - or 3 *Euler Angles*



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)

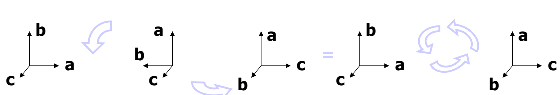


http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

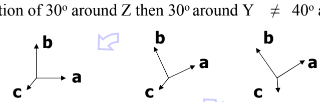
Interpolating Euler Angles

- Natural orientation representation: 3 angles for 3 degrees of freedom
- However, leads to unnatural interpolation:

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

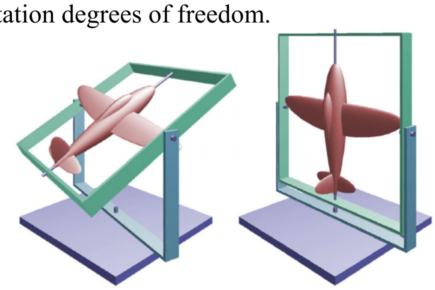


But rotation of 30° around Z then 30° around Y $\neq$ 40° around (1, 1, 1)



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!

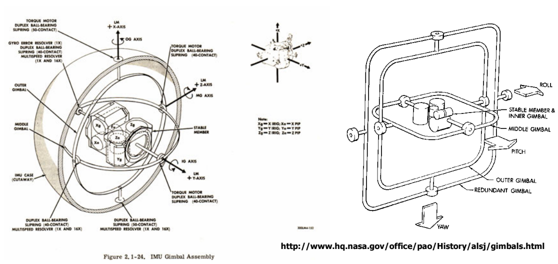
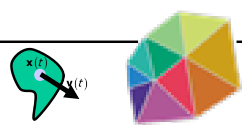


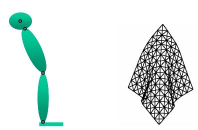
Figure 2.1-24. IMU Gimbal Assembly

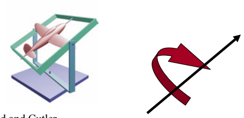
Questions?

Today

- More Dynamics
  - Rigid Body
  - Fracture
  - Deformation
- Forward & Inverse Kinematics
- Interpolation of Rotations
  - Euler Angles
  - Quaternions

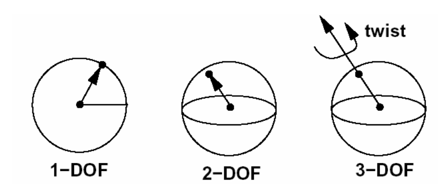






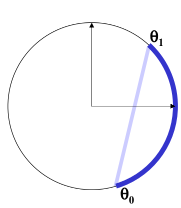
Solution: Quaternion Interpolation

- Interpolate orientation on the unit sphere
- By analogy: 1-, 2-, 3-DOF rotations as constrained points on 1, 2, 3-spheres



1D Sphere and Complex Plane

- Interpolate orientation in 2D
- 1 angle
  - But messy because modulo 2π
- Use interpolation in (complex) 2D plane
- Orientation = complex argument of the number



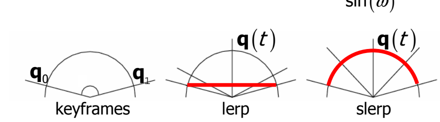
Velocity Issue: *lerp* vs. *slerp*

- Linear Interpolation (*lerp*) interpolates the straight line between the two orientations
  → *lerp* motion does not have uniform velocity:

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

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

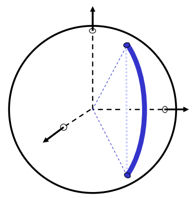
$$\text{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$



interpolate along arc line rather than secant

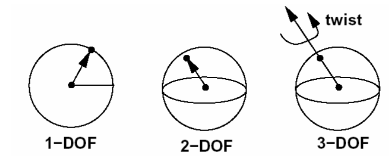
2-Angle Orientation

- Embed 2-sphere in 3D
- 2 angles
  - Messy because modulo 2π and pole
- Use linear interpolation in 3D space
- Orientation = projection onto the sphere
- Use slerp for velocity correction



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

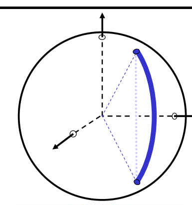


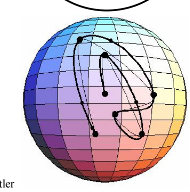
Quaternions

- Due to Hamilton (1843)
- Can be defined like complex numbers
  - a+bi+cj+dk
- Multiplication rules
  - $i^2 = j^2 = k^2 = -1$
  - $ij = k = -ji$
  - $jk = i = -kj$
  - $ki = j = -ik$
- …

Quaternion Principles

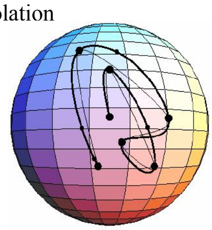
- A quaternion
  = point on unit 3-sphere in 4D
  = orientation
- We can apply it to a point, to a vector, to a ray
- We can convert it to a matrix
- How do we interpolate?
- How do we project?





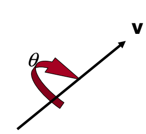
Quaternion Interpolation

- 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



Quaternions

- Quaternions are unit vectors on 3-sphere (in 4D)
  - Right-hand rotation of q radians about v:
    q = {cos(q/2); v sin(q/2)}, → often noted (s, v)
- What if we use -v ?
  - (s,v) = (-s,-v) → a rotation of -q around -v
- What is the quaternion of Identity rotation?
  - q_I = {1, 0, 0, 0}
- Is there exactly one quaternion per rotation?
  - No, q = {cos(q/2); v sin(q/2)} is the same rotation as
    -q = {cos((q+2π)/2); v sin((q+2π)/2)}
  - Antipodal on the quaternion sphere
- What is the inverse of quaternion q = {a, b, c, d}?
  - q^{-1} = {a, -b, -c, -d}



Quaternion Algebra I

- Special rule for quaternion multiplication (composition)
  $q_1 q_2 = \{s_1 s_2 - v_1 \cdot v_2 \, ; \; s_1 v_2 + s_2 v_1 + v_1 \times v_2 \}$

- Sanity check: a radians around v * 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)]}
    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

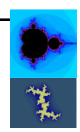
Quaternion Algebra II

- To rotate 3D point/vector p by q, compute:
  $q \; \{0; p\} \; q^{-1}$

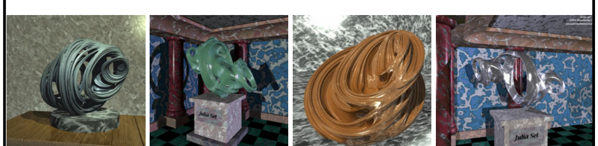
Example: p = (x,y,z)
q = { cos(q/2), 0, 0, sin(q/2) } = {c, 0, 0, s}
q^{-1} = { cos(q/2), 0, 0, -sin(q/2) } = {c, 0, 0, -s}

$q\,\{0; p\}\,q^{-1}$ = {c, 0, 0, s} {0, x, y, z} {c, 0, 0, -s}
= {c*0 – zs ; **cp**+0(0,0,s) + (0,0,s)×**p**} { c,0,0,-s}
= {-zs ; **cp** + (-sy, sx, 0) } { c,0,0,-s}
= {-zsc-(**cp**+(-sy,sx,0))·(0,0,-s) ;
-zs(0,0,-s)+c(**cp**+(-sy,sx,0)) + (**cp**+(-sy,sx,0)) × (0,0,-s) }
= {0, (0,0,zs^2)+c^2**p**+(-csy,csx,0)+(-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}

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



Next Week: Color!