

Hierarchical Modeling



Slide 1

6.837 Fall 2006

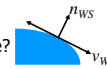
Complement to last time

How do I know I can transform the tangent plane?

- Well, at least it's trivial in the case of polygons: the tangent plane IS the polygon
- For curved surfaces, a formal treatment would require a formal definition of tangent plane. You don't want to go there.
- Intuitively, tangency is about touching at only one point, it's a "location" property, it's simpler than orthogonality.

Was my proof $(M^{-1})^T$ really rigorous?

- Not really: I took shortcuts and swept things under the rug
- In fact, we need to consider **all possible vectors** in the tangent plane
- But the high-level idea is right:
in the dot product $n^T v$, we introduce the identity $M^{-1}M$



Slide 2

6.837 Fall 2006

Transform tangent vector v

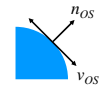
v is perpendicular to normal n :

$$\text{Dot product } n_{OS}^T v_{OS} = 0$$

$$n_{OS}^T (M^{-1} M) v_{OS} = 0$$

$$(n_{OS}^T M^{-1}) (M v_{OS}) = 0$$

$$(n_{OS}^T M^{-1}) v_{WS} = 0$$

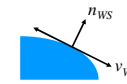


v_{WS} is perpendicular to normal n_{WS} :

$$n_{WS}^T = n_{OS}^T (M^{-1})$$

$$n_{WS}^T = (M^{-1})^T n_{OS}$$

$$n_{WS}^T v_{WS} = 0$$



Slide 3

6.837 Fall 2006

Plan

Hierarchical Modeling
OpenGL matrix stack
Scene graph
Forward kinematics
Inverse Kinematics
Interpolating rotations

Slide 4

6.837 Fall 2006

Hierarchical Modeling

Triangles, parametric curves and surfaces are the building blocks from which more complex real-world objects are modeled.

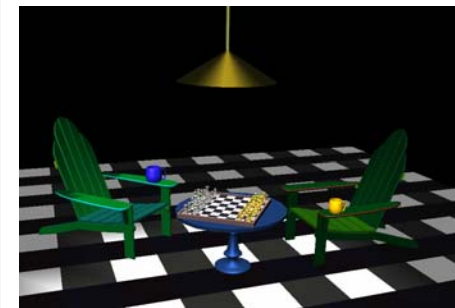
Hierarchical modeling creates complex real-world objects by combining simple primitive shapes into more complex aggregate objects.



Slide 5

6.837 Fall 2006

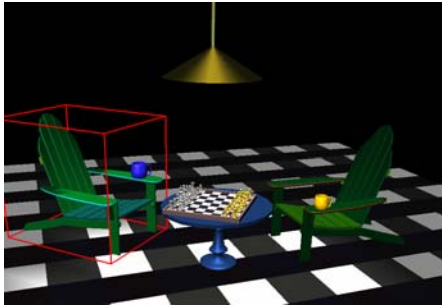
Hierarchical models



Slide 6

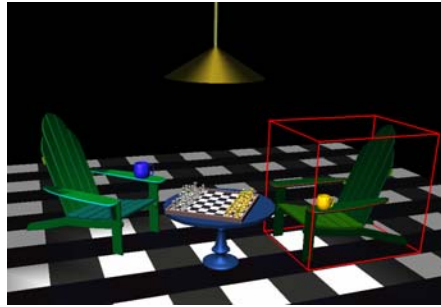
6.837 Fall 2006

Hierarchical models



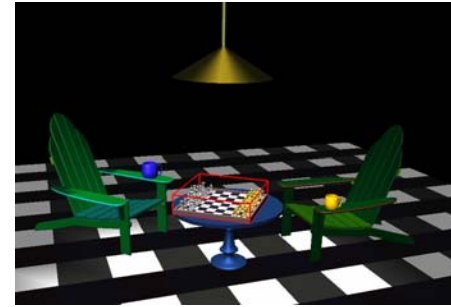
Slide 7 6.837 Fall 2006

Hierarchical models



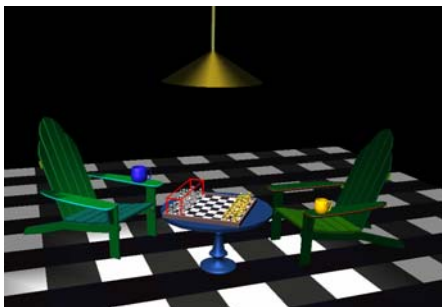
Slide 8 6.837 Fall 2006

Hierarchical models



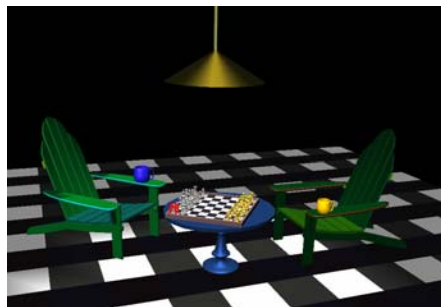
Slide 9 6.837 Fall 2006

Hierarchical models



Slide 10 6.837 Fall 2006

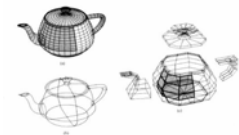
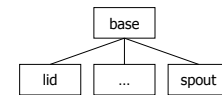
Hierarchical models



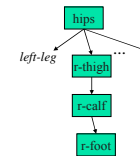
Slide 11 6.837 Fall 2006

Examples

Curves and surfaces



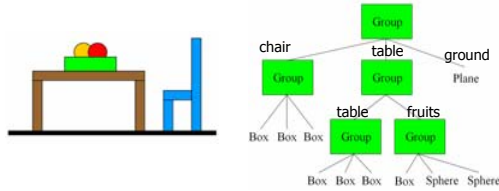
Animated Characters



Slide 12 6.837 Fall 2006

Hierarchical Grouping of Objects

Logical organization of scene



6.837 Fall 2006

Simple Example with Groups

```

group {
  numObjects 3
  group {
    numObjects 3
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
  }
  group {
    numObjects 2
    group {
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
    }
    group {
      Box { <BOX PARAMS> }
      Sphere { <SPHERE PARAMS> }
      Sphere { <SPHERE PARAMS> }
    }
  }
  Plane { <PLANE PARAMS> }
}

```

6.837 Fall 2006

Adding Materials

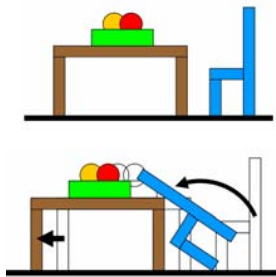
```

Group {
  numObjects 3
  Material { <BLUE> }
  Group {
    numObjects 3
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
    Box { <BOX PARAMS> }
  }
  Group {
    numObjects 2
    Material { <BROWN> }
    Group {
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
    }
    Group {
      Material { <GREEN> }
      Box { <BOX PARAMS> }
      Material { <RED> }
      Sphere { <SPHERE PARAMS> }
      Material { <ORANGE> }
      Sphere { <SPHERE PARAMS> }
    }
  }
  Plane { <BLACK> }
}

```

6.837 Fall 2006

Adding Transformations

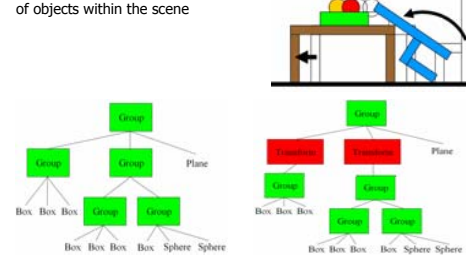


Slide 16

6.837 Fall 2006

Hierarchical Transformation of Objects

Transforms position logical groupings of objects within the scene



Slide 17

6.837 Fall 2006

Simple Example with Transforms

```

Group {
  numObjects 3
  Transform {
    ZRotate { 45 }
    Group {
      numObjects 3
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
      Box { <BOX PARAMS> }
    }
  }
  Transform {
    Translate { -2 0 0 }
    Group {
      numObjects 2
      Group {
        numObjects 3
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> }
        Box { <BOX PARAMS> }
      }
      Group {
        numObjects 2
        Box { <BOX PARAMS> }
        Sphere { <SPHERE PARAMS> }
        Sphere { <SPHERE PARAMS> }
      }
    }
  }
  Plane { <PLANE PARAMS> }
}

```

6.837 Fall 2006

Separating types of transformation

Note that we have treated translations, rotations, etc. as separate
But they are all represented by 4x4 matrices and there is no technical reason not to combine them into the resulting matrix
It's just simpler for the human programmer, and corresponds to the handle of 3D modeling/animation packages

Slide 19 6.837 Fall 2006

Demo applet

[http://www.cs.brown.edu/exploratories/...](http://www.cs.brown.edu/exploratories/)

Slide 20 6.837 Fall 2006

Questions?

Slide 21 6.837 Fall 2006

Plan

Hierarchical Modeling
OpenGL matrix stack
Scene graph
Forward kinematics
Inverse Kinematics
Interpolating rotations

Slide 22 6.837 Fall 2006

Hierarchical modeling in OpenGL

Commands to change current transformation
▪ glTranslate, glScale, etc.
Affects the **state**, i.e. all following commands will undergo this transformation
Utilities to maintain a matrix stack (to revert to previous state)
Difference between model and view matrix

Slide 23 6.837 Fall 2006

Model vs. Projection matrix

It is almost the same to rotate the camera or the objects
Main difference:
▪ Lighting
This is why OpenGL has two transforms: model and projection
`glMatrixMode(GL_MODELVIEW);`
Tells OpenGL that next matrix commands deal with the objects
typically used for modeling & animation
`glMatrixMode(GL_PROJECTION);`
Tells OpenGL we deal with the camera
typically used to change viewpoint & focal length

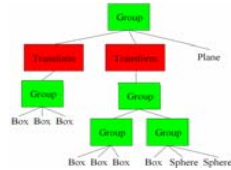
Slide 24 6.837 Fall 2006

Managing the state

To reset everything: `glLoadIdentity();`

OpenGL stores a stack of matrices

- You don't need to remember, OpenGL remembers
- `glPushMatrix()`
- `glPopMatrix`
- Typical use: push matrix when you start rendering a group
- Pop once you are done
- Geeky detail: the max stack size is typically bigger for... ModelView



Slide 25

6.837 Fall 2006

Questions?

Slide 26

6.837 Fall 2006

Plan

Hierarchical Modeling
OpenGL matrix stack
Scene graph
Forward kinematics
Inverse Kinematics
Interpolating rotations

Slide 27

6.837 Fall 2006

Scene Graph

Convenient Data structure for scene representation

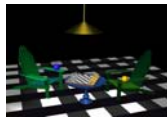
- Transformations
- Materials, color
- Multiple instances

Basic idea: Hierarchical Tree

- Useful for manipulation/animation
- Especially for articulated figures

Useful for rendering too

- Ray tracing acceleration, occlusion culling



Slide 28

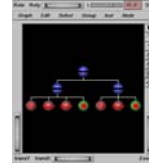
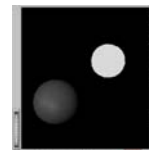
6.837 Fall 2006

Scene Graph Representation

Basic idea: Tree

Comprised of several node types:

- Shape: 3D geometric objects
- Transform: Affect current transformation
- Property: Appearance, texture, etc.
- Group: Collection of subgraphs



Slide 29

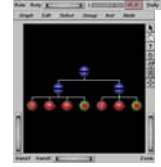
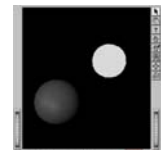
6.837 Fall 2006

Scene Graph Representations

In fact, generalization of a tree
Directed Acyclic Graph (DAG)

Why?

- Allows multiple instantiations
- Cycle forbidden
 - because infinite recursions



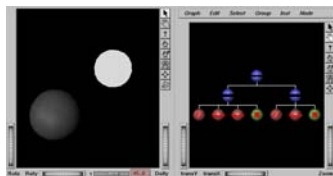
Slide 30

6.837 Fall 2006

Traversal

Depth first

- Top to bottom, left to right



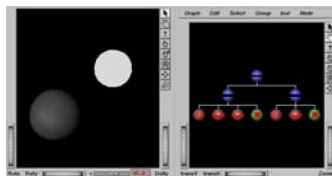
Slide 31

6.837 Fall 2006

Traversal State

The State is updated during traversal

- Transformations, properties
- Influence of nodes can be complex
- E.g. bottom to top



Slide 32

6.837 Fall 2006

Overview of Inventor

A suite of tools

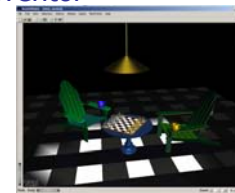
- Viewer(s)
- Utilities

An API

- C++ set of classes for 3D display and manipulation

File format

- ASCII or binary
- Later became VRML
 - Virtual Reality Modeling Language
 - Internet 3D format



Originally SGI (Silicon Graphics)

- <http://portal.acm.org/citation.cfm?id=166127>

Now TGS

- http://www.tgs.com/pro_div/oiv_main.htm

Slide 33

6.837 Fall 2006

Questions?

Slide 34

6.837 Fall 2006

Plan

Hierarchical Modeling
OpenGL matrix stack
Scene graph
Forward Kinematics
Inverse Kinematics
Interpolating rotations

Slide 35

6.837 Fall 2006

Animation

Hierarchical structure is essential for animation

- Eyes move with head
- Hands move with arms
- Feet move with legs
- ...

Without such structure the model falls apart:



Slide 36

6.837 Fall 2006

Forward Kinematics

Describes the positions of the body parts as a function of the joint angles.

Each joint is characterized by its degrees of freedom (dof)

- Usually rotation for articulated bodies

1 DOF: knee



2 DOF: wrist



3 DOF: arm

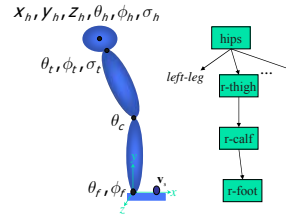


Slide 37

6.837 Fall 2006

Skeleton Hierarchy

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

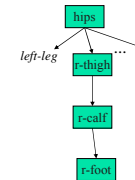


Derive world coordinates V_w for a point with local coordinates V_s ?

Slide 38

6.837 Fall 2006

Draw by Traversing a Tree



```

glLoadIdentity();
glPushMatrix();
glTranslatef(...);
glRotate(...);
drawHips();
glPopMatrix();
glPushMatrix();
glTranslatef(...);
glRotate(...);
drawThigh();
glTranslatef(...);
glRotate(...);
drawCalf();
glTranslatef(...);
glRotate(...);
drawFoot();
glPopMatrix();
left-leg
    
```

Assumes drawing procedures for thigh, calf, and foot use joint positions as the origin for a drawing coordinate frame.

Slide 39

6.837 Fall 2006

Forward Kinematics

$x_{h1}, y_{h1}, z_{h1}, \theta_{h1}, \phi_{h1}, \sigma_{h1}$

Transformation matrix for a point v_c is a matrix composition of all joint transformation between the point and the root of the hierarchy.

Note that the natural parameters of the degrees of freedom (e.g. angle) have a non-linear effect

$$v_w = T(x_h, y_h, z_h) R(\theta_h, \phi_h, \sigma_h) TR(\theta_t, \phi_t, \sigma_t) TR(\theta_c, \phi_c, \sigma_c) v_s$$

$$v_w = S \left(\underbrace{x_{h1}, y_{h1}, z_{h1}, \theta_{h1}, \phi_{h1}, \sigma_{h1}, \theta_{t1}, \phi_{t1}, \sigma_{t1}, \theta_{c1}, \phi_{c1}, \sigma_{c1}}_p \right) v_s = S(p) v_s$$

Slide 40

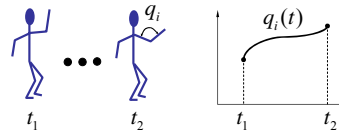
6.837 Fall 2006

Articulated Models

Articulated models:

- rigid parts
- connected by joints

They can be animated by specifying the joint angles as functions of time.



Slide 41

6.837 Fall 2006

Procedural Animation

A stopwatch with second and minute hands. Hands rotate together as a function of time. The hands are animated by varying the time parameter.



Slide 42

6.837 Fall 2006

Questions?

Slide 43 6.837 Fall 2006

Plan

Hierarchical Modeling
 OpenGL matrix stack
 Scene graph
 Forward kinematics
Inverse Kinematics
 Interpolating rotations

Slide 44 6.837 Fall 2006

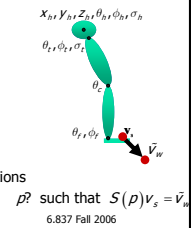
Inverse Kinematics

Forward Kinematics

- Given the skeleton parameters (position of the root and the joint angles) ρ and the position of the point in local coordinates v_s , what is the position of the point in the world coordinates v_w ?
- Not too hard, we can solve it by evaluating $S(\rho)v_s$

Inverse Kinematics

- Given the position of the point in local coordinates v_s and the desired position \vec{v}_w in world coordinates, what are the skeleton parameters ρ ?
- Much harder requires solving the inverse of the non-linear function:
- Underdetermined problem with many solutions



Slide 45 6.837 Fall 2006

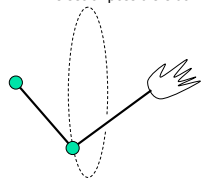
It's underconstrained

e.g., count degrees of freedom:

- we specify one point (3 equations)
- We usually need more than 3 angles

Simple geometric example (in 3D):
 specify hand position, need elbow & shoulder

- The set of possible elbow location is a circle in 3D



Slide 46 6.837 Fall 2006

How to tackle these problems?

Deal with non-linearity: Iterative solution (steepest descent)

- Compute Jacobian of position wrt angles
- Good news: in an interactive context, we want to move e.g. the hand and we can use the previous frame as an initial guess

Deal with ill-posedness: Pseudo-inverse (18.06!)

- Solution that displaces things the least

Deal with ill-posedness: Prior on "good pose"

- More advanced

Additional potential issues: bounds on joint angles, etc.

Slide 47 6.837 Fall 2006

Questions?

Slide 48 6.837 Fall 2006

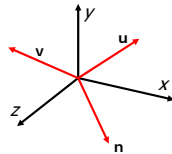
Plan

- Hierarchical Modeling
- OpenGL matrix stack
- Scene graph
- Forward kinematics
- Inverse Kinematics
- Interpolating rotations**

Slide 49 6.837 Fall 2006

Interpolating Orientations in 3-D

Critical in particular for camera animation
 Rotation matrices
 Given rotation matrices M_0 and time t_i ,
 find $M(t)$ such that $M(t_i)=M_i$



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

Slide 50 6.837 Fall 2006

Flawed Solution

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

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

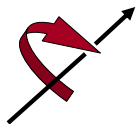
Is the result a rotation matrix?
 No, it does not preserve rigidity
 (angles and lengths)

Slide 51 6.837 Fall 2006

3D Rotations

How many degrees of freedom for 3D orientations?
 3 degrees of freedom:

- direction of rotation and angle
- or 3 *Euler Angles*



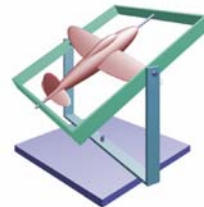
Slide 52 6.837 Fall 2006

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)



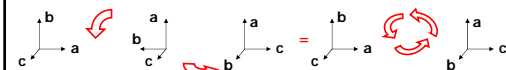
<http://www.fho-emsden.de/~hoffmann/jmbal09082002.pdf>

Slide 53 6.837 Fall 2006

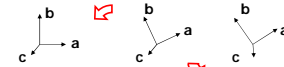
Interpolating Euler Angles

Natural orientation representation:
 interpolate independently 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)



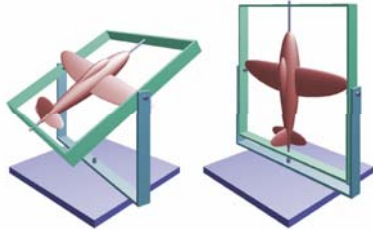
But rotation of 30° around Z then 30° around Y ≠ 40° around (1, 1, 1)



Slide 54 6.837 Fall 2006

Gimbal Lock

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



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

Slide 55 6.837 Fall 2006

Euler Angles in the Real World

Apollo inertial measurement unit
To "prevent" lock, they added a fourth Gimbal!

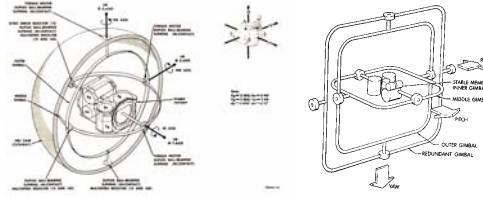


Figure 3.1.10. IMU Gimbal Assembly

<http://www.hq.nasa.gov/office/pao/History/alsj/gimbals.html>
Slide 56 6.837 Fall 2006

Questions?

Slide 57

6.837 Fall 2006

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



1-DOF

2-DOF

3-DOF

Slide 58

6.837 Fall 2006

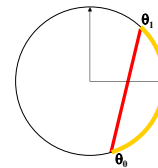
1D Sphere and Complex Plane

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 \Leftrightarrow complex multiplication
 - Trivial with exponential notation $re^{i\theta}$

Slide 59

6.837 Fall 2006

Velocity Issue: *lerp* vs. *slerp*

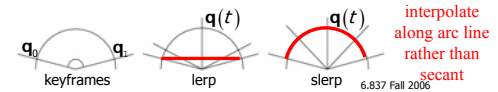
Linear Interpolation (*lerp*) interpolates the straight line between the two orientations

\rightarrow *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:

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



Slide 60

6.837 Fall 2006

2-Angle Orientation

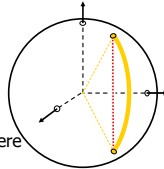
2 angles

- 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

Slide 61

6.837 Fall 2006

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



1-DOF

2-DOF

3-DOF

Slide 62

6.837 Fall 2006

Quaternions

Due to Hamilton (1843)

Can be defined like complex numbers

- $a+bi+cj+dk$

Multiplication rules

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

... More next time

Slide 63

6.837 Fall 2006