CS 445 / 645
Introduction to Computer Graphics

Lecture 21

Representing Rotations

University of Virginia

Parameterizing Rotations

Straightforward in 2D

- A scalar, θ, represents rotation in plane

More complicated in 3D

- Three scalars are required to define orientation
- Note that three scalars are also required to define position
- Objects free to translate and tumble in 3D have 6 degrees of freedom (DOF)

Representing 3 Rotational DOFs

3x3 Matrix (9 DOFs)

- Rows of matrix define orthogonal axes

Euler Angles (3 DOFs)

- Rot x + Rot y + Rot z

Axis-angle (4 DOFs)

- Axis of rotation + Rotation amount

Quaternion (4 DOFs)

- 4 dimensional complex numbers

Rotation Matrix

9 DOFs must reduce to 3

Rows must be unit length (-3 DOFs)

Rows must be orthogonal (-3 DOFs)

Drifting matrices is very bad

- Numerical errors results when trying to gradually rotate matrix by adding derivatives
- Resulting matrix may scale / shear
- Gram-Schmidt algorithm will re-orthogonalize your matrix

Difficult to interpolate between matrices

- How would you do it?

Euler Angles

(θ_x, θ_y, θ_z) = $R_z R_y R_x$

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

Axis order is not defined

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

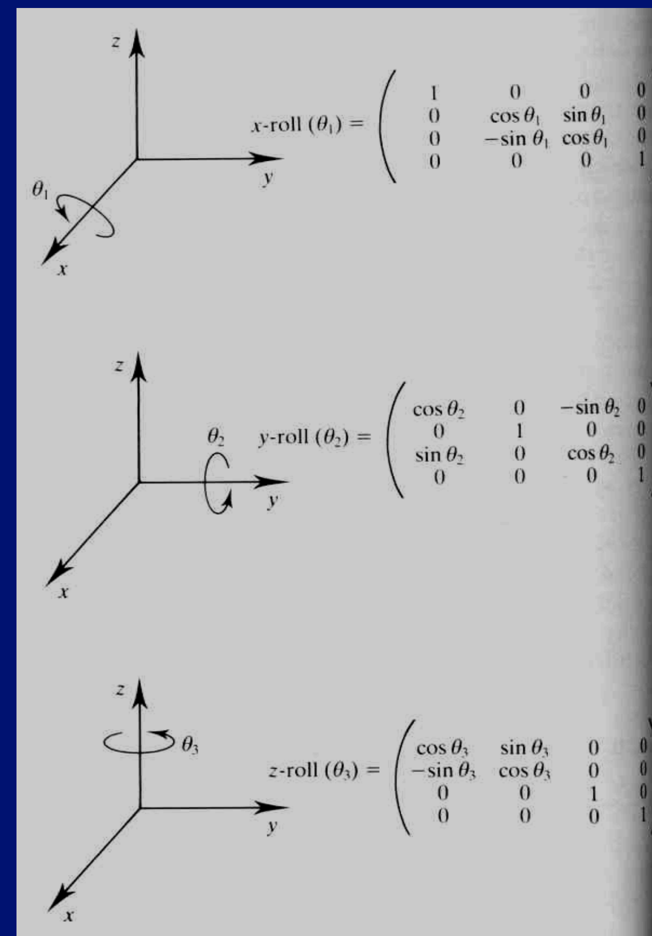


Figure 15.15 The principal rotation matrices.

Euler Angles

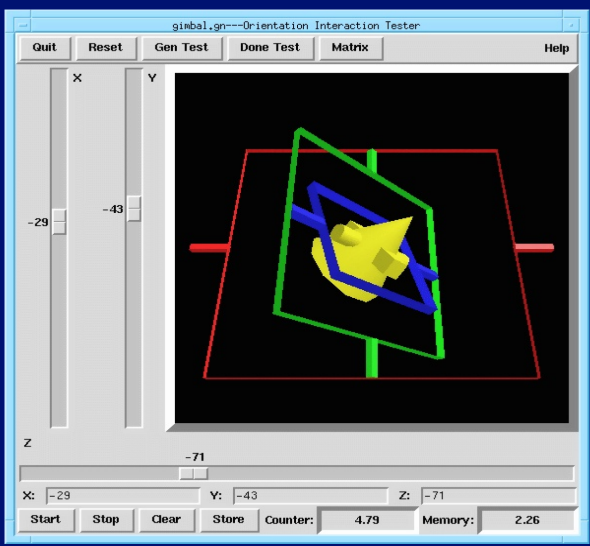
Rotations not uniquely defined

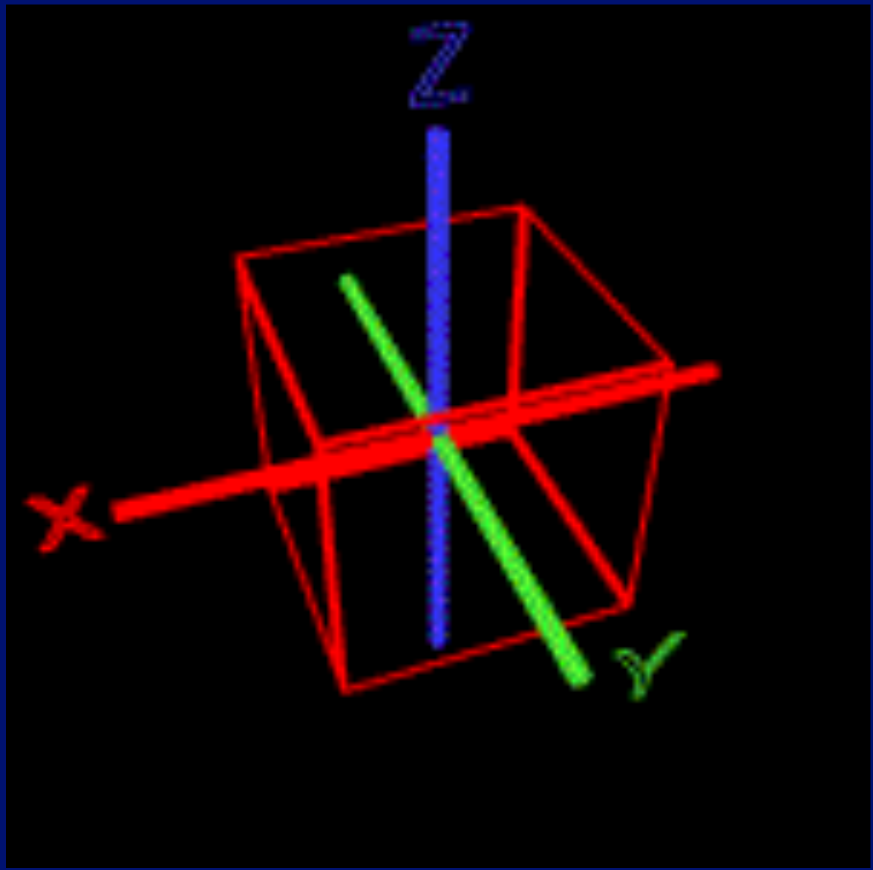
- ex: (z, x, y) = (90, 45, 45) = (45, 0, -45)
  takes positive x-axis to (1, 1, 1)
- Cartesian coordinates are independent of one another, but Euler angles are not
- Remember, the axes stay in the same place during rotations

Gimbal Lock

- Term derived from mechanical problem that arises in gimbal mechanism that supports a compass or a gyro

Gimbal Lock





http://www.anticz.com/eularqua.htm

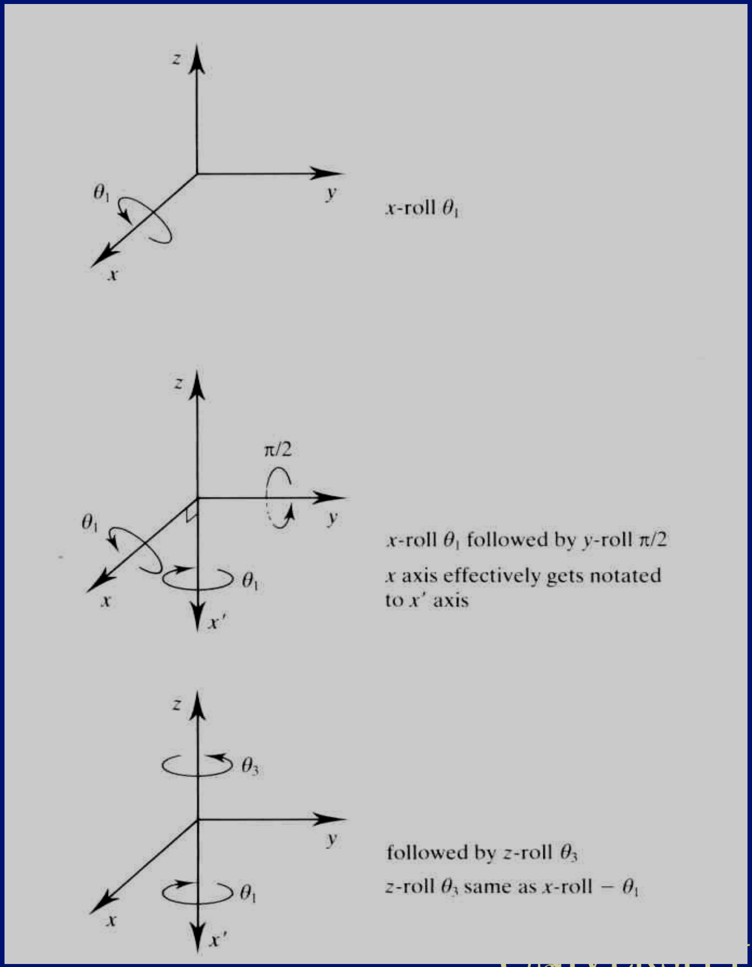
University of Virginia

Gimbal Lock

Occurs when two axes are aligned

Second and third rotations have effect of transforming earlier rotations

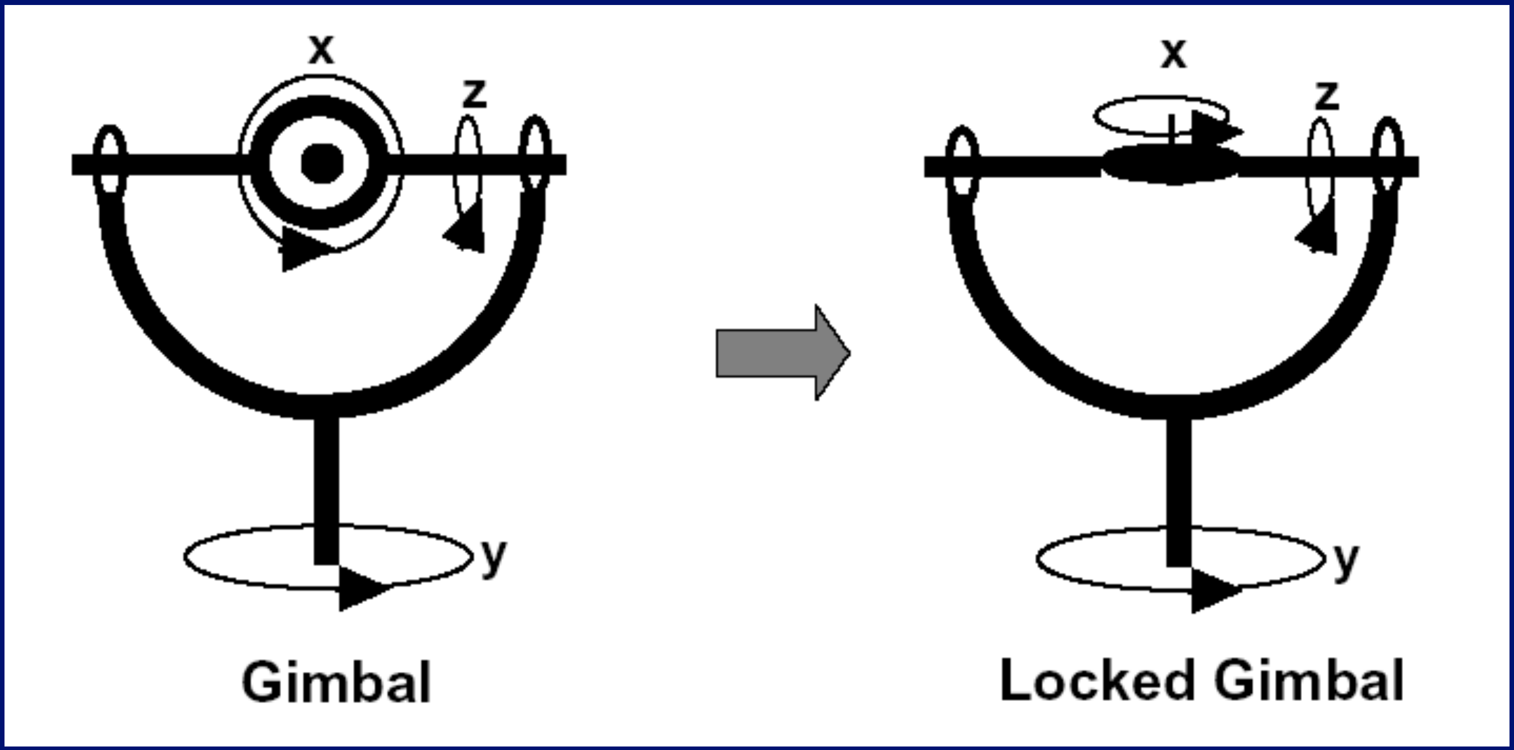
- ex: Rot x, Rot y, Rot z
  - If Rot y = 90 degrees, Rot z == -Rot x



UNIVERSITY of VIRGINIA

A Gimbal

Hardware implementation of Euler angles (used for mounting gyroscopes and globes)



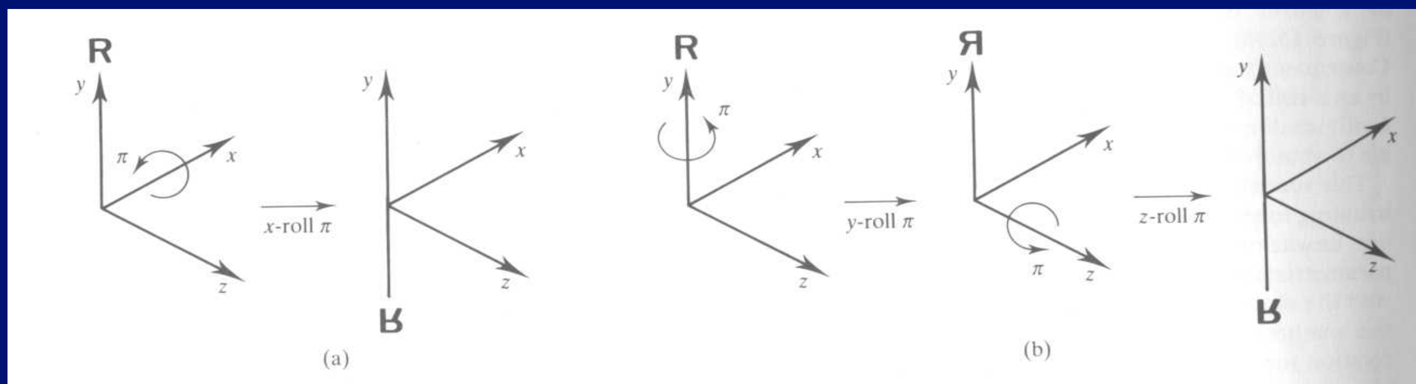
Interpolation

Interpolation between two Euler angles is not unique

ex: (x, y, z) rotation

- (0, 0, 0) to (180, 0, 0) vs. (0, 0, 0) to (0, 180, 180)
- Interpolation about different axes are not independent

Interpolation



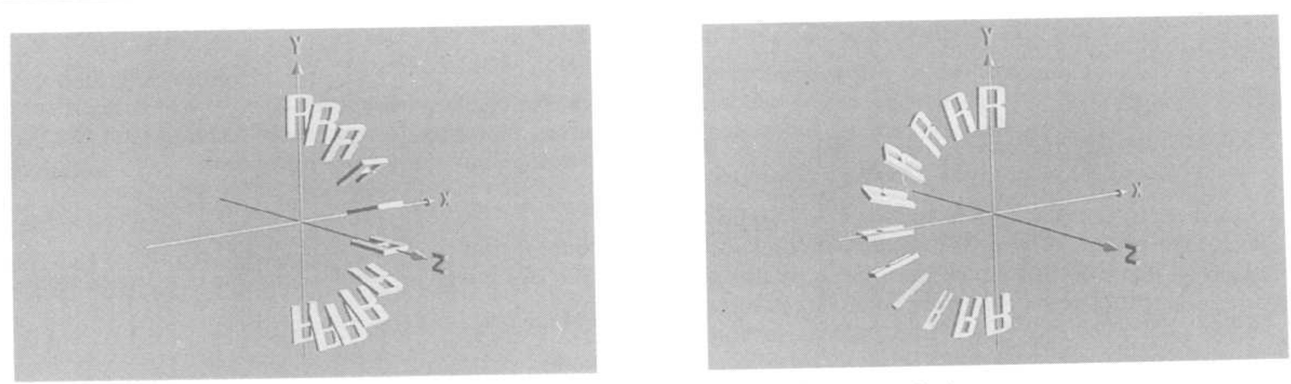


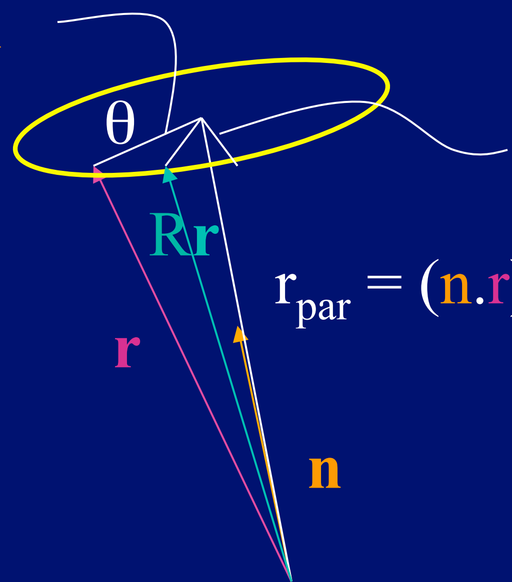
Figure 15.19 Euler angle parametrization. (a) A single x-roll of π. (b) A y-roll of π followed by a z-roll of π.

Axis-angle Notation

- *Define an axis of rotation (x, y, z) and a rotation about that axis, θ: $R(\theta, n)$*
- *4 degrees of freedom specify 3 rotational degrees of freedom because axis of rotation is constrained to be a unit vector*

Axis-angle Notation

$r_{perp} = r - (n.r)\, n$



$V = n \times (r - (n.r)\, n) = n \times r$

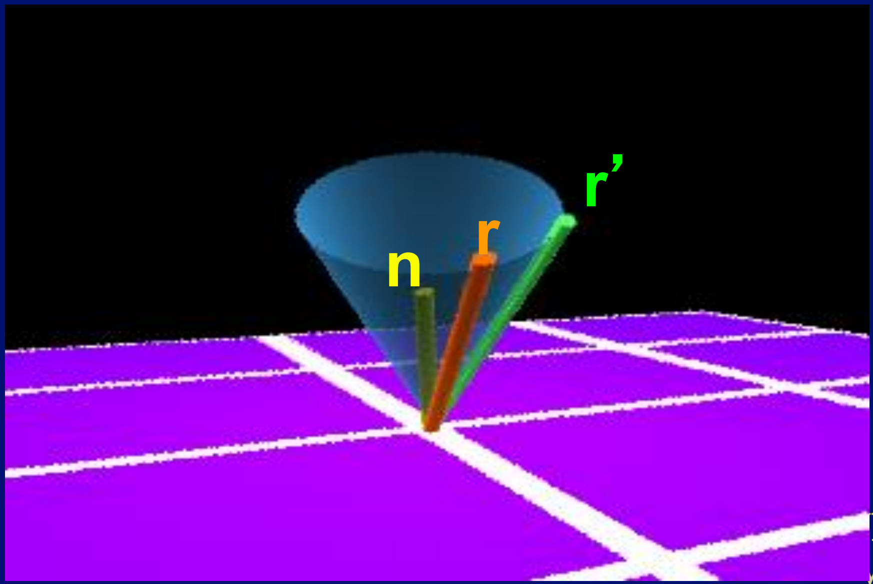
$r_{par} = (n.r)\, n$

$$\begin{aligned}
Rr &= Rr_{par} + Rr_{perp} \\
&= Rr_{par} + (\cos\theta)\, r_{perp} + (\sin\theta)\, V \\
&= (n.r)\, n + \cos\theta(r - (n.r)n) + (\sin\theta)\, n \times r \\
&= (\cos\theta) r + (1 - \cos\theta)\, n\, (n.r) + (\sin\theta)\, n \times r
\end{aligned}$$

Axis-angle Rotation

Given
- r – Vector in space to rotate
- n – Unit-length axis in space about which to rotate
- θ – The amount about n to rotate

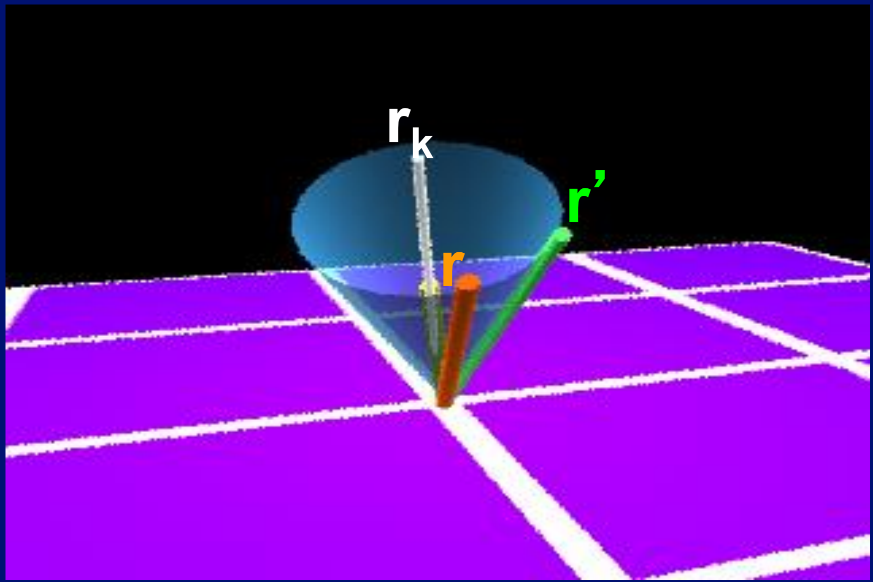
Solve
- r’ – The rotated vector



Axis-angle Rotation

Step 1

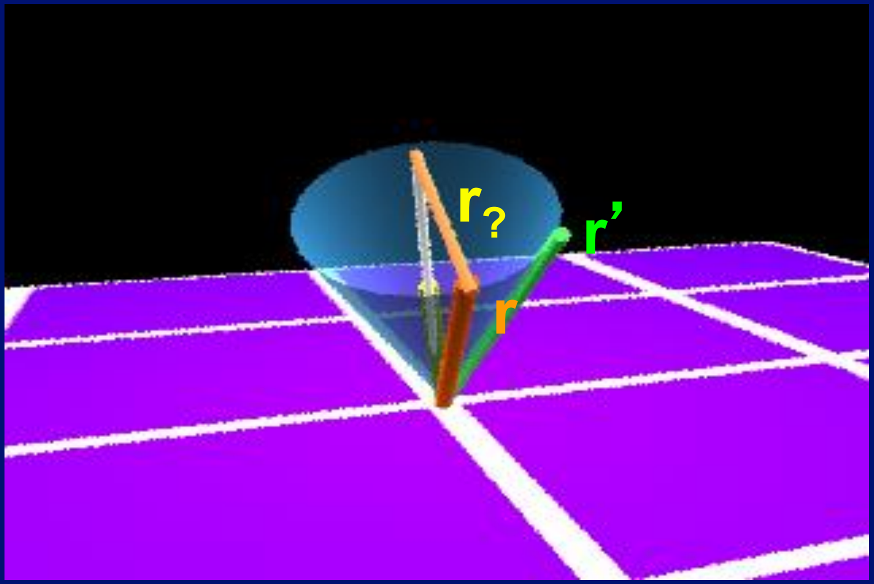
- Compute r_k an extended version of the rotation axis, n
- $r_k = (n ¢ r)\ n$



Axis-angle Rotation

Compute $r_?$

$$r_? = r - (n \cent r)\, n$$

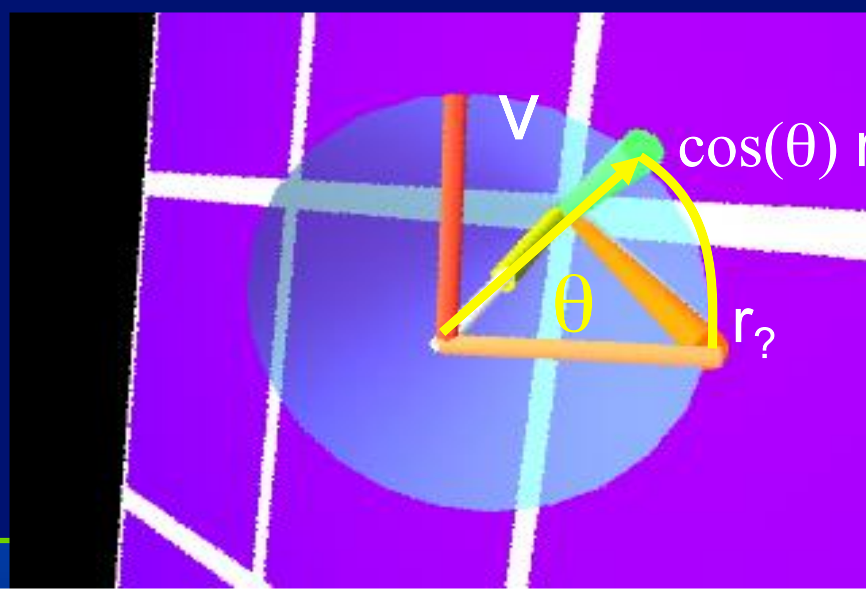


Axis-angle Rotation

Compute v, a vector perpendicular to r_k and $r_?$

v = r_k £ $r_?$

Use v and $r_?$ and θ to compute r'



Axis-angle Notation

No easy way to determine how to concatenate many axis-angle rotations that result in final desired axis-angle rotation

No simple way to interpolate rotations

Quaternion

Remember complex numbers: a + ib

- Where $i^2 = -1$

Invented by Sir William Hamilton (1843)

- Remember Hamiltonian path from Discrete II?

Quaternion:

- $Q = a + bi + cj + dk$
  - Where $i^2 = j^2 = k^2 = -1$ and $ij = k$ and $ji = -k$
- Represented as: $q = (s, \mathbf{v}) = s + v_x i + v_y j + v_z k$

UNIVERSITY *of* VIRGINIA

Quaternion

A quaternion is a 4-D unit vector q = [x y z w]

- It lies on the unit hypersphere $x^2 + y^2 + z^2 + w^2 = 1$

For rotation about (unit) axis v by angle θ

- vector part = $(\sin \theta/2)\ v$ = *[x y z]*
- scalar part = $(\cos \theta/2)$ = *w*
- $(\sin(\theta/2)\ n_x, \sin(\theta/2)\ n_y, \sin(\theta/2)\ n_z, \cos(\theta/2))$

Only a unit quaternion encodes a rotation - normalize

Quaternion

Rotation matrix corresponding to a quaternion:

- [x y z w] = $$\begin{bmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy \\ 2xy-2wz & 1-2x^2-2z^2 & 2yz+2wx \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 \end{bmatrix}$$

Quaternion Multiplication

- $q_1 * q_2 = [\mathbf{v_1}, w_1] * [\mathbf{v_2}, w_2] = [(w_1 v_2 + w_2 v_1 + (v_1 \times v_2)), w_1 w_2 - v_1 . v_2]$
- quaternion * quaternion = quaternion
- this satisfies requirements for mathematical *group*
- Rotating object twice according to two different quaternions is equivalent to one rotation according to product of two quaternions

UNIVERSITY of VIRGINIA

Quaternion Example

X-roll of π

- (cos (π/2), sin (π/2) (1, 0, 0)) = (0, (1, 0, 0))

Y-roll 0f π

- (0, (0, 1, 0))

Z-roll of π

- (0, (0, 0, 1))

R_y (π) followed by R_z (π)

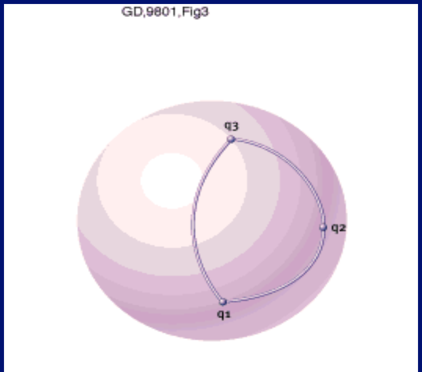
- (0, (0, 1, 0) times (0, (0, 0, 1)) = (0, (0, 1, 0) x (0, 0, 1)
  = (0, (1, 0, 0))

UNIVERSITY of VIRGINIA

Quaternion Interpolation

Biggest advantage of quaternions

- Interpolation
- Cannot linearly interpolate between two quaternions because it would speed up in middle
- Instead, Spherical Linear Interpolation, slerp()
- Used by modern video games for third-person perspective
- Why?



UNIVERSITY of VIRGINIA

SLERP

Quaternion is a point on the 4-D unit sphere

- interpolating rotations requires a unit quaternion at each step
  - another point on the 4-D unit sphere
- move with constant angular velocity along the great circle between two points

Any rotation is defined by 2 quaternions, so pick the shortest SLERP

To interpolate more than two points, solve a non-linear variational constrained optimization

- Ken Shoemake in SIGGRAPH '85 (www.acm.org/dl)

UNIVERSITY of VIRGINIA

Quaternion Interpolation

- ***Quaternion (white) vs. Euler (black) interpolation***
- ***Left images are linear interpolation***
- ***Right images are cubic interpolation***

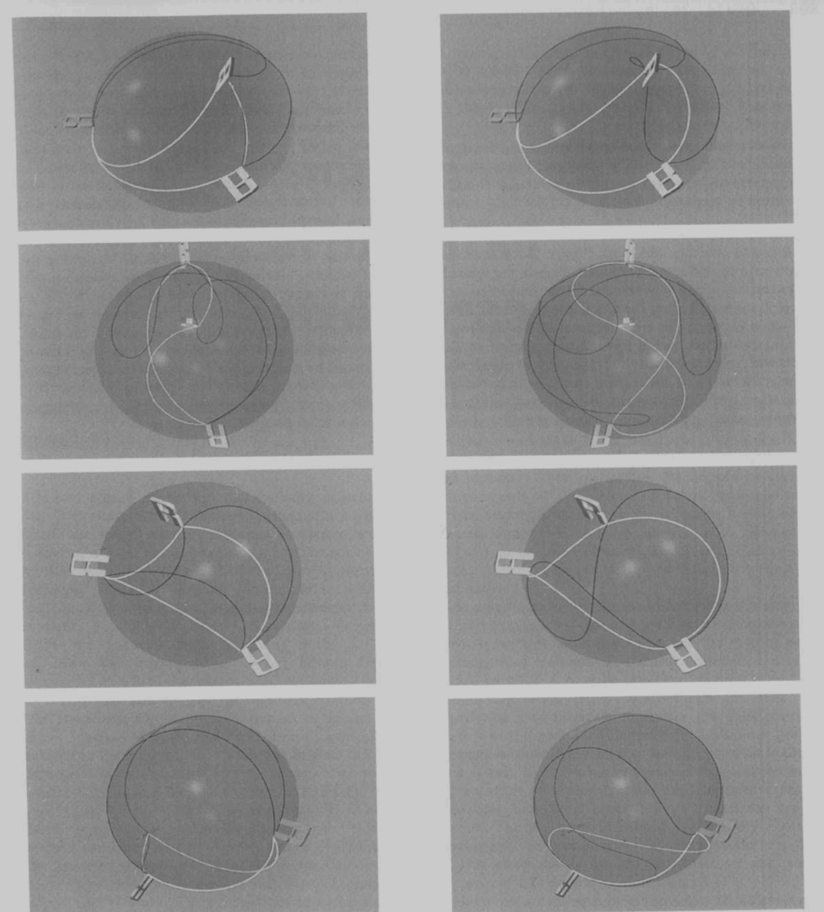


Figure 15.25 Shows how R moves through the three keys. In all cases the white line tracks the motion of R when the interpolation is carried out in quaternion space; the black line tracks the motion of R when Euler angles are interpolated. In each row the left illustration compares linear interpolation of Euler angles with spherical linear interpolation of quaternions. In each row the right illustration compares a cubic spline interpolation of Euler angles to the spherical cubic spline interpolation of quaternions (using squad( )).

Quaternion Code

http://www.gamasutra.com/features/programming/19980703/quaternions_01.htm

- Registration required

Camera control code

- http://www.xmission.com/~nate/smooth.html
  - File, gltb.c
  - gltbMatrix and gltbMotion