Virtual Trackball: Quaternions

6 March 2007
CMPT370
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review last time

- Homogeneous coordinates
- Changing frames via multiplying by 4x4 matrix
  - Translation
  - Scaling
  - Rotation
    - Euler angles

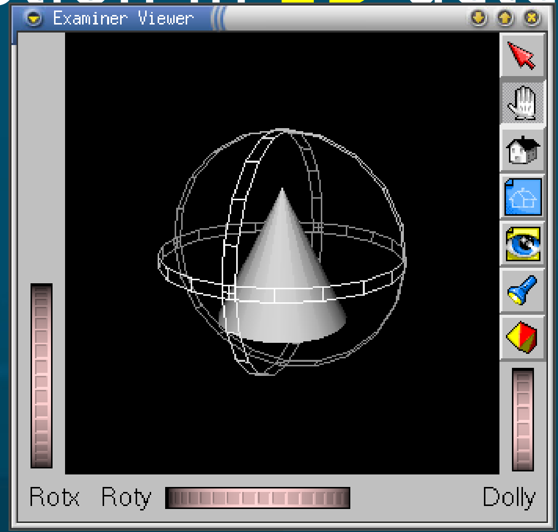
$$T = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

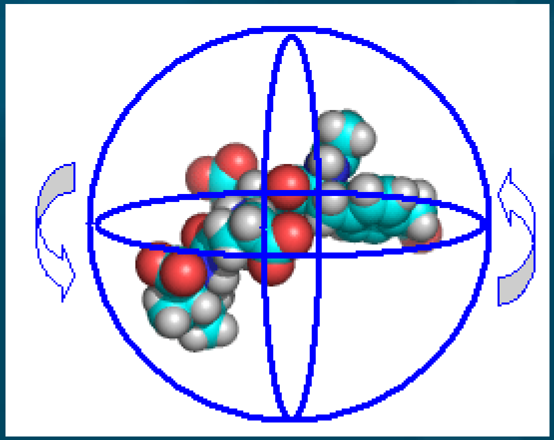
$$S = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

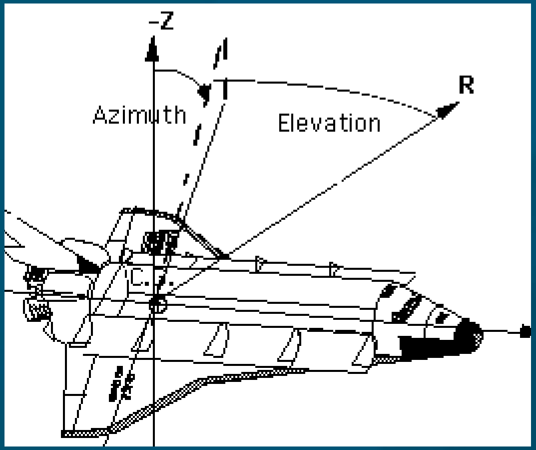
Rotations in 3D

- We've learned about rotating in 3D via three Euler angles: angles to rotate around x, y, z axes
  - Must pick an order: e.g., first x, then y, then z
  - User interface to specify three angles clunky
- A virtual trackball is like an upside-down mouse
  - Motion in 2D determines rotation of trackball

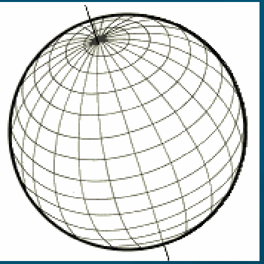


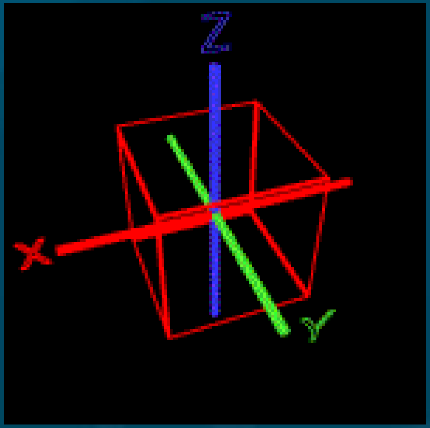


Gimbal lock



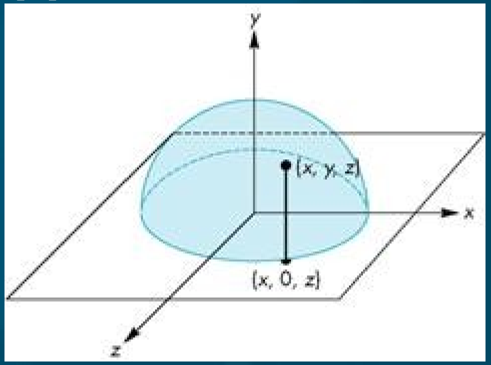
- One naïve way to do get a rotation from 2D mouse motion is:
  - Vertical motion --> elevation (latitude)
  - Horizontal --> azimuth (longitude)
- Problem: gimbal lock!
  - At the North/South poles, longitude has no meaning
  - Lose a degree of freedom
  - Apollo 11 landing on Moon nearly had an accident due to gimbal lock



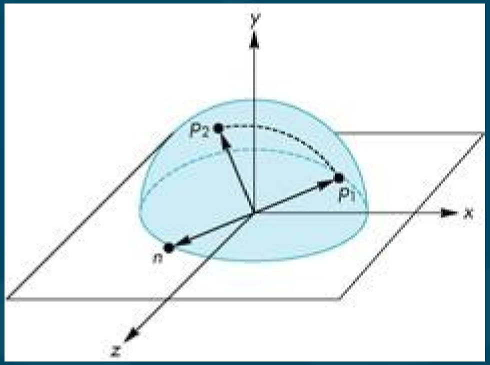


Virtual trackball

- Let the mouse position be in x–z plane
- Project up to the hemisphere of radius r:
  - $y = \mathrm{sqrt}(r^2 - x^2 - z^2)$



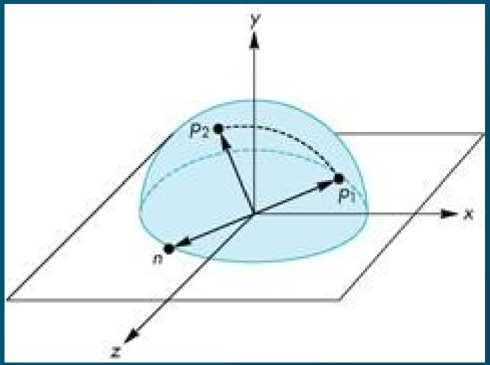
- Mouse motion corresponds to moving from p_1 to p_2 on the hemisphere
- Draw a "great circle" from p_1 to p_2
  - This determines the rotation



- Update in the event handler: every mouse motion yields a small rotation

Axis and angle of rotation

- The axis of rotation is found by the cross-product of p_1 and p_2
- The angle between p_1 and p_2 is found by: $|\sin\theta| = |n| / (|p_1| * |p_2|)$
  - If the mouse is moved slowly enough and we sample frequently, $\sin\theta \approx \theta$



- A quaternion is a compact way of representing the axis/angle of rotation

Quaternions

- Extension of complex numbers from 2D to 4D
    - (an example of a Clifford Algebra)
  - One real, three imaginary components i, j, k:
    - $\mathbf{b} = q_0 + q_1 i + q_2 j + q_3 k = (q_0, \mathbf{q})$
    - Where $\mathbf{q} = q_1 i + q_2 j + q_3 k$ is the pure quaternion part
- Properties:
    - $\mathbf{a} + \mathbf{b} = (p_0 + q_0, \mathbf{p} + \mathbf{q})$
    - $i^2 = j^2 = k^2 = -1$
    - $ij = k$, $ji = -k$, $jk = i$, $kj = -i$, $ki = j$, $ik = -j$
    - $|\mathbf{b}|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2$ (magnitude)

Multiplying quaternions

- **a** x **b** = $(p_0 q_0 - \mathbf{p} * \mathbf{q}),\ q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{p} \times \mathbf{q})$
  - **p*q**: dot-product (treat **p**, **q** as 3D vectors)
  - **p**x**q**: cross-product
- Order matters: multiplication is not commutative!
  - **a** x **b** ≠ **b** x **a**
- We'll use multiplication to compose multiple rotations

Properties of quaternions

- Conjugate: $\mathbf{b}(\text{conj}) = (q_0, -\mathbf{q})$
- Negative: $-\mathbf{b} = (-q_0, -\mathbf{q})$
- Multiplicative inverse: $\mathbf{b}^{-1} = \mathbf{b}(\text{conj}) / |\mathbf{b}|^2$
- Unit quaternions: $|\mathbf{b}|=1$, so $\mathbf{b}^{-1} = \mathbf{b}(\text{conj})$
  - We'll represent rotations with unit quaternions

Rotations with quaternions

- From the vector-angle form:
  - Rotate about the unit vector **u** by angle θ:
  - $\mathbf{b} = (\cos(\theta/2), \mathbf{u} \sin(\theta/2))$
- A point **p** in 3D space is represented by the quaternion $\mathbf{P} = (0, \mathbf{p})$
- The rotated point **p'** is represented by the quaternion
  - $\mathbf{P'} = \mathbf{b} * \mathbf{P} * \mathbf{b}^{-1}$

Converting to 4x4 matrix

- Rotate P by **b**: **$P' = b * P * b^{-1}$**
- Left-multiplication of a point $P = (x_p, y_p, z_p)$ by a rotation quaternion $q = (x, y, z, w)$:
  - $q * P$:

$$\begin{pmatrix} w_q & -z_q & y_q & x_q \\ z_q & w_q & -x_q & y_q \\ -y_q & x_q & w_q & z_q \\ -x_q & -y_q & -z_q & w_q \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \\ 0 \end{pmatrix}$$

- Right-multiplication by q^{-1}:
  - $P * q^{-1}$:

$$\begin{pmatrix} w_q & -z_q & y_q & -x_q \\ z_q & w_q & -x_q & -y_q \\ -y_q & x_q & w_q & -z_q \\ x_q & y_q & z_q & w_q \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \\ 0 \end{pmatrix}$$

Putting it all together

- Rotating a point P by a quaternion q = (x, y, z, w) is equivalent to multiplying by a 4x4 matrix:

$$\begin{pmatrix} w^2+x^2-y^2-z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & w^2-x^2+y^2-z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & w^2-x^2-y^2+z^2 & 0 \\ 0 & 0 & 0 & w^2+x^2+y^2+z^2 \end{pmatrix}$$

- (revised after lecture: I had this transposed before. This is the right way round for multiplying by a column vector on the right.)

TODO

- Lab3 if you haven't finished it already
  - OpenGL 3D model viewer
- Lab4: due next week Thu 15Mar
  - Add a virtual trackball using quaternions