Virtual Trackball: Quaternions

3 March 2009
CMPT370
Dr. Sean Ho
Trinity Western University

Review last time

- Math for 3D graphics: homogeneous coordinates
  - 4x4 transform matrices
  - Translate, scale, rotate
- Viewing: (see RedBook ch3)
  - Positioning the camera: model-view matrix
  - Selecting a lens: projection matrix
  - Clipping: setting the view volume

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

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

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

OpenGL vertex arrays

- Stores a vertex list in the graphics hardware
  - Six types of arrays: vertices, colours, colour indices, normals, texture coords, edge flags
- Our vertex list in C:
  - GLfloat verts[][3] = {{0.0, 0.0, 0.0}, {0.1, 0.0, 0.0}, ...}
- Load into hardware:
  - glEnableClientState(GL_VERTEX_ARRAY);
  - glVertexPointer(3, GL_FLOAT, 0, verts);
    - **3: 3D vertices**
    - **GL_FLOAT: array is of GLfloat-s**
    - **0: contiguous data**
    - **verts: pointer to data**

Using OpenGL vertex arrays

- Use glDrawElements instead of glVertex
- Polygon list references indices in the stored vertex array
  - GLubyte cubeIndices[24] = {0,3,2,1, 2,3,7,6, 0,4,7,3, 1,2,6,5, 4,5,6,7, 0,1,5,4};
  - Each group of four indices is one quad
- Draw a whole object in one function call:
  - glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);

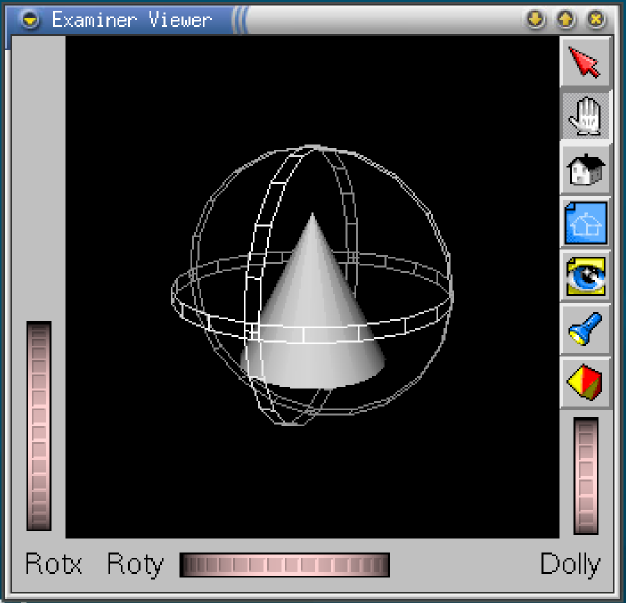
OpenGL display lists

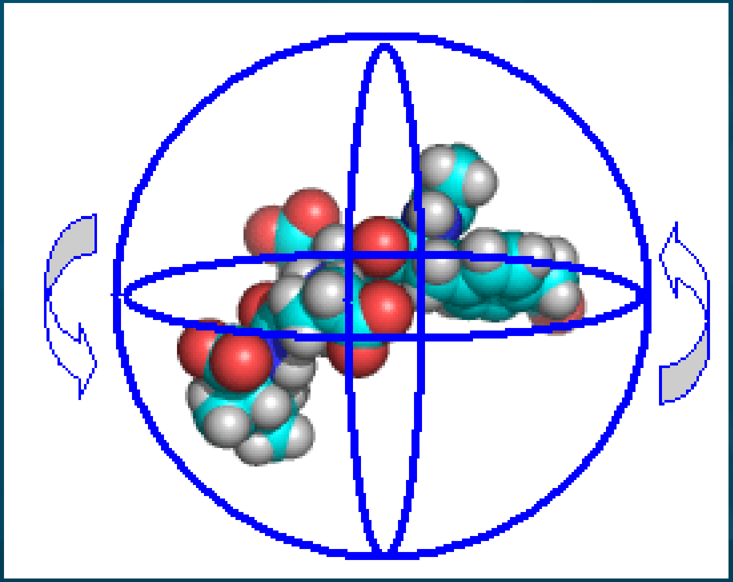
- Take a group of OpenGL commands (e.g., defining an object) and store in hardware
- Can change OpenGL state, camera view, etc. without redefining this stored object
- Creating a display list:
  - GLuint cubeDL = glGenLists(1);
  - glNewList(cubeDL, GL_COMPILE);
    - glBegin(...); ....; glEnd();
  - glEndList();
- Using a stored display list:
  - glCallList(cubeDL);

See RedBook ch7

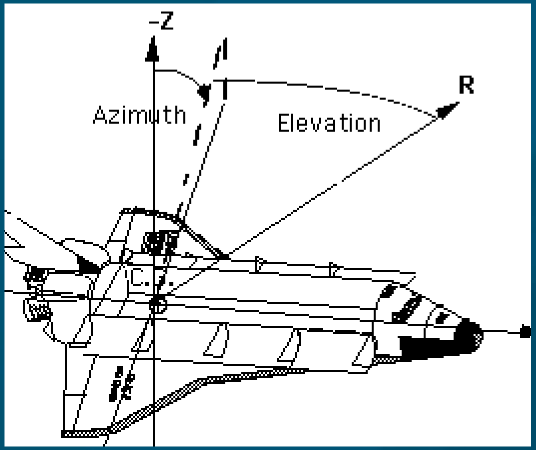
Rotations in 3D

- Euler angles: angles about x, y, z axes
  - Needs an order: e.g., first x, then y, then z
  - User interface to specify three angles clunky
- Virtual trackball: like an upside-down mouse
  - Motion in 2D determines rotation of trackball

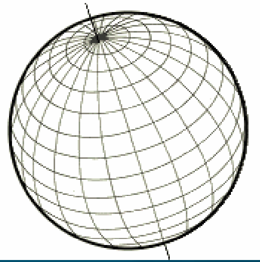


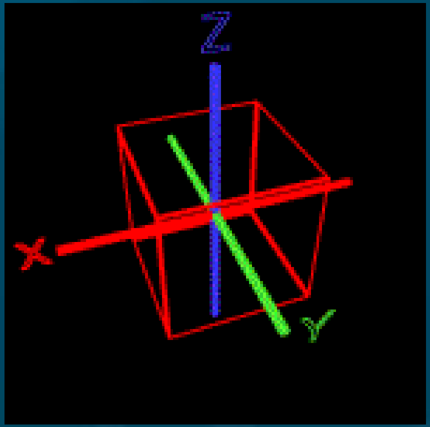


Gimbal lock



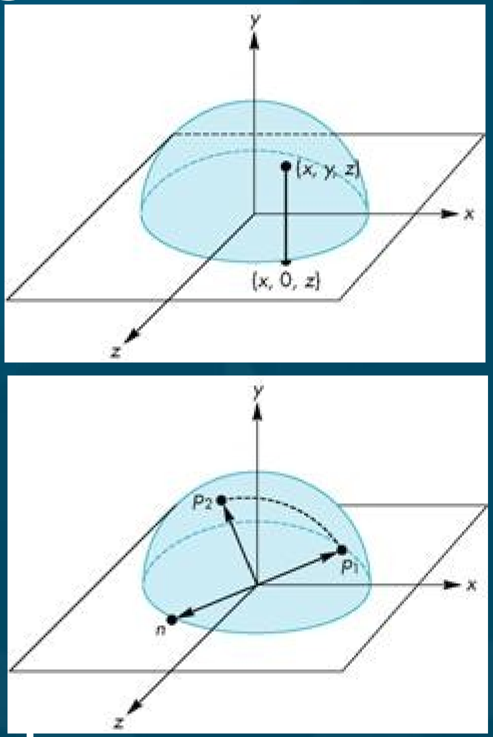
- One naïve way to do get a rotation from 2D mouse motion is:
  - Vertical motion --> elevation (latitude)
  - Horizontal --> azimuth (longitude)
- Problem: gimbal lock!
  - At the North/South poles, longitude has no meaning
  - Lose a degree of freedom
  - Apollo 11 landing on Moon nearly had an accident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



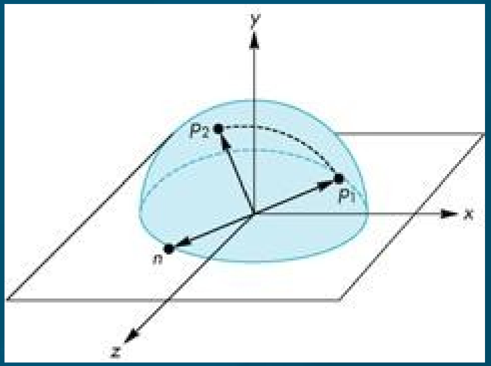


Virtual trackball

- Let the mouse position be in x-z plane
- Project up to the hemisphere of radius r:
  - $y = \text{sqrt}(r^2 - x^2 - z^2)$
- 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-move 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, $\sin\theta \approx \theta$
- glRotatef(θ, n_1, n_2, n_3): angle+axis
- Problem: how to compose two rotations? Convert axis+angle to a quaternion.

Quaternions

- Extension of complex numbers from 2D to 4D
  - (an example of a Clifford Algebra)
  - One real, three imaginary components i, j, k:
  - $\mathbf{b} = q_0 + q_1 i + q_2 j + q_3 k = (q_0, \mathbf{q})$
  - $\mathbf{q} = q_1 i + q_2 j + q_3 k$ 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)
  - **pxq**: cross-product (yields a quaternion)
- Order matters: multiplication is not commutative!
  - **a** x **b** ≠ **b** x **a**
- We'll represent composing multiple rotations by multiplication of the corresponding quaternions

Properties of quaternions

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

Rotations with quaternions

- From the axis-angle form:
  - Rotate about the unit vector **u** by angle θ:
  - $\mathbf{q} = (\cos(\theta/2), \mathbf{u} \sin(\theta/2))$
- 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 **p'**:
  - $\mathbf{p'} = \mathbf{q} * \mathbf{p} * \mathbf{q}^{-1}$
- These are quaternion multiplications; convert to matrix notation:

Converting to 4x4 matrix

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

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

- Followed by 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

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

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