Interpolating Unit Quaternions

To interpolate two orientations, we could use linear interpolation on all the entries of the quaternion (i.e. treat the quaternion as a four-dimensional vector). Unfortunately, intermediate quaternions would not have unit norm, and the angle velocity would not be constant even if we renormalized.

The solution: spherical linear interpolation, or the slerp. Consider the quaternions vectors, and find the angle between them:

\[ \omega = \cos^{-1}(q_1 \cdot q_2). \]

Given a parameter \( u \in [0, 1] \), the slerp interpolated value is defined as

\[ q(u) = q_1 \frac{\sin((1 - u)\omega)}{\sin(\omega)} + q_2 \frac{\sin(u\omega)}{\sin(\omega)}. \]

The slerp will have numerical difficulties when \( \omega \approx 0 \). In such a case, it is wise to replace the slerp with a lerp. There will also be a problem with \( \omega \approx n\pi/2 \); this should probably be flagged with an error and/or a request for more keyframes.
There are two possible great circles joining any two orientations. Normally, we want to choose the shortest one. This can be accomplished by testing the condition \((q_1 - q_2) \cdot (q_1 - q_2) > (q_1 + q_2) \cdot (q_1 + q_2)\). If true, \(q_2\) should be replaced with \(-q_2\). Recall that two quaternions that differ only in sign represent the same orientation.
3D Rotation User Interface

Goal: Want to specify angle-axis rotation “directly”.

Problem: May only have mouse, which only has two degrees of freedom.

Solutions: Virtual Sphere, Arcball.
The Virtual Sphere

1. Define portion of screen to be projection of virtual sphere.
2. Get two sequential samples of mouse position, $S$ and $T$.
3. Map 2D point $S$ to 3D unit vector $\vec{p}$ on sphere.
4. Map 2D vector $\overrightarrow{ST}$ to 3D tangential velocity $\vec{a}$. 
5. Normalize $\vec{d}$.
6. Axis: $\vec{a} = \vec{p} \times \vec{d}$.
7. Angle: $\theta = \alpha |\overrightarrow{ST}|$.
   (Choose $\alpha$ so a $180^\circ$ rotation can be obtained.)
8. Save $T$ to use as $S$ for next time.
The Arcball

1. Choose region of screen as projection of sphere: 2D point $O$ center, radius $\rho$.
2. Get initial 2D point $S$ on button-down.
3. Compute $\vec{s} = \overrightarrow{ST}$, $\vec{s} = (s_x, s_y)$.
4. Compute 2D radius: $r^2 = s_x^2 + s_y^2$. 
5. Map 2D vector $\vec{s}$ to 3D unit vector $\vec{p}$:
   If $r^2 > \rho^2$, map to silhouette of unit sphere:
   \[
   \begin{align*}
   p_x & \leftarrow s_x / r \\
   p_y & \leftarrow s_y / r \\
   p_z & \leftarrow 0.
   \end{align*}
   \]
   Else,
   \[
   \begin{align*}
   p_x & \leftarrow s_x / \rho \\
   p_y & \leftarrow s_y / \rho \\
   p_z & \leftarrow \sqrt{1 - p_x^2 - p_y^2}.
   \end{align*}
   \]

6. For each new mouse position $T$, map to unit 3D vector $\vec{q}$ as above.
7. Axis: $\vec{a} = \vec{p} \times \vec{q}$.
8. Angle: $\theta = 2 \cos^{-1}(\vec{p} \cdot \vec{q})$. 
9. Notes:

- Rotation given by \( \hat{\omega} \) is equal to the angle of the great arc between \( \hat{p} \) and \( \hat{q} \).
- Doubling the angle matches orientation's mathematical structure better.
- Points on opposite sides of the sphere silhouette allow a rotation by \( 360^\circ \).

(by ambiguity of \( \cos^{-1}(0) \) is finished...is always identity.)
Transforming Normals

- Normals are vectors perpendicular to all tangents at a point:

\[ \mathbf{N} \cdot \mathbf{T} \equiv \mathbf{n}^T \mathbf{t} = 0. \]

- Note that the natural representation of \( \mathbf{N} \) is as a row vector.
- Suppose we have a transformation \( M \), a point \( P \equiv \mathbf{p} \), and a tangent \( \mathbf{T} \equiv \mathbf{t} \) at \( P \).
- Let \( M_{\ell} \) be the “linear part” of \( M \), i.e. the upper \( 3 \times 3 \) submatrix.

\[
\begin{align*}
\mathbf{p}' &= M \mathbf{p} \\
\mathbf{t}' &= M \mathbf{t} \\
&= M_{\ell} \mathbf{t}.
\end{align*}
\]

\[
\begin{align*}
\mathbf{n}^T \mathbf{t} &= \mathbf{n}^T M_{\ell}^{-1} M_{\ell} \mathbf{t} \\
&= (M_{\ell}^{-1} \mathbf{n})^T (M_{\ell} \mathbf{t}) \\
&= (\mathbf{n}')^T \mathbf{t}'.
\end{align*}
\]
\[ \equiv \vec{N}' \cdot \vec{T}'. \]

- Transformation normals by inverse transpose of linear part of transformation: 
  \[ \mathbf{n}' = M^{-1T}_\ell \mathbf{n}. \]
- If \( M_T \) is orthogonal (usual case for rigid body transform), 
  \[ M^{-1T}_T = M_T. \]

Only worry if you have a shear transformation.