

Quaternions and Interpolation

Recall from Section 6.3 that unit quaternions corresponds to an axis $\vec{u} \in \mathbb{R}^3$ ($|\vec{u}| = 1$) and an angle $\theta \in [0; 2\pi]$ (Prop. 6.3), and that quaternions can be seen as representing rotations in \mathbb{R}^3 (with angle θ around axis \vec{u}) (Prop. 6.2 (and 6.4 (b))) in a way such that quaternion product (multiplication) corresponds to compositions of rotations in \mathbb{R}^3 (Prop. 6.4 (a)).

This can be used to interpolate nicely between two rotated positions (of a model/object in scene) :

Let R_1 be rotation giving first
 (from object's basic position)
 rotated position (a.k.a. orientation)

Let R_2 be rotation (from object's basic position/orientation) giving second orientation.

If we store rotations (eg. R_1 and R_2) as _{unit} quaternions [or as axis/angle, from which we can create the corresponding quaternions via Prop. 6.3] we can

- i) From the two quaternions q_1 and q_2 corresponding to R_1 and R_2 construct

$$q = q_2 \cdot q_1^{-1}$$

(see exercise 6.14 on p. 256 for how to find q_1^{-1})

$$\begin{aligned} \text{From } q \cdot q_1 &= (q_2 \cdot q_1^{-1}) \cdot q_1 \\ &= q_2 \cdot (q_1^{-1} \cdot q_1) = q_2 \cdot I \\ &= q_2 \end{aligned}$$

we see (from Prop. 6.4 (a)) that q represents the rotation taking the object from orientation R_1 to orientation R_2 .

ii) We can easily find the axis \vec{u} and angle θ for q (using Prop. 6.3).

iii) Given those, we can define nice intermediate rotations taking the object from R_1 to R_2 in small steps [for intermediate frames to render]:

Keep axis \vec{u} .
Use angle $\theta_t = t \cdot \theta$ for $t \in [0; 1]$.

The power of quaternions here lies in the ease with which the rotation represented by q can be found.

Of course, any rotation must in the GPU be represented by a matrix.

This is done^{e.g.} by the axis/angle \rightarrow rot. matrix conversion (S. 45) on p. 223

[Rodrigues Rotation Formula in matrix form]

or (if starting with a unit quaternion representation) by the quaternion \rightarrow rot. matrix version of it (p. 264).

[In OpenGL, the glRotatef - command directly takes an axis and angle, of course.]

Note that rotating θ degrees around \vec{u} is the same as rotating $360^\circ - \theta$ around $-\vec{u}$.

This is in unit quaternions reflected in q and $-q$ representing same rotation (but with angles θ and $360^\circ - \theta$).

When interpolating, one normally wants the smaller of θ and $360^\circ - \theta$. How this is determined is described at top of page 267.