A FLASH BASED DEMONSTRATION OF QUATERNIONS AND SPHERICAL LINEAR INTERPOLATION

Paul R. Bouthellier
Department of Mathematics and Computer Science
University of Pittsburgh-Titusville
Titusville, PA 16354
pbouthe@pitt.edu

Introduction:

Computer graphics can be used to illustrate the mathematics behind concepts such as: Rotations about the x, y, and z-axes, rotations about an arbitrary axis, movement along parametric curves in R^2 and R^3, projections of 3D objects onto 2D surfaces, linear transformations, back-face culling, creating cameras, object, world, camera, clipping, and screen space, combining maps, collision detection: when lines, cubes, spheres, and triangles intersect, modeling curves and surfaces, reflections, rotations, and filters, and numerical stability.

In this paper we will discuss using quaternions for rotations and the quaternion based technique SLERP-Spherical Linear Interpolation. SLERP allows for creating smooth rotations between two coordinate frames. The technique can also be extended to more than two rotation-coordinate frames pairs. SLERP is one of the main uses of quaternions in fields such as computer animation today.

The programs were written in the ActionScript language (version 3) of Flash CS4.

Definition of Quaternions:

Define $q=q_0+iq_1+jq_2+kq_3=q_{0+}\mathbf{q}$ where

$$\begin{gathered} i^2=j^2=k^2=ijk=-1 \\ ij=k=-ji \\ jk=i=-kj \\ ki=j=-ik \end{gathered} \tag{1}$$

Quaternion multiplication:

where

$$q=q_0+iq_1+jq_2+kq_3=q_{0+}\mathbf{q} \tag{2}$$

and

$$p=p_0+ip_1+jp_2+kp_3=p_{0+}\mathbf{p} \tag{3}$$

their product is given by

$$pq=p_0q_0-\mathbf{q}\bullet\mathbf{p}+p_0\mathbf{q}+q_0\mathbf{p}+\mathbf{p}\times\mathbf{q} \tag{4}$$

Given pure quaternions (quaternions which have a zero real part):

$$q=iq_1+jq_2+kq_3=\mathbf{q} \tag{5}$$

$$p=ip_1+jp_2+kp_3=\mathbf{p} \tag{6}$$

$$pq=-\mathbf{p}\bullet\mathbf{q}+\mathbf{p}\times\mathbf{q} \tag{7}$$

Rotations:

Given a unit quaternion

$$\cos(\frac{\theta}{2})+u\sin(\frac{\theta}{2}) \tag{8}$$

where u in a unit vector in R^3,

defining

$$q^{-1}=q_0-iq_1-jq_2-kq_3=q_0-\mathbf{q} \tag{9}$$

and where $v\in R^3$ is represented as a pure quaternion

then [1], [2]

$$\mathbf{q}v\mathbf{q}^{-1} \tag{10}$$

is a right-handed rotation of v by an angle of θ about the vector **q.**

The right-handed coordinate system we shall is the one defined by Flash and is illustrated below:

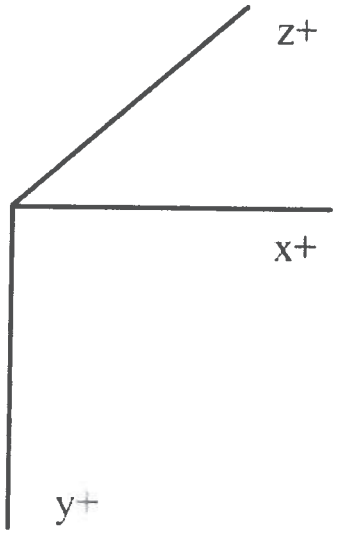

Figure 1: Flash Coordinate System

Quaternion Math

Given quaternions p, q, and r:

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{11}$$

$$\|pq\| = \|p\| \cdot \|q\| \tag{12}$$

$$(pq)r = p(qr) \tag{13}$$

$qv^{-1}q=(q_0+\mathbf{q})(0+\mathbf{v})(q_0+\mathbf{q})=$ (14)

$$0 + (q \cdot v)q + q_0^2 q + q_0(q \times v) - q_0(v \times q) \tag{15}$$

$$= (2q_0^2 - 1)v + 2q \cdot v + 2q_0^2(q \times v) \tag{16}$$

The matrix equivalent is given by [1]:

Where s=sin(θ), c=cos(θ) and $q = < q_1, q_2, q_3 >$

$$R_q^\theta = \begin{pmatrix} c+(1-c)q_1^2 & (1-c)q_1q_2 - sq_3 & (1-c)q_1q_3 - sq_2 \\ (1-c)q_1q_2 + sq_3 & c+(1+c)q_2^2 & (1-c)q_2q_3 - sq_1 \\ (1-c)q_1q_3 - sq_2 & (1-c)q_2q_3 + sq_1 & c+(1+c)q_3^2 \end{pmatrix} \quad (17)$$

To perform a sequence of rotations:

$$r_2(r_1vr^{-1})r_2^{-1} = (r_2r_1)v(r_1^{-1}r_2^{-1}) = (r_2r_1)v(r_2r_1)^{-1} \quad (18)$$

A Simple Cube

A first test is to create a simple object and use quaternions to rotate it about its x, y, and z-axes. A cube is the computer graphics equivalent of "Hello World." Our cube consists of:

- 8 points
- 6 faces and 4 connectors (for the wireframe version)
- quaternions for rotation about the x, y, and z-axes

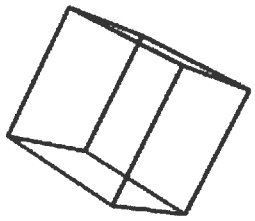

Figure 2: Basic Cube

Such a cube can also be created using 6 grouped squares in Flash CS4 or by a cube done in a package which can import such 3d objects into Flash.

Spherical Linear Interpolation

Our next goal is to:

- Rotate the z-axis of the cube 90° clockwise in world space
- Rotate the cube clockwise (several times) about its z-axis
- Move the cube through space via parametric equations

To accomplish these goals we use spherical linear interpolation:

Spherical linear interpolation interpolates between two quaternions q_0 and q_1 by treating them as two points on a unit 4-sphere. Where t goes between 0 and 1 [1]:

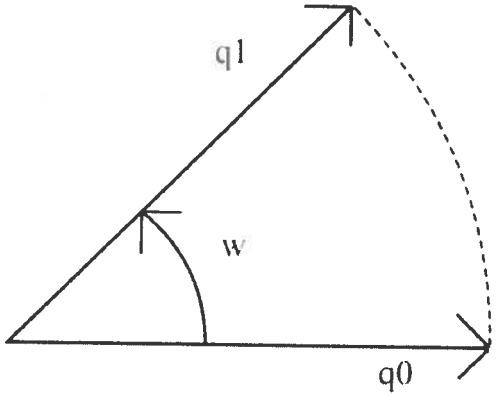

Figure 3: Spherical Linear Interpolation

$$slerp(q_0, q_1, t) = \frac{\sin((1-t)w)}{\sin(w)} q_0 + \frac{\sin(tw)}{\sin(w)} q_1 \tag{19}$$

- Choose the signs of q_0 and q_1 such that $q_0 \cdot q_1 > 0$ to get the shortest rotational arc from q_0 and q_1 [1].
- If $\sin(w) \approx 0$ we could have numerical problems-can use simple linear interpolation [1].

Conclusions:

It is also possible to create a rotating camera by the addition of two more quaternions: one for movement of the camera in world space and the second for the rotation of the camera about its own axes. To create the necessary equations it is often useful to first derive the equations in matrix form, and then translate the results to the necessary translations and quaternions.

References:

[1] *3D Math Primer for Graphics and Game Development*, Wordworth Publishing, Inc., 2002, by Dunn and Parberry.

[2] *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton University Press, 2002, by Kuipers.