VII.6

QUATERNIONS AND 4 X 4 MATRICES

Ken Shoemake
Otter Enterprises
Palo Alto, California

Quaternions are steadily replacing Euler angles as the internal representation of orientations, presumably because of such advantages as are detailed in Shoemake (1985, 1989). What is not so obvious from that paper, however, is that they mesh remarkably well with 4×4 homogeneous matrices.

Matrix multiplication can be used quite nicely for quaternion multiplication, since quaternions are, in fact, four-component homogeneous coordinates for orientations, and since they multiply linearly. Consider a quaternion q as a 4-vector, written (x_g, y_q, z_g, w_q), or as just (x, y, z, w) when context makes it clear. The quaternion product $p \blacklozenge q$ *is* a linear function of either p or q, so two different matrix forms are possible. (Transpose these when using row vectors.) They are:

$$p \blacklozenge q = \mathbf{L}(p)q = \begin{bmatrix} w_p & -z_p & y_p & x_p \\ z_p & w_p & -x_p & y_p \\ -y_p & x_p & w_p & z_p \\ -x_p & -y_p & -z_p & w_p \end{bmatrix} \begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix}$$

and

$$p \blacklozenge q = \mathbf{R}(q)p = \begin{bmatrix} w_q & z_q & -y_q & x_q \\ -z_q & w_q & x_q & y_q \\ y_q & -x_q & w_q & z_q \\ -x_q & -y_q & -z_q & w_q \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix}$$

Using these **L** and **R** matrices, we can readily convert a quaternion to a homogeneous rotation matrix. Recall that a quaternion q rotates a vector v using the product $q \blacklozenge v \blacklozenge q^{-1}$, where $q^{-1} = q^*/\mathrm{N}(q)$. In the common case of a unit quaternion, $q^{-1} = q^*$. This permits the rotation matrix to be computed from the components of q—since q^* *is* merely $(-x, -y, -z, w)$—as

$$\mathbf{Rot}(q) = \mathbf{L}(q)\mathbf{R}(q^*),$$

so that a hardware matrix multiplier can do all the conversion work. Isn't that nice?

More specifically, suppose you are using hardware matrix multiplication to compose a series of matrices that will be applied to row vectors in right-handed coordinates, as in $v\mathbf{SNQTP}$, where $\mathbf{Q}$ is to be derived from a quaternion, $q = (x, y, z, w)$. Then instead of $\mathbf{Q}$, compose with $\mathbf{L}(q)$ and $\mathbf{R}(q^*)$, so that the sequence is $v\mathbf{SNRLTP}$. For row vectors, we want the transpose form of **L** and **R**, so we have

$$\mathbf{Q}_{\mathrm{row}} = \mathbf{R}_{\mathrm{row}}(q^*)\mathbf{L}_{\mathrm{row}}(q) = \begin{bmatrix} w & z & -y & -x \\ -z & w & x & -y \\ y & -x & w & -z \\ x & y & z & w \end{bmatrix}$$

$$\times \begin{bmatrix} w & z & -y & x \\ -z & w & x & y \\ y & -x & w & z \\ -x & -y & -z & w \end{bmatrix}.$$

Because the desired result is a *homogeneous* rotation matrix, an overall scale factor can be ignored; thus, q^* can be used instead of q^{-1} even if $\mathrm{N}(q) = q \blacklozenge q^* = x^2 + y^2 + z^2 + w^2$ is not equal to one. Be aware, however, that some systems do not implement matrix manipulations carefully, and will misbehave if the bottom right entry of the matrix is not 1. Even when normalization is desired, it is not necessary to compute a square root; only addition, subtraction, multiplication, and division are used.

Notice that only the last row and column of the two matrices differ, and then only by transposition or sign change. (This may seem obvious, but one programmer's "obvious" is another's "obscure", and perhaps pointing it out will save someone time.) Although these two matrices may look peculiar to the average computer graphicist, multiplying them confirms that the result is a matrix with the expected zeros in the last row and column, with 1 in the corner for a unit quaternion.

$$\mathbf{Q}_{\mathrm{row}} = \begin{bmatrix} w^2+x^2-y^2-z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & w^2-x^2+y^2-z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & w^2-x^2-y^2+z^2 & 0 \\ 0 & 0 & 0 & w^2+x^2+y^2+z^2 \end{bmatrix}$$

Fans of *4D* should note that any rotation of a 4-vector v can be written as $p \blacklozenge q \blacklozenge v \blacklozenge q^{-1} p$, which is translated easily into matrices using this same approach. The q quaternion controls rotation in the planes excluding w—namely x–y, x–z, and y–z—while the p quaternion controls rotation in the planes including w—namely w–x, w–y, and w–z.

Converting a homogeneous matrix back to a quaternion also is relatively easy, as the $\mathbf{Q}_{\mathrm{row}}$ matrix has a great deal of structure that can be exploited. To preserve numerical precision, one must adapt to the specific matrix given, but the structure also makes that elegant. Observe that the difference, $\mathbf{Q}_{\mathrm{row}}$ minus its transpose, has a simple form:

$$\mathrm{Q}_{\mathrm{row}} - \mathrm{Q}_{\mathrm{row}}^{\mathrm{T}} = \begin{bmatrix} 0 & -4wz & 4wy & 0 \\ 4wz & 0 & -4wx & 0 \\ -4wy & 4wx & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Clearly, it is easy to find x, y, and z when w is known, so long as w is not zero—or, for better precision, so long as w is not nearly zero. On the other hand, the sum Of $\mathbf{Q}_{\mathrm{row}}$ plus its transpose also is simple, if we ignore the diagonal:

$$\mathbf{Q}_{\mathrm{row}} + \mathbf{Q}_{\mathrm{row}}^{\mathrm{T}} - \mathrm{diagonal} = \begin{bmatrix} 0 & 4xy & 4xz & 0 \\ 4xy & 0 & 4yz & 0 \\ 4xz & 4yz & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

So, knowing any one of x, y, or z also makes it easy to find the others, and to find w (using the difference matrix). In particular, if (i, j, k) is a cyclic permutation of (0, 1, 2), then

$$w = \frac{\mathbf{Q}[k,j] - \mathbf{Q}[j,k]}{q[i]}, \quad q[j] = \frac{\mathbf{Q}[i,j] + \mathbf{Q}[j,i]}{q[i]},$$

$$q[k] = \frac{\mathbf{Q}[i,\ k] + \mathbf{Q}[k,\ i]}{q[i]}.$$

Now observe that the trace of the homogeneous matrix (the sum of the diagonal elements) always will be $4w^2$. Denoting the diagonal elements by X, Y, Z, and W, one finds all possibilities:

$$4x^2 = X - Y - Z + W,$$

$$4y^2 = -X + Y - Z + W$$

$$4z^2 = -X - Y + Z + W$$

$$4w^2 = X + Y + Z + W$$

Except for w, the computation is $q[i] = \mathrm{Sqrt}(\mathbf{Q}[i,\ i] - \mathbf{Q}[j,\ j] - \mathbf{Q}[k,\ k] + \mathbf{Q}[3,\ 3])/2$. Since a divide almost certainly is cheaper than a square root, it is better to compute just one of these, preferably the largest, since that will give the best precision for the divides; but look at the diagonal elements again, and let $S = w^2 - x^2 - y^2 - z^2$. Then $X = S + 2x^2$, $Y = S + 2y^2$, $Z = S + 2z^2$, and $T = X + Y + Z = S + 2w^2$. Clearly, which of X, Y, Z, or T is largest indicates which of x, y, z, or w is largest. (T is the trace of the upper-left 3×3 corner of the homogeneous matrix.)

See also (*498*) Using Quaternions for Coding 3D Transformations, Patrick-Gilles Maillot