Rotations for N-Dimensional Graphics

Andrew J. Hanson
Computer Science Department
Indiana University
Bloomington, IN 47405
hanson@cs.indiana.edu

◊ Introduction ◊

In a previous article in Graphics Gems IV, "Geometry for N-Dimensional Graphics" (Hanson 1994), we described a fundamental family of techniques for dealing with the geometry of N-dimensional models in the context of graphics applications. Here, we build on that framework to look in more detail at the treatment of rotations in N-dimensional Eucidean space. In particular, we give a previously overlooked but very natural N-dimensional extension of the 3D rolling ball technique described in an earlier Gem (Hanson 1992), along with the corresponding analog of the Virtual Sphere method (Chen et al. 1988). Next, we discuss a practical method for specifying and understanding the parameters of N-dimensional rotations in terms of nested hyperspheres. Finally, we give the 4D extension of 3D quaternion orientation splines along with some additional details of the 4D Arcball method (Shoemake 1994), and a discussion of the problems involved in extending these treatments to N-dimensional rotations with $N > 4$.

For additional details and insights concerning N-dimensional geometry, we refer the reader to classic sources such as (Sommerville 1958,Coxeter 1991,Hocking and Young 1961,Efimov and Rozendorn 1975).

◊ The Rolling Ball in N Dimensions ◊

Basic Intuition from the 2D Rolling Ball. The basic intuitive property of a rolling ball (or *tangent space*) rotation algorithm in any dimension is that it takes a unit vector $\hat{\mathbf{v}}_0 = (0, 0, \ldots, 0, 1)$ pointing purely in the N-th direction and tilts it infinitesimally into the remaining $(N-1)$ dimensions so that it takes the form

$$\hat{\mathbf{v}} = (\hat{\mathbf{n}} \sin\theta, \cos\theta)$$

where $\hat{\mathbf{n}} = (n_1, n_2, \ldots, n_{N-1}, 0)$ is an arbitrary $(N-1)$-dimensional vector of unit length.

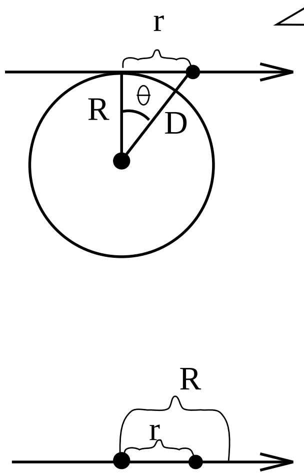


Figure 1. 2D rolling ball.

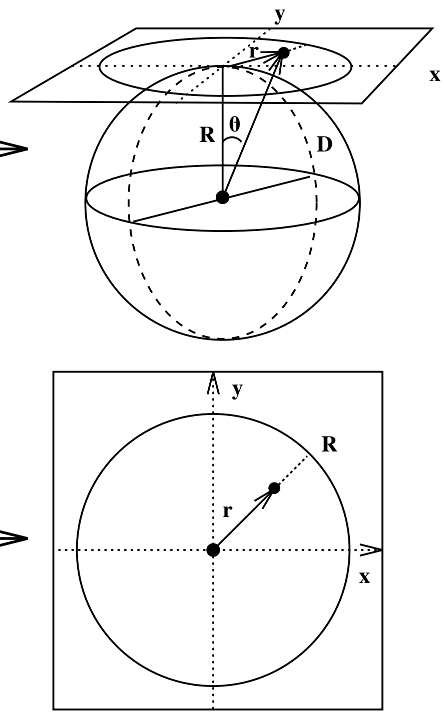


Figure 2. 3D rolling ball.

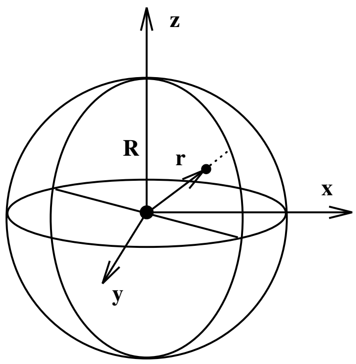


Figure 3. 4D rolling ball.

> *Qualitatively speaking, the rolling ball* pulls *vectors with an unseen N-th component in the direction $\hat{\mathbf{n}}$ of the controller motion, bringing the other components gradually into view.*

2D. The 2D case, illustrated in Figure 1, is realized simply by applying the rotation matrix

$$M_2 = R_0 = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} . \tag{1}$$

This transforms the vector $(0, 1)$ with *only* an N-th component into one with a small first component "pulled out."

Typically, we choose $c = \cos\theta = R/D$ and $s = \sin\theta = r/D$, where R is the radius of a rotating disk containing the object to be rotated, r is the distance moved by the controller (here, a 1D mouse), and $D^2 = R^2 + r^2$, as illustrated in Figure 1.

Implementation Note: For interactive systems, this choice has the particular advantage that, however rapidly the user moves the controller, $-1 < (r/D) < +1$, so

$|\theta| < \pi/2$. Depending upon the desired interface behavior, an alternative choice would be to take $\theta = r/R$ (in radians). This may be somewhat slower to calculate, and can cause large discontinuities in orientation for large controller motion.

3D. The 3D rolling ball can be derived by replacing Eq. (1) by the analogous 3×3 matrix for (x, z) rotations and then enclosing this in a conjugate pair of rotations that move an arbitrary 2D mouse displacement $\vec{\mathbf{r}} = (x, y, 0) = (rn_x, rn_y, 0)$ into the (x, z)-plane. With the 2D mouse displacement $\vec{\mathbf{r}}$ depicted in Figure 2 and

$$R_{xy} = \begin{bmatrix} n_x & -n_y & 0 \\ n_y & n_x & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; R_0 = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix},$$

then we find an alternative derivation of (Hanson 1992):

$$\begin{aligned} M_3 &= R_{xy} R_0 (R_{xy})^{-1} && (2) \\ &= \begin{bmatrix} c + (n_y)^2(1-c) & -n_x n_y(1-c) & +n_x s \\ -n_x n_y(1-c) & c + (n_x)^2(1-c) & +n_y s \\ -n_x s & -n_y s & c \end{bmatrix} && (3) \\ &= \begin{bmatrix} 1 - (n_x)^2(1-c) & -n_x n_y(1-c) & +n_x s \\ -n_x n_y(1-c) & 1 - (n_y)^2(1-c) & +n_y s \\ -n_x s & -n_y s & c \end{bmatrix}, && (4) \end{aligned}$$

where $n_x^2 + n_y^2 = 1$.

4D. The 4D case proceeds in two stages, beginning with a 3D mouse motion $\vec{\mathbf{r}} = (x, y, z, 0) = (rn_x, rn_y, rn_z, 0)$ shown in Figure 3; we transform first to move (n_y, n_z) into a pure y-component, and then rotate the entire result so that we end up with only a pure x-component. Defining

$$R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n_y}{r_{yz}} & -\frac{n_z}{r_{yz}} & 0 \\ 0 & \frac{n_z}{r_{yz}} & \frac{n_y}{r_{yz}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_{xy} = \begin{bmatrix} n_x & -r_{yz} & 0 & 0 \\ r_{yz} & n_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_0 = \begin{bmatrix} c & 0 & 0 & s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s & 0 & 0 & c \end{bmatrix}, \qquad (5)$$

where $r_{yz}^2 = n_y^2 + n_z^2$, we find

$$\begin{aligned} M_4 &= R_{yz} R_{xy} R_0 (R_{xy})^{-1} (R_{yz})^{-1} \\ &= \begin{bmatrix} 1 - (n_x)^2(1-c) & -(1-c)n_x n_y & -(1-c)n_x n_z & sn_x \\ -(1-c)n_x n_y & 1 - (n_y)^2(1-c) & -(1-c)n_y n_z & sn_y \\ -(1-c)n_x n_z & -(1-c)n_y n_z & 1 - (n_z)^2(1-c) & sn_z \\ -sn_x & -sn_y & -sn_z & c \end{bmatrix}, \qquad (6) \end{aligned}$$

where we used $n_x^2 + n_y^2 + n_z^2 = 1$ to get the symmetric form shown.

ND. The obvious extension of this procedure to any dimension is accomplished by having the controller interface supply an $(N-1)$-dimensional vector $\vec{\mathbf{x}} = (rn_1, rn_2, \ldots, rn_{N-1})$ with $\vec{\mathbf{x}} \cdot \vec{\mathbf{x}} = r^2$ and $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$ and applying the rotation

$$\begin{aligned} M_N &= R_{N-2,N-1} R_{N-3,N-2} \cdots R_{1,2} R_0 (R_{1,2})^{-1} \cdots (R_{N-3,N-2})^{-1} (R_{N-2,N-1})^{-1} \\ &= \begin{bmatrix} 1-(n_1)^2(1-c) & -(1-c)n_2n_1 & \cdots & -(1-c)n_{N-1}n_1 & sn_1 \\ -(1-c)n_1n_2 & 1-(n_2)^2(1-c) & \cdots & -(1-c)n_{N-1}n_2 & sn_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(1-c)n_1n_{N-1} & -(1-c)n_2n_{N-1} & \cdots & 1-(n_{N-1})^2(1-c) & sn_{N-1} \\ -sn_1 & -sn_2 & \cdots & -sn_{N-1} & c \end{bmatrix} \end{aligned} \tag{7}$$

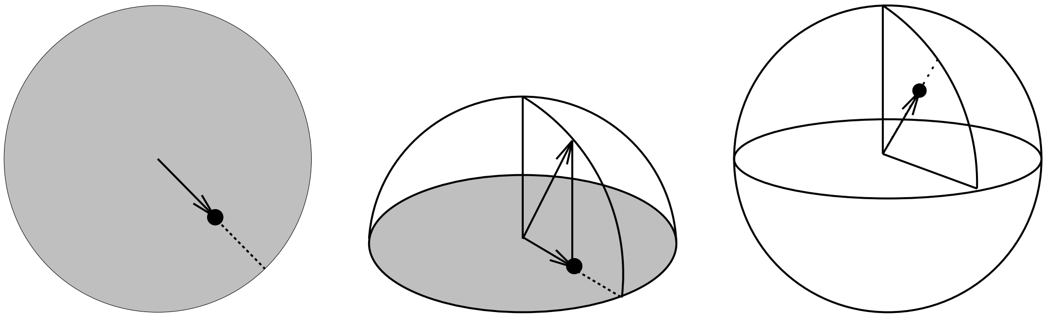
Recall that the controller input $\vec{\mathbf{x}} = r\hat{\mathbf{n}}$ that selects the direction to "pull" also specifies the parameters $c = \cos\theta = R/D$, $s = \sin\theta = r/D$, with $D^2 = R^2 + r^2$, or, alternatively, $\theta = r/R$.

Figures 1, 2, and 3 indicate schematically how the dynamics of the user interface work — moving the controller causes the portion of the figure that was initially projected to a point along the line of sight through the center of rotation to tilt into view as though following the lead of the control vector.

◇ Controlling Unit Vectors in N Dimensions ◇

In general the controller motion vector $\vec{\mathbf{x}}$ may have any magnitude, and this quantity controls a vector's rotation according to the chosen format for $\theta(r, R)$. However, there is a special class of control strategies that may be used for controlling N-dimensional unit vectors such as N-D lighting vectors: by picking an $(N-1)$-dimensional point *inside* an $(N-1)$-sphere projected from its N-dimensional embedding space to an $(N-1)$-dimensional plane, one can both *select* and *display* the unit vector.

Figure 4a shows a schematic diagram of a method for controlling the 3D lighting vector using a 2D mouse: the unit vector in 3D has only two degrees of freedom, so that picking a point within a unit circle determines the direction uniquely (up to the sign of its view-direction component). With a convention for distinguishing vectors with positive or negative view-direction components (e.g., solid or gray), we can uniquely choose and represent the 3D direction. Control of the vector is straightforward using the rolling ball: the lighting vector initially points straight out of the screen (up in the oblique view of Figure 4b), and moving the mouse in the desired direction tilts the vector to its new orientation, whose projection to the plane of Figure 4a is shown in the gray ellipse in Figure 4b. Rotating past 90 degrees moves the vector so its view-direction component is into the screen. However, control of the unit vector is even more natural if one simply tracks the position of the mouse constrained to the interior of the circle in Figure 4a. **Note**: To extend this method to the *back* hemisphere, map



(a) (b) (c)

Figure 4. Schematic diagram comparing (a) selecting 2D point in disk to specify 3D light direction, shown obliquely in (b), and (c) 3D flying mouse control of a 4D light direction by picking 3D point inside solid sphere.

the *exterior* of the circle to points on the back hemisphere out to some second radius chosen to correspond to the direction opposite to the view direction.

The analogous control system for 4D lighting, shown in Figure 4c, is based on a similar observation: since the 4D normal vector has only 3 independent degrees of freedom, choosing an *interior point* inside a solid sphere determines the vector uniquely up to the sign of its component in the unseen 4th dimension (the "4D view-direction component"). The rest of the control proceeds analogously. Since we cannot easily interpret 4D oblique views, we do not attempt to draw the 4D analog of Figure 4b.

◊ Controlling the Remaining Rotation Degrees of Freedom ◊

There are $N(N-1)/2$ parameters in a general N-dimensional orthogonal rotation matrix, one parameter for each possible pair of axes specifying a *plane of rotation* (the 3D intuition about "axes of rotation" does not extend simply to higher dimensions). The matrix M_N in Equation (7) has only $(N-1)$ parameters: where are the other $(N-1)(N-2)/2$ degrees of freedom needed for arbitrary rotations?

As discussed in (Hanson 1992), the non-commutativity of the rotation group allows us to generate all the other rotations by *small circular motions* of the controller in the $(N-1)$-dimensional subspace of $\vec{\mathrm{x}}$. Moving the controller in circles in the $(1,2)$-plane, $(1,3)$-plane, etc., of the $(N-1)$-dimensional controller exactly generates the missing $(N-1)(N-2)/2$ rotations required to exhaust the full parameter space. In mathematical terms, the additional motions are generated by the commutation relations of the $SO(N)$ Lie algebra for $i,j = 1,\ldots,N-1$,

$$\begin{aligned}[R_{iN}, R_{jN}] &= \delta_{ij}R_{NN} - \delta_{jN}R_{iN} + \delta_{iN}R_{jN} - \delta_{NN}R_{ij} \\ &= -R_{ij} \ .\end{aligned}$$

The minus sign in the above equation means that *clockwise* controller motions in the (i, j)-plane inevitably produce *counterclockwise* rotations of the object. Thus the philosophy presented in (Hanson 1992) of achieving the full set of context-free rotation group transformations with a limited set of controller moves extends perfectly to N-dimensions. *Implementation Note*: In practice, the effectiveness of this technique varies considerably with the application; the size of the counter-rotation experienced may be very small for parameters that give appropriate spatial motion sensitivity with current 3D mouse technology.

Alternative Context Philosophies. The rolling ball interface is a *context-free* interface which allows the user of a virtual reality context to ignore the absolute position of the controller and requires no supplementary cursor context display; thus one may avoid distractions that may disturb stereography and immersive effects in a virtual reality environment. However some applications are better adapted to *context-sensitive* interfaces like the Arcball method of (Shoemake 1994) or the *Virtual Sphere* approach of (Chen et al. 1988). The Virtual Sphere approach in particular can be straightforwardly extended to higher dimensions by using the rolling ball equations inside a displayed spatial context (typically a sphere) and changing over to an $(N-1)$-dimensional rolling ball outside the context; that is, as the controller approaches and passes the displayed inner domain context sphere, the rotation action changes to one that leaves the N-th coordinate fixed but changes the remaining $(N-1)$ coordinates as though an $(N-1)$-dimensional rolling ball controller were attached to the nearest point on the sphere. Similar flexibility can be achieved by using a different controller state to signal a discrete rather than a continuous context switch to the $(N-1)$-dimensional controller.

◊ Handy Formulas for N-Dimensional Rotations ◊

For some applications the incremental orientation control methods described above are not as useful as knowing a single matrix that immediately gives the entire N-dimensional orientation frame for an object. We note three ways to represent such an orientation frame:

Columns are new axes. One straightforward construction simply notes that if the default coordinate frame is represented by the orthonormal set of unit vectors $\hat{\mathbf{x}}_1 = (1, 0, \ldots, 0)$, $\hat{\mathbf{x}}_2 = (0, 1, 0, \ldots, 0)$, ..., $\hat{\mathbf{x}}_N = (0, \ldots, 0, 1)$, and the desired axes of the new (orthonormal) coordinate frame are known to be $\hat{\mathbf{a}}_1 = (a_1^{(1)}, a_1^{(2)}, \ldots, a_1^{(N)})$, $\hat{\mathbf{a}}_2$, ..., $\hat{\mathbf{a}}_N$, then the rotation matrix that transforms any vector to that frame just has the new axes as its columns:

$$M = \left[\begin{array}{cccc} \hat{\mathbf{a}}_1 & \hat{\mathbf{a}}_2 & \cdots & \hat{\mathbf{a}}_N \end{array}\right]$$

The reader may verify that the orthonormality constraints give M the required $N(N-1)/2$ degrees of freedom.

Concatenated subplane rotations. The rotations in the plane of a pair of coordinate axes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$, $i, j = 1, \ldots, N$ can be written as

$$R_{ij}(\theta_{ij}) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos\theta_{ij} & 0 & \cdots & 0 & -\sin\theta_{ij} & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \sin\theta_{ij} & 0 & \cdots & 0 & \cos\theta_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

and thus the $N(N-1)/2$ distinct $R_{ij}(\theta_{ij})$ may be concatenated in some order to produce a rotation matrix such as

$$M = \prod_{i<j} R_{ij}(\theta_{ij})$$

with $N(N-1)/2$ degrees of freedom parametrized by $\{\theta_{ij}\}$. However, since the matrices R_{ij} do not commute, different orderings give different results and it is difficult to intuitively understand the global rotation. In fact, as is the case for 3D Euler angles, one may even repeat some matrices and omit others, and still not miss any degrees of freedom.

Quotient Space Decomposition. A more intuitively controllable decomposition relies on the classic quotient property of the topological spaces of the orthogonal groups (Helgason 1962),

$$SO(N)/SO(N-1) = S^{N-1} , \tag{8}$$

where S^K is a K-dimensional topological sphere. In practical terms, this means that the $N(N-1)/2$ parameters of $SO(N)$, the mathematical group of N-dimensional orthogonal rotations that we are concerned with, can be viewed as a nested family of points on spheres.

Suppose we let $\mathrm{Par}(S^K) = K$ denote the number of free parameters describing a point on a K-sphere (e.g., the surface of the balloon-like 2-sphere S^2 has two parameters, azimuth and elevation). Then Eq. (8) can be applied repeatedly to compute the total number of parameters in the group as:

$$\begin{aligned}\mathrm{Par}(SO(N)) &= \mathrm{Par}(S^{N-1}) + \cdots + \mathrm{Par}(S^2) + \mathrm{Par}(S^1) \\ &= (N-1) + (N-2) + \cdots + 2 + 1 \\ &= N(N-1)/2 \,.\end{aligned}$$

An implementable formula may be derived starting with the explicit matrix needed to rotate a canonical point $(0, 0, \ldots, 0, 1)$ on the unit K-sphere S^K to an arbitrary point $\hat{\mathbf{n}}$, with $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$, on the K-sphere lying in a $(K+1)$-dimensional subspace of N-dimensional Euclidean space. First, we define the matrix

$$\begin{aligned} R_{K+1} = & \begin{bmatrix} \frac{n_2}{r_2} & \frac{n_1}{r_2} & 0 & & \\ -\frac{n_1}{r_2} & \frac{n_2}{r_2} & 0 & & \\ 0 & 0 & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & & \\ 0 & \frac{n_3}{r_3} & \frac{r_2}{r_3} & & \\ 0 & -\frac{r_2}{r_3} & \frac{n_3}{r_3} & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \cdots \\ & \cdots \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{n_K}{r_K} & \frac{r_{K-1}}{r_K} & 0 \\ 0 & \cdots & -\frac{r_{K-1}}{r_K} & \frac{n_K}{r_K} & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & n_{K+1} & r_K \\ 0 & \cdots & 0 & -r_K & n_{K+1} \end{bmatrix} , \end{aligned}$$

where $(r_i)^2 = \sum_{j=1}^{i} (n_j)^2$, so that $(r_{K+1})^2 = \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$. The reader may verify that this matrix (with an appropriate set of $(N-K-1)$ trailing 1's down the diagonal) takes the vector with a 1 in the $(K+1)$-st position into the unit N-vector $\hat{\mathbf{n}} = (n_1, n_2, \ldots, n_K, n_{K+1}, 0, \ldots, 0)$. *Note*: These matrices can be chosen in many alternate ways.

The fundamental rotation algorithm now requires the user to choose a sequence of subspaces terminating finally in a one-parameter rotation in a single plane; that is,

- Choose a point $\hat{\mathbf{n}}$ on the $(N-1)$-sphere embedded in N dimensions that defines, via $\hat{\mathbf{n}} \cdot \vec{\mathbf{x}} = 0$, an $(N-1)$-dimensional Euclidean subspace. The matrix $(R_N)^{-1}$ transforms $\hat{\mathbf{n}}$ to lie on the N-th axis.
- Choose a point $\hat{\mathbf{n}}'$ on the $(N-2)$-sphere embedded in the just defined $(N-1)$-dimensional space; $\hat{\mathbf{n}}' \cdot \vec{\mathbf{x}} = 0$ defines an $(N-2)$-dimensional Euclidean subspace. $(R_{N-1})^{-1}$ transforms $\hat{\mathbf{n}}'$ to lie on the $(N-1)$-st axis.
- Repeat K times, terminating when the current $\hat{\mathbf{n}} = (0, 0, 1, 0, \ldots, 0)$ lies on the 3rd axes, or when $K = N-2$. For example, when $N = 3$, one reduction suffices; for $N = 4$, two steps are needed, and so on.

The final matrix, with exactly $N(N-1)/2$ free parameters, is

$$M_N = R_N \cdots R_3 R_0 (R_3)^{-1} \cdots (R_N)^{-1} \ , \tag{9}$$

where

$$R_0 = \begin{bmatrix} c & -s & & & \\ s & c & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

(with $c = \cos\theta$, $s = \sin\theta$) performs the final clockwise rotation in the (x_1, x_2)-plane.

In 3D, for example,

$$M_3(\theta, \hat{\mathbf{n}}) = \begin{bmatrix} c + (n_1)^2(1-c) & n_1 n_2(1-c) - s n_3 & n_3 n_1(1-c) + s n_2 \\ n_1 n_2(1-c) + s n_3 & c + (n_2)^2(1-c) & n_3 n_2(1-c) - s n_1 \\ n_1 n_3(1-c) - s n_2 & n_2 n_3(1-c) + s n_1 & c + (n_3)^2(1-c) \end{bmatrix} \tag{10}$$

where $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = (n_1)^2 + (n_2)^2 + (n_3)^2 = 1$ reduces the number of free parameters to the three Euler angles.

In 4D, we would let $\hat{\mathbf{m}} = (m_1, m_2, m_3, m_4)$ be the required 4D unit vector and $\hat{\mathbf{n}} = (n_1, n_2, n_3)$ the 3D unit vector, and proceed to merge the matrices to compute $M_4(\theta, \hat{\mathbf{n}}, \hat{\mathbf{m}})$; it is probably best to implement this directly using the matrix multiplications, as the expression analogous to Eq. (10) is outrageously long. We verify that the three constraints $\hat{\mathbf{m}} \cdot \hat{\mathbf{m}} = \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = c^2 + s^2 = 1$ on the nine free variables $\hat{\mathbf{m}}$, $\hat{\mathbf{n}}$, and (c, s) reduce the total number of free parameters to the required six angles.

◇ Interpolating N-Dimensional Orientation Frames ◇

Given two N-dimensional orientation frames, how can we define a uniform-angular-velocity interpolation between them? One approach would be to take the hierarchy of points on the spheres $(S^{N-1}, \ldots, S^2, S^1)$ and apply a constant angular velocity spherical interpolation or "Slerp" to each:

$$\hat{\mathbf{n}}_{12}(t) = \mathrm{Slerp}(\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, t) = \hat{\mathbf{n}}_1 \frac{\sin((1-t)\theta)}{\sin(\theta)} + \hat{\mathbf{n}}_2 \frac{\sin(t\theta)}{\sin(\theta)}$$

where $\cos\theta = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2$. This formula is simply the result of applying a Gram-Schmidt decomposition while enforcing unit norm. Arbitrary splines can be defined in each subspace using this combined with the method of (Schlag 1991) in Gems II.

This achieves the goal of smooth appearance, but the solution is neither unique nor mathematically compelling, since the curve is not guaranteed to be a geodesic in $SO(N)$.

The arbitrariness of chosen matrix orders in M_N can be exploited to give qualitatively similar but different interpolations.

The specification of geodesic curves in $SO(N)$ is a difficult problem in general (Barr et al. 1992); fortunately, the two most important cases for interactive systems, $N = 3$ and $N = 4$, have elegant solutions using the covering or "Spin" groups. For $SO(3)$, geodesic interpolations and suitable corresponding splines are definable using Shoemake's quaternion splines (Shoemake 1985), which can be simply formulated using Slerps on S^3 as follows: let $\vec{\mathbf{q}}$ be a 3-vector of any length and $\hat{\mathbf{n}}$ a unit 3-vector, so that

$$q_0 = \cos(\theta/2), \quad \vec{\mathbf{q}} = \hat{\mathbf{n}} \sin(\theta/2)$$

is automatically a point on S^3 due to the constraint $(q_0)^2 + (q_1)^2 + (q_2)^2 + (q_3)^2 = 1$. Then each point on S^3 corresponds to an $SO(3)$ rotation matrix

$$R_3 = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}$$

which the reader can verify reduces exactly to the usual form in Eq. (10). In addition, two distinct points on S^3, $(q_0, \vec{\mathbf{q}})$ and $(-q_0, -\vec{\mathbf{q}})$, obviously correspond to the same rotation R_3. Slerping q generates sequences of matrices R_3 that are geodesic interpolations.

Quaternions in Four Dimensions. In the four-dimensional case, which we should really regard as the more fundamental one since it includes the 3D transformation as a special case, we can find the induced $SO(4)$ matrix by extending quaternion multiplication to act on full quaternions v^μ and not just 3-vectors ("pure" quaternions) $v = (0, \vec{\mathbf{V}})$ in the following way:

$$\sum_{\nu=0}^{3} R^\mu{}_\nu v^\nu = q \cdot v^\mu \cdot p^{-1} \ .$$

Working out the algebra, we find that R_3 given above is just the degenerate $p = q$ case of the 4D rotation matrix

$$R_4 = \begin{bmatrix} q_0p_0 + q_1p_1 + q_2p_2 + q_3p_3 & q_1p_0 - q_0p_1 - q_3p_2 + q_2p_3 & q_2p_0 - q_0p_2 - q_1p_3 + q_3p_1 & q_3p_0 - q_0p_3 - q_2p_1 + q_1p_2 \\ -q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3 & q_0p_0 + q_1p_1 - q_2p_2 - q_3p_3 & q_1p_2 + p_1q_2 - p_0q_3 - q_0p_3 & q_1p_3 + p_1q_3 + p_0q_2 + q_0p_2 \\ -q_2p_0 + q_0p_2 - q_1p_3 + q_3p_1 & q_1p_2 + q_2p_1 + q_0p_3 + q_3p_0 & q_0p_0 + q_2p_2 - q_1p_1 - q_3p_3 & q_2p_3 + q_3p_2 - q_0p_1 - q_1p_0 \\ -q_3p_0 + q_0p_3 - q_2p_1 + q_1p_2 & q_1p_3 + q_3p_1 - q_0p_2 - q_2p_0 & q_2p_3 + q_3p_2 + q_1p_0 + p_0q_1 & q_0p_0 + q_3p_3 - q_1p_1 - q_2p_2 \end{bmatrix} \tag{11}$$

We may take this form and plug in

$$p_0 = \cos(\phi/2), \quad \vec{\mathbf{p}} = \hat{\mathbf{m}} \sin(\phi/2)$$

to get a new form of the 4D orthogonal rotation matrix *parameterized in terms of two separate 3-sphere coordinates*:

$$R_4 = \frac{1}{2}\left[\begin{array}{cccc}
C_+ + C_- + \hat{\mathbf{m}}\cdot\hat{\mathbf{n}}(C_- - C_+) & -m^-_{23}C_- + m^-_{23}C_+ - m^+_1 S_- - m^-_1 S_+ & -m^-_{31}C_- + m^-_{31}C_+ - m^+_2 S_- - m^-_2 S_+ & -m^-_{12}C_- + m^-_{12}C_+ - m^+_3 S_- - m^-_3 S_+ \\
-m^-_{23}C_- + m^-_{23}C_+ + m^+_1 S_- + m^-_1 S_+ & C_+ + C_- + (m_1n_1 - m_2n_2 - m_3n_3)(C_- - C_+) & m^+_{12}C_- - m^+_{12}C_+ - m^-_3 S_- - m^+_3 S_+ & m^+_{31}C_- - m^+_{31}C_+ + m^-_2 S_- + m^+_2 S_+ \\
-m^-_{31}C_- + m^-_{31}C_+ + m^+_2 S_- + m^-_2 S_+ & m^+_{12}C_- - m^+_{12}C_+ + m^-_3 S_- + m^+_3 S_+ & C_+ + C_- + (-m_1n_1 + m_2n_2 - m_3n_3)(C_- - C_+) & m^+_{23}C_- - m^+_{23}C_+ - m^-_1 S_- - m^+_1 S_+ \\
-m^-_{12}C_- + m^-_{12}C_+ + m^+_3 S_- + m^-_3 S_+ & m^+_{31}C_- - m^+_{31}C_+ - m^-_2 S_- - m^+_2 S_+ & m^+_{23}C_- - m^+_{23}C_+ + m^-_1 S_- + m^+_1 S_+ & C_+ + C_- + (-m_1n_1 - m_2n_2 + m_3n_3)(C_- - C_+)
\end{array}\right] \tag{12}$$

where $C_\pm = \cos\frac{1}{2}(\phi \pm \theta)$, $S_\pm = \sin\frac{1}{2}(\phi \pm \theta)$, $m^\pm_i = (m_i \pm n_i)$, and $m^\pm_{ij} = (m_i n_j \pm m_j n_i)$.

Shoemake-style interpolation between two distinct 4D frames is now achieved by applying the desired Slerp-based interpolation method independently to a set of coordinates $q(t)$ on one three-sphere, and to a separate set of coordinates $p(t)$ on another. The resulting matrix $R_4(t)$ gives geodesic interpolations for simple Slerps, and smooth interpolations based on infinitesimal geodesic components when the spline methods of (Schlag 1991) or (Barr et al. 1992) are used.

Controls. As pointed out in (Shoemake 1994), the Arcball controller can be adapted with complete faithfulness of spirit to the 4D case, since one can pick *two* points in a three-sphere to specify an initial 4D frame, and then pick *two more* points in the three-sphere to define the current 4D frame. Note that Eq. (11) gives the complete 4D rotation formula. Alternately, one can replace the 4D rolling ball or virtual sphere controls described at the beginning by a pair (or more) of 3D controllers as noted in (Hanson 1992).

Higher Dimensions. The spin groups for dimensions higher than four contain highly suggestive subgroups that are isomorphic to the quaternion $SU(2)$ algebra that we have used here for $SO(3)$ and for $SO(4)$, whose covering group surprisingly decomposes into a direct product of two quaternions. However, the higher dimensional covering groups unfortunately consist of much more complex topological spaces. Our preliminary investigations of these spaces suggest that there should exist methods for specifying higher-dimensional geodesic interpolations and splines that are not radically different from dimensions three and four, but this is a topic for future work.

Acknowledgment. This work was supported in part by NSF grant IRI-91-06389.

◊ Bibliography ◊

(Barr et al. 1992) Alan H. Barr, Bena Currin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics*, 26(2):313–320, 1992.

(Chen et al. 1988) Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *Proceedings of Siggraph 88*, volume 22, pages 121–130, 1988.

(Coxeter 1991) H.S.M. Coxeter. *Regular Complex Polytopes.* Cambridge University Press, second edition, 1991.

(Efimov and Rozendorn 1975) N.V. Efimov and E.R. Rozendorn. *Linear Algebra and Multi-Dimensional Geometry.* Mir Publishers, Moscow, 1975.

(Hanson 1992) Andrew J. Hanson. The rolling ball. In David Kirk, editor, *Graphics Gems III*, pages 51–60. Academic Press, San Diego, CA, 1992.

(Hanson 1994) Andrew J. Hanson. Geometry for n-dimensional graphics. In Paul Heckbert, editor, *Graphics Gems IV*, pages 149–170. Academic Press, Cambridge, MA, 1994.

(Helgason 1962) Sigurdur Helgason. *Differential Geometry and Symmetric Spaces.* Academic Press, New York, 1962.

(Hocking and Young 1961) John G. Hocking and Gail S. Young. *Topology.* Addison-Wesley, 1961.

(Schlag 1991) John Schlag. Using geometric constructions to interpolate orientations with quaternions. In James Arvo, editor, *Graphics Gems II*, pages 377–380. Academic Press, Cambridge, MA, 1991.

(Shoemake 1985) K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics*, volume 19, pages 245–254, 1985. Proceedings of SIGGRAPH 1985.

(Shoemake 1994) Ken Shoemake. Arcball rotation control. In Paul Heckbert, editor, *Graphics Gems IV*, pages 172–192. Academic Press, Cambridge, MA, 1994.

(Sommerville 1958) D.M.Y. Sommerville. *An Introduction to the Geometry of N Dimensions.* Reprinted by Dover Press, 1958.