◊ II.4

Rotations for N-Dimensional Graphics

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

◊ Introduction ◊

In a previous Gem (Hanson 1994), "Geometry for N-Dimensional Graphics," we described a 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 rotations in N-dimensional Euclidean space. In particular, we give a 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 touch on practical methods for specifying and understanding the parameters of N-dimensional rotations. Finally, we give the explicit 4D extension of 3D quaternion orientation splines.

For additional details and insights, 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 of the 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 (towards the "north pole" of the ball) and tips it in the direction of an arbitrary unit vector $\hat{\mathbf{n}} = (n_1, n_2, \ldots, n_{N-1}, 0)$ lying in the $(N-1)$-plane tangent to the ball at the north pole, thus producing a new, rotated unit vector $\hat{\mathbf{v}}$, where

$$\hat{\mathbf{v}} = M_N \cdot \hat{\mathbf{v}}_0 = \hat{\mathbf{n}} \sin\theta + \hat{\mathbf{v}}_0 \cos\theta \ ,$$

as indicated schematically in Figure 1a. (Note: for notational simplicity, we choose to write the components of column vectors as horizontal lists.)

If we choose the convention that positive rotations are right-handed and progress counterclockwise, a positive rotation of the north pole actually tilts it into the negative direction of the remaining axis of the rotation plane. That is, if the 2D "rolling circle" acts on $\hat{\mathbf{v}}_0 = (0, 1)$ and

ISBN 0-12-543457-X

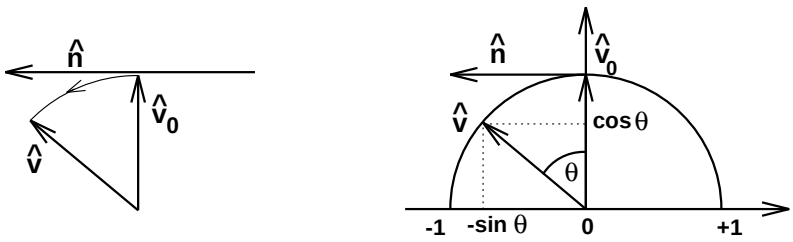

Figure 1. Tilting the "north pole" vector $\hat{\mathbf{v}}_0$ in the direction of the tangent vector $\hat{\mathbf{n}}$, as though rolling a ball by placing one's finger directly on the north pole and pulling in the direction $\hat{\mathbf{n}}$.

$\hat{\mathbf{n}} = (-1, 0)$ as shown in Figure 1b, then

$$\hat{\mathbf{v}} = M_2 \cdot \hat{\mathbf{v}}_0 = \hat{\mathbf{n}} \sin\theta + \hat{\mathbf{v}}_0 \cos\theta = (-\sin\theta,\, \cos\theta)\,,$$

where the rotation matrix M_2 can be written

$$\begin{aligned} M_2 &= \begin{bmatrix} \cos\theta & -\sin\theta \\ +\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & -s \\ +s & c \end{bmatrix} \\ &= \begin{bmatrix} c & +n_x s \\ -n_x s & c \end{bmatrix}\,. \end{aligned} \tag{1}$$

If we choose a right-handed overall coordinate frame, the sign of $\hat{\mathbf{n}}$ will automatically generate the correct sign convention.

Synopsis: Qualitatively speaking, if we imagine looking straight down at the north pole, the rolling ball *pulls* the unseen N-th component of a vector along the direction $\hat{\mathbf{n}}$ of the $(N-1)$-dimensional controller motion, bringing the unseen component gradually into view.

Implementation. In practice, we choose a radius R for the ball containing the object or scene to be rotated and move our controller (slider, 2D mouse, 3D mouse, ...) a distance r in the tangent direction $\hat{\mathbf{n}}$, as indicated in Figure 2a. Working from the simplified diagram in Figure 2b, we define $D^2 = R^2 + r^2$ and choose the rotation parameters $c = \cos\theta = R/D$ and $s = \sin\theta = r/D$.

For interactive systems, this choice has the particular advantage that, however rapidly the user moves the controller, $0 \le (r/D) < +1$, so $0 \le \theta < \pi/2$. Depending upon the desired interface behavior, an alternative choice would be to take $\theta = r/R$. This requires computing a trigonometric function instead of a square root, and may cause large discontinuities in orientation for large controller motion.

3D. The explicit 3D rolling ball formula can be derived starting from an arbitrary 2D mouse displacement $\vec{\mathbf{r}} = (x, y, 0) = (rn_x, rn_y, 0)$, where $n_x^2 + n_y^2 = 1$. Then one replaces Equation

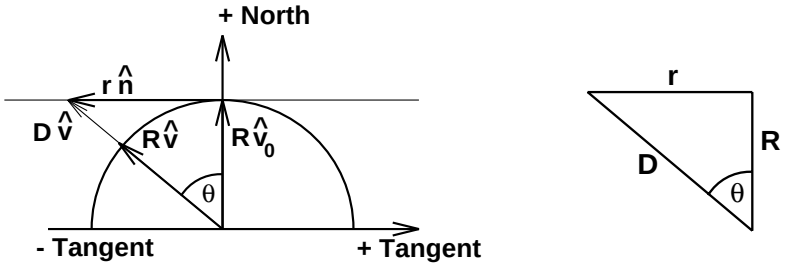

Figure 2. The notation used in implementing the rolling ball rotation model for N dimensions.

(1) with $n_x = +1$ by the analogous 3×3 matrix R_0 for (x, z) rotations and encloses this in a conjugate pair of rotations R_{xy} that transform the 2D mouse displacement $\vec{\mathbf{r}}$ into the strictly positive x-direction and back. Since even $\vec{\mathbf{r}} = (-1, 0, 0)$ is rotated to the positive x-direction before R_0 acts, all signs are correct. With the explicit matrices

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

we find an alternative derivation of the formula in our earlier Gem (Hanson 1992):

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

4D. The 4D case takes as input a 3D mouse motion $\vec{\mathbf{r}} = (x, y, z, 0) = (rn_x, rn_y, rn_z, 0)$, with $n_x^2 + n_y^2 + n_z^2 = 1$. Then one first transforms (n_y, n_z) into a pure y-component, rotates that result to yield a pure x-component, performs a rotation by θ in the (x, w)-plane, and reverses the first two rotations. Defining the required matrices as

$$R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n_y}{r_{yz}} & -\frac{n_z}{r_{yz}} & 0 \\ 0 & \frac{n_z}{r_{yz}} & \frac{n_y}{r_{yz}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_{xy} = \begin{bmatrix} n_x & -r_{yz} & 0 & 0 \\ r_{yz} & n_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_0 = \begin{bmatrix} c & 0 & 0 & +s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s & 0 & 0 & c \end{bmatrix}, \tag{3}$$

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

$$\begin{aligned} M_4 &= R_{yz} R_{xy} R_0 (R_{xy})^{-1} (R_{yz})^{-1} \\ &= \begin{bmatrix} 1-(n_x)^2(1-c) & -(1-c)n_x n_y & -(1-c)n_x n_z & s n_x \\ -(1-c)n_x n_y & 1-(n_y)^2(1-c) & -(1-c)n_y n_z & s n_y \\ -(1-c)n_x n_z & -(1-c)n_y n_z & 1-(n_z)^2(1-c) & s n_z \\ -s n_x & -s n_y & -s n_z & c \end{bmatrix} . \end{aligned} \tag{4}$$

ND. The extension of this procedure to any dimension is accomplished by having the controller interface supply an $(N-1)$-dimensional vector $\vec{\mathbf{r}} = (r n_1, r n_2, \ldots, r n_{N-1}, 0)$ with $\vec{\mathbf{r}} \cdot \vec{\mathbf{r}} = r^2$ and $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$ and applying the rotation

$$\begin{aligned} M_N &= R_{N-2,N-1} R_{N-3,N-2} \cdots R_{1,2} R_0 (R_{1,2})^{-1} \cdots (R_{N-3,N-2})^{-1} (R_{N-2,N-1})^{-1} \\ &= \begin{bmatrix} 1-(n_1)^2(1-c) & -(1-c)n_2 n_1 & \cdots & -(1-c)n_{N-1} n_1 & s n_1 \\ -(1-c)n_1 n_2 & 1-(n_2)^2(1-c) & \cdots & -(1-c)n_{N-1} n_2 & s n_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(1-c)n_1 n_{N-1} & -(1-c)n_2 n_{N-1} & \cdots & 1-(n_{N-1})^2(1-c) & s n_{N-1} \\ -s n_1 & -s n_2 & \cdots & -s n_{N-1} & c \end{bmatrix} \end{aligned} \tag{5}$$

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

♢ Controlling the Remaining Rotational 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 (5) has only $(N-1)$ independent parameters: we must now understand what happened to the other $(N-1)(N-2)/2$ degrees of freedom needed for arbitrary rotations.

In fact, 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 $\vec{\mathbf{r}} = r\hat{\mathbf{n}}$. Moving the controller in circles in the $(1,2)$-plane, $(1,3)$-plane, etc., of the $(N-1)$-dimensional controller space 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, and vice-versa. Thus the philosophy (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 relatively 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 that allows the user of a virtual reality application 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 (Shoemake 1994) or the Virtual Sphere approach (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 for 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 orthonormality constraints give M the required $N(N-1)/2$ degrees of freedom.

Concatenated subplane rotations. 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 the block matrix

$$R_{ij}(\theta_{ij}) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos\theta_{ij} & 0 & \cdots & 0 & -\sin\theta_{ij} & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \sin\theta_{ij} & 0 & \cdots & 0 & \cos\theta_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{6}$$

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 (with distinct parameters) and omit others, and still not miss any degrees of freedom.

Quotient Space Decomposition. Another useful decomposition relies on the classic quotient property of the topological spaces of the orthogonal groups (Helgason 1962),

$$SO(N)/SO(N-1) = S^{N-1}\,, \tag{7}$$

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, can be viewed as a nested family of points on spheres. The 2D form is the matrix (1) parameterizing the points on the circle S^1; the 3D form reduces to the standard matrix

$$M_3(\theta, \hat{\mathbf{n}}) = \begin{bmatrix} c + (n_1)^2(1-c) & n_1 n_2(1-c) - s n_3 & n_3 n_1(1-c) + s n_2 \\ n_1 n_2(1-c) + s n_3 & c + (n_2)^2(1-c) & n_3 n_2(1-c) - s n_1 \\ n_1 n_3(1-c) - s n_2 & n_2 n_3(1-c) + s n_1 & c + (n_3)^2(1-c) \end{bmatrix} \tag{8}$$

where the two free parameters of $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = (n_1)^2 + (n_2)^2 + (n_3)^2 = 1$ describe a point on the 2-sphere. These two parameters plus a third from the S^1 described by $c^2 + s^2 = 1$ (i.e., $c = \cos\theta$, $s = \sin\theta$) yield the required total of three free parameters equivalent to the three Euler angles. The 4D and higher forms are already too unwieldy to be conveniently written as single matrices.

♢ Interpolating N-Dimensional Orientation Frames ♢

To define a uniform-angular-velocity interpolation between two N-dimensional orientation frames, we might consider independently interpolating each angle in Equation (6), or we might take the quotient space decomposition given by the hierarchy of points on the spheres $(S^{N-1}, \ldots, S^2, S^1)$ and apply a constant angular velocity spherical interpolation to each spherical point in each successive dimension using the "Slerp"

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

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 in any dimension.)

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

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 $\hat{\mathbf{n}}$ be 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

$$R_3 = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{9}$$

which the reader can verify reduces exactly to the nested-sphere form in Equation (8). Note that the quaternions q and $-q$ each correspond to the same 3D rotation. Slerping q generates sequences of matrices $R_3(t)$ that are geodesic interpolations. Arbitrary splines can be defined using the method of Schlag (Schlag 1991).

Quaternions in Four Dimensions. In four dimensions, the correspondence between the rotation group $SO(4)$ and the spin group $\text{Spin}(4)$ that is its double covering may be computed by extending quaternion multiplication to act not just on 3-vectors ("pure" quaternions) $v = (0, \vec{\mathbf{V}})$, but on full 4-vector quaternions v^μ in the following way:

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

We thus find that the general double-quaternion parameterization for 4D rotation matrices takes the form

$$R_4 = \begin{bmatrix} q_0p_0 + q_1p_1 + q_2p_2 + q_3p_3 & q_1p_0 - q_0p_1 - q_3p_2 + q_2p_3 \\ -q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3 & q_0p_0 + q_1p_1 - q_2p_2 - q_3p_3 \\ -q_2p_0 + q_0p_2 - q_1p_3 + q_3p_1 & q_1p_2 + q_2p_1 + q_0p_3 + q_3p_0 \\ -q_3p_0 + q_0p_3 - q_2p_1 + q_1p_2 & q_1p_3 + q_3p_1 - q_0p_2 - q_2p_0 \end{bmatrix}$$
$$\begin{bmatrix} q_2p_0 - q_0p_2 - q_1p_3 + q_3p_1 & q_3p_0 - q_0p_3 - q_2p_1 + q_1p_2 \\ q_1p_2 + p_1q_2 - p_0q_3 - q_0p_3 & q_1p_3 + p_1q_3 + p_0q_2 + q_0p_2 \\ q_0p_0 + q_2p_2 - q_1p_1 - q_3p_3 & q_2p_3 + q_3p_2 - q_0p_1 - q_1p_0 \\ q_2p_3 + q_3p_2 + q_1p_0 + p_0q_1 & q_0p_0 + q_3p_3 - q_1p_1 - q_2p_2 \end{bmatrix} . \quad (10)$$

One may check that Equation (9) is just the lower right-hand corner of the degenerate $p = q$ case of Equation (10).

Shoemake-style interpolation between two distinct 4D frames is now achieved by applying the desired Slerp-based interpolation method independently to a set of quaternion coordinates $q(t)$ on one three-sphere, and to a separate set of quaternion coordinates $p(t)$ on another. The resulting matrix $R_4(t)$ gives geodesic interpolations for simple Slerps, and can be used as the basis for corresponding spline methods (Schlag 1991,Barr et al. 1992). Analogs of the $N = 3$ and $N = 4$ approaches for general N involve computing Spin(N) geodesics and thus are quite complex.

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. Equation (10) gives the complete form of the effective 4D rotation. Alternately, one can replace the 4D rolling ball or Virtual Sphere controls described earlier by a pair (or more) of 3D controllers (Hanson 1992).

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, 1992.

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

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