The 4D Rolling Ball

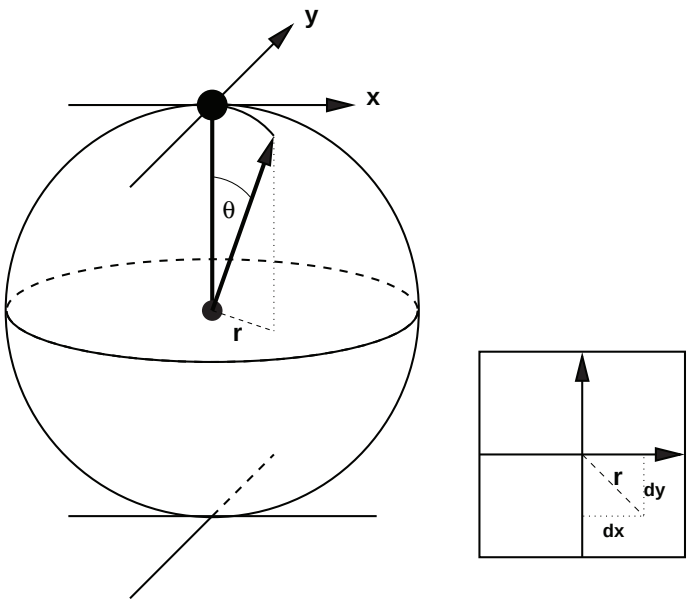

Figure 1: The 3D Rolling Ball approach to 3D orientation control.

3D Orientation Control. A fundamental problem in the design of interactive systems for exploring geometric objects is that we cannot see them all at once. A standard solution is to provide interactive orientation control, so that, by rotating the object at will, all sides and aspects of a 3D object can be seen and explored. Many different methods have been used to control the three degrees of freedom defining the orientation of a 3D object. These include products of rotations in the three Euclidean planes of three dimensions, (y, z), (z, x), and (x, y). Note that these three fundamental rotations correspond conveniently to rotations about the x axis, the y axis, and the z axis in 3D, but the rotation-about-an-axis concept breaks down in 4D. Standard 3D Euler angles describe the dynamics of a gyroscope and correspond to a (z-axis $\times$ y-axis $\times$ z-axis) sequence; many computer graphics applications choose some variation of the (x-axis $\times$ y-axis $\times$ z-axis) sequence. The problem with this type of rotation control is that the parameters do not change in a way that simulates a natural motion like holding a baseball in your hand and rotating it to examine the pattern of the seams. The 3D analog of the rotation

algorithm we will adopt for 4D control does exactly this: it rotates the object of interest as though encased in a glass ball with the object at its center. As shown in the Figure above, if you look at the point on this virtual glass ball closest to your viewpoint, the *rolling ball* control simply tilts that point in any direction you choose, incrementally hiding those parts of the object in the direction you tilt the center point, and incrementally exposing unseen parts of the object on the opposite side. One can thus instantly understand what is being made visible and what is becoming hidden as a consequence of the rotation. After each rotation, a new point on the virtual glass ball is closest to the viewpoint, and one starts afresh, with no need to maintain any knowledge of context. This contrasts with a number of other clever, but controller-position dependent, methods that rotate about the vertical axis under some conditions instead of always tilting the nearest point.

The equations of the 3D rolling ball algorithm are quite simple, and will help us in a moment to understand the strong analogy to the powerful 4D rolling ball orientation control method. We look straight down on the virtual glass ball containing our object at the center, locate the nearest point (at the middle of the screen), and define an arbitrary small displacement of the center in screen coordinates by a 2D vector $(dx,\, dy)$, let $r = (dx^2 + dy^2)^{1/2}$ be the Euclidean length of this vector, and choose a scaling size R to define the virtual radius of the glass sphere. Then if the standard rotation (supported, for example, in OpenGL) by an angle θ about a unit-length direction $\hat{n} = (n_x, n_y, n_z)$ is written as

$$\mathrm{glRotatef}(\theta, \hat{n}) \ ,$$

the required parameters are simply

$$\begin{aligned} \tan\theta &= \frac{r}{R} \\ \cos\theta &= \frac{R}{\sqrt{r^2 + R^2}} \\ \sin\theta &= \frac{r}{\sqrt{r^2 + R^2}} \\ \hat{n} &= \left(\frac{-dy}{r},\, \frac{dx}{r},\, 0\right) \ . \end{aligned}$$

This rotation will always tilt the invisible axis "pointing straight at you" in the direction $(dx,\, dy)$ that you choose, allowing you to explore any part of a 3D

object. Its explicit form, with $c = \cos\theta$ and $s = \sin\theta$, is

$$\begin{pmatrix} c\,n_x^2 + n_y^2 & (c-1)\,n_x n_y & n_x\,s \\ (c-1)\,n_x n_y & n_x^2 + c\,n_y^2 & n_y\,s \\ -n_x\,s & -n_y\,s & c \end{pmatrix} .$$

Using the fact that $n_x^2 + n_y^2 = 1$ gives the computationally less expensive form

$$\begin{pmatrix} 1 + (c-1)\,n_x^2 & (c-1)\,n_x n_y & n_x\,s \\ (c-1)\,n_x n_y & 1 + (c-1)\,n_y^2 & n_y\,s \\ -n_x\,s & -n_y\,s & c \end{pmatrix} ,$$

where we note that since $n_i\,s = (x_i/r)(r/(r^2 + R^2)^{1/2})$, we can eliminate the r from these terms for further efficiency.

A Small Piece of Group Theory Saves the Day. There is, however, one potential problem: this method has only *two* parameters, $(dx,\, dy)$. We know 3D rotations have *three* parameters. Have we lost something? Thanks to the secret properties of order-dependent, or non-Abelian, groups, to which 3D rotations belong, we can *always* find a sequence of these two-parameter rotations that will take us to *any* given orientation in 3D. This is *not* true of two-parameter rotation methods that use, say, the global latitude and longitude of a point on a sphere to implement rotations. More remarkably, we will find in 4D that the group properties of 4D rotations permit us to use just *three* degrees of freedom to obtain any arbitrary orientation in the fourth dimension's *six* degree-of-freedom parameter space! In the next section, we show how the 4D version of the Rolling Ball allows us to do this.

The 4D Rolling Ball. Four dimensions is peculiarly available to interactive exploration because the entire space of possible 4D orientations can be explored with an incremental 3D vector. That is, the precise analog of the 2D vector that controls the 3D Rolling Ball orientation-exploration algorithm is a 3D vector that controls the 4D Rolling Ball. The best way to think of the 4D Rolling Ball conceptually is to realize that the analog of looking at a 3D ball on a 2D screen, with the (x, y) axes visible and the z axis *pointing at you* and therefore *invisible*, is to imagine a 4D object projected to 3D space along the w axis, which is "pointing at you" and therefore invisible. The three (x, y, z) axes are *visible* in the projection

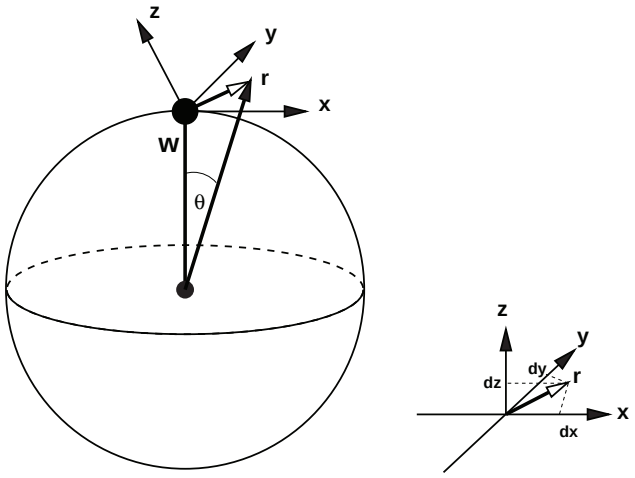

Figure 2: The 4D Rolling Ball approach to 4D orientation control.

from 4D to a 3D world along the w axis. Thus the perfect orientation exploration procedure is to specify a 3D vector (dx, dy, dz) that *tilts* the w axis slightly in the plane defined by the direction (dx, dy, dz) and the w axis itself. That makes a new piece of the geometry appear "on the opposite side" so to speak, and hides an old piece of the geometry on the side the vector is moving towards, exactly the same as for the 3D Rolling Ball, except that we must think of a 3D projection screen as our canvas instead of a 2D projection screen.

The 4D Rolling Ball matrix can literally be guessed from the form of the 3D Rolling Ball matrix, and it takes the form:

$$\left| \begin{array}{cccc} 1+(c-1)\,n_x^2 & (c-1)\,n_x n_y & (c-1)\,n_x n_z & n_x\,s \\ (c-1)\,n_x n_y & 1+(c-1)\,n_y^2 & (c-1)\,n_y n_z & n_y\,s \\ (c-1)\,n_x n_z & (c-1)\,n_y n_z & 1++(c-1)\,n_z^2 & n_z\,s \\ -n_x\,s & -n_y\,s & -n_z\,s & c \end{array} \right| .$$

Here $\hat{n} = (n_x,\, n_y,\, n_z) = \vec{dx}/r$ with $r^2 = \vec{dx} \cdot \vec{dx}$ is a 3D vector derived from the 3D controller input $\vec{dx} = (dx, dy, dz)$, and $c = \cos\theta$, $s = \sin\theta$ with $\theta = \arctan(r/R)$ exactly as in the 3D case. An exercise for the reader is to verify this formula by taking a 3D unit vector $\hat{n}$ embedded in 4D, and rotate it first in the yz plane so the z component is eliminated, then rotate that result in the xy plane to

give a pure x vector; applying a rotation in the xw plane to perform our "tilt" of the w axis, and then undoing the previous xy and yz rotations gives the formula. Symbolically,

$$R_{\mathbf{4Droll}} = R^{-1}(yz) \cdot R^{-1}(xy) \cdot R(\theta, xw) \cdot R(xy) \cdot R(yz) \; .$$

The main task of the interactive system, then, is to create natural ways with existing controllers to access these degrees of freedom; that is of course our main objective in this paper.

Finding All the 4D Degrees of Freedom. It is easy to show (see Box "4D Rotations") that, while 3D orientations have three degrees of freedom, 4D orientations have *six*. How can our three degree-of-freedom 4D Rolling Ball, which apparently only rotates in the three planes (xw, yw, zw), generate the other three degrees of freedom? In fact, any *pair* of directions such as (x, y) will produce a rotation in the *plane of the pair* if we move the controller in *circles* in that plane. Thus, since our three degrees of freedom (x, y, z) correspond to three *pairs* (yz, zx, xy), we actually can produce, incrementally, rotations in all six planes (xw, yw, zw, yz, zx, xy), and therefore any possible 4D orientation can be reached.

The ND Rolling Ball. While we will not need controllers for dimensions greater than four in our treatment here, it is straightforward to extend the idea to rotation controllers for arbitrary Euclidean dimensions. Using either a direct derivation method as noted above or deducing the obvious ND form once we know the 3D and 4D forms, we find the ND Rolling Ball formula with an $(N-1)$-vector $\hat{n} = (n_1, \ldots, n_{N-1})$ providing the control direction to be

$$\begin{vmatrix} 1+(c-1)n_1^2 & (c-1)\,n_1n_2 & \cdots & (c-1)\,n_1n_{N-1} & s\,n_1 \\ (c-1)\,n_1n_2 & 1+(c-1)n_2^2 & \cdots & (c-1)\,n_2n_{N-1} & s\,n_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (c-1)\,n_1n_{N-1} & (c-1)\,n_2n_{N-1} & \cdots & 1+(c-1)n_{N-1}^2 & s\,n_{N-1} \\ -s\,n_1 & -s\,n_2 & \cdots & -s\,n_{N-1} & c \end{vmatrix} .$$

The counting for obtaining the additional degrees of freedom works out as well. If we write the basic rotation matrix in a single plane $(x_i,\, x_N)$ defined by n_i in the formula as the R_{iN} (see Box: 4D Rotation Matrices), then it can be shown that the group algebra of $\mathbf{SO}(N)$ gives a commutator $[R_{iN},\, R_{jN}] = -R_{ij}$. This means

that, by applying the $(N-1)$ ND rolling ball rotation matrices R_{iN} repeatedly to one another, we can generate the $(N-1)(N-2)/2$ degrees of freedom in the R_{ij} by incremental motions. We thus can *always* use an $(N-1)$ degree-of-freedom controller to achieve all of the

$$(N-1) + \frac{1}{2}(N-1)(N-2) = \frac{1}{2}N(N-1)$$

degrees of freedom required to explore the *entire* ND orientation space.

Further Reading. Among the many references that a reader seeking further information might explore, we mention

- A. J. Hanson, "The Rolling Ball," in *Graphics Gems III*, ed. David Kirk, pp. 51–60, Academic Press,Cambridge, MA (1992).
- A. J. Hanson, "Geometry for N-dimensional Graphics," in *Graphics Gems IV*, ed. Paul Heckbert, pp. 149–170, Academic Press,Cambridge, MA (1994).
- A. J. Hanson, "Rotations for N-dimensional Graphics," in *Graphics Gems V*, ed. Alan Paeth , pp. 55–64, Academic Press,Cambridge, MA (1995).