Efficiently Building a Matrix to Rotate One Vector to Another

Tomas Möller and John F. Hughes

June 1999, revision December 1999

Abstract

We describe an efficient (no square-roots or trigonometric functions) routine that constructs the 3×3 matrix that rotates a unit vector $\mathbf{f}$ into another unit vector $\mathbf{t}$, rotating about the axis $\mathbf{f} \times \mathbf{t}$. An implementation in C is provided.

1 Introduction

Often in graphics, we have a unit vector, $\mathbf{f}$, that we wish to rotate to another unit vector, $\mathbf{t}$; in other words, we seek a rotation matrix $\mathbf{R}(\mathbf{f}, \mathbf{t})$ such that $\mathbf{R}(\mathbf{f}, \mathbf{t})\mathbf{f} = \mathbf{t}$. This paper describes a method to compute the matrix $\mathbf{R}(\mathbf{f}, \mathbf{t})$ from the coordinates of $\mathbf{f}$ and $\mathbf{t}$, without square-root or trigonometric functions, and compares it to other methods, one based on direct quaternion computation, another based on change of bases [1], and another described by Goldman [3]. Fast and robust C code can be found on the accompanying web site. In the event that unit vectors are not available, normalization requirements are comparable for all methods tested.

2 Derivation

Rotation from $\mathbf{f}$ to $\mathbf{t}$ can be generated by letting $\mathbf{v} = \mathbf{f} \times \mathbf{t}$, letting $\mathbf{u} = \mathbf{v}/||\mathbf{v}||$, and then rotating about the unit vector $\mathbf{u}$ by $\theta = \arccos(\mathbf{f} \cdot \mathbf{t})$. A formula for the matrix that rotates about $\mathbf{u}$ by θ is given in Foley et al. [2], namely

$$\begin{pmatrix} u_x^2 + (1 - u_x^2)\cos\theta & u_x u_y(1 - \cos\theta) - y_z \sin\theta & u_x u_z + u_y \sin\theta \\ u_x u_y(1 - \cos\theta) + u_z \sin\theta & u_y^2 + (1 - u_y^2)\cos\theta & u_y u_z(1 - \cos\theta) - u_x \sin\theta \\ u_x u_z(1 - \cos\theta) - u_y \sin\theta & u_y u_z(1 - \cos\theta) + u_x \sin\theta & u_z^2 + (1 - u_x^2)\cos\theta \end{pmatrix}$$

It involves $\cos(\theta)$, which is just $\mathbf{f} \cdot \mathbf{t}$, and $\sin(\theta)$, which is $||\mathbf{f} \times \mathbf{t}||$, i.e., $||\mathbf{v}||$. If we let

$$c = \mathbf{f} \cdot \mathbf{t} \tag{1}$$

and

$$h = \frac{1-c}{1-c^2} = \frac{1-c}{\mathbf{v}\cdot\mathbf{v}} \tag{2}$$

then, after considerable algebra, one can simplify the matrix to

$$\mathbf{R}(\mathbf{f},\mathbf{t}) = \begin{pmatrix} c + hv_x^2 & hv_xv_y - v_z & hv_xv_z + v_y \\ hv_xv_y + v_z & c + hv_y^2 & hv_yv_z - v_x \\ hv_xv_z - v_y & hv_yv_z + v_x & c + hv_z^2 \end{pmatrix} \tag{3}$$

Note that this formula for $\mathbf{R}(\mathbf{f},\mathbf{t})$ has no square-roots or trigonometric functions.

When $\mathbf{f}$ and $\mathbf{t}$ are nearly parallel (i.e., $|\mathbf{f}\cdot\mathbf{t}| > 0.99$), the computation of the plane that they define (and the normal to that plane, which will be the axis of rotation) is numerically unstable; this is reflected in our formula by the denominator of h becoming close to zero.

In this case, we observe that a product of two reflections (angle-preserving transformations of determinant -1) is always a rotation, and that reflection matrices are easy to construct: for any vector $\mathbf{u}$, the Householder matrix [4]

$$\mathbf{H}(\mathbf{u}) = \mathbf{I} - \frac{2}{\mathbf{u}\cdot\mathbf{u}}\mathbf{u}\mathbf{u}^t$$

reflects the vector $\mathbf{u}$ to $-\mathbf{u}$, and leaves fixed all vectors orthogonal to $\mathbf{u}$. In particular, if $\mathbf{a}$ and $\mathbf{b}$ are unit vectors, then $\mathbf{H}(\mathbf{b}-\mathbf{a})$ exchanges $\mathbf{a}$ and $\mathbf{b}$, leaving $\mathbf{a}+\mathbf{b}$ fixed.

With this in mind, we choose a unit vector $\mathbf{x}$ and build two reflection matrices: one that swaps $\mathbf{f}$ and $\mathbf{x}$, and the other that swaps $\mathbf{t}$ and $\mathbf{x}$. The product of these is a rotation that takes $\mathbf{f}$ to $\mathbf{t}$.

To choose $\mathbf{x}$, we determine which coordinate axis (x, y, or z) is most nearly orthogonal to $\mathbf{f}$ (the one for which the corresponding coordinate of f is smallest in absolute value) and let $\mathbf{x}$ be a unit vector along that axis.

We now build $\mathbf{A} = \mathbf{H}(\mathbf{x}-\mathbf{f})$, and $\mathbf{B} = \mathbf{H}(\mathbf{x}-\mathbf{t})$, and the rotation we want is $\mathbf{R} = \mathbf{B}\mathbf{A}$. The entries of $\mathbf{R}$ are

$$r_{ij} = \delta_{ij} - \frac{2}{\mathbf{u}\cdot\mathbf{u}}u_iu_j - \frac{2}{\mathbf{v}\cdot\mathbf{v}}v_iv_j + \frac{4\mathbf{u}\cdot\mathbf{v}}{(\mathbf{u}\cdot\mathbf{u})(\mathbf{v}\cdot\mathbf{v})}v_iu_j$$

where $\mathbf{u} = \mathbf{x}-\mathbf{f}$, $\mathbf{v} = \mathbf{x}-\mathbf{t}$, and $\delta_{ij} = 1$ when $i = j$ and $\delta_{ij} = 0$ when $i \neq j$.

3 Performance

The new routine was tested for performance against all (by the authors) previously known methods for rotating a unit vector into another unit vector. A naive way to rotate $\mathbf{f}$ into $\mathbf{t}$ is to use quaternions to build the rotation directly; letting $\mathbf{u} = \mathbf{v}/||\mathbf{v}||$, where $\mathbf{v} = \mathbf{f}\times\mathbf{t}$, and letting $\phi = (1/2)\arccos(\mathbf{f}\cdot\mathbf{t})$, we define $\mathbf{q} = (\sin(\phi)\mathbf{u}; \cos\phi)$ and then convert the quaternion $\mathbf{q}$ into a rotation via the method described in by Shoemake [5]. This rotation will take $\mathbf{f}$ to $\mathbf{t}$, and we refer to this computation as Naive. The second is called Cunningham and is simply a change of bases [1]. A routine for rotating around an arbitrary axis has been presented by Goldman [3], and in our third method we simplified his matrix for our purposes. The third method is denoted Goldman. All three of these require that some vector be normalized; the quaternion method requires normalization of $\mathbf{v}$; the Cunningham method requires that one input be normalized, and then requires normalization of the cross-product. Goldman requires the normalized axis of rotation. Thus the requirement of unit-vector input in our algorithm is not exceptional.

For the statistics below, we used $1,000$ pairs of random normalized vectors $\mathbf{f}$ and $\mathbf{t}$, each pair was feed to the matrix routines $10,000$ times in order to produce accurate timings. Our timings were done on a Pentium II 400 MHz with compiler optimizations for speed on.

Routine:	Naive	Cunningham	Goldman	New Routine
Time (s):	18.6	13.2	6.5	4.1

The fastest of previous known methods (Goldman) still takes about 50% more time than our new routine, and the naive implementation takes almost 350% more time. Similar performance can be expected on most other architectures, since square roots and trigonometric functions are expensive to use.

4 Acknowledgement

Thanks to Eric Haines for encouraging us to write this.

References

[1] Cunningham, Steve, “3D Viewing and Rotation using Orthonormal Bases”, in Andrew S. Glassner (editor), *Graphics Gems*, Academic Press, Inc., Boston, pp. 516–521, 1990.

[2] Foley, J.D., A. van Dam, S.K. Feiner, and J.H. Hughes, *Computer Graphics – Principles and Practice*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1990.

[3] Goldman, Ronald, “Matrices and Transformations”, in Andrew S. Glassner (editor), *Graphics Gems*, Academic Press, Inc., pp. 472–475, 1990.

[4] Golub, Gene, and Charles Van Loan, *Matrix Computations*, Third Edition, Johns Hopkins University Press, 1996.

[5] Shoemake, Ken, “Animating Rotation with Quaternion Curves”, *Computer Graphics (SIGGRAPH ’85 Proceedings)*, vol. 19, no. 3, pp. 245–254, July 1985.