TRANSFORMATIONS OF SURFACE NORMAL VECTORS
with applications to three dimensional computer graphics

Ken Turkowski
Advanced Technology Group
Graphics Software
Apple Computer, Inc.

Abstract: Given an affine 4x4 modeling transformation matrix, we derive the matrix that represents the transformation of a surface's normal vectors. This is similar to the modeling matrix *only* when any scaling is *isotropic*. We further derive results for transformations of light direction vectors and shading computations in clipping space.

6 July 1990

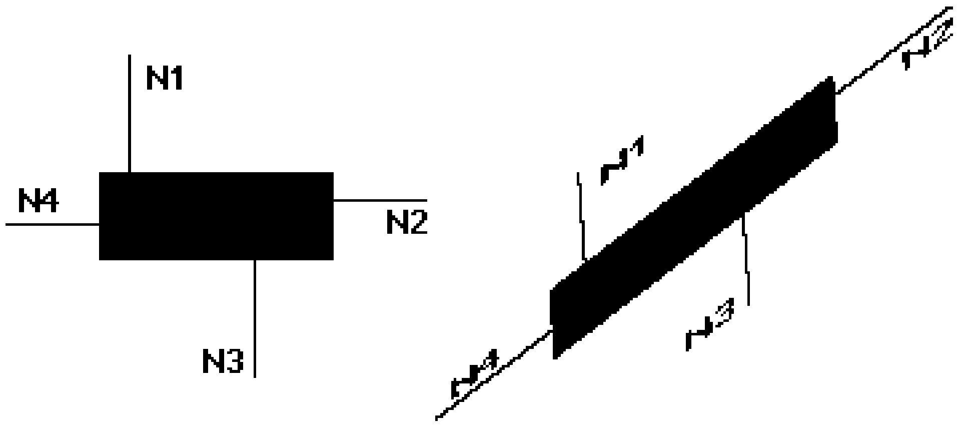
Apple Technical Report No. 22

Transformations of Surface Normal Vectors

Ken Turkowski
6 July 1990

Why is a Normal Vector not just a Difference Between Two Points?

In(fig. 1), below, on the left, we illustrate a rectangle, and its normals $\mathbf{N}_i$ that have been modeled as straight line segments. On the right is an affine transformation of the rectangle and its normals, where the endpoints of the straight line segments representing the normals have been transformed in the same way as other points that make up the rectangle. Note that these so-called "normal vectors" are no longer perpendicular to the surface.



(fig. 1)

Is this the type of behavior that we expect from normal vectors? I think not. This leads us to believe that normals behave differently under transformations than the surfaces that they correspond to. The transformations are related, though, by the requirement that the surfaces and their normals remain orthogonal. We use this property to derive the normal transformation matrix from the the point transformation matrix.

Transformation of Normal Vectors under Affine Modeling Transformations

Given a planar surface, a tangent vector can be expressed as the difference between two points on the surface:

$$\mathbf{t}_1 = \mathbf{p}_1 - \mathbf{p}_0 \tag{eq. 1}$$

The normal to the surface can be calculated from:

$$\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2 \tag{eq. 2}$$

for and two non-colinear tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$.

By construction, the normal is orthogonal to the tangent vectors, so that:

$$\mathbf{t} \quad \mathbf{n} = \mathbf{t}\mathbf{n}^t = 0 \qquad \text{(eq. 3)}$$

When a surface is transformed by the modeling matrix M:

$$\mathbf{p}' = \mathbf{p}\mathbf{M} \qquad \text{(eq. 4)}$$

where

$$\mathbf{M} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ m_{30} & m_{31} & m_{32} & 1 \end{bmatrix} \qquad \text{(eq. 5)}$$

and $\mathbf{p}$ are a set of points that define the surface, then the tangent vectors are transformed by the matrix $\mathbf{M}_1$:

$$\mathbf{t}' = \mathbf{t}\mathbf{M}_1 \qquad \text{(eq. 6)}$$

where

$$\mathbf{M}_1 = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \qquad \text{(eq. 7)}$$

is the submatrix of $\mathbf{M}$ that excludes translations, as can be verified by applying (eq. 1), (eq. 5) and (eq. 4).

In order to have the relation between tangent and normal vectors hold, we apply (eq. 3):

$$\mathbf{t} \quad \mathbf{n} = \mathbf{t}\mathbf{M}_1\mathbf{M}_1^{-1}\mathbf{n}^t = \left(\mathbf{t}\mathbf{M}_1\right) \quad \left(\mathbf{n}\mathbf{M}_1^{-1t}\right) = \mathbf{t}' \quad \mathbf{n}' = 0 \qquad \text{(eq. 8)}$$

or that

$$\mathbf{n}' = \mathbf{n}\mathbf{N} = \mathbf{n}\mathbf{M}_1^{-1t} \qquad \text{(eq. 9)}$$

which implies that the normal vector is transformed by the *transpose of the inverse* of the tangent's transformation matrix. This is a well-known theorem of tensor algebra, where $\mathbf{t}$ is called a contravariant tensor of rank 1, and $\mathbf{n}$ is a covariant tensor of rank 1.

Simplifying the Computations

Unless the matrix $\mathbf{M}_1$ is unitary (i.e. length and angle preserving), the transformation of a unit normal is generally not itself a unit normal. In this light, we do not need to compute the full inverse, when the adjoint will do, since a post-normalization step is necessary anyway. In terms of the tangent transformation matrix $\mathbf{M}_1$ or the modeling matrix $\mathbf{M}$, the adjoint is:

$$\operatorname{adj}\left(\mathbf{M}_1\right) = \begin{bmatrix} m_{22}m_{11} - m_{12}m_{21} & m_{02}m_{21} - m_{22}m_{01} & m_{12}m_{01} - m_{02}m_{11} \\ m_{12}m_{20} - m_{10}m_{22} & m_{22}m_{00} - m_{02}m_{20} & m_{10}m_{02} - m_{12}m_{00} \\ m_{10}m_{21} - m_{20}m_{11} & m_{20}m_{01} - m_{00}m_{21} & m_{00}m_{11} - m_{10}m_{01} \end{bmatrix} \qquad \text{(eq. 10)}$$

so a transformation $\mathbf{N}$ that preserves the relationship between the normal and the surface tangent under the modeling transformation $\mathbf{M}$ is:

$$\mathbf{N} = \begin{bmatrix} m_{22}m_{11} - m_{12}m_{21} & m_{12}m_{20} - m_{10}m_{22} & m_{10}m_{21} - m_{20}m_{11} \\ m_{02}m_{21} - m_{22}m_{01} & m_{22}m_{00} - m_{02}m_{20} & m_{20}m_{01} - m_{00}m_{21} \\ m_{12}m_{01} - m_{02}m_{11} & m_{10}m_{02} - m_{12}m_{00} & m_{00}m_{11} - m_{10}m_{01} \end{bmatrix} \qquad \text{(eq. 11)}$$

Non-Planar Surfaces

We can generalize these results to non-planar surfaces, as long as the geometry of the surface is described by a set of points. In particular, the definition of the surface normal can be replaced by

$$\mathbf{n} = \quad s\,, \qquad \text{(eq. 12)}$$

the gradient of a surface s at a given point. This makes (eq. 9) and (eq. 11) useful for Bézier surface patches.

Unitary Transformations

In the case where $\mathbf{M}_1$ is a unitary matrix (i.e. the modeling matrix $\mathbf{M}$ is composed of only rotations and translations), its inverse is simply the transpose; since the transpose of the transpose of a matrix is the original matrix itself, normals are transformed by the same matrix as the tangents, and hence:

$$\mathbf{N} = \mathbf{M}_1 \qquad \text{(eq. 13)}$$

Note that this does *not* hold true when the matrix includes skewing or other anisotropic scaling.

Inverse Transformations of the Lighting Vector

When computing the shading in a simple three dimensional graphics system, it is sometimes advantageous to transform a *directional* light vector from the world space into the modeling space in order to perform the dot product with the surface normals of an object in its canonical modeling space.

In world space, the shading computation takes the form:

$$d = \mathbf{n}_w \quad \mathbf{i}_w = \mathbf{n}_w \mathbf{i}_w^{\,t} \qquad \text{(eq. 14)}$$

Applying the normal transformation rule (eq. 9), we get the equivalent relation in modeling space:

$$d = \left(\mathbf{n}_m \mathbf{M}_1^{-1t}\right)\mathbf{i}_w^t = \mathbf{n}_m\left(\mathbf{i}_w \mathbf{M}_1^{-1}\right)^t = \mathbf{n}_m \quad \left(\mathbf{i}_w \mathbf{M}_1^{-1}\right) \qquad \text{(eq. 15)}$$

where the subscript w indicates world space, and the subscript m indicated modeling space. Then the light vector in world space is transformed by the inverse 3x3 modeling submatrix to bring it into modeling space.

However, this analysis assumes that the modeling and lighting vectors are already normalized appropriately for world space. In the general case, we have the equation:

$$d = \frac{\mathbf{n}_w \mathbf{i}_w^t}{\left\|\mathbf{n}_w\right\|\left\|\mathbf{i}_w\right\|} = \frac{\mathbf{n}_m \mathbf{M}_1^{-1t}\mathbf{i}_w^t}{\left\|\mathbf{n}_w\right\|\left\|\mathbf{i}_w\right\|} = \frac{\mathbf{n}_m \quad \left(\mathbf{i}_w \mathbf{M}_1^{-1}\right)^t}{\left\|\mathbf{n}_w\right\|\left\|\mathbf{i}_w\right\|} \qquad \text{(eq. 16)}$$

where the norm in the denominator is given by:

$$\begin{aligned} \mathbf{N} &= \left\|\mathbf{n}_w\right\|\left\|\mathbf{i}_w\right\| = \left\|\mathbf{n}_m \mathbf{M}_1^{-1t}\right\|\left\|\mathbf{i}_w\right\| \\ &= \left(\mathbf{n}_m \mathbf{M}_1^{-1t} \mathbf{M}_1^{-1} \mathbf{n}_m^t\right)^{\frac{1}{2}} \left(\mathbf{i}_w \mathbf{i}_w^t\right)^{\frac{1}{2}} \\ &= \left(\mathbf{n}_m \mathbf{G} \mathbf{n}_m^t\right)^{\frac{1}{2}} \left(\mathbf{i}_w \mathbf{i}_w^t\right)^{\frac{1}{2}} \end{aligned} \qquad \text{(eq. 17)}$$

The matrix

$$\mathbf{G} = \mathbf{M}_1^{-1t} \mathbf{M}_1^{-1} \qquad \text{(eq. 18)}$$

is called the *first fundamental matrix* [Faux 80], and represents the fact that the modeling space is in general *non-Euclidean*, i.e. the distance between points as measured in the modeling space is *not* simply the square root of the sum of the squares of the componentwise differences between the points. Such a space is called a *Riemannian* space, and comes about because the modeling submatrix $\mathbf{M}_1$ is not in general isotropic (i.e. scaling is not equal in all dimensions). In other words, if the metric in world space is to be regarded as Euclidean (as it should for shading calculations), then the metric in modeling space will not, in general, be Euclidean.

If the light vector is normalized in world space, then the shading equation then becomes:

$$d = \frac{\mathbf{n}_m \quad \left(\mathbf{i}_w \mathbf{M}_1^{-1}\right)^t}{\left(\mathbf{n}_m \mathbf{G} \mathbf{n}_m^t\right)^{\frac{1}{2}}} \qquad \text{(eq. 19)}$$

These calculations in Riemannian space are more complicated than the corresponding calculations in Euclidean space. Since the transformation from modeling to world space (eq. 9) occurs at each normal, whereas the Riemannian normalization (eq. 19) occurs at each pixel, it is computationally less expensive to do the shading in world space with the Euclidean norm:

$$d = \frac{\mathbf{n}_w \mathbf{i}_w^{\,t}}{\left(\mathbf{n}_w \mathbf{n}_w^{\,t}\right)^{\frac{1}{2}}}$$ (eq. 20)

Non-affine Transformations

The results of (eq. 9) are true regardless of the form of the modeling matrix $\mathbf{M}$. This allows us to apply it to non-affine (projective) transformations as well, i.e. modeling matrices in which the rightmost column has non-zero entries. Of course, (eq. 10) and (eq. 11) are not quite as simple, and the resultant normals are 4-vectors. Unfortunately, though, orthogonality is not invariant under projection.

Shading Computations in Clipping Space

We can compute diffuse shading in homogeneous coordinates by maintaining the dot product:

$$d = \frac{\mathbf{nN}}{\|\mathbf{nN}\|} \quad \mathbf{i} = \frac{\mathbf{nN}}{\|\mathbf{nN}\|}\mathbf{i}^t = \frac{\mathbf{nN}}{\|\mathbf{nN}\|}(\mathbf{VC})(\mathbf{VC})^{-1}\mathbf{i}^t = \underbrace{\frac{[\mathbf{n}(\mathbf{NVC})]}{\|\mathbf{nN}\|}}_{\substack{\textit{normalized}\\ \textit{surface normal}}} \underbrace{\left[\mathbf{i}(\mathbf{VC})^{-1}\right]^{t}}_{\substack{\textit{light vector in}\\ \textit{clipping space}}}$$ (eq. 21)

where the vectors $\mathbf{n}$ and $\mathbf{i}$ have their fourth component set to zero since they have only a direction, not a position. The matrices $\mathbf{V}$ and $\mathbf{C}$ transform from modeling to viewing, and viewing to clipping spaces, respectively. The normalization of the interpolated vector is a problem, though, since it is computed in world space, not clipping space, so this is not a practical technique for smooth shading by interpolating normal vectors in 4-space.

Back-Face Culling

It can significantly enhance the throughput of a renderer by eliminating back-facing polygons from consideration during hidden surface removal and scan conversion. *Back-facing* means that the polygon is not visible from the eye under the desired perspective viewing transformation.

Let us look at the situation in several spaces: modeling, world, viewing, and clipping.

Back-face Culling in Modeling Space

A point on the plane of the polygon can be characterized by the equation:

$$\mathbf{p}_m \mathbf{n}_m^{\,t} = 0$$ (eq. 22)

where

$$\mathbf{p}_m = [x \;\; y \;\; z \;\; 1], \qquad \mathbf{n}_m = [n_x \; n_y \;\; n_z \;\; d] \tag{eq. 23}$$

are given in modeling space.
The polygon is visible if the eye is on the proper side of the plane:

$$\left(\mathbf{e}_w \mathbf{M}^{-1}\right) \mathbf{n}_m^{\;t} < 0 \tag{eq. 24}$$

where $\mathbf{e}_w$ is given in world space. Of course, it isn't necessary to invert the entire 4x4 $\mathbf{M}$ matrix, since it can be computed from the inverse of a 3x3 instead:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{M}_1^{\;-1} & \mathbf{0} \\ -\mathbf{m}_0 \mathbf{M}_1^{\;-1} & 1 \end{bmatrix} \tag{eq. 25}$$

where the $\mathbf{M}$ matrix is partitioned as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{m}_0 & 1 \end{bmatrix}$$

Back-face Culling in World Space

To perform the visibility test in world space, we invoke associativity of matrix multiplication on (eq. 24):

$$\mathbf{e}_w \left(\mathbf{M}^{-1} \mathbf{n}_m^{\;t}\right) = \mathbf{e}_w \left(\mathbf{n}_m \mathbf{M}^{-1t}\right)^t < 0 \tag{eq. 27}$$

Back-face Culling in Viewing Space

In viewing space, the eye is at the origin

$$\mathbf{e}_v = [0 \; 0 \; 0 \; 1] \tag{eq. 28}$$

and we get:

$$\mathbf{e}_v \left[\mathbf{n}_m (\mathbf{MV})^{-1\,t}\right]^t < 0 \tag{eq. 29}$$

or

$$\left[\mathbf{n}_m (\mathbf{MV})^{-1\,t}\right] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} < 0 \tag{eq. 30}$$

We define a partition for the viewing matrix similar to that of the modeling matrix:

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{0} \\ \mathbf{v}_0 & 1 \end{bmatrix} \qquad \text{(eq. 31)}$$

The concatenation with the modeling matrix $\mathbf{M}$ is then:

$$\mathbf{MV} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{m}_0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 & \mathbf{0} \\ \mathbf{v}_0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1\mathbf{V}_1 & \mathbf{0} \\ \mathbf{m}_0\mathbf{V}_1 + \mathbf{v}_0 & 1 \end{bmatrix} \qquad \text{(eq. 32)}$$

Back-face Culling in Clipping Space

The concatenation with the clipping matrix produces:

$$\mathbf{e}_c \left[\mathbf{n}_m (\mathbf{MVC})^{-1t} \right]^t < 0 \qquad \text{(eq. 33)}$$

Conclusions

We have derived the normal transformation (eq. 9) corresponding to an affine point transformation of a surface. A simple way of calculating this matrix is given in(eq. 11). The computations in a 3D graphics system with infinite light sources can be simplified with (eq. 15).

Back-face culling is done by taking the dot product of the plane equation with the eye point. It seems as if the computations are simplest in modeling space.

References

Faux 80 Faux, I.D., and Pratt, M.J. *Computational Geometry for Design and Manufacture.* Chinchester, West Sussex, England: Ellis Horwood Limited, 1980.