# X.7

# PROPERTIES OF SURFACE-NORMAL TRANSFORMATIONS

Ken Turkowski
*Apple Computer*
*Cupertino,California*

## Why Is a Normal Vector Not Just a Difference between Two Points?

In Fig. 1a, we illustrate a rectangle and its normals, $\mathbf{N}_i$, that have been modeled as straight line segments. Figure 1b shows 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. This occurs because we applied an *anisotropic* transformation, that is, one that is not equal in all directions.

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 to which they correspond. 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 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

$$\mathbf{T}_1 = P_1 - P_0. \tag{1}$$

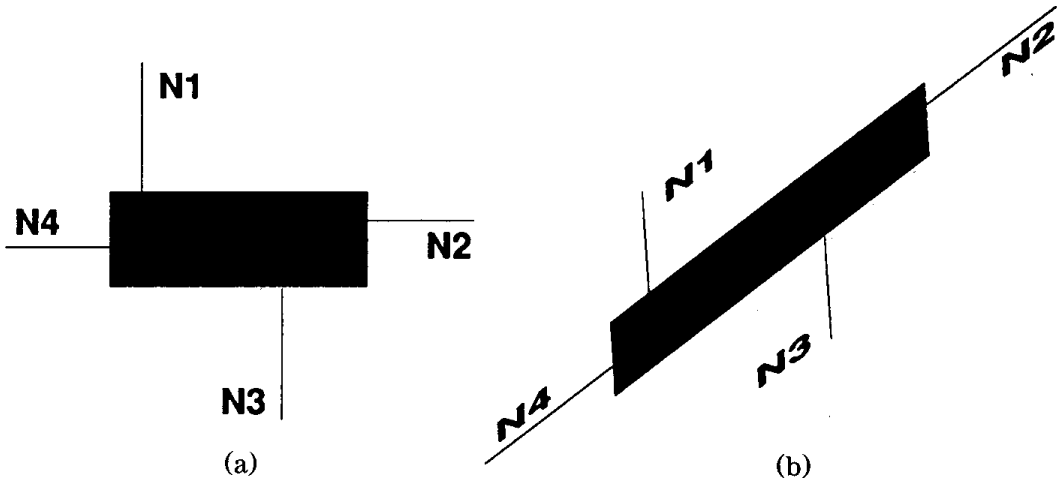


Figure 1.

The normal to the surface can be calculated from

$$\mathbf{N} = \mathbf{T}_1 \times \mathbf{T}_2, \tag{2}$$

for any two noncolinear tangent vectors $\mathbf{T}_1$ and $\mathbf{T}_2$.

If a surface is not planar, the tangents and normal can still be defined systematically. For a parametric surface (that is, $x$, $y$, and $z$ as a function of $u$ and $v$), we have

$$\mathbf{T}_u = \frac{\partial(x,y,z)}{\partial u}, \quad \mathbf{T}_v = \frac{\partial(x,y,z)}{\partial v}, \quad \mathbf{N} = \mathbf{T}_u \times \mathbf{T}_v, \tag{3}$$

while for an implicit surface described by $f(x, y, z) = 0$, we have

$$\mathbf{N} = \nabla f\ (x,\ y,\ z), \tag{4}$$

with a tangent represented by any nonzero vector such that

$$\mathbf{T} \cdot \mathbf{N} = 0. \tag{5}$$

Regardless of the type of surface, we know that Ihe normal is orthogonal to the tangent by construction, so that

$$\mathbf{T} \cdot \mathbf{N} = \mathbf{T}\mathbf{N}^{\mathrm{T}} = 0. \tag{6}$$

When a surface is transformed by the affine modeling matrix $\mathbf{M}$,

$$\tilde{\mathbf{p}}' = \tilde{\mathbf{p}}\mathbf{M}, \tag{7}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{00} & \mathbf{M}_{01} & \mathbf{M}_{02} & 0 \\ \mathbf{M}_{10} & \mathbf{M}_{11} & \mathbf{M}_{12} & 0 \\ \mathbf{M}_{20} & \mathbf{M}_{21} & \mathbf{M}_{22} & 0 \\ \mathbf{M}_{30} & \mathbf{M}_{31} & \mathbf{M}_{32} & 1 \end{bmatrix}, \tag{8}$$

and $\tilde{\mathbf{p}}$ is a representative of a set of points (in homogeneous coordinates) that define the surface, then the tangent vector is transformed by the matrix $\mathbf{M}_T$:

$$\mathbf{T}' = \mathbf{T}\mathbf{M}_T, \tag{9}$$

where

$$\mathbf{M}_T = \begin{bmatrix} \mathbf{M}_{00} & \mathbf{M}_{01} & \mathbf{M}_{02} \\ \mathbf{M}_{10} & \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{20} & \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \tag{10}$$

is the submatrix of $\mathbf{M}$ that excludes translations, as can be verified by applying Eq. 1, Eq. 8, and Eq. 7.

In order to have the relation between tangent and normal vectors hold, we apply Eq. 6:

$$\mathbf{T} \cdot \mathbf{N} = \mathbf{T}\mathbf{M}_T\mathbf{M}_T^{-1}\mathbf{N}^T = (\mathbf{T}\mathbf{M}_T) \cdot (\mathbf{N}\mathbf{M}_T^{-1T}) = \mathbf{T}' \cdot \mathbf{N}' = 0, \tag{11}$$

or

$$\mathbf{N}' = \mathbf{N}\mathbf{M}_N = \mathbf{N}\mathbf{M}_T^{-1T}, \tag{12}$$

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 covariant tensor of rank 1, and $\mathbf{N}$ is a contravariant tensor of rank 1.

Certain types of modeling transformations give rise to simple representations of the normal transformation matrix. Translations do not affect the normal or tangent transformations, so these two are the identity transformation. Below, we give examples of other special transformations.

## Isotropic Transformations

Often, a modeling transformation consists of uniform (isotropic) scaling. This type of transformation has the following representative tangent matrix

$$\mathbf{M}_{\mathrm{T}} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix}, \tag{13}$$

where $s$ is a nonzero scalar. The inverse transpose of this matrix (suitable for transforming normals) is

$$\mathbf{M}_{\mathrm{N}} = \mathbf{M}_{\mathrm{T}}^{-1\mathrm{T}} = \begin{bmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & \frac{1}{s} \end{bmatrix} = \frac{1}{s^2}\mathbf{M}_{\mathrm{T}}, \tag{14}$$

a scalar multiple of the tangent transformation matrix.

## Orthogonal Transformations

In the case where $\mathbf{M}_{\mathrm{T}}$ is an orthogonal matrix (that is, the modeling matrix $\mathbf{M}$ is composed only of 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,

$$\mathbf{M}_N = \mathbf{M}_T. \tag{15}$$

## Composition of Transformations

If a modeling transformation is composed of a concatenation of simpler transformations, the composite normal transformation is just the concatenation of the corresponding simpler normal transformations. That is, if

$$\mathbf{M}_T = \mathbf{M}_{T1}\mathbf{M}_{T2}. \; . \; .\mathbf{M}_{Tk}, \tag{16}$$

then

$$\mathbf{M}_N = \mathbf{M}_{N1}\mathbf{M}_{N2}. \; . \; .\mathbf{M}_{Nk}. \tag{17}$$

This can be verified by substitution of Eqs. 16, 17, and 12 into Eq. 11.

## Transformations of Metric Properties

In Euclidean space, the $L_2$ norm of a vector is given by

$$|\mathbf{V}|_2 = \sqrt{\mathbf{V}\cdot\mathbf{V}} = \sqrt{\mathbf{V}\mathbf{V}^T}. \tag{18}$$

Under the transformation

$$\mathbf{V} = \mathbf{W}\mathbf{A}, \tag{19}$$

we have

$$|\mathbf{W}|_2 = \sqrt{(\mathbf{W}\mathbf{A})(\mathbf{W}\mathbf{A})^T} = \sqrt{\mathbf{W}\mathbf{A}\mathbf{A}^T\mathbf{W}^T} = \sqrt{\mathbf{W}\mathbf{G}\mathbf{W}^T}, \tag{20}$$

where the matrix

$$\mathbf{G} = \mathbf{A}\mathbf{A}^T \tag{21}$$

is known as the *first fundamental matrix* (Faux and Pratt, 1979) or *metric tensor,* and represents the fact that the space is in general non-Euclidean, that is, the length of a vector in this space is not simply the square root of the sum of the squares of the components of the vector. Such a space is called a Riemannian space, and comes about because the modeling matrix **A** is not in general isotropic (scaling is not equal in all dimensions).

In a Euclidean space, **G** = **1** (the identity matrix), and the norm becomes the familiar Euclidean one.

## Applications to Computer Graphics: Back-Face Culling

Performance in a computer graphics display system can be improved if polygons facing away from the view point are removed before scan-conversion. This is facilitated with an operation involving the plane equation of the polygon in question:

$$N_x x + N_y y + N_z z - d = \mathbf{N} \cdot P - d = 0, \tag{22}$$

where

$$\mathbf{N} = [N_x \quad N_y \quad N_z] \tag{23}$$

is the outward-facing normal to the polygon,

$$P = [x \quad y \quad z] \tag{24}$$

is any point on the plane of the polygon, and $d$ is the closest distance of the plane to the origin, measured in the direction of the normal—positive if the normal points toward the origin from the plane, and negative if it points away.

Back-facing polygons can be identified by

$$\mathbf{N} \cdot E - d < 0, \tag{25}$$

where $E$ is the view point. This can also be represented as

$$\tilde{\mathbf{N}} \cdot \tilde{\mathbf{E}} < 0, \tag{26}$$

where

$$\tilde{\mathbf{N}} = [\mathrm{N}_x \quad \mathrm{N}_y \quad \mathrm{N}_z \quad -d] \text{ and } \tilde{\mathbf{E}} = [E_x \quad E_y \quad E_z \quad 1] \tag{27}$$

represent the plane and view point in homogeneous coordinates.

Suppose now that the polygon is defined in a coordinate system different than the "world," coordinate system. This is commonly called modeling space, and usually has an affine transformation $\mathbf{M}$ relating it to world space as in Eq. 7 by

$$\tilde{\mathbf{P}}_w = \tilde{\mathbf{P}}_m \mathbf{M}. \tag{28}$$

If the plane equation is given in modeling space, we can transform the view point into modeling space and do the back-face culling there:

$$\tilde{\mathbf{N}} \cdot \tilde{\mathbf{E}}\mathbf{M}^{-1} < 0. \tag{29}$$

We can alternatively do the culling in world space by noting that

$$\tilde{\mathbf{N}} \cdot \tilde{\mathbf{E}}\mathbf{M}^{-1} = \tilde{\mathbf{N}}(\tilde{\mathbf{E}}\mathbf{M}^{-1})^{\mathrm{T}} = \tilde{\mathbf{N}}\mathbf{M}^{-1\mathrm{T}}\tilde{\mathbf{E}}^{\mathrm{T}} = (\tilde{\mathbf{N}}\mathbf{M}^{-1\mathrm{T}}) \cdot \tilde{\mathbf{E}}, \tag{30}$$

so we can transform the plane from modeling to world space with the inverse transpose of the full modeling matrix.

Either way, we need to invert the modeling matrix, whether we do the back-face culling in modeling or world space.

## Applications to Computer Graphics: Shading

When computing shading in a simple three-dimensional graphics system, it is sometimes advantageous to transform a *directional* light vector $\mathbf{I}$ from the world space into the modeling space. Central to the Lambert

shading computation, for example, is the calculation of the dot product:

$$d = \frac{\mathbf{N}_w \mathbf{I}_w^{\mathrm{T}}}{|\mathbf{N}_w|\,|\mathbf{I}_w|}. \tag{31}$$

In a typical scan-conversion system, the light vector is normalized once when the light is placed in the scene, and doesn't need to be normalized again. The surface normal, however, is usually interpolated from the polygon's vertex normals, so it needs to be renormalized at every pixel in the polygon. (This is true if the normals are linearly interpolated in a Cartesian space. If the normals are interpolated on the surface of a sphere, using spherical coordinates or quaternions, no renormalization is required.)

Applying the normal transformation rule (Eq. 12), we get the equivalent relation in modeling space:

$$d = \frac{\mathbf{N}_w \mathbf{I}_w^{\mathrm{T}}}{|\mathbf{N}_w|\,|\mathbf{I}_w|} = \frac{\mathbf{N}_m \mathbf{M}_{\mathrm{N}} \mathbf{I}_w^{\mathrm{T}}}{|\mathbf{N}_m \mathbf{M}_{\mathrm{N}}|\,|\mathbf{I}_w|} = \frac{\mathbf{N}_m \cdot \left(\mathbf{I}_w \mathbf{M}_{\mathrm{N}}^{\mathrm{T}}\right)}{|\mathbf{N}_m \mathbf{M}_{\mathrm{N}}|\,|\mathbf{I}_w|}, \tag{32}$$

which implies that we need to transform only one vector (the light vector) instead of all of the polygon's vertex normals. But wait! There's a pesky transformation in the denonominator as well. Looking back to Eq. 20, we find that

$$|\mathbf{N}_m \mathbf{M}_{\mathrm{N}}| = \sqrt{\mathbf{N}_m \mathbf{M}_{\mathrm{N}} \mathbf{M}_{\mathrm{N}}^{\mathrm{T}} \mathbf{N}_m^{\mathrm{T}}} = \sqrt{\mathbf{N}_m \mathbf{G} \mathbf{N}_m^{\mathrm{T}}}, \tag{33}$$

where **G** is the first fundamental matrix of the Riemannian modeling space.

Normalizing the surface normal at each pixel costs:

| | |
|---|---|
| Modeling space: | one 3 × 3 matrix–vector multiplications and one 3-vector dot product |
| World space: | one 3-vector dot product |

Since one matrix-vector multiplication is composed of three dot products, it costs three extra dot products per pixel to do the shading calculations in modeling space. Thus, it seems like it is always advantageous to do the shading calculations in world space.

There is another reason not to compute shading in model space: the components of the vector as normalized by Eq. 33 do not necessarily have a magnitude less than or equal to one. This is an important consideration if the shading calculations are to be done in fixed-point because the magnitudes need to be bounded. For a given unit of energy (power consumption multiplied by time), fixed-point computations are always faster than floating-point, so high-performance graphics systems especially need to perform shading in world space.

If one can be assured, however, that all scaling will be isotropic, it is possible to save a few matrix-vector multiplications per polygon by subsuming the scale factor in Eq. 14 into the transformed light vector, thereby guaranteeing a Euclidean norm and bounded vector magnitudes. However, the increased flexibility afforded by nonrestricted modeling matrices far outweighs any performance improvement afforded by shading in modeling space.

## Conclusions

Normals are transformed by the inverse transpose of the modeling transformation.

Anisotropic transformations make metric computations (distance, length, norm) more complex: three extra 3-vector dot products.

It is slightly more advantageous to perform back-face culling in modeling space. The extra cost in world space is one multiplication and one $4 \times 4$ matrix transpose.

It costs three extra 3-vector dot products per pixel to perform shading computations in modeling rather than world space if anisotropic scaling is allowed. It costs $(n - 1)$ extra $3 \times 3$ matrix-vector multiplications per polygon in world space.

Unless the interface to the graphics library prevents anisotropic scaling (none currently do), the shading software should accommodate it. This is accomplished most efficiently and robustly in world space.