THE USE OF COORDINATE FRAMES IN COMPUTER GRAPHICS

Ken Turkowski
Media Technologies: Graphics Software
Advanced Technology Group
Apple Computer, Inc.
(Draft Friday, August 18, 1989)

Abstract: We demonstrate the use of coordinate frames to simplify transformations. We show how to specify coordinate transformations without trigonometry. The viewing transformation is simplified considerably by specifying it in terms of coordinate frames.

14 July 1989

Apple Technical Report No. KT-20

The Use of Coordinate Frames in Computer Graphics

Ken Turkowski

14 July 1989

Vectors and Points

We will have occasion to distinguish between 3D vectors and 3D points. A vector has a magnitude and a direction, but does not have a location, whereas a point only has a location, and no magnitude or direction. A vector cannot be moved, but it can be scaled and rotated. A point cannot be scaled or rotated, although it can be moved; a group of points can, however, be rotated or moved relative to each other. A linear transformation is appropriate for vectors, whereas an affine transformation is appropriate for points. There is a unique origin in a vector space, but an origin is arbitrary in an affine (point) space. These properties are summarized in the table below:

Attribute	**Vector**	**Point**
Represents	magnitude & direction	location
Origin	unique	arbitrary
Transformation	linear	affine
	scale	move
	rotate	

When points and vectors are represented by three components, they can only be distinguished by context. We may sometimes represent them by *four* coordinates (x, y, z, w), in what is called *homogeneous coordinates*. We interpret a 4-vector as a point in 3D by its projection onto the hyperplane w=1, viz [x/w, y/w, z/w]. The indeterminate form at w=0 is resolved by taking the limit as w approaches 0 from above: the point approaches infinity *in a particular direction*; hence the *vector* interpretation.

By convention, we will represent points as homogeneous 4-vectors with w=1.

For example, we represent the point 1 unit along the x-axis as:

$$\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}$$

whereas the x-axis (a vector) itself is represented as:

$$\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$$

In non-homogeneous coordinates, they are both represented as:

$$\begin{bmatrix}1 & 0 & 0\end{bmatrix}$$

so only context can distinguish them.

Coordinate Frames

We shall usually represent a coordinate frame for three-dimensional points with a 4x3 matrix:

$$\mathbf{F} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{O} \end{bmatrix} = \begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_y & Y_z \\ Z_x & Z_y & Z_z \\ O_x & O_y & O_z \end{bmatrix} \qquad \text{(eq. 1)}$$

This establishes a local reference frame within a more global frame by representing the local origin, *x*-, *y*-, and *z*-axes in terms of the global coordinates in the rows of the matrix. In particular,

$$\mathbf{O} = \begin{bmatrix} O_x & O_y & O_z \end{bmatrix} \qquad \text{(eq. 2)}$$

is a *point* that represents the origin of the local coordinate frame, represented in the coordinates of the global reference frame.

The local *x*-axis,

$$\mathbf{X} = \begin{bmatrix} X_x & X_y & X_z \end{bmatrix} \qquad \text{(eq. 3)}$$

is a *vector* (not a *point*), with both magnitude and direction.

Similar sorts of interpretations are appropriate for the **Y** and **Z** axes.

The matrix representation of this coordinate frame is more than just a convenient representation; it is in fact related to the more familiar 4x4 graphics transformation [Newman 79] matrix:

$$\mathbf{F}_4 = \left[\begin{array}{c|c} \mathbf{F} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{array}\right] = \left[\begin{array}{c|c} \begin{matrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline \mathbf{O} & 1 \end{array}\right] = \left[\begin{array}{ccc|c} X_x & X_y & X_z & 0 \\ Y_x & Y_y & Y_z & 0 \\ Z_x & Z_y & Z_z & 0 \\ \hline O_x & O_y & O_z & 1 \end{array}\right] \qquad \text{(eq. 4)}$$

where the 0's and 1 in the right column underscore the interpretation of **X**, **Y** and **Z** as vectors, and **O** as a point.

A 4-vector, with w=0, will not be affected by the translation portion (bottom row) of a 4x4 matrix transformation, whereas the 4-vector with $w \neq 0$ will.

We illustrate the effect of a coordinate transformation on the homogeneous representation of the *x*-axis (a vector) with a 4x4 matrix multiplication:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \left[\begin{array}{ccc|c} X_x & X_y & X_z & 0 \\ Y_x & Y_y & Y_z & 0 \\ Z_x & Z_y & Z_z & 0 \\ \hline O_x & O_y & O_z & 1 \end{array}\right] = \begin{bmatrix} X_x & X_y & X_z & 0 \end{bmatrix} \qquad \text{(eq. 5)}$$

With the fourth component zero, we see that a vector transforms into a vector. Not just any vector, the *x*-axis transforms into the vector $\begin{bmatrix} X_x & X_y & X_z & 0 \end{bmatrix}$, the top row of the transformation matrix. It is easy to see that the *y*- and *z*-axes transform into the second and third rows of the matrix, respectively.

The transformation of the origin (a point) yields:

$$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\left[\begin{array}{ccc|c} X_x & X_y & X_z & 0 \\ Y_x & Y_y & Y_z & 0 \\ Z_x & Z_y & Z_z & 0 \\ \hline O_x & O_y & O_z & 1 \end{array}\right] = \begin{bmatrix}O_x & O_y & O_z & 1\end{bmatrix} \qquad \text{(eq. 6)}$$

the bottom row of the matrix. This is consistent with the definition we gave it earlier.

Since the transformation of an arbitrary vector

$$\begin{bmatrix}v_x & v_y & v_z & 0\end{bmatrix}\left[\begin{array}{ccc|c} X_x & X_y & X_z & 0 \\ Y_x & Y_y & Y_z & 0 \\ Z_x & Z_y & Z_z & 0 \\ \hline O_x & O_y & O_z & 1 \end{array}\right] = \begin{bmatrix}v_x X_x + v_y Y_x + v_z Z_x & v_x X_y + v_y Y_y + v_z Z_y & v_x X_z + v_y Y_z + v_z Z_z & 0\end{bmatrix} \qquad \text{(eq. 7)}$$

doesn't depend at all on the last row (translation) we will generally omit it as well as the last column, to obtain the familiar 3x3 linear transformation matrix for 3-D vectors:

$$\begin{bmatrix}v_x & v_y & v_z\end{bmatrix}\begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_y & Y_z \\ Z_x & Z_y & Z_z \end{bmatrix} = \begin{bmatrix}v_x X_x + v_y Y_x + v_z Z_x & v_x X_y + v_y Y_y + v_z Z_y & v_x X_z + v_y Y_z + v_z Z_z\end{bmatrix} \qquad \text{(eq. 8)}$$

We will use this as the intrinsic coordinate frame for 3-D *vectors*, whereas the 4x3 matrix (eq. 1, page 3) will be used as the intrinsic coordinate frame for 3-D *points*, where we extend the operations of *linear* algebra to *affine* algebra as follows:

$$\begin{bmatrix}p_x & p_y & p_z\end{bmatrix}\begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_y & Y_z \\ Z_x & Z_y & Z_z \\ O_x & O_y & O_z \end{bmatrix} = \begin{bmatrix}p_x X_x + p_y Y_x + p_z Z_x + O_x & p_x X_y + p_y Y_y + p_z Z_y + O_y & p_x X_z + p_y Y_z + p_z Z_z + O_z\end{bmatrix} \qquad \text{(eq. 9)}$$

This is consistent with the treatment of 3-D points as homogeneous 4-vectors, as in (eq. 6).

Examples: Using Coordinate Frames to Solve Problems

(Example 1) Find the simple plane rotation that aligns the *x*-axis to the *y*-axis.

If the *x*-axis [1, 0] rotates into the *y*-axis [0, 1], the *y*-axis rotates into the negative *x*-axis [-1, 0]. Therefore, the desired transformation is:

$$\mathbf{R}_{90} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} X_x & X_y \\ Y_x & Y_y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \text{(eq. 10)}$$

(Example 2) Find the simple plane rotation that aligns the *x*-axis to the direction [1, 1].

From the previous example, we know that the new *y*-axis must be aligned with $[1 \quad 1]\mathbf{R}_{90} = [-1 \quad 1]$, so that the desired matrix is:

$$\mathbf{R}_{45} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

The normalization factor comes about because the Euclidean norm of each of the two rows is $\sqrt{2}$. Without this, the transformation would enlarge vectors as well as rotate them.

(Example 3) Find the simple plane rotation that rotates an arbitrary normalized vector **v** into another normalized vector **w**.

We approach this by first rotating **v** to align it to the *x*-axis, then rotating it to align it to **w**. In order to determine the first rotation, it is easier to specify the rotation from the *x*-axis to **v**, and then invert it.

Rotating the *x*-axis to **v** is accomplished by the matrix:

$$\mathbf{R}_{\mathbf{v}} = \begin{bmatrix} \mathbf{v} \\ \mathbf{v}\mathbf{R}_{90} \end{bmatrix} = \begin{bmatrix} v_x & v_y \\ -v_y & v_x \end{bmatrix}$$

Since this is an orthogonal matrix (all pure rotations are orthogonal), its inverse is equal to its transpose:

$$\mathbf{R}_{\mathbf{v}}^{-1} = \begin{bmatrix} \mathbf{v} \\ \mathbf{v}\mathbf{R}_{90} \end{bmatrix}^{T} = \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix}$$

The rotation from the x-axis to **w** is found similarly:

$$\mathbf{R}_{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ \mathbf{w}\mathbf{R}_{90} \end{bmatrix} = \begin{bmatrix} w_x & w_y \\ -w_y & w_x \end{bmatrix}$$

Therefore the desired transformation is the concatenation:

$$\mathbf{R}_{\mathbf{v}\to\mathbf{w}} = \mathbf{R}_{\mathbf{v}}^{-1}\mathbf{R}_{\mathbf{w}} = \begin{bmatrix} \mathbf{v} \\ \mathbf{v}\mathbf{R}_{90} \end{bmatrix}^{T} \begin{bmatrix} \mathbf{w} \\ \mathbf{w}\mathbf{R}_{90} \end{bmatrix} = \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} \begin{bmatrix} w_x & w_y \\ -w_y & w_x \end{bmatrix} = \begin{bmatrix} v_x w_x + v_y w_y & v_x w_y - v_y w_x \\ -(v_x w_y - v_y w_x) & v_x w_x + v_y w_y \end{bmatrix}$$

(eq. 11)

(Example 4) Find the skewing transformation suitable for italicizing letters by $\frac{1}{4}$ unit in *x* for every unit in *y*.

We basically just want to remap the *y*-axis to $\begin{bmatrix} \frac{1}{4} & 1 \end{bmatrix}$ while the *x*-axis remains the same:

$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ \frac{1}{4} & 1 \end{bmatrix}$$

(Example 5) Find the rotation that takes the vector $\frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ onto the *x*-axis, through the plane that contains them both.

The axis of rotation can be obtained as:

$$\mathbf{n} = \frac{\mathbf{v}\times\mathbf{x}}{\|\mathbf{v}\times\mathbf{x}\|} = \frac{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\times\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}{\left\|\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\times\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\right\|} = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & -1 \end{bmatrix}$$

where $\mathbf{x}$ is the x-axis. An orthogonal third vector can be obtained by crossing this with the given vector:

$$\mathbf{m} = \frac{\mathbf{n}\times\mathbf{v}}{\|\mathbf{n}\times\mathbf{v}\|} \frac{\begin{bmatrix} 0 & 1 & -1 \end{bmatrix}\times\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}}{\left\|\begin{bmatrix} 0 & 1 & -1 \end{bmatrix}\times\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\right\|} = \frac{1}{\sqrt{6}}\begin{bmatrix} 2 & -1 & -1 \end{bmatrix}$$

These three vectors then make up an orthonormal coordinate frame:

$$\mathbf{N} = \begin{bmatrix} \mathbf{v} \\ \mathbf{m} \\ \mathbf{n} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

This transformation takes the *x*-axis onto the vector [1 1 1], and the *z*-axis onto the axis of rotation. In order to simplify the rotation, we would like to have the inverse of this, namely, to transform the axis of rotation onto the *z*-axis. Since $\mathbf{N}$ is an orthogonal matrix, its inverse is simply the transpose:

$$\mathbf{N}^{-1} = \mathbf{N}^{T} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

It can be verified that the vector $\frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ maps onto the *x*-axis, that the axis of rotation, $\frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & -1 \end{bmatrix}$, maps onto the *z*-axis, and that the *x*-axis maps onto

$$\mathbf{x}' = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \end{bmatrix}.$$

A simple rotation about the *z*-axis in this frame would have the *y*-axis map into

$$\mathbf{y}' = \mathbf{x}'\mathbf{R}_{z90} = \begin{bmatrix} -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} & 0 \end{bmatrix}$$

where

$$\mathbf{R}_{z90} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(eq. 12)}$$

is the 3D analog of (eq. 10), and rotates the *x*-axis onto the *y*-axis around the *z*-axis. Putting this together into a transformation, we have:

$$\mathbf{R} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \\ -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We then need to go back to our original frame by using $\mathbf{N}$; the composite transformation is:

$$\mathbf{T} = \mathbf{N}^T\mathbf{R}\mathbf{N} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} & 0 \\ -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{2\sqrt{3}} + \frac{1}{2} & \frac{1}{2\sqrt{3}} - \frac{1}{2} \\ \frac{1}{\sqrt{3}} & \frac{1}{2\sqrt{3}} - \frac{1}{2} & \frac{1}{2\sqrt{3}} + \frac{1}{2} \end{bmatrix}$$

a somewhat formidable expression. No trigonometry per se has been used for this derivation: only cross products, vector normalization, and matrix multiplication. We generalize this in the next example.

(Example 6) Find the rotation that takes an arbitrary normalized vector $\mathbf{v}$ to another normalized vector $\mathbf{w}$, through the plane that contains them both.

Generalizing our experience with (Example 3) and (Example 5), we have the matrix that transforms $\mathbf{v}$ onto the *x*-axis and the axis of the plane of rotation onto the *z*-axis:

$$\mathbf{N}^T = \begin{bmatrix} \mathbf{v} \\ \mathbf{m} \\ \mathbf{n} \end{bmatrix}^T \qquad \text{(eq. 13)}$$

where

$$\mathbf{n} = \frac{\mathbf{v} \times \mathbf{w}}{\|\mathbf{v} \times \mathbf{w}\|} \qquad \text{(eq. 14)}$$

is the axis of rotation, and

$$\mathbf{m} = \frac{\mathbf{n} \times \mathbf{v}}{\|\mathbf{n} \times \mathbf{v}\|} \qquad \text{(eq. 15)}$$

is the third vector that completes a dextral orthogonal basis. The image of $\mathbf{w}$ in $\mathbf{N}^T$ is:

$$\mathbf{w}' = \mathbf{w}\mathbf{N}^T \qquad \text{(eq. 16)}$$

The transformation that rotates the *x*-axis (i.e. the image of $\mathbf{v}$) onto $\mathbf{w}'$ (the image of $\mathbf{w}$) is:

$$\mathbf{R} = \begin{bmatrix} \mathbf{w}' \\ \mathbf{w}'\mathbf{R}_{z90} \\ \mathbf{z} \end{bmatrix} \qquad \text{(eq. 17)}$$

where

$$\mathbf{z} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

is the *z*-axis. The desired transformation is then:

$$\mathbf{R}_{\mathbf{v} \to \mathbf{w}} = \mathbf{N}^T\mathbf{R}\mathbf{N} \qquad \text{(eq. 18)}$$

(Example 7) For a general transformation on a set of points, suppose that we would like to scale the *x*-axis by 2, the *y*-axis by 3, the *z*-axis by 4, and we want to reorient the object described in terms of those points so that the new *x*-axis points in the direction $\frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, the *y*-axis points in the direction $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, and the *z* axis points in the direction $\frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & -1 \end{bmatrix}$, and further, that the whole object be shifted in position by $\begin{bmatrix} 10 & 20 & -27 \end{bmatrix}$.

Just copying these specifications into a 4x3 matrix, we get:

$$\mathbf{T} = \begin{bmatrix} \frac{2}{\sqrt{3}} & \frac{2}{\sqrt{3}} & \frac{2}{\sqrt{3}} \\ 3 & 0 & 0 \\ 0 & \frac{4}{\sqrt{2}} & -\frac{4}{\sqrt{2}} \\ 10 & 20 & -27 \end{bmatrix}$$

Note that this transformation is more complex than others that we have encountered before, and would be virtually impossible to describe in terms of elementary rotation, scaling, skewing and translation operations, yet it was extremely simple to describe and implement in terms of a coordinate frame.

Another View of the Viewing Transformation

In Smith's paper on the Viewing Transformation [Smith 84], he analyzes the viewing transformation by breaking it up into elementary components. We base the construction of the viewing transformation on this paper, but we do so by analyzing the components in terms of their coordinate frames.

Foley and van Dam [Foley 82] have figures and text that strongly suggest construction of the components of the viewing transformation via coordinate frames, but they stop short of doing so.

The Camera's Modeling Frame

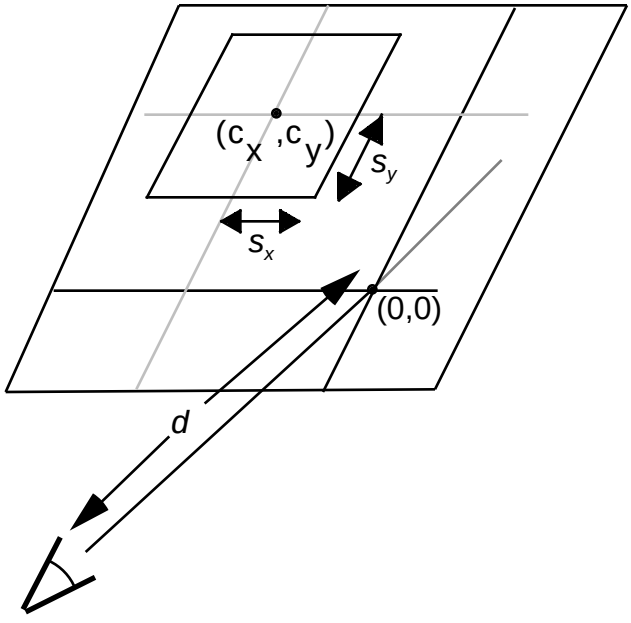
The camera is placed and oriented within the global coordinate frame with the 4x3 matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{O} \end{bmatrix} = \begin{bmatrix} x_x & x_y & x_z \\ y_x & y_y & y_z \\ z_x & z_y & z_z \\ o_x & o_y & o_z \end{bmatrix} \qquad \text{(eq. 19)}$$

where **O** is the camera's location, and **X**, **Y** and **Z** are its local *x*-, *y*-, and *z*-axes, respectively.

The Frame of the Camera's Viewing Plane

This frame accounts for the size of the window in the viewing plane and its displacement from the center:



$$\mathbf{V} = \frac{1}{d} \begin{bmatrix} s_x & 0 \\ 0 & s_y \\ c_x & c_y \end{bmatrix} \qquad \text{(eq. 20)}$$

The perpendicular distance to the viewing plane, d, is arbitrary, and serves only (along with s_x, s_y, c_x, and c_y) to define the pyramidal cone of a perspective view. Any distance can be used, in order to assure that the resultant image contains all of the objects of interest.

The scale factors s_x and s_y correspond to the half-width and half-height of the window on the viewing plane, and should have the same aspect ratio as the physical dimensions of the window on the display device.

Usually, the center of the view plane window is [0, 0], but other values are useful when partitioning the image up into sequential bands or subregions when there is not enough memory to render the entire image at once.

The Clipping Frame

This frame scales the axes in the viewing plane so that the coordinates of interest are in the rectangular parallelepiped { (x, y, z): $-1 \le x \le +1$, $-1 \le y \le +1$, $0 \le z \le +1$ }:

$$\mathbf{C} = \frac{1}{f}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

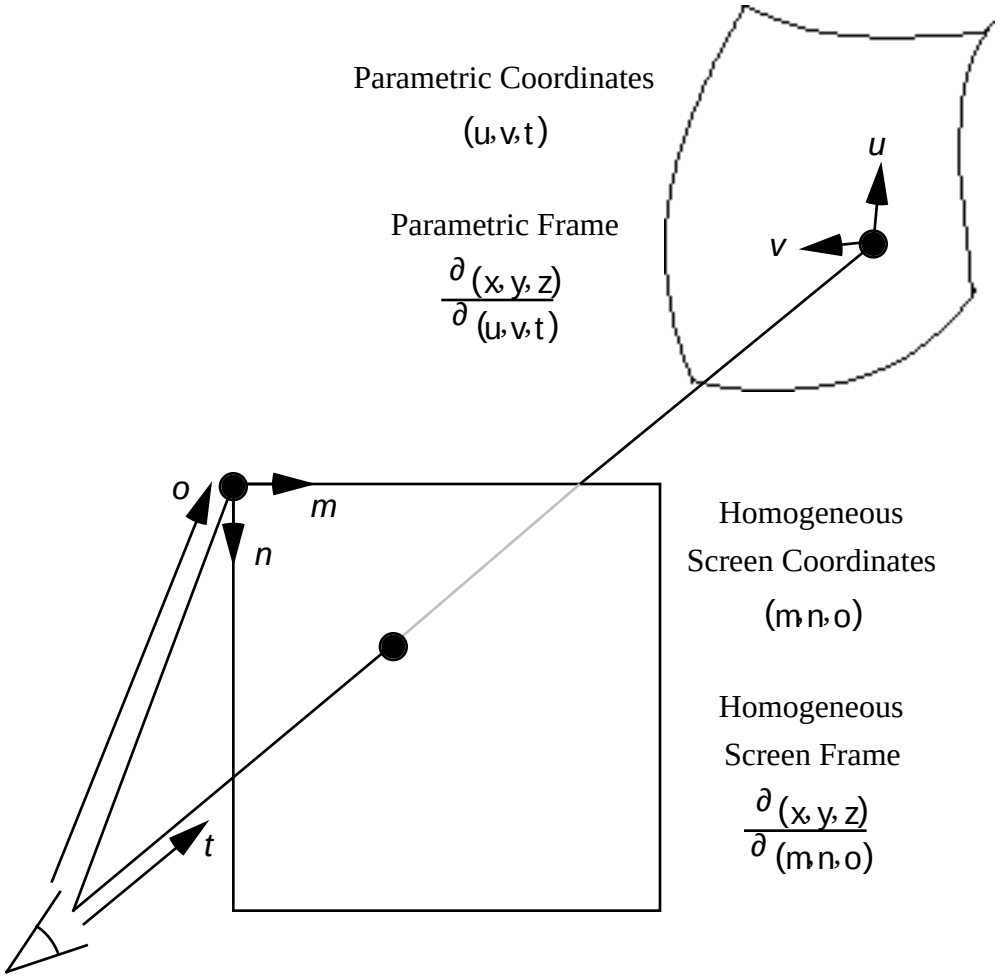
Since this is nothing more than an isotropic scale in 3D, it is either implemented as a scalar multiplication, or is incorporated into the $\mathbf{V}$ matrix above as:

$$\mathbf{CV} = \frac{f}{d}\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

Ray-Traced Texture-Mapping of Parametric Surfaces

Ray Tracing [Whitted 80] [Heckbert 84] [Amanatides 84] [Cook 84].

With texture-mapping, we link together three coordinate systems: the world, the screen, and the parameters of the ray and surface:



Coordinates on the image plane are homogeneous, such that, when $o = 1$, m and n correspond to the usual coordinates on the screen. The ray is parametrized in terms of t, and u and v are the parametric coordinates of the surface.

The unit vectors for m and n are the usual ones on the image plane, i.e. screen x (m) to the right by one pixel and screen y (n) down, by one pixel, although screen y could go up insead. The unit vectors for u and v are the usual ones for parametric surfaces. The unit vector for t is sometimes taken as the unit vector in (x, y, z) in the direction of the ray. The unit vector for o is the difference between the origin of the screen and the camera location. It should be noted that neither of the coordinate systems (m, n, o) and (t, u, v) are orthogonal nor right-handed, nor do they need to be. In the above diagram, the lines with arrows indicate unit vectors in each of these coordinates.

We can describe the unit vectors of the screen and parametric coordinate systems in terms of the world coordinates, (x, y, z):

$$\hat{\mathbf{m}} = \begin{bmatrix} x_m & y_m & z_m \end{bmatrix}, \qquad \hat{\mathbf{n}} = \begin{bmatrix} x_n & y_n & z_n \end{bmatrix}, \qquad \hat{\mathbf{o}} = \begin{bmatrix} x_o & y_o & z_o \end{bmatrix} \tag{eq. 21}$$

$$\hat{\mathbf{t}} = \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}, \qquad \hat{\mathbf{u}} = \begin{bmatrix} x_u & y_u & z_u \end{bmatrix}, \qquad \hat{\mathbf{v}} = \begin{bmatrix} x_v & y_v & z_v \end{bmatrix} \tag{eq. 22}$$

where the notation x_m indicates the change in x that corresponds to a unit change in m, or equivalently, the partial derivative of x with respect to m.

Note that these are considered to be unit vectors in their own coordinate system, and do not, in general, have unit length in the world coordinate system.

These vectors can be conveniently packed into Jacobian matrices to effect a change of coordinates:

$$\mathbf{S} = \begin{bmatrix} \hat{\mathbf{m}} \\ \hat{\mathbf{n}} \\ \hat{\mathbf{o}} \end{bmatrix} = \frac{\partial(x, y, z)}{\partial(m, n, o)} = \begin{bmatrix} x_m & y_m & z_m \\ x_n & y_n & z_n \\ x_o & y_o & z_o \end{bmatrix} \tag{eq. 23}$$

$$\mathbf{T} = \begin{bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{t}} \end{bmatrix} = \frac{\partial(x, y, z)}{\partial(u, v, t)} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_t & y_t & z_t \end{bmatrix} \tag{eq. 24}$$

One of the side-effects of using homogeneous screen coordinates is that:

$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\mathbf{S}$ = world coordinates of screen origin.

In order to do texture mapping with filtering, we need to map the *neighborhood* of a pixel to the *neighborhood* of a point in the surface's parametric space. This is accomplished with the aid of the Jacobian:

$$\mathbf{U} = \mathbf{S}\mathbf{T}^{-1} = \frac{\partial(u, v, t)}{\partial(m, n, o)} = \left[\begin{array}{cc|c} u_m & v_m & t_m \\ u_n & v_n & t_n \\ \hline u_o & v_o & t_o \end{array}\right] \tag{eq. 25}$$

The 2x2 submatrix, partitioned above in (eq. 25), is the portion that we are interested in. It is the differential coordinate frame of a pixel in terms of differential surface parametric coordinates. In particular, moving one pixel to the right corresponds to moving

$$\begin{bmatrix} u_m & v_m \end{bmatrix}$$

in (u, v) space, and moving one pixel down (or increasing screen y) corresponds to moving

$$\begin{bmatrix} u_n & v_n \end{bmatrix}$$

in (u, v) space. These would be the axes (major and minor) of an elliptically-weighted average filter [Heckbert 8x]. By finding the bounding box of these vectors, we can apply a summed-area table [Crow 84]. After finding the norm of the 2x2 matrix, we can apply MipMapping [Williams 83] [Turkowski 88].

The texture coordinates themselves are found from solving the system of three ray-surface intersection equations in the three unknowns *t*, *u*, and *v*.

••••••••

Given a ray

$$\mathbf{r}(t) = \mathbf{r}_0 + \mathbf{r}_\Delta t$$

a viewing trransformation

$$\mathbf{V}^{4x3}: (x,y,z) \rightarrow (m,n,o)$$

mapping world coordinates to homogeneous screen coordinates

•••••••••••••••••••••••••••••••••••• The rest is just a scrapbook ••••••••••••••••••••••••••••••••••••

Relationship between the Elementary Viewing Transformations and Coordinate Frames

In the following, we show that the elementary viewing transformations in [Smith 84] correspond to the inverse of appropriate coordinate frames.

In Smith's analysis, he has a matrix that represents the *position* of the camera:

$$\mathbf{A} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline -o_x & -o_y & -o_z & 1 \end{array}\right] \qquad \text{(eq. 26)}$$

where

$$\mathbf{o} = \begin{bmatrix} o_x & o_y & o_z \end{bmatrix} \qquad \text{(eq. 27)}$$

is the position (origin) of the camera in world space. The frame corresponding to this matrix is:

$$\mathbf{A}^{-1} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline o_x & o_y & o_z & 1 \end{array}\right] \qquad \text{(eq. 28)}$$

The Camera's Orientation Frame

Smith's representation of the *orientation* of the camera is:

$$\mathbf{B} = \left[\begin{array}{c|c} \mathbf{x} & 0 \\ \hline \mathbf{y} & 0 \\ \hline \mathbf{z} & 0 \\ \hline \mathbf{0} & 1 \end{array}\right]^{-1} = \left[\begin{array}{ccc|c} x_x & x_y & x_z & 0 \\ \hline y_x & y_y & y_z & 0 \\ \hline z_x & z_y & z_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]^{-1} \qquad \text{(eq. 29)}$$

where

$$\mathbf{x} = \begin{bmatrix} x_x & x_y & x_z \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y_x & y_y & y_z \end{bmatrix}$$

$$\mathbf{z} = \begin{bmatrix} z_x & z_y & z_z \end{bmatrix} \qquad \text{(eq. 30)}$$

represent the **x**, **y** and **z** vector axes of the camera in world coordinates. If these vectors are orthonormal, then **B** simplifies to:

$$\mathbf{B}_{ortho} = \left[\begin{array}{c|c|c|c} \mathbf{x}^t & \mathbf{y}^t & \mathbf{z}^t & \mathbf{0} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c|c|c} x_x & y_x & z_x & 0 \\ \hline x_y & y_y & z_y & 0 \\ \hline x_z & y_z & z_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \text{(eq. 31)}$$

The frame corresponding either of these matrices is:

$$\mathbf{B}^{-1} = \left[\begin{array}{c|c} \mathbf{x} & 0 \\ \hline \mathbf{y} & 0 \\ \hline \mathbf{z} & 0 \\ \hline \mathbf{0} & 1 \end{array}\right] = \left[\begin{array}{c|c|c|c} x_x & x_y & x_z & 0 \\ \hline y_x & y_y & y_z & 0 \\ \hline z_x & z_y & z_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \text{(eq. 32)}$$

The Camera's Modeling Frame

Concatenating the **A** and orthogonal **B** matrices, we arrive at:

$$\mathbf{AB}_{ortho} = \left[\begin{array}{c|c|c|c} x_x & y_x & z_x & 0 \\ \hline x_y & y_y & z_y & 0 \\ \hline x_z & y_z & z_z & 0 \\ \hline -x_x o_x - x_y o_y - x_z o_z & -y_x o_x - y_y o_y - y_z o_z & -z_x o_x - z_y o_y - z_z o_z & 1 \end{array}\right] \quad \text{(eq. 33)}$$

Note the coupling of the effects of the **A** and **B** matrices. Using the more general, non-orthogonal **B** matrix yields an even more complicated expression. If we instead look at the corresponding concatenated *frames*, we arrive at:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} = \left[\begin{array}{c|c|c|c} x_x & x_y & x_z & 0 \\ \hline y_x & y_y & y_z & 0 \\ \hline z_x & z_y & z_z & 0 \\ \hline o_x & o_y & o_z & 1 \end{array}\right] \quad \text{(eq. 34)}$$

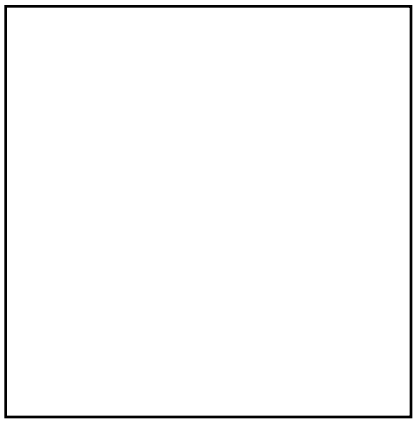
Unlike (eq. 33), this expression is valid for the more general, non-orthogonal **B** matrix. It uncouples the effects of camera position and orientation. The similarity to (eq. 4) allows us to represent this as a more compact 4x3 coordinate frame matrix, as in (eq. 1):

$$\mathbf{M} = \left[\begin{array}{c|c|c} x_x & x_y & x_z \\ \hline y_x & y_y & y_z \\ \hline z_x & z_y & z_z \\ \hline o_x & o_y & o_z \end{array}\right] \quad \text{(eq. 35)}$$

The significance of this frame is that it allows the camera to be moved in the scene in the same manner as any visible object in the scene.

Examples of the Camera Modeling Frame

Blah, blah, blah. Many figures.



(fig. 1)

The Frame of the Viewing Plane

Smith's next transformation accounts for projecting onto a portion of the viewing plane that is not centered on the viewing axis:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{c_x}{d} & -\frac{c_y}{d} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(eq. 36)

where $\begin{bmatrix} c_x & c_y \end{bmatrix}$ is the center of the projection onto the viewing plane, as offset from the normal viewing axis, and d is the perpendicular distance from the camera to the viewing plane. The distance d is arbitrary: all that matters is the *ratio*, which determines an *orientation* relative to the normal.

The inverse of this matrix is:

$$\mathbf{C}^{-1} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{c_x}{d} & \frac{c_y}{d} & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

(eq. 37)

At first glance, this does not look like a coordinate frame of the form (eq. 4). However, when we view this as a coordinate frame within the plane, we find that the 3x2 submatrix:

$$\begin{bmatrix} & \\ 1 & 0 \\ 0 & 1 \\ \frac{c_x}{d} & \frac{c_y}{d} \end{bmatrix}$$

(eq. 38)

represents a *translation in the plane* as indicated in the following diagram:

(fig. 2)

The Frame of the Canonical Viewing Volume

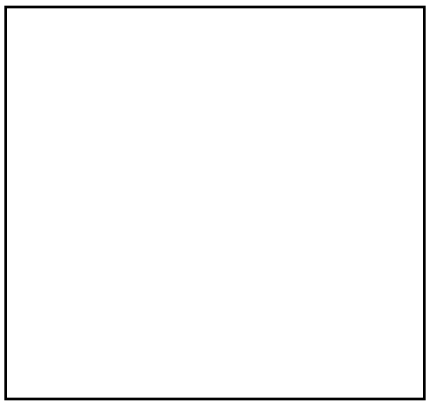
The transformation to the canonical viewing volume is given by Smith as:

$$\mathbf{D} = \left[\begin{array}{cccc} \frac{d}{s_x f} & 0 & 0 & 0 \\ 0 & \frac{d}{s_y f} & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \qquad \text{(eq. 39)}$$

Its inverse,

$$\mathbf{D}^{-1} = \left[\begin{array}{cccc} \frac{s_x f}{d} & 0 & 0 & 0 \\ 0 & \frac{s_y f}{d} & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \qquad \text{(eq. 40)}$$

Can be easily interpreted as a change in the scale of the coordinate axes. Each of the x, y, and z-axes are scaled by the distance to the far clipping plane (i.e. they eventually will all be divided by z in order to get the desired perspective foreshortening). This accounts for the factor of f. the sx/d and sy/d specify the size of the axes as normalized to the viewing plane.



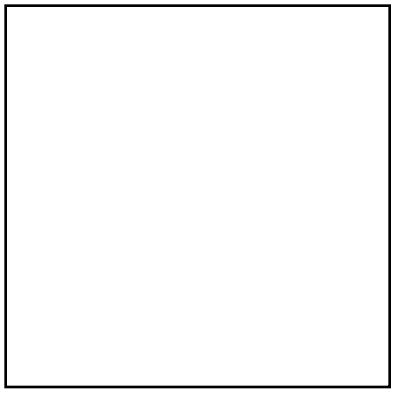
(fig. 3)

The Composite Frame of the Canonical Viewing Volume

The composition of the preceding two frames is:

$$(\mathbf{CD})^{-1} = \mathbf{D}^{-1}\mathbf{C}^{-1} = \left[\begin{array}{cccc} \frac{fs_x}{d} & 0 & 0 & 0 \\ 0 & \frac{fs_y}{d} & 0 & 0 \\ \frac{fc_x}{d} & \frac{fc_y}{d} & f & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \quad \text{(eq. 41)}$$

This is illustrated by the following figure:



(fig. 4)

References

Amanatides 84 Amanatides, John, “Ray Tracing with cones”, *Computer Graphics* (SIGGRAPH ‘84 Conference Proceedings), Vol. 18, no. 3, pp. 129-135 (1984).

Cook 84 Cook, Rob L., Tom Porter, and Loren Carpenter, “Distributed Ray Tracing”, *Computer Graphics* (SIGGRAPH ‘84 Conference Proceedings), vol. 18, no. 3, pp. 137-145 (1984).

Crow 84 Crow, Franklin C., “Summed-Area Tables for Texture Mapping”, *Computer Graphics* (SIGGRAPH Conference Proceedings ‘84), Vol. 18, No. 3, pp. 207-212, July 1984.

Foley 82 Foley, James D. and Anrdies van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

Heckbert 84 Heckbert, Paul S. and Pat Hanrahan, “Beam Tracing Polygonal Objects”, *Computer Graphics* (SIGGRAPH ‘84 Conference Proceedings), Vol. 18, no. 3, pp. 119-127 (1984).

Heckbert 8x Elliptically...

Newman 79 Newman, William M. and Robert F Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, 1979.

Smith 84 Smith, Alvy Ray, “The Viewing Transformation”, Technical Memo No. 84, Lucasfilm, Ltd., revised May 4, 1984.

Turkowski 88 Turkowski, Ken, “The Differential Geometry of Texture Mapping”, Apple Technical Report # 10, Apple Computer, Inc., Cupertino, CA., May 1988.

Whitted 80 Whitted, Turner, “An improved illumination model for shaded display”, Comm. ACM, Vol. 23, No. 6, pp. 343-349 (1980).

Williams 83 Williams, Lance, “Pyramidal Parametrics”, *Computer Graphics* (SIGGRAPH ‘83 Conference Proceedings), vol. 17, no. 3, July 1983.