

3D Modeling Transformations

Introduction to Computer Graphics

Arizona State University

Dianne Hansford

February 13, 2005

1 Introduction

In our last course notes, we stepped-through 2D transformations. Now, 3D transformations should be easy! The 3D transformation matrix A is a 3×3 matrix:

$$A = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \mathbf{a}_3] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

The linear map, which takes a vector \mathbf{v} in the $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$ system to a vector $\hat{\mathbf{v}}$ in the $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ system, has the form $\hat{\mathbf{v}} = A\mathbf{v}$. Therefore,

$$\mathbf{e}_1 \rightarrow \mathbf{a}_1 \quad \mathbf{e}_2 \rightarrow \mathbf{a}_2 \quad \mathbf{e}_3 \rightarrow \mathbf{a}_3.$$

One measure of the behavior of a linear map is the *determinant* of the matrix A , or $|A|$. In 3D the determinant measures the *volume* of the parallelepiped formed by the \mathbf{a}_i vectors (positioned at the origin). Recall that you use co-factor expansion to compute the determinant of a 3×3 matrix.

One other notable difference between 2D and 3D transformations: rotations are not commutative in 3D. More on rotations below.

2 3D Linear Maps

2.1 Scaling

$$A = \begin{bmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{bmatrix} \quad |A| = s_{11}s_{22}s_{33}$$

2.2 Reflection

$$A = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \quad \text{such that } |A| = -1$$

2.3 Shear

We can shear in one or more coordinate-axes. Let's look at some examples.

Shear parallel to the $\mathbf{e}_1, \mathbf{e}_2$ -axes:

$$\mathbf{e}_1 \rightarrow \mathbf{e}_1 \quad \mathbf{e}_2 \rightarrow \mathbf{e}_2 \quad \mathbf{e}_3 \rightarrow \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}.$$

Therefore,

$$A = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}.$$

An arbitrary vector $\mathbf{v} \rightarrow \begin{bmatrix} v_1 + av_3 \\ v_2 + bv_3 \\ v_3 \end{bmatrix}$.

Shear parallel to the $\mathbf{e}_2, \mathbf{e}_3$ -axes:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ \star & 1 & 0 \\ \star & 0 & 1 \end{bmatrix}.$$

Shear parallel to the $\mathbf{e}_1, \mathbf{e}_3$ -axes:

$$A = \begin{bmatrix} 1 & \star & 0 \\ 0 & 1 & 0 \\ 0 & \star & 1 \end{bmatrix}.$$

The determinant of a shear: $|A| = 1$, thus the volume of a sheared object does not change.

Here is a special shear that you have encountered before, most likely unknowingly:

$$\mathbf{e}_1 \rightarrow \begin{bmatrix} 1 \\ -b/a \\ -c/a \end{bmatrix} \quad \mathbf{e}_2 \rightarrow \mathbf{e}_2 \quad \mathbf{e}_3 \rightarrow \mathbf{e}_3,$$

resulting in the following mapping

$$\begin{bmatrix} 1 & 0 & 0 \\ -b/a & 1 & 0 \\ -c/a & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}.$$

This is a Gauss Elimination step!

2.4 Projection

Projections “flatten” our geometry. In computer graphics, we are interested in *parallel projections*, which are either *orthographic* or *oblique*. The most common (linear map) projection is the following orthographic projection:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

which projects all vectors in to the $\mathbf{e}_1, \mathbf{e}_2$ -plane.

We will discuss projections in more detail when we get to Viewing. Then we will introduce the *perspective projection*, which isn’t a linear map. (The foreshortening process doesn’t preserve linear combinations of vectors.)

2.5 Rotation

The rotation matrices will be set up so that the *right-hand rule* is preserved: Align your right thumb with the positive axis about which you will rotate. Your fingers will curl from one axis to another: A positive angle of rotation will rotate in this direction. Rotation about \mathbf{e}_3 :

$$A = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

rotates “from \mathbf{e}_1 to \mathbf{e}_2 ”. (Recall this is our 2D transformation.)

Rotation about \mathbf{e}_2 :

$$A = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix},$$

rotates from \mathbf{e}_3 to \mathbf{e}_1 .

Rotation about \mathbf{e}_1 :

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix},$$

rotates from \mathbf{e}_2 to \mathbf{e}_3 .

The column vectors of a rotation matrix are orthogonal and unit length, and thus *orthonormal*. A matrix with orthonormal column vectors is called *orthogonal*. These matrices have special a property that is quite handy for computer graphics:

$$R^{-1}(\alpha) = R(-\alpha) = R^T(\alpha),$$

so to reverse a rotation is as simple as applying the transpose of the rotation!

In 3D, matrix products *do not commute*, and this includes rotations. (Recall that rotations commute in 2D.) Sketch the following example. Let

- $R_{\mathbf{e}_3}$ be a 90° rotation about \mathbf{e}_3 , and
- $R_{\mathbf{e}_1}$ be a 90° rotation about \mathbf{e}_1 .

Try applying these rotations to the vector \mathbf{e}_1 . You'll see that

$$\mathbf{e}_2 = R_{\mathbf{e}_3}R_{\mathbf{e}_1}\mathbf{e}_1 \quad \text{and} \quad \mathbf{e}_3 = R_{\mathbf{e}_1}R_{\mathbf{e}_3}\mathbf{e}_1.$$

This simple example illustrates the non-commutivity of matrix multiplication.

2.6 Properties

A linear map, $\hat{\mathbf{v}} = A\mathbf{v}$,

- Maps vectors \rightarrow vectors
- Preserves linear combinations of vectors. Suppose we have

$$\mathbf{w} = \alpha\mathbf{u} + \beta\mathbf{v}$$

Apply the linear map:

$$\hat{\mathbf{v}} = A\mathbf{v} \quad \hat{\mathbf{u}} = A\mathbf{u} \quad \hat{\mathbf{w}} = A\mathbf{w}$$

And the relationship still holds:

$$\hat{\mathbf{w}} = \alpha\hat{\mathbf{u}} + \beta\hat{\mathbf{v}}$$

3 3D Affine Maps

An affine map in 3D takes the form

$$\hat{\mathbf{x}} = A(\mathbf{x} - \mathbf{o}) + \mathbf{p} = A\mathbf{x} + \mathbf{p}.$$

As before, for a point \mathbf{x} in the coordinate system defined by $\mathbf{o}, [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$, an affine map gives the coordinates for the corresponding point $\hat{\mathbf{x}}$ in the coordinate system defined by $\mathbf{p}, [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$.

Affine maps have the following nice properties.

- The ratio of three collinear points is invariant.
- Barycentric combinations are invariant.
- Parallel planes are mapped to parallel planes.

4 Homogeneous Coordinates

The homogeneous form of a 3D point $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ is $\bar{\mathbf{x}} = \begin{bmatrix} x_1x_4 \\ x_2x_4 \\ x_3x_4 \\ x_4 \end{bmatrix}$. Most of the

time, when you have a 3D point that you want to represent in homogeneous

coordinates, you will write it as $\bar{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}$, that is, $x_4 = 1$.

Homogenous coordinates represent 4D points. The fourth coordinate is called the w -coordinate. When we form a 3D point from a homogeneous point, we divide each coordinate by the w -coordinate,

$$\mathbf{x} \approx \bar{\mathbf{x}} = \begin{bmatrix} x_1x_4/x_4 \\ x_2x_4/x_4 \\ x_3x_4/x_4 \\ x_4/x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}.$$

We use the \approx sign and say that “ \mathbf{x} corresponds to $\bar{\mathbf{x}}$.” Commonly we think of the 3D point corresponding to the 4D point with $w = 1$. Although, all points on a line through the origin are considered to be equivalent. In other words, there are many homogeneous representations of the same (affine) 3D point. This idea is more easily understood (and sketched) by considering the homogeneous form of a 2D point, which is a 3D point. All points with $z = 1$ are affine points.

Homogeneous coordinates serve two purposes in computer graphics:

1. A notation that allows an affine map (A and \mathbf{p}) to be consolidated into one matrix.
2. A representation for a perspective projection in matrix form.

In OpenGL, matrices are the cornerstone of the mathematics of the pipeline: e.g., the modelview and projection matrices. Both of these matrices are 4×4 in size, and they arranged such that

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & p_1 \\ a_{21} & a_{22} & a_{23} & p_2 \\ a_{31} & a_{32} & a_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or in short-hand notation:

$$\begin{bmatrix} & A & & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Later, the zeros in the bottom row will be replaced with a perspective map definition.

Previously we defined an affine map as

$$\mathbf{x}' = A\mathbf{x} + \mathbf{p}.$$

To put this transformation in the context of OpenGL's matrix-oriented pipeline, let's rewrite the affine map in matrix form with homogeneous coordinates:

$$\begin{aligned} \bar{\mathbf{x}}' &= \begin{bmatrix} I & & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A & & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{x}} \\ &= \bar{T}\bar{A}\bar{\mathbf{x}} \\ &= \begin{bmatrix} & A & & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{x}}. \end{aligned}$$