Linear Algebra in Computer Graphics

In two dimensions, we can represent a vector by a 2x1 column matrix

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

We can scale a vector by multiplying by a suitable diagonal matrix S

$$\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} s_x x \\ s_y y \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = S\vec{x}$$

We can also rotate a vector through an angle θ using an orthonormal matrix R

$$\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R\vec{x}$$

We can not do translations this way

$$\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

since these are not accomplished by matrix multiplication alone. In order to get translations, we must extend the concept of a vector to

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This vector is known as a "homogeneous vector," and the general case is

$$\begin{bmatrix} x \\ y \\ W \end{bmatrix}$$

We represent a 2D point by a line $\{(tx, ty, tW), t \in \mathbb{R}\}$ Choosing $t = 1/W$, we have the normalized homogeneous vector

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

In terms of the 2D homogeneous vector, we have translation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x+d_x \\ y+d_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

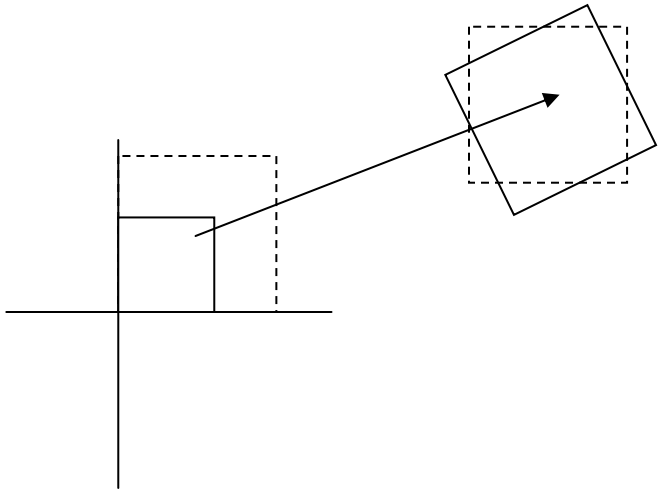
To summarize, we have 3 fundamental matrix operations, rotation, scaling and translation.

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

It is easy to see that any “rigid body” motion can be accomplished by first scaling, then translating, then rotating an object:



The extension to three-dimensional coordinates is quite easy. We have the 3D scaling matrix

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the 3D translation matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and three 3D rotation matrices

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We also have a special 4x4 matrix called the “perspective transform”

$$P = \begin{bmatrix} \frac{1}{\tan(\mu)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\nu)} & 0 & 0 \\ 0 & 0 & \frac{B+F}{B-F} & \frac{-2BF}{B-F} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where μ is the angle between the camera and the plane containing the camera and the right hand edge of the screen. ν is the angle between the camera and the top edge of the screen. B is the distance to the *back clipping plane* (the farthest distance any object can be from the camera) and F is the distance from the *front clipping plane* (the closest distance any object can be. [See http://en.wikipedia.org/wiki/3D_projection]

There are other 4x4 matrices which implement other camera/perspective models, as well as reflections.

Modern graphics chips can implement these 4x4 matrix operations in hardware (10^8 times a second!) as well as perform operations in parallel (called pipelining). These speeds allow objects with millions of polygons to be rendered in fractions of a second.

The main point of this note is to remark that A LOT OF COMPUTER GRAPHICS IS BASED ON LINEAR ALGEBRA!