3D Linear Algebra Cheat Sheet

A 3x3 graphics matrix can be used to TRANSFORM (scale, shear, and rotate) each vertex of a 3D model. A movement (translation) vector can also be added as the fourth column, making a 4x4 matrix (the bottom row is used to keep it square). Where appropriate, both 3x3 and 4x4 are discussed here.

BASIS VECTORS are the fundamental vectors of a coordinate system. In 3D they are called $\hat{\imath}$ (i-hat), $\hat{\jmath}$ (j-hat), and $\hat{k}$ (k-hat). So, in a standard (i.e., untransformed right-hand) coordinate system, they are:

$$\hat{\imath}=\begin{bmatrix}1\\0\\0\end{bmatrix},\ \hat{\jmath}=\begin{bmatrix}0\\1\\0\end{bmatrix},\ \hat{k}=\begin{bmatrix}0\\0\\1\end{bmatrix}$$

Any transformation (not counting translation) can be fully described by specifying only what would happen to standard basis vectors if they were transformed. So, the COLUMNS in the 3x3 part of a 3D transformation matrix are those vectors. For example:

$$\begin{bmatrix}1&2&3&0\\4&5&6&0\\7&8&9&0\\0&0&0&1\end{bmatrix}\ \Rightarrow\ \hat{\imath}=\begin{bmatrix}1\\4\\7\end{bmatrix},\ \hat{\jmath}=\begin{bmatrix}2\\5\\8\end{bmatrix},\ \hat{k}=\begin{bmatrix}3\\6\\9\end{bmatrix}$$

Any vector is transformed by a 3x3 matrix like this:

$$\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}xa+yb+zc\\xd+ye+zf\\xg+yh+zi\end{bmatrix}$$

And by a 4x4 matrix like this:

$$\begin{bmatrix}a&b&c&X\\d&e&f&Y\\g&h&i&Z\\0&0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}xa+yb+zc+X\\xd+ye+zf+Y\\xg+yh+zi+Z\end{bmatrix}$$

See how this standard (untransformed) $\hat{\jmath}$ transforms to the second column from this 3x3 matrix:

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}=\begin{bmatrix}2\\5\\8\end{bmatrix}$$

(You can certainly transform an already transformed coordinate system, but in that case the resulting basis vectors would not necessarily equal the columns in the transformation matrix.)

MULTIPLYING two matrices produces a *composite* matrix that, in one step, can transform a vector the same as transforming it with the original right matrix then transforming it with the left matrix. Since the columns in a matrix define the new basis vectors for that transform, each column in the result of that multiply is each column in the right-side (input) matrix used as a vector transformed by the left-side matrix:

$$\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}\begin{bmatrix}p&q&r\\s&t&u\\v&w&x\end{bmatrix}=\begin{bmatrix}pa+sb+vc&qa+tb+wc&ra+ub+xc\\pd+se+vf&qd+te+wf&rd+ue+xf\\pg+sh+vi&qg+th+wi&rg+uh+xi\end{bmatrix}$$

(You can also expand this formula to 4x4 matrices.)

As you can see, matrix multiplication is NOT commutative. However, it is associative.

A DOT PRODUCT (also called inner product) of two vectors is how of much of one's length is shared with the other (one vector is projected onto the other):

$$\begin{bmatrix}a\\b\\c\end{bmatrix}.\begin{bmatrix}d\\e\\f\end{bmatrix}=ad+be+cf$$

Thus, the dot product of orthogonal vectors is zero.

A (3D) DETERMINANT of three vectors indicates the volume of the parallelepiped created by those vectors (that is, it is twice the area enclosed by the vectors). The sign is positive if an even number of axes have been flipped, and negative if an odd number have been flipped. If the three vectors are arranged as the columns in a matrix, the determinant is calculated by adding all products of tiled left-leaning diagonals and subtracting all products of tiled right-leaning diagonals:

$$det\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}=aei+bfg+cdh-ceg-bdi-afh$$

An INVERSE matrix is analogous to a reciprocal, allowing you to do algebra on matrices. For example, if $AB=C$, then $A=CB^{-1}$. For a 3x3 matrix, it can be calculated like this:

If $M=\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}$

then $M^{-1}=\begin{bmatrix}ei-fh&ch-bi&bf-ce\\fg-di&ai-cg&cd-af\\dh-eg&bg-ah&ae-bd\end{bmatrix}/det(M)$

Search online for calculating an inverse 4x4 matrix, or just use a library!

The CROSS-PRODUCT of two vectors in 3-space is a vector whose length is their determinant in the plane those vectors define, and whose direction is orthogonal to those vectors using a right-hand rule:

$$\begin{bmatrix}a\\b\\c\end{bmatrix}\times\begin{bmatrix}d\\e\\f\end{bmatrix}=\begin{bmatrix}bf-ec\\dc-af\\ae-db\end{bmatrix}$$

Thus, in a standard coordinate system, $\hat{\imath}\times\hat{\jmath}=\hat{k}$

SPAN is all points that can be described by the sum of a set of vectors whose lengths can take on any value but directions do not change.

COLUMN SPACE is the space of all possible resulting spans of a given transformation. For example, if a transformation causes all points to lie on a plane, then that plane is the column space.

RANK is how many dimensions in a column space. For example, if column space is a plane, then the rank is 2.

If you use column vectors (rather than row vectors), as does this cheat sheet and the vast majority of math literature, you can think of OpenGL, Monogame, and Direct3D as STORING MATRICES in column-major order:

$\begin{bmatrix}a&b&c\\d&e&f\\g&h&i\end{bmatrix}$ resides in successive locations as *a,d,g,b,e,h,c,f,i.*

So, an element of MyArray[3][3] would be accessed as MyArray[row,column]. Performance issues to consider are: cache behavior vs. CPU processing order (multiplies, etc.), shaders vs. 3D library, and configuring the 3D library to internally transpose matrices.

Math literature uses a standard right-handed COORDINATE SYSTEM, where the thumb of the right hand points along x, index finger points along y, and remaining fingers point along z. OpenGL and Monogame use a right-handed system by default. Direct3D uses a left-handed system by default. Fortunately, there are ways to use each of those libraries in either coordinate system without much, if any, performance hit.

If you can get your equations into this form:

ax + by + cz = d
ex + fy + gz = h
ix + jy + kz = m

Then you can write them like this:

$$\begin{bmatrix}a&b&c\\e&f&g\\i&j&k\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}d\\h\\m\end{bmatrix}$$

And use algebra to solve it like this:

$$\begin{bmatrix}a&b&c\\e&f&g\\i&j&k\end{bmatrix}^{-1}\begin{bmatrix}d\\h\\m\end{bmatrix}=\begin{bmatrix}x\\y\\z\end{bmatrix}$$

This document freely available at https://github.com/Blotch3D/Blotch3D
Kelly Loum, 2008