APPENDIX: SOME USEFUL MATHEMATICS

Introduction

Various aspects of computer graphics make use of mathematics. This is not surprising because a lot of computer graphics involves geometry and 3D spaces. This material is here to help you understand what is going on. In particular, we will try to explain in computer graphics terms, rather than giving a piece of abstract mathematics. We will use the correct mathematical terminology and we will try to be accurate with the mathematics that does appear. Our aim is to give a feel for what the mathematics means, in the context of computer graphics. The description treads a fine line between full mathematical rigour and the rather limited descriptions given in quite a few computer graphics textbooks. So, let me offer apologies in advance if I occasionally fall the wrong side of that line.

Spaces and Coordinates

Much of what we do in graphics involves the idea of a space, usually 2D or 3D, sometimes 4D. Typically we think of this space as having a coordinate system, such as (x, y, z). This *Cartesian coordinate* system is by the far the commonest in graphics. Spaces can be thought about in the abstract – coordinates were a relatively late invention, long after the Ancient Greek geometers had produced their results – so coordinates are not fundamental. Even so, every computer graphics library uses them, so we will too.

For a given set of values (x_0, y_0, z_0) we can think of the coordinates as defining a *point* in that space. We also have the idea of a *vector*, which we think of as having a direction and a magnitude. Thus we can think of the 3-tuple (x_0, y_0, z_0) as representing either a point in space or, as seen from the origin, a vector in space. This equivalence of a point and a direction is important for projective geometry. The *length* of this vector is just the square root of the sum of the squares of its components. If the length is 1, we have a *unit vector*. The *axes* of our coordinate system are defined by three *orthogonal* unit vectors; orthogonal being what we informally call "at right-angles" to each other.

Linear Algebra

Linear algebra is concerned with vectors and vector spaces, when subjected to *linear transformations* (to be explained shortly). These areas are the reason it is of interest in computer graphics. It is also concerned with *systems of linear equations*, often the simultaneous solution of a set of linear equations.

Many introductory linear algebra courses often focus on simultaneous equations and computer graphics books do not. It may thus appear that simultaneous equations have little to do with computer graphics. In fact that is not true. Linear transformations are very important in computer graphics and so simultaneous equations are: you can think of a point as being the simultaneous solution to two equations defining intersecting lines; you can think of a line as the simultaneous solution to the equations of two planes etc. See also homogeneous coordinates, later.

Linear equations have no power greater than one, so

$$a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n = 0$$

is a the general linear equation. This has n independent variables x_i, which can be thought of in much the same way that we think of coordinates. For example, in 3D, a line can be represented as:

$$a_1x + a_2y + a_3z = 0$$

Notice we said this is a *line*; the equation is *linear*! The equation to a circle is not linear and requires powers higher than one; in fact, powers of 2:

$$x^2 + y^2 - r^2 = 0$$

We can also turn such equations into inequalities, to express the idea of "to one side" of the line. Points which satisfy:

$$a_1x + a_2y + a_3z > 0$$

are to one side of the line, while points which satisfy:

$$a_1x + a_2y + a_3z < 0$$

are to the other side, while points which satisfy:

$$a_1x + a_2y + a_3z = 0$$

are exactly on the line. (We can do the same with other equations, including non-linear ones. For the closed curves such as the circle, inequalities give us an inside/outside test.)

Matrices

If a set of simultaneous equations has a solution, there must be at least as many equations as there are unknowns. If the equations are *linearly independent* then we only need have exactly as many equations as unknowns. Three equations are not independent if one can be constructed as a linear sum of the other two. There is a trivial case if one equation is a simple multiple of the another. In computer graphics terms, this means that one line is a scaled version of another.

If our equations are indeed independent, we can represent the whole system as a matrix A (of the coefficients a_{ij}), multiplied by a vector x (of the x_j) to give a vector result of zero. Compare this with the general linear equation above to see that this is the same thing but with several linear equations instead of one.

$$Ax^T = 0$$

Note: The "T" means transpose; we swap the rows for columns and vice versa.

Note: We can represent the same matrix in row-order or in column order, so we could have written this:

$$xA^T = 0$$

The mathematics does not change but it does affect the way you program it, so be clear which you are using. You can't mix both conventions in the same equation.

Transformations

A *transformation* is a mapping one vector space into another. It can be thought of as a function which changes the position and/or direction and/or scale of the axes of the coordinate system.

A function is linear if f respects addition and scaling:

$$f(x_1) + f(x_2) = f(x_1 + x_2)$$

$$sf(x) = f(sx).$$

Transformations are often given a geometric interpretation, such as rotation, reflection and translation. Matrices are the practical way we do this in computer graphics, which is why you will hear phrases such as “rotation matrix”.

The transformations you permit give the space a particular “flavour” and can be thought of as defining the space. Let’s examine that next.

Euclidean, Affine and Projective Transformations

Euclidean transformations

Euclidean transformations preserve length and angle, so the shape of an object does not change: straight lines transform to straight lines, planes transform to planes and circles transform to circles, for example. Only the position and orientation of the object changes, so they are sometimes called *rigid body transformations*.

If we permit only translation, rotation and reflection, we have a Euclidean space. (Reflection is arguably less fundamental, being just a change of sign convention.)

All Euclidean transformations can be decomposed into these operations. We can represent any Euclidean transformation as a matrix of appropriate size. For example:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix} + T$$

where R is a rotation matrix and T is a translation vector.

Affine transformations

Affine transformations do not preserve length or angle. Lines transform to lines and parallel lines remain parallel. However circles may transform to ellipses.

If we permit shearing or independent scaling of each axis, we have an affine transformation. (These are obviously not rigid body transformations.) We can also think of this as relaxing the characteristic of a Euclidean transformation, that a rotation has an inverse. If we replace this rotation with an operation which does not have to have an inverse (i.e. a non-singular operation), then we get an affine transformation instead.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + T$$

All Euclidean transformations are affine transformations but not the other way round. Affine tranformations are the more general case.

Projective transformations

These are the most general. Straight lines transform to straight lines and that’s about all we can say. Parallel lines may transform to non-parallel ones. Circles may transform to circles, or to ellipses, or to parabolas or to hyperbolas: i.e. to any conic.

Because of this generality, affine (and hence Euclidean) transformations are special cases of projective ones, as we can see from the following matrix form.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} A & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The whole matrix is a projective transformation. If A is non-singular, then it becomes an affine transformation. If A is a rotation matrix, it becomes a Euclidean transformation.

If the "1" in the vectors looks like a special case, note that it can certainly be some other value; but these are just scalings of the same projective point, so they will produce no new results. Even so, it can sometimes be useful practically. For example, if we are in danger of getting rounding errors in a computer calculation, we can scale everything, normalise at the end and still be sure the results will be what we wanted.

If the unit vector across the bottom of the matrix looks like a special case: it is. A general projective transformation is not required to have a unit vector here. The example uses it only to show that affine and Euclidean transformations can be embedded in the projective space as special cases.

Rotations

We probably have a clear idea of a rotation in 3D: we often say that we are rotating "about an axis", for example about the x-axis. In fact this isn't strictly accurate. Where rotation is concerned, 3D space is a special case. In 2D for example, we rotate in the xy plane. In our 3D world, we might think that this is "about" z, but this axis does not exist in the 2D space.

It is better to think of a rotation as taking place parallel to a given plane, with the advantage that this also is what we do in 4D and higher dimensions. In computer graphics when we give the 3D rotation matrix R_x, it is better to think of this as shorthand for R_{yz}, the rotation parallel to the xy plane.

Suppose we go to 4D. There are clearly four axes to rotate about but this does not mean there are only four principal rotations. The 2D description above ("rotate in the xy plane) is the correct one. We can rotate in every such principal plane, whatever the dimensionality. In 3D, there are the same number of these planes as there are axes.

Exercise for the reader: Why?

Exercise for the reader: How many principal rotations are there in 4D?