2.4.5 Orthonormal Bases and Coordinate Frames

Managing coordinate systems is one of the core tasks of almost any graphics program. Key to this is managing *orthonormal bases*. Any set of two 2D vectors $\mathbf{u}$ and $\mathbf{v}$ form an orthonormal basis provided they are orthogonal (at right angles) and are each of unit length. Thus,

$$\|\mathbf{u}\| = \|\mathbf{v}\| = 1,$$

and

$$\mathbf{u} \cdot \mathbf{v} = 0.$$

In 3D, three vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ form an orthonormal basis if

$$\|\mathbf{u}\| = \|\mathbf{v}\| = \|\mathbf{w}\| = 1,$$

and

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{u} = 0.$$

This orthonormal basis is *right-handed* provided

$$\mathbf{w} = \mathbf{u} \times \mathbf{v},$$

and otherwise it is left-handed.

Note that the Cartesian canonical orthonormal basis is just one of infinitely many possible orthonormal bases. What makes it special is that it and its implicit origin location are used for low-level representation within a program. Thus, the vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are never explicitly stored, and neither is the canonical origin location $\mathbf{o}$. The global model is typically stored in this canonical coordinate system, and it is thus often called the *global coordinate system*. However, if we want to use another coordinate system with origin $\mathbf{p}$ and orthonormal basis vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$, then we *do* store those vectors explicitly. Such a system is called a *frame of reference* or *coordinate frame*. For example, in a flight simulator, we might want to maintain a coordinate system with the origin at the nose of the plane, and the orthonormal basis aligned with the airplane. Simultaneously we would have the master canonical coordinate system (Figure 2.21). The coordinate system associated with a particular object, such as the plane, is usually called a *local coordinate system*.

At a low level, the local frame is stored in canonical coordinates. For example,

$$\mathbf{u} = (x_u, y_u, z_u) \equiv x_u\mathbf{x} + y_u\mathbf{y} + z_u\mathbf{z}.$$

A location implicitly includes an offset from the canonical origin:

$$\mathbf{p} = (x_p, y_p, z_p) \equiv \mathbf{o} + x_p\mathbf{x} + y_p\mathbf{y} + z_p\mathbf{z}.$$

Note that if we store a vector $\mathbf{a}$ with respect to the *uvw* frame, we store a triple (u_a, v_a, w_a) which we can interpret geometrically as:

$$(u_a, v_a, w_a) \equiv u_a\mathbf{u} + v_a\mathbf{v} + w_a\mathbf{w}.$$

To get the canonical coordinates of a vector $\mathbf{a}$ stored in the $\mathbf{u}$ $\mathbf{v}$ $\mathbf{w}$ coordinate system, simply recall that $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are themselves stored in terms of Cartesian coordinates, so the expression $u_a\mathbf{u} + v_a\mathbf{v} + w_a\mathbf{w}$ is already in Cartesian coordinates if evaluated explicitly. Using matrices to manage changes of coordinate systems is discussed in Sections 6.2.1 and 6.5.

2.4.6 Constructing a Basis from a Single Vector

Often we need an orthonormal basis that is aligned with a given vector. That is, given a vector $\mathbf{a}$, we want an orthonormal $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ such that $\mathbf{w}$ points in the same direction as $\mathbf{a}$ (Hughes & Möller, 1999), but we don't particularly care what

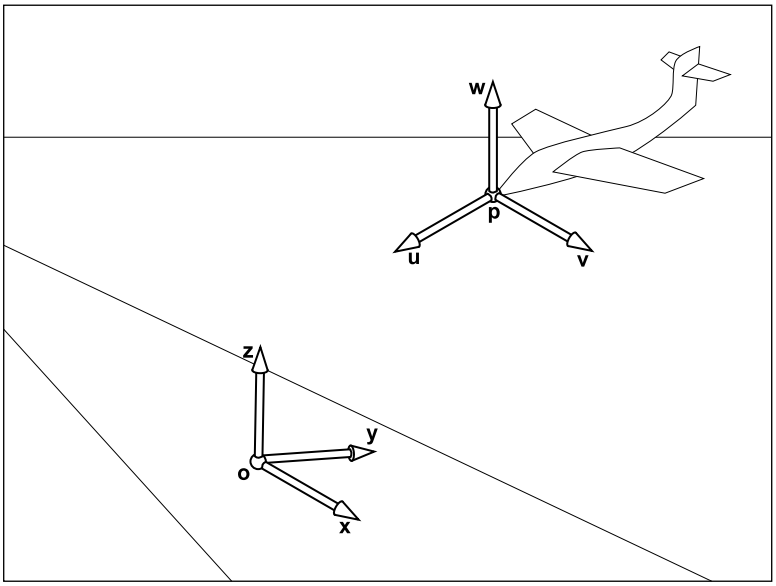

Figure 2.21. There is always a master or "canonical" coordinate system with origin **o** and orthonormal basis **x, y**, and **z**. This coordinate system is usually defined to be aligned to the global model and is thus often called the "global" or "world " coordinate system. This origin and basis vectors are never stored explicitly. All other vectors and locations are stored with coordinates that relate them to the global frame. The coordinate system associated with the plane are explicitly stored in terms of global coordinates.

$\mathbf{u}$ and $\mathbf{v}$ are. One vector isn't enough to uniquely determine the answer; we just need a robust procedure that will find any one of the possible bases.

This can be done using cross products as follows. First make $\mathbf{w}$ a unit vector in the direction of $\mathbf{a}$:

$$\mathbf{w} = \frac{\mathbf{a}}{\|\mathbf{a}\|}.$$

This same procedure can, of course, be used to construct the three vectors in any order; just pay attention to the order of the cross products to ensure the basis is right handed.

Then choose any vector $\mathbf{t}$ not collinear with $\mathbf{w}$ and use the cross product to build a unit vector $\mathbf{u}$ perpendicular to $\mathbf{w}$:

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}.$$

If $\mathbf{t}$ is collinear with $\mathbf{w}$ the denominator will vanish, and if they are nearly collinear the results will have low precision. A simple procedure to find a vector sufficiently different from $\mathbf{w}$ is to start with $\mathbf{t}$ equal to $\mathbf{w}$ and change the smallest magnitude component of $\mathbf{t}$ to 1. For example, if $\mathbf{w} = (1/\sqrt{2}, -1/\sqrt{2}, 0)$ then $\mathbf{t} = (1/\sqrt{2}, -1/\sqrt{2}, 1)$. Once $\mathbf{w}$ and $\mathbf{u}$ are in hand, completing the basis is simple:

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}.$$

An example of a situation where this construction is used is surface shading, where a basis aligned to the surface normal is needed but the rotation around the normal is not important.

2.4.7 Constructing a Basis from Two Vectors

The procedure in the previous section can also be used in situations where the rotation of the basis around the given vector is important. A common example is building a basis for a camera: it's important to have one vector aligned in the direction the camera is looking, but the orientation of the camera around that vector is *not* arbitrary, and it needs to be specified somehow. Once the orientation is pinned down, the basis is completely determined.

A common way to fully specify a frame is by providing two vectors $\mathbf{a}$ (which specifies $\mathbf{w}$) and $\mathbf{b}$ (which specifies $\mathbf{v}$). If the two vectors are known to be perpendicular it is a simple matter to construct the third vector by $\mathbf{u} = \mathbf{b} \times \mathbf{a}$. To be sure that the resulting basis really is orthonormal, even if the input vectors weren't

$\boldsymbol{u} = \boldsymbol{a} \times \boldsymbol{b}$ also produces an orthonormal basis, but it is left handed.

quite, a procedure much like the single-vector procedure is advisable:

$$\begin{aligned} \mathbf{w} &= \frac{\mathbf{a}}{\|\mathbf{a}\|} \\ \mathbf{u} &= \frac{\mathbf{b} \times \mathbf{w}}{\|\mathbf{b} \times \mathbf{w}\|} \\ \mathbf{v} &= \mathbf{w} \times \mathbf{u} \end{aligned} \tag{2.9}$$

In fact, this procedure works just fine when $\mathbf{a}$ and $\mathbf{b}$ are not perpendicular. In this case, $\mathbf{w}$ will be constructed exactly in the direction of $\mathbf{a}$, and $\mathbf{v}$ is chosen to be the closest vector to $\mathbf{b}$ among all vectors perpendicular to $\mathbf{w}$.

If you want me to set $\mathbf{w}$ and $\mathbf{v}$ to two non-perpendicular directions, something has to give—with this scheme I'll set everything the way you want, except I'll make the smallest change to $\mathbf{v}$ that so that it is in fact perpendicular to $\mathbf{w}$.

What will go wrong if ***a*** and ***b*** are parallel?

This procedure *won't* work if $\mathbf{a}$ and $\mathbf{b}$ are collinear. In this case $\mathbf{b}$ is of no help in choosing which of the directions perpendicular to $\mathbf{a}$ we should use: it is perpendicular to all of them.

A common situation in which a basis is constructed from two vectors is in specifying camera positions (Section 4.3). We want to construct a frame that has $\mathbf{w}$ parallel to the direction the camera is looking, and $\mathbf{v}$ should point out the top of the camera. To orient the camera upright, we build the basis around the view direction, using the straight-up direction as the reference vector to establish the camera's orientation around the view direction. Setting $\mathbf{v}$ as close as possible to straight up exactly matches the intuitive notion of "holding the camera straight."

2.4.8 Squaring up a Basis

Occasionally you may find problems caused in your computations by a basis that is supposed to be orthonormal but error has crept in—due to rounding error in computation, or to the basis having been stored in a file with low precision, for instance.

The procedure of the previous section can be used for this purpose; simply constructing the basis anew using the existing $\mathbf{w}$ and $\mathbf{v}$ vectors will produce a new basis that is orthonormal and is close to the old one.

This approach is good for many applications but is not the best available. It does produce accurately orthogonal vectors, and for nearly orthogonal starting bases the result will not stray far from the starting point. However, it is asymmetric: it "favors" $\mathbf{w}$ over $\mathbf{v}$ and $\mathbf{v}$ over $\mathbf{u}$ (whose starting value is thrown away). It chooses a basis close to the starting basis but has no guarantee of choosing *the* closest orthonormal basis. When this is not good enough, the SVD (Section 5.4.1) can be used to compute an orthonormal basis that *is* guaranteed to be closest to the original basis.