5

ORIENTATION

The quintessential setup in three-dimensional graphics is a scene composed of objects with a camera positioned somewhere looking in a particular direction. As a simple example, imagine a single cube sitting on the ground. This is our scene and we look at it from some point in space. We imagine either our eye or perhaps a camera at this point oriented in such a way that it is looking at the scene and probably focused on the cube in particular. There is much that we have to keep track of here. The cube has a given shape, it is positioned on the ground at a particular location with a particular orientation, the camera is centered at a given point, it is looking at some point in the scene, and it is positioned so the up direction is aligned in a way the user or programmer prefers. Our task now is to determine how to specify all of these orientations and how to keep them appropriately aligned with each other as we move the camera or move objects in the scene.

Start with the cube in our example, and note that most of the vector geometry developed so far was focused on determining the vertices for various objects like the cube. We actually want coordinates for these vertices, so we specified a Cartesian coordinate system which we now call the *local* coordinate system. It is local for the object at hand and usually we find it convenient to place the origin at the center of the object (assuming this center is convenient to find.) When we place the cube on the ground, we are really constructing yet another coordinate system called the *world* coordinate system where the ground is just a particular plane with a description in the world system and the center of the cube has a set of world coordinates (c_x, c_y, c_z) as well as a set of local coordinates $(0, 0, 0)$. The camera, too, has a set of world

Mathematical Structures for Computer Graphics, First Edition. Steven J. Janke.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

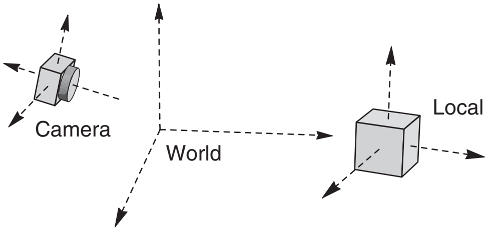


Figure 5.1 Coordinate systems in a scene

coordinates indicating where it is positioned in our scene, but in addition it has its own orientation. What we display on the computer screen is relative to the camera's orientation, so it is once again convenient to select another coordinate system called the *camera* coordinate system. (It is equally reasonable to call this last system either the *eye* coordinate system or the *view* coordinate system.) The origin of the camera coordinate system is at the center of the camera (Figure 5.1).

Using a local coordinate system, it is easy to find the vertices of our cube because we are free to first center it at the origin and orient it so that faces are parallel to the coordinate planes. We could take one vertex to be $(1, 1, 1)$ and then another is just $(-1, -1, -1)$, the reflection in the origin. In fact, all eight vertices have coordinates that are either 1 or -1. This is an easy representation to begin with in the local coordinate system and, if we need, we can apply rotation or scaling transformations to get the cube to any given size and orientation.

Positioning the cube in the scene requires finding world coordinates for the vertices, but this could be as simple as applying a translation to move the center of the cube to the required location. It could also be more complicated requiring some scaling and rotation to properly place the cube relative to other objects in the scene. Placing the camera is similar to placing objects, but once it is placed we need to point it towards the scene. This means we are picking the camera coordinate system appropriately, and consequently we are specifying the three coordinate axes for a Cartesian coordinate system. One axis should lie along a line from the camera to the center of the scene, one should point up, and the third should be perpendicular to the other two in such a way that we have a right-handed coordinate system.

Clearly, the orientation involved in viewing a scene depends on understanding the construction of coordinate systems (Cartesian systems in particular) and the transformation of one to another. It might seem that having one global coordinate system would simplify things, but as we saw previously, a local coordinate system, for example, can make designing an object much easier. The flexibility of several coordinate systems usually outweighs the simplicity of a single system.

5.1 CARTESIAN COORDINATE SYSTEMS

Throughout the development of vector geometry, whenever we give the coordinates of a vector such as $\vec{v} = (x, y, z)$, what we really mean is that $\vec{v} = x\vec{i} + y\vec{j} + z\vec{k}$. We say that $\vec{v}$ is a *linear combination* of the common vectors $\vec{i}, \vec{j}$, and $\vec{k}$. Since we selected the common vectors so that they did not lie in a single plane, every three-dimensional vector is a linear combination of these three common vectors, so we say they form a *basis*. Moreover, we actually selected the basis vectors so that they all had unit length and were perpendicular to each other. That is, their dot products with each other are zero and $\vec{i} \cdot \vec{i} = \vec{j} \cdot \vec{j} = \vec{k} \cdot \vec{k} = 1$. We say they form an *orthonormal* basis. One key feature of an orthonormal basis is the ease with which we can calculate dot products. For two vectors $\vec{w}_1 = x_1\vec{i} + y_1\vec{j} + z_1\vec{k}$ and $\vec{w}_2 = x_2\vec{i} + y_2\vec{j} + z_2\vec{k}$, the dot product is $\vec{w}_1 \cdot \vec{w}_2 = x_1x_2 + y_1y_2 + z_1z_2$. This is how we defined the dot product early on, but it really is a result of using an orthonormal basis.

A basis allows us to specify arbitrary vectors nicely, but we need an origin as well in order to specify points. An origin O is a point that we intuitively think of as the center of what we call a *coordinate system*. For an arbitrary point P in space, we first write P as a sum, $P = O + (P - O)$. This is the sum of a point and a vector, and earlier we understood that this type of sum gives us a point. The coordinates of P in the coordinate system are just the coordinates of the vector $(P - O)$ using the given basis.

Definition 5.1 (Coordinate System). *A coordinate system in three (or two) dimensions is a set of three (or two) basis vectors along with a designated point called the origin. If the basis vectors are perpendicular to each other and have unit length, the basis and the coordinate system are called orthonormal.*

In our graphics scenario with a cube sitting on the ground, we have three coordinate systems: the world coordinate system ($\mathscr{W}$), the local coordinate system ($\mathscr{L}$), and the camera coordinate system ($\mathscr{C}$). Their corresponding origins are O_w, O_l, and O_c. Since we imagine the world coordinate system as a global system, we will let our usual vectors $\{\vec{i}, \vec{j}, \vec{k}\}$ to be an orthonormal basis for $\mathscr{W}$. Then let $\{\vec{u}, \vec{v}, \vec{w}\}$ be an orthonormal basis for $\mathscr{L}$. We will hold off on specifying $\mathscr{C}$ until later because we want to let the user have some input into how the camera is oriented. Now consider one of the cube vertices which we have represented in local coordinates [e.g., $(1, 1, 1)$]. We would like to find the corresponding world coordinates for this vertex, and, more generally, what we need is an algorithm for moving from one coordinate system to another.

Suppose we have the coordinates of point P with respect to coordinate system $\mathscr{S}_1$ and we want the coordinates of P with respect to the system $\mathscr{S}_2$. We might guess that the change from one set of coordinates to the other requires an affine transformation because we once again want to preserve lines in the process. In fact, there are two parts to the transformation. First, since the origins of $\mathscr{S}_1$ and $\mathscr{S}_2$ may be different, a translation of the coordinates will be necessary. Second, we need to account for different sets of basis vectors in the two systems. It is possible that a rotation will account for the difference, but it is also possible that we may need some more complicated combination of scaling, shear, and rotation to account for the different orientation of basis vectors.

The first part of the transformation involving translation is relatively easy to describe. In $\mathscr{S}_1$, we represent the point P as $P = O_1 + (P - O_1) = O_1 + \vec{v}_1$. The $\mathscr{S}_1$ coordinates of P are the coordinates of $\vec{v}_1$. To move the origin from O_1 to O_2, we need to add the vector $\vec{t} = (O_2 - O_1)$. Rewriting the expression for P shows how to find the $\mathcal{S}_2$ coordinates for P.

vector de cambio

$$P = O_1 + (P - O_1) = O_1 + \vec{v}_1 = O_2 \underline{- \vec{t} + \vec{v}_1} = O_2 \underline{+ \vec{v}_2}$$

The $\mathcal{S}_2$ coordinates are the coordinates of $\vec{v}_2$, which are the coordinates of $\vec{v}_1$ minus the shift vector $\vec{t}$. Intuitively, this all makes sense because coordinates are relative to the origin; they should change by the amount of any translation. When moving from $\mathcal{S}_1$ to $\mathcal{S}_2$, we subtract the vector $\vec{t}$.

We still need to account for a different set of basis vectors in $\mathcal{S}_2$, but as a first guess at how to quantify this difference, it seems that some linear transformation will do the trick. Once we have applied the translation, then both origins coincide so the problem is reduced to applying a linear transformation to align the axes. We can express the total transformation from one set of coordinates to another in terms of matrix multiplication. Let the 4×4 matrix M_T (with homogeneous coordinates) represent the translation, and let the matrix M_B represent the linear transformation necessary to adjust for a different basis. Then multiplication by $M_B M_T$ transforms $\mathcal{S}_1$ coordinates into $\mathcal{S}_2$ coordinates.

Example 5.1 (A Simple Coordinate System Change). Take $\mathcal{S}_1$ to be a system with orthonormal basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. Let P have coordinates $(2, -1, 1)$ in this system. Now suppose that $\mathcal{S}_2$ is a second coordinate system with an origin that has $\mathcal{S}_1$ coordinates $(4, -2, 5)$; that is, its origin is displaced relative to the first coordinate system.

Then, vector $(O_2 - O_1) = (4, -2, 5)$ and we need to subtract this shift.

$t = -\ (1,0,0)$

$$M_T = \begin{bmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$M_T P_{S_1} = \underline{P_{S_2}}$

If coordinate system $\mathcal{S}_2$ has the same basis vectors as $\mathcal{S}_1$, then all we have to do in order to change from the first coordinate system to the second is to multiply by M_T. (M_B is then the identity.) Expressing P in homogeneous coordinates gives $(2, -1, 1, 1)$, and multiplying by M_T results in $(-2, 1, -4, 1)$. The $\mathcal{S}_2$ coordinates for P would be $(-2, 1, -4)$.

If coordinate system $\mathcal{S}_2$ does not have the same basis vectors as $\mathcal{S}_1$, we need another transformation. Suppose that system $\mathcal{S}_2$ has basis vectors $\{\vec{i}, \vec{k}, -\vec{j}\}$, which means that this system is just the $\mathcal{S}_1$ system rotated $\pi/2$ radians counterclockwise around the x-axis. Considering the effect on coordinates, rotating the basis vectors counterclockwise is equivalent to rotating the vector $(P - O_2)$ clockwise. We choose M_B to rotate clockwise around the x-axis (Figure 5.2).

$M^{-1} = M^T$

$$M_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$(1, 0, 0)$

$(0, 0, 1)$

$(0, -1, 0)$

$M = \begin{bmatrix} | & | & | \\ u & v & w \\ | & | & | \end{bmatrix}$ S_2 basis

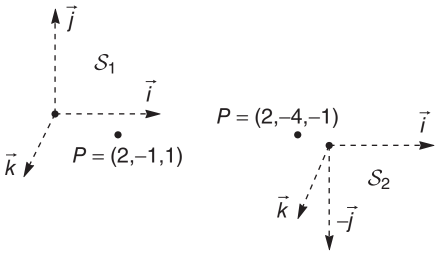


Figure 5.2 Changing coordinate systems

The complete coordinate change combines both the translation and the rotation. The order of the two is, of course, important. The translation vector was given with respect to the $\mathcal{S}_1$ system, so we apply it first to align the origins. Then we apply the rotation to align the axes. (If we apply a rotation first, then we have to adjust how we represent the translation. This can be done, but it is easier and more intuitive to apply the translation first.)

$M_B M_T P_{S_1} = P_{S_2}$

$$M_B M_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -4 \\ 0 & 0 & 1 & -5 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying this matrix times the vector with homogeneous coordinates $(2, -1, 1, 1)$ gives $(-2, -4, -1, 1)$. The point P has $\mathcal{S}_1$ coordinates $(2, -1, 1)$ and $\mathcal{S}_2$ coordinates $(-2, -4, -1)$. □

The M_B transformation matrix accounts for the differences in the basis vectors for the two systems. The simple example of a rotation around one axis is relatively easy to deal with, but we really need an approach that deals with more complicated situations. For example, if the systems are not orthonormal, then we have to adjust both for the length of the basis vectors and for the angle between them.

Assume that we have made an appropriate translation so that the origin of $\mathcal{S}_1$ coincides with the origin of $\mathcal{S}_2$. Imagine we have a vector $\vec{c}$ that we wish to express in both $\mathcal{S}_1$ coordinates and $\mathcal{S}_2$ coordinates. Let the basis vectors for system $\mathcal{S}_1$ be $\{\vec{q}, \vec{r}, \vec{s}\}$, and let the basis of $\mathcal{S}_2$ be $\{\vec{u}, \vec{v}, \vec{w}\}$. These are general bases, so although they are sets of independent vectors, they may not be orthonormal.

If in $\mathcal{S}_1$ the coordinates are $\vec{c} = (a_1, b_1, c_1)$, then this means that

$$\vec{c} = a_1\vec{q} + b_1\vec{r} + c_1\vec{s} \tag{5.1}$$

In the $\mathcal{S}_2$ system, let the coordinates be $\vec{c} = (a_2, b_2, c_2)$, meaning that

$$\vec{c} = a_2\vec{u} + b_2\vec{v} + c_2\vec{w} \tag{5.2}$$

Now suppose we have $\mathcal{S}_1$ coordinates for each of the basis vectors in the $\mathcal{S}_2$ system. That is, $\vec{u} = (u_q, u_r, u_s), \vec{v} = (v_q, v_r, v_s), \vec{w} = (w_q, w_r, w_s)$. Using these coordinates, we get equations for $\vec{u}, \vec{v}$, and $\vec{w}$ similar to Equation 5.1. For example, $\vec{u} = u_q\vec{q} + u_r\vec{r} + u_s\vec{s}$. Substituting these equations into Equation 5.2 and combining terms gives the following matrix equation:

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} u_q & v_q & w_q \\ u_r & v_r & w_r \\ u_s & v_s & w_s \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} = \underline{M} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{5.3}$$

Here, M is a 3×3 matrix because temporarily we do not need to consider translation and therefore we do not need the larger 4×4 matrix operating on homogeneous coordinates. Multiplication by the matrix M converts $\mathcal{S}_2$ coordinates to $\mathcal{S}_1$ coordinates. Of course, if M has an inverse, then multiplication by M^{-1} converts $\mathcal{S}_1$ coordinates to $\mathcal{S}_2$ coordinates.

A closer look at the matrix M reveals that the columns are just the $\mathcal{S}_1$ coordinates for each of the basis vectors in $\mathcal{S}_2$. Since these columns come from a basis, they are independent and M has a nonzero determinant. This ensures that M has an inverse and consequently M_B is a 4×4 matrix with M^{-1} in the upper left-hand corner. We have found the matrix for the second part of the general coordinate transformation.

Result 5.1 (Coordinate Transformation). *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two coordinate systems with origins O_1 and O_2 and basis vectors $\{\vec{q}, \vec{r}, \vec{s}\}$ and $\{\vec{u}, \vec{v}, \vec{w}\}$, respectively. Suppose the $\mathcal{S}_1$ coordinates of the vector $\underline{\vec{t} = (O_2 - O_1)}$ are (t_q, t_r, t_s) and that the $\mathcal{S}_1$ coordinates for the basis vectors of $\mathcal{S}_2$ are $u = (u_q, u_r, u_s), v = (v_q, v_r, v_s)$, and $w = (w_q, w_r, w_s)$. Form the matrix M (defined in Equation 5.3) by using these $\mathcal{S}_1$ coordinates as columns.*

The coordinate transformation from $\mathcal{S}_1$ to $\mathcal{S}_2$ is an affine transformation represented by the matrix $M_{\mathcal{S}_1 \to \mathcal{S}_2}$, where

$$M_{\mathcal{S}_1 \to \mathcal{S}_2} = \underline{M_B}\,\underline{M_T} = \left[\begin{array}{ccc|c} & \boxed{M^{-1}} & & 0 \\ & & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left.\underbrace{\begin{bmatrix} 1 & 0 & 0 & -t_q \\ 0 & 1 & 0 & -t_r \\ 0 & 0 & 1 & -t_s \\ 0 & 0 & 0 & 1 \end{bmatrix}}\right\} t$$

For the other direction from $\mathcal{S}_2$ to $\mathcal{S}_1$, the transformation matrix $M_{\mathcal{S}_2 \to \mathcal{S}_1}$ is the inverse of $M_{\mathcal{S}_1 \to \mathcal{S}_2}$.

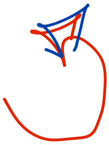
If we are lucky enough to have orthonormal bases, then the inverse of M is actually the transpose of M, $M^{-1} = M^T$. In this case, the matrix $M_{\mathcal{S}_1 \to \mathcal{S}_2}$ has a convenient form:

$$M_{\mathcal{S}_1 \to \mathcal{S}_2} = \begin{bmatrix} u_q & u_r & u_s & -\vec{u} \cdot \vec{t} \\ v_q & v_r & v_s & -\vec{v} \cdot \vec{t} \\ w_q & w_r & w_s & -\vec{w} \cdot \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

In the last column, we have dot products of basis vectors with the translation vector $\vec{t}$.

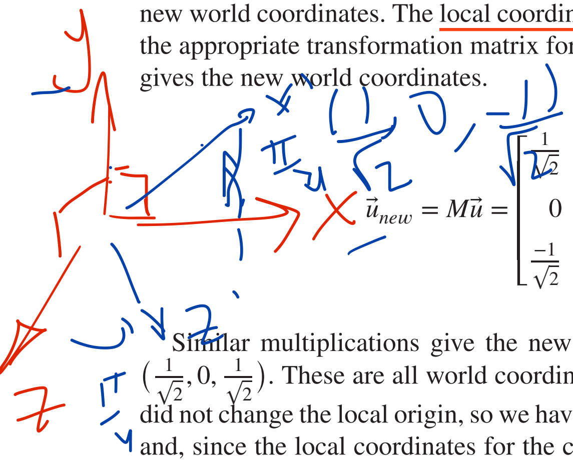
Example 5.2 (Local and World Coordinates). Return to the scenario where the graphics scene has a cube sitting on the ground in some orientation. We have both a world coordinate system, $\mathscr{W}$, and a local coordinate system, $\mathscr{L}$. We use the standard vectors $\{\vec{i}, \vec{j}, \vec{k}\}$ as an orthonormal basis for $\mathscr{W}$. Set the basis for $\mathscr{L}$ as $\{\vec{u}, \vec{v}, \vec{w}\}$, where temporarily $\vec{u} = \vec{i}$, $\vec{v} = \vec{j}$, and $\vec{w} = \vec{k}$. The bases of $\mathscr{L}$ and $\mathscr{W}$ are aligned. Recall that we set the cube vertices to have local coordinates $(1,1,1), (1,-1,1), (1,1,-1)$, and so on.

Since the coordinate systems use vectors $\{\vec{i}, \vec{j}, \vec{k}\}$, we will refer to the corresponding axes as x, y, and z. The origin O_w for the world system has, of course, world coordinates $(0,0,0)$, and the ground plane in our scene will be the xz plane leaving the y-axis pointing up. To position the cube so that it is sitting on the ground somewhere off-center in the scene, we set the world coordinates of the local system origin, O_l, to $(5,1,8)$. (The y coordinate equal to one moves the local system up a bit from the world system and pushes the cube up so that it is sitting on the ground plane.)



Rather than keeping the cube oriented with sides parallel to the coordinate axes, we may want it rotated around, say, the y-axis. We have two choices. We can rotate the cube vertices with respect to the local coordinate system, or we can rotate the local coordinate basis vectors with respect to world coordinates (keeping the local origin fixed). Rotating the basis vectors effectively rotates the cube as well, because we keep the same local coordinates for the cube vertices as before; the vertex $(1,1,1)$, for example, will be repositioned relative to the world coordinate system. The choice of approaches again depends on the context of the graphics application, but it may be useful to keep the cube aligned with the local axes so that rotating the entire local coordinate system seems like a decent choice.

Suppose we rotate the local coordinate system $\pi/4$ radians counterclockwise around the y-axis. Then vector $\vec{u}$ which was originally the same as $\vec{i} = (1,0,0)$ gets new world coordinates. The local coordinates of $\vec{u}$ are still $(1,0,0)$. We already know the appropriate transformation matrix for a rotation around the y-axis, so applying it gives the new world coordinates.



$$\vec{u}_{new} = M\vec{u} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$M = \begin{bmatrix} | & | & | \\ u & v & w \\ | & | & | \end{bmatrix}$$

S1

Similar multiplications give the new basis vectors $\vec{v}_{new} = (0,1,0)$ and $\vec{w}_{new} = (\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})$. These are all world coordinates for the new local coordinate basis. We did not change the local origin, so we have simply rotated the local coordinate system and, since the local coordinates for the cube vertices stayed the same, the cube was rotated as well.

The original task was to convert local coordinates to world coordinates. In the notation of Result 5.1, we are set to use the world system as $\mathcal{S}_1$ and the local system as $\mathcal{S}_2$. First, construct $M_{\mathcal{S}_1 \to \mathcal{S}_2}$, and then find the inverse to get $M_{\mathcal{S}_2 \to \mathcal{S}_1}$. We have $\mathcal{S}_1$ (world) coordinates for $\vec{t} = (O_l - O_w) = (5, 1, 8)$ and for the basis vectors of $\mathcal{S}_2$ (local coordinate system). The columns of M are the basis vectors of $\mathcal{S}_2$ and, because the systems are orthonormal, $M^{-1} = M^T$.

$$M_{(\mathcal{W} \to \mathcal{L})} = M_B M_T = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & \frac{3}{\sqrt{2}} \\ 0 & 1 & 0 & -1 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & \frac{-13}{\sqrt{2}} \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

The inverse matrix converts local coordinates to world coordinates:

$$M_{(\mathcal{L} \to \mathcal{W})} = (M_B M_T)^{-1} = M_T^{-1} M_B^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 5 \\ 0 & 1 & 0 & 1 \\ \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying $M_{(\mathcal{L} \to \mathcal{W})}$ times the local coordinates $(1, 1, 1)$ [homogeneous coordinates are $(1, 1, 1, 1)$] gives the world coordinates for one of the cube's vertices, $(5, 2, 9.41)$ (Figure 5.3). □

This example is somewhat simple in that rotation was around a coordinate axis. Later we will use the conversion technique to move to camera coordinates where the orientation of the basis vectors is not readily seen as a rotation.

$\boxed{M_B} \underline{M_T}$

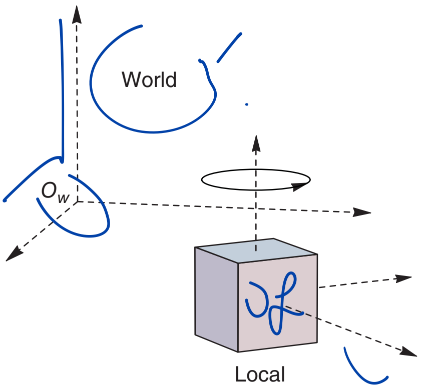


Figure 5.3 $\mathscr{L}$ and $\mathscr{W}$

5.2 CAMERAS

Usually, the selection of the world ($\mathscr{W}$) and local ($\mathscr{L}$) coordinate systems is a simplifying convenience. The graphics designer or graphics programmer picks the world system from a global perspective, and often the center of the world is the origin. The ground is possibly the xz plane formed by two basis vectors with the y-axis pointing up; in this case, the first and third basis vectors determine the ground plane, and the second basis vector is normal to the plane. Various attributes of the scene (mountain ranges, roads, buildings, etc.) may make one orientation of the axes more intuitive than another. For the local system, there may only be one object described by the system and therefore the symmetry of the object calls the shots in determining axis orientation. If there is more than one object, then maybe a subsystem inside the local coordinate system is an appropriate design choice.

When it comes to the camera coordinate system ($\mathscr{C}$), we need to know where we are looking and which way is up. It also is important to decide whether the system is a right-handed or left-handed coordinate system. Most graphics systems rely on right-handed systems, although in particular instances a left-handed system may be useful.

Suppose then that we first pick the camera's center position, P_c, and decide where in the scene we are looking, say at the point P_s. The vector $\vec{n} = P_c - P_s$ is normal to a plane (called the *view plane*) which we imagine holds the window into our scene. We see the scene through this window and our perspective transformation will map the scene onto this window. Just as a convention, the view plane normal $\vec{n}$ points in the positive z direction and we are looking in the negative z direction (Figure 5.4).

Let the vector $\vec{v}$ point in the up direction. This vector may be user-supplied and, although we know it should be perpendicular to $\vec{n}$, it may not be described that precisely, so we intend to adjust $\vec{v}$ if necessary. To set up an orthonormal basis for the

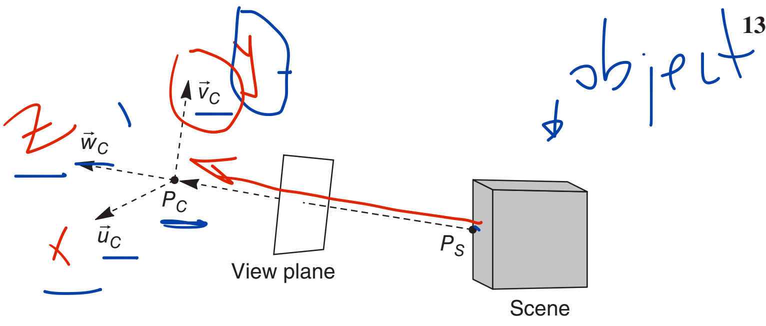


Figure 5.4 Camera orientation

camera system $\mathscr{C}$, we define the following basis vectors:

$$
\begin{aligned}
\vec{w}_c &= \frac{\vec{n}}{|\vec{n}|} \\
\vec{u}_c &= \frac{\vec{v} \times \vec{w}_c}{|\vec{v} \times \vec{w}_c|} \qquad (5.5) \\
\vec{v}_c &= \underline{\vec{w}_c \times \vec{u}_c} \quad \checkmark
\end{aligned}
$$

This set of vectors $\{\vec{u}_c, \vec{v}_c, \vec{w}_c\}$ forms an orthonormal basis for the coordinate system $\mathscr{C}$ and the point P_c is the designated origin. It is a right-handed coordinate system with $\vec{u}_c$ analogous to the x direction, $\vec{v}_c$ (the up vector) analogous to the y direction, and $\vec{w}_c$ (pointing at the camera) analogous to the z direction.

We started with the normal $\vec{n}$ and the up vector $\vec{v}$. Assuming the coordinates of these vectors are given in world coordinates, we have the world coordinates for the basis vectors in $\mathscr{C}$. To view the scene from the camera position, we need to convert the world coordinates for points in the scene to camera coordinates. Using the notation $\vec{u}_c = (u_x, u_y, u_z)$ for the world coordinates and letting $\vec{t} = (P_c - O_w)$, the appropriate matrix that performs the coordinate transformation is

M^T

$$
M_{\mathscr{W} \to \mathscr{C}} = \begin{bmatrix} u_x & u_y & u_z & -\vec{u} \cdot \vec{t} \\ v_x & v_y & v_z & -\vec{v} \cdot \vec{t} \\ w_x & w_y & w_z & -\vec{w} \cdot \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.6)
$$

$M^* = [u \; v \; w]$

$M^{-1} = M^T$

S1

Example 5.3 (Converting to Camera Coordinates). In Example 5.2, the cube vertex (1, 1, 1) in local coordinates became (5, 2, 9.41) in world coordinates. Now, center the camera at the point (10, 12, 18) and suppose we are looking at the origin of the world system. The view plane normal is $\vec{n} = (10, 12, 18)$. If we select an up vector of

(0, 0, 0) $\qquad \hat{n} = P_c - P_s$ ✓

$\vec{v} = (1, 1, 0)$, then

$$\vec{w}_c = \frac{(10, 12, 18)}{\sqrt{10^2 + 12^2 + 18^2}} \approx (0.42, 0.50, 0.76)$$

$$\vec{u}_c = \frac{\vec{v} \times \vec{w}_c}{|\vec{v} \times \vec{w}_c|} \approx (0.71, -.71, 0.08)$$

$t = P_c - O_w$

$$\vec{v}_c = \vec{w}_c \times \vec{u}_c \approx (0.57, 0.50, -0.65)$$

To transform (1, 1, 1) to camera coordinates, we multiply by the transformation matrix using $\vec{t} = (10, 12, 18)$.

$$M_{\mathscr{W} \to \mathscr{C}} \begin{bmatrix} 5 \\ 2 \\ 9.41 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.71 & -0.71 & 0.08 & -0.02 \\ 0.57 & 0.50 & -0.65 & 0 \\ 0.42 & 0.50 & 0.76 & -23.88 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \\ 9.41 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.95 \\ -2.42 \\ -13.63 \\ 1 \end{bmatrix}$$

The z coordinate of our vertex $(2.86, -2.27, -13.63)$ is negative because the camera position is far along the world system's positive z-axis. Also, the camera is relatively high with respect to the world system's origin, so the y coordinate of the vertex is also negative. (Note: The matrix M was calculated using the approximate values for the basis vectors in $\mathscr{C}$. The value -0.02 in the matrix should theoretically be 0. See Exercises for more detail.) □

5.2.1 Moving the Camera or Objects

Now that we have analyzed the mechanics of coordinate transformations, we can combine the transformation matrix with the standard transformations we studied earlier for changing the position or shape of an object, perhaps a cube. The task is to start with a view of the scene from the camera-oriented position and to transform the cube in some way. Suppose, first, that the cube is centered at the origin and we would like to rotate it around an axis through its center. Multiplication by a rotation matrix will alter the cube's original vertex coordinates appropriately. The following transformation matrix A will accomplish the conversion of local coordinates to rotated coordinates and then to world coordinates.

$$A = M_{\mathscr{L} \to \mathscr{W}} R$$

As we saw in Example 5.2, an alternate way of thinking about this transformation is as a rotation of the local coordinate system itself rather than just the vertices of the cube. That is, we rotate the basis vectors into new positions. Call the new rotated coordinate system $\mathscr{L}^*$. Now we interpret the matrix R as transforming $\mathscr{L}^*$ coordinates into $\mathscr{L}$ coordinates. So the matrix A is really the matrix converting $\mathscr{L}^*$ coordinates to $\mathscr{W}$ coordinates.

$$A = M_{\mathscr{L}^* \to \mathscr{W}} = M_{\mathscr{L} \to \mathscr{W}} \cdot R$$

As an example, take the R matrix to be a rotation around the z-axis.

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

Referring to our general coordinate transformation, let $\mathscr{S}_1 = \mathscr{L}$ and $\mathscr{S}_2 = \mathscr{L}^*$. Then R is the matrix M because the columns of R are just the rotated coordinates of the basis vectors $\{(1,0,0),(0,1,0),(0,0,1)\}$. The first column of R is the vector $(\cos\theta, \sin\theta, 0)$, which is just the rotation of vector $(1,0,0)$. These are coordinates in the local coordinate system $\mathscr{L}$. So R is indeed a coordinate transformation matrix which converts $\mathscr{L}^*$ coordinates to $\mathscr{L}$ coordinates. Multiplying by $M_{\mathscr{L} \to \mathscr{W}}$ then converts the $\mathscr{L}$ coordinates to $\mathscr{W}$ coordinates.

Focus on the cube vertex $(1,1,1)$. These are local coordinates (system $\mathscr{L}$). Multiplying by R either changes the local coordinates to $(\cos\theta - \sin\theta, \sin\theta + \cos\theta, 1)$ or it changes the basis vectors of $\mathscr{L}$ giving a new system $\mathscr{L}^*$. In the $\mathscr{L}^*$ system, the vertex of the cube has coordinates $(1,1,1)$. The advantage of the second interpretation is that the coordinates of the cube vertices stay unchanged.

Transformations other than rotations work just as well here and the lesson is that transforming coordinates can be thought of as transforming coordinate systems. Yet, coordinate systems are all relative to each other, and usually there is some system, often the world coordinate system $\mathscr{W}$, that is absolute and unchanging. It is important to realize that we transformed the cube by thinking of it locally; we rotated around an axis through the center of the cube. Applying the rotation first accomplished this. If we wanted to rotate with respect to the world coordinate system and swing the cube around the world origin, we multiply by R after converting to world coordinates. In this case, it makes most sense to think of R as rotating coordinates because the world coordinate system is most likely fixed.

We view the scene from the camera position, so any movement or repositioning in the scene must ultimately take into account the camera coordinate system. The following five orientation tasks offer an overview of the order of operations in developing appropriate coordinate transformations:

1. *Object Rotation*. To rotate around an axis through the object's center, the rotation matrix R is applied to local coordinates first and then the sequence of coordinate transformations ends with the camera coordinate system.

$$M_{\mathscr{L} \to \mathscr{C}} = M_{\mathscr{W} \to \mathscr{C}} \cdot M_{\mathscr{L} \to \mathscr{W}} \cdot R$$

This matrix is applied to local coordinates of the cube vertices to generate the resulting camera coordinates showing the cube in a rotated position. To rotate around the world origin, we move the rotation matrix to act on world coordinates.

$$M_{\mathscr{L}\to\mathscr{C}} = M_{\mathscr{W}\to\mathscr{C}} \cdot R \cdot M_{\mathscr{L}\to\mathscr{W}}$$

2. *Eye Position.* The viewer's eye sees the scene through the camera and, if the viewer pivots his or her head left or right, the camera pivots left or right. When we have camera coordinates, we apply a rotation matrix. In this case, the rotation is around the up vector, which is the vector $\vec{v}_c$ in our notation.

$$M_{\mathscr{L}\to\mathscr{C}} = R \cdot M_{\mathscr{W}\to\mathscr{C}} . M_{\mathscr{L}\to\mathscr{W}}$$

   Again, we imagine applying this matrix to local coordinates for an object.

3. *User Control.* As the viewer watches the scene, he or she may wish to move to the right or left via some sort of input (perhaps the mouse) as in a computer game. The resulting translation is a multiplication by a translation matrix T_n once we have camera coordinates.

$$M_{\mathscr{L}\to\mathscr{C}} = T_n \cdot M_{\mathscr{W}\to\mathscr{C}} \cdot M_{\mathscr{L}\to\mathscr{W}}$$

4. *Auxiliary Coordinate System.* We may want objects in our scene to move with respect to some temporary point. For example, to simulate the solar system with moons revolving around planets which revolve around the sun, we may want the position of Jupiter to perturb the orbit of some other body. To implement this type of movement, we need an auxiliary coordinate system $\mathscr{A}$ centered on Jupiter and we then convert to $\mathscr{A}$, transform perhaps through rotation, convert back to local coordinates, and continue the conversion to camera coordinates.

$$M_{\mathscr{L}\to\mathscr{C}} = M_{\mathscr{W}\to\mathscr{C}} \cdot M_{\mathscr{L}\to\mathscr{W}}\, M_{\mathscr{A}\to\mathscr{L}} \cdot R \cdot M_{\mathscr{L}\to\mathscr{A}}$$

   Recall here that $M_{\mathscr{A}\to\mathscr{L}} = M^{-1}_{\mathscr{L}\to\mathscr{A}}$.

5. *Hierarchical Control*. Each of two cubes may have their own coordinate systems, but they may then be placed into a group coordinate system. This allows positioning of the cubes relative to each other and allows rotation of either cube around its center. By building scenes in this hierarchical way, quick changes can be made by transforming the coordinate systems.

In each of these listed cases, the sequence of transformation matrices can all be multiplied together, producing one matrix for the overall transformation. Graphics systems often use a stack data structure to keep track of the various matrices.

5.2.2 Euler Angles

Rotations and translations serve mostly to orient objects in a scene. Specifying translations is fairly straightforward, requiring the coordinates of a single vector showing the displacement, but specifying the rotations can be more of a problem.

The three-dimensional rotations we developed were all relative to some axis. We rotated around the coordinate axes x, y, and z, and then we derived the rotation around an arbitrary axis (a_x, a_y, a_z), finding its transform matrix M_{arb}. From the derivation in Chapter 4, recall the expression for M_{arb} where $c = \cos\theta$ and $s = \sin\theta$.

$$M_{arb} = \begin{bmatrix} c + (1-c)a_x^2 & (1-c)a_xa_y - sa_z & (1-c)a_xa_y + sa_y \\ (1-c)a_xa_y + sa_z & c + (1-c)a_y^2 & (1-c)a_ya_z - sa_x \\ (1-c)a_xa_z - sa_y & (1-c)a_ya_z + sa_x & c + (1-c)a_z^2 \end{bmatrix} \tag{5.7}$$

What happens when we have a sequence of several rotations one after the other? We saw this earlier in the derivation of the rotation around an arbitrary axis; there, we multiplied five individual rotations together to get the single matrix result in Equation 5.7. Surprisingly, no matter how many rotations we apply, one after the other, the result is always the same as a single rotation around some axis. The eighteenth century mathematician Leonard Euler showed that no matter what way a sphere is rotated around its center, there always is an axis that remains fixed. For our purpose, the theorem is clearer when stated for two rotations.

Theorem 5.1 (Euler on Rotations). *In three dimensions, the composition of two rotations is equivalent to a single rotation around some axis.*

Applying this theorem several times to a long string of rotations shows that any composition of rotations is equivalent to a single rotation. We certainly know that multiplying all the rotation matrices together gives a single matrix, but the theorem claims this single matrix is actually the matrix for a rotation around some axis. There are proofs of Euler's theorem using linear algebra or using the geometry on a sphere, but here a simple example helps build some intuition about this result.

Example 5.4 (Two Rotations Equivalent to One). Consider a rotation around the x-axis (basis vector $\vec{i}$) followed by a rotation around the z-axis (basis vector $\vec{k}$.) In both cases, the angle of rotation is $\pi/2$ counterclockwise. Then, the two rotation matrices are R_x and R_z.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} R_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Respecting the order of the rotations, the product matrix is

$$R = R_z R_x = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Now notice that R fixes the vector $(1,1,1)$. If we normalize this vector, we get $\frac{1}{\sqrt{3}}(1,1,1)$ and, when we look at M_{arb} in Equation 5.7 using $a_x = a_y = a_z = \frac{1}{\sqrt{3}}$ and $\theta = 2\pi/3$, we see that $R = M_{arb}$. This composition of two rotations is equivalent to a single rotation of $\theta = 2\pi/3$ radians around the axis $(1,1,1)$. The point $(1,1,0)$ is sent to $(0,1,1)$ as expected (after checking a sketch of the coordinate system). □

There are various perspectives we can take on the transformation $R = R_z R_x$ from the last example. The easiest is to recall how the rotation matrices were derived by finding the new coordinates for a rotated point. Then the combination of two rotations, or many rotations, is seen as moving a point within a fixed (local) coordinate system. Points (hence, vertices) move and the coordinate system is fixed.

We can also imagine the coordinates of points staying fixed while the coordinate basis vectors change. In this case, the matrices are coordinate transformations and the coordinates for a vertex change because there is a new rotated coordinate system. Look at R_x in terms of the rotation angle.

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

The columns of R_x are the coordinates of the transformed basis vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ under a counterclockwise rotation around the x-axis. This means that, if M in Result 5.1 were a clockwise rotation, then $R_x = M^{-1}$ is the matrix that converts the original coordinates ($\mathcal{S}_1$) into coordinates in the transformed system ($\mathcal{S}_2$). The system $\mathcal{S}_2$ is the result of rotating the original system ($\mathcal{S}_1$) clockwise around the x-axis.

A similar interpretation of R_z as the inverse of a clockwise rotation around z allows us to give the composite conversion.

$$R_z R_x = M_{\mathcal{S}_2 \to \mathcal{S}_3} M_{\mathcal{S}_1 \to \mathcal{S}_2} = M_{\mathcal{S}_1 \to \mathcal{S}_3}$$

Coordinate system $\mathcal{S}_3$ is system $\mathcal{S}_2$ transformed by a clockwise rotation around its (new) z-axis. When thinking of coordinates, the vector $(0,0,1)$ in $\mathcal{S}_3$ is the z-axis and this is fixed by R_z. Figure 5.5 shows the three coordinate systems using the rotations from Example 5.4.

The point $P = (1,1,0)$ in system $\mathcal{S}_1$ has coordinates $(1,0,1)$ in transformed system $\mathcal{S}_2$ and coordinates $(0,1,1)$ in the system $\mathcal{S}_3$. These last coordinates are the same coordinates that the rotated point has in system $\mathcal{S}_1$.

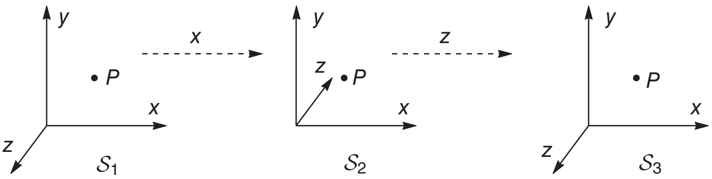


Figure 5.5 Two rotations

The rotation $R = R_z R_x$ from the example can be interpreted as rotating the basis vectors into a new coordinate system. These rotations are counterclockwise and the basis vector $(1, 0, 0)$ ends up as $(0, 1, 0)$. Looking at the final transformed coordinate system, multiplying R times the point with local coordinates $(1, 1, 0)$ gives original coordinates $(0, 1, 1)$. So the transformation R converts the local coordinates in the transformed coordinate system into original coordinates. This is exactly what we want. To rotate an object, we apply a rotation matrix to the basis vectors. Then, relative to the original coordinate system, to find the coordinates for any vertex in the rotated object, we multiply by the same rotation matrix.

In the previous analysis of two rotations, the fact that R_x, for example, has columns that are the transformed basis vectors is not unique. Actually, all rotation matrices have columns that are unit vectors perpendicular to each other. The same is true of the rows. This characterization is summarized in the following result.

Result 5.2 (Rotation Matrices). *Every rotation matrix is an orthogonal matrix, which means that the columns considered as vectors all have length* 1 *and are perpendicular to each other. The rows are also unit vectors that are perpendicular to each other. This implies that* $R^{-1} = R^T$. *Consequently,* $(R_1 R_2)^{-1} = R_2^T R_1^T$.

A combination of rotations is a single rotation, but since we are very comfortable with rotations around the x-, y-, and z-axis, it is has traditionally been practical to express any single rotation in terms of rotations around these axes. As an example, the orientations of spacecraft are given in terms of three angles of rotation around perpendicular axes. Unfortunately, there are various ways to order the three angles. We could, for example, rotate around the x-axis, then around the new z-axis, and finally around the new x-axis, or we could fix the three axes and rotate around each in some order. In both cases, we can achieve all possible rotations.

There are several (12) possibilities for the order of rotations around axes that can express all rotations. One common ordering is the one just given: rotate around the x-axis, then the new z-axis, and finally around the new x-axis. We will settle on rotating first around the x-axis through angle θ, then around the new y-axis through angle α, and finally around the newest z-axis through angle β. Form the product of the three matrices:

$$R_z R_y R_x = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

The entries in the above matrix are

$$
\begin{aligned}
r_{11} &= \cos\beta\cos\alpha \\
r_{12} &= \cos\beta\sin\alpha\sin\theta - \sin\beta\cos\theta \\
r_{13} &= \cos\beta\sin\alpha\cos\theta + \sin\beta\sin\theta \\
r_{21} &= \sin\beta\cos\alpha \\
r_{22} &= \sin\beta\sin\alpha\sin\theta + \cos\beta\cos\theta \\
r_{23} &= \sin\beta\sin\alpha\cos\theta - \cos\beta\sin\theta \\
r_{31} &= -\sin\alpha \\
r_{32} &= \cos\alpha\sin\theta \\
r_{33} &= \cos\alpha\cos\theta
\end{aligned}
\tag{5.8}
$$

If we are given a rotation matrix (around some axis), then we know the entries r_{ij} and, using the above equalities, we can solve for the angles θ, α, β. In particular, we get the following formulas for the three angles:

$$
\boxed{
\begin{aligned}
\theta &= \tan^{-1}\left(\frac{r_{32}}{r_{33}}\right) \\
\alpha &= -\sin^{-1}(r_{31}) \\
\beta &= \tan^{-1}\left(\frac{r_{21}}{r_{11}}\right)
\end{aligned}
}
$$

The three angles θ, α, β are called the *Euler angles*, but there are a few subtleties in the formulas. For example, the second formula has more than one solution; we might try to fix this by restricting the range of each angle appropriately. However, there is still a problem with both the first and third formulas if $\cos\alpha = 0$. This is more significant and we can assess the difficulty by temporarily taking $\alpha = \pi/2$. Then the rotation matrix looks as follows:

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

Since the entries in this matrix are constant or depend on $\theta - \beta$, there are an infinite number of solutions for the three Euler angles that give this rotation matrix. All values that result, for example, in $\theta - \beta = \pi/3$, give the same rotation matrix. No amount of restricting the range of angles will help here. This situation is referred to as the *gimbal lock*, where the name comes from the mechanical design of gyroscopes. In orienting space craft, gimbal lock is a situation where the gyroscopic system cannot rotate appropriately to compensate for all perturbations. Mathematically, it means that changing angles θ and β changes the rotation only in a certain way. Changes in θ and β do not give us enough degrees of freedom to reach all orientations close to the current one.

Gimbal lock is indicative of a larger problem for computer graphics. In animation, we need to move smoothly from one orientation to another. Some method of interpolation is necessary to generate the intermediate orientations for the transition. One method is to take a linear combination of the initial and final orientations so that an intermediate angle could be generated by $(1-t)\theta_1 + t\theta_2$. If we generate intermediate orientations by interpolating the three Euler angles in this way, the results are not always satisfactory. For initial positions that are near the gimbal lock, the trajectory for the animation can look odd. Many times the interpolations do look fine, but the odd cases can occur somewhat unexpectedly. Theoretically, there are fixes for these problems, but the resulting algorithms begin to become unwieldy. A better solution is to turn to quaternions to represent orientations.

Euler angles often prove useful in orienting the camera for a scene. We imagine our head positioned at the center of the camera looking toward the scene (down the z-axis). Then, we can turn our head side to side (called *yaw* in aeronautic terminology), or we can nod up and down (called *pitch*), or we can tilt our head side to side (called *roll*). All these orientations can be described by one of the Euler angles, and putting them together we get a single rotation matrix.

5.2.3 Quaternions

One of the predominant themes in the mathematics of computer graphics is the connection between algebra and geometry. Adding vectors and multiplying matrices both have relevant geometrical meaning when we consider the vertices of some object. Even the algebra of the real numbers can be connected to the geometry of points on the one-dimensional number line.

Moving up to the complex numbers is yet another story. Complex numbers have the form $a + bi$, where a and b are real numbers, and we then introduce i (not as a vector here) which is a new *imaginary* quantity such that when we square it we get -1. That is, $i^2 = -1$. We are not expected to immediately visualize this quantity or to have developed intuition about it. All we need to know is that its square is -1. The number $z = 2 + 3i$ is an example of a complex number, and we say it has real part equal to 2 and imaginary part equal to 3. There is an algebra for these numbers, which means we can add and multiply them.

$$
\begin{aligned}
(x_1 + y_1 i) + (x_2 + y_2 i) &= (x_1 + x_2) + (y_1 + y_2)i \\
(x_1 + y_1 i)(x_2 + y_2 i) &= (x_1 y_1 - y_1 y_2) + (x_1 y_2 + y_1 x_2)i
\end{aligned}
\tag{5.9}
$$

Notice that when multiplying two complex numbers we get a minus sign in the first term of the product because we replaced i^2 with -1. The theory of complex numbers was developed to help solve equations like $x^2 + 2 = 0$ where real numbers were simply insufficient. They now play a pivotal role in mathematics, both theoretical and applied, but our current interest is limited to their connections to the geometry that might benefit computer graphics. To that end, we need to go a little further in the algebra.

For real numbers, we talk about the absolute value as a way of measuring their size. The absolute value of a product is the product of absolute values. Similarly, we introduce an absolute value for complex numbers which is more accurately called a *norm*. We first define a conjugate and use the notation $\bar{z}$ for the conjugate of z. If $z = a + bi$ then $\bar{z} = a - bi$. The norm of z, denoted $|z|$ is defined by setting $|z|^2 = z\bar{z}$; the square of the norm is z times its conjugate.

$$|z|^2 = z\bar{z} = (a + bi)(a - bi) = a^2 + b^2 \implies |z| = \sqrt{a^2 + b^2} \tag{5.10}$$

It looks like the norm of z is exactly the same as the length of the vector (a, b).

In fact, there is an intimate connection between complex numbers and vectors because we can represent both with two coordinates. The point $z = a + bi$ is the point in the regular Cartesian coordinate system with coordinates (a, b). We can also think of z as analogous to the vector from $(0, 0)$ to (a, b). Finally, it is important to note that in the algebra of complex numbers, $|z_1 z_2| = |z_1||z_2|$. The norm of a product is the product of the norms. One nice use of the norm is in producing a standard form for nonzero complex numbers.

$$z = a + bi = |z|\left(\frac{a}{|z|} + \frac{b}{|z|}i\right) = |z|(\cos\theta + i\sin\theta) \tag{5.11}$$

The idea here is that the numbers $a/|z|$ and $b/|z|$ are numbers between -1 and 1 such that the sum of their squares is 1. This means we can find a θ so that one of the numbers is $\cos\theta$ and the other is $\sin\theta$. We have written the complex number z as its length times a complex number with length 1. This canonical form is often useful in calculations.

We are now in a position to focus on one of the key attributes of the connection between complex numbers and geometry. Let $w = \cos\theta + i\sin\theta$. (We have written the i before the $\sin\theta$ just for clarity.) Since $|w| = 1$, multiplying any $z = a + bi$ by w preserves the length of z.

$$wz = (\cos\theta + i\sin\theta)(a + bi) = (a\cos\theta - b\sin\theta) + (a\sin\theta + b\cos\theta)i$$

If we think of points and vectors, the vector (a, b) which represents z has been transformed through the multiplication by w to another vector $((a\cos\theta - b\sin\theta), (a\sin\theta + b\cos\theta))$. If we look closer and remember the two-dimensional rotation matrix, notice the result of multiplying by the rotation matrix:

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} = \begin{bmatrix}a\cos\theta - b\sin\theta\\ b\cos\theta + a\sin\theta\end{bmatrix}$$

Complex multiplication by w and multiplication by the rotation matrix give the same resulting vector.

We have discovered that complex multiplication is really rotation. What if we multiplied by a w that was not of unit length? Then, we would use the canonical form $w = |w|(\cos\theta + i\sin\theta)$ and the product wz will be a scalar $|w|$ times the previous result. This means that we rotated z and multiplied its length by $|w|$. So geometrically multiplication is a counterclockwise rotation through angle θ where the length of the rotated vector is $|wz| = |w||z|$. This is the key observation, and the more we look at the algebra involved, the more we see how the addition of i to the number system allowed us to introduce a minus sign at just the right place to form a geometrical rotation. The natural next step is to see if we can generalize this to three dimensions in the hope that a new algebra will lead to a more efficient way to manipulate three-dimensional geometric objects.

5.2.4 Quaternion Algebra

In the middle of the nineteenth century, the Irish mathematician William Hamilton invented a new algebraic system by starting with a collection of imaginary quantities called *quaternions*. Although it might seem reasonable to start with complex numbers and add another imaginary quantity, quaternions require two new imaginary quantities giving a set of three denoted by $\{i, j, k\}$. (Again, these are not vectors although the letters conventionally used are also used for orthonormal basis vectors.) The square of each of these quantities is -1, that is, $i^2 = j^2 = k^2 = -1$. A quaternion is a generalized complex number: $\hat{q} = a + bi + cj + dk$. Historically, the tricky part was to define the algebra properly so that we get all the appropriate properties associated with addition, multiplication, and a norm. Hamilton hit on a set of rules for the imaginary quantities that led to a useful algebraic system.

Definition 5.2 (Quaternions). *A quaternion is a mathematical object of the form $\hat{q} = a + bi + cj + dk$, where the imaginary quantities $\{i, j, k\}$ satisfy the following rules:*

$$\begin{aligned} i^2 = j^2 = k^2 &= -1 \\ ij = -ji &= k \\ jk = -kj &= i \\ ki = -ik &= j \end{aligned}$$

It is not an accident that the rules for the imaginary quantities look similar to the rules we adopted for cross products of the basis vectors in an orthonormal coordinate system. (In fact, quaternions came first and led to the definitions of dot product and cross product.) Just as in complex numbers, addition of two quaternions and scalar multiplication takes place component-wise. If $\hat{q}_1 = a_1 + b_1 i + c_1 j + d_1 k$ and $\hat{q}_2 = a_2 + b_2 i + c_2 j + d_2 k$ with scalar s, then

$$\begin{aligned} \hat{q}_1 + \hat{q}_2 &= (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k \\ s\hat{q}_1 &= sa_1 + sb_1 i + sc_1 j + sd_1 k \end{aligned} \tag{5.12}$$

Multiplication of two quaternions happens just as we might expect by using the distributive property to find $4 \times 4 = 16$. Using the rules for the imaginary quantities, the result simplifies to

$$\begin{aligned}\hat{q}_1\hat{q}_2 = &(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2)\\ &+ (a_1b_2 + b_1a_2 + c_1d_2 - c_2d_1)i\\ &+ (a_1c_2 + a_2c_1 + d_1b_2 - b_1d_2)j\\ &+ (a_1d_2 + d_1a_2 + b_1c_2 - b_2c_1)k \qquad (5.13)\end{aligned}$$

It is a little easier to present calculations if we refer to the quaternion $\hat{q}$ by $(a, \vec{w})$ where $\vec{w} = (b, c, d)$. So a quaternion has a scalar part (a) and a vector part $(\vec{w})$. Using the dot product and cross product, we can express the product of two quaternions in vector form:

$$\begin{aligned}\hat{q}_1\hat{q}_2 &= (a_1, \vec{w}_1)(a_2, \vec{w}_2)\\ &= ((a_1a_2 - \vec{w}_1 \cdot \vec{w}_2), (a_1\vec{w}_2 + a_2\vec{w}_1 + \vec{w}_1 \times \vec{w}_2)) \qquad (5.14)\end{aligned}$$

By looking at the rules for the imaginary quantities, or by noticing the cross product in the expression for the quaternion product, it is clear that quaternion multiplication is not commutative. The product $\hat{q}_1\hat{q}_2$ does not always equal $\hat{q}_2\hat{q}_1$. To finish describing the algebra, we need to define the conjugate for $\hat{q}$.

Definition 5.3 (Conjugate). *The conjugate for the quaternion* $\hat{q} = (a, \vec{w})$ *is* $\hat{q}^* = (a, -\vec{w})$.

Just as in the complex numbers, a quaternion times its conjugate gives the norm squared.

$$\begin{aligned}|\hat{q}|^2 = \hat{q}\hat{q}^* &= ((a^2 + |\vec{w}|^2), (a\vec{w} - a\vec{w} - \vec{w} \times \vec{w}))\\ &= a^2 + b^2 + c^2 + d^2 \qquad (5.15)\end{aligned}$$

If a quaternion has norm equal to 1, we say it is a *unit* quaternion. In the case of conjugates, a quick check shows $\hat{q}\hat{q}^* = \hat{q}^*\hat{q}$. Using the conjugate again, we can define another useful quaternion, the inverse $\hat{q}^{-1}$.

$$\hat{q}^{-1} = \frac{\hat{q}^*}{|\hat{q}|^2} \implies (\hat{q}^{-1})\hat{q} = \hat{q}(\hat{q}^{-1}) = \frac{\hat{q}\hat{q}^*}{|q|^2} = 1 \qquad (5.16)$$

Finally, one more standard form helps us with calculations. For complex numbers, we were able to write any number in the canonical form $z = |z|(\cos\theta + i\sin\theta)$. For quaternions, there is a similar canonical form.

$$\hat{q} = (a, \vec{w}) = |\hat{q}|\left(\frac{a}{|\hat{q}|}, \frac{|\vec{w}|}{|\hat{q}|}\frac{\vec{w}}{|\vec{w}|}\right) = |\hat{q}|(\cos\theta, \vec{u}\sin\theta) \qquad (5.17)$$

The vector $\vec{u} = \vec{w}/|\vec{w}|$ has length 1, and since $a^2 + |\vec{w}|^2 = |\hat{q}|^2$, we can set $|\vec{w}|/|\hat{q}| = \sin\theta$. In fact, the quaternion $\hat{u} = (0, \vec{u})$ acts like i does in complex numbers; they are both square roots of -1. For any unit quaternion with zero scalar ($\hat{u}$), we have $\hat{u}^2 = -1$.

5.2.5 Rotations

We are interested in rotations of three-dimensional vectors, but quaternions are really four-dimensional objects. When we use the form $\hat{q} = a + bi + cj + dk$, the four parameters (a, b, c, d) suggest the connection with four-dimensional vectors. The alternative form $\hat{q} = (a, \vec{w})$ simply partitions the four parameters so that we have one scalar a and one three-dimensional vector $\vec{w} = (b, c, d)$. If we restrict our attention to quaternions with zero scalar, then we can draw a one-to-one relationship with vectors in three space. Interestingly, a relatively simple quaternion transformation results in rotating quaternions with zero scalar.

Theorem 5.2 (Quaternion Rotations). *If $\hat{v}$ is a quaternion with zero scalar and $\hat{q} = (a, \vec{w})$ is any unit quaternion, then the transformation $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ rotates $\hat{v}$ around the axis $\vec{w}$.*

Along with establishing why the transformation T is a rotation, we will need to determine what the angle of rotation actually is. More practically, once we have an axis and angle, we need to find the quaternion $\hat{q}$ that defines a transformation giving the desired rotation.

Example 5.5 (Quaternion Rotation around x Axis). We have called the unit vector that defines the x-axis $\vec{i} = (1, 0, 0)$. So consider the unit quaternion $\hat{q} = \frac{1}{\sqrt{2}}(1, \vec{i}) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i$. Note here that we have used both the vector $\vec{i}$ which is actually $(1, 0, 0)$ and the imaginary quantity i, a square root of -1, to define the quaternion. The standard basis vectors $\{\vec{j}, \vec{k}\}$ correspond to the imaginary quantities j and k in exactly the same way.

$i^2 = -1$

To see what the transformation $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ does, we check its effect on the vectors $(3, 0, 0)$ and $(1, 0, 1)$, which correspond to quaternions $\hat{v}_1 = (0, 3\vec{i}) = 0 + 3i + 0j + 0k$, and on $\hat{v}_2 = (0, \vec{i} + \vec{k}) = 0 + i + 0j + k$.

First we find $\hat{q}^{-1} = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i$, and then we can calculate the effect of the transformation on $\hat{v}_1$.

$$T(\hat{v}_1) = \hat{q}\hat{v}_1\hat{q}^{-1} = \left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)(3i)\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\right) = 3i = \hat{v}_1$$

The quaternion, and hence the vector $(3, 0, 0)$, remains fixed under the transformation T. This is consistent with the theorem, which claims the axis of rotation is $\vec{i}$.

For $\hat{v}_2$, the calculation is a touch trickier because we do need to keep track of the order of multiplication between imaginary quantities.

$$
\begin{aligned}
T(\hat{v}_2) = \hat{q}\hat{v}_2\hat{q}^{-1} &= \left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i\right)(i+k)\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i\right) \\
&= \left(\frac{1}{\sqrt{2}}\right)^2 (1+i)(i+k)(1-i) \\
&= \frac{1}{2}(i+k+i^2+ik)(1-i) = \frac{1}{2}(i+k-1-j)(1-i) \\
&= \frac{1}{2}(i+k-1-j-i^2-ki+i+ji) = \frac{1}{2}(i+k-1-j+1-j+i-k) \\
&= (i-j)
\end{aligned}
$$

The transformation expectedly fixed the i component, but rotated the k component counterclockwise around $\vec{i}$ (x axis) through $\pi/2$. This is exactly the result of rotating the original vector $(1,0,1)$ around the x-axis. Just for practice, we can recalculate the transform using the vector form (Equation 5.14) of quaternion multiplication.

$$
\begin{aligned}
T(\hat{v}_2) = \hat{q}\hat{v}_1\hat{q}^{-1} &= \frac{1}{\sqrt{2}}(1,\vec{i})(0,(\vec{i}+\vec{k}))\frac{1}{\sqrt{2}}(1,-\vec{i}) \\
&= \left(\frac{1}{\sqrt{2}}\right)^2 ((0-1),(\vec{i}+\vec{k}-\vec{j}))(1,-\vec{i}) = \frac{1}{2}(-1,(\vec{i}-\vec{j}+\vec{k}))(1,-\vec{i}) \\
&= \frac{1}{2}((-1-(-1)),(\vec{i}+\vec{i}+\vec{k}-\vec{j}-(\vec{j}+\vec{k})) \\
&= (0,(\vec{i}-\vec{j}) \\
&= (i-j)
\end{aligned}
$$

As expected, we get the same result. Just to keep clear on the meaning of these quantities, the result of the transformation is $i-j$, which is the quaternion $i+(-1)j+0k = (0,(1,-1,0))$. The original vector $(1,0,1)$ was transformed to the vector $(1,-1,0)$. It was rotated counterclockwise by $\pi/2$ around the x-axis (Figure 5.6). □

The previous example gives a little intuition about how quaternion transformations can be used to rotate vectors, but we still need to establish carefully that the transformation $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ is a rotation where $\hat{q}$ is a unit quaternion and $\hat{v}$ is a quaternion with zero scalar. The details are given in Section 5.4, but the idea is first to notice that T is a linear transformation. To show that it is a rotation, we need to establish that T transforms the basis vectors in a coordinate system the same way that a rotation

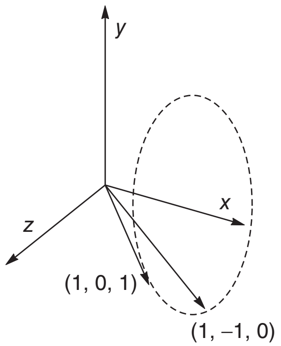


Figure 5.6 Quaternion rotation

does. The linear transformation property then implies that T transforms any vector the same way a rotation does.

Result 5.3 (Selecting $\hat{q}$). *If $\hat{q} = (\cos\theta, \vec{u}\sin\theta)$ and $\hat{v} = (0, \vec{v})$, then the transformation T defined by $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ rotates $\vec{v}$ around the axis $\vec{u}$ through an angle 2θ.*

This result highlights the connection between the quaternion $\hat{q}$ and the angle of rotation. It is then easy to construct T.

Example 5.6 (Rotating with Quaternions). To rotate the vector $(2, -1, 5)$ counterclockwise around the axis $(1, 1, 2)$ through angle $\pi/3$, we begin by setting up the quaternion $\hat{v} = (0, (2, -1, 5))$. We need a unit vector in the direction of the rotation axis, so $\vec{u} = \frac{1}{\sqrt{6}}(1, 1, 2)$. Since the rotation angle is $2\theta = \pi/3$, we have $\theta = \pi/6$ and $\hat{q} = \left(\frac{\sqrt{3}}{2}, \frac{1}{\sqrt{6}}(1, 1, 2)\frac{1}{2}\right)$. Applying the transformation, we get

$$\begin{aligned}
T(\hat{v}) &= \left(\frac{\sqrt{3}}{2}, \frac{1}{\sqrt{6}}(1, 1, 2)\frac{1}{2}\right)(0, (2, -1, 5))\left(\frac{\sqrt{3}}{2}, -\frac{1}{\sqrt{6}}(1, 1, 2)\frac{1}{2}\right) \\
&\approx (-2.25, (3.16, -1.07, 3.72))(0.87, (-0.2, -0.2, -0.41)) \\
&\approx (0, (4.38, 0.07, 3.31))
\end{aligned}$$

Within round-off error, the length of the rotated vector $(4.38, 0.07, 3.31)$ is equal to the length of the original vector $(2, -1, 5)$. This, of course, is expected of a rotation. □

Composing two rotations is relatively easy. If the two corresponding unit quaternions are $\hat{q}_1$ and $\hat{q}_2$, then applying the first and then the second gives a new transformation T^*.

$$T^*(\hat{v}) = \underbrace{\hat{q}_2}_{q_2}\underbrace{\hat{q}_1\hat{v}\overset{q_1}{\hat{q}_1^{-1}}}_{T(\vec{v})}\hat{q}_2^{-1} = (\hat{q}_2\hat{q}_1)\hat{v}(\hat{q}_2\hat{q}_1)^{-1}$$

Again, since quaternion multiplication is not commutative, we have to be careful which rotation we want to do first and which second. We can produce a single unit quaternion to do the combined rotations by simply multiplying $\hat{q}_2$ and $\hat{q}_1$ in that order. Finding the inverse is easy once we have the conjugate and the norm of the product. We have actually established a result we discovered earlier: the composition of two three-dimensional rotations is equivalent to a single three-dimensional rotation around the appropriate axis.

Using quaternions to rotate vectors is theoretically pleasing, but is it actually practical? We can compare elementary operations to give some indication of efficiency. To multiply two quaternions, think about the basic form of the quaternion as a sum $a + bi + cj + dk$ and count the number of multiplication and additions of numbers. It takes 16 multiplications and 12 additions. (Interestingly, multiplying using the vector form of quaternions takes the same number of multiplications and additions.) Consequently, to apply the transformation T for rotation, it takes 2 quaternion multiplications or 32 multiplications and 14 additions. On the other hand, multiplying by a single 3×3 rotation matrix takes only nine multiplications and six additions. The comparison is lopsided in favor of the matrix multiplication.

However, manipulation of a graphics scene requires rotation after rotation, so comparing the composition of two rotations is also key. To compose two quaternion rotations, we multiply two quaternions taking 16 multiplications and 12 additions. For matrices, we need to multiply two 3×3 matrices taking 27 multiplications and 18 additions. Now, it seems there is an advantage for quaternions. Often, primitive operations are implemented in hardware, so our comparisons are only indicative of how speeds might compare.

Regardless of the efficiency gains or losses, another key reason to use quaternions is to interpolate between two orientations of an object and this is what we turn to next.

5.2.6 Interpolation: Slerp

Summarizing what we have done, a unit quaternion $\hat{q}$ determines a rotation in three-space, with the vector part indicating the rotation axis and the scalar part determining the rotation angle. A quaternion $\hat{v}$ with zero scalar can be thought of as a three dimensional vector, and the transformation $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ produces the rotated vector. (The presence of $\hat{q}^{-1}$ in the definition of the transformation means that, even if the quaternion $\hat{q}$ does not have unit norm, we still get the same rotation because $|\hat{q}|$ cancels out.) We now want to expand our concept of quaternions from rotations to orientations.

If we start with the standard orthonormal Cartesian coordinate system with origin O and unit basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$, then a unit quaternion represents a rotation which transforms each of the basis vectors to form a new orthonormal coordinate system. As an animated cube rotates while it moves from one position to another in the scene, its local coordinate system is changing orientation from step to step of the animation. So, in fact, a starting orientation represented by a unit quaternion $\hat{q}_1$ changes into a final orientation represented by quaternion $\hat{q}_2$. Along the way, there is an entire sequence of quaternions representing intermediate orientations. The center of the cube may also move through space, but this change is represented by translations and we can consider the change separately from the orientation.

The interpolation problem is simply to construct a sequence of orientation quaternions beginning with $\hat{q}_1$ and ending with $\hat{q}_2$. It is desirable that the change from quaternion to quaternion is smooth and that the rate of change is constant. A good first guess at a scheme for doing this is to compute quaternions $\hat{q}(t) = (1 - t)\hat{q}_1 + t\hat{q}_2$ as t goes from zero to 1. This is the affine combination we saw when finding points on a line segment, and we say it *interpolates* the end points $\hat{q}_1$ and $\hat{q}_2$.

This linear approach to interpolation changes the orientation smoothly, but it does not have a constant rate of change. To see this, it helps to envision the interpolation more geometrically. A quaternion is a four-dimensional vector, so when we consider unit quaternions, we are looking at points on a four-dimensional sphere. Visualizing such spheres is tough, but thinking by analogy with three-dimensional spheres is easier. Restricting ourselves to unit quaternions, we have two points indicated by $\hat{q}_1$ and $\hat{q}_2$ on the sphere in four dimensions. We want a path on the sphere between the two points and it makes some sense to find the shortest path. (Think analogously to the three-dimensional sphere where the shortest path between two points is the intersection of the sphere with a plane containing the two points and the center of the sphere.) The linear interpolation procedure draws a straight line between the points, but it is not on the sphere; the length of $\hat{q}(t)$ is not necessarily 1. Normalizing $\hat{q}(t)$ by dividing by its length gives a (shortest) path of quaternions that are on the sphere.

Figure 5.7 shows a cross section of the situation. The key observation is that, since linear interpolation takes uniform steps along the straight line, the angle θ in the figure changes faster in the middle of the line and slower near the two end points. Instead, we would like the angular change to be constant. To obtain this uniform change, the angle between $\hat{q}_1$ and $\hat{q}(t)$ should be $t\theta$. This sort of interpolation is called *spherical linear interpolation* (or *slerp*, for short).

To develop a convenient formula for slerp, look at $\hat{v}$ in the figure. The quaternions are four-dimensional vectors and we want to choose quaternion $\hat{v}$ so that it

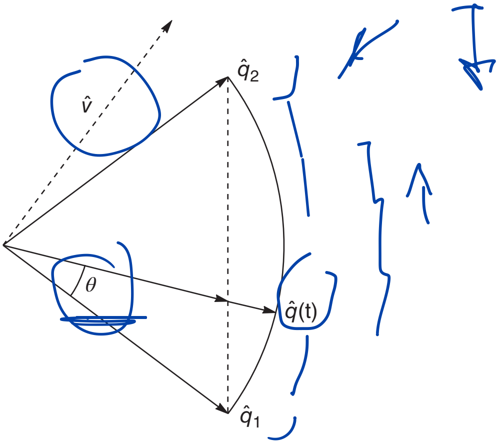


Figure 5.7 Slerp

is perpendicular to $\hat{q}_1$ with length 1; that is, $\hat{v} \cdot \hat{q}_1 = 0$. Define $\hat{q}(t)$ to be an affine combination of $\hat{q}_1$ and $\hat{v}$.

$$\hat{q}(t) = \cos(t\theta)\hat{q}_1 + \sin(t\theta)\hat{v}$$

Assuming we can find an appropriate $\hat{v}$, this definition has some nice properties. For $t = 0$, $\hat{q}(0) = \hat{q}_1$ and $\hat{q}(t)$ has unit length (see Exercises). By construction, the angle between $\hat{q}_1$ and $\hat{q}(t)$ changes uniformly. For $t = 1$, the definition should give $\hat{q}_2$, and we can use this constraint to solve for $\hat{v}$.

$$\begin{aligned} \hat{q}_2 &= (\cos\theta)\hat{q}_1 + (\sin\theta)\hat{v} \\ \implies \hat{v} &= \frac{\hat{q}_2 - (\cos\theta)\hat{q}_1}{\sin\theta} \end{aligned} \tag{5.18}$$

A quick check shows that, assuming $\hat{q}_1$ and $\hat{q}_2$ have length 1, $\hat{v}$ does as well. Remember that the trigonometric addition formulas simplifies the expression for $\hat{q}(t)$.

$$\begin{aligned} \hat{q}(t) &= \cos(t\theta)\hat{q}_1 + \sin(t\theta)\vec{v} \\ &= \cos(t\theta)\hat{q}_1 + \sin(t\theta)\frac{\hat{q}_2 - (\cos\theta)\hat{q}_1}{\sin\theta} \\ &= \frac{1}{\sin\theta}((\sin\theta\cos t\theta - \sin t\theta\cos\theta)\hat{q}_1 + (\sin t\theta)\hat{q}_2) \\ &= \frac{\sin(1-t)\theta}{\sin\theta}\hat{q}_1 + \frac{\sin t\theta}{\sin\theta}\hat{q}_2 \end{aligned}$$

This gives $\hat{q}(t)$ as a combination of $\hat{q}_1$ and $\hat{q}_2$. It smoothly changes the orientation and does so at a constant rate.

There is one ambiguity left. On the sphere, there are two paths between $\hat{q}_1$ and $\hat{q}_2$, one is on the "front" side of the sphere and the other is on the "back." We probably want the shortest one, and to determine that we calculate the dot product of $\hat{q}_1$ and $\hat{q}_2$ as four dimensional vectors. If the result is positive, we know the angle between them is less than $\pi/2$ and the slerp procedure will be mapping the shortest path on the sphere. Otherwise, changing $\hat{q}_2$ to $-\hat{q}_2$ will give a positive dot product. Since $-\hat{q}_2$ represents the same rotation as $\hat{q}_2$, using $-\hat{q}_2$ in this second case will give the shortest path.

Result 5.4 (Slerp Procedure). *Given two unit quaternions $\hat{q}_1$ and $\hat{q}_2$, spherical linear interpolation produces the quaternions*

$$\hat{q}(t) = \frac{\sin(1-t)\theta}{\sin\theta}\hat{q}_1 + \frac{\sin t\theta}{\sin\theta}\hat{q}_2$$

If $\hat{q}_1 \cdot \hat{q}_2 < 0$, use $-\hat{q}_2$ instead of $\hat{q}_2$.

We can use slerp to move objects or to move the camera position. However, when moving the camera position, there is no guarantee that the up vector will stay positioned up (i.e., parallel to its initial position) throughout the interpolation. This may be fine, but it may be necessary to adjust the orientations.

5.2.7 From Euler Angles and Quaternions to Rotation Matrices

To orient an object in space, we position its center using a translation and then rotate it into the desired orientation. The rotation can be described by a rotation matrix, by the three Euler angles, or by an corresponding quaternion. Sometimes, in the midst of coding a graphics application, it is necessary to move between these three descriptions. We have seen how to go from a rotation matrix to Euler angles, and the reverse direction simply requires multiplying the three individual rotation matrices together. Now we need an algorithm for converting quaternions to matrices.

Since rotation is a linear transformation, we only need to know what the transformation does to the basis vectors in order to determine what it does to any vector. So we will calculate the effect of a quaternion on the standard orthonormal basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. Let $\hat{q} = (a, \vec{w}) = (a, b, c, d)$. Assume $\hat{q}$ is a unit quaternion, so $a^2 + b^2 + c^2 + d^2 = 1$.

$$\begin{aligned}
\vec{v}_1 &= \hat{q}\vec{i}\hat{q}^{-1} = (a, b, c, d)(0, 1, 0, 0)(a, -b, -c, -d) \\
&= (-b, a, d, -c)(a, -b, -c, -d) = (0, a^2 + b^2 - c^2 - d^2, 2ad + 2bc, 2bd - 2ac) \\
\vec{v}_2 &= \hat{q}\vec{j}\hat{q}^{-1} = (a, b, c, d)(0, 0, 1, 0)(a, -b, -c, -d) \\
&= (-c, -d, a, b)(a, -b, -c, -d) = (0, -2ad + 2bc, a^2 - b^2 + c^2 - d^2, 2cd - 2ab) \\
\vec{v}_3 &= \hat{q}\vec{k}\hat{q}^{-1} = (a, b, c, d)(0, 0, 0, 1)(a, -b, -c, -d) \\
&= (-d, c, -b, a)(a, -b, -c, -d) = (0, 2bd + 2ac, 2cd - 2ab, a^2 - b^2 - c^2 + d^2)
\end{aligned}$$

Using these transformed vectors as columns gives us the correct rotation matrix.

$$R(\hat{q}) = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{bmatrix}$$

Going backwards to find the quaternion corresponding to a rotation matrix relies on the algebraic relations between entries in this rotation matrix. There are several algorithms for doing this, and one in particular starts by calculating the sum of the diagonal elements, $S = 3a^2 - b^2 - c^2 - d^2$. If the quaternion is a unit quaternion, then we can conclude that $S + 1 = 4a^2$. In a similar vein, we find formulas for the other parameters (r_{ii} is a diagonal entry in the rotation matrix).

$$\begin{aligned}
2r_{11} - S + 1 &= -a^2 + 3b^2 - c^2 - d^2 + 1 = 4b^2 \\
2r_{22} - S + 1 &= -a^2 - b^2 + 3c^2 - d^2 + 1 = 4c^2 \\
2r_{33} - S + 1 &= -a^2 - b^2 - c^2 + 3d^2 + 1 = 4d^2
\end{aligned}$$

We have an equation for each parameter, but solving them requires deciding whether to take the positive or negative square root. One of these decisions is fine, because we know that a quaternion and its negative represent the same orientation. However, once we solve for one parameter this way, we need other equations to guarantee the correct values of the remaining parameters. Notice that $r_{12} + r_{21} = 4bc$ and $r_{21} - r_{12} = 4ad$. Two other pairs of off-diagonal entries give similar equations, and no matter which parameter we solve for first, we can find all the others using these off-diagonal equations.

Since we can convert between Euler angles and rotation matrices in both directions and between quaternions and matrices in both directions, we can pass between all three of these representations.

5.3 OTHER COORDINATE SYSTEMS

Assigning numbers to points was a historical breakthrough that changed the way geometry was done. This led to a wide variety of coordinate systems with the venerable Cartesian coordinate system usually in the front and center. An orthonormal basis for the system adds considerably to the ease of calculation and fits reasonably with most geometric situations. Yet, although many geometric objects have nice descriptions in Cartesian coordinates, most objects in the natural world do not. We can try to approximate them with simpler objects and fancier mathematical techniques, but usually compact descriptions escape us. This argues for at least exploring the other options for coordinate systems.

Keep in mind some of the attributes of a coordinate system that make it particularly useful in graphics applications. First, it should be relatively easy to calculate with the coordinates. Note how dot products are easy to compute in orthonormal Cartesian systems. Second, although it is not necessary, it does help if there is a one-to-one relationship between coordinates and points. Cartesian coordinates have this property, but homogeneous coordinates do not. Without unique coordinates, algorithms often have to deal with several cases. Third, the coordinate system should allow simple or otherwise efficient descriptions of some common objects. Lines, planes, and therefore cubes are easy to describe in Cartesian systems and are even more unified when using homogeneous coordinates.

5.3.1 Non-orthogonal Axes

Rather than choosing basis vectors for a Cartesian coordinate system that are mutually perpendicular (i.e., orthogonal), we may decide that non-perpendicular axes match

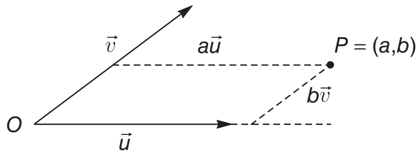


Figure 5.8 Non-perpendicular axes

the shape of our object better. As we will see later, sometimes a pattern we wish to superimpose on some flat face has a shape more conducive to skew axes. Much of our analysis of coordinate systems including converting from one set of coordinates to another allowed for non-perpendicular axes. In two dimensions, the situation is easy to assess, and Figure 5.8 shows two non-perpendicular axes $\vec{u}$ and $\vec{v}$ along with the origin O and the coordinates (a, b) of a point P. The point P is then expressed as $O + a\vec{u} + b\vec{v}$. A useful simplification is to require that $\vec{u}$ and $\vec{v}$ are unit vectors.

We can no longer use the standard formula for dot product in this system because $\vec{u} \cdot \vec{v} \neq 0$. If we do have two vectors described in this system $\vec{w}_1 = (a_1, b_1)$ and $\vec{w}_2 = (a_2, b_2)$, then the dot product requires a little more algebra.

$$\vec{w}_1 \cdot \vec{w}_2 = (a_1\vec{u} + b_1\vec{v}) \cdot (a_2\vec{u} + b_2\vec{v}) = (a_1a_2 + b_1b_2) + (a_1b_2 + a_2b_1)(\vec{u} \cdot \vec{v})$$

Here we assumed that $\vec{u}$ and $\vec{v}$ are unit vectors. The same approach gives a formula for the dot product of three-dimensional vectors described in a non-orthonormal coordinate system.

It is often convenient to convert from a set of non-orthogonal basis vectors into a set of orthogonal ones. The point is to find orthogonal vectors that in some sense match the original set reasonably well. In two dimensions, we simply keep one basis vector and write the other as a sum of a component projected on the first plus a component that is perpendicular to the first.

Example 5.7 (Two Dimensions: Finding Orthogonal Basis Vectors). If the object or pattern we are working with suggests we use the basis vectors $\vec{u} = (1, 2)$ and $\vec{v} = (-1, 3)$, we can first normalize them to get $\vec{u} = \frac{1}{\sqrt{5}}(1, 2)$ and $\vec{v} = \frac{1}{\sqrt{10}}(-1, 3)$. Then decide which vector fits the object or pattern best and take it as the first vector in our new basis. Say $\vec{u}$ is the key vector, so set $\vec{u}_* = \vec{u}$. Now project $\vec{v}$ onto $\vec{u}_*$ to get $(\vec{u}_* \cdot \vec{v})\vec{u}_*$. This projection is in the direction of $\vec{u}_*$, so subtracting this from $\vec{v}$ gives a vector $\vec{v}^*$ that is perpendicular to $\vec{u}_*$

$$\vec{v}_* = \vec{v} - (\vec{u}_* \cdot \vec{v})\vec{u}_* = \frac{1}{\sqrt{10}}(-1, 3) - \frac{1}{\sqrt{10}}(1, 2) = \frac{1}{\sqrt{10}}(-2, 1)$$

We could have found a vector perpendicular to $\vec{u}_*$ in our sleep, but using the projection helps us to ensure that we find the one in the direction that matches $\vec{v}$. After renormalizing $\vec{v}^*$, the two vectors form an orthonormal basis. □

The situation in three dimensions is not much harder and we have already dealt with the problem when looking for a suitable camera coordinate system. There we started with the vector pointing at the scene plus an approximation to the vector pointing up. Then using cross products, we produced a perpendicular up-vector and a third forming a right-hand coordinate system. In the current situation, we might have three vectors to start with, $\{\vec{u}, \vec{v}, \vec{w}\}$. Take $\vec{u}_* = \vec{u}$. Then either $\vec{u}_* \times \vec{v}$ or $\vec{v} \times \vec{u}_*$ gives a perpendicular vector that could serve as $\vec{w}_*$. We decide which based on the dot product with $\vec{w}$. Then again, either $\vec{u}_* \times \vec{w}_*$ or $\vec{w}_* \times \vec{u}_*$ gives $\vec{v}_*$.

Example 5.8 (Three Dimensions: Finding Orthogonal Basis Vectors). Take the vectors $\vec{u} = (1, 1, 1)$, $\vec{v} = (0, -2, 1)$, and $\vec{w} = (-2, 0, 3)$ as the initial basis vectors. Then, $\vec{u}_* = \vec{u} = (1, 1, 1)$. Finding the cross product gives $\vec{u}_* \times \vec{v} = (3, -1, -2)$ or $\vec{v} \times \vec{u}^* = (-3, 1, 2)$.

If it is in the same direction as $\vec{w}$, then the angle between them should be less than $\pi/2$ and the dot product will be positive.

$$\vec{w} \cdot (-3, 1, 2) = 12 > 0 \implies \vec{w}_* = (-3, 1, 2)$$

Now, $\vec{w}_* \times \vec{u}_* = (-1, 5, -4)$ and the dot product with $\vec{v}$ is negative, so we take the opposite direction, $\vec{v}_* = (1, -5, 4)$.

The three vectors $\{(1, 1, 1), (1, -5, 4), (-3, 1, 2)\}$ form an orthogonal system, and once we normalize, they form an orthonormal system. Checking the cross products, we find that the system is right-handed.

Instead of using cross products as we did in this example, we could proceed as we did in Example 5.7 by projecting $\vec{v}$ onto $\vec{u}_*$ and then projecting $\vec{w}$ onto the plane determined by $\vec{u}_*$ and $\vec{v}_*$. In linear algebra, this approach is called the *Gram–Schmidt process*. □

5.3.2 Polar, Cylindrical, and Spherical Coordinates

We have already used angles for designating orientations, and if we push further, we can use them for identifying points. In two dimensions, the origin plus the distance to a particular point gets us started and the angle then indicates direction. The coordinates called *polar coordinates* are (r, θ); r is the distance and θ is the direction given in radians. The direction angle is relative to a fixed direction, so this coordinate system requires both a point designated as the origin and a direction designated as the fixed direction. If there is a Cartesian coordinate system already established, then the fixed direction is usually along the positive x-axis (Figure 5.9). If $\theta = 0$, then the point is somewhere on the positive x-axis. Angles are measured in a counterclockwise manner, so $\theta = \pi/2$ radians means the point is on the positive y-axis. A negative angle would then be a clockwise direction, so $\theta = -\pi/2$ indicates a point on the negative half of the y-axis.

Polar coordinates are not unique. The point P with Cartesian coordinates $(1, 1)$ has polar coordinates $(\sqrt{2}, \pi/4)$, where the polar coordinate system is arranged so $\theta = 0$ indicates the positive x-axis. Clearly, adding a multiple of 2π to θ gives the

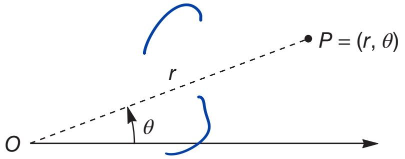


Figure 5.9 Polar coordinate system

same point, so, for example, $P = (\sqrt{2}, 9\pi/2)$. This minor annoyance can be partially patched up by insisting that the direction angle θ be constrained by $0 \leq \theta < 2\pi$. The possibility of r being negative means that the polar coordinates $(-\sqrt{2}, -3\pi/4)$ also denotes P. Again, nonuniqueness rears its head, forcing us to restrict the distance to be nonnegative if we want unique coordinates. However, even with constraints, the origin still has multiple polar coordinates since $r = 0$ and θ can be anything. It is probably not worth obsessing over this uniqueness problem and instead design algorithms to deal with it appropriately.

Polar coordinates are particularly nice for planar objects that have circular symmetry. A circle, for example, has the property that all points are at a fixed distance from the center. So in a local polar coordinate system with the origin at the center, a circle of radius 4 can be described with the simple equation $r = 4$; all points with $r = 4$ and with θ equal to anything are on the circle. A spiral becomes $r = \theta$, where we take only nonnegative values for θ, and one flower-like object is $r = \cos 2\theta$. See the Exercises for relatively simple polar equations that produce aesthetically pleasing (if not practical) patterns (Figure 5.10).

Of course, points have both polar and Cartesian coordinates, and to move between them a little trigonometry finds the relationships.

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \\ r &= \sqrt{x^2 + y^2} \\ \theta &= \tan^{-1} y/x \end{aligned} \tag{5.19}$$

Adding an axis perpendicular to the polar plane boosts polar coordinates into three dimensions. The new axis, which we might call z, gives a third coordinate, and (r, θ, z)

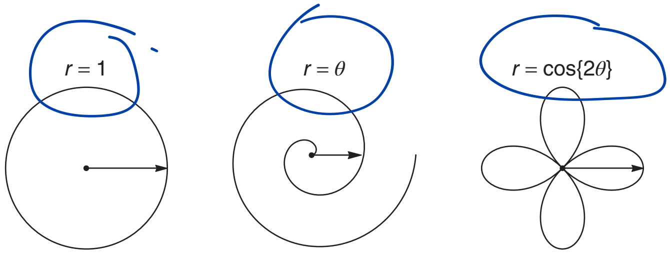


Figure 5.10 Polar curves

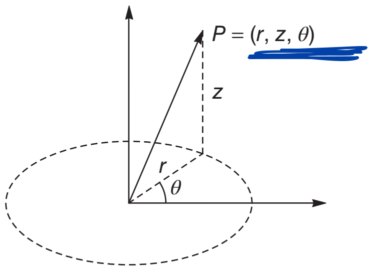


Figure 5.11 Cylindrical coordinate system

now describes points in space. This *cylindrical* coordinate system easily describes a cylinder because $r = 4$ now says that both θ and z can be anything. By using a set of equations, we can extend our descriptive capabilities as, for example, when we use $r = 4$ and $z = 5$ to describe a circle in space. The set of equations $r = 4$ and $z = \theta$ gives a spring (or helix) (Figure 5.11).

Thinking in terms of cylindrical coordinates can make some objects easier to describe and then, if we need, we can convert to Cartesian coordinates for the standard transformations like rotation, scaling, and translation. The conversion proceeds easily because x and y coordinates obey the polar coordinate conversion and z is the same in both cylindrical and Cartesian coordinates.

Instead of just circular symmetry, many objects have close to spherical symmetry. To design a coordinate system which better fits these objects, we start with polar coordinates and add another angular direction rather than a standard axis. Imagine that we have an established three-dimensional Cartesian right-handed coordinate system. Then once again $\theta = 0$ indicates points along the positive x-axis. Now we introduce a new direction designated with the angle ϕ where $0 \leq \phi \leq \pi$ and $\phi = 0$ indicates points along the positive z-axis (Figure 5.12). The distance coordinate is now a distance in three dimensions rather than just two, so we give it a new designation, ρ. The coordinates (ρ, θ, ϕ) are called *spherical coordinates*. (Unfortunately, presentations of spherical coordinates are not all consistent; the two directions θ and ϕ can appear reversed from this presentation and sometimes just the names are reversed.)

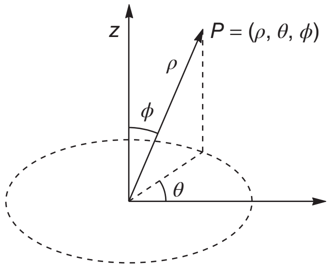


Figure 5.12 Spherical coordinate system

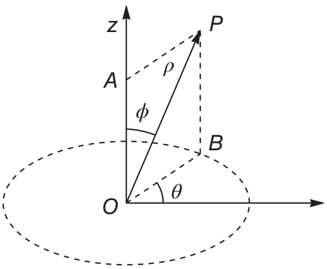


Figure 5.13 Conversion between spherical and Cartesian coordinates

The point with spherical coordinates $(\sqrt{2}, \frac{\pi}{2}, \frac{\pi}{4})$ is the point (0, 1, 1) in the Cartesian system. Again, you can construct several sets of coordinates that denote the same point in space. For example, any point on the z-axis has $\phi = 0$, so the value of θ is irrelevant; both $(1, 0, 0)$ and $(1, \pi, 0)$ are the same point.

In the spherical coordinate system, the equation $\rho = 4$ now describes a sphere of radius 4, and, together, the two equations $\rho = \theta$ and $\phi = \pi/6$ (for nonnegative θ) describe a conical spiral.

To convert from spherical coordinates to Cartesian coordinates and back, take a point P and drop perpendiculars to the z-axis and the xy plane. Then, consider the triangles ΔPOA and ΔPOB in Figure 5.13. The segment PA has length $\rho \sin\phi$ and segment PB is $\rho\cos\phi$ which is just the z coordinate. The equations to convert from spherical to Cartesian coordinates become

$$\begin{aligned} x &= \rho \sin\phi \cos\theta \\ y &= \rho \sin\phi \sin\theta \\ z &= \rho \cos\phi \end{aligned} \tag{5.20}$$

Going from Cartesian to spherical coordinates requires the following formulas:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \tan^{-1}\frac{y}{x} \\ \phi &= \cos^{-1}\frac{z}{\rho} \end{aligned} \tag{5.21}$$

5.3.3 Barycentric Coordinates

Cartesian, polar, cylindrical, and spherical coordinates all require reference objects; they need an origin and either axes or specified directions. What we are doing in most coordinate systems is selecting references that match particular geometric situations. When we are focused on line segments, for example, picking the two end points as references leads to a convenient way for identifying points on the segment.

With A and B as end points, a point P on the segment can be described by noting that $P = A + t(B - A) = (1 - t)A + tB$.

This expresses P as an affine combination of the two end points A and B. It is a weighted average of the two points, and the coordinates $1 - t$ and t are called *barycentric coordinates*. (The prefix "bary" refers to the Greek word for weight.) These coordinates uniquely determine P. When $t = 0$, then $P = A$, and when $t = 1$, $P = B$. The barycentric coordinates are weights and they always sum to 1. (Technically, we could just give the coordinate t knowing that the second one $(1 - t)$ can be readily calculated.)

Without constraining t to the interval $0 \leq t \leq 1$, we can describe any point on the line. If $t > 1$, then we are reaching a point P on the line extending past end point B. Similarly, negative values of t give points on the line extending past end point A. In both these situations, one of the barycentric coordinates is negative and the other positive; the sum is always 1. If we have Cartesian coordinates for the end points, then the weighted average produces Cartesian coordinates for P.

By specifying three reference points instead of two, we can use the same technique to locate points in the plane. Consider points A, B, and C that are not collinear and set them as reference points. Then, an arbitrary point P in the plane can be described by $\underline{P = \alpha_0 A + \alpha_1 B + \alpha_2 C.}$ The three $\underline{\text{barycentric coordinates } (\alpha_0, \alpha_1, \alpha_2)}$ sum to 1, and if they are all nonnegative, then the point is inside the triangle. This coordinate system turns out to be very useful in computer graphics and will be explored in more detail in the Chapter 6.

Generalizing to more reference points unfortunately proves difficult, but various schemes have proved useful in specific situations. If the points are vertices of a polygon with equal sides and angles, the generalization is almost straightforward.

5.4 COMPLEMENTS AND DETAILS

5.4.1 Historical Note: Descartes

The Cartesian coordinate system is named after René Descartes, a French philosopher who wrote a book called *Discourse on Method* published in 1637. The book is really the beginning of modern philosophy and contained three appendices which were more scientific treatises. One of those treatises discussed geometry and suggested the connections between algebra and geometry. There is no mention of perpendicular axes or coordinates, so placing his name on a coordinate system is more the result of good publicity than inventive mathematics. As with most results that bear someone's name, the true history is most likely more convoluted involving the contributions of many along the way.

5.4.2 Historical Note: Hamilton

Sir William Rowan Hamilton was born in Dublin in 1843. As a student at Trinity College, he proved to have exceptional ability in mathematics and went on to make major discoveries in optics and dynamics. He effectively unified dynamics with his equations and brought the same level of analysis to optics. The worldwide recognition that followed made him one of the most influential mathematicians of the time.

In the middle of the nineteenth century, complex numbers were proving their worth and many were drawing connections between complex numbers and two-dimensional geometry. The desire to generalize the ideas to three dimensions piqued Hamilton's interest and he worked steadily trying to devise an algebra that mimicked the complex number system where one imaginary quantity (i) plus the real numbers formed a coherent system of numbers. For three dimensions, the natural generalization was to add a second imaginary quantity. Yet, no one could define multiplication and addition in order to produce an algebra with useful properties like associativity, commutativity, and distributivity.

It turns out that there is no way to define an appropriate algebra for two imaginary quantities, but there is for three imaginary quantities (i, j, k) if the insistence on commutativity is abandoned. After years of work, the solution came to Hamilton one night while walking over the Brougham Bridge in Dublin. The idea was so exciting to him that he reportedly scratched the equations ($i^2 = j^2 = k^2 = ijk = -1$) in one of the bridge stones. Hamilton spent the next 22 years of his life developing techniques for using quaternions in physics and mathematics. Others joined the cause, but possibly due to the ponderous nature of the book Hamilton produced, not many reached the inner circle and quaternions were somewhat slow to catch on. Later in the century, the American physicist Josiah Gibbs (among others like Oliver Heaviside) drew from Hamilton's work to establish the foundations of modern vector analysis; in particular, the cross product operation grew out of quaternion algebra and became the key in many analytical calculations.

Quaternions themselves never quite caught on although Hamilton was convinced of their central position in mathematics. (Arguably, it was Hamilton's work on quaternions that introduced the words "vector" and "scalar" into the mathematical lexicon). In the twentieth century, their connection with rotations pulled them from curious examples to key analytical tools, and the development of computer graphics, no doubt, helped assure quaternions a permanent position among important mathematical objects.

5.4.3 Proof of Quaternion Rotation

To establish carefully that the transformation $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ is a rotation where $\hat{q}$ is a unit quaternion and $\hat{v}$ is a quaternion with zero scalar, we can start by simplifying the problem.

First, write $\hat{q}$ in its canonical form $|\hat{q}|(\cos\theta, \vec{u}\sin\theta)$. This means the conjugate is $\hat{q}^* = |\hat{q}|(\cos\theta, -\vec{u}\sin\theta)$ and the inverse is $\hat{q}^{-1} = \frac{1}{|\hat{q}|}(\cos\theta, -\vec{u}\sin\theta)$. Then the transform T simplifies to

$$\begin{aligned} T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1} &= |\hat{q}|(\cos\theta, \vec{u}\sin\theta)(\hat{v})\frac{1}{|\hat{q}|}(\cos\theta, -\vec{u}\sin\theta) \\ &= (\cos\theta, \vec{u}\sin\theta)(\hat{v})(\cos\theta, -\vec{u}\sin\theta) \qquad (5.22) \end{aligned}$$

In the transformation T, the length of $\hat{q}$ cancels out, implying that any scalar multiple of $\hat{q}$ will give the same transformation. Specifically, $\hat{q}$ and $-\hat{q}$ correspond to the same transformation.

Second, T is in fact a linear transformation. Using the definition of quaternion multiplication, we can show that $T(\hat{v}_1 + \hat{v}_2) = T(\hat{v}_1) + T(\hat{v}_2)$ and that $T(a\hat{v}) = aT(\hat{v})$ (see Exercises). Rotation of three-dimensional vectors is also a linear transformation (it is multiplication by a matrix). To show that these two linear transformations are the same, we only need to show that they act the same on the orthonormal basis vectors for a coordinate system. Then the linear transformation property implies that they act the same on all vectors.

We will pick a coordinate system by starting with the quaternion $\hat{u}$ that appears in the expression for $\hat{q}$. This is a unit quaternion with zero scalar, so it corresponds to a unit vector $\vec{u}$. For the second basis vector, any vector $\vec{r}$ perpendicular to $\vec{u}$ will do, and for the third basis vector we can use $\vec{u} \times \vec{r}$ (Figure 5.14). We will show that T fixes $\vec{u}$ and rotates the other two basis vectors through the same angle. Start with vector $\vec{u}$ and the canonical form $\hat{q} = (\cos\theta, \vec{u}\sin\theta)$. We calculate how T transforms $\hat{u} = (0, \vec{u})$.

$$\begin{aligned}
T(\hat{u}) = \hat{q}\hat{u}\hat{q}^{-1} &= (\cos\theta, \vec{u}\sin\theta)(\hat{u})(\cos\theta, -\vec{u}\sin\theta) \\
&= (-(\vec{u}\cdot\vec{u})(\sin\theta), \vec{u}\cos\theta + \vec{u}\sin\theta \times \vec{u})(\cos\theta, -\vec{u}\sin\theta) \\
&= (-\sin\theta, \vec{u}\cos\theta)(\cos\theta, -\vec{u}\sin\theta) \\
&= (-\sin\theta\cos\theta + (\cos\theta\sin\theta)(\vec{u}\cdot\vec{u}), (\sin\theta)^2\vec{u} + (\cos\theta)^2\vec{u} \\
&\quad - (\cos\theta\sin\theta)(\vec{u}\times\vec{u}) \\
&= (0, (\sin\theta)^2\vec{u} + (\cos\theta)^2\vec{u})) \\
&= (0, \vec{u}) = \hat{u}
\end{aligned}$$

T fixes $\hat{u}$ and therefore fixes the vector $\vec{u}$. It follows that it fixes any multiple of $\vec{u}$ as well. This is exactly what a three-dimensional rotation around the axes $\vec{u}$ does, so T and the rotation agree on multiples of $\vec{u}$.

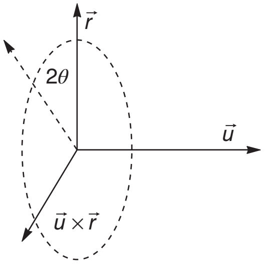


Figure 5.14 Quaternion rotation

Now, take a vector $\vec{r}$ perpendicular to $\vec{u}$ and consider the quaternion $\hat{r} = (0, \vec{r})$.

$$\begin{aligned}
T(\hat{r}) = \hat{q}\hat{r}\hat{q}^{-1} &= (\cos\theta, \vec{u}\sin\theta)(\hat{r})(\cos\theta, -\vec{u}\sin\theta)\\
&= (-(\vec{u}\cdot\vec{r})(\sin\theta), \vec{r}\cos\theta + \vec{u}\sin\theta \times \vec{r})(\cos\theta, -\vec{u}\sin\theta)\\
&= (0, \vec{r}\cos\theta + (\sin\theta)(\vec{u}\times\vec{r}))(\cos\theta, -\vec{u}\sin\theta)\\
&= (0 - (-\sin^2\theta)((\vec{u}\times\vec{r})\cdot\vec{u}), (\cos^2\theta)\vec{r} + (\cos\theta\sin\theta)(\vec{u}\times\vec{r})\\
&\quad - (\cos\theta\sin\theta)(\vec{r}\times\vec{u}) - (\sin^2\theta)(\vec{u}\times\vec{r})\times\vec{u})\\
&= (0, (\cos^2\theta)\vec{r} + 2\cos\theta\sin\theta(\vec{u}\times\vec{r}) - (\sin^2\theta)\vec{r})\\
&= (0, \cos(2\theta)\vec{r} + \sin(2\theta)(\vec{u}\times\vec{r}))
\end{aligned}$$

T sends $\hat{r}$ to the quaternion $(0, \vec{w})$, where $\vec{w}$ is a linear combination of $\vec{r}$ and $\vec{u}\times\vec{r}$. In the plane of these last two vectors, $\vec{r}$ has been rotated counterclockwise through an angle of 2θ. In other words, vectors perpendicular to $\vec{u}$ are rotated around the axis formed by $\vec{u}$. This is precisely what the rotation would do.

For vectors along the axis $\vec{u}$ and for those perpendicular to it, the transformation T and the three-dimensional rotation around $\vec{u}$ agree. For any vector, we can write it as a sum of a vector parallel to $\vec{u}$ and one that is perpendicular. Consequently, T and the rotation agree on every vector. We have shown that T is indeed a three-dimensional rotation. Moreover, we have discovered the appropriate unit quaternion q for accomplishing a given rotation.

5.5 EXERCISES

1. Let the coordinate system $\mathcal{S}_1$ have basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. Relative to system $\mathcal{S}_1$, the system $\mathcal{S}_2$ has origin $(2, -1, 5)$ and basis vectors $\vec{k}, \vec{j}, -\vec{i}$. Find the matrices $M_{\mathcal{S}_1\to\mathcal{S}_2}$ and $M_{\mathcal{S}_2\to\mathcal{S}_1}$. Use the matrices to find the $\mathcal{S}_2$ coordinates of $P = (-3, 6, 2)$ where these coordinates are in $\mathcal{S}_1$.

2. In Example 5.1, find the $\mathcal{S}_1$ coordinates for the basis vectors in $\mathcal{S}_2$ and use them to verify the $\mathcal{S}_2$ coordinates found for P.

3. Let the coordinate system $\mathcal{S}_1$ have basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. Relative to system $\mathcal{S}_1$, the system $\mathcal{S}_2$ has origin $(0, 0, 0)$ and basis vectors $-\vec{k}, \vec{i}, -\vec{j}$. Find the matrices $M_{\mathcal{S}_1\to\mathcal{S}_2}$ and $M_{\mathcal{S}_2\to\mathcal{S}_1}$. Use the matrices to find the $\mathcal{S}_2$ coordinates of $P = (1, 2, 3)$ where these coordinates are in $\mathcal{S}_1$.

4. Let the coordinate system $\mathcal{S}_1$ have basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. Relative to system $\mathcal{S}_1$, the system $\mathcal{S}_2$ has origin $(0, 0, 0)$ and basis vectors $\{(0, 1, 1), (1, 0, 2), (1, -1, 0)\}$. Find the matrices $M_{\mathcal{S}_1\to\mathcal{S}_2}$ and $M_{\mathcal{S}_2\to\mathcal{S}_1}$. Use the matrices to find the $\mathcal{S}_2$ coordinates of $P = (1, 1, 1)$ where these coordinates are in $\mathcal{S}_1$.

5. Assume the world coordinate system has basis vectors $\{\vec{i}, \vec{j}, \vec{k}\}$. There is a cube with vertices $(\pm 1, \pm 1, \pm 1)$ sitting in the world. A camera is centered at location

(20, 10, 5) looking at the point (1, 2, 2). The up vector is approximately (0, 1, 1). Find the matrix $M_{\mathscr{W}\rightarrow\mathscr{C}}$ and use it to find the camera coordinates of the cube's vertices.

6. In the previous problem, suppose the cube is in its own local coordinate system which coincides with the world coordinate system. Rotate the local system $\pi/4$ around the y-axis. Now determine the camera coordinates for the cubes vertices.

7. By using the properties of linear transformations, show that once we know where a linear transformation sends the basis vectors in a coordinate system, we can determine where any vector is sent.

8. If system $\mathscr{S}_1$ is right-handed and system $\mathscr{S}_2$ is left-handed, what can we say about the matrix $M_{\mathscr{S}_1\rightarrow\mathscr{S}_2}$?

9. In Example 5.3, the matrix $M_{\mathscr{W}\rightarrow\mathscr{C}}$ has −0.02 in the upper right corner. Show that, theoretically, this should be zero and, therefore, round-off error must explain the difference.

10. An *orthogonal* matrix is square with orthonormal columns. Show that if M is orthogonal, then $M^T\mathrm{M} = I$ and hence $M^{-1} = M^T$.

11. Reasoning by analogy with the three-dimensional case, give the 3 matrix that transforms coordinates from one system to another. In particular, find $M_{\mathscr{S}_1\rightarrow\mathscr{S}_2}$ where $\mathscr{S}_1$ has basis vectors $\{\vec{i}, \vec{j}\}$ (two dimensions) and $\mathscr{S}_2$ has origin (2, 6) and basis vectors $\{(2, 2), (1, 3)\}$. Verify the matrix by converting (4, 5) from $\mathscr{S}_1$ coordinates to $\mathscr{S}_2$ coordinates.

12. Start with the standard orthonormal coordinate system in three dimensions and rotate counterclockwise through $\pi/2$ around the x-axis. Now, rotate $\pi/2$ clockwise around the z-axis. This produces coordinate systems $\mathscr{S}_1$, $\mathscr{S}_2$, and $\mathscr{S}_3$. Find the $\mathscr{S}_1$ coordinates for a point given in $\mathscr{S}_3$ coordinates. Determine which matrices transform coordinates from each of these systems to the others.

13. Let $R = \begin{bmatrix} 0.577 & 0.408 & 0.707 \\ 0.577 & -0.816 & 0 \\ 0.577 & 0.408 & -0.707 \end{bmatrix}$. Verify that R is a rotation matrix and find the Euler angles appropriate for rotations around x, y, and z in that order.

14. Let z_1 and z_2 be complex numbers. Note that dividing z_1 by z_2 gives $z_1\bar{z}_2/|z_2|^2$. Now take four points in the plane A, B, C, and D. Consider their coordinates as complex numbers. Show that the line through A and B is parallel to the one through C and D if and only if $(A - B)/(C - D)$ is a real (not complex) number. Similarly, show that the lines are perpendicular if and only if $(A - B)/(C - D)$ is a purely imaginary number (no real part).

15. If $\hat{q}$, $\hat{r}$, and $\hat{s}$ are quaternions, show that $\hat{q}(\hat{r} + \hat{s}) = \hat{q}\hat{r} + \hat{q}\hat{r}$.

16. Use quaternions to rotate the vector (1, 1, 2) around the axis (−1, 2, 6) by $\pi/6$ radians.

17. Find the quaternions for a rotation of $\pi/3$ around the axes $(2, 0, -2)$ and $(-3, 1, 1)$. Determine the quaternion representing the composition of these two in order. Find the axis of the resulting rotation.

18. In the slerp derivation, we assumed $\hat{q}_1$ and $\hat{v}$ had unit length. Show that this implies $\hat{q}(t)$ has unit length. Verify that the expression for $\hat{v}$ in Equation 5.18 has unit length.

19. Sketch the graph of the polar equations $r = 1 + \sin\theta$ and $r = \cos(2\theta)$.

20. Find the polar equation of a unit circle.

21. In cylindrical coordinates, describe the shapes $z = 5$, $r = 5$, $z = r$.

22. Using spherical coordinates, describe a unit sphere with a cylindrical hole of diameter 0.25 through the north pole and the south pole of the sphere.

23. An implicit or explicit equation for a two-dimensional line can be put in the form $mx + ny = 1$. Then, the coordinates (m, n) uniquely describe the line since the intercepts on the x- and y-axis are $\left(\frac{1}{m}, 0\right)$ and $\left(0, \frac{1}{n}\right)$. The line can easily be drawn between these intercepts (allowing for m or n to be zero). Show that the equation $m^2 + n^2 = 1$ represents all lines tangent to the unit circle. (Other interesting patterns of lines can be described by various equations using the line coordinates.)

24. Show that $T(\hat{v}) = \hat{q}\hat{v}\hat{q}^{-1}$ is a linear transformation by showing that it satisfies the addition and scalar multiplication properties.

5.5.1 Programming Exercises

1. Write a program to keeping track of a cube of edge length 2 centered at $(0, 0, 1)$. Use a world coordinate system, a local coordinate system, and a camera coordinate system. Draw the cube on the screen and allow the user to rotate around the cube's center or around the world origin. (If possible, draw the cube in perspective.)

2. Write a program for graphing a polar equation. Use it to graph $r = e^{sin\theta} - 2\cos(4\theta) + \sin^5(\theta/12)$.