APPENDIX **C**

Trigonometry and Vector Reference

IN THIS APPENDIX

- Trigonometry
- Vectors

Although we covered a lot of math in the book, this fills in some of the minor details and is for your reference, so you won't have to page through the book to find something quickly.

Trigonometry

Trigonometry is the study of angles, shapes, and their relationships. Most trigonometry is based on the analysis of a right triangle, as shown in Figure C.1.

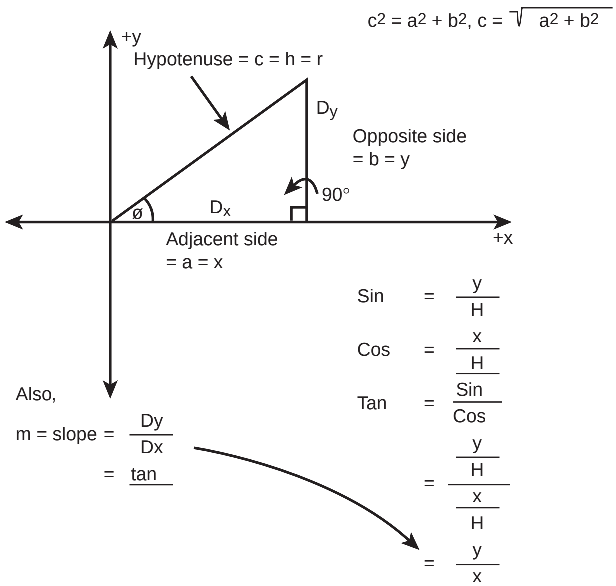

FIGURE C.1 The right triangle.

TABLE C.1 Radians Versus Degrees

360 degrees = 2*PI radians is approx. 6.28 radians
180 degrees = PI radians is approx. 3.14159 radians
$\frac{\text{360 degrees}}{\text{2*PI radians}}$ = 1 radian is approx. 57.296 degrees
$\frac{\text{2*PI radians}}{\text{360 degrees}}$ = 1 degree is approx. 0.0175 radians

Fact 1: There are 360 degrees in a complete circle, or 2*PI radians. Hence, there are PI radians in 180 degrees. The computer functions `sin()` and `cos()` work in radians, *not* degrees—remember that! Table C.1 lists the values.

Fact 2: The sum of the interior angles *theta1* + *theta2* + *theta3* = 180 degrees, or PI radians.

Fact 3: Referring to the right triangle in Figure C.1, the side opposite *theta1* is called the *opposite side*, the side below it is called the *adjacent side*, and the long side is called the *hypotenuse*.

Fact 4: The sum of the squares of the sides of a right triangle equals the square of the hypotenuse. This is called the *Pythagorean Theorem*. Mathematically, we write it like this:

$$\text{hypotenuse}^2 = \text{adjacent}^2 + \text{opposite}^2$$

sometimes using a, b, and c for dummy variables:

$$c^2 = a^2 + b^2$$

Therefore, if you have two sides of a triangle, you can find the third.

Fact 5: There are three main trigonometric ratios that mathematicians like to use: *sine*, *cosine*, and *tangent*. They are defined as

$$\text{cos(theta)} = \frac{\text{adjacent side}}{\text{hypotenuse}} = \frac{x}{r}$$

```
DOMAIN: 0 <= theta <= 2*PI
RANGE: -1 to 1
```

$$\text{sin(theta)} = \frac{\text{opposite side}}{\text{hypotenuse}} = \frac{y}{r}$$

```
DOMAIN: 0 <= theta <= 2*PI
RANGE: -1 to 1
```

$$\text{tan(theta)} = \frac{\text{sin(theta)}}{\text{cos(theta)}} = \frac{\text{opposite/hypotenuse}}{\text{adjacent/hypotenuse}}$$

$$= \frac{\text{opposite}}{\text{adjacent}} = \frac{y}{x} = \text{slope} = \text{M}$$

```
DOMAIN: -PI/2 <= theta <= PI/2
RANGE: -infinity to +infinity
```

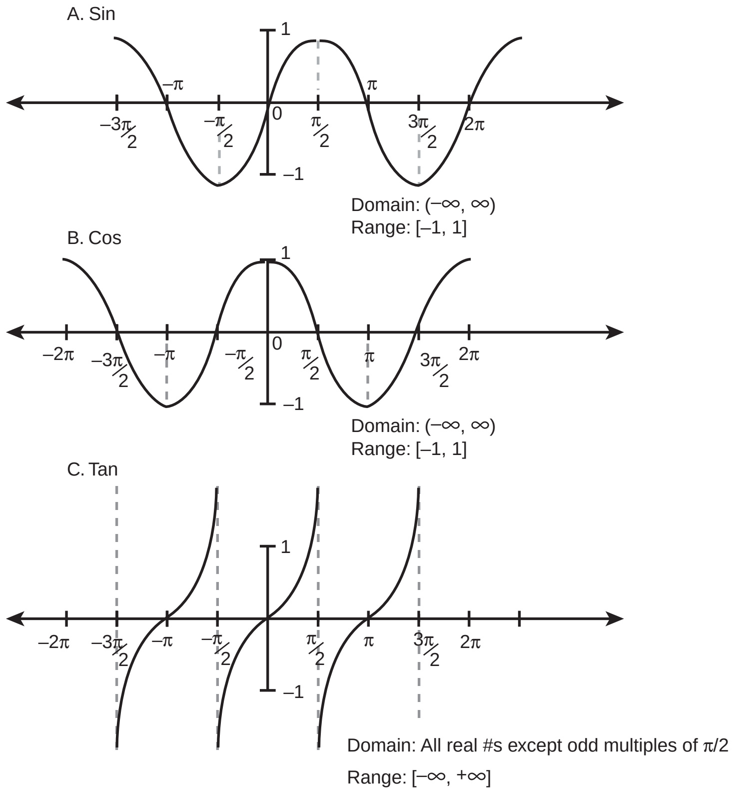

FIGURE C.2 Graphs of basic trigonometric functions.

Figure C.2 shows graphs of all the functions. Notice that all the functions are periodic (repeating) and that *sin(theta)* and *cos(theta)* have periodicity of 2*PI, whereas *tangent* has periodicity of PI. Also, notice that *tan(theta)* goes to ± infinity whenever *theta mod PI* is PI/2.

> **TIP**
>
> You might note the use of the terms *domain* and *range*. These simply mean the input and the output, respectively.

There are numerous trigonometric identities and tricks. Table C.2 lists some of the more useful trigonometric ratios, as well as some interesting identities.

TABLE C.2 Useful Trigonometric Identities

Cosecant: csc(theta) = 1/sin(theta)
Secant: sec(theta) = 1/cos(theta)
Cotangent: cot(theta) = 1/tan(theta)
Pythagorean Theorem in terms of trig functions:
$\sin(\text{theta})^2 + \cos(\text{theta})^2 = 1$
Conversion identity:
sin(theta1) = cos(theta1 – PI/2)
Reflection identities:
sin(-theta) = -sin(theta)
cos(-theta) = cos(theta)
Addition identities:
sin(theta1 + theta2) = sin(theta1)*cos(theta2) + cos(theta1)*sin(theta2)
cos(theta1 + theta2) = cos(theta1)*cos(theta2) - sin(theta1)*sin(theta2)
sin(theta1 - theta2) = sin(theta1)*cos(theta2) - cos(theta1)*sin(theta2)
cos(theta1 - theta2) = cos(theta1)*cos(theta2) + sin(theta1)*sin(theta2)

Of course, you can derive many more identities based upon these. In general, identities help you simplify complex trigonometric formulas, so you don't have to do the math. Hence, when you come up with an algorithm based on *sin*, *cos*, *tan*, and so on, always take a look in a trigonometry book and see whether you can simplify your math, so that fewer computations are needed to get to the result.

Vectors

Vectors are basically nothing more than finite line segments. They are defined by a starting point and an end point, as shown in Figure C.3.

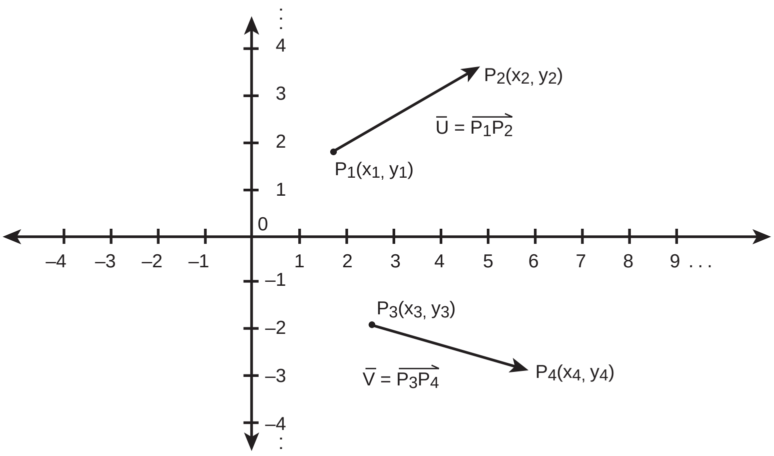

FIGURE C.3 Vectors in the plane.

In Figure C.3, you see a vector **U** defined by the two points p1 (the initial point) and p2 (the terminal point). The vector $\mathbf{U} = \langle u_x, u_y \rangle$ is from p1(x1,y1) to p2(x2,y2). To compute **U**, you simply subtract the initial point from the terminal point:

$$\mathbf{U} = p2 - p1 = (x2-x1,\ y2-y1) = \langle u_x,\ u_y \rangle$$

We usually represent vectors by bolded uppercase letters like **U**, and the components are written within angled brackets like $\langle u_x, u_y \rangle$. However, sometimes I drop to lowercase and lose the bolding if in the context we are only talking about vectors—saves a bit of typing.

Another way to represent the vector from p1 to p2 is by using bar notation. This is done by placing the two endpoints adjacent to each other and placing a bar over them:

$$\overline{p1p2}$$

Okay, so a vector represents a line segment from one point to another, but that segment can represent a lot of concepts, such as *velocity*, *acceleration*, or whatever. Be warned: Once defined, vectors are always relative to the origin. This means that after you create a vector from p1 to p2, the initial point in vector space is always at (0,0), or (0,0,0) in 3D. This doesn't matter because the math takes care of everything, but if you think about it, it makes sense.

A vector is only two or three numbers in 2D and 3D space, so it really only defines an endpoint in 2D or 3D space. Hence, the starting point is always thought of as the origin. This doesn't mean that you can't translate vectors around and perform various geometrical operations with the vectors themselves. It just means that you need to keep in mind what a vector really is.

The cool thing about vectors is the operations you can perform on them. Because vectors are really sets of ordered numbers, you can perform many of the standard mathematical operations on them by performing a mathematical operation on each component independently.

NOTE

Vectors can have any number of components. Usually in computer graphics you deal with 2D and 3D vectors, or vectors of the form $\mathbf{A} = \langle x,y \rangle$, $\mathbf{B} = \langle x,y,z \rangle$. An *n*-dimensional vector has the form

$$\mathbf{C} = \langle c1,\ c2,\ c3,\ \ldots,\ cn \rangle$$

n-dimensional vectors are used to represent sets of variables rather than geometrical space, because after 3D, you enter *hyperspace*.

Vector Length

The length of a vector is called the *norm*, and is represented by two vertical bars, like this: $|\mathbf{U}|$ This is read as "the length of **U**."

The length is computed as the distance from the origin to the tip of the vector. Hence, you can use the standard Pythagorean Theorem to find the length. Therefore, $|\mathbf{U}|$ is equal to

$$|\mathbf{U}| = \mathrm{sqrt}(u_x^2 + u_y^2)$$

And if **U** happened to be a 3D vector, the length would be

$$|\mathbf{U}| = \mathrm{sqrt}(u_x^2 + u_y^2 + u_z^2)$$

Normalization

After you have the length of a vector, you can do something interesting with it—*normalize* the vector, or in other words shrink the vector to make sure that its length is 1.0. Unit vectors have a lot of nice properties just like the scalar 1.0 does, so your intuition probably agrees with me. Given a vector $\mathbf{N} = \langle n_x, n_y \rangle$, the normalized version of **N** is usually written in lowercase as **n**, and is computed like this:

$$\mathbf{n} = \mathbf{N}/|\mathbf{N}|$$

Very simple. The normalized version of a vector is simply the vector divided (multiplied by the inverse) by the length of a vector.

Scalar Multiplication

The first operation that you might want to perform on a vector is scaling. This is performed by multiplying each component by a single scalar number. For example:

$$\text{Let } \mathbf{U} = \langle u_x, u_y \rangle$$
$$k*\mathbf{U} = k*\langle u_x, u_y \rangle = \langle k*u_x, k*u_y \rangle$$

Figure C.4 shows the scaling operation graphically.

In addition, if you want to invert the direction of a vector, you can multiply any vector by –1. This will invert the vector, as shown in Figure C.5.

Mathematically:

$$\text{Let } \mathbf{U} = \langle u_x, u_y \rangle$$

The vector in the opposite direction of **U** is

$$-1*\mathbf{U} = -1*\langle u_x, u_y \rangle = \langle -u_x, -u_y \rangle$$

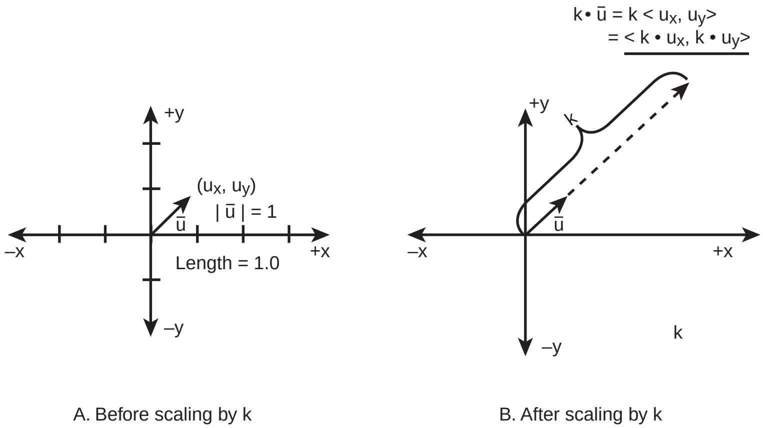

FIGURE C.4 Vector scaling.

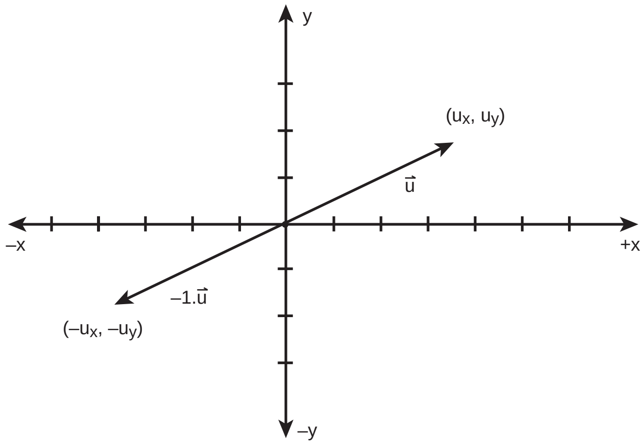

FIGURE C.5 Vector inversion.

Vector Addition

To add two (or more) vectors together, all you have to do is add the respective components together. Figure C.6 illustrates this graphically.

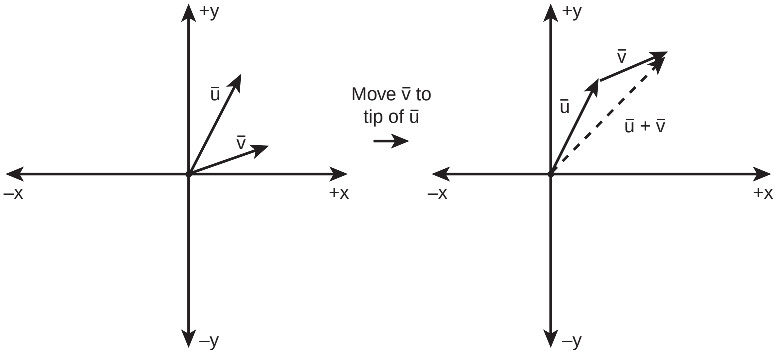

FIGURE C.6 Vector addition.

Vector **U** is added to **V**, and the result is **R**. Notice how the addition was performed geometrically. I took **V**, moved it to the terminal point of **U**, and then drew the other side of the triangle. Geometrically, this is equivalent to the following operation:

$$\mathbf{U} + \mathbf{V} = \langle u_x, u_y \rangle + \langle v_x, v_y \rangle = \langle u_x+v_x, u_y+v_y \rangle$$

Thus, to add any number of vectors together on graph paper, you can simply add them "tip to tail." Then, when you add them all up, the vector from the origin to the last tip is the result.

Vector Subtraction

Vector subtraction is really vector addition with the opposite pointing vector. However, it is sometimes illustrative to see subtraction graphically as well. Take a look at Figure C.7 to see **U**–**V** and **V**–**U**.

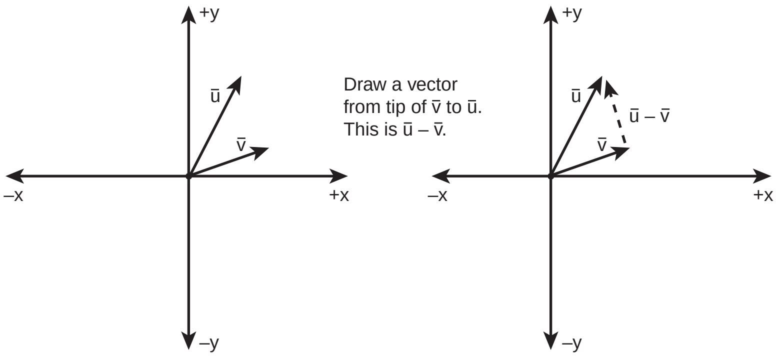

FIGURE C.7 Vector subtraction.

C

Notice that the **U**–**V** is computed by drawing a vector from **V** to **U**, and **V**–**U** is computed by drawing a vector from **U** to **V**. Mathematically, we have

$\mathbf{U} - \mathbf{V} = \langle u_x, u_y\rangle + \langle v_x, v_y\rangle = \langle u_x\text{-}v_x, u_y\text{-}v_y\rangle$

This might be easier to remember, but a piece of graph paper can sometimes be a much better "computer" when you're doing the math manually, because you can visualize the data more quickly. Hence, it's a good idea to know how to add and subtract vectors on graph paper when you're rendering algorithms—trust me!

The Inner Product, or the "Dot" Product

The dot product is defined like this:

$\mathbf{U} . \mathbf{V} = u_x*v_x + u_y*v_y$

The dot product, usually represented by a dot (.), is computed by summing the products of the individual terms. Moreover, the result is a scalar. Well, heck—how does that help? We don't even have vectors anymore! True, my young Jedi, but the dot product is also equal to this expression:

$\mathbf{U} . \mathbf{V} = |\mathbf{U}|*|\mathbf{V}|*\cos\theta$

This states that **U** dot **V** is equal to the length of **U** multiplied by the length of **V** multiplied by the cosine of the angle between the vectors. If we combine the two different expressions, we get this:

$\mathbf{U} . \mathbf{V} = u_x*v_x + u_y*v_y$
$\mathbf{U} . \mathbf{V} = |\mathbf{U}|*|\mathbf{V}|*\cos\theta$
$u_x*v_x + u_y*v_y = |\mathbf{U}|*|\mathbf{V}|*\cos\theta$

This is a very interesting formula—it basically gives us a way to compute the angle between two vectors, as shown in Figure C.8, which is a really useful operation.

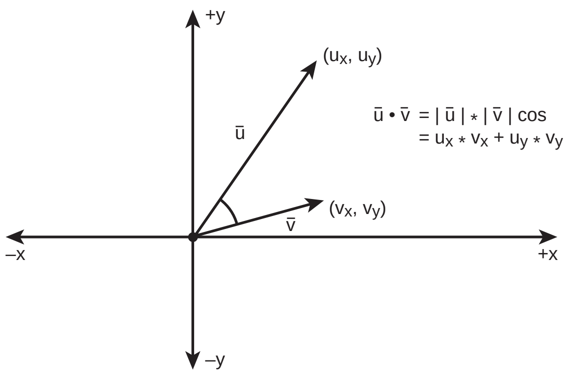

FIGURE C.8 The dot product.

If you can't see that, take a look at the equation after rearranging and taking the inverse cosine of both sides:

$$\theta = \cos^{-1} (u_x * v_x + u_y * v_y / |\mathbf{U}| * |\mathbf{V}|)$$

Or, more compactly, we assume that (**U.V**) means ($u_x * v_x + u_y * v_y$) and just write:

$$\theta = \cos^{-1} (\mathbf{U.V} / |\mathbf{U}| * |\mathbf{V}|)$$

This is a very powerful tool, and the basis of many 3D graphics algorithms. The cool thing is that if the length of **U** and **V** are already 1.0, their product is 1.0 and the formula simplifies further to

$$\theta = \cos^{-1} (\mathbf{U.V}), \text{ for } |\mathbf{U}|=|\mathbf{V}| = 1.0$$

And here are a few properties of the dot product:

Property 1: If the angle between **U** and **V** is 90 (perpendicular), **U.V** = 0.

Property 2: If the angle between **U** and **V** is < 90 (acute), **U.V** > 0.

Property 3: If the angle between **U** and **V** is > 90 (obtuse), **U.V** < 0.

Property 4: If **U** and **V** are equal, **U.V** = $|\mathbf{U}|^2 = |\mathbf{V}|^2$.

These properties are all shown graphically in Figure C.9.

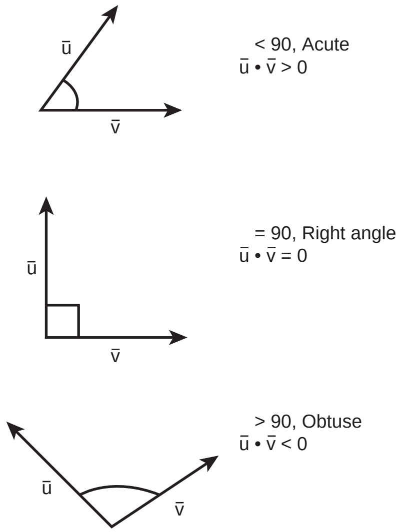

FIGURE C.9 Angles and their relationship to the dot product.

The Cross Product

The next type of multiplication that can be applied to vectors is called the *cross product*. However, the cross product only makes sense on vectors with three or more components, so let's use 3D space vectors as an example. Given **U** = <u_x, u_y, u_z> and **V** = <v_x, v_y, v_z>, the cross product written **U**×**V** is defined as

```
U×V = ¦U¦*¦V¦*sin θ * n
```

Analyzing the formula in steps, |**U**| denotes the length of **U**, |**V**| denotes the length of **V**, and sin θ is the sin of the angle between the vectors. Thus, the product (|**U**|*|**V**|*sin θ) is a scalar (that is, a number). Then you multiply it by **n**. But what is **n**? **n** is a unit vector, which is why it's in lowercase. In addition, **n** is a normal vector, which means that it's perpendicular to both **U** and **V**. Figure C.10 shows this graphically.

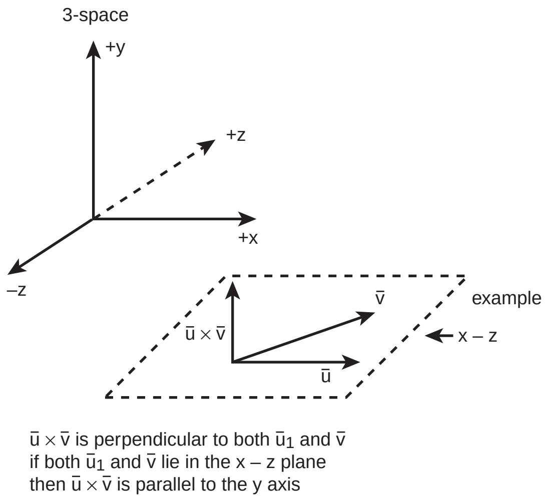

FIGURE C.10 The cross product.

So, the cross product tells us something about the angle between **U** and **V** and the normal vector to both **U** and **V**. But without another equation, you aren't going to get anywhere. The question is how to compute the normal vector from **U** and **V** so that you can compute the sin θ term, or whatever. The cross product is also defined as a very special vector product. However, it's hard to show without matrices, so bear with me. Assume that you want to compute the cross product of **U** and **V**, or **U**×**V.** First you build a matrix like this:

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$$

Here, **i**, **j**, **k** are unit vectors parallel to the x, y, and z axes, respectively.

Then, to compute the cross product of **U** and **V**, you perform this multiplication:

$$\mathbf{N} = (u_y*v_z-v_y*u_z)*\mathbf{i} + (-u_x*v_z+v_x*u_z)*\mathbf{j}\ (u_x*v_y-v_x*u_y)*\mathbf{k}$$

That is, **N** is just a linear combination of three scalars, each multiplied by mutually orthogonal (perpendicular) unit vectors that are each parallel to the x, y, and z axes, respectively. Thus, we can forget the **i**, **j**, **k** and rewrite the equation as

$$\mathbf{N} = \langle u_y*v_z-v_y*u_z,\ -u_x*v_z+v_x*u_z,\ u_x*v_y-v_x*u_y \rangle$$

N is the normal vector to both **U** and **V**. However, it is not necessarily a unit vector (if **U** and **V** were both unit vectors, **N** would be), so you must normalize it to find **n**. After that's done, you can plug everything into the cross product equation and do what you will.

In practice, though, few people ever use the $\mathbf{U}\times\mathbf{V} = |\mathbf{U}|*|\mathbf{V}|*\sin\theta * \mathbf{n}$ formula. They simply use the matrix form to find the normal vector. Again, normal vectors are very important in 3D graphics. Normals are not only great because they are normal to two vectors, but they are also used to define planes, and to compare the orientation of polygons—useful for collision detection, rendering, lighting, and so forth.

The Zero Vector

Although you probably won't use the zero vector much, it still exists. The zero vector has zero length, no direction, and is simply a point, if you want to get technical. Thus, in 2D, the zero vector is <0,0>, and in 3D, it's <0,0,0>, and so on for higher dimensions.

Position Vectors

Position vectors are really useful when tracing out geometrical entities like lines, segments, curves, and so on. I used them when we did clipping in Chapter 10, so it's something that's important. Take a look at Figure C.11, which depicts a position vector that can be used to represent a line segment.

The line segment is from p1 to p2. V is the vector from p1 to p1, and v is a unit vector from p1 to p2. You then construct **P** to trace out the segment. **P** looks like this mathematically:

$$\mathbf{P} = p1 + t*\mathbf{v}$$

Here, *t* is a parameter that varies from 0 to |**V**|. If *t* = 0, you have

$$\mathbf{P} = p1 + 0*\mathbf{v} = \langle p1 \rangle = \langle p1_x,\ p1_y \rangle$$

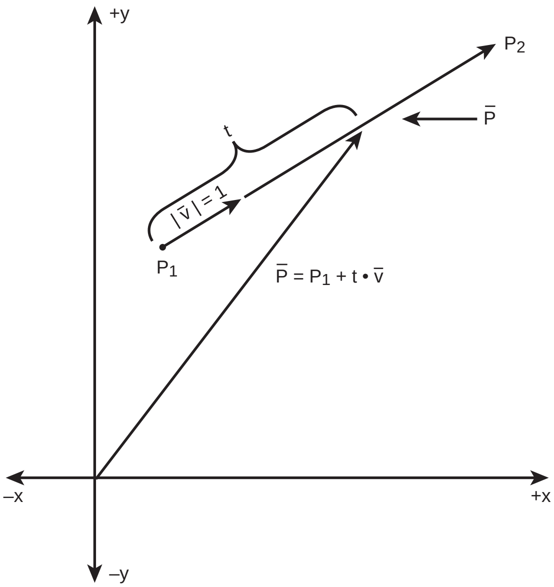

FIGURE C.11 Position vectors.

Thus, at t = 0, **P** is pointing to the beginning of the segment. On the other hand, when t = |**V**|, you have

$$
\begin{aligned}
\mathbf{P} &= p1 + |\mathbf{V}|*\mathbf{v} = p1 + V = \langle p1+V\rangle \\
&= \langle p1_x+V_x,\ p1_y+V_y\rangle \\
&= p2 = \langle p2_x,\ p2_y\rangle
\end{aligned}
$$

Vectors as Linear Combinations

As you saw in the cross product calculation, vectors can also be written in this notation:

$$\mathbf{U} = u_x*\mathbf{i} + u_y*\mathbf{j} + u_z*\mathbf{k}$$

Here, **i**, **j**, and **k** are unit vectors parallel to the x, y, and z axes. There is nothing magical about this; it's just another way to write vectors that you might need to know. All the operations still work exactly the same. For example:

$$
\begin{aligned}
&\text{let } \mathbf{U} = 3\mathbf{i} + 2\mathbf{j} + 3\mathbf{k} \\
&\text{let } \mathbf{V} = -3\mathbf{i} - 5\mathbf{j} + 12\mathbf{k}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{U} + \mathbf{V} &= 3\mathbf{i} + 2\mathbf{j} + 3\mathbf{k} - 3b - 5\mathbf{j} + 12\mathbf{k} \\
&= 0\mathbf{i} - 3\mathbf{j} + 15\mathbf{k} = \langle 0, -3, 15\rangle
\end{aligned}
$$

Nothing but notation, really.