2

Miscellaneous Math

Much of graphics is just translating math directly into code. The cleaner the math, the cleaner the resulting code; so much of this book concentrates on using just the right math for the job. This chapter reviews various tools from high school and college mathematics and is designed to be used more as a reference than as a tutorial. It may appear to be a hodge-podge of topics and indeed it is; each topic is chosen because it is a bit unusual in “standard” math curricula, because it is of central importance in graphics, or because it is not typically treated from a geometric standpoint. In addition to establishing a review of the notation used in the book, the chapter also emphasizes a few points that are sometimes skipped in the standard undergraduate curricula, such as barycentric coordinates on triangles. This chapter is not intended to be a rigorous treatment of the material; instead intuition and geometric interpretation are emphasized. A discussion of linear algebra is deferred until Chapter 5 just before transformation matrices are discussed. Readers are encouraged to skim this chapter to familiarize themselves with the topics covered and to refer back to it as needed. The exercises at the end of the chapter may be useful in determining which topics need a refresher.

2.1 Sets and Mappings

Mappings, also called *functions*, are basic to mathematics and programming. Like a function in a program, a mapping in math takes an argument of one *type* and maps it to (returns) an object of a particular type. In a program we say “type;” in math we would identify the set. When we have an object that is a member of a set, we use the $\in$ symbol. For example,

$$a \in \mathbf{S},$$

can be read "a is a member of set $\mathbf{S}$." Given any two sets $\mathbf{A}$ and $\mathbf{B}$, we can create a third set by taking the *Cartesian product* of the two sets, denoted $\mathbf{A} \times \mathbf{B}$. This set $\mathbf{A} \times \mathbf{B}$ is composed of all possible ordered pairs (a, b) where $a \in \mathbf{A}$ and $b \in \mathbf{B}$. As a shorthand, we use the notation $\mathbf{A}^2$ to denote $\mathbf{A} \times \mathbf{A}$. We can extend the Cartesian product to create a set of all possible ordered triples from three sets and so on for arbitrarily long ordered tuples from arbitrarily many sets.

Common sets of interest include:

- $\mathbb{R}$—the real numbers;
- $\mathbb{R}^+$—the non-negative real numbers (includes zero);
- $\mathbb{R}^2$—the ordered pairs in the real 2D plane;
- $\mathbb{R}^n$—the points in n-dimensional Cartesian space;
- $\mathbb{Z}$—the integers;
- S^2—the set of 3D points (points in $\mathbb{R}^3$) on the unit sphere.

Note that although S^2 is composed of points embedded in three-dimensional space, they are on a surface that can be parameterized with two variables, so it can be thought of as a 2D set. Notation for mappings uses the arrow and a colon, for example:

$$f : \mathbb{R} \mapsto \mathbb{Z},$$

which you can read as "There is a function called f that takes a real number as input and maps it to an integer." Here, the set that comes before the arrow is called the *domain* of the function, and the set on the right-hand side is called the *target*. Computer programmers might be more comfortable with the following equivalent language: "There is a function called f which has one real argument and returns an integer." In other words, the set notation above is equivalent to the common programming notation:

$$\text{integer } f(\text{real}) \quad \longleftarrow \text{equivalent} \longrightarrow \quad f : \mathbb{R} \mapsto \mathbb{Z}.$$

So the colon-arrow notation can be thought of as a programming syntax. It's that simple.

The point $f(a)$ is called the *image* of a, and the image of a set A (a subset of the domain) is the subset of the target that contains the images of all points in A. The image of the whole domain is called the *range* of the function.

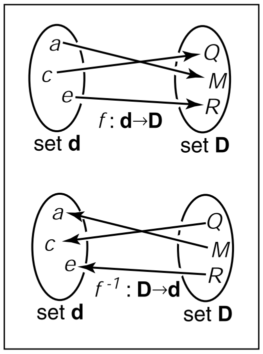

Figure 2.1. A bijection f and the inverse function f^{-1}. Note that f^{-1} is also a bijection.

2.1.1 Inverse Mappings

If we have a function $f : \mathbf{A} \mapsto \mathbf{B}$, there may exist an *inverse function* $f^{-1} : \mathbf{B} \mapsto \mathbf{A}$, which is defined by the rule $f^{-1}(b) = a$ where $b = f(a)$. This definition only works if every $b \in \mathbf{B}$ is an image of some point under f (that is, the range equals the target) and if there is only one such point (that is, there is only one a for which $f(a) = b$). Such mappings or functions are called *bijections*. A bijection maps every $a \in \mathbf{A}$ to a unique $b \in \mathbf{B}$, and for every $b \in \mathbf{B}$, there is exactly one $a \in \mathbf{A}$ such that $f(a) = b$ (Figure 2.1). A bijection between a group of riders and horses indicates that everybody rides a single horse, and every horse is ridden. The two functions would be *rider(horse)* and *horse(rider)*. These are inverse functions of each other. Functions that are not bijections have no inverse (Figure 2.2).

An example of a bijection is $f : \mathbb{R} \mapsto \mathbb{R}$,with $f(x) = x^3$. The inverse function is $f^{-1}(x) = \sqrt[3]{x}$. This example shows that the standard notation can be somewhat awkward because x is used as a dummy variable in both f and f^{-1}. It is sometimes more intuitive to use different dummy variables, with $y = f(x)$ and $x = f^{-1}(y)$. This yields the more intuitive $y = x^3$ and $x = \sqrt[3]{y}$. An example of a function that does not have an inverse is $sqr : \mathbb{R} \mapsto \mathbb{R}$, where $sqr(x) = x^2$. This is true for two reasons: first $x^2 = (-x)^2$, and second no members of the domain map to the negative portions of the target. Note that we can define an inverse if we restrict the domain and range to $\mathbb{R}^+$. Then $\sqrt{x}$ is a valid inverse.

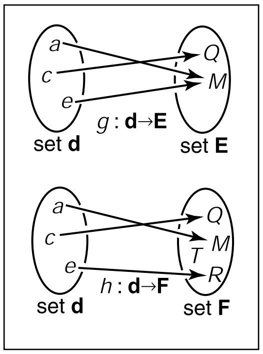

Figure 2.2. The function *g* does not have an inverse because two elements of **d** map to the same element of **E**. The function *h* has no inverse because element *T* of **F** has no element of **d** mapped to it.

2.1.2 Intervals

Often we would like to specify that a function deals with real numbers that are restricted in value. One such constraint is to specify an *interval*. An example of an interval is the real numbers between zero and one, not including zero or one. We denote this $(0, 1)$. Because it does not include its endpoints, this is referred to as an *open interval*. The corresponding *closed interval*, which does contain its endpoints, is denoted with square brackets: $[0, 1]$. This notation can be mixed, i.e., $[0, 1)$ includes zero but not one. When writing an interval $[a, b]$, we assume that $a \leq b$. The three common ways to represent an interval are shown in Figure 2.3. The Cartesian products of intervals are often used. For example, to indicate that a point $\mathbf{x}$ is in the unit cube in 3D, we say $\mathbf{x} \in [0, 1]^3$.

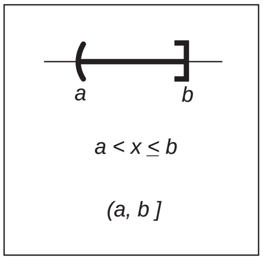

Figure 2.3. Three equivalent ways to denote the interval from *a* to *b* that includes *b* but not *a*.

Intervals are particularly useful in conjunction with set operations: *intersection*, *union*, and *difference*. For example, the intersection of two intervals is the set of points they have in common. The symbol $\cap$ is used for intersection. For example, $[3, 5) \cap [4, 6] = [4, 5)$. For unions, the symbol $\cup$ is used to denote points in either interval. For example, $[3, 5) \cup [4, 6] = [3, 6]$. Unlike the first two operators, the difference operator produces different results depending on argument order.

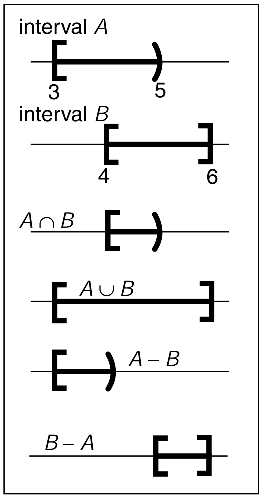

Figure 2.4. Interval operations on [3,5) and [4,6].

The minus sign is used for the difference operator, which returns the points in the left interval that are not also in the right. For example, $[3,5) - [4,6] = [3,4)$ and $[4,6] - [3,5) = [5,6]$. These operations are particularly easy to visualize using interval diagrams (Figure 2.4).

2.1.3 Logarithms

Although not as prevalent today as they were before calculators, *logarithms* are often useful in problems where equations with exponential terms arise. By definition, every logarithm has a *base* a. The "log base a" of x is written $\log_a x$ and is defined as "the exponent to which a must be raised to get x," i.e.,

$$y = \log_a x \quad \Leftrightarrow \quad a^y = x.$$

Note that the logarithm base a and the function that raises a to a power are inverses of each other. This basic definition has several consequences:

$$\begin{aligned} a^{\log_a(x)} &= x; \\ \log_a(a^x) &= x; \\ \log_a(xy) &= \log_a x + \log_a y; \\ \log_a(x/y) &= \log_a x - \log_a y; \\ \log_a x &= \log_a b \ \log_b x. \end{aligned}$$

When we apply calculus to logarithms, the special number $e = 2.718\ldots$ often turns up. The logarithm with base e is called the *natural logarithm*. We adopt the common shorthand $\ln$ to denote it:

$$\ln x \equiv \log_e x.$$

Note that the "$\equiv$" symbol can be read "is equivalent by definition." Like π, the special number e arises in a remarkable number of contexts. Many fields use a particular base in addition to e for manipulations and omit the base in their notation, i.e., $\log x$. For example, astronomers often use base 10 and theoretical computer scientists often use base 2. Because computer graphics borrows technology from many fields we will avoid this shorthand.

The derivatives of logarithms and exponents illuminate why the natural logarithm is "natural":

$$\begin{aligned} \frac{d}{dx} \log_a x &= \frac{1}{x \ln a}; \\ \frac{d}{dx} a^x &= a^x \ln a. \end{aligned}$$

The constant multipliers above are unity only for $a = e$.

2.2 Solving Quadratic Equations

A *quadratic equation* has the form

$$Ax^2 + Bx + C = 0,$$

where x is a real unknown, and A, B, and C are known constants. If you think of a 2D xy plot with $y = Ax^2 + Bx + C$, the solution is just whatever x values are "zero crossings" in y. Because $y = Ax^2 + Bx + C$ is a parabola, there will be zero, one, or two real solutions depending on whether the the parabola misses, grazes, or hits the x-axis (Figure 2.5).

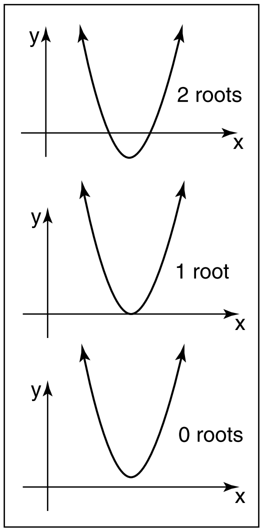

Figure 2.5. The geometric interpretation of the roots of a quadratic equation is the intersection points of a parabola with the *x*-axis.

To solve the quadratic equation analytically, we first divide by A:

$$x^2 + \frac{B}{A}x + \frac{C}{A} = 0.$$

Then we "complete the square" to group terms:

$$\left(x + \frac{B}{2A}\right)^2 - \frac{B^2}{4A^2} + \frac{C}{A} = 0.$$

Moving the constant portion to the right-hand side and taking the square root gives

$$x + \frac{B}{2A} = \pm\sqrt{\frac{B^2}{4A^2} - \frac{C}{A}}.$$

Subtracting $B/(2A)$ from both sides and grouping terms with the denominator $2A$ gives the familiar form:[1]

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}. \tag{2.1}$$

Here the "$\pm$" symbol means there are two solutions, one with a plus sign and one with a minus sign. Thus 3 ± 1 equals "two or four." Note that the term that determines the number of real solutions is

$$D \equiv B^2 - 4AC,$$

which is called the *discriminant* of the quadratic equation. If $D > 0$, there are two real solutions (also called *roots*). If $D = 0$, there is one real solution (a "double" root). If $D < 0$, there are no real solutions.

For example, the roots of $2x^2 + 6x + 4 = 0$ are $x = -1$ and $x = -2$, and the equation $x^2 + x + 1$ has no real solutions. The discriminants of these equations are $D = 4$ and $D = -3$, respectively, so we expect the number of solutions given. In programs, it is usually a good idea to evaluate D first and return "no roots" without taking the square root if D is negative.

[1] A robust implementation will use the equivalent expression $2C/(-B \mp \sqrt{B^2 - 4AC})$ to compute one of the roots, depending on the sign of B (Exercise 7).

2.3 Trigonometry

In graphics we use basic trigonometry in many contexts. Usually, it is nothing too fancy, and it often helps to remember the basic definitions.

2.3.1 Angles

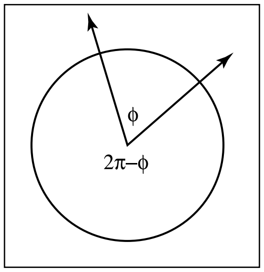

Figure 2.6. Two half-lines cut the unit circle into two arcs. The length of either arc is a valid angle “between” the two half-lines. Either we can use the convention that the smaller length is the angle, or that the two half-lines are specified in a certain order and the arc that determines angle ϕ is the one swept out counterclockwise from the first to the second half-line.

Although we take angles somewhat for granted, we should return to their definition so we can extend the idea of the angle onto the sphere. An angle is formed between two half-lines (infinite rays stemming from an origin) or directions, and some convention must be used to decide between the two possibilities for the angle created between them as shown in Figure 2.6. An *angle* is defined by the length of the arc segment it cuts out on the unit circle. A common convention is that the smaller arc length is used, and the sign of the angle is determined by the order in which the two half-lines are specified. Using that convention, all angles are in the range $[-\pi, \pi]$.

Each of these angles is *the length of the arc of the unit circle that is “cut” by the two directions*. Because the perimeter of the unit circle is 2π, the two possible angles sum to 2π. The unit of these arc lengths is *radians*. Another common unit is degrees, where the perimeter of the circle is 360 degrees. Thus, an angle that is π radians is 180 degrees, usually denoted $180°$. The conversion between degrees and radians is

$$\text{degrees} = \frac{180}{\pi}\text{ radians};$$

$$\text{radians} = \frac{\pi}{180}\text{ degrees}.$$

2.3.2 Trigonometric Functions

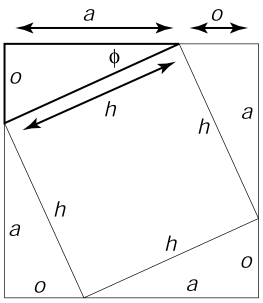

Figure 2.7. A geometric demonstration of the Pythagorean theorem.

Given a right triangle with sides of length a, o, and h, where h is the length of the longest side (which is always opposite the right angle), or *hypotenuse*, an important relation is described by the *Pythagorean theorem*:

$$a^2 + o^2 = h^2.$$

You can see that this is true from Figure 2.7, where the big square has area $(a+o)^2$, the four triangles have the combined area $2ao$, and the center square has area h^2.

Because the triangles and inner square subdivide the larger square evenly, we have $2ao + h^2 = (a + o)^2$, which is easily manipulated to the form above.

We define *sine* and *cosine* of ϕ, as well as the other ratio-based trigonometric expressions:

$$\begin{aligned}
\sin\phi &\equiv o/h;\\
\csc\phi &\equiv h/o;\\
\cos\phi &\equiv a/h;\\
\sec\phi &\equiv h/a;\\
\tan\phi &\equiv o/a;\\
\cot\phi &\equiv a/o.
\end{aligned}$$

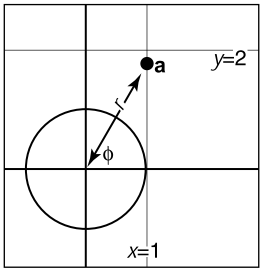

Figure 2.8. Polar coordinates for the point $(x_a, y_a) = (1, \sqrt{3})$ is $(r_a, \phi_a) = (2, \pi/3)$.

These definitions allow us to set up *polar coordinates*, where a point is coded as a distance from the origin and a signed angle relative to the positive x-axis (Figure 2.8). Note the convention that angles are in the range $\phi \in (-\pi, \pi]$, and that the positive angles are counterclockwise from the positive x-axis. This convention that counterclockwise maps to positive numbers is arbitrary, but it is used in many contexts in graphics so it is worth committing to memory.

Trigonometric functions are periodic and can take any angle as an argument. For example $\sin(A) = \sin(A + 2\pi)$. This means the functions are not invertible when considered with the domain $\mathbb{R}$. This problem is avoided by restricting the range of standard inverse functions, and this is done in a standard way in almost all modern math libraries (e.g., (Plauger, 1991)). The domains and ranges are:

$$\begin{aligned}
\text{asin} &: [-1, 1] \mapsto [-\pi/2, \pi/2];\\
\text{acos} &: [-1, 1] \mapsto [0, \pi];\\
\text{atan} &: \mathbb{R} \mapsto [-\pi/2, \pi/2];\\
\text{atan2} &: \mathbb{R}^2 \mapsto [-\pi, \pi].
\end{aligned} \tag{2.2}$$

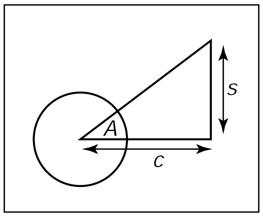

Figure 2.9. The function atan2(*s*,*c*) returns the angle *A* and is often very useful in graphics.

The last function, atan2(s, c) is often very useful. It takes an s value proportional to $\sin A$ and a c value that scales $\cos A$ by the same factor and returns A. The factor is assumed to be positive. One way to think of this is that it returns the angle of a 2D Cartesian point (s, c) in polar coordinates (Figure 2.9).

2.3.3 Useful Identities

This section lists without derivation a variety of useful trigonometric identities.

Shifting identities:

$$\begin{aligned}
\sin(-A) &= -\sin A\\
\cos(-A) &= \quad\cos A\\
\tan(-A) &= -\tan A\\
\sin(\pi/2 - A) &= \quad\cos A\\
\cos(\pi/2 - A) &= \quad\sin A\\
\tan(\pi/2 - A) &= \quad\cot A
\end{aligned}$$

Pythagorean identities:

$$\begin{aligned}\sin^2 A + \cos^2 A &= 1\\ \sec^2 A - \tan^2 A &= 1\\ \csc^2 A - \cot^2 A &= 1\end{aligned}$$

Addition and subtraction identities:

$$\begin{aligned}\sin(A+B) &= \sin A \cos B + \sin B \cos A\\ \sin(A-B) &= \sin A \cos B - \sin B \cos A\\ \sin(2A) &= 2\sin A \cos A\\ \cos(A+B) &= \cos A \cos B - \sin A \sin B\\ \cos(A-B) &= \cos A \cos B + \sin A \sin B\\ \cos(2A) &= \cos^2 A - \sin^2 A\\ \tan(A+B) &= \frac{\tan A + \tan B}{1 - \tan A \tan B}\\ \tan(A-B) &= \frac{\tan A - \tan B}{1 + \tan A \tan B}\\ \tan(2A) &= \frac{2\tan A}{1 - \tan^2 A}\end{aligned}$$

Half-angle identities:

$$\begin{aligned}\sin^2(A/2) &= (1 - \cos A)/2\\ \cos^2(A/2) &= (1 + \cos A)/2\end{aligned}$$

Product identities:

$$\begin{aligned}\sin A \sin B &= -(\cos(A+B) - \cos(A-B))/2\\ \sin A \cos B &= \quad (\sin(A+B) + \sin(A-B))/2\\ \cos A \cos B &= \quad (\cos(A+B) + \cos(A-B))/2\end{aligned}$$

The following identities are for arbitrary triangles with side lengths a, b, and c, each with an angle opposite it given by A, B, C, respectively (Figure 2.10):

$$\frac{\sin A}{a} = \frac{\sin B}{b} = \frac{\sin C}{c} \qquad \text{(Law of sines)}$$

$$c^2 = a^2 + b^2 - 2ab\cos C \qquad \text{(Law of cosines)}$$

$$\frac{a+b}{a-b} = \frac{\tan\left(\frac{A+B}{2}\right)}{\tan\left(\frac{A-B}{2}\right)} \qquad \text{(Law of tangents)}$$

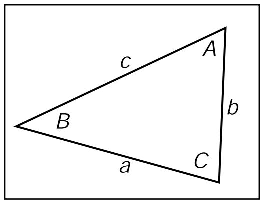

Figure 2.10. Geometry for triangle laws.

The area of a triangle can also be computed in terms of these side lengths:

$$\text{triangle area} = \frac{1}{2}\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}.$$

2.4 Vectors

A *vector* describes a length and a direction. It can be usefully represented by an arrow. Two vectors are equal if they have the same length and direction even if we think of them as being located in different places (Figure 2.11). As much as possible, you should think of a vector as an arrow and not as coordinates or numbers. At some point we will have to represent vectors as numbers in our programs, but even in code they should be manipulated as objects and only the low-level vector operations should know about their numeric representation (DeRose, 1989). Vectors will be represented as bold characters, e.g., $\mathbf{a}$. A vector's length is denoted $\|\mathbf{a}\|$. A *unit vector* is any vector whose length is one. The *zero vector* is the vector of zero length. The direction of the zero vector is undefined.

Vectors can be used to represent many different things. For example, they can be used to store an *offset*, also called a *displacement*. If we know "the treasure is buried two paces east and three paces north of the secret meeting place," then we know the offset, but we don't know where to start. Vectors can also be used to store a *location*, another word for *position* or *point*. Locations can be represented as a displacement from another location. Usually there is some understood *origin* location from which all other locations are stored as offsets. Note that locations are not vectors. As we shall discuss, you can add two vectors. However, it usually does not make sense to add two locations unless it is an intermediate operation when computing weighted averages of a location (Goldman, 1985). Adding two offsets does make sense, so that is one reason why offsets are vectors. But this emphasizes that a location is not a offset; it is an offset from a specific origin location. The offset by itself is not the location.

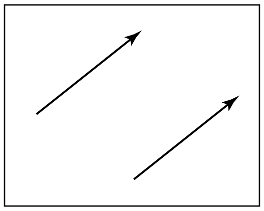

Figure 2.11. These two vectors are the same because they have the same length and direction.

2.4.1 Vector Operations

Vectors have most of the usual arithmetic operations that we associate with real numbers. Two vectors are equal if and only if they have the same length and direction. Two vectors are added according to the *parallelogram rule*. This rule states that the sum of two vectors is found by placing the tail of either vector against the head of the other (Figure 2.12). The sum vector is the vector that "completes the triangle" started by the two vectors. The parallelogram is formed by taking the sum in either order. This emphasizes that vector addition is commutative:

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}.$$

Note that the parallelogram rule just formalizes our intuition about displacements. Think of walking along one vector, tail to head, and then walking along the other.

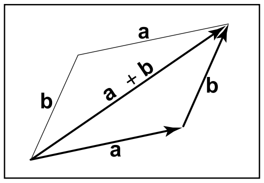

Figure 2.12. Two vectors are added by arranging them head to tail. This can be done in either order.

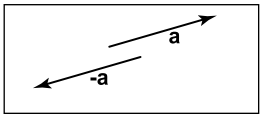

Figure 2.13. The vector –**a** has the same length but opposite direction of the vector **a**.

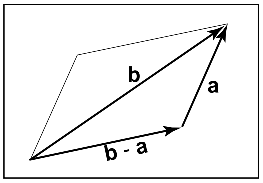

Figure 2.14. Vector subtraction is just vector addition with a reversal of the second argument.

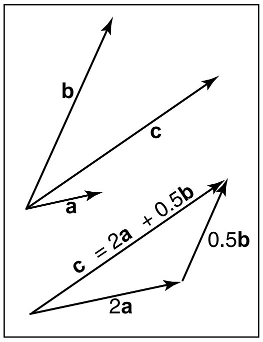

Figure 2.15. Any 2D vector **c** is a weighted sum of any two non-parallel 2D vectors **a** and **b**.

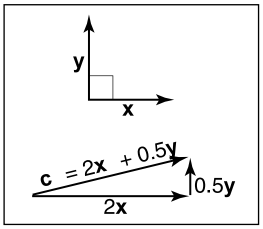

Figure 2.16. A 2D Cartesian basis for vectors.

The net displacement is just the parallelogram diagonal. You can also create a *unary minus* for a vector: $-\mathbf{a}$ (Figure 2.13) is a vector with the same length as $\mathbf{a}$ but opposite direction. This allows us to also define subtraction:

$$\mathbf{b} - \mathbf{a} \equiv -\mathbf{a} + \mathbf{b}.$$

You can visualize vector subtraction with a parallelogram (Figure 2.14). We can write

$$\mathbf{a} + (\mathbf{b} - \mathbf{a}) = \mathbf{b}.$$

Vectors can also be multiplied. In fact, there are several kinds of products involving vectors. First, we can *scale* the vector by multiplying it by a real number k. This just multiplies the vector's length without changing its direction. For example, $3.5\mathbf{a}$ is a vector in the same direction as $\mathbf{a}$ but it is 3.5 times as long as $\mathbf{a}$. We discuss two products involving two vectors, the dot product and the cross product, later in this section, and a product involving three vectors, the determinant, in Chapter 5.

2.4.2 Cartesian Coordinates of a Vector

A 2D vector can be written as a combination of any two non-zero vectors which are not parallel. This property of the two vectors is called *linear independence*. Two linearly independent vectors form a 2D *basis*, and the vectors are thus referred to as *basis vectors*. For example, a vector $\mathbf{c}$ may be expressed as a combination of two basis vectors $\mathbf{a}$ and $\mathbf{b}$ (Figure 2.15):

$$\mathbf{c} = a_c\mathbf{a} + b_c\mathbf{b}. \tag{2.3}$$

Note that the weights a_c and b_c are unique. Bases are especially useful if the two vectors are *orthogonal*, i.e., they are at right angles to each other. It is even more useful if they are also unit vectors in which case they are *orthonormal*. If we assume two such "special" vectors $\mathbf{x}$ and $\mathbf{y}$ are known to us, then we can use them to represent all other vectors in a *Cartesian* coordinate system, where each vector is represented as two real numbers. For example, a vector $\mathbf{a}$ might be represented as

$$\mathbf{a} = x_a\mathbf{x} + y_a\mathbf{y},$$

where x_a and y_a are the real Cartesian coordinates of the 2D vector $\mathbf{a}$ (Figure 2.16). Note that this is not really any different conceptually from Equation (2.3), where the basis vectors were not orthonormal. But there are several advantages to a Cartesian coordinate system. For instance, by the Pythagorean theorem, the length of $\mathbf{a}$ is

$$\|\mathbf{a}\| = \sqrt{x_a^2 + y_a^2}.$$

It is also simple to compute dot products, cross products, and coordinates of vectors in Cartesian systems, as we'll see in the following sections.

By convention we write the coordinates of $\mathbf{a}$ either as an ordered pair (x_a, y_a) or a column matrix:

$$\mathbf{a} = \begin{bmatrix} x_a \\ y_a \end{bmatrix}.$$

The form we use will depend on typographic convenience. We will also occasionally write the vector as a row matrix, which we will indicate as $\mathbf{a}^{\mathrm{T}}$:

$$\mathbf{a}^{\mathrm{T}} = \begin{bmatrix} x_a & y_a \end{bmatrix}.$$

We can also represent 3D, 4D, etc., vectors in Cartesian coordinates. For the 3D case, we use a basis vector $\mathbf{z}$ that is orthogonal to both $\mathbf{x}$ and $\mathbf{y}$.

2.4.3 Dot Product

The simplest way to multiply two vectors is the *dot* product. The dot product of $\mathbf{a}$ and $\mathbf{b}$ is denoted $\mathbf{a} \cdot \mathbf{b}$ and is often called the *scalar product* because it returns a scalar. The dot product returns a value related to its arguments' lengths and the angle ϕ between them (Figure 2.17):

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \; \|\mathbf{b}\| \; \cos\phi, \tag{2.4}$$

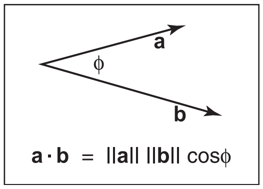

Figure 2.17. The dot product is related to length and angle and is one of the most important formulas in graphics.

The most common use of the dot product in graphics programs is to compute the cosine of the angle between two vectors.

The dot product can also be used to find the *projection* of one vector onto another. This is the length $\mathbf{a}{\rightarrow}\mathbf{b}$ of a vector $\mathbf{a}$ that is projected at right angles onto a vector $\mathbf{b}$ (Figure 2.18):

$$\mathbf{a}{\rightarrow}\mathbf{b} = \|\mathbf{a}\| \; \cos\phi = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|}. \tag{2.5}$$

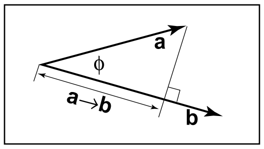

Figure 2.18. The projection of **a** onto **b** is a length found by Equation (2.5).

The dot product obeys the familiar associative and distributive properties we have in real arithmetic:

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= \mathbf{b} \cdot \mathbf{a}, \\ \mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) &= \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}, \\ (k\mathbf{a}) \cdot \mathbf{b} &= \mathbf{a} \cdot (k\mathbf{b}) = k\mathbf{a} \cdot \mathbf{b}. \end{aligned} \tag{2.6}$$

If 2D vectors $\mathbf{a}$ and $\mathbf{b}$ are expressed in Cartesian coordinates, we can take advantage of $\mathbf{x} \cdot \mathbf{x} = \mathbf{y} \cdot \mathbf{y} = 1$ and $\mathbf{x} \cdot \mathbf{y} = 0$ to derive that their dot product is

$$\begin{aligned}\mathbf{a}\cdot\mathbf{b} &= (x_a\mathbf{x}+y_a\mathbf{y})\cdot(x_b\mathbf{x}+y_b\mathbf{y})\\ &= x_ax_b(\mathbf{x}\cdot\mathbf{x})+x_ay_b(\mathbf{x}\cdot\mathbf{y})+x_by_a(\mathbf{y}\cdot\mathbf{x})+y_ay_b(\mathbf{y}\cdot\mathbf{y})\\ &= x_ax_b+y_ay_b.\end{aligned}$$

Similarly in 3D we can find

$$\mathbf{a}\cdot\mathbf{b} = x_ax_b+y_ay_b+z_az_b.$$

2.4.4 Cross Product

The cross product $\mathbf{a}\times\mathbf{b}$ is usually used only for three-dimensional vectors; generalized cross products are discussed in references given in the chapter notes. The cross product returns a 3D vector that is perpendicular to the two arguments of the cross product. The length of the resulting vector is related to $\sin\phi$:

$$\|\mathbf{a}\times\mathbf{b}\| = \|\mathbf{a}\|\,\|\mathbf{b}\|\sin\phi.$$

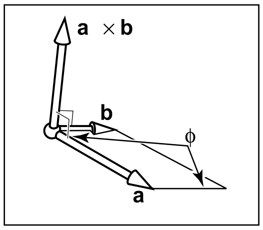

Figure 2.19. The cross product $\mathbf{a}\times\mathbf{b}$ is a 3D vector perpendicular to both 3D vectors **a** and **b**, and its length is equal to the area of the parallelogram shown.

The magnitude $\|\mathbf{a}\times\mathbf{b}\|$ is equal to the area of the parallelogram formed by vectors $\mathbf{a}$ and $\mathbf{b}$. In addition, $\mathbf{a}\times\mathbf{b}$ is perpendicular to both $\mathbf{a}$ and $\mathbf{b}$ (Figure 2.19). Note that there are only two possible directions for such a vector. By definition, the vectors in the direction of the *x*-, *y*- and *z*-axes are given by

$$\begin{aligned}\mathbf{x} &= (1,0,0),\\ \mathbf{y} &= (0,1,0),\\ \mathbf{z} &= (0,0,1),\end{aligned}$$

and we set as a convention that $\mathbf{x}\times\mathbf{y}$ must be in the plus or minus $\mathbf{z}$ direction. The choice is somewhat arbitrary, but it is standard to assume that

$$\mathbf{z} = \mathbf{x}\times\mathbf{y}.$$

All possible permutations of the three Cartesian unit vectors are

$$\begin{aligned}\mathbf{x}\times\mathbf{y} &= +\mathbf{z},\\ \mathbf{y}\times\mathbf{x} &= -\mathbf{z},\\ \mathbf{y}\times\mathbf{z} &= +\mathbf{x},\\ \mathbf{z}\times\mathbf{y} &= -\mathbf{x},\\ \mathbf{z}\times\mathbf{x} &= +\mathbf{y},\\ \mathbf{x}\times\mathbf{z} &= -\mathbf{y}.\end{aligned}$$

Because of the $\sin\phi$ property, we also know that a vector cross itself is the zero-vector, so $\mathbf{x} \times \mathbf{x} = \mathbf{0}$ and so on. Note that the cross product is *not* commutative, i.e., $\mathbf{x} \times \mathbf{y} \neq \mathbf{y} \times \mathbf{x}$. The careful observer will note that the above discussion does not allow us to draw an unambiguous picture of how the Cartesian axes relate. More specifically, if we put $\mathbf{x}$ and $\mathbf{y}$ on a sidewalk, with $\mathbf{x}$ pointing East and $\mathbf{y}$ pointing North, then does $\mathbf{z}$ point up to the sky or into the ground? The usual convention is to have $\mathbf{z}$ point to the sky. This is known as a *right-handed* coordinate system. This name comes from the memory scheme of "grabbing" $\mathbf{x}$ with your *right* palm and fingers and rotating it toward $\mathbf{y}$. The vector $\mathbf{z}$ should align with your thumb. This is illustrated in Figure 2.20.

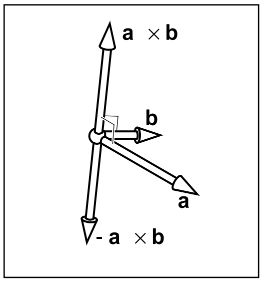

Figure 2.20. The "right-hand rule" for cross products. Imagine placing the base of your right palm where **a** and **b** join at their tails, and pushing the arrow of **a** toward **b**. Your extended right thumb should point toward **a** × **b**.

The cross product has the nice property that

$$\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c},$$

and

$$\mathbf{a} \times (k\mathbf{b}) = k(\mathbf{a} \times \mathbf{b}).$$

However, a consequence of the right-hand rule is

$$\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a}).$$

In Cartesian coordinates, we can use an explicit expansion to compute the cross product:

$$\begin{aligned}\mathbf{a} \times \mathbf{b} &= (x_a\mathbf{x} + y_a\mathbf{y} + z_a\mathbf{z}) \times (x_b\mathbf{x} + y_b\mathbf{y} + z_b\mathbf{z}) \\ &= x_a x_b \mathbf{x} \times \mathbf{x} + x_a y_b \mathbf{x} \times \mathbf{y} + x_a z_b \mathbf{x} \times \mathbf{z} \\ &\quad + y_a x_b \mathbf{y} \times \mathbf{x} + y_a y_b \mathbf{y} \times \mathbf{y} + y_a z_b \mathbf{y} \times \mathbf{z} \\ &\quad + z_a x_b \mathbf{z} \times \mathbf{x} + z_a y_b \mathbf{z} \times \mathbf{y} + z_a z_b \mathbf{z} \times \mathbf{z} \\ &= (y_a z_b - z_a y_b)\mathbf{x} + (z_a x_b - x_a z_b)\mathbf{y} + (x_a y_b - y_a x_b)\mathbf{z}.\end{aligned} \tag{2.7}$$

So, in coordinate form,

$$\mathbf{a} \times \mathbf{b} = (y_a z_b - z_a y_b, z_a x_b - x_a z_b, x_a y_b - y_a x_b). \tag{2.8}$$

2.4.5 Orthonormal Bases and Coordinate Frames

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

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

and

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

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

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

and

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

This orthonormal basis is *right-handed* provided

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

and otherwise it is left-handed.

Note that the Cartesian canonical orthonormal basis is just one of infinitely many possible orthonormal bases. What makes it special is that it and its implicit origin location are used for low-level representation within a program. Thus, the vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are never explicitly stored and neither is the canonical

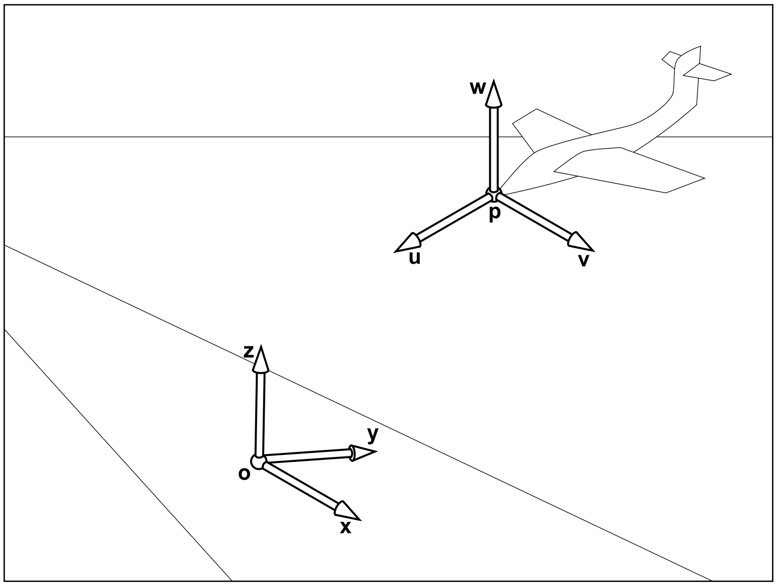

Figure 2.21. There is always a master or "canonical" coordinate system with origin **o** and orthonormal basis **x, y**, and **z**. This coordinate system is usually defined to be aligned to the global model and is thus often called the "global" or "world" coordinate system. This origin and basis vectors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

origin location $\mathbf{o}$. The global model is typically stored in this canonical coordinate system, and it is thus often called the *global coordinate system*. However, if we want to use another coordinate system with origin $\mathbf{p}$ and orthonormal basis vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$, then we *do* store those vectors explicitly. Such a system is called a *frame of reference* or *coordinate frame*. For example, in a flight simulator, we might want to maintain a coordinate system with the origin at the nose of the plane, and the orthonormal basis aligned with the airplane. Simultaneously, we would have the master canonical coordinate system (Figure 2.21). The coordinate system associated with a particular object, such as the plane, is usually called a *local coordinate system*.

At a low level, the local frame is stored in canonical coordinates. For example, if $\mathbf{u}$ has coordinates (x_u, y_u, z_u),

$$\mathbf{u} = x_u\mathbf{x} + y_u\mathbf{y} + z_u\mathbf{z}.$$

A location implicitly includes an offset from the canonical origin:

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

where (x_p, y_p, z_p) are the coordinates of $\mathbf{p}$.

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

$$\mathbf{a} = u_a\mathbf{u} + v_a\mathbf{v} + w_a\mathbf{w}.$$

To get the canonical coordinates of a vector $\mathbf{a}$ stored in the $\mathbf{u}$-$\mathbf{v}$-$\mathbf{w}$ coordinate system, simply recall that $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are themselves stored in terms of Cartesian coordinates, so the expression $u_a\mathbf{u} + v_a\mathbf{v} + w_a\mathbf{w}$ is already in Cartesian coordinates if evaluated explicitly. To get the $\mathbf{u}$-$\mathbf{v}$-$\mathbf{w}$ coordinates of a vector $\mathbf{b}$ stored in the canonical coordinate system, we can use dot products:

$$u_b = \mathbf{u}\cdot\mathbf{b}; \quad v_b = \mathbf{v}\cdot\mathbf{b}; \quad w_b = \mathbf{w}\cdot\mathbf{b}$$

This works because we know that for *some* u_b, v_b, and w_b,

$$u_b\mathbf{u} + v_b\mathbf{v} + w_b\mathbf{w} = \mathbf{b},$$

and the dot product isolates the u_b coordinate:

$$\begin{aligned}\mathbf{u}\cdot\mathbf{b} &= u_b(\mathbf{u}\cdot\mathbf{u}) + v_b(\mathbf{u}\cdot\mathbf{v}) + w_b(\mathbf{u}\cdot\mathbf{w})\\ &= u_b\end{aligned}$$

This works because $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are orthonormal.

Using matrices to manage changes of coordinate systems is discussed in Sections 6.2.1 and 6.5.

2.4.6 Constructing a Basis from a Single Vector

Often we need an orthonormal basis that is aligned with a given vector. That is, given a vector $\mathbf{a}$, we want an orthonormal $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ such that $\mathbf{w}$ points in the same direction as $\mathbf{a}$ (Hughes & Möller, 1999), but we don't particularly care what $\mathbf{u}$ and $\mathbf{v}$ are. One vector isn't enough to uniquely determine the answer; we just need a robust procedure that will find any one of the possible bases.

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

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

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

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

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

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

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

An example of a situation where this construction is used is surface shading, where a basis aligned to the surface normal is needed but the rotation around the normal is often unimportant.

2.4.7 Constructing a Basis from Two Vectors

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

A common way to fully specify a frame is by providing two vectors $\mathbf{a}$ (which specifies $\mathbf{w}$) and $\mathbf{b}$ (which specifies $\mathbf{v}$). If the two vectors are known to be perpendicular it is a simple matter to construct the third vector by $\mathbf{u} = \mathbf{b} \times \mathbf{a}$.

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

To be sure that the resulting basis really is orthonormal, even if the input vectors weren't quite, a procedure much like the single-vector procedure is advisable:

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

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

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

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

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

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

2.4.8 Squaring Up a Basis

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

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

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

2.5 Curves and Surfaces

The geometry of curves, and especially surfaces, plays a central role in graphics, and here we review the basics of curves and surfaces in 2D and 3D space.

2.5.1 2D Implicit Curves

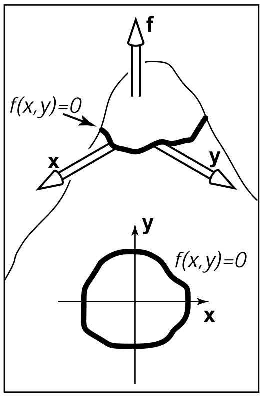

Figure 2.22. An implicit function *f*(*x*,*y*) = 0 can be thought of as a height field where *f* is the height (top). A path where the height is zero is the implicit curve (bottom).

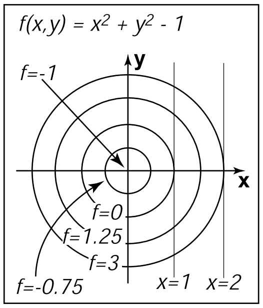

Figure 2.23. An implicit function *f*(*x*,*y*) = 0 can be thought of as a height field where *f* is the height (top). A path where the height is zero is the implicit curve (bottom).

Intuitively, a *curve* is a set of points that can be drawn on a piece of paper without lifting the pen. A common way to describe a curve is using an *implicit equation*. An implicit equation in two dimensions has the form

$$f(x, y) = 0.$$

The function $f(x, y)$ returns a real value. Points (x, y) where this value is zero are on the curve, and points where the value is non-zero are not on the curve. For example, let's say that $f(x, y)$ is

$$f(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2, \tag{2.9}$$

where (x_c, y_c) is a 2D point and r is a non-zero real number. If we take $f(x, y) = 0$, the points where this equality holds are on the circle with center (x_c, y_c) and radius r. The reason that this is called an "implicit" equation is that the points (x, y) on the curve cannot be immediately calculated from the equation and instead must be determined by solving the equation. Thus, the points on the curve are not generated by the equation *explicitly*, but they are buried somewhere *implicitly* in the equation.

It is interesting to note that f does have values for all (x, y). We can think of f as a terrain, with sea-level at $f = 0$ (Figure 2.22). The shore is the implicit curve. The value of f is the altitude. Another thing to note is that the curve partitions space into regions where $f > 0$, $f < 0$, and $f = 0$. So you evaluate f to decide whether a point is "inside" a curve. Note that $f(x, y) = c$ is a curve for any constant c, and $c = 0$ is just used as a convention. For example if $f(x, y) = x^2 + y^2 - 1$, varying c just gives a variety of circles centered at the origin (Figure 2.23).

We can compress our notation using vectors. If we have $\mathbf{c} = (x_c, y_c)$ and $\mathbf{p} = (x, y)$, then our circle with center $\mathbf{c}$ and radius r is defined by those position vectors that satisfy

$$(\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - r^2 = 0.$$

This equation, if expanded algebraically, will yield Equation (2.9), but it is easier to see that this is an equation for a circle by "reading" the equation geometrically. It reads, "points $\mathbf{p}$ on the circle have the following property: the vector from $\mathbf{c}$ to

$\mathbf{p}$ when dotted with itself has value r^2." Because a vector dotted with itself is just its own length squared, we could also read the equation as, "points $\mathbf{p}$ on the circle have the following property: the vector from $\mathbf{c}$ to $\mathbf{p}$ has squared length r^2."

Even better, is to observe that the squared length is just the squared distance from $\mathbf{c}$ to $\mathbf{p}$, which suggests the equivalent form

$$\|\mathbf{p} - \mathbf{c}\|^2 - r^2 = 0,$$

and, of course, this suggests

$$\|\mathbf{p} - \mathbf{c}\| - r = 0.$$

The above could be read "the points $\mathbf{p}$ on the circle are those a distance r from the center point $\mathbf{c}$," which is as good a definition of circle as any. This illustrates that the vector form of an equation often suggests more geometry and intuition than the equivalent full-blown Cartesian form with x and y. For this reason, it is usually advisable to use vector forms when possible. In addition, you can support a vector class in your code; the code is cleaner when vector forms are used. The vector-oriented equations are also less error prone in implementation: once you implement and debug vector types in your code, the cut-and-paste errors involving x, y, and z will go away. It takes a little while to get used to vectors in these equations, but once you get the hang of it, the payoff is large.

2.5.2 The 2D Gradient

If we think of the function $f(x, y)$ as a height field with height $= f(x, y)$, the *gradient* vector points in the direction of maximum upslope, i.e., straight uphill. The gradient vector $\nabla f(x, y)$ is given by

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right).$$

The gradient vector evaluated at a point on the implicit curve $f(x, y) = 0$ is perpendicular to the *tangent* vector of the curve at that point. This perpendicular vector is usually called the *normal vector* to the curve. In addition, since the gradient points uphill, it indicates the direction of the $f(x, y) > 0$ region.

In the context of height fields, the geometric meaning of partial derivatives and gradients is more visible than usual. Suppose that near the point (a, b), $f(x, y)$ is a plane (Figure 2.24). There is a specific uphill and downhill direction. At right angles to this direction is a direction that is level with respect to the plane. Any intersection between the plane and the $f(x, y) = 0$ plane will be in the direction that is level. Thus the uphill/downhill directions will be perpendicular to the line of intersection $f(x, y) = 0$. To see why the partial derivative has something to do

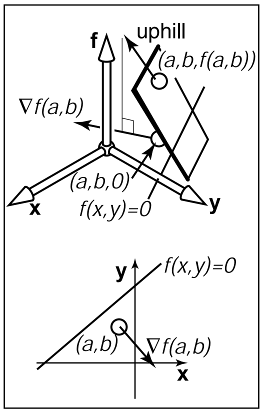

Figure 2.24. A surface height = *f*(*x*,*y*) is locally planar near (*x*,*y*) = (*a*,*b*). The gradient is a projection of the uphill direction onto the height = 0 plane.

with this, we need to visualize its geometric meaning. Recall that the conventional derivative of a 1D function $y = g(x)$ is

$$\frac{dy}{dx} \equiv \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{g(x + \Delta x) - g(x)}{\Delta x}. \tag{2.10}$$

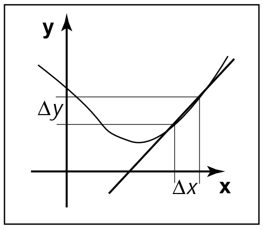

Figure 2.25. The derivative of a 1D function measures the slope of the line tangent to the curve.

This measures the *slope* of the *tangent* line to g (Figure 2.25).

The partial derivative is a generalization of the 1D derivative. For a 2D function $f(x, y)$, we can't take the same limit for x as in Equation (2.10), because f can change in many ways for a given change in x. However, if we hold y constant, we can define an analog of the derivative, called the *partial derivative* (Figure 2.26):

$$\frac{\partial f}{\partial x} \equiv \lim_{\Delta x \to 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}.$$

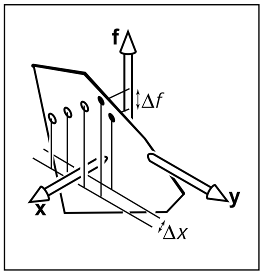

Figure 2.26. The partial derivative of a 2D function with respect to *f* must hold *y* constant to have a unique value, as shown by the dark point. The hollow points show other values of *f* that do not hold *y* constant.

Why is it that the partial derivatives of x and y are the components of the gradient vector? Again, there is more obvious insight in the geometry than in the algebra. In Figure 2.27, we see the vector $\mathbf{a}$ travels along a path where $\mathbf{f}$ does not change. Note that this is again at a small enough scale that the surface height $(x, y) = f(x, y)$ can be considered locally planar. From the figure, we see that the vector $\mathbf{a} = (\Delta x, \Delta y)$.

Because the uphill direction is perpendicular to $\mathbf{a}$, we know the dot product is equal to zero:

$$(\nabla f) \cdot \mathbf{a} \equiv (x_\nabla, y_\nabla) \cdot (x_a, y_a) = x_\nabla \Delta x + y_\nabla \Delta y = 0. \tag{2.11}$$

We also know that the change in f in the direction (x_a, y_a) equals zero:

$$\Delta f = \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y \equiv \frac{\partial f}{\partial x}x_a + \frac{\partial f}{\partial y}y_a = 0. \tag{2.12}$$

Given any vectors (x, y) and (x', y') that are perpendicular, we know that the angle between them is 90 degrees, and thus their dot product equals zero (recall that the dot product is proportional to the cosine of the angle between the two vectors). Thus, we have $xx' + yy' = 0$. Given (x, y), it is easy to construct valid vectors whose dot product with (x, y) equals zero, the two most obvious being $(y, -x)$ and $(-y, x)$; you can verify that these vectors give the desired zero dot product with (x, y). A generalization of this observation is that (x, y) is perpendicular to $k(y, -x)$ where k is any non-zero constant. This implies that

$$(x_a, y_a) = k\left(\frac{\partial f}{\partial y}, -\frac{\partial f}{\partial x}\right). \tag{2.13}$$

Combining Equations (2.11) and (2.13) gives

$$(x_\nabla, y_\nabla) = k'\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right),$$

where k' is any non-zero constant. By definition, "uphill" implies a positive change in f, so we would like $k' > 0$, and $k' = 1$ is a perfectly good convention.

As an example of the gradient, consider the implicit circle $x^2 + y^2 - 1 = 0$ with gradient vector $(2x, 2y)$, indicating that the outside of the circle is the positive region for the function $f(x, y) = x^2 + y^2 - 1$. Note that the length of the gradient vector can be different depending on the multiplier in the implicit equation. For example, the unit circle can be described by $Ax^2 + Ay^2 - A = 0$ for any non-zero A. The gradient for this curve is $(2Ax, 2Ay)$. This will be normal (perpendicular) to the circle, but will have a length determined by A. For $A > 0$, the normal will point outward from the circle, and for $A < 0$, it will point inward. This switch from outward to inward is as it should be, since the positive region switches inside the circle. In terms of the height-field view, $h = Ax^2 + Ay^2 - A$, and the circle is at zero altitude. For $A > 0$, the circle encloses a depression, and for $A < 0$, the circle encloses a bump. As A becomes more negative, the bump increases in height, but the $h = 0$ circle doesn't change. The direction of maximum uphill doesn't change, but the slope increases. The length of the gradient reflects this change in degree of the slope. So intuitively, you can think of the gradient's direction as pointing uphill and its magnitude as measuring how uphill the slope is.

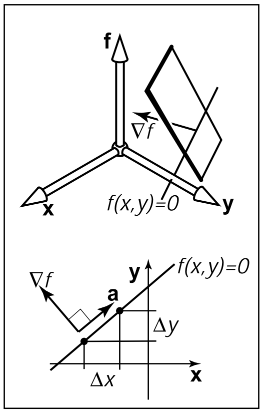

Figure 2.27. The vector **a** points in a direction where f has no change and is thus perpendicular to the gradient vector ∇f.

Implicit 2D Lines

The familiar "slope-intercept" form of the line is

$$y = mx + b. \tag{2.14}$$

This can be converted easily to implicit form (Figure 2.28):

$$y - mx - b = 0. \tag{2.15}$$

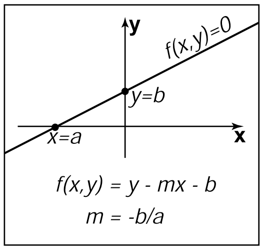

Figure 2.28. A 2D line can be described by the equation $y - mx - b = 0$.

Here m is the "slope" (ratio of rise to run) and b is the y value where the line crosses the y-axis, usually called the *y-intercept* . The line also partitions the 2D plane, but here "inside" and "outside" might be more intuitively called "over" and "under."

Because we can multiply an implicit equation by any constant without changing the points where it is zero, $kf(x, y) = 0$ is the same curve for any non-zero k. This allows several implicit forms for the same line, for example,

$$2y - 2mx - 2b = 0.$$

One reason the slope-intercept form is sometimes awkward is that it can't represent some lines such as $x = 0$ because m would have to be infinite. For this reason, a more general form is often useful:

$$Ax + By + C = 0, \tag{2.16}$$

for real numbers A, B, C.

Suppose we know two points on the line, (x_0, y_0) and (x_1, y_1). What A, B, and C describe the line through these two points? Because these points lie on the line, they must both satisfy Equation (2.16):

$$Ax_0 + By_0 + C = 0,$$
$$Ax_1 + By_1 + C = 0.$$

Unfortunately we have two equations and *three* unknowns: A, B, and C. This problem arises because of the arbitrary multiplier we can have with an implicit equation. We could set $C = 1$ for convenience:

$$Ax + By + 1 = 0,$$

but we have a similar problem to the infinite slope case in slope-intercept form: lines through the origin would need to have $A(0) + B(0) + 1 = 0$, which is a contradiction. For example, the equation for a 45-degree line through the origin can be written $x - y = 0$, or equally well $y - x = 0$, or even $17y - 17x = 0$, but it cannot be written in the form $Ax + By + 1 = 0$.

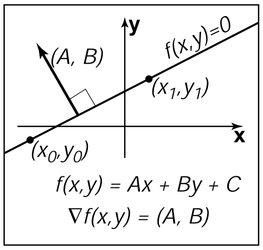

Figure 2.29. The gradient vector (A, B) is perpendicular to the implicit line $Ax + By + C = 0$.

Whenever we have such pesky algebraic problems, we try to solve the problems using geometric intuition as a guide. One tool we have, as discussed in Section 2.5.2, is the gradient. For the line $Ax + By + C = 0$, the gradient vector is (A, B). This vector is perpendicular to the line (Figure 2.29), and points to the side of the line where $Ax + By + C$ is positive. Given two points on the line (x_0, y_0) and (x_1, y_1), we know that the vector between them points in the same direction as the line. This vector is just $(x_1 - x_0, y_1 - y_0)$, and because it is parallel to the line, it must also be perpendicular to the gradient vector (A, B). Recall that there are an infinite number of (A, B, C) that describe the line because of the arbitrary scaling property of implicits. We want any one of the valid (A, B, C).

We can start with any (A, B) perpendicular to $(x_1 - x_0, y_1 - y_0)$. Such a vector is just $(A, B) = (y_0 - y_1, x_1 - x_0)$ by the same reasoning as in Section 2.5.2. This means that the equation of the line through (x_0, y_0) and (x_1, y_1) is

$$(y_0 - y_1)x + (x_1 - x_0)y + C = 0. \tag{2.17}$$

Now we just need to find C. Because (x_0, y_0) and (x_1, y_1) are on the line, they must satisfy Equation (2.17). We can plug either value in and solve for C. Doing this for (x_0, y_0) yields $C = x_0y_1 - x_1y_0$, and thus the full equation for the line is

$$(y_0 - y_1)x + (x_1 - x_0)y + x_0y_1 - x_1y_0 = 0. \tag{2.18}$$

Again, this is one of infinitely many valid implicit equations for the line through two points, but this form has no division operation and thus no numerically degenerate cases for points with finite Cartesian coordinates. A nice thing about Equation (2.18) is that we can always convert to the slope-intercept form (when it exists) by moving the non-y terms to the right-hand side of the equation and dividing by the multiplier of the y term:

$$y = \frac{y_1 - y_0}{x_1 - x_0}x + \frac{x_1 y_0 - x_0 y_1}{x_1 - x_0}.$$

An interesting property of the implicit line equation is that it can be used to find the signed distance from a point to the line. The value of $Ax + By + C$ is proportional to the distance from the line (Figure 2.30). As shown in Figure 2.31, the distance from a point to the line is the length of the vector $k(A, B)$, which is

$$\text{distance} = k\sqrt{A^2 + B^2}. \tag{2.19}$$

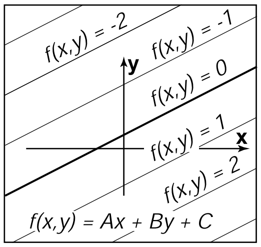

Figure 2.30. The value of the implicit function *f*(*x,y*) = *Ax* + *By* + *C* is a constant times the signed distance from *Ax* + *By* + *C* = 0.

For the point $(x, y) + k(A, B)$, the value of $f(x, y) = Ax + By + C$ is

$$\begin{aligned} f(x + kA, y + kB) &= Ax + kA^2 + By + kB^2 + C \\ &= k(A^2 + B^2). \end{aligned} \tag{2.20}$$

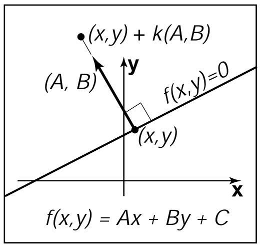

Figure 2.31. The vector *k*(*A,B*) connects a point (*x,y*) on the line closest to a point not on the line. The distance is proportional to *k*.

The simplification in that equation is a result of the fact that we know (x, y) is on the line, so $Ax + By + C = 0$. From Equations (2.19) and (2.20), we can see that the signed distance from line $Ax + By + C = 0$ to a point (a, b) is

$$\text{distance} = \frac{f(a, b)}{\sqrt{A^2 + B^2}}.$$

Here "signed distance" means that its magnitude (absolute value) is the geometric distance, but on one side of the line, distances are positive and on the other they are negative. You can choose between the equally valid representations $f(x, y) = 0$ and $-f(x, y) = 0$ if your problem has some reason to prefer a particular side being positive. Note that if (A, B) is a unit vector, then $f(a, b)$ is the signed distance. We can multiply Equation (2.18) by a constant that ensures that (A, B) is a unit vector:

$$\begin{aligned} f(x, y) = \frac{y_0 - y_1}{\sqrt{(x_1 - x_0)^2 + (y_0 - y_1)^2}}x + \frac{x_1 - x_0}{\sqrt{(x_1 - x_0)^2 + (y_0 - y_1)^2}}y \\ + \frac{x_0 y_1 - x_1 y_0}{\sqrt{(x_1 - x_0)^2 + (y_0 - y_1)^2}} = 0. \end{aligned} \tag{2.21}$$

Note that evaluating $f(x, y)$ in Equation (2.21) directly gives the signed distance, but it does require a square root to set up the equation. Implicit lines will turn out to be very useful for triangle rasterization (Section 8.1.2). Other forms for 2D lines are discussed in Chapter 14.

Implicit Quadric Curves

In the previous section we saw that a linear function $f(x, y)$ gives rise to an implicit line $f(x, y) = 0$. If f is instead a quadratic function of x and y, with the general form

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0,$$

the resulting implicit curve is called a quadric. Two-dimensional quadric curves include ellipses and hyperbolas, as well as the special cases of parabolas, circles, and lines.

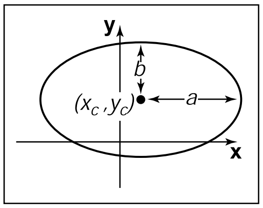

Figure 2.32. The ellipse with center (x_c, y_c) and semi-axes of length a and b.

Examples of quadric curves include the circle with center (x_c, y_c) and radius r:

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$$

Try setting $a = b = r$ in the ellipse equation and compare to the circle equation.

where (x_c, y_c) is the center of the ellipse, and a and b are the minor and major semi-axes (Figure 2.32).and axis-aligned ellipses of the form

$$\frac{(x - x_c)^2}{a^2} + \frac{(y - y_c)^2}{b^2} - 1 = 0.$$

2.5.3 3D Implicit Surfaces

Just as implicit equations can be used to define curves in 2D, they can be used to define surfaces in 3D. As in 2D, implicit equations *implicitly* define a set of points that are on the surface

$$f(x, y, z) = 0.$$

Any point (x, y, z) that is on the surface results in zero when given as an argument to f. Any point not on the surface results in some number other than zero. You can check whether a point is on the surface by evaluating f, or you can check which side of the surface the point lies on by looking at the sign of f, but you cannot always explicitly construct points on the surface. Using vector notation, we will write such functions of $\mathbf{p} = (x, y, z)$ as

$$f(\mathbf{p}) = 0.$$

2.5.4 Surface Normal to an Implicit Surface

A surface normal (which is needed for lighting computations, among other things) is a vector perpendicular to the surface. Each point on the surface may have a different normal vector. In the same way that the gradient provides a normal to an implicit curve in 2D, the surface normal at a point $\mathbf{p}$ on an implicit surface is given by the gradient of the implicit function

$$\mathbf{n} = \nabla f(\mathbf{p}) = \left(\frac{\partial f(\mathbf{p})}{\partial x}, \frac{\partial f(\mathbf{p})}{\partial y}, \frac{\partial f(\mathbf{p})}{\partial z} \right).$$

The reasoning is the same as for the 2D case: the gradient points in the direction of fastest increase in f, which is perpendicular to the direction's tangent to the surface, in which f remains constant. The gradient vector points toward the side of the surface where $f(\mathbf{p}) > 0$, which we may think of as "into" the surface or "out from" the surface in a given context. If the particular form of f creates inward facing gradients and outward facing gradients are desired, the surface $-f(\mathbf{p}) = 0$ is the same as surface $f(\mathbf{p}) = 0$ but has directionally reversed gradients, i.e., $-\nabla f(\mathbf{p}) = \nabla(-f(\mathbf{p}))$.

2.5.5 Implicit Planes

As an example, consider the infinite plane through point $\mathbf{a}$ with surface normal $\mathbf{n}$. The implicit equation to describe this plane is given by

$$(\mathbf{p} - \mathbf{a}) \cdot \mathbf{n} = 0. \tag{2.22}$$

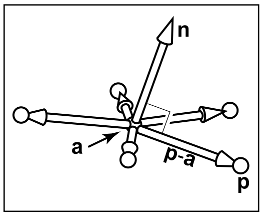

Figure 2.33. Any of the points **p** shown are in the plane with normal vector **n** that includes point **a** if Equation (2.2) is satisfied.

Note that $\mathbf{a}$ and $\mathbf{n}$ are known quantities. The point $\mathbf{p}$ is any unknown point that satisfies the equation. In geometric terms this equation says "the vector from $\mathbf{a}$ to $\mathbf{p}$ is perpendicular to the plane normal." If $\mathbf{p}$ were not in the plane, then $(\mathbf{p} - \mathbf{a})$ would not make a right angle with $\mathbf{n}$ (Figure 2.33).

Sometimes we want the implicit equation for a plane through points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$. The normal to this plane can be found by taking the cross product of any two vectors in the plane. One such cross product is

$$\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}).$$

This allows us to write the implicit plane equation:

$$(\mathbf{p} - \mathbf{a}) \cdot ((\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})) = 0. \tag{2.23}$$

A geometric way to read this equation is that the volume of the parallelepiped defined by $\mathbf{p} - \mathbf{a}$, $\mathbf{b} - \mathbf{a}$, and $\mathbf{c} - \mathbf{a}$ is zero, i.e., they are coplanar. This can only be true if $\mathbf{p}$ is in the same plane as $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$. The full-blown Cartesian representation for this is given by the determinant (this is discussed in more detail in Section 5.3):

$$\begin{vmatrix} x - x_a & y - y_a & z - z_a \\ x_b - x_a & y_b - y_a & z_b - z_a \\ x_c - x_a & y_c - y_a & z_c - z_a \end{vmatrix} = 0. \tag{2.24}$$

The determinant can be expanded (see Section 5.3 for the mechanics of expanding determinants) to the bloated form with many terms.

Equations (2.23) and (2.24) are equivalent, and comparing them is instructive. Equation (2.23) is easy to interpret geometrically and will yield efficient code. In addition, it is relatively easy to avoid a typographic error that compiles into incorrect code if it takes advantage of debugged cross and dot product code. Equation (2.24) is also easy to interpret geometrically and will be efficient provided an efficient 3 $\times$ 3 determinant function is implemented. It is also easy to implement without a typo if a function *determinant*$(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is available. It will be especially easy for others to read your code if you rename the *determinant* function *volume*. So both Equations (2.23) and (2.24) map well into code. The full expansion of either equation into x-, y-, and z-components is likely to generate typos. Such typos are likely to compile and, thus, be especially pesky. This is an excellent example of clean math generating clean code and bloated math generating bloated code.

3D Quadric Surfaces

Just as quadratic polynomials in two variables define quadric curves in 2D, quadratic polynomials in x, y, and z define *quadric surfaces* in 3D. For instance, a sphere can be written as

$$f(\mathbf{p}) = (\mathbf{p} - \mathbf{c})^2 - r^2 = 0,$$

and an axis-aligned ellipsoid may be written as

$$f(\mathbf{p}) = \frac{(x - x_c)^2}{a^2} + \frac{(y - y_c)^2}{b^2} + \frac{(z - z_c)^2}{c^2} - 1 = 0.$$

3D Curves from Implicit Surfaces

One might hope that an implicit 3D curve could be created with the form $f(\mathbf{p}) = 0$. However, all such curves are just degenerate surfaces and are rarely useful in practice. A 3D curve can be constructed from the intersection of two simultaneous implicit equations:

$$\begin{aligned} f(\mathbf{p}) &= 0, \\ g(\mathbf{p}) &= 0. \end{aligned}$$

For example, a 3D line can be formed from the intersection of two implicit planes. Typically, it is more convenient to use parametric curves instead; they are discussed in the following sections.

2.5.6 2D Parametric Curves

A *parametric* curve is controlled by a single *parameter* that can be considered a sort of index that moves continuously along the curve. Such curves have the form

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g(t) \\ h(t) \end{bmatrix}.$$

Here (x, y) is a point on the curve, and t is the parameter that influences the curve. For a given t, there will be some point determined by the functions g and h. For continuous g and h, a small change in t will yield a small change in x and y. Thus, as t continuously changes, points are swept out in a continuous curve. This is a nice feature because we can use the parameter t to explicitly construct points on the curve. Often we can write a parametric curve in vector form,

$$\mathbf{p} = f(t),$$

where f is a vector-valued function, $f : \mathbb{R} \mapsto \mathbb{R}^2$. Such vector functions can generate very clean code, so they should be used when possible.

We can think of the curve with a position as a function of time. The curve can go anywhere and could loop and cross itself. We can also think of the curve as having a velocity at any point. For example, the point $\mathbf{p}(t)$ is traveling slowly near $t = -2$ and quickly between $t = 2$ and $t = 3$. This type of "moving point" vocabulary is often used when discussing parametric curves even when the curve is not describing a moving point.

2D Parametric Lines

A parametric line in 2D that passes through points $\mathbf{p}_0 = (x_0, y_0)$ and $\mathbf{p}_1 = (x_1, y_1)$ can be written

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{bmatrix}.$$

Because the formulas for x and y have such similar structure, we can use the vector form for $\mathbf{p} = (x, y)$ (Figure 2.34):

$$\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0).$$

You can read this in geometric form as: "start at point $\mathbf{p}_0$ and go some distance toward $\mathbf{p}_1$ determined by the parameter t." A nice feature of this form is that $\mathbf{p}(0) = \mathbf{p}_0$ and $\mathbf{p}(1) = \mathbf{p}_1$. Since the point changes linearly with t, the value of t between $\mathbf{p}_0$ and $\mathbf{p}_1$ measures the fractional distance between the points. Points

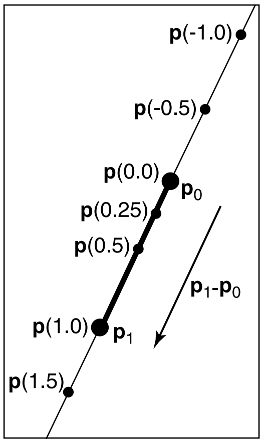

Figure 2.34. A 2D parametric line through $\mathbf{p}_0$ and $\mathbf{p}_1$. The line segment defined by $t \in [0,1]$ is shown in bold.

with $t < 0$ are to the "far" side of $\mathbf{p}_0$, and points with $t > 1$ are to the "far" side of $\mathbf{p}_1$.

Parametric lines can also be described as just a point $\mathbf{o}$ and a vector $\mathbf{d}$:

$$\mathbf{p}(t) = \mathbf{o} + t(\mathbf{d}).$$

When the vector $\mathbf{d}$ has unit length, the line is *arc-length parameterized*. This means t is an exact measure of distance along the line. Any parametric curve can be arc-length parameterized, which is obviously a very convenient form, but not all can be converted analytically.

2D Parametric Circles

A circle with center (x_c, y_c) and radius r has a parametric form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + r\cos\phi \\ y_c + r\sin\phi \end{bmatrix}.$$

To ensure that there is a unique parameter ϕ for every point on the curve, we can restrict its domain: $\phi \in [0, 2\pi)$ or $\phi \in (-\pi, \pi]$ or any other half open interval of length 2π.

An axis-aligned ellipse can be constructed by scaling the x and y parametric equations separately:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + a\cos\phi \\ y_c + b\sin\phi \end{bmatrix}.$$

2.5.7 3D Parametric Curves

A 3D parametric curve operates much like a 2D parametric curve:

$$\begin{aligned} x &= f(t), \\ y &= g(t), \\ z &= h(t). \end{aligned}$$

For example, a spiral around the z-axis is written as:

$$\begin{aligned} x &= \cos t, \\ y &= \sin t, \\ z &= t. \end{aligned}$$

As with 2D curves, the functions f, g, and h are defined on a domain $D \subset \mathbb{R}$ if we want to control where the curve starts and ends. In vector form we can write

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{p}(t).$$

The parametric curve is the range of **p**: $\mathbb{R} \to \mathbb{R}^3$.

In this chapter we only discuss 3D parametric lines in detail. General 3D parametric curves are discussed more extensively in Chapter 15.

3D Parametric Lines

A 3D parametric line can be written as a straightforward extension of the 2D parametric line, e.g.,

$$\begin{aligned} x &= 2 + 7t, \\ y &= 1 + 2t, \\ z &= 3 - 5t. \end{aligned}$$

This is cumbersome and does not translate well to code variables, so we will write it in vector form:

$$\mathbf{p} = \mathbf{o} + t\mathbf{d},$$

where, for this example, $\mathbf{o}$ and $\mathbf{d}$ are given by

$$\begin{aligned} \mathbf{o} &= (2, 1, \quad 3), \\ \mathbf{d} &= (7, 2, -5). \end{aligned}$$

Note that this is very similar to the 2D case. The way to visualize this is to imagine that the line passes though $\mathbf{o}$ and is parallel to $\mathbf{d}$. Given any value of t, you get some point $\mathbf{p}(t)$ on the line. For example, at $t = 2$, $p(t) = (2, 1, 3) + 2(7, 2, -5) = (16, 5, -7)$. This general concept is the same as for two dimensions (Figure 2.30).

As in 2D, a *line segment* can be described by a 3D parametric line and an interval $t \in [t_a, t_b]$. The line segment between two points $\mathbf{a}$ and $\mathbf{b}$ is given by $\mathbf{p}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$ with $t \in [0, 1]$. Here $\mathbf{p}(0) = \mathbf{a}$, $\mathbf{p}(1) = \mathbf{b}$, and $\mathbf{p}(0.5) = (\mathbf{a} + \mathbf{b})/2$, the midpoint between $\mathbf{a}$ and $\mathbf{b}$.

A *ray*, or *half-line*, is a 3D parametric line with a half-open interval, usually $[0, \infty)$. From now on we will refer to all lines, line segments, and rays as "rays." This is sloppy, but corresponds to common usage and makes the discussion simpler.

2.5.8 3D Parametric Surfaces

The parametric approach can be used to define surfaces in 3D space in much the same way we define curves, except that there are two parameters to address the two-dimensional area of the surface. These surfaces have the form

$$\begin{aligned} x &= f(u, v), \\ y &= g(u, v), \\ z &= h(u, v). \end{aligned}$$

The parametric surface is the range of the function **p**: $\mathbb{R}^2 \rightarrow \mathbb{R}^3$.

or, in vector form,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{p}(u, v).$$

Pretend for the sake of argument that the Earth is exactly spherical.

The θ and ϕ here may or may not seem reversed depending on your background; the use of these symbols varies across disciplines. In this book we will always assume the meaning of θ and ϕ used in Equation (2.25) and depicted in Figure 2.35.

Example. For example, a point on the surface of the Earth can be described by the two parameters longitude and latitude. If we define the origin to be at the center of the earth, and let r be the radius of the Earth, then a spherical coordinate system centered at the origin (Figure 2.35), lets us derive the parametric equations

$$\begin{aligned} x &= r \cos\phi \sin\theta, \\ y &= r \sin\phi \sin\theta, \\ z &= r \cos\theta. \end{aligned} \tag{2.25}$$

Ideally, we'd like to write this in vector form, but it isn't feasible for this particular parametric form.

We would also like to be able to find the (θ, ϕ) for a given (x, y, z). If we assume that $\phi \in (-\pi, \pi]$ this is easy to do using the *atan2* function from Equation (2.2):

$$\begin{aligned} \theta &= \text{acos}(z/\sqrt{x^2 + y^2 + z^2}), \\ \phi &= \text{atan2}(y, x). \end{aligned} \tag{2.26}$$

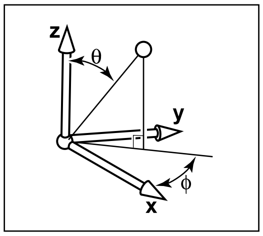

Figure 2.35. The geometry for spherical coordinates.

With implicit surfaces, the derivative of the function f gave us the surface normal. With parametric surfaces, the derivatives of $\mathbf{p}$ also give information about the surface geometry.

Consider the function $\mathbf{q}(t) = \mathbf{p}(t, v_0)$. This function defines a parametric curve obtained by varying u while holding v fixed at the value v_0. This curve, called an *isoparametric curve* (or sometimes "isoparm" for short) lies in the surface. The derivative of $\mathbf{q}$ gives a vector tangent to the curve, and since the curve lies in the surface the vector $\mathbf{q}'$ also lies in the surface. Since it was obtained by varying one argument of $\mathbf{p}$, the vector $\mathbf{q}'$ is the partial derivative of $\mathbf{p}$ with respect to u, which we'll denote $\mathbf{p}_u$. A similar argument shows that the partial derivative $\mathbf{p}_v$ gives the tangent to the isoparametric curves for constant u, which is a second tangent vector to the surface.

The derivative of $\mathbf{p}$, then, gives two tangent vectors at any point on the surface. The normal to the surface may be found by taking the cross product of these vectors: since both are tangent to the surface, their cross product, which is perpendicular to both tangents, is normal to the surface. The right-hand rule for cross products provides a way to decide which side is the front, or outside, of the surface; we will use the convention that the vector

$$\mathbf{n} = \mathbf{p}_u \times \mathbf{p}_v$$

points toward the outside of the surface.

2.5.9 Summary of Curves and Surfaces

Implicit curves in 2D or surfaces in 3D are defined by scalar-valued functions of two or three variables, $f : \mathbb{R}^2 \to \mathbb{R}$ or $f : \mathbb{R}^3 \to \mathbb{R}$, and the surface consists of all points where the function is zero:

$$S = \{\mathbf{p} \,|\, f(\mathbf{p}) = 0\}.$$

Parametric curves in 2D or 3D are defined by vector-valued functions of one variable, $\mathbf{p} : D \subset \mathbb{R} \to \mathbb{R}^2$ or $\mathbf{p} : D \subset \mathbb{R} \to \mathbb{R}^3$, and the curve is swept out as t varies over all of D:

$$S = \{\mathbf{p}(t) \,|\, t \in D\}.$$

Parametric surfaces in 3D are defined by vector-valued functions of two variables, $\mathbf{p} : D \subset \mathbb{R}^2 \to \mathbb{R}^3$, and the surface consists of the images of all points (u, v) in the domain:

$$S = \{\mathbf{p}(t) \,|\, (u, v) \in D\}.$$

For implicit curves and surfaces, the normal vector is given by the derivative of f (the gradient), and the tangent vector (for a curve) or vectors (for a surface) can be derived from the normal by constructing a basis.

For parametric curves and surfaces, the derivative of $\mathbf{p}$ gives the tangent vector (for a curve) or vectors (for a surface), and the normal vector can be derived from the tangents by constructing a basis.

2.6 Linear Interpolation

Perhaps the most common mathematical operation in graphics is *linear interpolation*. We have already seen an example of linear interpolation of position to form line segments in 2D and 3D, where two points $\mathbf{a}$ and $\mathbf{b}$ are associated with a parameter t to form the line $\mathbf{p} = (1-t)\mathbf{a} + t\mathbf{b}$. This is *interpolation* because $\mathbf{p}$ goes through $\mathbf{a}$ and $\mathbf{b}$ exactly at $t = 0$ and $t = 1$. It is *linear* interpolation because the weighting terms t and $1 - t$ are linear polynomials of t.

Another common linear interpolation is among a set of positions on the x-axis: x_0, x_1, . . ., x_n, and for each x_i we have an associated height, y_i. We want to create a continuous function $y = f(x)$ that interpolates these positions, so that f goes through every data point, i.e., $f(x_i) = y_i$. For linear interpolation, the points (x_i, y_i) are connected by straight line segments. It is natural to use parametric line equations for these segments. The parameter t is just the fractional distance between x_i and x_{i+1}:

$$f(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i). \tag{2.27}$$

Because the weighting functions are linear polynomials of x, this is linear interpolation.

The two examples above have the common form of linear interpolation. We create a variable t that varies from 0 to 1 as we move from data item A to data item B. Intermediate values are just the function $(1 - t)A + tB$. Notice that Equation (2.27) has this form with

$$t = \frac{x - x_i}{x_{i+1} - x_i}.$$

2.7 Triangles

Triangles in both 2D and 3D are the fundamental modeling primitive in many graphics programs. Often information such as color is tagged onto triangle vertices, and this information is interpolated across the triangle. The coordinate system that makes such interpolation straightforward is called *barycentric coordinates*; we will develop these from scratch. We will also discuss 2D triangles, which must be understood before we can draw their pictures on 2D screens.

2.7.1 2D Triangles

If we have a 2D triangle defined by 2D points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, we can first find its area:

$$\begin{aligned} \text{area} &= \frac{1}{2}\begin{vmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{vmatrix} \\ &= \frac{1}{2}\left(x_a y_b + x_b y_c + x_c y_a - x_a y_c - x_b y_a - x_c y_b\right). \end{aligned} \tag{2.28}$$

The derivation of this formula can be found in Section 5.3. This area will have a positive sign if the points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are in counterclockwise order and a negative sign, otherwise.

Often in graphics, we wish to assign a property, such as color, at each triangle vertex and smoothly interpolate the value of that property across the triangle. There are a variety of ways to do this, but the simplest is to use *barycentric* coordinates. One way to think of barycentric coordinates is as a non-orthogonal coordinate system as was discussed briefly in Section 2.4.2. Such a coordinate system is shown in Figure 2.36, where the coordinate origin is $\mathbf{a}$ and the vectors from $\mathbf{a}$ to $\mathbf{b}$ and $\mathbf{c}$ are the basis vectors. With that origin and those basis vectors, any point $\mathbf{p}$ can be written as

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}). \tag{2.29}$$

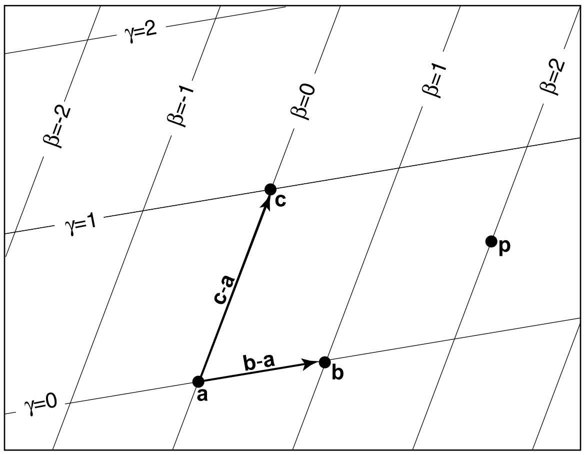

Figure 2.36. A 2D triangle with vertices **a**, **b**, **c** can be used to set up a non-orthogonal coordinate system with origin **a** and basis vectors (**b** – **a**) and (**c** – **a**). A point is then represented by an ordered pair (β, γ). For example, the point **p** = (2.0, 0.5), i.e., **p** = **a** + 2.0 (**b** – **a**) + 0.5 (**c** – **a**).

Note that we can reorder the terms in Equation (2.29) to get

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

Often people define a new variable α to improve the symmetry of the equations:

$$\alpha \equiv 1 - \beta - \gamma,$$

which yields the equation

$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}, \tag{2.30}$$

with the constraint that

$$\alpha + \beta + \gamma = 1. \tag{2.31}$$

Barycentric coordinates seem like an abstract and unintuitive construct at first, but they turn out to be powerful and convenient. You may find it useful to think of how street addresses would work in a city where there are two sets of parallel streets, but where those sets are not at right angles. The natural system would essentially be barycentric coordinates, and you would quickly get used to them. Barycentric coordinates are defined for all points on the plane. A particularly nice feature of barycentric coordinates is that a point $\mathbf{p}$ is inside the triangle formed by $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ if and only if

$$\begin{aligned} 0 < \alpha < 1, \\ 0 < \beta < 1, \\ 0 < \gamma < 1. \end{aligned}$$

If one of the coordinates is zero and the other two are between zero and one, then you are on an edge. If two of the coordinates are zero, then the other is one, and you are at a vertex. Another nice property of barycentric coordinates is that Equation (2.30) in effect mixes the coordinates of the three vertices in a smooth way. The same mixing coefficients (α, β, γ) can be used to mix other properties, such as color, as we will see in the next chapter.

Given a point $\mathbf{p}$, how do we compute its barycentric coordinates? One way is to write Equation (2.29) as a linear system with unknowns β and γ, solve, and set $\alpha = 1 - \beta - \gamma$. That linear system is

$$\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x_p - x_a \\ y_p - y_a \end{bmatrix}. \tag{2.32}$$

Although it is straightforward to solve Equation (2.32) algebraically, it is often fruitful to compute a direct geometric solution.

One geometric property of barycentric coordinates is that they are the signed scaled distance from the lines through the triangle sides, as is shown for β in Figure 2.37. Recall from Section 2.5.2 that evaluating the equation $f(x, y)$ for the line $f(x, y) = 0$ returns the scaled signed distance from (x, y) to the line. Also recall that if $f(x, y) = 0$ is the equation for a particular line, so is $kf(x, y) = 0$ for any non-zero k. Changing k scales the distance and controls which side of the line has positive signed distance, and which negative. We would like to choose k such that, for example, $kf(x, y) = \beta$. Since k is only one unknown, we can force this with one constraint, namely that at point $\mathbf{b}$ we know $\beta = 1$. So if the line $f_{ac}(x, y) = 0$ goes through both $\mathbf{a}$ and $\mathbf{c}$, then we can compute β for a point (x, y) as follows:

$$\beta = \frac{f_{ac}(x, y)}{f_{ac}(x_b, y_b)}, \tag{2.33}$$

and we can compute γ and α in a similar fashion. For efficiency, it is usually wise to compute only two of the barycentric coordinates directly and to compute the third using Equation (2.31).

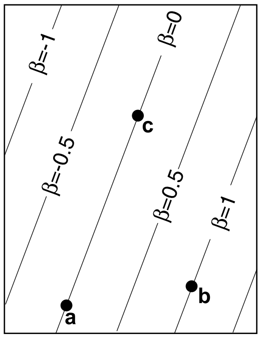

Figure 2.37. The barycentric coordinate β is the signed scaled distance from the line through **a** and **c**.

To find this "ideal" form for the line through $\mathbf{p}_0$ and $\mathbf{p}_1$, we can first use the technique of Section 2.5.2 to find *some* valid implicit lines through the vertices. Equation (2.18) gives us

$$f_{ab}(x, y) \equiv (y_a - y_b)x + (x_b - x_a)y + x_a y_b - x_b y_a = 0.$$

Note that $f_{ab}(x_c, y_c)$ probably does not equal one, so it is probably not the ideal form we seek. By dividing through by $f_{ab}(x_c, y_c)$ we get

$$\gamma = \frac{(y_a - y_b)x + (x_b - x_a)y + x_a y_b - x_b y_a}{(y_a - y_b)x_c + (x_b - x_a)y_c + x_a y_b - x_b y_a}.$$

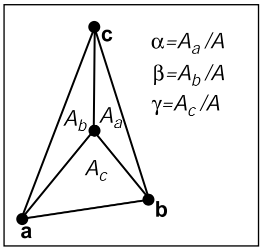

Figure 2.38. The barycentric coordinates are proportional to the areas of the three subtriangles shown.

The presence of the division might worry us because it introduces the possibility of divide-by-zero, but this cannot occur for triangles with areas that are not near zero. There are analogous formulas for α and β, but typically only one is needed:

$$\begin{aligned} \beta &= \frac{(y_a - y_c)x + (x_c - x_a)y + x_a y_c - x_c y_a}{(y_a - y_c)x_b + (x_c - x_a)y_b + x_a y_c - x_c y_a}, \\ \alpha &= 1 - \beta - \gamma. \end{aligned}$$

Another way to compute barycentric coordinates is to compute the areas A_a, A_b, and A_c, of subtriangles as shown in Figure 2.38. Barycentric coordinates obey the rule

$$\begin{aligned} \alpha &= A_a/A, \\ \beta &= A_b/A, \\ \gamma &= A_c/A, \end{aligned} \tag{2.34}$$

where A is the area of the triangle. Note that $A = A_a + A_b + A_c$, so it can be computed with two additions rather than a full area formula. This rule still holds for points outside the triangle if the areas are allowed to be signed. The reason for this is shown in Figure 2.39. Note that these are signed areas and will be computed correctly as long as the same signed area computation is used for both A and the subtriangles A_a, A_b, and A_c.

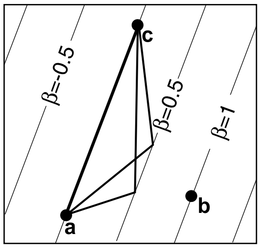

Figure 2.39. The area of the two triangles shown is base times height and are thus the same, as is any triangle with a vertex on the $\beta = 0.5$ line. The height and thus the area is proportional to β.

2.7.2 3D Triangles

One wonderful thing about barycentric coordinates is that they extend almost transparently to 3D. If we assume the points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are 3D, then we can still use the representation

$$\mathbf{p} = (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}.$$

Now, as we vary β and γ, we sweep out a plane.

The normal vector to a triangle can be found by taking the cross product of any two vectors in the plane of the triangle (Figure 2.40). It is easiest to use two of the three edges as these vectors, for example,

$$\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}). \tag{2.35}$$

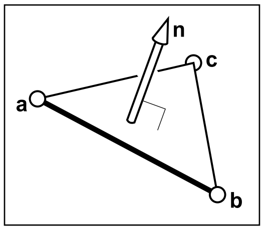

Figure 2.40. The normal vector of the triangle is perpendicular to all vectors in the plane of the triangle, and thus perpendicular to the edges of the triangle.

Note that this normal vector is not necessarily of unit length, and it obeys the right-hand rule of cross products.

The area of the triangle can be found by taking the length of the cross product:

$$\text{area} = \frac{1}{2}\|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})\|. \tag{2.36}$$

Note that this is *not* a signed area, so it cannot be used directly to evaluate barycentric coordinates. However, we can observe that a triangle with a "clockwise" vertex order will have a normal vector that points in the opposite direction to the normal of a triangle in the same plane with a "counterclockwise" vertex order. Recall that

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \; \|\mathbf{b}\| \;\; \cos\phi,$$

where ϕ is the angle between the vectors. If $\mathbf{a}$ and $\mathbf{b}$ are parallel, then $\cos\phi = \pm 1$, and this gives a test of whether the vectors point in the same or opposite directions.

This, along with Equations (2.34), (2.35), and (2.36) suggest the formulas

$$\alpha = \frac{\mathbf{n} \cdot \mathbf{n}_a}{\|\mathbf{n}\|^2},$$

$$\beta = \frac{\mathbf{n} \cdot \mathbf{n}_b}{\|\mathbf{n}\|^2},$$

$$\gamma = \frac{\mathbf{n} \cdot \mathbf{n}_c}{\|\mathbf{n}\|^2},$$

where $\mathbf{n}$ is Equation (2.35) evaluated with vertices $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$; $\mathbf{n}_a$ is Equation (2.35) evaluated with vertices $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{p}$, and so on, i.e.,

$$\begin{aligned}\mathbf{n}_a &= (\mathbf{c} - \mathbf{b}) \times (\mathbf{p} - \mathbf{b}),\\ \mathbf{n}_b &= (\mathbf{a} - \mathbf{c}) \times (\mathbf{p} - \mathbf{c}),\\ \mathbf{n}_c &= (\mathbf{b} - \mathbf{a}) \times (\mathbf{p} - \mathbf{a}).\end{aligned} \tag{2.37}$$

Frequently Asked Questions

• Why isn't there vector division?

It turns out that there is no "nice" analogy of division for vectors. However, it is possible to motivate the quaternions by examining this questions in detail (see Hoffman's book referenced in the chapter notes).

• Is there something as clean as barycentric coordinates for polygons with more than three sides?

Unfortunately there is not. Even convex quadrilaterals are much more complicated. This is one reason triangles are such a common geometric primitive in graphics.

• Is there an implicit form for 3D lines?

No. However, the intersection of two 3D planes defines a 3D line, so a 3D line can be described by two simultaneous implicit 3D equations.

Notes

The history of vector analysis is particularly interesting. It was largely invented by Grassman in the mid-1800s but was ignored and reinvented later (Crowe, 1994). Grassman now has a following in the graphics field of researchers who are developing *Geometric Algebra* based on some of his ideas (Doran & Lasenby, 2003). Readers interested in why the particular scalar and vector products are in some sense the right ones, and why we do not have a commonly-used vector division, will find enlightenment in the concise *About Vectors* (Hoffmann, 1975). Another important geometric tool is the *quaternion* invented by Hamilton in the mid-1800s. Quaternions are useful in many situations, but especially where orientations are concerned (Hanson, 2005).

Exercises

1. The *cardinality* of a set is the number of elements it contains. Under IEEE floating point representation (Section 1.5), what is the cardinality of the *floats*?

2. Is it possible to implement a function that maps 32-bit integers to 64-bit integers that has a well-defined inverse? Do all functions from 32-bit integers to 64-bit integers have well-defined inverses?

3. Specify the unit cube (x-, y-, and z-coordinates all between 0 and 1 inclusive) in terms of the Cartesian product of three intervals.

4. If you have access to the natural log function $\ln(x)$, specify how you could use it to implement a $\log(b, x)$ function where b is the base of the log. What should the function do for negative b values? Assume an IEEE floating point implementation.

5. Solve the quadratic equation $2x^2 + 6x + 4 = 0$.

6. Implement a function that takes in coefficients A, B, and C for the quadratic equation $Ax^2 + By + C = 0$ and computes the two solutions. Have the function return the number of valid (not NaN) solutions and fill in the return arguments so the smaller of the two solutions is first.

7. Show that the two forms of the quadratic formula on page 17 are equivalent (assuming exact arithmetic) and explain how to choose one for each root in

order to avoid subtracting nearly equal floating point numbers, which leads to loss of precision.

8. Show by counterexample that it is not always true that for 3D vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \times \mathbf{c}$.

9. Given the non-parallel 3D vectors $\mathbf{a}$ and $\mathbf{b}$, compute a right-handed orthonormal basis such that $\mathbf{u}$ is parallel to $\mathbf{a}$ and $\mathbf{v}$ is in the the plane defined by $\mathbf{a}$ and $\mathbf{b}$.

10. What is the gradient of $f(x, y, z) = x^2 + y - 3z^3$?

11. What is a parametric form for the axis-aligned 2D ellipse?

12. What is the implicit equation of the plane through 3D points $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$? What is the parametric equation? What is the normal vector to this plane?

13. Given four 2D points $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{b}_0$, and $\mathbf{b}_1$, design a robust procedure to determine whether the line segments $\mathbf{a}_0\mathbf{a}_1$ and $\mathbf{b}_0\mathbf{b}_1$ intersect.

14. Design a robust procedure to compute the barycentric coordinates of a 2D point with respect to three 2D non-collinear points.