Library of Congress Cataloging-in-Publication Data

Buckland, Mat.
Programming game AI by example / by Mat Buckland.
p. cm.
Includes index.
ISBN 1-55622-078-2 (pbk.)
1. Computer games—Design. 2. Computer games—Programming. 3. Computer graphics. I. Title.
QA76.76.C672B85 2004
794.8'1526—dc22 2004015103

2320 Los Rios Boulevard
Plano, Texas 75074

Printed in the United States of America

ISBN 1-55622-078-2

10 9 8 7 6 5 4 3
0409

All inquiries for volume purchases of this book should be addressed to Wordware Publishing, Inc., at the above address. Telephone inquiries may be made by calling:

(972) 423-0090

Programming Game AI by Example

Mat Buckland

Wordware Publishing, Inc.

Chapter 1

A Math and Physics Primer

There's no hiding from it — if you want to learn AI, it helps to know some mathematics and physics. Sure, you can use many AI techniques in a "cut and paste" fashion, but that's not doing yourself any favors; the moment you have to solve a problem slightly different from the one you've borrowed the code from you're going to run into difficulties. If you understand the theory behind the techniques, however, you will stand a much better chance of figuring out an alternative solution. Besides, it feels good to understand the tools you're working with. What better reason do you need to learn this stuff but that?

I'm going to write this chapter assuming you know hardly anything at all about math or physics. So forgive me if you already know most of it, but I figure this way I'll catch everyone, no matter what your experience is. Skim through the chapter until you come to something you don't know or you find a topic where you think your memory needs to be refreshed. At that point, start reading. If you are already comfortable with vector math and the physics of motion, I suggest you skip this chapter entirely and come back later if you find something you don't understand.

Mathematics

We'll start with mathematics because trying to learn physics without math is like trying to fly without wings.

Cartesian Coordinates

You are probably already familiar with the Cartesian coordinate system. If you've ever written a program that draws images to the screen then you will almost certainly have used the Cartesian coordinate system to describe the positions of the points, lines, and bitmaps that make up the image.

In two dimensions, the coordinate system is defined by two axes positioned at right angles to each other and marked off in unit lengths. The horizontal axis is called the x-axis and the vertical axis, the y-axis. The point where the axes cross is called the *origin*. See Figure 1.1.

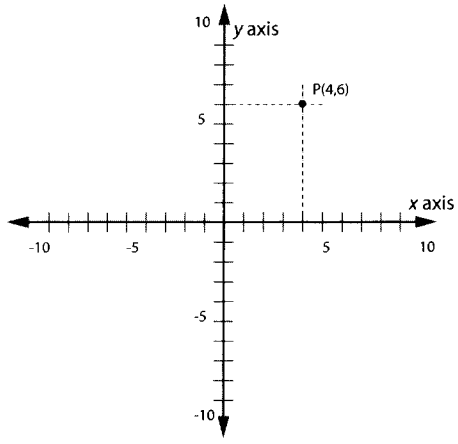

Figure 1.1. The Cartesian coordinate system

The arrowheads at each end of the axes in Figure 1.1 indicate they extend in each direction infinitely. If you imagine yourself holding an infinitely large sheet of paper with the x and y axes drawn on it, the paper represents the xy plane — the plane on which all points in the two-dimensional Cartesian coordinate system can be plotted. A point in 2D space is represented by a *coordinate pair* (x, y). The x and y values represent the distances along each of the respective axes. Nowadays, a series of points or lines plotted on the Cartesian coordinate system is usually referred to as a graph, which saves a lot of typing for sure. :o)

NOTE To represent three-dimensional space, another axis is needed — the z-axis. The z-axis extends from behind your screen to way behind your head, passing through the origin en route. See Figure 1.2.

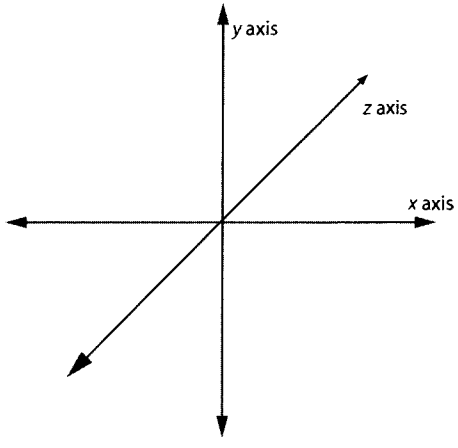

Figure 1.2. A three-axis (3D) coordinate system

Functions and Equations

The concept of functions is fundamental to mathematics. A *function* expresses the relationship between two (or more) terms called *variables*, and is typically written in the form of an *equation* (an algebraic expression set equal to another algebraic expression). Variables are named as such because, as the name implies, their values may vary. Variables are usually expressed with letters of the alphabet. The two most common variables you will see used in mathematical equations are x and y (although any letter or symbol is just as valid).

If each value of x can be associated with one value of y, then y is a function of x. y is said to be the *dependent* variable since its value depends on the value of x. Here are a couple of examples:

$$y = 2x \tag{1.1}$$

$$y = mx + c \tag{1.2}$$

In the second example, the m and the c represent *constants* (sometimes called coefficients) — values that never change no matter what the value of x is. They are effectively similar to the 2 in equation (1.1). Therefore, if $a = 2$, equation (1.1) can be written as follows:

$$y = ax \tag{1.3}$$

Given any value of x, the corresponding y value can be calculated by putting the x value into the function. Given $x = 5$ and $x = 7$ and the function $y = 2x$, the y values are:

$$\begin{aligned} y &= 2(5) = 10 \\ y &= 2(7) = 14 \end{aligned} \tag{1.4}$$

This type of function, where y is only dependent on one other variable, is called a *single-variable* function. Single-variable functions may be visualized by plotting them onto the xy Cartesian plane. To plot a function, all you have to do is move along the x-axis and for each x value use the function to calculate the y value. Of course, it's impossible to plot the graph for every value of x — that would take forever (literally) — so you must select a range of values.

The left-hand side of Figure 1.3 shows how function $y = 2x$ looks when plotted on the xy plane, using the range of x values between –5.0 and 5.0.

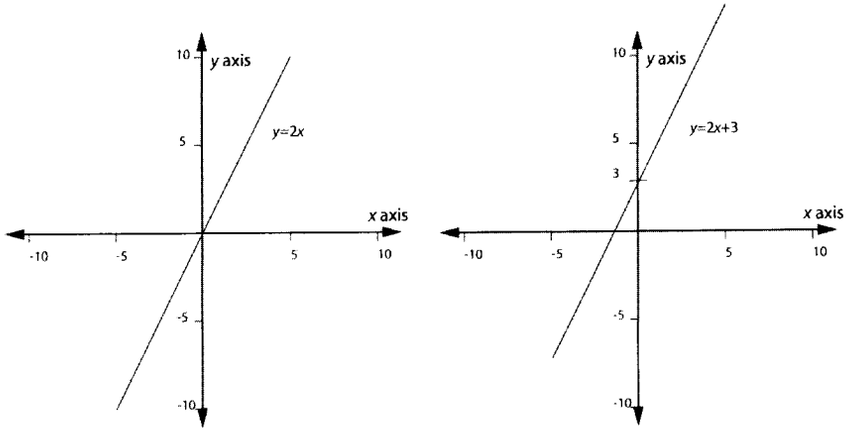

Figure 1.3. Functions plotted in Cartesian space

To plot the function $y = mx + c$ to a graph, you must first have some values for the constants m and c. Let's say $m = 2$ and $c = 3$, giving the function $y = 2x + 3$. The right-hand side of Figure 1.3 shows the resulting graph.

The graphs look very similar, don't they? That's because $y = mx + c$ is the function that defines all straight lines in 2D space. The constant m defines the line's gradient, or how steep the slope of the line is, and the constant c dictates where the line intersects the y-axis. The function $y = 2x$, shown on the left in the figure, is equivalent to the function $y = mx + c$, when $m = 2$ and $c = 0$. The plot on the right is almost identical but because its c value is 3, the point where it intersects the y-axis is shifted up by three units.

Sometimes you will see a function such as $y = mx + c$ written like this:

$$f(x) = mx + c \qquad (1.5)$$

The notation $f(x)$ is stating that the dependent variable — in this example, the y — depends on the variable x in the expression given on the right-hand side, $mx + c$. Often, you will see symbols other than an f to represent the function, so don't become confused if you come across something like the following.

$$g(x) = x^2 + bx \qquad (1.6)$$

The $g(x)$ represents exactly the same thing as if the equation was written as:

$$f(x) = x^2 + bx \qquad (1.7)$$

Functions can depend on more than one variable. Take the calculation for the area of a rectangle for example. If its length is denoted by the letter l, and its width by w, then the area A is given by the equation:

$$A = lw \qquad (1.8)$$

To plot a two-variable function like (1.8) on a graph, a third dimension, z, must be added, perpendicular to the other axes. Now it's possible to plot A to the z-axis, l to the x-axis, and w to the y-axis. See Figure 1.4.

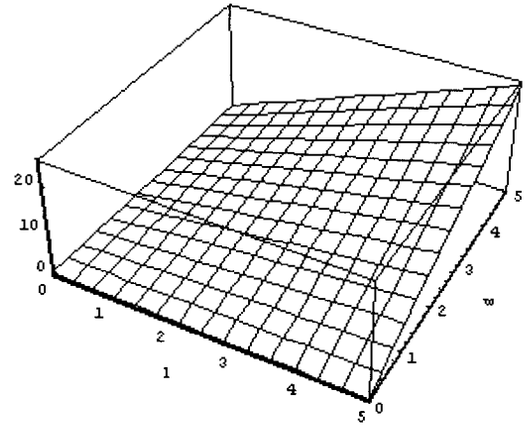

Figure 1.4. The function A = lw plotted in three dimensions

The volume of a cube is given by the three-variable function:

$$V = lwh \qquad (1.9)$$

where the h represents the height of the cube. To plot this on a graph you need to add a fourth axis. Unfortunately, unless under the influence of psychotropic compounds, humans cannot see in more than three dimensions. However, we do have the ability to imagine them, so that's what you have to do if you want to plot functions with more than three variables on a graph. Mathematicians seem to find this easy to do, but many programmers, myself included, don't!

NOTE The space an n-dimensional function occupies, where n is greater than 3, is often referred to as *hyperspace* by mathematicians.

Exponents and Powers

An exponential function is defined like this:

$$f(x) = a^x \qquad (1.10)$$

The a is known as the *base* and the x as the *power*. If the equation is spoken, you would say that $f(x)$ equals a to the power x. This means that a is multiplied with itself x amount of times. So 7^2 is the same as writing 7x7, and 3^4 is the same as writing 3x3x3x3. A number to the power of 2 is known as the square of that number, and a number to the power of 3 is known as the cube. Therefore, the cube of 5 is:

$$5^3 = 5 \times 5 \times 5 = 125 \tag{1.11}$$

Figure 1.5 shows equation (1.10) plotted on a graph for $a = 2$. The curve clearly shows how the value of y increases rapidly with x. This type of curve is often referred to as *exponential growth*.

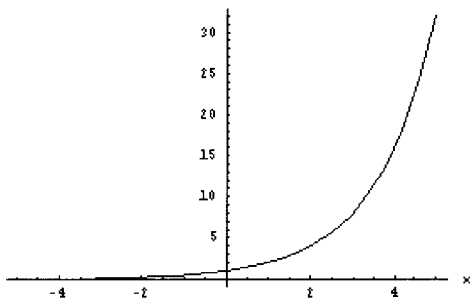

Figure 1.5. The function $f(x) = 2^x$ plotted on the xy plane

HISTORICAL NOTE For a reason lost to time, mathematicians decided they would use the latter part of the alphabet to represent variables and the rest of the alphabet to represent constants. This is why the axes in the Cartesian coordinate system are labeled x, y, and z.

Roots of Numbers (Radicals)

The *square root* of a number is a value that when multiplied by itself results in the original number. Square roots are written using the *radical* symbol $\sqrt{\ }$. Therefore, the square root of 4 is written as:

$$\sqrt{4} = 2 \tag{1.12}$$

We can square both sides of this equation to show the relationship between the power and the root:

$$4 = 2^2 \tag{1.13}$$

The square root of a number is also known as the *second root* of that number. We can also calculate the third, fourth, fifth, or any size root of a number. The third root of a number is known as its cube root and is written like this: $\sqrt[3]{\ }$. Notice how we need the 3 there to tell us that the root to be

taken is the third. The cube root of a number gives a number that when multiplied to the power of three gives the original number. For instance:

$$\sqrt[3]{27} = 3 \tag{1.14}$$

Once again we can cube both sides of the equation to show the relationship between the power and the root:

$$27 = 3^3 \tag{1.15}$$

It's also possible to write the root of a number as a *fractional exponent*. For example, the square root of a number can be written as $x^{\frac{1}{2}}$, the third root as $x^{\frac{1}{3}}$, and so on.

Simplifying Equations

Often, to solve an equation you must first simplify it. One of the golden rules for achieving this is that you can add, subtract, divide, or multiply terms to either side. (There is one exception to this rule: The term must not be zero when multiplying or dividing.) As long as the same thing is done to *both* sides, then the sides will remain equal. This is best understood with the aid of a couple of examples.

Example 1

Consider the following equation:

$$3x + 7 = 22 - 2x \tag{1.16}$$

This equation can be simplified by subtracting 7 from both sides.

$$\begin{aligned} 3x + 7 - 7 &= 22 - 2x - 7 \\ 3x &= 15 - 2x \end{aligned} \tag{1.17}$$

It can be further simplified by adding $2x$ to both sides:

$$\begin{aligned} 3x + 2x &= 15 - 2x + 2x \\ 5x &= 15 \end{aligned} \tag{1.18}$$

We can also divide both sides by 5, giving us the answer for x:

$$\begin{aligned} \frac{5x}{5} &= \frac{15}{5} \\ x &= 3 \end{aligned} \tag{1.19}$$

Let's take a look at a slightly more complex example.

Example 2

Let's say we want to solve the following for y:

$$y = 2(3x - 5y) + \frac{x}{3} \tag{1.20}$$

First of all we can remove the parentheses by multiplying the term inside the parentheses $(3x - 5y)$, by the term outside (2), giving:

$$y = 6x - 10y + \frac{x}{3} \tag{1.21}$$

Next, it's a good idea to remove all fractional terms by multiplying all the terms on both sides with the denominators of the fractions (the denominators are the values beneath the line). In this example, multiplying all terms on both sides of equation (1.21) by 3 gives:

$$3y = 18x - 30y + x \tag{1.22}$$

At this point we have a y term on the left and x *and* y terms on the right. We need to transpose similar terms so they share the same side of the equation. In this example we can do this by adding $30y$ to both sides.

$$\begin{aligned} 3y + 30y &= 18x - 30y + x + 30y \\ 3y + 30y &= 18x + x \end{aligned} \tag{1.23}$$

Now that like terms are grouped together we can combine them. This gives:

$$33y = 19x \tag{1.24}$$

Finally, we should divide both sides by the coefficient in front of the unknown variable. In this example we are solving for y so we must divide both sides by 33, giving:

$$y = \frac{19}{33}x \tag{1.25}$$

Example 3

Here are a few more rules that come in handy when simplifying equations:

$$\frac{x}{y} = \frac{1}{y}(x) \tag{1.26}$$

$$\frac{a}{x} + \frac{b}{y} = \frac{ay + bx}{xy} \tag{1.27}$$

$$(x + y)^2 = x^2 + y^2 + 2xy \tag{1.28}$$

$$\left(\frac{x}{y}\right)^2 = \frac{x^2}{y^2} \tag{1.29}$$

$$\sqrt{\frac{x}{y}} = \frac{\sqrt{x}}{\sqrt{y}} \tag{1.30}$$

Let's take a look at some of the new rules in action. This time the equation to simplify is:

$$5x - 2y = \left(\frac{y - x}{\sqrt{x}}\right)^2 \tag{1.31}$$

Using rule (1.29) gives:

$$\begin{aligned} 5x - 2y &= \frac{(y - x)^2}{\left(\sqrt{x}\right)^2} \\ 5x - 2y &= \frac{(y - x)^2}{x} \end{aligned} \tag{1.32}$$

Multiplying both sides by x to dispose of the fractional part gives:

$$x(5x - 2y) = (y - x)^2 \tag{1.33}$$

Now to get rid of the parentheses on the left:

$$5x^2 - 2xy = (y - x)^2 \tag{1.34}$$

To remove the parentheses on the right we use the rule from (1.28):

$$5x^2 - 2xy = x^2 + y^2 - 2xy \tag{1.35}$$

Adding $2xy$ to both sides gives:

$$5x^2 = x^2 + y^2 \tag{1.36}$$

By subtracting x^2 from both sides and rearranging we get the simplified equation:

$$y^2 = 4x^2 \tag{1.37}$$

The final step is to take the square root of both sides:

$$y = 2x \tag{1.38}$$

Simplifying equations can get a lot harder than this of course, but these few rules are enough for you to understand any of the simplifications presented in this book.

Trigonometry

Trigonometry is based on the study of triangles. The word comes from the Greek words *trigon*, for triangle, and *metry*, for measure. It is an enormously useful field of mathematics and has many practical applications in computer science. In the game AI field, you will find it used for line-of-sight (LOS) calculations, collision detection, some aspects of pathfinding, etc. Lots of AI is really math-dependent when you boil it down; you will be wise to learn it well.

Rays and Line Segments

A *ray* is a line with one endpoint. It is of infinite length and is defined by a direction (usually expressed as a normalized vector; see the section on vectors later in this chapter) and an origin. Figure 1.6 shows a ray situated at the origin.

A *line segment* is a *piece* of a line and is defined by two endpoints. Figure 1.6 also shows a line segment defined by the two endpoints p1 and p2.

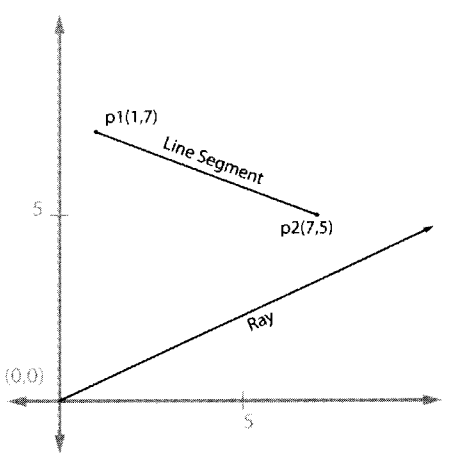

Figure 1.6. A line segment and a ray

Angles

An angle is defined as the measure of divergence of two rays that share the same origin. See Figure 1.7.

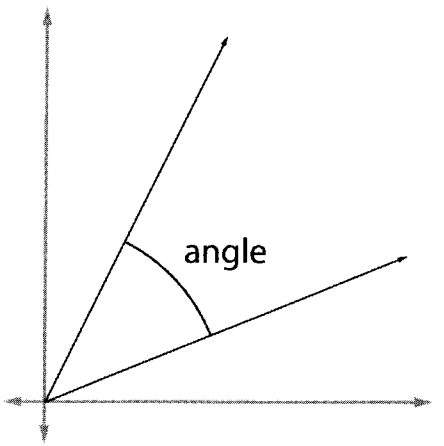

Figure 1.7. An angle

You may be used to thinking of angles in terms of degrees. Walls in most homes are typically at 90 degree angles, for example, and circles are 360 degrees around. Mathematicians prefer to measure the magnitude of an angle using *radians*. Radians are a unit of measurement based upon a circle of unit radius — a radius of 1 — centered at the origin. The radius of a circle is the distance from the center of the circle to its perimeter. Drawing the two rays from Figure 1.7 onto the same diagram as the unit circle, we get Figure 1.8. The length of the curved line segment between the two rays — shown in the diagram as a dotted line — is the angle measured in radians between them.

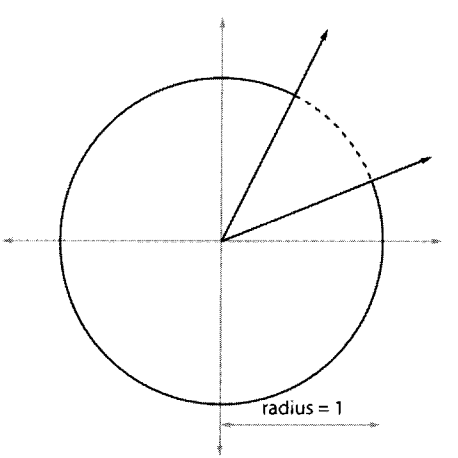

Figure 1.8. The length of the dotted line is the angle in radians between the two rays.

Now that you know what a radian is, let's calculate how many radians there are in a circle. You may remember the Greek symbol π (pi) from your school days. It's a well-known and frequently used mathematical constant, and has a value of 3.14159 (to five decimal places). You can use pi to calculate the circumference of a circle — the distance around the entire perimeter — using the equation:

$$perimeter = 2\pi r \tag{1.39}$$

Using this equation to determine the perimeter of a unit circle gives the number of radians in a circle. That's because the number of radians in a circle *is* the length of the perimeter of a circle with a radius of 1. So we just substitute 1 for r in equation (1.39) to get:

$$perimeter = 2\pi r = 2\pi(1) = 2\pi = num\ radians \tag{1.40}$$

Therefore, there are 2 π radians in every circle.

⌘ **TIP** Now that you know how many radians make up a circle, you can convert between radians and degrees if you ever have to. There are 360 degrees in a circle, so that means:

360° = 2 π rads

Dividing both sides by 360 we get:

1° = 2 π /360 rads

Angles are usually denoted using the Greek letter *theta*, which looks like this: θ.

Triangles

A triangle consists of three line segments connected at their ends. A triangle's inner angles always add up to π radians (180 degrees). Figure 1.9 shows the different types of triangles you can encounter.

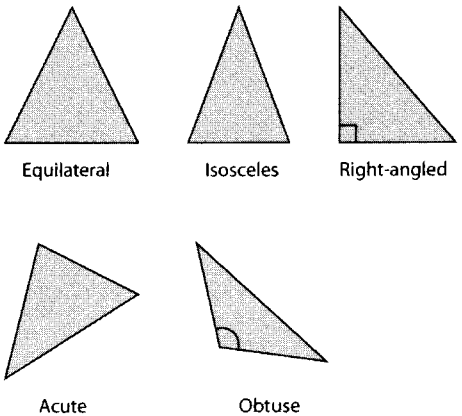

Figure 1.9. Different types of triangles

- An **equilateral** triangle has sides of equal length. Triangles with this property also have angles of equal sizes.
- An **isosceles** triangle has two sides and two angles of equal size.
- A **right-angled** triangle has one angle that is π/2 radians (90 degrees) — a *right angle*. The right angle is always represented by a box.
- An **acute** triangle's inner angles are all acute (less than π/2 radians).
- An **obtuse** triangle has one angle that is obtuse (greater than π/2 radians).

Pythagorean Theorem

The triangles you will be using most are of the right-angled variety. They have many interesting properties you can put to good use. Possibly the most famous property of right-angled triangles was discovered by Pythagoras, a Greek mathematician who lived from 569 to 475 BC. He was a very clever chap indeed, and is most famous for stating this:

The square of the hypotenuse of a right-angled triangle is equal to the sum of the squares of the other two sides.

The hypotenuse of a triangle is its longest side, as shown in Figure 1.10.

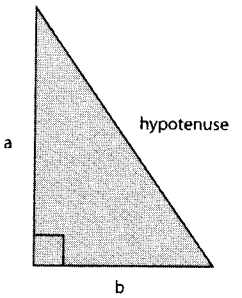

Figure 1.10

If the hypotenuse is denoted as h, the Pythagorean theorem can be written as:

$$h^2 = a^2 + b^2 \tag{1.41}$$

Taking the square root of both sides gives:

$$h = \sqrt{a^2 + b^2} \tag{1.42}$$

This means that if we know the length of any two sides of a right-angled triangle, we can easily find the third.

TIP When working on the AI for games you will frequently find yourself using the Pythagorean theorem to calculate if Agent A is closer to an object than Agent B. This would normally require two calls to the square root function, which, as we all know, is slow and should be avoided wherever possible. Fortunately, when comparing the lengths of the sides of two triangles, if side A is bigger than side B, then it will always be bigger, whether the lengths are squared or not. This means that we can avoid taking the square roots and just compare the squared values instead. This is known as working in *squared-distance space* and is something you will see frequently in the code shown in this book.

A Practical Example of the Pythagorean Theorem

Let's say you have an archer at position A (8, 4) and his target at position T (2, 1). The archer can only fire an arrow a maximum distance of 10 units. Consequently, to determine if he can hit the target, the distance between them must be calculated. This is easy to determine using the Pythagorean theorem. First, the lengths of the sides TP and AP shown in Figure 1.11 are calculated.

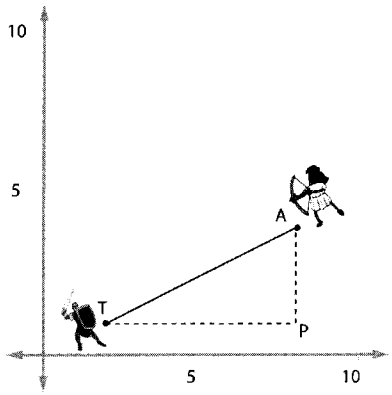

Figure 1.11

To find the distance AP, the y component of the archer's position is subtracted from the y component of the target's position:

$$AP = 4 - 1 = 3 \tag{1.43}$$

To find the distance TP, we do the same, but with the x components:

$$TP = 8 - 2 = 6 \tag{1.44}$$

Now that TP and AP are known, the distance from the archer to the target can be calculated using the Pythagorean theorem:

$$\begin{aligned} TA &= \sqrt{AP^2 + TP^2} \\ &= \sqrt{3^2 + 6^2} \\ &= \sqrt{9 + 36} \\ &= 6.71 \end{aligned} \tag{1.45}$$

Well within target range. Let that arrow fly!

The Mysteries of SohCahToa Unveiled

If you know the length of one of the sides of a right-angled triangle and one of the remaining two angles, you can determine everything else about the triangle using trigonometry. First, take a look at Figure 1.12. It shows the names of each side of a right-angled triangle.

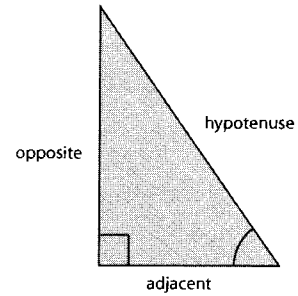

Figure 1.12. Names of the sides of a triangle

The side opposite the angle is called the *opposite* (surprise, surprise), and the side lying between the angle and the right angle is known as the *adjacent*. There are three trigonometric functions to help calculate the features of a right-angled triangle. You probably know them from school. They are sine, cosine, and tangent, and are often abbreviated to *sin*, *cos*, and *tan*. This is what they represent:

$$\sin(\theta) = \frac{opposite}{hypotenuse} \tag{1.46}$$

$$\cos(\theta) = \frac{adjacent}{hypotenuse} \tag{1.47}$$

$$\tan(\theta) = \frac{opposite}{adjacent} \tag{1.48}$$

It will pay you well to memorize these three relationships because you'll be using them frequently. My math teacher taught me to memorize them as a mnemonic: Soh-Cah-Toa, pronounced "sowcahtowa" (where "sow" and "tow" rhyme with "know"). Although it looks weird, it's easy to say, and very easy to remember.

The best way of seeing how the sine, cosine, and tangent functions can be utilized is by looking at some examples.

TIP When working out any of the following problems on a calculator, make sure it's set to work in radians, and not degrees!

Take a look at Figure 1.13.

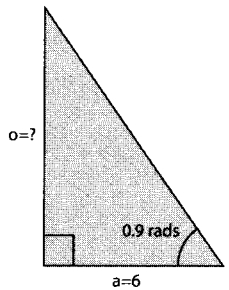

Figure 1.13

We want to calculate the length of the opposite given the length of the adjacent and the angle. From SohCahToa we can remember that the tangent of an angle is equal to the opposite divided by the adjacent. Rearranging the equation a little gives us:

$$o = aTan(\theta) \tag{1.49}$$

So all we have to do to get o is pick up a calculator (to determine the tangent) and plug in the numbers, like so:

$$\begin{aligned} o &= 6Tan(0.9) \\ &= 7.56 \end{aligned} \tag{1.50}$$

Easy peasy. Okay, let's try another, only this time you try to solve it first. Calculate the length of the side h shown in Figure 1.14

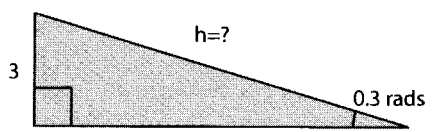

Figure 1.14

Did you manage it? In this example we know the angle and the opposite. Remembering SohCahToa, we see that it's the sine function that should be used because the sine of the angle is equal to the opposite divided by the hypotenuse. Rearranging the equation gives:

$$h = \frac{o}{\sin(\theta)} \tag{1.51}$$

And plugging in the numbers gives:

$$\begin{aligned} h &= \frac{3}{\sin(0.3)} \\ &= 10.15 \end{aligned} \tag{1.52}$$

So far so good. How about the problem shown in Figure 1.15? This time you have to find the angle given the lengths of the adjacent and hypotenuse.

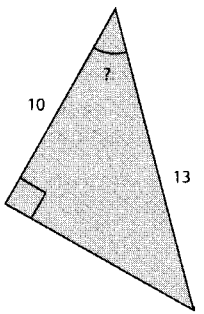

Figure 1.15

This time our friend is the cosine function, but plugging in the numbers creates a problem.

$$\cos(?) = \frac{10}{13} = 0.769 \tag{1.53}$$

We know that the *cosine* of the angle is 0.769, but what is the angle itself? How do we find that out? Well, the angle is determined using the *inverse cosine*. This is normally written as $\cos^{-1}$. So, all you do is use the inverse cosine button on a calculator (if you can't see $\cos^{-1}$ on your calculator, you may have to press the inverse button before the cosine button) to get the result:

$$? = \cos^{-1}(0.769) = 0.693 \; radians \tag{1.54}$$

At this point I'm going to end the lesson in trigonometry. Although it is a vast subject, the Pythagorean theorem and SohCahToa are all the trig theory you are going to need for the rest of this book.

Vectors

You'll be using vector math frequently when designing the AI for your games. Vectors are used everywhere from calculating which direction a game agent should shoot its gun to expressing the inputs and outputs of an artificial neural network. Vectors are your friend. You should get to know them well.

You have learned that a point on the Cartesian plane can be expressed as two numbers, just like this:

$$P = (x, y) \tag{1.55}$$

A 2D vector looks almost the same when written down:

$$\mathbf{v} = (x, y) \tag{1.56}$$

However, although similar, a vector represents two qualities: direction *and* magnitude. The right-hand side of Figure 1.16 shows the vector (9, 6) situated at the origin.

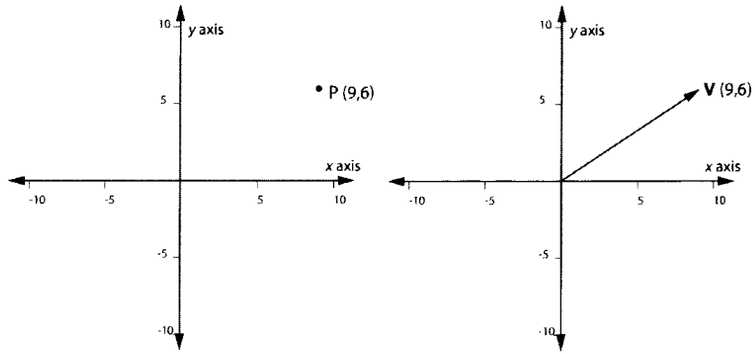

Figure 1.16. A point, P, and a vector, V

NOTE Vectors are typically denoted in bold typeface or as a letter with an arrow above it like so: $\vec{v}$. I'll be using the bold notation throughout this book.

The bearing of the arrow shows the direction of the vector and the length of the line represents the magnitude of the vector. Okay, so far so good. But what does this mean? What use is it? Well, for starters, a vector can represent the velocity of a vehicle. The magnitude of the vector represents the speed of the vehicle and the direction represents the heading of the vehicle. That's quite a lot of information from just two numbers (x, y).

Vectors aren't restricted to two dimensions either. They can be any size at all. You would use a 3D vector, (x, y, z) for example, to represent the velocity of a vehicle that moves in three dimensions, like a helicopter.

Let's take a look at some of the things you can do with vectors.

Adding and Subtracting Vectors

Imagine you are a contestant in a TV reality game. You are standing in a clearing in the jungle. Several other competitors stand beside you. You're all very nervous and excited because the winner gets to date Cameron Diaz... and the losers have to watch. Sweat is dripping from your forehead, your hands are clammy, and you cast nervous glances at the other competitors. The bronzed, anvil-chinned TV host steps forward and hands a gold-trimmed envelope to each competitor. He steps back and orders you all to rip open your envelopes. The first person to complete the instructions will be the winner. You frantically tear away at the paper. Inside is a note. It says:

I'm waiting for you in a secret location. Please hurry, it's very hot in here. You can reach the location by following the vectors (–5, 5), (0, –10), (13, 7), (–4, 3).

Cameron

With a smile on your face you watch the rest of the competitors sprint off in the direction of the first vector. You do a few calculations on the back of the envelope and then set off in a completely different direction at a leisurely stroll. By the time the other competitors reach Cameron's hideout, sweating like old cheese and gasping for breath, they can hear your playful giggles and the splash of cool shower water...

You beat the opposition because you knew how to add vectors together. Figure 1.17 shows the route all the other competitors took by following the vectors given in Cameron's note.

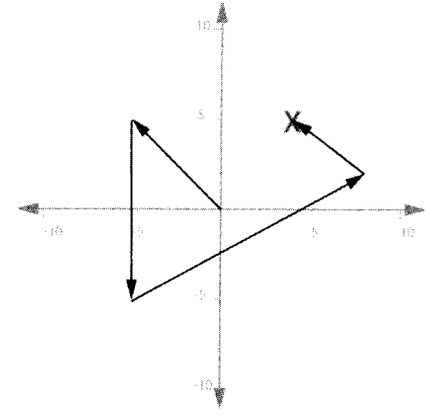

Figure 1.17. The route of the opposition

You knew, however, that if you added all the vectors together you would get a single vector as the result: one that takes you directly to the final destination. To add vectors together you simply add up all the x values to give the result's x component, and then do the same with the y values to get the y component. Adding the four vectors in Cameron's note together we get:

$$\begin{aligned} new\ x &= (-5) + (0) + (13) + (-4) = 4 \\ new\ y &= (5) + (-10) + (7) + (3) = 5 \end{aligned} \qquad (1.57)$$

giving the vector (4, 5), exactly the same result as if we followed each vector individually. See Figure 1.18.

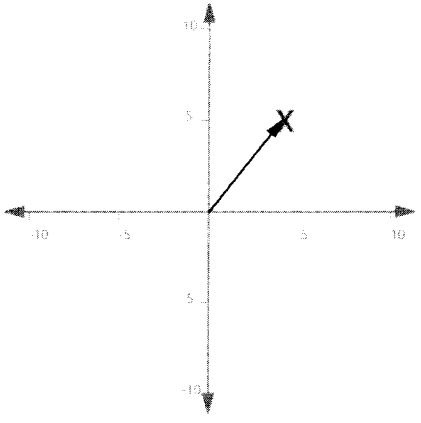

Figure 1.18. Your route

Multiplying Vectors

Multiplying vectors is a cinch. You just multiply each component by the value. For example, the vector **v** (4, 5) multiplied by 2 is (8, 10).

Calculating the Magnitude of a Vector

The *magnitude* of a vector is its length. In the previous example the magnitude of the vector **v** (4, 5) is the distance from the start point to Cameron's hideout.

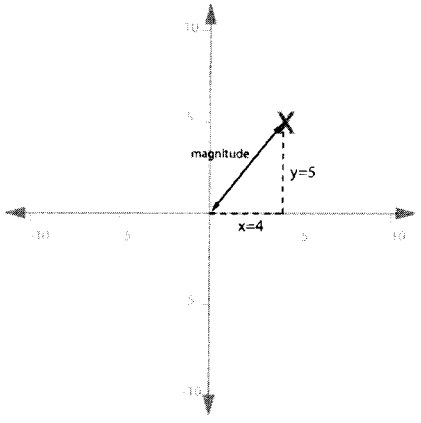

Figure 1.19. Finding the magnitude of a vector

This is easy to calculate using the Pythagorean theorem.

$$magnitude = \sqrt{4^2 + 5^2} = 6.403 \qquad (1.58)$$

If you had a three-dimensional vector then you would use the similar equation:

$$magnitude = \sqrt{x^2 + y^2 + z^2} \qquad (1.59)$$

Mathematicians place two vertical bars around a vector to denote its length.

$$magnitude = |\mathbf{v}| \qquad (1.60)$$

Normalizing Vectors

When a vector is normalized, it retains its direction but its magnitude is recalculated so that it is of unit length (a length of 1). To do this you divide each component of the vector by the magnitude of the vector. Mathematicians write the formula like this:

$$\mathbf{N} = \frac{\mathbf{v}}{|\mathbf{v}|} \qquad (1.61)$$

Therefore, to normalize the vector (4, 5) you would do this:

$$\begin{aligned} new\ x &= 4 / 6.403 = 0.62 \\ new\ y &= 5 / 6.403 = 0.78 \end{aligned} \qquad (1.62)$$

This may seem a strange thing to do to a vector but in fact, normalized vectors are incredibly useful. You'll find out why shortly.

Resolving Vectors

It's possible to use trigonometry to resolve a vector into two separate vectors, one parallel to the *x*-axis and one to the *y*-axis. Take a look at the vector, **v**, representing the thrust of the jet-fighter shown in Figure 1.20.

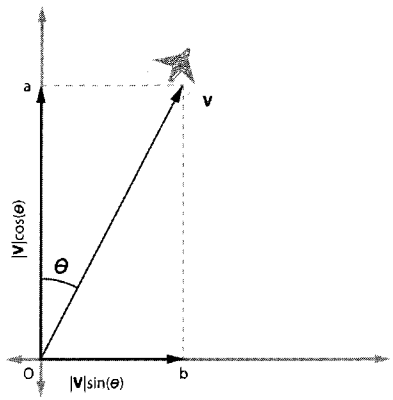

Figure 1.20

To resolve **v** into its *x*/*y* components we need to find Oa and Ob. This will give us the component of the aircraft's thrust that is acting along the *y*-axis, and the component along the *x*-axis, respectively. Another way of putting it is that Oa is the amount of thrust acting along the *x*-axis, and Ob is the amount along the *y*-axis.

First, let's calculate the amount of thrust along the *y*-axis: Oa. From trigonometry we know that:

$$\cos(\theta) = \frac{adjacent}{hypotenuse} = \frac{Oa}{|\mathbf{v}|} \tag{1.63}$$

Rearranged, this gives:

$$Oa = |\mathbf{v}| Cos(\theta) = y\ component \tag{1.64}$$

To calculate Ob this equation is used:

$$\sin(\theta) = \frac{opposite}{hypotenuse} = \frac{Ob}{|\mathbf{v}|} \tag{1.65}$$

Giving:

$$Ob = |\mathbf{v}| \sin(\theta) = x\ component \tag{1.66}$$

The Dot Product

The *dot product* gives the angle between two vectors — something you will need to calculate often when programming AI. Given the two 2D vectors **u** and **v**, the equation looks like this:

$$\mathbf{u} \bullet \mathbf{v} = \mathbf{u}_x \mathbf{v}_x + \mathbf{u}_y \mathbf{v}_y \tag{1.67}$$

The • symbol denotes the dot product. Equation (1.67) doesn't give us an angle though. I promised an angle, so you'll get one! Here's another way of calculating the dot product:

$$\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}||\mathbf{v}|\cos(\theta) \tag{1.68}$$

Rearranging we get:

$$\cos(\theta) = \frac{\mathbf{u} \bullet \mathbf{v}}{|\mathbf{u}||\mathbf{v}|} \tag{1.69}$$

Remember, the vertical lines surrounding a vector indicate its magnitude. Now is the time when you discover one of the useful uses for normalizing vectors. If **v** and **u** are *both normalized*, then the equation simplifies enormously to:

$$\begin{aligned} \cos(\theta) &= \frac{\mathbf{u} \bullet \mathbf{v}}{1 \times 1} \\ &= \mathbf{u} \bullet \mathbf{v} \end{aligned} \tag{1.70}$$

Substituting in the equation from (1.67) for the right-hand side gives:

$$\cos(\theta) = \mathbf{u} \bullet \mathbf{v} = \mathbf{u}_x \mathbf{v}_x + \mathbf{u}_y \mathbf{v}_y \tag{1.71}$$

giving us an equation for the angle between the vectors.

One great use of the dot product is that it will quickly tell you if one entity is behind or in front of the facing plane of another. How so? Check out Figure 1.21.

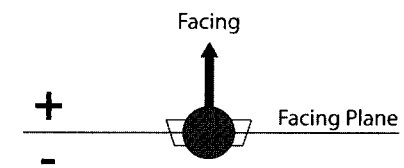

Figure 1.21

The figure shows a game agent facing directly north. The horizontal line is relative to the agent and describes the facing plane of the agent. Everything situated ahead of this line can be said to be in front of the agent.

Using the dot product it's easy to determine if an object is situated in front or behind the agent. The dot product of the agent's facing vector and the vector from the agent to the object will be positive if the object is forward of the facing plane of the agent and negative if it is behind.

A Practical Example of Vector Mathematics

Here's an example of some of the vector methods you've just learned about working together. Let's say you have a game agent, Eric the Troll, who stands at position *T* (the origin) and facing in the direction given by the normalized vector **H** (for heading). He can smell a helpless princess at position *P* and would very much like to throw his club at her, to tenderize her a little, before he rips her to pieces. To do this, he needs to know how many radians he must rotate to face her. Figure 1.22 shows the situation.

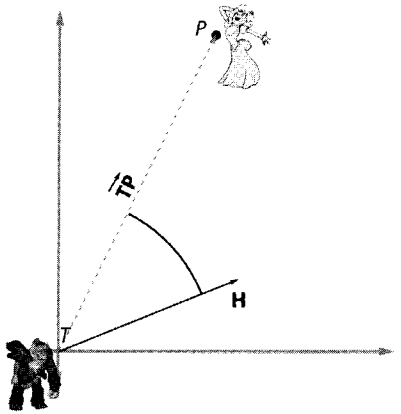

Figure 1.22

You've discovered that you can calculate the angle between two vectors using the dot product. However, in this problem you only have one vector to start with, **H**. Therefore we need to determine the vector $\overrightarrow{TP}$ — the vector that points directly at the princess. This is calculated by subtracting point *T* from point *P*. Because *T* is at the origin (0, 0), in this example *P*–*T* = *P*. However, the answer *P*–*T* is a vector, so let's show this by typing it in bold and calling it **P**.

We know that the cosine of the angle the troll needs to turn to face the princess is equivalent to the dot product of **H** and **P**, provided both vectors are normalized. **H** is already normalized so we only need to normalize **P**. Remember, to normalize a vector its components are divided by its magnitude. Consequently, the normal of **P** ($\mathbf{N_P}$) is:

$$\mathbf{N}_p = \frac{\mathbf{P}}{|\mathbf{P}|} \tag{1.72}$$

The dot product can now be used to determine the angle.

$$\cos(\theta) = \mathbf{N}_p \bullet \mathbf{H} \tag{1.73}$$

So

$$\theta = \cos^{-1}(\mathbf{N}_p \bullet \mathbf{H}) \tag{1.74}$$

To clarify the process, let's do the whole thing again but with some numbers. Let's say the troll is situated at the origin *T* (0, 0) and has a heading of **H** (1, 0). The princess is standing at the point *P* (4, 5). How many radians does the troll have to turn to face the princess?

We know that we can use equation (1.74) to calculate the angle but first we need to determine the vector, **TP**, between the troll and the princess and normalize it. To obtain **TP** we subtract *T* from *P*, resulting in the vector (4, 5). To normalize **TP** we divide it by its magnitude. This calculation was shown earlier in equation (1.62), resulting in $\mathbf{N}_{TP}$ (0.62, 0.78).

Finally we plug the numbers into equation (1.74), substituting equation (1.71) for the dot product.

$$\theta = \cos^{-1}(\mathbf{N}_{TP} \bullet \mathbf{H})$$
$$\theta = \cos^{-1}((0.62 \times 1) + (0.78 \times 0))$$
$$\theta = \cos^{-1}(0.62)$$
$$\theta = 0.902 \text{ radians}$$

The Vector2D Struct

All the examples given in this book make use of the `Vector2D` struct. It's very straightforward and implements all the vector operations we've discussed. I'll list the majority of its declaration here so you can familiarize yourself with it.

```
struct Vector2D
{
  double x;
  double y;

  Vector2D():x(0.0),y(0.0){}
  Vector2D(double a, double b):x(a),y(b){}

  //sets x and y to zero
  inline void Zero();

  //returns true if both x and y are zero
  inline bool      isZero()const;
```

```
  //returns the length of the vector
  inline double    Length()const;

  //returns the squared length of the vector (thereby avoiding the sqrt)
  inline double    LengthSq()const;

  inline void      Normalize();

  //returns the dot product of this and v2
  inline double    Dot(const Vector2D& v2)const;

  //returns positive if v2 is clockwise of this vector,
  //negative if counterclockwise (assuming the Y axis is pointing down,
  //X axis to right like a Window app)
  inline int       Sign(const Vector2D& v2)const;

  //returns the vector that is perpendicular to this one
  inline Vector2D  Perp()const;

  //adjusts x and y so that the length of the vector does not exceed max
  inline void      Truncate(double max);

  //returns the distance between this vector and the one passed as a parameter
  inline double    Distance(const Vector2D &v2)const;

  //squared version of above
  inline double    DistanceSq(const Vector2D &v2)const;

  //returns the vector that is the reverse of this vector
  inline Vector2D  GetReverse()const;

  //we need some operators
  const Vector2D& operator+=(const Vector2D &rhs);
  const Vector2D& operator-=(const Vector2D &rhs);
  const Vector2D& operator*=(const double& rhs);
  const Vector2D& operator/=(const double& rhs;
  bool operator==(const Vector2D& rhs)const;
  bool operator!=(const Vector2D& rhs)const;
};
```

Local Space and World Space

It's important you understand the difference between *local space* and *world space*. The world space representation is normally what you see rendered to your screen. Every object is defined by a position and orientation *relative to the origin of the world coordinate system* (see Figure 1.23). A soldier is using world space when he describes the position of a tank with a grid reference, for instance.

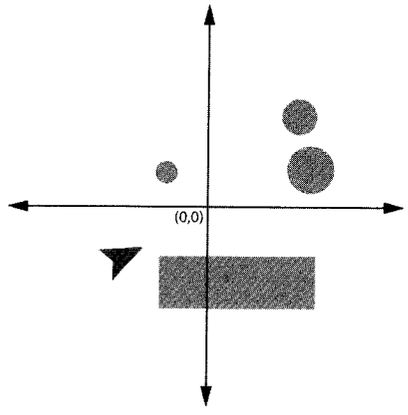

Figure 1.23. Some obstacles and a vehicle shown in world space

Local space, however, describes the position and orientation of objects relative to a specific entity's local coordinate system. In two dimensions, an entity's local coordinate system can be defined by a facing vector and a side vector (representing the local *x*- and *y*-axis, respectively), with the origin positioned at the center of the entity (for three dimensions an additional up vector is required). Figure 1.24 shows the axis describing the local coordinate system of the dart-shaped object.

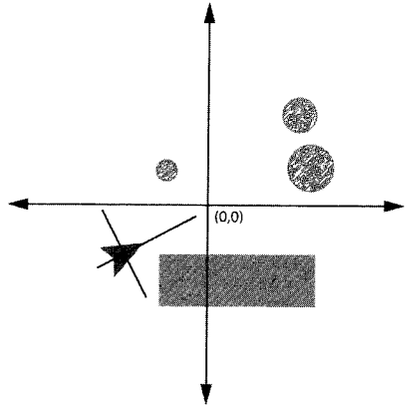

Figure 1.24. The vehicle's local coordinate system

Using this local coordinate system we can transform the world so that all the objects in it describe their position and orientation relative to it (see Figure 1.25). This is just like viewing the world through the eyes of the entity. Soldiers are using local space when they say stuff like "Target 50m

away at 10 o'clock." They are describing the location of the target relative to their own position and facing direction.

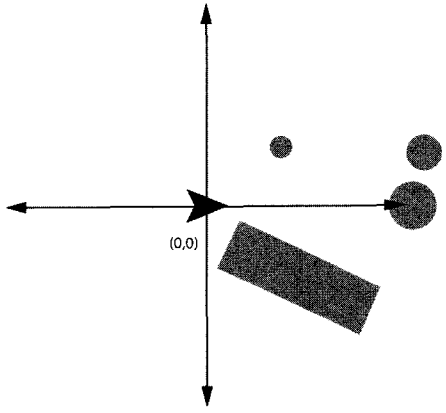

Figure 1.25. Objects transformed into the vehicle's local space

This ability to transform objects between local and world space can help simplify many calculations as you'll see later in the book. (Although you need to understand the concept, how it's actually done is beyond the scope of this book — check out the matrix transformations chapter of a computer graphics book.)

Physics

My dictionary defines the science of physics as:

> *The science of matter and energy and of the interactions between the two.*

As a game AI programmer you'll frequently be working with the laws of physics, and especially ones concerned with motion, which is what will be covered in this section. You'll often find yourself creating algorithms for predicting where an object or agent will be at some time in the future, for calculating what the best angle is to fire a weapon, or what heading and force an agent should kick a ball with to pass it to a receiver. This isn't AI per se of course, but it *is* all part of creating the illusion of intelligence and is normally part of the AI programmer's workload, so you need to know this stuff.

Let's take a look at some of the fundamental concepts used in physics.

Time

Time is a scalar quantity (completely specified by its magnitude and with no direction) measured in seconds, abbreviated to s. Until recently, a second was defined in terms of the rotational spin of the Earth, but as the

Earth's rotation is slowing down slightly every year, by the late sixties this became problematic for scientists who needed increasingly precise measurements for their experiments. Today, therefore, a second is measured as:

> *The duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium 133 atom.*

This definition provides today's scientists with the constant time interval they require for their precise experiments.

Time in computer games is measured in one of two ways: either in seconds (just as in the real world) or by using the time interval between updates as a kind of *virtual second*. The latter measurement can simplify many equations but you have to be careful because, unless the update rate is locked, the physics will differ between machines of varying speeds! Therefore, if you choose to use a virtual second, make sure your game's physics update frequency is locked to a reasonable rate — usually the rate of the slowest machine you're developing for.

NOTE Not all that long ago the majority of computer games used a fixed frame rate and every component — rendering, physics, AI, etc. — was updated at the same frequency. Many of today's sophisticated games, however, specify a unique rate for each component. For example, the physics might be updated 30 times a second, the AI 10 times a second, and the rendering code allowed to go as fast as the machine it runs on. Therefore, whenever I refer to an "update rate" in the text, if I don't specify a context, it will be in the context of the subject I'm talking about.

Distance

The standard unit of distance — a scalar quantity — is the meter, abbreviated to m.

Mass

Mass is a scalar quantity measured in kilograms, abbreviated to kg. Mass is the measure of an *amount* of something. This can be a confusing quality to measure since the mass of an object is calculated by weighing it, yet mass is not a unit of weight; it is a unit of *matter*. The weight of an object is a measurement of how much force gravity is exerting on that object. Because gravity varies from place to place (even here on Earth), this means the weight of an object can vary in different places, even though its mass never changes. So how can mass be measured accurately?

Scientists have overcome this problem by creating a platinum-iridium cylinder that everyone has agreed to call THE kilogram. This cylinder is kept in Paris and all measurements are made relative to it. In other words, you can go to France and have your own duplicate kilogram made, which weighs exactly the same as THE kilogram. Now you know that wherever

you are located, no matter what the gravity, your duplicate will have exactly the same mass as THE kilogram back in France. Problem solved.

Position

You might think the position of an object is an easy property to measure, but *where exactly* do you measure its position from? For example, if you wanted to specify your body's position in space, from where would you take the measurement? Would it be from your feet, your stomach, or your head? This presents a problem because there would be a big discrepancy between the position of your head and that of your feet.

Physicists solve this problem by taking the location of the *center of mass* of the object as its position. The center of mass is the object's balance point. This would be the place where you could attach an imaginary piece of string to the object and it would balance in any position. Another good way of thinking about the center of mass is that it is the average location of all the mass in a body.

Velocity

Velocity is a vector quantity (a quantity that has magnitude *and* direction) that expresses *the rate of change of distance over time*. The standard unit of measurement of velocity is meters per second, abbreviated to m/s. This can be expressed mathematically as:

$$v = \frac{\Delta x}{\Delta t} \tag{1.75}$$

The Greek capital letter Δ, read as delta, is used in mathematics to denote a *change in quantity*. Therefore, Δt in equation (1.75) represents a change in time (a time interval) and Δx a change in distance (a displacement). Δ is calculated as the *after quantity minus the before quantity*. Therefore if an object's position at t = 0 is 2 (before) and at t = 1 is 5 (after), Δx is 5 – 2 = 3. This can also result in negative values. For instance if an object's position at t = 0 is 7 (before) and at t = 1 is 3 (after), Δx is 3 – 7 = –4.

NOTE Delta's little brother, the lowercase letter delta, written as δ, is used to represent very small changes. You often see δ used in calculus. Because δ looks similar to the letter d, to prevent confusion, mathematicians tend to avoid using d to represent distance or displacement in their equations. Instead, a less ambiguous symbol such as Δx is used.

Using equation (1.75), it's easy to calculate the average velocity of an object. Let's say you want to work out the average velocity of a ball as it rolls between two points. First calculate the displacement between the two points, then divide by the amount of time it takes the ball to cover that

distance. For instance, if the distance between the points is 5 m and the time taken for the ball to travel between points is 2 s, then the velocity is:

$$v = \frac{5}{2} = 2.5 \text{ m/s} \tag{1.76}$$

It's also easy to calculate how far an object has traveled if we know its average speed and the length of time it has been traveling. Let's say you are driving your car at 35 mph and you'd like to know how far you've moved in the last half hour. Rearranging equation (1.75) gives:

$$\Delta x = v\Delta t \tag{1.77}$$

Popping in the numbers gives:

$$\text{distance traveled} = 35 \times \frac{1}{2} = 17.5 \text{ miles} \tag{1.78}$$

Relating this to computer games, if you have a vehicle at position **P** at time *t* traveling at constant velocity **V**, we can calculate its position at the next update step (at time *t* + *1*) by:

$$\mathbf{P}_{t+1} = \mathbf{P}_t + \mathbf{V}\Delta t \tag{1.79}$$

Where $\mathbf{V}\Delta t$ represents the displacement between update steps (from equation (1.77)).

Let's make this crystal clear by showing you a code example. Following is a listing for a `Vehicle` class that encapsulates the motion of a vehicle traveling with constant velocity.

```
class Vehicle
{
  //a vector representing its position in space
  vector m_vPosition;

  //a vector representing its velocity
  vector m_vVelocity;

public:

  //called each frame to update the position of the vehicle
  void Update(float TimeElapsedSinceLastUpdate)
  {
    m_vPosition += m_vVelocity * TimeElapsedSinceLastUpdate;
  }
};
```

Note that if your game uses a fixed update rate for the physics, as do many of the examples in this book, Δt will be constant and can be eliminated from the equation. This results in the simplified `Update` method as follows:

```
//update for a simulation using a constant update step
void Vehicle::Update()
{
  m_vPosition += m_vVelocity;
}
```

Remember though, that if you choose to eliminate Δt like this, the unit of time you will be using in any calculations is no longer the second but rather the time interval between update steps.

Acceleration

Acceleration is a vector quantity that expresses *the rate of change of velocity over time* and is measured in meters per second per second, written as m/s^2. Acceleration can be expressed mathematically as:

$$a = \frac{\Delta v}{\Delta t} \tag{1.80}$$

This equation is stating that acceleration is equivalent to the change in velocity of an object divided by the time interval during which the change in velocity occurred.

For example, if a car starts from rest and accelerates at 2 m/s^2, then every second, 2 m/s is added to its velocity. See Table 1.1.

Table 1.1

Time(s)	Velocity(m/s)
0	0
1	2
2	4
3	6
4	8
5	10

Plotting this data to a velocity versus time graph, we get Figure 1.26. If we examine a time interval, say the interval between $t = 1$ and $t = 4$, we can see that the gradient of the slope, given by $\frac{\Delta v}{\Delta t}$, is equivalent to the acceleration during that interval.

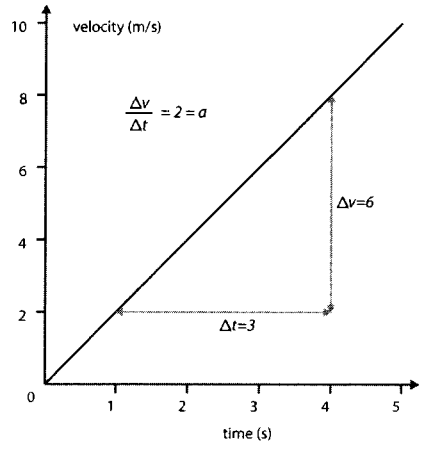

Figure 1.26. The velocity of the car plotted against time

You learned earlier how the equation $y = mx + c$ defines all straight lines in the 2D Cartesian plane, where m is the gradient and c the intersection on the y-axis. Because we can infer from Figure 1.26 that constant acceleration is always plotted as a straight line, we can relate that equation to the acceleration of the car. We know that the y-axis represents the velocity, v, and that the x-axis represents time, t. We also know that the gradient m relates to the acceleration. This gives the equation:

$$v = at + u \tag{1.81}$$

The constant u represents the velocity of the car at time $t = 0$, which can be shown as the intersection of the line on the y-axis. For instance, if the car in the example started off with a velocity of 3 m/s, then the graph would be identical but offset upward by 3 as shown in Figure 1.27.

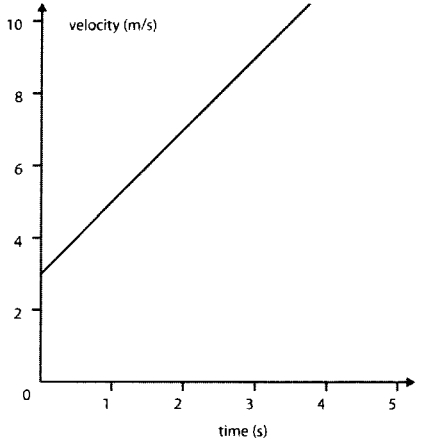

Figure 1.27. The same car but traveling with an initial velocity of 3 m/s at time $t = 0$

To test the equation let's determine what the velocity of a car starting with a velocity of 3 m/s and accelerating at 2 m/s^2 will be after 3 seconds. Plugging in the numbers to equation (1.81) gives:

$$v = 2 \times 3 + 3$$
$$v = 9 \text{ m/s} \tag{1.82}$$

This is exactly what we can infer from the graph. See Figure 1.28.

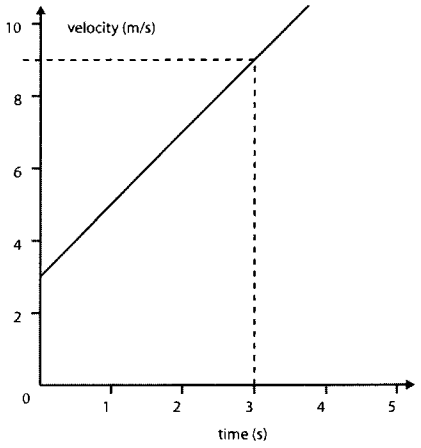

Figure 1.28

Another interesting thing about a velocity-time graph is that the area under the graph between two times is equivalent to the distance traveled by the object during that time. Let's look at a simple example first. Figure 1.29 shows the time versus velocity graph for a vehicle that spends 2 seconds at 4 m/s then stops.

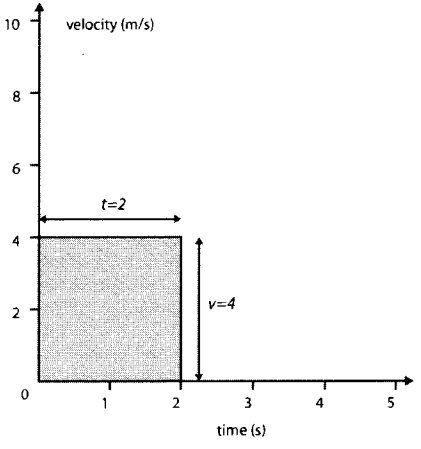

Figure 1.29

The area under the graph (the region shaded in gray) is given by height × width, which is equivalent to velocity × time, which as you can see gives the result of 8 meters. This is the same result from using the equation $\Delta x = v\Delta t$.

Figure 1.30 shows the example from earlier where a vehicle accelerates from rest with a constant acceleration of 2 m/s^2. Let's say we'd like to calculate the distance traveled between the times $t = 1$ and $t = 3$.

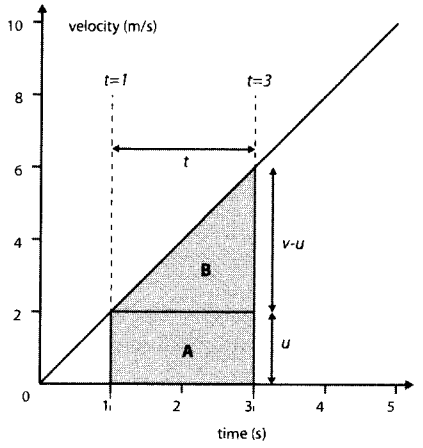

Figure 1.30

We know that the distance traveled between $t = 1$ and $t = 3$ is the area beneath the graph between those times. As is clearly shown in the figure, this is the sum of the areas of rectangle A and triangle B.

The area of A is given by the time displacement, t, multiplied by the starting velocity, u, written as:

$$Area(A) = \Delta t \times u \tag{1.83}$$

The area of B, a triangle, is half the area of the rectangle described by the sides of the triangle. The sides of the triangle are given by the time displacement, t, and the difference between the finish velocity and the start velocity, $v - u$. This can be written as:

$$Area(B) = \frac{1}{2}(v-u)\Delta t \tag{1.84}$$

Therefore, the total area under the graph between times $t = 1$ and $t = 3$, which is equivalent to the distance traveled, is the sum of these two terms, given as:

$$\Delta x = u\Delta t + \frac{1}{2}(v-u)\Delta t \tag{1.85}$$

We know that $v - u$ is equivalent to the change in velocity Δv, and that, from equation (1.80)

$$v - u = \Delta v = a\Delta t \tag{1.86}$$

This value for $v - u$ can be substituted into equation (1.85) to give us an equation that relates distance to time and acceleration.

$$\Delta x = u\Delta t + \frac{1}{2}a\Delta t^2 \tag{1.87}$$

Putting the numbers into this equation gives:

$$\begin{aligned} \Delta x &= 2 \times 2 + \frac{1}{2} \times 2 \times 2^2 \\ \Delta x &= 4 + 4 \\ \Delta x &= 8 \text{ m} \end{aligned} \tag{1.88}$$

We can do another useful thing with this equation: We can factor time out to give us an equation relating velocity to distance traveled. Here's how.

From equation (1.81) we know that:

$$\Delta t = \frac{v - u}{a} \tag{1.89}$$

We can substitute this value for Δt in equation (1.87) to give:

$$\Delta x = u\left(\frac{v-u}{a}\right) + \frac{1}{2}a\left(\frac{v-u}{a}\right)^2 \tag{1.90}$$

This nasty-looking equation can be simplified greatly. (If you are new to algebra I suggest trying to simplify it yourself. If you find yourself getting stuck, the full simplification is given at the end of the chapter.)

$$v^2 = u^2 + 2a\Delta x \tag{1.91}$$

This equation is extremely useful. For example, we can use it to determine how fast a ball dropped from the top of the Empire State Building will be traveling when it hits the ground (assuming no air resistance due to wind or velocity). The acceleration of a falling object is due to the force exerted upon it by the Earth's gravitational field and is equivalent to approximately 9.8 m/s^2. The starting velocity of the ball is 0 and the height of the Empire State Building is 381 m. Putting these values into the equation gives:

$$\begin{aligned} v^2 &= 0^2 + 2 \times 9.8 \times 381 \\ v &= \sqrt{7467.6} \\ v &= 86.41 \text{ m/s} \end{aligned} \tag{1.92}$$

The preceding equations hold true for all objects moving with a constant acceleration but of course it's also possible for objects to travel with varying acceleration. For example, an aircraft when taking off from a runway has a high acceleration at the beginning of its run (which you can feel as a force pushing you into the back of your seat), which decreases as the limits of its engine's power are reached. This type of acceleration would look something like that shown in Figure 1.31.

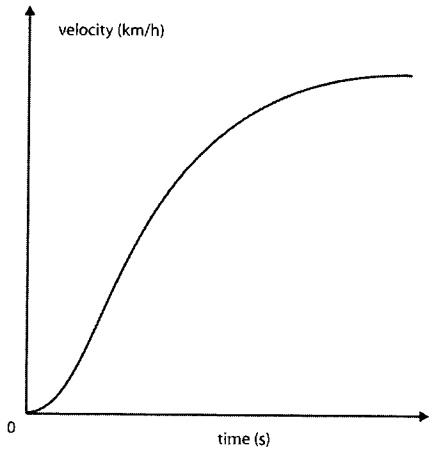

Figure 1.31. An aircraft accelerating up the runway

As another example, Figure 1.32 shows the velocity versus time graph for a car that accelerates to 30 km/h, brakes sharply to avoid a stray dog, and then accelerates back to 30 km/h.

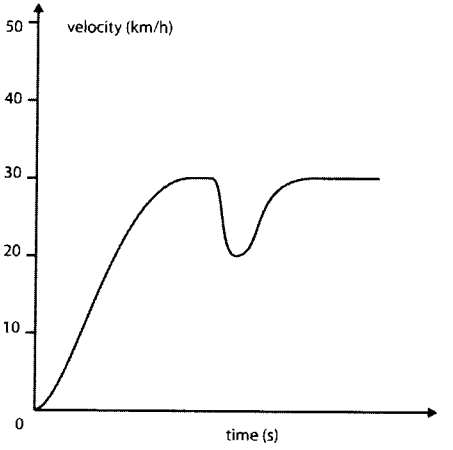

Figure 1.32

When you have varying accelerations like these it's only possible to determine the acceleration at a specific time. This is achieved by calculating the gradient of the tangent to the curve at that point.

Force

According to Isaac Newton:

> *An impressed force is an action exerted upon a body in order to change its state, either of rest, or of uniform motion in a right line.*

Therefore, force is that quality that can alter an object's speed or line of motion. Force has nothing to do with motion itself though. For example, a flying arrow does not need a constant force applied to it to keep it flying (as was thought by Aristotle). Force is only present where *changes in motion* occur, such as when the arrow is stopped by an object or when a drag racer accelerates along the strip. The unit of force is the Newton, abbreviated to N, and is defined as:

> *The force required to make a one-kilogram mass move from rest to a speed of one meter per second in one second.*

There are two different types of force: contact and non-contact forces. Contact forces occur between objects that are touching each other, such as the frictional force present between the snow and skis of a downhill skier. Non-contact forces are those that occur between objects not touching each other, such as the gravitational force of the Earth upon your body or the magnetic force of the Earth upon a compass needle.

It's important to note that many forces can act upon a single object simultaneously. If the sum of those forces equals zero, the object remains in motion with the same velocity in the same direction. In other words, if an object is stationary or moving in a straight line with a constant velocity, the sum of all the forces acting upon it must be zero. If, however, the sum of the forces is not equal to zero, the object will accelerate in the direction of the resultant force. This can be confusing, especially in relation to static objects. For instance, how can there be *any* forces acting upon an apple sitting on a table? After all, it's not moving! The answer is that there are two forces acting upon the apple: the force of gravity trying to pull the apple toward the Earth and an equal and opposite force from the table pushing it away from the Earth. This is why the apple remains motionless. Figure 1.33 shows examples of varying amounts of forces acting upon everyday objects.

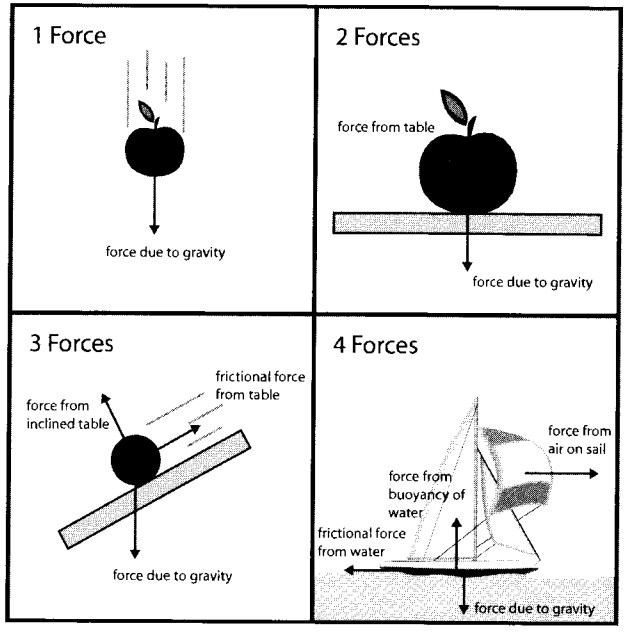

Figure 1.33. From left to right and top to bottom: a falling apple, an apple resting on a table, a ball rolling down an inclined table, and a yacht sailing on water

We know that if the sum of the forces acting upon an object is non-zero, an acceleration will be imparted in the direction of the force; but how much acceleration? The answer is that the amount of acceleration, a, is proportional to the object's mass, m, and to the total force applied, F. This relationship is given by the equation:

$$a = \frac{F}{m} \tag{1.93}$$

More commonly though, you will see this equation written as:

$$F = ma \tag{1.94}$$

Using this equation, if we know how fast an object is accelerating and its mass, we can calculate the total force acting upon it. For instance, if the boat in Figure 1.33 has a mass of 2000 kg, and it is accelerating at a rate of 1.5 m/s^2, the total force acting upon it is:

$$F_{total} = 2000 \times 1.5 = 3000 \text{ N}$$

Also using the equations for force, acceleration, velocity, and position, if we know how much force is acting on an object, we can determine the acceleration due to that force and update the object's position and velocity accordingly. For example, let's say you have a spaceship class with attributes for its mass, current velocity, and current position. Something like this:

```
class SpaceShip
{
private:

  vector m_Position;

  vector m_Velocity;

  float  m_fMass;

public:

  ...
};
```

Given the time interval since the last update and a force to be applied, we can create a method that updates the ship's position and velocity. Here's how:

```
void SpaceShip::Update(float TimeElapsedSinceLastUpdate, float ForceOnShip)
{
  float acceleration = ForceOnShip / m_fMass;
```

First of all, calculate the acceleration due to the force using equation (1.93).

```
  m_Velocity += acceleration * TimeElapsedSinceLastUpdate;
```

Next, update the velocity from the acceleration using equation (1.80).

```
  m_vPosition += m_Velocity * TimeElapsedSinceLastUpdate;
}
```

Finally, the position can be updated with the updated velocity using equation (1.77).

Summing Up

This chapter covers a lot of ground. If much of this stuff is new to you, you'll be feeling slightly confused and perhaps a little intimidated. Don't worry though. Soldier on and, as you read through the book, you'll see how each principle is applied to a practical problem. When you see the theory used in real-world contexts, you'll find it a lot easier to understand.

Simplification of Equation (1.90)

Let me show you how that pesky-looking equation is simplified. Here it is again in all its glory.

$$\Delta x = u\left(\frac{v-u}{a}\right)+\frac{1}{2}a\left(\frac{v-u}{a}\right)^2$$

First, let's work on the rightmost term. From the rule shown by equation (1.29) we can change the equation to read:

$$\Delta x = u\left(\frac{v-u}{a}\right)+\frac{1}{2}a\frac{(v-u)^2}{a^2}$$

We can now tidy up the a's a little:

$$\Delta x = u\left(\frac{v-u}{a}\right)+\frac{(v-u)^2}{2a}$$

Let's now dispose of the parentheses in the $(v-u)^2$ term using the rule given by equation (1.28).

$$\Delta x = u\left(\frac{v-u}{a}\right)+\frac{v^2+u^2-2vu}{2a}$$

Let's remove the other parentheses too.

$$\Delta x = \frac{uv-u^2}{a}+\frac{v^2+u^2-2vu}{2a}$$

Now to get rid of the fractional parts by multiplying every term by $2a$:

$$2a\Delta x = 2a\left(\frac{uv-u^2}{a}\right)+2a\left(\frac{v^2+u^2-2vu}{2a}\right)$$

$$2a\Delta x = 2uv-2u^2+v^2+u^2-2vu$$

Almost there now! We just need to group like terms together.

$$2a\Delta x = v^2-u^2$$

And rearrange to give the final equation.

$$v^2 = u^2+2a\Delta x$$