Chapter 1

Sound and Your Computer

If you've ever taken an introductory physics course, you've learned that sounds travels through the air in waves. You may even have debated the age-old conundrum, "If a tree falls in the forest . . . ." But just to make sure we're all speaking the same language, this chapter begins by describing the elementary characteristics of sound. Most of the chapter, however, deals with sound in personal computers and on the Internet.

What you'll learn:

- Some basic sound concepts: what it is and how we measure it
- How to digitize sound for computers and the Internet
- How to encode files using some popular audio codecs: PCM, MACE, ADPCM, μ-law, A-law, TrueSpeech, and MPEG
- What audio hardware and software you need to play and record sound on your computer

A Few Sound Concepts

Sound happens when some motion disturbs the air to create waves of compressed air that spread outward like ripples in a pond (see Figure 1-1). The source of sound is always a vibrating object, such as

vocal chords, a musical instrument, or an audio speaker membrane. Figure 1-1 shows just a few waves, but in reality the waves are generated as fast as the source vibrates, so the horn in the drawing is likely to be producing hundreds or thousands of waves per second, depending on what note it is playing. The waves radiate outward from the source until they strike our eardrums and make them vibrate. The vibrations are converted by the structure of the ear into signals that our brains interpret as sound.

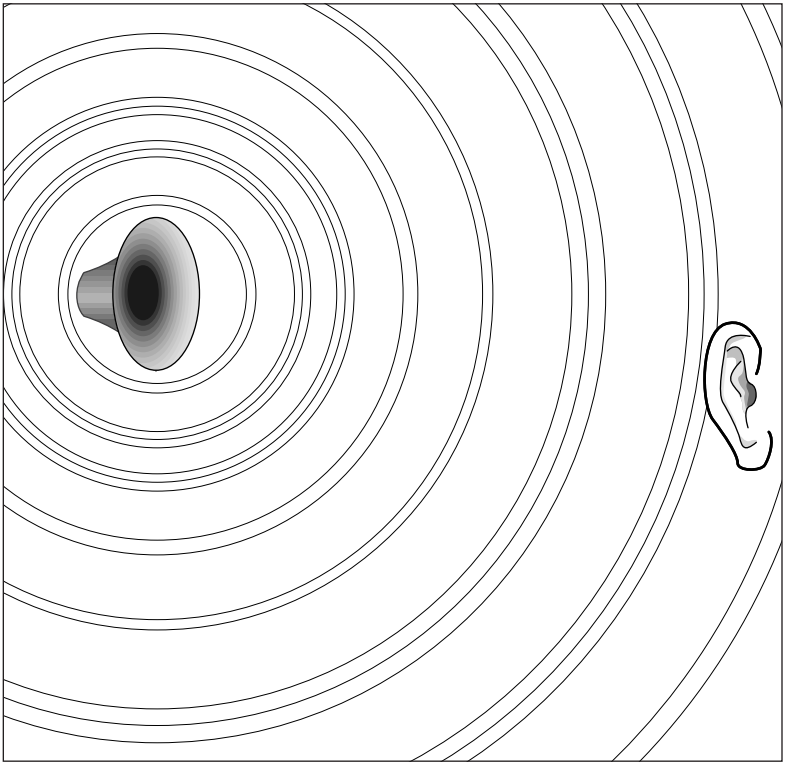

Figure 1-1 *Sound ripples outward from a source as waves of air.*

The characteristics of a sound are determined by the amplitude, frequency, and shape of the waves. Figure 1-2 shows what the cross-

section of three different sound waves might look like. Each drawing represents one hundredth of a second. That's a short period of time; if you could hear these waves played by themselves, each would sound like a tiny blip.

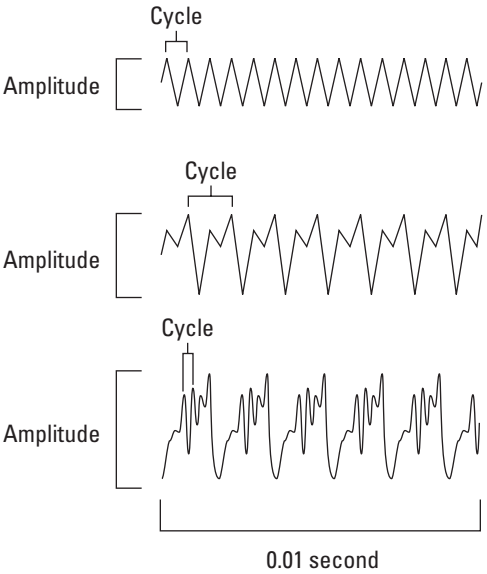

Figure 1-2 *Sound waves are characterized by amplitude, frequency, and shape.*

Amplitude refers to the height of the waves, which determine their volume. The higher a wave is, the louder it sounds. In Figure 1-2, the top wave is soft, the middle one is a little louder, and the bottom is quite a bit louder in general, although it has some soft moments.

Frequency refers to how close together the waves are, measured in *hertz* (Hz). One hertz is one cycle per second. A cycle goes from a peak to a valley and up to the next peak. In Figure 1-2, I have marked one cycle in each wave. High-frequency waves are close

together and produce high-pitched sounds. Low-frequency waves are further apart, producing low-pitched sounds. The top wave in Figure 1-2 represents a moderately high tone—you can see 14 cycles in one hundredth of a second, so the frequency is 1,400 Hz or 1.4 *kilohertz* (kHz). The middle is a lower note. You can see only four cycles in one hundredth of a second, so the frequency is 700 Hz. The bottom wave varies in frequency because it is not a simple note but a recording of me singing. (Don't worry, you don't have to listen to it.)

Note

The hertz is named after German physicist Heinrich Rudolf Hertz, who first described the properties of electromagnetic waves.

The air is filled with waves that you can't hear because they are not in the audible frequency range. Young human ears respond to waves from about 20 to 20,000 Hz. But as you get older, your lose the ability to hear the extreme frequencies, especially the high frequencies.

The *shape* of the wave determines the quality or *timbre* of the sound. In Figure 1-2, the top wave has a simple and regular shape. It would be a pure but relatively uninteresting tone, perhaps produced by a tuning fork or an electronic oscillator. The middle wave has more character, with a few peaks and valleys within the cycle. It might be produced by a flute or chime. The bottom wave demonstrates the highly complex and colorful timbre of the human voice in full song.

The shape of the wave, and thus the timbre it produces, are influenced by *harmonics*—the overtones that accompany a root tone. When someone blows the note A on clarinet, it's the harmonics added to the note by the material and shape of the instrument that tells us its a clarinet, not a flute or a saxophone. Each instrument has its own pattern of harmonics. In the middle wave in Figure 1-2, that slight dip near the peak of each cycle is caused by a simple harmonic added to the root tone. The bottom wave is full of harmonics

pulling the wave into interesting little nooks and crannies. Harmonics is a fascinating study where not all the answers are known, as you'll see in Chapter 3, where we talk about synthesizing musical instruments on your computer.

Digitizing Sound

When we use a microphone to record sound, the microphone converts the sound waves into an electromagnetic waveform much like those illustrated in Figure 1-2. We use the term *analog* to describe this type of signal because it continuously varies. In an analog recording system, the signal is processed, stored, and played back in analog form. The most common analog recording media for sound are phonograph records and audiotape.

The pros and cons of analog recordings is argued endlessly by audiophiles, but for our purposes in this book, one disadvantage outweighs everything else. Today's personal computers are *digital* devices. They process and store numeric data only. Everything that you input to a personal computer — including text, images, and sound — must be converted into digital form. Everything you transmit or receive via the Internet mustst be digital. Your computer cannot process analog waveforms, nor can they be transmitted on the Internet.

Sampling

We convert analog sound waves into digital data by a process called *sampling*, illustrated in Figure 1-3. The wave in Figure 1-3 is from the same sound clip as the bottom wave in Figure 1-2, but this time I spread it out to .001 second (a thousandth of a second) so you could clearly see the samples. Each dot represents a point where the wave was measured and a number recorded. The number scale on the right side shows the numbering system that was used, from -128

to 127. I marked a few numbers in the wave to show you how they are generated. The scale along the bottom identifies the frequency of the samples. You can see 22 samples in the drawing.

The result of sampling is a series of numbers. The wave in Figure 1-3, for example, yields the numbers -70, -70, -70, -65, -54, -40, and so on. Now we have data that a digital computer can sink its teeth into. It could be stored in a file, copied to a disk, e-mailed to our friends, and posted on a Web page. And eventually, it could be turned back into an analog waveform again. When we play back the recording, the player uses the sampled numbers to re-create the original waveform — more or less, as you'll see in the next section.

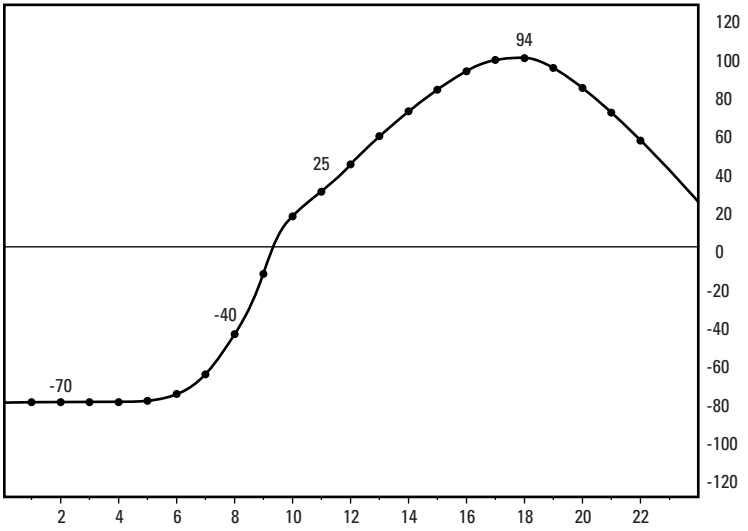

Figure 1-3 *Analog waveforms are digitized by sampling the waveform at regular intervals.*

Sampling rate

Two factors control how well the digital data reproduces the original analog waveform: the sampling rate and the sample size. The *sampling rate* describes the number of samples that are taken per second,

expressed in kilohertz (kHz). The example in Figure 1-3 uses a sampling rate of 22,050 samples per second. The higher the sampling rate, the closer we approximate the original waveform. Compare the wave in Figure 1-3 to the one in Figure 1-4. Both were taken from the same original waveform, but the wave in Figure 1-4 was sampled at only 4 kHz, producing only a rough similarity to the original wave. The most common sampling rates used today are 8, 11.025, 22.050, and 44.1 kHz. The lowest rate of 8 kHz should be used for speech only. Audio CDs are recorded at 44.1 kHz, with superb results.

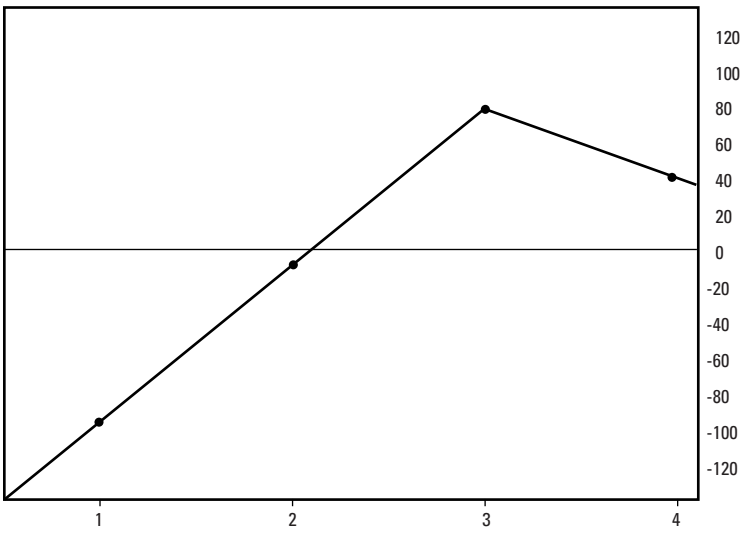

Figure 1-4 *Slower sampling rates produce significantly lower-quality sound.*

Note

For convenience throughout this book, round off the sampling rates to 11, 22, and 44 kHz.

When talking about sound samples, the term *bandwidth* refers to the range of frequencies of the recording. You could say that the human bandwidth is 19,980 Hz, since we could hear from about 20

Hz to about 20,000 Hz when we were young. But we usually ignore the low end of the range and simply say that the bandwidth is 20 kHz. Similarly, if a recording includes frequencies up to 10 kHz, its bandwidth is 10 kHz. An important rule called the *Nyquist Theorem* says that a sound clip's sampling rate must be twice its bandwidth. Or, to put it backwards, the bandwidth of a recording will be half the sampling rate you used. Thus, to adequately sample a clip with a 10 kHz bandwidth, you need a sampling rate of at least 20 kHz. Plain old telephone service provides a bandwidth of about 3.5 kHz, so a sampling rate of 8 kHz is more than adequate for a telephone-quality recording. A sampling rate of 44 kHz easily covers the entire human bandwidth of 20 kHz.

Note

The term *bandwidth* is bandied about a lot on the Internet. Most often, you'll hear it used to express connection speeds in *kilobits per second* (Kbps). Don't confuse that with sampling bandwidth, which is expressed in *kilohertz*.

Sample size

The *sample size* describes the numeric accuracy of each sample. The sound clip in Figure 1-2 uses a scale from -128 to 127. But since analog waveforms are continuous, many sampled values actually fall between two numbers. For example, a sample value may be part way between 2 and 3. We have to round the value either up to 3 or down to 2. This introduces a *quantization error* into the data that results in noise (an audible hiss) when the sampled recording is played back.

Suppose instead that we use a scale from -32,768 to 32,767. Fewer samples will fall between the numbers on our scale. And when they do, the quantization errors are smaller and result in less noise. The general rule of thumb is, larger sample sizes produce less noise and more accurate sound.

Note

Sample size is also known as *bit depth* or *sample resolution*.

Why do we use such strange ranges of numbers? Why not use -200 to 199 or -10,000 to 9,999? The answer lies in the way that computers represent numbers. Because computers have to represent numbers with electrical voltages, and switches, and magnetic spots, they use the binary number system to represent values. A binary digit, known as a bit, can have only two values, 0 or 1. We have to combine bits into bytes to be able to represent meaningful data. In a personal computer, eight bits make a byte, which is capable of storing 256 values, from 0 to 255 for unsigned numbers or from -128 to 127 for signed numbers. Some audio clips use an 8-bit sample size, resulting in a low-quality recording. It's fine for speech, but music sounds pretty bad. Higher quality recordings use 12 or 16 bits to reduce the amount of noise and more accurately represent the waveform. With 16 bits, you can represent 65,535 values, -32,768 to 32,767 for signed numbers, which is more than adequate for CD-quality sound.

The sample size also determines the *dynamic range* of the recording — the range of volume from the softest to the loudest sound. Each bit that you add to the sample size adds 6 decibels (dB) to the dynamic range, which effectively doubles the dynamic range. Sixteen-bit samples allow a dynamic range of 96 dB, which is just about the dynamic range of the human ear. By comparison, audiotape achieves a dynamic range of only 65 dB.

So we just use a 44 kHz sampling rate and a 16-bit sample size for fantastic recordings. Right? Well yes . . . but. Consider a three-minute song. That's 180 seconds times 44,100 samples per second times 2 bytes per sample equals 15,876,000 bytes! Do you want to download a 15MB file just to hear your nephew sing "Old MacDonald"? Neither do I. The highest-quality sampling rate and sample size are perfect for recording audio CDs. But for files that we share over the Internet, we have to find a trade-off that produces reasonable file sizes with acceptable — not perfect — quality. Most often, we record at 11 or 22 kHz with an 8-bit sample size for speech and a 12-bit or 16-bit sample size for music.

Channels

My calculations for that three-minute, 15MB song assume that it is a monaural (mono) recording. If it's recorded in stereo, as most audio CDs are, it takes up 30MB. Stereo requires two recorded channels — one for the left track and one for the right — so a stereo recording is twice as large as a mono recording.

Some recordings have more than two channels. 3D sound, for example, uses four channels: front left and right, and rear left and right. Surround Sound uses six channels. Other forms of spatial sound use anywhere from three to 16 channels — used mostly in games and simlations. Just think how much RAM and disk space you'd need to record 16 channels!

Using Audio Codecs to Encode Files

A software module that encodes and decodes audio data is known as a *codec*, for *co*der/*dec*oder. In other words, it does the sampling when you record an audio file, and interprets the samples when you play back the file. Sound recorders such as Windows Sound Recorder or Macintosh Sound Manager make use of a variety of codecs so you can choose the sampling rate, sample size, and number of channels you want.

Many codecs also compress the sampled data to keep file sizes as small as possible. Compression reduces a file's size by removing unnecessary data. Compression methodologies fall into two general categories: lossless and lossy. A *lossless* compression method removes data in such a way that it can be restored completely by decompression. Lossless techniques are crucial for compressing text files, drawings, databases, and other documents where missing data damages the document. Programs such as PKZip and StuffIt provide lossless compression.

But lossless compression is not nearly as effective as *lossy* compression techniques, in which some data is permanently lost. Sound

clips, videos, and photographs lose some quality when data is permanently removed, but they are still usable. The trick is to lose the least noticeable data, and that's what audio codecs strive to do.

Pulse Code Modulation

Pulse Code Modulation (PCM) is the simplest codec, as it just samples the sound at whatever rate, size, and number of channels you specify. It does not attempt to compress the sampled data. It's the most commonly used codec, especially in Windows systems, even though it offers no compression.

ADPCM

When encoding human speech, *Adaptive Differential PCM* (ADPCM) manages to get near 16-bit accuracy using only 4-bit samples by storing not the samples themselves but the differences from one sample to the next. Rather than store -70, -70, -70, -65, for example, it stores -70, 0, 0, 5. To keep the numbers small, it also takes advantage of the fact that human speech waveforms are highly predictable. After sampling the beginning of a section, ADPCM can predict what the next samples will be. This isn't quite the same as when your mother finishes your sentences for you — ADPCM works at the microsecond level. Take a look at the bottom waveform in Figure 1-2 and you can see how repetitive the human voice is at that level. The inaccuracy of the predictions introduce distortion in the higher frequencies of the recording. The higher the sampling rate, the less distortion you get. At 11 kHz, for example, distortion is noticeable, but at 44 kHz it almost disappears.

Since it can reduce a 12-bit or 16-bit sample size to four bits, ADPCM codecs reduce a file's size to one third to one fourth compared to PCM encoding. But it works only on files consisting of human speech. Music is much more unpredictable, and ADPCM shows no noticeable savings and a considerable reduction in quality when applied to music clips.

μ-law and A-law

μ-law encoding was originally developed for telephone systems, but it is popular for Internet audio. It reduces 16-bit samples to 8-bit samples, yielding 50 percent compression. A variation of *μ*-law called A-law is popular in Europe, where it was developed for European telephone systems.

Note

The Greek letter μ is pronounced "mu," and you will often see *μ*-law written as mu-law or u-law in situations where printing a *μ* is difficult or impossible.

μ-law uses a method called *companding* (*com*pressing/ex*panding*) to get 16-bit accuracy in eight bits, thus saving 50 percent of the file size. Companding uses a logarithmic sampling scale instead of the usual linear scale. Figure 1-5 shows how this works. Notice that the numbers are close together near the zero point, where the volume is soft, and farther apart at the extremes, where the volume is loud. The widely spaced numbers introduce more noise than the closely spaced numbers, but you won't hear it because of the loudness of the sound.

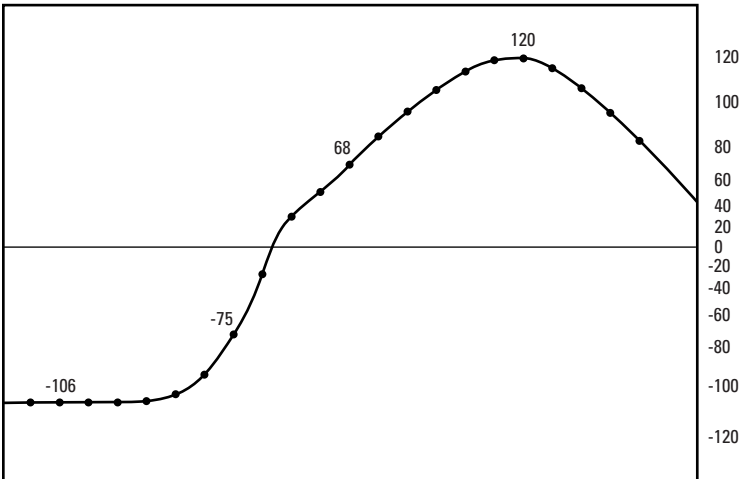

Figure 1-5 *μ-law's companding technique spaces the sampling scale to allow noise at volume extremes, where you won't hear it.*

MACE

The *Macintosh Audio Compression and Expansion* (MACE) codec was developed by Apple specifically for Macintosh. It supports an 8-bit sample size only, and you can select either 3:1 or 6:1 compression. That is, you can compress the file to one third or one sixth of its PCM size. As with all lossy compression techniques, some sound quality is lost by the compression/decompression process. As you might expect, you lose more fidelity with 6:1 compression than with 3:1 compression. The 6:1 compression ratio, which cuts the sampling rate in half to double the compression ratio, is recommended for speech only.

TrueSpeech

TrueSpeech was developed by DSP Group, Inc., for its telephony applications. But since Windows 95, 98, and NT include the TrueSpeech codec with Sound Recorder and Media Player, it is being used to compress many types of speech-only files. It achieves a whopping 93 percent compression, so it reduces a 15MB file to just 1MB. That's considerably more compression than ADPCM or μ-law can achieve.

TrueSpeech takes a completely different approach to recording sound data. It does not record sound samples, but analyzes the vocal characteristics of the talker and records information on how to recreate what was said and how it was said. This takes significantly less space than recording sound samples, but the quality suffers.

MPEG

The Motion Picture Experts Group (MPEG) was commissioned to develop video compression standards, but since video includes audio, they ended up developing audio compression standards, too. MPEG so far has created three general standards, known as MPEG-1, MPEG-2, and MPEG-3. (A fourth is on the way.) MPEG-1 defines three audio standards known as Layer 1, 2, and 3,

representing increasing levels of compression (and also complexity). The MPEG-1 Layer 3 standard provides the highest compression and audio quality available on the Internet, but at a cost that some users aren't able to pay — encoding a file takes so much computation that many personal computers simply can't handle the job. But if all you want to do is listen to MPEG-encoded files, not encode them yourself, your computer should be able to handle the decoding, which isn't nearly as difficult.

MPEG compression schemes examined much psychological research in acoustics — known as psychoacoustics — to determine which parts of a sampled sound are most important to human hearing, and which parts can be thrown away. But the answers weren't conclusive, and MPEG made its final decisions by simply trying out different techniques. The results are impressive.

The MPEG codec works by taking a second look at the problem of quantization noise. As I said earlier, you need larger sample sizes to avoid noise from the encoding process. But psychoacoustics reveals that louder tones mask nearby softer tones. In other words, if you play a loud note at 440 Hz and a softer note at 445 Hz, the human ear hears only the 440 Hz tone. MPEG takes advantage of this fact to permit noise where it will be masked by legitimate sounds. It turns out that a large proportion of an audio file can have its sample size reduced with little or no audible effect. And as you know, reducing the sample size produces less data and results in smaller files.

Your Sound Hardware and Software

If you read the Introduction, you know that this book is not about hardware. But you should be aware of what hardware you need to record and play back sound on your computer. Without getting into a lot of details, brand names, and specifications, I'll just briefly overview the bare minimum. Most of these items are built into a

Macintosh, although they can be supplemented or replaced with higher-quality ones. If you have a PC, these may have been installed at the factory, or you may need to buy and install them yourself.

- A sound board, the newer the better. The capabilities of your sound board determine how mch you can do in recording and playing back. If you have an older 8-bit sound board, for example, you won't be able to record 16-bit files, and you can only play back 16-bit files with 8 bits.
- An onboard synthesizer for playing MIDI, MOD, and karaoke files. It's probably on your sound board, but it might be directly on your motherboard.
- A CD-ROM drive if you want to play and record from audio CDs. If you want to record your own audio CDs, you need a CD-R (CD write once read many times) or CD-RW (CD rewritable) drive.
- Speakers or headphones. Speakers may either be built into your monitor, or plugged into your sound board. If you have a PC, the tinny little speaker that beeps when you start up is not good enough. Many CD-ROM drives include a headphone jack.
- A microphone for recording voice messages. You don't need a microphone to record from your CD player, though.

In terms of software, you need at least one recorder and enough players to handle the various types of sounds you want to play, such as sampled sounds, MIDIs, and audio CDs. You may already have several recorders and players. Windows comes with Sound Recorder, CD Player, and Media Player. Macintosh includes Sound Manager, which both records and plays back. It's built into System 7.5.3 and higher, and can be added as an extension to earlier versions of Mac OS. Macintosh OS 7 and higher also includes AppleCD Audio Player.

Your sound board might also come with its own software, including a recorder and one or more players. Be sure to check any disks

that came with the player to see what software they contain. If your sound board was installed at the factory, look for a folder on your hard drive that contains its files.

If you plan to do any serious recording, you'll also need a sound editor. Windows Sound Recorder can do some basic editing tasks. Macintosh Sound Manager's recorder does not. In either case, you'll probably want to get a better sound editor.

Now here's the good news. You're holding a great collection of recorders, players, and editors in your hand right now. The CD-ROM at the back of this book includes some of my favorite audio software for Windows and Macintosh. And the rest of this book shows you, chapter by chapter, how to use them. I hope you'll install them and try them out as you read the chapters that describe them.

What's Next?

Chapter 2 explores the popular audio file formats for storing sampled sounds on the Internet: WAV, SND, AU, and several other formats.