This chapter introduces the method of sound sampling and several forms of additive synthesis. These techniques are fundamental to computer music and should be understood by every musician interested in synthesized sound.

Sampling Synthesis

In popular parlance, *sampling* means making a digital recording of a relatively short sound. The term "sampling" derives from established notions of digital *samples* and *sampling rate*. Sampling instruments, with or without musical keyboards, are widely available. All sampling instruments are designed around the basic notion of playing back prerecorded sounds, shifted to the desired pitch.

Sampling synthesis is different from the classical technique of fixed-waveform synthesis explained in chapter 1. Instead of scanning a small fixed wavetable containing one cycle of a waveform, a sampling system scans a large wavetable that contains thousands of individual cycles—several seconds of prerecorded sound. Since the sampled waveform changes over the attack, sustain, and decay portion of the event, the result is a rich and time-varying sound. The length of the sampling wavetable can be arbitrarily long, limited only by the memory capacity of the sampler. Most samplers provide an interface to an optical or magnetic disk drive so that groups of samples can be loaded into the sampler relatively quickly.

Musique Concrète and Sampling: Background

Composed manipulation of recorded sounds dates back at least as early as the 1920s, when composers such as Darius Milhaud, Paul Hindemith, and Ernst Toch experimented with variable-speed phonographs in concert (Ernst 1977). Magnetic tape recording, originally developed in Germany in the 1930s, permitted cutting and splicing, and therefore flexible editing and rearrangement of sequences of recorded sounds. Tape recorders were not available to musicians until the post–World War 2 period.

After experiments with variable-speed phonographs in the late 1940s, Pierre Schaeffer founded the Studio de Musique Concrète at Paris in 1950 (see figure 4.1). He and Pierre Henry began to use tape recorders to record and manipulate *concrète* sounds. *Musique concrète* refers to the use of microphone-recorded sounds, rather than synthetically generated tones as in pure electronic music. But it also refers to the manner of working with

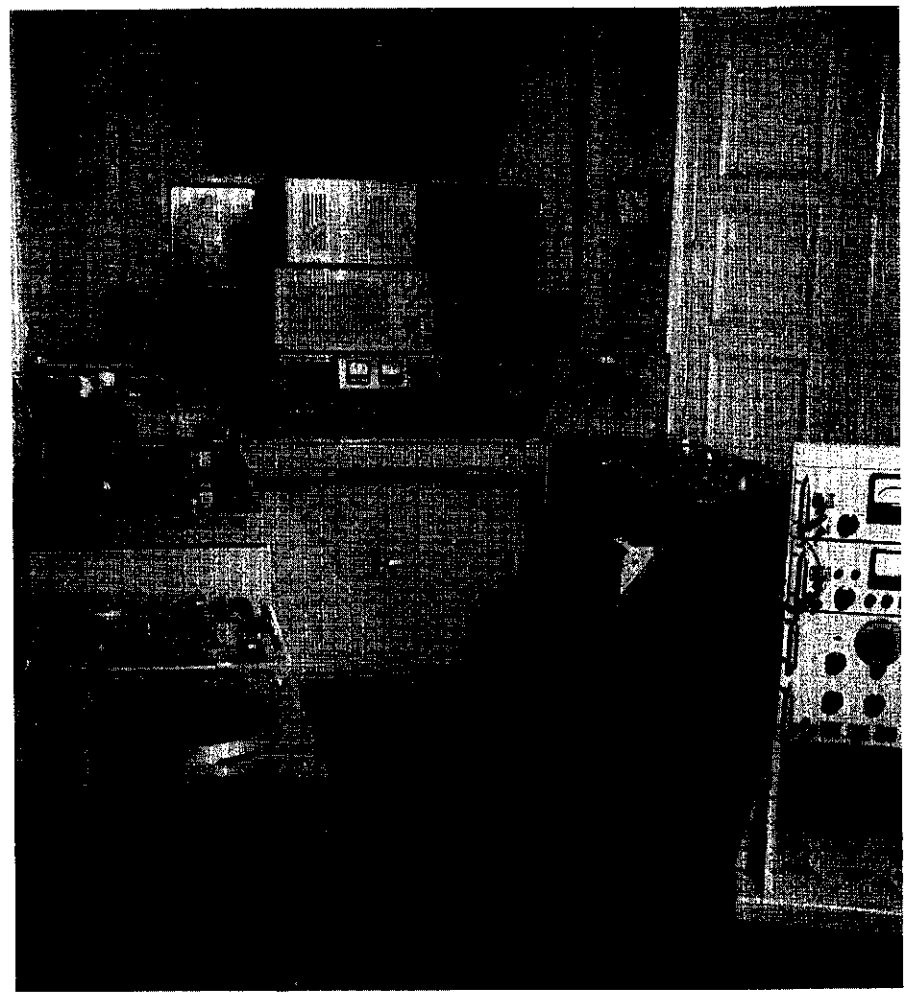


Figure 4.1 Pierre Schaeffer's studio for musique concrète at rue de l'Université, Paris, 1960. The studio features three tape recorders on the left, along with a disk turntable. On the right is another tape recorder and the multiple-head Phonogène device (see figure 4.2). (Photograph courtesy of the Groupe de Recherches Musicales, Paris.)

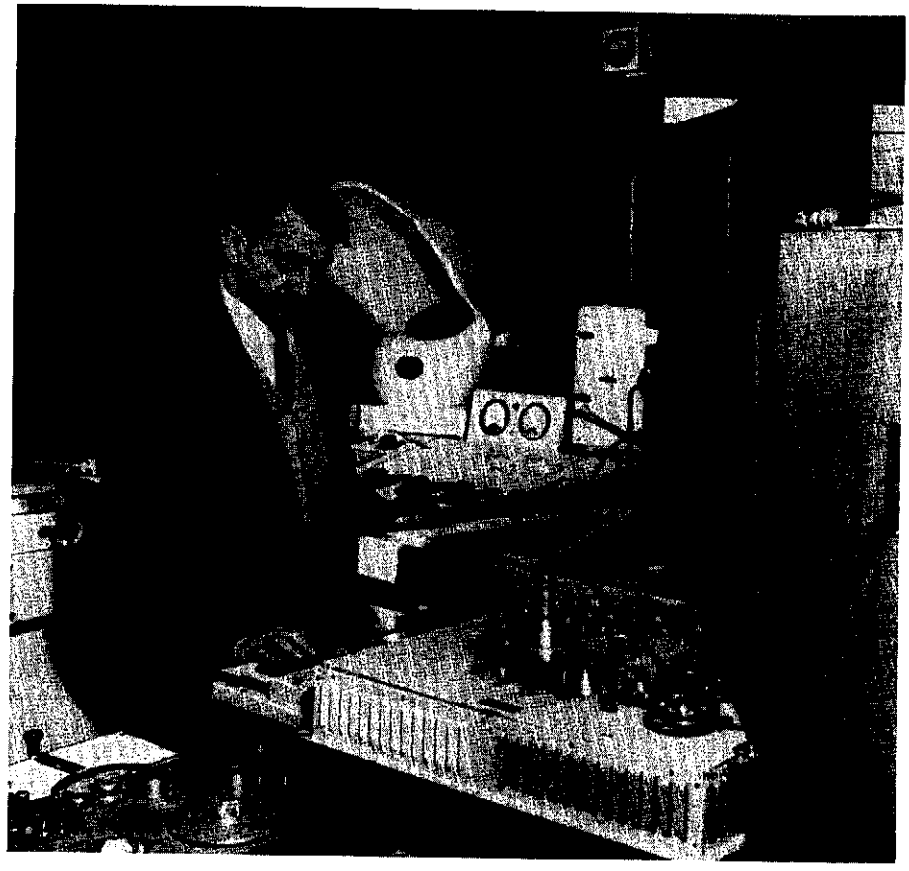


Figure 4.2 Pierre Schaeffer with the Phonogène, a tape-based transposer and time-stretcher, 1953, Paris. (Photograph by Lido, supplied by the courtesy of the Groupe de Recherches Musicales.)

such sounds. Composers of musique concrète work directly with sound objects (Schaeffer 1977; Chion 1982). Their compositions demand new forms of graphic notation, outside the boundaries of traditional scores for orchestra (Bayle 1993).

Modern sampling instruments are based on a principle used in photo-electric and tape-loop devices such as the Edwin Welte's Light-tone Organ (Berlin, 1930s), Sammis's Singing Keyboard (Hollywood, 1936), Pierre Schaeffer's Phonogène (figure 4.2, Paris, early 1950s), Hugh LeCaine's Special Purpose Tape Recorder (Ottawa, 1955), the Chamberlain (Los Angeles, late 1960s), and the Mellotron (London, early 1970s). These devices played either optical disks (encoded with photographic images of waveforms), or magnetic tape loops of sound. Depending on the disk or tape selected and

the key pressed on the musical keyboard, a playback head inside these instruments would play the sound on the disk or tape running at a rate that matched the pitch specified by the depressed key.

The designer of the Singing Keyboard, Frederick Sammis, described the potential of such an instrument in 1936:

Let us suppose that we are to use this machine as a special-purpose instrument for making "talkie" cartoons. At once it will be evident that we have a machine with which the composer may try out various combinations of words and music and learn at once just how they will sound in the finished work. The instrument will probably have ten or more sound tracks recorded side by side on a strip of film and featuring such words as "quack" for a duck, "meow" for a cat, "moo" for a cow.... It could as well be the bark of a dog or the hum of a human voice at the proper pitch. (Frederick Sammis, quoted in Rhea [1977].)

Perhaps the most famous predigital "sampler" was the Mellotron—an expensive instrument containing a number of rotating tape loops. The Mellotron enjoyed popular success with rock groups in the 1970s. They used the instrument to create "orchestral" or "choral" backings on popular songs. But the complicated electromechanical design of the Mellotron made it a temperamental instrument. The tape loops wore out due to head abrasion, and there were failures in the moving parts used in selecting and running multiple tape loops. Despite their problems, Mellotrons piqued interest in the prospect of playing recorded natural sounds on stage.

Several years later, the rise of digital electronics made it feasible to record and store sound in digital memory chips. In the 1970s, however, memory chips were still expensive, so the first "sampling" devices were simple delay units for the recording studio, designed to enrich a sound by mixing it with a sampled version of itself delayed by several milliseconds. (See chapter 10 for a discussion of delay effects.) As memory became cheaper it became possible to store several seconds of sounds for playback on a musical keyboard–based digital sampling instrument. The Fairlight Computer Music Instrument (CMI) was the first commercial keyboard sampler (1979, Australia). The CMI had a resolution of 8 bits per sample and cost over $25,000. Taking advantage of declining costs for digital hardware, the E-mu Emulator (figure 4.3), introduced in 1981, lowered the cost of 8-bit monophonic sampling (Vail 1993). For about $9000, the Emulator offered a total of 128 Kbytes of sample memory.

In order to create a commercial sampling instrument, three basic issues must be addressed: looping, pitch-shifting, and data reduction, which we discuss in the next three sections.

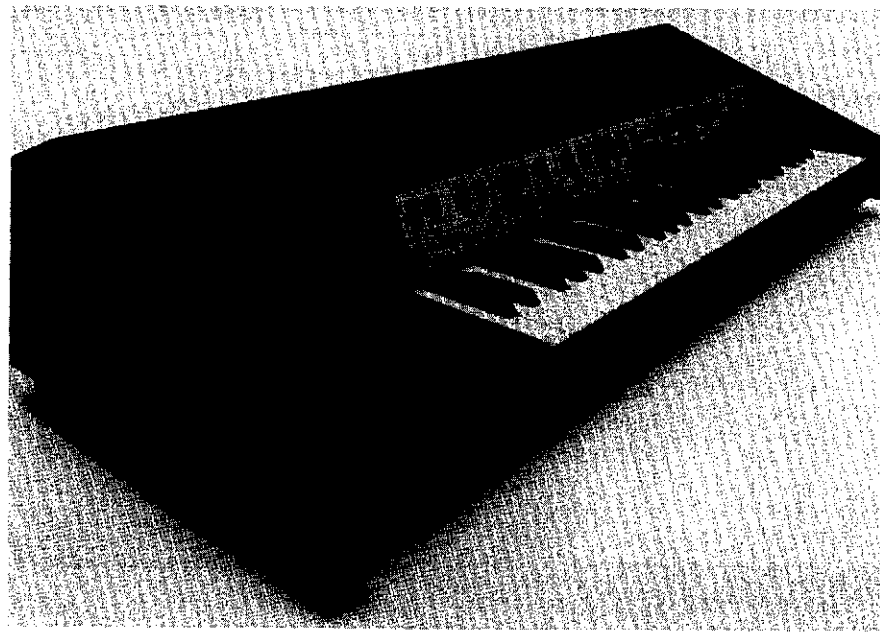


Figure 4.3 The E-mu Emulator sampling keyboard instrument (1981).

Looping

Looping extends the duration of sampled sounds played by a musical keyboard. If the musician holds down a key, the sampler should scan "seamlessly" through the note until the musician releases the key. This is accomplished by specifying beginning and ending *loop points* in the sampled sound. After the attack of the note is finished, the sampler reads repeatedly through the looped part of the wavetable until the key is released; then it plays the note's final portion of the wavetable.

Factory-supplied samples are often "prelooped." But for newly sampled sounds, the responsibility of specifying the begin and end loop points is usually left to the musician who sampled them. Creating a seamless but "natural" loop out of a traditional instrument tone requires care. The loop should begin after the attack of the note and should end before the decay (figure 4.4).

Some samplers provide automatic methods for finding prospective loop points. One method is to perform *pitch detection* on the sampled sound (Massie 1986). (See chapter 12 for a discussion of pitch detection methods.) The pitch detection algorithm searches for repeating patterns in the wavetable that indicate a fundamental *pitch period.* The pitch period is the time

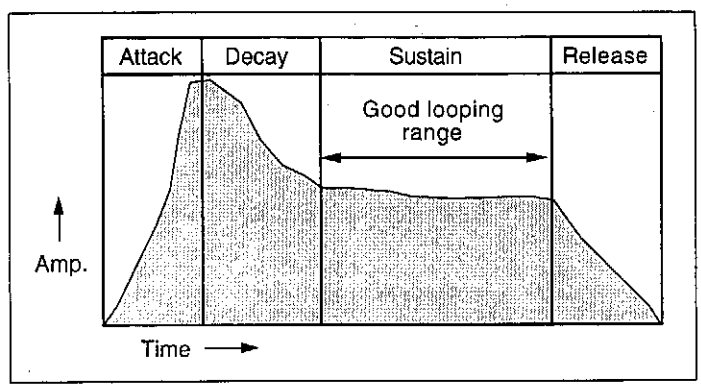


Figure 4.4 Sound with a characteristic ADSR amplitude envelope. The best area for a smooth loop is the sustained portion.

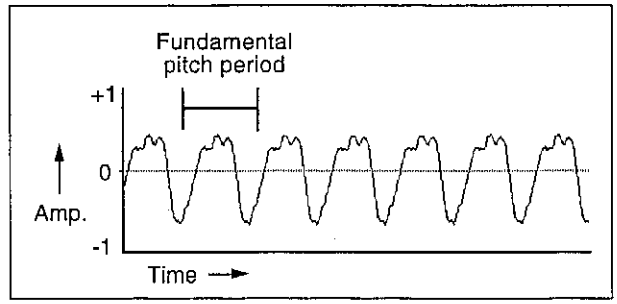


Figure 4.5 The fundamental pitch period is equal to one cycle of a periodic waveform, in this case, a waveform emitted by an alto saxophone.

interval that spans one cycle of a periodic waveform (figure 4.5). Once the pitch has been estimated, the sampler suggests a pair of loop points that match some number of pitch periods in the waveform. This kind of looping algorithm tends to generate smooth loops that are constant in pitch. If the body of the loop is too short, however, the result is similar to the sterile tones of fixed-waveform synthesis. For example, a loop encompassing one or two pitch periods of a violin note negates the time-varying qualities of a bowed string, yielding an artificial tone that has lost its identity.

The beginning and ending points of a loop can either be *spliced* together at a common sample point or *crossfaded.* A splice is a cut from one sound to the next. Splicing waveforms results in a click, pop, or thump at the splice point, unless the beginning and ending points are well matched. Crossfading means that the end part of each looped event gradually fades out while the beginning part slowly fades in again; the crossfade looping process repeats

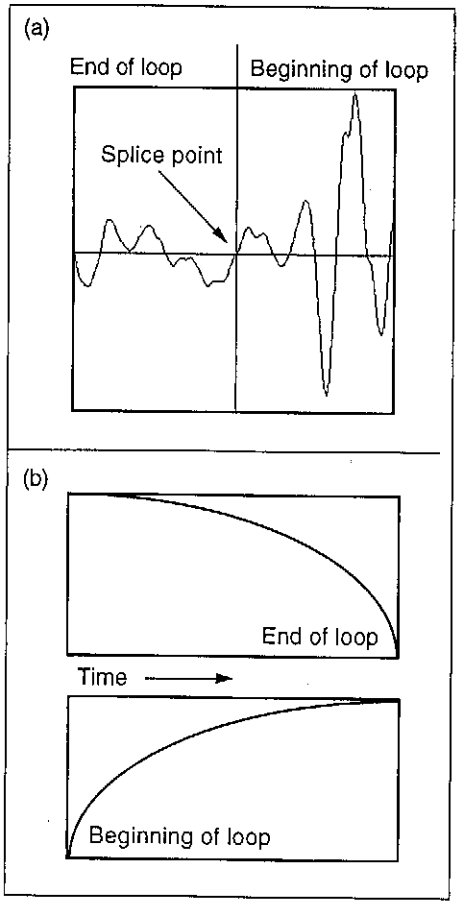


Figure 4.6 Splicing versus crossfading loops. (*a*) A vertical splice of two parts of a waveform at a common zero point. The ending point of the loop splices to the beginning of the same wavetable loop. (*b*) Crossfade looping can be viewed as a fade out of the end of the loop overlapped by a fade in of the beginning of the loop.

over and over as the note is sustained (figure 4.6). Typical crossfade times range from 1 to 100 ms, but crossfades can be extended as long as is desired.

When none of these techniques create a smooth loop, due to vibrato or other variations in the signal, more complicated methods can be brought to bear, such as *bidirectional looping*. A bidirectional loop alternates between forwards and backwards playback (figure 4.7a). Forwards and backwards loops can be layered on top of one another to mask discontinuities in either direction (figure 4.7b). Even more elaborate looping techniques based on spectrum analysis are available. For example, one can analyze the sound, randomize the phase of each spectral component in the loop, and resynthesize (Collins 1993).

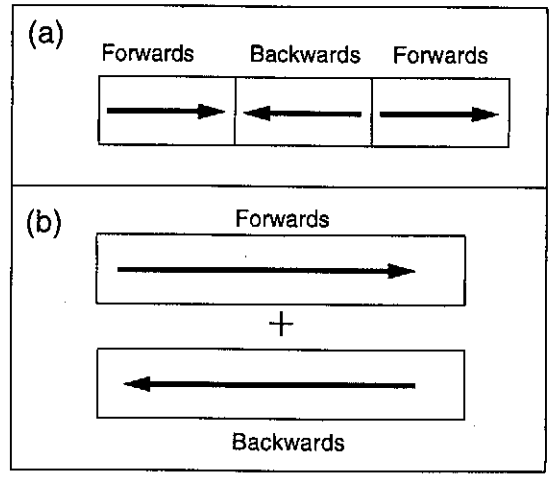


Figure 4.7 Looping methods for smoothing out variations. (*a*) Three cycles of a bidirectional loop. (*b*) In a layered forwards and backwards loop the two versions are added together.

Pitch-shifting

In an inexpensive sampler it may not be possible to store every note played by an acoustic instrument. These samplers store only every third or fourth semitone and obtain intermediate notes by shifting the pitch of a nearby stored note. If you record a sound into a sampler memory and play it back by pressing different keys, the sampler carries out the same pitch-shifting technique. A side effect of simple pitch shifting is that the sound's duration increases or decreases, depending on the key pressed. Chapter 10 describes methods of pitch shifting that preserve the original duration of the sound. Here we stay with simple pitch shifting.

Two methods of simple pitch shifting exist.

Method 1. Varying the clock frequency of the output DAC changes the playback sampling rate; this shifts the pitch up or down and changes the duration.

Method 2. Sample-rate conversion (*resampling* the signal in the digital domain) shifts the pitch inside the sampler and allows playback at a constant sampling rate for all pitches.

Some samplers employ the first method, others use the second method. Both of these methods are called *time-domain* techniques, since they operate directly on the time-domain waveform. This is different from the *frequency-*

domain pitch-shifting techniques discussed in chapter 10. Next we compare these two time-domain methods.

Since method 1 changes the playback sampling rate, it requires a separate DAC for each note that can be played simultaneously on the musical keyboard (typically up to 10 DACs). Each DAC must permit a variable clock rate and have a variable-frequency smoothing filter associated with it. For full transposibility, the DAC and the filter must work over extremely wide operating ranges. For example, if a tone with a pitch of 250 Hz sampled at 44.1 KHz is shifted up six octaves to 16 KHz, the clock frequency of the output DAC must shift up six octaves to 2.82 MHz.

Because of these ranges, either expensive parts must be used or (more typically) the audio performance of the system must be compromised in some ways. For example, one sampler that employs this pitch-shifting method allows only a single semitone of transposition (less than a 6% change of clock frequency) for sounds recorded at its highest sampling rate of 41.67 KHz. In this case the DAC and the filter are never forced to work at a sampling rate higher than 44.1 KHz. Other samplers do not permit any transposition above an arbitrary frequency.

Pitch-shifting method 2 performs sample-rate conversion. Sample-rate conversion, in effect, resamples the signal in the digital domain. This is essentially the same pitch variation technique as used in wavetable-lookup synthesis described in chapter 3. The output DAC's sampling frequency remains constant. Speeding up a sound and increasing its pitch is achieved by resampling at a lower sampling rate. This is analogous to time-lapse photography in which the frame rate is slowed down to achieve a speedup on playback. In a digital audio system samples are skipped in resampling. The number of samples that are skipped is proportional to the amount of pitch shifting that is desired (just as in wavetable-lookup synthesis). The process of skipping samples in resampling is called *decimation* (figure 4.8a). Resampling with decimation is also called *downsampling*. For example, to shift the pitch upwards by three octaves, the signal is downsampled by reading every third sample in playback.

To lower the pitch of a sound and slow it down, the sound is resampled at a higher frequency to stretch it out. This is analogous to the operation of a slow-motion camera that speeds up the frame rate to achieve a slowdown upon playback. In a digital audio system, new intermediate samples are inserted between existing samples by means of *interpolation* (figure 4.8b). Resampling with interpolation is also called *upsampling*.

The relationship between the various resampling rates and pitch-shifting can be confusing at first, because pitch-shifting method 1 and method 2 seem to go in opposite directions to achieve the same aim. Method 1 raises

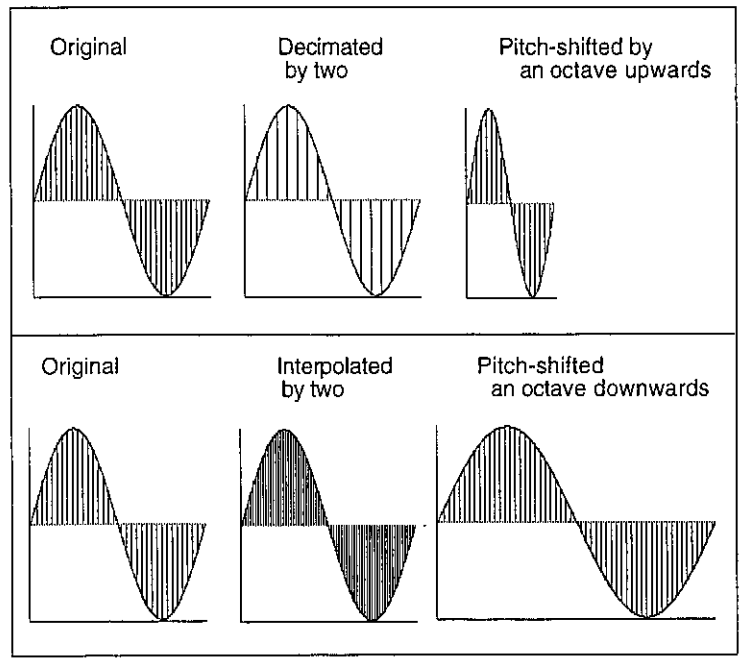


Figure 4.8 Pitch-shifting by sample-rate conversion with a constant playback sampling frequency. (Top) If every other sample is skipped on playback, the signal is decimated and the pitch is shifted up an octave. (Bottom) If twice the number of samples are used by means of interpolation on playback, the signal is shifted down an octave.

pitch by increasing the sampling rate on playback. Method 2, however, raises pitch by decreasing the resampling rate with decimation (downsampling), even though the playback sampling frequency is constant.

So far we have seen how to shift pitch by octave intervals. To shift pitch by any integer ratio, a combination of interpolation and decimation are used (Schafer and Rabiner 1973a; Moorer 1977; Rabiner 1983; Lagadec 1983; Crochiere and Rabiner 1983; Hutchins 1986a; Duncan and Rossum 1988). In particular, to pitch shift by a ratio N/M, we first interpolate by M and then decimate by N. For example, to shift a sound down by an interval of 3/4 (a perfect fourth) we upsample and interpolate by a factor of 4 and then downsample and decimate by a factor of 3. To shift up by a factor of 4/3 we first interpolate by 3 and then decimate by 4.

Sample-rate Conversion without Pitch-shifting

Many digital audio recorders operate at the standard sampling rates of 48 or 44.1 KHz. How can we resample a recording at one of these frequencies

so as to play it back at the other frequency with no pitch shift? In this case the resampling rate is the same as the new output DAC sampling rate.

To convert a signal between the standard sampling rates of 44.1 and 48 KHz without a pitch change, a rather elaborate conversion process is required. First the rates are factored:

$$\frac{48000}{44100} = \frac{2^5 \times 5}{3 \times 7^2} = (4/3 \times 4/7 \times 10/7).$$

These ratios can be implemented as six stages of interpolations and decimations by factors of 2, 3, 5, and 7.

1. Interpolate by 4 from 44,100 to 176,400 Hz
2. Decimate by 3 from 176,400 to 58,800 Hz
3. Interpolate by 4 from 58,800 to 235,200 Hz
4. Decimate by 7 from 235,200 to 33,600 Hz
5. Interpolate by 10 from 33,600 to 336,000 Hz
6. Decimate by 7 from 336,000 to 48,000 Hz

The signal can then be played back at a sampling rate of 48 KHz with no change of pitch.

As long as the input and output sampling rates can be written as a simple fraction, then the conversion process is straightforward. If the rates do not have an integer ratio or they are constantly changing, then more sophisticated mathematical techniques must be used, which we will not venture into here (see Crochiere and Rabiner 1983; Rabiner 1984; Lagadec 1984). This is the case with *flanging effects* (see chapter 10) and audio *scrubbing* (simulating the manual back-and-forth motion of a magnetic tape rocking across a playback head to locate a splice point).

Problems in Resampling

The audio fidelity of resampling is limited by the precision of the hardware used in the conversion. When there are many intermediate resampling stages, a slight loss in fidelity in the form of added noise is to be expected. Aliasing (see chapter 1) can also be a problem. This is because resampling, like the original sampling process, can generate unwanted spectral artifacts due to aliasing. When a sampler skips samples in decimation, for example, it is throwing away intermediate samples. These intermediate samples may have smoothed the waveform's transition between two disjoint points. Thus

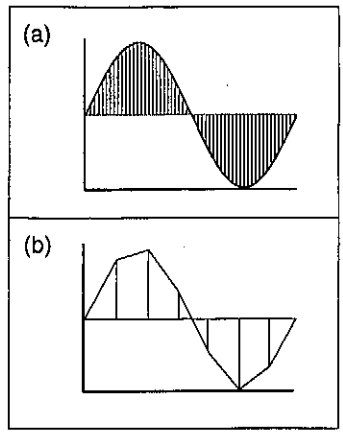


Figure 4.9 With enough decimation, even a sine wave can be turned into a jagged waveform. (*a*) Original sinusoidal waveform. (*b*) Decimation of (*a*) by a factor of eight.

a decimated signal is often full of jagged discontinuities (figure 4.9). At the same time, all frequencies are shifted up, meaning that aliasing can occur on playback. This problem can be reduced to a minimal effect by lowpass filtering the signal after decimation. Filtering smooths the jagged edges of the decimated waveform.

Filtering is also needed in interpolation because simple linear interpolation creates aliased components. Rather than devising a more complicated interpolation scheme, the usual approach in sample-rate conversion is to combine linear interpolation with filtering to shift the frequency content and also minimize aliasing.

Data Reduction and Data Compression in Samplers

The price of semiconductor memory has declined dramatically since it was introduced in the early 1970s. It is still not practical, however, to store a large library of sounds in semiconductor memory. To fit even a subset of such a library into their limited memories, many samplers use *data reduction* or *data compression* strategies to reduce the storage burden. The two are quite different. Data reduction throws away what it considers to be "non-essential" data, while data compression merely makes use of redundancies in the data to code it in a more memory-efficient form. Data compression can reconstitute the original data, while data reduction involves a loss of the original data. Both methods are sometimes grouped under the rubric of *coding* or *encoding* schemes in the audio literature.

Data Reduction

Most samplers do not have facilities for sound analysis and "intelligent" data reduction. In order to reduce the amount of memory needed for storage of audio samples, manufacturers have sometimes taken crude measures that directly affect audio quality. For example, an obvious way to reduce the data stored in a sampler is to limit the sample resolution or quantization (see chapter 1). Some inexpensive sample players use 12 bits or fewer to represent a sample. A variation on this is a *floating-point* encoding scheme that stores the samples in a low-resolution form along with several bits that indicate the original amplitude of the sound (Pohlmann 1989a). Despite shifts in apparent dynamic range, however, the signal-to-noise ratio of the low-resolution samples remains low. Another method is lowering the sampling rate. This diminishes the number of samples stored per unit of time, at the cost of shrinking the audio bandwidth. A third way is to store only every third or fourth note in the range of an instrument and then pitch-shift those samples to play in between pitches. This has the side effect of shifting the spectrum, which is not ideal. If the sound contains any variation like tremolo or vibrato, the rate of these variations is also affected noticeably by pitch-shifting. As the cost of memory declines, there is less and less justification for methods that uniformly compromise audio quality.

A more sophisticated approach to data reduction starts from an analysis stage, which stores sounds in a data-reduced form along with *control functions* that approximately reconstitute it. There are many possible approaches to this analysis and resynthesis. For example, the analysis may take into account *masking phenomena* and throw away those parts of a sound that are supposedly masked by louder parts. (For an introduction to masking, see chapter 23; for further details see Buser and Imbert 1991.) Later in this chapter we look at four experimental data reduction methods based on an additive synthesis model. Several commercial data reduction schemes are built into consumer audio products. This is not the place to enter into a broader discussion of the completeness of the perceptual models on which data reduction schemes are based. Suffice it to say that in any data reduction scheme there is a loss of data leading to a reduction in audio quality. These losses are especially apparent in musical material that exploits the full range of a fine audio system.

Data Compression

To conserve memory space, some systems use data compression techniques to limit the amount of space taken up by a stream of samples. This is done

through the elimination of data redundancies and should not involve any sacrifice in audio quality. One common compression method is *run-length encoding*. The basic idea of run-length encoding is that every sample value is not stored. Rather, each sample that is different from the previous sample is stored, together with a value that indicates how many subsequent samples repeat that value. (For more on audio data compression see Moorer 1979b.)

Sample Libraries

Since a sampler is a type of recording system, the quality of the samples depends on the quality of the recording techniques. Making high-quality samples requires good players with fine instruments, excellent microphones, and favorable recording environments. Arranging all these elements for a large library of sounds takes a great deal of effort. Thus most users of samplers prefer to supplement their collection of samples with libraries prepared by professionals and distributed on magnetic or optical disks.

An Assessment of Samplers

Despite advances in sampling technology, samplers retain a "mechanistic" sound quality that makes them distinguishable from the animated sounds produced by good human performers. Most percussionists, for example, would not mistake the frozen sound of a sampled drum solo from that of a human drummer. In a live performance on acoustic drums, each drum stroke is unique, and there are major differences in the sound depending on the musical context in which the stroke is played. This is not to say that robotic performance is invalid. The commercial success of drum machines proves that lock-step rhythms and unvarying percussion sounds have a major audience.

In any case, it is understandable that the "naturalness" or "realism" of a sampler should be held up as a criterion for judging between different brands. It is well known that a given instrument tone may sound much more realistic on one sampler than it does on another.

Certain instruments, like organs, can be modeled more or less realistically by most samplers. That is, they all can generate a high-quality recording of a pipe or electronic organ. Other instruments like voices, violins, saxophones, electric guitars, and sitars are intrinsically more difficult to capture with existing sampling technology. Individual notes can be captured reasonably well, but when we put the notes together into phrases, melodies, and chords, it is apparent that major chunks of acoustic and performance information have been left out.

Factory-supplied samples model the generic singer, the generic saxophone played by the generic saxophonist, the generic orchestra played in the generic concert hall, and so on. Yet most knowledgeable listeners can tell the difference between two vocalists, saxophonists, and conductors with orchestras. It would be difficult to mistake a MIDI sequencer/sampler rendition of a saxophone solo for the signature style of the John Coltrane original. This points out a fundamental limitation in existing samplers. Beyond a certain point, it is impossible to increase the realism of present samplers without major advances in technology and in understanding of the relationship between sound structure and musical performance. One obvious evolutionary path for samplers is analysis/resynthesis (see chapter 13), which permits flexible, context-sensitive transformations of musical sounds.

In expressive instruments like voices, saxophones, sitars, guitars, and others, each note is created in a musical context. Within a phrase, a note is reached from another note (or from silence) and leads into the successive note (or leads to silence). In addition to these contextual cues, transitional sounds like breathing, tonguing, key clicks, and sliding fingers along strings punctuate the phrasing. Constraints of style and taste determine when context-sensitive effects such as rubato, portamento, vibrato, crescendi and diminuendi, and other nuances are applied.

These problems can be broken into two parts: (1) How can we model the sound microstructure of note-to-note transitions? (2) How can we interpret (analyze) scores to render a context-sensitive performance according to style-specific rules? These questions are the subject of the next two brief sections.

Modeling Note-to-note Transitions

The problem of what happens in note-to-note transitions was the subject of the doctoral research of John Strawn at Stanford University (1985b). He analyzed the transitions in nine nonpercussive orchestral instruments. The time- and frequency-domain plots that emerged from this research graphically depicted the context-sensitive nature of tone successions.

In wind instruments, one of the ways to articulate a transition is by *tonguing*—a momentary interruption of the windstream by an action of the tongue, as if the player were pronouncing the letter *t* or *k*. Figure 4.10 shows a time-domain plot of transitions of a trumpet played tongued (a) and untongued (b). The contrast between the two types of transitions is clear.

Figure 4.11 plots the spectrum of this transition. Strawn's research demonstrated that some transitions are very smooth, with a dip of as little as

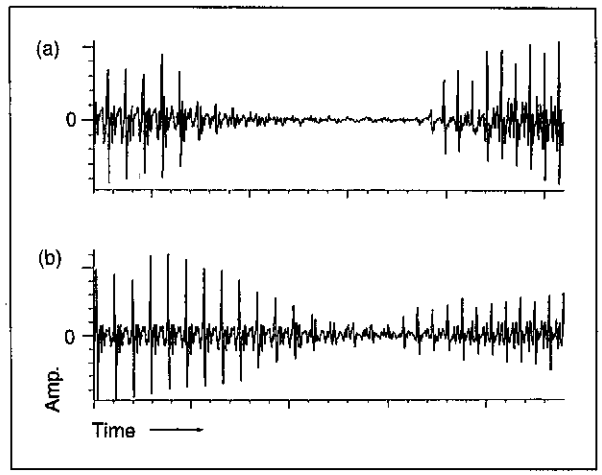


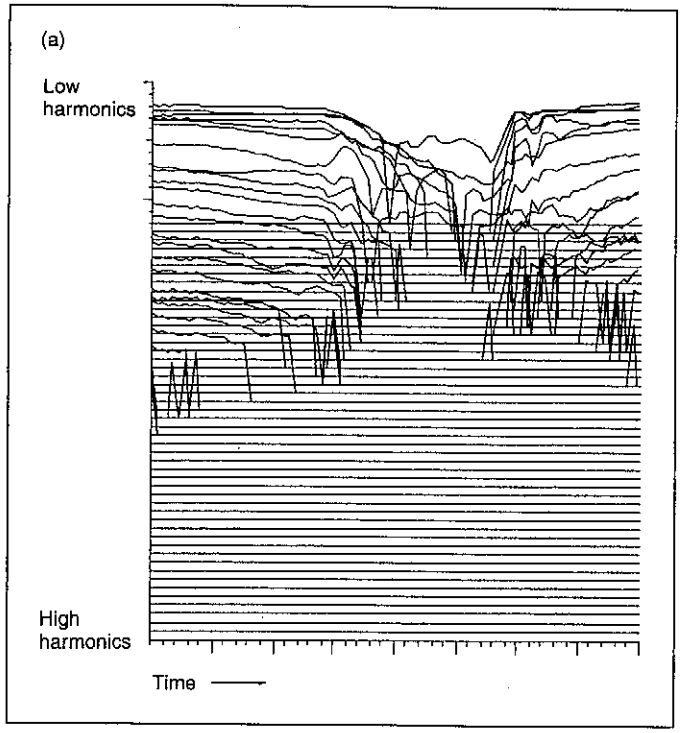
Figure 4.10 Time-domain plot of note-to-note transition of an ascending major third interval for a trumpet played tongued (*a*) and untongued (*b*). The time span for the plots is about 120 ms. (Courtesy of John Strawn.)

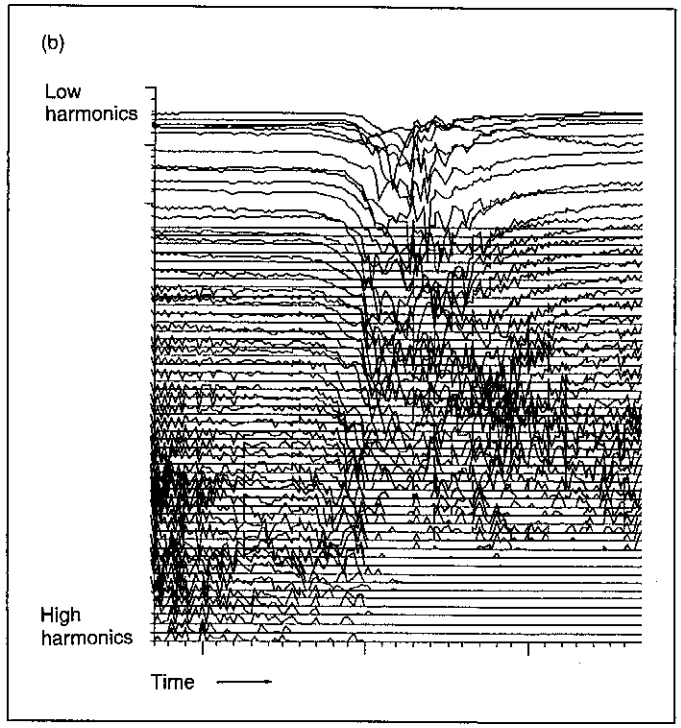
10 dB of amplitude between notes. Other transitions are laden with strong transitional cues in amplitude and spectrum changes that articulate the attack of the second note.

Modeling the microstructure of note-to-note transitions appears to be a tractable problem, since its solution depends on an expectable advance in technology. The problem could be solved by an increase in sampler memory capacities (storing all two-note transitions), fast signal processing, or some combination of the two. The diphone method, for example, stores transition data in a form that allows them to be stretched or compressed (Rodet, Depalle, and Poirot 1988). Holloway and Haken (1992) model transitions as overlapping tracks in a tracking phase vocoder (see chapter 13).

If transitions are to be calculated automatically—as a musician plays on a keyboard, for example, the instrument must be able to make a very quick determination of context. (Chapter 15 discusses the related issue of machine interpretation of musical scores.)

Figure 4.11 Spectrum plots of the transitions shown in figure 4.10. The plots show 50 harmonics plotted over a time span of 300 ms, with lower harmonics at the back. (*a*) Tongued. (*b*) Untongued. Notice how the "hole" in the middle of (*a*) is filled in when the note transition is untongued (more continuous). (Courtesy of John Strawn.)


(a)
Low harmonics
High harmonics
Time


(b)
Low harmonics
High harmonics
Time