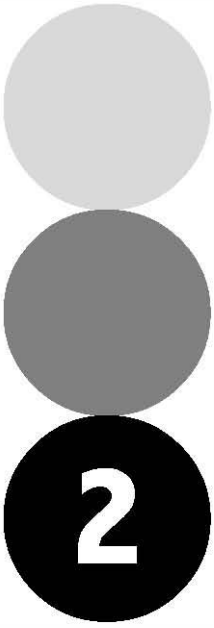

2 Sampling (Wavetable) Synthesis

2.0 Introduction

Chapter 1 covered background on sampling, quantization, compression, and storage of digital PCM sound waveforms. This chapter will introduce some basic aspects of reproducing and modifying PCM sounds. The majority of digital sound and music synthesis today is accomplished via the playback of stored PCM (*Pulse Code Modulation*) waveforms. Single-shot playback of entire segments of stored sounds is common for sound effects, narrations, prompts, musical segments, etc. Most high-quality modern electronic music synthesizers, speech synthesis systems, and PC software systems for sound synthesis use prestored PCM as the basic data. This data is manipulated to yield the final output sound(s). Here we'll look at some of the common manipulations and techniques. We'll also discover some of the problems with using PCM exclusively for interactive expressive audio systems.

2.1 Wavetable Synthesis

For musical sounds, it is common to store just a loop, or table, of the periodic component of a recorded sound waveform and play that loop back repeatedly.

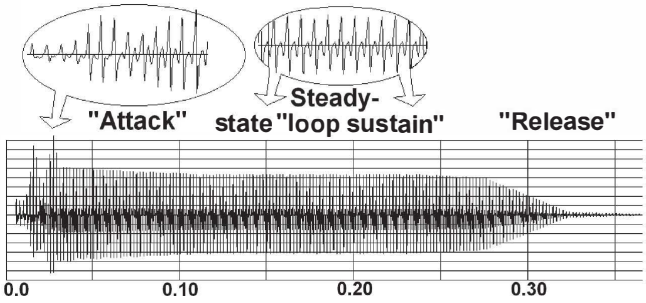

Figure 2.1. Wavetable synthesis of trumpet tone.

This is called *wavetable* synthesis. For more realism, the attack or beginning portion of the recorded sound can be stored in addition to the periodic steady state part. Figure 2.1 shows the synthesis of a trumpet tone starting with an attack segment, followed by repetition of a periodic loop, ending with an enveloped decay (or release). "Envelope" is a synthesizer/computer music term for a time-varying gain change applied to a waveform or other parameter. Envelopes are often described by four components: the *Attack Time;* the *Decay Time* ("decay" here means the initial decay down to the steady state segment); the *Sustain Level;* and the *Release Time* (final decay). Hence, envelopes are sometimes called ADSRs.

Originally called *sampling synthesis* in the music industry, any synthesis using stored PCM waveforms has now become commonly known as *wavetable synthesis*. Filters are usually added to high-quality wavetable synthesis, allowing control of spectral brightness as a function of intensity, and to achieve more variety of sounds from a given set of samples.

2.1.2 Multisampling

Pitch shifting of a PCM sample or wavetable is accomplished via interpolation as discussed in Chapter 1. A given sample can be pitch shifted only so far in either direction before it begins to sound unnatural. This can be dealt with by storing multiple recordings of the sound at different pitches, and switching or interpolating between these upon resynthesis. This is called *multisampling*. Multisampling also might include the storage of separate samples for "loud" and "soft" sounds. Linear interpolation is usually performed between these loudness multisamples as a function of the desired synthesized volume. This is necessary because loudness is not simply a matter of amplitude or power, and most sound sources exhibit spectral variations as a function of loudness. For example, there is usually more high frequency energy, or "brightness" in

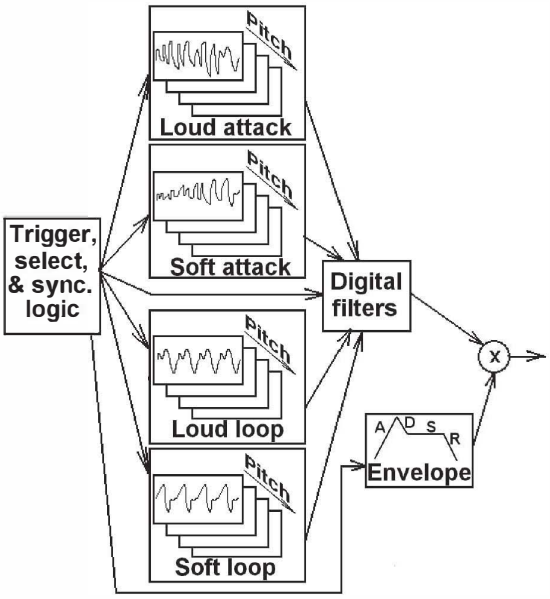

Figure 2.2. Sampling synthesis block diagram.

loud sounds than in soft sounds. Filters can also be used to add spectral control and variation. Figure 2.2 shows a block diagram of a complete sampling synthesizer.

2.2 Concatenative Speech Synthesis

For speech, the most common technique used is a form of wavetable synthesis called *concatenative synthesis*. Concatenative phoneme synthesis relies on the retrieval and concatenation of roughly 40 phonemes (for English). Examples of vowel phonemes are /i/ as in beet, /I/ as in bit, /a/ as in father, etc. Examples of nasals are /m/ as in mom, /n/ as in none, /ng/ as in sing, etc. Examples of fricative consonant phonemes are /s/ as in sit, /ʃ/ as in ship, /f/ as in fifty, etc. Examples of voiced fricative consonants are /z/, /v/, etc. Examples of plosive consonants are /t/ as in tat, /p/ as in pop, /k/ as in kick , etc. Examples of voiced plosives include d, b, g, etc. Vowels and nasals are essentially periodic pitched sounds, so a minimal required stored waveform is only one single period of each. Consonants require more storage because of their noisy nature. Figure 2.3 shows synthesis of the word "synthesize" by the concatenation of the phonemes /s/, /I/, /n/, /th/, /U/, /s/, /a/, /i/, /z/.

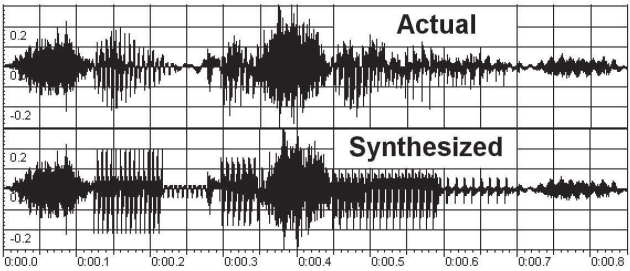

Figure 2.3. Concatenative speech synthesis of the word "synthesize" (lower). The top waveform is an actual recording of the spoken "synthesize."

The quality of concatenative phoneme synthesis is generally considered quite low, due to the simplistic assumption that all of the pitched sounds (vowels, etc.) are purely periodic. Also, simply "gluing" /s/, /I/, and /n/ together does not make for a high quality realistic synthesis of the word "sin." In actual speech, phonemes gradually blend into each other as the jaw, tongue, and other "articulators" move gradually with time (see the upper waveform of Figure 2.3).

Accurately capturing the transitions between phonemes with PCM requires recording transitions from phoneme to phoneme, called *diphones*. A *concatenative diphone synthesizer* blends together stored diphones. Examples of diphones include see, she, thee, and most of the roughly 40×40 possible combinations of phonemes. Much more storage is necessary for a diphone synthesizer, but the resulting increase in quality is significant. As with PCM music synthesis, PCM speech synthesis can be improved further by storing multisamples for different pitch ranges, genders, voice qualities, etc.

2.3 Manipulating PCM

The need and desire to manipulate PCM arises for artistic reasons (music composition, digital DJ and turntable art, sound design for movies and theater, etc.). In games and *Virtual Reality* (VR), the PCM that is available usually does not exactly match the desired application and circumstance. For example, a sound of a squeaking door stored on disk is nearly guaranteed to not be of the correct length, loudness, or character for a given event of a door opening. Even if a sound basically "works" once for the door opening, playing the same identical sound again when the door closes will cue the listener that it's the same identical sound. An even more obvious example might be the sound of a person walking up and down hills of snow, where the sound of each footstep would normally be different from all others. Timing, spectral

characteristics, etc., should all be flexible in order to produce convincing, realistic, and artistic interactive sounds.

2.3.1 Pitch Shifting by Sample Rate Conversion

One way to manipulate sound is to modify the pitch by dynamic sample-rate conversion, as discussed in Chapter 1. But pitch shifting has a number of problems, one being that for a generic PCM sample, resampling not only changes the pitch, but also changes time by the same factor. This is just like turning the speed up or down on a mechanical playback device such as a turntable or tape machine. Shifting up by a factor of 20% yields a sound that plays for 20% shorter time. Shifting down by doubling the time results in a sound that has a pitch one octave lower. By the way, an "octave" is a ratio of 2.0 or 0.5 in frequency, such as the space between any two adjacent "C" notes on the piano keyboard. In fact, it's called an "oct"ave because there are a total of eight white keys, or "diatonic notes" contained in an octave.

One problem of shifting the sampling rate of a PCM sound is the intrinsic link between pitch and time. Pitch shifting upward by one octave yields a sound which plays at twice as high a pitch, and for half the time. Another problem is the so-called *munchkinification effect*. This relates to the apparent shift in the perceived sound-producing object when the pitch and time is shifted. For example, the octave-up pitch shift also yields the perception of a talker whose head sounds half the size of a normal person. This other dimension, which has to do with the quality of sounds rather than pitch, loudness, or time, is called *timbre*. Once we've learned more about the frequency domain, we'll have better tools to explainmunchkinification and timbre, and we'll also have a lot more tools to avoid (or control independently) all of these dimensions.

((9))

2.3.2 Time Shifting by Overlap-Add (OLA)

To devise a way of shifting the playback time of PCM without shifting pitch, we must first resort to a little *PsychoAcoustics*. PsychoAcoustics is the study of how sound stimuli, upon reaching our ears (and possibly our other senses), becomes information that is sensible, musical, threatening, emotionally moving, etc., to our brains. An interesting aspect of perception in audio, visual, and tactile modes is what I like to call "the 30 Hz transition." This refers to the fact that events that repeat, or roughly repeat, at rates much lower than 30 Hz (cycles per second) are perceived as time events. Images flashing at a low rate; a medium speed single-stroke drum roll, or a low frequency periodic pressure on our skin, are all perceived as a series of events occurring in time if they are in the range of 1–20 Hz or so. Events that happen much faster than 30 Hz are perceived quite differently. Video and movie images are flashed every 1/60 of a second, yielding a perception of smooth and flicker-free motion.

As a vibration on the skin increases in frequency toward 100 Hz, it feels more and more like a steady constant pressure. Sound, at frequencies of 35, 60, and 100 Hz, is perceived increasingly as a pitch, and certainly not as individual repetitions of some event in time. The key as to how we might manipulate time in PCM samples without changing pitch and timbre lies in the time = 1/30 second transition.

Overlap-Add (OLA) methods exploit the 30 Hz transition phenomenon by segmenting, repeating, overlapping, and adding an existing waveform to yield a new waveform that has the same perceived pitch characteristics, but modified time. Cutting a waveform into 1/20 second segments, moving all segments closer together by a factor of 40%, and adding them together will yield a 40% shorter file, but the same basic pitch characteristics. This is because the repetitive events beneath each 1/20 second "window" still maintain their exact frequencies, yielding the same sense of pitch. Figure 2.4 shows this process. Notice that the segments in Figure 2.4 are enveloped with triangular gain functions (called *windows*). This avoids the inevitable clicks and pops which would result if we just clipped out the audio using a so-called rectangular window. We will discuss more on windows in later chapters.

10

Given this wonderfully simplistic (while still PsychoAcoustically informed) view of time shifting, we might be content to assume that we've generally solved the problem of pitch-preserving time shifting. That is at least until we begin to listen to the results. For one, we cannot simply cut up a waveform and glue it back together without suffering some artifacts. This would be similar to cutting a photo of a face into equal tiles, sliding all the tiles closer together, and blending the overlaps. We'd probably find the fuzziness and loss of important detail renders some faces unrecognizable. In the audio domain, we should expect that the overlap and add of waveforms might yield audible cancellations or reinforcements (called *destructive* and

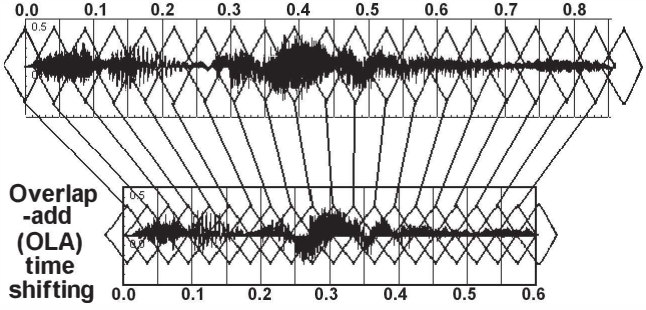

Figure 2.4. Overlap-add time shifting of a PCM waveform.

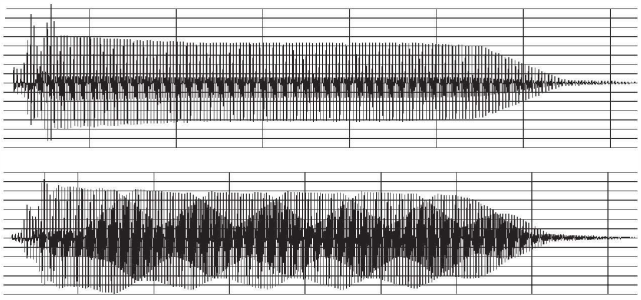

Figure 2.5. Top: Original trumpet tone. Bottom: Overlap-add timeshift shows clear modulation due to constructive and destructive interference.

constructive interference). Figure 2.5 shows examples of destructive and constructive interference due to windowed overlap-add.

2.3.3 Fixing Overlap-Add Modulation Artifacts

The modulations due to interference from overlap-add can be quite audible, and depending on the rate of artifact occurrence, can manifest as either a roughness or an actual pitch (overlap windows spaced every one hundredth of a second will cause a clear 100 Hz tone to be heard in the processed waveform). This can be fixed somewhat by randomizing the window rate. This usually results in a decrease in the "pitchiness" of the artifacts, but introduces a noisy roughness.

Another solution is to move the windows around dynamically to minimize the amount of interference. This involves deducing the pitch of a region of waveform, then setting the local window positions to be spaced apart by an integer multiple of the pitch period. This is called *Pitch Synchronous Overlap-Add* (PSOLA). For example, if the trumpet steady state is a periodic oscillation at 330 Hz, and the desired window spacing is 60 milliseconds, the window spacing would be modified from 19.8 periods of 330 Hz to 20 periods (60.60606 ms) for some windows, and 19 periods (57.5757 ms) for others, for an average of 60 ms. But how do we determine the pitch? We'll discuss that a little when we talk about autocorrelation and the *Fast Fourier Transform* (FFT) in Chapter 5.

2.3.4 Ouch!! Even More Artifacts from Overlap-Add

Another artifact that comes from overlap-add time scaling arises because certain parts of sounds are more naturally extended or shortened in time, while

others are not. One example is the trumpet attack segment, which is roughly the same length independent of the length of the note played by the trumpet player. More profound examples exist in speech, where elongation of consonants causes the speech to sound slurred and drunk. Shortening of consonants can render the speech incomprehensible (shortening an /s/ sound, for example, can result in a perceived /t/ sound). Speaking slowly, or singing, involves elongating the vowels and making pauses longer, but not elongating the consonants. This argues that a good overlap-add time stretch algorithm should be intelligent enough to "figure out" when it should stretch/contract and when it should not, and do the right thing to make the average time modification come out right.

Note that multisampled wavetable synthesis, phoneme speech synthesis, and diphone speech synthesis all take into account a priori the notion of consonants and attacks, versus vowels and steady states. This allows such synthesis techniques to "do the right thing," at the expense of someone having to explicitly craft the set(s) of samples by hand ahead of time. This is not generally possible, or economically feasible, with arbitrary PCM.

Again, in later chapters we'll learn more about how we might determine vowels and consonants, or other sections of audio that should or should not be time shifted. But by that time, we will have seen that there are so many other ways to "skin the cat" of interactive parametric audio, we might not care so much about just performing blender-like manipulations on PCM audio.

2.4 Let's Pause and Think Here

The rest of this book is going to be about why we don't have to, or want to, record and play back manipulated PCM for interactive games, VR, music, and production. This chapter discussed some of the rich legacy of manipulated PCM audio, since that will be our default "fallback" if we run out of time, imagination, or algorithms to do it in more intelligent ways. We will refer back to the examples, techniques, and artifacts presented in this chapter as we continue to gain insight and tools for doing model-based parametric audio synthesis and manipulation. The motivation for doing model-based sound is, of course, the kinds of expressive controls we'll have in doing synthesis.

Reading:

Robert Bristow-Johnson. "Wavetable Synthesis 101: A Fundamental Perspective." AES 101 Conference Preprint #4400, 1996.

Perry Cook, ed. *Music, Cognition, and Computerized Sound*. Cambridge: MIT Press, 1999.

Code:

srconvrt.c

timeshif.c

Sounds:

[Track 7] Wavetable Synthesized Trumpet Passage.

[Track 8] "Synthesize," Original and Phoneme Synthesis.

[Track 9] Pitch Shifted Speech and Trumpet.

[Track 10] Time Shifted "Synthesize," Time Shifted Trumpet Passages.