MULTIMEDIA AUDIO

Multimedia audio is an emerging and evolving topic that encompasses techniques of both analog and digital signal processing, speech, music, and computer systems architecture. This article will begin by describing the components of a multimedia audio system. This will be followed by an overview of the concepts of digital sampling and quantization, and the filtering issues that arise when dealing with audio data at different sampling rates. Computer architectures for moving audio streams within systems are then described. Popular audio compression algorithms will then be reviewed. Next follows a section on the musical instrument digital interface (MIDI), a section on methods for music synthesis, and a section on speech synthesis and recognition. Finally there is a section on 3-D audio and audio for virtual environments.

THE MULTIMEDIA AUDIO SUBSYSTEM

Figure 1 shows a simplified architecture of a multimedia subsystem. As technologies evolve, some components and functionalities have transitioned back and forth between dedicated entertainment systems, computers, and video gaming systems. Examples of dedicated entertainment systems include televisions, home theater systems, future generation stereos, and so on. Computer systems with multimedia include not only desktop computers, but also web servers, palmtops, personal digital assistants (PDAs), and the like. Dedicated video gaming systems use many multimedia components, and add additional controllers and display options to the mix. Other systems which would fit in the multimedia system category might include interactive kiosks and information systems, video conferencing systems, and immersive virtual reality systems. The components shown in Fig. 1 and described below reflect a merging of audio functions from all of these areas.

Multimedia Audio Subsystem Inputs

Line Input(s). This is a high impedance external analog input. Common connection sources include an external mixer, home stereo preamp output, video tape audio outputs, compact disk players, cassette decks, and so forth. Approximate maximum signal levels for this input are 1 V peak to peak.

Microphone Input(s). This connection is an external analog input from a microphone. Signal levels are low, at fractions of millivolts, so low-noise preamplification is required within the multimedia audio hardware.

PC Speaker. The internal connection is to a traditional IBM PC speaker channel, which is capable of synthesizing only square waves at somewhat arbitrary frequencies. Because of the square-wave limitation, this channel is usually used only for "alert" messages consisting of beeps at different frequencies.

CD Audio. This is usually an internal line-level analog connection to CD-ROM audio.

MIDI Synthesizer. This is an internal line-level connection when housed on same board as the PC audio subsystem.

Digital Input(s). This is an external digital serial connection, usually via the Sony/Phillips Digital Interface Format (SPDIF), connected via RCA coaxial connectors and wire, or optical fiber. This is generally considered a "high-end" feature, which can be added to many com-

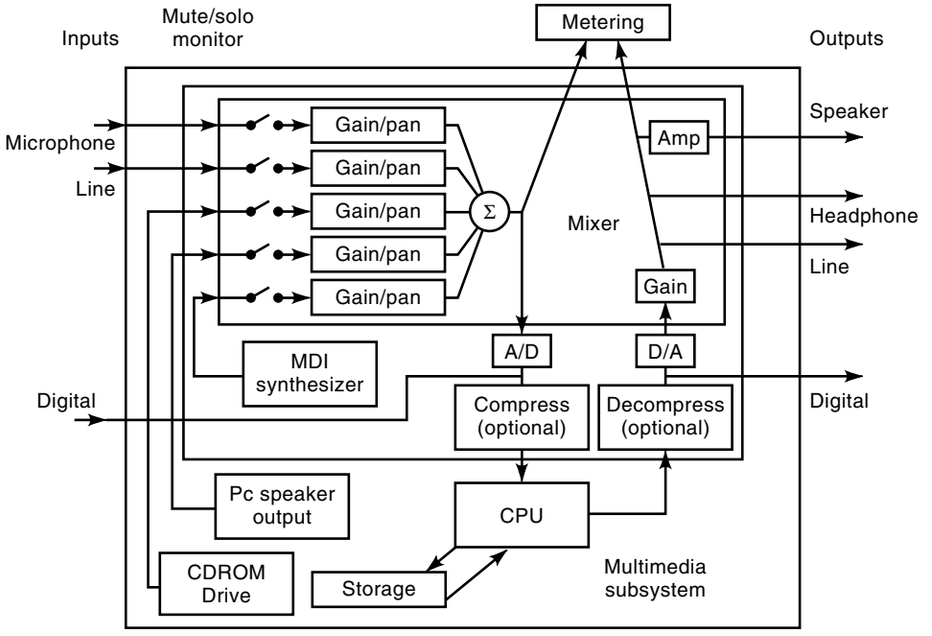

Figure 1. Multimedia subsystem showing inputs, outputs, and connections within the system.

puter systems by the addition of a plug-in board, and is not included in most multimedia systems.

Multimedia Audio Subsystem Outputs

Speaker Output. This is typically a low-impedance 1 W to 4 W external analog output capable of driving loudspeakers directly.

Line Output. This high-impedance analog output connects to an external amplifier, mixer, stereo preamplifier, or other device. Maximum signal levels approximate 1 V peak to peak.

Headphone Output. This output drives stereo headphones. Sometimes the connector for this output automatically mutes the main speaker outputs when headphones are plugged in.

Digital Output(s). See the Digital Input(s) subsection above.

Mixer/Router—Between the Inputs and Outputs

Gain / Volume. This includes input gain controls for all inputs, and master output gain controls. It usually has a log scale, because this most closely matches the human perception of loudness.

Pan. This control allocates the signal from left to right, and is also called balance control. The simplest pan control is linear panning, which applies gain of α to the signal routed to one channel, and a gain of $(1.0 - \alpha)$ to the other. So a panning α value of 0.0 would route all signal to the left speaker, 0.5 would route half-amplitude signals to both speakers, and 1.0 would route all signal to the right speaker. This panning scheme is problematic, however, because perceived loudness is related to power at the listener, and a center-panned signal can appear to be decreased in volume. To provide a more natural panning percept for humans, curves are designed so that a center-panned signal from each channel is 3 dB attenuated in each speaker (rather than 6 dB, as would be the case with linear panning).

Mute / Monitor / Solo. These are switch-like interfaces (usually buttons on a graphical user interface), which determine which signals get into or out of the system. Mute causes a signal to not be passed to the output; monitor causes a signal to be passed to the output. Solo causes only the selected signal channel(s) to be heard, and performs a mute operation on all other signal channels.

Metering. Metering is graphical indication of volume levels of inputs and/or outputs. Usually displayed as the log of the smoothed instantaneous power, metering is often designed to mimic the historic VU (volume unit) meters of radio and recording consoles.

Other Audio-Related System Inputs

Joystick. The joystick is technically not part of the audio subsystem of a multimedia computer, but the connector is often used to provide connections to MIDI (see below).

MIDI (Musical Instrument Digital Interface). On PC architecture machines, MIDI is usually connected via a special cable that connects to the joystick port, but dedicated plug-in MIDI cards are also available. Other machines connect MIDI directly to the serial ports via an electrical adapter box, and still others require special hardware plug-in boards. MIDI input and output are almost always supported on the same joystick port, plug-in board, or serial adapter. See the section titled Musical Instrument Digital Interface for more information on the MIDI hardware and software protocols.

Other Audio-Related System Outputs

Joystick MIDI. See the Joystick Input subsection above.

MIDI. See the MIDI Input subsection above.

Force Joystick and Other Haptic Devices. Haptics (1) is the combined touch senses of taction (sense of vibration and pressure on the skin) and kinesthesia (the sense of positions of limbs, body, and muscles). Haptic display devices are just beginning to appear as computer system peripherals. The simplest include simple vibrators attached to joysticks, embedded in gloves, in chair seats, and low-frequency audio speakers worn on the back. These are intended to provide the sensation of contact, vibration and roughness, and explosions for games. More sophisticated haptic devices do not use audio frequencies per se, but allow for precise forces to be exerted upon the user in response to algorithmic computations taking place in the computer. Connectors range from standard serial ports to specialized connectors from plug-in boards.

Video Helmet / Glasses and Other Devices. These often include integrated headphone connections, with the headphones fed from the audio outputs of the system.

PCM WAVE STORAGE AND PLAYBACK

Representing Audio as Numbers

In a modern multimedia system, audio is captured, stored, transmitted, analyzed, transformed, synthesized, and/or played back in digital form. This means that the signal is sampled at regular intervals in time, and quantized to discrete values. The difference between quantization steps is often called the quantum. The process of sampling a waveform, holding the value, and quantizing the value to the nearest number that can be represented in the system is referred to as analog to digital (A/D) conversion. Coding and representing waveforms in this manner is called pulse code modulation (PCM). The device that does the conversion is called an analog-to-digital converter (ADC, or A/D). The corresponding process of converting the sampled signal back into an analog signal is called digital to analog conversion, and the device that performs this is called a DAC. Filtering is also necessary to reconstruct the sampled signal back into a smooth continuous time analog signal, and this filtering is usually contained in the DAC hardware. Figure 2 shows the process of analog-to-digital conversion on a time/amplitude grid.

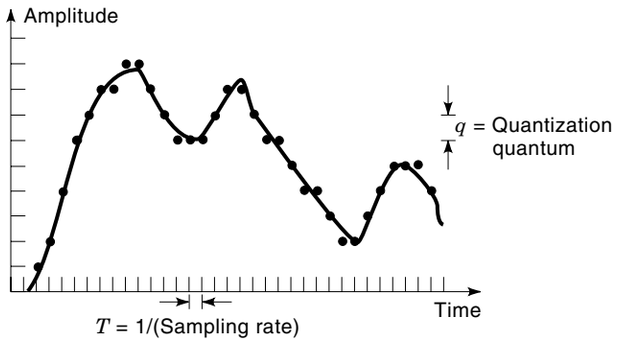

Figure 2. Linear sampling and quantization. At each timestep, the waveform is sampled and quantized to the nearest quantum value.

Sampling and Aliasing

An important fundamental law of digital signal processing states that if an analog signal is bandlimited with bandwidth B, the signal can be periodically sampled at sample rate $2B$, and exactly reconstructed from the samples (2). Intuitively, a sine wave at the highest frequency B present in a bandlimited signal can be represented using two samples per period (one sample at each of the positive and negative peaks), corresponding to a sampling frequency of $2B$. All signal components at lower frequencies can be uniquely represented and distinguished from each other using this same sampling frequency of $2B$. If there are components present in a signal at frequencies greater than one-half the sampling rate, these components will not be represented properly, and will alias as frequencies different from their true original values.

Most PC-based multimedia audio systems provide a large variety of sampling rates. Lower sampling rates save processor time, transmission bandwidth, and storage, but sacrifice audio quality because of limited bandwidth. Humans can perceive frequencies from roughly 20 Hz to 20 kHz, thus requiring a minimum sampling rate of at least 40 kHz. Speech signals are often sampled at 8 kHz or 11.025 kHz, while music is usually sampled at 22.05 kHz, 44.1 kHz (the sampling rate used on audio compact disks), or 48 kHz. Other popular sampling rates include 16 kHz (often used in speech recognition systems) and 32 kHz. The maximum sampling rate available in multimedia systems is 50 kHz. To avoid aliasing, ADC hardware usually includes filters that automatically limit the bandwidth of the incoming signal as a function of the selected sampling rate.

Quantization

To store a value in a computer, it must be represented as a finite precision quantity. On a digital computer, that means that some number of binary digits (bits) are used to represent a number that approximates the value. This quantization is accomplished either by rounding to the quantity nearest to the actual value, or by truncation to the nearest quantity less than or equal to the actual value. With uniform sampling in time, a bandlimited signal can be exactly recovered, provided that the sampling rate is twice the bandwidth or greater; but when the signal values are rounded or truncated, the difference between the original signal and the quantized signal is lost forever. This can be viewed as a noise component upon reconstruction. A common analytical technique assumes that this discarded signal component is uniformly distributed randomly at $\pm 1/2$ the quantum for quantization by rounding, or from 0 to the quantum for quantization by truncation. Using this assumption gives a signal to quantization noise approximation of $6N$ dB, where N is the number of bits (2). This means that a system using 16 bit linear quantization will exhibit a signal to quantization noise ratio of approximately 96 dB.

Most PC-based multimedia audio systems provide two or three basic sizes of audio words. Sixteen-bit data is quite common, as this is the data format used in Compact Disk systems. Eight-bit data is equally common, and is usually used to store speech data. Data size will be discussed in more detail in the section titled Audio Compression.

ANALOG MIXING VERSUS DIGITAL FILTERS

As mentioned above, there are a number of sampling rates common in a multimedia audio system. Often these sampling rates must be simultaneously supported, as might be the case with speech narration and sound effects taking place along with music synthesis. To mix these signals in the digital domain requires sampling rate conversion (3,4), which can be computationally expensive if done correctly. The solution in some systems is to require that all media that will be played at the same time must be at the same sampling rate. Another solution is to do the conversion back to analog using multiple DACs, and then do the mixing in the analog domain. As processor power increases and silicon costs decrease, more systems are beginning to appear which provide software or hardware sampling rate conversion to one common system sampling rate (usually 44.1 kHz), and then mix all signals in the digital domain for playback on a single fixed sampling rate stereo DAC.

SOFTWARE STREAMING MODELS

Buffered (Vectorized) Data Architectures

Nearly all multimedia systems contain some type of central processor. For audio input, the CPU performs acquisition of the samples from the audio input hardware, possibly compresses the samples, and then stores the resultant data, or transmits the data to another processing stage. For audio output the CPU would collect samples from another process or storage, and/or might possibly perform direct synthesis of sound in software. If the samples were compressed the CPU would perform decompression, and then transmit the sound samples to the audio output hardware. A karaoke application might perform many of these tasks at once, taking in a vocal signal via the ADCs, performing processing on the vocal signal, performing software synthesis of music, mixing the synthesized music with the processed voice, and finally putting the resulting signal out to the DACs.

The CPU in many systems must not only perform audio functions, but also other computing tasks as well. For example, in running a game application on a desktop PC system, the CPU would respond to joystick input to control aspects of the game, perform graphics calculations, keep track of game state and score, as well as performing audio synthesis and processing. If the CPU were forced to collect incoming samples and/or calculate output samples at the audio sample rate, the overhead of swapping tasks at each audio sample would consume much of the processor power. Modern pipelined CPU architectures perform best in loops, where the loop branching conditions are stable for long periods of time, or at least well predicted. For this reason almost all audio subsystems work on a block basis, where a large number of samples are collected by the audio hardware before the CPU is asked to do anything about them. Similarly for output, the CPU might synthesize, decompress, process, and mix a large buffer of samples and then pass that buffer to the audio hardware for sample-rate synchronous transmission to the DACs, allowing the CPU to turn to other tasks for a time. A typical system might process buffers of 1024 samples, collected into system memory from the audio ADCs by DMA (direct memory access) hardware. This buffer size would correspond to a processor service rate of only 44 times/s for a sampling rate of 44.1 kHz.

Blocking Push/Pull Versus Callback

There are two common architectures for handling buffered audio input and output—push/pull with blocking, or callback. In a push/pull with blocking architecture, the main audio process writes output (and/or reads input) buffers to the audio output software drivers. The audio driver software keeps a number of buffers (minimum two) for local workspace, and when those buffers are full, it "blocks" the calling write/read function until one or more of the available work buffers are empty/full. The audio driver software will then "release" the calling write/read function to complete its task. Another way to view this is that the function called to write and/or read doesn't return until more samples are needed or available. Because of the blocking, the code in which the audio driver functions are called is often placed in a different execution "thread," so that other tasks can be performed in parallel to the audio sample read/write operations.

In a callback architecture, the audio drivers are initialized and passed a pointer to a function which is called by the audio subsystem when samples are needed or ready. This function usually just copies a buffer of samples in or out, and sets a flag so that another process knows that it must fill the local buffer with more samples before the next callback comes.

SYNCHRONIZATION

Since a multimedia system is responsible, by definition, for collecting and/or delivering a variety of media data types simultaneously, synchronization between the different types of data is an important issue. Audio often is the master timekeeper in a multimedia system, for two primary reasons. The first reason is that audio samples flow at the highest rate, typically tens of thousands of samples per second, as opposed to 30 frames/s or so for video and graphics, or 100 events/s for joystick or virtual reality (VR) controller inputs. This means that the granularity of time for audio samples is much smaller than those of the other data types in a multimedia system, and a timekeeper based on this granularity will have more resolution than one based on coarser time intervals of the other data types. The other reason that audio is often the master timekeeper is that the human perception system is much more sensitive to lost data in audio than it is to graphics or video data loss. An occasional dropped frame in video is often not noticed, but even one dropped sample in audio can appear as an obvious click. A dropped or repeated buffer of audio is nearly guaranteed to be noticed and considered objectionable.

In some systems where audio serves as the master timekeeper, other data-delivery systems keep informed of what time it is by querying the audio drivers, and throttle their outputs accordingly, sometimes ignoring frames of their own data if the process gets behind the audio real-time clock. In other systems, the audio buffering blocking or callback process can be set up so that the frames of other data are synchronous with the audio buffers. For example, a video game might run at about 22 graphics frames/s, synchronous to audio buffers of size 1024 samples at 22.05 kHz sampling rate.

In some systems such as game systems, the playback of large contiguous and mixed streams of audio is not required, but rather all that is required is the playback of small wave files and the synthesis of MIDI-controlled music (see the section titled "Musical Instrument Digital Interface"). In these systems there is often a master timekeeper that synchronizes all events, and the audio event cues are dispatched to the wave and MIDI playback systems at (roughly) the correct times. Once a cue is received, each small wave file plays back in its entirety, and/or each MIDI event is serviced by the synthesizer. If the lengths of the individual sound files are not too long, reasonable synchronization can be achieved by this type of "wild sync," where only the beginnings of the events are aligned.

AUDIO COMPRESSION

The data required to store or transmit CD quality audio at 44.1 kHz sampling rate, stereo, 16 bits, is 176 kbyte/s. For storage, this means that a three-minute song requires 30 Mbytes of storage. For transmission, this means that an ISDN line at 128 kbit/s is over 10 times too slow to carry uncompressed CD-quality audio. Audio compression, as with any type of data compression, strives to reduce the amount of data required to store and/or transmit the audio. Compression algorithms can be either lossless, as they must be for computer text and critical data files, or lossy, as they usually are for images and audio.

μ-Law Compression

Rather than using a linear quantization scheme, where the amplitude dynamic range is divided into equal pieces, μ-Law (or a-Law) nonlinear quantization (5) provides a simple method of data compression. Figure 3 shows a nonlinear quantization scheme.

Since the quantization noise is related to the size of the quantum, then having a small quantum ensures small quantization noise, and a large signal to quantization noise ratio for small-amplitude signals. For large-amplitude signals, the quantum is large, but the signal to quantization noise ratio is held relatively constant. A common metric applied to μ-Law nonlinear sampling is that N bits performs roughly as well as $N + 4$ linearly quantized bits. Eight-bit μ-Law is common for speech coding and transmission, providing 2 : 1 compression over 16 bit linear audio data with a perceptual quality roughly equal to that of a 12 bit system. μ-Law compression is lossy, since the original samples cannot be reconstructed from the compressed data.

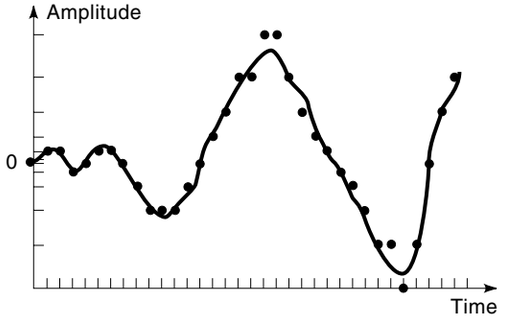

Figure 3. Nonlinear sampling and quantization. At each timestep, the waveform is sampled and quantized to the nearest quantum, but the quanta are small for small signal values, and larger for large signal values. This results in a roughly constant signal-to-noise ratio, independent of signal amplitude.

Adaptive Delta Pulse Code Modulation

Adaptive delta pulse code modulation (ADPCM) (6) endeavors to adaptively change the quantum size to best match the signal. Further, the changes (deltas) in the signal are coded rather than the absolute signal values. Each sample, the change in the signal value versus the last value, is computed. This delta is compared to the current adapted delta value, and the new adapted delta value is increased or decreased accordingly. All that is coded is the sign of the delta for the given step, and a three-bit value reflecting the current quantized adapted value. This allows 4 bits of information to store a 16 bit linearly quantized sample value, providing 4 : 1 compression over 16 bit linear audio data. ADPCM is supported by many multimedia audio systems, but it has not found broad popular use because the quality is not considered good enough for most program material. ADPCM compression is lossy.

MPEG and Other Transform Coders

Transform coders endeavor to use information about human perception to selectively quantize regions of the frequency spectrum, in order to "hide" the quantization noise in places that the human ear will not detect. Such coders strive for what is often called "perceptual losslessness," meaning that human listeners cannot tell the difference between compressed and uncompressed data. A transform coder performs frequency analysis using either the Fourier transform, subband decomposition using a filter bank, wavelet analysis, or a combination of these. The spectral information is inspected to determine the significantly loud features, and masking threshold curves (regions under which a sound could be present and not be detected due to the loudness of a nearby spectral peak) are drawn outward in frequency from these significant peaks. Quantization noise is then "shaped" by allocating bits to subband regions, so that the quantization noise lies under the masking threshold curves. Masking is also computed in time, recognizing that noise can linger significantly in time after a loud event without being heard, but cannot precede the loud event by as much time without being heard. Given some assumptions and suitable program material, the quantization noise will be allocated in such a way that a human listener will not be able to detect it. Transform coding is used in multimedia standard audio formats such as MPEG (Moving Picture Expert Group) (7) and Dolby's AC-2 and AC-3 (8). Such coders can achieve perceptually lossless compression ratios of approximately 4 : 1 or 8 : 1 on some program material, and even higher compression ratios with some small acceptable degradation.

MUSICAL INSTRUMENT DIGITAL INTERFACE

The musical instrument digital interface (MIDI) standard (9), adopted in 1984, revolutionized electronic music and soon

thereafter affected the computer industry. It is a shining example of what can happen when manufacturers of competing products agree on a simple hardware and software protocol standard, and in the case of MIDI it happened without the aid of a preexisting external organization or governmental body.

The Basic Standard

A simple two-wire serial electrical connection standard allows interconnection of musical devices over cable distances of up to 15 m, and longer over networks and extensions to the basic MIDI standard. The software protocol is best described as "musical keyboard gestural," meaning that the messages carried over MIDI are the gestures that a pianist or organist uses to control a traditional keyboard instrument. There is no time information contained in the basic MIDI messages, and they are intended to take effect as soon as they come over the wire. Basic MIDI message types include NoteOn and NoteOff, Sustain Pedal Up and Down, Modulation amount, and PitchBend. NoteOn and Off messages carry a note number corresponding to a particular piano key, and a velocity corresponding to how hard that key is hit. Another MIDI message is Program Change, which is used to select the particular sound being controlled in the synthesizer. MIDI provides for 16 channels, and the channel number is encoded into each message. Instruments and devices can all "listen" on the same network, and choose to respond to the messages sent on particular channels.

Certain messages are used to synchronize multiple MIDI devices or processes. These messages include start, stop, and clock. Clock messages are not absolute time messages, but simply "ticks" measured in fractions of a musical quarter note. A process or device wishing to synchronize with another would start counting ticks when it first received a MIDI start message, and would keep track of where it was by looking at the number of ticks that had been counted. These timing messages and some others do not carry channel numbers, and are intended to address all devices connected to a single MIDI network.

Extensions to MIDI

The most profound basic extension to MIDI has been the advent of General MIDI (9,10), and the Standard MIDI File Specifications (9). By the dawn of multimedia, MIDI had already begun to be a common data type for communication between applications and processes within the computer, with the majority of users not being musicians, and not even owning a keyboard or MIDI interface for their computer. General MIDI helped to assure the performance of MIDI on different synthesizers, by specifying that a particular program (algorithm for producing the sound of an instrument) number must call up a program that approximates the same instrument sound on all General MIDI compliant synthesizers. There are 128 such defined instrument sounds available on MIDI channels 1 to 9 and 11 to 16. For example, MIDI program 0 is grand piano, and MIDI program 57 is trumpet. On General MIDI channel 10, each note is mapped to a different percussion sound. For example, bass drum is note number 35 on channel 10, cowbell is note number 56, and so forth.

The MIDI file formats provide a means for the standardized exchange of musical information. The growth of the World Wide Web has brought an increase in the availability and use of MIDI files for augmenting web pages and presentations. A MIDI level 0 file carries the basic information that would be carried over a MIDI serial connection, including program changes, so that a well-authored file can be manipulated by a simple player program to configure a General MIDI synthesizer and play back a song with repeatable results. A MIDI level 1 file is more suited to manipulation by a notation program or MIDI sequencer (a form of multitrack recorder program that records and manipulates MIDI events). Data are arranged by "tracks," which are the individual instruments in the virtual synthesized orchestra. Metamessages allow for information that is not actually required for a real-time MIDI playback, but might carry information related to score markings, lyrics, composer and copyright information, and so forth.

To help ensure consistency and quality of sounds, the Downloadable Sounds Specification provides means to supply PCM-sampled sounds to a synthesizer (see section on PCM Sampling and Wavetable Synthesis below). Once downloaded, author/application-specific custom sounds can be controlled using standard MIDI commands.

Improvements, Enhancements, and Modifications of MIDI

So far the most successful extensions to MIDI have been those that do not require a change in hardware of software to support existing legacy equipment and files. Compatibility has been at the heart of the success of MIDI so far, and this has proven to be important for improvements and extensions to MIDI.

- MIDI Time Code (MTC) is an extension to the basic MIDI message types, which allows for absolute time information to be transmitted over existing MIDI networks (9).
- MIDI Machine Control (MMC) specifies new messages that control audio recording and video production studio equipment, allowing for synchronization and location functions (9).
- MIDI Show Control (MSC) specifies new messages that are designed specifically to control stage lighting and special-effects equipment in real time (9).
- MIDI GS and XG are system specifications which extend the range of sounds and sound control beyond basic general MIDI. Extended control of effects and additional bands of sounds are compatible with general MIDI, and to some degree between GS and XG. Up-to-date information can be obtained from Roland (GS) and Yamaha (XG).

Less successful extensions to MIDI include XMIDI and ZIPI, sharing the fact that they require new hardware to be useful. XMIDI (11) endeavors to extend the electrical and message standards by transparently introducing additional higher bandwidth transmissions on the existing MIDI wiring, but has not yet found adoption among manufacturers and the MIDI Manufacturers Association. ZIPI (12) is a more generic high-bandwidth network, requiring new hardware and protocols, but also including MIDI as a subsystem. Various serial systems and protocols for modern recording studio automation and control allow for MIDI data in addition to digital audio to flow over optical cabling. As of this writing, manufacturers of professional music and recording equipment have

yet to agree on a standard, and are pursuing their own system designs.

SOUND AND MUSIC SYNTHESIS

PCM Sampling and Wavetable Synthesis

The majority of audio on computer multimedia systems comes from 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, and so forth. 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. This is called wavetable synthesis. For more realism, the attack or beginning portion of the recorded sound is stored in addition to the periodic steady-state part. Figure 4 shows short periodic loops and longer noise segments being used for speech synthesis, as described later. Originally called sampling synthesis in the music industry, all synthesis involving stored PCM waveforms has become known as wavetable synthesis. Filters are usually added to sampling and wavetable synthesis to allow for control of spectral brightness as a function of intensity, and to get more variety of sounds out of a given set of samples.

In order to synthesize music, accurate and flexible control of pitch is necessary. In sampling synthesis, this is accomplished by dynamic sample rate conversion (interpolation). Most common is linear interpolation, where the fractional time samples needed are computed as a linear combination of the two closest samples. Sample interpolation is often misunderstood as a curve-fitting problem, but for best audio quality it should be viewed as a filter design problem (4,13). A given sample can only be pitch shifted so far in either direction before it begins to sound unnatural. This is dealt with by storing multiple recordings of the sound at different pitches, and switching or interpolating between these upon resynthesis. This is called "multisampling."

Additive and Subtractive Synthesis

Synthesis of signals by addition of fundamental waveform components is called additive synthesis. Since any function can be uniquely represented as a linear combination of sinusoidal components, the powerful tool of Fourier analysis (14) gives rise to a completely generic method of sound analysis and resynthesis. When there are only a few sinusoidal components, additive synthesis can be quite efficient and flexible. However, most sounds have a significant number of components which vary quite rapidly in magnitude and frequency. Essentially no multimedia subsystem sound synthesizers use strictly additive synthesis, but this could become a reality as processor power increases, and costs decrease.

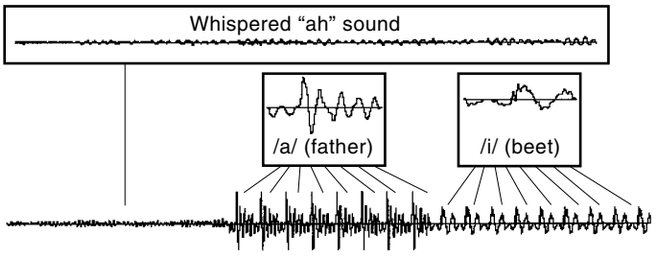

Figure 4. Synthesizing the word "hi" from individual PCM wave files. This is done by using a recording of the whispered "ah" sound, followed by the vowel "ah," then by the vowel "ee." Since vowel sounds are quasiperiodic, only one period of the vowel sounds is stored.

Some sounds yield well to subtractive synthesis, where a complex source signal is filtered to yield a sound close to the desired result. The human voice is well modeled in this way, because the complex waveform produced by the vocal folds is filtered and shaped by the resonances of the vocal tract tube. The section titled Speech will discuss subtractive synthesis in more detail.

FM Synthesis

For the first few years of PC-based audio subsystems, sound synthesis by frequency modulation (FM) was the overriding standard. Frequency modulation relies on the modulation of the frequency of simple periodic waveforms by other simple periodic waveforms. When the frequency of a sine wave of average frequency f_c (called the carrier), is modulated by another sine wave of frequency f_m (called the modulator), sinusoidal sidebands are created at frequencies equal to the carrier frequency plus and minus integer multiples of the modulator frequency. This is expressed as:

$$y(t) = \sin(2\pi t(f_c \cos(2\pi t f_m)]$$

where I is the index of modulation, defined as $\Delta f_c/f_c$. Carson's rule (a rule of thumb) states that the number of significant bands on each side of the carrier frequency is roughly equal to $I + 2$. For example, a carrier sinusoid of frequency 600 Hz, a modulator sinusoid of frequency 100 Hz, and a modulation index of 3 would produce sinusoidal components of frequencies 600, 700, 500, 800, 400, 900, 300, 1000, 200, 1100, 100 Hz. Inspecting these components reveals that a harmonic spectrum (all integer multiples of some fundamental frequency) with 11 significant harmonics, based on a fundamental frequency of 100 Hz can be produced by using only two sinusoidal generating functions. By careful selection of the component frequencies and index of modulation, and by combining multiple carrier/modulator pairs, many spectra can be approximated using FM. The amplitudes and phases of the individual components cannot be independently controlled, however, so FM is not a truly generic waveform synthesis method. The amplitudes of sideband components are controlled by Bessel functions, and since the very definition of FM is a nonlinear function transformation, the system does not extend additively for complex waveforms (or sums of modulator or carrier sine functions). Using multiple carriers and modulators, connection topologies (algorithms) have been designed for the synthesis of complex sounds such as human voices, violins, and brass instruments (15).

Physical Modeling and Generalized Algorithmic Synthesis

Physical modeling synthesis (16) endeavors to model and solve the acoustical physics of sound-producing systems in order to synthesize sound. Classical theories and recent advances in knowledge about acoustics, combined with algorithm advances in digital signal processing and increases in processor power, allow certain classes of sound-producing sys-

tems to be physically modeled in real time. Unlike additive synthesis, which can use one powerful generic model for any sound, physical modeling requires a different model for each separate family of musical instrument or sound-producing object. The types of fundamental operations required to do physical modeling are varied, and include waveform playback, linear filtering, lookup tables, and other such operations. These represent nearly a complete superset of the fundamental operations required to do other types of synthesis. In the history of computer music, there is a recurring notion of a set of simple "unit generators," which are used to build more complex algorithms for synthesis and processing (17). The need for multiple algorithms gives rise to a new type of generalized synthesizer architecture.

New systems for multimedia sound synthesis, particularly those that run in software on a modern host microprocessor, are beginning to surface based on a generalized algorithmic framework. In such systems, the algorithm that best suits the particular sound can be used. Such systems face problems of resource allocation, because different algorithms exhibit different memory and processor load characteristics. Since, however, each synthesis algorithm is best suited to certain types of sounds, generalized algorithmic synthesis can yield the theoretical best quality for all sounds. Further, the flexible dynamic framework of a generalized software synthesizer allows for new algorithms to be added at a later time.

The upcoming MPEG-4 audio layer standard includes a component called structured audio language (SAOL) (18), which provides a framework for manipulating audio in more parametric and flexible ways. There is a component allowing for algorithmic synthesis and processing specifications, thus supporting generalized algorithmic synthesis directly in the audio layer of the multimedia data.

SPEECH

Through science fiction, long before the advent of the personal computer, the public was introduced to the idea that all machines of the future would speak naturally and fluently, and understand all human speech. For this reason, of all the multimedia data types, speech is the one for which the public has the highest expectations, yet has proven to be the most difficult to implement effectively. Since the areas of speech synthesis, recognition, and coding are quite large, only a brief introduction to these topics will be provided here.

Concatenative Phoneme Speech Synthesis

A large amount of artificial digital speech is generated by concatenative waveform synthesis. Since the steady states of speech vowels are quasiperiodic waveforms, only one period of each vowel can be stored, and these can be looped to generate arbitrary length segments of these vowels. Since consonant sounds are not periodic, however, they must be stored in longer wave files. Each vowel and consonant is called a phoneme, and at least 40 phonemes are required to represent the sounds encountered in American English speech. To generate speech, these phoneme segments are concatenated together. Pitch shifting is accomplished by sample rate interpolation of the vowel and voiced consonant sounds. Figure 4 shows how the word "hi" might be synthesized using concatenative phoneme synthesis.

Concatenative Diphone Speech Synthesis

The transitions from one phoneme to another are not abrupt, because it takes time for the speech articulators (tongue, jaw, etc.) to move from one point to another. Concatenative phoneme synthesis does not allow for smooth transitions between phonemes, and mere cross-fading by interpolation from one phoneme to the other does not produce realistic transitions. More natural synthetic speech can be generated by storing "diphones," which are recordings of entire transitions from one phoneme to the other. This takes much more storage, because if there are N phonemes, there will be N^2 diphones, and the diphone transitions involving vowels are longer than simple single periods of a periodic waveform. In any given language, certain diphone transitions will not be required, and others occur so infrequently that it is acceptable to store only phonemes and do the lower-quality synthesis in cases where rarely needed diphones are needed. Triphones can also be stored and synthesized.

Source/Filter Speech Synthesis

When the pitch of stored speech waveforms is shifted by sample rate interpolation, undesired distortions of the waveform and accompanying spectra occur. If the pitch shift range is small, this does not generate troubling perceptual artifacts. However, the pitch range of speech can be quite large, perhaps a factor of two for emotional speech. For large pitch shifts, the distortion manifests itself as making the sound seem like the size of the speaker's head is changing. This might be familiar to the reader from listening to tape recordings that have been sped up or slowed down. The speeding up might make the voice sound like it belongs to a chipmunk or mouse, for example. If the voice of a child or woman is desired, mere pitch shifting is not enough to give the right sound.

For this reason, a more parametric synthesis model is required to allow for truly flexible speech synthesis over a large range of pitches and types of voices. In human vowels, the source waveform of the vocal fold vibration is filtered by the resonances of the vocal tract acoustic tube. For unvoiced consonants (s, f, etc.) the source generated by turbulence noise is filtered by the vocal tract resonances. Voiced consonants (z, v, etc.) contain components of both noise and quasiperiodic oscillation. A speech synthesizer based on a source/filter model can be used to generate more natural-sounding and flexible artificial speech. In such a synthesizer, source generators such as a waveform oscillator and a noise source are filtered by digital filters to shape the spectrum to match speech sounds. Transitions between phonemes are accomplished by driving the filter and source parameters from one point to another. Pitch changes are easily accomplished by simply changing the frequency of the waveform generator.

Speech Coding and Compression

Speech can be compressed and coded (19) using the same techniques that were described in the section titled Audio Compression. Having a parametric synthesis model closely based on the actual human speech mechanism, however, allows higher quality at higher compression ratios. Linear predictive coding (LPC) is a signal-processing technique that allows the automatic extraction of filter and source parame-

ters from a speech waveform (20). LPC is used to code speech with recursive filter of the order of 8 to 20, updated a few times per second, typically 20 to 200. The extracted source waveform can be parametrically modeled using a simple oscillator and noise source, or more realistically modeled using vector codebook quantization methods. High-quality speech transmission can be accomplished using only a few thousand bits per second, representing significant compression improvements over 8 kHz μ-Law coded speech.

Speech Recognition

Speech recognition systems (21) can be divided into four main groups, based on whether they are intended for isolated words or continuous speech, or designed to be speaker independent or trained for a particular speaker. A speaker-dependent, isolated word system is essentially the farthest from our experience as human speech recognizers, but such systems can be quite accurate and useful for command-type applications such as a voice interface to computer menu commands. Some command systems such as voice dialing must function as speaker-independent, isolated word systems, because the huge number of possible voices makes it impossible to train the system on a few individual voices. Speaker-independent, continuous speech systems have proven to be the most difficult to implement, because of many inherent problems. Some of these problems include identifying where words or phonemes start and end (called segmentation), determining the actual speech from the natural disfluencies and repetitions present in natural speech (uhm, er, I mean, well, uh, you know?), and dealing with the differences in individual speakers and voices.

Speech recognizers are usually based on a "front-end," which processes the speech waveform and yields a few parameters per time period which are matched against a library of templates. Typical front-ends include spectrum analysis and linear predictive coding, as described in the "Speech Coding and Compression" subsection. The templates are often phoneme related. Statistical methods involving hidden Markov models (HMMs) are used to pick the most likely candidate words from a set of template matches. The application of HMMs can extend to the sentence level, helping to determine which are the most likely utterances, given cases of multiple possible ways to say the same thing, or given words that sound alike and might be easily confused by the recognizer.

3D AUDIO AND AUDIO FOR VIRTUAL ENVIRONMENTS

The human auditory system is good at using only the information at the eardrums to determine the locations of sound-producing objects. The primary perceptual cues are time delays between the two ears, amplitude differences between the two ears, filter functions related to the shadowing effects of the head and shoulders, and complex filter functions related to the twists and turns of the pinnae (outer ears). Three-dimensional audio (22) uses headphones or speakers to place sources at arbitrary perceived locations around the listener's head. If headphones are used and the appropriate cues are synthesized into the "binaural" signals presented to the two ears, virtual source locations can be manipulated directly. Using speakers requires more signal processing to cancel the effects of each speaker signal getting to both ears. Virtual reality systems using helmets with vision systems often have an integrated stereo audio capability, and the head-tracking systems used to manipulate the visual displays in these systems can be used to manipulate the audio displays as well.

Multispeaker systems such as quadrophonic sound and ambisonics (23) have historically been used in certain research settings, but have not been broadly adopted for various reasons of economy, multiple competing standards, and lack of commercially available program material. Recently, various surround sound formats such as Dolby ProLogic, Dolby 5.1 Digital, DTS, and Dolby AC-3 are entering the home entertainment market. These promise to give a multispeaker capability to future multimedia systems, making more immersive audio experiences in the home likely. In addition to the movie and audiophile music titles, which will initially drive the market for these new systems, one should expect to see games and other multimedia content emerge, which can take advantage of the multiple speaker systems.

BIBLIOGRAPHY

1. G. Burdea, *Force and Touch Feedback for Virtual Reality,* New York: Wiley, 1996.
2. K. Steiglitz, *A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music,* Menlo Park, CA: Addison-Wesley, 1995.
3. R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1983.
4. J. Smith and P. Gossett, A flexible sampling rate conversion method, *IEEE Proc. Acoust., Speech Signal Process.,* **2**: 1984, pp. 19.4.1–19.4.4.
5. L. Rabiner and R. Schafer, *Digital Processing of Speech Signals,* Englewood Cliffs, NJ: Prentice-Hall, 1978.
6. International Multimedia Association, ADPCM, IMA Compatibility Proc., Sect. 6, May 1992.
7. ISO/IEC Working Papers and Standards Rep., JTCI SC29, WG11 N0403, MPEG 93/479, 1993.
8. G. Davidson, W. Anderson, and A. Lovrich, A low-cost adaptive transform decoder implementation for high-quality audio, IEEE Publication # 0-7803-0532-9/92, 1992.
9. MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification,* La Habra, CA: MMA, 1996.
10. S. Jungleib, *General MIDI,* Madison, WI: A-R Editions, 1995.
11. The XM spec: Is MIDI 2.0 finally at hand? *Keyboard Magazine,* June, 1995.
12. Special Issue on the ZIPI Music Interface Language, *Comput. Music J.,* **18** (4): Cambridge, MA: MIT Press, 1994.
13. L. Rabiner, A digital signal processing approach to interpolation, *Proc. IEEE,* **61**: 692–702, 1973.
14. R. Bracewell, *The Fourier Transform and Its Applications,* New York: McGraw-Hill, 1986.
15. C. Roads and J. Strawn (eds.), *Foundations of Computer Music,* Cambridge, MA: MIT Press, 1985.
16. Two Special Issues on Physical Modeling, *Comput. Music J.,* **16** (4): 1992, **17** (1): 1993.
17. M. Mathews, *The Technology of Computer Music,* Cambridge, MA: MIT Press, 1969.
18. B. Grill et al. (eds.), *ISO 14496-3 (MPEG-4 Audio),* Committee Draft, *ISO / IEC JTCI / SC29 / WG11,* document W1903, Fribourg CH, October 1997.
19. A. Spanias, Speech coding: A tutorial review, *Proc. IEEE,* **82**: 1541–1582, 1994.

20. J. Makhoul, Linear prediction: A tutorial review, *Proc. IEEE,* **63**: 561–580, 1975.
21. J. Picone, Continuous speech recognition using hidden Markov models, *IEEE Mag. Acoust. Speech Signal Process.,* **7** (3): 26–41, 1990.
22. D. Begault, *3-D Sound for Virtual Reality and Multimedia,* San Diego: Academic Press, 1994.
23. M. Gerzon, Ambisonics in multichannel broadcasting and video, *J. Audio Eng. Soc.,* **33** (11): 859–871, 1985.

Reading List

D. O'Shaughnessy, *Speech Communication: Man and Machine,* Reading, MA: Addison-Wesley, 1987.
C. Roads, *A Computer Music Tutorial,* Cambridge, MA: MIT Press, 1996.

PERRY R. COOK
Princeton University

MULTIMEDIA HOLOGRAPHIC STORAGE. See HOLOGRAPHIC STORAGE.

MULTIMEDIA INFORMATION DELIVERY. See VIEWDATA.