Chapter 3

Sound / Audio

Sound is a physical phenomenon produced by the vibration of matter, such as a violin string, or a block of wood. As the matter vibrates, pressure variations are created in the air surrounding it. This alteration of high and low pressure is propagated through the air in a wave-like motion. When a wave reaches the human ear, a sound is heard.

Sound methodology and audio techniques engage in processing these sound waves (acoustic signals). Important topics in this area are coding, storage on recorders or digital audio tapes, music and speech processing.

In this chapter a discussion of sound, music coding and speech processing is presented. Basic concepts and formats of sound, as well as representation of sound in the computer [Boo87, Tec89] are presented. Because multimedia applications use audio in the form of music and/or speech, music and its MIDI standard, as well as speech synthesis, speech recognition and speech transmission [Loy85, Fla72, FS92, O’S90, Fal85, Bri86, Ace93, Sch92], are described.

The topic of audio data storage on optical discs is presented in Chapter 7, for the reason that the principles and technology used are not restricted to audio. The compression of audio/video signals is also described separately in Chapter 6, because similar methods are used for compressing the data of different media. Further, the commonalities among the media are emphasized by being treated together.
3.1 Basic Sound Concepts

Sound is produced by the vibration of matter. During the vibration, pressure variations are created in the air surrounding it. The pattern of the oscillation is called a waveform (Figure 3.1 [Tec89]).

![Oscillation of an air pressure wave.](image)

The waveform repeats the same shape at regular intervals and this portion is called a period. Since sound waves occur naturally, they are never perfectly smooth or uniformly periodic. However, sounds that display a recognizable periodicity tend to be more musical than those that are nonperiodic. Examples of periodic sound sources are musical instruments, vowel sounds, the whistling wind and bird songs. Nonperiodic sound sources include unpitched percussion instruments, coughs and sneezes and rushing water.

**Frequency**

The frequency of a sound is the reciprocal value of the period; it represents the number of periods in a second and is measured in hertz (Hz) or cycles per second (cps). A convenient abbreviation, kHz (kilohertz), is used to indicate thousands of oscillations per second: 1 kHz equals 1000 Hz [Boo87]. The frequency range is divided into:
3.1. BASIC SOUND CONCEPTS

<table>
<thead>
<tr>
<th>Sound Type</th>
<th>Frequency Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infra-sound</td>
<td>from 0 to 20 Hz</td>
</tr>
<tr>
<td>Human hearing frequency</td>
<td>from 20 Hz to 20 kHz</td>
</tr>
<tr>
<td>Ultrasound</td>
<td>from 20 kHz to 1 GHz</td>
</tr>
<tr>
<td>Hypersound</td>
<td>from 1 GHz to 10 THz</td>
</tr>
</tbody>
</table>

Multimedia systems typically make use of sound only within the frequency range of human hearing. We will call sound within the human hearing range audio and the waves in this frequency range acoustic signals [Boo87]. For example, speech is an acoustic signal produced by humans; music signals have a frequency range between 20 Hz and 20 kHz. Besides speech and music, we denote any other audio signal as noise.

Amplitude

A sound also has an amplitude, a property subjectively heard as loudness. The amplitude of a sound is the measure of the displacement of the air pressure wave from its mean, or quiescent state.

3.1.1 Computer Representation of Sound

The smooth, continuous curve of a sound waveform is not directly represented in a computer. A computer measures the amplitude of the waveform at regular time intervals to produce a series of numbers. Each of these measurements is a sample. Figure 3.2 illustrates one period of a digitally sampled waveform.

The mechanism that converts an audio signal into digital samples is the Analog-to-Digital Converter (ADC). The reverse conversion is performed by a Digital-to-Analog Converter (DAC). The AM79C30A Digital Subscriber Controller chip is an example of an ADC and is available on SPARCstations™. Desktop SPARC™ systems include a built-in speaker for audio output. DAC is also available as a standard UNIX™ device. For example, SPARCserver 6xx systems do not have an internal speaker, but support an external microphone and speaker.
Sampling Rate

The rate at which a continuous waveform (Figure 3.1) is sampled is called the sampling rate. Like frequencies, sampling rates are measured in Hz. The CD standard sampling rate of 44100 Hz means that the waveform is sampled 44100 times per second. This seems to be above the frequency range the human ear can hear. However, the bandwidth (which in this case is 20000 Hz - 20 Hz = 19980 Hz) that digitally sampled audio signal can represent, is at most equal to half of the CD standard sampling rate (44100 Hz). This is an application of the Nyquist sampling theorem. (“For lossless digitization, the sampling rate should be at least twice the maximum frequency responses.”) Hence, a sampling rate of 44100 Hz can only represent frequencies up to 22050 Hz, a boundary much closer to that of human hearing.

Quantization

Just as a waveform is sampled at discrete times, the value of the sample is also discrete. The resolution or quantization of a sample value depends on the number of bits used in measuring the height of the waveform. An 8-bit quantization yields 256 possible values; 16-bit CD-quality quantization results in over 65536 values.

Figure 3.3 presents a 3-bit quantization. The sampled waveform with a 3-bit quan-
3.1. BASIC SOUND CONCEPTS

Figure 3.3: Three-bit quantization.

Quantization results in only eight possible values: .75, .5, .25, 0, -.25, -.5, -.75 and -1. The shape of the waveform becomes less discernible with a lowered quantization, i.e., the lower the quantization, the lower the quality of the sound (the result might be a buzzing sound).

Sound Hardware

Before sound can be processed, a computer needs input/output devices. Microphone jacks and built-in speakers are devices connected to an ADC and DAC, respectively for the input and output of audio.

3.1.2 Audio Formats

The AM79C30A Digital Subscriber Controller provides voice-quality audio. This converter uses an 8-bit $\mu$-law encoded quantization and a sampling rate of 8000 Hz. This representation is considered fast and accurate enough for telephone-quality speech input.

CD-quality audio is generated if the stereo DAC operates at 44100 samples per second with a 16-bit linear PCM (Pulse Code Modulation) encoded quantization [JB89].
The above examples of telephone-quality and CD-quality audio indicate that important format parameters for specification of audio are: sampling rate (e.g., 8012.8 samples/second) and sample quantization (e.g., 8-bit quantization).

### 3.2 Music

The relationship between music and computers has become more and more important, specially considering the development of MIDI (Music Instrument Digital Interface) and its important contributions in the music industry today. The MIDI interface between electronic musical instruments and computers is a small piece of equipment that plugs directly into the computer’s serial port and allows the transmission of music signals. MIDI is considered to be the most compact interface that allows full-scale output.

#### 3.2.1 MIDI Basic Concepts

MIDI is a standard that manufacturers of electronic musical instruments have agreed upon. It is a set of specifications they use in building their instruments so that the instruments of different manufacturers can, without difficulty, communicate musical information between one another [Loy85].

A MIDI interface has two different components:

- **Hardware** connects the equipment. It specifies the physical connection between musical instruments, stipulates that a **MIDI port** is built into an instrument, specifies a **MIDI cable** (which connects two instruments) and deals with electronic signals that are sent over the cable.

- A **data format** encodes the information traveling through the hardware. A MIDI data format does not include an encoding of individual samples as the audio format does (Section 3.1.2). Instead of individual samples, an instrument-connected data format is used. The encoding includes, besides the instrument specification, the notion of the beginning and end of a note, basic frequency
and sound volume. MIDI data allow an encoding of about 10 octaves, which corresponds to 128 notes.

The MIDI data format is digital; the data are grouped into MIDI messages. Each MIDI message communicates one musical event between machines. These musical events are usually actions that a musician performs while playing a musical instrument. The action might be pressing keys, moving slider controls, setting switches and adjusting foot pedals.

When a musician presses a piano key, the MIDI interface creates a MIDI message where the beginning of the note with its stroke intensity is encoded. This message is transmitted to another machine. In the moment the key is released, a corresponding signal (MIDI message) is transmitted again. For ten minutes of music, this process creates about 200 Kbytes of MIDI data, which is essentially less than the equivalent volume of a CD-audio coded stream in the same time.

If a musical instrument satisfies both components of the MIDI standard, the instrument is a MIDI device (e.g., a synthesizer), capable of communicating with other MIDI devices through channels. The MIDI standard specifies 16 channels. A MIDI device (musical instrument) is mapped to a channel. Music data, transmitted through a channel, are reproduced at the receiver side with the synthesizer instrument. The MIDI standard identifies 128 instruments, including noise effects (e.g., telephone, air craft), with unique numbers. For example, 0 is for the Acoustic Grand Piano, 12 for the marimba, 40 for the violin, 73 for the flute, etc.

Some instruments allow only one note to be played at a time, such as the flute. Other instruments allow more than one note to be played simultaneously, such as the organ. The maximum number of simultaneously played notes per channel is a main property of each synthesizer. The range can be from 3 to 16 notes per channel.

To tune a MIDI device to one or more channels, the device must be set to one of the MIDI reception modes. There are four modes:

- Mode 1: Omni On/Poly;
- Mode 2: Omni On/Mono;
- Mode 3: Omni Off/Poly;
- Mode 4: Omni Off/Mono

The first half of the mode name specifies how the MIDI device monitors the incoming MIDI channels. If Omni is turned on, the MIDI device monitors all the MIDI channels and responds to all channel messages, no matter which channel they are transmitted on. If Omni is turned off, the MIDI device responds only to channel messages sent on the channel(s) the device is set to receive.

The second half of the mode name tells the MIDI device how to play notes coming in over the MIDI cable. If the option Poly is set, the device can play several notes at a time. If the mode is set to Mono, the device plays notes like a monophonic synthesizer – one note at a time.

### 3.2.2 MIDI Devices

Through the MIDI interface, a computer can control output of individual instruments. On the other hand, the computer can receive, store or process coded musical data through the same interface. The data are generated with a keyboard and reproduced through a sound generator. A sequencer can store data. Further, it may also modify the musical data. In a multimedia system, the sequencer is a computer application.

The heart of any MIDI system is the MIDI synthesizer device. A typical synthesizer looks like a simple piano keyboard with a panel full of buttons, but it is far more (more detailed information on synthesizers can be found in [Boo87]). Most synthesizers have the following common components:

- **Sound Generators**

  Sound generators do the actual work of synthesizing sound; the purpose of the rest of the synthesizer is to control the sound generators. The principal purpose of the generator is to produce an audio signal that becomes sound when fed into a loudspeaker. By varying the voltage oscillation of the audio
signal, a sound generator changes the quality of the sound — its pitch, loudness and tone color — to create a wide variety of sounds and notes.

Internally, sound generation can be done in different ways. One way is to store the acoustic signals as MIDI data in advance. Afterwards, the stored MIDI data are transformed with a digital-analog adapter into acoustic signals. Individual notes are composed in a timely fashion. Another method is to create acoustic signals synthetically.

- **Microprocessor**
  The microprocessor communicates with the keyboard to know what notes the musician is playing, and with the control panel to know what commands the musician wants to send to the microprocessor. The microprocessor then specifies note and sound commands to the sound generators; in other words, the microprocessor sends and receives MIDI messages.

- **Keyboard**
  The keyboard affords the musician's direct control of the synthesizer. Pressing keys on the keyboard signals the microprocessor what notes to play and how long to play them. Some synthesizer keyboards can also signal to the microprocessor how loud to play the notes and whether to add *vibrato* or other effects to the notes. The sound intensity of a tone depends on the speed and acceleration of the key pressure. The keyboard should have at least five octaves with 61 keys.

- **Control Panel**
  The control panel controls those functions that are not directly concerned with notes and durations (controlled by the keyboard). Panel controls include: a slider that sets the overall volume of the synthesizer, a button that turns the synthesizer on and off, and a menu that calls up different patches for the sound generators to play.

- **Auxiliary Controllers**
  Auxiliary controllers are available to give more control over the notes played on the keyboard. Two very common variables on a synthesizer are *pitch bend* and *modulation*. Pitch bend controllers can bend pitch up and down, adding
portamento (a smooth, uninterrupted glide in passing from one tone to another) to notes; modulation controllers can increase or decrease effects such as vibrato.

- **Memory**

Synthesizer memory is used to store patches for the sound generators and settings on the control panel. Many synthesizers also have a slot for *external memory cartridges*. By using several memory cartridges, the musician can plug in a different cartridge each time s/he wants a set of new sounds for the synthesizer.

There are many other MIDI devices that augment the standard synthesizer in a MIDI system. Examples are drum machines which specialize in percussion sounds and rhythms, the master keyboard which increases the quality of the synthesizer keyboard, guitar controllers, guitar synthesizers, drum pad controllers and so on.

An important MIDI device is a *sequencer*, which can be a drum machine, computer or dedicated sequencer. A sequencer was used originally as a storage server for generated MIDI data. Today, a sequencer, being a computer, becomes additionally a music editor. Data can be modified in a proper way because of their digital data representation. There are several possibilities to represent musical data. The most common representation and manipulation of data are musical notes. The musical piece appears on the screen in the form of a sheet of music. The sequencer transforms the notes into MIDI messages (Sections 3.2.1, 3.2.3). Another representation is a direct input of MIDI messages. Here, the user specifies required musical events per channel with their time dependencies. This input depends on the keyboard type.

### 3.2.3 MIDI Messages

MIDI messages transmit information between MIDI devices and determine what kinds of musical events can be passed from device to device. The format of MIDI messages consists of the *status byte* (the first byte of any MIDI message), which describes the kind of message, and *data bytes* (the following bytes). MIDI messages are divided into two different types:
• **Channel Messages**

Channel messages go only to specified devices. There are two types of channel messages:

- *Channel voice messages* send actual performance data between MIDI devices, describing keyboard action, controller action and control panel changes. They describe music by defining pitch, amplitude, timbre, duration and other sound qualities. Each message has at least one and usually two data bytes that accompany the status byte to describe these sound qualities. Examples of channel voice messages are *Note On, Note Off, Channel Pressure, Control Change*, etc.

- *Channel mode messages* determine the way that a receiving MIDI device responds to channel voice messages. They set the MIDI channel receiving modes for different MIDI devices, stop spurious notes from playing and affect local control of a device. Examples of such messages are *Local Control, All Notes Off, Omni Mode Off*, etc.

• **System Messages**

System messages go to all devices in a MIDI system because no channel numbers are specified. There are three types of system messages:

- *System real-time messages* are very short and simple, consisting of only one byte. They carry extra data with them. These messages synchronize the timing of MIDI devices in performance; therefore, it is important that they be sent at precisely the time they are required. To avoid delays, these messages are sent in the middle of other messages, if necessary. Examples of such messages are *System Reset, Timing Clock (MIDI clock)*, etc.

- *System common messages* are commands that prepare sequencers and synthesizers to play a song. The various messages enable you to select a song, find a common starting place in the song and tune all the synthesizers if they need tuning. Examples are *Song Select, Tune Request*, etc.

- *System exclusive messages* allow MIDI manufacturers to create customized MIDI messages to send between their MIDI devices. This coding starts
with a *system-exclusive-message*, where the manufacturer is specified, and ends with an *end-of-exclusive message*.

### 3.2.4 MIDI and SMPTE Timing Standards

MIDI reproduces traditional note length using *MIDI clocks*, which are represented through *timing clock* messages. Using a MIDI clock, a receiver can synchronize with the clock cycles of the sender. For example, a MIDI clock helps keep separate sequencers in the same MIDI system playing at the same tempo. When a master sequencer plays a song, it sends out a stream of ‘Timing Clock’ messages to convey the tempo to other sequencers. The faster the Timing Clock messages come in, the faster the receiving sequencer plays the song. To keep a standard timing reference, the MIDI specifications state that 24 MIDI clocks equal one quarter note.

As an alternative, the *SMPTE timing standard* (Society of Motion Picture and Television Engineers) can be used. The SMPTE timing standard was originally developed by NASA as a way to mark incoming data from different tracking stations so that receiving computers could tell exactly what time each piece of data was created [Boo87]. In the film and video version promoted by the SMPTE, the SMPTE timing standard acts as a very precise clock that stamps a time reading on each frame and fraction of a frame, counting from the beginning of a film or video. To make the time readings precise, the SMPTE format consists of `hours:minutes:seconds:frames:bits` (e.g., 30 frames per second), uses a 24-hour clock and counts from 0 to 23 before recycling to 0. The number of frames in a second differs depending on the type of visual medium. To divide time even more precisely, SMPTE breaks each frame into 80 bits (not digital bits). When SMPTE is counting bits in a frame, it is dividing time into segments as small as one twenty-five hundredth of a second.

Because many film composers now record their music on a MIDI recorder, it is desirable to synchronize the MIDI recorder with video equipment. A SMPTE synchronizer should be able to give a time location to the MIDI recorder so it can move to that location in the MIDI score (pre-recorded song) to start playback or recording. But MIDI recorders cannot use incoming SMPTE signals to control their recording and playback. The solution is a MIDI/SMPTE synchronizer that converts SMPTE into MIDI, and vice versa. The MIDI/SMPTE synchronizer lets the user specify
different tempos and the exact points in SMPTE timing at which each tempo is to start, change, and stop. The synchronizer keeps these tempos and timing points in memory. As a SMPTE video deck plays and sends a stream of SMPTE times to the synchronizer, the synchronizer checks the incoming time and sends out MIDI clocks at a corresponding tempo.

### 3.2.5 MIDI Software

Once a computer is connected to a MIDI system, a variety of MIDI applications can run on it. Digital computers afford the composer or sound designer unprecedented levels of control over the evolution and combination of sonic events.

The software applications generally fall into four major categories:

- **Music recording and performance applications**

  This category of applications provides functions such as recording of MIDI messages as they enter the computer from other MIDI devices, and possibly editing and playing back the messages in performance.

- **Musical notations and printing applications**

  This category allows writing music using traditional musical notation. The user can then play back the music using a performance program or print the music on paper for live performance or publication.

- **Synthesizer patch editors and librarians**

  These programs allow information storage of different synthesizer patches in the computer’s memory and disk drives, and editing of patches in the computer.

- **Music education applications**

  These software applications teach different aspects of music using the computer monitor, keyboard and other controllers of attached MIDI instruments.

The main issue in current MIDI-based computer music systems is *interactivity*. Music is a temporal art, and any computer program dealing with music must have
sophisticated facilities for representing time and for scheduling processes to occur at a particular point in time. This capability of music applications became possible because of increased computational speeds (e.g., the computational speeds needed to execute compositional algorithms in real-time are available). Therefore, current computer music systems are able to modify their behavior in response to input from other performing musicians [Row93].

The processing chain of interactive computer music systems can be conceptualized in three stages:

- **The sensing stage**, when data are collected from controllers reading gesture information from human performers on stage.
- **The processing stage**, when the computer reads and interprets information coming from the sensors and prepares data for the response stage.
- **The response stage**, when the computer and some collection of sound-producing devices share in realizing a musical output.

Commercial manufacturers dominate in providing MIDI devices, such as MIDI controllers and synthesizers, for sensing and response stages. The processing stage has commercial entries as well, most notably MIDI sequencers. It is in processing, however, that individual conceptions of interactive music are most readily expressed, in any of a variety of programming languages with temporal and MIDI extensions.

Commercial interactive music systems appeared in the mid-1980s. Two groundbreaking efforts in this field were *M* and *Jam Factory* [Zic87]. Among the breakthroughs implemented by these programs was the graphic control panel, which allowed access to the values of global variables affecting musical output. Manipulating the graphic controls had an immediately audible effect. The sensing performed by *M* and *Jam Factory* centered around reading manipulations of the control panel and interpreting an incoming stream of MIDI events. Responses were sent out as MIDI.

In 1990, Opcode Systems released *Max™*, a graphical programming environment for interactive music systems. *Max* is an object-oriented programming language, in which programs are realized by manipulating graphic objects on a computer screen and making connection between them [DZ90].
3.3. SPEECH

An interactive computer music system, emphasizing composition and performance, is the MIT system *Cypher* [Row93]. The program has two main components: a listener and a player. The listener characterizes performances represented by streams of MIDI data, which could be coming from a human performer, another computer program or even *Cypher* itself. The player generates and plays musical material.

NeXT™ Computer has a music system based on MIDI that combines the synthesis power and generality of a mainframe computer with the performance flexibility of a keyboard synthesizer. The system, *Music Kit*, helps the composer or performer construct applications that create, organize, process and render music data [JB89].

3.3 Speech

Speech can be “perceived,” “understood” and “generated” by humans and also by machines. A human adjusts himself/herself very efficiently to different speakers and their speech habits. Despite different dialects and pronunciation, the speech can be well understood by humans. The brain can recognize the very fine line between speech and noise. For this purpose, both ears are used, because filtering with only one ear is substantially more difficult for the listener. The human speech signal comprises a subjective lowest spectral component known as the *pitch*, which is not proportional to frequency. The human ear is most sensitive in the range from 600 Hz to 6000 Hz. Fletscher and Munson have shown that the human ear is substantially less sensitive to low and very high frequencies than to frequencies around 1 kHz. Speech signals have two properties which can be used in speech processing:

- Voiced speech signals show during certain time intervals almost periodic behavior. Therefore, we can consider these signals as *quasi-stationary signals* for around 30 milliseconds.

- The spectrum of audio signals shows characteristic maxima, which are mostly 3-5 frequency bands. These maxima, called *formants*, occur because of resonances of the vocal tract.

For description and modeling of human speech generation, see [All85, BN93].
A machine can also support speech generation and recognition. With computers, one can synthetically generate speech, where the generated signals do not sound quite natural but can be easily understood. An example of such an artificial sounding voice can be heard at the Atlanta (Georgia, USA) airport. On the other hand, a voice can sound natural but may be very difficult to understand. Speech recognition often uses matching rules or statistically based methods. Today, workstations and personal computers can recognize 25,000 possible words. Problems are caused when dialects, emotional pronunciation and environmental noises are part of the audio signal. There are, and will continue to be in the near future, considerable differences between the speech generation and recognition efficiencies/capabilities of the human brain and a high-performance computer [Aceg3, Mamg93].

In the following two sections we describe in more detail some crucial issues of computer-generated speech and recognition.

### 3.3.1 Speech Generation

Speech generation research has a long history. By the middle of the 19th century, Helmholtz had already built a mechanical vocal tract coupling together several mechanical resonators with which sound could be generated. In 1940, Dudley produced the first speech synthesizer through imitation of mechanical vibration using electrical oscillation [Fal85].

An important requirement for speech generation is **real-time signal generation**. With such a requirement met, a speech output system could transform text into speech automatically without any lengthy preprocessing. Some applications only need a limited vocabulary; an example is the spoken time announcement of a telephone answering service. However, most applications need a large vocabulary, if not an unlimited vocabulary.

Generated speech must be **understandable** and must sound **natural**. The requirement of understandable speech is a fundamental assumption, and the natural sound of speech increases user acceptance.
3.3. SPEECH

Basic Notions

For further discussion we introduce some notions of importance:

- The lowest periodic spectral component of the speech signal is called the fundamental frequency. It is present in a voiced sound.

- A phone is the smallest speech unit, such as the m of mat and the b of bat in English, that distinguish one utterance or word from another in a given language.

- Allophones mark the variants of a phone. For example, the aspirated p of pit and the unaspirated p of spit are allophones of the English phoneme p.

- The morph marks the smallest speech unit which carries a meaning itself. Therefore, consider is a morph, but reconsideration is not.

- A voiced sound is generated through the vocal cords. m, v and l are examples of voiced sounds. The pronunciation of a voiced sound depends strongly on each speaker.

- During the generation of an unvoiced sound, the vocal cords are opened. f and s are unvoiced sounds. Unvoiced sounds are relatively independent from the speaker.

Exactly, there are:

- Vowels – a speech sound created by the relatively free passage of breath through the larynx and oral cavity, usually forming the most prominent and central sound of a syllable (e.g., u from hunt);

- Consonants – a speech sound produced by a partial or complete obstruction of the air stream by any of the various constrictions of the speech organs (e.g., voiced consonants, such as m from mother, fricative voiced consonants, such as v from voice, fricative voiceless consonants, such as s from nurse, plosive consonants, such as d from daily and affricate consonants, such as dg from knowledge, or ch from chew).
Reproduced Speech Output

The easiest method of speech generation/output is to use prerecorded speech and play it back in a timely fashion [BN93]. The speech can be stored as PCM (Pulse Code Modulation) samples. Further data compression methods, without using language typical properties, can be applied to recorded speech (see Chapter 6).

Time-dependent Sound Concatenation

Speech generation/output can also be performed by sound concatenation in a timely fashion [Ril89]. Individual speech units are composed like building blocks, where the composition can occur at different levels. In the simplest case, the individual phones are understood as speech units. Figure 3.4 shows the individual phones of the word *crumb*. It is possible with just a few phones to create an unlimited vocabulary.

![Figure 3.4: Phone sound concatenation.](image)

However, transitions between individual phones prove to be extremely problematic. Therefore, the phones in their environment, i.e., the allophones, are considered in the second level. But the transition problem is not solved sufficiently on this level either. Two phones can constitute a diphone (from di-phone). Figure 3.5 shows the word *crumb*, which consists of an ordered set of diphones.

![Figure 3.5: Diphone sound concatenation.](image)

To make the transition problem easier, syllables can be created. The speech is generated through the set of syllables. Figure 3.6 shows the syllable sound of the
word *crumb*. The best pronunciation of a word is achieved through storage of the whole word. This leads toward synthesis of the speech sequence (Figure 3.7).

Transitions between individual sound units create an essential problem, called *coarticulation*, which is the mutual sound influence throughout several sounds. This influence between individual sound units arises because physical constraints, such as mass and speed of the articulator in the vocal tract, influence the articulation of consecutive phones.

Additionally, *prosody* should be considered during speech generation/output. Prosody means the stress and melody course. For example, pronunciation of a question differs strongly from a statement. Therefore, prosody depends on the semantics of the speech and this has to be taken into consideration during time-dependent sound concatenation [Wai88].

**Frequency-dependent Sound Concatenation**

Speech generation/output can also be based on a frequency-dependent sound concatenation, e.g., through a formant-synthesis [Ril89]. Formants are frequency maxima in the spectrum of the speech signal. Formant synthesis simulates the vocal
tract through a filter. The characteristic values are the filter's middle frequencies and their bandwidths. A pulse signal with a frequency, corresponding to the fundamental speech frequency, is chosen as a simulation for voiced sounds. On the other hand, unvoiced sounds are created through a noise generator.

Individual speech elements (e.g., phones) are defined through the characteristic values of the formants. Similar problems to the time-dependent sound concatenation exist here. The transitions, known as coarticulation, present the most critical problem. Additionally, the respective prosody has to be determined.

New sound-specific methods provide a sound concatenation with combined time and frequency dependencies. Initial results show that new methods generate fricative and plosive sounds with higher quality.

Human speech can be generated using a multi-pole lattice filter. The first four or five formants, occurring in human speech are modeled correctly with this filter type. Further, unvoiced sounds, created by vocal chords, are simulated through a noise and tone generator. The method used for the sound synthesis in order to simulate human speech is called the Linear-Predictive Coding (LPC) method. This method is very similar to the formant synthesis described above. A further possibility to simulate human speech consists of implementing a tube model. Here, a simplified mechanical tube model approximates the human tube as the speech generation system.

Using speech synthesis, an existent text can be transformed into an acoustic signal. Figure 3.8 shows the typical components of the system. In the first step, transcription is performed, in which text is translated into sound script. Most transcription methods work here with letter-to-phone rules and a Dictionary of Exceptions stored.
3.3. SPEECH

in a library. The generation of such a library is work-extensive, but using the interactive control of the user it can be improved continuously. The user recognizes the formula deficiency in the transcription and improves the pronunciation manual; therefore, his/her knowledge becomes part of the letter-to-phone rules and the Dictionary of Exceptions. The solution can be either individual or generally accessible rules and a Dictionary of Exceptions.

In the second step, the sound script is translated into a speech signal. Time or frequency-dependent concatenation can follow. While the first step is always a software solution, the second step is most often implemented with signal processors or even dedicated processors.

Besides the problems of coarticulation and prosody, ambiguous pronunciation must be considered. Pronunciation can be performed correctly only with additional knowledge of the content, i.e., it is semantic-dependent. An example is the word lead. It can be used as a noun to describe a metal, but when used as a verb (with a different pronunciation as the noun) it means “to guide people.”

3.3.2 Speech Analysis

Speech analysis/input deals with the research areas shown in Figure 3.9 [Bri86]:

- Human speech has certain characteristics determined by a speaker. Hence, speech analysis can serve to analyze who is speaking, i.e., to recognize a speaker for his/her identification and verification. The computer identifies and verifies
the speaker using an acoustic fingerprint. An acoustic fingerprint is a digitally stored speech probe (e.g., certain statement) of a person; for example, a company that uses speech analysis for identification and verification of employees. The speaker has to say a certain sentence into a microphone. The computer system gets the speaker’s voice, identifies and verifies the spoken statement, i.e., determines if the speech probe matches the speaker’s spoken statement.

- Another main task of speech analysis is to analyze what has been said, i.e., to recognize and understand the speech signal itself. Based on speech sequence, the corresponding text is generated. This can lead to a speech-controlled typewriter, a translation system or part of a workplace for the handicapped.

- Another area of speech analysis tries to research speech patterns with respect to how a certain statement was said. For example, a spoken sentence sounds differently if a person is angry or calm. An application of this research could be a lie detector.

Speech analysis is of strong interest for multimedia systems. Together with speech synthesis, different media transformations can be implemented.

The primary goal of speech analysis is to correctly determine individual words with probability \( \leq 1 \). A word is recognized only with a certain probability. Here, environmental noise, room acoustics and a speaker’s physical and psychological conditions play an important role.

For example, let’s assume extremely bad individual word recognition with a probability of 0.95. This means that 5% of the words are incorrectly recognized. If we have a sentence with three words, the probability of recognizing the sentence correctly is \( 0.95 \times 0.95 \times 0.95 = 0.857 \). This small example should emphasize the fact that speech analysis systems should have a very high individual word recognition probability. Figure 3.10 shows schematically a speech recognition system. The system is divided into system components according to a basic principle: “Data Reduction Through Property Extraction”. First, speech analysis occurs where properties must be determined. Properties are extracted by comparison of individual speech element characteristics with a sequence of in advance given speech element characteristics. The characteristics are quantified where the concrete speech elements are present.
Second, the speech elements are compared with existent references to determine the mapping to one of the existent speech elements. The identified speech can be stored, transmitted or processed as a parametrized sequence of speech elements.

Concrete implementations mostly use dedicated building blocks or signal processors for characteristics extraction. Usually, the comparison and decision are executed through the main system processor. The computer’s secondary storage contains the letter-to-phone rules, a Dictionary of Exceptions and the reference characteristics. The concrete methods differ in the definition of the characteristics. The principle of “Data Reduction Through Property Extraction,” shown in Figure 3.10, can be applied several times to different characteristics. The system which provides recognition and understanding of a speech signal (Figure 3.11) applies this principle several times as follows:

- In the first step, the principle is applied to a sound pattern and/or word model. An acoustical and phonetical analysis is performed.

- In the second step, certain speech units go through syntactical analysis; thereby, the errors of the previous step can be recognized. Very often during the first step, no unambiguous decisions can be made. In this case, syntactical analysis provides additional decision help and the result is a recognized speech.
The third step deals with the semantics of the previously recognized language. Here the decision errors of the previous step can be recognized and corrected with other analysis methods. Even today, this step is non-trivial to implement with current methods known in artificial intelligence and neural nets research. The result of this step is an understood speech.

These steps work mostly under the consideration of time and/or frequency-dependent sounds. The same criteria and speech units (formants, phones, etc.) are considered as in speech generation/output (discussed in Section 3.3.1).

There are still many problems into which speech recognition research is being conducted:

- A specific problem is presented by room acoustics with existent environmental noise. The frequency-dependent reflections of a sound wave from walls and objects can overlap with the primary sound wave.
- Further, word boundaries must be determined. Very often neighboring words flow into one another.
- For the comparison of a speech element to the existing pattern, time normalization is necessary. The same word can be spoken quickly or slowly. However, the time axis cannot be modified because the extension factors are not proportional to the global time interval. There are long and short voiceless sounds (e.g., s, sh). Individual sounds are extended differently and need a minimal time duration for their recognition.
Speech recognition systems are divided into *speaker-independent recognition systems* and *speaker-dependent recognition systems*. A speaker-independent system can recognize with the same reliability essentially fewer words than a speaker-dependent system because the latter is trained in advance. Training in advance means that there exists a training phase for the speech recognition system, which takes a half an hour. Speaker-dependent systems can recognize around 25,000 words; speaker-independent systems recognize a maximum of about 500 words, but with a worse recognition rate. These values should be understood as gross guidelines. In a concrete situation, the marginal conditions must be known. (e.g., Was the measurement taken in a sound deadening room?, Does the speaker have to adapt to the system to simplify the time normalization?, etc.)

### 3.3.3 Speech Transmission

The area of speech transmission deals with efficient coding of the speech signal to allow speech/sound transmission at low transmission rates over networks. The goal is to provide the receiver with the same speech/sound *quality* as was generated at the sender side. This section includes some principles that are connected to speech generation and recognition.

- **Signal Form Coding**
  
  This kind of coding considers no speech-specific properties and parameters. Here, the goal is to achieve the most efficient coding of the audio signal. The data rate of a PCM-coded stereo-audio signal with *CD-quality* requirements is:

  \[
  \text{rate} = 2 \times \frac{44100}{s} \times \frac{16\text{bit}}{8\text{bit/byte}} = 176,400 \text{ bytes/s} = 1,411,200 \text{ bits/s}
  \]

  Telephone quality, in comparison to CD-quality, needs only 64 Kbit/s. Using *Difference Pulse Code Modulation (DPCM)*, the data rate can be lowered to 56 Kbits/s without loss of quality. *Adaptive Pulse Code Modulation (ADPCM)* allows a further rate reduction to 32 Kbits/s.

- **Source Coding**
Parameterized systems work with source coding algorithms. Here, the specific speech characteristics are used for data rate reduction.

Channel vo-coder is an example of such a parameterized system (Figure 3.12). The channel vo-coder is an extension of a sub-channel coding. The signal is divided into a set of frequency channels during speech analysis because only certain frequency maxima are relevant to speech. Additionally, the differences between voiced and unvoiced sounds are taken into account. Voiceless sounds are simulated by the noise generator. For generation of voiced sounds, the simulation comes from a sequence of pulses. The rate of the pulses is equivalent to the a priori measured basic speech frequency. The data rate of about 3 Kbits/s can be generated with a channel vo-coder; however the quality is not always satisfactory.

Major effort and work on further data rate reduction from 64 Kbits/s to 6 Kbits/s is being conducted, where the compressed signal quality should correspond, after a decompression, to the quality of an uncompressed 64 Kbits/s signal.

- **Recognition/Synthesis Methods**

There have been attempts to reduce the transmission rate using pure recognition/synthesis methods. Speech analysis (recognition) follows on the sender
3.3. SPEECH

side of a speech transmission system and speech synthesis (generation) follows on the receiver side (see Figure 3.13).

![Recognition/synthesis systems: components of a speech transmission system.](image)

Figure 3.13: Recognition/synthesis systems: components of a speech transmission system.

Only the characteristics of the speech elements are transmitted. For example, the speech elements with their characteristics are the formants with their middle frequency bandwidths. The frequency bandwidths are used in the corresponding digital filter. This reduction brings the data rate down to 50 bits/s. The quality of the reproduced speech and its recognition rate are not acceptable by today's standards.

- **Achieved Quality**

The essential question regarding speech and audio transmission with respect to multimedia systems is how to achieve the minimal data rate for a given quality. The published function from Flanagan [Fla72] (see Figure 3.14) shows the dependence of the achieved quality of compressed speech on the data rate. One can assume that for telephone quality, a data rate of 8 Kbits/s is sufficient. Figure 3.15 shows the dependence of audio quality on the number of bits per sample value. For example, excellent CD-quality can be achieved with a reduction from 16 bits per sample value to 2 bits per sample value. This means that only 1/8 of the actual data needs to be transmitted.
Figure 3.14: Dependence of the achieved speech quality on the data rate.

Figure 3.15: Dependence of audio quality on the number of bits per sample value.