To appear in: Wiley Encyclopedia of Computer Science and Engineering
Note: Accepted original version prior to copyediting and revision

Digital Sound Synthesis for Multimedia Audio

Eduardo R. Miranda

Interdisciplinary Centre for Computer Music Research

University of Plymouth

Introduction

Techniques for generating audio for multimedia applications can be grouped into two broad approaches, referred to in this article as *sampling manipulation* and *software synthesis*, respectively.

The most common approach to adding sounds to multimedia applications is to use recorded sounds. Indeed, there are various sound libraries and repositories available, both commercially and in the public domain, where sound designers may take sound samples to include in their developments. This practice works well for cases where sounds do not need to change dynamically in the application domain; e.g., in a movie. However, there are a number of applications where the sound needs to change dynamically. In these cases it might necessary to foster closer links between the process of generating sounds and the process of generating images; for example, in computer games where the sound is deemed to be an integral part of the story of the game. For such cases, *sampling manipulation* techniques take the sound design process a step further: they allow for software transformation of given recorded sounds. Sampling manipulation techniques can be of two types: *frequency domain* or *time domain*.

An even higher degree of sophistication is afforded by *software synthesis*, which includes those techniques for generating digital audio from scratch. A classic example is the Additive Synthesis technique, whereby a sound is constructed as the result adding the output of a number of sinusoidal oscillators working in parallel. In this case, each oscillator is tuned to a frequency component of the spectrum of the sound in question. Other examples of software synthesis techniques include Frequency Modulation, Amplitude Modulation, Waveshaping and Physical Modelling, to cite but four (Miranda, 2002).

This article begins with an introduction to the basics of digital representation of sound, including the notions of sampling and storage. Then, it presents the fundamentals of sound processing and analysis by means of the Short-term Fourier Transform method, followed by an introduction to sampling manipulation techniques. Next, it introduces the basics of Frequency Modulation, a powerful software synthesis technique that is relatively easy to implement and is capable of producing a variety of musical tones and unusual sound effects.

2 Digital Representation of Sound

Sound results from the mechanical disturbance of some object in a physical medium such as air. These mechanical disturbances generate vibrations that can be converted into an *analog signal* (time-varying voltage) by means of devices such as a microphone.

Analog signals are continuous in the sense that they consist of a continuum of values, as opposed to stepwise values. Digital computers, however, are not analog machines. They fundamentally deal only with *binary numbers*. In contrast to the decimal numeric system, which uses ten different symbols (i.e., from 0 to 9) to represent numbers, the binary system uses only two symbols: 0 and 1. Currently available computers are made of tiny electronic switches, each of which can be in one of two states at a time: on or off, represented by the digits '1' and '0', respectively. Consequently, the smallest unit of information that such computer can handle is the *bit*, a contraction of the term "binary digit". For example, the decimal numbers 0, 1, 2 and 3 are represented in the binary system as 0, 1, 10 and 11 respectively.

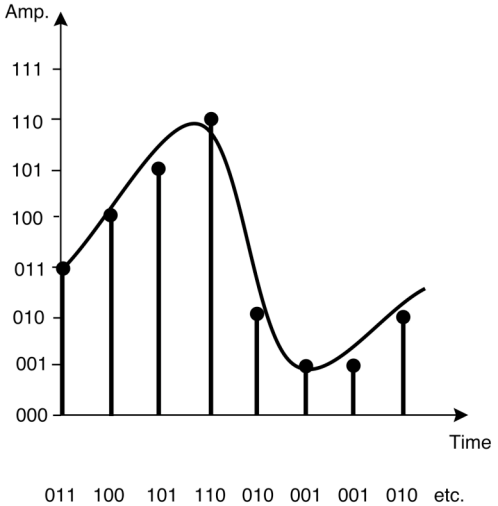
A digital computer is normally configured to function based upon strings of bits of fixed size, referred to as *words*. For example, a computer configured for 4-bit words would represent the decimal numbers 0, 1, 2 and 3 as 0000, 0001, 0010 and 0011, respectively. Note that the maximum number that four bits can represent is 1111, which is equal to 15 in the decimal system. In this case, a 4-bit computer seems to be extremely limited, but in fact, even a much larger word size would present this type of limitation. Most currently available computers use 32- or 64-bit words, but they represent numbers in a slightly different way: they consider each digit of a decimal number individually. Hence a 4-bit computer would use two separate words to represent the decimal number 15, one for the digit 1 and another for the digit 5: 15 = 0001 0101.

The binary representation discussed above is extremely useful because computers also need to represent things other than numbers. Manufacturers assign arbitrary codes for such symbols: for instance, the letter A = 10000001 and the letter B = 10000010. Whilst part of this codification is standard for most machines (e.g., the ASCII codification), a significant proportion is not, which leads to one of the causes of incompatibility between different systems.

In order to process sounds on the computer, the analog sound signal must be represented using binary numbers. Conversely, the digital signal must be converted into analog voltage in order to play a sound from the computer. The computer must therefore be provided with two types of data converters: analog-to-digital (ADC) and digital-to-analog (DAC).

The conversion is based upon the concept of *sampling*. Sampling functions by measuring the voltage (that is, the amplitude) of a continuous signal at intervals of equal duration. Each measurement value is called a *sample* and is recorded in binary format. The result of the whole sampling process is a sequence of binary numbers corresponding to the voltages at successive time lapses (Figure 1).

Figure 1: Sampling functions by measuring the amplitude of an analog signal at intervals of equal duration. Measurements are rounded to fit the numbers it can process.



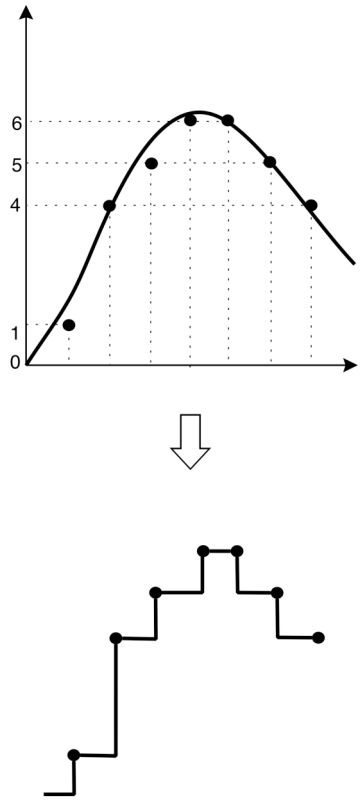
Audio samples can be stored on any digital medium, such as tape, disk or computer memory, using any recording technology available, such as electromagnetic and optic technology. The advantage of digital sound representation over analog representation is that the former allows for computer manipulation and generation of streams of samples in a variety of ways.

2.1 The Sampling Theorem: Quantisation and Aliasing Distortion

The number of times a signal is sampled in each second is called the *sampling rate* (or sampling frequency) and it is measured in Hertz (Hz).

The sampling theorem states that in order to accurately represent a sound digitally, the sampling rate must be higher than at least twice the value of the highest frequency contained in the signal (Oppenheim and Willsky, 1996). The faster the sampling rate, the higher the frequency that can be represented, but the greater the demands for computer memory and power. The average upper limit of human hearing is approximately 18 kHz, which implies a minimum sampling rate of 36 kHz (i.e., 36 000 samples per second) for high fidelity. The sampling rate frequently recommended for multimedia audio is 44 100 Hz.

Figure 2: The sampling process rounds off the measurements according to the resolution of the converter. The higher the resolution, the better the accuracy of the digitised signal.

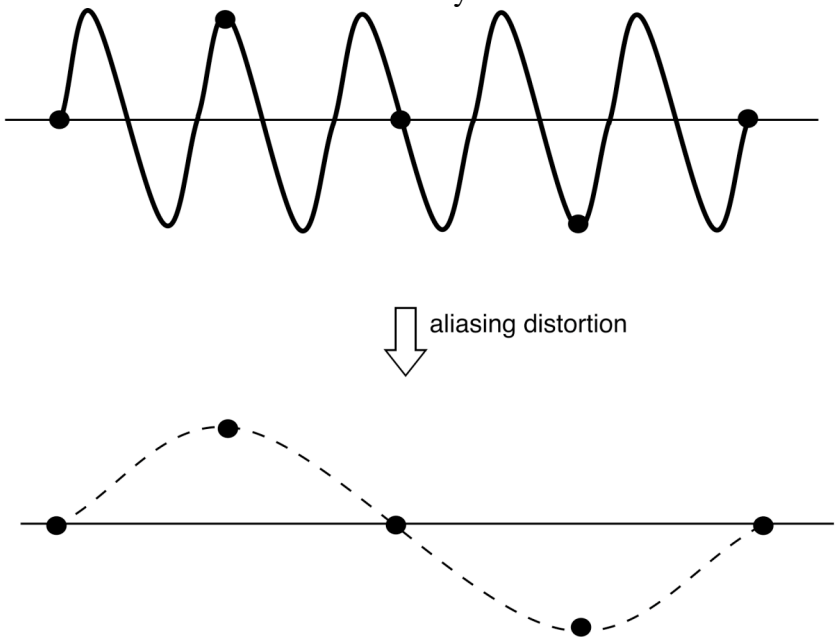


The amplitude of a digital signal is represented according to the scale of a limited range of different values. This range is determined by the resolution of the ADC and DAC. The resolution of the converters depends upon the size of the word used to represent each sample. For instance, whereas a system with 4-bit resolution would have only 16 different values (2^4) to measure the amplitudes of the samples, a system with 16-bit resolution would have 65 536 different values (2^{16}). The higher the resolution of the

converters, the better the quality of the digitised sound. The sampling process normally rounds off the measurements in order to fit the numbers that the converters can deal with (Figure 2). Unsatisfactory lower resolutions are prone to cause a damaging loss of sound quality, referred to as the *quantisation noise*. The ADC needs at least two samples per waveform cycle in order to represent the frequency of the sound. Significant frequency information might be lost otherwise. Digital recording systems place a low-pass filter before the ADC in order to ensure that only signals below the Nyquist frequency enter the converter. This is so because the conversion process can create foldback frequencies that cause a phenomenon known as *aliasing distortion* (Figure 3).

Nyquist frequency is the name of the highest frequency that can be represented in a digital audio system. It is calculated as half of the value of the sampling rate. For instance, if the sampling rate of the system is equal to 44 100 Hz, then the Nyquist frequency is equal to 22 050 Hz.

Figure 3: The resolution of the ADC needs to be at least two samples per cycle, otherwise aliasing distortion may occur.



2.2 Pulse-Code Modulation (PCM) and Pulse-Density Modulation (PDM)

Standard audio sampling uses Pulse Code Modulation (PCM) (Waggener, 1999). Starting in the late seventies with 14-bit words, then moving up to 16-bit, 24-bit words, etc. PCM today is capable of high quality digital audio coding. There is, however, increasingly little room from improvement due to a number of cumbersome technical issues. In short, PCM requires decimation filters at the sampling end, and interpolation filters at the playback end of the modulation process. These add unavoidable quantisation noise to the signal.

Pulse Density Modulation (PDM) eliminates the decimation and interpolation altogether: it records the pulses directly as a 1-bit signal (Sinclair, 2000). The analog-to-digital converter uses a negative feedback loop to accumulate the sound. If the input accumulated over one sampling period, rises above the value accumulated in the negative feedback loop during previous samples, then the converter outputs 1. Conversely, if the sound falls relative to the accumulated value, then the converter outputs 0. As a result, full positive waveforms will all be 1s and full negative waveforms will all be 0s. The crossing point will be represented by alternating 1s and 0s. The amplitude of the original analog signal is represented by the density of pulses.

At first sight, a sampling rate of 2 822 400 Hz may seem to require prohibitive storage capacity and computer power. But this should not be the case. A standard stereo CD uses 16-bit word samples, thus the bit rate per channel here is 16 times 44.1 kHz, that is 705 600 bits per second. As PDM uses 1-bit per sample, the bit rate of 2 8224 MHz is only about four times higher than for a standard CD.

The result of PDM gives an unparalleled impression of depth and fluidity that comes from a far greater frequency response and dynamic range. PDM captures harmonics inaudible as pure tones by accurately reproducing the leading edges of transients, which enriches the sound spectrum and enhances the listening experience.

2.3 Sound Storage: File Formats and Compression

Digital audio may be stored on a computer in a variety of formats. Different systems use different formats, which define how samples and other related information are organised in a file. The most basic way to store a sound is to take a stream of samples and save them into a file. This method, however, is not flexible because it does not allow for the storage of information other than the raw samples themselves; for example, the sampling rate used, the size of the word or whether the sound is mono or stereo. In order to alleviate this problem, sound files normally include a descriptive data structure, referred to as the sound file header. Some sound file headers allow for the inclusion of text comments and cue pointers in the sound. Most used sound file formats are WAVE (.wav) and AIFF (.aif)

A major disadvantage of raw sound file storage is that it is uneconomical, as it might contain a great deal of redundant information that would not need high sampling rate accuracy for representation. As a rough illustration, imagine a mono sound file, with a sampling rate of 44.1 kHz and 16-bit words, containing a sound sequence with recurrent three minute-long silences separating the sounds. In this case, 2 116 899 bits would be required to represent each of these silent portions. A compression scheme could be devised in order to replace the raw sampling representation of the silent portions with a code that instructs the playback device to produce three seconds of silence.

There are a number of methods for compressing the representation of sound samples in order to reduce the size of the file. One of the most popularly known methods in use today is MPEG3 (short for Moving Picture Experts Group layer 3), popularly known as MP3 (Ruckert, 2005). Originated by Internet Standards Organization (ISO), MPEG3 works by eliminating sound components that would not normally be audible to humans. MPEG3 can compress a sound file considerably without much noticeable difference.

Other well-known compression schemes include Real Audio (by Progressive Networks), ATRAC3 (by Sony) and WMA (by Microsoft). RealAudio is acknowledged as the first compression format to support live audio over the Internet. ATRAC3 is a sound compression technology based on ATRAC, the technology that Sony devised originally for the MiniDisc.

3 Sound Processing Technology Background

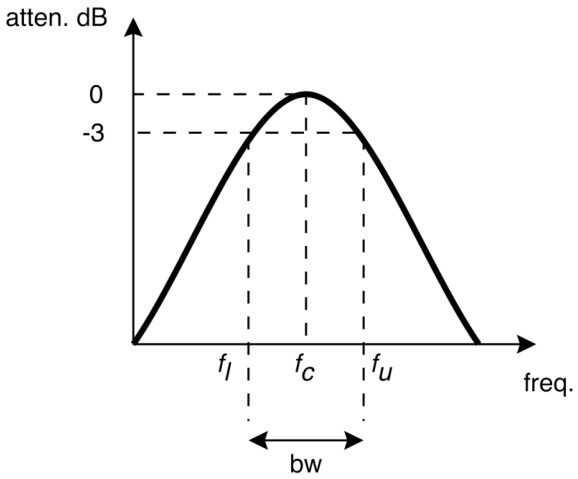
3.1 Brief Introduction to Filters

In general, a filter is any device that performs some sort of transformation on a signal. For simplicity, however, we refer only to those filters that remove or favour the resonance of specific components of the spectrum of a signal, namely: low-pass (LPF), high-pass (HPF), band-pass (BPF) and band-reject (BRF).

The BPF, also known as the resonator, rejects both high and low frequencies with a passband in between. Two parameters are used to specify the characteristics of a BPF: passband centre frequency (represented as f_c) and resonance bandwidth (represented as *bw*). The *bw* parameter comprises the difference between the upper (represented as f_u) and lower (represented as f_l) cut-off frequencies (Figure 4).

The BRF amplitude response is the inverse of a BPF. It attenuates a single band of frequencies and discounts all others. Like a BPF, it is characterised by a central frequency and a bandwidth; but another important parameter is the amount of attenuation in the centre of the stopband.

Figure 4: The pass-band filter response.



An LPF permits frequencies below the point called the cut-off frequency to pass with little change. However, it reduces the amplitude of spectral components above the cut-off frequency. Conversely, an HPF has a passband above the cut-off frequency where signals are passed and a stopband below the cut-off frequency, where the signals are attenuated. There is always a smooth transition between passband and stopband. It is often defined as the frequency at which the power transmitted by the filter drops to one half (about –3 dB) of the maximum power transmitted in the passband.

Some applications may require a composition of interconnected filters in order to produce a desired effect. There are two basic combinations for filter composition: *parallel connection* and *serial connection*. In a serial connection, filters are connected like the links of a chain. The output of one filter feeds the input of the next, and so on. With parallel connection, the signal to be filtered is simultaneously applied to all filter inputs. The output is the addition of the frequency responses of all filters, resulting in the resonation of any frequency found within the passband of the individual filters.

3.2 Short-Time Fourier Transform

In the eighteenth century, Jean Baptiste Joseph Fourier proposed that complex vibrations could be analysed as a set of parallel sinusoidal frequencies, separated by a fixed integer ratio; for example, 1*x*, 2*x*, 3*x*, etc., where *x* is the lowest frequency of the set. Hermann Ludwig Ferdinand von Helmholtz then developed Fourier's analysis for the realm of musical acoustics (Helmholtz, 1885). Basically, the differences are perceived because the loudness of the individual components of the harmonic series differs from timbre to timbre.

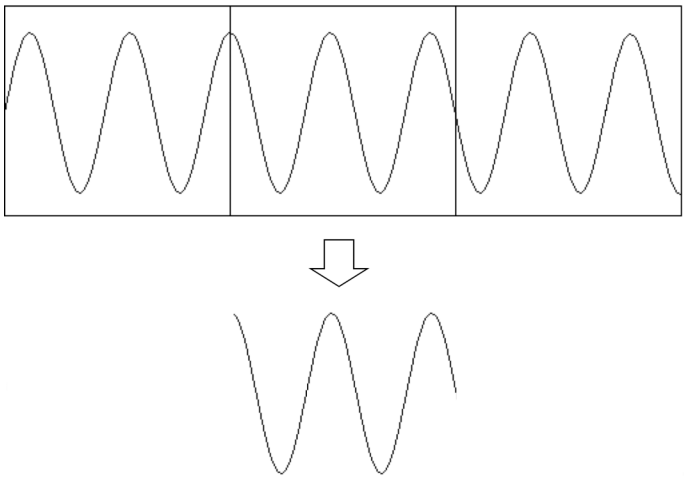
Spectrum analysis is fundamentally important for spectral manipulation because samples alone do not inform the spectral constituents of a sampled sound. In order to manipulate the spectrum of sounds, we

need adequate means to dissect, interpret and represent them. In a nutshell, spectrum analysis is aimed at the identification of the frequencies and amplitudes of the spectrum components. *Short-time Fourier Transform* (STFT) is a widely used spectral analysis technique (Allen and Mills, 2004). STFT stands for an adaptation, suitable for computer implementation, of the original Fourier analysis for calculating harmonic spectra.

One of the main problems with the original Fourier transform theory is that it does not take into account that the components of a sound spectrum vary substantially during its course. The result of the analysis of a sound of, for example, five minutes duration would inform the various components of the spectrum but would not inform when and how they developed in time. The analysis of a sound that changes from an initial timbre to another during its course would only display the existence of the components that form both types of timbres, as if they were heard simultaneously. STFT implements a solution for this problem. It chops the sound into short segments called windows and analyses each segment sequentially. It uses the Fourier transform to analyse these windows, and plots the analysis of the individual windows in sequence in order to trace the time evolution of the sound in question. The result of each window analysis is called a frame. Each frame contains two types of information: a *magnitude spectrum* depicting the amplitudes of every analysed component and a *phase spectrum* showing the initial phase for every frequency component.

In the context of STFT, the process whereby shorter portions of a sampled sound are detached for FFT analysis is referred to as windowing (Figure 5). The windowing process must be sequential, but the windows may overlap. The effectiveness of the STFT process depends upon the specification of three windowing factors: the envelope for the window, the size of the window and the overlapping factor.

Figure 5: The sound is windowed for FFT analysis



Note in Figure 5 that the windowing process may cut the sound at non-zero parts of the waveform. The Fourier transform algorithm considers each window as a unit similar to a wavecycle. The problem with this is that interruptions between the ends of the windowed portion lead to irregularities in the analysis. This problem can be remedied by using a lobe envelope to smooth both sides of the window. From the various functions that generate lobe envelopes, the Gaussian, the Hamming and the Kaiser functions are more often used because they tend to produce good results.

The size of the window defines the frequency and the time resolutions of the analysis. This value is normally specified as a power of two; e.g. 256, 512, 1024, and so on. Longer windows have better frequency resolution than smaller ones, but the latter have better time resolution than the former. For example, whilst a window of 1024 samples at a rate of 44 100 Hz allows for a time resolution of

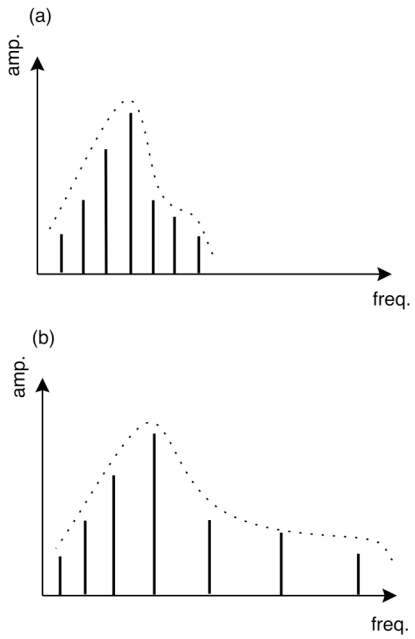
approximately 23 milliseconds (1024/44 100 = 0.023), a window of 256 samples gives a much better resolution of approximately 6 milliseconds (256/44 100 = 0.0058). Conversely, the Fourier analysis will be tuned to scan frequencies spaced by a bandwidth of approximately 43 Hz (44 100/1024 = 43) in the former case and to approximately 172 Hz (44 100/256 = 172) in the latter. This means that a window of 256 samples is not suitable for the analysis of sounds lower than 172 Hz, but it may suit the analysis of a sound that is likely to present important fluctuation within less than 23 milliseconds.

In order to find out precisely when an event occurs, the Fourier analysis algorithm cuts down the frequency resolution and vice versa. The overlapping of successive windows can alleviate this problem. For example, if an overlap factor is set to equal 16 (i.e., 1/16th of the size of the window) and the window size is set to equal 1024 samples, then the windowing process will slice the sound in steps of 64 samples (i.e., 1024/16 = 64). In this case the time resolution of a window of 1024 samples would improve from 23 milliseconds to approximately 1.4 milliseconds (i.e., 0.023/16 = 0.0014).

4 Sampling Manipulation in the Frequency Domain: Spectral Manipulations and Resynthesis

Most spectral manipulation techniques work in two stages, analysis and resynthesis, which can be performed in a number of ways. Whereas the analysis stage extracts spectral coefficient from a given sound, the resynthesis stage uses these coefficient to recreate the sound using a suitable synthesis technique, such as overlap-add, additive and spectral modelling synthesis (SMS); these are introduced below. The ability to create a precise imitation of the original sound is not, however, always interesting for multimedia applications. Here one would probably prefer to modify the analysis data in order produce variations or modifications of a given sound. Typical modifications include spectral time stretching, spectral shift and spectral stretching (Figure 6).

Figure 6: Spectral stretching.



4.1 Overlap-Add Synthesis

Overlap-add synthesis takes the information from each frame of the STFT analysis and recreates an approximation of the original sound window by window (Miranda, 2002). This process works well for those cases where the windows are reconstructed in conformity with the original analysis specification

(e.g., use of similar windowing and overlapping). The analysis data must be cautiously manipulated in order to produce good results, otherwise the integrity of the signal may be completely destroyed. This may, of course, be desirable for some specific purposes, but in most circumstances such spurious effects are difficult to control.

4.2 Additive Re-Synthesis

Additive re-synthesis employs STFT analysis data to control the oscillators of an additive synthesiser. In this case, the analysis stage includes an algorithm that converts the STFT data into amplitude and frequency functions – or envelopes – that span across the STFT frames. This is generally referred to as the *peak-tracking algorithm* and can be implemented in a number of ways. Additive re-synthesis is more flexible for parametric transformation than the overlap-add technique discussed above, because envelope data are generally straightforward to manipulate. Transformations such as stretching, shrinking, rescaling and shifting can be successfully applied to either or both frequency and time envelopes. Additive resynthesis is suitable for dealing with harmonic sounds with smoothly changing, or almost static, components, such as vowel-like sounds. Noisy or non-harmonic sounds with highly dynamic spectra are more difficult to handle here.

4.3 Spectral Modelling Synthesis (SMS)

The SMS technique (Serra and Smith, 1990) attempts to improve additive resynthesis by adding a mechanism that reintegrates the non-sinusoidal components that are discarded (or converted into stationary vibrations during) during the STFT analysis. The outcome of the analysis provides two types of information: one about the sinusoidal components of the sound and another about non-sinusoidal (or "noisy") components. Therefore, the re-synthesis process results from two simultaneous synthesis processes: one for sinusoidal components and the other for the non-sinusoidal components of the sound.

Each frame of the STFT is resynthesised immediately after the analysis, using the additive resynthesis method described above. This signal is then subtracted from the original in order to obtain the residual non-sinusoidal components, which is represented as an envelope for a time-varying filter.

The sinusoidal components are synthesised by generating sinewaves using the amplitude and frequency trajectories of the harmonic analysis (as with additive re-synthesis). The non-sinusoidal components are synthesised by filtering a white noise signal. The envelope produced during the analysis controls the filter.

5 Sampling Manipulation in the Time-Domain

5.1 Granular Sampling

In order to better understand the concept of granular sampling it is advisable to comprehend the concept of *sound grains* and their relationship to our auditory capacity. This notion is largely based upon a sound representation method published in 1947, in a paper by the Nobel Prize physicist Dennis Gabor.

Although it is taken for granted now that the components of a sound spectrum vary substantially during the sound emission, these variations were not considered relevant enough to be represented until recently. In his paper, Gabor proposed the basis for a representation method, which combines frequency-domain and time-domain information. His point of departure was to acknowledge the fact that the ear has a time threshold for discerning sound properties. Below this threshold, different sounds are heard as clicks, no

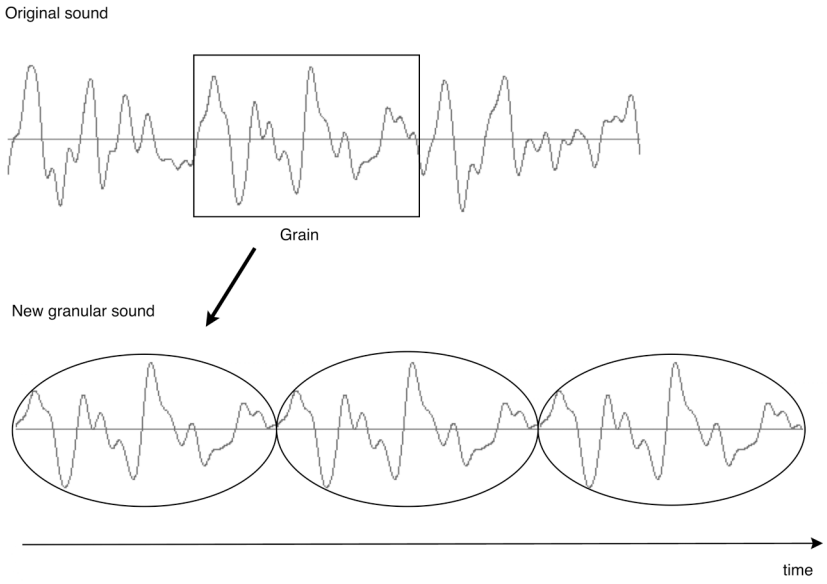
matter how different their spectra might be. The length and shape of a wavecycle define frequency and spectrum properties, but the ear needs several cycles to discern these properties. Gabor called this minimum sound length *acoustic quanta* and estimated that it usually falls between 10 and 30 milliseconds, according to the nature of both the sound and the subject.

Gabor's theory fundamentally suggested the STFT technique introduced earlier. It suggested that a more accurate sound analysis could be obtained by slicing the sound into very short segments, and by applying a Fourier-type of analysis to each of these segments. A sequence of frequency-domain representation snapshots, one for each segment, would then show – like the frames of a film – how the spectrum evolves with time.

Granular sampling generates sequences of short sound segments - or "sound grains". It employs a *granulator* mechanism that extracts small portions from a sampled sound and applies an envelope to them. The granulator may produce the grains in a number of ways. The simplest method is to extract only a single grain and replicate it many times (Figure 7). More complex methods involve the extraction of grains from various portions of the sample. In this case, the position of the extraction can be either randomly defined or controlled by an algorithm. Interesting results can be obtained by extracting grains from more than one sound source. Other common operations on "granulated" sounds include: reversing the order of the grains, time expansion (by changing the time interval between the grains) and time compression (by truncating grains).

The envelope is of critical importance in granular sampling because it prevents the glitches that would otherwise be produced by possible phase discontinuities between the grains. The shape of the envelope should be carefully chosen because unwanted distortions may be produced by unavoidable amplitude modulation effects. Also, envelopes that use linear segments with angled joints should be used with caution, if at all, as they often produce "undesirable" spectral components. Sometimes such undesirable effects may be useful in order to add a certain colouration to the resulting sound, but they should be avoided as a rule.

Figure 7: Granular sampling works by extracting grains from a given sound.



5.2 Waveset Distortion

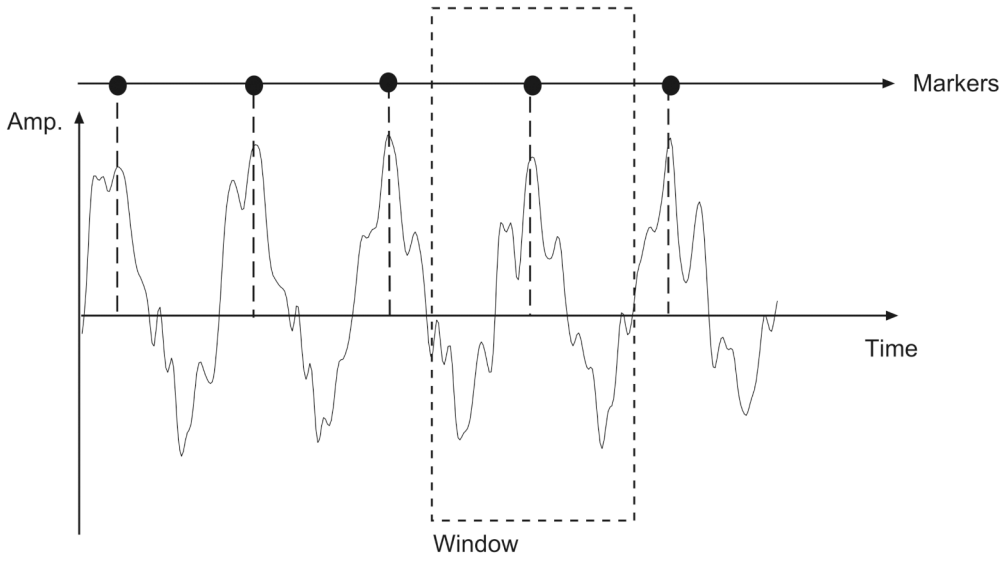
Waveset distortion involves the transformation of a sound at its wavecycle level; that is, the unit for waveset distortion is a set of samples between two successive zero crossings (Wishart, 1994). For example, a simple sinusoid sound is viewed in this context as a succession of wavecycles, and not as a succession of samples.

Waveset distortion is akin to granular sampling in many respects. The main difference is that radical changes are applied to the individual wavecycles, rather than re-arranging either the order or location of grains. Also, there is an explicit unit definition to work with here: the wavecycle.

5.3 Pitch Synchronous Overlap and Add (PSOLA)

PSOLA, an acronym for Pitch Synchronous Overlap and Add, was originally designed for speech synthesis (Ben and Morgan, 2000). In short, it works by concatenating small segments with an overlapping factor. The duration of these segments is proportional to the pitch period of the resulting signal. PSOLA is interesting because a given signal can be decomposed in terms of these elementary segments and then re-synthesised by streaming these elements sequentially (Figure 8). Parametrical transformation can be applied during the streaming of the segments in order to modify the pitch and/or the duration of the sound. PSOLA is particularly efficient for shifting the pitch of a sound, as it does not, in principle, change its spectral contour. PSOLA closely resembles waveset distortion dicussed earlier, with the difference that the segmentation is much more constrained here.

Figure 8: PSOLA works by decomposing the signal into small overlapping segments, called markers, using a window whose middle point is the segment's point of maximum energy. The sound is then resynthesised by streaming these segments. Change to the sound can be made by changing the original position of the markers.



At the analysis stage, the elementary segments are extracted using a window centred at the local maxima, called markers. These markers should be pitch-synchronous in the sense that they should lie close to the fundamental frequency of the sound. The size of the window is proportional to the local pitch period. The

pitch of the signal can be changed during re-synthesis by changing the distances between the successive markers. Time-stretching or time-compression is achieved by repeating or skipping segments.

The caveat of the PSOLA analysis and resynthesis method is that it only works satisfactorily on sounds containing regular cycles displaying a single energy point, referred to as the local maxima; the human voice fulfils this requirement, but not all sounds do.

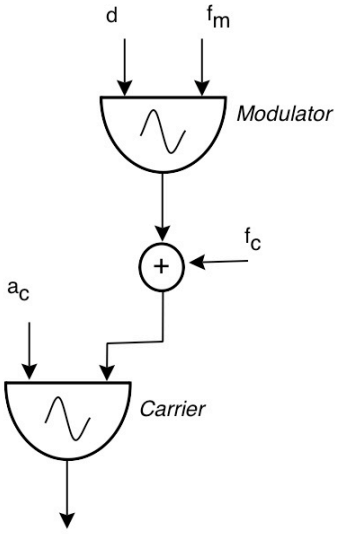
6 Software Sound Synthesis: Frequency Modulation

Frequency Modulation, of FM, was invented in the late 1960s by the composer John Chowning (1973) at the University of Stanford. There are a number of variations in FM synthesizer design. The most basic design comprises two oscillators, called *modulator* and *carrier* oscillators, respectively (Figure 9). This simple architecture is capable of producing a surprisingly rich range of distinctive timbres. More complex FM synthesizers may employ various modulators and carriers, combined in a number of ways. For didactic purposes we classify the various approaches to FM synthesizer design into two categories: *simple FM* and *composite FM*.

6.1 Simple FM

In simple FM (Figure 9) the output of the modulator is offset by a constant, represented as f_c , and the result is then applied to control the frequency of the carrier. If the amplitude of the modulator is equal to zero, then there is no modulation. In this case, the output from the carrier will be a simple sinewave at frequency f_c . Conversely, if the amplitude of the modulator is greater than zero, then modulation occurs and the output from the carrier will be a signal whose frequency deviates proportionally to the amplitude of the modulator. In the context of FM synthesis, the amplitude of the modulator is referred to as *frequency deviation* (represented as *d*), and its value is usually expressed in Hz.

Figure 9: The basic FM synthesizer consists of two oscillators, where the output of one modulates the frequency of the other.



The parameters for the FM synthesiser portrayed in Figure 9 are as follows:

d = frequency deviation
f_m = modulator frequency

a_c = carrier amplitude
f_c = offset carrier frequency

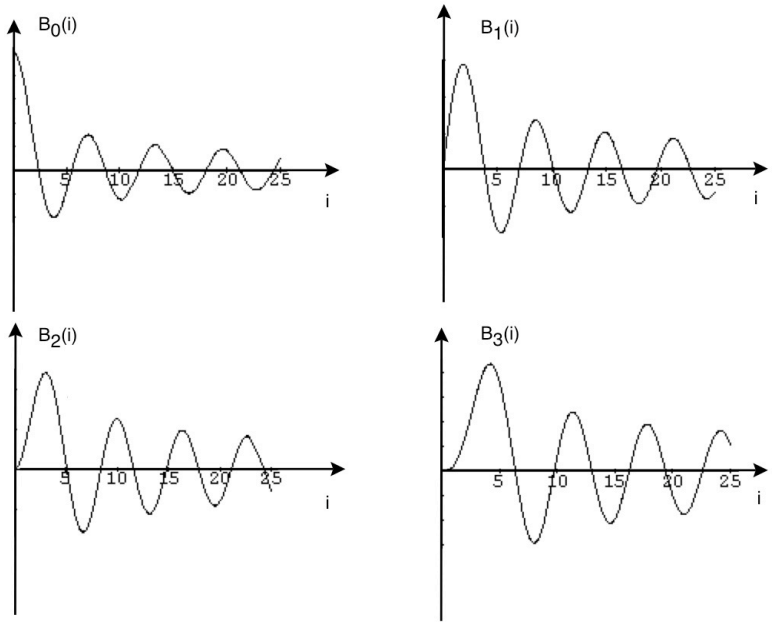
The parameters d and f_m can drastically change the form of the carrier wave. If the f_m is kept constant whilst increasing d, then the period of the carrier's output will increasingly expand and contract, proportional to d. If d remains constant and f_m is increased, then the rate of the deviation will become faster.

The simplicity of its architecture and its capability to produce a great variety of different timbres made FM synthesis more attractive than other techniques available at the time of its invention. Hence the great interest of Yamaha to exploit this technique for the manufacturing of MIDI synthesisers in the early 1980s, such as the legendary and commercially successful DX7 keyboard synthesizer (It seems, however, that Yamaha used a variant of the original FM technique, referred to as Phase Modulation).

The spectrum of an FM sound is composed of the f_c and a number of partials on either side of it, spaced at a distance equal to f_m. The partials generated on each side of the carrier frequency are usually called sidebands. The sideband pairs are calculated as follows: $f_c + k \times f_m$ and $f_c - k \times f_m$ where k is an integer, greater than zero, which corresponds to the order of the partial counting from f_c. The amplitudes of the components of the spectrum are determined by a set of functions known as *Bessel functions* and represented as $B_n(i)$. A basic introduction how the Bessel functions determine the amplitude of the partials of an FM-generated sound is given below; for more details, please consult the book by Chowning and Bristow (1987).

Figure 10 shows the graphical representation of four Bessel functions: $B_0(i)$, $B_1(i)$, $B_2(i)$ and $B_3(i)$, respectively. They determine scaling factors for the amplitude of pairs of sidebands, according to their position relative to the offset carrier's frequency. The a_c usually defines the overall loudness of the sound, but the amplitudes for individual partials are calculated by scaling the given carrier amplitude according to the factors established by the Bessel functions. For instance, $B_0(i)$ determines the scaling for f_c, $B_1(i)$ the scaling for the first pair of sidebands ($k = 1$), $B_2(i)$ the scaling for the second pair of sidebands ($k = 2$), etc.

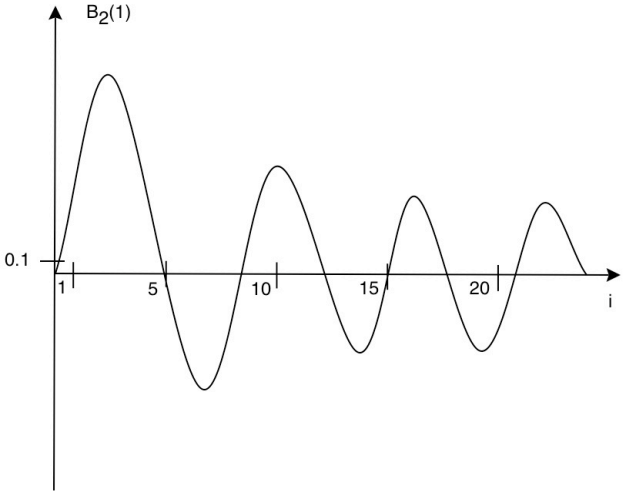
Figure 10: Bessel functions determine the amplitude scaling factors for pairs of sidebands.



The vertical axis indicates the amplitude scaling factor according to the value of the modulation index represented by the horizontal axis. For example, when $i = 0$ the offset carrier frequency will sound at its maximum factor and the amplitudes of all sidebands will be zero. If $i = 1$, then the scaling factor for f_c

will decrease to approximately 0.76, the factor for the first pair of sidebands will have a value of 0.44, the second pair will have a value of 0.11, etc. (Figure 11). The value of the modulation index must be large in order to obtain significant amplitudes in high-order sidebands.

Figure 11: The vertical axis indicates the amplitude scaling in function of the value of the modulation index.



One important rule to bear in mind when calculating an FM spectrum is that the scaling factors for the odd partials on the left of the sideband pair are multiplied by –1. In addition, notice that Bessel functions indicate that sidebands may have either positive or negative values, depending on the modulation index. For example, if $i = 5$, the scaling factor for the first pair of sidebands will be approximately –0.33. The negative signal here indicates that the sidebands are out of phase. This may be represented graphically by plotting it downwards on the frequency axis. In this case, the amplitudes of the conflicting partials will either add or subtract, depending on their respective phases. If the offset carrier frequency is very low and the modulation index is set high, then the modulation may produce sidebands that fall in the negative frequency domain of the spectrum. As a rule, all negative sidebands will fold around the 0 Hz axis and mix with the sidebands in the positive domain. Reflected partials from the negative domain will, however, reverse their phase. For instance, the settings $f_c = 440$ Hz, $f_m = 440$ Hz and $i = 3$, produce the following negative sidebands: –1320 Hz, –880 Hz, and –440 Hz. These partials fold into the positive domain, with reversed phase, and add algebraically to the partials of the same values that were there before the reflection. Another phenomenon that is worth remembering here is that partials falling beyond the Nyquist frequency (introduced earlier) fold over and reflect into the lower portion of the spectrum.

The ability to provide control for time-varying spectral components of a sound is of critical importance for sound synthesis. The amplitudes of the partials produced by most acoustic instruments vary through their duration: they often evolve in complicated ways, particularly during the attack of the sound. The *modulation index* is an effective parameter to control spectral evolution. As the modulation index defines the number of partials in the spectrum, time-varying functions can be employed to vary i in order to produce varying spectra, according to the slope of its Bessel function at specific modulation values.

6.2 Composite FM

Composite FM involves two or more carrier oscillators and/or two or more modulator oscillators. There are a number of possible combinations and each of them will create different types of spectral compositions. Complex FM produces more sidebands but the complexity of the calculations to predict the

spectrum also increases. There are at least five basic combinatory schemes for building composite FM synthesizers:

- Additive carriers with independent modulators
- Additive carriers with one modulator
- Single carrier with parallel modulators
- Single carrier with serial modulators
- Self-modulating carrier

The "additive carriers with independent modulators" scheme is composed of two or more simple FM synthesizers working in parallel. The spectrum is therefore the result of the addition of the outputs from each synthesizer. The "additive carriers with one modulator" scheme employs one modulator oscillator to modulate two or more carrier oscillators. The resulting spectrum is the result of the addition of the outputs from each carrier oscillator. The "single carrier with parallel modulators" scheme employs a more complex signal to modulate a carrier oscillator: the result of two or more sinewaves added together. The formula for the calculation of a simple FM spectrum is expanded in order to accommodate multiple modulator frequencies and modulation indices. For example, in the case of two parallel modulator oscillators, the sideband pairs are calculated as follows:

$$f_c - (k_1 \times f_{m1}) + (k_2 \times f_{m2})$$
$$f_c - (k_1 \times f_{m1}) - (k_2 \times f_{m2})$$
$$f_c + (k_1 \times f_{m1}) + (k_2 \times f_{m2})$$
$$f_c + (k_1 \times f_{m1}) - (k_2 \times f_{m2})$$

Each of the partials produced by one modulator oscillator forges a local carrier for the other modulator oscillator. The larger the number of parallel modulators, the greater the amount of nested local carriers. The amplitude scaling factors here result from the multiplication of the respective Bessel functions: $B_n(i_1) \times B_m(i_2)$.

The "single carrier with serial modulators" scheme also employs a complex signal to modulate a carrier oscillator. But in this case the modulating signal is a frequency-modulated signal. The sideband frequencies are calculated using the same method used above for parallel modulators, but the calculation of the amplitude scaling factors is different. The order of the outermost modulator is used to scale the modulation index of the next modulator.

Finally, the "self-modulating carrier" scheme employs the output of a single oscillator to modulate its own frequency. The oscillator output signal is multiplied by a feedback factor (represented as f_b) and added to a frequency value (f_m) before it is fed back into its own frequency input; f_b may be considered here as a sort of modulation index. This scheme is sometimes preferable to a simple FM synthesizer. The problem with simple FM is that the amplitudes of its partials vary according to the Bessel functions, but this variation is not linear. The number of sidebands increases by augmenting the modulation index, but their amplitudes do not rise linearly. This grants an unnatural colouration to the sound which may not always be desirable. The amplitudes of the partials produced by a self-modulating oscillator increase more linearly according to f_b.

6.3 Frequency ratios and sound design

Basically, timbre control in FM is governed by two simple ratios between FM parameters. One is the ratio between the frequency deviation and the modulator frequency and has already been introduced: it defines the modulation index. The other is the ratio between the offset carrier frequency and the modulator frequency, called frequency ratio and represented as $f_c{:}f_m$. The frequency ratio is a useful tool for the

implementation of a phenomenon that is very common among acoustic instruments, that is, achieving variations in pitch whilst maintaining the timbre virtually unchanged. If the frequency ratio and the modulation index of a simple FM synthesizer are maintained constant but the offset carrier frequency is modified then the sounds will vary in pitch, but their timbre will remain unchanged. In this case, it is much easier to think in terms of frequency ratios rather than in terms of values for f_c and f_m separately. Basic rules of thumb for FM sound design in terms of these simpler ratios are given as follows:

- Rule 1: if f_c is equal to any integer and f_m is equal to 1, 2, 3 or 4, then the resulting timbre will have a distinctive pitch, because the offset carrier frequency will always be prominent.

- Rule 2: if f_c is equal to any integer and f_m is equal to any integer higher than 4, then the modulation produces harmonic partials but the fundamental may not be prominent.

- Rule 3: if f_c is equal to any integer and f_m is equal to 1, then the modulation produces a spectrum composed of harmonic partials; e.g., the ratio 1:1 produces a sawtooth-like wave.

- Rule 4: if f_c is equal to any integer and f_m is equal to any even number, then the modulation produces a spectrum with some combination of odd harmonic partials; e.g., the ratio 2:1 produces a square-like wave.

- Rule 5: if f_c is equal to any integer and f_m is equal to 3, then every third harmonic partial of the spectrum will be missing; e.g., the ratio 3:1 produces narrow pulse-like waves.

- Rule 6: if f_c is equal to any integer and f_m is not equal to an integer, then the modulation produces non-harmonic partials; e.g., 2:1.29 produces a "metallic" bell sound

7 Concluding Remarks

This article introduced two broad approaches to digital sound synthesis for multimedia audio applications - namely *sampling manipulation* and *software synthesis* - and showed representative examples of sound synthesis techniques. There are many more techniques out there. Only the tip of the iceberg has been introduced in this article. A discussion on how to apply these techniques in multimedia projects is beyond the scope of this article. However, it must be said that the implementation of such techniques are very much dependent on the software being used to develop the multimedia systems in question. Some developers may prefer to implement their systems using a single package, which may or may not support the techniques introduced here. Conversely, there are a number of stand-alone programming tools for implementing sound synthesis. For instance, there is the CDP sound transformation software, which to the best of this author's knowledge, is the most comprehensive package available for sampling manipulation (http://www.composersdesktop.com/ - last visited 25 January 2008). For implementation of software synthesis techniques there is Csound, which is a powerful and well-documented programming language for audio (http://csounds.com/ - last visited 25 January 2008).

In addition to this author's own book *Computer Sound Design: Synthesis Techniques and Programming* (Focal Press, 2002) and other publications listed in the references, the following books are suggested for complementary information: *The Csound Book* (edited by R. Boulanger, MIT Press, 2000), *Real Sound Synthesis for Interactive Applications* (by P. Cook, AK Peters, 2002), *A Programmer's Guide to Sound* (by T. Kientzle, Addison-Wesley Professional, 1997), *Computer Music: Synthesis, Composition and Performance* (by C. Dodge and T. Jerse, Schirmer, 1997), *On Sonic Art* (by T. Wishart, Routledge, 1996), and *Digital Audio Signal Processing* (by U. Zolzer, Willey, 1997).

8 References

Allen, R. L. and Mills, D. (2004). *Signal Analysis: Time, Frequency, Scale and Structure*. John Wiley and Sons.

Ben, G. and Morgan, N. (2000). *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley and Sons.

Chowning, J. (1973). "The Synthesis of Complex Audio Spectra by means of Frequency Modulation". *Journal of the Acoustical Society of America*, 21(7):526-534.

Chowning, J. and Bristow, D. (1987). *FM Theory and Applications: By Musicians for Musicians*. Hal Leonard Corp.

Miranda, E. R. (2002). *Computer Sound Design: Synthesis Techniques and Programming*. Elsevier/Focal Press.

Oppenheim, A. V. and Willsky, A. S. (1996), 2nd edition. *Signals and Systems*. Prentice Hall.

Ruckert, M. (2005). *Understanding MP3: Syntax, Semantics, Mathematics and Algorithms*. Springer-Verlag.

Serra, X. and Smith, J. O. (1990). "Spectral Modeling Synthesis:A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition". *Computer Music Journal*, 14(4):12-24.

Sinclair, I. (Ed.) (2000). *Audio and Hi-fi Handbook*, Newnes.

Waggener, W. N. (1999). *Pulse Code Modulation Systems Design*. Artech House Publishers.

Wishart, T. (1994). *Audible Design*. Orpheus the Pantomime.