

Chapter

1

Introduction

‘Some digital techniques are little more than precise versions of previous mechanical and analog processes. But precision is not the most important attribute of the computer. What the computer offers the composer is programmability – the extension of functionality in any direction.’

Curtis Roads [Roads, 1985a, p. xiii]

OBJECTIVES

This chapter covers:

- the definition of computer music, how computer music relates to music in general, and why we should pursue it;
- a quickstart guide and a summary of the important concepts;
- the history of computer music.

1.1 WHAT IS COMPUTER MUSIC?

Computer music could be most generally defined as *music that involves a computer at any stage of its life cycle*. This encompasses a very wide variety of musical activities, even if ‘unplugged’ acoustic music and electric (e.g., amplified) music without microprocessors are excluded. We are not yet at the stage where singing to yourself in the shower is automatically

accompanied by computers, though this may well change in an era of smart homes. We are, however, in a digital era¹ where as soon as music is recorded computers tend to become intimately employed in the service of sound, and music is also produced directly from computer music software. The reader is no doubt already familiar with many cases of computers in music, whether using a virtual recording studio on a laptop, running a score-editing program, or using the massed ranks of computers making up the Internet. Perhaps less traditional examples of computing stations are provided by mobile phones, personal digital assistants and handheld games consoles, and most new electronic appliances for our homes now have embedded microprocessors (even if not all of them sing). The predominant type of music technology for convenience and price is now digital gear, so that even most hardware boxes lurking within our music stores (synthesizers, fx pedals, mixers) turn out to be computational devices with embedded software.

There are other definitions of computer music, and rather more restrictive ones too, which consider only music that is entirely created on a computer, relies centrally on computer technology for the artistic conception, and cannot have any independent existence. But the working definition in the previous paragraph is a useful talking point to get us started, and this book will cover all manner of roles for computers in musicking.

To define music itself, by the way, would be to get ourselves into all sorts of semantic difficulties at an early stage. Suffice to say that since music is so vital a human activity, it is best to leave as much scope as possible for people to disagree on its definition. So this book will not restrict itself to music that fits clear models of Western concert music, but will cover much alternative work, such as sound installations and other sound art. We must always be conscious of the many and rich circumstances of music making throughout the world; certainly, Common Practice Music Notation written for conservatoire-trained musicians is not the only valid way of going about matters.

With music varying widely in its intended function and setting, it may be spontaneous or require intensive planning, may be pre-composed or improvised within different parameters, and may be disseminated through various routes including community participation, performers touring, record stores or broadcast technology. We shall explore many different ways in which computational technology intersects with musical practice. Where computers have thoroughly inserted themselves throughout society, they are naturally to be found as central tools in music production and performance. The convenience of much modern music making tends to rely on the substantial editing capabilities of computers, and many musical developments are now absolutely dependent on them.

In one ironic work conceived by Reinhold Friedl, however, ‘computer music’ was produced by drumming on old Atari computer shells (Peter Hollinger was the percussionist at the

¹This is not the forum to consider any solutions to the disparity in wealth and opportunity around the world that undermines any over-optimistic recognition of a universal ‘digital age’. Computers are predominantly the preserve of richer countries, and though initiatives like the One Laptop Per Child project are to be welcomed as increasing accessibility, clean drinking water and other more fundamental concerns take precedence over the relative luxury of computer music in the grand scheme of things. Nevertheless, you probably have access to computers – otherwise you wouldn’t be interested in this book – and you probably fall into the 20% or so of the world population with access to the Internet; and on this basis we proceed.

premiere performance in Berlin, for the International Computer Music Conference in 2000). This example shows how easy it is to undermine any ‘canonical’ definition with counter examples and borderline cases, and perhaps indicates why we need to be careful in general. What is music, after all, but a social construction, and why separate computer music from it? Mainly because categories are sometimes useful for partitioning knowledge for the purposes of reasonable learning, but not because we have to take the borderlines too seriously; we shouldn’t worry if we find ourselves straying across them.

Computer music is an interdisciplinary pursuit, at the intersection of the arts and the sciences. All sorts of disciplines inform it: music, computer science, psychology, physics (acoustics), engineering (signal processing, electronics) [Moore, 1990, p. 24]. It is important to realize that academic disciplines are themselves just categories of convenience, and there is no cause for any sort of inferiority complex just because computer music is a younger one. We can take any results from whichever field can help us in our quest. We can walk a line between the arts and the sciences and jump off in whatever direction our investigations take us. So it is perfectly possible for a physicist and a composer to collaborate; they might even be the same person, such as the ‘composer–scientist’ Bob Sturm, one of whose works is evocatively entitled *50 Particles in a Three-Dimensional Harmonic Potential: An Experiment in 5 Movements* (1999)!

1.1.1 Some Examples of Computer Music

The provisos in the previous section have shown that we must be careful with categories, which, if too quickly taken as real, highly constrain our ability to survey matters. As the artificial intelligence guru Marvin Minsky has pointed out, musical style only really means anything with respect to a vast field of examples [Minsky and Laske, 1992]. A data-driven approach constructs models from a host of examples, and it is healthy to use examples always to clarify points. In this spirit, some central exemplars of computer music are now listed, as a prelude to the many fascinating inventions and explorations detailed in this book. These examples are not in any way meant to encapsulate the field, but to pique your curiosity.

An early example of writing a computer program to generate music is the *Illiac Suite*, named after the University of Illinois Automatic Computer (hence, Illiac), by the musical chemists Lejaren Hiller and Leonard Isaacson, which was first performed on 9 August 1956. This painstakingly crafted work is described in great detail in their book *Experimental Music: Composition with an Electronic Computer* from 1959, and collects various musical experiments from species counterpoint to serialism. Yet they weren’t the only ones to investigate the potential of computer-assisted composition at the time; an unsuccessful attempt on the pop charts was made in 1956 by Douglas Bolitho and Martin Klein with the song *Push Button Bertha!* In truth, only the song’s melody line had been composed following machine instructions, and a lyricist, Jack Owens, was brought in to anthropomorphize the Datatron computer on which the program had run [Ames, 1987]. Contemporary manifestations of computer-generated music include the Koan software, MadPlayer portable

music hardware, RjDj for the iPhone, and many more, where we really have become used to receiving new music composed and played back on demand at the touch of a button.

John Chowning adapted the frequency modulation (FM) technique that underpins FM radio into an algorithm for digital sound synthesis in 1967. It took until 1983 for this work to be taken up by a mass audience, in what remains the biggest-selling synthesizer of all time, the Yamaha DX7. The technique has gained the status of a classic method of sound synthesis, capable when carefully handled of some realistic sounds as well as empowering new soundscapes, and was used in countless 1980s records as well as in Chowning's own 1970s computer music compositions. It has been the subject of many implementations, from mobile phone soundcards to virtual emulations of the original hardware DX7. The Fall 2007 edition of the *Computer Music Journal* pays homage to Chowning's *Stria* (1977), promoting this FM composition to the status of a modern classic.

We are now used to computers which can record and process many channels of sound in realtime. But before the 1990s typical computers were too slow to allow such manipulation (particularly at higher sound quality), and often confined themselves to sending control messages. Analog synthesizers were preferred by musicians in the 1960s and 1970s because they responded immediately. One pragmatic use of computers was to send control information for synthesis hardware rather than directly synthesize everything virtually. In the 1980s MIDI (Musical Instrument Digital Interface) was the control messaging standard of choice, but it is worth featuring one earlier system. Max Mathews's GROOVE (Generated Realtime Operations On Voltage-controlled Equipment), in operation from October 1968 [Mathews and Moore, 1970], employed a DDP-224 computer as the control engine for an array of analog synthesis equipment. This allowed complex musical time functions far beyond the basic existing analog step sequencer. Laurie Spiegel further adapted this set-up to create VAMPIRE (Video And Music Program for Interactive Realtime Exploration/Experimentation, extant 1974–9) [Spiegel, 1998], an early live audiovisualizer. There are deep historical precedents to the current fascination with live audiovisuals in club and cinema contexts.

One recurrent dream in computer music is the creation of a virtual performer. The idea of a machine musician has a long history in human thought, and has heady implications in artificial intelligence. It is a fascinating endeavor, to shape a system that will become an independent musical agent, and, hopefully, sufficiently adept and interesting even to promote new modes of interaction. To mention one computerized twist on this tale, the trombonist George Lewis created a famous system called Voyager (the system has been around under this name since 1985, though there are earlier experiments). Voyager is a highly personal creation, a complex system with a wide scope in live improvisation. It can play with multiple human players, typically utilizing pitch-to-MIDI convertors to track notes. Digesting this information, and forming multiple statistical representations of the state of play, the system willfully interjects and responds. Lewis isn't seeking to make a deterministic system by any means, but an independent-minded and stimulating equal participant in music making.

We are becoming accustomed to having access to new sound synthesis, analysis and processing possibilities, and they can change musical culture. Auto-Tune² is a plug-in of great popularity with producers, primarily used for correcting a singer's wobbly intonation (most notoriously when that singer is rather untalented to begin with). The realtime digital signal processing first attempts to find the pitch of an input audio signal, and then if this is mistuned with respect to a template (e.g., notes in the current key of a song), carries out a subtle resynthesis to ensure the pitch is appropriate. The effect can also be applied in a less subtle way quite deliberately, for example by setting the pitch correction speed too fast for the processor to cope. This was famously used by the producers Mark Taylor and Brian Rawling for Cher's 'Believe', released in November 1998 [Sound, 1999], and by Nigel Godrich and Radiohead on the *Kid A* album from 2000.

We could go on, through musical computer games, musical robots and mobile phone orchestras. The rest of this book should provide plenty of further examples.

1.1.2 Sociable Computer Musicians

Music is a social art, at least outside the musician's preparation time for practicing and composing. Music can be the social glue which binds human gatherings together, and is often highly participatory for all involved, even if some settings, such as classical concertizing, can harbor a more one-way transmission process from stage to audience. For a musician, social networking is a central part of the business, and most music making is a collaborative pursuit, even if ensembles vary in the degree of hierarchy they contain.

In computer music, the same eventual social outcomes and negotiations must be borne seriously in mind. Though the lone bedroom producer or the late-night coder are stereotypes which sometimes seem to deny this, online digital music releases are certainly intended for other ears. Indeed, computers as a communication tool enable many new forms of social interaction (the popularity of social networking sites like MySpace and Facebook attest to this). Yet computer technology also has many parts to play within more traditional musical settings, and a sympathy for and understanding of the social values of music making can ease engagement between the acoustic and the digital worlds. We will also come to explore many new ensembles empowered by computers, from the laptop orchestra to mobile phone groups, and see how network music brings a new vitality to musical networking!

New communication systems tend to impact in a major way on society, and the Internet has been no exception. Musical distribution models have been revitalized, and intellectual property laws challenged in many ways. Economics and legislation are lagging behind the wave of experimentation, but new music in the Internet age is in rude health. The explosion of downloadable media content, popularly invoked by mention of such brands as MP3, iPod/iTunes and YouTube, is a major factor in the reconfiguration of the music industry in the current era. Later parts of this book will tackle this online music presence, and explore

²www.antarestech.com

new technologies intended to automatically keep abreast of the voluminous data available, under the theme of ‘music information retrieval’.

It seems unfair to pick any one example amongst many, and thus perhaps to further a viral campaign, but let’s point to Buraka Sound System. Whether they are still there when you read this can be determined by checking the validity of this web link (<http://www.myspace.com/burakasomsistema>). The popular YouTube video for ‘Sound of Kuduro’ features the odd glimpse of a laptop screen, but also promotes the essential human presence in music making and enjoyment (<http://www.youtube.com/watch?v=4CkXhtw7UNk>). I wouldn’t have found out about them and obtained their first album, *Black Diamond* (2008), had it not been for the rapidity and ease of online promotion that bypasses conventional media. Musicians actively explore any new technology that helps them to reach other people.

1.1.3 Why Investigate Computer Music?

There are many practical benefits that digital technology, and particularly computers, have brought to music, which will be explored at length in this book. These often have the form of convenient storage and editing capabilities, as in the use of computers for audio recording and score typesetting. The transformative powers of computer algorithms to modify and tweak sound, and generative prowess in the creation of new sounds and music, will also be discussed.³ We shall see many analytical tools for music which are only available by employing computers. Moore [1990, p. 4] notes that computer music offers new standards of ‘temporal precision’ and ‘precise, repeatable experimentation with sound’.

The notion of experimentation is further exemplified by the natural employment of computers in the modeling of music, for multifarious purposes including acoustics, music analysis, psychology and, not least, composition. It is natural to seek computer assistance in exploring novel musical systems and theories. The computer can be a laboratory to try out alternative tunings, or new esoteric twists on serialism. Whilst Iannis Xenakis began his work in stochastic music in the 1950s manually, by 1962 he had turned to the computer to automate the hard work of using probability distributions. The Autechre album *Confield* (2001), a powerful abstract electronica somewhere between skewed electronic dance music and club-tinged electroacoustics, is the result of intense experimentation with computer generation of music.

Let us consider a clear example of the use of the computer as a basic labor-saving device. In order to compose *Music of Changes*, a work for piano from 1951, John Cage literally spent six months throwing coins. The Book of Changes is the Chinese *I Ching*, a system of 64 hexagrams, and Cage was mapping from the hexagrams to parameters of piano music. He wanted to give up certain compositional decisions to the oracle of the *I Ching*. The saving of time must have been a great relief to him when Lejaren Hiller, the computer composer and a close collaborator of his (for example on *HPSCHD* from 1969) wrote a computer program

³‘Unlike real oscillators, computer-simulated oscillators can produce low frequencies with ease and precision’ [Mathews, 1969, p. 53].

to ‘roll the dice’ on his behalf. Long printouts of pseudorandom numbers so generated could be provided for Cage to work from. Divination had been automated.

It is interesting to consider the gap between production and perception. There are undoubtedly sequences of actions which no individual human player could accomplish, impossible synchronizations that no human ensemble could navigate with or without conductor and click track, which can be programmed with a computer. In some cases these are still perceptible, despite being beyond the means of human players to enact. In terms of sound itself, there are regions of timbre denoting truly novel electronic sounds which nonetheless remain clearly audible. In this sense, there is leeway between our evolutionary capacities to perceive structure and to act in the environment, even though it is hard to deny close links between the two in biological heritage. We may interpret novel sounds through various glasses, from basic auditory mechanisms to cultural acceptability, and computers allow us to hit some new spots in this domain. It is useful, though, to be aware of the psychological factors which ultimately constrain those musical structures that can have a practical effect. To this end, the reader is encouraged to read and take inspiration from the psychoacoustic and music psychology literature [Moore, 2004; London, 2004]; [Moore, 1990, pp. 10–11].

Computers can replace human performers to allow the production of new sounds or hyper-precise actions. Introducing mechanical means of performance is not new in itself; for example, the Mexican-American composer Conlon Nancarrow was hammering out a series of glorious studies for player piano from the 1950s (prompted himself by Henry Cowell), and anticipated many of the later possibilities of sequencer music. The roll of honor here might go on to mention Frank Zappa’s *Jazz From Hell* (1986) composed with Synclavier, Alistair Riddell’s computer-controlled piano works, or Vidovsky’s music for MIDI Piano, but the rigid metronomic virtuosity of electronic dance music is perhaps the most public manifestation of tight mechanical specification. Music for computer offers the most perfect control of any desired aspect, whether it be pitch, timing, timbre, space or any other. Sometimes, particularly in earlier non-realtime works for tape, this power comes at the price of losing the human performance element (some composers might also call this a benefit, as they escape the additional intermediacy of interpretation). As Paul Lansky has written of tape music, ‘You are no longer scripting a performance’ [Lansky and Roads, 1989, p. 41], but fashioning a finished opus, give or take the act of diffusing it in playback. There are also, however, many works which seek to integrate computer resources with human players at the time of performance, and these can still exploit inhuman transformations enabled by computer as long as the computer is somehow aware (through human action, or clever algorithms) of when to act.

1.2 QUICKSTART GUIDE TO COMPUTER MUSIC

Most people have a horror of reading manuals closely. You already understand that you can skip around this book as you need and pick out the sections that interest you. But if you are

a newcomer to this field, you will find there are some important concepts to ‘get your head around’. So, to help you move quickly to the outcomes that most interest you, some core ideas are introduced here in the manner of a ‘quickstart’ guide. These provide a central basis for computer music which should help your explorations, though it may repeat things you already know; just skim to where you feel you have something to gain.

1.2.1 Sound Waves and the Brain

Sound waves propagate in air as changes in pressure, essentially by air molecules bashing against each other, forming regions of higher (compression) and lower (rarefaction) density. So the more air molecules are crammed together, the higher the density and pressure. The branch of physics known as acoustics describes such phenomena, and sound waves can actually propagate in any state of matter. You may have seen reports from astronomers about (very low frequency) sound waves in interstellar gas clouds, or have heard sound underwater where the waves propagate four times faster, enabling you to hear more distant objects. There are even underwater concerts, though none yet in interstellar gas clouds to my knowledge, but as humans are most usually making music on land, air molecules seem to be what we’re most used to.

Yet, once the sound waves reach our eardrums, they are translated through a number of other mediums, such as a solid, the tiny bones in the middle ear, and a liquid, the fluid-filled cochlea. An incoming sound eventually ends up as electrical signals in the brain, produced by the electrochemical triggering of the hair cells by the motion of the basilar membrane. The physiology of this transduction process is fascinating, and well worth further study [Pickles, 1988; Moore, 2004], but for now you should simply be aware that the domain of physics at the point of intersection with human audition becomes psychophysics, or more specifically psychoacoustics. And once on the auditory pathways of the central auditory system in the brain, sound becomes a subject for neurobiology and eventually at the highest level cognitive psychology. Science has penetrated many of the mysteries concerning how humans perceive sound, but much work remains to be done, and it would be misinformation to say that human hearing was a solved problem. Neuroscience is a fascinating discipline, but cannot yet provide a full working model of auditory cognition, despite some incredible successes in probing human capacities.

Where does this leave us? Results from acoustics will be used when modeling sound with computers. Results from psychoacoustics inform computer models of human hearing. The psychology of music is an active research field, with many overlaps with computer music.

1.2.2 The Time Domain

There are two main ways to plot sound that crop up sufficiently often to make explaining them an important part of this introductory chapter. Before either is discussed, please bear in mind that human auditory perception itself is the best judge we have of sound *as an experience*; nevertheless, using graphical plotting methods to explore sound has many benefits and can provide great insights.

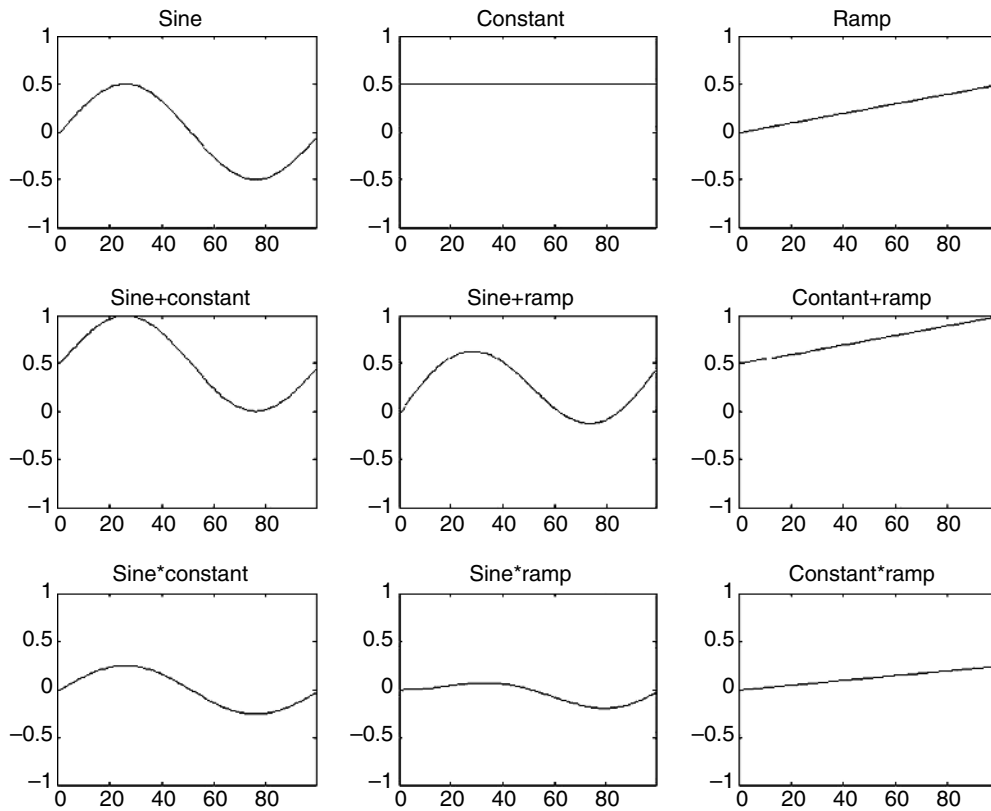


Figure 1.1 Basic signal operations in the time domain. The three signals on the top row are combined by addition in the middle row and multiplication in the lowest row. The signal operations act pointwise over time. The ‘constant’ signal is also called a DC (direct current) offset. When it is added to another signal, it offsets that signal; when it multiplies, it scales the range.

Pressure waves can be measured at a single point in the air; just put a microphone there! An electrical analog, such as changing voltage, can be measured across a component in a circuit. The time-varying value so gained is a **signal**. At this point, the first plot we shall explore is that of signal strength on the y -axis against time on the x -axis. This is the **time domain**, and the signal is also called the **waveform** (some example time-domain diagrams appear in Figure 1.1). The signal strength is measured with respect to some baseline, typically the zero line on the graph. For instance, in air, the pressure is measured with respect to baseline atmospheric pressure, and will be positive if there is greater pressure (density of air molecules) and negative if there is reduced pressure (density). Signal strength goes by the special name **amplitude**. It is possible to convert to a measure of signal **power** (capacity to do work, energy available) by squaring the amplitude. Most signals are not a simple

Linear and Logarithmic

Consider the sequence 1, 10, 100, 1000, Each number is formed by adding another zero at the right, effectively multiplying the previous number by 10 – equivalent in the decimal system to shifting the 1 across the columns to the left. Since these numbers otherwise will span a very large range, they can most conveniently be compared simply by noting the number of zeroes in the written expression. 1, 10, 100, 1000, . . . is replaced by the sequence of **exponents** 0, 1, 2, 3, The original numbers can always be recovered by taking 10 to the **power** of a given exponent; that is, multiplying 10 by itself as many times as the exponent asks ($10^0 = 1$, $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$, . . .).

The operation described in the previous paragraph is a transformation of ordinary numbers into an alternative domain, that of **logarithms**. A logarithm always acts with respect to some base; for our convenience, 10 is the base used here. The logarithmic domain is a much improved vantage point from which to survey a wide range of numbers. This is the motivation for using decibels as a measurement of signal strength instead of basic amplitude. The human auditory system can cope with a range from 10^{-12} to 1 watt per square meter of power, from the threshold of hearing to the threshold of painfully intense sounds. The ratio of quantities here, comparing painfully loud to vanishingly soft, is 1 followed by 12 zeroes to one; the logarithmic domain gives a better handle on such a wide range. **Exponentiation** allows the recovery of normal numbers again.

Evolution has equipped us to be sensitive to the log (logarithm) of quantities in many instances, rather than the linear scale. Logarithms are used often with musical quantities, for instance in the perception of pitch as well as loudness, but are also useful, for example, in modeling sensitivities to light or to pressure on the skin.

constant direct current, but alternate around the baseline; even though signal strength could be positive or negative, power (the square of amplitude) remains positive. Power can also be measured as an average over a stretch of time, giving rise to a well-defined statistical average amplitude such as the **root mean square**.

For audio signals dealt with by computer software, the range of amplitude is typically normalized to floating point values between -1.0 and 1.0 .

We shall deal with signals throughout this book. As a warning at this stage, it is interesting to note how they can be combined. It is possible to talk of scaling a signal (multiplying all its values by a constant) or offsetting (adding a constant signal). These operations are fundamental for mapping signal values to other number ranges, required as the preconditions of later processing. Be forewarned that the mathematics of combining signals can be disconcerting; to multiply two signals, each a function of time, $a(t)$ and $b(t)$ to form a new signal $c(t) = a(t) * b(t)$ is to multiply together the instantaneous values of a and b at each moment of time. Figure 1.1 illustrates this for three basic signals. There are some important consequences of multiplying signals in the time domain, but we shall return to this point later once we have more machinery to deal with it.

Human hearing is approximately logarithmic (see sidebar) over much of its scope. For this reason, it is convenient to convert amplitude to units known as **decibels**.⁴ The standard equation for this is:

$$\text{value in decibels} = 10 \log_{10} \left(\frac{\text{input power}}{\text{reference power}} \right) \quad (1.1)$$

where the power of a signal is the square of the amplitude, and \log_{10} is used for logarithms to base 10. To work with amplitude directly:

$$\text{value in decibels} = 20 \log_{10} \left(\frac{\text{input level}}{\text{reference level}} \right) \quad (1.2)$$

The logarithm must be taken to a base, 10 in this case, and the measurement of decibels is always with respect to a reference level. If we imagine signals within the range -1 to 1 , we could set a reference level at full power of 1 , or make it very small, perhaps 10^{-12} watts per square meter (this latter case is often denoted by dB SPL, which stands for sound pressure level). In the first case, full amplitude would be 0 dB and all lesser amplitudes would be represented by negative numbers of decibels. In the second, 1 would be 120 dB ($10 \log_{10} 1/10^{-12}$) relative to the reference. In both cases, an amplitude of zero would be negative infinity, since that is the logarithm of zero. You may have seen audio software and hardware meters using various conventions here; a mixer's working scale could be from $+9$ dB down to infinity, allowing some 'headroom' rather than immediate overloads. I have also skipped a couple of technicalities – for example, taking an average measure of amplitude during some time span – here for the sake of the exposition [Loy, 2007a, Chapter 4].

Just a word or two of warning: decibels are useful since they are more perceptually relevant than amplitudes. But loudness as a psychoacoustic phenomenon is still more complicated than this. Also, the logarithmic scale should not be used in exactly the same way as the original values – do not add decibels together as if this added the associated

⁴Some authors write deciBel to respect the human surname Bell, and this also explains the standard abbreviation dB. Lower case is used in this book for decibel, as it is for amp, voltage etc.

signals' amplitudes. For such operations, it is necessary to return to the amplitude domain. However, certain rules of thumb are applicable: for example, doubling the amplitude of a signal is equivalent to a decibel increase of 6 dB. Successive gain operations, each of which multiplies the amplitude of a signal, can be dealt with by adding decibels, since they are multiplications rather than additions in the original amplitude domain.

1.2.3 Periodicity

We are sensitive to any object which vibrates in a repetitive manner at sufficient rate. The object follows the same physical pattern over and over again, many times per second, an **oscillation**. For example, in speech or singing, when you produce a vowel sound, the vocal folds in your larynx are set into oscillation, passing through puffs of air at a steady rate to create a periodic source. Periodic vibration is characterized by the **period**, measuring the length of an individual repetition in seconds, or the **frequency** of vibration, the number of repetitions per second, measured in **Hertz** (abbreviated Hz). These two quantities are inversely related to one another, so that a 100 Hz oscillation has a period of $1/100 = 0.01$ seconds, or one centisecond.

Figure 1.2 depicts two periodic signals. One looks wobblier, with more peaks than the other, but both repeat once per second (since the reciprocal of one is one, the period is one second and the fundamental frequency is 1 Hz). In some sense that we must unpack, the smooth-looking function here is much simpler and more basic than the wobbly function.

A periodic wave will play back in a loop, but there is always a choice of where to start it from when first setting it in motion. The starting position within a single cycle is the **phase**. You will also see phase referred to as the instantaneous position within a period at a given time.

It is necessary at this stage to introduce one method for analyzing sound which is particularly applicable to periodic signals. This method is Fourier analysis, a theory that allows any perfectly periodic sound to be broken down into the simplest possible basic oscillations. To find what these most basic periodic signals are, we can consider the simplest physical system with a periodic solution, as described by a mass solely under a linear restoring force, such as a mass on a spring without any friction. The solution to this system is the pure sound known as the **sine** (also called a sine tone or sine wave, or more generally a sinusoid or sinusoidal oscillation; you may also be aware of the dual function known as the cosine, a sine function shifted along a bit, which is also a sinusoid). The sound of this waveform is similar to that of a tuning fork, and some electronic tuners or alarm beeps; older analog music studios often had dedicated sine tone generators. It is easily produced on a computer from its basic equation. It sounds, well, plain, and has in some sense the simplest sound color, because if there were a simpler periodic sound, we could decompose the sine in turn into simpler entities yet; but in fact the sine is the end of the line.

Having introduced the sine, we can use it. The Fourier theory states that a perfectly periodic signal can always be analyzed in such a way that it is broken down into sine

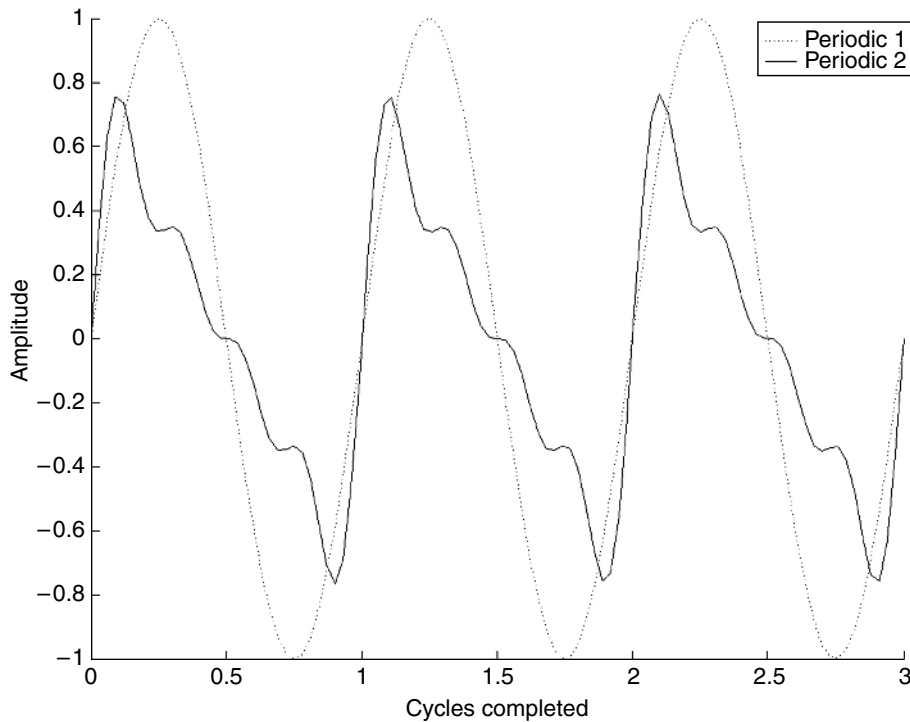


Figure 1.2 Periodic waveforms.

components. Each sine component can be described by a frequency and a phase, though the phase is often less important in perception for periodic sounds, as well as an amplitude weighting to say how strongly that component is present. Further, given the period of the signal to be analyzed, the lowest frequency component which can appear will correspond to this **fundamental frequency**. All other sine components will have frequencies which are at the possible integer multiples of this fundamental frequency, the spacings of a **harmonic series**. So if the fundamental frequency was 100 Hz, the components would have frequencies of 100, 200, 300, 400 and so on. Such multiples are termed **harmonics**, numbered from the fundamental as the first harmonic, or **overtones** when numbered such that the second harmonic (double the fundamental frequency) is the first overtone. Not all these components will necessarily have the same strength in the mixture which recreates the original signal; some frequencies may not be present at all.

Now, it is a short step from mentioning frequency to invoking the word 'pitch', but we must be careful. **Pitch** is a psychoacoustic attribute of sound in which the brain tries to find an explanatory fundamental frequency as if an incoming sound were periodic and its components followed a harmonic series. Since many independent sounds can arrive at once,

Sinusoids

A mathematical description of the sinusoid must take account of the natural unit of angle in trigonometry, the radian. The sine function is periodic, with a period of 2π radians. It is always possible to ‘run through’ a period at faster speed by multiplying the 2π by a frequency f , to form an angular velocity $2\pi f$, usually denoted by the Greek letter ω . We can formulate a general sinusoid as

$$\sin(\omega t + \phi) \tag{1.3}$$

where ϕ is a phase offset, since it is possible to start off this function at any point in its cycle. From trigonometry, we know that a cosine wave is a sine wave shifted in phase by $\frac{\pi}{2}$ radians. The phase offset allows the expression of a linear combination of sine and cosine waves ($\sin(A + \phi) = \sin(A) \cos(\phi) + \cos(A) \sin(\phi)$; thus, since ϕ is a fixed angle, it determines the coefficients of combination), which reduces to a pure sine for $\phi = 0$ and a cosine for $\phi = \frac{\pi}{2}$ (remember, a cosine is a sine advanced in phase by $\frac{\pi}{2}$ radians). Because of the duality of sine and cosine, it would have been possible to start from cosine waves instead of sines; hence the more general term sinusoid (when speaking informally, sine is often used instead of sinusoid). Note that the only two parameters of the function are the frequency and the phase.

By scaling the output, it is possible to add a third parameter for the amplitude of the signal, but since this could be done with any signal, it is not presented here as part of the equation for a sinusoid itself.

and not all sounds approximate a harmonic series, this is in general a rather hard problem, and we shall return to consider it in more detail in Section 3.5.1.

As a demonstration, the more complex periodic sound (periodic 2) from Figure 1.2 is broken down into sinusoidal components in Figure 1.3. Figure 1.4 then demonstrates a short-cut version, called a **line spectrum**, which shows the different harmonics and their respective strengths in the mix, though discarding any phase information. Because the exact mixture is known in this example, the diagram shows perfect identification of the components; but in general, as we shall next investigate, finding the constituents of a complex sound can be a little more tricky.

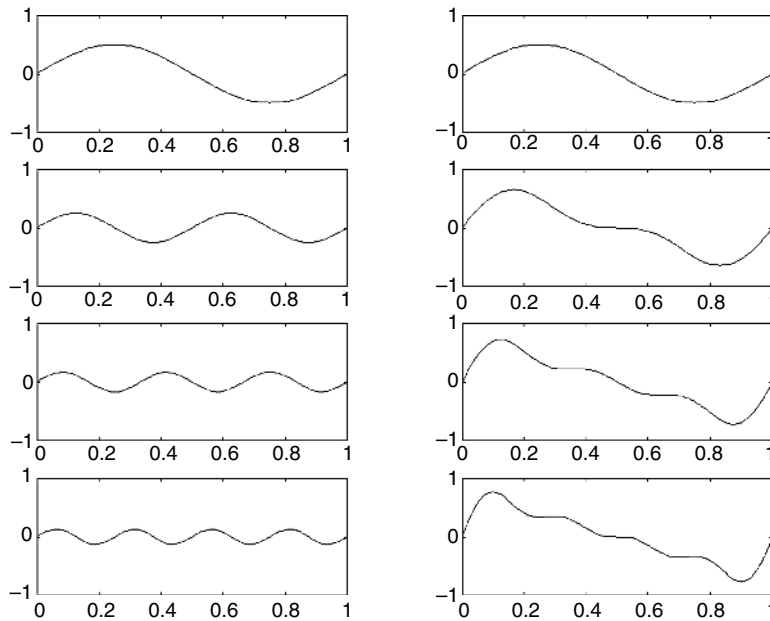


Figure 1.3 Sinusoidal components of a periodic sound. The left column shows each individual sine component (they vary in their amplitude). The right column gives the mix so far at each stage, as the sines are added together down the page.

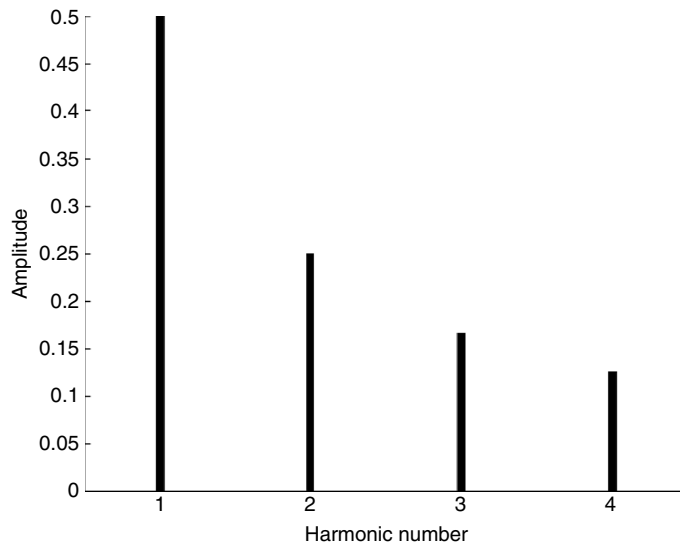


Figure 1.4 Line spectra. The frequency and amplitude of each component of the periodic sound are indicated (though phase information is dropped).

1.2.4 The Frequency Domain

We are not always in a position where we have a perfectly periodic sound of known period. We may not know the period of a sound in advance; we may have a sound which varies its period (a glissando, for instance), or a noise source which is not periodic at all. Many musical sounds are only at most psuedo-periodic (there is some subtle fluctuation of period), and the period is only relatively stable during a **steady state** of sustained oscillation, rather than during the fast-changing **transient** beginning of the sound.

In the physical world, a vibrating object can simultaneously oscillate at a number of different frequencies to make a complex vibration (the coexistence of multiple frequencies is described by the principle of superposition in physics). Depending on the material properties of the object, the relation between different excitable frequencies, also often known as **modes**, can follow the simple linear spacing of the harmonic series, or take on more complicated inharmonic patterns. The term ‘harmonic’ is not applicable in the latter case if analyzing component frequencies, so **partial** or **mode** are the preferred general terms.

Despite this failure for sounds in general to be well-behaved and periodic, it is still possible to measure whether a sound has energy at any particular frequency during a given time. Note that because of the dual nature of time span and frequency, the accuracy of this measurement in frequency must depend on how much of the period is taken into account. In principle, though, you might imagine ‘testing’ a sound beginning at one point in time, for a certain duration for each frequency you want to measure. By carrying out a succession of these tests at different starting points, it is possible to make a map of a sound’s frequency content across time.

This process of measurement is a particularly useful tool for analyzing sound, and the most common form of this analysis employs the Fourier analysis mentioned above. Whilst it would be possible in principle to try and measure every frequency starting at every point, a computer only has finite memory and finite processing resources. A compromise to allow for practical constraints is to use a large basic period (relative to most frequencies we want to study), and carry out a Fourier analysis *as if the signal were periodic during that segment of time*. As long as the audio signal doesn’t change too much during this period, we will obtain measurements for all the harmonics of the analysis frequency. Whilst this doesn’t get us a measurement for every frequency, we at least gain a useful subset of measurements (I am skipping a few complications for the time being).

The **spectrogram** is a plot over time indicating the energy in a sound at different frequency components (which are the multiples of the analysis frequency). One Fourier analysis is carried out for each step in time; these steps are evenly spaced out, though the analysis regions might partially overlap. Typical figures on a computer might be an analysis fundamental frequency of around 43 Hz, and 86 analysis **frames** per second (so the Fourier analysis regions overlap by one half). Figure 1.5 shows a spectrogram obtained in this way. The shade plotted at a particular (time, frequency) point is a logarithmic power measurement (essentially, decibels). The three-dimensional graph (shade here substitutes for seeing the

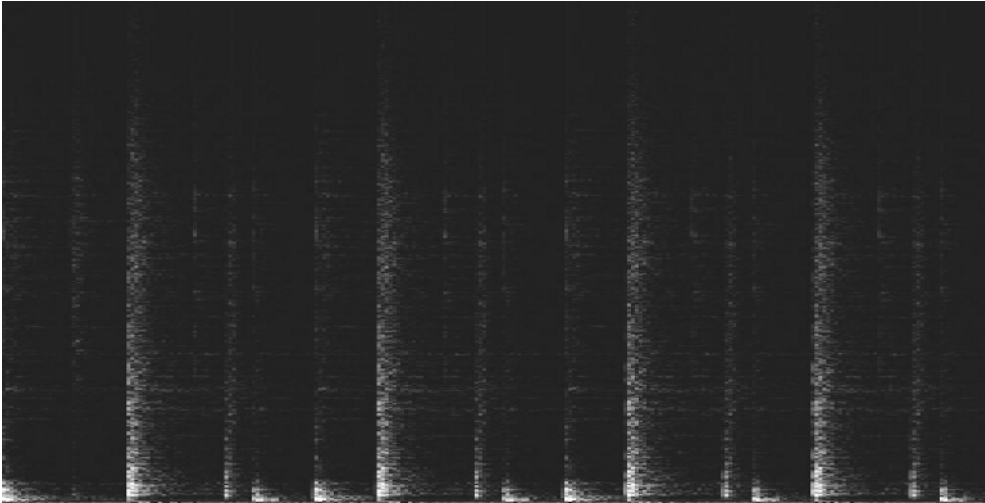


Figure 1.5 Spectrogram view of a drum loop. The percussive strikes are visible as (noisy) vertical bands. Noise sounds tend to be broadband, having energy at all frequencies.

third dimension directly) is best thought of for now as a grid, a time-frequency plane of cells, each of which is an energy measurement at a particular time and frequency.

There is a lot more to say concerning such analysis, and Chapter 3 will take up these issues, including the mathematics. For now, it is important to recognize that the **frequency domain** provides an alternative viewpoint on sound to the time domain. You will also see references to the **spectrum** of a sound, as an energy distribution over frequencies.

Fourier analysis has been used here to break all sounds down in terms of sines. However, this does not mean that the description is necessarily compact for all sounds. White noise is the most brutish form of noise; it has equal energy at all frequencies on the average over time. A Fourier analysis, even with a tiny fundamental frequency of analysis, will only pick up this mean flat spectral distribution. So for noisy sounds, the spectrogram can be a rather inefficient descriptor. The separate modeling of noise components in a sound is an issue for Section 3.2.4.

The frequency decomposition view of sound is of central importance, and evolution has not missed out on this good trick. The cochlea in your inner ear is at heart a frequency analyzer, with 3500 or so inner hair cells strung out on the basilar membrane and differentially sensitive to incident frequencies [Moore, 2004, p. 33].

I have already warned of the complexities of the psychoacoustic attribute of loudness. The dynamic range of the ear varies with frequency, and a set of contours can be experimentally established which describe how different amounts of power at different frequencies equate. These **equal loudness contours** (also called Fletcher–Munson curves) show that it takes substantially more pressure at bass frequencies for us to react, and that we are particularly sensitive to areas of the spectrum associated with speech, peaking at around 3500 Hz, the resonant frequency of the auditory canal before the eardrum. This frequency-dependency also means that the same music played back at different overall volumes automatically has a different equalization (EQ). Some hi-fis attempt to compensate for this to some extent (e.g., boosting bass at low overall volume), but it is an important warning to mix engineers.

1.2.5 Digital Audio

The computer has to deal with ones and zeroes. Whilst friendly software might hide this from you in various ways, nevertheless, at the heart of the machine, a binary regime is in play, and the computer is manipulating **bits** (**binary digits**, in other words, which have the value 0 or 1). But how can the real world, where movement is smooth and apparently continuous, be digitized? The first thing to realize is that the computer doesn't have to store a number into just one of two values, but can use a succession of bits to represent a given number. For instance, eight bits is one **byte**, and binary arithmetic tells us that we could represent 2^8 or 256 different values by using one byte. By using 256 different values we can obtain a much better approximation to the instantaneous value of a continuous sound than we could with two!

Computers have to digitize, to turn continuous **analog** values into **digital** ones. The process implies some loss of information; but, importantly, by increasing the **bit resolution** we can have as high an accuracy as desired, and our approximation gets better by a factor of two with every bit we add.

This is not the only approximation; we must also digitize time itself. We cannot store a value as if there were an infinite number of instances per second, for this would require infinite storage. So we must select a **sampling rate**, often denoted by R or f_s in mathematical notation. A typical sampling rate would be 44 100 Hz, that is, measuring a value, taking a sample, 44 100 times per second. As these sampling instances are equally spaced in time (driven by an accurate clock) there is one value sampled every $1/44\,100 \approx 0.0000226757$ seconds. The sampling of an input waveform in time and amplitude is represented in Figure 1.6 by the grid, and the diagram also demonstrates the **quantization** that must take place, rounding off signal values when captured to the nearest grid levels on the y -axis.⁵

⁵The term quantizing recurs in sequencing, where it refers to forcing the start time values of events to comply with a grid.

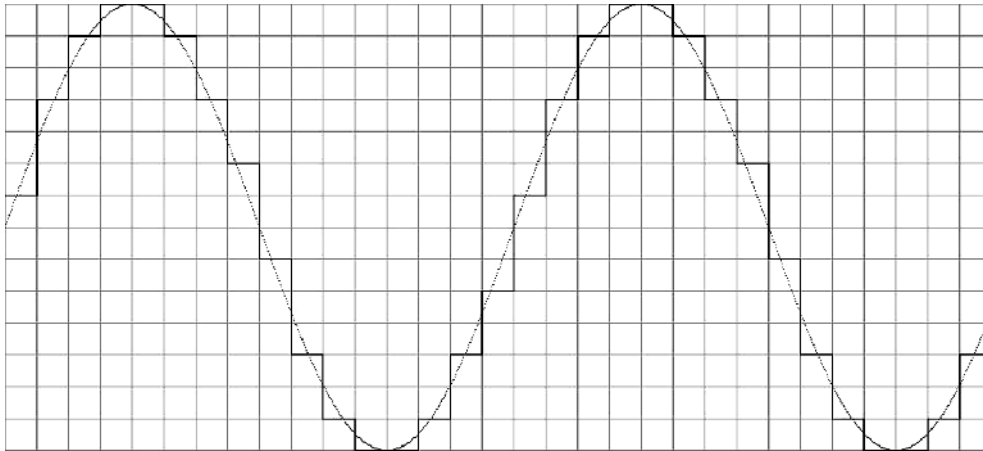


Figure 1.6 Continuous function sampled onto an underlying grid. Note that the values at each sampling step have been quantized to the nearest grid line on the y -axis to give the step function.

The digital sine

For a digital system, it is only possible to deal with discrete samples of time n ; with a sampling rate of R per second, time passes such that sample n occurs after n/R seconds. For a sine, the distance traversed in the angular domain of the function input is time * angular velocity = $\omega n/R$, and we can formulate a general sinusoid as

$$\sin(\omega n/R + \phi) \quad (1.4)$$

Computer music will always remain dependent on some form of microphone and loud-speaker if interfacing with the outside sound world. Note that there is simply no way to get audio into and out of a computer without an ADC (analog to digital convertor) and a DAC (digital to analog convertor).

Digital audio is supported by a mathematical theory which supplies conditions on the reproduction capabilities of the sampling process [Loy, 2007b; Watkinson, 2001; Pohlmann, 2005]. The **sampling theorem** is the essential statement of the reproducible bandwidth, that

Proof of Aliasing

A trigonometric derivation of aliasing, due to Richard Hamming [Hamming, 1989, pp. 22–24], is presented here for mathematically inclined readers. Consider the cosine form of a general sinusoid: $\cos(2\pi f n + \phi)$ where f is the frequency as a proportion of the sampling rate and n is the sample index, $n = 0, 1, \dots$. Thus a frequency of $1/2$ in this case would correspond to the Nyquist rate, and a frequency of 1 to the sampling rate. The number f can always be broken down into an integer and a fractional part (there will only be integer parts involved for frequencies at the sampling rate and above). So taking the integer part as m and the fractional part as a we obtain $\cos(2\pi(m + a)n + \phi)$. Because of the cosine periodicity, and since n is always an integer, the $2\pi mn$ in the argument can be removed. We are left with $\cos(2\pi an + \phi)$ where a is known to be fractional. If a is less than $1/2$, we are done, because our original cosine was equivalent to a sampled cosine at a frequency under Nyquist. If a is greater than $1/2$, write $a = 1 - b$; then we have $\cos(2\pi(1 - b)n + \phi)$. The $2\pi n$ drops out again, and because a cosine of a negative angle is the same as that of a positive, and a phase shift does not affect frequency, $\cos(-2\pi bn + \phi) = \cos(2\pi bn - \phi)$. So all frequencies f reduce to frequencies within the range 0 to $1/2$, and aliasing is inevitable when sampling.

is, exactly which frequencies can be represented by a digital system. As you might suspect, the higher the sampling rate, the better the range of representable frequencies. The critical boundary is the **Nyquist limit**, which is at half the sampling rate. No frequency above Nyquist can be represented by a digital system. To restate this in another way, there must be at least two samples per period if a sine tone is to be accurately stored.

What happens to any higher frequency? The digital system will attempt to store such frequencies, but will end up recording an alias, that is, another frequency below the Nyquist limit, instead. This gives rise to **aliasing** distortion, also called **foldover** – basically, misrepresented frequencies.

In order to avoid this happening, which would cause audible noise, an ADC is always preceded by a low-pass filter, which makes sure that there is no energy in the input at frequencies over half the sampling rate to cause aliasing. The DAC also makes use of a low-pass filter, as the final reconstruction step from the sampled digital values to the smooth waveform of the continuous analog signal.

The computer can only run at a certain clock speed, so it has a maximum rate at which it could ever sample time. Measuring devices are also only accurate to a certain resolution. Yet the approximation is sufficiently good with high sampling resolution and bit resolution, and, critically, high-quality convertors, to convince even golden ears. Engineers now have many decades of experience in the design of the conversion process between analog and digital, now favoring one-bit oversampling convertors which neatly sidestep some of the difficulties of designing appropriate low-pass filters (with additional benefits). Older arguments about the quality of digital audio, which was admittedly a little shakier when first commercially available due to inferior convertors, have little credibility now. And the versatility of computer-based production for audio, as well as perfect copying fidelity with error correction, are too useful to permit a move back to analog. Digital file management is easier, allows random access and takes up less space; multiple generations do not build up noise, avoiding compromises in the process of recording. Of course, some analog recording formats are still commonly available, and valued for their own characteristics (and their own recording quirks such as print-through, wow and flutter), but digital audio is king [Watkinson, 2001; Pohlmann, 2005].

There is one more restriction of a digital audio system to relate, which is that bit resolution is intimately related to dynamic range. The rule of thumb is that one bit corresponds to six decibels of range. Since all rooms (even recording studio control rooms) have a background noise floor of at least 20 dB SPL, the approximately 98 dB range of 16-bit audio [Pohlmann, 2005, p. 37] can bring us close to the ear's own dynamic range at its most sensitive frequency region.⁶ The dynamic range of digital audio far surpasses the 55–65 dB standards for commercial analog audio, or even 80 dB for high quality professional tape machines [Roads, 1996, p. 40]. 24-bit digital audio has a potentially better dynamic range than the human ear, across the spectrum (in practice, conversion issues may restrict this a little). Whilst 32-bit (even 64-bit!) floating point audio is often used internally in software, the quality is dependent on the original bit resolution of the ADC used to get the analog data into digital format (there are 32-bit digital signal processor (DSP) chips, however).

Whether recording engineers in certain styles of music really use the available dynamic range or not is another issue, with a backdrop of a 'loudness war' in marketing and broadcast. Many mastering engineers, dance producers and noise musicians deliberately compress music to compete with each other for maximum playback volume; they should note that compression makes music tiring to listen to relatively fast, destroys ambience except for an ambience of onslaught, and has other implications on quality of playback and presentation of mix [Katz, 2007].

1.2.6 Filters

A filter is any sound processor which affects different frequency components of a sound in different ways. The filter might be used to inject or attenuate energy at different points

⁶We must cover 120 dB, so we take a new reference level for our digital system at the noise floor of 20 dB SPL, assuming that sound below this level (which the digital system cannot represent) would have otherwise been masked. This is not quite true in practice, due to nonlinear masking effects and frequency content of the noise floor, but does as a rule of thumb. Note that resetting the reference level is not breaking the 'don't add decibels' rule.

across the spectrum. Some filters (all-pass filters) do not affect energy at all, but introduce different delays at different frequencies. Indeed, this disturbance of the phase relationships in a signal is another basic characterization of a filter; cancellations and reinforcements are set up by combining the original signal with time-delayed versions of itself, which usually has the effect of changing the energy at particular frequencies [Roads, 1996, pp. 19–20].

There are certain basic filter types that frequently occur. Figure 1.7 details four basic types and their effect on the magnitude spectrum. Filter diagrams should not be idealized; in practice, filters have non-trivial stop bands, regions of transition between passing and blocking frequencies. They may not be perfectly flat, with some ripple providing additional coloring. The effect of a filter on magnitude is only one part of the story, and the effect on phase (frequency-dependent time delay) is also of note.

In general, an arbitrary filter can have a complicated effect upon the spectrum. Filters are of basic utility in modeling real sounds. The body of an acoustic instrument can be considered to be a filter which gives flavor to the timbre of a sound. A violin body is well coupled to the air (the strings are not) and colors the raw string vibrations. By changing the shape of your mouth and throat, you filter the air stream to produce different vowel sounds and consonants.

It is possible to view a spectrogram as a bank of filters; each frequency measurement is the output of one band-pass filter. The cochlea is also essentially a filter bank, with each inner hair cell a distinct band-pass filter only responsive to components of incident sound within a particular frequency range.

1.2.7 Timbre

Timbre might be described as a means of differentiating sound sources, which takes into account aspects of their acoustical production, even when controlling for certain common features. For instance, imagine a piano, a clarinet and a trumpet all playing a 261.626 Hz fundamental at 90 dB SPL in an anechoic chamber (so that there is no reverberation to complicate matters, the walls being lined with fully absorbent material). You would still hope to tell them apart, despite the common period and intensity.⁷ This image of instruments actually seems to propose rather static sounds with perfectly fixed frequency and amplitude, when in reality all sounds are time varying. And indeed, the changing character over time of features of the sound, including the envelope (time-function) of pitch and loudness, is an essential part of this differentiation. Applying knowledge of the spectrum of a sound, periodic sounds might be told apart by the way in which different harmonics fluctuate in amplitude, or more generally, for non-periodic sounds, by the time-varying energy exhibited in the spectrum. Periodic sounds still tend to begin with a noise-like transient, so different physical regimes can be passed through in succession. All of this

⁷This definition paraphrases the American Standards Association who placed timbre in negation as the attribute of sound which isn't pitch or loudness!

Filter Mathematics Part 1

Warning: the mathematics here is summarized for brevity. More technical detail is available in Roads [1996], Loy [2007b] and the signal-processing books referenced particularly in Chapters 3 and 4.

The next output sample is calculated as a linear combination of current and past inputs and feedback from past outputs. Conventionally, $x(n)$ is the value of the input and $y(n)$ is the value of the output at step n . Past values are indexed as $x(n-k)$ or $y(n-k)$ for k samples back in time. Note that to convert to time in seconds, rather than indexing directly in samples, the index would be multiplied by the sampling period $1/R$ (R is the sampling rate). But we do not need to worry about this purely on a sample processing level.

The (causal) linear time-invariant general filter equation is then

$$y(n) = \sum_{k=0}^M a(k)x(n-k) - \sum_{k=1}^N b(k)y(n-k) \quad (1.5)$$

where the $a(k)$ are $M+1$ feedforward coefficients and the $b(k)$ are N feedback coefficients. If all the feedback coefficients are zero or $N=0$, removing any feedback, the digital filter is called a **finite impulse response (FIR)** filter. If there is any feedback, energy can theoretically recirculate forever (even if in practice on a computer there is a limit based on the number precision) and the filter is called an **infinite impulse response (IIR)** filter. The impulse response describes what happens to a single sample of one followed by all zeroes (i.e., 1, 0, 0, ...) when passed through the filter.

More complex responses for general sequences are formed from many scaled and delayed copies of the impulse response through a process called convolution; but we will return to this in Chapter 4.

By the way, just as with feedback from microphone and mixer, if the recirculating energy is too great it may continue to build up, causing a blow-up – an unstable filter. Since they have no feedback, however, all FIR filters are stable.

The general filter equation can be more profoundly analyzed to explore the spectral consequences of the filter on a signal. The full mathematics of filters is only recommended for the mathematically literate reader. Nevertheless, we shall return to this in due course in Chapter 4, and try to say more about it intuitively, too.

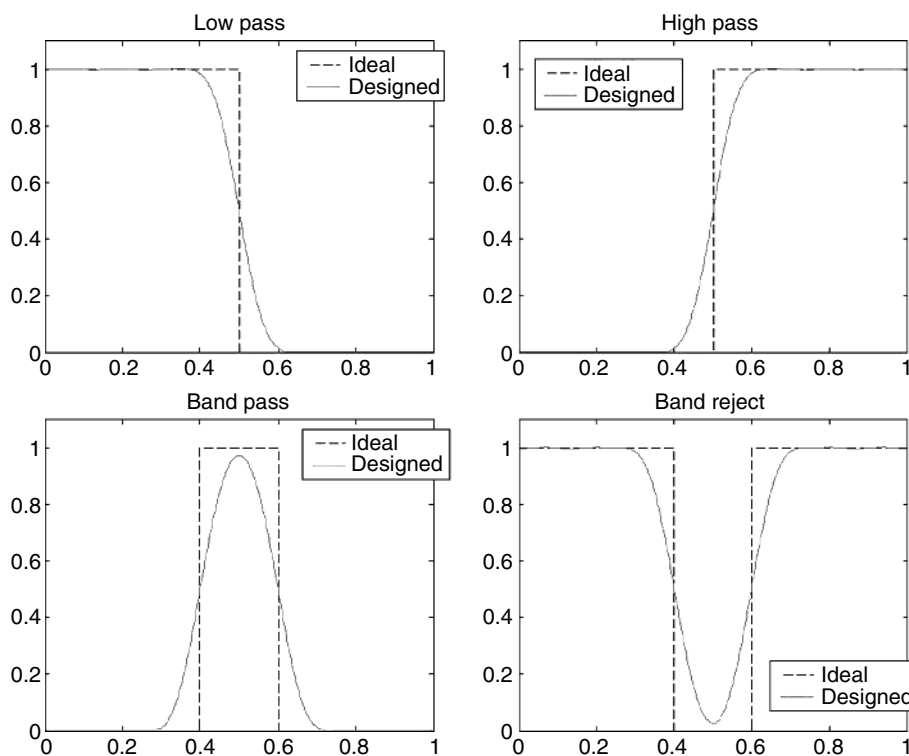


Figure 1.7 Four basic filter types. Magnitude spectra illustrate the frequency responses for four standard filter types. The band-reject filter is also often called a band-stop or notch filter. Each plot shows the frequency response for an ideal filter, and the actual frequency response for some real finite impulse response filters approximating the ideal specification. In this diagram, the x -axis is marked in terms of 'normalized frequency', with a reference of the Nyquist rate, so $0 = 0$ Hz and $1 =$ half the sampling rate. This makes the diagram independent of the sampling rate.

adds up to a complex sound-description that is the timbre of the sound, and requires a multi-dimensional property list, a particular model of sound analysis, to outline.

Psychological studies of timbre have investigated what properties are *most* important in differentiating sounds [Risset and Wessel, 1999; Beauchamp, 2007a]. The findings do not always agree, and are of course rather dependent on the stimuli (orchestral instruments often being the subject matter) [Bregman, 1990, pp. 122–126]. Whilst some features are peculiar to certain investigations, others do tend to recur, including the spectral centroid⁸ and the attack envelope, which correlate well with the physical properties of struck objects. Through experience, even instruments which have distinct qualities of sound in different

⁸In this case a measure of the centre of mass of the spectral energy averaged over time.

portions of the range (such as the clarinet's chalumeau, clarino and altissimo registers) are perceived as coming from a common source, no doubt assisted by additional spatial and visual cues. In electronic music, which allows many novel timbres, this viewpoint on physical production is suddenly absent. It is likely that evolved mechanisms play a large part in deciphering and categorizing new timbres for a listener [Windsor, 2000], though humans are also undoubtedly adaptable learners when exploring new territory.

Western classical music before the 20th century only really confronted timbre in terms of orchestration. The increased emphasis on dazzling instrumental effect in the music of Debussy, Stravinsky and their peers began to set timbre apart, and Schoenberg's daring musical line constructed from timbre changes ('Farben', the third movement of his *Five Orchestral Pieces* of 1909) brought timbre to centre stage. An overt treatment and hunt for timbral resources became a feature from the mid 20th century, becoming associated both with contemporary classical music (e.g., the sound masses of György Ligeti's *Atmosphères* from 1961) and with the explicit sound manipulation of electronic music (e.g., Pierre Schaeffer's 1948 *Étude aux Chemins de Fer* and Bernard Parmegiani's 1975 *De Natura Sonorum*). However, the expressive carrying potential of timbre has a healthy reality in the beating of partials of metallophones in the Indonesian Gamelan, the ornamental instructions of Japanese Gagaku and the husky flow of Australian Aboriginal music, to name but three examples which substantially predate Western music's own preoccupations.

1.2.8 Space

Another important aspect of sound, particularly in electronic art music and film music production, is space. We speak of **spatialization** as the act of rendering a sound in space. The perceptual side of the coin is **localization**, the perceived location of a sound for an observer. Most prosaically, acoustic instrumentalists (or mobile phone speakers!) are naturally heard as emitting sound from their current positions, but virtual sound sources can also be evoked by configurations of loudspeakers and more complicated rendering strategies. Environmental acoustics also play a part. Within enclosed rooms, reflections from the walls lead to reverberation alongside the sound traveling directly from the source to your ears. Reverberation can add to the sense of liveness of a sound, as well as obstructing some cues to localization. Architectural acoustics, surround-sound set-ups, compositional and mixing strategies in the treatment of space and more can all interact in the portrayal of space in music.

The site of presentation of music, whether delivered within an enclosed chamber of particular reverberant characteristics, or even outside or underwater, has always been recognized by musicians as a major practical factor in performance. Space on human scales is unavoidable in music, since with a speed of sound of around 340 meters per second in normal temperature and altitude conditions, the wavelengths of pressure waves become comparable to room sizes (20 Hz has a wavelength of 17 m!). Space has been employed as a compositional factor [Brant, 1998], from antiphonal practice in church music (such as J. S. Bach's *St. Matthew Passion* (1727) setting for two offset choir and orchestra groups), to

offstage musicians for timbral effect (trumpets, horns and percussion in Mahler's Second Symphony) and deliberate placement of musicians in space to make dense counterpoint more discernible (Henry Brant's *Antiphony I* (1953) for five orchestral groups). Recorded music, however, brings the issue of space to the fore at the point of playback; schemes must be devised to compensate for the potential unnaturalness of loudspeaker reproduction. In the extreme, with the challenge of evoking virtual acoustics, it is probably with electronic music that the issue of space has become most intensely a compositional concern in and of itself.

Aside from earlier investigations into stereo that date back as far as 1881, a brief history of spatial experimentation might mention Moholy Nagy's *Stage Scene – Loud Speaker* (1924–6) a loudspeaker placed on a bidirectional revolving table, an art project that anticipates Stockhausen's famous circular sound distribution method for the quadraphonic *Kontakte* (Contacts, 1958–1960). Fantasound, an adaptable multichannel configuration used for the film *Fantasia* in 1940 anticipated later surround-sound experiments in the film industry.⁹ The *Pupitre d'Espace* (Space Console, 1951) of Jacques Poullin allowed the live diffusion of *musique concrète* within a four-speaker set-up of front-left–front-right–rear–overhead, by a human controller gesturing within this special music stand's induction coils. January 1953 saw the creation of the first octophonic (eight-channel) piece, John Cage's *Williams Mix*, this four-minute-fifteen-second work only realized after a lengthy project of analog tape splicing from a 192-page score. Space has continued to exert fascination as an important aspect of electroacoustic music, from loudspeaker orchestras (the first being the Gmebaphone in 1973) to purpose-built venues. The latter include temporary demonstration housings such as the 425-speaker Philips Pavilion (1958) or the German Pavilion at the Osaka Expo in 1970, as well as new sound systems for theatres, cinemas and planetariums! Whilst quadraphonic sound was an unsuccessful commercial experiment in the 1970s, cinema standards, especially supported by the popularity of DVDs and computer games, have helped to bring at least 5.1 (left–centre–right, back-left–back-right and '.1' subwoofer) to many homes.

The psychoacoustics of spatial hearing and different multichannel formats will be investigated more thoroughly, starting in Section 2.5.

1.2.9 Patching, Signal Flow and Unit Generators

Dedicated items of equipment for particular audio tasks often come as black boxes, where the inner workings are concealed, but the expected inputs and outputs are carefully specified in the manual. Consider the more traditional recording studio, with many racks of esoteric gear lining the walls, and the problem of plugging one unit into another as amplified over all the available equipment. You may have seen dedicated patch bays in such studios which help to avoid running long wires everywhere. Imagine a behemoth of a modular analog synthesizer, perhaps with a 'Battleships'-like pin matrix, or more untidily, wires strung from one side of the beast to another to connect up its many modules. Even the sound from a

⁹Because of the limitations of the cinemas of the day, the full Fantasound release of *Fantasia* was only shown at a few venues, reproduced over many loudspeakers; later general release was mono.

humble electric guitar might be passed through a series of effects pedals prior to reaching the amp. It is clear that **patching** is central to electronic music, and may become arbitrarily complex the more components are available to be plugged together.

Modularity can be a healthy sign of careful design: a particular functionality is encapsulated in a single module. A module can just be considered as a black box, with the fine details of its working concealed from view. The module is still perfectly usable as long as the expected inputs and outputs are known and appropriate values are therefore passed in and extracted. This encapsulation of functionality is useful in software as in hardware, and carries across from the physical world of real wires to the virtual world. A network of boxes can itself be considered a single black box (see Figure 1.8). This is useful to gradually build up a more and more complex system, with each stage of the design carefully delineated.

In computer music, Max Mathews named useful basic modules in sound synthesis software **unit generators** [Mathews, 1969, p. 35]. A unit might be a sine generator or a noise generator, or act as a processing unit of some form. In each case, the nature of the inputs and outputs to that unit vary based on the functionality encapsulated. But each unit is a ready-made module, easily employed as a building block to create complex patches for musical tasks.

A **signal** can be generally defined as any transmission of information, and in computer music typically refers to a time-varying value. This might be the alternating current output by a microphone, or the succession of sample values obtained by an analog to digital convertor. Electronic music involves many signals being routed between different sources and destinations. Establishing how information propagates is crucial to designing a computer music system.

To depict this we can draw a block diagram, showing which units plug into which, but with a notion of **signal flow**. The set of units and their connections is a processing graph demonstrating how the system deals with information; that is, how signals flow through the system and what processes them on their way. The system must start somewhere and end up with some final output (or else nothing will be observed to happen!). Typically, there might be an input from the outside world (a microphone plugged into the ADC for the computer) and a final output back to the outside world (the DAC).

These signal flow networks can support multiple sampling rates in one graph. In general, there is no overall rule as to when changes occur in signals, so that values end up being updated at different rates, and the limiting case is a constant which remains fixed. You may see reference to different official rates in computer music systems, including **audio rate** (sometimes a-rate), corresponding to the sampling rate of a digital system and **control rate** (sometimes k-rate), typically being a slower and more efficient update rate for control signals. There are various forms of message which are singular data sent when cued at a particular time. There is even an initialization rate (i-rate), namely a constant starting value, as the constant limiting case.

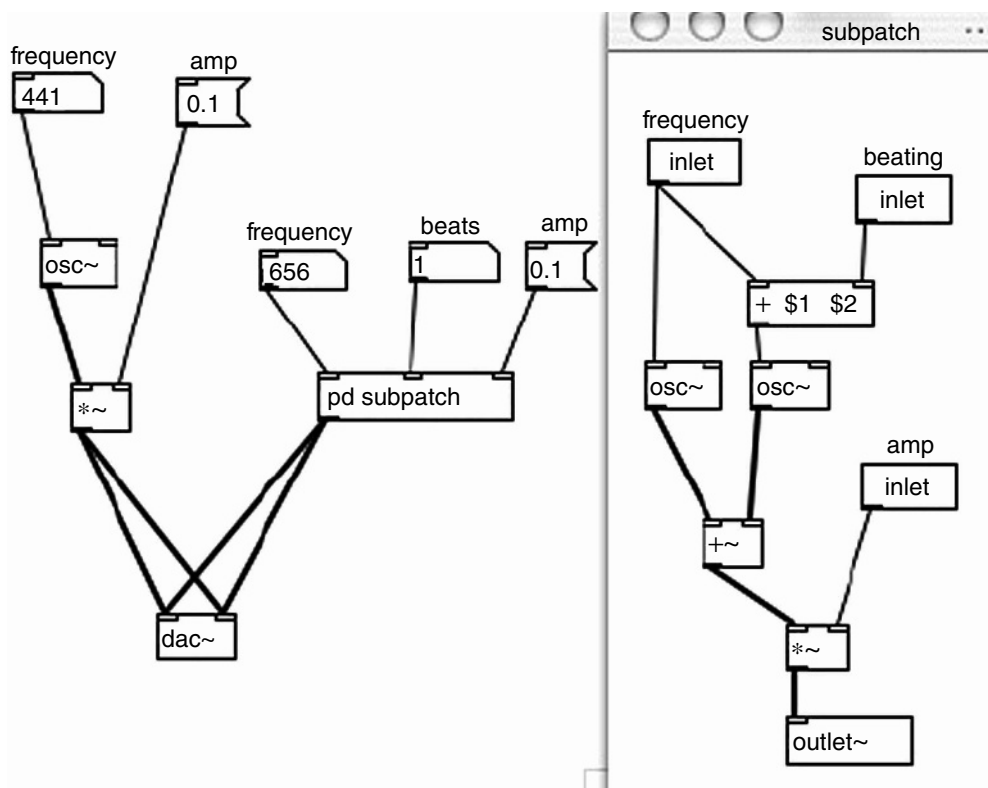


Figure 1.8 Patching in Pd. This screenshot from the Pd software package shows the boxes and wires of a patching network, with a subpatch abstraction (itself explicated in the right-hand window). The patch adds one sine oscillator at 440 Hz to a beating oscillator (two sines close together in frequency) at 656/657 Hz. Each box is specialized by the text inside it. The ~ sign and thicker wires in Pd mark out audio signal flow rather than control messages. Other systems might differentiate boxes more by the graphical icon.

The idea of units and patching is a standard paradigm, and the block diagrams describing how modules plug together have often formed the basis for graphical user interfaces (GUIs) in computer music software. Table 1.1 lists a selection of standard software that has this kind of front end; Figure 1.9 provides another graphical example, from Ross Bencina's AudioMulch software. Much other software also employs it, though in a more implicit manner; for instance, some text-based systems allow you to plug unit generators together by specifying the graph with program code, rather than drawing it out visually.

1.2.10 Computer Music Software

You have probably already been confronted by the dazzling and bewildering array of available computer music software. Software goes in and out of fashion, commercial imperatives

Table 1.1 A selection of modular patching environments for computer music.

Program	Availability	Description	Reference
'Max' paradigm family	Commercial and FOSS variants	Max/MSP, Pd, jMax visual programming languages	Puckette [2002]
Reaktor	Commercial (Native Instruments)	Toolkit to build your own software synthesizer	http://www.native-instruments.com
AudioMulch	Commercial (Ross Bencina)	Software for building your own signal processing network	http://www.audiomulch.com/
OpenMusic	FOSS (IRCAM)	Computer composition software with a graphical patching front end (Lisp under the surface)	http://sourceforge.net/projects/ircam-openmusic/
Logic environment	Commercial (Apple)	Processing graph can be customized from a patching interface (similar facilities exist in some other studio environments, such as Cubase, and the virtual patchcords are explicit on the back panel in Reason!)	http://www.apple.com/logicstudio/logicpro/
Tassman	Commercial (Applied Acoustics Systems)	Sound synthesis environment where an acoustic model is built up from components	http://www.applied-acoustics.com/tassman.htm
Bidule	Commercial (Plogue)	Patching environment for sound synthesis and processing	http://www.plogue.com/

drive a constantly shifting backdrop of operating systems, and even free software offers the perpetual opportunity for upgrades and also for discontinuation of a favorite package. Even a brief look at the history of computer systems will reveal an abundance of different machines with a panoply of software; museums and other archives are fighting a losing battle just to hold onto a fraction of these systems over time.

Fortunately, software doesn't change so fast as to make all advice untrustworthy, and there is much venerable and proven computer music software. If you are taking a course, your tutor may be expecting you to learn some particular packages, or you may be wondering what to try next. As you progress through this book you will see a variety of software featured. Since no one application can possibly cover all desirable goals, exploring a variety is healthy. So I present here two warnings, in the form of two aspects of software you should be aware of. I shall provide a few examples but not an exhaustive survey, and other packages crop up throughout this book.¹⁰

¹⁰If some of these are historical by the time you read this, as is very likely in this fast-paced world, at least I've provided a flavor of the era I live in!

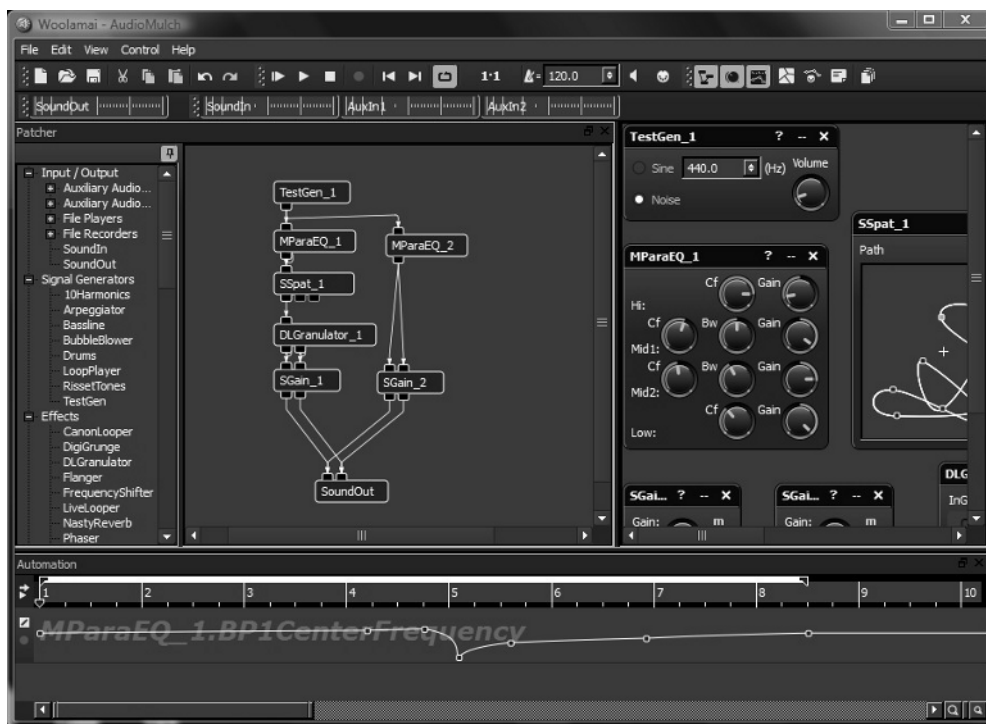


Figure 1.9 Patching in AudioMulch. This screenshot shows the AudioMulch 2 application, with a central patching view for plugging many generators and processing units together; control GUIs for the individual units can be seen on the right. Reproduced by permission of Ross Bencina, audiomulch.com.

First, some software is commercial and some is free. Neither implies superior quality; some of the most powerful sound synthesis environments are open-source software, for example, and so are free to use and extend (for example, Pd, Csound and SuperCollider). The abbreviations FOSS (for Free and Open-Source Software), and also FLOSS, with the addition of a Libre for good measure, often appear. On the other hand, a number of commercial packages are established in the music industry, and even if they have free clones or rivals, their staying power and market share is often related to being professional, reliable and well-documented products (examples are Pro Tools, Logic Audio, Cubase, Digital Performer, Sibelius and Finale). Demo versions (with limited use time or with disabled saving functionality) are often available to ‘try before you buy’. Whilst there are some standards for exchanging data between packages, there are many issues of compatibility between different ones, and commercial software often uses proprietary file formats and protocols which make it more difficult to exchange data.

Second, different software takes different amounts of effort to learn. This may seem a truism, but you must appreciate that it is not just due to problems of design in the software and its documentation. A more significant underlying reason is the tradeoff between the learning curve for a program and the degree of customization achievable with it. This means, essentially, that you can find programs that are straightforward to use because they have few options. They seem powerful at first, but quickly pale as you reach a point where you see the design limitations (examples might be drawn from many effects plug-ins). On the other hand, there are programs which require a great investment of your time, even to get quite limited results at first. Yet they significantly reward long-term work, eventually repaying your apprenticeship and allowing fundamental experimental investigations. This is really also just the choice between immediate results and deferred reward. Either type of software could eventually become a standard part of your armory: a sound editor or sequencer can be relatively easy to learn to use, and yet be a reliable tool for standard tasks; a complex sound synthesizer might provide interesting compositional stimulus for you each time you employ it.

The best advice is to learn a variety of software, and not just that which is easy to pick up. You tend to understand less about the underlying processes if you only use packages on the surface level, and deeper understanding is a positive contribution to computer music composition and research. Knowing a wider variety of tools also supports more informed choices when judging the needs of computer music projects. Ultimately, use whatever gets the job done quickly and well, and always bear in mind the artistic and scientific outcomes above the implementation; it is easy to be distracted by technical issues from real aims.¹¹

1.2.11 Programming and Computer Music

Programming is an essential skill for the computer musician who wants to investigate new sound worlds, and perhaps be first to set foot in them. It is always possible to rest comfortably behind the bleeding edge using whatever systems are made available by other explorers (some commercially minded). And much interesting music can still be made this way, even with facilities that the designers of software have overlooked or discounted (think of the glitch movement in electronica). Yet the truly interesting new possibilities, and the repurposing of technology to individual projects, ultimately require the facility to build new software.

No one would seriously claim that everything should be coded from the ground up; aside from some aesthetic tinkerers, building your own operating system before you make computer music is probably an indirection too far. It is typically good practice to make use of any existing code you can, to spend your time concentrating on the truly original aspects of your project. To assist this, there are many extension libraries for well-known general-purpose programming languages. For efficient audio signal processing code, C remains a popular choice. Yet there are many other computer music tasks, such as

¹¹Kyle Gann's rant and the comments following on from it are an enlightening read in this regard (see http://www.artsjournal.com/postclassic/2006/05/postsemester_rampage_electroni.html).

algorithmic composition, which can be profitably pursued using any favorite language, and are also supported by many third-party libraries. There are also some scientific modeling environments which are often used by researchers for prototyping, such as MATLAB and GNU Octave, again with plenty of third-party extensions.

There are specialized programming environments explicitly for computer music which might rest upon a standard language such as Java or Lisp, or even be new languages in their own right, even if they bear a family resemblance to prior work (for example, SuperCollider is a descendent of Smalltalk and C which also explores syntactical tricks from more esoteric languages [McCartney, 2002]). There are also scripting languages integrated into particular software, such as the scripting language available in Sibelius, the Nyquist language now integrated into the free Audacity audio editor, or the use of JavaScript in Max/MSP. Operating systems themselves often provide the facility to script instructions for applications, and may have extensive command-line facilities for scripting processes. The latter might be useful, for example, if building a computer-controlled installation which has to be started up by a gallery curator each day with the minimum of fuss (turning the power on) and automatically set itself up.

Table 1.2 lists a selection of some popular options for computer music programming at the time of writing. I have not attempted to collect together all the third-party libraries available, but will mention a few useful ones at other points in this book. There are many historically important computer music languages as well [Loy, 1989a; Roads, 1996; Lyon, 2002]. It should also be noted that many of these systems support arts computing beyond sound alone, having facilities for audio and visuals, and helpful functionality for interfacing to all types of controllers and information beyond the computer itself. In this situation, systems like Processing or Flash, which have increasing amounts of audio support, and may be very well suited to building web applications, can also be helpful.

Programming itself is not taught in this book, and I have chosen to provide pseudocode for algorithms rather than concentrate on any one language, because of the wide range of options available. It is not especially difficult to come to terms with the syntax of a particular language, but it can take a substantial investment of time to explore the associated libraries of available functions. In these cases, the best documentation is usually that which provides plenty of real examples, such as those with which major computer music systems like Max/MSP, Csound and SuperCollider are replete. These make the learning of such systems much more fun and immediate than the study of dry textbooks for general-purpose languages like C. Modern development environments compile code very quickly, or the languages are interpreted, so that the response to running a command can seem to be immediate. Allied to realtime audio synthesis, the facility for fast feedback is now the norm, and development usually has few of the frustrating waits that were historically associated with programming activity.

Table 1.2 A selection of popular computer music programming languages.

Program	Availability	Description	Reference
Pd	Mac/Linux/PC (FOSS)	Pure Data, a popular visual programming language for audio	Puckette [2007]
Max/MSP	Mac/PC (commercial, Cycling74)	Visual programming language with JavaScript scripting, a distant commercial cousin of Pd	Winkler [1998]; Zicarelli [2002]
SuperCollider	Mac/Linux/PC (FOSS)	Draws on Smalltalk and C, but is its own interpreted language. Optimized for realtime audio synthesis and smoothly combines algorithmic composition and synthesis	McCartney [2002]
Csound	Mac/Linux/PC (FOSS)	Contemporary descendent of the historic MusicN series. Old school syntax, but now with integrated Python scripting. Includes a rich catalogue of sound synthesis and analysis methods	Boulanger [2000]
Common Music	Mac/Linux/PC (FOSS)	LISP/Scheme-based, interpreted system for musical algorithm design supporting many output formats from MIDI to Csound	Taube [2004]
ChucK	Mac/Linux/PC (FOSS)	Novel computer music language exploring notions of concurrency, with the colorful Audicle programming environment	Wang [2008]
Impromptu	Mac (free)	Scheme-based, interpreted system which can act as an AudioUnit plug-in host	Sorensen [2005]

1.2.12 Representing Music on a Computer

I have already cautioned that there are many different musics in the world, and it would be surprising if ‘one sound fits all’ were true. Whilst common practice notation and 12-tone equal temperament have been very successful musical memes in terms of the volume of music dependent on them, they are not the last word on the representation of music. The interfaces of software often make rather constrained decisions about what music typically consists of, tied to the culture being serviced by the product, and this is another reason to have the ability to create or adapt your own software if need be. In general, it befits enlightened musicians to be as aware as possible of the assumptions underlying their musical practice. This critical facility may even lead to the recognition of genuine new musical possibilities.

Representation cannot be escaped; whenever creating music, we are forced to choose our theories and tools (because we get a choice: there is no one music theory, and no one best tool). But computers in particular are so stubborn in their insistence that every detail must

be specified that they bring home the need to create explicit well-formed theories of music in an unavoidable way. There can be no woolliness and appeals to irrational philosophy, at least in the operation of the computer itself (wherever decisions are deferred to human minds, well, here anything can happen!).

A prominent theme in electronic music is the ‘sound object’, an extended concept of musical materials, respecting the continuous transformation of sound in high-dimensional spaces of timbral description [Landy, 2007; Wishart, 1996]. The sound object can encompass a musical note, but also treats interconnected sequences of notes with portamento, environmental sounds, richly synthesized spectral complexes, and many other transformational possibilities.

Henkjan Honing has stated that there is no great justification for the note event as the founding concept in music cognition [Honing, 1993]; but here is a riposte from more than twenty years earlier, from Max Mathews:

The final principle for specifying sound sequences is the *note concept*. Sound exists as a continuous function of time starting at the beginning of a piece and extending to the end. We have chosen, for practical reasons, to chop this continuous sound into discrete pieces, called notes, each of which has a starting time and a duration time. This division is admittedly a restriction on the generality of sound synthesis, but one we are not brave enough to avoid. Needless to add, notes have been around for some time. [Mathews, 1969, p. 36]

The ease with which we slip into thinking of notes will recur in this book (soon, in the context of MIDI, for example). What is really going on is the choice of the timescale and sonic features for description. Music is a time-based artform, but quite what the best vantage point in time is may depend on who is doing the analysis, and much composition takes place ‘out of time’. Of course, ultimately in presentation music is bound by the speed with which the brain can process and handle the ‘perceptual present’. There are certain conveniences to the segmentation of music into events which may have great benefit from an evolutionary perspective. The way in which we comprehend complex musical textures is related to the way our marvelous auditory systems let us break down complex scenes in the environment, whether on the savannah or at cocktail parties [Bregman, 1990]. There are many theories as to how the information processing takes place and what is broken down when analyzing an auditory scene. Again, we shall come back to this.

The chief point for now is that we may extract musical information at a number of timescales and viewpoints. In a digital system, the lowest level consists of the samples themselves. Music in conventional understanding is a much higher-level process, whose entities operate over longer timescales; nevertheless, computer music gives us the opportunity to specify individual samples on the way to building up a larger-scale picture. It will be convenient at points to look at **blocks** of samples, say 64 at a time, or larger **frames**,

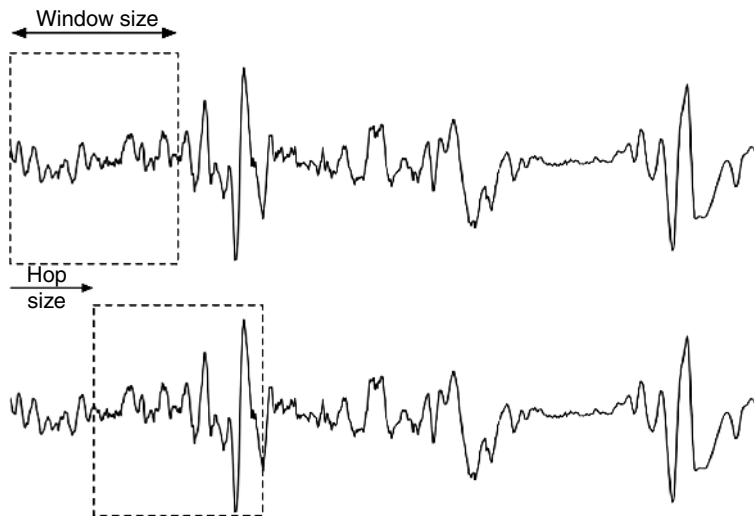


Figure 1.10 Sliding window. Two frames are separated by *hop size* samples; each frame manipulates the samples in a window (shown outlined by the dashed box). A single feature value might be obtained that says something about the variation of the signal in the window, or a whole set of features might be extracted at once.

perhaps windows containing 1024 samples. Looking at a group of samples at once can be convenient for making statistics which describe longer-term motion, all in aid of working up to the higher-level musical descriptors. We shall encounter processes aimed at extracting particular summary features for successive windows of samples. Figure 1.10 demonstrates how this might work with a sliding window; two successive frames are depicted, separated by a hop of a number of samples.

Converting from analog signals, through digital samples, to more abstract layers of description has been called the **signal-to-symbol problem** [Mataric, 2007, p. 73]. An example would be trying to backwards engineer the notes of a classical score given only the audio signal of a clarinet, a problem which seems fair enough on the surface, but proves rather more complicated in practice. It is often convenient to use computers to treat symbols at a high level of abstraction, but computer systems may include a number of levels of description, and complete systems will incorporate both real-world and virtual spaces. With all of this understood as operating with respect to some model of what music is, the representation problem is not a trivial one, and underlies any modeling efforts in computer music.

1.3 A WHIRLWIND HISTORY OF COMPUTER MUSIC

At the time of writing (2008), computer music is 58 years old; that is, if we count the popular tunes played live via the warning hooter on the CSIR Mk 1 computer in Australia in 1951 [Doornbusch, 2004].¹² You might wonder how the computer forced a sound out of the hooter; essentially, by blasting out impulses (a one and then zeroes) repeating at a desired frequency, with no low-pass reconstruction filter. In the same year, the first (analog) recording of computer music was made, of popular tunes performed by the Ferranti Mark I computer following a program by Christopher Strachey, by a BBC unit visiting the ‘electronic brain’ in Manchester [Fildes, 2008].

These early mainframe computers were room-sized behemoths of the era before integrated circuits and the marvels of miniaturization. The first real computers, in the modern sense of electronic programmable data processing machines, only date to the 1940s, though there are many precedents in mechanical technology, from the Jacquard Loom of 1801 to Charles Babbage’s plans for an Analytical Engine. Indeed, there are many musical precedents and anticipations of computer music, by people ranging from experimental music composers to mechanical instrument builders – sometimes the same individuals [Hugill, 2007]. As Moore [1990, p. 8] notes, ‘human imagination is the driving force behind human technology’. Famous figures in computer science had anticipated the consequences for the arts, from Ada Lovelace’s prescient remarks in 1843 on the Analytical Engine’s potential for automated composition, to Alan Turing’s theoretical and then practical involvement in the birth of computing. The history of the computer itself, then, is a prime driver of practical success, as what had only been imagined or theorized came to be engineered.

The year 1957 is a special point in our history. The first serious investigation of the potential of computer music is usually attributed to Max Mathews, sometimes dubbed the ‘father’ of computer music [Lyon, 2002, p. 21]. Having access to facilities at Bell Labs in 1957, and with the lenient and musically inquisitive research director John Pierce to support him, Mathews was instrumental in many computer music firsts. Rather than using an analog recording of a hooter, he was able to work with an early digital to analog convertor¹³ and directly write programs whose output could eventually be transferred to magnetic tape. The earliest recording from this process was the 20-second *In the Silver Scale* composed by Newman Guttman, premiered on 17 May 1957. The work immediately showed the potential of the computer by utilizing alternative tunings, rather than the standard equal-tempered scale. In order to create the piece, Mathews wrote the first computer music programming language, Music 1, the first of a whole series usually described as **Music N**, and whose distant and powerful descendants such as Csound are perfectly usable today. Along the way, Mathews originated the Unit Generator concept, was involved in early experiments in the synthesis of

¹²<http://www.csse.unimelb.edu.au/dept/about/csirac/music/>

¹³Perhaps the only set-up available in the world at that point, it was held by another company across the other side of town!

singing, and explored alternative interfaces such as graphical input and a radio conductor's baton.

From a few pioneers, engagement in computer music has rocketed in parallel with the massive reduction in space and costs of digital technology and electronic computers, and at the same time the marvelous acceleration of processor speeds. The computer music literature is replete with excitable citations of Moore's Law, as slow non-realtime rendering has given way to faster-than-realtime calculation. The power for significant realtime audio processing at CD quality has been available on standard home computers since around 1996. Whilst there are always ways to demand too much of whichever machine you have, it is no longer necessary to gain access to a few select institutions to explore computer music.

Really widely available and affordable home computers are traced to the beginning of the 1980s, era of the Commodore 64 and ZX Spectrum, though some musicians had begun early with the Kim-1 and its ilk in 1975. These early microcomputers had rather constrained sound chips, but this didn't stop the creation of some fantastic eight-bit music for computer games and the first tracker programs. Later revivals of those challenging compositional constraints are perhaps more familiar to some readers than the original, and there are thriving eight-bit music and demo scenes today. The take-up of the MIDI control protocol from 1982 led to many MIDI recording and generating computer music programs. By the 1990s, the typical home computer was powerful enough to run multichannel audio recording, processing and synthesis in realtime; there are also many precedents in the 1980s [Yavelow, 1989; Leider, 2004]. We shall survey these developments in Chapter 2.

Whilst there are many musical precedents, from Kraftwerk to hiphop, the explosion of electronic dance music with the rave era (1988 as the Second Summer of Love) saw frequent computer performances at raves. Many an Atari ST or Amiga was run on a dodgy electrical supply in a field, playing back new tracks via sequencer or tracker software! The now-familiar laptop performer became an increasingly common sight in the 1990s. By the new millennium many festivals seemed overrun with computational devices at centre stage, the operators often rather muted physical presences somehow making huge noises. But diversity is paramount in healthy musical scenes, and in a physical backlash there are now equally many alternative interface devices, handheld computers, self-built controllers and the like to offer plenty of movement potential, even where the sounds themselves are often so much larger than an acoustic musician alone could summon.

The computer became part of the arsenal of any popular musician and home recordist, alongside the computer music researchers and ostensibly serious composers. Conservatoire musical education is in no way a necessary qualification for computer music work; autodidact amateurs can easily find themselves become professionals, learning as they go (Autechre spring to mind). Whilst the grand *musique par ordinateur* institution IRCAM has been guilty at times of propagating a model separating engineers and composers, many of today's computer music pioneers are equally at home with technical issues in either domain. Though no one human being can be expert in all facets of a rapidly advancing field, musical

curiosity will drive many musicians deep into the heart of machines, whilst many supposed engineers have found their artistic expression fulfilled via computer music.

So we live in a time of unadulterated activity, with relatively portable, powerful and affordable machines (we'll get to mobile phones and personal digital assistants later, too). With so much resting on computers, whether they are centre stage or backroom devices, there is plenty to explore in this book.

1.4 SUMMARY

This chapter has set the scene for the book, covering issues of definition for the field of computer music, and providing some central examples. We have had a glimpse of the rich history of the field, which matches well that of advances in computers. Musicians were quick to exploit the potential of the new generalized calculating machines, and there are plenty of examples of early projects from the 1950s. Present-day activity in computer music is extensive and fast paced, enriched by the speed of communication of ideas and the realtime audio power of even portable machines. The chapter has also provided a quickstart guide to computer music. This was intended as an introduction to an introduction, a useful starting point; nevertheless, I cannot possibly cover in the space all aspects of further disciplines that impact on computer music, from acoustics to musicology. The coming chapters will probe much more into the varied and exciting technologies and projects within computer music as a field in itself.

1.5 EXERCISES

1. Write down your own definition for computer music. Extra groupwork: get a friend to make their own independent definition; then compare your work and discuss. Optional extra: if you have the opportunity, discuss your definition with at least one experienced musician and one experienced computer user. Was the musician *really* just experienced in pure music? Was the computer scientist only expert in computers? Perhaps boundaries are not as solid as you might think.
2. Write down a list of all the music software you can off the top of your head. Try to identify common functionality between different programs. Can you form a taxonomy of computer music software?
3. Create a sine tone in as many different ways as you can. Beyond trying various software packages, this might extend to finding an analog sine tone generator or tuning device You should try and produce different frequencies at different

amplitudes and phases. Be careful that you don't play anything at painful volumes, particularly over headphones, but probe the limits of your hearing. Can you hear a 16 kHz sine tone at both low and high amplitude?

4. Research the following three artists: Laurie Spiegel, Markus Popp (Oval) and George Lewis. What different roles have computers played in their work?
5. Calculate the root mean square amplitude for one cycle of a sine wave, for the cases where the period contains 2, 10 and 100 samples. Do the same for one cycle of a cosine wave. How do your results compare? Is there a limiting value as the number of samples increases?
6. You have one signal at -10 dB, and one at -6 dB (the reference level is 1). Add up their amplitudes. What is the final result in decibels?
7. Draw a patching diagram showing a white noise generator plugged into a low-pass filter, and from there to a DAC.
8. Draw a patching diagram showing two sine tone generators, the first with its frequency set to 440 Hz and its initial phase 0, and the second a 550 Hz tone with phase $\frac{\pi}{2}$. The output of the first sine generator is multiplied by a scaling factor of 0.5 and the second by 0.1, and the signals are then summed before going to a DAC.
9. Draw a patching diagram for the IIR filter which consists of feedforward coefficients $a(0)=0.5$ and $a(7)=0.2$, and feedback coefficient $b(1)=-0.1$. In order to represent the delays you will need a delay unit with a signal input and output, and a delay time control in samples. The filter should be processing input from an ADC and then sending the result to a DAC.
10. Use the Hamming derivation of aliasing to construct a diagram of the phenomena showing an original wave with frequency over Nyquist, and the alias that is actually heard to result. For convenience, you will probably want to work with normalized frequencies such that 1 is the sampling rate, and $1/2$ is Nyquist.

1.6

FURTHER READING

No book is an island, and some complementary materials are suggested at the close of each chapter in addition to the in-text references. Reading multiple perspectives on a topic will help to consolidate your knowledge.

Acoustics, Psychoacoustics and the Psychology of Music

- Campbell, M. and Greated, C. (1987). *The Musician's Guide to Acoustics*. Oxford University Press, Oxford.
- Cook, P. R. (ed) (1999). *Music, Cognition and Computerized Sound*. MIT Press, Cambridge, MA.
- Deutsch, D. (ed) (1999). *The Psychology of Music*, 2nd Edition. Academic Press, San Diego, CA.
- Everest, F. A. (2001). *The Master Handbook of Acoustics*, 4th Edition. McGraw-Hill, New York, NY.
- Hallam, S., Cross, I. and Thaut, M. (eds) (2009). *The Oxford Handbook of Music Psychology*. Oxford University Press, Oxford.
- Moore, B. C. J. (2004). *An Introduction to the Psychology of Hearing*, 5th Edition. Elsevier, London.
- Pierce, J. R. (1992). *The Science of Musical Sound*, Revised Edition. W. H. Freeman and Company, New York, NY.
- Roederer, J. G. (1995). *The Physics and Psychophysics of Music: An Introduction*, 3rd Edition. Springer Verlag, New York, NY.
- Sloboda, J. A. (1985). *The Musical Mind*. Oxford University Press, Oxford.
- Thompson, W. F. (2009). *Music, Thought, and Feeling: Understanding the Psychology of Music*. Oxford University Press, New York, NY.
- Yost, W. A. (2007). *Fundamentals of Hearing: An Introduction*, 5th Edition. Academic Press, Burlington, MA.

Mathematics and Music

- Benson, D. J. (2007). *Music: A Mathematical Offering*. Cambridge University Press, Cambridge. <http://www.maths.abdn.ac.uk/~bensondj/html/maths-music.html>
- Loy, G. (2007). *Musimathics* (Volumes 1 and 2). MIT Press, Cambridge, MA.

Digital Audio

- Pohlmann, K. C. (2005). *Principles of Digital Audio*, 5th Edition. McGraw-Hill, New York, NY.
- Watkinson, J. (2001). *The Art of Digital Audio*, 3rd Edition. Focal Press, Oxford.

Electronic Music History

- Chadabe, J. (1997). *Electric Sound: The Past and Promise of Electronic Music*. Prentice Hall, Englewood Cliffs, New Jersey.
- Collins, N. and d'Escriván, J. (eds) (2007). *The Cambridge Companion to Electronic Music*. Cambridge University Press, Cambridge.
- Cox, C. and Warner, D. (eds) (2004). *Audio Culture: Readings in Modern Music*. Continuum, London and New York.
- Emmerson, S. (2007). *Living Electronic Music*. Ashgate, Aldershot, Hampshire.
- Holmes, T. (2008). *Electronic and Experimental Music*, 3rd Edition. Routledge, New York, NY.
- Manning, P. (2004). *Electronic and Computer Music*. Oxford University Press, Oxford.
- Norman, K. (2004). *Sounding Art: Eight Literary Excursions Through Electronic Music*. Ashgate, Aldershot, Hampshire.
- Schrader, B. (1982). *Introduction to Electro-Acoustic Music*. Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Shapiro, P. (ed) (2000). *Modulations. A History of Electronic Music: Throbbing Words on Sound*. Distributed Art Publishers, Inc., New York, NY.

Computer Music Textbooks and Reference Texts

- Brown, A. (2007). *Computers in Music Education*. Routledge, New York, NY.
- Dean, R. (ed) (2009). *The Oxford Handbook of Computer Music*. Oxford University Press, New York, NY.
- Dodge, C. and Jerse, T. (1997). *Computer Music Synthesis, Composition, and Performance*. Schirmer Books, New York, NY.
- Hugill, A. (2008). *The Digital Musician*. Routledge, New York, NY.
- Lincoln, H. B. (ed) (1970). *The Computer and Music*. Cornell University Press, Ithaca, NY.
- Mathews, M. V. (1969). *The Technology of Computer Music*. MIT Press, Cambridge, MA.
- Moore, F. R. (1990). *Elements of Computer Music*. P T R Prentice Hall, Englewood Cliffs, NJ.
- Pope, S. T. (2008). *The Big MAT Book: Courseware for Audio and Multimedia Engineering*. Extensive online resources at <http://heaveneverywhere.com/TheBigMATBook/>

Puckette, M. S. (2007). *The Theory and Technique of Computer Music*. World Scientific Publishing Co., Inc., Hackensack, NJ. Available from <http://crca.ucsd.edu/~msp/techniques.htm>

Roads, C. (1996). *The Computer Music Tutorial*. MIT Press, Cambridge, MA.

Roads, C. and Strawn, J. (eds) (1985). *Foundations of Computer Music*. MIT Press, Cambridge, MA.

Roads, C. (ed) (1989). *The Music Machine*. MIT Press, Cambridge, MA.