EDITED BY

ROGER T. DEAN

The Oxford Handbook *of* COMPUTER MUSIC

THE OXFORD HANDBOOK OF COMPUTER MUSIC

THE OXFORD HANDBOOK OF

COMPUTER MUSIC

Edited by

ROGER T. DEAN

OXFORD
UNIVERSITY PRESS

2009

OXFORD
UNIVERSITY PRESS

Oxford University Press, Inc., publishes works that further
Oxford University's objective of excellence
in research, scholarship, and education.

Oxford New York
Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in
Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan Poland Portugal Singapore
South Korea Switzerland Thailand Turkey Ukraine Vietnam

Published by Oxford University Press, Inc.
198 Madison Avenue, New York, New York 10016

www.oup.com

Library of Congress Cataloging-in-Publication Data
The Oxford handbook of computer music / edited by Roger T. Dean.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-19-533161-5
1. Computer music—History and criticism. I. Dean, R. T.
ML1380.O9 2009
786.7'6—dc22 2008046594

1 3 5 7 9 8 6 4 2
Printed in the United States of America
on acid-free paper

Contents

This page intentionally left blank

Contributors

Freya Bailes, University of Western Sydney
Michael Casey, Dartmouth College
Nick Collins, University of Sussex
Roger T. Dean, University of Western Sydney
Paul Doornbusch, Victoria University, Wellington
Alan Dorin, Monash University
Alice Eldridge, Monash University
Simon Emmerson, De Montfort University
James Harley, University of Guelph
Douglas Keislar, Berkeley
Leigh Landy, De Montfort University
Peter Lennox, University of Derby
George E. Lewis, Columbia University
Peter Manning, University of Durham
Jon McCormack, Monash University
Peter McIlwain, Monash University
Daniel Müllensiefen, Goldsmiths, University of London
Pauline Oliveros, Rensselaer Polytechnic Institute
Garth Paine, University of Western Sydney
Marcus T. Pearce, Goldsmiths, University of London
Tim Perkis, Berkeley
Palmyre Pierroux, University of Oslo
Jøran Rudi, University of Oslo
Noam Sagiv, Brunel University
Wayne Siegel, Royal Academy of Music, Aårhus
Mary Simoni, University of Michigan
Hazel Smith, University of Western Sydney
Atau Tanaka, University of Newcastle
Geraint A. Wiggins, Goldsmiths, University of London
Trevor Wishart, Durham University
David Worrall, University of Canberra

This page intentionally left blank

THE OXFORD HANDBOOK OF COMPUTER MUSIC

This page intentionally left blank

CHAPTER 1

INTRODUCTION: THE MANY FUTURES OF COMPUTER MUSIC

ROGER T. DEAN

COMPUTER music offers possibilities for music-making that can hardly (if at all) be achieved through other means. These possibilities commune between real-time creation in improvisation or other forms of interactive performance, production of scores for others to perform, and acousmatic composition. Broadly, we use *acousmatic* to refer to pre-fixed digital sound structures ready for acoustic diffusion through loudspeakers without performers energizing conventional musical instruments to make sounds.

In this brief introduction, I give some perspectives on the scope and futures of computer music, indicating how the topics are addressed within the book. Computer music has passed its 50th anniversary and is part of a slightly longer tradition of electroacoustic music. This book provides a broad introduction to the whole electroacoustic field and its history (see part I and the appendix in particular), but its explicit emphasis is on computer music in the period since the 1980s during which the Yamaha DX7 synthesizer and the availability of desktop (and later laptop) computers at prices individuals could afford meant that the practice of computer music was no longer restricted to those who could access a mainframe computer. Many composers and improvisers, like myself, first gained access to computer music opportunities in this period of the 1980s. Hence, the breadth of activity expanded until the present in part because the field of participants widened and in part because of the rapid acceleration in central processing unit (CPU) speed and therefore of the power of computers for real-time sound generation. Chapter 2, by Douglas Keislar, and the appendix, by Paul Doornbusch, provide interesting timelines of these issues of CPU speed.

1. The Appeal of Computer Music

A. For Musicians

For many years, it seemed as if computer music could only usefully be the domain of the composer willing to await the generation of the sounds programmed (as described in chapters of part I). Such a composer can be said to operate "out-of-time," in contrast to those who use computers in "real-time" performance. But, these out-of-time composers also used the computer to assist the generation of scores for live instrumental performers (see chapter 5) as well as for the making of acousmatic sound. One of the appeals was the apparent potential of computers to generate any conceivable sound and to realize performances of precision or complexity not feasible as a human performance. Many parts of the book address computer music composition, notably chapters 5, 7, and 9, by James Harley, Trevor Wishart, and Simon Emmerson, respectively. Possibly the most influential composer within computer music was Iannis Xenakis, and Harley discusses his works, juxtaposing them with contributions from Karlheinz Stockhausen and other pioneers. The long-standing prejudice that electronic music did not sound sufficiently "human" was progressively overcome; furthermore, computational and perceptual analyses of what those prejudices identified (e.g., slowing of tempi at phrase boundaries, relationships between pitch and performed intensity) could then be applied in software for composition and its realization, such as Director Musices. So when desired, aspects of this particular range of human features could appear in computer music. The realization of computer music scores is discussed here particularly by Emmerson and by Douglas Keislar (chapter 2).

However, some remarkably powerful real-time performing vehicles, such as the Music Mouse, appeared along with the Macintosh computer in the '80s; by the '90s, real-time midi-manipulation (e.g., using the software MAX on such computers) and later digital sound manipulation (using MSP or many other platforms) became widespread, fluent, and stable. One of the appeals of computers in real-time music-making is the possibility that the computer can itself enter a dialogue with other musicians, as discussed in chapters 6, 18, 19, and 20, and in chapters 8, 21, and 22 by Tim Perkis, George Lewis, and Pauline Oliveros, respectively. Another is that the computer can generate or manipulate events long after some initiation point, and that this can be done either in a way that the performers control or without their foreknowledge of the nature of the impending events.

Some contemporary opportunities in computer music include "soundspotting" (see chapter 20 by Michael Casey), in which rapid identification of features of incoming streams of sound is used to find related features in stored databases of other sounds so that the stored materials can be used in performance in a variety of controlled and performable ways. Such principles of relating different materials can operate in real-time intermedia performance also and are among the topics discussed by Nick Collins, in chapter 17 on laptop music-making; by Jon McCormack and colleagues, in chapter 18's context of A-life (artificial life) and generative approaches to computer music; and by Noam Sagiv and colleagues in chapter 15. In chapter 10, Wayne Siegel considers dance and computer music, on the basis of long experience, and other aspects of the conversion of bodily or other gestures into computer music are considered by Garth Paine (chapter 11) and by Atau Tanaka (chapter 12).

Although all the chapters in this book operate at a sophisticated technical level that should not offend the sensibility of a professional computer music composer, the book is not focused on computer music techniques, software, and so on. It is about computer music itself; thus, it is throughout intended to stimulate the listener at large, especially those with modest prior experience of the work.

B. For Listeners and Users

Is computer music just another form of music, like Balinese music or Western classical music, or does it have some particular appeals or difficulties? As mentioned, it is unlike all previous musics in at least one respect: its capacity to generate and utilize, in principle, any sound. For most people even now, computer music thus presents unfamiliar components, articulated into meaning structures that need to be penetrated or, according to your perspective, that the listener is free to envisage. Broadly, there is similarity between the process of an Indian classical musician becoming engaged with the music of John Coltrane and of someone approaching computer music from a position of unfamiliarity. What is required is the capacity to recognize sonic features and their recurrence, since music is one of the most repetitive of the temporal arts, and then to construct meaning therefrom.

As Freya Bailes and I have shown recently (chapter 23), most listeners have a remarkable ability both to recognize and to co-relate computer-manipulated sounds. For example, we found that they identified close connections between speech sounds, manipulated speech sounds, and what we call NoiseSpeech, which is manipulated so extensively there are no remaining detectable phonemes: all these sounds are readily perceived as speechlike. Speech is characterized by vast rates of change in spectral quality compared with classical music or even with computer music at large. Yet, we even found that noise on which an invariant set of speech formants is superimposed is still heard as speechlike and qualifies as NoiseSpeech. Identifying sonic continuities and categories is an important facility for the listener seeking to gain meaning from their sonic exposure.

Such listener ability for construction of semiotic fields is useful in our approach to the long tradition of computer music using the voice, as discussed by Hazel Smith in chapter 14. It will also be useful in the more practical aspects of sonification, the representation of data for informational purposes of more objective kinds, as discussed by David Worrall in chapter 16. As he notes, an extension of these informational intents provides ideas by which the traditions of sonification and of computer music can interact.

Even given these widespread capacities to hear structure and extract meaning from sound, our involvement and enjoyment of computer music is necessarily influenced by our cultural and educational experience. Thus, Mary Simoni discusses gender discrimination within and about the music (chapter 24), and Jøran Rudi and Palmyre Pierroux analyze current processes in computer music education, with particular reference to their decade-long experience with interactive school study in Norway (chapter 26). Leigh Landy provides an optimistic outlook on the future for creation and appreciation of computer music and does so in the broadest possible artistic and sociopolitical contexts of digital culture (chapter 25).

2. Some Editorial Principles and Practices in This Book

I have invited (and cajoled) authors always from the ranks of those highly qualified to write on their topics, but also, in a few cases, in such a way as to create interaction among them. Thus, we have an expert on the psychology of synesthesia contributing to an article on algorithmic intermedia processes; one on the processes of imaging and imagining music joining a discussion of empirical approaches to the perception of computer-generated sound; and several contributions involving exchanges between computer science, modeling, generative algorithms, A-life, and music. Chapters also address sociocultural and educational issues. The authors originate from most parts of the world in which computer music has been important (notably North America, Europe, Australasia, and Asia), with the exception of South America.

As some of the authors may have lived to regret, I have interacted closely with them: first in providing some specific suggestions on which to base the range of ideas they cover and then in proposing editorial changes and additions once a draft was received. The authors have also had to put up in some cases with incessant pressure from me such that all articles could be completed reasonably efficiently. Timely book production has also been a consistent objective of our enthusiastic and supportive commissioning editor at Oxford University Press, Norm Hirschy.

I was quite clear in my approach to several authors, such as those writing in the opening part, that there would and should be overlap in the topics they addressed with those of other chapters. By seeking distinct approaches from these authors, I am happy that we achieved complementarity in the descriptions of individual events or developments. Thus, Keislar provides a perceptively conceptualized analysis of the development of computer music in the broad context of music at large; this is complemented by the "machine" emphasis I requested of Doornbusch and the "software" emphasis I asked of Peter Manning. I take responsibility for the points of overlap, of which I edited a few (with the authors' approval), cross-referenced some (as editor, I added several of the cross-references in the book), and purposely retained several more.

Another set of tangential and occasionally overlapping views are provided by the four personal statements (in the two sections titled "Sounding Out"), which complement the more conventional thematic review chapters. In inviting these informal statements, I indicated that I was seeking some expression of the most pressing thoughts about computer music these eminent musicians were digesting and developing at the time. They were given complete freedom regarding minimum length or breadth of topics to be addressed. I think you will agree the results are a distinctive and stimulating component of the book.

I would also like to thank Paul Doornbusch for his unstinting work on the chronology in the appendix, which I believed should be a component of such a book but which I anticipated researching and compiling myself, with some trepidation. I was very glad that he agreed to develop this into what I consider a useful and informative document. No chronology can be complete, and every chronology in an area of interest should be seen as challengeable. Paul's is already the most substantial and probably the most balanced currently available, and it is hoped he can develop it as a continuing resource.

3. Outlook

My purpose in editing this book has been to facilitate access to computer music for the listener and for the future and to bring together the range of compositional, theoretical, and practical issues that a practitioner can always benefit from considering. We are at a point in the history of this endeavor at which the opportunities are endless, and in most cases, one can actually readily envisage a means to fulfill them (the fulfillment of course may take immense effort). This has not always been the case in music composition and improvisation, and I hope some of the excitement of this moment will come across to our readers. If so, my contributors deserve the highest credit. You will not have great difficulty in finding substantial works by all the composers and improvisers discussed in the book, legitimately available without charge on the Internet, or finding examples of all the creative techniques and technologies described. Again, such an opportunity has not always existed in relation to music. I hope you enjoy the exploration the book encourages, and that it leads you to experience future performances and the many high-quality commercial recordings now available.

This page intentionally left blank

PART I

SOME HISTORIES OF COMPUTER MUSIC AND ITS TECHNOLOGIES

CHAPTER 2

A HISTORICAL VIEW OF COMPUTER MUSIC TECHNOLOGY

DOUGLAS KEISLAR

A single chapter on the history of computer music technology can hardly do justice to its topic. The most comprehensive volume on computer music techniques (Roads 1996a) was billed as a tutorial yet occupied 1,200 pages, a significant number of which examined the history and antecedents of these techniques. Recent book-length historical treatments of electronic music, encompassing computer music, include those by Chadabe (1997) and Manning (2004). Accordingly, the present chapter necessarily gives but a glimpse at some of the major mileposts. In lieu of extensive detail, it proffers a conceptual framework, a way of thinking about this multifarious field.

We start by discussing terminology. The term *computer music* arose in the context of 20th-century developments in Western art music. In the 20th century, the very term *music*, formerly understood to be straightforward and unambiguous, was subjected to stretching and questioning on a number of fronts owing to new musical directions such as serialism, aleatoric composition, percussion-only works, noise, and electronic sound. The compound noun *computer music* has its ambiguities, and it substantially overlaps with other terms such as *electronic music* and *electroacoustic music*. Difficulties with all these terms have been discussed by Manning (2004, pp. 403–404) and Landy (2007, pp. 9–17), among others. We can distinguish two main usages of the term computer music: (1) a musical genre or category, analogous to the symphony, jazz combo, and the like, in which the computer plays a part in composition, performance, or sonic realization; and (2) a technical discipline, analogous to computer graphics, that encompasses many aspects of the computer's use in applications related to music. Landy's discourse is not unusual in confining the scope of the term computer music to the first meaning. The present book also is oriented primarily toward this first definition, but it is informative to consider the second. Technical capabilities often stimulate aesthetic directions and vice versa. We focus here on technologies that directly enhance musical creation (as opposed to music scholarship, education, commerce, etc., as well as scientific endeavors), but as we shall see, even this limited scope is ultimately broader than might be expected.

The specific technical fields falling under the rubric of computer music can be gleaned by reading publications such as the *Proceedings of the International Computer Music Conference, Computer Music Journal*, and the aforementioned textbook by Roads (1996a). Areas relevant to computer music include not only music composition and computer science but also music theory, music representation, digital signal processing, acoustics, psychoacoustics, and cognitive science, to name but a few. Writers such as Pope (1994) have proposed taxonomies of the field. The keyword index at the ElectroAcoustic Resource site (http://www.ears.dmu.ac.uk/), also published in Landy (2007), hierarchically lists various relevant disciplines of study.

Kassler and Howe (1980) briefly characterized computer music technologies as replacements for conventional human musical activities, an approach akin to the more general writings of the media theorist Marshall McLuhan (1964). The next section of this chapter charts a course influenced by both these sources. Although it treats musical technologies prior to the computer, it forms an essential backdrop to the ensuing discussion, because its analytical perspectives and terminology reappear throughout the chapter. After that section, we examine the development of the computer as a musical instrument, broadly speaking, and then the means by which human musicians have operated this "instrument." Next, we consider the computer as a musician itself, whether composer or performer. The chapter concludes with a synopsis of some trends in the 20th and 21st centuries.

1. Antecedents: Abstraction, Disjunction, and Proliferation in Music Technology

Music-making involves a chain or network of relationships among musicians, musical technologies, and listeners. The historical development of music technology can be broadly viewed as a chronological series of abstractions and disjunctions that affect these linkages, as shown in table 2.1. Some of the abstractions represent instances of McLuhan's "extensions of man," in which a technology extends the reach or power of a human faculty, and some of the disjunctions represent instances of his "self-amputations," in which the technology replaces the use of a human faculty. A disjunction (i.e., decoupling) may permit the interjection of a new intermediary between the separated entities. Likewise, a pair of disjunctions around an entity in a chain may permit its disintermediation (i.e., bypass) or elimination, typically by the conjunction of the two entities it formerly separated. Either case—intermediation or disintermediation—may effect a proliferation of some capability or some musical feature. The proliferation sometimes results from a one-to-many mapping. Changes in music technology can clearly alter musicians' interactions with their instruments, but such changes may also profoundly affect musicians' interactions with each other and may transform music itself.

The prototypical music technology was the ancient musical instrument, which in the case of pitched instruments served as an abstraction of the singing voice. Advancement of technology led to instruments that were increasingly removed from human physiology, such as keyboard instruments, whose progenitor, the hydraulis, was invented in the 3rd century B.C.E. With keyboards, the controller's separation from the sound generators allows the latter to consist of pipes, strings, bars, or whatnot: there is a one-to-many mapping from performance technique to timbre. Also, the keyboard's pitch representation replaces the linear frequency distribution occurring along a pipe or string with logarithmic frequency, which corresponds better to human perception. These mappings were precursors to the arbitrary mappings enabled by computer technology. For that matter, one can consider the lever, central to the keyboard mechanism, as a prototype of mapping.

Self-playing mechanized musical instruments represent important predecessors of computer music. They effect a disintermediation of the performer; the "score" becomes directly conjoined to the instrument. The ancient Greeks had the technological means to construct such automata and may have done so. The first description of an automated instrument, an organ, is found in *The Book of Ingenious Devices* by the Banú Músà brothers (Hill 1979). These scholars worked in Baghdad's House of Wisdom in the 9th century C.E.—as did the mathematician al-Khwārizmī, from whose name the term *algorithm* derives. Automatic musical instruments of all sorts appeared with increasing frequency in Europe during the 18th and 19th centuries. The quintessential composer for an automated instrument was Conlon Nancarrow (1912–1997), whose studies for player piano exhibit a fascination with transcending the physiological and cognitive limitations of human pianists. This liberation of the composer from the constraints of performers' abilities figures prominently as a recurrent theme in music technology, particularly in computer music.

Analog electronic instruments were direct ancestors of computer music technology. The electronic musical instrument can be viewed as an abstraction of the acoustic instrument, one that can further decouple the control interface from the sound generator, permitting a proliferation of timbres as well as of possible

Table 2.1 **An interpretive summary of some major developments in music technology prior to the computer**

Technology	Introduced in	Added role	Abstraction	Disjunction	Proliferation	One-to-many mapping
Musical instrument	Prehistory	Instrumentalist, instrument builder	of voice	of sound production from body	of sounds: increased range of timbre, pitch, duration, loudness	of performer to sound generators (in some instruments)
Keyboard	Antiquity, Middle Ages			of control mechanism from sound generators (pipes, strings, etc.)	of timbres; of simultaneous pitches; increased range of duration and loudness (organ)	of performance skill to instruments and timbres
Music notation	Antiquity (symbolic); Middle Ages (graphic, with pitch axis)	Composer	of performance	of musical conception from sonic realization (temporally); of composer from instrument (performer is intermediary)	of the composition across space and time; of the composition's complexity, texture (polyphony), and duration	of one musician (composer) to many (performers); of one composition to many performances of it
—	—	Conductor (an outcome of notation-enabled complexity)	(conductor is an abstraction of the performer, a meta-performer)	of composer from performer (conductor is new intermediary); of performers' parts from the conductor's score	of parts (texture)	of one performer's (the conductor's) gestures to multiple voices or instruments
Mechanically automated musical instruments	Middle Ages, becoming more		of performer	of performer from score and instrument, which are now directly conjoined and the	of identical performances across time and (later, with mass production) across space; increased ranges of	

	common in 18th and 19th centuries			performer eliminated	musical parameters, sometimes exceeding human limitations	
Sound recording	Late 19th to early 20th century	Sound engineer	(a "concretion" of the score; captures the concrete performance rather than the abstract composition)	of performer from audience; of performance in time (editing); in musique concrète, elimination of performer and instrument	of identical performances across time and space	of performer to multiple audiences
Electrical transmission of sound (e.g., broadcast radio)	Early 20th century		of acoustical transmission	of performer from audience	of performance across space	of performer to multiple audiences
Electronic musical instruments	20th century	With synthesizer, composer or performer can become a builder of virtual instruments (patches)	of acoustic musical instruments	of control mechanism from sound generator; in theremin, a complete physical decoupling of performer from instrument	of timbres; of possible controllers; increased ranges of all musical parameters	
Electronic sound processing (reverb and spatialization)	20th century		of room	of sound source from original position and space; of listener from physical space		

The fourth row does not present a new technology, only another human role that resulted from the technology in the previous row.

controllers. Analog electronic instruments have a rich history in the 20th century (Rhea 1972, Roads 1996b, Chadabe 1997, Davies 2001). Two particularly noteworthy electrical devices were Thaddeus Cahill's colossal telharmonium and Lev Termen's theremin, first demonstrated publicly in 1906 and 1920, respectively. The telharmonium's sound was created by tonewheels (as in the later Hammond organ) and transmitted over telephone wires to listeners. This distribution mechanism foreshadowed Internet "radio" by virtue of having pre-dated broadcast radio. The theremin was notable for achieving the first tactile disjunction of the performer from the instrument. The theremin player moved his or her hands in the air without touching the instrument. However, this physical decoupling was not accompanied by an arbitrary mapping of gesture to sonic result; the distance of each hand from one of two antennae tightly controlled the frequency or amplitude, respectively, of a monotimbral signal.

What we now think of as analog sound synthesis was originally accomplished in the electronic music studios of the 1950s using custom collections of signal-processing modules such as oscillators, filters, and modulators. In the mid-1960s, Robert Moog and Donald Buchla each introduced modular analog synthesizers. In such instruments, the previously separate modules are housed together and interconnected using voltage control, in which one module's output modifies an input parameter of another module. (As discussed in the section "Digital Sound Synthesis," this notion of interconnected units had already been introduced in computer music software, albeit in a more abstract form and with sound generation that was much too slow for live performance.) The analog sequencer that appeared in the 1960s as a synthesizer module allowed one to store a repeatable sequence of control voltages. These could, for example, specify the pitches of a series of notes; therefore, the sequencer functioned analogously to the player piano roll, conjoining the "score" to the instrument and disintermediating the performer. The modular synthesizer's configurability lets the composer or performer construct new virtual instruments (synthesis "patches"), conceptually disintermediating the instrument builder by assuming that role. Similarly, one can consider the designer of the synthesizer itself to be a builder of meta-instruments.

One of the most dramatic of all technologies affecting music is sound recording. Recording made its debut with Édouard-Léon Scott de Martinville's phonautograph, patented in 1857; however, this device lacked a corresponding playback mechanism. In 1877, Thomas Edison invented the first practical technology for both recording and playback: the phonograph. Magnetic wire recorders were invented in 1898, and the Magnetophon tape recorder appeared in 1935. Phonograph disks, wire recorders, and tape recorders were all used in composition before 1950. Sound recording formed the sine qua non of musique concrète, the school of composition based on assemblage of recorded sounds, which Pierre Schaeffer (1910–1995) pioneered in his 1948 composition *Étude aux Chemins de Fer.*

"Tape music" disintermediates both the performer and the musical instrument. Working directly with sound, alone in a studio and unconcerned with performers' interpretation or skills, the composer becomes more like the visual artist. The impact of recording on music composition was similar to that of photography on visual art. However, the visual arts had already been primarily representational of the physical world; the camera as the ultimate representational tool stimulated painters to explore increasingly nonliteral representation and ultimately abstraction. By contrast, music had always been largely abstract; sound-recording technology introduced literal representation of the physical world. Music gained the ability to play with levels of allusion to this world, to juxtapose fragments of it, and to transform them. With tape recording, time itself could be disjoined and rejoined in an arbitrary fashion. In conjunction with electronic processing, recording technology endowed the already-rich timbres of the physical world with the new trait of plasticity.

Moreover, in conjunction with sound synthesis, recording technology gave the composer a new degree of control over the parameters of music. Whereas the musique concrète of midcentury Paris focused on sculpting sound from real-world sources, the contemporaneous *elektronische Musik* championed by Karlheinz Stockhausen (1928–2007) in Köln (Cologne) focused on constructing sound from the raw building blocks of acoustics, such as sine waves and noise. The frequencies, durations, and amplitudes of sounds could all be arbitrarily controlled and combined in ways not possible with traditional instruments. An interest in this degree of control stemmed naturally from the concerns of the serialist school of composition that Arnold Schoenberg (1874–1951) had founded earlier in the century.

Space also became a parameter for manipulation, through the use of multichannel playback to position a sound in the perceived space according to its level in each channel. A number of composers used this spatialization technique in the 1950s. It was put to dramatic effect by Stockhausen, who during the composition of *Kontakte* (1960) projected sounds from a rotating speaker that he recorded quadraphonically. The result is a dynamic swirling of sound around the audience, who listens to the recording while seated amid four loudspeakers. The midcentury composers also used artificial reverberators. Reverberation and spatialization techniques tend to disjoin sounds perceptually from both the space in which they were originally recorded and the space where the recording is heard. Room acoustics may represent the primary form of natural (acoustic) signal transformation, of which these specific electronic techniques are an abstraction (as suggested in table 2.1), but more generally, electronic signal processing leads to a proliferation of possible sonic transformations, most of which involve effects other than spatial imagery.

The technology of computer music incorporates and extends the capabilities of previous tools and previous human roles. As discussed in the rest of this chapter, the general-purpose nature of computer technology enables abstractions, disjunctions, and remappings at all points in the music-making process, including all the ones from earlier stages of music technology and more. With the exception, in most cases, of transducers (microphones, speakers, and, more generally, sensors, controllers, and actuators), all the elements can become virtual, that is, nonphysical (decoupled from hardware). The computer culminates the process of disjunction from the body that was begun with the instrument. While this separation might seem dismal from a certain humanistic perspective (especially in light of McLuhan's harsh term "amputation"), its important converse is a perspective that sees human energy spreading out beyond the body into the world, infusing it with potentially positive new manifestations of creativity. (The amputations are optional and temporary, being reversible, and the disjunctions enable new forms of connectedness, such as online musical collaboration.) The challenge for technological development is to stay cognizant of what may have been lost (e.g., a somatically intimate connection to sound) and to investigate how to regain it. Indeed, such considerations underlie much recent research, especially on controllers; some losses can be attributed to the relatively primitive state of tools in a young field, tools that will continue to improve.

Computer technology represents the generalization of "tool" or "instrument," by which the tool's domain can extend into all areas of human endeavor and the tool can be continually adapted (if not adapt itself) to solve new problems. A host of implications ensue, among them not only new horizons in musical practice but also challenges to preconceptions about art, creativity, and human identity. In computer music, the computer can function as an abstraction of the instrument, the musician, or both. We now examine these abstractions more closely.

2. The Computer as Musical Instrument

In 1963, Max Mathews (b. 1926), the electrical engineer who has been dubbed "the father of computer music," published an influential paper in *Science*, "The Digital Computer as a Musical Instrument." This article summarized some of the groundbreaking research and development in computer-generated sound at the Bell Telephone Laboratories in New Jersey. Mathews (see fig. 2.1) had invented digital sound synthesis: the numerical construction of sounds "from scratch." To this day, the use of the computer as an instrument—that is, a sound generator—remains at the heart of the multifaceted field we call computer music.

Still, many musicians rely less on synthesis than on the playback and processing of digitally recorded sound. Playback of prerecorded sounds and real-time processing of live sounds allow external sound input to become part of a musical instrument. The instrument now may encompass both kinds of transducer (i.e., the microphone and the loudspeaker, which are themselves abstractions of the ear and the voice) as well as abstractions of these transducers (e.g., input or output of symbolic music data rather than audio). Thus, after examining digital sound synthesis, we discuss digital recording and processing and then the implications of all these audio technologies for the craft of composition.

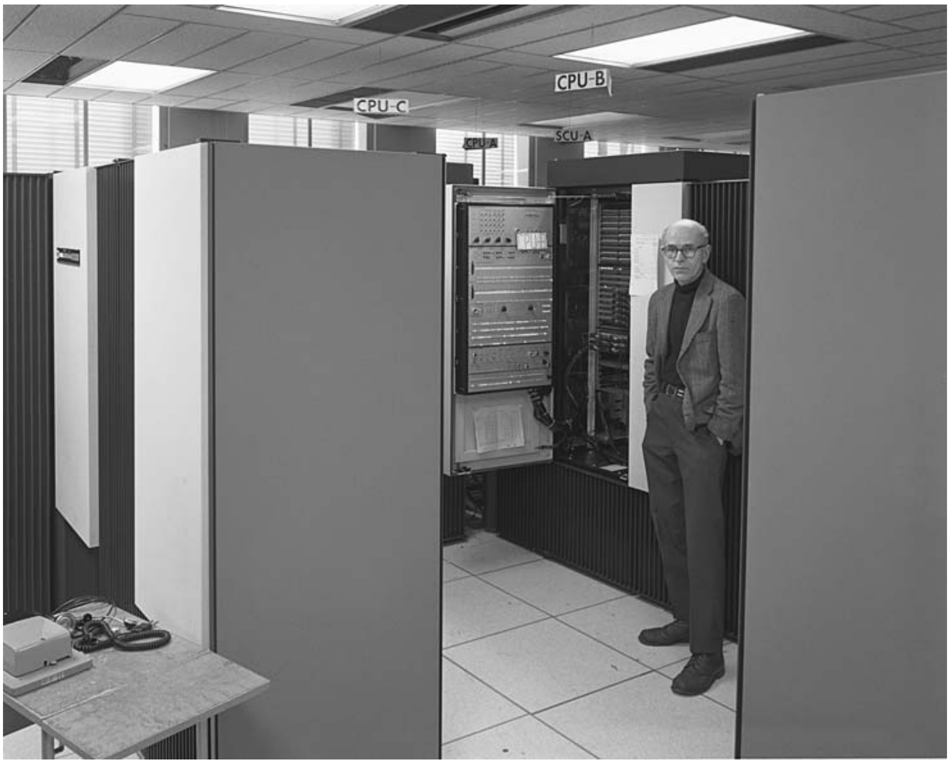


Figure 2.1 Max Mathews, standing amid the units of the IBM 7094 mainframe computer at Bell Labs around 1965. (Courtesy of Max Mathews.)

A. Digital Sound Synthesis

The earliest productions of musical sound by computer—in 1950 or 1951 in Australia (see the appendix) and England (*Computer Music Journal* 2004)—represented noteworthy technical achievements, but they were severely constrained in timbre as they relied on the machines' "system beep" facilities. This work did not result in any significant participation by composers, and until recently it remained unknown to the computer music community that subsequently developed worldwide.

True digital sound synthesis, availing itself of the computer's general-purpose nature to construct arbitrary sounds, awaited the introduction of digital-to-analog converters (DACs) in the 1950s. Max Mathews began research into digital sound synthesis in 1957 with his colleagues at Bell Laboratories, producing that year MUSIC I, the first of his computer programs for digital sound synthesis. MUSIC III, a system for interconnecting virtual modules like digital oscillators and filters, appeared in 1960. This approach was similar to the modular analog synthesizer (which appeared later in the decade) but was implemented purely in software. Owing to processor speeds, the synthesis could not be real time. In other words, it took much longer to create a sound than to play it back—a limitation not overcome for years.

Special-purpose, real-time digital synthesizers (separate musical instruments) became available in the mid-1970s. Digital synthesis on dedicated musical instruments attained wide distribution following the introduction of Yamaha's DX7 in 1983. By the late 1990s, the microprocessors in general-purpose desktop computers were fast enough for real-time sound synthesis. Of course, the computational speed of a synthesis algorithm depends on the algorithm's complexity and how efficiently it is implemented, as well as how many voices (i.e., instances of the algorithm) are running simultaneously. However, even the most demanding families of synthesis techniques are now feasible for real-time implementation. As in the past, much of the craft in designing sound synthesis algorithms involves finding computational shortcuts that have minimally adverse perceptual effects.

Many early digital synthesis algorithms were, like the typical analog synthesizer, subtractive: a timbrally rich waveform is filtered to remove its frequency components in a possibly time-varying manner. Additive synthesis, in which a spectrum is instead constructed by adding a sine wave oscillator for each of its component frequencies, was available, but its greater realism came at a much greater computational expense. John Chowning (b. 1934) at Stanford University discovered an entirely different method, frequency modulation (FM) synthesis (Chowning 1973), an efficient technique that remained popular for the next quarter century, until increased processor speeds rendered its efficiency moot. Most sound synthesis techniques can be used to emulate a desired musical sound with some degree of success, and in doing so they must approximate the acoustical waveform that reaches the ear, but the most fully virtual family of techniques, known as physical modeling (e.g., Smith 1992), goes back to the origin and explicitly emulates the acoustics within the instrument. This approach allows a virtual instrument to be played with the same kind of controls as the real instrument, with potentially striking realism. Current computers are also able to achieve comparable realism through real-time additive synthesis, given a database that stores all the timbral variants needed for a performance and given rules for joining these together in natural-sounding phrasing. Eric Lindemann's Synful Orchestra (http://www.synful.com) illustrates this approach applied to the emulation of orchestral instruments. Although realism might seem to be a moot criterion for composers who avoid imitative sounds, the significance of emulation is discussed under "Implications" in this section of the chapter.

B. Digital Sound Recording and Processing

The use of digital sound recording in composition postdated that of digital sound synthesis. Although analog-to-digital converters (ADCs) existed in the 1950s, other technical barriers to digital audio recording existed, such as limited computer memory and the need for error correction. The first commercial digital audio recorders were introduced in the mid-1970s. In the mid-1980s, after the introduction of the CD format in 1982, Sony introduced the digital audiotape (DAT) format, which became popular in the computer music community, as did direct-to-disk technologies enabled by affordable ADCs for personal computers in the late 1980s.

Once digital recording became practical, digital editing, mixing, and processing fell into place, and the methods of musique concrète could be transferred to the computer music studio. Digital signal processing in computer software enabled the development of many new techniques for transforming sound that in the analog realm had been impractical or of inferior quality. For instance, the phase vocoder, a frequency domain tool for audio analysis and resynthesis, permitted high-quality time stretching without pitch shifting and vice versa (Moorer 1978, Dolson 1982). Digital sound processing blurred the lines between synthesized and natural sound. In wavetable synthesis, for example, one period of a recorded waveform (a split second of audio) is stored in a numerical table for playback by an oscillator. With increases in computer memory, this early technique gave way to what was sometimes called *sampling synthesis*, in which a recorded note's entire duration could be stored for greater realism. As another example of the aforementioned blurring, granular techniques, which manipulate tiny snippets of sound in massive quantities (Roads 1978), can be applied equally to recorded or synthesized sound—often with perceptually similar results.

Chowning (1971) conducted the first research on the placement of sounds in a computer-generated virtual space, incorporating the simulation of reverberation and moving sound sources. He then employed these techniques in his own quadraphonic compositions. Other techniques such as ambisonics (Gerzon 1973) were invented for mathematically specifying spatial position, with the goals of greater flexibility in loudspeaker placement and listener position.

C. Implications

The synthesis, recording, and processing of sound by computer engendered a proliferation of new musical possibilities, further freeing composers from the constraints of human performers. The computer's increased precision, power, fidelity, flexibility, extensibility, and reproducibility augmented analog technologies' impact on music. The computer offered not only unlimited textural complexity and arbitrarily complicated yet precise rhythms but also unprecedented control and resolution in parameters such as pitch, timbre, and spatial location. The increased resolution implied the availability not only of discrete increments that were finer than previously possible but also of continuous trajectories along these musical dimensions.

Consider the proliferation of possibilities for pitch. The use of microtonal scales and arbitrary tuning systems, so long a challenge for acoustical instruments, became trivial with computer software, in which any frequency can be specified as easily as any other, and tuning is both precise and stable. (Some early analog instruments, notably the telharmonium, explicitly accommodated new tuning systems, but the oscillators in analog synthesizers typically suffer from frequency drift.)

Regarding computers' proliferation of timbres, Mathews realized early on that computers could, in principle, create any sound. He also recognized that the main limitation was humans' understanding of how to produce a desired sound: the field of psychoacoustics (the connection between the physics and the perception of sound) had, and still has, much to learn. As a result, psychoacoustical topics arose frequently in early computer music research. Using computer-synthesized sound, the recently introduced concept of timbre as a multidimensional space could be explored in research and in musical practice (Grey 1975, Wessel 1979). Additive synthesis made it possible to morph between timbres—an example of the trajectories mentioned. Also, it became clear that temporal variations in the amplitude or frequency of spectral components within a single tone could be critical for timbre. Understanding and manipulating sound at this microscopic level enriches the composer's vocabulary. As Chowning (2005) said, digital synthesis lets the musician not only compose with sound but also compose the sound itself.

Similarly, the parameter of space is uniquely tractable through computer technology. Although sonic liveliness can be achieved in the analog domain through multichannel recording and through dynamic routing to multiple loudspeakers during playback, computers grant the composer complete control of spatial imagery. Just as digital sound synthesis creates virtual instruments, so does digital positioning in multiple channels, along with carefully simulated reverberation, create a virtual space.

Digital sound synthesis also promotes a conjunctive blurring of the boundaries between musical dimensions that are normally thought of as distinct. For example, in the 1969 composition *Mutations* by Jean-Claude Risset (b. 1938), selected frequency components are first heard as distinct pitches but then fused in a unitary timbre (see fig. 2.2). Similarly, composers can harness the acoustical connections between timbre and tuning, as in some early experiments at Bell Labs and in Chowning's 1977 composition *Stria*. Also, *Stria* exhibits formal unity through self-similarity at multiple timescales. Computer sound generation allowed Chowning to apply the golden ratio (which itself expresses the self-similar relation $a+b:a = a:b \approx 1.618:1$) to FM synthesis parameters, spectral structure, and musical intervals in addition to overall form. This unification across temporal levels recalls Stockhausen's theoretical writing and early analog electronic music, for example, *Studie II* from 1954. Stockhausen was intrigued by a continuum of time that would span timbre, pitch, rhythm, and form.

In the same vein, digital processing of recorded sound allows the composer to play with a continuum between natural- and synthetic-sounding timbres (eroding the old ideological divide between elektronische Musik and musique concrète) and between recognizable and unrecognizable sound sources. Schaeffer was interested in disjoining sounds from their sources, focusing perception on their timbral

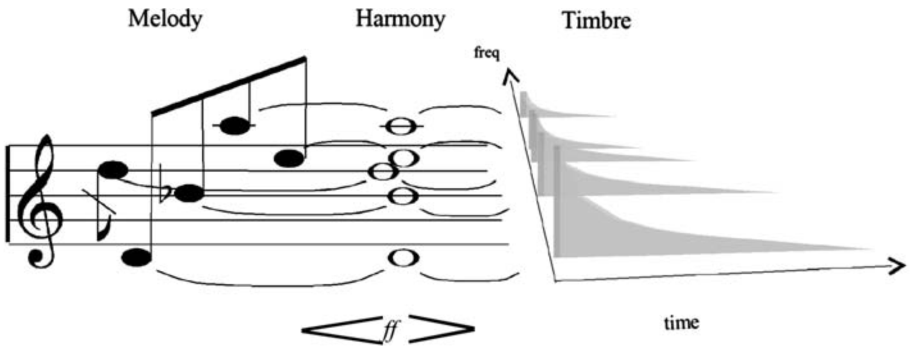


Figure 2.2 An excerpt from *Mutations* (1969) by Jean-Claude Risset, illustrating the use of computer sound synthesis to bridge pitch and timbre. Pitches first heard melodically are sustained as a harmony. Then a gong-like sound enters, containing the same frequencies, which however are now perceptually blended into a single note. (Courtesy of John Chowning.)

characteristics instead of their identities in the physical world. The term *musique concrète* has today been largely superseded by *acousmatic music*, a term with an explicitly disjunctive connotation that derives from the *akousmatikoi*, disciples of Pythagoras who listened to him from behind a veil. One way to veil the identity of a sound is to granulate it, to apply disjunction at a microscopic level. Granular synthesis and other microsound techniques (Roads 2001) became increasingly popular in the last decade or two, notably among composers of acousmatic music. As pointed out by Roads (2001, p. 340), these techniques foster a new conception of music. The traditional intervallic mode of musical thinking, concerned with numerical scales and proportions in precise parameters such as pitch and duration, gives way to a morphological mode of thinking in which sharp definition is replaced by fuzzy, supple, and ambiguous materials. It is not difficult to view these ideas through the lens of postmodernism or to compare them to an Einsteinian versus a Newtonian universe.

Computers are in fact excellent tools for both morphological and intervallic approaches, the latter including note-based musical conceptions. Notes are a disjunctive type of musical material at an intermediate timescale: the music consists of a sequence of discrete, measurable units, typically with abrupt changes in pitch from note to note. Composers in the acousmatic tradition, in particular, sometimes dismiss note-based music as lying outside the arena of interest for electroacoustic music, especially when the sounds are modeled after traditional instruments. However, it is important to understand the rationale for studying conventional features of music when undertaking technological research, regardless of their presence in particular styles of contemporary composition. As a case in point, consider the emulation of traditional instruments, which has figured heavily in sound synthesis research (Risset 1969, Chowning 1973, Smith 1992, and many others), as already implied by the discussion of realism in synthesis techniques (see "Digital Sound Synthesis"). A complaint is sometimes heard that such work should instead focus on novel timbres. The response to this argument is twofold. First, one can much better assess the success of a technique when gauging its results against a familiar reference point. Emulating the well known is often more complicated than emulating the little known, placing higher demands on the system. (Consider a "Turing test" in which the human subject has to judge whether text comes from a human or a computer, with two cases: one with the words in the subject's native language and the other with them in a completely unfamiliar language.) Second, and more germanely for the composer, the knowledge gained by refining the emulative technique will often inform a more sophisticated construction of brand-new sounds. Similarly, as we discuss under the heading "The Computer as Composer," the emulation of traditional musical styles has held an important position in algorithmic composition. The complaint about that research direction and the response to the complaint are directly analogous to those about the emulation of traditional instruments.

3. Human Control of the Digital "Instrument"

Having discussed the development of the computer as an instrument and its implications for composition, we now turn to the musician's connection to this instrument. The era of computer music extends from the 1950s to the present, a period that is roughly bisected by the appearance, within a matter of months of each other, of three paradigm-shifting technologies. The musical instrument digital interface (MIDI) protocol was introduced in 1983, as was the first affordable digital synthesizer, the Yamaha DX7. In January 1984, the first commercially successful computer with a graphical user interface—the Apple Macintosh—was launched, and it soon became especially popular among musicians as it was easy to use and could be connected to synthesizers via MIDI. For the first half of the history of computer music, therefore, users interacted with the computer via punch cards or, later, typed commands, and most had no access to real-time digital sound synthesis, having to wait, often for many hours, to hear the fruits of their labor. Music creation with the computer was almost exclusively a cerebral activity, a welcoming realm for composers perhaps, but not for performers.

In the second half of the computer music era, the two new aspects of musician-computer interaction—the graphical and the real time—made musical production less disjunct from the body. The visual representation and manual manipulation of virtual objects, combined with real-time sound synthesis or processing, come close to a natural experience for the musician. Logical next steps might involve paradigms that have been introduced in computer music but not yet widely deployed, such as haptic controllers with a realistic "feel" (Cadoz et al. 1984, Nichols 2002) and virtual reality technologies (Bargar 1993).

We divide the following discussion into tools for the composer and tools for the performer, recognizing that the boundary is not rigid: some tools may be used by both, and the two roles may overlap. In improvisation, the performer assumes some or all of the role of a composer. This discussion emphasizes software rather than hardware.

A. The Composer's Interface

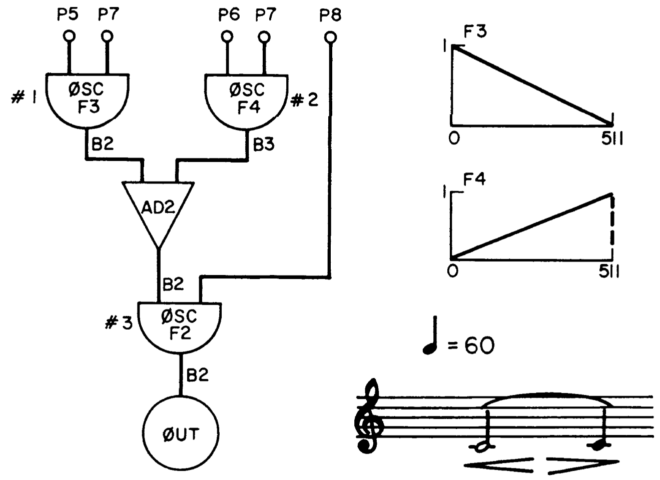
As mentioned, for decades the primary interface for composers of computer music was textual input. Typically, a composer used a special-purpose computer language for music. Sometimes, a general-purpose programming language for which music routines had been written was used.

The software engineer who develops sound synthesis tools can be considered an abstraction of the traditional instrument builder. A composer who assumes this role disintermediates that engineer. Many computer music composers have reveled in the freedom to build their own virtual instruments and even to build languages for building instruments. However, there is a continuum of involvement available to the composer, ranging from writing languages, to writing programs, to creating algorithms, to simply tweaking parameters of existing algorithms (analogous to twiddling knobs on an analog synthesizer). At the very end of this continuum is the composer served by a technical assistant, a model available at large institutions like the Institut de Recherche et Coordination Acoustique/Musique (IRCAM), founded by Pierre Boulez (b. 1925). The last scenario is enviable for the composer who has no technical inclination, and its presence is unsurprising in a society like that of France, where support for the arts is widely considered a necessity rather than a luxury. The opposite end of the scale, the musician-cum-hacker, may be emblematic of societies with a do-it-yourself ethos and a cultural history that celebrates the pioneer.

Tool development and creative ideas are mutually influential. When the composer is also a programmer (or hardware developer), or if the engineer works closely with the user, it is more likely that creative musical goals will inform the tool's design. Otherwise, musicians are likely to be frustrated by the assumptions of the tools they are provided. (MIDI is a common example [Moore 1988].) However, it is also true that composers find inspiration in the implications of preexisting tools: much music has been written that was inconceivable without the specific technology the composer employed. We now turn our attention to some specific types of tool, with an emphasis on those having programmatic interfaces.

The MUSIC-N Languages

Mathews conceived of sound synthesis software as a twofold abstraction: of the orchestra and of the score. The "orchestra" was a collection of "instruments," and the instrument in turn was an interconnected set of "unit generators," that is, signal-processing modules (see fig. 2.3). The "score" was computer code that contained specifications of the notes that the instruments would play. In early sound synthesis languages, the instruments were also defined (as connections between unit generators) within the score. Later, the orchestra-score dichotomy became more explicit, with the code defining the instruments typically stored in one file (the orchestra file) and the note list in another (the score file). Mathews's MUSIC III, created in 1960, was the first in a long lineage of synthesis languages based on the paradigm of the unit generator. (See Roads 1996a, pp. 789–790 and



```
1   INS 0 3 ;
2   ØSC P5 P7 B2 F3 P30 ;
3   ØSC P6 P7 B3 F4 P29 ;
4   AD2 B2 B3 B2 ;
5   ØSC B2 P8 B2 F2 V1 ;
6   ØUT B2 B1 ;
7   END ;
8   GEN 0 1 3 .999 0 0 511 ;
9   GEN 0 1 4 0 0 .999 511 ;
10  GEN 0 1 2 0 0 .99 50 .99 205 −.99 306 −.99 461 0 511 ;
11  NØT 0 3 2 0 2000 .0128 6.70 ;
12  NØT 2 3 1 2000 0 .0256 6.70 ;
13  TER 3 ;
```

Figure 2.3 A simple example of a MUSIC V software "instrument." *Top left:* a block diagram showing the interconnected unit generators that create the sound. *Top right:* Two functions, for diminuendo and crescendo. *Middle right:* The music to be synthesized. *Bottom:* The MUSIC V "score" for synthesizing the music; it contains both the instrument definition (lines 1–7) and the list of notes (11–12). (Reproduced from Mathews [1969], p. 59. Used with the permission of the MIT Press.)

807–808 for a list of these MUSIC-N languages.) Perhaps the most influential was MUSIC V, described by Mathews in his 1969 book, and the most widespread was Csound (Vercoe 1985a, Boulanger 2000), still in active use today.

The part of the MUSIC-N languages that abstracted the traditional score allowed the composer to list the notes in the composition along with the per note parameters for the instrument that would play that note. Depending on the language, different degrees of algorithmic control were available for generating the notes and their parameters. However, a score-generating language can lie outside the arena of MUSIC-N, and it can be conjoined to something other than sound synthesis software. For example, a note list can be translated into MIDI data for controlling hardware synthesizers, or it can be translated into music notation for human performers to read. The very first composition language to be developed, Robert Baker and Lejaren Hiller's MUSICOMP from 1963, is in the latter category. We return to the subject of algorithmic composition in a subsequent section ("The Computer as Composer").

Later Textual Languages

More recent languages for music programming offered increased flexibility. Nyquist (Dannenberg 1997) dissolved the separation between the score and the orchestra and between control signals and audio signals. Like some previous languages, it also gave the user access to all the features of a general-purpose programming language (in this case, LISP). SuperCollider (McCartney 2002) consists of an object-oriented language for composition and a real-time sound synthesis server that client applications can access over a network. There are a number of other noteworthy sound synthesis packages accessed from a general-purpose programming language such as C++ or LISP.

ChucK (Wang and Cook 2003) is a recent text-based language for real-time sound synthesis. Having a special syntax for manipulating the flow of time, its temporal model diverges from that of MUSIC-N. ChucK is designed to be useful for "on-the-fly" programming, which in a performance context is often referred to as "live coding." Live coding takes advantage of the virtualization of the instrument to eliminate the old temporal disjunction between the roles of instrument builder and performer. In live coding, their roles merge, as though Stradivarius were building violins on stage as he played them. Furthermore, both these roles are merged with that of the composer: the live coder is an improvising programmer of sound.

Graphical Patching Languages

Graphical patching languages take MUSIC-N's idea of interconnected units and render it less abstract through visual, manually manipulable representations of modules and their connections. For many composers, these environments represented a crucial advance in musician-computer interfaces as the user interface was reminiscent of a modular analog synthesizer. Graphical patching languages can also support some of the algorithmic logic characteristic of a general-purpose textual programming language, but in a form accessible to the less technically inclined musician. Roads (1996a, p. 753) and Puckette (2002) traced a bit of the history of these environments, starting with the MITSYN (Multiple Interactive Tone Synthesis System) graphical patching program that was developed at the Massachusetts Institute of Technology (MIT) by 1970. The best-known graphical patching language for music is Miller Puckette's Max program, which dates from the mid-1980s. Early versions of Max provided only control data, in the form of MIDI, but real-time audio signal processing and synthesis were added to a special-purpose version around 1990 and to the commercial version in 1997, as well as to an open source variant, Pure Data (Puckette 1997).

Puckette (2002) stressed that the most important influence on Max was in fact not graphical patching but the real-time scheduling of parallel control tasks found in Mathews's RTSKED program. (The Max software is named after Mathews.) Much of the appeal of Max is indeed its real-time nature; it is a tool for performers as well as composers. Similarly, Scaletti (2002) stressed that her environment Kyma, which among other features includes graphical patching of signal-processing modules, is not a graphical language, but rather a language offering multiple means for viewing and manipulating data.

Nonprogrammatic Software

For lack of space, we gloss over the numerous software packages with nonprogrammatic user interfaces. (See Roads [1996a] for historical summaries and publications such as *Electronic Musician* for in-depth descriptions of newer software.) After the introduction of MIDI and the graphical user interface, a number of commercial MIDI sequencers, synthesis patch editor/librarians, sound editors, and simple algorithmic composition programs appeared. As microprocessors got faster and hard disks more capacious, audio playback, audio editing, and signal processing were added to sequencers and other software, leading to a new generation of digital audio workstations (DAWs), which originally required special-purpose hardware but eventually were implemented in software. (The DAW can be thought of as a functionally extended virtual mixing console.) Similarly, faster microprocessors allowed synthesizer functionality to be implemented in software as virtual instruments, which typically provide graphical and MIDI control of simulations of older acoustic, analog, and digital musical instruments. The notion of interconnected signal-processing modules has been extended to plug-ins, modules that are accessed transparently across the boundaries of different software packages.

Of course, the user interface to which composers are most accustomed is traditional Western music notation. The input and output of music notation dates back to experiments by Hiller and Baker in 1961. Most notation software is used for creating printed output, often with supplementary auditioning via MIDI. However, some compositional software, such as OpenMusic and PWGL (PatchWork Graphical Language), integrates music notation with a graphical patching environment.

Multimedia

Algorithmic approaches to the control of sound (audio, MIDI, etc.) can cross over into the control of visuals (images, video, lighting, etc.). The same software and even the same algorithm may control both sound and image, in but one example of a one-to-many mapping of functionality using computers. The term *visual music* is often used to describe such techniques and, more generally, visual arts that emulate music's dynamism and abstractness. The Jitter software package for the Max environment has been used in this manner (Jones and Nevile 2005). Visual data can be derived from audio data and vice versa. The term *sonification* refers to the mapping of nonaudio data into sound. This term originated outside music, referring to practical techniques for data display with the goal to help the listener understand the information in the nonaudio data. However, the idea has inspired many composers, whose goal is instead an aesthetically rewarding sonic result. Some pieces based on this kind of mapping have employed natural phenomena such as magnetic fields, coastlines, or molecular vibrations, while others have used technological sources such as network delays or the code for Web pages.

B. The Performer's Interface

Whereas the first half of the computer music era was dominated by the composer, the second half has witnessed the ascent of the performer. There had already been a long history of performers participating in analog electronic music. Most often, they simply performed on conventional acoustic instruments alongside a tape recorder that played back electronic sounds. "Tape music" was for years the main outlet for computer music as well—often pieces were conceived for the tape recorder alone, but music for performer plus "computer-generated tape" was not uncommon. However, live analog electronics also played an important role in the 20th century. Here, the performer could interact with the machine (see, e.g., Chadabe 1997). After the introduction of MIDI synthesizers, computer music quickly expanded into this interactive domain. The present discussion examines performers' interaction with computers from the standpoint of connectivity. First, we briefly look at types of input to the performer, then ways of capturing output from the performer to feed to the computer, and finally interconnections that are more complex.

Input to the Performer

When playing along with a computer, the performer usually simply listens to the generated sound and responds accordingly, just as in conventional performance with other musicians. However, electronics can explicitly cue the performer. In the analog domain, performers sometimes wore headphones to hear tape-recorded cues. The composer Emmanuel Ghent (1925–2003) in the 1960s was the first to use the computer to cue performers in this way (Roads 1996a, p. 680). Visual cues can also be given. Of course, the canonical input to the performer is a visual score in traditional music notation. More recently, composers have used real-time updating of various sorts of computer-displayed notation to effect interactivity between the computer and the performer (Freeman 2008).

Output from the Performer

The electronic capture of performers' gestures has become an especially rich field of study in the 21st century. The annual international conference, New Interfaces for Musical Expression (NIME), had its inception in 2001. In addition to the NIME proceedings, refer to the textbook on controllers by Miranda and Wanderley (2006) and to the article by Jordà (2004). We consider here three categories of output: performance on traditional acoustic instruments or voice, performance with new controllers, and performance analogous to the traditional conductor's.

Performances by singers and traditional instrumentalists can be captured as audio and video and analyzed to create control data for sound generation. Alternatively, the addition of sensors allows the performer's gestures and motions to be measured more directly. The technological extension of traditional instruments turns them into "hyperinstruments," as they are called by Tod Machover of the MIT Media Laboratory. (This idea recalls McLuhan's "extensions of man" but applied to the tool rather than to the human, adding an additional layer of abstraction.)

There is a large body of work on alternative controllers, that is, new performance interfaces (see chapter 11 this volume, and the previously mentioned references). The great variety of such systems, and the present chapter's emphasis on software, preclude a satisfactory treatment here. However, a key point is that the introduction of computer technology bestowed the instrument designer with the option of a complete disjunction between the controller and the synthesis engine. Within this gap can lie one or more software layers that map the physical gestures to synthesis parameters or to higher-level musical events in a thoroughly flexible fashion. These include going beyond the one-to-one mapping of traditional instruments, where each note requires a separate gesture, to a one-to-many mapping in which a gesture can trigger a complex set of events. Given the disjunction between player and sound generation, an important direction in controller design is to mimic what Moore (1988) called the "control intimacy" of traditional instruments. Minimizing the temporal latency between gesture and sonic result is crucial (Moore 1988, Wessel and Wright 2002).

There is a fuzzy boundary between interfaces for instrumentalists and interfaces for conductors. Mathews was interested long ago in the metaphor of "conducting" electronic instruments. The GROOVE (Generated Real-Time Output Operations on Voltage-Controlled Equipment) system (Mathews and Moore 1970) was a hybrid digital/analog system that included a CONDUCT program; in the late 1980s, Mathews developed his Radio Baton controller, which was still not designed for traditional conducting. For a nonmetaphorical conductor, the technology exists to conduct music in real time through video- or sensor-based capture of traditional conducting gestures. The conducting gestures can control real-time sound synthesis, or they can control an audio recording by modifying its timing and dynamics through digital signal processing (Lee et al. 2006).

Networked Performance

Performers using networked computers can collectively control the generation of synthesized music. Conceptually, this is not necessarily distant from four-hand piano. However, networked performers can also control each other's instruments, perhaps only subtly modifying a certain parameter while allowing the performer of each instrument to retain overall control or perhaps changing each other's performances more drastically. This interdependent sort of performance represents a relatively new paradigm in music, a new kind of conjunction of musicians and of instruments. The League of Automatic Music Composers pioneered interdependent computer performance in the late 1970s (Bischoff et al. 1978). Weinberg (2005) proposed a taxonomy for such performance, in which control can be centralized or decentralized and interaction can be sequential or simultaneous (see fig. 2.4). The network can, of course, be the Internet. Viewed from another perspective, the Internet can serve as a point of disjunction, allowing performers to be decoupled in space from each other as well as from their audience.

The audience can become part of the composition, providing input to the computer system and thereby becoming (to some extent) performers and even composers themselves. Their input can affect both automatic sound synthesis and, via real-time updates to performance instructions, human sound production (Freeman 2008).

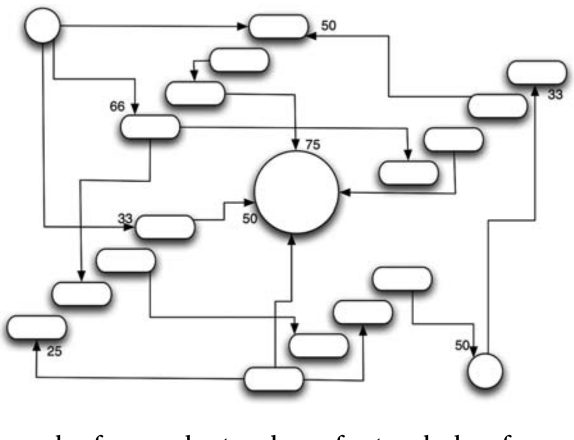


Figure 2.4 One example of a complex topology of networked performers, from Weinberg (2005). The circles represent computers serving as hubs, and the oblong shapes represent computers or digital instruments played by performers. The latter are arranged here in staircase patterns that indicate sequential operations, with one player's actions followed by another's. The numbers specify how much influence is exerted in a particular connection. (Used with the permission of MIT Press.)

C. Implications

The technology of computer music continues the older trends in music technology discussed in this chapter. Through a new intermediary, the computer, composers can avail themselves of new sonic possibilities and new levels of abstraction. The computer also permits the disintermediation of human performers and their instruments, with all their cognitive and physical constraints. In this case, the composer regains direct contact with sonic production and, in principle, gains unlimited control over the result. Conversely, the composer can relinquish control by only specifying the score algorithmically at a high level, for example, or by creating sound installations that vary depending on current conditions, including observers' actions. This release of control has many precedents in music, such as figured bass in the Baroque era and aleatoric composition in the 20th century.

With the computer, human physiology can be removed from the process of musical creation. However, given the physiological substrates of music, it is quite certain that musical performance will never become obsolete. Instead, humans will continue to find new expressive ways to map physical gestures into sonic results. Although the computer can certainly disintermediate the human performer, it is equally possible to disintermediate only the instrument, allowing the performer to transcend the limitations of conventional instruments.

Just as the composer can, in a sense, become a performer (directly creating the sound), the performer can, in a sense, become a conductor: controlling the sound in real time without being bound by the need to develop virtuosic technique on one or more conventional musical interfaces. The intrinsic idiosyncrasies of the instrument can vanish, to be replaced by whatever idiosyncratic mappings serve the performer's imaginative needs. Just as the computer frees the composer from a one-to-one mapping between compositional indication (notes on paper) and sonic result (sounding notes), so it frees the performer from a necessarily one-to-one mapping between gesture and sound. Like the composer (operating out of real time), the performer (operating in real time) can control arbitrarily complex musical processes. Composers and performers often play with the ambiguity and potential confusion between the expected conventional mappings or roles and newly possible ones. As a result, the clarity and theatricality of gesture become more important. The performer tends toward the dancer. Similarly, the dancer equipped with sensors can become a musical performer, controlling the music—or, as Siegel and Jacobsen (1998) say, "dancing the music."

4. The Computer as Musician

The previous section was titled "Human Control of the Digital 'Instrument.'" What about interactivity? To the extent that the human musician's interaction with the computer cannot be termed "control," the computer is in at least one sense autonomous, and it starts to make sense to refer, at least metaphorically, to the computer as a musician. Of course, the ultimate control of the computer lies with the human who has programmed the computer. But, given the possible inclusion of machine-learning techniques and other forms of software complexity, the computer's behavior might not be predicted by its programmer. The threshold of autonomy is fuzzy. When does algorithmic composition software stop being just a human composer's tool and start becoming a composer itself? When does an instrument—or "hyperinstrument"—stop being just a human performer's tool and start becoming a performer itself?

A. The Computer as Composer

Even in his 1963 article on the "computer as a musical instrument," Mathews discussed other researchers' and composers' use of the computer for automated composition rather than for sound synthesis. Those people included Lejaren Hiller (1924–1994), James Tenney (1934–2006), and others. The book by Hiller and Isaacson (1959) includes the complete score for the world's first "serious" computer-generated composition, their *Illiac Suite for String Quartet*, constructed from September 1955 to November 1956 and published in 1957. (A short computer-generated song in a popular style, created in 1956 by two other programmers, is shown in Ames [1987].) Each of the *Illiac Suite's* four movements is the result of a different "experiment," with different musical results. The first two movements involve conventional counterpoint; the last two illustrate modern experimental styles. Thus, this early piece of computer-composed music illustrates two different aspects of algorithmic composition that have since been extensively developed: (1) emulation of a traditional style, as a sort of validation of the software's ability to solve a well-known musical problem; and (2) composition in a contemporary style, generally in service of a particular composer's aesthetic and creative needs. In the former category, significant contributions were made by investigators such as Schottstaedt (1984), Ebcioğlu (1985), Hörnel and Menzel (1998), and especially Cope (1996, 2004). (Some of these investigators are composers to whom category 2 also applies.) In the latter category, there are too many composers to list, but for this historical essay, special note must be made of the pioneers Iannis Xenakis (1922–2001), Gottfried Michael Koenig (b. 1926), and Herbert Brün (1918–2000), in addition to Hiller (who also collaborated with John Cage, 1912–1992) (see chapter 5, this volume).

In the category of style emulation, the software by David Cope (b. 1941), called Experiments in Musical Intelligence (1996), deserves special mention, not for the controversy it has engendered through its mimicking of famous classical composers, but for the number of years the composer has put into refining his software and for the sheer volume of the musical corpus it has produced. Counting individual movements as separate works, Cope calculated that he had created over a thousand finished compositions with his software. It seems likely that many of these works could pass a Turing test for all but the most musically knowledgeable listeners (see fig. 2.5). Regardless of one's view of these particular pieces, or of style emulation in general, the work is thought provoking because the future will certainly see many more such efforts of increasing sophistication, not all using Cope's methods, and because even Cope's techniques are not constrained to traditional styles. He has applied them to his own musical style, which

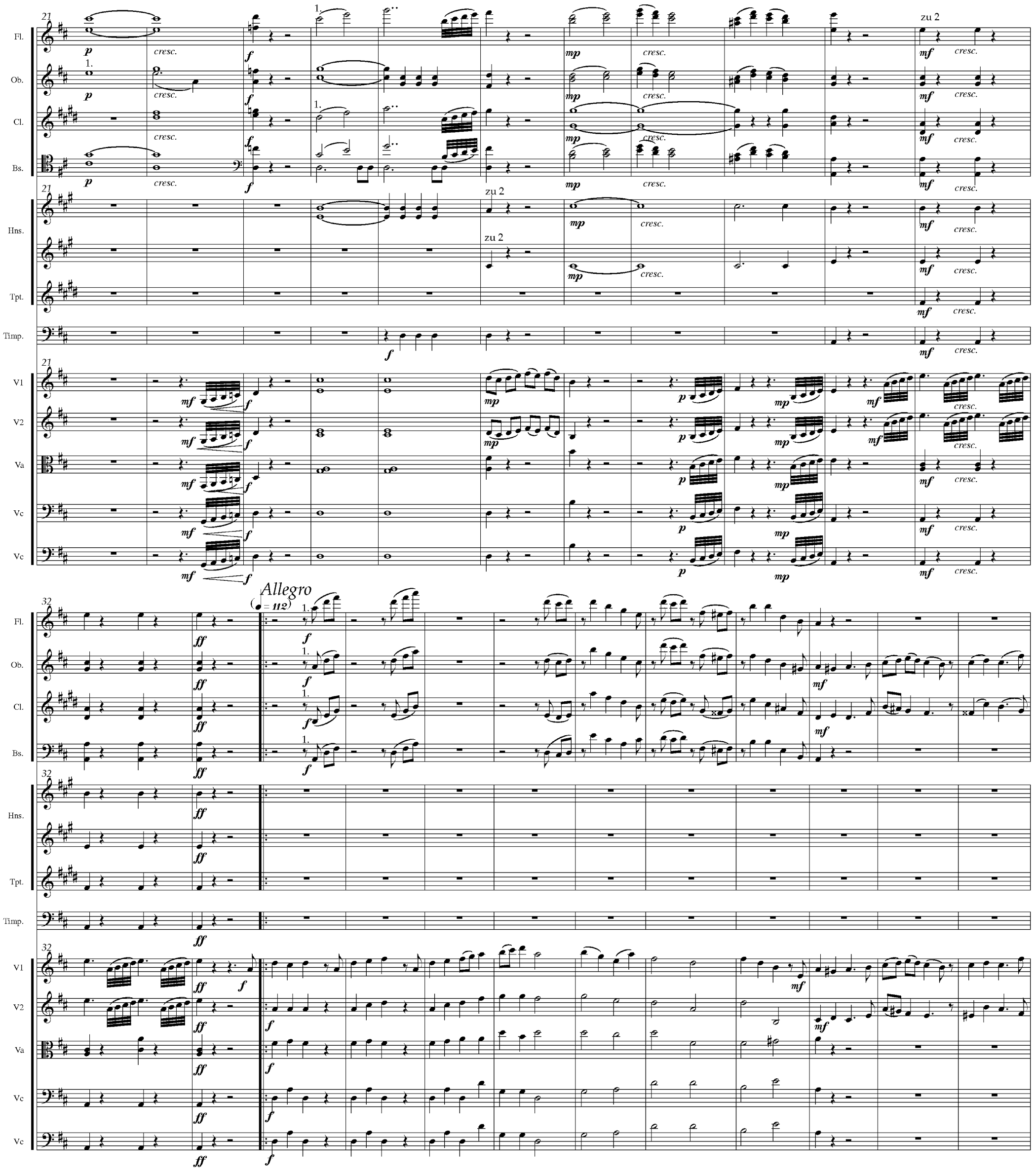


Figure 2.5 An excerpt from the first movement of David Cope's Emmy-Beethoven Symphony. (Used with the permission of David Cope and Spectrum Press.)

demonstrates that style emulation can be relevant to the second category of algorithmic composition mentioned.

The advent of computer-aided algorithmic composition of that second type represented a logical continuation of mathematical and formalized procedures in 20th-century music, such as serialism, Xenakis's stochastic composition, and Cage's chance music. Most composers of computer music who employ algorithmic compositional techniques would assert that in the case of their compositions, the computer is not a composer, only a compositional aid. There is indeed a spectrum of involvement available, ranging from simple calculations that solve a small musical problem or generate a bit of source material for further human manipulation, to systems that are intended to run as stand-alone generators of entire pieces, as was the case for each movement of the *Illiac Suite*. Composers of computer music have used a wide variety of algorithmic approaches. Representative categories include formal grammars, stochastic algorithms, genetic and other evolutionary algorithms, cellular automata, neural nets and other machine-learning techniques, and expert systems. Overviews of algorithmic composition include those by Ames (1987), Loy (1989), Roads (1996a), Chadabe (1997), and Essl (2007). Ariza (2005) has proposed a taxonomy of algorithmic composition systems.

B. The Computer as Performer

In considering a computer to be a "performer," there are several dimensions for classification. One can categorize the system based on its inputs and outputs; for example, it might accept keyboard and mouse events but not audio as input, and it might produce audio and video as output but not MIDI. One can examine what produces the sound: the computer itself, external synthesizers (e.g., via MIDI), physical musical instruments (via actuators), or humans (e.g., by automated conducting). One can also consider whether the machine is represented anthropomorphically, whether just through visual appearance or also through functional, robotic simulations of human body parts. Most important, though, one can assess the degree of "intelligence" the system exhibits. Does it have knowledge of musical conventions and styles in terms of either composition or performance practice (such as expressive deviation from metronomic timing)? Does it "listen" to other performers and react as a human musician might? Can it learn about music by listening to it? Can it improvise? Or, does it simply play back data that has been fixed in advance of the performance, oblivious of what might be going on around it? If it does more than the last, it exhibits some degree of what Rowe (2001) and others call machine musicianship. (Also see Jordà [2004] for an analytical discussion that treats intelligent and interactive systems within the framework of digital instruments.)

Robotics and Animated Characters

The computer that controls the playing of physical musical instruments represents a refinement of the old tradition of mechanical instruments, discussed in the section on antecedents. The player piano of a century ago has become a MIDI piano such as the Yamaha Disklavier (introduced in the 1980s), with the software-based MIDI sequence serving as an abstraction of the piano roll. Other instruments, conventional or innovative, can similarly be fitted with computer-controlled actuators (Kapur 2005). Musical robots have become increasingly sophisticated, with Japanese efforts taking the lead. In 1985, researchers at Japan's Waseda University startled the computer music community by demonstrating WABOT-2, an anthropomorphic robot musician that had not only fingers to play an organ keyboard but also "eyes" (a camera) that read conventionally notated sheet music (Roads 1986). More recently, the Waseda group demonstrated an anthropomorphic robot flutist, complete with artificial lungs and lips (Solis et al. 2006). Other researchers are focusing on integrating machine musicianship into robots that can improvise (Weinberg and Driscoll 2006).

Indeed, the software challenges of machine musicianship are more broadly pertinent to computer music than are robot mechanics. Furthermore, anthropomorphism is often more motivated by psychology than by functional necessity. However, anthropomorphism can also be obtained without robotics, through on-screen characters. As an example, Bos et al. (2006) implemented a virtual conductor that directs a human ensemble whose players watch the animated character's traditional conducting patterns on a computer monitor. This virtual conductor incorporates machine-listening techniques, so it can react to the playing of the musicians it is conducting.

Machine Recognition of Music

There are numerous applications of machine recognition of music, whether the music takes the form of audio, sheet music, MIDI, or something else. Computer accompaniment of human performers, in the classical vein where the music is fully notated, required the development of score-following techniques (Dannenberg 1985, Vercoe 1985b). In these, the "score" was some computer representation of the music rather than traditional musical notation, and the computer "listened" to a human soloist. Actual optical music recognition (OMR), having sheet music as input, originated with research at MIT in the 1960s and became commercially available in the 1990s. Although designed for other musical applications, OMR can be incorporated in live performance, as prefigured by the WABOT-2 system.

More crucial than the ability to read music is the ability to hear it and parse what is heard into musical units. On the part of a computer, this ability is generally called *automatic transcription* (Moorer 1975, Klapuri and Davy 2006), a term that suggests but does not require traditional notation as the output. Real-time recognition of previously unknown music, combined with some musical knowledge, would permit the computer to improvise. Such a task is simplified when the input is MIDI rather than audio. Once the musical units, such as notes, are recognized, there are many paths the computer could take in response. For example, it could select harmonies to accompany an input melody, or it could improvise by extending the melody further, as in the case of François Pachet's Continuator (Pachet

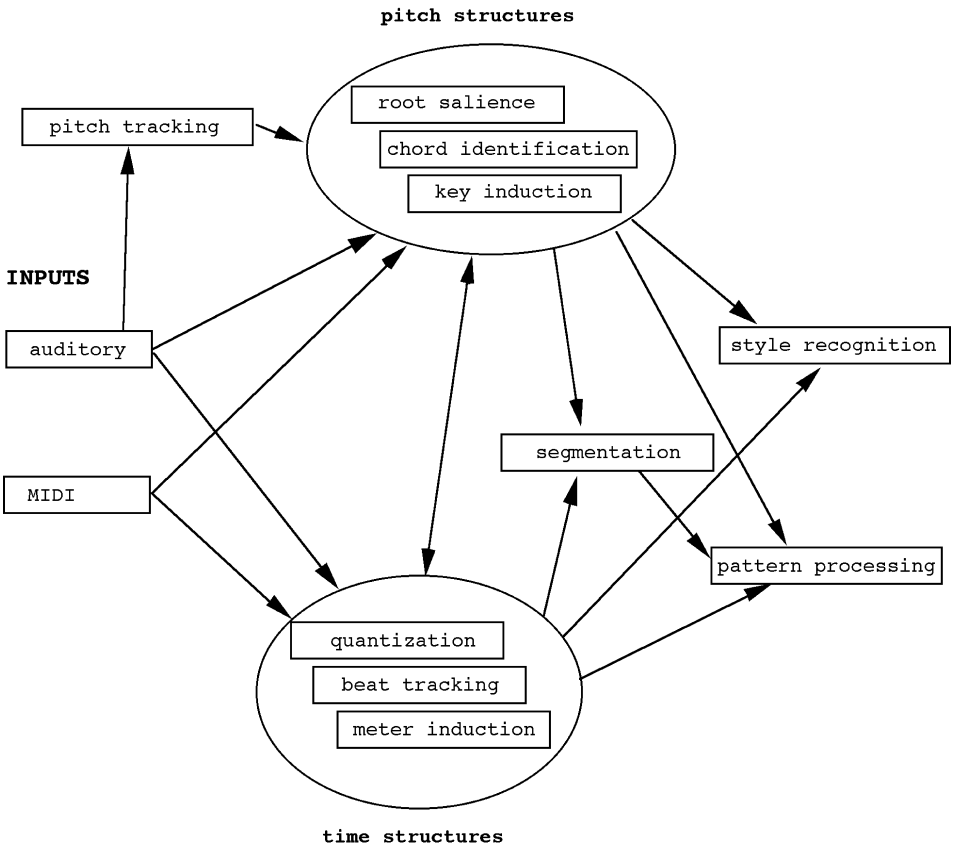


Figure 2.6 The overall architecture of a machine-musicianship system, from Rowe (2001). (Used with the permission of MIT Press.)

2002). Figure 2.6, from Rowe (2001), shows a typical architecture for a machine musicianship system in which the inputs are either MIDI or audio.

Expression

Traditional notation specifies music with symbols that are perfectly quantized in pitch and time. The score says to play a quarter-note A on the fourth beat, not to play 441 Hz for 0.8 s starting at 0.08 s after the beat, with a vibrato rate of 5.1 Hz and a deviation of ±3 Hz. A crucial part of a musician's knowledge consists of understanding how to render the music with expressive deviations from the putative regularity of the abstract symbols, based on the musical context and style. Machine musicianship benefits from research that quantifies and makes explicit these normally intuitive and implicit rules. Such research at Stockholm's Kungliga Tekniska Högskolan (KTH), or Royal Institute of Technology, has resulted in over 60 publications (http://www.speech.kth.se/music/performance/). The KTH rules for expressive performance have been embedded in MIDI sequencers (Friberg et al. 2000) and can accommodate real-time updates similar to conducting (Friberg 2006).

C. Implications

Research areas such as OMR, automatic transcription of musical audio, and other types of music information retrieval (MIR) were developed primarily for nonperformance applications, but they can be turned to creative use in performance, as other technical areas have been, especially if real-time implementations are possible. The goal of machine musicianship encourages "computer music," in the sense of a creative practice, to harness the wide-ranging resources of "computer music," in the sense of a technical field.

Computer performance can range from the trivial, with the machine exhibiting no more intelligence than a tape recorder, to the complex, with the computer interacting with human musicians, playing in a way that listeners consider expressive, and improvising. However, simply matching human performance may be more interesting to science than to art. As Nancarrow's work for player piano demonstrates, composers may find machine performance of music to be of most interest when the machine does something that human performers cannot.

As for machine composition, Cope's prolific experiments point to a future in which the validity of computers' musical output is no longer at issue. Just as the intermediary of notation led to the development of a new musical role, the meta-performer (i.e., the conductor), so has the intermediary of the computer led to a new role that we could call the meta-composer. Instead of working directly on a composition, the meta-composer (the prime example thus far being Cope) works with an automated or semiautomated system that can produce many more compositions than a human could in one lifetime working unaided. This again is an example of abstraction, disjunction, one-to-many mapping, and proliferation. The composer removes himself or herself from the details of creating a specific composition while producing a great many compositions, just as the conductor does not perform on a specific instrument but at a high level controls them all. To state that this new paradigm exists is not to denigrate the creative art of composers who work otherwise, including without a computer, or to claim that human postprocessing cannot improve the output of automated systems. However, a reflexively negative response would invite comparison with historical reactions to the Copernican revolution and similar upsets. Moreover, limitations of current computer models are hardly cause for a self-satisfied proclamation of some uniquely human ability that must lie beyond the reach of automata; similar claims in the past have fallen victim to advances in science and technology, and computer music is a young field.

5. Conclusion

Contemplating what Chadabe (1997, p. 21) called the "great opening up of music to all sound," as well as Schoenberg's modernist application of algorithms in the creative process, it is clear that the 20th century fundamentally redefined music. The new definition subsumed, rather than eliminated, the old. An important contributor to the redefinition was the introduction of electronically generated or captured sound, culminating in computer music technology, which has likewise expanded the definition of the musical instrument.

Similarly, the 21st century is fundamentally redefining the musician in a way that subsumes the old definition. There are several parts to this redefinition (all of which continue trends from the 20th century):

1. *The composer*: The interjection of arbitrarily complex algorithms into the compositional process allows the composer to operate on the music at as high or as low a level as desired. Whereas the traditional composer operates on the same level as the performer, writing the individual notes that the performer will play, the composer who uses algorithms may work at a higher level. At the high end of the scale, the composer can become what we have referred to as a meta-composer. At a lower level, in creating synthesis models and fashioning sounds directly, the composer takes on the roles of the instrument builder and performer.
2. *The performer*: The completion of the process of disjoining the control interface from the sound generator means that human musicianship need not be so focused on developing physical dexterity and can focus on higher-level control with flexible mapping of gesture to sonic result—the instrumentalist tends to become a conductor. Performers may develop and perfect their own individual interfaces and mappings. Performers may be networked together and control each other's performances.
3. *The public*: The ubiquity of sufficiently powerful computer hardware and the availability of sufficiently flexible software help to democratize music-making, allowing people with less training to craft and perform music. (Even a young schoolchild can use the GarageBand software, for example.) This is not to say that the amateur's musical output will be as sophisticated as the trained musician's, only that the amateur's output can achieve a previously inconceivable degree of polish. This is because the expert's musical knowledge and control have been virtualized, that is, abstracted and disjoined from the person and moved into the tools. Also, the disjunction between control and sound generation allows the network, such as the Internet, to act as an intermediary, conjoining users across space in collaborative music-making. Similarly, the flexible mapping of control information afforded by computer technology means that an audience can participate in a performance in numerous ways, directly affecting the rendering and even the composition of the music that the computer or the human performers play.
4. *The machine*: The ongoing advances in machine musicianship suggest that the term *musician* will increasingly be used metaphorically to refer to automated processes. When people habitually use a metaphorical term, the

> sense of metaphor diminishes, so it is entirely possible, if not likely, that the word *musician* will come to refer, in common usage, to either a human or an automated entity. Anthropomorphic robots and on-screen characters, while unnecessary in principle, will probably accelerate the adoption of the new meaning.

Paradoxically, the ability of the computer to mimic, transform, and in some respects exceed conventional human music-making could serve to refocus humans' concern on the corporeal and emotional aspects of music—bringing to a full circle the process of disjunction that began with the externalization of sound production from the body. However, the arts do not follow preordained paths, and the subjective aspects of human experience are not off-limits for computer modeling, so such prognostication would simply be speculative. It is more likely that such a trend would intersect with others in different directions.

In summary, the historical development of music technology can be viewed as human creativity's drawing on the primal act of singing and proliferating it out from the body through a multitude of ingenious devices. These started with musical instruments and included tools to extend music across space and time. There is a web of interrelationships among musicians and these devices. As the generalization of the tool, computer technology can be inserted at any juncture in this network, augmenting or replacing instruments or musicians through disjunctions and new conjunctions. The resulting profusion of potential physical and virtual interconnections holds the seeds of countless sonic manifestations, which collectively constitute the musically possible. From these, it is the role of the artist to extract the musically meaningful.

Acknowledgments: In addition to this volume's editor (Roger Dean), the following individuals provided helpful suggestions after reading an earlier draft of this chapter: Thom Blum, Tim Perkis, Curtis Roads, and Robert Rowe.

BIBLIOGRAPHY

Ames, C. 1987. Automated composition in retrospect: 1956–1986. *Leonardo* 20(2): 169–185.

Ariza, C. 2005. Navigating the landscape of computer-aided algorithmic composition systems: a definition, seven descriptors, and a lexicon of systems and research. In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 765–772.

Bargar, R. 1993. Virtual sound composition for the CAVE. In *Proceedings of the 1993 International Computer Music Conference.* San Francisco: International Computer Music Association, p. 154.

Bischoff, J., R. Gold, and J. Horton. 1978. Microcomputer network music. *Computer Music Journal* 2(3): 24–29.

Bos, P., D. Reidsma, Z. M. Ruttkay, and A. Nijholt. 2006. Interacting with a virtual conductor. In *Proceedings of 5th International Conference on Entertainment Computing.* Lecture Notes in Computer Science 4161. Berlin: Springer Verlag, pp. 25–30.

Boulanger, R., ed. 2000. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming.* Cambridge: MIT Press.

Cadoz, C., A. Luciani, and J. Florens. 1984. Responsive input devices and sound synthesis by simulation of instrumental mechanisms: The Cordis system. *Computer Music Journal* 8(3): 60–73.

Chadabe, J. 1997. *Electric Sound: The Past and Promise of Electronic Music.* Upper Saddle River, NJ: Prentice-Hall.

Chowning, J. 1971. The simulation of moving sound sources. *Journal of the Audio Engineering Society* 19: 2–6.

Chowning, J. 1973. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society* 21(7): 526–534.

Chowning, J. 2005. Composer le son lui-même. In *Portraits Polychromes no. 7: John Chowning*, ed. E. Gayou. Paris: Michel de Maule, pp. 25–30.

Computer Music Journal. 2004. Editor's note, in DVD Program Notes. [Includes a transcription of Frank Cooper's 1994 audio commentary on the production of computer music in Manchester in 1951. The music and commentary appear on the accompanying disc.] *Computer Music Journal* 28(4): 126–127.

Cope, D. 1996. *Experiments in Musical Intelligence.* Madison, WI: A-R Editions.

Cope, D. 2004. *Virtual Music: Computer Synthesis of Musical Style.* Cambridge: MIT Press.

Dannenberg, R. 1985. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference 1984.* San Francisco: International Computer Music Association, pp. 193–198.

Dannenberg, R. 1997. Machine tongues XIX: Nyquist, a language for composition and sound synthesis. *Computer Music Journal* 21(3): 50–60.

Davies, H. 2001. Electronic instruments. In *The New Grove Dictionary of Music and Musicians*, vol. 8, 2nd ed., ed. S. Sadie. London: Macmillan, pp. 67–107.

Dolson, M. 1982. "A tracking phase vocoder and its use in the analysis of ensemble sounds." Ph.D. thesis, California Institute of Technology.

Ebcioğlu, K. 1985. An expert system for Schenkerian synthesis of chorales in the style of J. S. Bach. In *Proceedings of the 1984 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 135–142.

Essl, K. 2007. Algorithmic composition. In *The Cambridge Companion to Electronic Music*, ed. N. Collins and J. d'Escrivan. Cambridge: Cambridge University Press, pp. 107–125.

Freeman, J. 2008. Extreme sight-reading, mediated expression, and audience participation: real-time music notation in live performance. *Computer Music Journal* 32(3): 25–41.

Friberg, A. 2006. pDM: An expressive sequencer with real-time control of the KTH music performance rules. *Computer Music Journal* 30(1): 37–48.

Friberg, A., V. Colombo, L. Frydén, and J. Sundberg. 2000. Generating musical performances with Director Musices. *Computer Music Journal* 24(3): 23–29.

Gerzon, M. 1973. Periphony: With-height sound reproduction. *Journal of the Audio Engineering Society* 21(1): 2–10.

Grey, J. M. 1975. "An exploration of musical timbre." Ph.D. thesis, Stanford University.

Hill, D., trans. and comm. 1979. *The Book of Ingenious Devices (Kitáb al-Hiyal) by the Banú (sons of) Músà bin Shákir.* Dordrecht, Netherlands: Reidel.

Hiller, L. A., and L. M. Isaacson. 1959. *Experimental Music: Composition with an Electronic Computer.* New York: McGraw-Hill.

Hörnel, D., and W. Menzel. 1998. Learning musical structure and style with neural networks. *Computer Music Journal* 22(4): 44–62.

Jones, R., and B. Nevile. 2005. Creating visual music in Jitter: Approaches and techniques. *Computer Music Journal* 29(4): 55–70.

Jordà, S. 2004. Instruments and players: Some thoughts on digital lutherie. *Journal of New Music Research* 33(3): 321–341.

Kapur, A. 2005. A history of robotic musical instruments. In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 21–28.

Kassler, M., and H. Howe. 1980. Computers and music. In *The New Grove Dictionary of Music and Musicians*, vol. 4, 1st ed., ed. S. Sadie. London: Macmillan, pp. 603–615.

Klapuri, A., and M. Davy. 2006. *Signal Processing Methods for Music Transcription.* New York: Springer.

Landy, L. 2007. *Understanding the Art of Sound Organization.* Cambridge: MIT Press.

Lee, E., T. Karrer, and J. Borchers. 2006. Toward a framework for interactive systems to conduct digital audio and video streams. *Computer Music Journal* 30(1): 21–36.

Loy, D. G. 1989. Composing with computers: A survey of some compositional formalisms and music programming languages. In *Current Directions in Computer Music Research*, ed. M. V. Mathews and J. R. Pierce. Cambridge: MIT Press, pp. 291–396.

Manning, P. 2004. *Electronic and Computer Music.* Rev. and expanded ed. New York: Oxford University Press.

Mathews, M. 1963. The digital computer as a musical instrument. *Science* 142(3592): 553–557.

Mathews, M. 1969. *The Technology of Computer Music.* Cambridge: MIT Press.

Mathews, M., and F. R. Moore. 1970. GROOVE—a program to compose, store, and edit functions of time. *Communications of the ACM* 13(12): 715–721.

McCartney, J. 2002. Rethinking the computer music language: SuperCollider. *Computer Music Journal* 26(4): 61–68.

McLuhan, M. 1964. *Understanding Media: The Extensions of Man.* Cambridge: MIT Press.

Miranda, E. R., and M. Wanderley. 2006. *New Digital Musical Instruments: Control and Interaction beyond the Keyboard.* Middleton, WI: A-R Editions.

Moore, F. R. 1988. The dysfunctions of MIDI. *Computer Music Journal* 12(1): 19–28.

Moorer, J. A. 1975. "On the segmentation and analysis of continuous musical sound by digital computer." Ph.D. thesis, Stanford University.

Moorer, J. A. 1978. The use of the phase vocoder in computer music applications. *Journal of the Audio Engineering Society* 26(1/2): 42–45.

Nichols, C. 2002. The vBow: A virtual violin bow controller for mapping gesture to synthesis with haptic feedback. *Organised Sound* 7(2): 215–220.

Pachet, F. 2002. The Continuator: Musical interaction with style. In *Proceedings of the 2002 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 211–218.

Pope, S. T. 1994. Editor's notes: A taxonomy of computer music. *Computer Music Journal* 18(1): 5–7.

Puckette, M. 1997. Pure Data. In *Proceedings of the International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 224–227.

Puckette, M. 2002. Max at seventeen. *Computer Music Journal* 26(4): 31–43.

Rhea, T. 1972. "The evolution of electronic musical instruments in the United States." Ph.D. thesis, George Peabody College, Nashville, TN.

Risset, J.-C. 1969. *An Introductory Catalogue of Computer Synthesized Sounds.* Murray Hill, NJ: Bell Telephone Laboratories.

Roads, C. 1978. Automated granular synthesis of sound. *Computer Music Journal* 2(2): 61–62.

Roads, C. 1986. The Tsukuba Musical Robot. *Computer Music Journal* 10(2): 39–43.

Roads, C. 1996a. *The Computer Music Tutorial.* Second printing (pbk.). Cambridge: MIT Press.

Roads, C. 1996b. Early electronic music instruments: Time line 1899–1950. *Computer Music Journal* 20(3): 20–23.

Roads, C. 2001. *Microsound.* Cambridge: MIT Press.

Rowe, R. 2001. *Machine Musicianship.* Cambridge: MIT Press.

Scaletti, C. 2002. Computer music languages, Kyma, and the future. *Computer Music Journal* 26(4): 69–82.

Schottstaedt, W. 1984. *Automatic Species Counterpoint.* Stanford Technical Report STAN-M-19, Stanford University, Palo Alto, CA.

Siegel, W., and J. Jacobsen. 1998. The challenges of interactive dance: An overview and case study. *Computer Music Journal* 22(4): 29–43.

Smith, J. 1992. Physical modeling using digital waveguides. *Computer Music Journal* 16(4): 74–91.

Solis, J., K. Chida, K. Taniguchi, S. Hashimoto, K. Suefuji, and A. Takanishi. 2006. The Waseda flutist robot WF-4RII in comparison with a professional flutist. *Computer Music Journal* 30(4): 12–27.

Vercoe, B. 1985a. *Csound: A Manual for the Audio-processing System.* Program documentation. Cambridge: MIT Media Lab.

Vercoe, B. 1985b. The synthetic performer in the context of live musical performance. In *Proceedings of the International Computer Music Conference 1984.* San Francisco: International Computer Music Association, pp. 199–200.

Wang, G., and P. Cook. 2003. ChucK: A concurrent and on-the-fly audio programming language. In *Proceedings of the 2003 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 219–226.

Weinberg, G. 2005. Interconnected musical networks: Toward a theoretical framework. *Computer Music Journal* 29(2): 23–39.

Weinberg, G., and S. Driscoll. 2006. Toward robotic musicianship. *Computer Music Journal* 30(4): 28–45.

Wessel, D. 1979. Timbre space as a musical control structure. *Computer Music Journal* 3(2): 45–52.

Wessel, D., and M. Wright. 2002. Problems and prospects for intimate musical control of computers. *Computer Music Journal* 26(3): 11–22.

CHAPTER 3

EARLY HARDWARE AND EARLY IDEAS IN COMPUTER MUSIC: THEIR DEVELOPMENT AND THEIR CURRENT FORMS

PAUL DOORNBUSCH

THE great adventure of music in the 20th and 21st centuries is the use of computers. There have been enormous challenges to overcome for the pioneers of computer music, from original and unique technical requirements to new aesthetics. The result of these developments is now a dominant musical activity. There are many ideas and events in the early practice of computer music that are still evident today in one guise or another.

There is no linear trajectory or chronology to the history of computer music; it consists of a disparate conglomerate of discrete events. However, it is this milieu of events that makes computer music possible. There have always been those musicians and composers who engage with the latest technical advances, whether they are drawn steel strings, advanced mechanical instrument construction possibilities, or general-purpose machines such as the computer. These people have imaginative, creative uses for the technology, and they use it to expand their creativity.

While now routine, there was a time when the use of computers to make music was, if not exactly outrageous, then certainly highly unusual and controversial. This seems incongruous when we realize that composers and musicians have *always* embraced new technology to creative ends. While some traditionalists still find the use of computers in music abhorrent, to others it is an inevitable development of sublime beauty. For the enthusiastic, computer music still maintains an aura of a particular aesthetic and technical possibilities. For most others, computers in music are a ubiquitous fact, from CD playback to digital recording, editing, and production techniques.

Whatever the case, it was mostly a trail of small and uncoordinated steps that led to this use of computers in music. Often enough, a technical development in one area would spur a creative development in another, and sometimes a creative need would encourage a technical response. These developments were usually not seen at the time as scientific, artistic, or commercial. Ideas relating to complex compositional techniques, sound synthesis, musical instrument design, microtonality, and the enthralling power of the computation machine all contributed to a climate in which something called "computer music" emerged.

However incomplete this narrative must be, as a complete treatment would take a book, some of it will be of intentional developments, but other parts will be engineering advances, which were sometimes accidentally useful in a musical context. Oscillators and waveform generators, noise generators, filters, random number generators, numerical techniques, algorithm development, and so on are all examples of technical developments that found use in musical contexts. It is easy, from this artistic and historically distant standpoint, to devalue computer music's initial buzzes, squawks, and arcana as irrelevant. Nevertheless, an examination conscious of how the current artistic, scientific, and consumer-oriented reality has been shaped by the past, and still expresses many of the initial ideas from the past, would do well to note the achievements, effort, and dedication of these pioneers for it is their achievements that are the basis for what we have today.

1. Early Machines, Developments, and the First Steps in Computer Music

Early computer music developments were bound to the early machines on which they occurred. These computers were much more primitive than what is commonplace today and were what are now called mainframe computers. Typically, they occupied a large room, used vacuum tubes (valves) or discrete transistors as logic devices, were very slow by today's standards, and often ran in a time-share environment in which users submitted their program, or *job*, and returned later (usually the next day) to collect the output of their program. They did not have screens to interact with the user; they were typically controlled by a console with switches, lights, and buttons; and the data and programs were stored on punched-paper tape, punched cards, or magnetic tapes mounted on tape drives the size of large refrigerators. Memory was typically cathode ray tubes, mercury delay lines, magnetic core memory, and so on and usually only a few kilobytes. Mainframes and the slightly more friendly minicomputers were the typical computer until the very early 1980s.

Australia's CSIRAC (Council for Scientific and Industrial Research Automatic Computer) was the first computer to play music,[1] but through a series of conservative administrative decisions and missed opportunities it contributed little to what we now know as computer music. CSIRAC was developed by the Australian Commonwealth organization CSIR (Council for Scientific and Industrial Research), and it played music publicly in November 1951 at Australia's first computing conference, although the musical programming of CSIRAC had been occurring for about a year or more. CSIRAC, now regarded as the fourth or fifth all-electronic digital computer in the world, was programmed by Geoff Hill to play popular tunes of the day, "Colonel Bogey," "Girl with Flaxen Hair," and so on. In 1957, another programmer, Tom Cherry, extended the music programming on CSIRAC and developed a program that would take a "score" tape (punched-paper tape) as input to be performed. CSIRAC was one of the very earliest computers and making it play music was a feat. Some of CSIRAC's details are as follows: it used over 2,000 "valves"; required 30 kW to run; had 768 ten-bit words of memory (one word was equal to 2 "bytes") and 2,048 words of disk storage; had a major-cycle frequency of 0.001 MHz and a computational power of about 0.0005 millions of instructions per second (MIPS); occupied 45 m^2; and weighed 7,000 kg.

CSIRAC, a primitive computer, had limited interaction possibilities, and one of them was a loudspeaker used by sending raw pulses from the serial bus to make what the programmers called a "blurt." This was used to indicate that a program had terminated or as a debugging aid to indicate the part of the program being executed. A few pulses were sent to the speaker to make the sound. Hill, Australia's first software engineer, came from a musical family and had perfect pitch. It would have been natural for him to ponder if the blurt could be turned into a steady tone by sending pulses to the speaker at a regular period. This was not as simple as it sounds because CSIRAC was a low-speed (1-kHz major clock), serial architecture computer, and each of the memory locations in the mercury acoustic delay line memory took a different *time* to access. Thus, programming pulses to arrive at the loudspeaker with a regular period (to make a steady tone) took considerable programming gymnastics and cunning. Only the few best programmers were able to achieve this in the fifteen years of CSIRAC's service. Because the sound generation mechanism was not using a digital-to-analog converter (DAC), there were no variable sound synthesis possibilities. One important characteristic of the musical activity was that it was all real time, and it was possible to make limited changes in tempo and pitch from the console while the piece was playing. It would be many years before computer music was again a real-time activity. Unlike the computers that immediately followed it, CSIRAC was operated by a single user who sat at the console and interacted with it, much like today's personal computers (PCs).

CSIRAC, developed as a scientific tool, found some use as a musical instrument but because of the lack of managerial enthusiasm for the activity, composers were never involved; thus, the musical activity was not developed further. There were early attempts by the engineers to do more with the music, but funding was directed elsewhere, and the computing activities were slowly reduced until CSIRAC was transferred from Sydney to Melbourne in 1955 and resumed service at The University of Melbourne. Musical activity also occurred there, particularly with a new program by Cherry, but there was a general ignorance of the significance of the development, and Percy Grainger, an adventurous local composer, was never put in touch with the machine despite regularly walking past the computation laboratory.

The real groundbreaking phase of computer music developments began at Bell Laboratories, where Max Mathews, with the support of John Pierce, developed the MUSIC-N family of music programming languages. Bell Labs, with vast research resources, was interested in sound research and computers for telephony usage; while related to this, the music activity was unofficial. It was in 1957 when Mathews finished MUSIC I, which was a basic sound-generating program for the International Business Machines (IBM) 704 mainframe. A key hardware development that was useful for Mathews was the DAC, which was not available on CSIRAC. Today, DACs are built into every computer and allow for digital information to be turned into analog voltage and sound. The IBM 704 had 4,096 words (of 36 bits) of magnetic core memory, hardware floating-point support, magnetic tape storage (5 megabits per tape), and magnetic "drum" storage and could perform over 4,000 multiplications per second or almost 40,000 simpler operations per second. It was considered an important and leading-edge machine.

The MUSIC-N languages developed from MUSIC I in 1957 to MUSIC V in 1968, and each new version included a number of advances, many of which are still evident in music software. MUSIC I was a monophonic program with only triangle waves for sound output, but importantly it took as input a *score* file of instructions of what to play. This is similar to what happened with CSIRAC in Melbourne at the same time; the Music Programme would play a score tape (of punched paper) in real time. MUSIC II in 1958 was a little more advanced, but MUSIC III (for the IBM 7094), in 1959, was a major step forward and introduced the important concept of modularity with the unit generator or UGen. This is the sort of nitty-gritty idea that in hindsight seems unreasonably wise for its day, along with the score and orchestra files for sound events and sound synthesis, respectively, and the table lookup oscillator. A UGen is a predefined unit of functionality, such as an oscillator, envelope generator, filter, audio input or output, and so on. UGens are combined in the orchestra file to make *instruments*, which create the sounds for the events defined in the score file. Some UGens take audio input, and they all take control input.

All of the MUSIC-N programs up to and including MUSIC IV were written in assembly language that was machine specific. This means that when the computer changed, the software had to be rewritten. MUSIC V was written in 1968 for the IBM System/360, but it was written in FORTRAN, a popular high-level language that was portable. The IBM S/360 was the first machine to standardize on 8-bit bytes and 32-bit words, and while it came in many variants, it typically had about 4 MB of disk storage, 4–8 KB of core memory, tape storage, and punched-card readers. This FORTRAN version of MUSIC became popular with researchers and universities as Mathews gave it away and it was relatively easy to make it work on many other and different computers. Later, MUSIC 10 was developed from this version to run on the Digital Equipment Corporation's (DEC) PDP-10 mainframe computers. However, while achievable, this was not trivial as the program was stored on over 3,000 punched cards.

While the popularity and availability of MUSIC V was a great advance in computer music, it was still very difficult to use. Often, computers were not available, a composer had to take a box of punched cards of the score and orchestra files to a computing facility where the job would be run, usually overnight because it could take many hours to compute a minute of audio. If there was an error, it would need to be corrected and resubmitted. Once the program ran properly, the composer would be handed a magnetic tape, but this was a digital tape of *samples.* Then, the composer would usually have to go to another facility, perhaps a long drive away, to have the digital tape played through a DAC and recorded to an analog tape. Only then, several days to several weeks after finishing the program, would the end result be heard.

One of the key decisions made by Mathews and Pierce at an early stage was to have composers and musicians involved in the program. They had met Milton Babbitt in 1959 along with Vladimir Ussachevsky, James Tenney, Otto Leuning, and later Edgard Varèse. James Tenney joined Bell Labs in 1961 to work on psychoacoustics, but in reality he worked on computer music and stayed until 1964. The input from musicians greatly helped shape the development of the MUSIC-N languages. For example, composers wanted polyphony, variable tunings, flexible and variable timbres and timing, and so on. This directed the software development effort. Tenney was the first recognized composer to work at Bell Labs, and he produced some of the important work there, including *Analog #1: Noise Study* (1961) and *Four Stochastic Studies* (1962). *Analog #1: Noise Study* used MUSIC III to synthesize various kinds of controlled noise, and the form is also realized with a random number method. The *Four Stochastic Studies* required that he write a computer program, PLF 2, to compose them.

The MUSIC-N languages have been very influential because of their extremely elegant concepts: the UGen concept, the orchestra file that takes (time-varying) parameters for the sound synthesis, the score file that defines the timed events with the sounds from the orchestra file, and the table lookup oscillator. Indeed, the plug-ins of today are logically very similar to a MUSIC-N instrument: both output sounds in samples, and these may be mixed and controlled with pitch and duration commands; also they both take parameters (automation) to modify the sounds. Another influence is with the Moving Picture Experts Group 4 (MPEG-4) specification for the Structured Audio Orchestra Language (SAOL), a language for synthesis and signal processing that borrows greatly from MUSIC-N.

Another important development, at about the same time as MUSIC-N, took place at the University of Illinois, where Lejaren Hiller and Leonard Isaacson worked on computer-assisted algorithmic composition. In 1955, Hiller and Isaacson undertook the first experiments in computer-generated music by applying musical rules and later randomness. Hiller and Isaacson used the ILLIAC I computer, which was similar in some ways to CSIRAC, with 2,800 valves or vacuum tubes, storage of 1,024 (40-bit) words in memory (5 KB), magnetic drum storage of 12,800 (40-bit) words (64 KB), and punched-paper tape use.

Hiller asked the important question, "Why would anyone want to make music with a computer?" The answer he supplied is that because computers are excellent at organizing and selecting data, and this is similar to at least some aspects of composition, so computers should be useful for composition. After some initial experimentation with writing parts of the ILLIAC String Quartet, he concluded that the most successful results were obtained using controlled randomness and applying general abstractions to compositional problems rather than applying music theory rules. This idea was enthusiastically used (and separately arrived at) by Iannis Xenakis at about the same time and later by Gottfried Michael Koenig. Of course, there were no graphical displays or MIDI (Musical Instrument Digital Interface); musical information was coded as numbers and entered into programs via punched cards or punched-paper tape. The output would be printed (possibly after being first output to tape) as numbers representing notes and durations and then transferred to musical notation by hand.

2. Computing Architectures and How They Changed Music

Part of the history of computer music is also the history of computing because hardware and software developments in the computing world directly affected the computer music world. As discussed, the nature of computing in its early days—mainframe computers in time-share facilities, punched-card inputs, and fanfold printouts or (nine-track) digital magnetic tape output—was also part of the nature of computer music. These limitations of the technology, and the lack of what is now called "user friendliness," had a direct impact on what sort of computer music was created. This is also true as computing developed, and a brief overview of computing architectures will help this discussion. The performance of computers, especially older and larger machines, is difficult to quantify in modern terms.

Often, there were architectural differences (e.g., overlapping instruction execution and memory accesses) that allowed for performance greater than the “cycle time,” memory access time, or central processing unit (CPU) speed would simplistically indicate. To give an approximate idea of the performance, there will be ratings of millions of instructions per second (MIPS) for some computers. While these are sometimes controversial measurements, seen as flawed or misleading and often meaningless, they are only an approximate indication of relative performance and sufficiently illustrative for the current purpose.

A. Mainframe Computer

A mainframe is a very large computer, typically these days with many CPUs and large storage capacity, along with large data input and output (I/O) capability. Mainframes have always had large word sizes of, typically, 32–36 bits, and large storage capacity (initially many megabytes of data, even if it was relatively slow magnetic tape), and as much central memory as possible, from a few kilobytes at the beginning. These are what governments and large companies use for their data processing. They have very secure operating systems, and while the CPU may be no faster than what is now in a PC, the data processing capacity is vastly superior because of the I/O speed (i.e., much faster memory, disk, and network access). This of course makes them very expensive and large, requiring temperature-controlled rooms and so on. Mainframes from the 1960s included very advanced concepts in their architecture that have taken several decades to appear in PCs today. Mainframes typically have proprietary operating systems that may be arcane for programming, but their reliability, security, and I/O capacity makes them attractive in certain situations, such as banking, for which some mainframes have operated for ten years without interruption, even when undergoing maintenance or upgrades.

The first computers were basically mainframes, even if they did not initially have a time-sharing operating system. Examples of mainframe computers are the IBM 704 (1954), IBM 1401 (1959), IBM 7040 (1961), IBM S/360 (1967), IBM zSeries (current), and many computers from the likes of Honeywell, Control Data Corporation, Burroughs, Amdahl, Hitachi, and so on. Today, mainframes are mostly used to serve data to other computing systems. As an indication of the performance of some of these machines, an IBM 704 was capable of 0.006–0.04 MIPS, and the IBM S/360 came in different configurations from about 0.025 to 1.25 MIPS. By the mid-1970s, mainframes were typically capable of a performance of about 1–5 MIPS.

B. Minicomputers

Minicomputers were developed in the 1960s and became popular in the 1970s. Minicomputers were smaller than mainframes and took up one or two electrical cabinets, about the size of one or two large refrigerators. They were smaller than mainframe computers in others ways, such as being 8- or 16-bit computers and having lower memory, storage, and I/O capability. The first minicomputer, the PDP-1, was developed by DEC. Because they were relatively low cost, minicomputers were popular and found their way into university computing and research departments as well as industry and commerce.

There were many new computing applications developed on minicomputers, and music applications were no exception. The DEC PDP-11, PDP-15, and VAX-11/780 computers became a staple in universities, and the MUSIC-N languages made a natural progression to them. With the differing configurations available, there was a range of performance for all of these machines, but the PDP-11s were capable of about 0.3–1 MIPS, and the VAX-11/780s were capable of about 0.5–1 MIPS. Minicomputers also developed sophisticated operating system environments, which were multiuser and multitasking—UNIX was developed on, and for, minicomputers. DEC, IBM, Hewlett-Packard, Honeywell-Bull, Data General, and others all manufactured minicomputers for many years. However, with the rise in power of the microprocessor and the PC, minicomputers were not widespread for so long.

C. Microcomputers and Personal Computers

The microprocessor—a single integrated-circuit processor—is the main processing unit of the microcomputer and the PC. Semiconductor fabrication progress and miniaturization allowed the development of a small microprocessor on a single chip in the very early 1970s. These were very limited, but by the mid-1970s generally useful microprocessors were readily available. Because they were small and inexpensive, computers built around these microprocessors were regarded in their early years as little more than toys or curiosities for hobbyists. Often sold as kits and with as little as a few hundred bytes of memory, they also usually had no I/O devices other than switches and lights and used cassette tapes for storage. The KIM-1, for example, had an 8-bit 6502 microprocessor that operated at 1 MHz, 1,024 bytes of memory, a cassette interface for storage (very slow), a hexadecimal keypad for input, no power supply or case, and a cost of U.S. $245 in 1975. Many single-purpose business machines were developed around microprocessors (e.g., the dedicated word processor), but these were eventually supplanted by the PC. As semiconductor memory became less expensive, along with floppy disk storage and the increasing power of microprocessors, these small machines became popular as home devices; such computers were the Apple II, Commodore 64, Atari, BBC Micro, Tandy TRS-80, and so on. These used a microprocessor from either Intel (typically an 8080), Motorola (often the 6502 or 6800), or Zilog (Z80) and had a simple operating system that allowed for program launching and directory/file listing. IBM entered the microcomputer market relatively late in 1981 with the IBM personal computer (PC), based on the Intel 8088 microprocessor. In the early to mid-1980s, microcomputers from, for example, IBM, were capable of about 0.25 MIPS, while the Apple Macintosh was capable of about 0.5 MIPS. The term *personal computer* has replaced the term *microcomputer*, and they have developed remarkably from their humble beginnings, to the point at which now PCs are the dominant computing force, with PC technology affecting and now appearing in mainframes and supercomputers. Now PCs have the power to process many millions of complex mathematical operations a second (5,000–50,000 MIPS), far outstripping mainframes and minicomputers of just a few years ago. This has had remarkable consequences for computer music.

Musicians and composers usually like to hear the results of their work as they do it. However, in the time of mainframe computers, computer music was an arduous activity, and it sometimes took weeks to hear the results, but the allure of computer music was so strong they persevered. A step closer to the real-time ideal was taken with the development of minicomputers. Barry Vercoe ported MUSIC-IV to the IBM S/360 and later, in the early 1970s, ported that version of MUSIC to the PDP-11 minicomputer. In the process he consolidated several branches and variations of the MUSIC-N programs and enhanced them with the addition of "control rate" (*k-rate*) capability. The concept of control rate is a rate of change lower than the sampling rate, that is utilized for parameter change in defined instruments. This is useful for computational efficiency because signals that control parameters (such as filter cutoff frequency) need not be calculated at the full sampling rate. The PDP-11s used 16-bit words and had a time-sharing operating system (UNIX), which made them attractive to university departments because they offered enough utility and precision for many tasks even if they were not always the fastest computers available. Thus, they were relatively plentiful in universities, and this special and extended version of the MUSIC IV language, called MUSIC 11, became available to a much wider community of users through its portability, the popularity of the platform, and the spirit of generosity engendered by Mathews and others who freely gave away their software. Along with Vercoe's earlier MUSIC 360 for the IBM mainframes, this meant that MUSIC V and variants were the dominant computer music software throughout the 1970s, and DEC PDP computers (along with some IBM mainframes) were the dominant general computer music platform, although all MUSIC programs and derivatives typically required significant memory resources due to their using long lists of instructions in their processing.

The popularity of minicomputers also encouraged the adoption of the C programming language and the UNIX operating system, which stimulated the development of software to run on them. The C language is a highly portable language, and UNIX was rewritten in C in its early days to encourage its adoption. Both UNIX and C were widely adopted by universities around the world that had minicomputers. The multiuser and multitasking capabilities of UNIX encouraged its use, and its system of using small programs, which could be interconnected to perform a complex task or create a larger system, was popular and highly influential.

At the University of California at San Diego (UCSD) in the late 1970s at the Computer Audio Research Laboratory (CARL), F. Richard Moore developed CMUSIC, a computer music system based on the UNIX approach. Rather than a single monolithic system like MUSIC V, Moore developed CMUSIC as a number of