Craig R. Harris
5769 Liberty Rd.
Ann Arbor, Michigan 48103 USA

A Composer's Computer Music System: Practical Considerations

Introduction

Several factors contribute to the emergence of small but powerful computer music workstations, either as single units in a larger network of computers, or as standalone computer music systems. The current state of hardware and software development has reached a level of sophistication that makes it possible to have access to computer power and programs previously only available on larger computers for a fraction of the price.

Major sound synthesis software systems have been rewritten in the C programming language, contributing to their portability and extensibility. This affects systems built within the UNIX environment, as well as those built upon other operating systems, since C has become a significant development language across operating system boundaries. Recent developments that allow for the writing and reading of standard operating system soundfiles make the necessity of designing and maintaining separate soundfile management systems virtually obsolete. The implementation of a variety of user-interface styles to computer music applications has made the creation environment much more accessible to musicians as a whole, and to composers in particular. These factors, combined with decreasing prices for powerful microcomputers, larger and faster disk drives, and the availability of high-quality, lower-cost sound conversion systems, point in a new direction: powerful, dedicated machines for individual artists, smaller computer music facilities, and larger systems based on networks of semi-independent computer music workstations.

In order to identify the practical considerations pertaining to the implementation of such a system, it is useful to address the issues in the context of an actual computer music studio installation. The system described has been designed and realized, concentrating on building the tools and environment for music creation using "delayed performance," direct synthesis techniques for music generation and sound processing.

The decision to use software sample synthesis as the primary means of generation and manipulation of materials was made because of the extreme flexibility it provides in designing, controlling, and adapting sound. Once the restriction for real-time synthesis or control is removed, it is possible to be as particular about details as the compositional context requires, in generating, mixing, and processing sound as the work evolves. Working in this manner with the rich sound realm made available using these techniques is compositionally and aurally compelling, and not currently attainable using an event-based communications scheme such as MIDI (Musical Instrument Digital Interface).

Basic Premises

During the planning stage of any system implementation it is important to analyze the manner in which the users of the system will be working. The choice of hardware and software, as well as the working environment, needs to be designed to enhance the activity and the artist's natural creative tendencies, rather than force conformity into working and psychological modes that are not appropriate to achieve the maximum results.

Composers are concerned with the manipulation of sound into a comprehensible, musical context. As ideas progress and become refined it is often necessary to rework "sketches," making various alterations to all aspects of the music (e.g., timbre, pitch, rhythm, harmonic and structural elements). A supportive environment needs to be flexible and capable of retaining as much of the generative information as possible, allowing for minimum specifications for changes, and reducing duplication of entry.

Computer Music Journal, Vol. 11, No. 3, Fall 1987,

The system should be easy to program for testing of sounds quickly, allowing the composer to experiment on both the microcosmic and macrocosmic layers of a composition without inordinate time delays. As work on a composition builds momentum, more time is generally required in terms of access to the computer for audition, input, and preparation for large-scale processing.

An important aspect of working in this way is that the composer is designing the *instruments* or sound environment as part of the compositional process. In many ways this defines the generative principles on which the composition is based. It is very much an experimental procedure of creative extensions of the imagination, of sounds, combinations of sounds, and sound events.

A comprehensive system would supply a variety of means for the creation and manipulation of sound. In the ideal, this would incorporate multiple sound synthesis techniques, extensive digital mixing capabilities, various sound manipulation utilities, capacity for postprocessing of existing soundfiles (reverberation, delay, filter, panning), extensions into computer-assisted composition, and an integrated environment in which to work. It is important to build into the design of the system strong support for both interactive and batch mode processing. Flexibility and extensibility are critical issues, and the control needs to be left in the hands of the composer, who must retain the ability to easily design and implement tools that suit the direct requirements of the specific musical context.

Operating Environment

Computer Hardware

The computer hardware for the studio consists of a network of two computers, an Advanced Logic Research (ALR) Dart, and an Altos 986. The ALR is an IBM/AT compatible computer with a 10-MHz clock, and 1 Mbyte of random-access-memory (RAM). The Altos is a multiuser microcomputer, also with 1 Mbyte of RAM, capable of handling up to nine users. Both run on INTEL processors, with the Altos based on the INTEL 8086, and the ALR based on the INTEL 80286, with the 80287 floating-point coprocessor.

The computers are connected by a local area network that permits the ALR to share resources with the Altos. This consists primarily of disk sharing, file transfer resources, remote terminal access, and printer access. Currently there are two terminals, a modem, a letter-quality printer, and a dot-matrix printer connected to the Altos, and a monochrome monitor and mouse connected to the ALR, in addition to the network and sound conversion hardware.

Hard disk storage consists of 66 Mbytes on the Altos, divided between two 33 Mbyte disks, and 100 Mbytes on the ALR, divided between a 60 Mbyte Priam disk drive and a 40 Mbyte SeaGate drive. Archiving is done utilizing a cartridge tape drive or floppy disk drive on the Altos, or 1.2-Mbyte and 360-kbyte floppy disk drives on the ALR.

The ALR system is a completely functional computer music system and is not dependent on the Altos either for sound processing, synthesis, or conversion. The multitasking or multiuser environment has significant advantages, however, even for a system dedicated to an individual artist. In a non-real-time synthesis system there are periods during which one is waiting for the results of the computation. In order to utilize this time one could be preparing further scores and mixes for computation upon completion of the current operation. The benefit in this situation, then, is not merely to be able to compute multiple soundfiles simultaneously, but also to be able to prepare several procedures while the computer is still computing current soundfiles.

Sound Conversion Hardware

Sound conversion is achieved utilizing the DIGISOUND-16 analog-to-digital and digital-to-analog (A/D/A) converter from Micro Technology Unlimited (MTU). This is a 16-bit stereo conversion system supporting speeds of up to 50 Ksamples per second (Ks/S) stereo. This sampling rate has not been reached on direct-to/from-disk transfer on the ALR, but tests have demonstrated that at least 32 Ks/S stereo A/D/A is consistently and dependably functional utilizing standard disk controllers,

even without track buffering on the controller. The DS-16 connects to the ALR utilizing a single IBM PC/XT/AT compatible interface/controller board, providing both programmed input/output (I/O) and direct memory access (DMA) modes of transfer between the converter and the computer. Software is provided for direct-to/from-memory and direct-to/from-disk transfer by MTU. The A/D/A direct-to/from-memory conversion software was written by Gordon Rudd (Design Science), and Russell Pinkston added the direct-to/from-disk software.

Operating Systems Software

The Altos runs under the Xenix 3.0a operating system, which is an AT&T UNIX System III derivative. Several Berkeley UNIX utilities are available, and support is present for the C and Fortran programming languages.

MS-DOS 3.20 runs on the ALR computer, with support for the C programming language (Microsoft 4.0) and a macro assembler.

The Music System

One can work in several ways with the system. Software was chosen and tools were developed that would be conducive to creating a unified environment. Support is provided for both textual commands (via the UNIX shell) and menu-driven modes for several operations. All soundfiles can be manipulated by the various processing programs without having to change file formats and headers. Conversion utilities are provided, however, to allow for the network file transfer between the MS-DOS and Xenix operating systems, and for soundfile transfers to other computer music environments.

The programs for synthesis and processing of sound have been written in the C programming language. Soundfiles are written either as standard MS-DOS files on the ALR, or as standard Xenix files on the Altos. The sound conversion software reads and writes MS-DOS operating system soundfiles directly from and to the disk, respectively. The music software is functional on both systems, so it is possible to work on the two computers, sharing various parts of the computational load. The network is not completely NFS (New File System) oriented, but there are utilities providing for access from Xenix onto the MS-DOS partitions on the Altos. In this way it is possible to operate on the same soundfiles, either by working on the shared disk partitions or by sending files across to the appropriate computer for processing as required. Since the programs on both computers are written in Microsoft C, portability of software is maximized, facilitating the maintenance of consistency of software between computers.

Musique Concrète Environment

Sounds can be digitized directly in monaural or stereo and stored for later use. Soundfile manipulation utilities allow one to alter these stored sounds in a variety of ways. Several of these utilities have been adapted from the *Cylinder Contiguous Soundfile System* (*CCSS*), written by Robert Gross at the Eastman School of Music (1980–1982). Other utilities are part of the CMIX environment, or were written specifically for this installation by the author. Some routines perform operations on soundfiles to simulate variable-speed tape recorder functions by altering the sampling speed ratios, either as a one-time, single-rate operation, or utilizing user-defined rates and functions to control the simulation of the frequency alteration. The following list of utilities demonstrates the kinds of resources available. In most cases options are available for the specification of such details as specific durations and skipping (offsets) into soundfiles:

sfchrate — alter pitch/duration of a soundfile
sfchvrate — variable speed version of sfchrate
sfreverse — reverse soundfile
ampplot — display windowed histogram of soundfile
sfheader — display header information about soundfiles
mksfhead — make new soundfile header
sfhedit — edit soundfile header
sfxfer — soundfile transfer utility
sfsplit — channel separation utility

A digital mixing program allows for large-scale mixing of multiple soundfiles, with necessary amplitude scaling, fade-in/fade-out, channel send, and looping specifications. This mixer was adapted from the CCSS mixer, which operates on all active soundfiles "simultaneously" before writing to the disk. Two modes of operation are available, a *quick* mode that mixes soundfiles directly and a *complex* mode, which incorporates amplitude factors and fades.

Sound Resources—The Soundfile Library

As part of the music environment, a catalog of soundfiles is available for use either in stored form, in processed form, or as source material for affecting various aspects of other sounds. The Soundfile Library consists of digitized sounds, sampled as single channel (monaural) files. They are kept in a local directory that can be readily loaded or unloaded as required, according to desire and disk space requirements. This directory is on a read-only disk partition that is shared between both computers, making the sound resources available for multiple processes.

The percussion soundfile library consists of 35 digitized percussion soundfiles, including a tam-tam, gongs, cymbals of different sizes struck with various mallets, crotales, triangles, metal plates, a bell tree, temple and wood blocks, and wind chimes.

In addition to the standard percussion soundfiles, a variety of instruments have been digitized and are available. The list includes plucked and bowed strings, vocal tones, woodwind, brass, and piano tones. Samples were taken from varying ranges to retain as much of the formant characteristics as possible when processing these sounds.

Sound Synthesis Environment

Sound synthesis has been achieved by adapting the CMIX software, written and provided by Paul Lansky (Princeton University), to work on the MS-DOS and Xenix operating systems. Using CMIX one is able to directly synthesize sound according to the user-defined instrument and score specifications. Furthermore, postprocessing of existing soundfiles is possible by writing programs in C to link with the CMIX libraries. These can read in soundfiles and either operate on them directly or create new soundfiles. Digital mixing is also available, and in some cases it is more desirable to use CMIX as opposed to the CCSS mixer. CMIX mixer operates on each consecutive soundfile, with the capability of performing more complex operations on each soundfile regarding envelope shaping and channel specification.

The CMIX programs are extremely flexible in terms of the available synthesis techniques, and in the ease of implementing additional procedures for sound manipulation. The system provides additive synthesis, subtractive synthesis, and frequency modulation, and utilizes linear prediction techniques for analysis, resynthesis, and cross synthesis.

The CMIX libraries contain routines for synthesis, processing, and file handling, and provide command-level procedures for operations such as the defining and plotting of waveshape and envelope functions. Users can easily build their own libraries for providing access to often-used processing routines.

CMIX is an environment as well as a package for music synthesis. The level of control is largely left to the user, and in some ways the degree of efficiency is also determined by the user's understanding of the routines driving the system. The more one knows about how the programs work and is able to define the limits and terms of the intended goals, the faster and more effectively one achieves results. Naturally, this is true in many systems, but in CMIX the level of control extends from the higher-level instructions for pitch, rhythm, duration, and spectral properties, all the way to the lower-level routines that control such fundamental operations as I/O instructions and read/write modes.

For example, depending on the operations one wishes to perform, an instruction to write destructively to the soundfile can save substantial computation by a call to the **wipeout** write routine rather than the **addout** write routine. This not only saves the time of cleaning out a soundfile or creating a new soundfile (writing zeroes before operating), but also an extra **add** operation for every sample.

Speed of operation can also be enhanced by such advance planning procedures as distributing sound-

Table 1. Sound synthesis timing specifications

Processing Category	*Write Mode*	*Duration*	*Channels*	*Sampling Rate*	*Timing*	*Performance to Computation Ratio*
Synthesis	Addout	1	1	25 Ks/S	18"	1:18
		3	1	"	55"	"
		5	1	"	1'32"	"
		10	1	"	3'4"	"
		1	2	"	24"	1:24
		3	2	"	1'12"	"
		5	2	"	2'	"
		10	2	"	4'3"	"
	Wipeout	1	1	"	16"	1:15
		3	1	"	45"	"
		5	1	"	1'16"	"
		10	1	"	2'30"	"
		1	2	"	20"	1:20
		3	2	"	1'	"
		5	2	"	1'42"	"
		10	2	"	3'22"	"

files between multiple physical disk drives if they are available. While at the outset this may seem like a less important detail, upon examination this proves to be a significant point. A simple test of a 5-sec mix that took 1 min 57 sec to compute on a single disk drive took only 24 sec reading from one disk and writing to another. This results in a time-savings ratio of 5:1.

While it is difficult to discuss computational and timing specifications for a system as it would perform in a real production context, it is useful to examine the characteristics of the installation within specified boundaries, if only to clarify the working environment and capacity of the system. Tables 1–3 show some timings associated with various operations.

For purposes of comparison all timing specifications have been calculated at 25 Ks/S sampling rate. The system has a capacity for a total of 10 min of monaural or 5 min of stereo sound on each of the two 30 Mbyte partitions of the Priam disk. It is possible to have 6.66 min of monaural or 3.33 min of stereo sound on each of the two SeaGate 20 Mbyte partitions. The operating system and software utilizes less than 5 Mbytes of this available space, leaving a substantial amount of space for sound storage remaining.

Examination of Tables 1–3 shows that computation times are within reasonable limits for a delayed performance computer music system, for testing as well as generation of actual compositional material. It is also evident that the efficiency can be substantially increased with even a basic understanding of the factors involved. It should be stated that when these timing specifications were calculated, the floating-point resources were not yet fully im-

Table 2. Digital mixing timing specifications

Processing Category	*Read/Write Mode*	*Duration*	*Channels*	*Sampling Rate*	*Timing*	*Number of Input Sound Files*
		Write to same disk				
CCSS Mixer	Quick	5	2	25 Ks/S	1'57"	1
	"	4	2	"	2'3"	2
	"	4	2	"	2'30"	4
	Complex	5	2	"	2'27"	1
	"	4	2	"	2'43"	2
		Write to different disk				
CCSS Mixer	Quick	5	2	25 Ks/S	24"	1
	Complex	5	2	"	51"	1
	"	4	2	"	1'44"	2
CMIX Mixer		5	2	25 Ks/S	1'	2
		4	2	"	2'30"	4

Table 3. Soundfile Utility Timing Specifications

Processing Category	*Duration*	*Channels*	*Sampling Rate*	*Timing*	*Operation*
			Write to same disk		
sfchrate	4	1	25 Ks/S	1'23"	transpose 1 octave up
	4	1	"	2'4"	transpose 1 octave down
	4	2	"	3'42"	transpose 1 octave up
sfreverse	2	2	"	52"	
			Write to different disk		
sfchrate	4	1	25 Ks/S	1'9"	transpose 1 octave up
	4	1	"	1'40"	transpose 1 octave down
	4	2	"	2'28"	transpose 1 octave up
sfreverse	2	2	"	19"	

requirements of the composer utilizing computers as an instrument. A single synthesis technique, or a single "mode" or means of operating with sound and soundfiles doesn't respond to the various ways of working that are conducive to artistic creation. It is critical that the technology, including the hardware, software, and operating environment assist the artist and enhance the creative activity, accommodating the various conditions present during the evolution of a composition.

The availability of software and hardware on both large and small systems, and the portability of soundfiles between systems offer new possibilities for composers who wish to work with delayed-performance synthesis techniques. One can take advantage of an independent system in a standalone environment, and also benefit from the use of a computer or site with more powerful computation power and a wealth of additional synthesis and manipulation programs.

Considering the current state of development as described, and the fact that the fully equipped standalone computer music system such as this ALR AT can be purchased for under $10,000 with sound conversion, it is practical, realistic, cost-effective, and extremely desirable to work with this kind of system, either as an individual composer's creative instrument, or as part of a larger system designed to service many composers.

Acknowledgments

I would like to thank Paul Lansky for providing CMIX, and Robert Gross for CCSS. Thanks are also due to David Cox and Larry Isaacs at MicroTechnology Unlimited, Gordon Rudd of Design Science, and Russell Pinkston for support in many aspects of the development on this project.

References

Harris, C., and A. Brinkman. 1986. "A Unified Set of Software Tools for Computer-assisted Set-theoretic and Serial Analysis of Contemporary Music." In P. Berg, ed., *Proceedings of the 1986 International Computer Music Conference*, pp. 331–336.