Computer-Music Interfaces: A Survey

Article *in* ACM Computing Surveys · June 1985

DOI: 10.1145/4468.4470 · Source: DBLP

Bruce Pennycook
University of Texas at Austin
47 PUBLICATIONS **290** CITATIONS

Computer–Music Interfaces: A Survey

BRUCE W. PENNYCOOK

Department of Music and Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada

This paper is a study of the unique problems posed by the use of computers by composers and performers of music. The paper begins with a presentation of the basic concepts involved in the musical interaction with computer devices, followed by a detailed discussion of three musical tasks: music manuscript preparation, music language interfaces for composition, and real-time performance interaction. Fundamental design principles are exposed through an examination of several early computer music systems, especially the Structured Sound Synthesis Project. A survey of numerous systems, based on the following categories, is presented: compositions and synthesis languages, graphics score editing, performance instruments, digital audio processing tools, and computer-aided instruction in music systems. An extensive reference list is provided for further study in the field.

Categories and Subject Descriptors: J.5 [**Computer Applications**]: Arts and Humanities—*music*

General Terms: Design, Languages

Additional Key Words and Phrases: Composition and synthesis languages, computer-aided instruction in music systems, design principles, graphic score editing, real-time performance systems

INTRODUCTION

The User Interface in a Music Context

There have been numerous studies addressing a variety of difficult aspects of man–machine communications and user-interface design. The types of tasks examined in these articles are primarily familiar computing environment activities such as text editing and formatting [Embley and Nagy 1981; Furata et al. 1982; Meyrowitz and van Dam 1982], interactive programming [Sandewall 1978], natural language interfaces [Ledgard et al. 1980], and input strategies involving nonkeyboard input devices such as mice, light pens, tablets, and the various display format strategies associated with them [Baecker 1980]. A special issue of *Computing Surveys* that focused on psychological aspects of human–computer interaction appeared in 1981 [Moran 1981].

A recent article by William Buxton [1983], whose contributions to music interface design are discussed below, offers some pertinent observations regarding the selection of input devices. Buxton summarizes his report as follows:

> When we have developed a methodology which allows us to determine the gesture which best suits the expression of a particular concept, then we will be able to build the user interfaces which today are only a dream. [Buxton 1983]

This comment serves as a useful starting point for this survey. Attempts to design

CONTENTS

and implement new user interfaces for computer music applications have indeed required a fresh examination of the musical activities for which they are intended.

Musicians develop musical skills through formal study and practical application. Musical knowledge is translated into sound through the physical gestures of vocal and instrumental performance or, in the case of electroacoustic music, through signal generation, amplifiers, and finally loudspeakers. Much of the musician's craft is absorbed unconsciously as part of the music-making experience. Unraveling these complex interrelationships of knowledge, experience, and gesture poses a formidable challenge. Codifying the web of musical attributes loosely referred to as *musicianship* is further compounded by the fact that each and every musical style is a product of unique sociotemporal forces.

From these notions, some critical observations may be drawn regarding the nature of music information as it applies to the design and implementation of user interfaces:

(1) Little is known about the deep structures of musical cognition.
(2) Many of the interface requirments for a music system are unique to music and have no equivalent within the general realm of computer usage. Furthermore, composition of music, performance of digital musical instruments, and manipulation of recorded sound each pose new and substantially different problems for the user-interface designer.
(3) Composers, performers, and recording engineers usually exhibit highly idiosyncratic work habits, which ultimately must be accommodated by the user interface(s) on an individual basis.
(4) As a result in part of the rich, descriptive vocabulary of musical concepts, information about music and sound is not easily translated into computer data that are accessible to and readily manipulated by musicians.

The work reported in this survey represents only a small fraction of the diverse and relatively voluminous efforts in this field. It is important to note that most of the contributions to user-interface design and implementation have been developed by musicians with varying degrees of training as computer scientists according to their own needs and musical perspectives.

It is difficult to align the information in a survey such as this with each individual reader's background and interests. The path chosen here is somewhat biased toward a general treatment of the relationship between musicians and machines rather than a documentation of the various technologies that have been used to construct computer music interfaces.

The objective is to establish some basic criteria for music interface design and to support these criteria with discussions of a few implemented systems.

Partitioning the Discussion

For the purposes of this survey, the discussion of music interface design has been subdivided into the following categories:

- technical considerations,
- music user interface concepts,
- design strategies,
- systems survey.

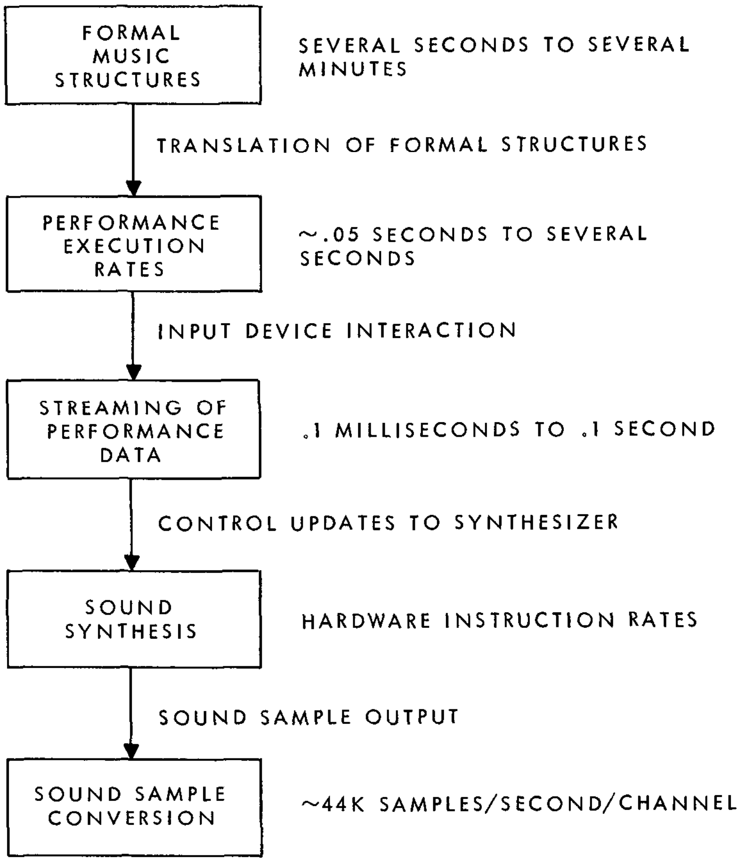
FORMAL MUSIC STRUCTURES
SEVERAL SECONDS TO SEVERAL MINUTES
TRANSLATION OF FORMAL STRUCTURES
PERFORMANCE EXECUTION RATES
~.05 SECONDS TO SEVERAL SECONDS
INPUT DEVICE INTERACTION
STREAMING OF PERFORMANCE DATA
.1 MILLISECONDS TO .1 SECOND
CONTROL UPDATES TO SYNTHESIZER
SOUND SYNTHESIS
HARDWARE INSTRUCTION RATES
SOUND SAMPLE OUTPUT
SOUND SAMPLE CONVERSION
~44K SAMPLES/SECOND/CHANNEL

Figure 1. Throughput rates for real-time music systems.

The section on technical considerations presents certain factors that must be satisfied in real-time computing environments, although many of the interfaces presented in this survery are software packages operating in nonreal time.

In Section I, first the unique nature of musical tasks as opposed to general computing tasks is illustrated, and then four innovative systems are discussed to establish some basic principles.

Section II has been subdivided into five broad categories: composition and synthesis languages, graphics score editing tools, real-time performance systems, digital audio processing tools, and computer-aided instruction systems. There is a great deal of overlap within these five divisions; for example, graphics score editing tools are used for composing as well as for transcribing and printing music notations. This multiplicity of purpose is most pronounced in recent commercial instruments such as the Fairlight Computer Music Instrument (manufactured by Fairlight Instruments Limited, Sydney, Australia).

Technical Considerations

The system requirements for music interfaces vary according to the task and to the nature of the musical environment. In a real-time setting the following hierarchy of timing requirements exists, as shown in Figure 1: (1) control of formal music structures such as instrumentation, number of output channels; (2) performance execution from the input devices; (3) merging the performance and synthesis data into a continuous stream and transmitting it to the synthesis device; (4) computation of the sound samples; (5) output to and from audio conversion subsystems. Integrating the minimum response times at all levels of the system can pose substantial throughput problems. Throughput reliability can be compounded by the nature of musical gesture in that performance data often arrives in "bursts."

Consider the example of a pianist playing a different chord once each second. Although the *average bandwidth* is relatively low, the instantaneous bandwidth of each chord is directly proportional to the total number of elements, which, in combination, produce the actual sound. Thus only the *worst case throughput* can be considered in the interface design.

Provisions must also be made for the integrity of real-time requests at all levels of the system, as shown in Figure 1. A detailed study of the requirements for real-time control of all of the contributing factors outlined above exceeds the scope of this article. The reader is directed to Mathews and Bennett [1978], Moorer [1982a, 1982b], and Loy and Abbott [1985] for information on the technical issues.

1. MUSIC USER INTERFACE CONCEPTS

Interface specifications vary dramatically with respect to the five levels of communication shown in Figure 1. This makes it impossible to devise evaluation schemes that can be generally applied to all music interface specifications. Unlike text-editing environments, in which measures of productivity can be gathered empirically, in most musical settings *productivity* and *aesthetic* value become hopelessly confused. How can we measure the effectiveness of a piano keyboard? Measures used in evaluating text editing environments (e.g., correct keystrokes per unit time [Embley and Nagey 1981]) do not apply. Instead, the interface designer must rely on individual assessments by users performing a variety of tasks. The choice of tasks is necessarily limited by the biases inherent in the system and by the musical preferences of the user. A user interface that satisfies the needs of one musician in an efficient, well-ordered way may be awkward or even counterproductive for another musician.

Attempts to identify and codify the relationship of musical activities and computer music interfaces have appeared in Smoliar [1973], Buxton [1978], Vercoe [1975], Chadabe [1977], Laske [1977, 1978], and Hanes [1980].

So far, no generally applicable theories have emerged. There is, however, a consistent theme: Musical actions (whether encoded as musical symbols or performed on instruments) and music cognition are both based on a hierarchy of temporal structures. The design of a computer music interface or performance system requires careful examination on many temporal levels.

1.1 An Examination of Some Musical Tasks

1.1.1 Music Manuscript Preparation

Certain musical tasks can be described as a sequence of simple actions with specifiable goals. For example, transcribing individual parts for the members of an ensemble from the composer's master score could be summarized as follows:

- Select suitable manuscript paper for the current instrument (considering the number of staves, staff size, spacing between staves, etc.).
- Determine if the instrument requires a transposed part (i.e., a part written in a different key as required by transposing instruments such as clarinets and French horns).
- Choose the correct number of measures per page, so that page turns are preceded by a rest of sufficient duration.
- Choose the corrrect number of measures per line of music to facilitate reading and visual interpretation.
- Select the correct pen nib sizes for the manuscript.
- Prepare the manuscript paper by adding the appropriate clef, key, and time signatures.
- Copy the part from score.
- Check the transcription against the original part in the score and make corrections as necessary.

This process is essentially a specialized form of document preparation. Although the general problem of music manuscript preparation is somewhat difficult because of the idiosyncracies of music typesetting, the basic tasks of the music copyist should require an interface similar to display-oriented text editors and formatters. Yet there are very few music manuscript preparation systems general enough to accommodate a wide variety of musical styles and forms and produce camera-ready copy (see Smith [1973], Byrd [1977], Maxwell and Ornstein [1983], and Hamel [1984]).

One of these systems, *MS*, has taken Leland Smith, an accomplished performer, composer, and pioneer in the development of score-processing software, over a decade to develop and refine [Smith 1973]. Part of the problem stems from the rapid changes in display technology and the resultant expectations of the users. The principal difficulty, however, lies in the nature of musical notation. The size, location, and orientation of each musical symbol must be adjusted so that the visual appearance of the manuscript is not only accurate, but also contributes to the performer's understanding of the work and his or her ability to execute the symbols as musical gesture. (It is often said by musicians that one can usually guess the composer of a work simply by the *appearance* of the score.)

What appears to be a relatively straightforward (although very long) document-processing problem actually embodies the same aesthetic difficulties as other music-processing tasks. Composers and copyists have their own notions of how a particular musical passage ought to look on the page. Of course, there are certain standards and conventions in the realm of music publishing that are reducible to sets of rules and procedures. These may work well enough for traditional music notation such as a Schubert symphony, but are wholly inadequate for many contemporary scores or for scores of non-Western music such as South Indian classical music. The visual appearance of the score must effectively convey the stylistic idiom and the composer's intentions. Providing the copyist with a computer music typesetting system becomes an immensely difficult problem, which must take into account such vague and flexible notions as musical style and layout aesthetics.

In Section 3.2, several score editors are presented. Each offers certain unique features.

1.1.2 *Music Language Interfaces for Composition*

The problems of musical style and personal preference are most acute in the design of music languages for composition. There are countless anecdotes surrounding the work habits of the great composers. Beethoven would fret over a crucial melodic passage for months and then, having solved it, could complete the entire movement in a very short time. Stravinsky's work scores are not available for study, but it is known that he used liberal amounts of colored ink as a private code for organizing musical structures. One suspects that these examples are not extreme cases, and that individual style and aesthetics are acute considerations in designing music languages for composition.

The relationship of the tasks comprising compositional process to the manipulation of the materials is not readily apparent. The goal—a *composition*—emerges from seemingly arbitrary processes that the composer constructs to suit his or her immediate creative needs. The interface between the composer's imagination and the finished product is a tool that allows the composer to experiment with musical materials while being restricted to the requirements of the medium. For some, paper and pencil suffices; for others a piano serves the purpose. Many contemporary composers work with multitrack sound recording systems and a complex of sophisticated electronic devices (see Chadabe [1977, 1983]).

Identifying composition tasks in terms of goals seems fruitless. We can, however, identify the kinds of tools that composers find useful: musical instruments, music manuscript, graphics, sound recording systems, programming languages, device controllers, etc. Thus the criteria for evaluating the effectiveness of the user interface must be based on a measure of its capacity to adapt to the needs of the composer throughout the compositional process.

1.1.3 Performance Systems

For centuries instrument builders have labored to provide performers with acoustically precise, responsive instruments that enable them to play the best music with the least amount of effort. A poorly constructed violin can be coerced to produce beautiful tones, but it requires much greater exertion and skill by the player. Instrument inventors have also been motivated by a desire to provide composers with new sonic capabilities. The pianoforte was considered to be a novelty in the middle of the eighteenth century, but the expressive qualities offered by control of hammer velocity soon made it preferred over the harpsichord by composers and performers. (Sachs [1940] contains a detailed historical survey of musical instruments.)

Electroacoustic instruments pose new problems for the instrument designer. In addition to certain traditional modes of instrumental performance, electronic instruments offer a wide variety of new possibilities. Many synthesizers are "hardwired," presenting the performer with a set of mechanisms such as keyboards, potentiometers, switches, foot pedals, joy sticks, thumb wheels, etc., which control specified sound-generating and processing modules. More sophisticated synthesizers, including several new entries based entirely on digital technology, permit the user to assign the output of a controlling device (a time-varying direct-current control voltage or a stream of binary data) to different inputs of the modules.

This capacity to reconfigure the controls of the instrument manually or under program control radically alters our notions of performance. For example, advances in the design of electronic piano keyboards, such as those developed by R. Moog,[1] which track pressure sensitivity and motion in two planes along with key position and key depression velocity, provide the performer with unprecedented control mechanisms. These new mechanisms require the performer to develop new technical skills and, more important, new modes of artistic expression.

Computer-controlled analog and digital synthesizers offer another level of performer control. Performance data, synthesis configuration, device assignment, output channel distribution, etc., can all be prepared in advance. A *performance*, then, consists of varying degrees of intervention over the automatic control of the synthesizer; the operator becomes performer, composer, and conductor of all the musical forces at once.

2. DESIGN PRINCIPLES

2.1 Early Systems

The earliest interactive computer music systems were *Groove*, developed by Max Mathews and his co-workers at AT&T Bell Laboratories between 1968 and 1970 [Mathews and Moore 1969, 1970; Mathews and Rosler 1969], and *A Computer Aid for Musical Composers*, developed at approximately the same time at the National Research Council of Canada [Pulfer 1970, 1971a, 1971b; Tanner 1971]. These systems addressed the problems of the user environment in terms of real-time interaction. Both systems provided a basic set of tools for describing, manipulating, playing, and storing musical information. The devices may seem rudimentary compared with current standards of technology. As Figure 2 shows, however, each system provided the composer with a flexible and easily understood work space. In an address to the American Society of University Composers, Mathews characterizes Groove as follows: "The desired relationship between the performer and the computer is not that between a player and his instrument, but rather that between the conductor and his orchestra" [Mathews and Moore 1969a]. Groove provides instantaneous feedback through the visual display device and the loudspeakers. The musician's responses to the feedback are recorded and saved for future replays.

[1] The 100 Series Keyboard Controller, Big Brier Inc., Leicester, N.C., 1982.

USER INTERFACE DEVICES

DEVICE	GROOVE	NRC
VISUAL	CRT:MENUS, WAVEFORMS	CRT/MENU, SCORES,
MANUAL	TTY KNOBS SWITCHES CLAVIER 3-d WAND	TTY LIGHT-PEN
AUDIO	REAL-TIME (analog) STEREO	REAL-TIME (digital) STEREO

Figure 2. A comparison of Groove and the NRC music systems.

The NRC system provided similar types of real-time interaction but with limited digital rather than analog audio synthesis. A number of important features were incorporated which indicate that this system was very much in the vanguard of today's experimental user-interface designs:

(1) Four voices of real-time digital sound output.
(2) A color display that enabled individual music passages (voices) to be color coded during the editing procedure (red for the current voice being edited, blue for all of the background).
(3) Optional simultaneous control of an analog synthesizer called the *Paramus* that could play from the same score as the digital synthesis devices.
(4) Direct hard-copy output of the music in the form of a conventional musical score.
(5) All four voices displayed in conventional musical notation scrolled horizontally across the screen *at the same time that the score was being digitally synthesized.* This is perhaps more astonishing, given the state of graphics systems, computer power, and synthesis capabilities in 1971.
(6) A 61-note organ keyboard that could be performed in real-time producing four voices of notation on the screen (to the best of the system's ability).
(7) Two orthogonally mounted thumb wheels and a mouse for controlling the cursor on the graphics screen (the first application of a mouse controller in music interface design).
(8) A two-handed data entry system, with the left hand controlling a key set selecting note durations for pitches placed on the screen by the right hand, which controlled the cursor position.

The NRC system was well ahead of its time, not only with respect to the production of musical scores and sounds, but in the overall integration of computer graphics, input devices, real-time control, and digital synthesis.

There were other approaches to real-time interaction which produced Piper [Gaburo 1973], Musys [Grogono [1973], EMS [Wiggen 1968], all relying on computer control of analog sound generators rather than the somewhat limited real-time digital synthesis capabilities available at that time. In all cases the user had direct control over sound-modifying parameters.

2.2 The Structured Sound Synthesis Project

An important attempt to systematically investigate man–machine communication within the musical domain was made by William Buxton and a team of research assistants at the University of Toronto in 1978. The *Structured Sound Synthesis Project* (SSSP) borrowed from Buxton's experiences with the NRC system, POD [Truax 1977], and Piper, and from certain concepts proposed by Otto Laske [1978]. Buxton's efforts have established some fundamental criteria for effective music interface design. These first appeared in the report Design Issues in the Foundation of a Computer Based Tool for Music Composition [Buxton 1978]. This report was followed by Buxton [1981], Buxton et al. [1978a, 1978b, 1979, 1980, 1982], and Fedorkow et al. [1978], all of which address user-interface issues to some extent. The major features of the user interface are summarized in the following sections.

2.2.1 SSSP Graphics Display

The graphics interface provides the basis for nearly all interaction with the music system. Commands and graphics actions such as score editing and sound synthesis specification have been organized into a combined iconic selection and typed response menu-driven format. Since one of the primary objectives of the SSSP has been to minimize the learning curve for musically sophisticated but computer-naive users, most of the dialog is system initiated and is couched in familiar musical terms and symbols.

Most communications are executed through manipulation of a cursor controlled by a tablet input device, thus reducing the need for typed input. Actions by the user always invoke a response from the system (unlike UNIX[2] [Ritchie and Thompson 1974], e.g., where an absence of response from the operating system usually means that the input was syntactically correct and that the request is being serviced). The most significant feature of the graphic interface is that all of the actions needed to compose and listen to a musical event appear as symbolically informative images. An example of the display during a session of editing waveforms for the synthesis process is shown in Figure 3.

2.2.2 SSSP Sound Synthesis Interface

Audio feedback from the SSSP is instantaneous. User input and internal data are organized and stored in efficient data structures so that performance specifications, called *m_events*, can be interpreted directly by the synthesizer control processor and subsequently by the synthesizer units. An important feature of the synthesis interface is the generous use of defaults in the data specifications. Users may focus on certain aspects of the composition process, such as entering, testing, and correcting the notes and rhythms, without having to be concerned with other parameters. This approach is very helpful for the novice as well as being convenient for the experienced user.

[2] UNIX is a trademark of AT&T Bell Laboratories.

2.2.2.1 The SSSP Conduct System. Several unique devices and control structures have been designed and implemented by the SSSP team that can be used to control the output of the synthesizer during playback of prepared musical events. Live performance using the SSSP *Conduct* system generally involves the manipulation of these devices in a similar fashion to Mathew's Groove system. As it would be highly impractical to move the graphics display system to each performance site, programs were devised that enabled the user to rapidly execute commands at a standard video display terminal from the following *motion-sensitive* input sources:

(1) two continuously variable sliders;
(2) *x–y* mouse (actually a digitizing pad and tracker);
(3) *x–y* touch-sensitive pad;
(4) clavier (piano) keyboard;
(5) four variable-slope, straight-line segments for control of output rates or amplitude contours implemented in the Conduct software.

Other real-time user variables include simple switches, set by placing the cursor over a label and depressing the appropriate button on the mouse, and continuously variable parameters set by direct typing, depressing a button to invoke the "last-typed" value or the "default" value, or by *dragging*, whereby the cursor is placed over a parameter field and by moving the mouse up or down vertically the current value is shifted appropriately.

The most interesting feature of Conduct is that the parameters that describe musical attributes such as pitch, duration, and timbre selection may be arbitrarily grouped together. Each motion detected from one of the many input channels can modify a

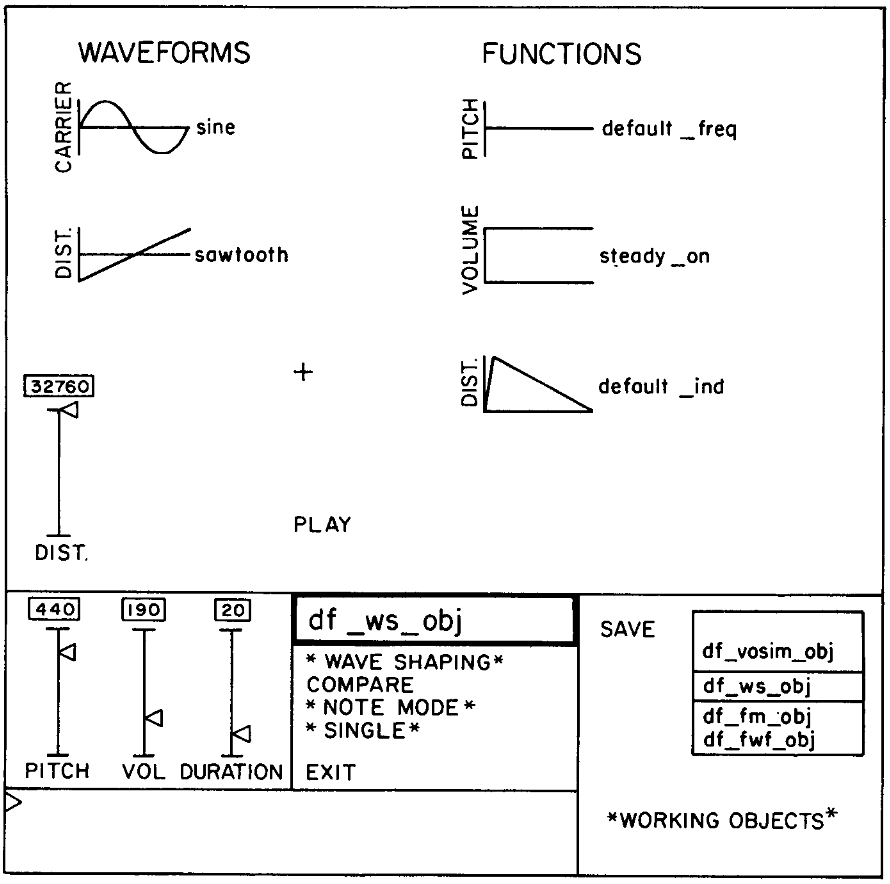

Figure 3. The SSSP interactive display. (Reproduced from Buxton [1981, p. 54].)

set of parameters. According to Buxton, "thus, any transducer can control many parameters, all having different instantaneous values, without any concern for context" [Buxton 1981, p. 178]. These features provide the performer a great deal of freedom and substantially reduce the effort and time needed to affect a direct, real-time response from the synthesis/playback system.

2.2.3 Contributions of the SSSP to User-Interface Design

The achievements of the SSSP team may be assessed by the four basic criteria established in the introduction to this article:

(1) Rather than assume a priori knowledge about music interfaces and music creativity in general, the authors designed a system that could be easily adapted according to observed user preferences. The "quick and dirty" approach [Buxton 1978a, p. 5] reduces the commitment to work that has been completed, thus encouraging the designers to improve the system in response to the user community.

(2) Unique, special-purpose interface mechanisms were designed and implemented. The Conduct system offers a set of interface techniques which have no direct counterpart outside of music. Of particular interest are Buxton's experiments with nonkeyboard and "one-handed" input devices. These are useful for multiple-input-device live performance settings and a position-sensitive surface, which in one mode serves as a "drum" and in another as

set of "potentiometers" for controlling performance parameter levels.

(3) The system is presented to the user in the form of a "software onion," permitting the composer access to as many of the details of the processes as deemed suitable. Wherever possible, default values are provided, permitting the user to defer decisions. The wide range of interface methods also contributes to accommodating individual work habits.

(4) The use of iconic display menus, familiar music terminology, highly streamlined command sequences, and *gestural* input mechanisms (tablet, mouse, sliders, etc.) greatly reduces the barriers between music and computers.

2.3 Other Real-Time Control Devices

A comprehensive approach to computer music production is under development at Lucasfilms, Inc. The objectives of the digital audio group (under the direction of James A. Moorer) encompass all aspects of musical activity: composition, synthesis, analysis, signal processing, sound recording, editing, processing and mixing, and score preparation. The first step was the design and construction of the Lucasfilm Audio Signal Processor, [Moorer 1982a]. A general-purpose interface was required to serve musicians and recording engineers alike. Two articles have been published that report on the human engineering problems, "Remembering Performance Gestures" [Abbott 1982] and "The Lucasfilm Real-Time Console for Recording Studios and Performance of Computer Music" [Snell 1982].

Abbott recognizes that, in general, studio engineers have "... evolved ways of thinking about memorizing and editing gestures" [Abbott 1982, p. 1], and may be reluctant to formulate new work habits, whereas computer music composers are more often interested in exploring new means for musical expression. The basic approach has been to identify *parameter definition problems*, defined as the kinds of processing between the user input and the synthesis/processing device, and to divide these into a *gesture objectification problem* and a *parameter-mapping problem*. As a general strategy, gesture objectification—the acquisition and utilization of real-time gestures—has been modeled as a set of time-varying functions, much like the Groove system described above. An important provision of the Lucasfilm system is that all actions can be "remembered" (i.e., stored in memory) for recall at a later time. Although other systems have included this feature, it has been central to the design strategies of the digital sound project at Lucasfilm.

The operator of the digital recording system at Lucasfilm is provided with a high-resolution bit-mapped screen that can be used for displaying attributes of the acoustical signal (time-domain and spectral-domain representations), musical scores, and various aspects of the mixing procedure. An example of the latter is the display of icons representing prerecorded events stored on disk, which can be visually time-synchronized to other events and to the film frame count.

A device demonstrated at the 1983 International Computer Music Conference at the Eastman School of Music in Rochester, New York, offered a different approach to music control devices. Steve Hafflich displayed a prototype system [Hafflich and Burns 1983], in which the motions of an orchestral conductor were detected from high-frequency audio signals emitted by a specially built baton. The sonar signals were tracked and displayed on a graphics terminal. Thus the motion of the conductor's baton through space was converted into time-varying vectors, which could then be used to control various aspects of the synthesis process in real time.

3. SYSTEMS SURVEY

3.1 Composition and Synthesis Languages

Numerous software packages have been written for applications in music composition, music analysis, sound synthesis, and sound manipulation. In most cases musical or acoustical information is specified as alpha-numeric data. The increased availability of medium- and high-resolution display terminals has contributed to the development of all aspects of music-related interface design, especially score representations. New products for the IBM PC and Apple Macintosh computers offer a wide variety of applications software for music notation display and editing, real-time control of music synthesizers, and the manipulation of digitally recorded sound.

3.1.1 Sound Synthesis Languages

Sound synthesis languages have been surveyed elsewhere in this publication and do not properly belong to a discussion of music interfaces (see Loy and Abbott [1985]). The capabilities of synthesis software (or hardware), however, affect the complexity and run-time requirements of the music system, as well as the complexity of the programs or devices that supply the necessary parametric data. A general-purpose, programmable synthesis language, such as Music10 [Tovar and Smith 1977] or MUSBOX [Loy 1981], offers much greater flexibility than "hard-wired" systems with a limited number of active parameters, such as the SSSP synthesizer. Whether implemented in hardware or software though, the purpose of the user interface is to provide time-ordered lists of synthesis descriptors, time-varying functions, and other performance parameters used to control the synthesis process.

3.1.2 Music Data Encoding and Manipulation

A list and brief description of the principle music data-encoding and data-manipulating languages follows. The mode of communication in each of these schemes is alphanumeric text, but their input specifications and operation syntax are designed for markedly different user strategies. More extensive discussion of music languages and preprocessors can be found in Pennycook [1983a], Hiller and Isaacson [1970], and Roads [1985].

DARMS. DARMS [Brinkman 1983; Erickson 1975] is a music-encoding scheme used primarily by music theorists to enter data for statistical, thematic, and structural analyses. Musical symbols such as pitches, rhythms, articulations, and measures are assigned codes and/or values.

FORMES. Another approach to music language design is represented by FORMES [Cointe and Rodet 1983], which is a music-programming language written in VLISP [Chailloux 1978]. This *object-oriented* language adds a time dimension component called a *dynamic-calculation tree* to program and data constructs borrowed from other object-oriented languages such as Smalltalk [Kay and Goldberg 1976]. A discussion of Smalltalk in a music context appears in Lieberman [1982].

MPL. "Musical Program Library is a comprehensive system for processing musical information" [Nelson 1980]. MPL is based on APL [Iverson 1962] and consists of a *work space* and a set of functions for organizing and transforming musical data in single-dimension *vectors* and in 4 × *n* *matrices*, where the four rows represent a set of parameters used to control a synthesis process. Musical graphics can be displayed on a Tektronix 4013 or printed on a Calcomp 563 plotter and a Diablo Hyterm II printer/plotter.

PlaComp. The PlaComp system is a set of integrated languages and command structures for composing, editing encoded scores, describing and storing synthesis data, and synthesizing music [Murray and Beauchamp 1978; Peters 1975]. PlaComp utilizes the PLATO resources developed at the University of Illinois to facilitate user input and to display music and synthesis data.

Pla. Pla [Schottstaedt 1983] is a music programming language based primarily on SAIL [Reiser 1976] and LISP [Weinreb and Moon 1981] constructs. The principle contribution of Pla to user-interface designs is the incorporation of structured programming constructs and the LISP construct Flavors [Wood 1982]. Further discussion of Pla can be found in Pennycook [1983a], Roads [1985], and Loy and Abbott [1985].

POD. The POD system was developed by Truax [Truax 1977; Truax and Barenholtz 1977] at the Institute of Sonology, Utrecht, on a PDP-15. The current version forms the basis of a PDP11/DMX-1000 [Walraff 1979], real-time composition and synthesis system called PODX [Truax

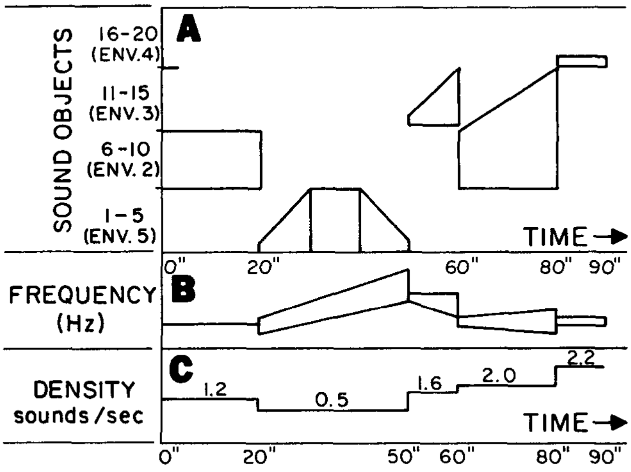

Figure 4. POD mask. (Reproduced from Truax [1977, p. 30].)

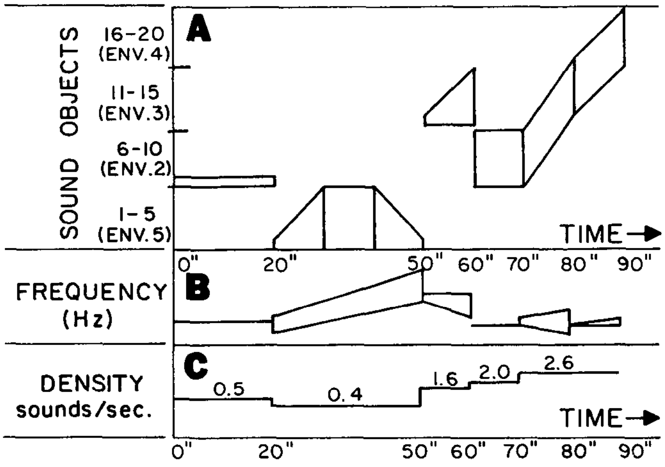

Figure 5. POD mask. (Reproduced from Truax [1977, p. 30].)

1985]. POD is based on an aesthetic and pedagogical preference for compositional strategies that focus on the specification of time-varying functions, which control stochastic distributions of several musical parameters (pitch, timbre, vertical density, amplitude, spatial distribution). This strategy provides the composer with strong, active parameters that produce immediate auditory results, as opposed to weak, general systems that often frustrate the composer with system complexities and poor response turnaround times. Figures 4 and 5 illustrate two different sets of probability masks controlling sound density, frequency, and object (sound synthesis routines) selection.

Score. Score [Smith 1972] is a FORTRAN programme that reads a file of codes and values derived from musical symbols and produces a time-ordered list of parametric data for a variety of synthesis languages. Although the input formats are rigid and somewhat restrictive, Score has been successfully used for many important computer music compositions. There are several similar music-encoding and -manipulation languages modeled on Score, such as Scot [Good 1978] and Scorell [Brinkman 1981]. Score and its descendants use terminology that is familiar to musicians, and automate repetitive tasks. This simplifies the work of translating musical information into data that can use a graphics device or control a synthesis device. Sample input and output files are shown in Figure 6.

Further discussion of music languages

```
   <input to Score>

TOOT 0 1 8;............. instrument 1 plays 8 notes from time 0.

P2 RHY/8/16//8//4//2;... 8th, 2 16ths, 2 8ths, 2 quarters, 1 half note.

P3 NOTES/C4/D/E
   /F/G/A/B/C5; ....... C-major scale starting on middle C

P4 500;................. amplitude (2048 maximum conversion
                         word size)

P5 F3;.................. F1 and F3 are names of function tables.
P6 F1;

END;


   <output - first note only>

PLAY;
TOOT 0.00 .125 C F3 F1; . parameter fields designate the instrument
                          name, start time in seconds, note duration
                          in seconds, pitch, functions.
```

Figure 6. Score input and output files. (Reproduced from Pennycook [1983a, p. 9].)

and their properties can be found in Hiller [1972], Krasner [1980], Pennycook [1983a], Roads [1985], and Loy and Abbott [1984].

3.2 Graphics Score Editing

Reducing the cost and increasing the resolution of bit-mapped display systems is a major factor in the further development of interactive score-editing tools. Pointing systems (mice, track balls, etc.) offer increases in user throughput similar to those achieved in menu-driven text editing systems.

MS. MS was developed by Smith in the 1970s [Smith 1973]. Musical notation is graphically displayed on a vector screen. Commands and data are entered directly from the keyboard or indirectly from files. The output can be produced several times larger than life size, which produces sharp, high-quality print when photoreduced.

Scriva. Scriva [Buxton et al. 1979] is the score-preparation software incorporated in the SSSP system (see Buxton et al. [1982]). The user uses a digitizing tablet and mouse to manipulate symbols and text displayed on an interactive vector terminal. Figure 7 shows an example of the notational flexibility of the screen formats.

Mockingbird. Mockingbird [Maxwell and Ornstein 1983] is described as a *powerful scribe* rather than a creative tool (see figure 8; see also Roads [1981]). Music is performed on and played back through a Yamaha CP-30 synthesizer, and displayed on the high-resolution Dorado [Lampson et al. 1981] terminal. High resolution, large memory space, very fast processing speeds, and mouse-activated editing capabilities similar to those developed for text at Xerox Palo Alto Research Center provide the musician with a powerful music notation editor. The system is somewhat restrictive in that only piano type (one treble clef and one bass clef) scores are supported. The elegance, efficiency, and great speed of Mockingbird, however, make it a worthy model for further developments.

Gregory's Scribe. A well-designed, inexpensive system for encoding and printing traditional music notation is described by Crawford and Zeef [1983]. The user interacts with an Apple computer through a low-cost digitizing tablet. Output is printed on a dot-matrix printer at a resolution high

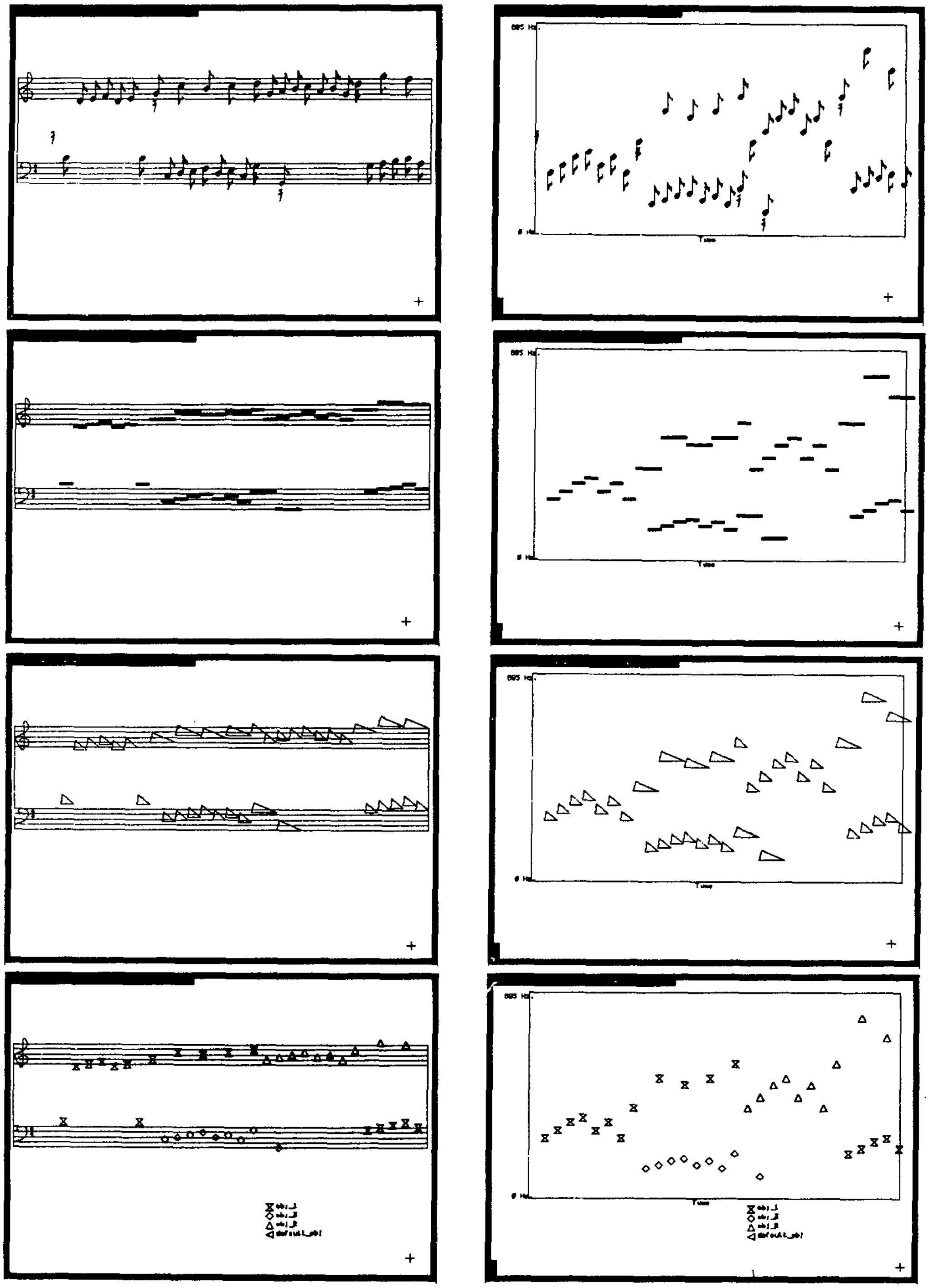
Figure 7. Example of a Scriva screen format. (Reproduced from Buxton et al. [1979, p. 23].)

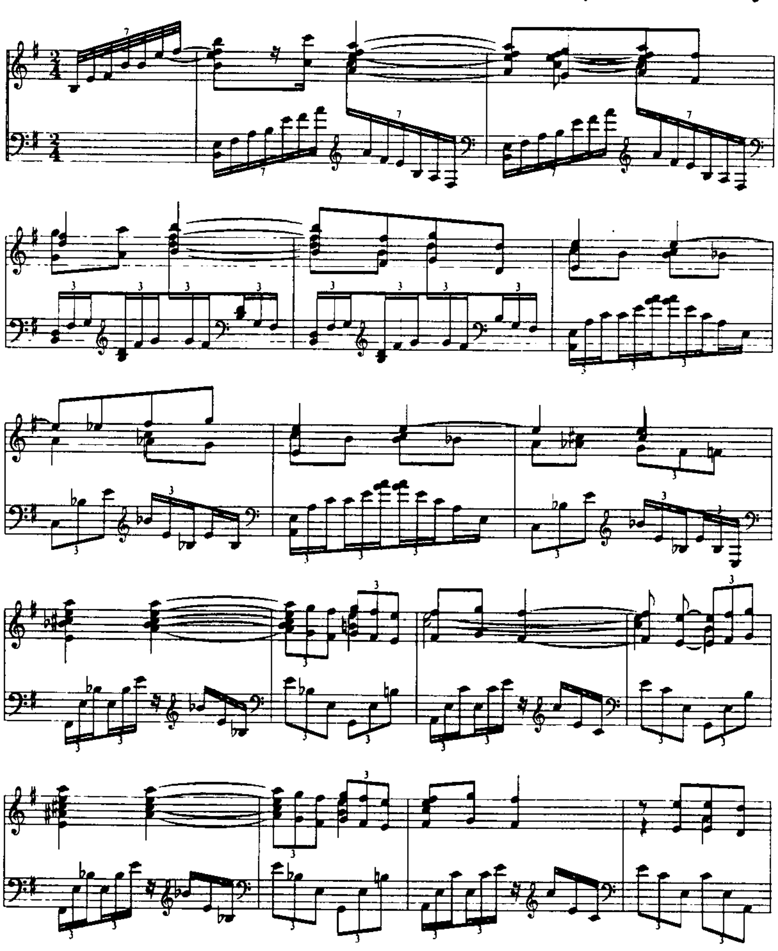

Figure 8. Mockingbird notation. (Reproduced from Roads [1981, p. 58].)

enough for performance applications. This system offers several traditional music notation formats and facilities for transcribing them to modern music notation.

Musprint. Musprint [Hamel 1984] is a new software package written by Keith Hamel for the Apple Macintosh. The author constructs musical symbols using the MacPaint program, and then (like the SSSP tools) uses a mouse to place them on the bit-mapped screen. Although the resolution of the Macintosh is much less than a terminal such as the Dorado, the vertical resolution of the output is a function of the printer. Hence dot-matrix output that shows discontinuities in oblique and curved lines appears smooth when printed on a laser-jet type device. A reproduction from one of the Musprint manual pages appears in Figure 9.

Personal Composer. Personal Composer [Miller 1984] is a software package for the IBM PC. It requires a Hercules™ monochrome graphics card, which increases the screen resolution of an IBM PC to 820 × 640 pixels, to obtain sufficient display quality for music notation. Of greater interest here, though, is that this product includes a direct interface to MIDI (see Section 3.3) data. Music performed on a piano-type keyboard that can generate MIDI data can be directly displayed as musical notation—a highly desirable feature. Conversely, any of the 32 channels of music notation can be

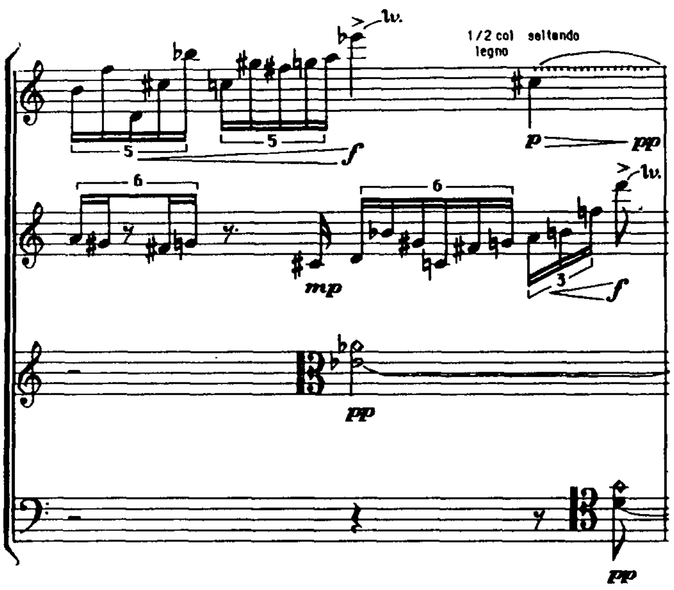

Figure 9. Musprint notation (dot-matrix output). (Reproduced from Hamel [1984, p. 5-4].)

converted to MIDI data and heard through a synthesizer equipped to receive MIDI.

3.3 Performance Instruments

Digital performance devices can be divided into categories based on their degrees of complexity and user access to the internal operations of the hardware.

Digital Music Systems. Digital music systems, such as New England Digital Synclavier II [Appleton, 1977; Lehrman 1984], and Fairlight Computer Music Instrument, Crumar General Development System [Kaplan 1981], offer the performer a wide range of user-accessible capabilities. These include digital recording, graphics score editing and signal manipulation tools, fully programmable synthesis, and composition software. Although they are rather expensive ($10,000–$50,000), these systems are being used in computer music research studios and commercial recording studios. The capacity to record, store, edit, process, and play back musical tones has made machines like the Fairlight and Synclavier attractive as cost-effective commercial tools.

Portable Digital Keyboard Devices. Commercial synthesizers range from simple inexpensive portables, from major manufacturers such as Yamaha, Casio, and Tandy Corporation, to sophisticated programmable models for professional studio use, such as the Crumar Synergy [Kaplan 1981] and the new Yamaha digital keyboard pianos and synthesizers. Most of the professional models permit the player to store preset signal treatment and record and playback performance actions, and augment the acoustical resources and control mechanism from external sources. A recent market entry, the Kurzweil Piano,[3] achieves a nearly perfect reproduction of an acoustic piano through a patented method. (Information on digital synthesizers is presented regularly in *Keyboard Magazine* and *The Computer Music Journal.* A broad survey of commercial synthesis equipment appeared in the February 1981 issue of *Studio Sound.*)

Microcomputer Peripherals. Several manufacturers offer music systems for personal computers. These range from software options, which use the internal speaker of the machine, to digital synthesis boards that use piano keyboards and graphics packages. Although very few design innovations for the user interface have accompanied these products, they have

[3] Kurzweil Music Systems, 411 Waverly Oaks Rd., Waltham, Mass.

integrated keyboard performance, programmable sound-synthesis specification, composition aids, multitrack recording and playback of performance gestures, and music instruction, in streamlined, low-cost packages, making computer music available to the microprocessor consumer. The most popular of these systems are *MusicSystem*, manufactured by Mountain Computer Corporation, *Alpha Syntauri*, from Syntauri Corporation, and *Soundchaser*, by Passport Designs Incorporated. An interesting design for an interactive network of microprocessors to produce music appeared in Bischoff et al. [1978], but this is not offered commercially.

Nonkeyboard Instruments. Products such as the Drumulator, Lynn Drum, and Drummatics offer a range of drum performance capabilities including programmable sequencing (from reiterative patterns to entire songs), digital or analog drum synthesis, and programmable playback of digitized drums (see Anderton [1983]). The user interfaces for these machines vary from switch and rotary knob controls to touch-sensitive surfaces played with the hands or drum sticks. Like the keyboard devices, performed gestures can be captured, stored, and replayed under program control in some machines. Active parameters include tempo, amplitude of each drum or cymbal sound in the pattern, accent selection, rhythmic characteristics, and, in the more expensive units, pitch and signal treatment controls.

Custom-Built Devices. Many noncommercial performance devices and peripherals have been constructed to fulfill unique compositional needs. Buxton's SSSP Conduct system described above represents one strategy for controlling musical actions in a live-performance setting. Another interesting system is *Drapes/Events* [Scheidt 1983], in which a set of active musical parameters such as timbral characteristics, note density, rhythmic rate, and volume are displayed as horizontal bars on an Apple II terminal. The user can manipulate these graphs/parameters with a few keystrokes, which result in instantaneous modifications to the sounds produced by a custom-built digital synthesizer. Numerous other devices have been constructed and demonstrated [Bartlett 1979; Mathews and Abbott 1980; Buxton, personal communication, 1984]. Another interesting development is a set of electronic, solid-body, stringed instruments developed by R. Armin in Toronto [Emerson 1984]. Although the output is analog, these fine instruments offer unprecedented precision and audio signal quality in the area of violin, viola, and cello simulation.

MIDI. The Musical Instrument Digital Interface is described by Junglieb [1983] and Wright [1983]. A more probing analysis of the problems and promises of the MIDI system appeared in Milano [1984]. MIDI is a hardware and software interface system for connecting a microprocessor to several music synthesis instruments over a daisy-chained 31.25-kbaud serial link. At the lowest level, the music interface is a protocol specification for distributing musical data. At the user level, it is a set of software packages for describing sequences of musical events. Up to 16 channels of control data can be distributed at a fast enough rate to avoid timing discrepancies among the various devices. Many new software products for Commodore, Apple, and IBM computers are currently available from major electroacoustic instrument manufacturers such as Roland, Yamaha, Sequential Circuits, Korg, and Oberheim. The MIDI interface has permitted the decoupling of the computer from the performance/synthesis instrument, thus opening up the market for more specialized products. Ultimately, the presence of a mechanism that permits manufacturers to concentrate on the technology of performance devices, free from the concerns of the computer interface, will yield better products.

3.4 Digital Audio Processing Tools

Interfaces developed for music and speech signal processing serve a wide range of purposes. In most cases, interactive graphics display of signal representations form the basis of the interface. From the software-based "all-digital" sound studio developed at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford [Moorer 1977], to real-time systems such

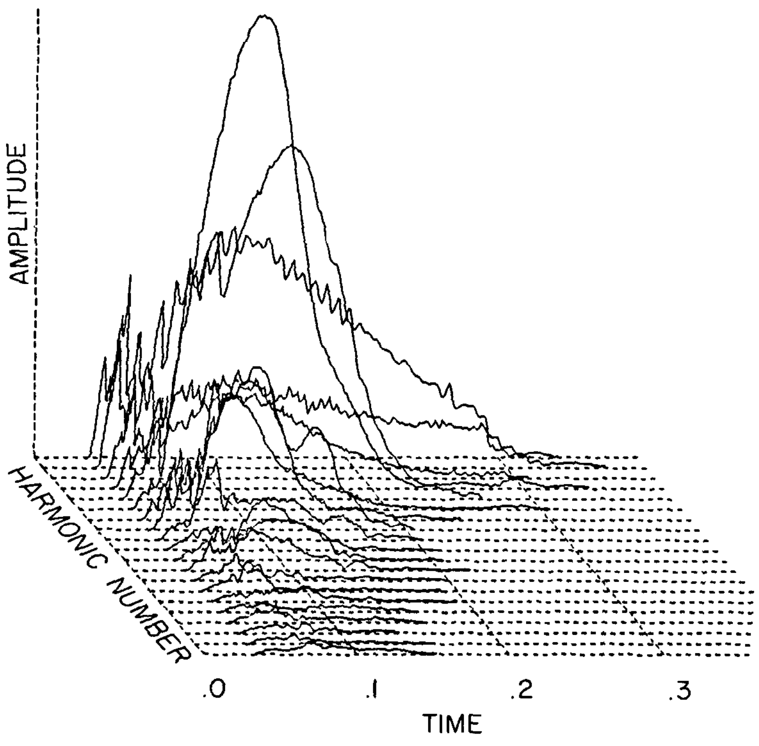

Figure 10. Perspective plot of amplitude X harmonic number X time for a violin tone. The fundamental harmonic is plotted in the background with higher frequency harmonics in the foreground. (Reproduced from Grey [1975, p. 122].)

as the Lucasfilm audio system [Moorer 1982a, 1982b] the primary interface goals have been to provide error-free digital signal editing, digital signal processing, splicing, and mixing tools.

Other approaches include the New York Institute of Technology *Digital Editor* [Kowalski and Glassner 1982], a system designed in conjunction with the British Broadcasting Corporation [McNally 1979], interactive tools designed for the Institut de Recherche et Coordination Acoustique-Musique, Paris [Abbott 1978], *Gcomp* Banger and Pennycook [1983], an interactive graphics package for specifying and editing time-varying sound, file processing, and mixing control functions, and various commercial packages described above that include digital recording a playback.

Another class of interfaces has emerged from research in acoustics, psychoacoustics, and speech recognition. These tools provide graphics display of time-domain and frequency-domain representation (especially those derived from the Fourier transform and phase vocoder analyses [Moorer 1977; Piszczalski 1979]). An example of a violin tone analyzed by the phase vocoder is shown in Figure 10. Several significant advances in our understanding of psychoacoustics are in part attributable to the development of interactive graphics tools for manipulating signal representations [Chafe et al. 1982; Grey 1975; Strawn 1980].

The speech recognition research team at the Massachusetts Institute of Technology has been developing powerful graphics aids on a LISP machine [Roads 1983; Weinreb and Moon 1981], which, when coupled with high-resolution hard copy printers, greatly enhance productivity. Noninteractive real-time display capabilities have been added to digital instrumentation and signal analysis products, such as those offered by leading audio instrumentation manufacturers like Bruel and Kjaar and Texas Instruments.

A trend toward small, powerful UNIX-based computers has provided the means to produce specialized audio workstations with capabilities previously limited to large computers. One such system, the *ORFIAS 32/320* [Pennycook et al. 1985], utilizes a Texas Instruments TMS32020 digital-signal-processing microprocessor within a commercially available 32-bit computer. The signal-processing power of the TMS32020 is enhanced by other circuitry that provides interfaces for disks and audio conversion systems as well as numerous oscillators. In addition to an extended UNIX command structure, the user interface offers graphics aids for signal manipulation.

There is a host of commercially available digital sound-processing products being used in both the recording studio and live performances. These include digital delay units, phasors, flangers, reverberation systems, harmonizers (which generate one or more pitches in parallel at a specifiable interval to the input signal), and units that combine one or more of these effects. Manufacturers' attempt to provide user-friendly control panels is of particular interest. For example, the *Ursa Major Space Station*, manufactured by Ursa Major, Belmont, Massachusetts, offers both a variety of reverberation and delay effects, and custom-built circuit boards that simulate the reverberation characteristics of specific rooms. The front panel reduces the complex description of reverberation specifications to a set of rotary knobs which control the delay taps and switches for selecting programmed settings.

3.5 Computer-Aided Instruction in Music Systems

Several entries in the computer-aided instruction field have used interface techniques developed for text editing and graphics display of musical notation. Menu-driven systems substantially reduce the learning curve for the user, an extremely important consideration in the design of interactive instructional software. A detailed study, The Role of Graphics in Computer Aided Instruction in Music [Pennycook 1983b], offers a detailed discussion of the most widely distributed packages.

A new product built in Canada, the *Excercette* [Sallis 1983], offers instantaneous pitch detection for training in *solfège* and voice intonation. A pitch is displayed in Common Musical Notation on a medium-resolution screen and played by the synthesis mechanism. The student responds to a request for a new pitch by singing into a microphone. The response is promptly displayed with an indication of accuracy (correct, high, low). The user can select four degrees of error resolution ranging from +/− one-quarter tone to "just detectable." The capacity of the device is currently being expanded to include pitch sequences (melodies).

Musicland [Lamb 1982] is a music instruction system available for the Alpha Syntauri system. (A description of the University of Canterbury computer music system in which Lamb first formulated certain aspects of his methodologies appears in Frykberg and Bates [1978].) The Musicland system is unique in that musical constructs are assembled by the manipulation of colored boxes with a mouse input device. The boxes contain musical fragments that can be presented to the student from a library or constructed with free-hand graphics drawing routines. The interface allows a young person to experiment with compositional constructs before encountering the complex vocabulary and syntax of music.

4. CONCLUSIONS

In this survey a number of fundamental criteria in the design of music interfaces have been established. The success and effectiveness of each of the implemented systems described here are the result of expertise and insight in both music and computer engineering. Many of the interface designs developed in other fields, especially document preparation, have contributed to music interface design by defining the man–machine communications issues.

Current research trends suggest that the role of the "artist in the laboratory" [Buxton 1984] has been grossly underestimated, and that the extreme demands placed on technology by musical requirements should serve as a suitable model for the man-machine interface.

The design requirements of musical information and performance, however, have required unique solutions. Although achievements in reliable real-time music hardware implementations have had the most pronounced commercial impact, efforts to solve all aspects of music interface specification have exposed some general interface problems, and pointed the way toward some substantive solutions for all user interface designers.

ACKNOWLEDGMENTS

The author wishes to thank Curtis Abbott, Guest Editor of this issue of *Computing Surveys*, for the opportunity to serve as a contributor and for his advice and encouragement, and William Buxton for his valuable comments, information, and access to his personal library.

REFERENCES

ABBOTT, C. 1978. A software approach to interactive processing of musical sounds. *Comput. Music J. 2*, 2, 19–23.

ABBOTT, C. 1982. *Remembering Performance Gestures.* Lucasfilm Ltd., San Rafael, Calif.

ANDERTON, C. 1983. Digital drums: An overview. *Polyphony 8*, 5, 22–26.

APPLETON, J. 1977. Problems of designing a composer's language for digital synthesis. Audio Eng. Soc. Preprint No. 1230, Audio Engineering Soc., New York, pp. 1–5.

BAECKER, R. 1980. Human–computer interactive systems: A state-of-the-art review. In *Processing of Visible Language II*, P. Kolers, E. Wrolftrad, and H. Bouma, Eds. Plenum, New York, pp. 423–444.

BANGER, C., AND PENNYCOOK, B. 1983. Gcomp: Graphics control of mixing and processing. *Comput. Music J. 7*, 4, 33–39.

BARTLETT, M. 1979. A microcomputer controlled synthesis system for live performance. *Comput. Music J. 3*, 1, 25–37.

BISCHOFF, J., GOLD, R., AND HORTON, J. 1978. Music for an interactive network of microprocessors. *Comput. Music J. 2*, 3, 24–29.

BRINKMAN, A. L. 1982. Data structures for a Music-11 preprocessor. In *Proceedings of the 1981 International Computer Music Conference* (Denton, Tex.). Computer Music Association, San Francisco, pp. 178–196.

BRINKMAN, A. 1983. A design for a single-pass scanner for the DARMS coding language. In *Proceedings of the 1983 International Computer Music Conference* (Venice, Italy). Computer Music Association, San Francisco, pp. 7–30.

BUXTON, W. 1978. Design issues in the foundation of a computer-based tool for music composition. Tech. Rep. CSRG-97, Computer Systems Research Group, Univ. of Toronto, Toronto, Ontario, Canada.

BUXTON, W. 1981. Music software user's manual. Tech. Rep. CSRG 22, Computer Systems Research Group, Univ. of Toronto, Toronto, Ontario, Canada.

BUXTON, W. 1983. Lexical and pragmatic considerations of input structures. *Comput. Graphics 17*, 1, 31–37.

BUXTON, W. 1984. *The role of the artist in the laboratory.* Unpublished manuscript, Computer Systems Research Institute, Univ. of Toronto, Toronto, Ontario, Canada.

BUXTON, W., REEVES, W., BAECKER, R., AND MEZCI, L. 1978a. The use of hierarchy and instance in a data structure for computer music. *Comput. Music J. 2*, 4, 10–20.

BUXTON, W., FOGELS, E. A., FEDORKOW, G., SASAKI, L., AND SMITH, K. C. 1978b. An introduction to the SSSP digital synthesizer. *Comput. Music J. 2*, 4, 28–38.

BUXTON, W., SNIDERMAN, R., REEVES, W., PATEL, S., AND BAECKER, R. 1979. The evolution of the SSSP score editing tools. *Comput. Music J. 3*, 4, 14–25.

BUXTON, W., REEVES, W., FEDORKOW, G., SMITH, K. C., AND BAECKER, R. 1980. A microcomputer-based conducting system. *Comput. Music J. 4*, 1, 8–21.

BUXTON, W., PATEL, S., REEVES, W., AND BAECKER, R. 1982. Scope in interactive score editors. *Comput. Music J. 5*, 3, 50–56.

BYRD, D. 1977. An integrated computer music software system. *Comput. Music J. 1*, 2, 55–60.

CHADABE, J. 1977. The nature of the landscape within which computer music systems are constructed. *Comput. Music J. 1*, 3, 5–11.

CHADABE, J. 1983. Interactive composing, Albany, New York. In *Proceedings of the 1983 International Computer Music Conference* (Rochester, N.Y.) Computer Music Association, San Francisco.

CHAFE, C., MONT-REYNAUD, B., AND RUSH, L. 1982. Toward an intelligent editor of digital audio: Recognition of musical constructs. *Comput. Music J. 6*, 1, 30–41.

CHAILLOUX, J. 1978. VLISP: 10.3 Manuel de référence. RT 16-78, Université de Paris, 8-Vincennes.

COINTE, P., AND RODET, X. 1983. FORMES: A new object-language for managing of hierarchy of events. Institut de Recherche et de Coordination Acoustique-Musique, Paris.

CRAWFORD, D., AND ZEEF, J. 1983. Gregory's Scribe: Inexpensive graphics for Pre-1600 music notation. *Comput. Music J. 7*, 1, 21–24.

EMBLEY, D. W., AND NAGEY, G. 1981. Behavioral aspects of text editors. *ACM Comput. Surv. 13*, 1 (Mar.), 33–70.

EMERSON, M. 1984. The shock of the new. *The Strad 94*, 1126, 698–701.

ERICKSON, R. F. 1975. The DARMS project: A status report. *Comput. Humanities 9*, 6 291–298.

FEDORKOW, G., BUXTON, W., AND SMITH, K. C. 1978. A computer-controlled sound distribution system for the performance of electroacoustic music. *Comput. Music J. 2*, 3, 33–42.

FRYKBERG, S. K., AND BATES, R. H. 1978. The computer and the composer: A description of a developing, working system. Tech. Rep., Dept. of Electrical Engineering, Univ. of Canterbury, Christchurch, New Zealand.

FURATA, R., SCOFIELD, J., AND SHAW, A. 1982. Document formatting systems: Surveys concepts, and issues. *ACM Comput. Surv. 14*, 3 (Sept.), 417–472.

GABURO, J. 1973. An analog/hybrid instrument for electronic music synthesis. Ph.D. dissertation, Univ. of Toronto, Toronto, Ontario, Canada.

GOOD, M. 1978. SCOT: A score translator for Music-11. Undergraduate thesis. Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass.

GREY, J. M. 1975. An exploration of musical timbre. Stan-M-2, Ph.D. dissertation, Dept. of Music, Stanford Univ., Stanford Calif.

GROGONO, P. 1973. Software for an electronic music studio. *Softw. Pract. Exper. 3*, 369–383.

HAFFLICH, S., AND BURNS, M. 1983. Following a conductor: The engineering of an input device. M.I.T. Experimental Music Studio. Presented to the International Computer Music Conference, Rochester, N.Y.

HAMEL, K. 1984. *Musprint Manual.* Triangle Resources, Cambridge, Mass.

HANES, S. 1980. The musician-machine interface in digital sound synthesis. *Comput. Music J. 4*, 4, 23–44.

HILLER, L. 1972. Computer programs used to produce the composition HPSCHD. Tech. Rep. 4, Dept. of Music, State Univ. of New York, Buffalo, N. Y.

HILLER, L., AND ISAACSON, L. 1970. Music composed with computers: A historical survey. In *The Computer and Music,* H. Lincoln, Ed. Cornell Univ. Press, Ithaca, N. Y., pp. 42–96.

IVERSON, K. E. 1962. *A Programming Language.* Wiley, London.

JUNGLIEB, S. 1983. MIDI hardware fundamentals. *Polyphony 8*, 4, 34–38.

KAPLAN, S. J. 1981. Developing a commercial digital synthesizer. *Comput. Music J. 5*, 3, 62–73.

KAY, A., AND GOLDBERG, A., EDS. 1976. Smalltalk-72 instruction manual, SSL 76-6, Xerox Palo Alto Research Center, Palo Alto, Calif.

KOWALSKI, M. J., AND GLASSNER, A. 1982. The N.Y.I.T. digital sound editor. *Comput. Music J. 6*, 1, 66–73.

KRASNER, G. 1980. Machines Tongues VII: "The design of a Smalltalk music system. *Comput. Music J. 4*, 4, 4–14.

LAMB, M. 1982. Musicland—A network of educational games. *Options*, vol. 13: Spring. Office of Educational Development, Univ. of Toronto, Toronto, pp. 12–13.

LAMPSON, B. W., PIER, K. P., ORNSTEIN, S. M., CLARK, D. C., AND MCDANIEL, G. 1980. The Dorado: A high performance personal computer (three papers). Rep. CSL-80-2, Xerox Palo Alto Research Center, Palo Alto, Calif.

LASKE, O. E. 1977. Toward a theory of interfaces for computer music systems. *Comput. Music J. 1*, 4, 53–60.

LASKE, O. E. 1978. Considering human memory in designing user interfaces for computer music. *Comput. Music J. 2*, 4, 39–45.

LEDGARD, H., WHITESIDE, J., SINGER, A., AND SEYMOUR, W. 1980. The natural language of interactive systems. *Commun. ACM 23*, 10 (Oct.), 556–563.

LEHRMAN, P. D. 1984. Inside the Synclavier. *Studio Sound 26*, 2, 68–74.

LIEBERMAN, H. 1982. Machine Tongues IX: Object-oriented programming. *Comput. Music J. 6*, 3, 8–21.

LOY, G. D. 1981. Notes on the implementation of MUSBOX: A compiler for the Systems Concepts digital synthesizer. *Comput. Music J. 5*, 1, 34–50.

LOY, G. D. AND ABBOTT, C. 1985. Programming languages for computer music synthesis, performance, and composition. *Comput. Surv. 17*, 2 (June), 235–265.

MATHEWS, M., AND ABBOTT, C. 1980. The sequential drum. *Comput. Music J. 4*, 4, 45–59.

MATHEWS, M., AND BENNETT, G. 1978. Real-time synthesizer control rapports. IRCAM 5178, Institut de Recherche et Coordination Acoustique-Musique, Paris.

MATHEWS, M., AND MOORE, F. R. 1969. GROOVE, a program for realtime control of a sound synthesizer by a computer. In *Proceedings of the 4th Annual Conference of the American Society of University Composers* (Santa Barbara, Calif., Apr.). ASUC, Music Dept., Columbia Univ., New York, pp. 22–31.

MATHEWS, M. AND MOORE, F. R. 1970. GROOVE—A program to compose, store, and edit functions of time. *Commun. ACM 13*, 12 (Dec.), 715–721.

MATHEWS, M., AND ROSLER, L. 1969. Graphical language for the scores of computer-generated sounds. In *Music by Computers*. H. von Foerster, and J. W. Beauchamp, Eds. Wiley, New York, pp. 84–114.

MAXWELL, J. T., AND ORNSTEIN, S. M. 1983. Mockingbird: A composer's amanuensis. CSL-83-2, Xerox Corporation, Palo Alto, Calif.

MCNALLY, D. 1979. Microprocessor mixing and processing of digital audio signals. *J. Audio Eng. Soc. 27*, 10, 793–803.

MEYROWITZ, N., AND VAN DAM, A. 1982. Interactive editing systems: Parts I and II. *ACM Comput. Surv. 14*, 3 (Sept.), 321–416.

MILANO, D. 1984. Turmoil in MIDI-Land. *Keyboard*, 42–63, 106.

MILLER, J. 1984. *Personal Composer*, American Bullycode (tm), Washington, D.C.

MOORER, J. A. 1977. Signal processing aspects of computer music: A survey. *Proc. IEEE 65*, 8, 1108–1132.

MOORER, J. A. 1982a. The Lucasfilm audio signal processor. *Comput. Music J. 6*, 3, 22–32.

MOORER, J. A. 1982b. *The ASP System: A Tutorial.* Lucasfilm Ltd., San Rafael, Calif.

MORAN, T. P., ED. 1981. Special issue: The Psychology of Human–Computer Interaction. *ACM Comput. Surv. 13*, 1 (Mar.).

MURRAY, D. J., AND BEAUCHAMP, J. 1978. A user's guide to the PLATO/I1980 music synthesis system: Using the PlaComp language. Manual, Dept. of Music, Univ. of Illinois, Urbana.

NELSON, G. 1980. MPL: Musical program library manual. Dept. of Music, Oberlin College, Oberlin, Ohio.

PENNYCOOK, B. W. 1983a. Music languages and preprocessors: A tutorial. In *Proceedings of the 1983 International Computer Music Conference* (Berkeley, Calif.). Computer Music Association, San Francisco, pp. 275–298.

PENNYCOOK, B. 1983b. The role of graphics in computer aided music instruction. Paper presented at 1983 Nicograph Conference, Tokyo, Japan, Oct.

PENNYCOOK, B., KULICK, J., AND DOVE, D. 1985. The Image and Audio Systems audio workstation. In *Proceedings of the 1985 International Computer Music Conference*. Computer Music Association, San Francisco, pp. 145–149.

PISZCZALSKI, M. 1979. Spectral surfaces from performed music. *Comput. Music J. 3*, 1 18–24.

PULFER, J. K. 1970. Computer aid for musical composers. *Bull. Radio Electr. Eng. Div. 20*, 2. National Research Council, Ottawa, Canada, pp. 44–48.

PULFER, J. K. 1971a. The NRC computer music system. National Research Council Rep., Ottawa, Canada.

PULFER, J. K. 1971b. Man–machine interaction in creative applications. *Int. J. Man-Mach. Stud. 3*, 1–11.

REISER, J., ED. 1976. SAIL. Rep. STAN-CS-574, Stanford Artificial Intelligence Laboratory, Stanford Univ., Stanford, Calif.

RITCHIE, D. M., AND THOMPSON, K. 1974. The UNIX time-sharing system. *Commun. ACM 17*, 7 (July), 365–375.

ROADS, C. 1981. A note on printing music by computer. *Comput. Music J. 5*, 3, 57–59.

ROADS, C. 1983. A report on Spire: An interactive audio processing environment. *Comput. Music J. 7*, 2 70–74.

ROADS, C. 1985. Research in music and artificial intelligence. *ACM Comput. Surv. 17*, 2 (June), 163–190.

SACHS, C. 1940. *The History of Musical Instruments.* Norton, New York.

SALLIS, F. 1984. LIM. Unpublished manuscript, Univ. de Laval, Quebec City, Canada.

SANDEWALL, E. 1978. Programming in an interactive environment: The "LISP" experience. *ACM Comput. Surv. 10*, 1 (Mar.), 35–71.

SCHEIDT, D. 1983. The blue card. Unpublished manuscript, Kingston, Canada.

SCHOTTSTAEDT, B. 1983. Pla: A composer's idea of a language. *Comput. Music J. 7*, 1 11–20.

SMITH, L. C. 1972. Score—A musician's approach to computer music. *J. Audio Eng. Soc. 20*, 1, 7–14.

SMITH, L. C. 1973. Editing and printing music by computer. *J. Music Theor. 17*, 2, 292–309.

SMOLIAR, S. W. 1973. A data structure for an interactive music system. *Interface 2*, 2, 127–140.

SNELL, J. 1982. The Lucasfilm real-time console for recording studios and performance of computer music. *Comput. Music J. 6*, 3, 33–45.

STRAWN, J. 1980. Approximation and syntactic analysis of amplitude and frequency functions for digital sound synthesis. *Comput. Music J. 4*, 3, 3–24.

TANNER, P. 1971. Some programs for the computer generation of polyphonic music. Rep. ERB-862, National Research Council, Ottawa, Canada.

TOVAR AND SMITH, L. 1977. Music10 manual. Unpublished user's manual. Center for Computer Research in Music and Acoustics, Stanford Univ., Stanford, Calif.

TRUAX, B. 1977. The POD system on interactive composition programs. *Comput. Music J. 1*, 3, 30–39.

TRUAX, B. 1985. The PODX system: Interactive compositional software for the DMX-1000. *Comput. Music J. 9*, 1, 29–38.

TRUAX, B., AND BARENHOLTZ, J. 1977. Models of interactive computer composition. In *Computing in the Humanities: Proceedings of the Third ICCH*, S. Lusignan and J. North, Eds. Univ. of Waterloo Press, Waterloo, Ontario, Canada, pp. 209–219.

VERCOE, B. 1975. Man–computer interaction in creative applications. Experimental Music Studio, Massachusetts Institute of Technology, Cambridge, Mass.

WALRAFF, D. 1979. The DMX-1000 signal processing computer. *Comput. Music J. 3*, 4, 44–49.

WEINREB, D., AND MOON, D. 1981. Lisp machine manual. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.

WIGGEN, K. 1968. The electronic music studio at Stockholm Development and Construction. *Interface 1*, 127–165.

WOOD, R. J. 1982. Franz Flavors: An implementation of abstract data types in an applicative language. Rep. TR-1174, Maryland Artificial Intelligence Group, Univ. of Maryland, College Park, Md.

WRIGHT, J. 1983. What MIDI means for musicians. *Polyphonic 8*, 4, 8–15.

Received April 1984; final revision accepted August 1985.