An Overview of the Structured Sound Synthesis Project

by

W. Buxton, G. Fedorkow, R. Baecker,
W. Reeves, K. C. Smith,
G. Ciamaga, and L. Mezei

The Structured Sound Synthesis Project (SSSP)
Computer Systems Research Group
University of Toronto
Toronto, Ontario
Canada
M5S 1A4

November 1, 1978

An Overview of the Structured Sound Synthesis Project

by

W. Buxton, G. Fedorkow, R. Baecker,
W. Reeves, K. C. Smith,
G. Ciamaga, and L. Mezei

The Structured Sound Synthesis Project
Computer Systems Research Group
University of Toronto
Toronto, Ontario
Canada
M5S 1A4

Abstract

An introduction to the Structured Sound Synthesis Project of the University of Toronto is presented. The Structured Sound Synthesis Project (SSSP) is an interdisciplinary project whose aim is to conduct research into problems and benefits arising from the use of computers in musical composition. This research can be considered in terms of two main areas: the investigation of new representations of musical data and processes, and the study of man-machine communication as it relates to music.

Integral to the research described is the evolution of a computerized environment which one could term a "composer's assistant." Therefore, the development of an interactive music system -- in combination with observing its use by trained composers -- constitutes one of the main activities of our research. This paper not only gives an overview of the system, but describes some of the objectives in its design and conclusions from our experience with it to date. Key words applicable to the project are: interactive, real-time, music (composition and synthesis), mini-computer, special digital sound synthesizer, and graphics-oriented command language.

1. *Scope and Objectives of Study*

The objective of the Structured Sound Synthesis Project (SSSP) is to conduct research into problems and benefits arising from the use of computers in musical composition. A study in this area is interdisciplinary by nature. The scope of the current project involves three main disciplines: music (composition and theory), computer science, and electrical engineering.

The musical problem with which we are concerned is our current lack of understanding of the process of musical composition. That is, while there exist theories which deal with the artifacts of this process (the study of scores, performance, etc.) and resulting esthetic questions, our ability to explicate the strategies and decision making employed in composition is very limited. Our lack of knowledge in this domain was perhaps best brought out when researchers began (circa 1957) attempting to provide computerized tools to aid the composer in this process. Just as with comparable efforts to develop systems for the automated translation of natural language, such attempts have met with with very limited success. That this is the case can be illustrated by contrasting the relatively small amount of music which has been written using such facilities (Melby, 1976) with the amount of research which has gone into their development (Battier and Arveiller, 1978).

However, just as attempts to develop such facilities have served to point out the problems, they also serve -- in our opinion -- as ideal environments for filling in some of the current gaps in our knowledge. Given a suitable interactive environment, this is seen in two ways. First, a vehicle for collecting information is provided: by observing the interaction of trained composers with the system. Second, such a system serves as a tool which enables the testing of interim results of the research. Such an approach represents the main methodology utilized by the SSSP. As a result, a great deal of our initial efforts have been devoted to the development of a suitable interactive environment to serve as such a research tool.

Given that the success of the project is based largely upon the interaction of both musicians and technologists with the system, it can be seen why the man-machine communication problem constitutes the second main aspect of the research. In the realm of computer science this problem centres on two main areas: the efficient internal representation of data and processes, and the establishment of a suitable task-oriented command language for accessing them. To enable the above, true interaction and immediate turn-around are essential. The system which has resulted makes heavy use of interactive computer graphics. In addition, the initial engineering effort has seen the development of a special purpose digital sound synthesizer, which is one of the most important elements in the system.

In summary, we hope to demonstrate the possibilities of a man-machine environment in which the composer can more fully exploit the creative potential of the medium and create works of musical significance. Success in doing so will not only demonstrate an understanding of the underlying musical problems, but have important implications concerning the use of computers in the humanities.

2. *Design Approach*

The work of the SSSP has been guided by two basic design attitudes (Buxton, 1978). These can be characterized as first, a "behavioural", and second, an "iterative" approach to systems design. By a behavioural approach, we mean an approach whose primary focus is the user's behaviour while undertaking the task for which the tool was designed; that is, his mental imaging of the problem space, and the strategies which he employs in negotiating within this space. By an iterative approach, we simply mean an approach which discards the current version of the system as soon as it has served its purpose, then replacing it with a new version which is based on increased experience and insights.

These two approaches are linked in a non-trivial way. In order to tailor a tool to fit an application, we must know how it is to be used. Hence our interest in behavioural issues. However, we must acknowledge our lack of adequate *a priori* understanding to enable such tailoring. The approach, therefore, is to adopt a strategy to gain, bit-by-bit, the requisite knowledge. This we accomplish following the iterative approach: implement to our current ability, observe, draw conclusions, re-implement (with added insight), observe, etc.

From a practical point of view, the main consideration in such an approach is the cycle time of successive iterations. The ability to obtain rapid feedback on the merits of various ideas is essential to the success of the approach. We would argue that the research environment can be designed so as to make such feedback realistic. In doing so, perhaps the two must important guidelines are the provision of adequate software tools, and the avoidance of special-purpose hardware (wherever possible). Getting software up and running as quickly and inexpensively as possible is dependent on a flexible system including a powerful editor, online debugger, and high-level programming language. (This precludes the use of low-level languages, such as assembly language, as the basis for software development.)

3. *User Software Organization*

In determining the organization of the user software (Buxton, Green & Hume, 1978), the central concern has been to minimize the burden on the user imposed by the mechanics of carrying out various tasks. The composer should simply not have to memorize a large number of commands, the sequence in which they may be called, or the order in which their arguments must be specified. In our approach we have chosen to categorize all commands according to four basic tasks. The intention in so doing is to provide a simple conceptual framework within which the composer can view his activities. The task breakdown is as follows:

1. Definition of the palette of timbres to be available. This we call *object definition*, which is analogous to choosing the instruments which are to comprise the composer's orchestra. The main expansion on the analogy is that the composer also has the *option* to "invent" his own instruments.

2. Definition of the pitch-time structure of a composition, a process which we can call *score definition*. In conventional music, this task would be roughly analogous to composing a piano version of a score.

3. The *orchestration* of the "score". Generally stated, attaching attributes (such as *objects* defined in Step 1) to *scores* defined in Step 2.

4. The *performance* of the material developed thus far, whether an entire (orchestrated or unorchestrated) score, or simply a single note (to audition a particular object, for example) [1].

[1] We include performance as part of the compositional process based on the opinion

In order to undertake these four tasks the novice user need only remember four commands. For each task there exists an "editor" which is in effect a software tool to aid in that tasks undertaking. Reeves, Buxton, Pike, & Baecker (1978), for example, presents an editor for scores (task 3) called *ludwig*. Like most of the SSSP software, *ludwig* makes heavy use of interactive computer graphics. The benefits in so doing are two-fold. First, data can be presented in a clear, musically intuitive manner. Secondly, by making use of selected *menus* of *lightbuttons*, a straightforward method of working is provided. The options currently available to the composer are always clear, as is the protocol for their use. Furthermore, the system is flexible enough to allow the user to jump back-and-forth from one task's editor to another's. This brings to light another important aspect of the design. Since different composers work in different ways, the system is structured such that no order is imposed on the sequence in which the user undertakes the above four tasks. A composer is allowed to perform a score before it has been orchestrated, or even orchestrate it before the *objects* indicated have been defined! The implication is that -- through the judicious use of defaults -- the system is capable of coping with incompletely specified data, thereby freeing the composer from having to worry about that which is not of immediate concern.

As the user becomes more familiar with the system, alternative methods of undertaking these four tasks can be presented, while still keeping within the original conceptual framework. Scores may be defined deterministically (using an editor based on either piano-roll or common music notation), or generated using available compositional programs. Since all of these scores have the same internal representation (Buxton, Reeves, Baecker, & Mezei, 1978), they can be combined, transformed, edited, etc. using any of the available tools. Similarly, the timbres of sounds (or *objects)* can be defined using a variety of techniques. Finally, it was seen as important to provide the composer with a "handle" to his data which permits them to be manipulated at more than the basic note-by-note level prevalent in most systems today. This was accomplished to a certain degree by making use of the notion of *instantiation* and allowing the composer to structure his data in a *hierarchical* manner (Buxton, Reeves, *et al*, 1978).

---

that a piece of music is not completed until it is heard. While some theorists would dispute its need, we would argue that composers of conventional music have always had such aural feed-back -- in the mind's ear -- as enabled by a familiarity with the long tradition of western music; a tradition which does not exist for the composer of contemporary music.

4. *The Computing Environment*

4.1. *Hardware*

4.1.1. *Introduction*

A general overview of the physical environment is presented in Figure 1. Here the main functional components are shown. These can be subdivided into four main categories: the input/output (I/O) transducers, the host computer, the slave processor, and the digital synthesizer.

4.1.2. *I/O Transducers*

4.1.2.1. *General*

In selecting the I/O transducers to be used by the system, it was consciously attempted to avoid special purpose hardware wherever possible. The transducers used have been chosen so as to minimize as much as possible the physical gestures that the user must make in executing a particular task. In this regard, we have attempted to assemble a configuration in which typing could be kept to a minimum. As a result, the transducers are oriented very heavily towards interactive computer graphics. Again, this is in keeping with -- and a result of -- the importance which we place on the flexibility to utilize different representations of data and processes. Graphics are well suited for the task. We shall now discuss each of the key transducers independently.

4.1.2.2. *Graphics Display*

Perhaps one of the most important components in the system is the graphics display. The display used is a fast refresh vector-drawing display produced by the Three Rivers Computer Corporation of Pittsburgh. Use of refresh over the alternative storage or raster-scan (video) techniques was chosen primarily due to the potential for *dynamic* graphics. One can, for example, animate complex processes, have graphic scores scroll across the screen, or selectively erase, move, or edit individual picture components. With the Three Rivers display and display processor, all of this is possible free of flicker (even with the most complex images) in real-time.

4.1.2.3. *Digitizing Tablet*

Equally important to the display of information graphically is to enable the use of graphical techniques for data entry. Various alternatives are available. These include light-pens, "joy-stick" controllers, tracker-balls, touch sensitive panels, digitizing tablets, and mice (tracker balls mounted in a small housing enabling them to be used much like a digitizing tablet). For our purposes, we have chosen to make use of a digitizing tablet. This device facilitates sketching and pointing, and provides for very high resolution when required.

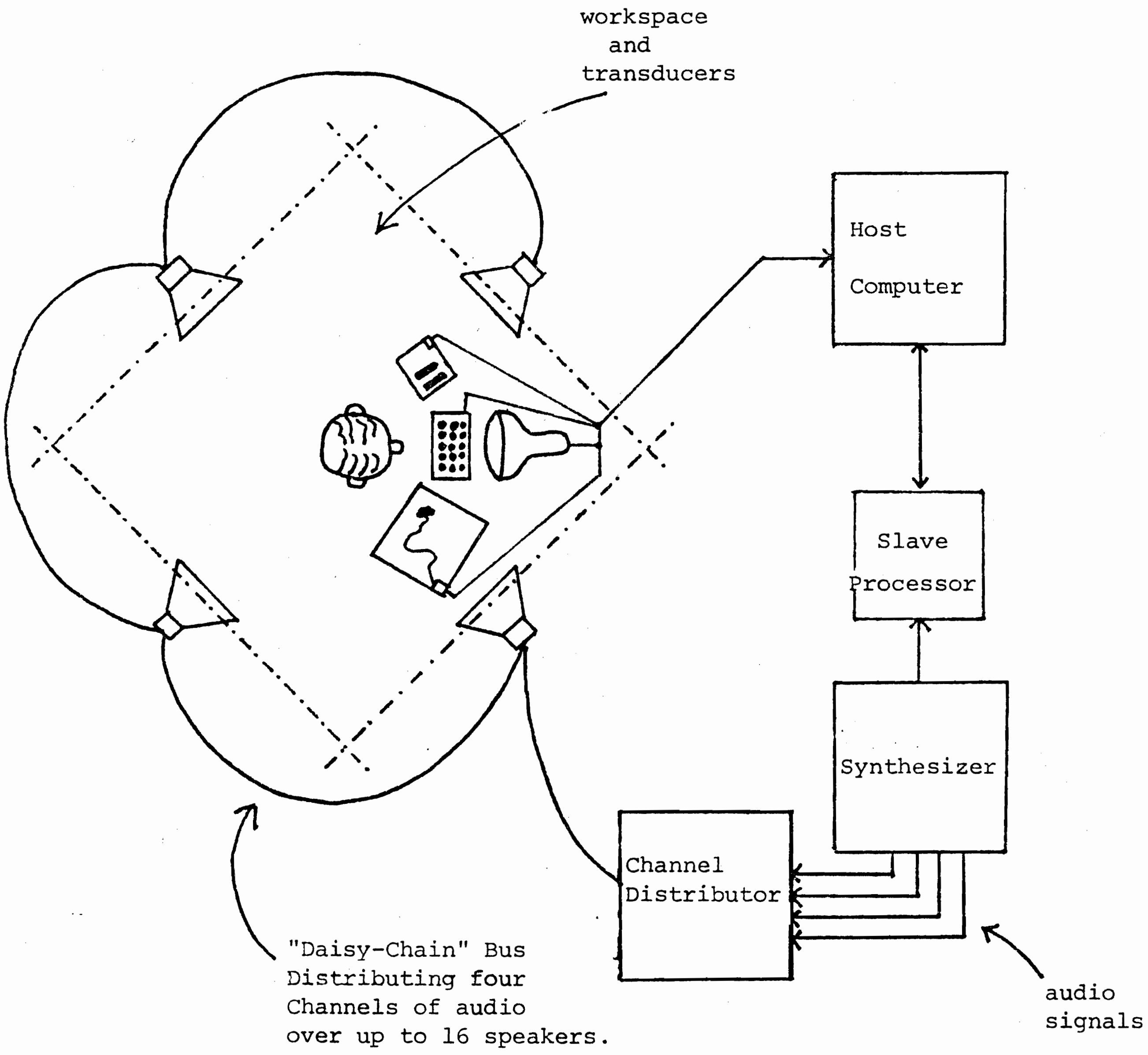
workspace and transducers
Host Computer
Slave Processor
Synthesizer
Channel Distributor
"Daisy-Chain" Bus Distributing four Channels of audio over up to 16 speakers.
audio signals

Figure 1 : Functional block diagram of physical environment

The tablet consists of a flat panel over which one can move a small device called the *cursor.* Most important, the device can be used in combination with the graphics display in such a way that placing the cursor at a particular position on the tablet will cause a graphic icon called the *tracking-cursor* to be positioned at the corresponding position on the graphics display. Thus, all of the facilities of a light-pen are provided (for example, drawing or pointing), without the associated arm fatigue and visual problems.

Besides the uses already mentioned, the tablet has certain other attractive features. For example, mounted on top of the cursor are four buttons which can be used to initiate different events or indicate responses to questions. One of the values in this is that the hand need not leave the cursor for the typewriter-type keyboard in order to reply to queries requiring a key-stroke response. (Note that of the graphic input devices listed above, only the tablet and the mouse conveniently lend themselves to the incorporation of such buttons.) Finally, coupled with appropriate software, the tablet can function as the input device to a pattern-recognizer which the user can "train" to learn and recognize a set of user-defined shorthand graphics symbols (such as for eight-note, rest, etc.).

4.1.2.4. *Sliders*

The "slider box" is another input device useful in interactive computer graphics. Designed by Fedorkow (see Fedorkow, 1978 for details), the device consists of a box containing three general purpose switches and two infinite sliders. Each slider (shown in Figure 2) consists of a touch sensitive continuous plastic belt in combination with a motion detector [2]. The user may touch the exposed part of the slider (circa 13 c.m.) and move the belt up or down. All of these interactions can be monitored by the host computer and used as control information. For example, in "conducting" a score, one slider can be used for tempo and the other for dynamics. Controlling the sliders with one hand leaves the other free to use the tablet. Thus very high bandwidth is possible with no typing and very little physical effort. Furthermore, even though there are only two physical sliders, they can be used to control several different "virtual" potentiometers which may be displayed on the CRT. In this case, one can rapidly change their context (i.e., reassign a slider to a different potentiometer) simply by pointing at the potentiometer with the cursor and touching one of the sliders. In so doing we see one of the most attractive features of the slider: it is *motion* rather than *position* sensitive. Therefore, unlike a normal potentiometer, there is no need to "null" it, or reset its position when switching it to control graphic fader A from controlling graphic fader B (which may be in a different position).

[2] While all electrical components were developed by Fedorkow, we are indebted to Allison Research, Inc., Nashville, Tennessee, for their co-operation in letting us utilize the slider mechanism developed by them.

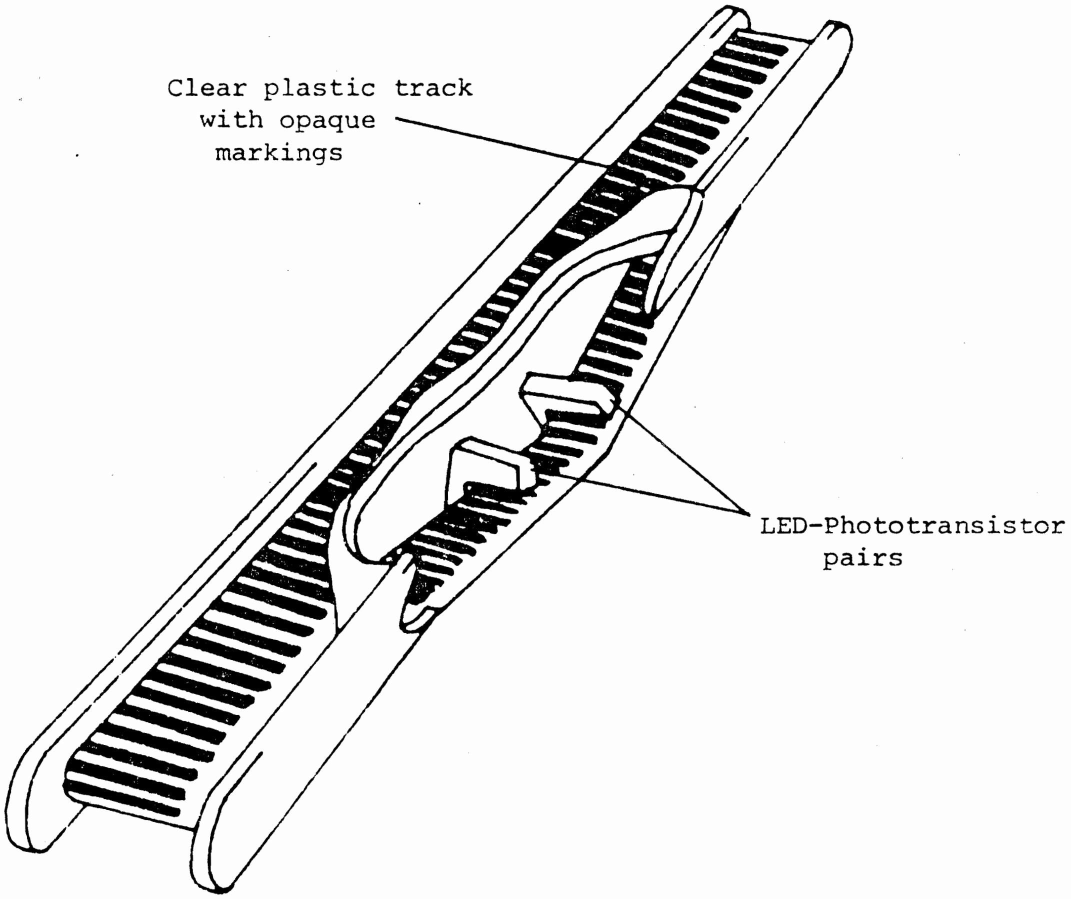

Figure 2 : The Slider

4.1.2.5. *Typewriter-type Keyboard*

In addition to the above, a normal typewriter-type keyboard is provided. This enables the graphics display to be used as a conventional terminal. When working graphically, however, a system can be designed such that about the only time the user need use the keyboard is in initially calling up a program and when assigning names to files, such as scores.

4.1.2.6. *Monitor System*

The function of the monitor system is straightforward. The implementation, however, includes a microprocessor-based system to enable the distribution of four channels of audio over (up to) sixteen discrete channels of amplification. This multiplexed, bus-oriented system is described in more detail in Fedorkow (1977), and Fedorkow, Buxton, and Smith (1978).

4.1.3. *Digital Synthesizer*

The digital synthesizer -- described more fully in Buxton, Fogels, Fedorkow, Sasaki, & Smith (1978) -- is one of the key components of the system. The device has sixteen digital sound generators. The generators are essentially fixed sampling rate, accumulator-type digital oscillators. The sampling rate of each oscillator is 50 kHz, giving a bandwidth of 25 kHz. Dynamic range is well over 60 dB (and should improve with further adjustment) while frequency resolution is approximately .7 Hz (linear scale) over the entire bandwidth. This will shortly be improved to enable resolution of less than 1 "cent" at even extremely low frequencies [3]. The output signal of each oscillator can be fed to one of four analogue output busses which may then either be fed directly to an amplifier, or to a *channel distributor* (Fedorkow, 1978; Fedorkow, Buxton and Smith, 1978). The waveform output by each generator may not only be user defined -- up to 8 waveforms available at one time -- but one may switch waveforms in mid-cycle. This is possible since the 16 oscillators share a 16k buffer of 12-bit words to store waveforms. This 16k of RAM is partitioned into 8 2k blocks, one for each of the 8 possible waveforms defined by the user or system. Apart from this memory configuration, this synthesizer is particularly interesting because the oscillators may be used to generate sounds according to five different synthesis modes: fixed waveform, frequency modulation, VOSIM, additive synthesis, and waveshaping [4]. This goes a long way towards a "universal module" -- that is, all modules of a uniform type (with the resulting ease of conceptualization and communication). While this is in direct contrast with analogue synthesizers, a very wide repertoire of sounds is possible

---

[3] One "cent" is an interval equal to 1/100 of a tempered semi-tone. One cent is slightly below the limit of human pitch discrimination and therefore represents the preferred pitch resolution of the oscillators. See, for example Backus (1969).

[4] An important point to note is that the use of the various modes is not mutually exclusive. That is, it is perfectly possible to be synthesizing an 8 partial tone using additive synthesis, while we have VOSIM, FM, and fixed waveform modes being utilized at the same time. The flexibility of the arrangement is obvious, as is its benefit.

(including all phonemes in Indo-European languages, for example).

4.1.4. *Host Computer*

The key demand on the host computer is the ability to support the demands of the software and peripheral hardware described. Of particular importance is the ability to meet the real-time demands of the system. Finally, in keeping with our axiom of accessibility, the system should be the smallest, least expensive machine capable of meeting these demands. For the initial implementation, we have chosen to utilize a Digital Equipment Corporation (DEC) PDP-11/45 machine. This processor is complemented by three disk drives, a magnetic tape drive, hard copy output (both graphic and alpha-numeric), cache memory (to increase speed of program execution), and full memory complement.

There are various good reasons for choosing this machine, apart from the fact that it was available. First, it supports the UNIX operating system, the attractions of which are discussed below. Second, it provides a powerful tool during the research and development phase, and yet is downward compatible (hardware and software) to a smaller, less expensive, more portable machine, the DEC LSI-11/2. Thus, once the initial research and development phase is completed and a more stable system evolved, accessibility can be greatly increased.

4.1.5. *Slave Processor*

The main function of the slave processor is to enable redundant computation and "mindless crunching" to be farmed out by the host computer. This frees valuable computational overhead for higher level processing. That is, the host machine serves a supervisory role while the slave does the leg work. An example of the type of computation carried out by the slave is the expansion and interpolation of stored functions and the direct servicing of the real-time demands of the synthesizer.

For the function of slave processor, a minimal configuration of a DEC LSI-11/2 was chosen. The processor includes 16K RAM, floating-point option, and interfaces. It is mounted in the chassis of the synthesizer and utilizes the synthesizer's power supplies. While this processor is perhaps more expensive than some micro-processors, it has the attraction that it is able to execute code written in the high level language of the host processor. Code can be written, tested, and debugged on the host and then down-line loaded to the slave. The 16-bit machine is also significantly faster in carrying out arithmetic operations than most of its 8-bit counterparts. Finally, it will retain compatibility even when the host is scaled down to the LSI-11/2.

4.2. *Software*

4.2.1. *Operating System*

One of the main attractions of the PDP-11/45 is that it supports the UNIX operating system (Thompson and Ritchie, 1974). Simply stated, in our opinion, UNIX is the most state-of-the-art operating system available commercially today for use on mini-computers. UNIX is a time-sharing system for which there exist single-user subsets for smaller machines (eg., Lycklama, 1978). Therefore, the same basic system can be used on both the large PDP-11/45 and the future LSI-11/2 host processor. In addition, there exists a UNIX based system for the slave processor (Lycklama & Christensen, 1978), which provides for overall system unity. With the time-sharing version, the attraction is that several users can utilize the music software simultaneously, thereby increasing accessibility. Furthermore, both research and development and application work can be undertaken simultaneously. An impression of the power of UNIX in combination with the PDP-11/45 can be gleaned by considering that the system can support up to eight other users -- doing text editing, graphics, etc. -- and still perform a sixteen part composition in real-time *without resorting to use of the slave processor!* Again, remember that all this is on a time-sharing system running on a mini-computer.

In terms of its other features, UNIX has a very convenient hierarchically structured file system which facilitates the handling of the numerous files used in a composition. Through the feature known as the "shell", often-used sequences of commands can be abbreviated by the user into a single command. In addition, it is the shell which enables us to achieve two other demands. First, to be able to use a control condition (such as type of terminal) to determine which version of a particular command is to be executed. Second, to enable the current process to be suspended while the user temporarily branches off to another. The system not only supports, but is written in one of the most sophisticated high-level languages available on a mini-computer: the language "C", which is discussed below. Partially due to UNIX being written in C, the system can be modified to suit the needs of a particular installation. UNIX is documented (Thompson, 1978) and has a good text editor. In summary, it is felt that our development would be nowhere near its current state were it not for the tools provided by this system.

4.2.2. *High Level Language*

As stated above, the language chosen for implementing the base structures is the language "C" (Kernighan and Ritchie, 1978). The attraction of this language is that it supports complex data structures (such as aggregates of data types) coupled with dynamic storage allocation. "C" is a structured programming language with a clear control structure. Since it was written for the PDP-11, it generates very efficient code, which is important, given our real-time constraint. Furthermore, it is well documented (Kernighan and Ritchie, 1978). Again, a great deal of our success has been due to the availability of such tools.

4.2.3. *Graphics Support*

One of the main software packages on our system which serves to illustrate the power of UNIX and the PDP-11/45 in combination with well-chosen peripherals is the graphics support package, GPAC (Reeves, 1978). GPAC is a comprehensive package for supporting interactive computer graphics, including animation. The package includes many high level routines (such as scale, rotate, etc.), the composite effect of which is to free the programmer from low level details in order to concentrate on more important issues. The package is device independent. That is, the same routines will function whether the output device is a CRT (raster-scan, storage, or vector-drawing), or a hard-copy plotter. Second, the facility for interaction is augmented in that GPAC is what is known as "event-driven". That is, programs (with the resulting graphical display of information) can be structured so as to have the flow of control determined by the type and value of various user generated "events". Examples of such events would be: pushing one of the buttons on the cursor, moving the cursor or fader, lifting the cursor from the tablet, the absence of user activity for a given period of time, etc. Finally, GPAC is well documented (Reeves, 1978). In summary, GPAC makes it very easy for the programmer to provide immediate visual feedback to the user following a particular action. That this feedback can be in graphical form greatly increases our bandwidth of communication and improves the potential for our achieving a good user interface.

5. *Conclusions*

The system introduced in this paper is rather young. All of the music work described has been undertaken since January, 1977. This includes not only the software, but the design and construction of the digital synthesizer as well. We feel that the success demonstrated thus far reflects well on the basic approach to design. While there remains a great deal left to be done and learned, we are certain of one thing: that in order to develop a useful tool for composers, musicians must be involved in the design right from the beginning, and yet not be forced to sacrifice their musicianship in order to do so. We believe that in following our current approach, we are making progress in realizing this philosophy in practice.

6. *Acknowledgments*

The work described in this paper has been undertaken as part of the research of the Structured Sound Synthesis Project (SSSP) of the University of Toronto. The SSSP is an interdisciplinary project whose aim is to conduct research into problems and benefits arising from the use of computers in music composition. This research can be considered in terms of two main areas: the investigation of new representations of musical data and processes, and the study of human-machine interacton as it relates to music.

The research of the SSSP is funded by the Humanities and Social Sciences Research Council of Canada (grant S76-0099), while logistic support comes from the Computer Systems Research Group (CSRG) of the University of Toronto. This support is gratefully acknowledged.

7. *References and Bibliography*

Backus, J. (1969). *The Acoustical Foundations of Music.* New York: W. W. Norton & Co.

Battier, M. & Arveiller, J. (1978). *Documents Musique et Informatique: une bibliographie indexee.* Ivry S/Seine: Elmeratto, 27 Rue Paul-Bert. Buxton, W. (1978). Design Issues in the Foundation of a Computer-Based Tool for Music Composition. *Technical Report CSRG-97.* Toronto: University of Toronto.

Buxton, W. & Fedorkow, G. (1978). The Structured Sound Synthesis Project (SSSP): an Introduction. *Technical Report CSRG-92,* Toronto: University of Toronto.

Buxton, W., Fogels, A., Fedorkow, G., Sasaki, L., & Smith, K. C. (1978). An Introduction to the SSSP Digital Synthesizer. Paper presented at the Third International Conference on Computer Music, Dept. of Music, Northwestern University, Evanston Illinois.

Buxton, W., Green, M. & Hume, S. (1978). *Music Software User's Manual.* Toronto: unpublished manuscript, SSSP/CSRG, University of Toronto.

Buxton, W., Reeves, W., Baecker, R., & Mezei, L. (1978). The Use of Hierarchy and Instance in a Data Structure for Computer Music. Paper presented at the Third International Conference on Computer Music, Dept. of Music, Northwestern University, Evanston Illinois.

Fedorkow, G. (1978). *Audio Network Control.* Toronto: M.Sc. Thesis, Dept. of Electrical Engineering, University of Toronto.

Fedorkow, G., Buxton, W. & Smith, K. C. (1978). A Computer Controlled Sound Distribution System for the Performance of Electroacoustic Music. *The Computer Music Journal* 2.3. Paper presented at the Third International Conference on Computer Music, Dept. of Music, Northwestern University, Evanston Illinois.

Kernighan, B. & Ritchie, D. (1978). *The C Programming Language.* Englewood Cliffs, N. J.: Prentice-Hall Inc.

Lycklama, H. (1978). UNIX on a Microprocessor. *The Bell Systems Technical Journal* 57.6, part 2: 2087-2102.

Lycklama, H. & Christensen, C. (1978). A Minicomputer Satellite Processor System. The Bell Systems Technical Journal 57.6, part 2: 2103-2114.

Melby, C. (1976). *Computer Music Compositions of the United States 1976.* Beverly Hills Va., Theodore Front.

Reeves, W. T. (1978). A Device-Independent General Purpose Graphics System in a Minicomputer Time-Sharing Environment. *Technical Report CSRG-93.* Toronto: University of Toronto.

Reeves, W., Buxton, W., Pike, R. & Baecker, R. (1978). Ludwig: an Example of Interactive Computer Graphics in a Score Editor. Paper presented at the Third International Conference on Computer Music, Dept. of Music, Northwestern University, Evanston Illinois.

Thompson, K. (1978). UNIX Implementation. *The Bell Systems Technical Journal* 57.6, part 2: 1931-1946.

Thompson, K. & Ritchie, D. M. (1974). The UNIX Time-Sharing System. *Communications of the ACM* 17.7