IEEE Computer, Aug 1978

A computer music system allows the composer to work with musical structure in a highly controllable and predictive way, and enables him to compose, modify, and synthesize an expanded range of musical expression.

Computer Music Composition: The Polyphonic POD System

Barry Truax
Simon Fraser University

Experiments in the musical applications of computer technology have been carried out over the past 20 years. Today, digital technology is regarded as the central field of music acoustical research, and many composers are actively pursuing their compositional goals primarily with that technology. Meanwhile, a wide range of amateur and semi-professional enthusiasts interested in computer music have joined the field, and a growing audience of listeners are finding that computer music is providing new and exciting aural experiences.

This paper is not intended to comprise a complete survey of the field—no book has appeared as yet that dares tackle that task; instead, it will bring together some general observations on the problems of musical creation within the composer-machine environment, and present a detailed account of the polyphonic composition/synthesis system known as POD,[1] which has been developed by the author over the last five years. The version of the system presented here comes the closest to being the central core of what might be envisioned as a "complete" system, in the sense of a system which allows a composer to take an initial idea from its very inception to the final recorded product. Naturally, the POD system continues to undergo development and expansion. However, our experience with it through actual and extensive compositional use suggests that it establishes a framework within which future changes may be incorporated, but which in itself will remain fixed.

Musical design within a composer-machine environment

A useful place to begin examining the field of computer music composition is to focus on three conventional and somewhat arbitrary subdivisions: sound synthesis, compositional structure, and composer-machine communication. Figure 1 represents these categories as a triangle, oriented so that the communicational aspect occupies a different level of generality (a meta-level) from the other two. By specifying sound synthesis and compositional structure, we follow the traditional division of music into sound and structure, by which we mean the acoustic repertoire, and the set of relationships between elements of this repertoire. The arbitrary sound-structure distinction, made mainly for convenience of representation, should not prevent us from considering methods of musical organization where sound and structure cannot be easily separated, as is often the case in electronic music.

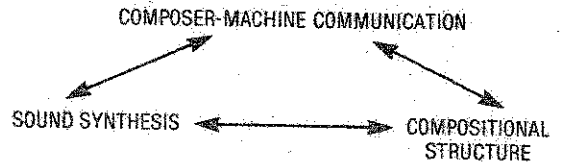

Figure 1. Subdivision of computer music composition.

0018-9162/78/0800-0040$00.75

Consider the variety of ways in which sound is structured in both language and music. Traditional scored music makes a clear division between the organization of the sound, as represented by the score, and its acoustic realization in performance. Even in non-notated traditional music, the tradition itself, intertwined with that of the entire culture, provides the structural basis for the organization of the sounds used (e.g., church modes, raga scales, gamelan melodies). In language, grammatical and syntactical rules provide a tightly defined structure for the limited phonemic repertoire. Electronic music allows the composer to create a wide range of sound as well as to control its organization, and hence has tended to blur the distinction between sound and structure. If, for instance, a tape composition consists of a complex, continuous sound texture, the structure is inseparable from the sound; and analysis of such work can be best made by describing the process that produced the sound structure.

Computer music work tends to re-emphasize the sound-structure distinction (that electronic music began to blur) by separating the actual synthesis or sound-producing method from the data representation used to prescribe the sound, or, in other cases, from the program procedures used to produce the sound. The method of data representation tends to resemble the score model referred to above, and hence tends to overlap more with that of instrumental composition, whereas the procedural approach (e.g., in the form of subroutines or machine operations) tends to resemble electronic synthesis procedures, such as those where a circuit diagram shows interconnected modules (e.g., generators, modification devices, logical decision units) and describes the process by which the acoustic output is realized. In a procedural model, process and structure become identical.

A computer program is not necessarily limited to either a score model or a procedural model, since it can realize either as a hierarchical control structure. Program designers, however, often tend to assume that composers wish to work with score-like data; therefore, many of the compositional features of computer programs today (e.g., MUSIC 5, MUSIC 360) encourage the kind of thinking associated with scores. Some composers, often those with an electronic music background, have developed systems that incorporate process-oriented elements of the voltage-controlled studio approach; examples are Koenig,[2] Chadabe,[3] Berg,[4] and to a lesser extent, Xenakis.[5]

Much of the history of computer music can be seen as having placed, at various times, differing emphasis on each of the three components noted in Figure 1. The extensive work initiated in the late 1950's at Bell Labs by Max Mathews et al[6] tended to emphasize sound synthesis and used procedures found in the electronic synthesizer (viz., the unit generator concept) to describe synthesis instruments. The result of their work (MUSIC 4, MUSIC 5) established a basic acoustic and compositional model which has become a norm for many other systems. Structural models were only implemented later by specific composers who designed their own compositional routines, such as those of Hubert S. Howe.[7] Composer-machine communication in the early systems was limited by the hardware restrictions, and often was characterized by slow batch processing. The GROOVE system, however, which allowed real-time interaction, was a significant step in the direction of interactive composer-machine communication.

The work of Lejaren Hiller,[8] on the other hand, started with a greater concern for compositional structure, as did that of Koenig[9] during the 1960's. Synthesis was not involved since the output was considered to be scored material for instruments. Composer-machine communication was interactive on a slower time scale, since the machine was used to compute particular compositional data structures which the composer examined on receiving the output, and which could only be modified through new input and additional program runs.

The minicomputer allowed a fundamentally different communicational relationship between composer and machine—interaction in real time.

Computer music programs today often use both acoustic and compositional models together, and although the range of potential models seems quite large, an up-to-date catalogue of well-defined ones in current use might be surprisingly small. Some models are based on the simulation or testing of traditional music acoustical models (in the case of synthesis) or traditional composition models (in the case of structural models). Such programs effectively allow us to verify our understanding of traditional methods. New models that have been introduced extend the range of possible musical expression, and it is these that seem most likely to establish computer music as a truly unique, non-derivative art form.

Although the early work tended to focus on either synthesis or compositional structure, there has been an increasing recognition in recent years that the kind of control that composers need over the complex resources with which they are concerned means that the communicational interface within the machine environment should not be arbitrary. The way in which ideas have to be expressed, the amount of data required, the time scale on which information is received by and demanded of the composer, even the modality of input (alpha-numeric, graphic, tactile)—all of these have an influence on both the strategy of the composer and even the character, in some cases, of the final output. The minicomputer was perhaps the first development to allow a fundamentally different communicational relationship to exist between composer and machine—namely the real-time inter-

active model. Composers now worked out their ideas interactively *within* the framework of the program, and were assisted in most cases by program strategies designed to give the composer as much information (including acoustic results) as possible to facilitate the design process. The obvious educational implications of this "knowing by doing" approach have found their outlet in the LOGO music system.[10] Some teaching applications of compositional and performance machines, along the lines of computer-aided instruction, have also been implemented, such as with the Dartmouth digital synthesizer.[11]

A newer, less developed communicational relationship exists within the work currently being done to establish microcomputer-based systems. These are usually linked to design of a hardware synthesizer, and the general orientation is towards a performance device in the sense of a musical instrument. In other words, the potential seems to exist that the precision of digital synthesis methods, together with the organizational ability of a CPU plus memory, can be applied to the development of performed music. We are faced with the possibility of "smart instruments" just as we have seen "smart terminals."

This cursory overview has tried to place the various kinds of computer music work within a perspective based on models of synthesis, compositional structure, and composer-machine communication. Musical activity within any of these systems takes on different characteristics, ranging from simulation of traditional models to the creation of entirely new models that require new performance strategies on the part of the composer, and that result in a new range of possibilities for musical expression.

The POD system for composition and synthesis

The POD system was first developed by the author at the Institute of Sonology, Utrecht, during 1972-73,[12] where it established itself as an interactive program for composition aided by real-time synthesis. This initial version was designed for a PDP-15 machine with 12K memory with data storage on DECtape. During 1974-75, the program was translated to a Hewlett-Packard 2116 machine with 16K memory in the Department of Psychology at Simon Fraser University, where it underwent considerable compositional improvement, aided by disk storage of compositional data. From 1976 onwards, the development on non-real-time synthesis at a fixed sampling rate from a digital magnetic tape unit greatly improved the synthesis quality that could be achieved. Secondly, during this period, the polyphonic system described in the next section of this paper was developed. During 1978, the program will be translated to a NOVA 3 machine at Simon Fraser University, and a PDP-11 version of POD6 should be completed. In May 1978, the entire polyphonic system was installed at EMS in Stockholm on a PDP-15 machine.

The compositional model of the basic POD system has been described in detail elsewhere,[13] so we will only touch on certain aspects of it here. A schematic representation of the compositional strategy is illustrated in Figure 2. Its most basic characteristic is that it maps the timbral, acoustic domain into the structural, syntactic domain, represented as stochastic distributions (or sequences) of sound events with varying density. More specifically, the composer sets up general patterns of frequency regions, amplitude ranges, and density of attack points (the "syntactic field specification" of Figure 2), and then determines how a specific set of defined "sound objects" (or timbral entities) is to be selected and displayed within the given structure. The specific occurrences of events are determined by the "distribution algorithm," which is based on a Poisson distribution. A further level of control is called "performance variables," and with these variables the composer can achieve alternative versions of the same set of events, mainly through control of the time structure and the ratio of event duration to entry delay (i.e., the time between attack points). Since there exists a variety of control possibilities, stochastic choices need not be adhered to, and so the composer is also involved in determining the degree to which random choices determine any given parameter, and to what extent and at what level deterministic choices are made.

The program encourages and facilitates a structural working method in the sense that one may design and work with the characteristics of large-scale sections, even an entire work, just as easily as with relationships between individual events. However, it is characteristic of POD that, although one may restrict the stochastic choices made by the program, one never actually determines a specific event completely. Instead, the composer sets up the rules by which specific events are chosen, and only if those rules are sufficiently restrictive does the determination of a specific event become entirely predictable. Unlike the electronic music synthesizer, the computer program allows structure to be worked with in a highly controllable and predictive way. In the case of POD, it also facilitates what is often called a "top-down" approach, where higher-level compositional decisions are made first, and the details are determined last. In this sense, the machine environment actually allows a working method and compositional process to occur that is fundamentally different from traditional practice. We have argued elsewhere that it is this ability to

The most significant aspect of computer music applications is the effect they have on the process of composition.

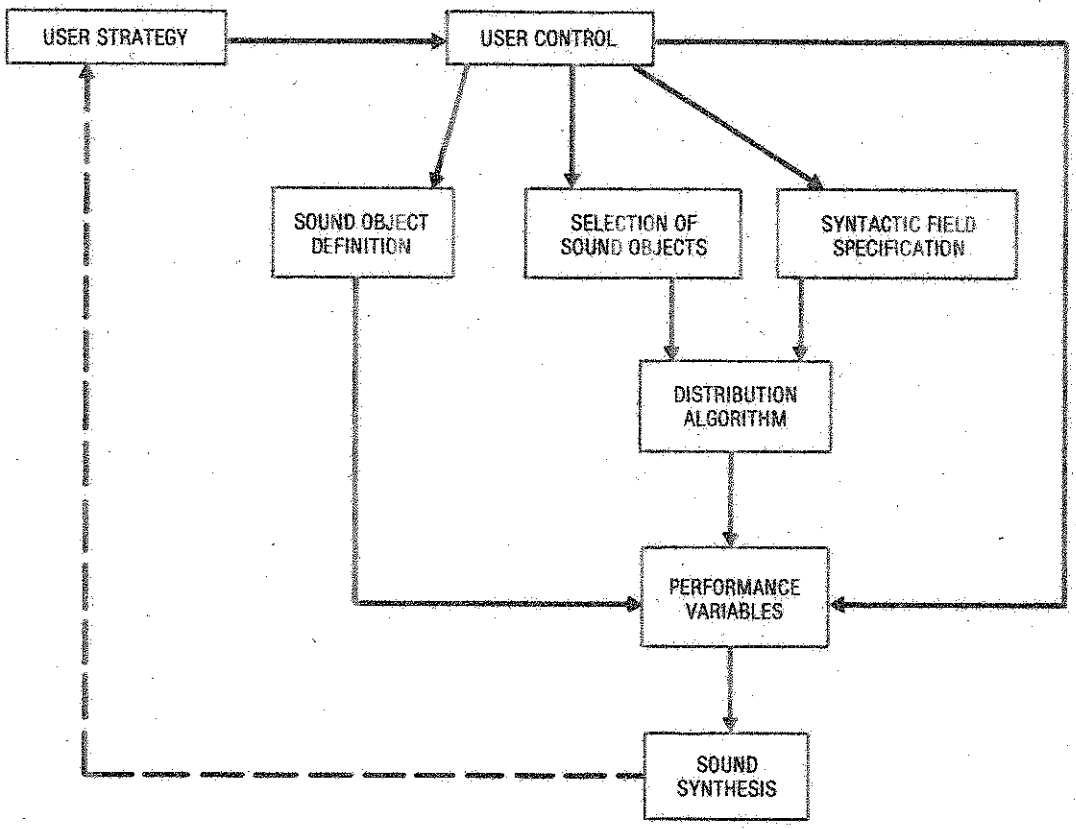

Figure 2. Compositional model of the POD system.

affect the process of composition that is the most significant aspect of computer music applications.[14]

Another way of looking at the program is to examine where it falls along the generality and strength continuum as shown in Figure 3. This continuum, which has been discussed in greater detail elsewhere,[15] establishes the basic inverse relation between the strength or efficiency of a method of problem solution and its range of applicability (i.e., its generality). The distinction is useful in analyzing composer-machine communication because a portion of the knowledge used (in the procedural sense of knowledge of how to perform certain actions) resides in the machine as programs, and the rest is provided by the user; the generality and strength relationship predicts the amount of user-machine interaction based on the relative amount of procedural knowledge implemented in the machine. Weak methods are characterized as necessitating a large amount of user information with low guarantee of well-formed result, but offering a wide range of potential output. Strong methods offer a better guarantee of a well-formed result (because the strategies used are based on implemented knowledge about such results), and hence require less user input (i.e., they seem more automated); however, the range of output is necessarily restricted. User interaction with the system increases to a maximum somewhere in the middle of this continuum, since at either end the user is either too burdened with data specification or else is not needed at all. This argument should not be taken to mean that user interaction is always of the utmost importance or that it should always be maximized; interaction promotes learning, and thus should be encouraged only in situations that attempt to optimize the educational potential of a system.

POD attempts to maximize user interaction and learning by incorporating strong synthesis methods

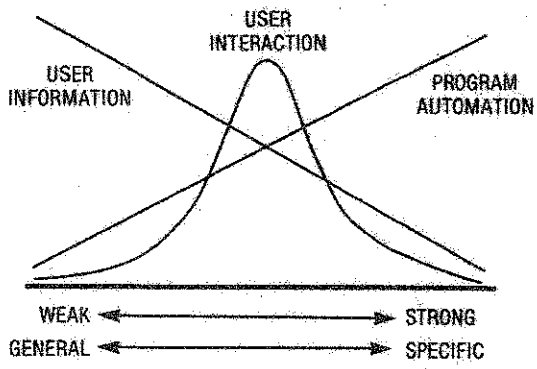

Figure 3. Variation of user information, program automation, and user interaction as a function of the generality and strength of a system.

(e.g., frequency modulation) and strong structural compositional methods, as described above. Generality is also maintained at a reasonable level by the use of a range of control procedures from deterministic to stochastic, and by the hierarchical nature of the data structure. The data structure is hierarchical in the sense that variables at one level interpret data at a lower level (e.g., performance variables interpret event data, and sound objects interpret acoustic data). Although the same range of output could be realized without such variables, it can be accessed much more quickly and efficiently with them, and hence a wider range of output (which reflects an increased generality) is simply more readily available to the user.

In interactive computer composition, the composer is confronted with the very important relation between the structure of the sound output and the mental processes that produced it.

In terms of the distinctions made earlier between different models relating sound and structure—namely the score model and the process model—POD can be seen to incorporate some aspects of each, or else POD users can be seen to place their own work somewhere along the continuum between these two models. Composers with a score orientation often tend to use the more deterministic procedures offered by the program, since they are thinking along the lines of establishing data structures that describe how sound events are to be generated. The more process-oriented composer will tend to set up procedures, such as those with various stochastic elements, with constraints operating on the process at different levels, and then let the program determine the actual details. All such users will interact with their data structures or procedures through the acoustic output they receive, and so in every case, a considerable learning process takes place. In this process, the composer is confronted with the very important relation between the structure of the sound output and the mental processes that produced it—a relation which Laske has asserted is central to a cognitive approach to studying musical activity.[16]

The polyphonic POD system

Figure 4 shows the most advanced state of development of the POD system to date—that realized at Simon Fraser University and at EMS, Stockholm. The system is completely polyphonic and includes many powerful compositional and synthesis options. Approximately eight compositions by four different composers (Truax, Piché, Goldberg, Keeble) have been realized on the system, as shown in its present version, since January 1977; however, all or part of at least another dozen have been realized by many more composers at Simon Fraser and Utrecht during 1974-76 when only the POD5 and POD6 systems (monophonic, real-time) were available. Three

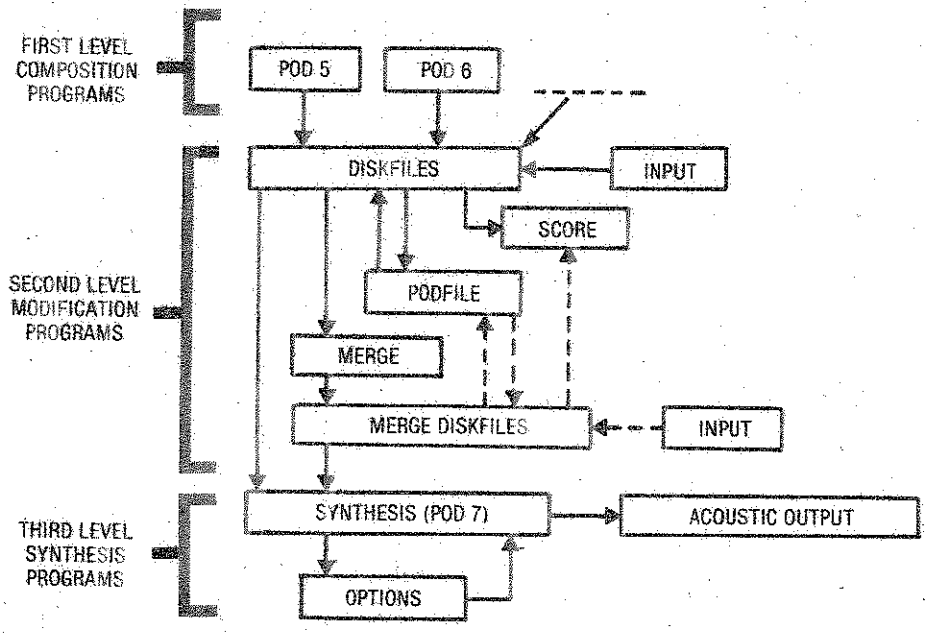

Figure 4. Block diagram of the polyphonic POD system. Solid lines denote existing data exchange permitted between programs (POD5, POD6, SCORE, PODFILE, MERGE, POD7) and storage files (diskfiles, merge diskfiles). Dashed lines indicate proposed additions to the present system. "Input" denotes event-by-event data specification, an alternative to use of the first-level composition programs.

of the author's computer music compositions are available on a recording.[17]

The system consists of three levels of programs, which may be broadly characterized as being concerned with composition (i.e., producing data structures describing sound patterns), modification, and synthesis. The *first level composition programs* are characterized as being interactive, real-time programs for composition and synthesis. The two presently in existence, POD5 and POD6, have been described extensively elsewhere.[18] In most cases, all of the basic compositional structures or layers of a structure are worked out at this first level. The emphasis on real-time synthesis at this level is to maximize the effectiveness of sound results on the composer's strategy. In other words, the composer should be able to hear intermediate results at this early stage when design plans are being worked out, or simply when preliminary experiments are being made. What determines whether the composer will hear polyphonic or monophonic output at this stage depends on the hardware or software synthesis strategies being used. Until recently, all POD installations were restricted to monophonic real-time output at this first level because all were using software methods of sound synthesis. As soon as appropriate hardware generators in sufficient numbers can be substituted to handle this task, then the composer can hear polyphonic results at this stage as well. Because of the availability of digitally controlled analog FM generators at EMS, Stockholm, the new POD system there is polyphonic in real time. Further improvements are possible with digital generators, such as are being designed at Stockholm and other centers.

The POD system consists of three levels of programs—the first concerned with composition, the second with modification, and the third with synthesis.

At the first level of composition programs, any number of related programs could exist. Within the present POD system, the idea is that each program at this level has the same set of compositional strategies available, but they each use different acoustic models of synthesis. For instance, POD5 uses fixed waveform synthesis (with common waveforms plus harmonic constructions from these) and an option for amplitude modulation; on the other hand, POD6 uses John Chowning's method of frequency modulation synthesis developed at Stanford,[19] notable for its production of time-dependent spectra which are heard as possessing the energetic "lively" quality of instrumental and environmental sounds. There is no conceivable limit to the number of synthesis methods that could be implemented in similar programs. However, it is probably certain that those which will work best within an interactive composition environment are those which I have termed as using 'strong' methods of acoustic synthesis[20]—that is, those which guarantee a well-formed (and hopefully wide-ranging) output for a minimum of input data specification. The reason for this strength is always the elegance and effectiveness of the acoustic model on which the synthesis program is based. Suggested additional methods, apart from those already in use, are Kaegi's VOSIM model[21] and the user-designed band-limited spectra proposed by Marc Le Brun of Stanford.[22]

The output of the first level of program is a data file stored on disk, called simply a "diskfile." This file can be compared to a traditional musical score to the extent that it is merely a description of the desired acoustic output and not the sound itself. Hence it is more compact (in data specification) than the synthesized output and can be more easily manipulated. Such files serve as basic input to the next level of programs.

The *second-level modification programs*, as the name suggests, perform various useful operations on the diskfiles created at the first program level. First of all, some kind of hardcopy output may be requested from the SCORE program, either as a numerical printout in a coded form for transcription into conventional musical notation, or else as graphic sound event drawings, which have in at least one case been useful for diagramming a tape accompaniment in an instrumental score.[23] Considerable work has been done in other centers regarding actual manuscript preparation. However, the most useful operations at this level are those which edit existing files (the PODFILE program) or combine them into polyphonic "merge" files (the MERGE program).

PODFILE allows any piece of data in diskfile to be accessed and edited. In addition it performs simple operations such as copying files into duplicate files in preparation for the creation of variants. The editing done by this program is either manual (editing one event at a time) or systematic (generating or operating on blocks of data). The facility for generating sets of data is limited to certain simple data-generating algorithms of less sophistication than those found in the basic first-level composition programs. However, in many day-to-day situations, such data generation is often useful. The methods currently employed to generate a block of data are constant value, linearly changing values, exponentially changing values, random choice from within a range, random choice from a given list of possible values, and sequential choice of a list of possible values. In addition to data generation methods, operators are also a useful part of the PODFILE program. These allow systematic variants to be created, whereby the data in two or more files are related by the fact that one has been operated upon to realize the other. Any of the four basic acoustic parameters used (time delay, maximum amplitude, sound object number, frequency) may be subjected to such operations, with frequency being operated upon linearly

(in Hz) or logarithmically (in musical semitones). Examples of the operators are linear transposition or increment, percentage transposition or increment, random change within some specified percentage increment, and inversion around a point (i.e., data less than the point are changed to being an equal amount greater, and vice versa). Many useful musical and acoustic operations require such operators. Musical transposition or inversion of pitch is easily accomplished by applying these operators to the pitch domain. The acoustic choral effect is realized by creating a number of variants where the frequencies are varied, say within one percent, of exact tuning. These may later be combined with a slight offset time for slightly delayed attacks that enhance the sense of realism and suggest an ensemble texture. Random change allows a stochastic element to enter as well.

Two diskfiles may be combined by the MERGE program into a polyphonic "merge" diskfile. Note that two definitions of polyphonic need to be distinguished here. First we have the simple case where adjacent sounds overlap. In music this occurs over a range from complete overlap in chords to a slight overlap that produces a smooth *legato* transition between events. The second case occurs when two "voices" or "layers" are combined in the sense of being mixed. It is this second sense that is intended by the MERGE program. Even there, however, a wide range of effects can be achieved. When the two layers have dissimilar rhythms, conventional polyphony or counterpoint results. When the two components are isorhythmic (i.e., they have the same time structure), then the result will be heard as chordal, or as an echo (if a slight delay is introduced), or as a compound timbre, if the two spectra fuse completely, or else as the ensemble texture called the choral effect if the two differ by only slight amounts in exact tuning and attack points. It is fortunate that such a diverse range of cases can all be realized by a straightforward mixing of files. It should be noted that the arrows in Figure 4 indicate that the MERGE program may be used recursively; that is, a merge diskfile may be combined with itself, or another like it, or else with a non-merged diskfile. Any number of recombinations or voice additions may be requested by successive runs of the program. These components in a merge file may be labeled as having a particular voice number, which has certain possible uses later on; at the moment eight such voices may be so tagged with a voice number. The main consideration to be given in the merge file operation is that one cannot combine sounds in an unlimited fashion without exceeding the ability of the synthesis program to calculate the composite sound. Since the word size of the synthesis routine is limited, usually by the number of bits in the D/A converter being used, so is the maximum amplitude of the composite sound, and hence its loudness. Therefore, if many sounds are to be combined by the MERGE operation, they must have sufficiently low maximum amplitudes that their composite amplitude remains within the given limits. Unfortunately, there is no such limiting loudness level known in the environment except our reaction in terms of pain, discomfort, or deafness.

In future developments, the merge diskfiles will also be candidates for input to the SCORE program and the PODFILE editing operations. In addition, direct score input to either the diskfile or merge diskfile will be made available by an INPUT routine, an operation similar to most MUSIC 5 type programs where all input is via score-card or note-card input, which specifies the desired result event by event.

Finally we come to the *third-level synthesis programs* which actually calculate the sound pressure data, store it on digital magnetic tape, and play the tape through a D/A converter equipped with a low-pass filter to smooth the waveform. The audio signal can then be recorded or monitored. This synthesis routine, which has been named POD7, can be divided into the following subroutines: 1) data file decoding; 2) synthesis calculation; 3) digital reverberation calculation; 4) disk-to-magnetic tape data transfer; and 5) tape playback.

First of all, data file decoding will be required for each different type of data file, which in our case is either the normal diskfile or the merge diskfile (Figure 4). For various reasons the data formats are different for each of these, necessitating certain differences in the decoding algorithm. Secondly, a synthesis calculation routine must exist for every synthesis type found in the first-level composition programs, since the program must be able to calculate the same sounds the composer heard at that level. In fact, a non-real-time program such as POD7 realizes such a calculation much more accurately than the real-time equivalents since there is no need to get samples out fast enough for real-time synthesis. Of course, if the calculation is too slow, or inefficient, the so-called "turnaround time" until results are heard becomes impracticably long. POD7 is presently only used with frequency modulation synthesis, which it calculates using an array of 2048 places describing a quarter sine wave. This accuracy is especially important for low frequency sounds or those of lower amplitude. The synthesis program need only be able to calculate single sounds and mix them with previously calculated sounds. Whether the composition is polyphonic in either of the two senses mentioned above (overlapping sounds or combined voices) is immaterial. In both cases, individual sounds are simply mixed with (i.e., added to) other sounds.

The sound pressure data so calculated are actually stored on disk, since access to it is both fast and random. In practice we have used a "mixing file" of a half-million words. It is cleared at the beginning, and every sound that is calculated is added to the values already stored at the place in the file where the sound is to begin. Thus the polyphonic effect is built up, sound by sound. The program keeps an elaborate accounting of time within the

mixing file and elapsed time in the composition. Since the composition is assumed to be proceeding sequentially in time, it is known that at any given point all future sounds will lie further on in the file. This means that once the mixing file is filled to a certain point, its contents can be dumped onto the magnetic tape up to the point where the last sound began. These transferred blocks then can be deleted from the disk, and the next blocks of acoustic data can take their place. Thus, the mixing file need only encompass, for example, one minute of actual sound, whereas the composition may extend indefinitely, or at least to such a length of magnetic tape as is available.

The third part of the program is that for digital reverberation. Because of the rather unsatisfactory quality of normal analog studio reverb units, many composers wish, in conjunction with the precision of digital synthesis, to have a cleaner and more even reverberation. This can be achieved digitally by a program with a set of "delay loop" arrays. When each array is filled, one retrieves the values written at the beginning (some number of samples previously) as an echo which can be combined with the original signal. By careful construction of the delay loops, 1000 echoes per second, giving flutter-free reverberation, can be achieved. Various models of reverberation have been discussed by Schroeder[24] and implemented by Moorer and others at Stanford.[25] In our case, the reverberation is calculated after a block of sound data has been completed in the mixing file. Two difficulties in this system encountered to date, however, have been the problem of integer arithmetic used in the digital reverberation, which leaves a residual noise at the end of the reverb, and transients in the reverb experienced through the segmentation of the sound data on the disk as described. In many cases, however, neither of these is objectionable.

The POD system is dynamic—the designer may add new options, such as stereo output.

The fourth and fifth parts of the program encompass the data transfer onto magnetic tape and its being read back at a continuous data transfer rate to the D/A converter. This last operation is the most difficult since no interruptions can be tolerated. A standard "double buffering" technique is used whereby through interrupt programming one array is being filled while the other is being clocked at a uniform rate out to the converter. It should be noted that high-quality digital magnetic tape with no imperfections in emulsion needs to be used, since these imperfections cause blocks to have errors and these blocks must then be skipped during playback. There is usually no extra time available during playback for error checking; hence, only the best tape with no "dropout" can be used.

In addition to the polyphonic output with the digital reverb option, other options can be profitably introduced into the synthesis routine. The one which we have experimented with most is the possibility of binaural output (stereo). This requires two converters which can be accessed simultaneously, or at least sample by sample. The relationship which we have experimented with between the two signals being fed to the two converters is that of small time delays simulating those experienced when a sound is oriented at some particular direction around the head. Time delays of this sort (termed interaural time delays) are zero for sounds directly in front or at the back, and maximally about 0.6 milliseconds for sounds to the side directly facing one ear. In between one can distinguish many separate directions. We have chosen 12 positions between far left and far right, because this is the largest number of positions that can be simulated when the sample period is of the order of 100 microseconds.[26] That is, the time delay arises when the given sample is displaced by a certain number of sample positions and sent to the opposite converter channel. Hence the smallest delay is one sample, and the maximum is determined so that it will reproduce the maximum interaural time delay quoted. It should be noted, however, that with such "double" samples being calculated, the playback will resemble a tape being played at half speed, and will therefore have to be recorded at half speed as well. When the sounds are heard on headphones, the definition of azimuth is exact; on loudspeakers it is less exact since additional time delays are introduced by the different distances the sound has to travel from each speaker to the listener. Front-back distinctions can only be achieved through additional, much smaller time delays that simulate the reflections from the pinna ridges of the outer ear. These are maximally 100 microseconds, and hence extremely small time displacements are required to realize these. The proposed acquisition of a faster magnetic tape drive, and hence higher sampling rate and shorter sampling periods, may allow experimentation in this direction.

In summary, we have described a system of compositional file handling which allows a wide range of polyphonic, timbral, and spatial possibilities with efficient data manipulation (in our case, directed from a high-speed CRT terminal). Besides the software, the system described requires at least one disk and one magnetic tape unit, with preferably a high-speed line-printer for hardcopy and CRT unit for rapid user input. High-quality D/A converters, of 12 to 16 bits with efficient low-pass filters, and clocked output are also required. Such a system is designed to operate within a well-equipped minicomputer environment. Large computers could handle the data calculation, of course, and could decrease the computing time as well, but they are more expensive to use, and conversion is always done on a smaller machine

anyway. Unfortunately, microcomputers as yet do not have the computational speed or efficiency to approach the basic requirements of such a polyphonic system. By the time they acquire enough software and hardware to do so, they will resemble a minicomputer more than a micro.

Unfortunately, micros as yet do not have the speed or efficiency required by a sophisticated polyphonic system.

However, it can be proposed that substitution of hardware generators, such as are being developed in many centers, interfaced to the more modest resources of the microcomputer, would render the need for disks and magnetic tape units (the most expensive parts of this system) unnecessary. Such a proposal is feasible, and because of its attractiveness of price, its implications should be carefully understood. The main problem with the hardware implementation is exactly that it is hardware. That is, it is designed only to perform certain operations which cannot be easily changed. Every type of synthesis manipulation described would require a separate hardware unit to be designed, built, and debugged. Since the polyphonic requirements would demand a large number of such units, even a multiplexed oscillator design would eventually limit the complexity of the output. For instance, in a choral effect of eight overlapping mistuned unisons of frequency-modulated sound, 16 oscillators would be required per note; 16 such voices would require 256 such oscillators, linked in FM pairs. Reverberation or binaural time delays would require additional units. In addition, it is debatable whether reasonably priced hardware units can achieve the same accuracy of calculation as a standard mini equipped with 32-bit integer multiply and divide arithmetic. And finally, the system described here is limited in synthesis capability only by the number of synthesis programs that have been written.

In the hardware case, every new synthesis algorithm requires new hardware units to be built. Although hardware design should be encouraged, the danger exists that its requirements (in time, money, and design considerations) may compromise the final acoustic result.

However, hardware devices of remarkable power and flexibility are in fact being built. Alles and di Giugno have recently described a one-card synthesizer which provides 64 FM oscillators and other attractive options.[27] Software for controlling such systems is also being planned.[28] Hardware synthesis devices are frequently being incorporated in performance-oriented digital synthesizers—some on a very large scale, such as the Samson synthesizer at Stanford; and the Alles "digital sound synthesis system,"[29] developed at Bell Labs; and others on a smaller scale, such as the Synclavier, developed by Jon Appleton, Sydney Alonso, and Cameron Jones at Dartmouth. One could easily compare the two major trends in computer music today—performance-oriented digital synthesizers on the one hand, vs. non-real-time, general-purpose composition and synthesis systems, such as MUSIC 5, MUSIC 4BF and MUSIC 360 on the other—to the situation in electronic music 15 years ago when a split occurred between analog synthesizers, usually with keyboards, and the classical or voltage-controlled studio. With digital technology, however, and its use of memory, storage and recall, programs and logical operations, the amount of difference between the two approaches may well be lessened, and composers will often use some combination of real-time performance techniques and non-real-time compositional methods. With highly flexible systems available, the composer will have a choice of working method just as much as of acoustic repertoire or compositional technique. The POD system, established firmly on the side of non-real-time composition, will likely remain in some middle ground between the newer performance systems and the general-purpose, score-oriented systems. Its specialized approach to composition will hopefully remain useful to composers as a framework within which their ideas may be realized, and to system designers as a source of ideas. ■

References

1. Acronym for Poisson Distribution.

2. G. M. Koenig, "Skizzen zum Schema eines Computer-Programmes für die Klangsynthese (SSP1)," Institute of Sonology, Utrecht, unpublished manuscript, n.d.

3. J. Chadabe, "Some Reflections on the Nature of the Landscape within which Computer Music Systems are Designed," *Computer Music Journal*, Vol. 1, No. 3, 1977, pp. 5-11.

4. Paul Berg, "ASP Report," Institute of Sonology, Utrecht, unpublished ms., May 1975; idem, "PILE—A Description of the Language," Dec. 1976; idem, "PILE—A Description of the Compiler," Feb. 1977.

5. I. Xenakis, *Formalized Muisc*, Indiana Univ. Press, Bloomington, 1971.

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

7. H.S. Howe, "Composing by Computer," *Computers and the Humanities*, Vol. 9, 1975, pp. 281-290.

8. L. A. Hiller and L. M. Isaacson, *Experimental Music*, McGraw-Hill, 1959; L. Hiller and R. A. Baker, "Com-

puter Music," in *Computer Applications in the Behavioral Sciences*, H. Borko, ed., Prentice-Hall, 1962, pp. 424-451; L. Hiller, "Programming a Computer for Musical Composition," in *Papers from the West Virginia University Conference on Computer Applications in Music*, G. Lefkoff, ed., W. Virginia Univ. Library, 1967, pp. 63-88.

9. G. M. Koenig, "Project 1," *Electronic Music Reports*, Institute of Sonology, Utrecht, No. 2, 1970, pp. 32-44; "Project 2: Computer Programme for the Calculation of Music Structure Variants," *Electronic Music Reports*, No. 3, 1970, pp. 4-161.

10. S. Papert, "LOGO Manual," MIT Artificial Intelligence Laboratory, 1973; Jean Bamberger, "Learning to Think Musically (A Computer Approach to Music Study)," *Music Educators Journal*, Vol. 59, 1973, pp. 53-57; S. Beckwith, "Composing Computers for Kids—A Progress Report," *Canada Music Book*, No. 9, 1974; idem, "Talking Music with a Machine," *College Music Symposium*, Vol. 15, 1975.

11. S. Alonso, Jon Appleton and C. Jones, "A Special Purpose Digital System for Musical Instruction, Composition and Performance," *Computers and the Humanities*, Vol. 10, 1976, pp. 209-215.

12. B. Truax, "The Computer Composition-Sound Synthesis Programs POD4, POD5, & POD6," *Sonological Reports*, No. 2, Institute of Sonology, Utrecht, 1973; idem, "Some Programs for Real-Time Synthesis and Composition," *Interface*, Vol. 2, 1973, pp. 159-162.

13. B. Truax, "A Communicational Approach to Computer Sound Programs," *Journal of Music Theory*, Vol. 20, No. 2, 1976, pp. 227-300; reprinted in part in *Computer Music Journal*, Vol. 1, No. 3, 1977, pp. 30-39.

14. B. Truax and J. Barenholtz, "Models of Interactive Computer Composition," *Proceedings of the Third International Conference on Computing in the Humanities*, University of Waterloo Press, 1977, pp. 209-219.

15. B. Truax, "The Inverse Relation between Generality and Strength in Computer Music Programs," *Proc. First International Computer Music Conference*, MIT Press, in press.

16. O. E. Laske, *Music, Memory and Thought*, University Microfilms, Ann Arbor, Michigan, 1977, pp. 3-18.

17. Melbourne SMLP 4033.

18. B. Truax, "A Communicational Approach," pp. 258-281; reprinted in *Computer Music Journal*, Vol. 1, No. 3, 1977.

19. J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Journal of the Audio Engineering Society*, Vol. 19, 1971, pp. 1-6; reprinted in *Computer Music Journal*, Vol. 1, No. 2, 1977.

20. B. Truax, "A Communicational Approach," p. 235; idem, "The Inverse Relation Between Generality and Strength."

21. W. Kaegi, "A Minimum Description of the Linguistic Sign Repertoire," *Interface*, Part 1: Vol. 2, 1973, pp. 141-153; Part 2: Vol. 3, 1974, pp. 137-157; idem, "VOSIM—A New Sound Synthesis System," paper delivered to the AES, Zurich, 1976.

22. From a talk presented at the Computer Music Conference, San Diego, 1977.

23. B. Truax, "A Communicational Approach," p. 268.

24. M.R. Schroeder, "Natural Sounding Artifical Reverberation," *Journal of the Audio Engineering Society*, Vol. 10, 1962, p. 219.

25. J. A. Moorer, "Signal Processing Aspects of Computer Music—A Survey," *Proc. IEEE*, 1977; reprinted in *Computer Music Journal*, Vol. 1, No. 1, 1977.

26. M. Gerzon, "Dummy Head Recording," *Studio Sound*, May 1975, pp. 42-44.

27. H. G. Alles and P. di Giugno, "A One-Card 64-Channel Digital Synthesizer," *Computer Music Journal*, Vol. 1, No. 4, 1977.

28. J. Lawson and M. Mathews, "Computer Program to Control a Digital Real-Time Sound Synthesizer," *Computer Music Journal*, Vol. 1, No. 4, 1977.

29. H. G. Alles, "A Portable Digital Sound Synthesis System," *Computer Music Journal*, Vol. 1, No. 4, 1977.

Bibliography

1. Battier, Marc and Jacques Arveiller, *Musique et Informatique: Une Bibliographie Indexée*, Université Paris VIII, Vincennes, 1976.
2. Buxton, William, "A Composer's Introduction to Computer Music," *Interface*, Vol. 6, 1977, pp. 57-72.
3. Chadabe, Joel, "Some Reflections on the Nature of the Landscape within which Computer Music Systems are Designed," *Computer Music Journal*, Vol. 1, No. 3, 1977, pp. 5-11.
4. Grey, J. M., "An Exploration of Musical Timbre," STAN-M-2, Stanford, Dept. of Music, 1975.
5. Hiller, Lejaren A. and R. A. Baker, "Computer Music," in *Computer Applications in the Behavioral Sciences*, H. Borko, ed., Prentice-Hall, 1962, pp. 424-451.
6. Howe, Hubert S., "Composing by Computer," *Computers and the Humanities*, Vol. 9, 1975, pp. 281-290.
7. Laske, Otto E., *Music, Memory and Thought*, University Microfilms International, Ann Arbor, 1977.
8. Lincoln, Harry B., *The Computer and Music*, Cornell University Press, 1970.
9. Mathews, Max, *The Technology of Computer Music*, MIT Press, 1969.
10. Moorer, James A., "Signal Processing Aspects of Computer Music—A Survey," *Proc. IEEE*, July 1977; reprinted in *Computer Music Journal*, Vol. 1, No. 1, 1977.
11. Truax, Barry D., "A Communicational Approach to Computer Sound Programs," *Journal of Music Theory*, Vol. 20, No. 2, 1976, pp. 227-300; excerpt reprinted in *Computer Music Journal*, Vol. 1, No. 3, 1977.
12. _____ and J. Barenholtz, "Models of Interactive Computer Composition," *Proc. Third International Conference on Computing in the Humanities*, University of Waterloo Press, 1977, pp. 209-219.
13. Wiggen, Knut, "The Musical Background of Computer Music," *Fylkingen International Bulletin*, No. 2, 1969, pp. 8-12.