Every musician who has tried to dream in the Procrustean beds of Computer Music has uttered a moan and thought of ways to arrange things more comfortably. This may take the form of trying to correct particular discomforts or designing a whole new four-poster complete with gilt cherubs.

To make software that will do everything for everybody is impossible, but it is an interesting exercise for a composer to try to express his particular musical desires in the form of a computer language. The goal may be a particular composition or, more generally, a composing and performing style out of which a number of pieces can be made. It will allow certain things and not others; perhaps all we will ask of it is an interesting and unique range of possibilities and a tolerable level of reliability.

My own bad design had not so much the form of the four-poster as that of a hammock under the trees, with birds and perhaps a frog or two twittering in the shrubbery. Although it seemed simple in conception it proved difficult to construct, as a large number of questions had to be asked and answered in order to make something that would "support the weight" of practical use. This paper is about those questions and the provisional set of answers to them that comprise the computer music program called MABEL. The fundamental concept of this program is the idea of the computer as an instrument capable of manipulating composed ideas but at the same time highly subjective to performance gesture. I will deal with the issues involved in developing this language as they came up.

The instrument for which the software was designed is the Buchla 400, a versatile computer-controlled digital performance stationized with many desirable features of the "soft instrument" on which one can freely present the qualities of the relationships of performer, instrument and sound.

1. THE MUSICAL IDEA

I find it useful to think of music as topologies, streams, layers, events, trajectories, processes. Traditional conceptions of notes, instruments, orchestras tend to lead one to traditional musical results. I would like to work towards a global, rather than an acridic way of describing musical process. The practical problem here is that global concepts have to be modularized and atomized in order to be made into working programs. No sooner has the grand design appeared in a hallucinatory flash than it must be dismembered and dissected into subroutines and device drivers. So at the level of programming we are dealing once again with the separate parameters of pitch, timbre, loudness, etc. We want to keep the global idea in mind though, and we do this by making it easy to design shapes and forms and to project them onto any dimension of the music.

We need to consider carefully the relationship in the music between composition and performance. The dividing line is not to be fixed. Composition is the work before the music is played; performance takes place in the moment of "real" time. Composition requires the preparation of data, processes, or events for use in performance, and part of that preparation is making decisions about which aspects of the music are to be left to the moment. The programs MABEL can provide convenient ways to compose, and convenient ways to perform, and the requirements of these activities may be very different. One may, for example, write an algorithm in a high level computer language using a data terminal; this is a compositional activity. To attempt to do so in performance would court disaster, however, because the essence of performance is immediacy, an intuitive manner of decision-making, and a reduction of the possibilities for error. So performance requires ways of
controlling the instrument that can be physical and intuitive rather than cerebral and deliberate.

Special consideration needs to be given to pitch organization. Microtonality, of course, is taken for granted, and we must have very immediate ways to construct non-tampered pitch sets. Since my earliest computer music experiments, I have found it useful to control pitch bidimensionally. The first dimension is the actual pitch data, the second the index or pointer into that data. So that, for example, if a certain algorithm is selecting the pointers into a list, the meaning of the result will be totally changed by changing the list’s contents. Alternately, with the data remaining constant, a different process may be employed to order it sequentially. If, while the program is running, we can immediately and freely charge sets of pitch data as well as the sequencing algorithms, we will be able to effect "modulations" in the musical sense of pitch content independently of transformations of the continuity.

We need to maintain flexibility in the time domain as well. One of the essentials of performance is rubato. One of the greatest curses of some computer music is mechanical, or electronic, regularity. Even if we are allowed to escape rigid periodicity (or equally, non-rhythmic periodicity), time and pitch are often thought of as inextricably linked (the "note"). I would like independent (or interdependent) time streams operating simultaneously on different facets of the music and to be able to control the tempo independently of independent streams or sequences of events very freely in performance.

2. INTERACTION BETWEEN THE MUSICIAN AND THE PROGRAM

Human/machine communication takes place on a number of levels. The program is installed in a mixture of FORTH and assembly language, on the mechanical memory of the Buchla 400. The M600 becomes a CP/M system by attaching a terminal to the RS232 port and adding a disk controller and drives. When composing and performing, however, these apertures disappear, and all control is exercised through the Buchla’s touch-sensitive keys and analog voltage inputs. As with the MIDS and PATCH languages, developed by Buchla and Associates for the 100, the user is then presented with a main menu, the options of which may be read from the data key pad on the instrument. Non-volatile data storage is provided by a 3.5” microfloppy disk system.

At the level of composition, the user will select algorithms via a computer mouse, user-written functions, pitch sets, and sequences, for each of which an "event editor" then allows one to assign these composed elements to the controllable parameters of the hardware by means of a patching matrix, and then to store these matrices for future use. The final compositional act in the construction of a "Comp", or framework of conditions under which structural changes will occur. Examples of such changes are the assignment of different patching matrices to the output devices, composition of sequences using user-written algorithms, global changes of pitch set, and major timing changes. Examples of the conditions which may cause these changes are keypress, input voltages of a specific value or exceeding specific limits, or arriving at a particular stage of a sequence.

At performance time, while nothing prevents the musician from manipulating deeper levels of the program, the likelihood is that most of his time will be spent playing the keyboard, initiating the major compositional changes described above, or manipulating some level of detail in the music by making changes to an active patching matrix. Since the system is sensitive to analog input and the pressure needed to pre-programmed input can be assigned as part of the composition, it is also possible to control single performance from external voltage sources and triggers, including those derived from an instrument. By this way the MABEL system can be "played" by someone who has no formal knowledge of the software at all.

1. COMPOSING DATA

Editing Pitch Sets

Digital frequency data is sent to the oscillators of the M600 by indexing tables containing at least 5464 values. Thirty-two of these pitch tables may be simultaneously in memory, with other sets available on the microfloppy disk.

The pitch editing screen displays a "ruler" showing as a reference the division of an octave into equi-tempered semitones. Below this, space is provided for the user to enter interval sizes specified in cents. As each interval is entered, a mark appears on the screen, indicating an interval graphically. The user may
enter any number of intervals up to the maximum of 64. If a smaller number is entered, the program will repeat the given interval pattern until the available space is used up. It is also possible to specify the starting, or reference point, from which this series of intervals will be calculated.

**Editing Functions**

When the function editing screen is entered from the main menu, another ruler is displayed, this one displaying a range of times from 2 milliseconds to 64 seconds, arranged logarithmically. Using a joystick, the user draws an envelope shape against this time ruler. The drawing can be quite approximate, since it will in any case be quantized into discrete line segments.

When this approximate shape has been drawn, a keypress displays along it sixteen flashing, spots which will correspond to the end points of the line segments. Four keys enable these end points to be moved; two keys expand and contract them linearly, and two other keys expand and contract them logarithmically. Using these keys, the spots are positioned to make the best possible fit to the sketched envelope. At this point, another keypress displays the envelope in its line-segment form.

It is also possible to designate particular segments as beginning and ending ones within the function. Up to sixty-four of these envelopes may be stored in memory concurrently.

**Editing Waveshapes**

Waveshaping, and timbre modulation based on it, is a powerful timbral resource on the MU2400. The MU2400 wave shape editor resembles those of recent versions of MiniMoog and PPG, allowing the specification of waveshapes by percentage amplitudes of the first sixteen harmonics or odd or even upper harmonics.

These are drawn in bar-graph form using the joystick; the number of the harmonic and its amplitude are also displayed numerically. At any point a keypress results in the calculation and display of the timbre of the function corresponding to that particular harmonic distribution. Up to 24 waveshapes may be stored in memory at one time.

**Editing Sequences**

Sequences are of two types, representing indices into a table (usually, but not always, a pitch table), and timing values. The sequence editor permits these to be edited together, but their separate identity is recognized by the execution code. The idea here is that one may compose a number of sequences of indices and times and then freely combine any time sequence with any index sequence in performance.

The sequence editor requests the user to enter, from the numeric keypad, an initial value and duration. The value is a number in the range 0-64 which represents the value of the index into whatever table. The time value is given as a number of "time units," the particular value of each time unit being separately specifiable. Index and duration are represented graphically by a bar whose vertical position represents the index and whose length represents duration.

An initial value having been specified, subsequent steps in the sequence are defined by means of a positive or negative increment, so the shape of the sequence is indicated relatively rather than by specification of absolute values. As each increment is entered the display is updated and scrolled horizontally.

It is also possible to move the cursor backward and to copy and transpose a sequence that has already been composed. The result of this procedure is a "sequence in the musical sense."

We may not wish to trigger certain events on every stage of the sequence but only at specified times. To take a single example, if pitch and amplitude are triggered at every stage, we will have a series of "tones" of identical envelope. If, however, amplitude is triggered less often than pitch, the result will not be played as smoothly. To do this is provided by two levels of triggers, "events", and "hooks", which may be installed at any sequence stage and are represented by downward and upward pointing arrows attached to the horizontal graphic bars which represent the individual sequence events. These triggers also provide a way to synchronize different sequences running simultaneously.

Having composed a sequence in this manner, a master-time unit may be assigned for it and the index data and the time data may be stored in memory.
One other feature of the sequence editor needs to be explained. It is possible to generate sequences automatically by means of user-written algorithms. If the "automatic" mode is selected, the program will request the user to specify the number of the algorithm to be used in generating the sequence, and a new line will appear in the display showing the numeric values of up to six parameters which will be passed to that algorithm. These having been supplied, a keystroke will execute the numbered algorithm and a sequence will be composed automatically.

Sequences are currently limited to 256 events (binary Procrusteanism). The idea is though that when longer continuities are required the sequence will "recompose itself" when a certain specified stage is reached, so that given appropriate algorithms a kind of continually evolving line can be generated. The sequence acts in this case as a buffer. Further ideas about this process will be outlined later.

Editing events

The event editing screen is the central patching matrix from which waveshapes, functions, sequences, and other data can be assigned to the hardware.

The matrix is represented as a rectangle on the left side of which are displayed labels representing the various parameters of the machine: frequency (LFO), filter setting (PL7), wave shape (WSH) and so on. It should be noted that the Buchla 400 provides both hardware and software facilities to manipulate this function processor, which offers multi-dimensional control of timbre. There are, for example, two pitch control parameters and a delay line, and an offset used for bending or ornamenting the primary pitch. Likewise level and dynamics are differentiated; two sets of amplitude controls allowing larger and smaller loudness inflections among other possibilities.

Four vertical columns are indicated for each parameter. These are: Data (DTA), Scaling (SCA), Algorithm (ALG) and Timing (TIM). A cursor driven from the keypad allows different possibilities to be entered in each column. The available options being displayed by a submenu which changes as the cursor is moved.

The Data column indicates the source of the data which is to be applied to that particular parameter. In the case of pitch, one will specify a particular pitch set. For other parameters the number of a time-variant function will be provided. In the pitch example, it takes only a key-stroke or two to make a change on pitch content, enabling "modulations" to be executed very quickly.

The data may be scaled by including a slope in the circle.

The algorithm column defines the manner in which data is to be selected. The simplest algorithm is RBF which means that the elements of the active data set will be indexed by the number of the active touch-sensitive performance key. Alternately one may specify a particular sequence number as the index.

The last column indicates the times at which the data will be sent to the hardware. Again the simplest is RBK (keyboard), while the other main possibility is to execute events from the timings of a sequence, or perhaps from the triggers (marks and hooks) embedded in that sequence. Various combinations of possibilities yield interesting results. For example, data may be selected by a sequence while timings come from the keyboard, or vice versa. Another possibility is that different functions that comprise a given event will be timed by a number of sequences, so that filter envelopes might be initiated by one and dynamics or changes in modulator ratio by another.

Once an "event" has been constructed by assigning values to the data, scaling, algorithm and time fields for each of the parameters, the event as a whole may be stored in memory (up to 96 maximum) and assigned at will to any number of the six oscilators. In addition, further, this assignment may be either polyphonically (through the assigned voices) or data may be sent to all oscillators simultaneously.

Editing a composition

The available repertoire of "events" already represent a certain layer of compositional structuring, incorporating as they do sequences of data and complex functions. A higher level of visual control is represented by a "COMP" or composition editor, in which it is possible to construct conditional statements which will manipulate the lower levels of the program. These statements include
assign an event to the hardware: starting, stopping, initializing or composing a sequence (using any of the available algorithms); making global changes of pitch etc.; and changing the tempo of running sequences.

A number of different stimuli may initiate any of these changes: in particular, touching a performance key, the arrival of a positive or negative trigger at a pulse input, an input voltage crossing a certain threshold, or the arrival of a sequence at a certain state.

By means of this editor, complex "presets" may be programmed which will exert powerful structural controls on the music in performance.

1. OPTIMIZING PERFORMANCE AT THE MACHINE LEVEL

MABEL is written in Music-FORTH, a multi-tasking version of FORTH provided by Buchla and Associates, though the main loop and other time-sensitive execution code is written in 280 assembly language. The multi-tasker provides a way to instatiate different tasks in order of their execution priority. For example, the highest priority is given to execution of new segments of time-varying functions. The hardware function processor of the 8400 interrupts the CPU when it is idle for a new segment. Once this has been seen to the hardware, the multi-tasker relinquishes the CPU to the next lower priority function in the loop which polls the keyboard and updates screen information. The execution of compositional algorithms is a lower priority still, while at the bottom are the FORTH routines associated with the various data editors.

In this environment, editing data will not interfere with the actual performance of the music.

5. INCORPORATING COMPOSITIONAL ALGORITHMS

It is quite easy to install user-written compositional algorithms in MABEL. These are written on the extended 8400 development system either in FORTH or as assembly language subroutines. The only precautions are that the subroutine must return control to the algorithm-execution task, and any parameters it may require must be contained in a small array the address of which is known. When the algorithm has been tested, its execution address and the address of its parameter block are installed in the relevant arrays, and it is thereafter referred to by index into these arrays. One can then use the algorithms as part of the normal operation of MABEL by writing a statement in the COMP editor to execute the algorithm in when a certain condition, such as a keypress or input pulse, is met.

I have only begun to explore the possibilities of this aspect of the program. The first compositional algorithm to be written was a 1/f sequencer in which the number of steps in the sequence, starting point, and average excursion were passed as parameters.

A more sophisticated approach was used in my composition "Hexachords", for an instrumentalist interacting with the MABEL system. For this piece, the musician's sound was analyzed using a pitch and envelope follower connected to the Buchla's analog-to-digital converters. The performer also had a footswitch connected to a pulse input. When he pressed the switch, the analysis software writes a sequence that follows approximately the contour of his line, when he releases the switch an algorithm is executed which composes simple variations on what has been recorded. The variations are nothing more than transposition of the line or segments of it, but this is sufficient to provide a lively heterophonic environment in which the player can freely explore the material of the composition.

Other algorithms in progress include permutation of a sequence by sequentially exchanging values on each repetition, and a type of cumulative composition in which the "incipit" is gradually extended and elaborated.

6. EVALUATING THE WORKING PROGRAM

MABEL has been in usable form since January 1985. It has been used to create a work for instrumentalists and electronics, as a vehicle for improvisation with an ensemble, and to compose and produce the music for a multimedia performance work by the Canada Shadows, which was presented in a European tour in May and June of this year. The goal of creating a versatile and amenable performance system has been met. It has the criteria expressed at the beginning of this paper: it is unique and quite reliable. The system is the result of a personal approach, however, and it is a method which will not accommodate everyone comfortably. In particular it is clear that MABEL has a tonal bias; composers thinking vertically would find it wanting. Further, as the large task of coding the program...
Proceeded, concessions were made to programming convenience, especially accommodation of data structures to patterns amenable to the microprocessor in order to improve real-time execution speed. The large number of variables in the program allows programming meaningless combinations of parameters, and as yet the program is not entirely free of other quirks and idiosyncrasies with which the user must acquaint himself through practice. In this regard, however, MAREL is no worse than many systems of more reputable lineage.

MAREL is, in essence, a personal system. I have no objection to making it available to anyone else who might enjoy trying it, but it is not conceived for the marketplace. It emanates from a particular musical perspective and caters to a specific set of musical needs. I regard it as a step, and an important one for me, in the attempt to create genuinely musical "soft instruments", and to facilitate the realization of experimental musical ideas in composition and performance.

ACKNOWLEDGMENTS

Don Buchia and Lynx Crowe of Buchia and Associates provided the elegant instrument which inspired the whole effort, the music-FORTH language in which the program is written, and extensive assistance and advice. A research grant from the Integrated Media Division of the Canada Council provided funds to carry the project through. Neil van Wooy, programmer, was the able assistant who coded many iterations of the execution code as the program was dragged kicking and screaming from idea to reality.