A Workstation in Live Performance: Composed Improvisation *

XAVIER CHABOT
Computer Audio Research Laboratory
Center for Music Experiment, Q-037
University of California, San Diego
La Jolla, CA 92093
ucbvax!sdcarl!xc

ROGER DANNENBERG
Computer Science Dept., Carnegie Mellon University
Pittsburgh, PA 15213

GEORGES BLOCH
Music Dept., University of California, San Diego
La Jolla, CA 92093

ABSTRACT

The present work-in-progress on an instrument-computer-synthesizer system for live performance is a continuation of previously reported work by the authors at the International Computer Music Conference in 1984 and 1985. The lecture/demonstration is dedicated to the live presentation and performance of the piece *"Jimmy Durante Boulevard"* written by Georges Bloch. The piece features flute, trumpet and keyboard interfaced with MIDI, voice, a personal computer and a MIDI synthesizer and is based on the notion of composed improvisation which is defined in the paper. The programming environment for live performance in which the piece has been developed is presented. The piece features various types of control inputs. It has been our main concern throughout this year to study implications these controls have on the character of the music produced. A wide range of examples are covered from simple switch pedals to complex instrumental pattern data. Finally, the construction of the piece is detailed and partial examples as well as a version of the whole piece are performed live by the authors.

I. INTRODUCTION

This presentation is a continuation of previous work by Roger Dannenberg (Dannenberg 1984, Dannenberg 1985) at Carnegie Mellon University, Xavier Chabot and Lawrence Beauregard (Chabot 1984) at IRCAM and Xavier Chabot and Georges Bloch (Chabot 1985a) at CARL, presented at the International Computer Music Conference in 1984 and 1985. These works were based on performance with realtime computerized systems and were specifically designed to demonstrate interactions between instrumental playing and computer processes. The present work is mainly concerned with the relationship between *what controls* and *what is controlled*, an issue not included in these previous works.

1.1. Performance Practice and Technology

While relationships between instrumental (physical) gestures and musical (compositional) gestures have been traditionally implicit in instrumental practice and music notation, the computer allows the response algorithm to be explicitly specified. This independence between process and control is fascinating, but presents us with the responsibility for choosing a particular strategy. Classifications like the continuum described by Gareth Loy (Loy 1985) or the three basic modes: *organ, accompaniment* and *intervention* given by Xavier Chabot and Georges Bloch (Chabot 1985b) must be revised in terms of performance and composition practice.

Performance—or instrumental—practice involves tradition, experience, and sensitivity to context. *Tradition* is involved when one studies an instrument from somebody who already knows how to play it. *Experience* is acquired with the use of the instrument. *Sensitivity to context* characterizes live performance. Realtime technology has been developed to give to the machine a sensitivity to context. Two examples, the tape recorder and the pedal, will give us further insight into the notion of performance practice.

The tape recorder has been heavily used by the recording industry for rock-and-roll, music which is very equally successful in the area of broadcast, vinyl and live performance. Electronic tape music often reveals itself often as natural evolution of classical music practice: it is taught in conservatories and played in concert halls featuring traditional instrumental acoustic characteristics. An area of real success is the interdisciplinary art show (tape music with light, painting, dance, theater and above all motion picture). We think that the advantage of rock-and-roll is to be born with electro-acoustic technology and to have been developed with it. The term "context sensitivity" is enlarged to sensitivity to the characteristics of the era.

We use DX7 pedals in all sound examples. Pedals enlighten the close relationship between "Technology" and "Instrumental". This kind of pedal is an easy access to the computer through MIDI. This pedal exists because the computer exists. This Music makes use of pedals because this is computer music. Now, using a pedal implies a gesture. We call these motions of feet acting on pedals the *foot dancing*. Thus we discover that Music, when involving a computer, provokes foot dancing, quite unexpected for a flutist. Music using Technology (computer and pedal) has to take into account Technology: the use of pedals. Computer Music that we demonstrate must take into account foot dancing.

Practice of the computer is bringing out a new music, in this case foot dancing for a flutist. *"Foot dancing"* as abstraction does not exist, but comes from the instrumentality of the pedal. *"Computer Music"* as abstraction does not exist, but comes from the Technology—expression that we would like to be interchangeable with the substantive Instrumental— of the computer.

1.2. Improvisation

All examples presented have an improvisatory character. We make a distinction between improvisation by the performer and improvisation by the machine. The performer is improvising if he/she has no directives. He/she is then highly sensitive to the musical context as mentioned by George Lewis in (Roads 1985) and is able to react to what he hears from the machine, like in the piece *"Jimmy Durante Boulevard"*. A piece is primarily specified by its mode of functioning. Its form or structure is produced by its actual functioning during a performance and can be different each time (Chadabe 1984). Knowledge of the mode of functioning can be considered as a notation— the performer is required to follow the rules—but in most cases we prefer to view knowledge of rules as part of performance practice. The machine is usually said to improvise if its response has some unexpected components. We will say rather that the important thing is to create expectations and to fulfill them or not. But at this point the *inertia* in time of the Music seems more significant that the distinction notated/improvised. Inertia of music is similar to inertia in physics. Later examples will clarify our definition. The inertia is embodied in data files or in compositional processes.

Examples based on timbre control by pedals and amplitude feature no inertia, those producing "rhythm echo" to demonstrate Morse have some inertia and the excerpt of Globokar's *"Automusic"* (Globokar 1986) has a lot of inertia. The intertia in *"Jimmy Durante Boulevard"*

*Published as: Xavier Chabot, Roger B. Dannenberg, Georges Bloch, "A Workstation in Live Performance: Composed Improvisation," in *Proceedings of the 1986 International Computer Music Conference,* Den Haag, Netherlands, October 1986. San Francisco: International Computer Music Association, 1986. pp. 57-60.

has the form of *explosions*. Their evolution is defined by their starting state, but can be modified by performers.

1.3. Programming Environment

The programming environment (*i.e.* operating system and language) is of primary importance for the music produced because it dictates how functions are defined in time, how they are controlled and how they communicate with each other. The idea of inertia comes directly from the procedural character of the C language and the extensions that we use for parallelism and scheduling: Player (Loy 1985b) or Moxc (Dannenberg 1986, Collinge 1985). Furthermore, the use of this environment leads us naturally to conceive forms in terms of hierarchic structures, and controls—interrupt driven or not—in terms of top-down information paths, from general to specific, from *musical context* to *musical event*, from *collective* to *individual*. With this system, the composer/programmer composes a mode of functioning (the program) before the actual functioning (the data). This would be totally different with a function driven system like GROOVE (Mathews 1970), for example, where manipulation of "musical details" (function samples) does not require a predefined hierarchy.

Performers are thinking of musical context and musical event at the same level. The piece *"Jimmy Durante Boulevard"* features many types of controls: the computer is sensitive to specific events and musical contexts, and is able to respond with specific results as well as with context creation.

Let us summarize what we think of as essential to integrate control input into the musical decisions which participate to the composition: (1) Computer Music has to be thought of in terms of Technology and Performance Practice. (2) For the distinction between notation and improvisation, we prefer a characterization of computer music in terms of (a) inertia and sensitivity to realtime, as well as (b) control in terms of sensitivity to "musical context/musical event".

2. A REALTIME PERFORMANCE ENVIRONMENT

Figure 1 shows our set-up MIDI based configuration.

Inputs: Flute and trumpet players make use of the Yamaha DX7 pedals and have their pitch and amplitude converted in MIDI data by Pitchrider 4000 modules (from IVL Technologies, Canada). The keyboard player plays the DX7 keyboard and also has a pedal. The voice, not shown on the figure, is input directly into the audio system.

Outputs: The sound output is a stereo mix of voice, flute, trumpet and four Yamaha TX7 modules.

Computing: The computer is an IBM PC (or equivalent) interfaced with MIDI through three Roland MPU 401's (MIDI Processing Unit).

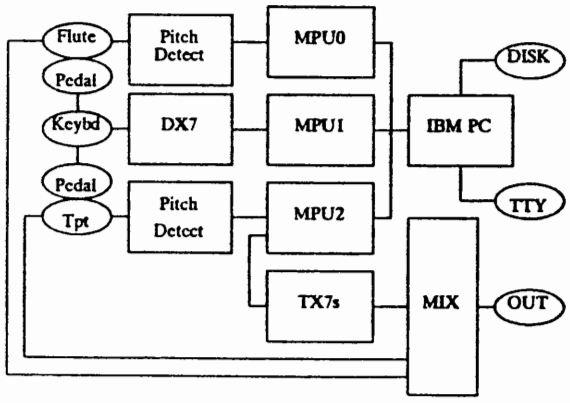

Figure 1 "Performance Setup"

2.1. Year 2001

The two main pieces of equipment, the personal computer and the keyboard/synthesizer are very representative of the musician's equipment of the year 2001. Their availability, versatility and handy packaging, make them definitively a product of our society from the 80's. The *practice* of this equipment by its negative and positive aspects reveals to us the *Technology* of the next generation of so-called the Computer Music Workstation. The future is in the continuance of the present. János Négyesy and Laurie Anderson playing the Mathews Electronic Violin and Morton Subtonick writing a concerto for two DX7's and orchestra (with Ursula Opens, Alan Feinberg and the Los Angeles Philarmonic for the 1988-89 season) are part of this continuity.

The IBM PC features a inadequate processor (the Intel 8088), but a nice development environment, enhanced and refined by a large number of utilities collected or written by Andrew Voelkel and Tom Erbe for CARL.

A very complete, versatile, programmable MPU driver has been written by Xavier Chabot (CARL) and Andrew Voelkel (FIPSE). This MPU environment is open. It is provided as a structured library of functions with source code. It is programmable (default actions can be replaced by users). It has been thought of as a C programming environment for MIDI. The manual and source code for IBM PC and single board M68000 are available from CARL. A version for the CARL/Harmony realtime operating system (Loy 1986) will appear soon. The CARL MPU driver's portability (dependence on compiler, processor and operating system) is embodied in the source code data base following the model of Harmony (Gentleman 1985). Interfaces have been built at CARL for Multibus and VME standard buses. The structure of this software can be used to structure other types of MIDI interfaces.

2.2. Schedulers

A lot of work has already been done on scheduling problems in realtime applications and such analysis will not be reported here. Our three actual candidates are Moxc, Player and Harmony. They can all be considered as supersets of the general purpose language C. Moxc is very small, easy to port, and basically implements a time tagged function call. Player is a little bit more sophisticated in that it gives primitives for process and parameter access. As mentioned above, they stimulate the composer/programmer to think in terms of "trees" of events, each level being more dense in time than the previous one. There is little or no provision for communication between events (message passing or synchronisation). We are looking forward to use Harmony on the CARL workstation prototype (Loy 1986). Harmony is not just a scheduler, but a whole realtime operating system, which has obvious advantages. It is based on message passing abstractions and we are curious to see what kinds of new applications this will stimulate. From the Language comes the Poetry. All following examples are using Moxc.

2.3. MIDI Control Input: Demonstration

In the following, content of our live demonstration on control input is detailed. Examples are ordered by increasing inertia. The first two are shown as playing modes, while the last two last are shown as part of a composition.

2.3.1. Examples 1 and 2: Timbre Control The first example demonstrates the action and accuracy of the pitch detector. A switch pedal controls the activation of the pitch detector. It is fundamental that any process can reveal its existence by being enabled and disabled. A continuous pedal (modulation pedal of the DX7) globally controls timbre changes. Timbres, numbered from 1 to 32, are distributed by categories: flute or organ sounds to timbre 1-10, keyboard to percussion sounds to 11-20, percussion and noises to 21-32. Another switch pedal increments by one the current timbre number, allowing variety within a given category of timbres. It is interesting to see the influence of the occurence of a particular timbre on the type of flute improvisation: each event "occurence of timbre" favors the creation of a particular musical context. For example improvisation with piano sounds keyed from the flute is very different from a keyboard player improvisation.

The Technology (pitch detector driving a synthesizer) reveals to the instrumentalist new modes of playing.

The second example is the same as above but with four TX7 modules. The number of modules activated is controlled by the amplitude coming from the pitch detector. The pedals affect sets of four timbres. The composition of sets is stored in files specified when calling the program. Here the combination of pitch, amplitude and pedal playing creates an improvisatory musical context. This example also shows the response time of the software (MPU servicing, parsing, echo to the MPU and the synthesizer): following one melodic line from the flute is no problem, while the same process on keyboard chords shows delays in the playback (arpeggiation of chords).

2.3.2. Example 3: Echos and Rhythms The use of Moxc to build rhythmed echos triggered by incoming pitches is shown in this example. Rythms are read from a file (like a set of timbres in the previous example) and indexed by switch pedals. The continuous pedal controls global tempo.

2.3.3. Example 4: Multitrack Sequencer A version of *"Automusique"*, piece number 8 of Globokar's *"Laboratorium"* is played. This version is somehow different from the original specifications. The performer depresses a pedal to enable time tagged pitch recording. When the pedal is lifted, cyclic playback starts. If the same track is retriggered, the new sequence is inserted. The other switch pedal increments the track number (maximum eight tracks) and allows layers of sequences. Each track has a different timbre. The continuous pedal controls the playback tempo. The performer creates a context in which he improvises. All tracks are written and read on disk in realtime. Technically, this shows realtime simultaneous MPU and disk servicing.

3. JIMMY DURANTE BOULEVARD

This piece written by Georges Bloch summarizes most features previously presented. It makes use of the CARL MPU software and the Moxc scheduler written by Roger Dannenberg and modified at CARL by Xavier Chabot.

3.1. Multiplication

The piece is based on the definition of a process able to *multiply* one material MAT1 by another MAT2 based on geometric series whose parameters are read from a data file. The multiplication process is notated (MAT1 * MAT2). In example 6a the trumpet will enter various MAT1 sequences, while MAT2 and the development parameters are fixed: *e.g.* a single note, a scale, a rhythmic pattern, or a complex sequence. In example 6b various developments are played while MAT1 and MAT2 are fixed. Development parameters are convergence/divergence, geometric ratio, proportions of silence and proportions of cuts into the trumpet phrase. Example 6c plays various developments entering MAT2 from the keyboard.

3.2. Chord Analysis

A chord analysis method by Julio Estrada is used. Any chord can be reduced and given a type number between 1 and 78. The type number gives a measurement of the dissonance and the evolution of the type number in a chord sequence gives measurement of the changes in harmonic context. Results of the analysis of any incoming chord affects the content of the MAT2 (example 7a), activity of subvoices of the development (example 7b) and parameter of the geometric series (example 7c). The chord analysis software has been defined by Georges Bloch and written by Roger Dannenberg.

3.3. Melodic Context Analysis

There are two simultaneous melodic context analyses: contour and concavity. They have been designed and written by Roger Dannenberg. Both processes are enabled/disabled by the flute pedal. The contour is defined as the integral in time with the pitch referenced to the first pitch entered when the flute pedal is depressed. The concavity is evaluated on a moving window of three pitches. Examples 8a and 8b show contour and flute activity (variation of the concavity) acting on the playback of development subvoices (activation and relative dynamics). The action of the flute is called *ornamentation*. See figure 2 below.

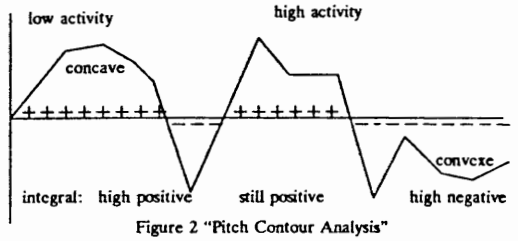

Figure 2 "Pitch Contour Analysis"

4. CONCLUSION

The last example is a version of the piece. Although there are many possible versions—performers are improvising— we strongly feel the uniqueness which makes every example belong to the same piece written by a composer: this is *COMPOSED IMPROVISATION*. Creation of the piece is shared between non realtime (composition) and realtime (improvisation). Each of the above examples stimulates a particular type of improvisation from each improviser. The only directive for the performers is to know what mode the program is in. The whole piece features complex relationships between musical context and musical event.

REFERENCES

CHABOT, XAVIER AND BEAUREGARD, LAWRENCE (1984). "The flute-4X project", *ICMC 84 Conference*, Paris.

CHABOT, XAVIER (1985a). "User Software for Realtime Input by a Musical Instrument", *ICMC 85 Proceedings*, Vancouver, B.C.

CHABOT, XAVIER (1985b). "Uses of an Instrument-Computer-Synthesizer System", *Music Department Seminar*, UCSD.

CHADABE, JOEL (1984). "Interactive Composing: An Overview", *Computer Music Journal*, 8(1), pp. 22-27, MIT Press.

COLLINGE, D.J (1985). "Moxie: A Language for Computer Music Performance", *ICMC 85 Proceedings*, Vancouver, B.C.

DANNENBERG, ROGER (1984). "An On-line Algorithm Realtime Accompaniment", *ICMC 84 Proceedings*, pp. 193-198, Paris.

DANNENBERG, ROGER AND BLOCH, JOSHUA J (1985). "Realtime Computer Accompaniment of Keyboard Performances" *ICMC 85 Proceedings*, pp. 279-289, Vancouver, B.C.

DANNENBERG, ROGER (1986). "The CMU MPU Toolkit", *ICMC 86 Proceedings*, The Hague.

GENTLEMAN, W.M. (1985). "Using the Harmony Operating System", ERB-996 (24685), National Research Council of Canada.

GLOBOKAR, VINKO (1975). "Individuum Collectivum", unpublished, Paris.

GLOBOKAR, VINKO (1986). "Laboratorium", unpublished, Paris.

LOY, D. GARETH AND WRIGHT, RUSTY (1985a). "An Operating Environment for a Realtime Performance Processing System" *ICMC 85 Proceedings*, Vancouver, B.C.

LOY, GARETH (1985b). "Player - Extension to the C programming language for Parallel Processing and Music Synthesis Control", unpublished, *Center for Music Experiment*, University of California.

LOY, GARETH (1986). "Designing a Computer Music Workstation from Musical Imperatives" *ICMC 86 Proceedings*, The Hague, Holland.

ROADS, CURTIS (1985). "Improvisation with George Lewis", in *Composers and the Computer*, Kaufman.

MATTHEWS, M. V., AND MOORE, F. R. (1970). "GROOVE—A Program to Compose, Store, and Edit Functions of Time", *Communications of the ACM*, (13)12, pp. 715-721.