Improvising with Computers: A Personal Survey (1989-2001)

Sergi Jordà
Music Technology Group, Audiovisual Institute, Pompeu Fabra University

Passeig de la Circumval·lació 8, 08003 Barcelona, Spain
email: sergi.jorda@iua.upf.es

Abstract

This paper exposes some topics and questions about music improvisation using computers that the author has considered during more than a decade of experience as a performer, composer, software designer and educator. After a quick review of several of his previous interactive music systems, some of those issues are then confronted in FMOL, a program initially designed as a tool for on-line collaborative composition that has been used by hundreds of on-line composers, and which is also being employed by the author in free-form audiovisual improvisation concerts.

1 Introduction

Improvisation is today present in almost every area of music, although it may have completely different meanings, behaviors and rules depending on the idiom it applies to (e.g. baroque, bebop, flamenco, blues, rock, etc.) and can even attempt at being non-idiomatic as in the so-called free improvisation (Bailey, 1992). Considering this infinite variety of approaches, further categorizations, generalizations and even definitions become really hard if not an impossible task; we will try to stick therefore to what could be the common denominator of all these practices, and consider improvisation as any form of *instant composition*, no matter how free or constrained this compositional process can result.

Computer-based interactive music systems go back to the late 1960s, initially involving computer-controlled analog synthesizers in concerts or installations. The use of real-time algorithmic composition spread in the 1970s with the work of composers and performers such as David Behrman, Joel Chadabe, Salvatore Martirano, Gordon Mumma or Laurie Spiegel (Mumma, 1975; Bernardini, 1986), but its greatest growth occurred during the mid 1980s with the MIDI standardization and, slightly later, with the advent of data-flow graphical programming languages such as MAX (Puckette, 1988; Puckette and Zicarelli, 1990). This made the design and implementation of custom interactive systems simpler than ever before (Winkler, 1998).

However, one and a half decades later, improvisation using computers still seems a burgeoning and under explored multidisciplinary area where the design of new controller interfaces, real-time sound synthesis and processing techniques, music theory, cognitive science, algorithmic composition techniques, and existing models of improvisation (computer and non-computer based) can converge, in order to bring new interactive music-making paradigms. But as computer music improvisation tries to define new models, it obviously cannot ignore the existing ones. Computer music improvisers have therefore in front of them a myriad of practices and know-how's. Several of them are as old and rich as the history of music making, while others are half-century-old and belong to the tradition of computer music.

The next part of this paper introduces three significant topics (others could have been included), discusses their current situation related to either instrumental or computer music, and suggests how fresh approaches can surpass traditional limitations and influence in the conception and implementation of new improvisation models.

2 In Search of New Computer Music Improvisation Paradigms: Three Topics to Think About

2.1 Controllers and Generators

Real-time interactive music needs musical input devices (i.e. controllers). The separation between gesture controllers and sound generators standardized by MIDI boosted the creation and development of many new alternative controllers with which to explore new creative possibilities (Chadabe, 1997). However, this separation should not always be seen as a virtue or an unavoidable technical imposition.

Before MIDI, the term *musical instrument* always referred to both sides of the sound and music creation process (i.e. controller and generator), and it is obvious that the final expressiveness and richness of any musical interface (controller) is not independent of the generator involved and the mapping applied between them. Some of the consequences that result from this fact are truisms (e.g. not

all combinations of controllers-generators are equally satisfying), but there are also some important questions that are raised: is it possible to develop highly sophisticated and efficient controllers without a prior knowledge of how the sound or music generators attached to them will work? We firmly believe that it is not possible and that a parallel design is necessary. Are today's standard music communication protocols (i.e. MIDI) wide and flexible enough, or are the potential dialogues between controllers and generators limited or narrowed by these existing protocols? Although MIDI has many limitations (low bandwidth, low resolution, half-duplex communications, etc.) (Moore, 1988) we optimistically believe that there still is room for experimentation and improvements while remaining compatible with MIDI's constraints.

Bi-directional mappings may be one of the solutions leading to wider and richer communications between controllers and generators and, as a result, between players and their music output. Duplex communications allow the existence of haptic and force manipulations and feedback, which may bring richer feedback loops between the instrument and the performer (Gillespie, 1999).

But force feedback and by extension, performer-instrument feedback, are not the only feedback loops present in traditional music playing: acoustical instruments, for example, resonate (i.e. they are "conscious" of the sound they produce) and this acoustic feedback that exists between the controller and the generator subsystems, is in fact fundamental in the overall sound production process. While audio analysis, which could be extended to the concepts of music analysis or *machine-listening*, is a regular topic in many computer-music disciplines, this task is not usually undertaken by the controller subsystem, meaning that too often, controllers do not "know" the output they are generating.

By searching for smarter mappings and wider communications between the different components of an interactive music system, we are not implying that low-cost and widely available input devices such as mice, joysticks or computer keyboards have to be considered as dead-ends. On the contrary, there is still a lot of research to be done in the area of interactive GUIs. We will see, how in the case of the mouse-driven FMOL instrument, the controller audio feedback is presented to the performer in a visual form. This intuitively helps the understanding and the mastery of the interface, enabling the simultaneous control of a high number of parameters that could not be possible without this visual feedback.

2.2 Macrostructural vs. Microstructural Level Control

For centuries, Western music has talked about *notes*, almost ignoring what was inside of them. Many instrumental and vocal improvisation idioms maintain, however, a good balance between *sound* and *notes* -or form- (consider for instance the growls of free-jazz saxophone players or the melismas of flamenco singers, both essential to their respective improvisational styles).

While it is true that since its beginning one half century ago, computer music was primarily concerned with sound and gave a relevance to the notes' *inside*, the prevalent use of the two complementary concepts *score* and *orchestra*, common to most Music N languages, did certainly not encourage the merger of these two macrostructural and microstructural musical levels.

With the advent of MIDI, almost two decades ago, computer assisted music creation (as opposed to computer assisted sound creation) was able to become a real-time activity. MIDI does indeed allow some sound parameter modifications by means of controller messages in degrees that vary with each synthesizer model, but its paradigm is completely note-based. The panorama may have started changing in recent years, when the increasing power of personal computers has allowed the definitive bloom of real-time software synthesizers.

Even though virtual synthesizers do not really represent any new concept, they have important differences with respect to their hardware counterparts, as they have the potentiality to achieve what no manufacturer could ever think of, giving more room for freedom, experimentation and imagination, and bringing forth new sonic and control paradigms. Sadly, control over most of these software tools still sticks to the idea of notes which can have an internal evolution, thus keeping the tacit computer-music distinction between *instruments* (that control sound) and *composing tools* (that control form). But nowadays, the fact is that the separation of both worlds, when applied to computer music creation, may be becoming a burden and an anachronism.

In this context, several questions arise: would it be possible to conceive of performing systems (controllers + generators) where the control of both sound and form could be done by means of the same tools and gestures? And wouldn't that bring a completely new dimension to real-time computer music creation?

This aim is by no means new. Since the second half of the last century, many composers such as Iannis Xenakis (Xenakis, 1992), Giacinto Scelsi, James Tenney or Curtis Roads (Roads, 2001a), to name only a few, have been concerned with the integration of these two macrostructural and microstructural musical levels. Xenakis and Roads have even designed or conceived software tools, such as the UPIC (Marino, Raczinski and Serra, 1990) or the PulsarGenerator (Roads, 2001b) that allow to capture both microstructural detail and macrostructural evolution. However, being composers but not necessarily improvisers, they have not concentrated their attention in the peculiar control necessities raised by *instant composition;* they therefore tend to obviate the *controller* side, even when their systems offer real-time synthesis and control capabilities. This shortage can also be seen in many powerful "micro-macro-friendly" software composition environments such as

James McCartney's SuperCollider (McCartney, 1996) or Carla Scaletti's KYMA (Scaletti, 1989). The simpler Music Mouse interactive software, developed by Laurie Spiegel in the mid 1980s, could indeed be considered both an instrument and a composing tool (Gagne, 1993), but the technology available fifteen years ago was not mature enough for this kind of mixed approach.

This *micro-macro* dichotomy is therefore still implicitly present in the approach of most improvisers and interactive systems designers nowadays. Composition-oriented musicians tend to the macrostructural level and others favor more the performance or instrument-oriented microstructural one, while a third group may try to find an uncertain equilibrium between both conceptions. My background as an amateur free-jazz saxophonist in the early 1980s can surely be one of the reasons for which I search within this no man's land. And this is also one of the fundamental aspects FMOL tries to explore.

2.3 Individual vs. Collective and Dilettante vs. Professional Performers

Creating music in real-time using computers and new interfaces poses also new questions about whom will be using them and how they will be used.

While performing music has typically been a collective event, traditional musical instruments have been mostly designed for an individual use (even if some of them, as the piano or the drum kit can be easily used collectively). This restriction can now be freely avoided when designing new interfaces, which leads to a new generation of distributed instruments, with a plethora of possible different models following this paradigm (statistical control, equally-allowed or role-playing performers, etc.). Implementations of musical computer networks date back to the late 1970s with performances by groups like the League of Automatic Music Composers (Bischoff, Gold and Horton, 1978) or the Hub (Perkins, 1999). This may also be the common case of many interactive sound installations which respond to the public movements and gestures, which leads us to another important point: that of the skills and the know-how of the performer(s).

I have developed several computer-based interactive music systems since 1989. Some of them were conceived for trained musicians or even for specific performers, while others, like Epizoo, were to be controlled by members of an audience in public performances (Jordà, 1996; Lozano-Hemmer, 1996). The demands for the two genres are obviously different. Complicated tools, which offer great freedom, can be built for the first group, while the second group demands simple but appealing tools that -while giving their users the feeling of control and interaction- must produce "satisfactory" outputs. These two classes are often mutually exclusive. Musicians become easily bored with the "popular" tool, while the casual user may get lost with the sophisticated one. But is this trend compulsory? Isn't it possible to design interfaces that can appeal to both sectors: tools that would not dishearten hobbyist musicians, but that would still be able to produce completely different musics, allowing a rich and intricate control and offering various stages of training and different learning curves? Tools that, like many traditional acoustical instruments, could offer a *low entry fee with no ceiling on virtuosity* (Wessel, Right, 2001).

There is the common belief that more hard-to-play instruments lead to richer and more sophisticated musics (e.g. the piano vs. the kazoo). But expressiveness does not necessarily imply difficulty, and in that sense, one of the obvious research trends in real-time musical instruments design can be the creation of easy-to-use and, at the same time, sophisticated and expressive systems. Considering that the best way to understand and appreciate any discipline, whether artistic or not, and music is not an exception, is by *doing* and being part of it, more efficient instruments that can subvert the previous effort-result statement will bring new sophisticated possibilities and the joy of real-time active music creation to non-trained musicians. Let's try, as Robert Rowe suggests, *to develop virtual instruments and/or musicians that do not just play back music for people, but become increasingly adept at making new and engaging music with people, at all levels of technical proficiency* (Rowe, 1992).

3 Three Previous Works by the Author

3.1 PITEL

Influenced by the works and writings of George Lewis (Roads, 1985) and Robert Rowe (Rowe, 1992), PITEL (1989-1992) is the first interactive music system I developed (Jordà, 1991). It is a software environment for polyphonic real-time composition and improvisation, which employs some simple mathematical machine-listening techniques. It can generate, up to four monophonic MIDI voices under the control of a *mouse-conductor*, while listening and reacting to one or two external MIDI players.

PITEL performances were usually jam sessions that involved three musicians: a saxophone and a trumpet player both fitted with pitch-to-MIDI converters, and me conducting the program. The aesthetical model was closer to free music than to the jazz tradition (no scales, modes or predefined rhythms were used). The instrument remained at the note and form level, without attempting any of the sonic microstructural control suggested in 2.2.

I wanted PITEL to be as flexible as possible and not oriented to any special musical aesthetic, so I decided to build a system with no musical knowledge at all (except the fact that *one octave has 12 MIDI notes*), as I felt that introducing additional musical ideas into the system, would have favoured particular musical styles.

The program was based on two-term feedback relations of the form $X_{a,i} = F(X_{a,i-N}, X_{b,i-k})$ (where X is a musical parameter as pitch, amplitude or duration, a and b indicate two different musical voices, and N and k determine the memory of the system); the output of any voice could therefore be influenced by its own past values but also by the past values of another voice.

Figure 1. PITEL listening matrix. In this configuration, all four internal voices are listening to voice #2 (although using different algorithms), creating a slightly homophonic effect. Voices A and B correspond to the human players.

This was a simple method for interrelating voices or creating *listening patterns* between different voices. The matrix displayed in figure 1, allowed to configure in real-time to what voice (including the two human ones represented by *A* and *B*) was each one of the four PITEL voices listening to, and which listening algorithm (*F*) it was using. By changing the elements of the matrix many different listening patterns could be defined, from homophonic hierarchies (as shown in figure 1) to circular closed dependencies (e.g. 1 listens to 2, 2 listens to 3, 3 listens to 4 and 4 listens to 1). Several listening algorithms *F* were also implemented, from the simplest average function $X_{a,i} = 1/2(X_{a,i-N} + X_{b,i-k})$ -reminiscent of the Karplus-Strong algorithm for plucked-string sounds (Karplus and Strong, 1983)- to sophisticated non-linear functions which led to more complex, unpredictable and interesting behaviours.

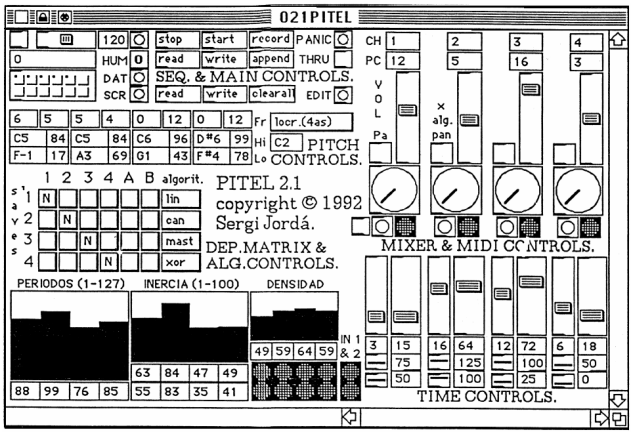

Figure 2. PITEL Main Screen.

PITEL's simple approach produced remarkably coherent music between the four internal voices. The results were usually not so satisfactory when the internal voices were listening to the human players, as they were not sharing the same listening logic (or *algorithm*). Two of the other criticisms that could be applied to the system were its big inertia (i.e. fast changes were hard to obtain) and, as shown in figure 2, the shortage of a serious interface design and conception: all high-level compositional parameters were adjusted by simple means of virtual buttons and sliders (which we will see, is not necessarily the case of all mouse-driven GUI).

3.2 The QWERTYCaster

While PITEL algorithmic and inertial approach follows perfectly the *composition tool* paradigm, the *low-tech QWERTYCaster* (1996) is a clear example of the *instrument* model. It is a very simple guitar-shaped controller that I designed and used for free improvisation. The aim was to build a fast reactive system using affordable and cheap technology, with direct audio output in response to every movement or gesture, not unlike Nicolas Collins *Trombone* (Collins, 1991) or Michael Waisvisz *Hands* (Krefeld, 1990).

Figure 3. Sergi Jordà with the *low-tech QWERTYCaster* (1996).

It consists of a standard QWERTY computer keyboard (*strings*), a trackball (*frets*) and a joystick pad (*lever*), all held together in a guitar-like piece of wood, connected to the computer with a five-meter cable which allows the performer to play the "rock guitar hero" role. Its five continuous controllers (two degrees of freedom for the trackball and three for the joystick), together with the joystick triggers, buttons and keys, steered an old 486 computer with a sampler soundcard and a simple but

effective custom MIDI software. At the opposite side of PITEL, the QWERTYCaster only focused in notes and sound control, leaving all the macro-formal organization for the human improviser. In spite (or maybe because) of its simplicity, I still use it sometimes in improvisation concerts, especially in duos.

3.3 Afasia

Afasia (1998), my third collaboration with the visual artist and performer Marcel.lí Antúnez (Lozano-Hemmer, 1996), is a one-man-show multimedia play inspired by Homer's Odyssey, which has been performed in many countries in Europe and America and has received international awards. In Afasia, the performer-conductor (Antúnez), fitted with a plethora of radio-emitting sensors, plays the one-man-orchestra controlling with his movements a whole mechanical music robot quartet (consisting on a *seventy-fingers* electric guitar+bass controlled with electro-valves and compressed air, a walking drum kit, a one-string electric violin excited by an E-Bow and a three-bagpipe "horn section"), a sampler, an audio CD player, three audio effects racks, a Yamaha Promix MIDI-controlled audio mixer, interactive multimedia animations, a DVD, a DMX light table and the video projector input-switcher (for changes between SVGA multimedia and DVD video).

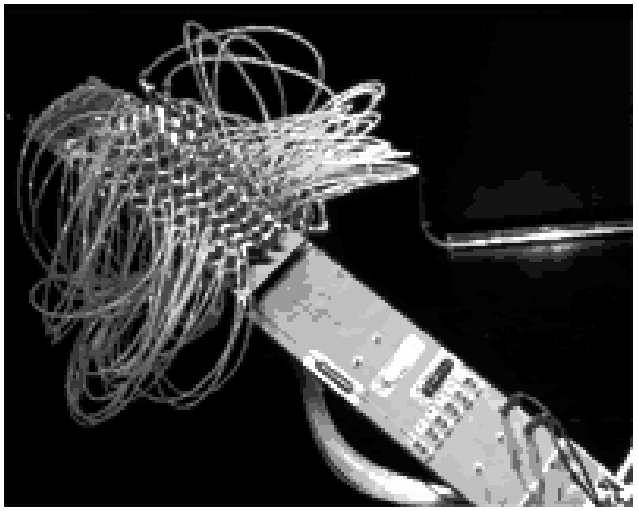

Figure 4. Fragment of the guitar+bass robot, designed and constructed by Roland Olbeter (1998). Each flexible tube is controlled by one electro-valve and transmits compressed air into one of the "fingers", tapping the string at a given fret.

All of the sonic elements of the system are interactively MIDI controlled. Each one of the four robots has its own virtual MIDI driver which bridges musical MIDI messages generated by the interactive software, with the digital output cards that control the relays and the pneumatic mechanisms. The music mechanically played by the electric guitar+bass and the violin robots (the two remaining robots, the drum kit and the three bagpipes are not amplified), is mixed together with the music coming from an internal sampler card and with occasional prerecorded audio fragments triggered from an internal CD audio player, into the Yamaha mixer. This output is eventually processed through three multieffects processing units, which are also MIDI controlled by the performer.

The complexity of this setup as opposed to the limited semantics of the sensors employed (gloves, buttons, potentiometers in each of the performer's articulation and mercury switches in his extremities) makes Afasia a peculiar example of a score-driven interactive model, in which every *island* (taken from Ulysses' journey) behaves as an independent interactive environment, which allows the performer to interact with different preset mappings and several restricted degrees of freedom.

Figure 5. Marcel.lí Antúnez controlling the three-bagpipe robot in Afasia (1998).

3.4 Afasia Interactive MIDI Files

Afasia's interactive music capabilities can be seen as an extension of the QWERTYCaster experience. The aforementioned setup (sensors + robots + MIDI gear) led me to implement its musical interactivity (images and video are treated differently) by means of format 1 Standard MIDI Files, which are expanded with the inclusion of text meta-events and reproduced through a custom interactive sequencer.

Text events are often used by sequencers to name parts of the tracks or to display synchronized lyrics. In the Afasia software they are used as commands that tell the custom sequencer player how to deal with, and process, the recorded MIDI events, according to the performer's

gestures. This approach enables to write, record and program interactive Afasia music using any standard sequencer software, although the interactivity is only available when the file is played in the custom Afasia sequencer player (when Afasia Standard MIDI files are reproduced on a conventional sequencer no interactivity is attained, as sequencers simply skip text meta-events). Here is a brief description of the Afasia MIDI files structure:

- Each part (or *island*) of the show is associated with one Standard MIDI file.
- Each MIDI file is made of any number of ***blocks*** (an Afasia concept not present in standard MIDI files).
- A ***block*** is a group of sequencer's tracks that behaves like a permutable section of a score, only one block being active at any time.
- Each block is made of a special ***control track*** and any number of conventional MIDI tracks.
- ***Control tracks*** only contain text meta-events that indicate how to interact with the other block's tracks, alter the block structure or jump to other blocks.
- Conventional tracks can also contain text meta-events that indicate how to interact or modify the MIDI data inside the track.
- As conventional sequencer tracks, each track (except for control tracks) is directed to a specific port or device, and to a specific channel within that device.
- Six MIDI ports are used in Afasia (a custom port for each of the four robots, the standard internal MIDI port for controlling the soundcard sampler, and the standard external MIDI port for controlling the audio mixer and the three effects processors).
- Each device can have several MIDI channels (e.g. the electric guitar+bass robot has one channel for each of its strings, while the bagpipe section MIDI port uses three channels, one for each bagpipe).

The text meta-events embedded in the sequencer tracks, specify interactive commands and do always have the following structure:

```
COMMAND_NAME=param1,param2,...,paramN$
```

The number of parameters (*N*) and their meanings varies with each command type. The first parameter usually corresponds to an input sensor number while the second indicates which track of the block the command applies to. Following parameters define how these incoming sensor values have to be mapped.

For example, the command `TRANSPOSE=4,3,12$` indicates that the third track of the block will be transposed according to the incoming values of sensor #4, in a maximum range of +-12 semitones. Many commands (TRANSPOSE is one of them) have an *end* version command (e.g. ENDTRANSPOSE) that disables the effect.

More than forty commands are implemented in Afasia. They allow to dynamically switch between blocks, mute/unmute tracks, transpose the tracks, loop any sequence and modify their lengths, modify the current play position, generate or modify any MIDI controller (either setting directly the new value, its increment or the range of random variation), quantize tracks, delay tracks, control tempo, compress/expand temporal events, trigger audio CD tracks or define chords or scales to which new generated notes will be matched.

Several additional macro-commands called *solo modes* can also be activated, each one defining a peculiar way of playing (i.e. entering MIDI note messages into the system). While the previous commands use simple one-to-one mappings, *solo modes* apply more sophisticated gesture detection mechanisms, that allow for example to *play* the robot-guitar in an almost conventional manner, controlling pitch with the performer's left elbow angle, while triggering notes with different right-arm fast movements that enable the performer to play chords, arpeggios or monophonic lines. Twelve solo modes (three for each robot) have been defined.

Besides the interactive music, the system also allows the performer to control multimedia sprite animations or access a DVD video by means of complementary text script files. All this turns Afasia into a powerful and flexible interactive system, which permits anything, from free audiovisual improvisation to completely pre-recorded sequence playing. Although I have not performed with the system myself, I have composed all the interactive pieces or environments for the show, to be performed by Marcel.lí Antúnez. The system has proved to be a flexible framework for interactive music composition and performance. The model seems also especially suitable for interactive soundtracks in videogames or other multimedia environments.

4 FMOL (F@ust Music On-Line)

The FMOL project (1997-2001) started when the Catalan theatre group La Fura dels Baus, proposed me to develop an Internet-based music composition system that could allow cybercomposers to participate in the creation of the music for their next show, F@ust 3.0, freely inspired by Goethe's work. Initially I did not have a clear idea of what I wanted; I just knew what I did not want: *to allow people to compose tempered music on the keyboard and send us attached MIDI files via E-mail*. Besides, although I felt that the project should have a fairly "popular" approach, and I did not want to be too demanding and restrictive about the participants' gear, I was not looking for a dull General MIDI sound, but for richer sounds and textures that could introduce newcomers into more experimental electronic music (Jordà, 1999). Mouse-driven software for real-time interaction and synthesis seemed therefore the natural solution, which led to

the development of a C++ stand-alone program with http capabilities (Jordà and Wüst, 2001).

4.1 FMOL's Sound Engine and Interface Design

The conception of Bamboo, FMOL's main graphical mouse-controlled interface, was very tight with the synthesis engine architecture design. Both were in fact developed in parallel with the primary aims of conceiving a real-time composition and synthesis system, appealing to both trained electronic composers and more casual or hobbyist musicians, suitable for the Internet (small scorefiles, etc.), that could run on a standard computer fitted with any conventional multimedia soundcard.

FMOL's sound engine supports eight stereo real-time synthesized audio tracks. Each track uses an independent audio buffer and includes a sound generator (sine, square, karplus-strong, sample player, etc.) and three serial processors (filters, reverbs, delays, resonators, frequency or ring modulators, etc.), which can be chosen by each composer between more than 100 different synthesis algorithms or variations.

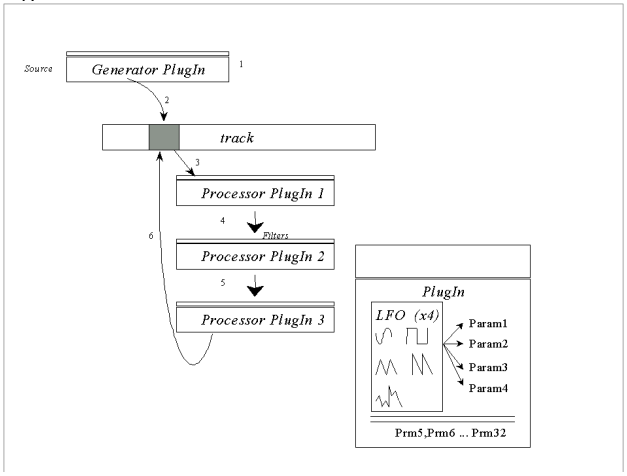

Figure 6. Scheme showing the generators, processors and oscillators structure of one of the eight FMOL audio channels or tracks. The grey square represents the frame being computed at one time, and the numbers (1 to 6) show the order in which the internal steps are carried.

Each generator or processor allows for eight control parameters, four of which can be modulated by four independent low frequency oscillator (LFOs), in which frequency, range and shape (sinusoidal, square, triangular, saw tooth or random) can also be interactively modified for each. The important point is that the control of this complicated architecture is directly reflected in an intuitive, symbolic, dynamic and non-technical way, in Bamboo's graphical interface (Jordà and Aguilar, 1998).

The Bamboo screen, presents a lattice in which vertical lines are associated with the synthesis generators and horizontal lines with the synthesis processors. Like a virtual guitar, these vertical lines/strings can be plucked or fretted with the mouse while they continuously draw the sound they generate like a multichannel oscilloscope. Horizontal segments that control the synthesizer's serial processors can be dragged and oscillate up and down, creating an abstract dance that tightly reflects the temporal activity and intensity of the piece.

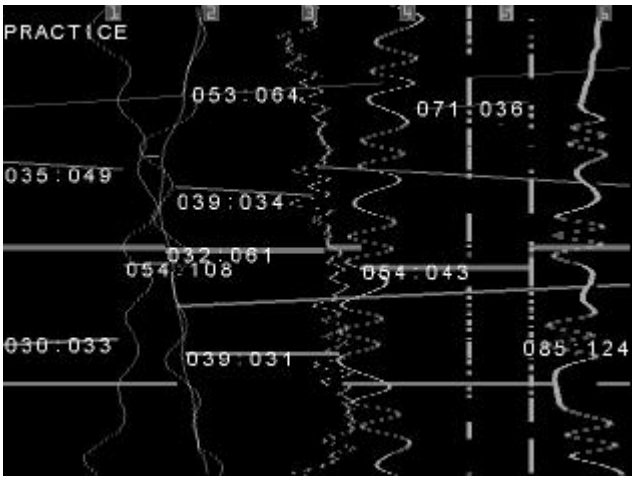

Figure 7. FMOL's Bamboo in full action

A complete Bamboo user's manual cannot be included but I will just mention that the combination of both mouse buttons and the computer keyboard allows for an intricate control, including sustaining sounds, modifying any parameter, recording mouse gestures loops, applying LFOs (with frequency, amplitude and wave-shape control), creating arpeggios or recording and restoring sequences and screen configuration snapshots. Although it may sound quite complex, the abstract visual feedback of all the instrument activity, which turns converts FMOL into an audiovisual synthesizer, has proved to be an invaluable help for users.

4.2 Low Frequency Oscillators and Time Control in FMOL

The FMOL software can act as a sequencer and as a player, and this facility is indeed fundamental for submitting, retrieving and playing compositions (scorefiles) via the web. However, for conceptual and esthetical reasons, record and playback capabilities were somehow restricted in order to avoid the triggering of pre-recorded sequences while improvising. Instead of that, users can dynamically record and retrieve all their mouse movements, as many

times as they want with the help of two keyboard keys, but this material vanishes when s/he leaves the session. Other possibilities include the storage and retrieval of snapshots that hold the state of all the parameters at a given time, and all these snapshots may be indeed stored with the scorefile. Although snapshots are instantaneous (they could be called presets using a synthesizer terminology), they do include the information for the twenty-four active LFOs (frequency, amplitude and wave shape), and the combination and interaction of all these LFOs are in fact responsible for much of the time control and the dynamic evolution of FMOL pieces. In that sense, FMOL is an attempt of integrating real-time composition with simultaneous microstructural and macrostructural control, into one single interface.

4.3 Musical and Social Implications

Unlike any other system I have designed, FMOL has been used by hundreds of Internet composers. From January to April 1998, the FMOL first Internet-database received more than 1,100 brief pieces by around 100 composers, some of whom connected nightly and spent several hours a week creating music. One of our main goals (i.e. to conceive a musical system which could be attractive to both trained and untrained electronic musicians) was fully attained. We know now that several of the participants had no prior contact with experimental electronic music and a few were even composing or playing for the first time, but all of them took it, however, as a rather serious game, and the final quality level of the contributions was impressive. After a difficult selection process (only 50 short pieces could be chosen and included on the show's soundtrack), and considering that a great number of interesting compositions had to be left aside, we decided some months later to produce a collective CD with a mixture of old and new compositions (Jordà, La Fura dels Baus et al., 1998).

A new web with a new version of the software has been back on-line during September 2000 for La Fura´s new show, the opera DQ, premiered in October 2000 at the Gran Teatre del Liceu in Barcelona. During one month, more than 600 compositions have been submitted, and the selected ones constitute now the electroacoustic parts of an otherwise orchestral score.

In September 2000, a one-week workshop specially aimed at visual artists with no prior musical knowledge, took place in Lisbon. The workshop concluded with several conducted collective improvisations, with astonishing results.

4.4 The FMOL Trio

Although FMOL was originally designed as a freely available system for on-line collaborative composition and "experimental electronic music proselytism", it also turned to be my favorite instrument for live concerts. Since 1999, the FMOL Trio (Cristina Casanova and myself on FMOL computers, plus Pelayo Arrizabalaga on saxophones/bass clarinet and turntables) performs free-form improvised electronic music, while two projectors connected to each of the computers give the complementary visual feedback (FMOL Trio, 2000). This setup, which enables the audience to watch the music and how it is being constructed, has proven to be a valuable addition to the concerts, giving the public a deeper understanding of the ongoing musical processes and adding new exciting elements to the show.

Figure 8. The FMOL Trio in performance. From left to right: Sergi Jordà (FMOL computer), Cristina Casanova (FMOL computer) and Pelayo F. Arrizabalaga (alto sax) (2000).

4.5 Collective Composition and On-line Real - Time Jamming

The use of the Internet for collective creation and the production of open and continuously evolving works was also one of the important keys behind the initial FMOL project. Versions 1 (1998) and 2 (2000) of the FMOL architecture do not implement any structures for real-time jamming. In order for several authors to collaborate on a same piece, it is required that they store their respective contributions on the FMOL database server before any other author can continue working on them. Each participant is allowed to start new compositions from scratch as well as overdubbing, modulating or processing any of the pieces stored in the database, which are accessible through a web browser.

The collaborative paradigm is based on a vertical-multitrack model (as opposed to a *horizontal-exquisite corpses* model which would allow the pasting of sonic fragments one after the other). FMOL scorefiles are stored on a relational database. The main entity is the compositions table, which has a recursive relationship to itself, allowing for a tree like representation of the pieces, as shown in figure 9. Each piece is a node storing a scorefile that holds the data for the eight real-time synthesized audio tracks, which can be played by the FMOL audio engine. A user can pick up any existing composition, listen to it, work on it and save a new version in the database. This new version will be automatically stored as a child node of the one the user picked. This permits new composers to modify and enlarge already existing pieces an endless number of times, while keeping at the same time the integrity of the original pieces. Besides, FMOL's synthesizer architecture not only allows to add new sound layers to previous compositions, but also to apply further processing to any of the previous tracks, modulating or distorting what other composers did, in unpredictable manners. That way, a musical idea brought by

one composer can grow and evolve in many different directions unexpected by its original creator.

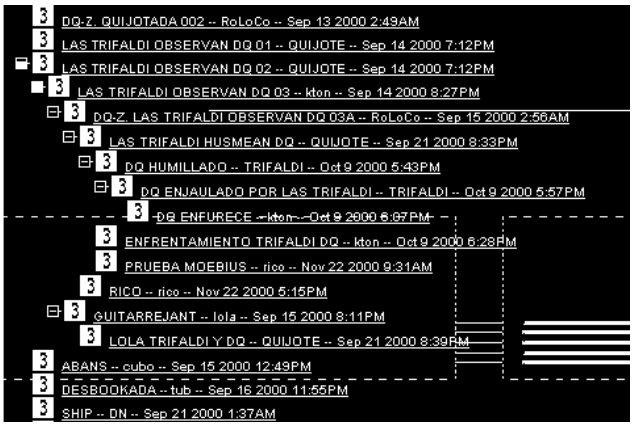

Figure 9. Screenshot showing a fragment of the compositions tree as seen from the web browser.

The newer version, currently under development will finally support real-time jamming. The problem of performing a jam session over the Internet has some constraints imposed by the current network technologies. The most relevant problem is related to the high network latencies. An event played on a computer placed on one end of an Internet connection will arrive with a perceptible delay to workstations on other ends of the net. This is actually unacceptable for playing most musical styles. Nevertheless, the type of music that is produced with FMOL is more timbrical than rhythmical and can therefore better tolerate timing inaccuracies, in a similar way that Gregorian chant could deal with the several seconds reverberation times produced by cathedrals.

In October 2001 several freely improvised duets have taken place between Barcelona and Dresden and Barcelona and Berlin, using a peer-to-peer prototype of the system. Events played on each computer were locally monitored and synthesized in exact real-time (so that each player could not feel the latency), and sent to the other computer which resynthesized them locally. As a consequence of this mechanism, every participant listened to a slightly different version of the collaborative piece, but latencies were always below 100 ms using a standard 56K connection and the system successfully proved its playability.

The full version which should be available during 2002, will include a real-time messaging server, which will be accessible through a web-based interface and will provide services such as an FMOL session manager; active jam-sessions will be monitored, and users will be able to create new sessions or join any of the currently open ones, if the maximum number of participants per session has not been reached (Jordà, Wüst, 2001).

5 Conclusion and Future Work

In parallel with the new Internet version, we are also developing a special concert version, which will include sensor-gloves, and will substitute the mouse with video detection, in order to allow the performers to play directly with their hands over a 3x2 meters retro-projected FMOL screen.

After years designing and improvising with interactive music systems, my favorite one has become the one I did not explicitly design for myself, but for everyone. For the last four years, FMOL has become my main research and production area, and further developments of the FMOL instrument have taken different courses: as a model for Internet collective music creation, as a free software musical instrument and as a professional system specially oriented to our own performances. FMOL is not the answer to all my questions concerning computer music improvisation but it has proved to be a valuable environment for experimenting with these topics as well as for raising new questions.

References

Bailey, D. (1992). *Improvisation: Its Nature and Practice in Music*. Da Capo Press.

Bernardini, N. (1986). "Live electronics." In R. Doati and A. Vidolin, eds. *Nuova Atlantide*. Venice: la Biennale di Venezia. pp. 61-77.

Bischoff, J., Gold, R., and Horton, J. (1978). "Music for an interactive Network of Computers". *Computer Music Journal, Vol. 2, No.3*, pp 24-29.

Chadabe, J. (1997). *Electric Sound: The Past and Promise of Electronic Music*, Prentice Hall.

Collins, N. (1991). "Low Brass: Trombone-Propelled Electronics." *Leonardo Music Journal, Vol. 1, No. 1.*

FMOL Trio (2000). Live at Metronom (Compact Disc Hazard CD10), Hazard Records.

Gagne, C. (1993). "Interview with Laurie Spiegel". In *Soundpieces 2: Interviews With American Composers*. The Scarecrow Press, Inc. N.J., pp. 295-333.

Gillespie, B. (1999). "Haptic Manipulation". In P.Cook, eds. 1999. *Music Cognition and Computerized Sound: An Introduction to Psychoacoustics*. Cambridge, MA: MIT Press, pp. 247-260.

Jordà, S. (1991). "A Real-Time MIDI Composer and Interactive Improviser by Means of Feedback Systems." *Proceedings of the 1991 International Computer Music Conference*. International Computer Music Association, pp. 463-466.

Jordà, S. (1996). "EPIZOO: Cruelty and Gratuitousness in Real Virtuality." *Cyberconf proceedings.*

Jordà, S., La Fura dels Baus et al. (1998). *F@UST 3.0 – FMOL* (Compact Disc), Fundación Autor.

Jordà, S., Aguilar, T. (1998). "A graphical and net oriented approach to interactive sonic composition and real-time synthesis for low cost computer systems." *Digital Audio Effects Workshop Proceedings*, pp. 207-210.

Jordà, S. (1999). "Faust music On Line: An Approach to Real-Time Collective Composition on the Internet." *Leonardo Music Journal, Vol. 9*, pp. 5-12.

Jordà, S., Wüst, O. (2001). "Architectural Overview of a System for Collaborative Music Composition over the Web." *Proceedings of the 2001 International Computer Music Conference*. International Computer Music Association.

Karplus, K., Strong, A. (1983). "Digital synthesis of plucked strings and drum timbres." *Computer Music Journal, Vol. 7, No. 7*, pp. 43-55.

Krefeld, V. (1990). "The Hand in The Web: An Interview with Michel Waisvisz.", *Computer Music Journal, Vol. 14, No. 2.*

Lozano-Hemmer, R. 1996. "Perverting Technological Correctness". *Leonardo, Vol. 29, No. 1.*

Marino, G., Raczinski, J.M. and Serra, M.H., (1990). "The new UPIC system". *Proceedings of the 1990 International Computer Music Conference*. International Computer Music Association.

McCartney, J. (1996). "SuperCollider: a new real time synthesis language." *Proceedings of the 1996 International Computer Music Conference*. International Computer Music Association.

Moore, F. R. (1988). "The Disfunctions of MIDI." *Computer Music Journal, Vol. 12, No. 1, pp. 19-28.*

Mumma, G. (1975). "Live-electronic music." In J.Appleton and R.Perera, eds. 1975. *The Development of Electronic Music*. Englewood Cliffs: Prentice Hall. pp. 286-335.

Perkins, T. (1999). "The Hub". *Electronic Musician Magazine, August 1999*.

Puckette, M. (1988). "The Patcher." *Proceedings of the 1988 International Computer Music Conference*. International Computer Music Association, pp. 420-429.

Puckette, M., Zicarelli, D. (1990). *MAX – An Interactive Graphical Programming Environment*. Menlo Park: Opcode Systems.

Roads, C. (1985). Improvisation with George Lewis. In *Composers and the Computer*, ed. C. Roads. Los Altos, Calif., William Kaufmann, Inc.

Roads, C. (2001a). *Microsound*, The MIT Press, Mas.

Roads, C. (2001b). "Sound Composition with Pulsars", Journal of the Audio Engineering Society, Vol. 49, No.3.

Rowe, R. (1992). *Interactive Music Systems – Machine Listening and Composing*, The MIT Press, Mas. p. 263.

Scaletti, C. (1989). "The kyma/platypus computer music workstation". *Computer Music Journal*, Vol. 13, No. 2. pp. 23-38.

Wessel, D., Wright, M. (2001). "Problems and Prospects for Intimate Musical Control of Computers." *Proceedings of the New Instruments for Musical Expression Workshop*. CHI, Seattle.

Winkler, T. (1998). *Composing Interactive Music: Techniques and Ideas Using MAX*, The MIT Press, Mas.

Xenakis, I. (1992). *Formalized Music*. Revised edition. New York: Pendragon Press.

On-line Additional Information

Audio and video excerpts for this paper:
http://www.iua.upf.es/~sergi/download/chi2001/

FMOL main page: http://www.iua.upf.es/~sergi/FMOL

FMOL-DQ Internet composition project:
http://teatredigital.fib.upc.es/dq

FMOL Trio: http://www.iua.upf.es/~sergi/FMOL/fmoltrio/