FMOL: A graphical and net oriented approach to interactive sonic composition and real-time synthesis for low cost computer systems

Sergi Jordà (1)
Toni Aguilar
(1) Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
sergi@iua.upf.es http://www.iua.upf.es/~sergi

Abstract

Faust Music On Line (FMOL), is a software project for real-time synthesis and interactive and collective music composition through the net, conceived by Sergi Jordà and developed by the former and Toni Aguilar, after a proposal by the Catalan theatre and performance group, *La Fura dels Baus*. This work has been sponsored by the S.G.A.E., the Spanish authors' association, and has received the first price in at the 3rd international software competition of *the Institut international de Musique Electroacoustique of Bourges*. Although the nature of the project deals with many different topics as can be a collective and net-oriented approach to composition, this paper will concentrate on the two aspects more related with this conference: the implementation of a complete real-time synthesis software on low cost computer systems, and the design of an intuitive graphical interface for interactive sonic composition.

1 Introduction

During the spring of '97, *La Fura dels Baus* was beginning to work on its new project, *Faust v3.0*, a free revision of Goethe's work (it has been premiered in Barcelona, on April 1998 and presented in NYC on July 1998). They proposed us to collaborate, conceiving and developing a tool that would allow musicians from across the world to participate through the Internet, on the elaboration of the play's soundtrack.

They did not have a clear idea of what they wanted, but we did know what we did not want. We did not want people to compose temperate music on the keyboard and sending us attached MIDI files in e-mails. We did not want a dull General MIDI sound, but a full, great, strange, sometimes noisy electronic sound. Moreover, we wanted, however, not to be too demanding and restrictive about the participants' gear, in order to bring the project to a wider audience, and not only to fully geared electronic musicians.

An "electronic and weird" sound with the minimal possible gear-hardware (i.e. a multimedia soundcard) means *real time synthesis.* After discarding for speed reasons the Java applet solution, we choose the platform dependent C program approach, and due to the short developing time we had, we had to focus on one only platform, betraying Internet's platform-independent nature.

The result, FMOL 1.0, is a standalone program written for W95, which runs on a Pentium 100 or higher provided with any 16 bit (and DirectX compatible) multimedia sound card. Once downloaded and installed it automatically manages all the Internet connections with the database server, with whom it interchanges small scorefiles, not audio. All development was done on *Microsoft Visual C++ 5.0*, and makes extensive use of *Microsoft DirectX* libraries (*Direct Sound* and *Direct Draw*).

2 The synth architecture

2.1 Hardware and Performance

When designing and programming the engine, performance and creative possibilities were more important than top quality algorithms. The main goal was to build a basic sound generation kernel that could be flexible enough for real time manipulation, and appealing and enriching for different users, possibly including many "MIDI minded", that were not acquainted of software synthesis.

Current FMOL synth engine supports eight stereo audio channels (real time synthesised at 16 bit and 22,050 Hz). Each channel is made of a generator (sine, square, sample player, etc.) and three serial processors (filters, reverbs, resonators, etc.), to be chosen by each composer between more than a hundred different synthesis methods, algorithms or variations. The synth also offers a non-real time render to WAV option, which converts its real-time recorded scorefiles to 22,050 Hz 16-bit stereo wave files. Rendering is done at 32-bits and the output files are normalized.

2.2 The audio Kernel

Each of these eight buffers (one for each channel) is two second long, and is partially written 25 times per second. That means that every 1/25 second, a fragment of 1,764 samples (22,050 samples/sec chan / 25 frames / sec * 2 chan) is computed and written down to each buffer.

An additional mix buffer of similar length and structure receives the sum of these eight buffers. The mix process is first carried with 32-bit integers, although the results are finally stored in the mix buffer, again with 16-bit resolution.

Finally, the mix buffer sends the resulting block to a primary buffer of only 2/25 of second, which writes the result onto the card's memory.

This architecture provides a small 1/25-second latency and delay, while keeping a memory of 2 seconds on each track (for delays, feedback algorithms, etc.). *Figure 1* shows this structure.

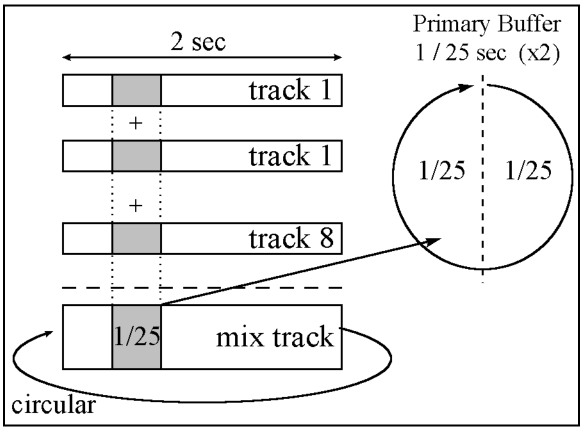

Figure 1. Structure of the internal buffers. This writing process is repeated 25 times/second.

2.3 Generators, processors, modulators

Each track is made of a generator and up to three serial processors. Moreover, for each track (except for track 1 and the final mix track) the called *generator* can in fact behave as a *parallel processor*. A processor of this kind, takes its input from the output of any of the lower buffers (i.e. channel 5 can be configured to process channel 1, 2, 3 or 4). The main difference between a parallel and a serial processor is that the former has a buffer of its own, while serial processors READ from and WRITE to the same buffer.
Each generator or processor can be configured with a maximum of 32 parameters, four of which can be modulated by four independent LFOs with frequencies ranging between 0.1 and 12.5 Hz. The type of each oscillator or LFO can also be dynamically configured (sinusoidal, square, triangular, saw tooth or random). 128 LFOs (4 LFOs/plugin * 4 plugins/track * 8 tracks) can therefore be active simultaneously. *Figure 2* illustrates this structure.

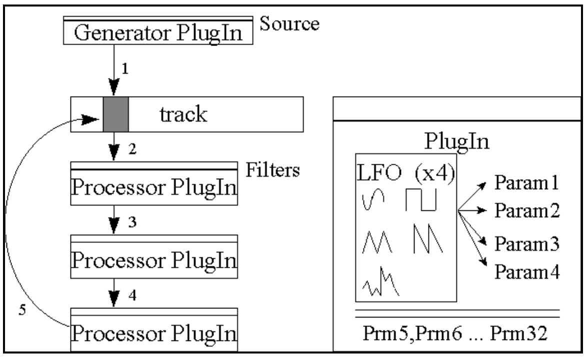

Figure 2. Each track is made of a generator three serial processors, and each can be controlled by four low frequency oscillators.

Algorithm	Type	Param. 1	Param. 2
BASIC OSCILLATORS			
Sine Wave	Generator	MS Pitch	Amplitude
Square Wave	Generator	MS Pitch	Amplitude
Sawtooth Wave	Generator	MS Pitch	Amplitude
Triangular Wave	Generator	MS Pitch	Amplitude
Pulse Train	Generator	MS Pitch	Amplitude
White Noise	Generator	NO	Amplitude
SAMPLE PLAYERS			
Sample Player	Generator	MS Pitch	Amplitude
Waveta.Samp.Pl	Generator	Wav.Sound #	Amplitude
Scratch Samp.Pl.	Generator	Frequency	Amplitude
BASIC MODULATORS			
Ring Modulation	Processor	MS Pitch	Modul %
Amplitute Modul.	Processor	MS Pitch	Modul %
FEEDBACK ALGORITHMS			
Karplus Strong	Generator	MS Pitch	Amplitude
Binary Modul.	Generator	Primary Pitch	Amplitude
K.Str. Processor	Processor	MS Pitch	NO
B.Mod. Processor	Processor	Primary Pitch	NO
LINEAR DIFFERENCE EQUATION FILTERS			
High-Low Filter	Processor	Cutoff Freq.	NO
Resonant Filter	Processor	Res. Freq.	Q
Comb Reverb	Processor	Feedback	Delay
Comb Eco	Processor	Feedback	Delay
Comb Delay	Processor	NO	Delay
Comb Comb	Processor	Gain	Delay
OTHERS			
Pitch Shift	Processor	Pitch Shift	NO
Panning	Processor	Angle	NO
Line In	Generator	NO	NO

Table 1. FMOL algorithms and their first two primary parameters.

3 The Algorithms

FMOL v1.0 has about twenty different synthesis or processing algorithms. As mentioned earlier, these algorithms were selected primary for its computing speed (they all operate, for instance, on the time domain), but together they constitute a heterodox and flexible palette. Moreover, due to their many configuration parameters, these 20 basic algorithms are used to build more than 100 patches or presets.

A complete list is included in *Table 1,* showing algorithms grouped in five different families:

- basic oscillators
- sample players
- modulation techniques
- feedback algorithms
- linear difference equation filters
- other heterogeneous processing techniques

4 The collective "vertical" composition approach

4.1 The multitrack sequencer

The FMOL Synth has been designed not only for real-time synthesis, but also for real-time composition and control. That means that scorefiles are generated on the fly, while composers interact with the interface and sound is being produced. That involves the existence of a sequencer, which operates at a 25-frames/second rate. Besides, FMOL is conceived for collective "vertical" composition, which means that new composers may add new tracks to one composition, but the duration of one composition is the same for all its composers.

These considerations lead us to a model where new composers are not only allowed to add new sound layers to existing compositions, but are also able to apply further processes to any of the existing tracks (until their three serial processors are used).

4.2 The compositions' tree database

To handle this model, FMOL scorefiles are organised on the server database, not as a simple list, but as a tree. Each time the program accesses this database, it receives and updates the compositions' tree structure, allowing the user to see all the compositions' dependencies, with the particular information of each node (author and time and date of creation). The user is then able, not only to upload his brand new compositions, but also to download and hear existing ones, and, if he wants, to enrich/modify/distort them. That way, a musical idea brought by one composer can grow and evolve in different and possibly orthogonal directions. *Figure 3* shows a fragment of this tree, taken from a screenshot of the program's Internet window.

5 The graphical Interface

With all these considerations in account, the great challenge was to find a coherent and appealing interface that could exploit both the synthesis and the collective compositional paradigms.

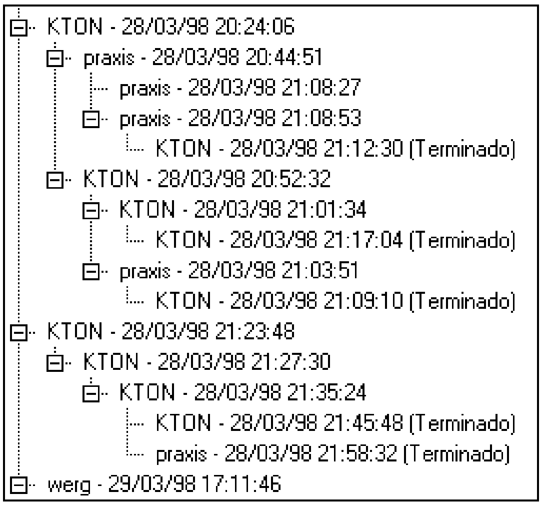

Figure 3. A screenshot from the program, showing one and a half-hour musical chat between two users (KTON and praxis, composing the 1998-03-28, between 08:24 PM and 9:58 PM)

5.1 "Musicians" vs. "Curious" or casual creators

The authors have been working on virtual instruments and interactive music systems for ten years. Some of them, like PITEL were thought for trained musicians, while others like EPIZOO, had to be controlled by members of an audience, in public performances. The demands for both genres are obviously different. For the first group, complicated tools, which offer great freedom can be built. The second group demands for simple but appealing tools which, although giving its users the feeling of control and interaction, should not produce "too bad" outputs. And both classes are normally exclusive. That is, the *musician* gets quickly and easily bored with the "popular" tool, while the casual user gets lost with the sophisticated one.

With these previous experiences in mind, for this project we tried to conceive a graphical interface, which could be attractive to both sectors. A tool that would not look frustrating to non-musicians, but would still be able to produce completely different music ("bad" and "better"), allowing a rich and intricate control, and offering various steps of training and learning curve.

5.2 Faust Dilemma

On the Fura’s play, many FMOL short pieces are planned to sound one after another by way of a leitmotif whenever the leading character, Faust, distraught and tormented, feels in the depths of his spirit the fierce battle taking place between the chaotic forces of life and the efforts of his intelligence to understand and dominate them. Therefore, in order to illustrate Faust dilemma, *la Fura* wanted two completely different graphic control interfaces, one more cerebral and another wilder.

6 Bamboo

We will concentrate here on one of the two FMOL's interfaces (the other being *Medusa*). *Bamboo* came to the rescue when the project was on its half, and the synth engine almost finished, and we now consider it as one of FMOL's big acknowledges, as it has proved able to exploit intuitively and in real time, with nothing more than a single mouse, most of the synth possibilities.

Bamboo represents the rational side. Though not simple, its control can be fully mastered. Its aspect is of a rectangular web, where the horizontal lines correspond to the generators, and the vertical lines to the processors. It behaves like a guitar or a harp, as its strings can be plucked or fretted with the mouse. It also behaves like a multichannel oscilloscope, as every string continuously draws the sound it is generating. *Figure 4,* is a screenshot from *Bamboo* ".

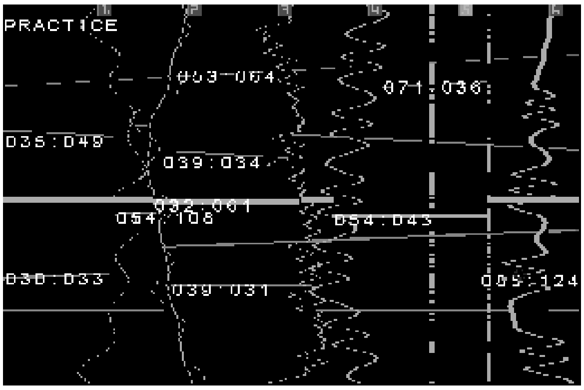

Figure 4. Bamboo "in action".

Moreover, this graphic interface is not only an input instrument, as it also follows the previous tracks' information. What a composer hears and sees, is therefore a combination of his actions and the previous composers' ones.

7 Conclusion

Although FMOL synth does not try to compete with fully programmable synthesis engines and environments, either soft or hard based, we believe it is considerably powerful and customizable, compared to typical closed architectures. In addition, its eight voices work well on a Pentium 100 fitted with a cheap SoundBlaster!

FMOL was on line two months, before Faust's première in Barcelona this April (**http://www.sgae.es/fmol**). During this period, more than 1,200 brief compositions by around 100 authors have been received, and 50 of them have been selected for the first representations, although the group plans to keep the soundtrack changing.

We know now, that many of these composers had no previous experience in "working with sound and noise" (as opposed to "working with notes and scales"). Some of them were even composing for the first time. Nevertheless, they took their job seriously; as proved by the late and long hours they periodically connected to the server. Moreover, many have told us, they now listen to *sound* in a different manner and with a new sensibility.

It is a fact that Internet’s immense creative potential is often discarded by crowds too used to instantaneous pleasure, who navigate through the web as if it was an infinite zapping TV. Projects like FMOL try to change these deficiencies, trying also to approach electronic music and sonic composition to a potential wider audience.

8 Acknowledgments

The authors thank *la Fura dels Baus* for the germinal proposal, and the S.G.A.E. for its support. Thanks also to Curtis Roads for his excellent *The Computer Music Tutorial* [1], which has been a permanent source of inspiration during the all project.

9 Internet Resources

1. Download: <http://www.sgae.es/fmol>
2. Jordà, Sergi " FMOL - Faust Music On Line - Project Description." <http://www.iua.upf.es/~sergi/FMOL/Project%20Description.htm >
3. Musicals Software Competition of Bourges 1998. <http://www.gmeb.fr/Sommaire-dir/index2.html>

References

[1] Roads, C. 1996. “The Computer Music Tutorial”. The MIT Press, MA.