The IRCAM Musical Workstation: A Prototyping and Production Tool for Real-Time Computer Music

Cort Lippe, Zack Settel, Miller Puckette and Eric Lindemann

IRCAM, 31 Rue St. Merri, 75004 Paris, France

During the past three years IRCAM has developed the IRCAM Musical Workstation (IMW). Recently we have begun work on musical tools and the realization of music with the system. Miller Puckette has written a version of MAX for the IMW which includes signal processing objects, as well as the standard objects found in the Macintosh version of MAX (including MIDI objects). The MAX environment for the IMW offers composers and researchers a powerful real-time prototyping and production tool for signal processing, synthesis, and the detection of continuous control parameters from musical instruments.

The IRCAM Musical Workstation (IMW) [1] is built around a dual Intel i860 card developed at IRCAM [2], and commercialized by ARIEL Corporation, which plugs into a NeXT machine cube. The IMW attempts to break with the paradigm wherein a DSP is controlled by a separate host computer. The i860 is a general-purpose RISC processor, in other words, general enough and yet fast enough to do control and synthesis at the same time.

MAX

MAX [3], as developed by Miller Puckette, extended by David Zicarelli, and commercialized by Opcode, is a graphical programming environment for the Macintosh with MIDI handles. MAX now runs on the IMW with an additional library of signal processing objects. So far, Puckette has written about 30 signal processing objects, including objects for filtering, sampling, pitch tracking, delay lines, FFTs, etc. MAX runs in the FTS [4] real-time monitor program under the CPOS operating system [5].

With the IMW version of MAX, the flexibility with which one can create control patches in the original Macintosh version of MAX is carried over into the domain of signal processing. Creating an oscillator object and a DAC object, hooking them together, and controlling the oscillator's frequency and amplitude via a MIDI keyboard in real-time becomes an easy task. (Figure 1.)

Prototyping Environment

In musical applications the ability to test and develop ideas interactively plays an important role. Because of its single architecture the IMW is a powerful prototyping and production environment for musical composition. Prototyping in a computer music environment often combines musical elements which traditionally have fallen into the categories of "orchestra" (sound generation) or "score" (control of sound generators). Mainly due to computational limitations, real-time computer music environments traditionally have placed hardware boundaries between "orchestra" and "score": the sound generation is done on one machine while the control is done remotely from another. The 4X machine at IRCAM and the MIDI studio are classic examples of this dichotomy. When developing a synthesis algorithm which makes extensive use of real-time control, it is extremely helpful, if not essential to develop the synthesis algorithm and the control software together. This is greatly facilitated when both the sound generation and control run on the same machine and in the same environment. Using MAX on the IMW, both synthesis and control can be developed and fine-tuned simultaneously in real time on the same machine with straightforward bi-directional communication between the two domains (without, for example, having to format communications protocol for a hardware link such as MIDI or RS232). (Figure 2.) A synthesis algorithm and its control algorithm may be grouped and saved together, forming a single complex object which can be utilized in other configurations (patches), ultimately allowing the user to blur the boundaries of the traditional computer music orchestra/score paradigm.

Production Environment

Musical production has just begun on the IMW. Outside of several sketches (Lippe and Settel) our early experiences have consisted of porting compositions made at IRCAM using the 4X to the IMW. This has meant porting the 4X micro-code, the C code from the 68000 host machine which controls the 4X, and finally MAX control programs which run on a Macintosh and communicate with the 4X host via MIDI. Reduction of all this code from a network of three machines to a single machine and a single environment on the IMW has been quite successful. For example, one module that was coded across the three machines which allows for independence of pitch and time in playback of sampled sounds was reduced to a single rather simple MAX patch upon being transferred to the IMW.

A typical 4X piece making use of a variety of modules for signal processing and sound synthesis might contain the following: harmonizers, delay lines, frequency shifters, samplers (with recording and playback in real-time), one or two synthesis algorithms such as additive synthesis or *Phase Aligned Formant Synthesis* (developed by Puckette) , filtering, reverberation, spatialisation, and possibly an FFT (for the analysis of incoming signals in order to extract control information for additive synthesis). Finally, a crossbar enabling all signals to be sent from any module to any other is normally included to maximize the number of possible signal paths. A configuration of this complexity for the composition *Pluton* [6] by Philippe Manoury (a 50 minute piano/4X piece) has been ported to the IMW.

Current Developments

Real-time signal analysis of instruments for the extraction of musical parameters gives composers useful information about what an instrument is doing. One of the signal processing objects recently developed in MAX on the IMW is a pitch tracking algorithm. In a clarinet sketch produced by Lippe, the pitch detection algorithm analyzes the incoming clarinet signal and outputs MIDI-style pitches which are sent to a score follower using the Explode object [7] which in turn triggers and controls the electronic events in the score. The pitch follower also outputs continuous control information analogous to pitch-bend in the MIDI domain. (All of this is done without the use of a MIDI cable, and thus without the bandwidth limits of MIDI.) This information coupled with envelope following can act as a secondary aid to score following. Certain kinds of musical material are very difficult to recognize with pitch following alone: thus one can rely on other analyses, depending on the context, to augment pitch-oriented score following, or follow other aspects of an instrument's signal when pitch is irrelevant or difficult to detect.

A subject of great interest at IRCAM presently is real-time audio signal analysis of acoustic instruments for the extraction of musical parameters. Musical parameters which can be derived from continuous tracking include a rich variety of elements. In the frequency domain, using pitch tracking, one can determine the stability of pitch on a continuous basis for the extraction of pitch-bend, portamento, glissando, trill, tremolo, etc. In the amplitude domain, envelope following of the continuous dynamic envelope for articulation detection enables one to determine flutter-tongue, staccato, legato, sforzando, crescendo, etc. In the spectral domain, FFT, pitch tracking, and filtering can be used to track continuous changes in the spectral content of sounds allowing for detection of multiphonics, inharmonic/harmonic ratios, timbral brightness, etc. High-level event detection combining the above mentioned analyses of frequency, amplitude, and spectral domains can provide rich control signals that reflect subtle changes found in the input signal. These control signals carry musically expressive information which can be used to drive signal processing and sound generating modules, ultimately providing an instrumentalist with a high degree of expressive control over an electronic score. In a clarinet sketch by Settel, high level event detection plays an important role in the soloist's control interface to signal processing and synthesis modules. (Figure 3.)

The dynamic relationship between performer and musical material, as expressed in the musical interpretation, can become an important aspect of the man/machine interface for the composer and performer (as well as the listener) in an environment where musical expression is used to control an electronic score. The richness of compositional information useful to the composer is obvious in this domain, but other important aspects exist: compositions can be fine-tuned to individual performing characteristics of different musicians, intimacy between performer and machine can become a factor, and performers can readily sense consequences of their performance and their musical interpretation.

Future Developments

Future developments in MAX on the IMW include the implementation of a complex vocoder (Figure 4.), real-time direct-to-disk soundfile recording and playback, as well as interfaces developed in ANIMAL [8], an interface prototyping environment designed to control MAX patches. An ANIMAL interface is in the planning stages which will offer a toolbox for high-level manipulation of continuous control parameters and allow composers to fine-tune signal analysis of instruments dependent on their acoustical properties and the musical context.

References

[1] Eric Lindemann *et al*, "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features", Proceedings, ICMC 1990 (Glasgow).

[2] Cort Lippe, "Musical Performance Using the IRCAM Workstation", Proceedings, ICMC 1991 (Montreal).

[3] Miller Puckette, "The Patcher", Proceedings, ICMC 1988 (Cologne).

[4] Miller Puckette, "A Real-time Monitor for Multiprocessor Music Synthesis", To appear in the Computer Music Journal 15(3)

[5] Eric Viara, "A Real-time Operating System for Computer Music", Proceedings, ICMC 1990 (Glasgow).

[6] Cort Lippe, "A Technical Description of *Pluton* by Philippe Manoury" IRCAM Annual Report 1988, IRCAM, Paris

[7] Miller Puckette, "EXPLODE: A User Interface for Sequencing and Score Following", Proceedings, ICMC 1990 (Glasgow).

[8] Eric Lindemann, "ANIMAL: A Rapid Prototyping Environment for Computer Music Systems", Proceedings, ICMC 1990 (Glasgow).

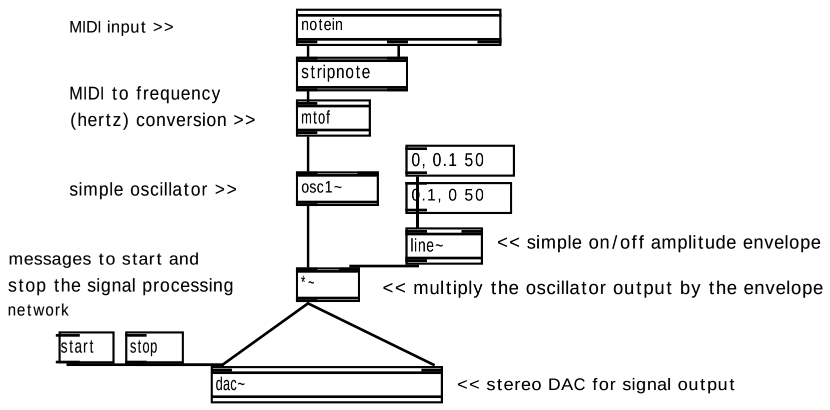


figure 1.

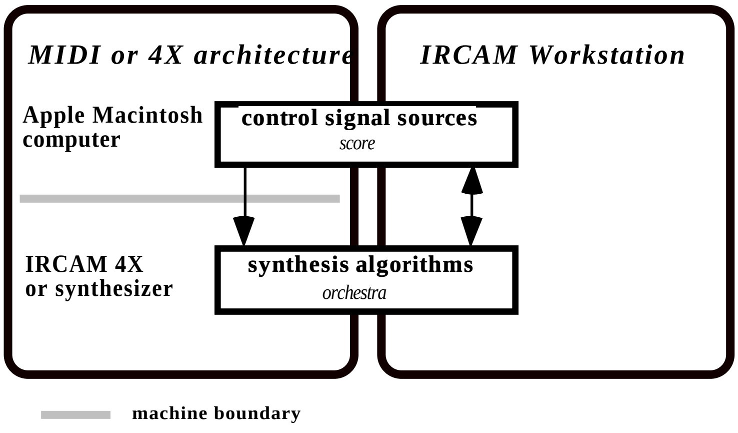


figure 2.

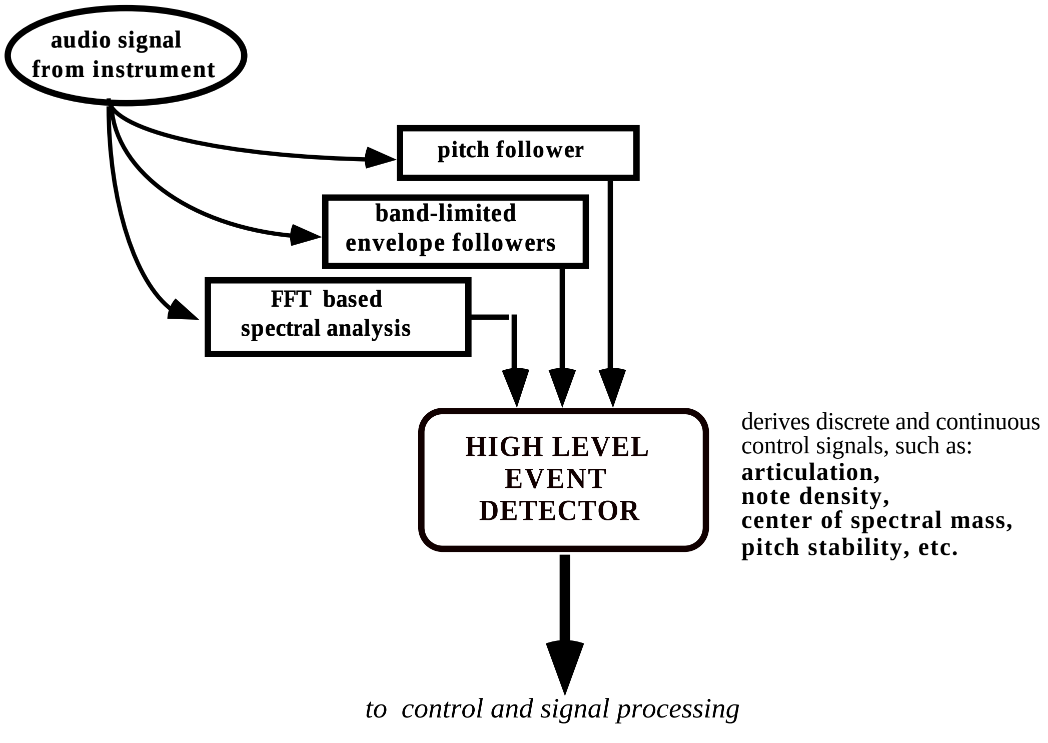
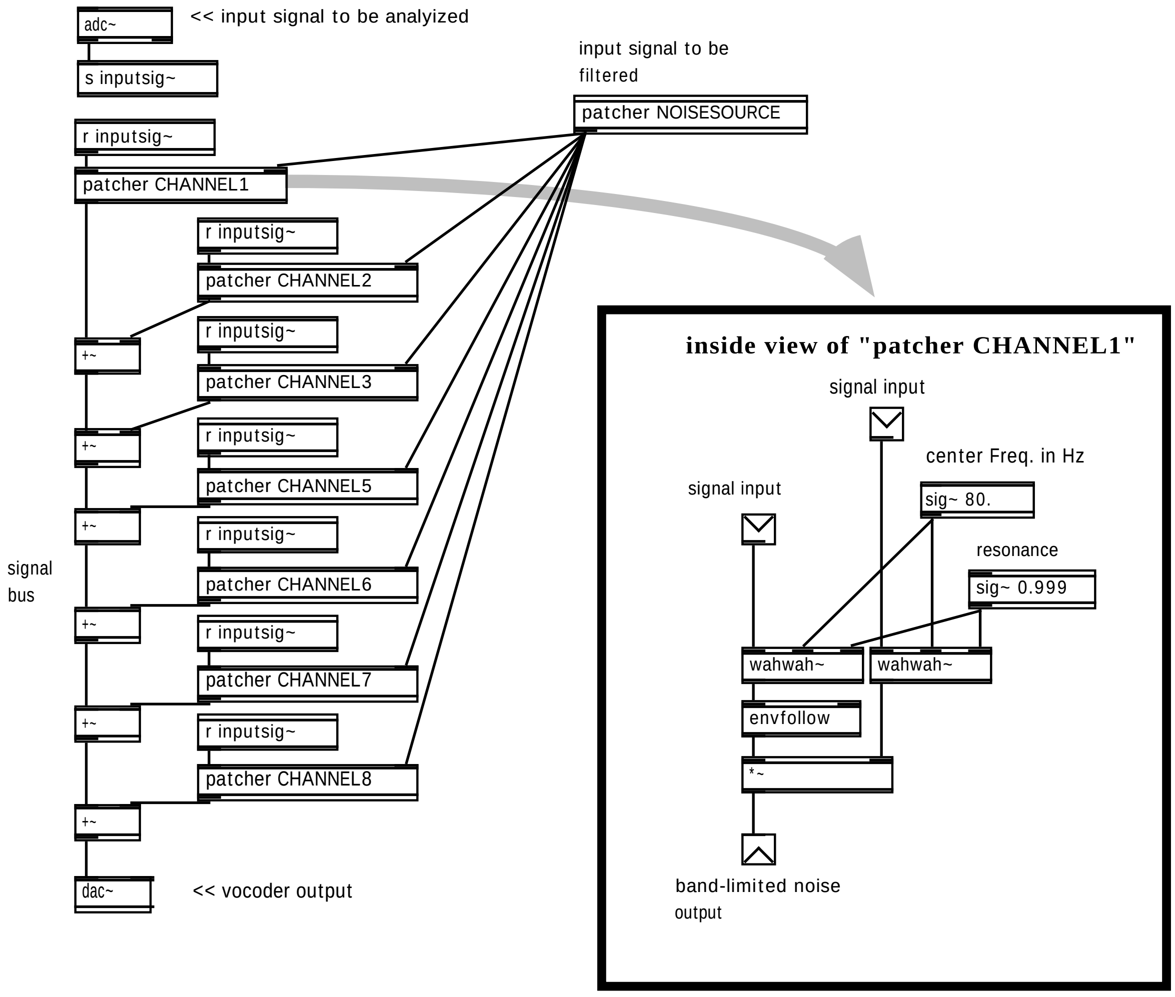


figure 3.



adc~
<< input signal to be analyzed
s inputsig~
input signal to be filtered
patcher NOISESOURCE
r inputsig~
patcher CHANNEL1
r inputsig~
patcher CHANNEL2
+~
r inputsig~
patcher CHANNEL3
+~
r inputsig~
patcher CHANNEL5
+~
r inputsig~
patcher CHANNEL6
+~
r inputsig~
patcher CHANNEL7
+~
r inputsig~
patcher CHANNEL8
+~
signal bus
dac~
<< vocoder output
inside view of "patcher CHANNEL1"
signal input
center Freq. in Hz
signal input
sig~ 80.
resonance
sig~ 0.999
wahwah~
wahwah~
envfollow
*~
band-limited noise output

figure 4.