

Managing Live Electronic while Composing and Performing - a Comprehensive MAX/MSP Framework

Bruno Friedmann, HFU Furtwangen, Germany; bruno.friedmann@hs-furtwangen.de

This project was done at IRCAM, Centre Pompidou, Paris, oct'07 - febr'08

- 1 Introduction
- 2 Musical composition
- 3 The MAX/MSP framework
- 4 Controlling and usability
- 5 Further interesting work
- 6 Conclusion
- 7 Acknowledgments
- 8 References

Abstract

A framework of MAX patches and functionalities was developed closely related to the creation of a 2 voice composition for clarinet and accordion. The realized affector patches use several IRCAM software tools for real time interaction and sound design. Aside from the affector patches development, great importance was attached to controlling and usability of the electronic parts to become an intrinsic and habitual part of the creative composition process itself, especially for an integral composing work flow. It appeared, that the controlling of the many powerful and intensively used live electronic applications is a bottle neck for the composer which constricts the usability.

1 Introduction

To use the powerful and sophisticated live electronic tools, developed at IRCAM over many years, exhaustingly, in rapid succession and in parallel, needs an attentive and advanced approach to manage the handling and makes it necessary to have a powerful interface for the purpose of composing. Otherwise this will be the bottle neck between the available powerful electronic tools and the innovative and demanding expectations of the composers.

To come close to this subject, a composition for two instruments, clarinet and accordion, was designed and a framework of MAX/MSP patches for an automated realtime performance was developed simultaneously. The patches are supposed to process and create an additional audio signal for each instrument in realtime, accomplished in strong relation to the given score. For both voices a score follower is implemented to follow the score events and create cues accordingly. System events (of the framework) send cues as well. A tracking system, which is able to create tracking cues, is also easily to implement.

Having all the powerful real time live electronic tools, developed at IRCAM in recent years available, there is nothing to be said against using it continu-

ously, maybe even for every note. Other performance techniques like bow and phrasing techniques for a violin or like taking into account suitable harmonies, are also used continuously as a matter of course. So, live electronic tools as important means of expression, widen the composer's power, but this requires an excellent and versatile usability.

2 Musical composition

As done multiple times in compose history, the Fibonacci sequence was used to define the basic musical scales. Here, the Fibonacci numbers define the tone distances in the dodecaptonic (12-tone) system, blue in Fig. 2.1, resulting in a 5-tone scale **F**. Added together with the transposed scale T_8 , azure blue in Fig. 2.1, one gets a 9-tone scale **D**. Described as tone distances one gets

$$D = F + F T_8 = (0\ 1\ 1\ 2\ 3\ 5) + T_8(0\ 1\ 1\ 2\ 3\ 5) \\ = (0\ 1\ 1\ 1\ 1\ 3\ 1\ 1\ 1\ 2)$$

or described as dodecaptonic tone numbers:

$$D = F + F T_8 = \{0\ 1\ 2\ 4\ 7\} + \{8\ 9\ 10\ 12\ 15\} \\ = \{0\ 1\ 2\ 3\ 4\ 7\ 8\ 9\ 10\}$$

this scale was used for the clarinet voice. For the accordion voice, **D** was inverted, see Fig. 2.1, resulting in the scale **ID**.

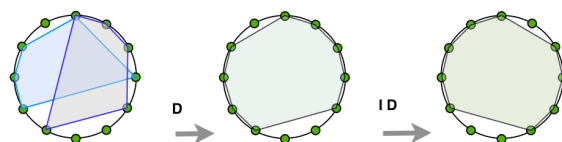


Fig. 2.1 Tone circle with Fibonacci intervals, defining the used 9-tone scales **D** and its inversion **ID**.

The rules of the 12-tone music are considered to be relevant for the composition. There are several patches which are based on the defined scales, like *FiboPitch*, which transposes sounds within these scales and *voiceGen* which creates second voices with a defined algorithm within the accordant scale.

Conceptional, there is intended to have a real time processing of both instruments in parallel to the direct sound and furthermore, the use of past musical phases and sounds, processed as well (patcher backward).

3 The MAX/MSP framework

As mentioned, the affector tools were developed closely related to the concept and special use of the musical composition, simply to enhance the instrument's music and the musical concept at it's best. Fig. 3.1 shows the current functional patcher arrangement.

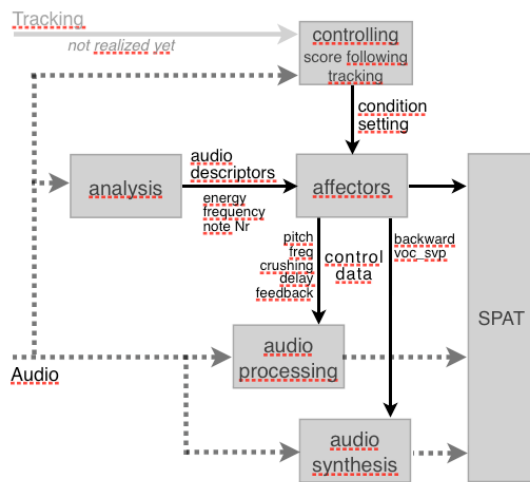


Fig. 3.1, Functional diagram of Framework 2.3

The patches affect real time audio and some control parameters of the *Spatialisateur*, like source position or room characteristics and process past sounds and musical phrases (audio synthesis) to be added at score defined positions.

The both incoming audio signals are prepared by the patcher *in*. *Analyse_in* delivers the necessary audio descriptors and the score follower component *scFo* provides the cues for controlling. The affector patches are often controlled by audio descriptors and deliver control data for the audio processing, Fig. 3.2. At the bottom, the patcher *signal_scheduler2Spat* mixes 7 audio signals to 3 sound sources, which are sent to the *Spatialisateur*.

Details

The real time affector tools, mostly controlled by audio energy, influencing pitch and frequency, are : *enCtlsPch* (energy controls pitch), *crusher* (pitch

shifting is compensated by frequency shifting) and *detuneInv* (pitch deviations are overcompensated by pitch shifting in the opposite direction, interesting for the clarinet, only). *FiboPitch* pitches the sound in defined steps (parameter) of the relevant tone scale. Furthermore, sound can be placed spatially in *Spat*, by *FiboPlacing*, depending on the detected Fibo tone number. Also the perceptual characteristic of the room acoustic qualities (*Spat_OPer*) can be changed in a subtle way - the louder the sound, the more is the reverberation reduced - controlled by the loudness, with patcher *enCtlsSpace*.

Some patches need the input of both instruments, like *voiceGen* (generates, algorithmically defined, second voices) or *duoSpace* (creates delay and feedback effects, depending on the energy and frequency of both audio signals) which in fact realizes a spatial communication of the two voices. The parameters of a vocoder, patcher *voc_svp* (using *supervp.cross~*), are continuously controlled by an algorithm (*sphere*), which integrates and processes the energy descriptors of the audio signals.

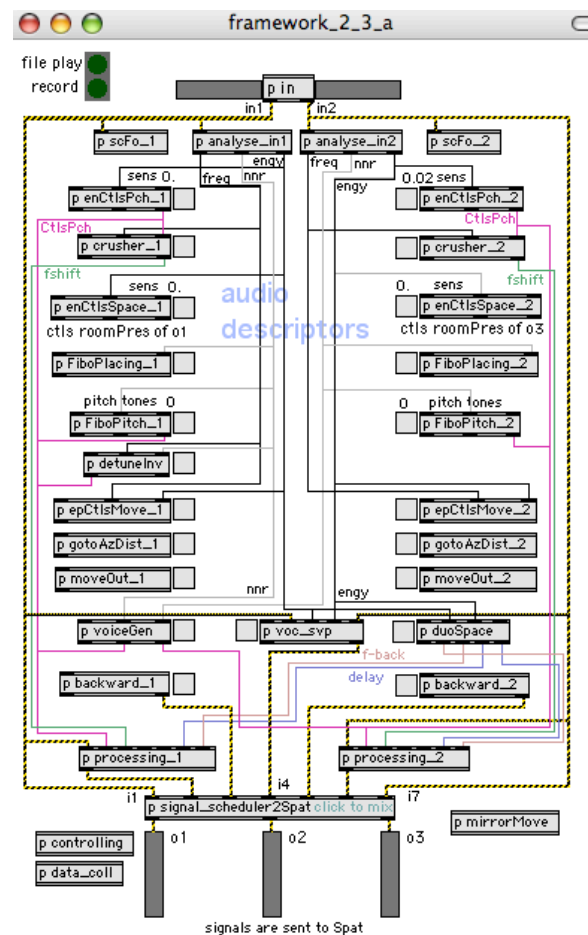


Fig. 3.2, Patcher arrangement and interconnectivity

The patcher *backward* processes past audio elements from the buffer by using *sogs~*, the granular synthesizer and add them accordingly to the score.

In Fig. 3.2, the audio descriptors (frequency, midi note numbers, audio energy) are shown as data lines, system events which enable and trigger some patcher functionalities and change the system state (called conditions) in time, are global variables and not shown. Neither the control variables to *Spat*, which are mostly communicated by midi signals. All the main patches are described in appendix A1.

4 Controlling and usability

The aim of this project is, to make an electronic framework, to support and enhance the music, composed in parallel, but also to become an intrinsic and habitual part of the creative composition process itself. To achieve this, a comfortable and suggestive way to handle the cues obtained from the score follower, the framework system and, if necessary, the tracking system, has to be found and implemented comprehensively. That means, the cues, which indicate changes of score or system or tracking events, must be related to the system variables to define easily numerous system conditions and to change and modify it while composing, according to the supposed course of the performance. This isn't realized yet in a satisfactory way.

A realization, in the style of the traditional live electronic controlling, is shown in Fig. 4.1. Normally, the performance is driven by the cues, resulting from score following and from system output. But a condition number can also be set by a musical assistant. This approach is based on the concept of defining all necessary system variables for every condition (momentary state of the system).

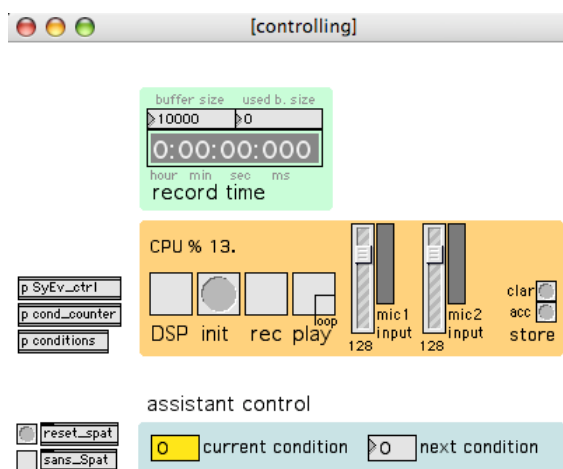


Fig. 4.1 Control interface for optional assistant interruption

These *system variables* describe the state of the system, of the involved patches, respectively, that means the current condition of the system. To go further to the next state or condition, an interrupt, done by a cue, resulting of a change of an appropriate score or tracking or system event, is necessary. For example, the recognition of a certain sound, the end of it, etc. This control concept can be done advantageously by using a scene description with graph theory. Since there is a composed work with a definite flow, the graph would be very simple - only one directed branch, a vertices can only be reached from a definite preceding one.

All conditions have to be defined by setting all relevant system variables, see Fig. 4.2.

>> state of the project <<

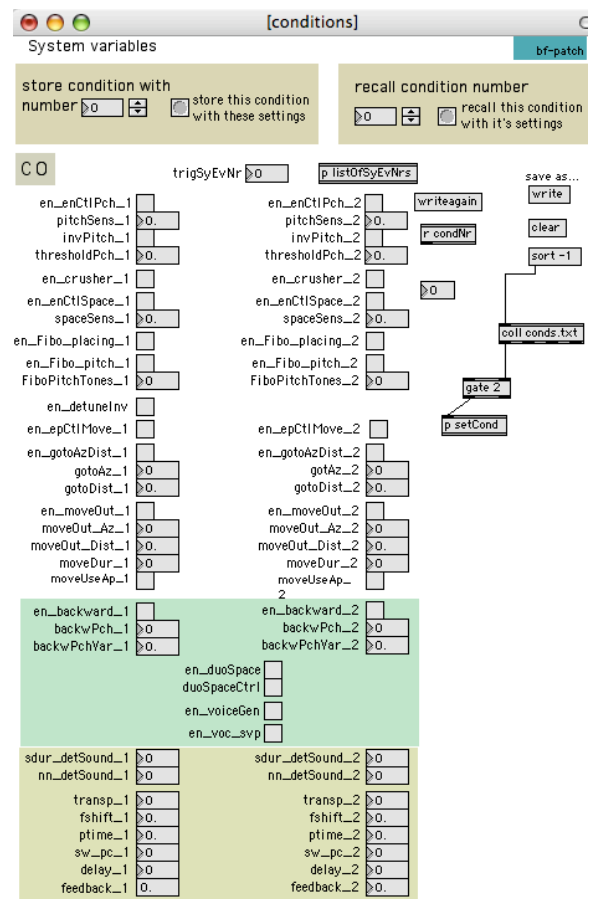


Fig. 4.2 Condition setting and controlling (condition number 9)

5 Further interesting work

As it is to see in Fig. 4.2, the definition of just one condition, that means the system state at one defined moment, is in fact laborious. Using score following and automatic scheduling, with a more or less continuous exertion of electronic influence, several hun-

dreds of conditions might be necessary to be defined. A composer wouldn't like to do that, he merely will reduce the use of the technical possibilities while composing, since there is no proper hierarchical meta level for a comfortable and easy going description including copying and changing of conditions.

Starting from this point, a concept of managing the relations between the cues and the system variables is to find and to implement. From the view of a composer: to find a easygoing way to enable, disable the affectors and to set and define the according parameters.

6 Conclusion

The time, I spent at IRCAM was really successful. The project I intended to work on, enabled me to become acquainted with the many audio and musical software systems and concepts developed at IRCAM.

I tried to create live electronic tools while creating a musical composition concurrently, both very specialized to each. This led directly to usability aspects due to the extensive use of the given processing tools. Between the processing of audio signals (view of the sound tool developers) and the use of the processing tools in a temporary context and in a dramatic composition (view of the composers and musicians) is still a gap to bridge, to make the powerful electronic affectors and enhancements, developed at IRCAM during the last years, really available and immersive for the composer.

7 Acknowledgments

I would like to thank Hugues Vinet and Olivier Warusfel for having made it possible to stay at IRCAM and to work in such an inspiring and highly competent environment. And I have to acknowledge the researchers at IRCAM for their support and helpful discussions, especially Olivier Warusfel concerning the Spatialisateur.

9 Appendix

A1 Description of the main patches, seen in patcher framework_2_3.

main patches	description
p in	buffer settings, recordings of the two instruments or playing soundfiles
p analysis_in	based on gbr.psy~ (Gabor); delivers audio descriptors and audio events
p scFo	score follower suivi.score~ not implemented comprehensively, yet
enCtlsPch	Energy controls pitch. With the loudness of his sound, the musician can control the pitch of the own recorded sound or another one

8 References

- N. Schnell, D. Schwarz: Gabor, multi-representation, real-time analysis/synthesis. In *Proc. of the 8th Int. Conference on Digital Audio Effects (DAFx#05)*, Madrid, 2005
- D. Schwarz, N. Orio, N. Schnell: Robust polyphonic midi score following with hidden Markov models
- D. Schwarz, A. Cont, N. Schnell: From Boulez to Ballads: Training IRCAM's score follower. In *Proceedings of the International Computer Music Conference (ICMC)*, Barcelona, 2005
- J.-M. Jot et al.: Spatialiseur, Introduction. IRCAM, Centre Georges Pompidou, 2006
- J.-M. Jot et al.: Spatialiseur, Reference Manual. IRCAM, Centre Georges Pompidou, 2006
- J.-M. Jot: Efficient models for reverberation and distance rendering in computer music and virtual audio reality. In *Proceedings of the International Computer Music Conference (ICMC)*, 1997
- G. Mazzola: Les «Structures» de Pierre Boulez - un lemme de Yoneda en musique ? presentation at m-mux, Dec'07, at IRCAM
- M. Andreatta: Méthodes mathématiques pour l'informatique musicale. Formation ATIAM, Université Pierre & Marie Curie, IRCAM - Centre Pompidou, 2006/2007

main patches	description
p crusher	a pitched signal will be transformed back via frequency shifting. This leads to a strange arrangement of the harmonics
enCtlsSpace	Energy controls spacial components of Spat. „the louder it sounds the smaller the auditive room“. With the loudness of his sound, the musician can control room presence and late re-verberance of Spat_Oper
p FiboPlacing	takes a note number of an audio signal (gained from analysis) and gives the audio source in Spat a place. Playing sounds increasingly the sound source will follow a spiral
p FiboPitch	takes note number, pitches it up with n steps (Fibonacci D or ID scale) and sends appropriate pitch signal to „processing
p detuneInv	dedicated to „detunable“ instruments, so, here for the clarinet. Takes a detuning from real tone frequency to create a pitch (cent) in the opposite tuning direction. So, the quality/roughness of a sound can be controlled
p epCtlsMove	Energy and pitch control the spacial components of Spat. Moves a sound source in SPAT_Control. Azimuth depends on frequency changes, distance on sound energy
p gotoAzDist	sets a Spat source to a certain position in space, immediately
p moveOut	moves a sound source in Spat_control from current position by a given amount dAz and dDist in the time given by „moveDur“. Using aperture optionally, reduces the aperture in Spat to 30° immediately, moves the source to the new position in movDur/2 and opens the aperture to 270°
p backward	based on sogs~ (FTM, Norbert Schnell), smooth overlap granular synthesis. Allows playback of parts of the buffer, in any speed, also backward with the possibilities to pitch, shift and other effects.
p processing	based on harmv2~ (The “jimmies” by Zack Settel) features: pitch, frequency shifting, crushing, delay, feedback
<i>to be used with both instruments</i>	
p duoSpace	The loudness of instrument a controls feedback (in patcher processing) of instrument b. And instrument b controls its spatial placement (Spat) with its own loudness. Additionally, b’s loudness controls its delay. The louder b sounds the more apart from the center b is placed and the longer the delay.
p voiceGen	takes pitch (frequency) from in1 and in2 as note numbers and creates via an algorithm two second voices in the chosen tone scale Fibo_D or Fibo_ID as relative pitch values to be sent out to the patcher processing.
p voc_svp	based on supervp.cross~ (Axel Roebel). Real time cross-synthesis module for both audio signals. Includes sphere_cross which combines both signal energies to enable a continuous and versatile cross-synthesis (see sphere_cross)
p mirrorMove	at a dramatical point of the music, the musicians moves, while playing from left to right and back and the sound sources are mirrored in the way that they move from right to left in the rear of the room
p signal_scheduler2Spat	includes a mixer - 7 input, 3 output - with presets and the possibility to „remote“ control the signal transfer gains. The mixer, signal_scheduler2Spat mixes 7 audio signals to 3 sound sources which are delivered to the Spatialiseur. The 7 audio signals are: two original signals (i1, i7), the processed one (i2, i6); the patcher backward create two signal via granular synthesis (i3, i5) which can deliver musical phrases played in the past. The vocoder voc_svp sends the audio signal i4.