OMax

The Software Improviser

Version **ß 1.0**

Omax is designed by *The OMax Brothers:*
G. Assayag, G. Bloch, M. Chemillier

http://www.ircam.fr/ircam/equipes/repmus/OMax

TABLE

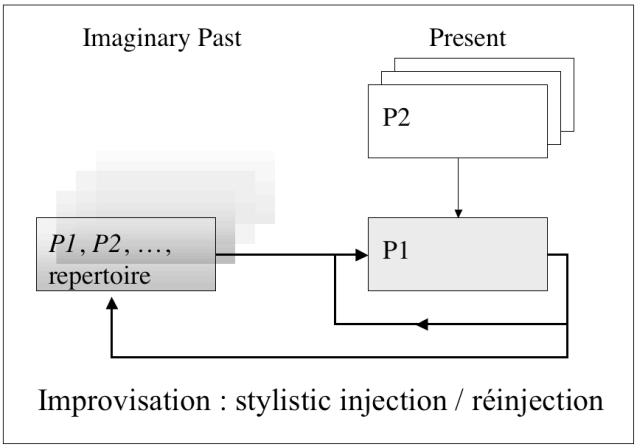
Introduction: OMax and the Stylistic Reinjection Process

Machine improvisation and related style learning problems usually consider building representations of time-based media data, such as music, either by explicit coding of rules or applying machine learning methods. Stylistic learning of musical style use statistical models of melodies or polyphonies to recreate variants of musical examples. Recent advances in learning techniques indicate that particular behaviors can be emulated and that credible behavior could be produced by a computer for a specific domain.

In the field of music improvisation with computers there has been recently notable advances in statistical music modeling that allows capturing stylistic musical surface rules in a manner allowing musically meaningful interaction between humans and computers. The contributors to OMax have experimented with several of these models during the past years, and recently cristallized the results of their researches in OMax, an environment which benefits both from the power of OpenMusic (Assayag, Agon), for modeling and high level programming, and MaxMSP (Puckette, Zicarelli) for performance and real time processing. OMax allows for interaction with a human player, on-the-fly high-level feature extraction and stylistic learning, virtual improvisation generation, stylistic model archiving and hybridizing. It operates on two distinct time scales: the Max one, which is close to real time and involves fast decision/reaction, and the OpenMusic one, which has a deeper analysis/prediction span over the past and the future. These two conceptions of time interact and synchronize over communication channels (OSC), through which musical data as well as control signals circulate in both directions. A decisive advantage we have found in this hybrid environment experience is its double-folded expandability. In the OM domain, it is easy, even while the system is running, to change the body of a lisp function and test incremental changes. Furthermore, it is easy to enrich the generation by connecting the system to a wide variety of compositional algorithms available in this environment. The same thing is true in the Max domain, with a comprehensive collection of real-time generation and processing modules. We think of this setup more as an indefinitely modular and extendible experimental environment for testing new ideas about interaction, than as a fixed application. However, in the setup documented here, there is only a subset of OpenMusic involved (we don't need the graphics), materialized as a small footprint lisp image, and a Max patch that can be run on top of Max or even on top of Max Runtime if you don't own Max.

OMax provides a virtual musical partner that learns all its knowledge from the musicians it's playing with, in a non-supervised mode, and that is fitted to a real-time audio context as well as Midi. One way of looking at the interaction that emerges between the musician and OMax is to characterize it as a process of Stylistic Reinjection. The musical hypothesis behind stylistic reinjection is that an improvising performer is informed continually by several sources, some of them involved in a complex feedback loop (see Figure below). The performer listens to his partners. He also listens to himself while he's playing, and the instantaneous judgment he bears upon what he is doing interferes with the initial planning in a way that could change the plan itself and open up new directions. Sound images of his present performance and of those by other performers are memorized, thus drifting back in memory from present to the past. From the long-term memory they also act as inspiring sources of material that would eventually be recombined to form new improvised patterns. We believe that musical patterns are not stored in memory as literal chains, but rather as compressed models, that may, upon reactivation develop into similar but not identical sequences: this is one of the major issues behind the balance of recurrence and innovation that makes an interesting improvisation. The idea behind stylistic reinjection is to reify, using the computer

as an external memory, this process of reinjecting musical figures from the past in a recombined fashion, providing an always similar but always innovative reconstruction of the past. To that extent, the virtual partner will look familiar, as well as challenging, to his human partner. The interesting thing is to see how the human partner reacts to his "clone" and changes his improvisation accordingly. Top improvisers, who have used this system at an early prototype stage (Bernard Lubat, Mike Garson, Philippe Leclerc, Hélène Schwarz, Guerino Mazzola and others) have developed their own way of interacting, as can be seen and listened to on the OMax web site: http://www.ircam.fr/equipes/repmus/OMax.



Pragmatically speaking, OMaxMax (the Max component) "listens" to a musician partner through Midi or audio channels. It extracts high-level features from the incoming signal and segments it into events and phrases. OMaxMax feeds continuously OMaxLisp (the OM/Lisp component) with a stream of Midi-like information (we call this augmented Midi). On its side, OMaxLisp continuously builds a memory model of the sequence of events. This Model is called an Oracle, based on works on string sequence modeling by Crochemore, Allauzen and al. (Oracle have been used, among others, in order to discover patterns in DNA strings). OMaxLisp is also ready to continuously "improvise", that is browse the model in order to generate variant sequences that it sends as a continuous augmented Midi flow to OMaxMax.

A lot of interface controls let the OMax operator steer the virtual improvisation by navigating into the memory of the session, from the immediate present to the far past. As a session can be archived and continued later, this far past could be weeks or years ago. As oracles can be hybridized, the memory model could be trans-individual as well and incorporate pieces of the Repertoire. Using these controls, the operator will be able to conduct a co-improvised performance with one or several human musicians.

We leave it to the bibliography to document the many technical aspects underlying the OMax technology.

OMax has been mainly developed by Gérard Assayag, Georges Bloch and Marc Chemillier. The stylistic modeling researches underlying it have been carried mainly by Gérard Assayag and Shlomo Dubnov.
OMax uses Yin~ the pitch tracker technology by Alain de Cheveigné (Max implementation by Norbert Schnell).

System Requirements

OMax runs on Apple PowerPC.
Mac Intel version is not available yet.
1GHz, 1GB G4 is a minimum for the audio version.
Omax needs Max 4.6 or Max Run Time 4.6.
OMax is multi channel: if your audio hardware permits it, you will be able to play on 8 channels.

Getting Started

Launch OMaxLisp and OMaxMax. Keep the two windows side by side.
Be sure to put the OMaxMax folder in your Max/MSP File Preferences.

Choosing between the Midi World and the Audio World

Before any use of OMax you have to decide if you want to experiment in Midi mode or in Audio mode. In Midi mode you'll have a Midi input (e.g. a performer playing a Midi keyboard) and a Midi output (OMax improvising to a Midi port connected to e.g. an external Midi expander or a software expander or a sequencer). In Audio mode, you'll have one (preferably two, see further) audio input through Max dac1/dac2, and OMax will be improvising by recombining the audio captured at the input and sending it to one to eight audio output channels. In addition, in the audio mode, you'll be able to listen simultaneously to 2 Midi outputs corresponding to the direct audio input as analyzed by the pitch tracker, and to the OMax improvisation.

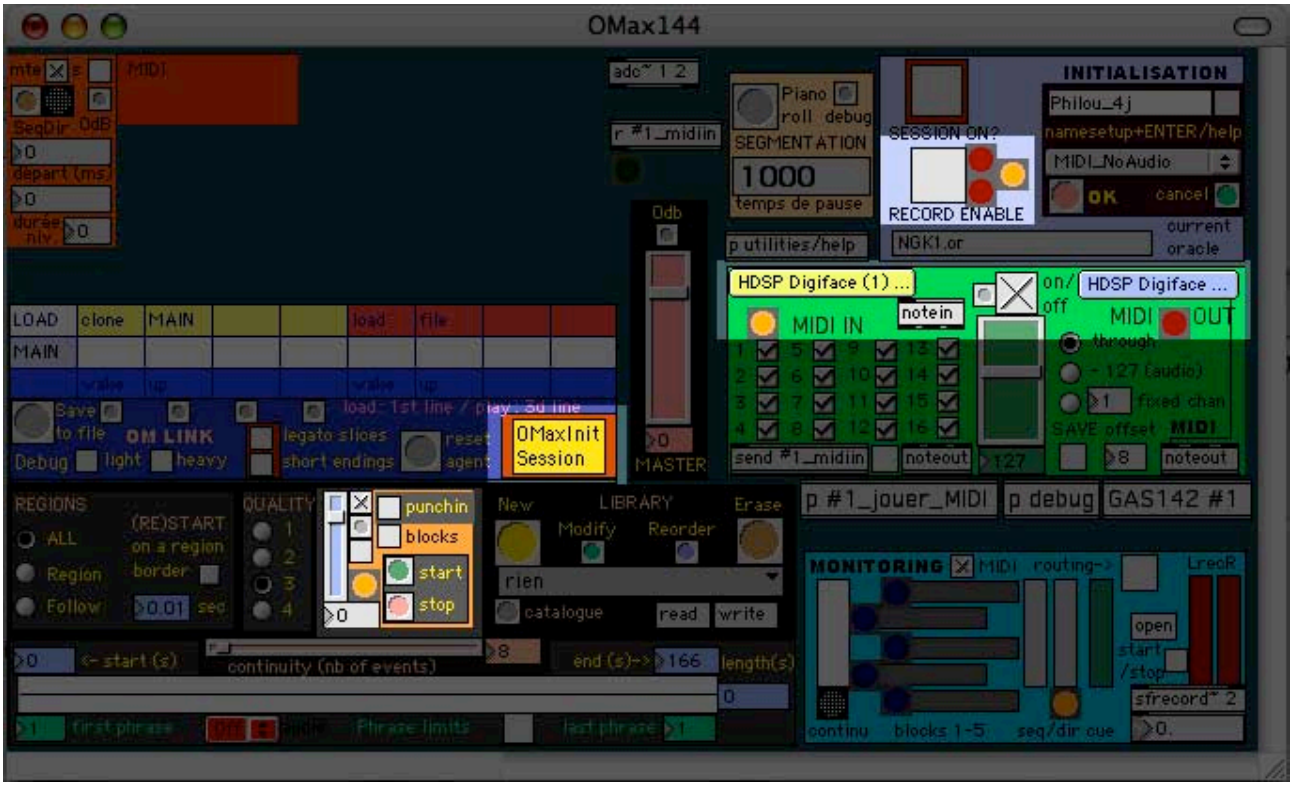
In the Initialisation panel, in the popup menu, choose MIDI_NoAudio and press OK, in order to choose the Midi mode. Every other item in the menu selects the Audio mode. The different audio items corresponds to internal settings that have been carefully tuned for selected input instruments: Tenor Sax, B flat Clarinet, Bass Clarinet, Voice, Percussion, Spoken Voice. The « Default » audio item is a good average setting to start with. The "pianobar" set is designed for the Moog pianobar, or an audio instrument equipped with its own Midi detector.

When switching from one mode to another, the Red top left panel changes its configuration. Choosing a mode activates a script that changes the patch organization. Advice is given to do this once at the beginning of a session, just after loading the patch, an to not fool around switching modes while the patch is running. You can save-as your patch (inside the folder it came from) and it will remember its Midi-or-Audio configuration. So you could have a patch specifically for the voice, and you won't have to configure it anymore.

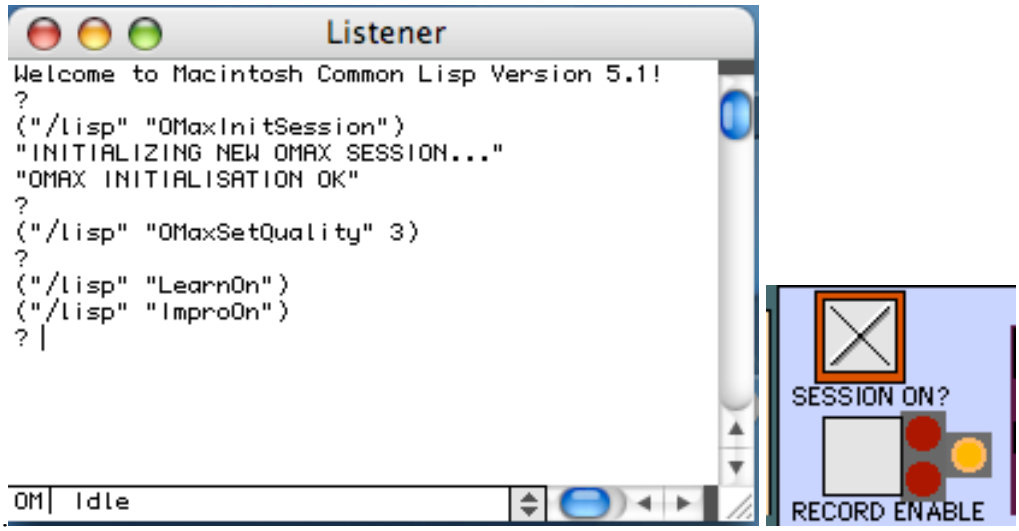
Before proceeding to Tutorial 1, go into Midi Mode (Don't forget to press OK).

Tutorial 1. Simple Midi Processing

On the figure below, we have highlighted the 4 zones of interest for launching a simple improvisation process.



1. Use the Midi port menus to indicate your input and your output.
2. Press ***OMaxInitSession***. This will initiate the communication between OMaxLisp and OMaxMax. The OMaxLisp listener and OMaxMax will react as pictured below:



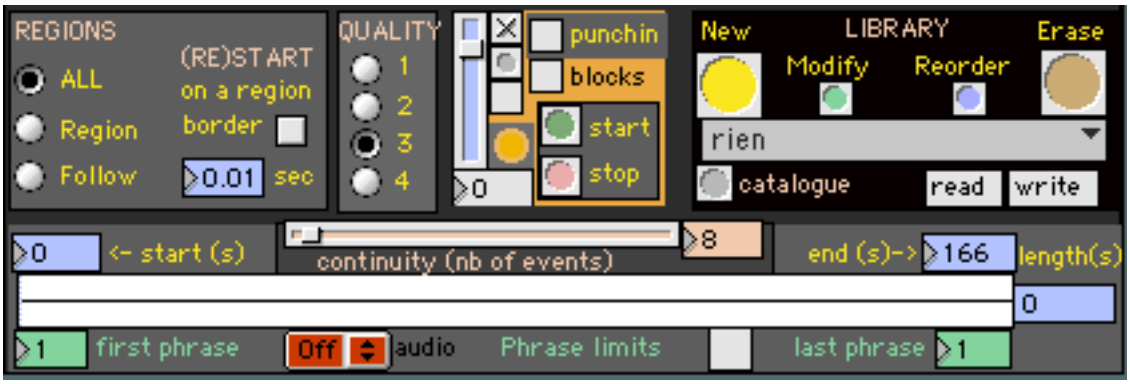
3. Check the toggle box ***Record Enable***.
4. If there's Midi flowing at the input, OMax will begin (almost) instantly to reimprovise it, sending it's improvisation on the selected out port.

5. Use ***start*** / ***stop*** to control the improvisation, as well as the *level slider*, and the *Mute/0dB/Cue* little buttons nearby the slider.

If you would like to use a Midi file instead of live Midi input in order to test the system, use the helper application *Read_sequence*. It will let you open a midi file (must have the .mid, lower case, extension) and send it to an output port (which of course must be the same as the input port selected in OMaxMax). Of course, you can also use any type of Midi sequencer and send it to OMax through IAC or toMaxMSP ports.

If you would like to keep OMax improvising, but stop « listening » to the input (and thus « learning » new material), just toggle *Record Enable* off.
If you would like to start another session, forgetting all the material learned so far, push the *OMaxInitSession* button again. The session will be reinitialized, and the toggle *Record Enable* will stay in the state (on or off) it was in.

Tutorial 2. Controlling your Improvisation



If Tutorial 1 was succesful, you have now Midi streaming in (e.g. a Midifile, or a musician playing a Midi keyboard) and OMax generating a co-improvisation simultaneously.
Now you would like to control this co-improvisation. This is the purpose of the Panel above.

We have already seen ***start*** and ***stop***, which just starts or stops the OMax Improvisation (but do not prevent OMax from listening and learning).

The ***Continuity*** slider helps to control the density of recombination in the OMax improvisation. If set to 8 (default), it will, in average, replicate 8 successive musical units from the learned sequence before trying to recombine, that is jumping to another place in the sequence. It is good policy to set it to a higher value (e.g. 16) when the input is dense (high tempo, lot of notes per time unit). Of course this is just a statistical indicator: OMax has a very sophisticated scheme for looking at the best combinations, so it could choose to increase the continuity internally in order to favor a better location to jump. Very low values could result in a chaotic behavior, which may be of interest.

The ***Quality*** radio buttons sets the quality of recombination. High quality recombination (lower values of the button, e.g. 1) jump between places that share a long common musical context, and that are rhythmically more coherent. One should prefer higher quality, of course, but, depending on the type of music at the input, high quality could result in no recombination at all (thus just replicating the input). The default value is 3; you should use 2 as much as possible, and 1 is adapted for some types of music with high redundancy rate.

The ***Regions*** radio buttons lets you control in which part of the input sequence learned so far OMax should peek musical material to recombine. They can be seen as a control of the memory of the improvisation (long term, short term).

- ***All*** means the whole sequence. In this mode, OMax navigates freely in the totality of the material learned so far.
- ***Region*** limits the available material to a certain time region of the sequence learned so far. Time region are selected using the horizontal rslider (see below).
- ***Follow*** is a particular case of a region that is continuously redefined in order to follow the real-time input. The length of this region is set by the numbox just right to the follow button. With a length of 5 seconds, OMax will always be re-improvising the material that has been played during the 5 last seconds, following as in a pursuit of the performer, while he/she moves ahead in time. Note that choosing ***Follow*** for a while, then choosing ***Region*** will « freeze » the current follow region: OMax will cease to

follow the performer and navigate in the freezed region. If the region is small, this is a particularly interesting way of installing a groove derived from very recent material, on top of which the performer will feel at ease in elaborating a chorus.

The long horizontal **Rslider** represents the whole session from the beginning (left) to the current time (right). It is a linear mapping of the Oracle itself, which you can see as a sequence representing in an ordered way the music that has been input since the beginning of the session. By selecting regions with the mouse in this slider, you will select input material that has been played at some point in the past, and force OMax to re-improvise this material, instead of wandering freely into the whole sequence.
Because the slider is of fixed size, time is continuously scaled to fit in as the session moves ahead, so a 20 seconds region could occupy a smaller or greater horizontal space depending on the overall duration of the session so far. The total length of the oracle is indicated on the right side of the Rslider.

Regions selected in the rslider become active only when the ***Region*** radio button is pressed.
Two blue numbox indicate the current region start and end time in seconds. The blue numbox at the far right (length) indicates the duration of the sequence learned so far.

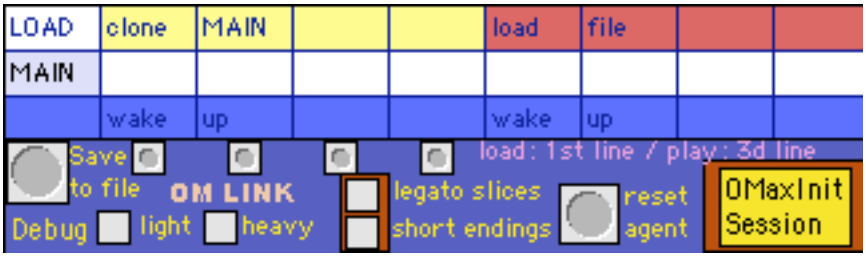
Play a while with this interface until you feel comfortable, then go to Tutorial 3.

Tutorial 3. More Control of your Improvisation: Phrase Segmentation

As the input from the performer is flowing in, it is analyzed into « phrases ». Phrases are just musical sequences surrounded by pauses of a certain length (default 1000 ms). This is a more convenient way than continuous time in seconds to represent things and to remember specific moments of interest.
If you check the toggle ***Phrase Limits***, then your selection in the rslider will be automatically aligned with phrase boundaries. The green numboxes will indicate the phrase range. If you just click (without dragging) in the rslider, you will select the phrase surrounding the point in time corresponding to the click, and its number will be indicated in the green numboxes.
For instance if you remember that the performer has played three long phrases since the beginning, it becomes extremely easy to select, say, the second phrase, then force OMax to improvise on that phrase with the ***Region*** radio button.
Whether your are in phrase mode or in continuous time mode, you might want to trigger on and off the improvisation several times (using ***start*** and ***stop***) and have OMax begin every time at the same place before going into some improvised phantasmagoria. This is the meaning of the toggle ***(Re)Start on a region border***. Actually OMax will always restart on the first event of the region specified. Every time you restart, OMax will begin with the same phrase, replicating the performer for a while (depending on continuity) then digress into his own improvisation, doing differently so for each restart.

If a region is selected and you start an improvisation in mode All, the region info will be used to choose a starting point. After the starting point, OMax is then free to improvise wherever it likes. So, even if the All mode favors a completely region-free improvisation, you can at least decide where to start, especially if you check ***(Re)Start on a region border***.

Tutorial 4. Oracle Housekeeping



The musical material input by the performer, taken as a sequence in time, is modeled in OMaxLisp into a data structure called an *Oracle*. Basically, it is the sequence itself, plus a certain number of arrows indicating where and how to jump. Thus, the Oracle is the long-term memory of the session, including a way to represent the logical organization of patterns.

OMaxLisp can handle several oracles at a time, for instance, the oracle of the current session plus oracles of previous sessions, and it can switch from one to another. It can even learn new material at the end of an oracle recorded at a previous session and reloaded for the circumstance. So the long-term memory may become a very long-term indeed, including sessions over months or years. One could even imagine that Omax learns the whole available music played by a performer and thread its own improvised path through this database.

One must remember that, even if OMaxLisp can maintain several Oracles at the same time, it learns only in one unique Oracle, called **Main**.

In order to perform such oracle housekeeping, you will use the panel pictured above.

- ***Save to File*** saves the main oracle to a file (convention: .or)
- ***Load*** *(red buttons)* load an oracle saved from a previous session into a memory slot.
- ***Wake up*** *(blue buttons)* the oracle loaded (or cloned see below, or the Main Oracle) in the corresponding memory slot becomes the active improvising oracle.
- ***Clone Main*** *(yellow buttons)*. The current Main oracle (the one that learns from the performer) is cloned into a memory slot. This is useful to keep oracles of different successive states of the session.
- ***Save clone*** *(the little Bang Buttons under each clone memory slot)* saves cloned oracle to a file.
- ***Load Main*** *(the white button)* is similar to the red load buttons, that is it loads an oracle from a file. But in this case, the loaded oracle becomes the Main, the oracle that learns. Learning will occur at the end of the sequence contained in the loaded Oracle. This is how you can continue a past session with a new session.

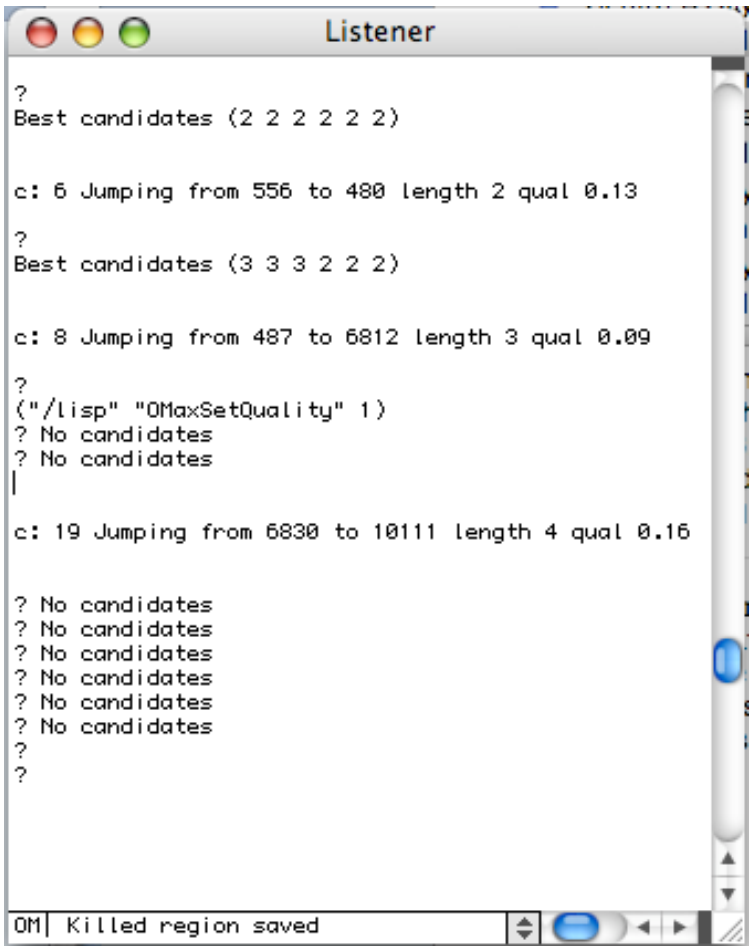
Notes:

- when an oracle is awakened to play, the interface in the improvisation control panel is updated to reflect its characteristics (duration, number of phrases). The region mode is set to All, the current region to (0,0) and the quality to 3.

- When you load a Main oracle, the previous one is deleted so you loose your data except if you saved it to a file or you cloned it.
- You can have simultaneously at hand: a Main oracle, 4 clones and 4 loaded oracles among which you can switch using the wake-up buttons.
- While you switch from Main to another Oracle, the new oracle becomes the improvising one, but the Main keeps on silently learning whatever input is played by the performer.

Other functions:

- ***Reset Agent*** resets the Main Oracle. This is comparable to ***OMaxInitSession***, except that you won't loose all your data stored in the clones and the loaded oracles memory slots. It clears only the Main (learning) Oracle.
- ***Legato slices*** if not set, will play durations exactly as learned (i.e. if a chord is slightly arpeggiated, it will be played like this). If set will quantify the duration (i.e. the chord will be played with simultaneous onsets and offsets).
- ***Short Endings*** when the performer stops playing, the oracle does not record the silence (which could be very long) before he plays again. Rather it records a max 2 seconds pause after the last event. If short endings is set, when the impro crosses such an ending event, it will not play a 2 seconds pause. Rather, it will give the event the duration of the preceding event.
- ***Debug light*** shows the recombination process in the OMaxLisp listener.
- ***Debug Heavy*** shows all the OSC messages flowing from OMaxLisp to Max.

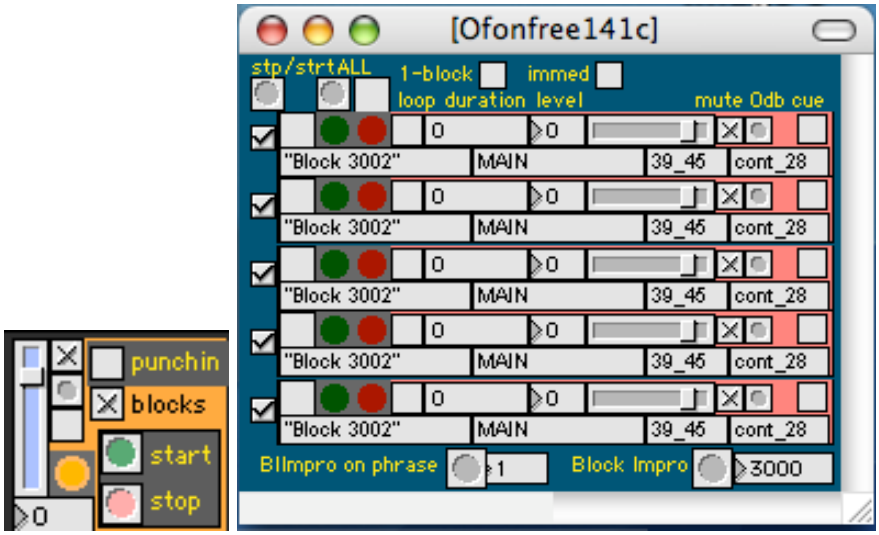


About the debug mode :

When using ***Debug Light***, you will see in the Listener (Lisp window) the recombination process. This can help to tune up the Quality parameter (see Tutorial 2). When it says « No candidates », it means that OMax does not find any recombination and will pursues the original sequence until it founds a possible jumping point. When it finds a recombination, it says « Jumping from x to y » where x and y are positions in the oracle. In addition it gives the length of the context and the rhythmical quality.
Too many « No candidates » may either show that the OMax improviser is currently in a particularly unique (non redundant) passage, but it might be that the quality factor is too restrictive (the value is too small, e.g. 1 or 2). Try a bigger value (e.g. 3).
Keep it mind that the debug print takes processing power.

Tutorial 5. Block Impros and Loops

By checking the toggle **block** on, you open a new window with a 5 tracks mixer-like interface.



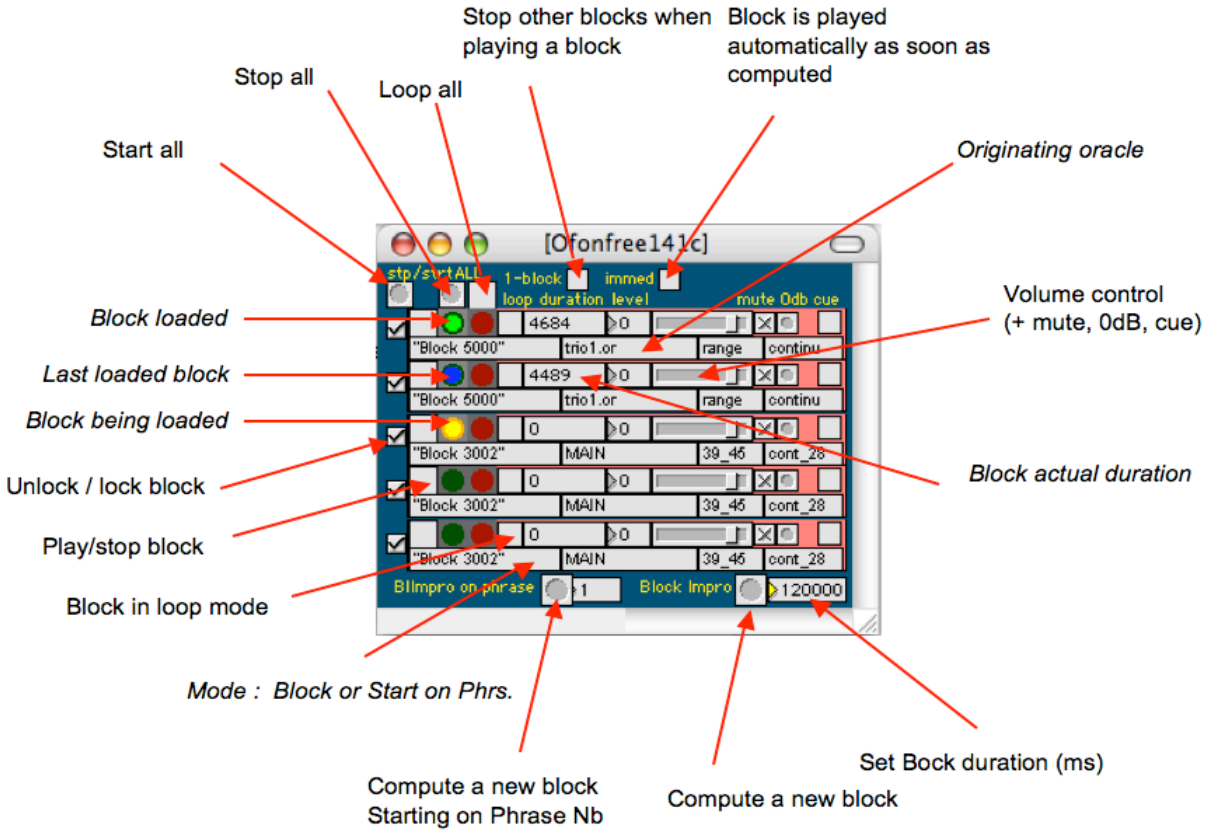
This allows you to "silently" compute new OMax improvisations and store them as sequences (called *Blocks*) to be played whenever you like. A Block will always play the same sequence, thus it is "improvised" upon first computation, but not every time you play it. This is a way to get constant things in an otherwise constantly changing system. Due to the concurrent agent architecture in OMaxLisp, computing a block does not interrupt other processes going on, such as "continuous" improvisation. Each time you compute a new block impro, it is stored as a sequence into the next available checked track (see little check boxes on the left of the tracks). Uncheck the track if you would like to protect it from being erased by a subsequent block computation and be sure to be able to replay it later.

When computing/loading a block, you shouldn't try to load another block before the computation is completed. The yellow light indicates that the loading process taking place, completion by the blue light. A green light indicates a block previously loaded.

To compute/load a block, type a duration and then press the ***Block Impro*** button. Wait for the blue light. The Impro is computed from the currently active Oracle, with the current Region/Quality/Start on Border parameters.
The ***BLImpro on phrase*** button computes a block starting on the indicated Phrase number. Duration is set the same way than before. The current active Oracle is used in mode All.

By using the loop mode and playing a polyphony of blocks, it is possible to get an extremely dense texture.

In the following picture, indicators are in italic, while active controls are in regular font face.



Tutorial 6. Presets for Improvisation Control

The Control window of OMaxMax has a library built into it for storing control presets.

Its primary use is during a performance, to remember particularly successful or interesting moments. The complete setup of the control window is remembered, with the exception of continuity, that is: Quality, Region mode (***All***, ***Region*** or ***Follow***), Region boundaries (and whether they are Phrase or Time limited), eventual follow time and the value of ***(Re)Start on a region border***. However, they can be saved in a file to be recalled with the corresponding Oracle. (By default, the file #1_lj.xml present in the OMaxMax folder will be loaded at start). Any setup can be instantly stored with the Button ***New***. The new setup will take the upper position in the menu. The current setup can be modified to the actual situation (button ***Modify***) or erased (***Erase***).

The menu shows the value of the parameters in this order:
Region mode / Start on a border / Phrase or time limit / Limits / quality

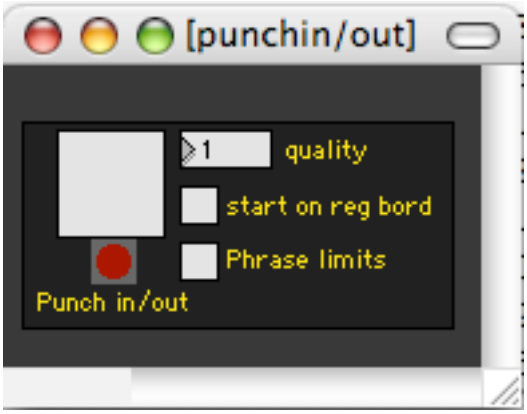
- Region mode: ALL / Reg / FOLLOW
- Start on a border: [(=yes) : (=no)
- Phrase limit: Ph (=yes) (nothing = no)
- Limits for ALL and Reg: XX-YY in milliseconds or #A-#B phrases numbers
- Time for FOLLOW: XXms
- Quality: Q1, Q2, Q3 or Q4

Examples:
- Reg:63000 54000_Q3
- ALL[Ph2-4_Q2

Notes on the use of the Library:

As said above, the Library is very much Oracle-dependant. So it is possible that a setup be irrelevant for a given oracle: for instance, the phrase or time borders could not exist in the given Oracle. In general, OMaxLisp sends back a default value when the borders asked for are impossible.

Another way of specifying and storing a region for later use is the Punch-in Punch-out window. To open this window, check the ***punchin*** toggle.

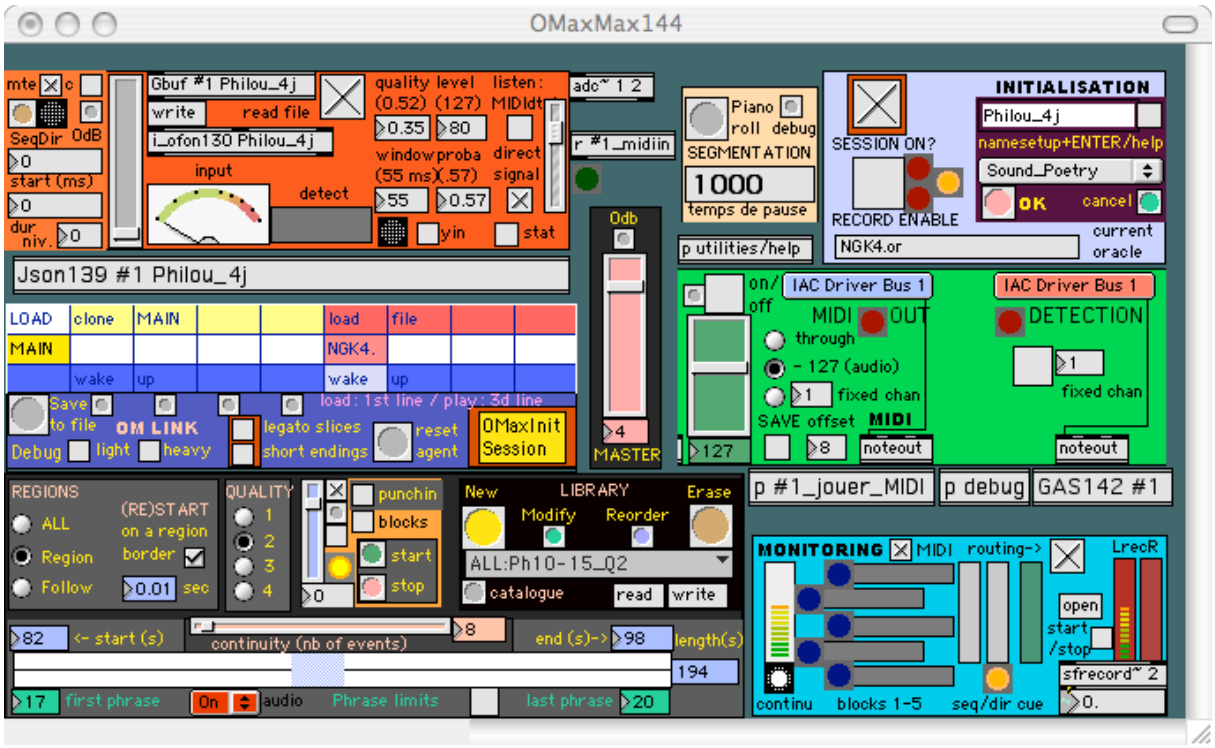


By clicking on the ***Punch in/out*** toggle, a region is created on the fly and stored in the library. This way you can specify a region in real time according to what you hear. The ***quality***, ***start on region border*** and ***phrase limits*** parameters associated to the punched region can be set beforehand using this interface.

Tutorial 7. Going Audio

In audio mode, all you have learned in Tutorials 1-6 stays valid, except now you're learning directly from an audio signal, and OMax improvisations are played in audio using the same sound as recorded from the performer.

After opening the OmaxMax patch, select the default audio instrument in the Initialization menu, press OK. A new audio panel will appear.



Check the audio on.

Check in the Max DSP options menu that every thing is set up for audio input output. You should have 2 inputs: adc1 for recording the performer's sound, adc2 for pitch detection.

A classical setup for e.g. a saxophon player would be:

- A good aerial microphone on adc1 for recording a quality sound (this is the sound you will hear when OMax wil be improvising)
- A contact microphone on the instrument on adc2 (pitch detection is better with a close take and you don't want the pitch detection canal to take the surrounding sound, including OMax impro, because this would result in a feed-back process).

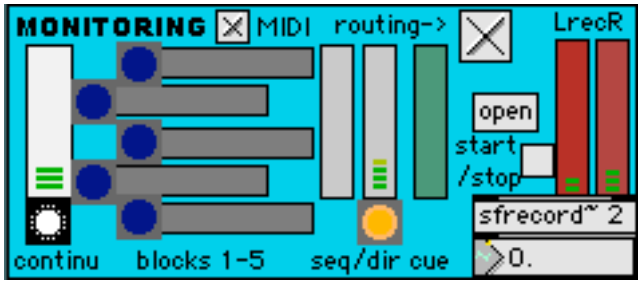
An alternative would be to use a single close mike (e.g. a mike clipped inside the bell) and send it to both adc1 and adc2. In this cheaper compromise, the sound for recording will be medium quality, and the detection, hopefully, will not catch too much of the sound environment.

For test purposes, you can just use the Mac's internal microphone (which is sent to adc1 and 2) and use a headset to avoid feedback.

Proceed as in Tutorial 1: ***OMaxInitSession***, ***Record Enable***, and you're gone.

Tutorial 8. Sound Output Control

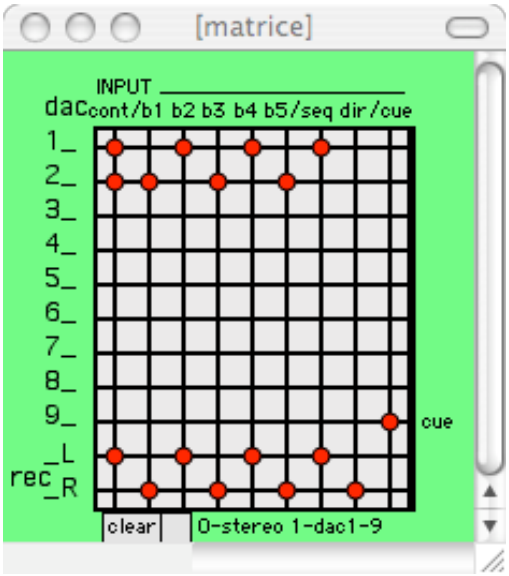
The Monitoring window allows to observe the sound busses. The Midi activity is also visible through the leds (it can be disabled by unsetting the MIDI toggle).



In order to understand the different signal-level meters in the *Monitoring* panel, here is the OMax terminology:

- **Continu** the continuous improvisation generated by the active Oracle
- **Blocks** the (up to 5) sequences played using the Block panel (see Tutorial 5)
- **Seq** the sequence is played straight using the SeqDir Panel (see further)
- **Dir** the direct input signal as captured on adc1
- **Rec** the signal sent to the record-to-file module
- **Cue** the signal is exclusively routed to a cue bus (for example headphones), and its normal output is muted

Checking the ***Routing*** toggle opens up a sound Matrix window.



The 9 signal tracks (continuous, block1-block5, seq, dir, cue) can be routed to up to 9 audio output (depending on your hardware setup) and/or to 2 record channels (Left / Right).
In order to record your mix, open a new sound file using the ***Open*** button in the Monitoring panel. Press the ***Start / Stop*** toggle to start/stop the recording.
By default, the direct signal is NOT sent to the sound system (this is logical for an acoustic instrument).

Interlude. Audio Buffers

As said previously, there is almost no difference in manipulation between the Midi mode and the Audio mode, except for a few more controls in the audio mode. However, going audio involves internal differences that have to be understood in order to comprehend the system behavior.
In the Midi mode, there is a model of the sequence played so far by the performer, maintained by OMaxLisp, called an Oracle. This learning oracle is called the Main. Other Oracles can be present at the same time: clones of the Main, and oracles loaded from files. One can switch from one to the other, making it the active improvising Oracle. Only the Main learns from the real time input. One can also load a main from a file, in order to augment it by learning from the real time input.
In audio mode, all this stays true. However, each oracle (Main, clone, loaded) has an audio counterpart under the form of a Max audio buffer. An oracle and its corresponding buffer are both sequences representing the same musical process that, at some time, has occurred, been recorded and analyzed. Oracles are symbolic sequences augmented with analytic structures, where buffers are just plain audio buffers, but they represent the musical sequence: in particular, they have the same overall duration.
Buffers are the result of recording the input audio signal on adc1, while Oracles are the result of recording and analyzing the symbolic data send by OMaxMax to OMaxLisp. This symbolic data is the result of continuously analyzing the input signal and extracting high level features (for the moment, events boundaries, pitch, intensity).

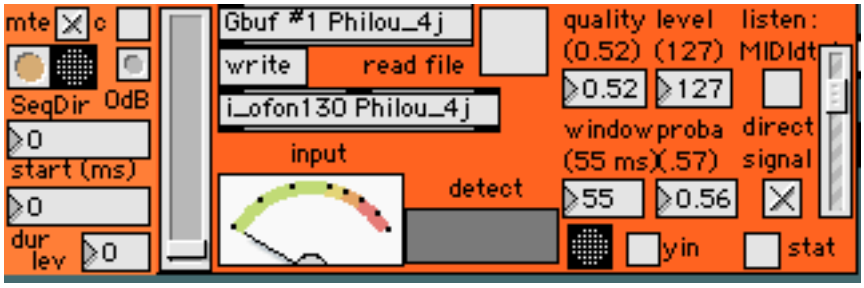
In the audio mode, every operation involving an Oracle involves and audio buffer as well. For instance, cloning an oracle will clone a buffer. Loading an oracle from a file, will load a soundfile into a buffer as well. Saving an oracle will also save the corresponding buffer to a soundfile. When a 150 MB buffer (the maximum, thus 20mn of sound) is involved, you bet this is more time consuming than in simple Midi. Memory limitations will not allow you to do everything you were doing in simple Midi. It's up to you to handle this, providing there is no limitation in OMax (even the 150MB max buffer size can be changed if you have enough RAM). However, it is recommended to start playing with not too large buffers if you use the clone/load features. It also depends on your machine and, even more, on the size of your RAM.

Tutorial 9. Saving / Loading in Audio Mode

When you save an oracle in audio mode (see Save in Tutorial 4), e.g. in file MySession.or, OMax will automatically save a soundfile called Mysession.aiff in the same directory. This soundfile is an image of the audio buffer corresponding to the Oracle saved.

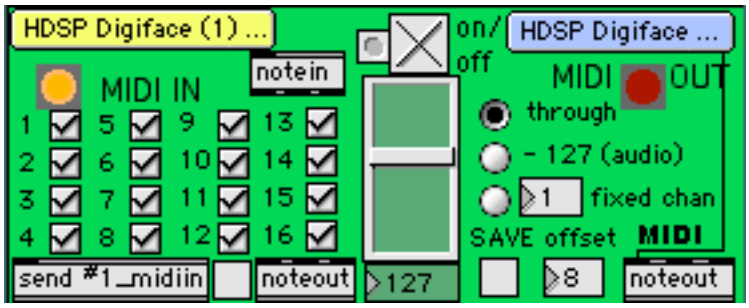
When you load an oracle in audio mode (see Load in Tutorial 4), e.g. from file MySession.or, there has to be a soundfile named Mysession.aiff in the same directory. This sound file will be automatically loaded in order to fill the audio buffer homologous to the oracle.

Tutorial 10. The Audio Control Panel

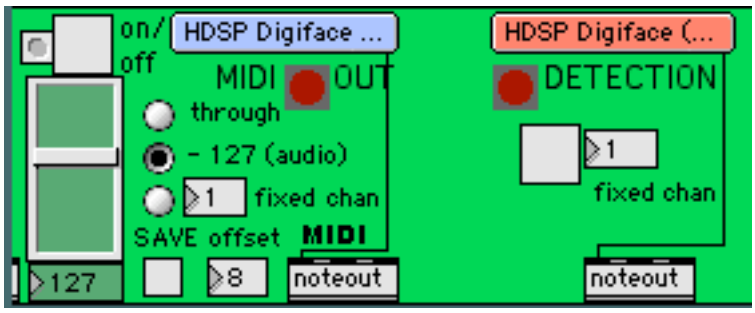


- ***SeqDir*** allows to directly play through the sound buffer, that is, to play back the recorded sequence as the performer played it. One sets the starting point and the length of the excerpt (in millisecs). It can be muted, sent to the cue bus, and the level numbox can be adjusted in dB.
- The vertical slider on the left indicates the progressive filling of the Main buffer
- The ***input*** meter is the input level of the main (recorded) signal on adc1
- The ***detect*** meter indicates the level of signal sent to to pitch detection (adc2). The white led just at the right blinks every time an event is recognized. Warning! A very low level signal is generally sufficient for the pitch detection. This is the use of the ***level*** value.
- ***Quality, window, proba:*** these are set automatically when you choose an audio instrument in the initialisation panel.
- ***Read file*** for learning from a soundfile instead of a real time input. Opens up a panel where you load, start and stop a soundfile. This is a nice way of testing the system with a good input quality.
- ***Listen MidiDtct*** sends the output of the pitch detection to the Midi Module. This is for debug purposes. A better way to listen to the Midi detection is to turn on the Detection toggle on the Midi control panel (see below).
- ***Listen Direct Signal*** sends the direct real time input from adc1 to the Sound Matrix. The vertical slider controls the level. It is advisable to let it on, and to select the output on the matrix.
- ***Yin, Stat:*** debug purposes.

Tutorial 11. The Midi Control Panels



The Midi panel when in Midi mode



The Midi Panel when in audio mode

The green Midi panel changes its appearance when switching from Midi mode to Audio mode. In both modes, one can set the Midi velocity using the vertical slider. The big toggle above the slider sets the Midi output on or off. One can notice in the above pictures that by default, the toggle is On in Midi mode and Off in Audio mode.

Midi mode:

- Midi In subpanel (left side)
  - Midi port selection menu
  - Midi in channel filters (toggles)
  - Noteout toggle (through) : sends the Midi input to the output.

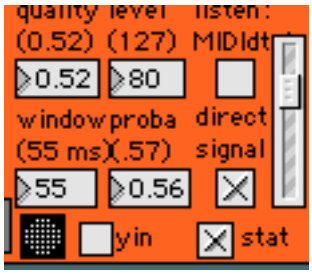
- Midi Out subpanel (right side)
  - Midi port selection menu
  - ***Through/-127/fixedc chan:*** affects the Midi channels of the OMax improvisation output. Through: same channels as the input. -127: not used in Midi mode. Fixed: user defined single channel.
  - ***Save:*** save the Midi input and Output happening in a session to a file. It has to be checked at the beginning of the session. At the end, upon unchecking, it will ask for a filename (.mid). In order not to create confusion between the input and output channels, there is an offset parameter. For example, if the input is on channel 1, 3, 5 and the offset is 8, the Midifile will contain the performer's input in channels 1, 3, 5, and the corresponding OMax improvisation on 9, 12, 14.

Audio mode:

In audio mode, two Midi flows can be generated. The output of the pitch detection (Detection, right panel) which corresponds to the performer's input, and the output of the OMax improvisation (Midi Out, left panel).
The controls have basically the same meaning than in Midi Mode. The default Midi Channel option is set to -127 because in audio, the event have a Midi channel >= 128. So 128 is really Midi channel 1. You shouldn't change this option except if you would like to set a fixed Midi channel (e.g. 13).
The Detection toggle allows to play in Midi the output of the pitch detector : the acoustic instrument is doubled by its Midi counterpart. The Midi output channel can be set in the numbox.

Tutorial 12. Tuning the Detection Parameters to your Needs

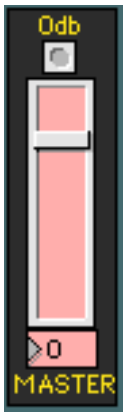
It is preferable to set the detection parameters before the start of the session. Any change of preset (marked by the OK button) empties the buffer. The presets correspond to existing instruments, and can often work with success with others.
However, the values of the detection parameters can be adjusted on the sound control panel.



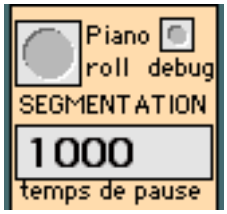
Generally speaking:

- The default values are a good starting point
- The window size (40-80ms) must be larger for slower and lower instruments
- The quality (0.35-0.90) must be lower for less harmonic (“dirtier”) sounds
- The probability (0.40-0.80) can be lower if the sounds are hard to detect.
- The level should be low enough to prevent the ambiant noises (including Omax own improvisation) to trigger the detection. The detection can be monitored on the black and white led.

Misc Global Controls



Master Volume (Midi and audio)



Threshold of silence duration in ms for detecting phrase boundaries.

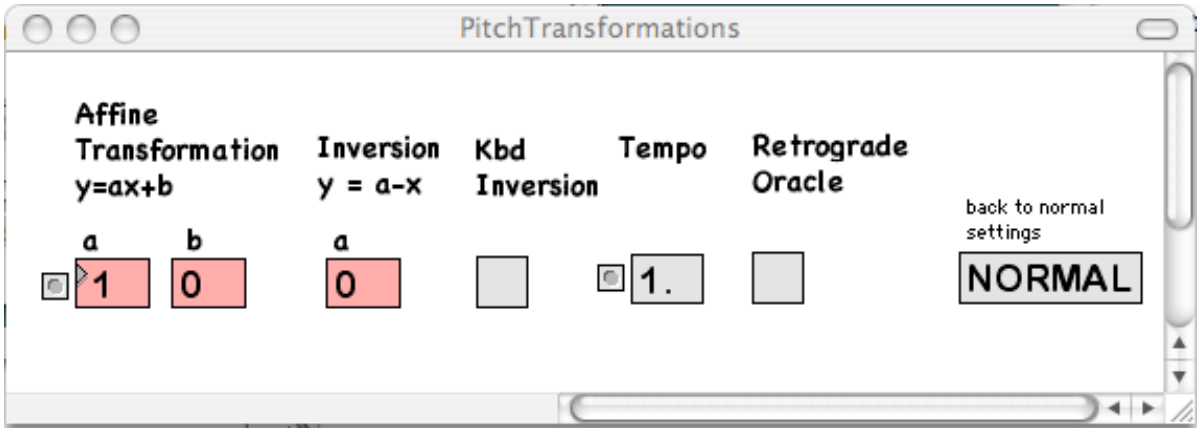
Note: the Pianoroll button is disabled

Helper Patches and Application

You will find a helper patch and a helper application in the OMax folder.

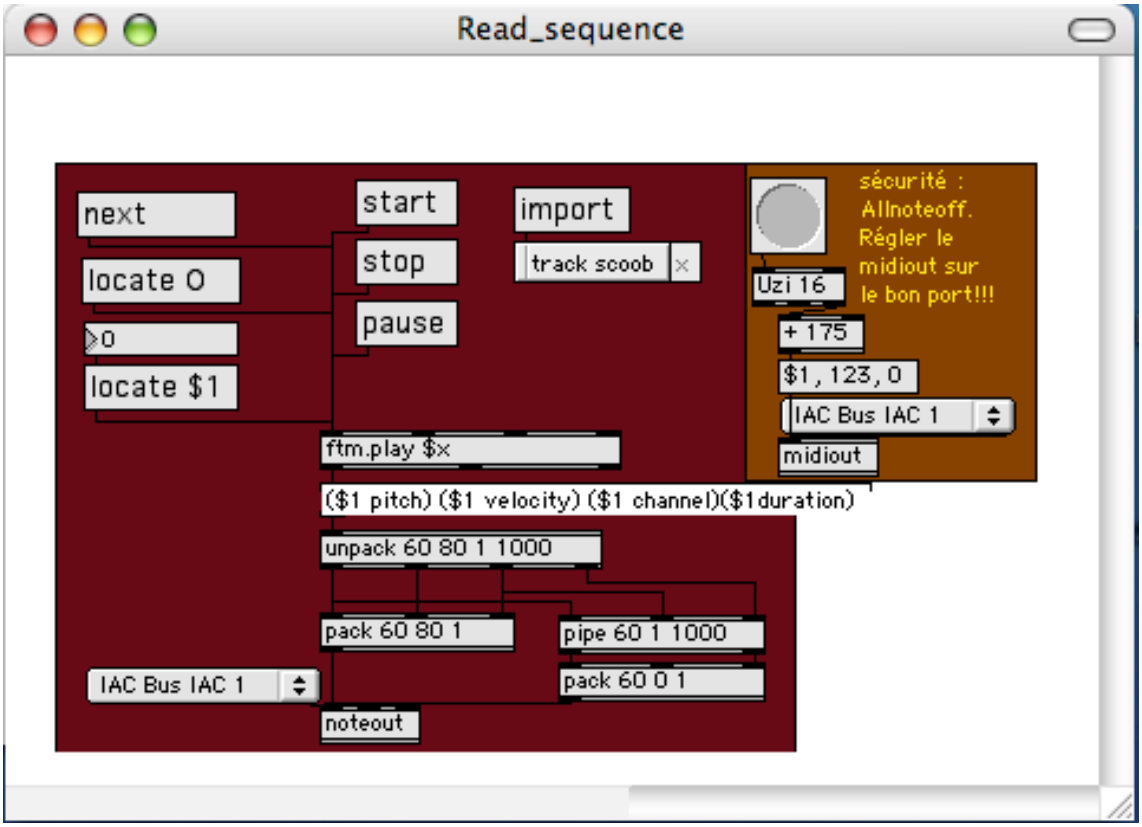
Using the patch **PitchTransformations** you will be able to perform on the fly transformations on the material generated by OMax:

- set-theoretical pitch transformation
    - Affine transform (a=2, 4, 8, 10 whole tone scales, a=3, 6, 9 diminished, a= 5, 7 permutation of the chromatic scale)
    - Inversion: mirror inversion on the chromatic scale. The inversion with parameter a = n is equal to the affine transform 11a+n
- Kbd inversion: maps notes that correspond to the center of the keyboard to the extremes and vice-versa
- Tempo: slow down or accelerate tempo (2 means twice as slow, 0.1 means 10 times faster)
- Retrograde: improvises the oracle backwards



These transformations are set for the Midi mode. The retrograde works in audio mode, and the tempo more or less.

The **Read_sequence** application lets you import a midifile (must be .mid, in lower case letters), start, pause, stop it, send it to a selected midi port. If you select the same midi port as input port in OMax, you will be able to « reimprovise » the MidiFile.
There is also a practical button for sending All-note-off to a Midi port, which is useful when OMax has been interrupted and your expander hangs on infinite notes.
This same All-note-off button can also be found in the Help patcher (“p utilities/help”) in the OmaxMax window.



Bibliography: OMax related Publications

Assayag, G. , Bloch, G. « Navigating the Oracle: a Heuristic Approach », ICMC 2007, Copenhagen, submitted.

Cont, S. Dubnov, G. Assayag « Anticipatory Model of Musical Style Imitation using Collaborative and Competitive Reinforcement Learning », Anticipatory Behavior in Adaptive Learning Systems, (Martin Butz and Olivier Sigaud and Gianluca Baldassarre, Berlin), 2007

E. Amiot, T. Noll, M. Andreatta, C. Agon, Fourier Oracles for Computer-Aided Improvisation », ICMC 2006, New Orleans, 2006

G. Assayag, G. Bloch, M. Chemillier « OMax-Ofon », Sound and Music Computing (SMC) 2006, Marseille, 2006

G. Assayag, G. Bloch, M. Chemillier « Improvisation et réinjection stylistiques », Le feed-back dans la création musicale contemporaine - Rencontres musicales pluri-disciplinaires, Lyon, 2006

G. Assayag, G. Bloch, M. Chemillier, A. Cont, S. Dubnov « OMax Brothers: a Dynamic Topology of Agents for Improvisation Learning », Workshop on Audio and Music Computing for Multimedia, ACM Multimedia 2006, Santa Barbara, 2006

Cont, S. Dubnov, G. Assayag « A framework for Anticipatory Machine Improvisation and Style Imitation », Anticipatory Behavior in Adaptive Learning Systems (ABiALS), Rome, 2006

Rueda, G. Assayag, S. Dubnov « A Concurrent Constraints Factor Oracle Model for Music Improvisation », XXXII Conferencia Latinoamericana de Informática CLEI 2006, Santiago, 2006

G. Assayag, S. Dubnov « Improvisation Planning and Jam Session Design using concepts of Sequence Variation and Flow Experience », Sound and Music Computing 2005, Salerno, 2005

Mondher, A. Gérard, M. Stephen, L. Olivier, C. Jean-Marc, R. Francis « De la théorie musicale à l'art de l'improvisation: Analyse des performances et modélisation musicale », ed. Mondher AYARI (DELATOUR-France, Paris), 2005

G. Assayag, S. Dubnov « Using Factor Oracles for machine Improvisation », Soft Computing, vol. 8, n° 9, Septembre, 2004

S. Dubnov, G. Assayag, O. Lartillot, G. Bejerano « Using Machine-Learning Methods for Musical Style Modeling », IEEE Computer, vol. 10, n° 38, Octobre, 2003

S. Dubnov, G. Assayag « Universal Prediction Applied to Stylistic Music Generation », Mathematics and Music. A Diderot Mathematical Forum, ed. Assayag, G., Feichtinger, H.G., Rodrigues, J.F. (Springer, Berlin), 2002

G. Assayag, G. Bejerano, S. Dubnov, O. Lartillot « Automatic modeling of musical style », 8èmes Journées d'Informatique Musicale, Bourges, 2001

O. Lartillot, S. Dubnov, G. Assayag, G. Bejerano « Automatic Modeling of Musical Style », International Computer Music Conference, La Havane, 2001

G. Assayag, S. Dubnov, O. Delerue « Guessing the Composer's Mind: Applying Universal Prediction to Musical Style », ICMC: International Computer Music Conference, Beijing, 1999

S. Dubnov G. Assayag « Universal Classification Applied to Musical Sequences », ICMC: International Computer Music Conference, Ann Arbor Michigan, 1998

J. Godet « Grammaires de substitution harmonique dans un improvisateur automatique », Paris 6 [DEA ATIAM], 2004

Laurier « Attributs multiples dans un improvisateur automatique », UPMC Paris 6 [DEA ATIAM], 2004
Seleborg « Interaction temps-réel/temps différé », DEA ATIAM [Mémoire de stage], 2004
N. Durand « Apprentissage du style musical et interaction sur deux échelles temporelles », Paris 6 [DEA ATIAM], 2003
Poirson « Simulations d'improvisations à l'aide d'un automate de facteurs et validation experimentale », UPMC [DEA ATIAM], 2002
O. Lartillot « Modélisation du style musical par apprentissage statique: Une application de la théorie de l'information à la musique », Paris 6/Ircam [DEA Atiam], 2000