OMax

The Software Improviser

© Martin Iartigues

Version **2**

Documentation by G. Assayag & G. Bloch, May 2008
Omax is designed and developed by *The OMax Brothers:* G. Assayag, G. Bloch, M. Chemillier in collaboration with Shlomo Dubnov
http://www.ircam.fr/ircam/equipes/repmus/OMax

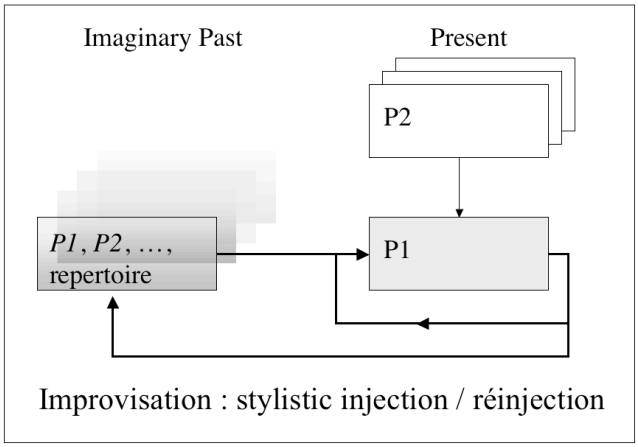
Introduction: OMax and the Stylistic Reinjection Process

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. The spectral version of OMax owes a lot to Shlomo Dubnov's pioneering work. OMax uses Yin~ the pitch tracker technology by Alain de Cheveigné (Max implementation by Norbert Schnell). The Spectral Mode uses FTM/Gabor library by the Ircam Real-Time Musical Interaction Team.

System Requirements

OMax 1.54 runs on Apple Macintosh PowerPC and Intel.
1GHz, 1GB G4 is a minimum for the audio version. Waveform display is CPU expensive and not recommended on PowerPC machines. Audio version uses memory and the largest RAM possible is recommended.
OMax needs Max 4.6 or Max RunTime 4.6.
Spectral OMax requires the FTM library.
OMax is multi-channel: if the audio hardware allows it, one can play back on 8 channels.
Several instances of OMax can be run simultaneously (on a powerful machine).

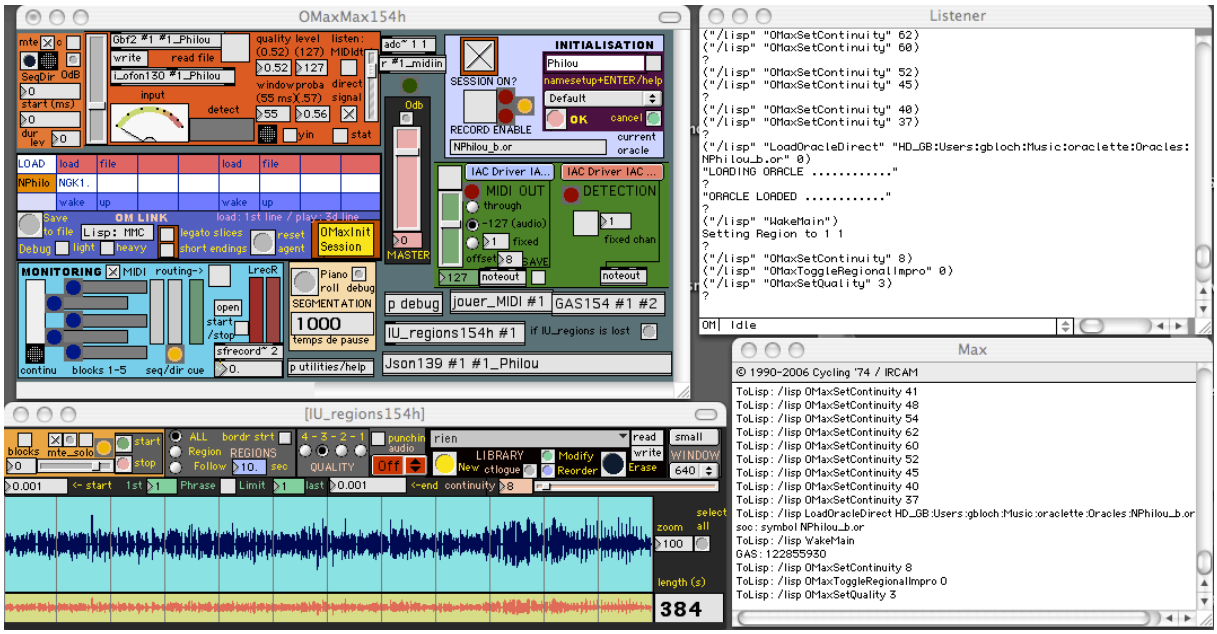
What is new in version 2

Several things have changed since OMax 1.0. Among them:

- Full Mac intel implementation in LispWorks
- Spectral audio version of the oracle, in which the events are defined by spectral descriptors (using FTMlib 2.2.4 or later)
- Display of the recorded waveform in audio modes
- Automation messages
- Suppression of the "cloning" of the main oracle
- Corrections of minor and less minor bugs (especially for the UI and the library)
- Video version of the oracle
- And, hopefully, not too many major bugs added….

Quick Start

OMax is constituted of two main programs: OmaxLisp and OmaxMax



The OmaxMax program is the same for both platforms, there is a just a single modification for communicating with Lisp. If you own max/MSP, you can modify the program yourself as explained later).
Launch OMaxLisp and OMaxMax. OMaxMax opens two windows, one for general commands and one improvisation commands and display. Keep the windows close to each other.

The Three Worlds of OMax: Midi, Event-audio and Spectral-audio

Before any use of OMax you have to decide if you want to use it in Midi mode or in Audio mode. Furthermore, Audio mode is declined in two ways: note-events or spectral descriptors. Therefore OMax is divided into three worlds

1. Midi mode: you 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).
2. Event-audio mode: you 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, you'll be also 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.
3. Spectral-audio mode: you have one 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.

How to choose a mode

The modes are chosen on the popup menu of the Initialisation panel.
Midi: choose MIDI_NoAudio and press OK.

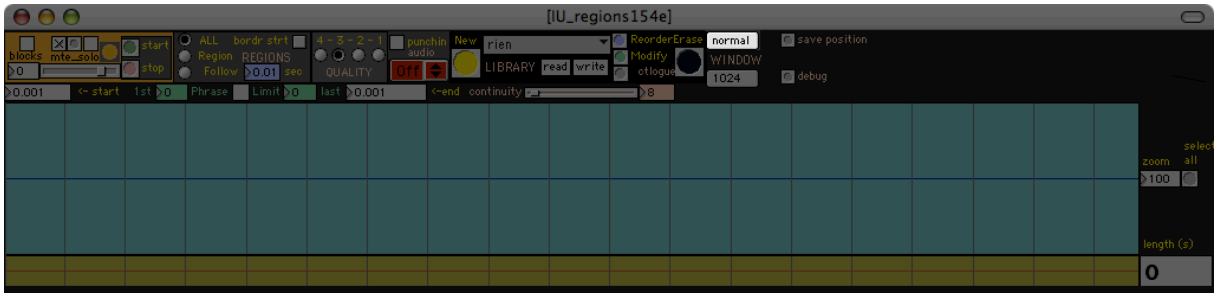
Event-audio: different presets correspond to internal settings carefully tuned for selected input instruments (Tenor Sax, Bflat Clarinet, Bass Clarinet, Voice, Percussion, Sound Poetry-spoken voice). The « Default » audio item is a good average setting to start with. The "pianobar" set is designed for the Moog pianobar, or any audio instrument equipped with its own Midi detector. The Bass Clarinet also works for double bass.
Spectral-audio: there are two choices, spectral_mfcc (Mel frequency cepstral coefficient) and lpc (Linear predictive coding). For the moment lpc gives better results. [Remainder: you NEED to have FTMLib installed in order to be able to use the spectral settings].

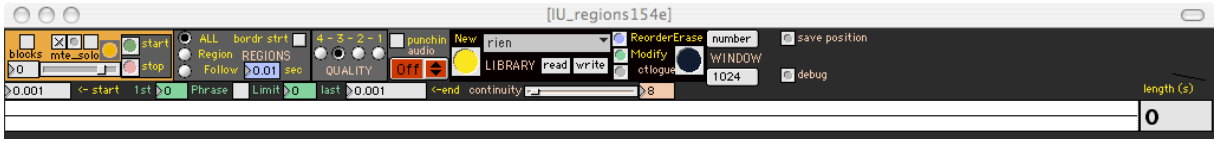
When switching from one mode to another, the Red top left panel actually changes: choosing a mode activates a script that modify the patch organization. Advice is given to do this once at the beginning of a session, just after loading the patch, an not to fool around switching modes while the patch is running. You can save-as your patch (inside the folder it came from) and it will remember its Midi-or-Audio configuration. So, for instance, 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

Setting up OMax for MIDI.

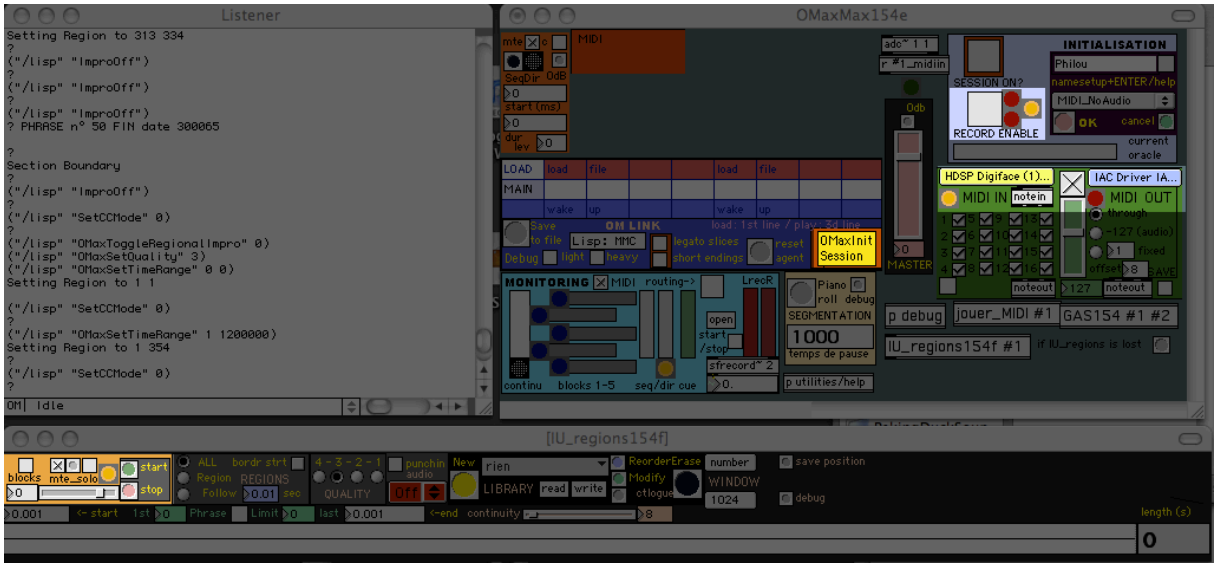
1. As said above, go into MIDI mode by Chosing MIDI_NoAudio in the popup menu of the Initialisation panel
2. Since, in MIDI, there is no waveform to be displayed, chose “number” in the upper right popup menu of the improvisation command window. A simple timeline is displayed.



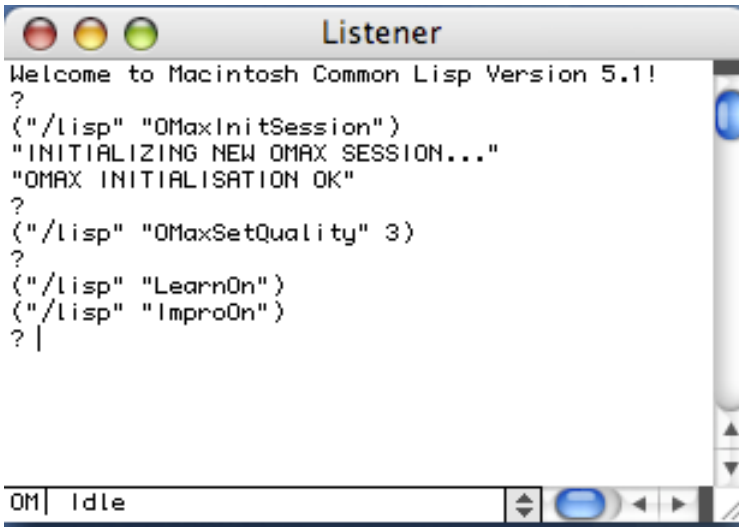


Starting the improvisation.

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



3. Use the Midi port menus to select your input and your output (warning: even if the correct ports are displayed, it is usually necessary to reselect them when restarting the program).
4. Press ***OMaxInitSession***. This will initiate the communication between OMaxLisp and OMaxMax. The OMaxLisp listener and OMaxMax will react as pictured below (the left window will a little bit differ in an Intel Mac)



.

5. Check the toggle box ***Record Enable***.
6. 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.
7. 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 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 without preventing 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) and even to a higher value for spectral improvisations (30-80 are decent values in spectral mode). 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) because OMax doesn't find jumps that match your exigence. 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. There is a debug mode to inform you about the attained rate of recombination that will be described later.

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) or directly on the waveform display (in audio oracles).
- ***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 current last seconds, following, as in a pursuit, 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). This Rslider can display a waveform when in audio mode. 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.

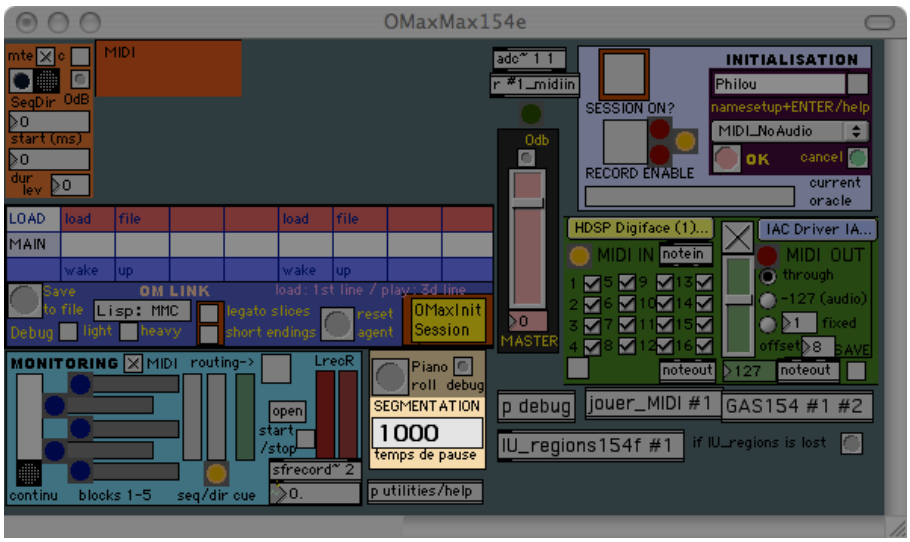
Regions selected in the rslider become active only when the ***Region*** radio button is pressed.

Two numboxes indicate the current region *start* and *end* time in seconds/milliseconds. The bigger numbox at the far right indicates the duration in seconds of the sequence learned so far.

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

Tutorial 3. More Improvisation Control: 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. This segmentation threshold can vary between 500 and 2000 ms and be modified on the segmentation panel:



If you check the toggle ***Phrase Limits*** in the improvisation panel, then your selection in the rslider will be automatically aligned with phrase boundaries. The two 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 by a simple click, 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 ***bordr strt*** (start on border). Actually OMax will always restart on the first event of the current selected region. 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 chosen region 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 ***bordr strt***.

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 a *Factor 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 recorded 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 (and improvise on them), it learns only in one unique Oracle, called **MAIN**.

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

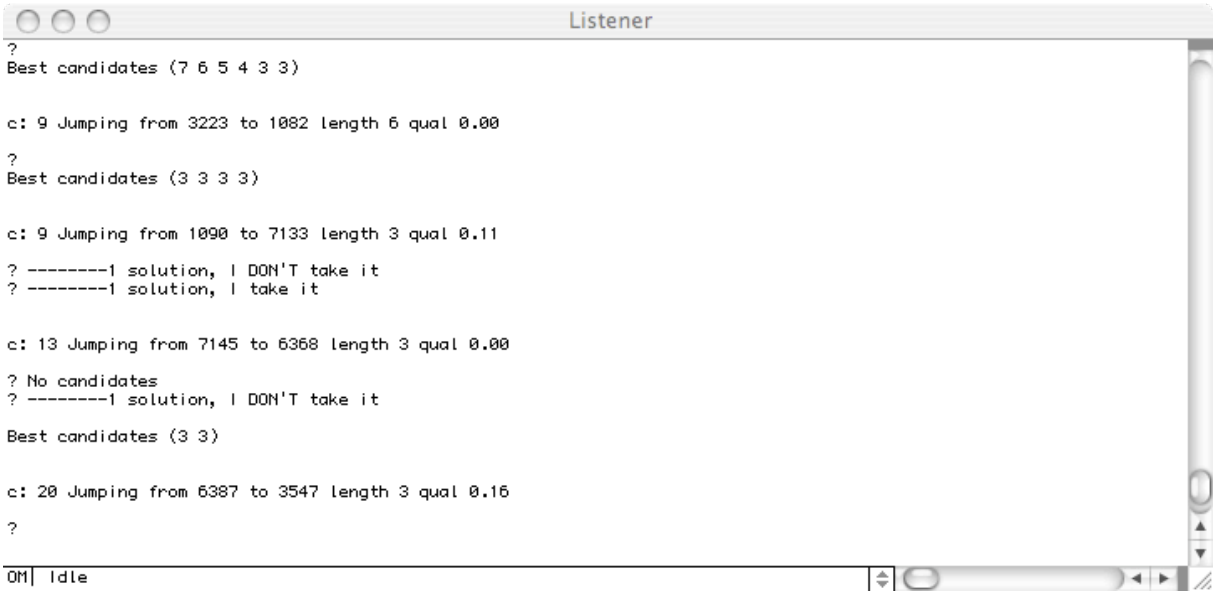
- ***Save to File*** saves the main oracle to a file (important convention: MyNameOracleFile.or)
- ***Load*** *(red buttons)* load an oracle saved from a previous session into a memory slot
- ***Wake up*** *(blue buttons)* the oracle loaded in the corresponding memory slot (including the Main Oracle on the left) becomes the active improvising oracle
- ***Load Main*** *(upper left 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 augment a past session with the contents of 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 beforehand
- You can have simultaneously at hand: a MAIN oracle and 8 loaded oracles among which you can switch using the blue ***wake-up*** buttons
- When 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 (if ***record enable*** is on).

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 loaded oracles memory slots. It clears only the MAIN (learning) Oracle. ***OMaxInitSession*** resets everything for a brand new session.
- ***Legato slices*** if not set, will play durations exactly as learned (i.e. if a chord is slightly arpeggiated, it will be played as is). If set it will quantify the duration (i.e. the chord will be played with simultaneous onsets and offsets). This affects only MIDI mode, not audio.
- ***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 improvisation 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 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 pursue 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 printing takes processing power (on PPC).

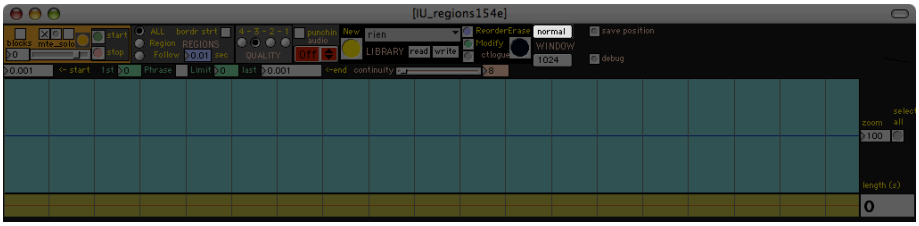
Tutorial 5. Going Audio

In audio mode, all you have learned in Tutorials 1-4 stays valid, except now you're learning directly from an audio signal, and OMax improvisations are played in audio using the very sound recorded from the performer.

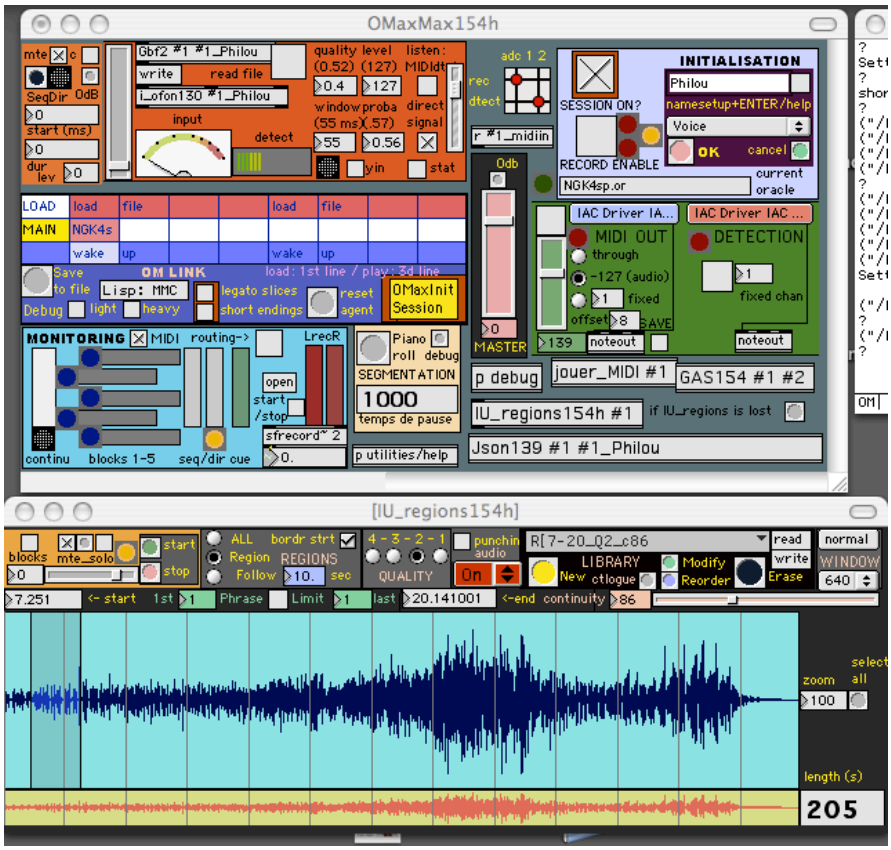
After opening the OmaxMax patch:

• select the default audio instrument in the Initialization menu, press OK. A new audio panel will appear.

• In the improvisation command window, select "normal", or any size according to the room available on your screen. A double waveform display appears. The top (blue) large waveform displays a current zoomed portion of the buffer. The small (yellow) always shows the whole recorded buffer. By selecting a portion in the yellow pane, it is zoomed into the blue pane. The ***all*** button redisplays the whole buffer in the blue pane. In order to select an improvization region, you have to click-drag in the blue pane, the yellow one being only for visual zooming purpose.



Let us review all the available parts of the interface:



• The top right blue is the initialisation process.
• The orange is the input panel, changing according to the setup in te blue.
• Between them is the input matrix, above the master.
•The green is the MIDI window
•The navy blue is the housekeeping window (Tut. 4)
• The square beige yellow is the segmentation window (Tut. 3)
• The hawaiian blue is the output monitoring window

• The variable size improvisation control window shows the waveform.
• The window also allows monitoring for continuous impro,control of the improvisation type (ALL, Region or Follow, Tut. 2), of the quality.

Check the audio on.

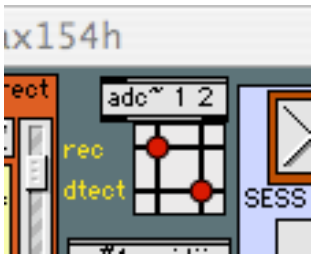
Check in the Max DSP options menu that every thing is set up for audio input output. By default you should have 2 inputs: adc1 for recording the performer's sound, adc2 for pitch detection.

Setting up the recording

Two sound inputs are needed: one for recording itself (*rec*) and the other for pitch – or spectral – detection (*dtect*).
A classical setup for e.g. a saxophon player would be:

- A good (best possible) aerial microphone on *rec* for recording a quality sound (this is the sound you will hear when OMax wil be improvising)
- A contact microphone on the instrument on *dtect* (pitch detection is better with a close take and you don't want the pitch detection canal to take the surrounding sound, including the OMax improvisation, because this would result in a feed-back process). For pitch detection, a cheap piezo pickup on the reed is perfect; for spectral audio, the pickup can be fixed on the bell.

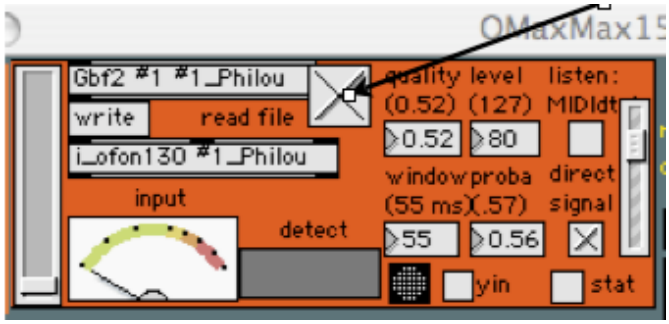
An alternative would be to use a single close mike (e.g. a mike clipped inside the bell) and send it to both *rec* and *dtect*. In this compromise, the sound for recording will be medium quality and the detection, hopefully, will not catch too much of the sound environment.



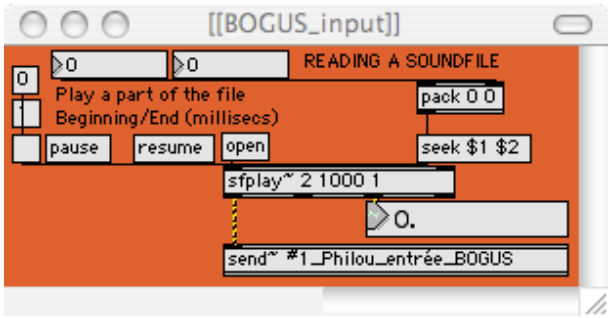
In any case you can chose wich adc~ goes to which input:
• except in the mode MIDI and pianobar, both inputs (*rec* and *dtect)* are needed.
• for pianobar mode, the *dtect* is disconnected. Only recording is needed since detection is made by the MIDI device
• for MIDI, there is no sound and all this is irrelevant.

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.

Using a File as Input



It is perfectly possible to use a recorded file as an input. Therefore, check the big toggle written read file: it opens a window poetically called BOGUS_input.



The use of this window is pretty straightforward:

• Use the ***open*** button to open a soundfile.
• You can play either by using the left toggle or the 1 and 0 buttons.
• You can play selected parts of the file by entering their dates in ms. The entering of the left date (beginning) forces the file to play
• You can pause, resume... during the reading of the file,

Finally…

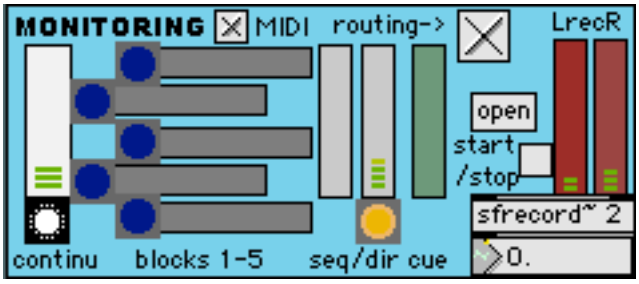
Proceed as in Tutorial 1: ***OMaxInitSession***, ***Record Enable***, and you're gone.

For more details, check the sections
Ref1: Audio Control Panel Reference
Ref3: Recording setup

Tutorial 6. Sound Output Control

Monitoring

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).

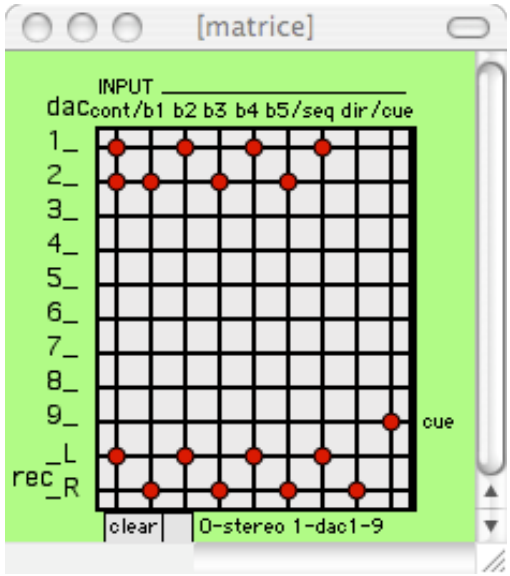


In order to understand the different signal-level meters in the *Monitoring* panel, here is the OMax terminology:

- ***Continu*** the continuous improvisation generated by the active Oracle
- ***Blocks*** the (up to 5) sequences played using the Block panel (see Tut. 9)
- ***Seq*** the recorded sequence is played straight using the SeqDir Panel (see further)
- ***Dir*** the direct input signal as captured on the input *rec*
- ***Rec*** the record-to-file channels
- ***Cue*** the signal is exclusively routed to a cue bus (for example headphones), and its normal output is muted. By default the output 9 is for the cue.

Routing

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).
By default, the direct signal (*dir*) is NOT sent to the sound system (as normal for an acoustic instrument).

Recording

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 input *dir* IS recorded, and push the toggle start/stop to start the recording. WARNING: the recording is quite demanding for the machine, on PPC.

Interlude 7. Audio Buffers

As previously said, there is almost no difference in manipulation between MIDI mode and Audio mode, except for a few more controls in the last one. However, going audio involves internal differences that have to be understood in order to comprehend the system's 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, loaded from files. One can switch from one to the other, making it the active improvising Oracle. Only the Main oracle learns from the real time input. One can also load a Main oracle from a file, in order to augment it by learning from the real time input (the new information is learned at the end of the one loaded).

Audio is in RAM

In audio mode, all this stays true. However, each oracle (Main or loaded) has an audio counterpart in 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, whereas buffers are just plain audio buffers, but they represent the same musical sequence: in particular, they have the same total duration.
Buffers are the result of recording the input audio signal on input *rec*, 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 time being, events boundaries, pitches, intensities or, in spectral mode, events boundaries and spectral descriptors).
In the audio mode, every operation involving an Oracle involves and audio buffer as well. For instance, 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,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 load features. It also depends on your machine and, even more, on the size of your RAM. In any case, it is recommanded that you do not learn ot improvize when you are loading files: you could get serious overload problems.

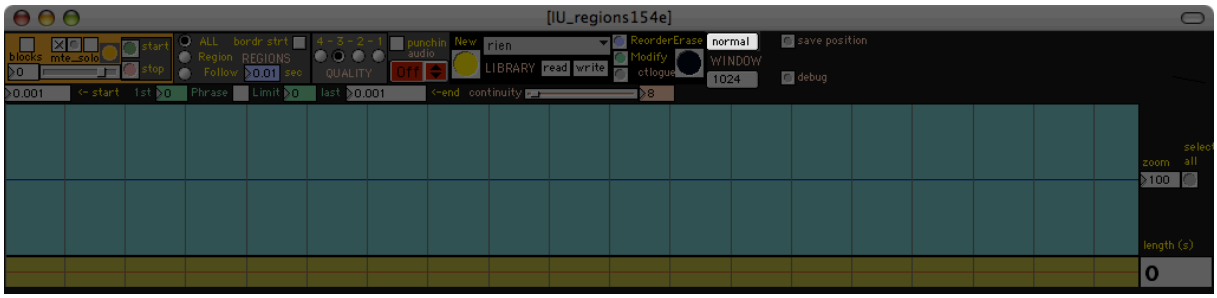
Saving / Loading in Audio Mode

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. WARNING: the ".or" termination is madatory. Both files have the exact same name with different endings. On Intel version, avoid special characters (specially accents) in filenames.
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.

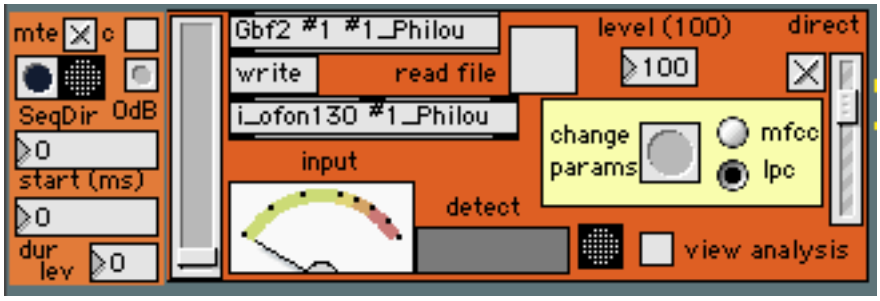
Tutorial 8. Going Spectral

In spectral mode, all you have learned in Tutorials 1-7 stays valid; you're now learning directly from a full audio signal spectrum, not from pitch detection, like in audio mode (Tut. 5). The incoming spectral frames are turned into vectors of spectral descriptors through the FTM/ Gabor Library (this is why you need to install FTM in order to take advantage of the spectral mode). Therefore there is no relevant MIDI information. However, like in audio mode, OMax improvisations use the same audio sound recorded from the performer.

After opening the OmaxMax patch:
• select SPECTRAL_lpc in the Initialization menu, press OK. A new orange spectral panel will appear.
• If the improvisation window is on "number", that is, shows a timeline, select "normal", or any size according to the room available on your screen. A double waveform display appears.



•This is the way the orange panel looks:



• This has common features with the audio panel (Tut. 5), notably the Gbf2 to check the waveform.
• It is possible to change parameters and even descriptors (going back and forth between mfcc and lpc) from this window. However, this should be done ONCE before the session.
• The detect ***level*** correspond to the signal sent to the descriptors. A smaller value will permit to have detection of silences and therefore "phrases" and stops in the recording process. Beware that by cutting too soon, the sound itself risks being strangely cut.

Check the audio on.

Check in the Max DSP options menu that every thing is set up for audio input output. By default you should have 2 inputs: adc1 for recording the performer's sound, adc2 for spectral analysis. Again a single mike input can be routed to *rec* and *dtect* if you can cope with feedback.

Spectral audio, quality and continuity

In MIDI and Audio modes, the events roughly corresponds to “notes”. This is actually the case for a monophonic instrument. Each detected note is an event. For polyphonic instruments like keyboards, events correspond more to points of attack (or release) of at least one note.
The spectral approach is radically different: like a sound-film, the sound is analysed in frames, up to more than 80 per seconds (fortunately for the machine, OMax aggregates similar contiguous frames into “super-frames” and therefore does rarely have that many).
The values for the continuity factor are thus very different in spectral setup. Continuity of 8 or 20 notes from an original recording can be many. 8 or 20 spectral frames generally are not even a second of sound.
A good rule of thumb is to think that spectral continuities are 8 to 10 times more than “event” continuity values. Of course, there no direct relationship: one long note with vibrato could be one event and many frames, while a very fast flute trill could generate almost as many events as frames.
What is said for continuity goes of course for quality: the quality factor is the context quality ; 4 notes in common are a lot. 4 frames too, actually, because it means not only a timbre, but a timbral evolution on something like 100 milliseconds. However, one should prefer (very) good qualities in spectral environement. Where 3 is an acceptable quality in event-mode, 2 is better in Spectral mode.

So ***quality*** = 2 and ***continuity*** = 60 are good values to start with.

Setting up Recording

As in Tutorial 5.

Input from File

As in Tutorial 5.

Finally...

Proceed as in Tutorial 1: ***OMaxInitSession***, ***Record Enable***, and you’re gone. Take care,in spectral mode the recording generally starts as soon as the ***Record Enable*** button is set.

For more details, check the sections

Ref4: Spectral Control Panel Reference
Ref3: Recording setup

Tutorial 9. Block Impros and Loops

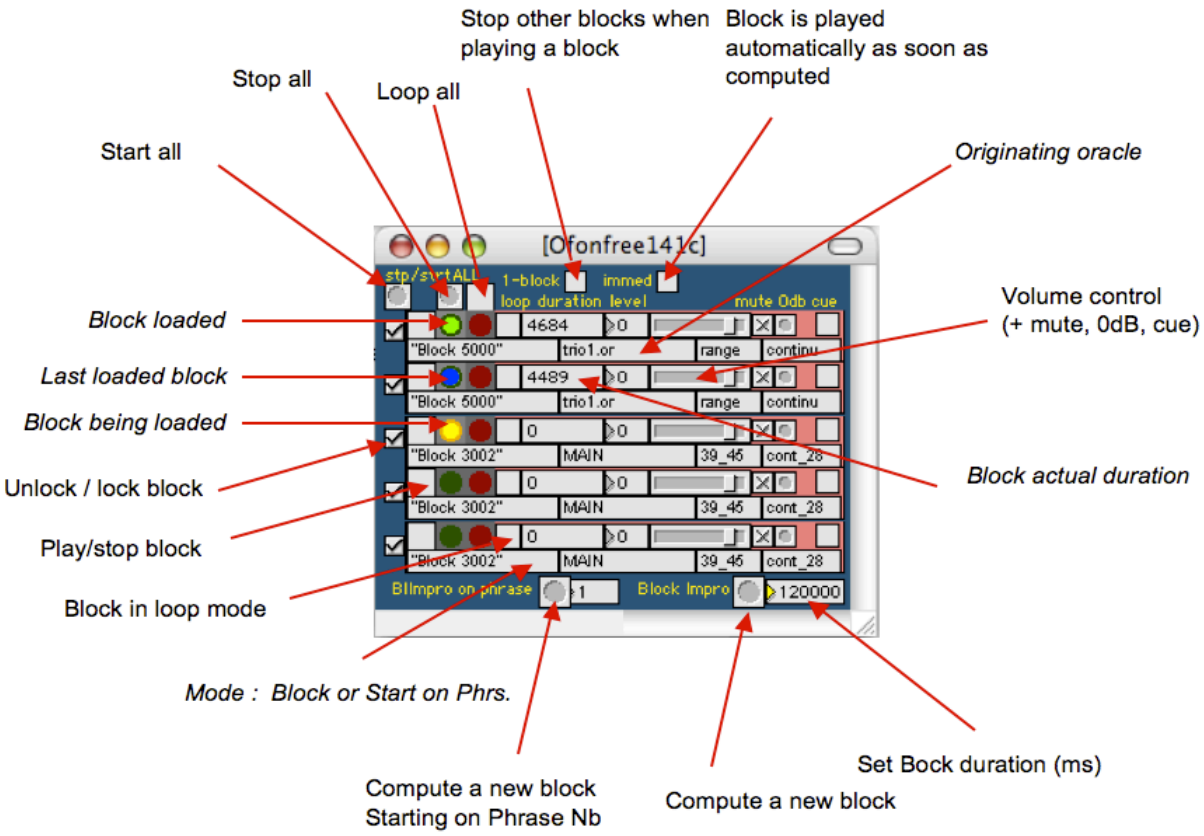
By checking the toggle **block**, you open a new window with a 5 tracks mixer-like interface.

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. Constant pattern can then be kept in an otherwise always evolving system. Due to the concurrent agent architecture in OMaxLisp, computing a block does not interrupt other processes going on, such as Main "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 to be sure to be able to replay it later.
When computing/loading a block, you shouldn't try to compute/load another block before the process is completed. A yellow light indicates that computing/loading process is taking place, followed by a blue light when the process is completed. A green light indicates a block previously loaded. So the blue light aways denote the last computed block.

To compute/load a block, type a duration in msecs in the lower-right numbox and press the ***Block Impro*** button. A yellow light appears, wait until it turns blue. 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, with the current active Oracle set in mode *All*.

By using the loop mode and playing a polyphony of blocks, it is possible to get an extremely dense texture., It is possible to ***loop*** one block, or all of them, to start a block immediately after it is computed ***(immed)***, to allow only one block to play at a time ***(1-block)***. Of course each block can have its individual level adjusted, can be muted or send to a cue chanel (to be listenend silently in headphones for instance)

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



Tutorial 10. Presets for Improvisation Control

The Improvisation Control window has a librarian module 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, that is: Quality, Region mode (***All***, ***Region*** or ***Follow***), Region boundaries (and whether they are Phrase or Time limited), possibly follow time (in ms), ***Bordr Strt*** (0 or 1) and the ***continuity*** value. 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). This is the role of the buttons ***write*** and ***read***.
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 current situation (button ***Modify***) or erased (***Erase***).
The first setup of the menu [rien] has no action on the user interface.

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: A (=ALL)/ R (=Region) / FL (=Follow)
- Start on a border: [(=yes) : (=no)
- Phrase limit: Ph (=yes) (nothing = no)
- Limits for ALL and Regions: XX-YY in seconds or in phrase numbers
- Time for FOLLOW: XXms
- Quality: Q1, Q2, Q3 or Q4
- Continuity: 1 to 248

Examples:

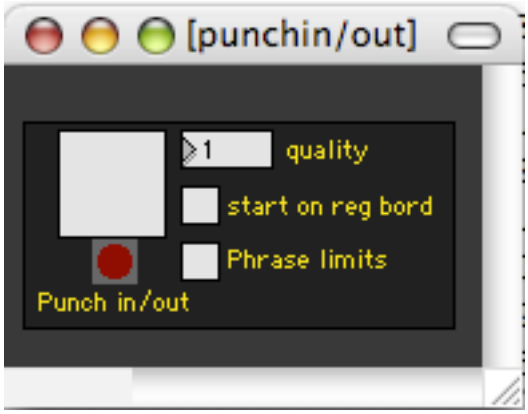
- R:43-54_Q3_c25 (Mode Region, from 43349 to 54289 millisec in the sequence, quality 3, continuity 25)
- A[Ph2-4_Q2_c8 (mode ALL, Start region Phrases 2 to 4, start on phrase begining, quality 2, continuity 8)
- FL16235ms_Q3_c10 (mode Follow, follow time = 16235 msecs, quality 3 and continuity 10)

Notes on the use of the Library:

The Library is very much Oracle-dependant. Any setup may be irrelevant for a given oracle: for instance, the phrase or time borders could not exist in the given sequence. In general, OMaxLisp switches to a default value when the borders asked for are impossible to reach.

Punchin / punchout

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.

Ref1: Audio Control Panel Reference

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 the *rec* channel (usually adc1)
- The ***detect*** meter indicates the level of signal sent to to pitch detection (or spectral analysis) from the *dtect* channel (usually 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 what the ***level*** value is for (see also Ref3)
- ***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 *rec* channel (usually adc1) to the Sound Matrix. The vertical slider controls the level. It is advisable to let it on, and to control this directly on the Sound matrix.
- ***Yin, Stat:*** debug and level purposes (see also Ref3).

Pitch Detection

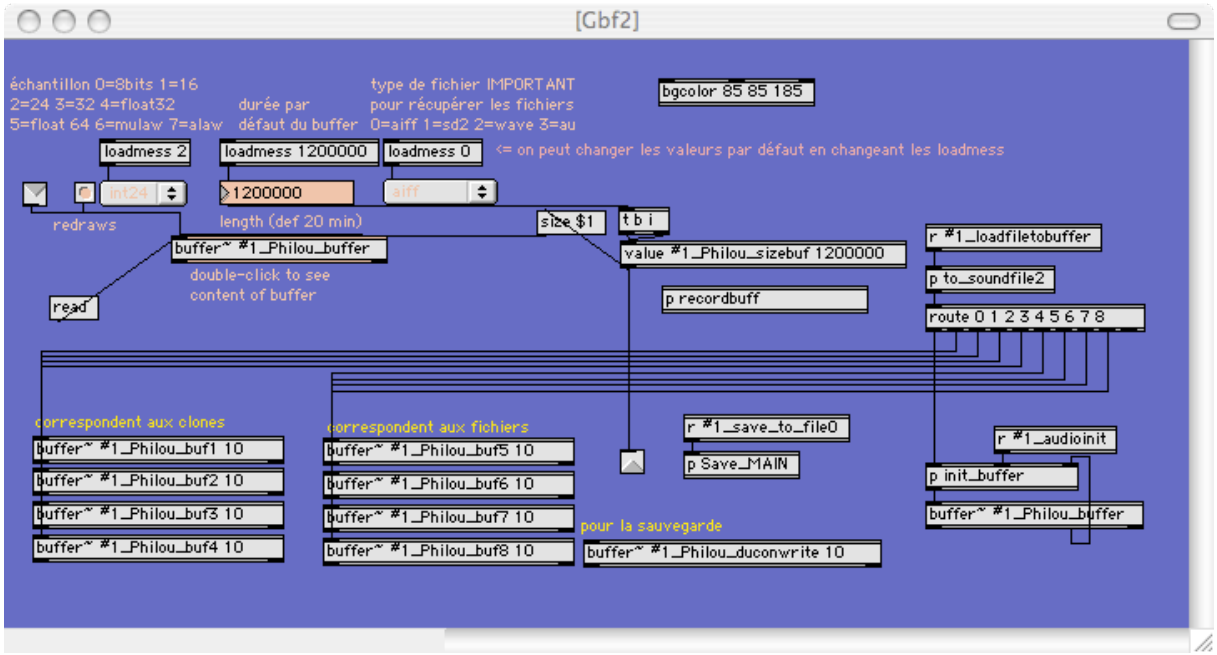
In Audio mode, like in other modes, it is preferable to set the general parameters before the start of the session. Any change of preset (marked by the OK button) empties the buffer. The presets correspond to existing instruments, and can often work with success with others.
However, the values of the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

Generally speaking:

- The default values are a good starting point
- Some of these values, notably Sound-Poetry and Percussion, work for extremely compact and dirty sound, working more as spectral enveloppes than real pitch detectors
- 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.

Buffers

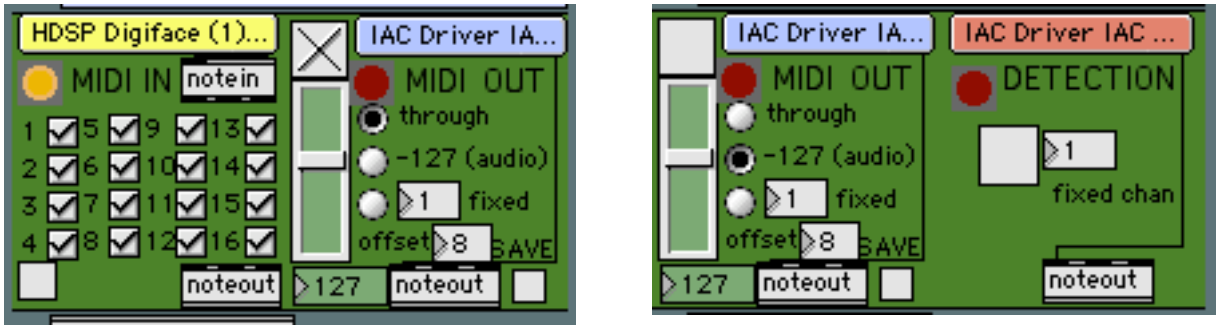
The buffers and their content are in the subpatch Gbf2 at the top of the panel. By clicking on it, one gets acces to the content of the buffers.



At the beginning of the session, the default values for the length of the main buffer (20 minutes), the sample size (24bits) and the format (aiff) can be changed.
By double clicking in the buffer object, you can verify that the buffers acctually contain sound (if not, check that Audio is on).

Ref2: MIDI Control Panel Reference

MIDI panel in MIDI mode... and 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, bottom left) : 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 the MIDI output of the pitch detector to be played: the acoustic instrument is doubled by its MIDI counterpart. The MIDI output channel is set in the numbox.

Spectral mode
MIDI flow is irrelevant and should not be used. Therefore the MIDI window disappears.

Ref3: Recording Setup Reference

Principles

There are two sound inputs to OMax audio. By default they are patched to adc1 and adc2

1. Recording itself (*rec*)
2. pitch – or spectral – detection (*dtect*)

Sending both inputs to the right place

• Except in the mode MIDI and Pianobar, you need an actual audio input (adc 1 or 2) to be plugged to *rec* AND one to *detect*
• in pianobar mode, *dtect* is disconnected. Since detection is made by the device attached to the instrument, only recording is needed
• for MIDI, there is no sound and all this is irrelevant.

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

- A (very) good aerial microphone on *rec* for recording a quality sound (the sound one hears when OMax is improvising)
- A contact microphone on the instrument on *dtect* (pitch detection is better with a close take and you don't want the pitch detection canal to take the surrounding sound, including the OMax improvisation, because this would result in a feed-back process). For pitch detection, a cheap piezzo pickup on the reed is perfect; for spectral audio, the pickup can be fixed on the bell.

An alternative could be to use a single mike very close to the instrument (e.g. a mike clipped inside the bell) and send it to both *rec* and *dtect*. In this case, the sound for recording could be OK 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.

General recording Advice

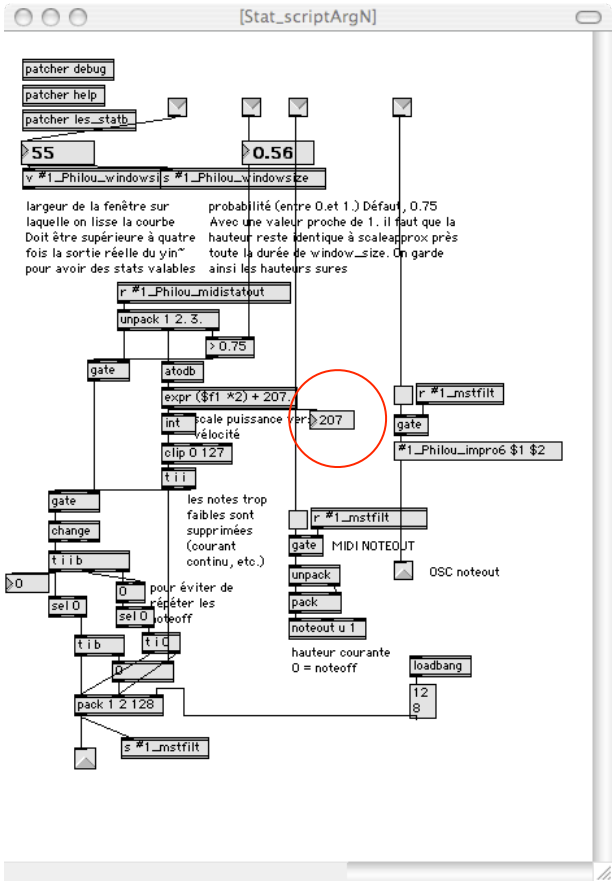
For recording, best is the rule
The best mike in the best position close enough to the instrument is the best solution.

But for pitch and spectral detection, low-fi is the rule
Curious as it may be, a short bandwith is enough (and actually better) for a good pitch detection AND for spectral descriptors extraction. This should not be curious: although the recordings are terrible, we are perfectly aware of the quality of Caruso's voice or Louis Armstrong's trumpet on their recording. The information given by a better bandwith (although extremely nice when listening) is as enormous as superfluous, making a hard time for the analysis process. This is the reason why a very cheap contact microphone will generally give better results than a good one.

Adjusting the detection/spectral level

Audio mode

The detection level for the *dtect* signal is very important. It can be set in the right part of the Audio control panel. WARNING: the level is set in the number box. The slider on the right correspond to the level of the direct signal eventually sent to the output (if the output matrix is on for direct sound)



By crossing the ***stat*** toggle, a window opens where you can control the levels.

This is the windows where the statistics for pitch detection are made.

The most important number is the one in the center, written 207 on the figure. This number roughly corresponds to a MIDI velocity, that is:

- *piano* sounds should be between 10 and 50 (40 is a good value in case you risk feedback)
- *fortissimo* sounds should be between 120 and 140 (but this is less important).
- 207 is a special value corresponding to NOTEOFF

So the best way to set the level is to play a fortissimo sound and set the level to have something between 120 and 140, then to adjust the level to get something like 40 in *pianissimo*.

Finally, you should check that when OMax plays fortissimo, it does not trigger any pitch detection. If it is not the case, lower the level. If trouble persists, use a contact microphone for the detection.

Spectral mode

In spectral mode, the recording is almost ALWAYS on, as soon as you push the ***Record Enable*** button. However, you can get phrases and even stopping with a very silent contact mike, but you should not count on it too much. To stop recording, putting ***Record Enable*** to zero is the best solution. However, you can also use the ***level*** number in order to get phrases and to stop the recording when nothing happens. It is advisable not to go too low: there is the risk of getting unpleasant cuts into the sound events.

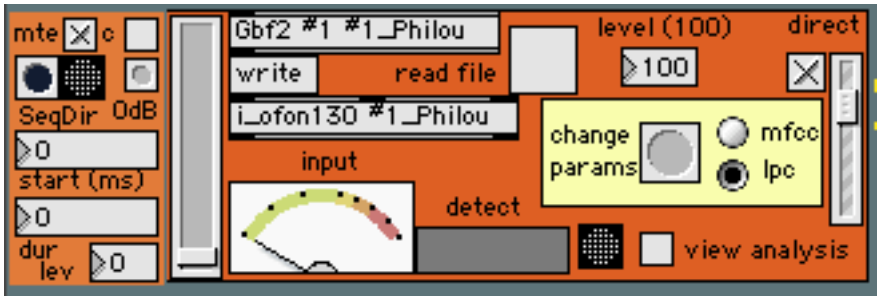
Ref4: Spectral Control Panel Reference

Spectral Oracle Basics

The spectral oracle uses what is called “spectral descriptors”. These are simple descriptions of the sound spectrum. They are calculated by “frame”. We have two ways of transforming these in oracle events: quantizing them, and, if the quantification of several successives frames gives the same result, we take them as one single longer frame. In any case, since the Oracle works with an alphabet, we must be able to transfrom these descriptors values in disctinct

equivalence classes. Quantification is one way to get these classes, and also allows to get less values for the alphabet: a spectral descriptor consisting of 20 coefficients varying between 0 and 9 would create an alphabet of 10^{20} possibilities. Chances to get repeated patterns be minimal! Another way of getting a shorter alphabet is, of course, to start with smaller descriptors, that is, with less coefficients. The whole process of getting equivalence classes with spectral description is still a very active field of research. Our use of "crude" quantification and of a small number of coefficient is a very basic method, although relatively efficient.
The two descriptors now implemented are *mel frequency cepstrum coefficients* ***(mfcc)*** and *linear prediction coding* ***(lpc)***. The quality of the results depends on the music played, of course, but also on the choice of the coefficient and of the way they are quantized. One must be aware that the idea behind the oracle is to find repetitions (only possible with a limited number of possibilities) such that these repetitions must sound close enough that one can be taken for the other.
More details on the spectral descriptors is available in the bibliography.

Control panel

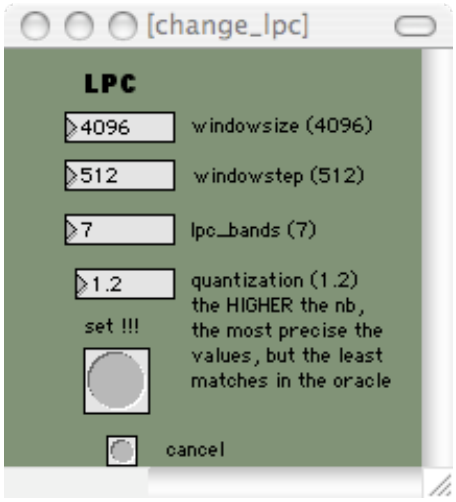


- ***SeqDir*** allows to directly play through the sound buffer, i. e. to play back the recorded sequence as the performer played it. One sets the starting point and the length of the excerpt (in millisecs). Mute, Cue bus send and Level in dB can also be adjusted
- The ***read file*** button opens a window allowing to read from a file (see Tut. 5)
- The ***level*** number decides of the level for the sound going to the spectral analysis. It should not be confused with the right slider, which decides what level of the direct sound is be sent to the sound system for possible amplification.
- by default, the ***direct*** button is checked and the incoming sound is sent to the output matrix (Tut 6). However, in this matrix NO output is selected. To actually amplify the incoming sound, both the ***direct*** button must be on and an output chosen on the output matrix.
- The ***Gbf2*** subpatch allows to check the contents of the buffers and change the format of the main buffer (see Ref1)
- The change parameters allow to change the values of the spectral descriptors. Once again, it is strongly advised to do it before recording a session. Before, however, you can play with the values of the descriptors, even change of descriptor type with the radio buttons.

Changing Spectral Parameters

For the moment, the best result we have are with the default parameters on lpc. However, for very continuous sound (like some electronic sounds) mfcc are more efficient.

lpc



Parameters value:

• ***windowsize***: as the name tells, the size of the window on which the analysis is made. At 44.1kHz, a window of 4096 corresponds to 93 milliseconds and covers all possible frequencies.

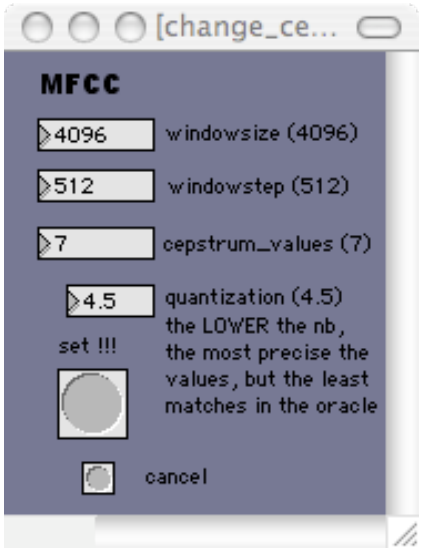
• ***windowstep***. More than a spectrum is calculated by windowsize. Because of the use of windowing techniques, most of the useful information is in the middle of the window. Here, a spectrum is calculated every 512 samples (12 msec). A small value is more demanding to the computer but permits a better time precision.

• ***lpc_bands***. Given the characteristics of lpc, a uneven value is preferable. Small means a cruder approximation of the spectrum. If you get very strange recombinations and have many of them, try 9 or 11

- ***quantization***. It is a MULTIPLYING factor (in mfcc it is a dividing factor). The result will be more precise with a higher value… and the spectra will be more differentiated (you will get less recombinations with, normally, a better resemblance)
- ***set!!!*** or ***cancel*** nothing is done until one of these two buttons is banged.

The default values are a good start. It is very easy to get values that will not give any result : 15 bands and a quantization factor of 5 will give as many letters as frames and will result in a total absence of recombination.

mfcc



Parameters value:

• ***windowsize***: as for lpc.

• ***windowstep***. as for lpc.

• ***lpc_bands***. 7 is a relatively low number for mfcc. A higher number could be tried (8 to 11)

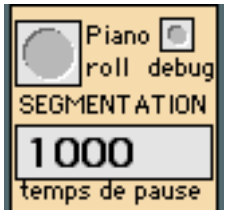
•***quantization***. It is a DIVIDING factor (in lpc it is the opposite). The result will be more precise with a lower value; the spectra will be more differentiated (you will get less recombinations with, normally, a better resemblance)

•***set!!!*** or ***cancel*** nothing is done until one of these two buttons is banged.

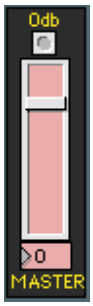
The default values are a good start. A interesting procedure is to compare two oracles made with the same recorded file with slightly different values.

Ref5: Miscellaneous

Global Controls

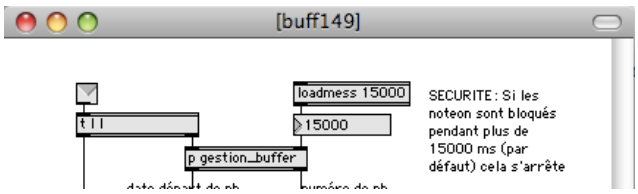


Threshold of silence duration in ms for detecting phrase boundaries
Note: the Pianoroll button is disabled

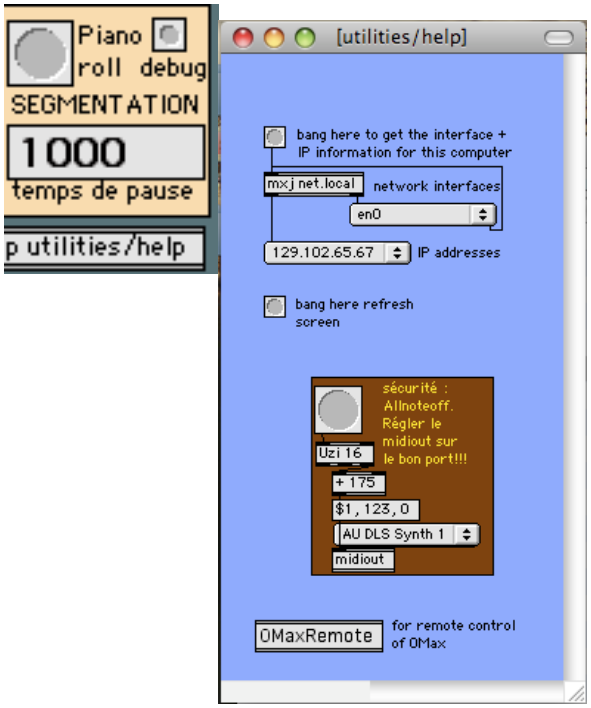


Master Volume
(Midi and Audio)

Utilities



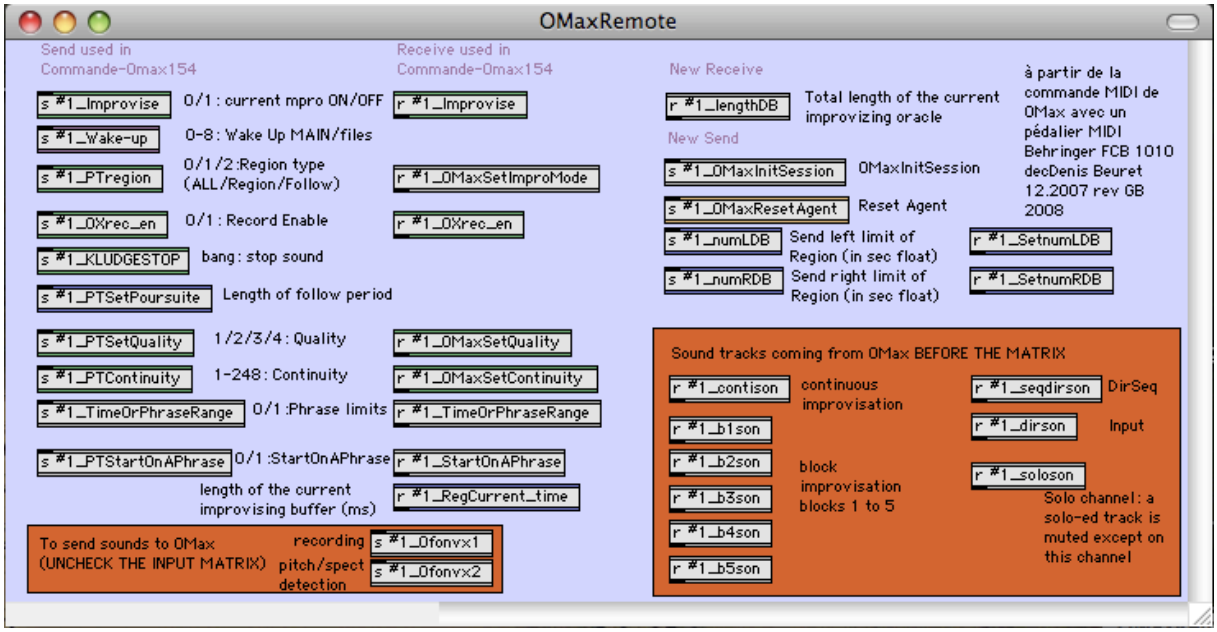
Clicking on the *debug* button in the segmentation panel opens up this subpatch where you can set the midi input note-off segmentation threshold (default 15000 msec). It means that if no note-off has occured in the midi input to stop note-on that arrived more than 15secs before, and no other, shorter note is pending, then the missing note-off are generated. This is to give an end to 'unfinite chords' in the midi input, a phenomenon which occurs with unperfect midi-devices (such as the Moog Pianobar or Midi Guitars) when note-off are eaten in the transmission and the performer stops playing for a while. You can change this value.



The patcher *utilities* opens up a window where you find the (very useful in Midi Mode) all notes off panic button, a utility that informs you about your IP address and a patcher that opens the OMaxRemote automation messages helper)

Automation Messages

This help patch documents the OMax automation messages. The red background contains signal messages (in and out) the blue one control messages (in and out). Using these messages you can take full control of OMax using your own interfaces or external devices. Do not close the OMax windows, just hide them if you don't need them. For a single OMax utilization just copy these messages to your patch. For a multi OMax utilization, the parameter #1 must be changed to reflect the parameter sent to each instance of OMax (see Helper patches below, MultiOMax), e.g. 's toto_improvise' or 's titi_PTRegion 0'.



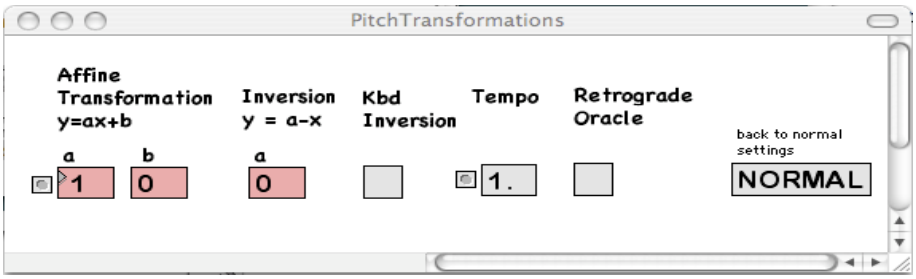
Helper Patches

You will find helper patches in the OMax folder.

PitchTransformations

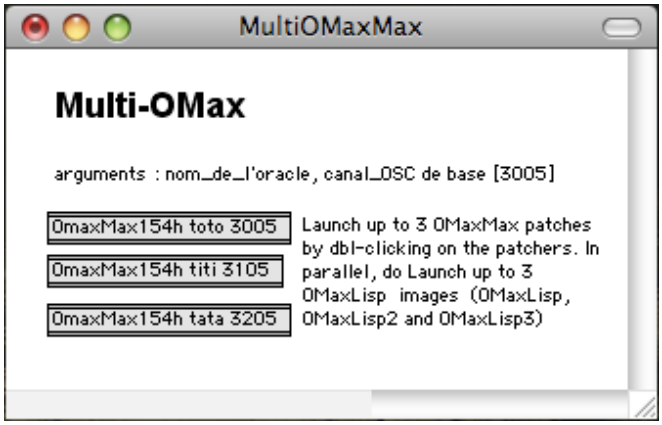
With this patch, you will be able to perform on the fly transformations on the material generated by OMax:

- set-theoretical pitch transformation (MIDI mode only)
    - 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 (MIDI mode only)
- Tempo: slow down or accelerate tempo (2 means twice as slow, 0.1 means 10 times faster)
- Retrograde: improvises the oracle backwards (works also in audio mode)



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

MultiOmaxMax



You can launch up to three OMax systems hat will operate in parallel with independent inputs and outputs but without any interaction one with another. In order to proceed, open the helper patch *MultiOMaxMax* then double-click on any subset of the three patchers. The corresponding OMaxMax patches will open up. On the lisp side, you have to load the corresponding OMaxLisp images. These are numbered (OMaxLisp, OMaxLisp2 and OMaxLisp3) and correspond to the three OMaxMax patches in the order from top to bottom. The OSC port numbers in the patchers arguments are mandatory since these are the ones recognized by the corresponding Lisp Images.

Bibliography: OMax related Publications

Bloch, G., Dubnov, S., Assayag, G., «Introducing Video Features and Spectral Descriptors in the Omax Improvisation System », Proceedings of the ICMC'08, Int. Comp. Music Assoc., Belfast 2007 (to appear).

Assayag, G. , Bloch, G. « Navigating the Oracle: a Heuristic Approach », Proceedings of the ICMC'07, Int. Comp. Music Assoc., Copenhagen 2007.

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 feedback dans la création musicale contemporaine - Rencontres musicales pluridisciplinaires, 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 (ABi-ALS), 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 expérimentale », 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