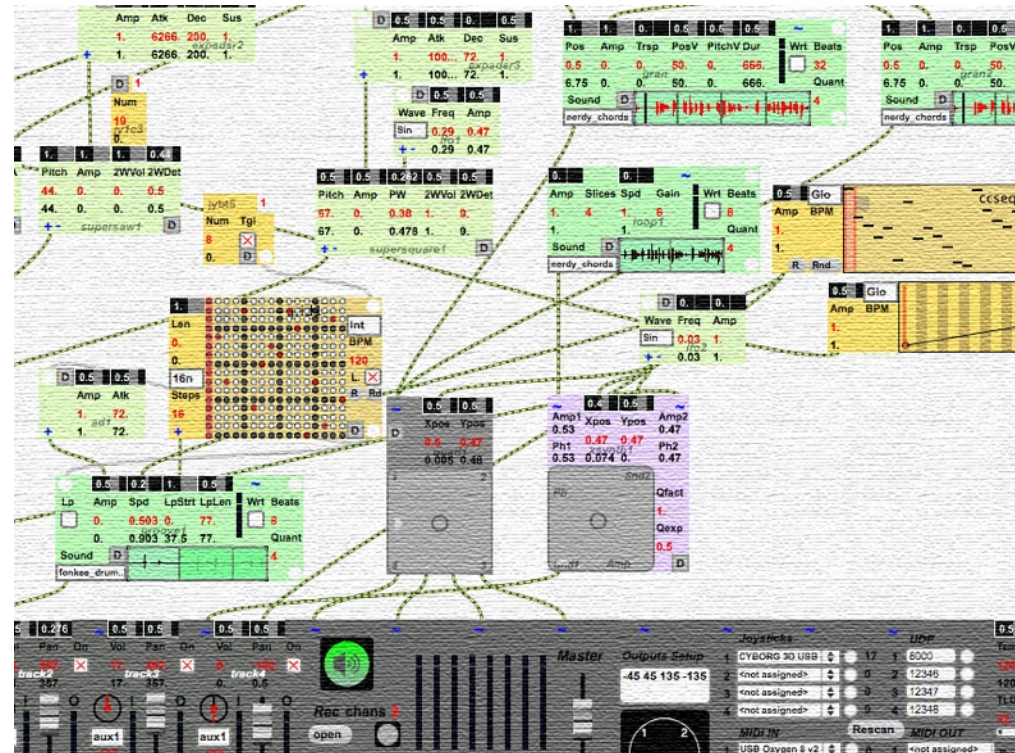


# Najo Modular Interface v2.1

## Quick Start Modules reference



# Najo Modular Interface 2

Concepts and programming : Jean Lochard

[jean.lochard@ircam.fr](mailto:jean.lochard@ircam.fr)

Externals objects used in NMI :

sogs~, psych~, supervp.scrub~, supervp.trans~, yin~ by Norbert Schnell

SuperVP engine by Axel Roebel

irc.shell, spat.oper, spat.spat~ by Thibaut Carpentier

moogfilter~ by Thomas Hélie, Jean Lochard, Thibaut Carpentier

najo.multipan by Jean Lochard

« The jimmies » by Zack Settel

© Ircam 2014

---

# Najo Modular Interface 2

Najo Modular Interface is a modular environnement for Max, build for live performance or sound design in studio. Its clear interfaces make it also a great tool for teaching synthesis and give a simple access to more complex Ircam technologies.

This "Quick Start" manual discribes how to install the software, how to create a new project and explains the main concepts you have to know when manipulating the NMI interface.

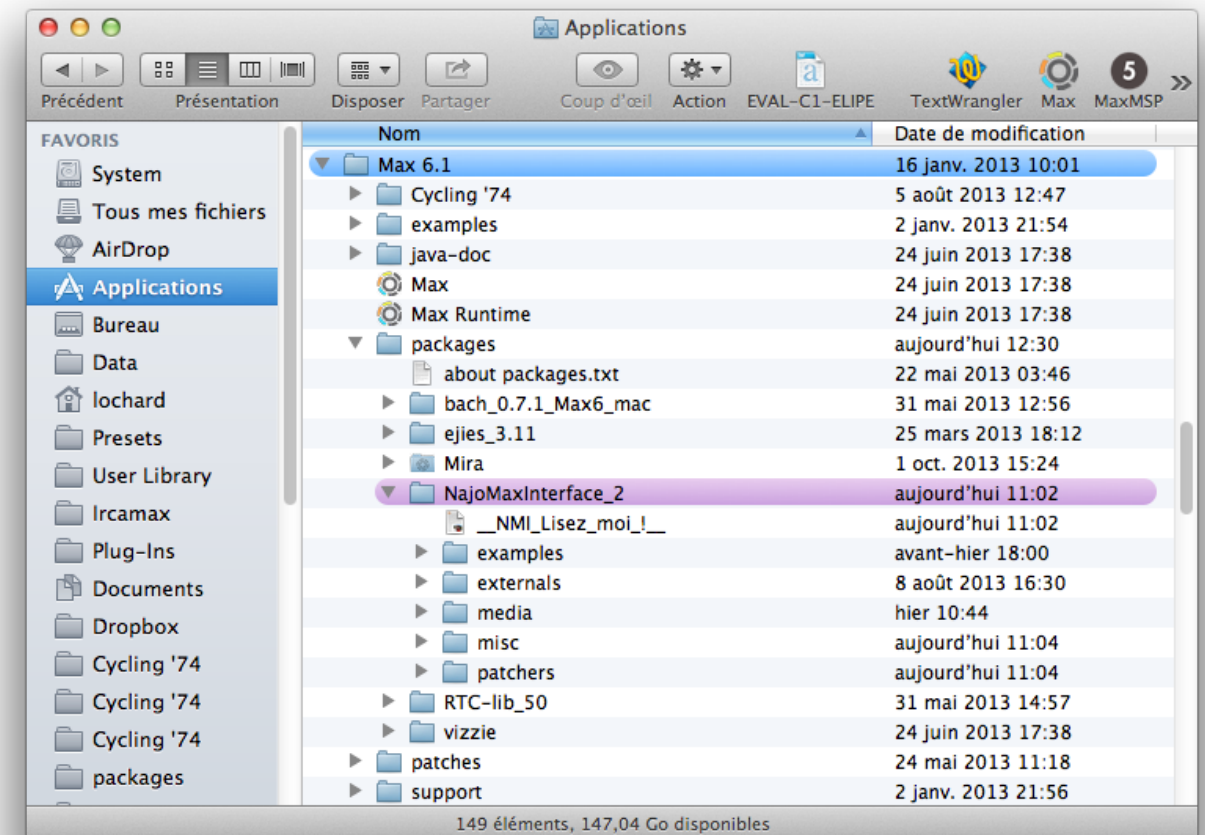
# Installation

## INSTALLATION

1. How to install
2. Content of the package

## How to install

NMI comes now as a package. The only thing to do is to copy the "NajoMaxInterface\_2" folder in:  
/Applications/Max6.1/packages



# Content of the package

The package contains a few folder:

The **examples** folder contains a few NMI projects to explore.

The **externals** folder contains all the externals used in the NMI modules. Those objects are copy protected. If your machine is not authorized yet, you should enter your Ircam Forum key the first time you create a module which use one of those objects.

The **misc** folder has subfolders where you find all the modules available in NMI. We put them in five categories : controller, effects, mix, sound players and synthesis modules.

**media** and **patchers** folder contains picture and Max patch that NMI needs to run properly.

# First steps with NMI2

## FIRST STEPS IN NAJO MODULAR INTERFACE 2

1. Create a new project
2. Default modules
3. Create a few modules...
4. Buid a monophonic synthesizer

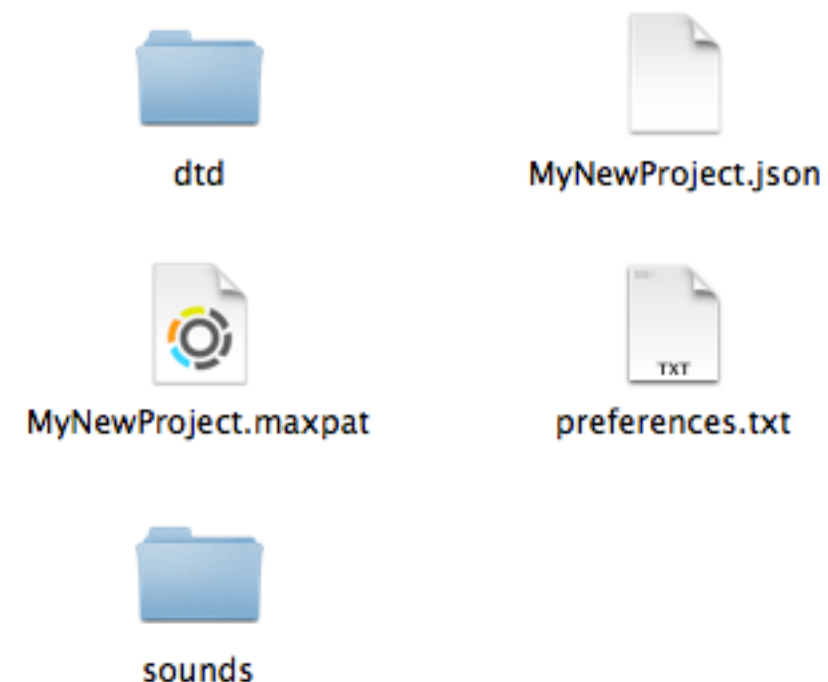
## Create a new project

To create a new NMI project just copy and paste the **NMI-EmptyProject** anywhere on your computer. Then rename the folder with the name you want, for example "NMI-NewProject".

The folder contains a **.maxpat** file which will memories the modules routings. Rename this file with the name you want (but keep the extension unchanged).

The project folder also contains a **.json** file which memories a bank of presets of your project. Rename it with same name as the .maxpat. It will be then loaded automatically when you launch the .maxpat of your project.

In our example, we call our two files "MyNewProject.maxpat" and "MyNewProject.json".

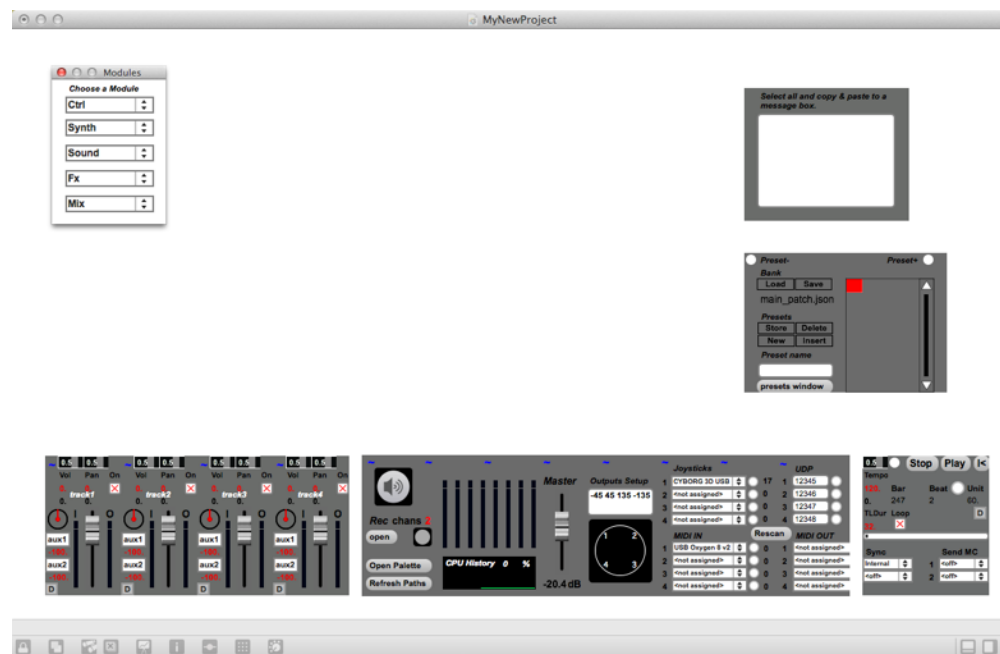


The subfolders **sounds** and **dtd** must stay at the same place and keep the same name otherwise NMI will not be able to find the sound files used in the project.

There is also a **preference.txt** file which must stay at the same place. It memories a part of the configuration of the project : MIDI ports and joysticks assignments, the positions of the loud-speakers, etc.

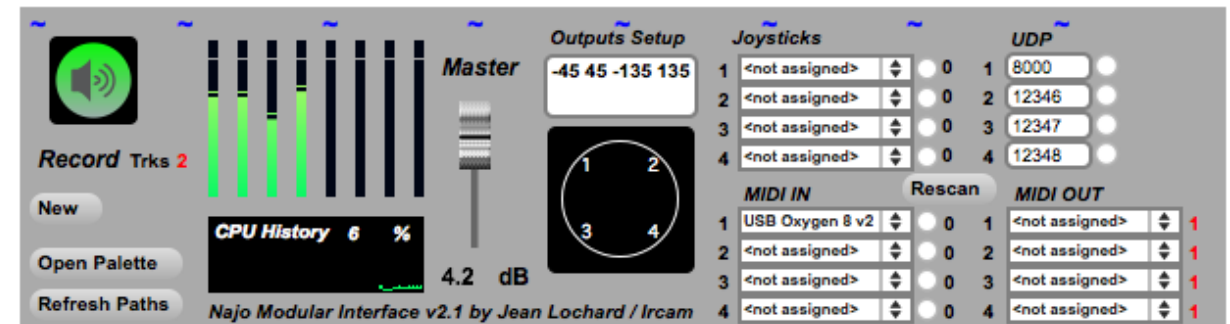
## Default modules

You can now double click on the **.maxpat** file of your project folder. Max opens and displays a patch with a default setup.



## Master

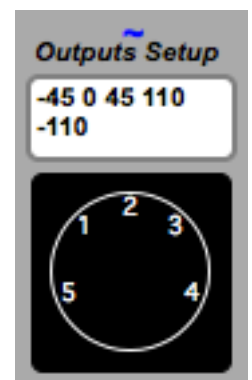
This module is the hart of NMI. It controls the DSP state and the master volume but it also defines which devices connected to computer is able to interact with the patch. Its also the place where you can define your speaker setup.



## Speaker setup

NMI is able to spatialize sound sources on a setup of many speakers from 2 to 8. Use the upper part of this space to set the position of the speakers in degrees (0° correspond to the front). The setup is memorized in the project preferences.

For example, to define a classical 5.1 speakers setup enter values as follow:



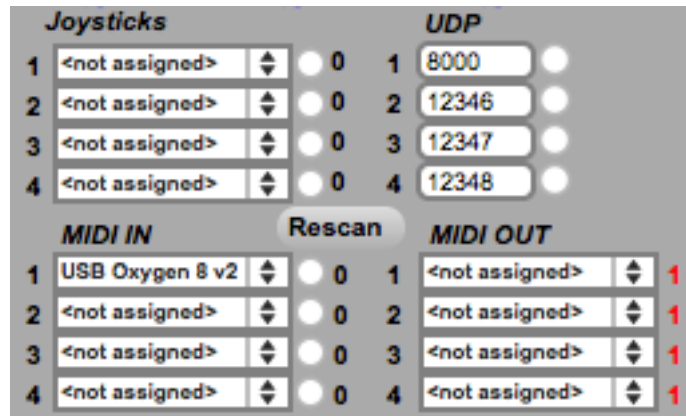


## Joysticks, UDP, MIDI IN, MIDI Out

All the menu off this section do the connection between NMI and your hardware.

You can connect up to **4 joysticks** and assign them to the 4 internal joystick ports.

You can define **4 network port number** and assign them to four internal ports.



The same is possible for **input MIDI devices** and **four output MIDI devices**. Each MIDI output can transmit MIDI data on its own MIDI channel (change the red values next to the menus). The configuration is memorized in the project's preferences.

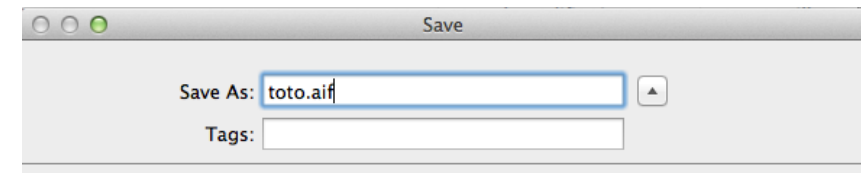
## Direct access to the outputs

The *Master* module has 8 inputs (symbolized by a blue wave) which allow to directly access to the outputs of the sound card. You can directly connect some modules as the *quad* or *sampler* module here.

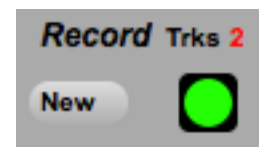


## Record outputs

The red value sets the number of outputs you will record. Max will save a multi track audio file with the same number of tracks. The "new" button opens a file selector to choose the name of the file for your next recording (yes, this is like this in Max !).



Name and choose a location for the new file.



Use the flashing record button in order to start and stop the recording. If the *Rec Start* function is on in the *transport* module, NMI starts its transport automatically.

## Transport

This module groups all the transport commands of NMI. It is also where you can define a Timeline duration in beats. This is also where you can setup a few options regarding the synchronization with the outside world. NMI can be synchronized on an external MIDI clock or it can generate it on two MIDI ports for





external devices such as a drum machine or any device understanding MIDI clock messages.

### Sync menus

Use the upper menu to set a synchronization source.

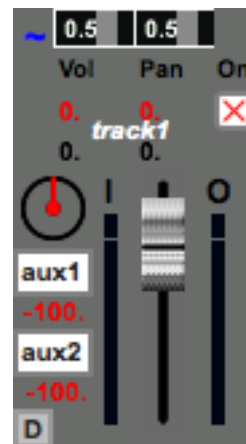
Internal: NMI is synchronized by Max

MIDI Clock: NMI uses MIDI clock informations coming from the MIDI port selected in the second menu.

## Tracks

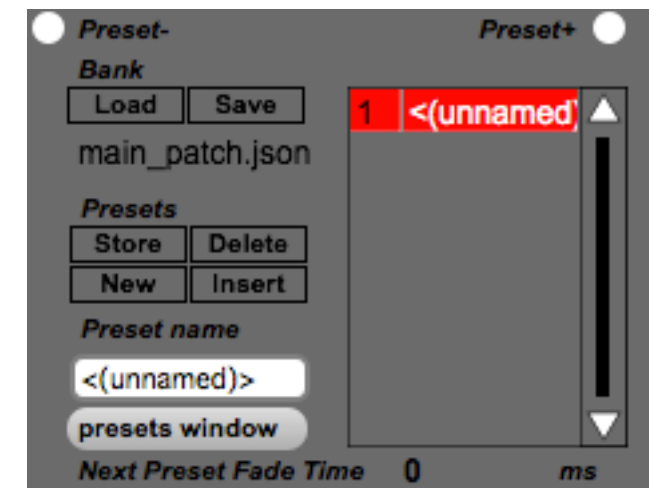
A default NMI patch starts with four Track modules. This module controls the volume of all the signals connected to it. It allows to define a 360° panning for the audio source depending of your speaker setup. It also specifies the gain and destination of two auxiliary buses (pre and post fader).

You can create more tracks if needed using the **Mix** section of the module palette.



## Presets

The preset section is used to make snapshots of every modules displayed in a patch in order to recall your favorite configurations. It can **create, store, insert, delete** or rename a preset.



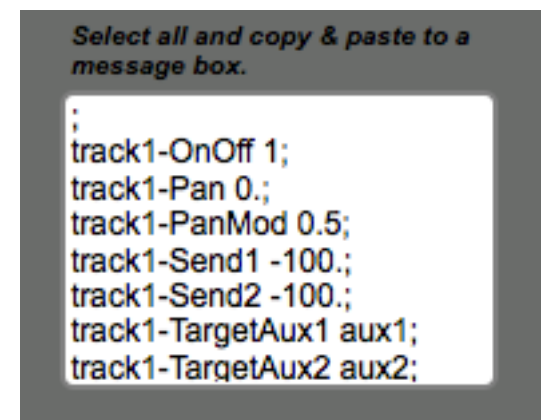
You can **save** a bank of preset or **load** a new one using the load and save buttons.

The module also defines a **Next Preset Fade Time** for each snapshot in order to perform morphings between presets.

## Messages

This box receives the messages generated by a module when pressing its **Dump** button.

For example, try it with track1 by clicking on its **D** button. A list of messages that describe the actual configuration of the module appears in the Messages section. You can then copy/paste those mes-



sages in a conventional message box to change the value of a particular parameter in the patch.

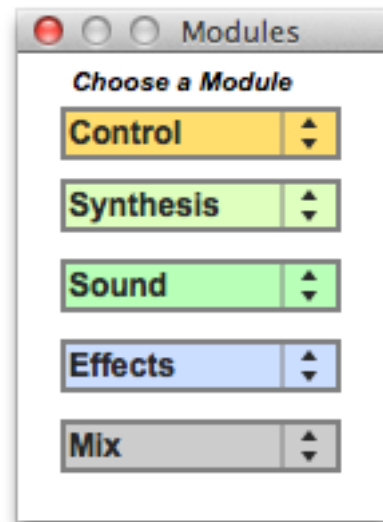
All parameters in NMI modules can be controlled remotely using the ";" in a message box. This is a way to perform small changes in a patch without using presets.

# Let's create a few modules...

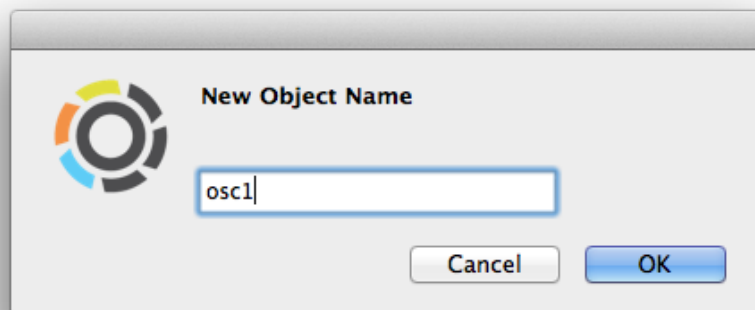
## First, an oscillator

In order to create a new module, you have to use the Module Palette. If it is not visible on the screen, use the **Open Module Palette** button in the Master section to make it appear again.

First choose a **osc** module in the Synthesis menu.



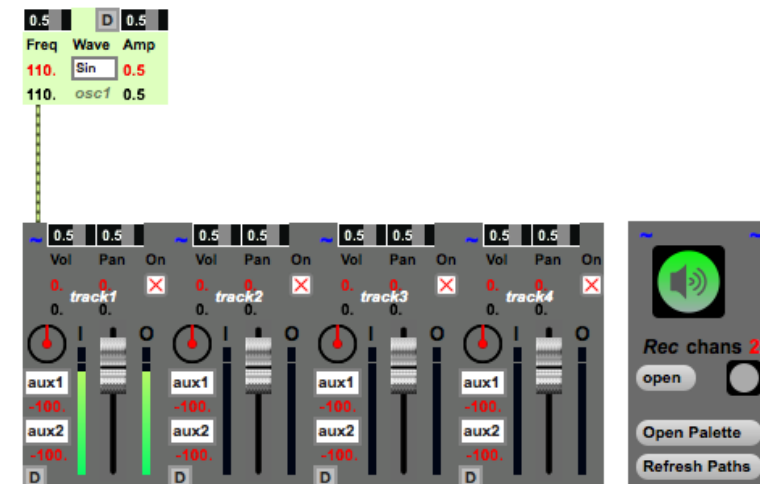
A box appears asking you the name to give to the new object. Add a number to the proposed name and validate with return.



**WARNING : All modules must have a different name. Otherwise the preset system will not perform correctly.**

The new module appears in the patch. Switch to "edit mode" to be able to change its position. Connect it to *track1*.

Switch the DSP on in the master section to hear your new oscillator.



## Parameters

Modules have parameters that you can modify with the mouse by changing the **red** value.

For example, our oscillator has two parameters that you can change this way : its Frequency and its Amplitude.

0.5	D	0.5
Freq	Wave	Amp
110.	Sin	0.5
110.	osc1	0.5

The other way to change a parameter value is to apply a modulation to the parameter by connecting a signal to the corresponding input of the module. The black value will display the current value of the parameter. Let's make it clear with an example.

## Apply a modulation

Create an LFO using the **Synthesis** menu of the Palette. Then connect it to the frequency input of the oscillator.

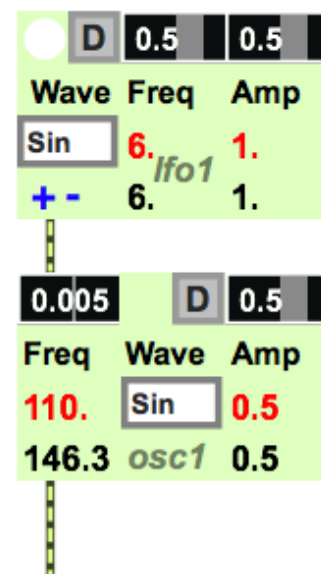
You hear now that the frequency of the oscillator is modulated around the specified red value.

## How modulation works

By definition a modulation signal modulates a parameter around a fixed value. Those values are the red values in NMI. Then, for each parameter, you define a modulation amount with the black & white slider. This value, from -1 to 1, multiplies the input signal. The result, multiplied by the range of the parameter defines what is added or subtracted to the red value.

Some modules, as the LFOs, produce a bi-polar signal. This means that those module generate a positive and negative modulation when they are connected to another module.

Others, like envelopes, create a positive signal. Their effect depends of the sign of the modulation slider.

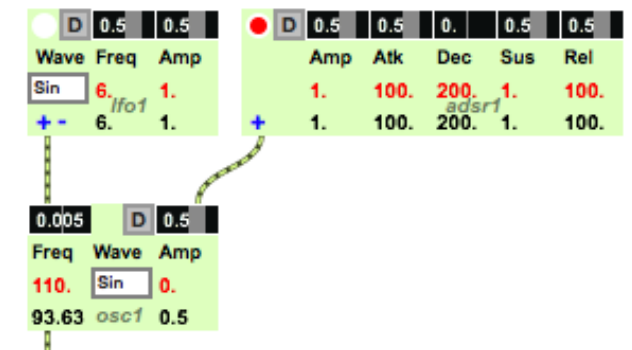


## Add an envelope

Create an **adsr** module using the **synth menu** of the palette. Call it **adsr1** and connect its output to the **Amp** input of the oscillator. We want no signal when the envelope is off. Therefore, we set the **Amp** value of the oscillator to zero.

Then trig the envelope using the button on the left top of the module. The *adsr* envelope starts and reaches its sustain level.

We hear back again our oscillator. Turn the envelope off to perform its release stage and switch off the oscillator again.



## Add a trigger to trig the envelope automatically

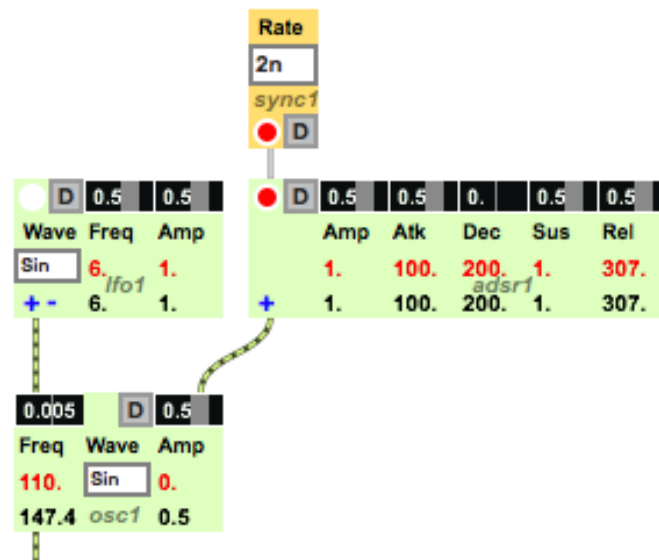
We find two kinds of datas in NMI. The first are signals between -1 and 1. The second are triggers that use controls wire of Max. They send 1 when something has to be on and 0 when something has to be off. Triggers are symbolize by a white button. You will find them on many modules in NMI.

Let's add a module to trig our envelope automatically: create a **sync** module using the **Ctrl** menu of the module palette. This

module produces a trigger synchronized with the global tempo of NMI. Connect it to the envelope as follow.

Then, turn on the global tempo using the **play** button in the transport section. The module flashes and switches the envelope on and off.

By connecting those few modules we already overviewed all the main concepts in Najo Modular Interface. To go further, lets build a small monophonic synthesizer.



# Build a monophonic synthesizer

Let's build a synthesizer controlled by MIDI, with a square oscillator, two envelopes, an LFO and a filter.

## Super square

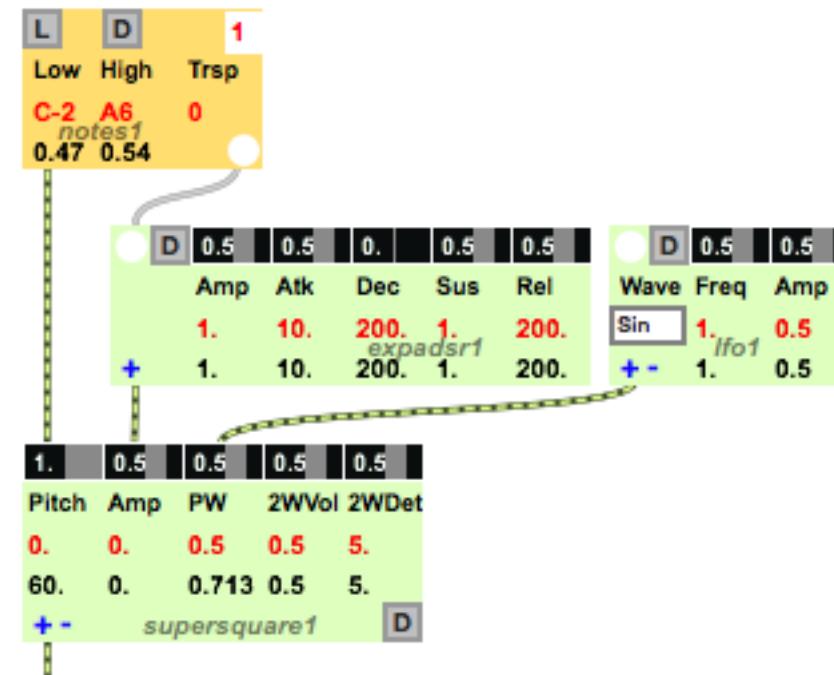
First, create a **supersquare** oscillator. This module combines three square waves with pulse width modulation. Two of them can be detuned and you can also control there volume. Connect the output of *supersquare* on a **track**.

Create a first envelope but this time choose the exponential **expadsr** (Synth) module. Connect it to the **Amp** input of the *supersquare* and set the default value of the Amp to 0.

Create an **lfo** and connect it to the **pulse width** (PW) modulation input of the *supersquare*.

To trig the envelope, we want to use the notes coming from our MIDI keyboard. To achieve that, create a **notes** module (Ctrl). The first output of this object generates a signal between 0. and 1. proportional to the played MIDI note. Connect this signal to the **Pitch** of the the *supersquare* and set the Pitch value to 0. Connect the trigger output of the module to the trigger input of the envelope.

You should have now something more or less like the following on your screen:



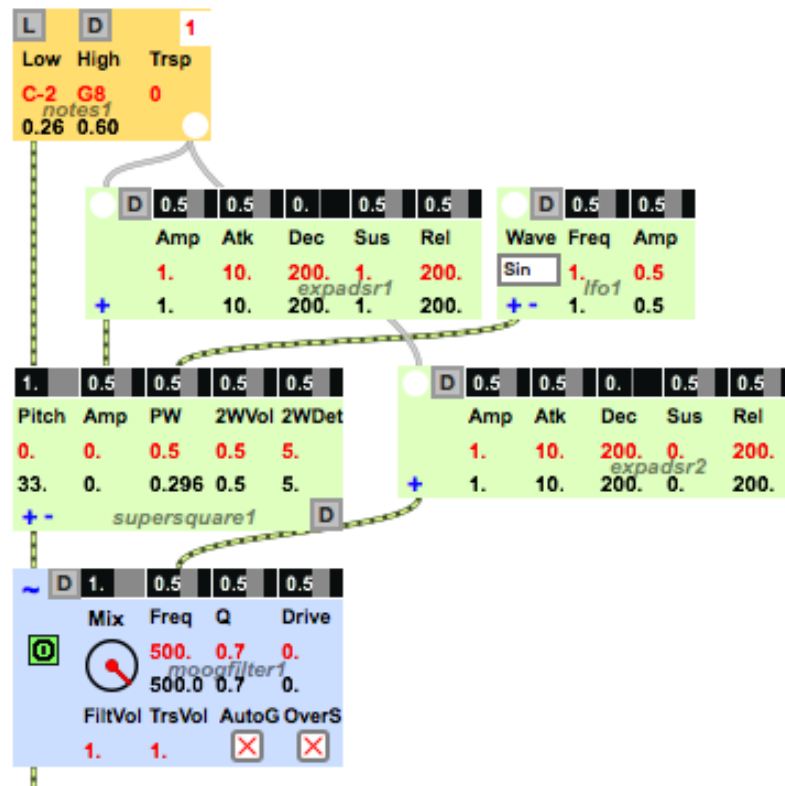
By default, the **notes** module receives MIDI datas from the **internal port 1**. You have to assign an external keyboard to that port in the *Master* section of NMI in order to receive notes correctly in the module.

You should be able to play your synthesizer from your keyboard now.



## Add a filter and a second envelope

Choose **moogfilter** in the *Effects* menu and connect it between *supersquare* and the *track* module. Create a second **expadsr**, set its sustain value to 0. and connect it as follow to modulate the cutoff of the filter by this new envelope.

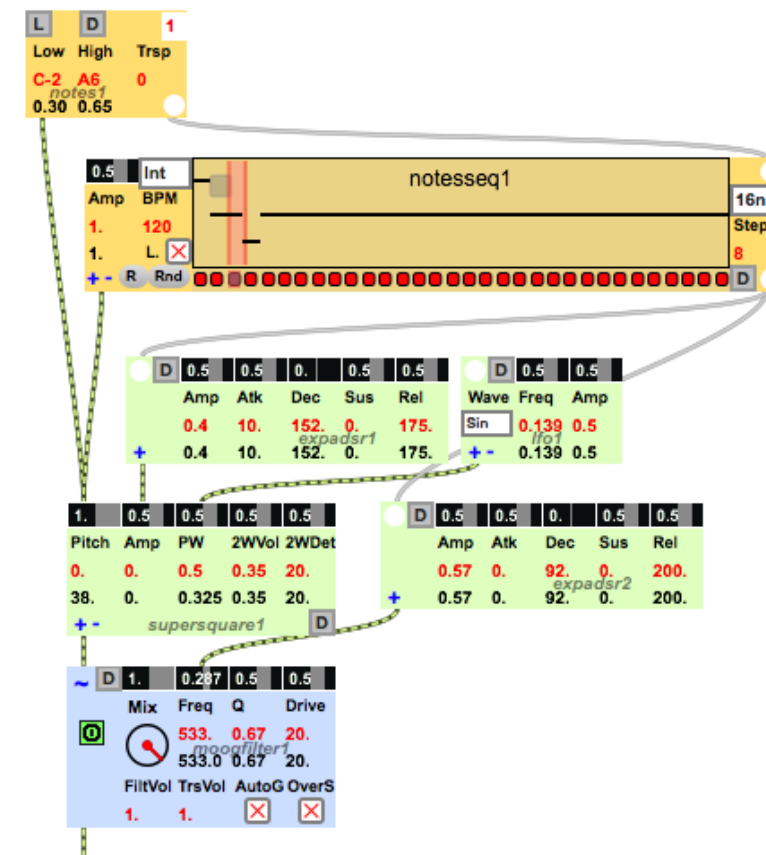


Adjust the Q to increase the modulation effect.

## Add a note sequencer

Imagine that we now want to trig a sequence of notes from our MIDI keyboard. Here is the way to do it.

Create a **noteseq** (Ctrl) object and connect it to the other modules as follow.



This last module allows you to program a sequence which will be interpreted as a transpositions amount when connected to a pitch input of a synthesis module. This means that **noteseq** generates positive and negative values.

By default, the module is synchronize on the global tempo of NMI. If we want to trig the transposition sequence with our keyboard, we need to put the module in **Internal** mode. This can be done with the **Glo/Int** menu on the module.

When in **Int mode**, a trigger input appears in to be able to trig the sequence with another module. Connect the trigger output of the **notes** object to this input.

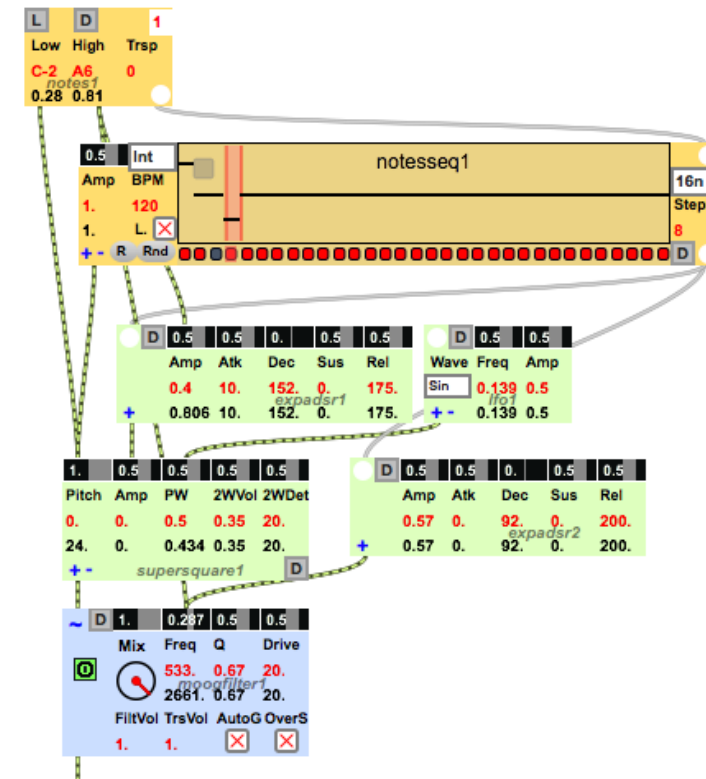
The trigger output of **noteseq** can be connected to our two previous **expadsr** in order to trig them on each step of the sequence.

## Use velocity

To finish with our little synthesizer, we can make it sensitive to the velocity of our keyboard.

It easy to add this feature. Connect the velocity output of the **notes** to the **Amp** input of the first **expadsr** envelope to change the volume according to the incoming velocity.

You can also experiment with the **Freq** input of the filter.



# Modules reference

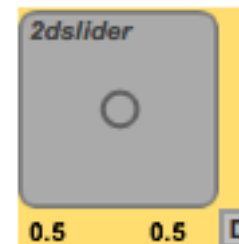
## MODULE REFERENCE

1. Control modules
2. Synthesis modules
3. Sound player modules
4. Effect modules
5. Mix Modules

## Control Modules

You will find in this family all the modules to control your patch from the outside world. The modules of that category generate signals from MIDI, joysticks, UDP, the new **mira-touch** object, etc.

### 2dslider



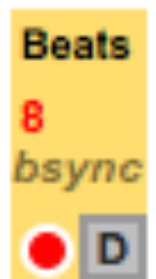
This module generates two signals from 0. to 1. depending of the position of the mouse on the 2 dimensions slider.

### bsync

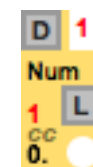
This object generate a trigger synchronized on the global tempo of NMI.

*beats*

This parameter defines the time in beats between two triggers.



### cc (continious controller)



This module is the way to use Continuous Controller MIDI message in NMI. It generate a signal between 0 and 1. proportional to the controller value. It also generate a trigger when the signal value is bigger than 0.5.

## *MIDI port*

Use this menu to define on which internal MIDI port you want to receive the MIDI datas. Remember that you must define the mapping between the MIDI ports and your hardware in the Master section of NMI.

## *Num*

Defines which controller number is used. The number can also be learned using the **L (Learn)** button on the module.

## **ccout**

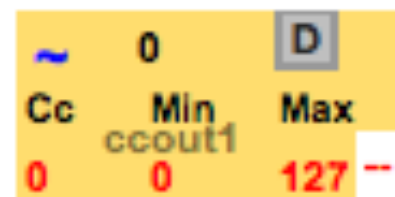
This object allows to generate a MIDI controller from any signal of a patch to control your external MIDI gears.

## *Cc (Continuous Controller)*

Defines the number of the controller to generate.

## *Min*

Defines a minimum value for the controller which means the value generated when the module receives the audio signal 0.



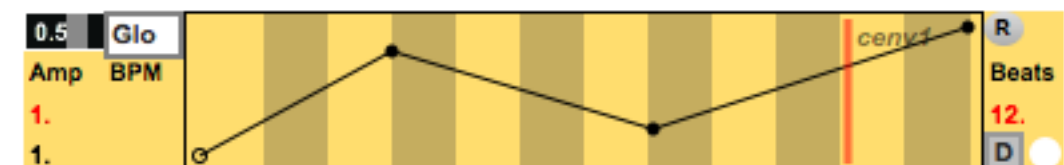
## *Max*

Defines the value to send when the module receives the 1. audio signal.

## *Out MIDI port*

This menu defines on which MIDI output port to send the new controller datas. The mapping between port numbers and physical devices is done in the MIDI OUT section of the Master. You can also define a target MIDI channel in this area.

## **cenv (control envelope)**



This module allows to draw an envelope with the function object of Max. The lecture of the envelope is (by default) synchronized on the global tempo of Najo Modular Interface.

## *Amp*

The Amp parameter sets a value which defines the maximum value that the function can reach.

## *Glo/Int menu*

This menu sets the synchronization mode of the module. When the **Glo** mode is selected, the lecture is synchronized with the global tempo of NMI. To start reading the function, press **play** in the transport section. In **Int mode**, the module is set to **Internal** mode. It can be then switch on and off by using its trigger input.

## *BPM*

In **Int** mode, specify the tempo in beat per minutes.

## *L (Loop)*

In **Int** mode, defines if the sequence must be looped or not.

## *Beats*

Defines the duration of the function in beats.

## **compkey**

Allows to use the computer keyboard to generate a signal in a patch. You can use the **L (Learn)** button to defines which key to use.

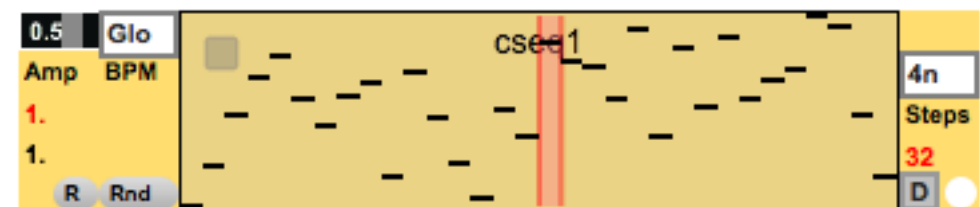


The module generates a signal value of 1. or 0. and a trigger, depending of the state of the chosen key and the value of the toggle mode.

## *Tgl (toggle)*

Switch the toggle mode on and off.

## **cseq (control sequence)**



This object is a step sequencer which delivers signals between 0. and the **Amp** parameter.

### *Amp*

Defines the maximum value of the step sequencer. The values of the sliders are mapped between 0. and Amp.

### *R (reset)*

Puts all the sliders at 0.

### *Rnd (Random)*

Generate a random sequence.

### *Glo/Int menu*

This menu sets the synchronization mode of the module. When the **Glo** mode is selected, the lecture is synchronized with the global tempo of NMI. To start reading the multislider, you have to press **play** in the transport section. In **Int mode**, the module is set to **Internal** mode. It can be then switch on and off by using its trigger input.

### *BPM*

In **Int** mode, specify the tempo in beat per minutes.

### *L (Loop)*

In **Int** mode, defines if the sequence must be looped or not.

### *Sync Rate menu*

Defines the rate of the sequencer in the notevalues format used in Max.

### *Steps*

Defines how many steps of the sequence are used.

## **envfol**

This module uses an average~ and a slide~ object to perform an envelope follower on its incoming signal.





### *Gain*

Defines a Gain in dB applied to the signal to "calibrate" the volume to follower. Change that value to amplify or reduce the amplitude of the data produced by the module.

### *Atk*

Defines a value in samples which sets how the slide~ object reacts when it receives increasing values. The more this value is big the less the module is reactive to the attacks of the signal.

### *Rel*

Defines a value in samples which sets how the slide~ object reacts when it receives decreasing values. The more this value is big the less the module is reactive to the release of the signal.

### *Off*

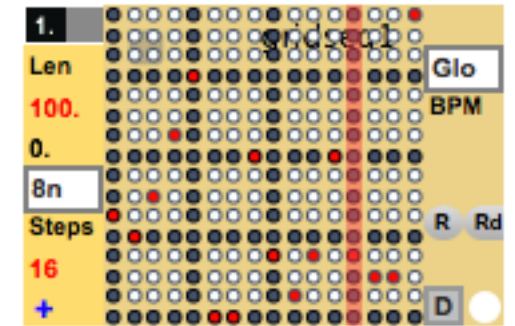
Defines the minimum value to reach to switch off the trigger output. This works only if an attack has been detected before.

### *On*

Defines the value to reach to switch on the trigger output. To be switched on again, the trigger must be released first.

## **gridseq**

This module uses a grid object to generate a signal between 0. and 1. It is used usually to control the **position** parameter of a **gran** or a **supervp.scrub** module.



### *Glo/Int menu*

This menu sets the synchronization mode of the module. When the **Glo** mode is selected, the lecture is synchronized with the global tempo of NMI. To start reading the live.grid, you have to press **play** in the transport section. In **Int mode**, the module is set to **Internal** mode. It can be then switch on and off by using its trigger input.

### *len*

Defines the length of the ramp between two matrix values as a percentage of the sync rate value. Set to zero, this parameter will produce freezes if the **gridseq** is connected to the position parameter of a **gran** or a **supervp.scrub** module.

### *Steps*

Defines how many steps of the sequence are used.

### *BPM*

In **Int** mode, specify the tempo in beat per minutes.

### *L (Loop)*

In **Int** mode, defines if the sequence must be looped or not.

### *R (reset)*

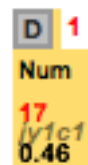
Reset the grid.

### *Rnd (Random)*

Generate a random sequence.

## **jy1c (joystick to 1 control signal)**

This module embeds a hi (human interface) max ob-



ject to transform a continuous direction of a joystick into a signal from 0 to 1.

### *Joystick port*

Choose from which joystick or other device you want to receive the data.

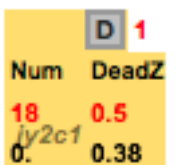
### *Num*

Each button or direction has a number id. There is no learn function on that module because it's usually hard to send data from just one direction with a joystick. But you can see the last send id in the Master section, next to the joysticks menus.

Note : because joysticks may have different resolution (256, 512 or 1024 values), the module calibrate itself automatically. Before starting your performance, you must manipulate your joystick with the maximum amplitude you will use in order to calibrate the output values correctly.

## **jy2c (joytick to 2 control signals)**

This object acts as the **jy1c** module but it generates 2 signals instead of 1. The idea is to set two different parameters with one direction of a joystick. (ex: one signal for the left, one signal for the right).



A dead zone can be defined to avoid to send values when the joystick is centered.

### *Joystick port*

Choose from which joystick or other device you want to receive the data.

### *Num*

Each button or direction has a number id. There is no learn function on that module because it's usually hard to send data from just one direction with a joystick. But you can see the last send id in the Master section, next to the joysticks menus.

Note : because joysticks may have different resolution (256, 512 or 1024 values) the module calibrate itself automatically. Before starting your performance, you must manipulate your joystick with the maximum amplitude you will use in order to calibrate the output values correctly.

### *DeadZ (Dead Zone)*

Defines the size of the center Dead Zone.

## **miratouch**

This module embeds a miratouch object to track the position of 5 fingers on an iPad. Each finger generates 2 signals depending of the x and y position of the fingers on the device. There are also 5 trigger outputs corresponding to each finger.

Each finger is able to switch on a process and has two signals to control it.



### **note**

This module converts a single MIDI note into two signals, one proportional to the pitch, one proportional to the velocity. A trigger output is also available.



### *Num*

Defines the MIDI note number to use in the module.

### *L (Learn)*

Use the learn button to define the note number automatically.

*Tgl (toggle)*

Defines the behavior of the module. With **Tgl** off, a note-on message switches on the trigger output, a note-off message switches it off. When **Tgl** is one, the note-ons switches the module on and off alternatively and the note-off have no effect.

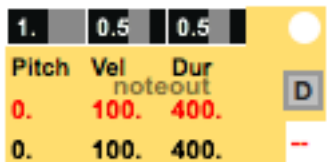
The velocity signal is generated only when the module switches on.

*MIDI port*

Use this menu to define on which internal MIDI port you want to receive the MIDI datas. Remember that you must define the mapping between the MIDI ports and your external hardware in the Master section of NMI.

**noteout**

This module use a signal and triggers to generate notes for your MIDI devices.



*Pitch*

This parameters defines the minimum or the maximum pitch which will be created by the module depending of the value of the Modulation slider.

*Vel*

Defines the velocity of the MIDI notes

*Dur*

Defines the length of the generated MIDI note. A new note is send only when the Trigger input switches on.

*Out MIDI port*

This menu defines which MIDI output port is used to send the MIDI notes. The mapping between port numbers and physical devices is done in the MIDI OUT section of the Master. You can also define a target MIDI channel in this area.

**notes**

This object receives MIDI notes from an area of a MIDI keyboard. The Pitch values are transformed into a signal between 0.



and 1.

### *Low*

Defines the lowest pitch of the keyboard area.

### *High*

Defines the highest pitch of the keyboard area.

### *Trsp*

Defines a transposition amount added or subtracted of the last received Pitch.

### *L (Learn)*

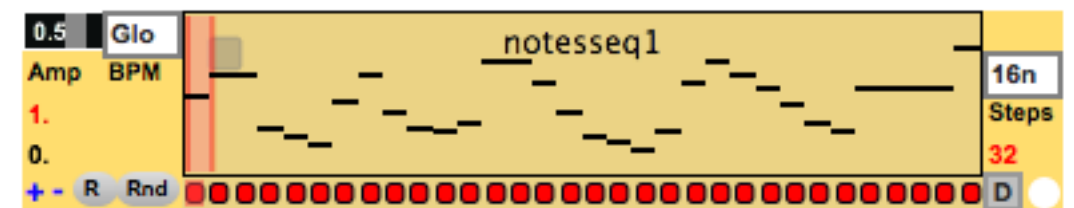
Use the learn button to define the keyboard area automatically. Just press the button and play a chord of two notes on your MIDI keyboard. It defines the minimum and maximum pitches to use in the module.

## *MIDI port*

Use this menu to define on which internal MIDI port you want to receive the MIDI datas. Remember that you must define the mapping between the MIDI ports and your external hardware in the Master section of NMI.

## **noteseq**

The noteseq module allows to define a sequence of signal which can be interpreted as a sequence of transpositions when it is connected to a **synth** module.



## *Amp*

By default, this module provide a signal from -1 to 1. You can reduce this ambitus by chancing this value.

## *Glo/Int menu*

This menu sets the synchronization mode of the module. When the **Glo** mode is selected, the lecture is synchronized with the global tempo of NMI. To start reading the sequence, you have to press **play** in the transport section. In **Int mode**, the module is set to **Internal** mode. It can be then switch on and off by using its trigger input.

## *Steps*

Defines how many steps of the sequence are used.

## *BPM*

In **Int** mode, specify the tempo in beat per minutes.

## *Sync Rate menu*

Defines the rate of the sequencer in the *notevalues* format used in Max.

## *R (reset)*

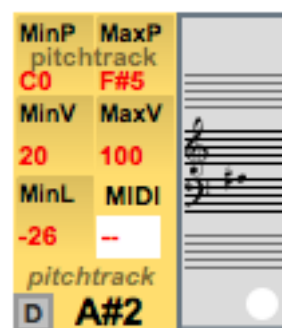
Reset the sequence.

## *Rnd (Random)*

Generate a random sequence.

## **pitchtrack**

This module uses a special Ircam technologie called yin~ to perform a funda-



mental frequency analysis. It generate a signal from 0. to 1. proportional to the detected pitch and a trigger. This module can also be used as an audio to MIDI converter and directly send the MIDI notes to one of the four MIDI out of NMI.

## *MinP*

Defines the minmum pitch of the tracked instrument. If pitchtrack detects a picth which is under this value, no MIDI notes is generated.

## *MaxP*

Defines the maximum pitch of the tracked instrument. If pitchtrack detects a picth which is above this value, no MIDI notes is generated.

## *MinV (Minimum Velocity)*

When a note is detected, its amplitude is converted to MIDI velocity. This value fixes the minimum velocity that the module can produce.

## *MaxV (Maximum Velocity)*



This fixes the maximum velocity that the module generates when converting the amplitude of the signal to velocities.

### *MinL (Minimum Level)*

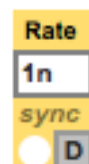
This defines the minimum level that the signal has to reach to be considered as a possible new pitch.

### *MIDI (MIDI Out Port)*

This allows to select an output MIDI port to directly send MIDI notes to an external device.

## **Sync**

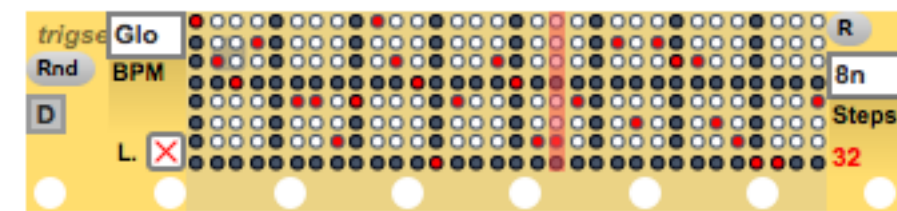
This module produces a trigger synchronized on the global tempo of Najo Modular Interface. The duration of the impulsions is equal to half the duration of the note value defined by the user.



### *Rate*

Defines the time between two triggers in Max notevalues.

## **Trigseq**



This module generates a sequence of triggers that can be used

to start any other modules in Najo (envelopes, recordings, etc). Each line of the matrix represents one of the 8 triggers outputs of the module. The tempo can come from the Global Transport or it can be the one of the module itself.

### *Glo/Int menu*

This menu sets the synchronization mode of the module. When the **Glo** mode is selected, the lecture is synchronized with the global tempo of NMI. To start reading the sequence, you have to press **play** in the transport section. In **Int mode**, the module is set to **Internal** mode. It can be then switch on and off by using its trigger input.

### *Steps*

Defines how many steps of the sequence are used.

### *BPM*

In **Int** mode, specify the tempo in beat per minutes.

### *L (Loop)*

In **Int** mode, defines if the sequence must be looped or not.

### *R (reset)*

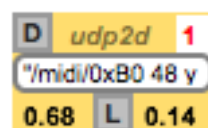
Reset the sequence.

### *Rnd (Random)*

Generate a random sequence.

## **udp**

This module converts values received by **udp** to a signal. OSC data can be generated by a lot of applications on a computer (OSCulator, Synaps for kinect) or by an iPad (TouchOSC™, Lemur™...) for an extended control of your NMI patches. **Udp port addresses** have to be correctly defined in the **master section** of NMI. You can receive OSC (Open Sound Control) data from up to four devices.



### *Udp port*

Select one of the four possible udp input port.

### *OSC address*

Enter here the address of the slider you want to receive with module. This address can also be learned automatically.

### *L (Learn)*

Allows to learn the udp path of a slider. To learn an adress, switch that button on and move a slider on your iPad or other device.

## **udp2d**

This module is the same as the **udp** module but dedicated to 2d sliders. It is able to receive separately an x and y position sent by the 2d slider in Lemur™ or TouchOSC™ and convert them into usual signals for NMI.

### *Udp port*

Select one of the four possible udp input port.

*OSC address*

Enter here the address of the slider you want to receive with the module. This address can also be learned automatically.

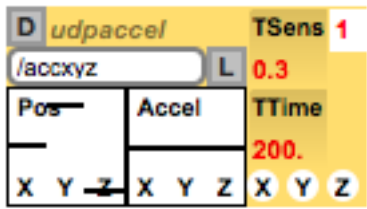
*L (Learn)*

Allows to learn the udp path of a slider. To learn an address, switch that button on and move a slider on your iPad or an other device.

**udpaccel**

This module was programmed to deal with accelerometer's datas sent by an iPad or iPhone (or any other 3 axis accelerometers). The datas are filtered to separate the position and the acceleration transmitted by the sensor along the 3 axis. **Signals from -1. to 1.** are generated from those six values.

A threshold system allows to detect events along the 3 axis and generates triggers according to the gestures.



*TSens (Trigger section sensibility)*

Adjust the acceleration threshold in the trigger section of the module. The more it is big, the more you need an hard gesture to generate a trigger. It also separates better the gestures along the different axis.

*TTime (Trigger section time)*

Defines the minimum time between two gestures in the trigger section.

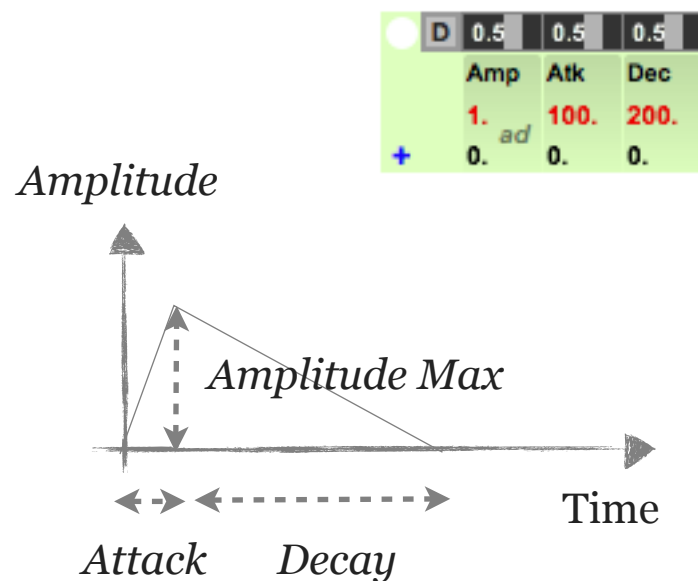
*L (Learn)*

Allows to learn the udp path of a slider. To learn an adress, switch that button on and move a slider on your iPad or other device.

# Synthesis Modules

## ad (attack decay envelope)

This module produces a simple linear attack/decay envelope with values between **0.** and **1.** (by default).



The envelope is played from the beginning to the end each time a trigger is received at the leftmost input of the module.

### Amp (Amplitude)

Defines the maximum value of the envelope which is reached after the Attack time.

### Atk (Attack)

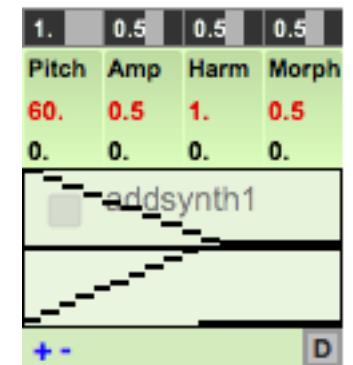
Defines the attack time of the envelope.

### Dec (Decay)

Defines the decay time of the envelope.

## addsynth (additive synthesizer)

This module is an additive oscillator. Two multisliders are used to define two configurations of amplitude for 16 sinusoids. A morph parameter allows to move between the two configurations.



### Pitch

Defines the pitch of the oscillator as a float MIDI note number. It sets the frequency of the first sinusoid also called the fundamental.

### Amp (Amplitude)

Defines the amplitude of the module.

*Harmo (Harmonicity)*

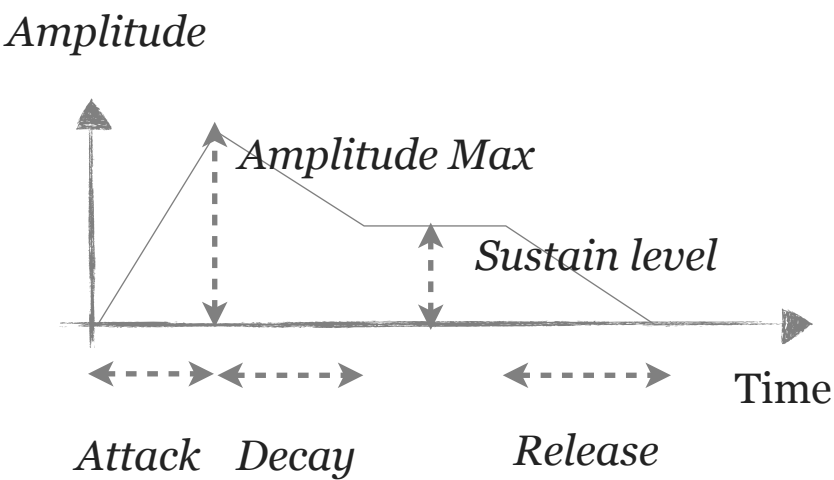
Sets the harmonicity of the synthesis. If set to 1., this parameter produces an harmonic spectrum : the frequencies of the sinusoids are multiple of the fundamental frequency. If this value is smaller than 1., it compresses the spectrum. If the value is bigger than 1., it expand the spectrum.

*Morph (Morphing)*

This parameter defines a position between the two amplitude situation. 0. means that we are on the first setup. 1. means that we use the second setup. Values in between will produce a linear interpolation between the two multislidiers.

**adsr (attack decay sustain release envelope)**

This module produces a signal that follows a classical linear ASDR envelope.



	D	0.5	0.5	0.	0.5	0.5
	Amp	Atk	Dec	Sus	Rel	
	1.	10.	200.	1.	200.	
+	1.	10.	200.	1.	200.	

The envelope is played from the beginning to the sustain stage when a 1 message is received from the leftmost inlet of the module. The release stage starts when 0 is received.

*Amp*

Defines the maximum level of the envelope. This is the level reached by the signal after the attack stage of the envelope.

*Atk (Attack)*

Defines the attack time of the envelope.

*Dec (Decay)*

Defines the decay time of the envelope.

*Sus (Sustain Level)*

Defines the level after the attack and decay time. This value sets a level proportional to the maximum amplitude value of the envelope. 1. means that the envelope stay at its maximum value after the attack phase.

## *Rel (Release)*

Defines the Release time of the envelope.

## **amp**

This very simple module provides a signal multiplication by an amplitude value between 0. and 1.

## *Amp*

Value of the amplitude that multiply the signal.

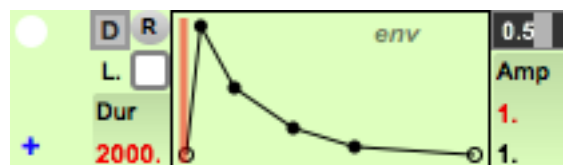


## **env (breakpoint function envelope)**

This box allows you to describe an envelope using a breakpoint function. It is possible to draw more complex envelopes than the classical ADSR. Use this module for short envelopes with no link with the Global Transport. For longer envelopes, use the **cen**v module of the **Control** menu.

## *Amp*

Envelope maximum Amplitude.



## *Loop*

Loop the envelope as long as it is on.

## *Dur (Duration)*

Defines the length of the envelope in milliseconds.

## *R (Reset)*

Resets the shape of the envelope.

## **expad**

This module is the same as the **ad** module except that it provides exponential transitions between the different stages of the envelope.

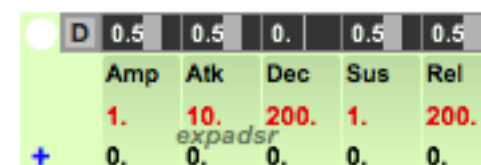
See **ad** for the list of parameters.



## **expadsr**

This module is the same as the **adsr** module except that it provides exponential transitions between the different stages of the envelope.

See **adsr** for the list of parameters.





## lfo

This box is a classical Low Frequency Oscillator controlled in frequency. If you need an lfo synchronized on the Global Transport tempo, use the **sync\_lfo** module.



### Wave

Choose a wave for the lfo between **Sin** (sinusoid), **Saw** (sawtooth), **Sqr** (square), **S&H** (Sample & Hold).

### Freq

Defines the frequency of the Low Frequency Oscillator from 0.01 to 100 Hz.

### Amp

Defines the maximum amplitude of the LFO.

### Input trigger

A trigger connected to the lfo restarts its phase.

## osc

This module provides a classical oscillator controlled in frequency.



### Wave

Choose a wave between **Sin** (sinusoid), **Tri** (triangle), **Sqr** (square), **Saw** (sawtooth).

### Freq

Defines the frequency of the oscillator from 10 to 10 000 Hz.

### Amp

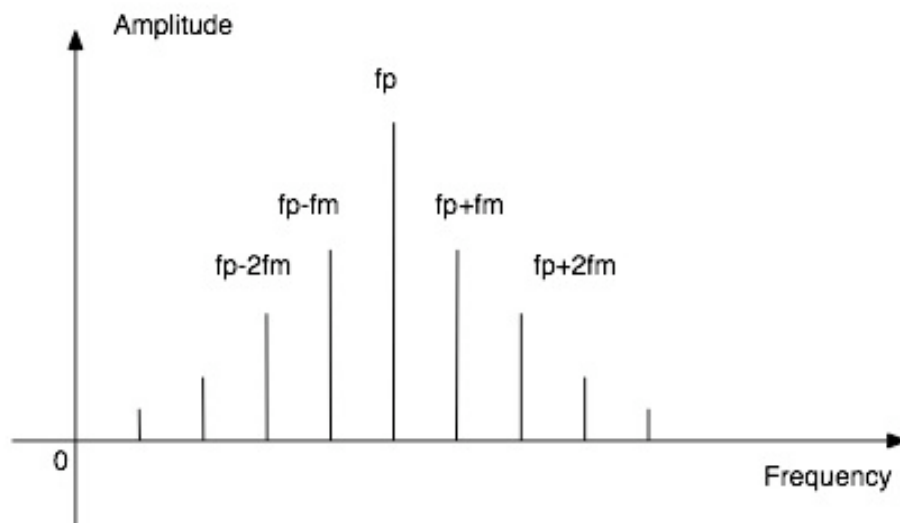
Defines the maximum amplitude of the oscillator.

## phasemod

This special oscillator provides Phase Modulation synthesis. It produces the same type of sounds than the well known FM (frequency modulation) synthesis but it is easier to control. The frequency of the module is controlled by a MIDI pitch which can be a float in order to produce quarter or eighth of tones.

1.	0.5	0.5	0.5
Pitch	Amp	Harm	ModA
60.	0.5	1.	0.5
0.	0.	0.	0.
+ - phasemod			D

Here is the spectrum that you obtain if the frequency of the carrier is  $fp$  and the frequency of the modulator is  $fm$ .



Spectrum generated by Phase Modulation

### *Pitch*

The frequency of the carrier is defined as a float MIDI pitch where 60 corresponds to the central C3 of a MIDI keyboard.

### *Amp*

Defines the final amplitude of the oscillator.

### *Harm (Harmonicity)*

This parameter defines a multiply factor between the frequency of the carrier  $fp$  and the frequency of the modulator :

$$fm = h * fp$$

The value of  $h$  will help us to predict the result of the phase modulation :

$h=1$  generates an harmonic spectrum.

$h=2$  generates an harmonic spectrum with just the odd frequencies.

### *ModA*

Defines the amplitude of the phase modulation. This parameter directly affects the brightness of the synthesis. It changes the amplitudes of the peaks generated around the frequency of the carrier.

**supersaw**

This module generates a wave form by combining 3 sawtooth oscillators.

1.	0.5	0.5	0.5
Pitch	Amp	2WVol	2WDet
60.	0.5	0.5	0.5
0.	0.	0.	0.
+ - supersaw D			

*Pitch*

Defines the frequency of the central oscillator as a float MIDI pitch.

*Amp*

Defines the final amplitude of the oscillator.

*2WVol (Secondary Waves Volume)*

Defines the volume of two other sawtooth detuned on each side of the central saw-teeth.

*2WDet (Seconday Waves Detune)*

Defines the detuning of the two other sawtooth in cents above and below the frequency of the central sawtooth oscillator.

**supersin**

This module is based on the same idea than **supersaw** but using sinusoids.

1.	0.5	0.5	0.5
Pitch	Amp	2WVol	2WDet
60.	0.5	0.5	0.5
60.	0.5	0.5	0.5
+ - supersin D			

*Pitch*

Defines the frequency of the central oscillator as a float MIDI pitch.

*Amp*

Defines the final amplitude of the oscillator.

*2WVol (Secondary Waves Volume)*

Defines the volume of two other sawtooth detuned on each side of the central sinusoid.

*2WDet (Seconday Waves Detune)*

Defines the detuning of the two other sawtooth in cents above and below the frequency of the central sinusoidal oscillator.

## supersquare

This module is based on the same parameters as supersaw but with replacing the sawt

1.	0.5	0.5	0.5	0.5
Pitch	Amp	PW	2WVol	2WDet
60.	0.5	0.5	0.5	5.
0.	0.	0.	0.	0.
+ - supersquare D				

### *Pitch*

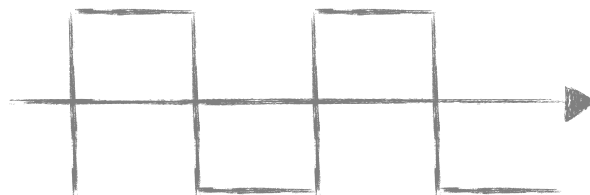
Defines the frequency of the central oscillator as a float MIDI pitch.

### *Amp*

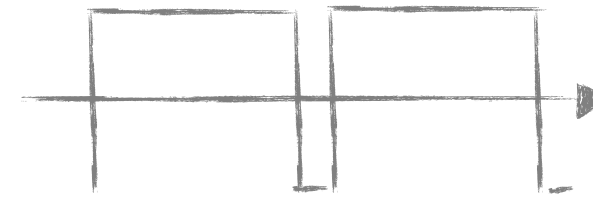
Defines the final amplitude of the oscillator.

### *PW (Pulse Width)*

Defines the Pulse Width of the square wave. A value of 0.5 means that the pulse width is 50% of the signal period.



50% pulse width



80% pulse width

### *2WVol (Secondary Waves Volume)*

Defines the volume of two other square waves detuned on each side of the central sinusoid.

### *2WDet (Secondary Waves Detune)*

Defines the detuning of the two other square waves in cents above and below the frequency of the central square oscillator.

## syncfno

This module is Low Frequency Oscillator which can synchronized with the Global tempo of NMI.

### *Wave*

Choose a wave for the lfo between **Sin** (sinusoid), **Saw** (sawtooth), **Sqr** (square), **S&H** (Sample & Hold).

*Rate*

Defines the frequency of the Low Frequency Oscillator specifying a Max notevalue.

*Amp*

Defines the maximum amplitude of the LFO.

*Input trigger*

A trigger connected to the lfo restarts its phase.

**vsti**

This module allows to play any VST instrument in NMI. It can be controlled in two manner : you can connect it to a trigger and to a signal as the previous oscillators we covered or you can use an internal MIDI port. This second solution allows to use the polyphony of the instrument.



*Pitch*

Define the pitch that is send to the vsti when a new trigger is received from the leftmost input of the device.

*Par1 to Par4*

Allows to assign an input signal to a parameter of the plugin. Use the little menu bellow the value to assign a parameter.

*Par5 and Par6*

Those two extra parameters allow you to change two other parameters of the plugin between two presets.

*Plugin Menu*

Choose the plugin you want to play with that menu. Please be aware of the plugin type. Nothing allows to know if a plugin is an instrument or a classical VST effect...

*Program Menu*

With most of the plugins, a few presets are available. They can be loaded or reloaded using this menu.

### *Trsp (Transposition)*

This value in semitones allows to transpose all the incoming MIDI notes. It also transposes the pitch generated by the Pitch and Trigger inputs.

### *RcvCtl (Receive Controllers)*

This toggle must be on if you want to receive the MIDI controllers directly from the MIDI Port. It could be useful to play with classical MIDI controllers as the Modulation Wheel (Controller 1) or any other controllers.

### *Low*

Defines the lowest note of the keyboard area used by the vsti.

### *High*

Defines the highest note of the keyboard area used by the vsti.

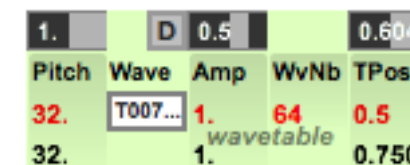
### *MIDI Port*

Defines which MIDI port is used to receive external MIDI notes.

## **wavetable**

This module use the 2d.wave~ object to read wavetables assembled in the same sound file. The wavetable synthesis was first introduced in some hardware synths as the PPG Wave 2.3 and PPG Waveterm.

Many Wavetables can be found as .wav files on the web now. Look at <http://www.ppg.synth.net/download.shtml> to find the original PPG Waveterm library.



1.	D	0.5	0.604
Pitch	Wave	Amp	WvNb TPos
32.	T007...	1.	64 0.5
32.	1.	1.	0.750

### *Pitch*

Defines the pitch of the oscillator in float MIDI note number.

### *Wave Menu*

This menu lists the waves available in your NMI project.

### *Amp*

Defines the amplitude of the module.

### *WvNb (Number of Waves)*

This defines the number of waves in the current wavetable.  
PPG wavetables has usually 64 waves but this number can be different depending of the origin of your wavetables files.

### *TPos (Time Position)*

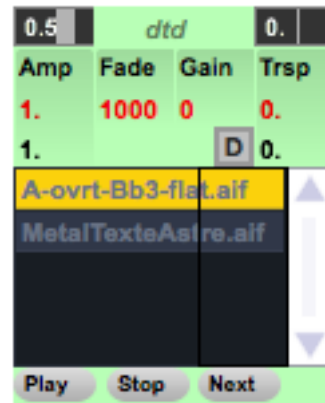
Defines which wave is played as a number between 0. and 1. .  
Going from 0. to 1. means that all the waves will be played successively. 1. corresponds to the first wave again allowing to loop between the end and the beginning of the wavetable.



# Sound player modules

## dtd (Direct to Disk)

This module was programmed to play long sound files during the performance.



### Amp

This parameter defines the final amplitude of the module.

### Fade

Between two sound files, the module performs a fade in fade out. This parameter defines its length.

### Gain

Applies a gain change in dB to the current file.

### Trsp (Transposition)

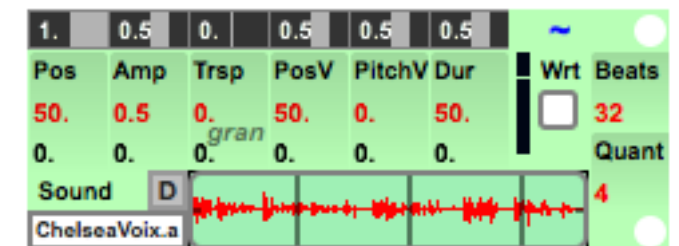
Transposes the current file in semitones by changing the file speed. To transpose a file without changing the speed see **supervp.trans** or **harmo** in the Effects section of the module palette.

## Sound List

This list reflect the content of the **dtd folder** of your project. They appear in the list in alphabetic order. To play a sound, select it and push the **play** button below the playlist. Use the **stop** button to stop the lecture. Use the **next** button to play the next file of the list and perform a fade out on the previous one.

## granular

This module performs a granular synthesis with the Ircam **sogs~** Max object.



## Sound Menu

This is where you choose the sound to granularize.

### *Pos*

Defines the position where to read in the sound file as a percentage of the file length.

### *Amp*

Set the amplitude of the module.

### *Trsp (Transposition)*

Transposes the grain by a certain value in cents.

### *PosV (Position Variation)*

Adds random to the grain position. The value is the maximum value in milliseconds of that random.

### *PitchV (Pitch Variation)*

Introduces random Pitch variations in cents.

### *Dur*

Defines the Duration of the grains.

### *Audio Input*

This input allows to record a new buffer to granularize in real time.

### *Wrt (Write)*

Switches on a function which write the new recording on the disk, in the **sounds** folder of your NMI project. It becomes then available for the other players.

### *Beats*

Defines the recording duration in beats at the current tempo of the Transport module.

### *Quant*

Fixes a quantize value when starting a new recording. For example 4 means that NMI will wait the first beat of the next measure to start the recording.

### *Input Trigger*

Starts a new recording.

### *Trigger Output*

Sends a trigger when the recording is finished.

### **groove**

This module embeds the groove~ Max object to play loops in a sound file.



### *Sound Menu*

This is where you choose the sound to play.

### *Trigger input*

Restart the loop from its beginning.

### *Amp*

Set the final amplitude of the module.

### *Spd (Speed)*

Change the speed of the player.

### *LpStart (LoopStart)*

Defines the loop start as a percentage of the sound length.

### *LpLen (LoopStart)*

Defines the loop length as a percentage of the sound length.

### *Audio Input*

This input allows to record a new buffer to granularize in real time.

### *Wrt (Write)*

Switches on a function which write the new recording on the disk, in the **sounds** folder of your NMI project. It becomes then available for the other players.

### *Beats*

Defines the recording duration in beats at the current tempo of the Transport module.

### *Quant*

Fixes a quantize value when starting a new recording. For example 4 means that NMI will wait the first beat of the next measure to start the recording.

### *Input Trigger*

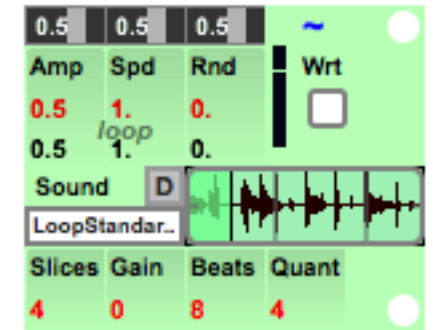
Starts a new recording.

### *Trigger Output*

Sends a trigger when the recording is finished.

## **loop**

The loop module splits a loop into slices and replay them synchronized with the tempo of NMI.



### *Amp*

Set the final amplitude of the module.

### *Spd (Speed)*

Change the speed of the player.

### *Rnd (Random)*

Defines purcentage of chance to trig a random slice.

### *Slices*

Defines in how many slices each beat is splited.

### *Gain*

Applies a gain in dB to the loop.

Audio Input

This input allows to record a new buffer to granularize in real time.

Wrt (Write)

Switches on a function which write the new recording on the disk, in the **sounds** folder of your NMI project. It becomes then available for the other players.

Beats

Defines the recording duration in beats at the current tempo of the Transport module.

Quant

Fixes a quantize value when starting a new recording. For example 4 means that NMI will wait the first beat of the next measure to start the recording.

Input Trigger

Starts a new recording.

Trigger Output

Sends a trigger when the recording is finished.

sampler

Sound	BPitch	Low	High	Vel	Det	Gain	Atk	Rel	Mode	LpStrt	LpEnd	Fade	1	
ictus-flute-C3.	C3	C-2	G3	L	0	0	0	0.	1183.	Loop	1287.	3616.	21	1-2
ictus-flute-C4.	C4	G#3	G4	L	0	0	0	0.	1071.	Loop	0.	4906.49	34	1-2
ictus-flute-C5.	C5	G#4	G8	L	0	0	0	0.	1302.	Loop	0.	4418.34	19	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
...	C3	C-2	C-2	L	0	0	0	0.	100.	Normal	0.	0.	0	1-2
sampler													D	

This module allows to play 12 samples as with a classical sampler. For each keygroups, those function are available:

Sound

Defines which sample is played by the keygroup. This menu reflects the content of the *sound* folder of your project.

BPitch (Base Pitch)

This defines the root pitch of the sample meaning that if the same pitch is played with a MIDI keyboard, no transposition will be applied. This parameter is also used in order to transpose the sample. There is no separate transposition parameter in this module.

### *Low (Low pitch)*

Defines the low pitch limit of the keygroup.

### *High (High pitch)*

Defines the high pitch limit of the keygroup.

### *Learn*

Use this button to learn the keygroup area automatically. Press the button and play two notes simultaneously on your MIDI keyboard in order to define the area.

### *Vel*

Defines the amount of modulation on the volume of the keygroup by the MIDI velocity.

### *Det (Detune)*

Detunes the keygroup of the value in cents.

### *Gain*

Defines a gain in dB for the keygroup.

### *Atks*

Defines the Attack time of the simple Attack/Release envelope included in each keygroup.

### *Rel*

Defines the Release time of the simple Attack/Release envelope included in each keygroup.

### *Mode (Play Mode)*

Defines how the sample is played by the sampler :

Normal: the sample is played from its beginning to its end. The sound stops when the key is released.

One Shot: Normal : the sample is played from its beginning to its end even the key is released.

Loop: plays the sample from its beginning and loops between the *Loop Start (LpStart)* and the *Loop End (LpEnd)*. The sound stops when the key is released.

TgLoop (Toggle Loop): same as loop but this time the MIDI key acts like a on off switch.

RevLoop (Reverse Loop): plays the sample from its beginning and loops backward and forward between the *Loop Start (LpStart)* and the *Loop End (LpEnd)*. The sound stops when the key is released.

TgRevLoop (Toggle Reverse Loop): same as loop but this time the MIDI key acts like a on off switch.

### *LpStrt (Loop Start)*

Defines the loop start time in ms of the keybroup.

### *LpEnd (Loop End)*

Defines the loop end time in ms of the keybroup.

### *Fade*

Defines a fade time as a percentage of the loop length. In Loop or TgLoop, the module uses this value to perform a realtime crossfade loop to avoid clicks when looping in the sound file.

## *Output Menu*

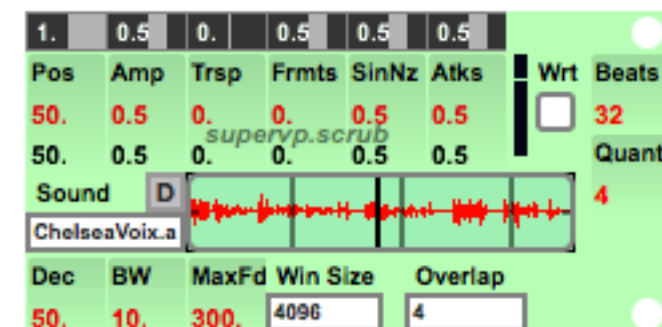
The sampler module has four individual outputs that you can use by pairs or individually. Choose a specific output for each keygroups using the menu.

## *MIDI Port*

Choose the MIDI port to use to receive the incoming MIDI notes in the sampler.

## **supervp.scrub (not in the FREE version)**

This module uses the Ircam supervp.scrub Max object to play a sound file.



## *Sound Menu*

This is where you choose the sound to granularize.

## *Pos (Position)*



Defines the position where to read in the sound file as a percentage of the file length.

### *Amp*

Set the amplitude of the module.

### *Trsp (Transposition)*

Transposes the sound by the value in cents.

### *Frmts (Formants)*

Defines a transposition amount in cents for the formant or the spectral envelope.

### *SinNz (Sinus/noise ratio remix option)*

Defines the balance between the sinusoidal and noisy components of the sound. 0.5 means a normal mix.

### *Atks (Attacks remix option)*

Defines the level of attacks in the sound. 0.5 means a normal mix. 0 removes all the attacks. With 1., just the attacks are heard.

### *Dec (Attacks Decay Time)*

Defines a decay time applied after each attacks of the sound. To well listen the effect of that parameter, the *Atk* parameter should be set to 1.

### *BW (Sinus/noise Bandwidth)*

Defines a threshold between the sinusoidal and noise part of the sound.

### *MaxFd (Maximum of the fundamental)*

This value helps to perform a better transposition of the formants. When playing a monophonic sound, this corresponds to the highest pitch in the sequence.

### *Win Size (Window Size)*

Sets the number of sample used in each analysis in the `supervp.scrub` object. A value of 4096 samples works fine for most polyphonic sounds. If you work on rhythmic material or

if you want to reduce the latency, you should change this value to 2048 or 1024.

### *Overlap*

Sets how many analyses are preformed during the time of one analysis.

### *Audio Input*

This input allows to record a new buffer to granularize in real time.

### *Wrt (Write)*

Switches on a function which write the new recording on the disk, in the **sounds** folder of your NMI project. It become then available for the other players.

### *Beats*

Defines the recording duration in beats at the current tempo of the Transport module.

### *Quant*

Fixes a quantize value when starting a new recording. For example 4 means that NMI will wait the first beat of the next measure to start the recording.

### *Input Trigger*

Starts a new recording.

### *Trigger Output*

Sends a trigger when the recording is finished.

# Effects Modules

## functions common to all effects module



### *On Off switch*

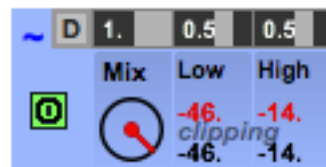
Switches the effect on and off.

### *Mix*

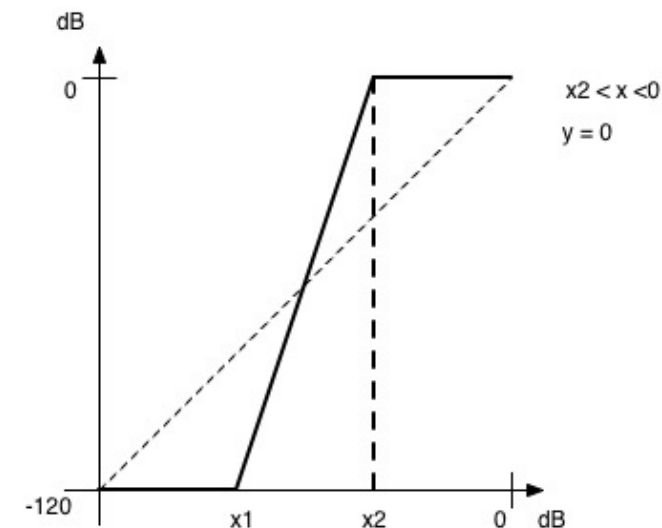
Defines the balance between the direct and the transformed sound.

## clipping

This effect is a recreation of the AudioSculpt™'s clipping effect with standard Max objects. The main idea of this spectral effect is to change the repartition of the energy for each frequency band of an FFT analysis. It acts as like a spectral waveshaping or noise gate. Refer to the AudioSculpt documentation for more informations :



<http://support.ircam.fr/docs/AudioSculpt/3.0/co/Clipping.html>



Input/output level table used for each frequency band in the clipping effect

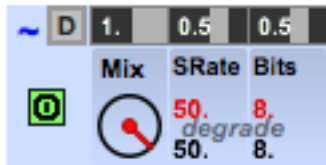
### *Low (Low Threshold)*

Defines the *Low threshold* of the clipping in dB (x1 on the graphic). This corresponds to the minimum volume that a frequency band as to reach to begin to be heard. By increasing this value, you remove the lowest noises in the sound.

### *High (High Threshold)*

Defines the *high threshold* in dB of the clipping. All frequency band above this level are set to the maximum output volume.

## degrade



This effect embeds a *degrade* Max object to reduce the sampling rate and the bit rate of the signal.

### *SRate (sampling rate)*

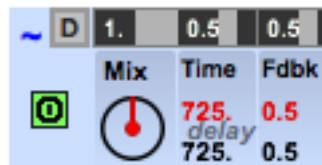
Defines the effective sampling rate. This parameter degrades the frequency bandwidth of the signal.

### *Bits*

Defines the word size in bits. When reducing this value, you will degrade the signal/noise ratio and add noise to the signal.

## delay

This module is a classical delay line with feedback.



### *Time*

Defines the delay time.

## Feedback

Defines the feedback amount.

## disto

This effect provide a distortion by clipping, amplifying and filtering the signal.



### *Dist*

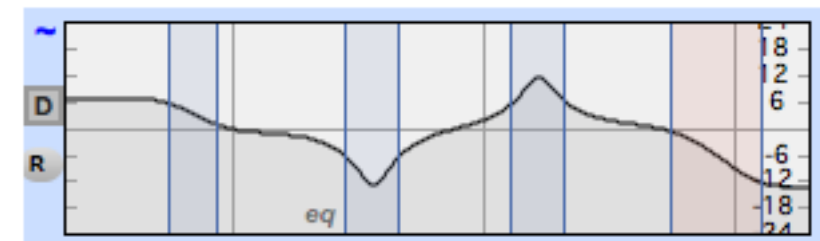
Defines the amount of distortion.

### *Tone*

Defines the color of the distortion. This value changes the frequency of a onepole~ filter connected at the end of the signal path.

## eq

This module provides a classical equalizer with a low and high shelving filters plus two parametric band



filters.

Use the mouse to change the shape of the equalizer. Use the presets or a message box with the content of a dump in order to recall configurations.

## filter

This effect uses the biquad~ Max object to provide a multi-mode filter.



### *Freq (Frequency)*

Define the cut-off frequency in lowpass and highpass mode or center frequency of the filter in bandpass and bandstop mode.

### *Q (Quality factor)*

Defines the amount of resonance around the cut-off frequency in lowpass and highpass mode or the bandwidth in bandpass and bandstop mode.

### *Gain*

Defines a gain in dB applied on all the signal.

## *Filter Mode menu*

Sets the filter mode between lowpass, highpass, bandpass and bandstop.

## *Filter Slope menu*

Defines the slope of the filter in dB/Octave. To produce a 24 db/Octave filter, two biquad in serial are used inside the module.

## freeze



This effect captures up to 100 spectral frames and recombines them randomly in order to produce a freezed sound.

### *Frames*

Defines how many frames are used to generate the freeze. Increase this value to add liveness to the sound.

### *Gain*

Applies a gain in dB to the signal.

*Input trigger*

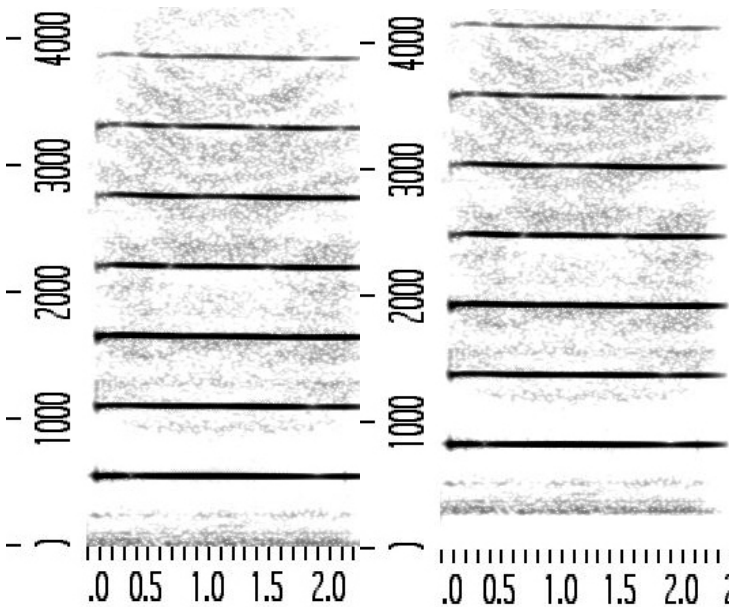
Use this input to trig the freeze effect.

**freqshift**

This module combines a traditional frequency shift effect. It is completed by a delay line with feedback to apply successive frequency shift on the signal.



A frequency shift sounds very different from a transposition.



Because all the frequencies are shifted by the same amount, we loose the harmonic relationship between the partials and the perceived pitch becomes blur.

Spectrum of an harmonic sound and spectrum the same sound frequency shifted

*Freq*

Defines the shift of the spectrum in hz. This value can be positive or negative.

*Time*

Defines the time of the feedback delay line in ms.

*Fdbk (Feedback)*

Defines the feedback amount.

**harmo**

This effect is a classical harmonizer that uses a modified version of the famous Ircam harmv2~ abstraction. It is completed by a delay line to apply many transpositions to the signal.



### *Trsp (Transposition)*

Defines a transposition amount in cents.

### *Win (Window)*

Sets the duration of the time window used inside the module in ms.

### *Time*

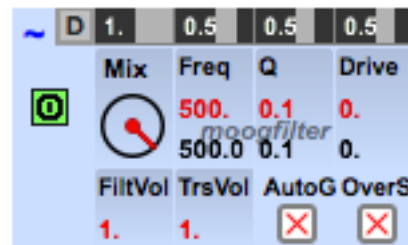
Defines the delay time of the feedback delay line.

### *Fdbk (Feedback)*

Sets the amount of feedback.

## **moogfilter**

This effect uses an Ircam moogfilter~ Max object to emulate the classical Moog™ 24dB/octave ladder filter. The particularity of this implementation is that its possible to set the volumes of the linear part and non-linear part (containing distortions) of the filter separately.



### *Freq*

Sets the cut-off frequency of the filter.

### *Q (Quality factor)*

Sets the resonance amount around the filter cut-off frequency.

### *Drive*

Amplify the signal at the input of the filter in order to excite more the non-linear part of the filter and add more natural distortion.

### *FiltVol (Filter Volume)*

Amplitude of the linear part of the filter.

### *TrsVol (Transients Volume)*

Amplitude of the non-linear part of the filter more excited by the transients of the signal. Combined with the *FiltVol* value this parameter can be used as a transient designer.



### *AutoG (Auto Gain)*

When this option is on, the filter automatically compensates the drive gain. This allows to add colors to a sound by using the drive without changing the global volume of the signal.

### *OverS (OverSampling)*

This switches on the oversampling in the Max object to increase the accuracy of the filter in the high frequencies.

### **psych**

This effect use the Ircam psych~ Max object to perform a transposition. This object was optimized to transpose monophonic sounds like a voice or wind instruments. It can transpose the pitch and the spectral envelope separately. This results in a more natural transposition avoiding the famous "Mickey Mouse" effect.



### *Trsp (Transposition)*

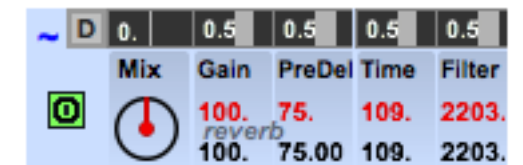
Defines the transposition amount in cents.

### *Frmts (Formants)*

Defines the transposition amount for the formants.

### **reverb**

This module provides a stereo reverb effect. It embeds the Ircam rev4~ abstraction completed by a delay line to add a pre delay and filters to smooth the reverb in the high frequencies.



### *Gain*

Gain of the reverb.

### *PreDel (Pre delay)*

Adds a pre delay in ms. Increasing this value with the decay *Time* value helps to emulate a bigger space.

### *Time (Decay Time)*

Sets the decay time of the reverb with values between 0 127. A 127 value produces a infinite reverb effect.

## *Filter*

Sets the cut-off frequency of two lowpass filters connected after the rev4~ object. Those filters help to smooth the sound of rev4~ in the high frequencies.

## **ringmod**



This module performs a classical ring modulation with an internal sinusoidal oscillator or using an external signal connected to the rightmost input of the box.

## *Freq*

Changes the frequency of the sinusoid that multiply the signal.

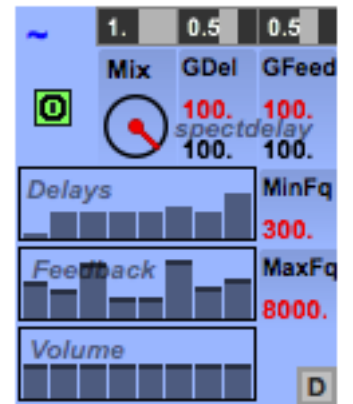
## *Mode*

Sin: in that mode, the module uses its internal sinusoid.

Ext: in that case, an external signal is used.

## **spectdelay**

This effect provides an 8 bands spectral delay with feedback.



## *Delays Multislider*

This multislider sets the duration of the delay for the 8 bands in 16th notes at the tempo defined by the Global Transport of NMI.

## *Feedback Multislider*

Set the feedback amount for the 8 frequency bands.

## *GDel (Global Delay)*

This parameter allows to change all the delays by applying the same factor to all the values. 100% means that the module uses the delay values displayed on the slider ; 50% means half of the values, etc.

## *GFeed (Global Feedback)*

This parameter allows to change all the feedbacks by applying the same factor to all the values. 100% means that the module

uses the feedback values displayed on the slider ; 50% means half of the values, etc.

*MinFq (Minimum Frequency)*

Defines the frequency limit between the first and the second frequency band.

*MaxFq (Max Frequency)*

Defines the frequency limit between the 7th and the 8th frequency band.

**supervp.trans**

This effect uses the Ircam supervp.trans object to apply transposition on the signal. This effect is able to transpose the pitch and the spectral envelope separately as psych but with a better quality. It also allows to use the remix options to drastically change the spectral content of the signal.



a

*Trsp (Transposition)*

Transposes the sound by the value in cents.

*Frmts (Formants)*

Defines a transposition amount in cents for the formant or the spectral envelope.

*SinNz (Sinus/noise ratio remix option)*

Defines the balance between the sinusoidal and noisy components of the sound. 0.5 means a normal mix.

*Atks (Attacks remix option)*

Defines the level of attacks in the sound. 0.5 means a normal mix. 0 removes all the attacks. With 1., just the attacks are heard.

### *Dec (Attacks Decay Time)*

Defines a decay time applied after each attacks of the sound. To well listen the effect of that parameter, the *Atk* parameter should be set to 1.

### *BW (Sinus/noise Bandwidth)*

Defines a threshold between the sinusoidal and noise part of the sound.

### *MaxFd (Maximum of the fundamental)*

This value helps to perform a better transposition of the formants. When playing a monophonic sound, this corresponds to the highest pitch in the sequence.

### *Win Size (Window Size)*

Sets the number of sample used in each analysis in the supervp.scrub object. A value of 4096 samples works fine for most polyphonic sounds. If you work on rhythmic material or if you want to reduce the latency, you should change this value to 2048 or 1024.

### *Overlap*

Sets how many analyses are preformed during the time of one analysis.

### **vst**

This module allows to load a VST plugin in NMI.



### *Par1 to Par4*

Allows to assign an input signal to a parameter of the plugin. Use the little menu bellow the value to assign a parameter.

### *Par5 to Par8*

Those two extra parameters allow you to change two other parameters of the plugin between two presets.

## Plugin Menu

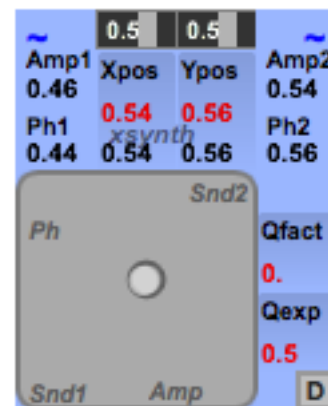
Choose the plugin you want to play with that menu. Please be aware of the plugin type. Nothing allows to know if a plugin is an instrument or a classical VST effect...

## Program Menu

With most of the plugins, a few presets are available. They can be loaded or reloaded using this menu.

## xsynth

This effect uses standard Max object to calculate a cross synthesis between two signals. It mixes amplitudes and phases of FFT of the inputs as the Generalized Cross Synthesis found in AudioSculpt™ as follow:



$$A = (1 - XPos) * Amp1 + XPos * Amp2 + Qfact^{Qexp} * Amp1 * Amp2$$

$$P = (1 - YPos) * Ph1 + YPos * Ph2$$

The 2d slider allows to manipulate amplitudes and phases with one gesture.

## XPos

Defines the X position on the 2d slider.

## YPos

Sets the YPos on the 2d slider.

## Qfact

Sets the value of the q factor. This factor multiply the multiplication of the two amplitudes. These means that this parameter adds more amplitude when the two sounds have energy in the same frequency bands of the FFT.

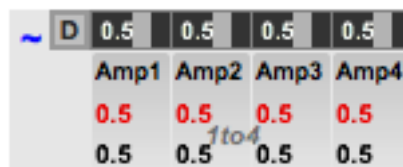
## Qexp

This parameter is the exponent of the q factor. It was added to attenuate the effect of the q factor and avoid eventual distortions.

# Mix Modules

## 1to4

This module allows to route a signal to 4 outputs and control the amplitude on each of them.



### Amp1 to Amp4

Sets the amplitude for each output of the module.

## aux

Allows to get back in the patch the signal coming from the 8 auxiliary buses available on the *Track* module. This allows to apply the same FX scheme to several tracks.



### Aux menu

Sets the aux number.

## gain

Applies a gain to a signal.



### Gain

Gain in dB.

## in

This module catches the signal coming from the input of a sound card connected to Max.



### Input menu

Input number.

## matrix4x4

This box is a 4 by 4 audio matrix. It connects any of the four inputs to any of the four outputs.

Vol	Vol	Vol	Vol	D
0.42	0.69	0.2	0.	
Vol	Vol	Vol	Vol	
0.26	0.93	0.11	0.	
Vol	Vol	Vol	Vol	
0.97	0.045	0.17	1.	
Vol	Vol	Vol	Vol	
1.	0.075	0.27	0.59	

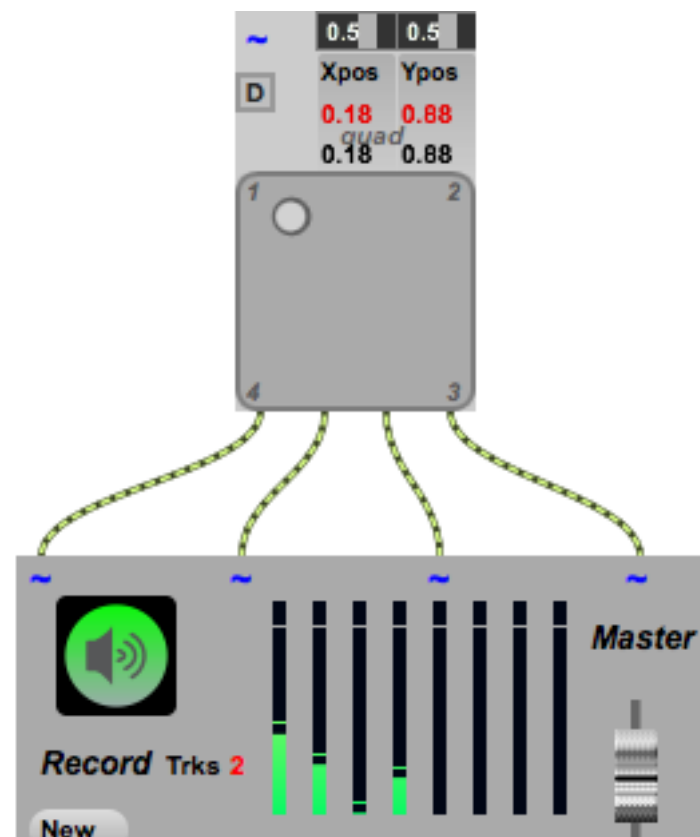
## Amplitude Matrix

Use one of the 16 positions to define the routing inside the matrix. For example, to connect the input 2 with input 4, set an amplitude value as on the following screen capture.

Vol	Vol	Vol	Vol	D
0.	0.	0.	0.	INPUTS
0.	0.	0.	0.	OUTPUTS
0.	0.	0.	1.	matrix 4X4
0.	1.	0.	0.	

## quad

This module is a quadriphonic panner. It can be directly connected on the Master module of NMI.



## XPos

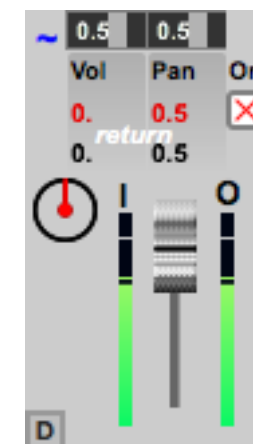
Defines the X position on the 2d slider.

## YPos

Sets the YPos on the 2d slider.

## return

This module is similar to the *Track* module but without the auxiliary buses. It was done to avoid signal loops accidents. Use the *return* module after the signal path connected after an *aux* module.



## Vol (Volume)

Sets the gain in dB of the track return.



### *Pan (Panoramic)*

Sets the pan position. This is a 360° pan which allows to send the signal in any speakers of the setup defined in the *Master* module.

### *On (Trigger input)*

Switches the return track on and off. A trigger connected to the corresponding input of the module can be used to control that switch.

### **spattrack**

This special track module embeds an Ircam spat.spat~ Max object. It offer a complete source and room simulation.

### *Vol (Volume)*

Sets the gain in dB of the track return.

### *Pan (Panoramic)*

Sets the pan position. This is a 360° pan which allows to send the signal in any speakers of the setup defined in the *Master* module.

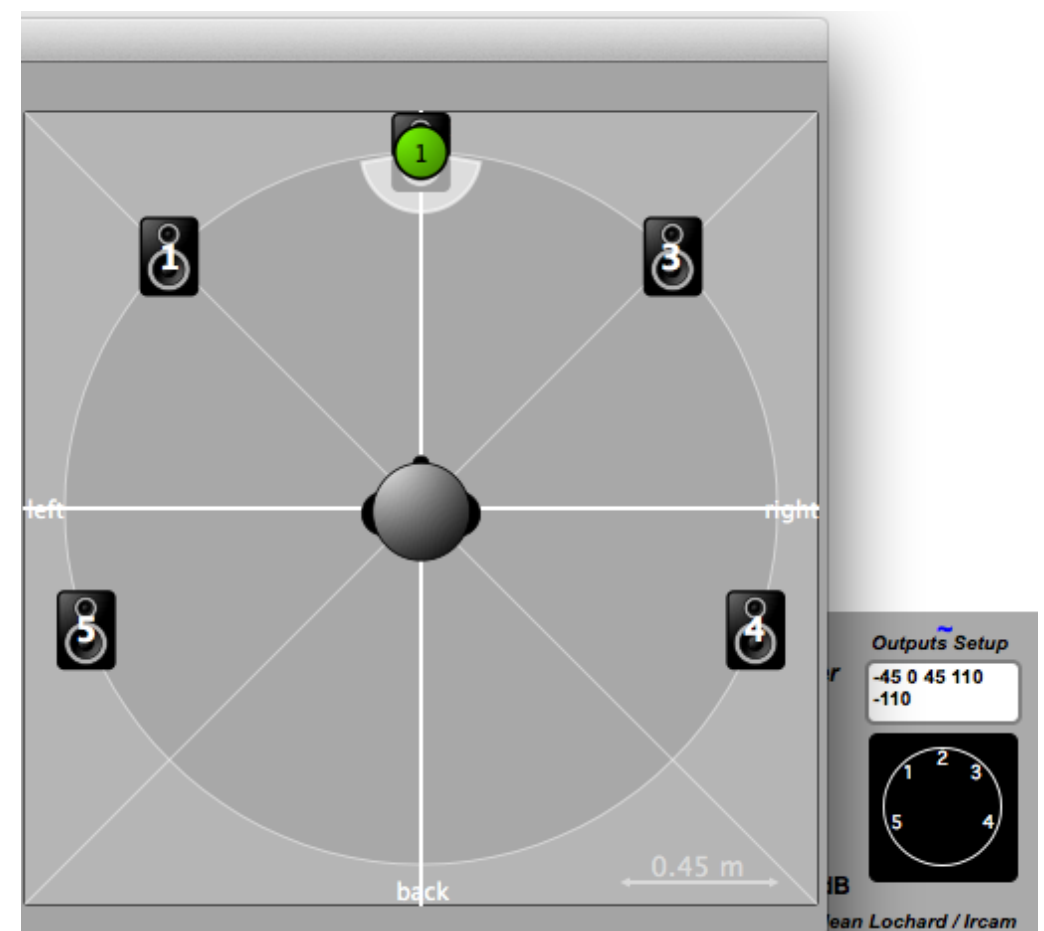


### *On (Trigger input)*

Switches the return track on and off. A trigger connected to the corresponding input of the module can be used to control that switch.

### *Open button*

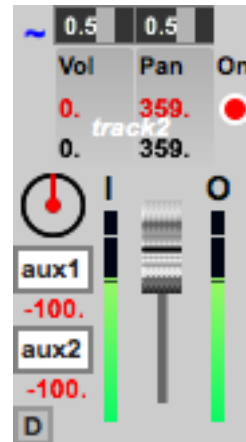
Opens the Spat™ panel. It reflect the speaker configuration that you set in the *Master* module of NMI, as for this classical 5.1 setup :



See the Spat™ documentation for more informations about the function of that panel.

## track

This module makes the connection between the different sound sources available in a patch with the *Master* module of NMI.



### *Vol (Volume)*

Sets the gain in dB of the track return.

### *Pan (Panoramic)*

Sets the pan position. This is a 360° pan which allows to send the signal in any speakers of the setup defined in the *Master* module.

### *On (Trigger input)*

Switches the return track on and off. A trigger connected to the corresponding input of the module can be used to control that switch.

## *Aux Buses menus*

Those menus allows to choose two destination auxiliary buses. The signal is received in an *aux* module.

## *Aux Levels*

Sets a gain in dB for the two auxiliary buses.