

# **Pulsaret M4L**

*Ableton Live Instrument  
granular synthesis device  
{glisson, grainlet, trainlet, pulsar}*

**user manual v.2.3**

[www.apesoft.it](http://www.apesoft.it)

apeSoft © 2019

## **Starting**

[overview](#)

[system requirement](#)

[installation](#)

[copyright](#)

[purchase a license](#)

[update to v.2](#)

[background](#)

[GUI layout](#)

[Quick access](#)

## **Granular streams**

[parameters](#)

[settings](#)

[envelope select](#)

[wave select](#)

## **Windowing**

[general](#)

[shape select](#)

[draw](#)

[additive](#)

[audio slot](#)

[WavePad](#)

[WavePad management](#)

[WavePad display](#)

[WavePad Snap](#)

## **Snapshots**

[store/recall](#)

[micro pad](#)

[transitions](#)

[snapshots manage](#)

[subscribe/un-subscribe transition clients](#)

[transition curve](#)

**Snapshots sequencer improviser unit**[beats cycle](#)[time intervals](#)[step sequencer](#)[output](#)[Note and Tick Value](#)**Hyper Vectorial Pad**[general explanation](#)[pad mouse behavior](#)[X/Y time](#)[miscellaneous](#)**Matrix**[general explanation](#)[parameter linkage](#)[graillet/pulsaret mode](#)[parameters rescale](#)[LFO modulation](#)[status bar](#)**OSC I/O**[header](#)[sync](#)[widget mapping](#)**Overview**[save/load device](#)[info window](#)[float window](#)[history](#)[acknowledgments](#)

## overview

The Granular Synthesis device for [Ableton Live](#) (**Instrument**), implements a prototypes granulator trainlet, grainlet, pulsar glisson etc...

## system requirements

### **Macintosh**

Pulsaret 2 requires a Mac PPC or Intel machine running OS X 10.4 or later, and 1 GB RAM.

### **Windows**

Pulsaret 2 requires a Windows XP/Vista/7 machine and 1 GB RAM.

Pulsaret 2 requires Live 8.2.2 and Max For Live. Details about Max For Live can be found at [Ableton.com](#).

This bundle will only work in [Ableton Live](#) (not in the MaxMSP application).

Max for Live puts the power and potential of Max/MSP inside Live. Create all the instruments, effects and extensions you've ever wanted. Go beyond the common and predictable, and transcend the limits that conventional tools impose. Build completely unique synths and effects, create algorithmic composition tools, or fuse Live and controller hardware into radical, new music machines. Join a society of makers and share ingenuity.

Max for Live was co-developed by [Ableton Live](#) and [Cycling '74](#).

Pulsaret 2 uses QuickTime in order to read correctly media files (including MP3), therefore QuickTime must be installed on your system. On Windows we recommend a complete installation of QuickTime choosing all optional components.

## copyright

This program is copyright shareware and it is not freeware.

You can download the unregistered version of the program and give it to your friends or to any other person as long as for no charge. This program cannot be distributed in shareware compilations CDs without prior written approval from the author.

No responsibility is taken for any damage or losses caused by this package.

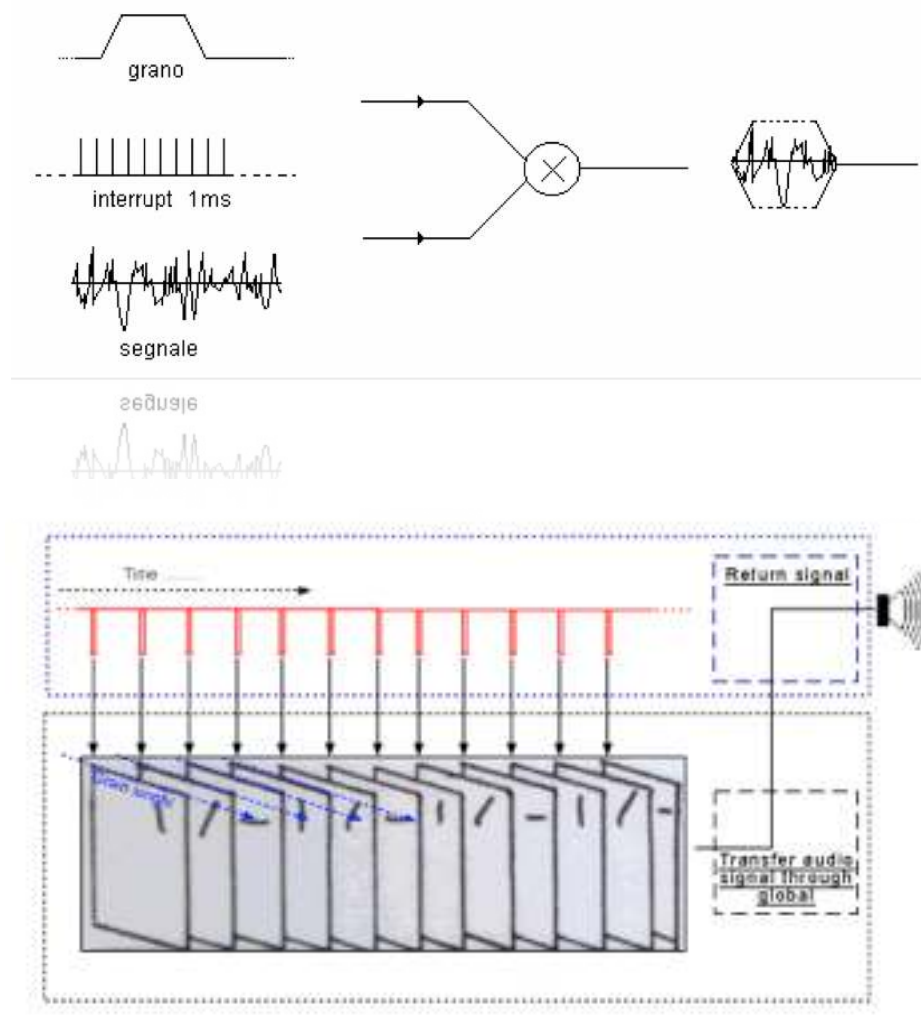
All program trademarks belongs to its respective author.

## background

The first official release of Density (2001), developed in [Csound](#) language and based on [Eugenio Giordani](#) 's GSC4 (Granular Synthesis for Csound). GSC4 was the first patch for granular synthesis on Csound implementing [Barry Truax](#) model.

Pulsaret 2 can generate thousands of grains dynamically, I preferred this way, rather than a fixed number of "voices" (oscillators). The overlapping factor grains, depends only on the actual CPU power. Thus you have not limits in grains number for second (density).

Below an easy granulation model.



DensityGSC (Csound version) is still available for free at: [www.alessandro-petrolati.it/densitygsc.html](http://www.alessandro-petrolati.it/densitygsc.html) it works on Windows XP but not in Windows Vista, it seems magically resurrected with Windows7. DensityGSC is a discontinued product.

New Pulsaret 2 is completely rewritten in [Max/Msp 5](#), available for both Macintosh and Windows standalone applications and M4L (Max For [Ableton Live](#)) devices. More stable, flexible, improved audio quality, restyling GUI look (Graphic User Interface) with native effects Hv\_pads, FiltersEQ, Snapshots Sequencers improviser unit etc...

## GUI layout

The most important Pulsaret 2 parameters are placed inside of only one window. Pulsaret 2 GUI (Graphic User Interface) is divided into parts, Quick Access, Windowing Module, Granular parameters and Snapshots-presets.

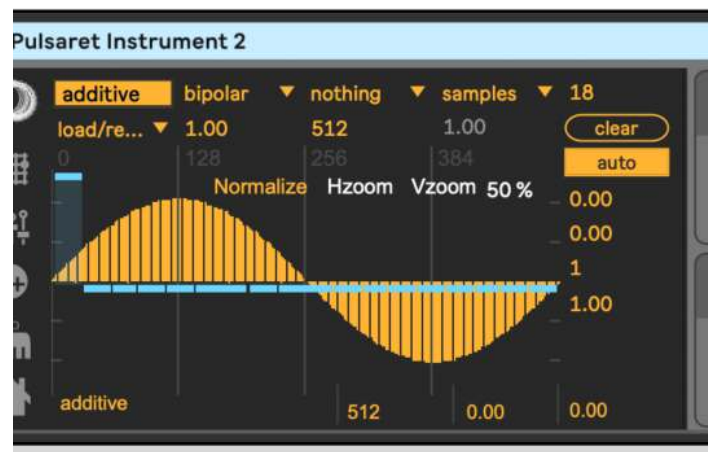
### *Pulsaret 2 Instrument*



### Quick Access

You can quickly open (From left to right)

- Snapshots sequencer improviser unit
- Hyper Vectorial Pads
- OSC I/O mapping
- About



## Windowing

### general

The windowing module generates some classic envelopes and prototypes used for smoothing the grain amplitude. The currently selected window, it will be employed for the grain envelope during granulation. You can deforming some envelopes shapes like **gauss**, **curve** or **additive**, draw a new shape freehand or through **draw** prototype. Also you can load an audio file from disk (until 6 audio files, Aiff Wav or Mp3 supported). Almost all Windowing WavePad functions are identical to [Granular Streams](#) WavePad.

See [WavePad](#), [WavePad display](#), [WavePad Snap](#), [WavePad mouse](#) for more details.

Unlike Granular Streams WavePad, the Windowing WavePad buffer management have two additional functions:

## reset shape

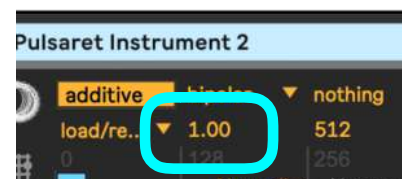
When you select from menù the first of them, current envelope or shape will be restored at the original shape.

When you select reset all shapes, all prototypes and envelopes they will be restored at the original shape.

**resetshape** and **resetallshapes**, they work on the sixteen envelopes/prototypes restoring originals shape. While **clear** work either on the sixteen envelopes/prototypes either on the six sound files slots, writing 0 in the all buffer samples.



Windowing have an important parameter that allow you to chose a **size** in samples for prototypes or envelope shape. You can also resize buffer dynamically, entering a new value expressed in samples (512 default).



N.B. when you resize a shape, you will reset original default shape.

## shape select

There are 16 pre-generated shapes:

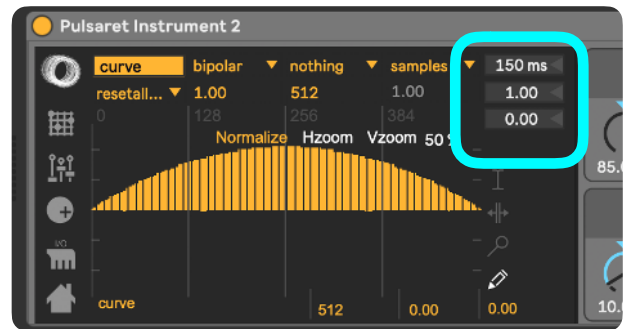
rectangle,  
blackman,  
**curve**,  
**gauss**,  
hamming,  
hanning,  
**draw**,  
sine,  
cosine,  
triangle,  
saw,  
square,  
sigmoid,  
sin(x)/x,  
random,  
**additive**

Those above listed in **bold** have additional parameters which are displayed when selected:



**curve** shows three number boxes, on the top (top right 150 ms) we have deformation update rate in milliseconds.

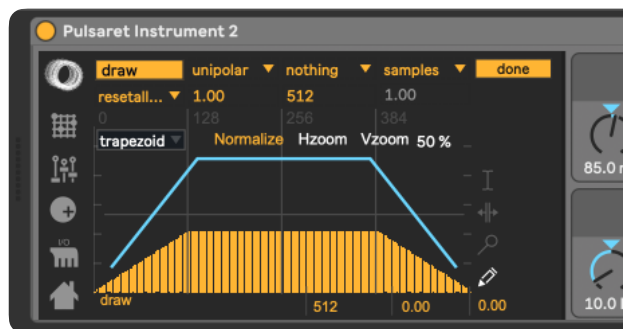
Since these parameters work in real time, you can set a speed limit time to avoid cpu overhead. The other two parameters (1.00 and -0.72) set the middle/border curve deformation.



Like curve, **gauss** shows on the top the speed limit update rate and only a “Gaussian standard deviation” parameter.

## draw

In Pulsaret 2, the grain envelope shape default is **draw** (bold italic), when selected appear yellow button (draw/done). This allows you to superimpose at the WavePad a function **break-points** where you can track the envelope segments. You can add new break-points by clicking on the function superimposed pad or shift + click on a specific break-point to remove it.

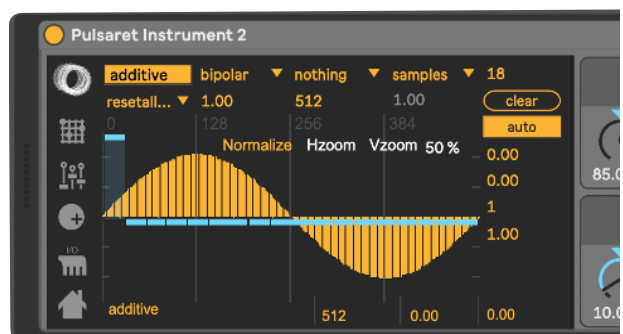


Defaults shapes are available and include, **adsr** (Attack Decay Sustain Release) and a **trapezoid** (default grain envelope). You can switch between **unipolar/bipolar** automatically when you change view perspective in windowing WavePad, (see [Granular Streams WavePad display](#) for more explanations).

## additive

**additive** generates a composite waveform through weighted sums of sinusoids. It works similarly to Csound Gen19. Each slider in the additive pad, controls a specific partial amplitude. You can change the total harmonics number for the spectrum (top right numbox 18), **clear** all amplitude contributes of spectrum (bringing to 0).

Toggle on **auto**, you will enable amplitude auto-rescale, thus the resulting shape it will be always normalized to one. N.B. **phase** and **offset** are normalized ( $0 \div 1$ ). The two below num-box of **gen** button, sets the fine amplitude of a specific partial. You need selects a **partial number** (1 in the example), enter the **amplitude value** (0.5 in the example) at last click on **gen** to apply change.





## audio slot

In the shape select menu, after the sixteen envelopes/prototypes you will see six empty slots named: **usershape1**, **usershape2** etc.. You can employ them to load until six audio files from disk, just like [Granular streams WavePad](#). See for further details. If cannot load, or drag, a sound file in a shape slot, you must employ one of the six available slots.

## WavePad

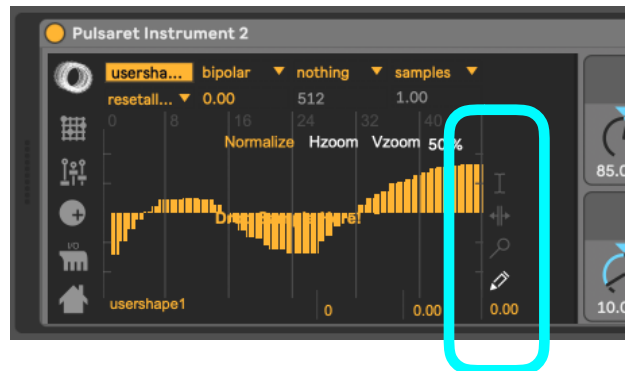
On the right side of the WavePad, you can choose the type of action: select, loop, move and draw.

The tools **select** (from top to bottom) and

**loop** allows you to select a buffer length

**move** (hand) allows you to select a part (zoom) of prototypes or loaded audio file.

**draw** (pencil) allows you to draw directly on the pad.



## WavePad management

**load/replace** = open a file browser to load an audio file on the selected slot (aiff, wave mp3 supported). Also you can drop a valid file on the wave pad, if it's unsupported, a message will be shown;

**savetofile** = export the contents of the buffer as Wav or Aif audio file;

**openbuffer** = opens the WavePad buffer window, or brings it to the front if it is already open. The window is resizable but not editable, you can **scrolling the mouse over the buffer window**;

**crop** = will trim the audio data to the current selection. It resizes the buffer to the selection length, copies the selected samples into it, and displays the result at default settings. The buffer is erased, except for the selected range. This is a "destructive edit," and cannot be undone;

**clear buffer** = erases the contents of buffer;

**lastsel** = it causes the selection start and end points to revert to their immediately previous values. This is helpful when you are making fine editing adjustments with the mouse and accidentally click in the wrong place, or otherwise cause the selection to change unintentionally. Repeated undo commands will toggle between the last two selection states;

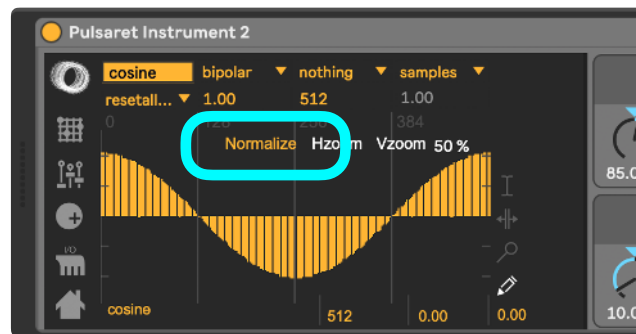
**zoom out** = resets the entire Hzoom (horizontal zoom) time display;





Clicking **Normalize** to scale the sample values in the WavePad buffer (0 dB), so that the highest peak matches the value of 1. Scroll number on the top to manually adjust value.

**e.g.** normalize = 1.50 scale the sample values in the buffer so the highest peak matches the value given by the argument.



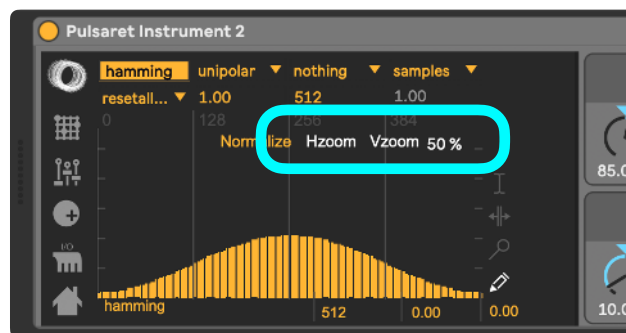
This can cause either amplification or attenuation of the audio, but in either case, every value is scaled. When a sound file is loaded, default value is 0, it means original wave amplitude, value greater than 0 are saved and loaded in the project.

N.B. Normalized value of 0 can be interpreted like: original file amplitude if you have not change it before, normalizing at 0 value (i.e. 0.001 amplitude gain, -60 db).

### WavePad display

Clicking **Hzoom** to reset horizontal zoom (entire time display of file);

Clicking on **Vzoom** you resets vertical amplitude zoom (default 50 %). Mouse scrolling on right number (50 %) to set an amplitude zoom amount. N.B. unlike **Normalize**, **Vzoom** is not a destructive amplitude rescaling but only a graphics rescale.



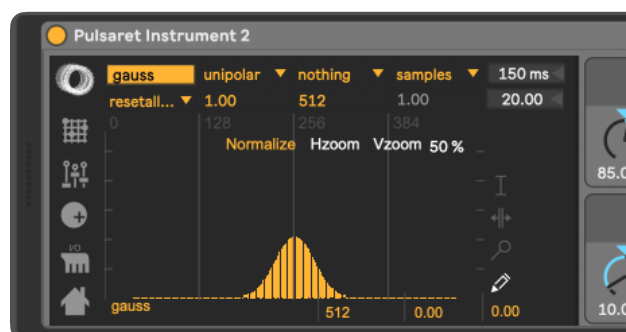
In the image below, the last right menu (**bipolar**) switch to unipolar/bipolar view mode, unipolar mode shows values between 0 ÷ 1 while in bipolar shows values between -1 ÷ 1;

### WavePad Snap

You can set the WavePad **snap mode** selection range:

**bipolar** switch view mode unipolar/bipolar;

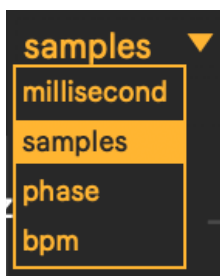
Snap causes the start and end points of the selection to automatically move to specific points in the buffer, defined by the snap mode:



**nothing** disables snap to allow free selection. This is the default;

**grid** specifies that the selection start and end points (grain length) should snap to the vertical grid lines;

**zerocrossing** instead of snapping the selection to a uniform grid, this mode searches for zero-crossings of the [buffer](#) data;



The second menu from the top (millisec...), sets the unit of time measurement used by the display:

**milliseconds** sets the display unit to milliseconds;

**samples** causes time values to be shown as sample positions in the target buffer. The first sample is numbered 0, unless the display has been shifted by the offset message;

**phase** causes time to be displayed according to phase within the buffer, normalized so that the 0 refers to the first sample, and 1 refers to the last; N.B. grid values for phase, must be normalized (0. ÷ 1.), (see later);

**bpm** specifies beats per minute as the time reference unit, relative to a master tempo and number of beats per bar, both of which you can set with the bpm message.

The numbox (1.00) specifies spacing of the grid lines for current unit of time measurement used by the display. N.B. grid values for **phase**, must be normalized (0. ÷ 1.)

## Granular Streams

### parameters

In this section we will examine the Granular Synthesis parameters. Please refer to web for more explanations about granular synthesis.

You can interact with Pulsaret 2 parameters on the GUI (Graphic User Interface) in the follows ways:

Click and drag (scroll) on vertical sliders (or dials) to change the value, (see [mousing](#) mode for more details);

Holding down the Command key (Macintosh) or the Control key (Windows) while mouse scroll, for precise value control;

Select a widget clicking on the name and use keyboard up/down arrows;

Drag up/down the display value (the number box below each parameter);

Click on the number box value, enter a numeric value and press enter key;

Send a MIDI CC or Ableton Live automation, see [Ableton](#) user manual;

The granulation engine is updated every new generated grain, the frequency of grains scattering depending from **density** parameter, expressed in Hz (grains for seconds).

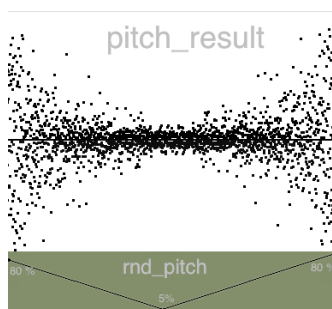
The vertical sliders provides a **deterministic values**, while the dials (knobs) provides **random** values. Summing the two values (deterministic-random) will be produced a value that is passed to the granulator.

In the follow example, the **pitch** value (deterministic) is rescaled with **rnd\_pitch** (random):

$$\text{pitch\_result} = \text{pitch} + (\text{birnd}(\text{rnd\_pitch} * \text{pitch}/2) )$$

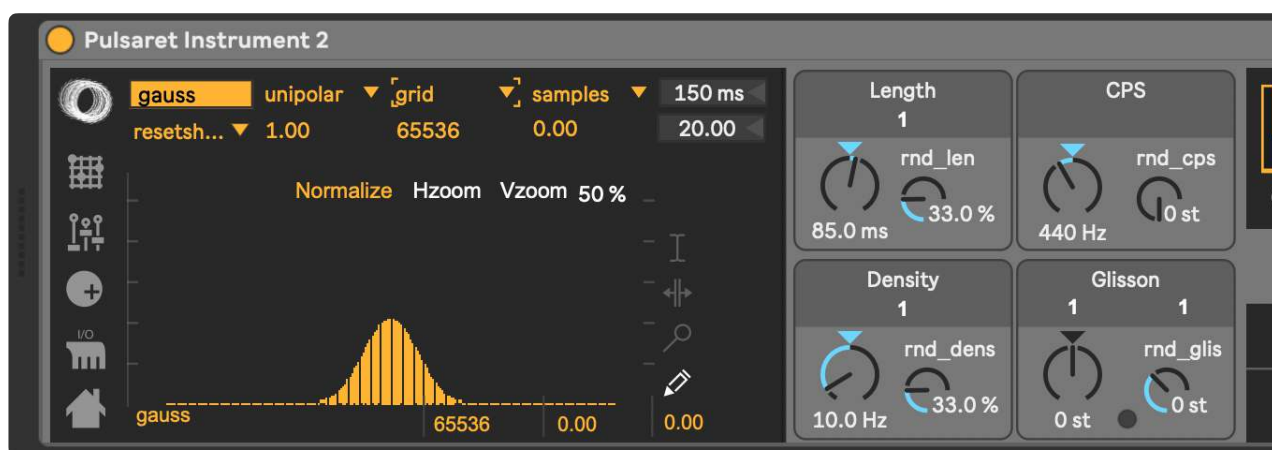
Where **birand** is bipolar random generator. The random range is expressed in %, therefore 0 % it means only deterministic value, 100 % pitch\_result can vary in the range:

$$\text{pitch\_result} = \text{pitch} \pm \text{pitch}/2$$



In the example **rnd\_pitch** (green frame) starting from 80%, is reduced to 5 % in the middle.

With a great **rnd\_pitch**, each grain can have an high random pitch mask range. Carrying **rnd\_pitch** toward lowest values, the random pitch mask is reduced until 0 % which means no random pitch grain scattering.



**length** = grain duration in milliseconds;

**rnd\_length** = random range deviation, in % of length;

**density** = grain for seconds in Hz;

**rnd\_density** = random range deviation, in % of density;

**cps** = grain frequency in Hz;

**freeze button** = (small button in the middle of scanning control) switch between the value 0 (freezes) and the last control value;

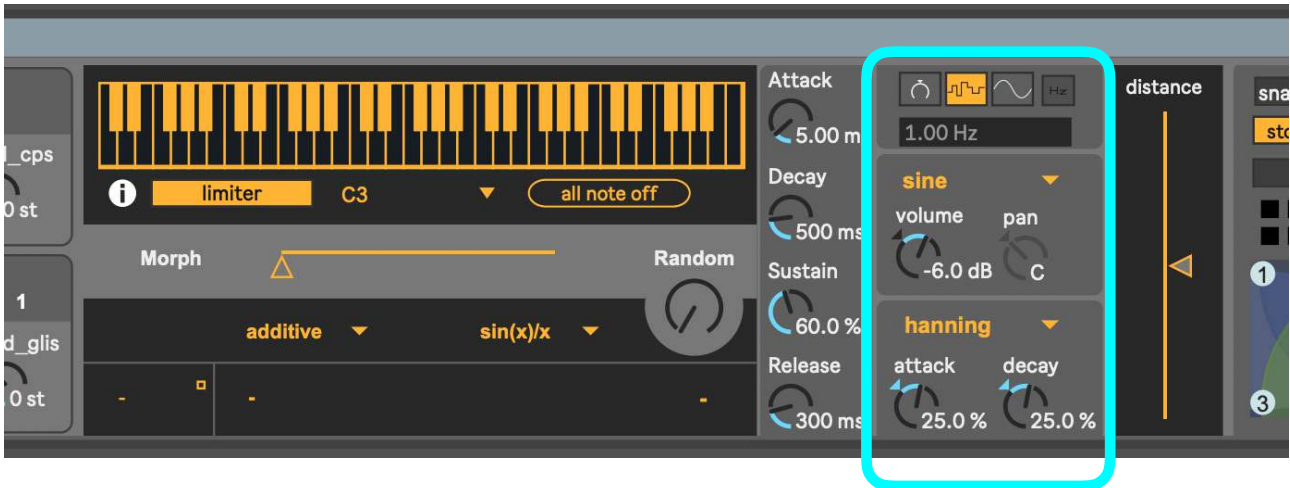
**rnd\_cps** = random range deviation, in % (12 semitone);

**glisson** = glissando semitones inside of the grain;

**rnd\_glisson** = randomly moves semitones glisson (-/+ 12 semitones);

**morph** = morphing between two waveshapes (left=wave\_1, right=wave\_2, middle=mix 50%);

**rnd\_morph** = morphing randomly between two waveshapes, in %;



**volume** = grains amplitude in decibel;

**pan** = left/right grains distributions on the stereo front (only pan = manually);

**distance** = stereo field width;

**panningmode** *tab* = select between: pan (manually), jittering (random), lfo (low frequency modulation);

**beat/Hz** (only lfo) = toggle between beat/Hz; sets the panning duration (left right azimuth) expressed in note-ratios or frequency (Hz). Beat mode is synced with the Ableton Live global transport time, you can change global time (BPM<sup>1</sup>) from Ableton Live transport.

**beat** (only lfo) = each item switch represents the Relative-Tempo expressed in note-ratios; you can change BPM (Beat per minute) from the Ableton Live transport. N.B. you need activate the Ableton Live transport master clock.

**panning rotate frequency** *slider* = when selected lfo, set the panning azimuth in Hz, you can chose among five shapes: **sine**, **triangle**, **saw**, **random** and **pencil**, by selecting **pencil** you can draw inside pan-shapes window your custom curve.

All the granular parameters have a prefixed values range. Some of them have an additional features, you can divide or multiply the parameters output in order to extend range.

The rescaled value it will be passed to granulator engine, but on display you will see old prefixed widget range.

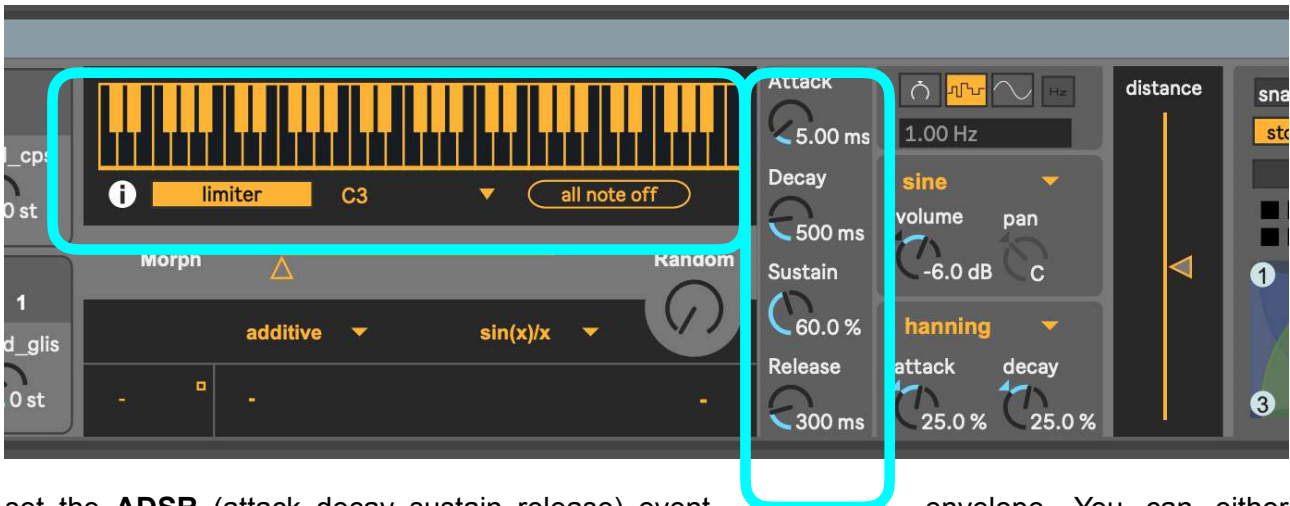
<sup>1</sup> **Beats per minute (BPM)** is a unit typically used as a measure of tempo in music.

The BPM tempo of a piece of music is conventionally shown in its score as a [metronome](#) mark, as illustrated to the right. This indicates that there should be 120 [crotchet](#) beats ([quarter notes](#)) per minute. In simple [time signatures](#) it is conventional to show the tempo in terms of the note duration on the bottom. So a 4/4 would show a [crotchet](#) (or quarter note), as above, while a 2/2 would show a [minim](#) (or [half note](#)). In compound time signatures the beat consists of three note durations (so there are 3 [quavers](#) ([eighth notes](#)) per beat in a 6/8 time signature), so a dotted form of the next note duration up is used. The most common compound signatures: 6/8, 9/8, and 12/8, therefore use a dotted crotchet (dotted quarter note) to indicate their BPM. (Wikipedia)

## settings

Streams setup:

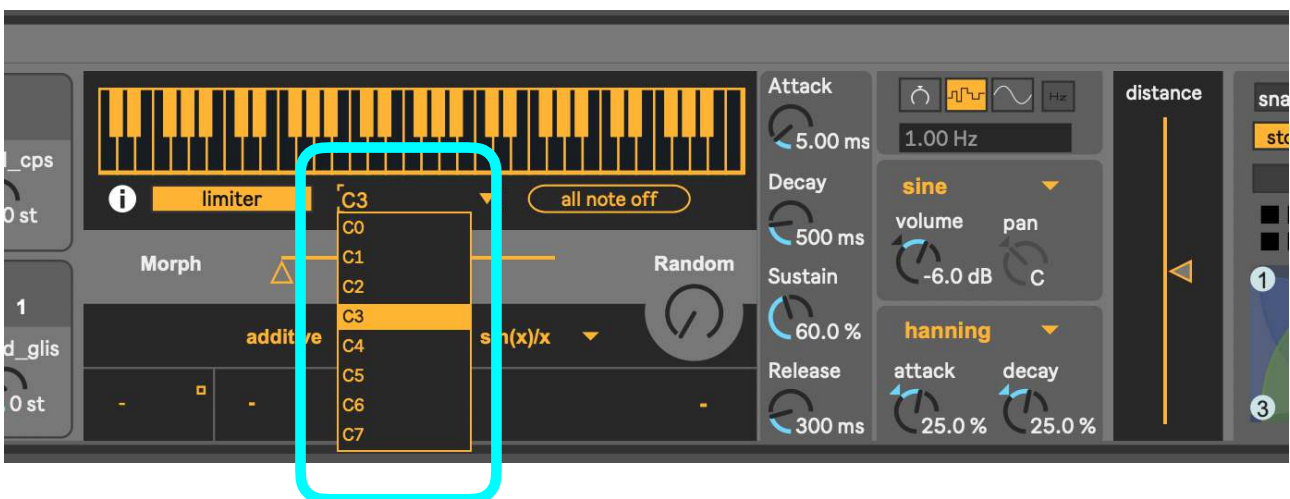
Through the **keyboard** you can **active/de-active** granular streams **polyphonically**, you can also



set the **ADSR** (attack decay sustain release) event envelope. You can either use the MIDI message note on/off to trigger streams remotely.

From left to right, number box (**6** in the image), sets general Synth polyphony; **i** toggle, enable/disable linear interpolation for *sample* and *envelope* oscillators;

**C3** menu is the keyboard octave transposition, C3 key is gray emphasized;



**limiter** enable/disable the peak-limiter which allows for the specified control of signal amplitude;

**all note off** turn off all playing notes by sending a message to each instance with a playing note. The message consists of the MIDI pitch most recently received via the keyboard note or midinote message followed by a 0 (meaning zero velocity or note-off);

most right we have displayed the last note played from keyboard or MIDI event;

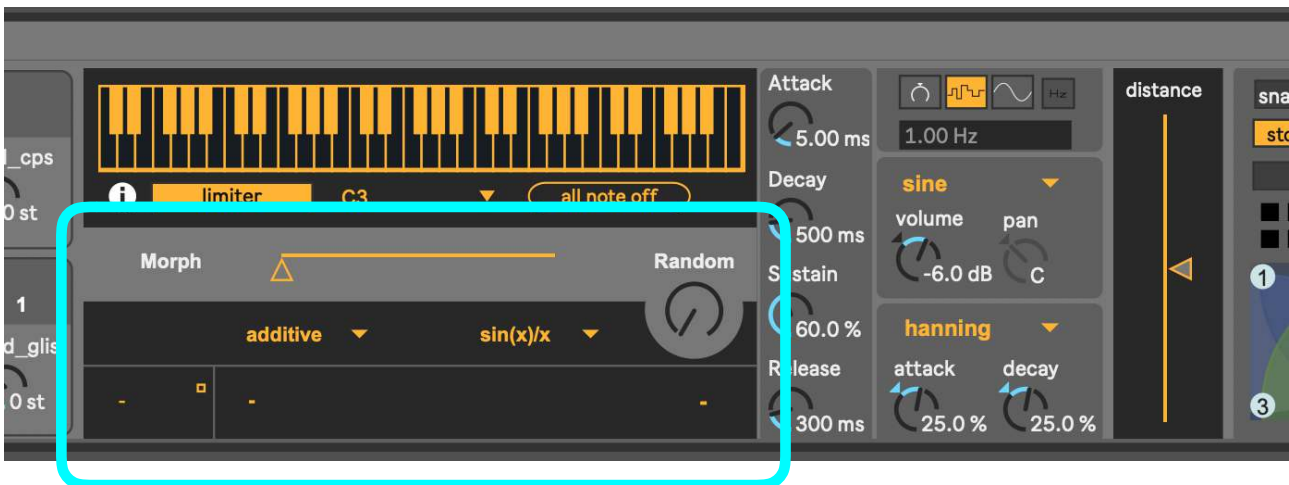
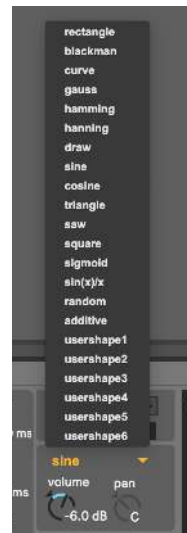
## envelope select

From the windowing envelope pop-up menù, you can choose a shape for granulation or loading until 12 sound files. Each stream can have a different envelope, the default shape is “draw”. when you load an audiofile (or mp3) in windowing, you will see file name in the streams envelope select popup menùs.

see “Windowing” for more explanation.

## wave select

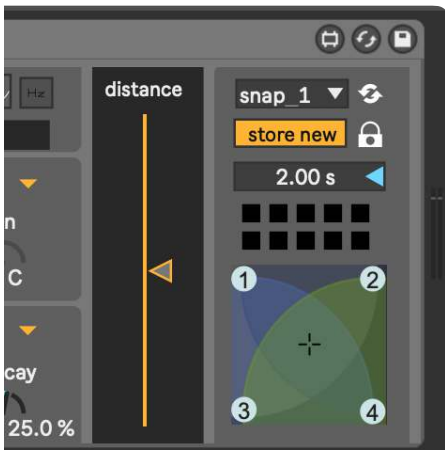
Pulsaret provide a powerfull way to mix two waveshapes, every grain it will be initialized with own % mix of twi shapes. You can reach edge shapes by selecting differents wave shapes from menùs





## Snapshots

### store/recall



A snapshot is a photo of graphical interface (GUI parameters) in the current state. Each snapshots module can store up to 24 snapshots.

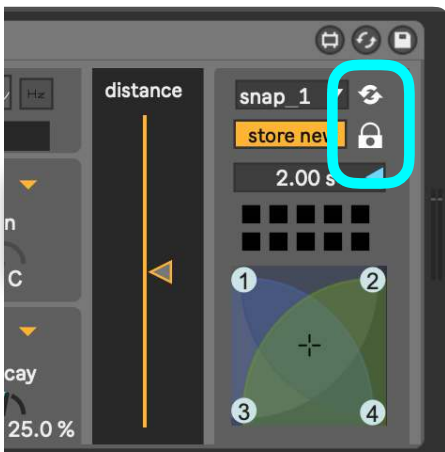
You can make transitions between two or more snapshots, you can subscribe or unsubscribe determinate Pulsaret 2 parameters.

When you find an Interesting "sound" you can store new snapshot just shift + click on a button in the **snap-pad**, thus the preset button will light orange. You can recall a snapshot by simply clicking on the **snap-pad** buttons; all the widgets

subscribed (see later) will be restored at current snapshot value, immediately.

Alternatively a snap can be stored from pop-up menù, a message box will ask you to digit a snap number in (see below for more explanations).

### micro pad



The micro-pad's goal is to obtain intermediate values between four snapshots (interpolation). You can configure four nodes (snapshots).

**locked** (lowest in the image): nodes: the mouse can only edit the nodes position and size. **unlocked**: slider; the mouse can only changes the slider location.

**refresh** (highest in the image) enable/disable **auto-recall** snapshots, see below for more details.

### transitions

Transitions are a linear interpolation between two snapshots, with one time duration.

You can start a transition only from the menù **recall** item.

Is important to understand that the transition occurs from the **current parameters positions**, toward the selected snapshot.

**actual GUI widgets positions >>> (toward) selected snapshot (time duration)**

You can change the **transition-time** (2.00 sec in above image). The transitions can occur simultaneously on the streams and/or on the main. If **auto-recall** (see above) is enabled, selecting from the menu a snapshot slot to start the transition.

## snapshots manage

**store new** snapshot using the next empty preset slot;

**store snap number**, enter a number or a list separated by space, to store;

**clear** current selected snapshot;

**clear all snapshot**, a message box will ask you to confirm;

**renumber** sort all snapshots stored into consecutive, beginning with 1;

**recall**<sup>1</sup> start a transition toward current snap number;

**pause/resume** current transition (if occur);

**stop transition** cancel current transition (if occur);

**reinit parameters**, resets subscribed **parameters** at default value, only for the current streams or main<sup>2</sup>;

**save to disk** current snapshots-bank;

**load from disk** current snapshots- bank;

**client window**, open clients list to subscribe, unsubscribe, setting interpolation type and more;

**storage window** open storage window displays any stored presets;

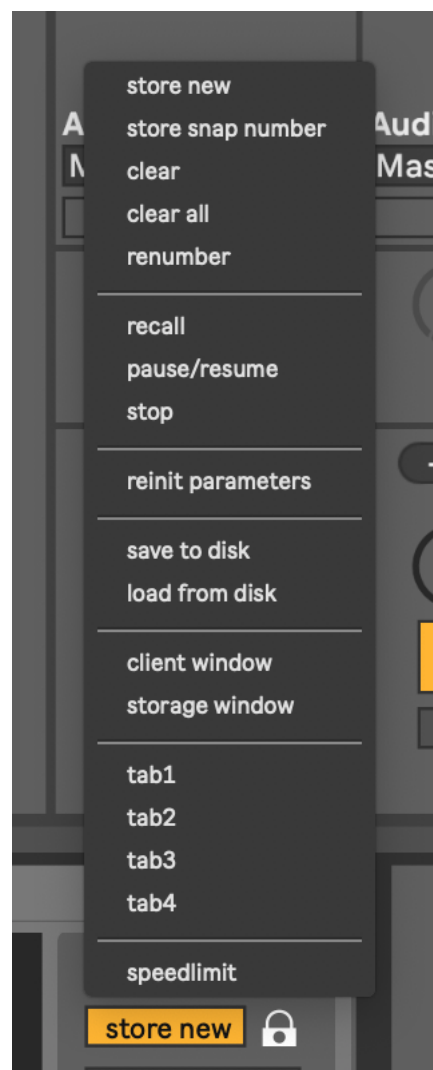
**open HV\_PADS** (Hyper Vectorial Pads, see [HV\\_Pad](#));

**open sequencer** improviser unit (see [Snapshots Sequencer](#));

**tab1, tab2, tab3 and tab4** open transition curve window, see [transition curve](#) below;

**speedlimit** set update time limit for transitions and micropad nodes interpolations.

N.B. all the presets are saved in the Ableton Live project or device, however, you can manage individually save/load snap-bank, for example useful for exchanging presets between the streams.

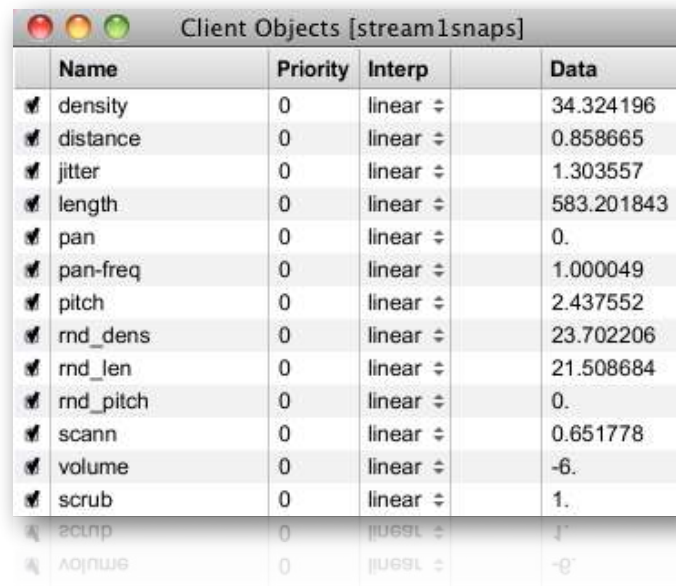


<sup>1</sup> you can enable or disable auto-recall, if enabled selecting a snap from the menu, it will start the transition, if disabled it will be **only selected**.

<sup>2</sup> gsnap will restore all Pulsaret 2 subscribed parameters at default values (granular parameters and setup parameters).

## subscribe/un-subscribe transition clients

The two windows **client** and **storage**, are both non-interactive:



	Name	Priority	Interp	Data
✓	density	0	linear ↕	34.324196
✓	distance	0	linear ↕	0.858665
✓	jitter	0	linear ↕	1.303557
✓	length	0	linear ↕	583.201843
✓	pan	0	linear ↕	0.
✓	pan-freq	0	linear ↕	1.000049
✓	pitch	0	linear ↕	2.437552
✓	rnd_dens	0	linear ↕	23.702206
✓	rnd_len	0	linear ↕	21.508684
✓	rnd_pitch	0	linear ↕	0.
✓	scann	0	linear ↕	0.651778
✓	volume	0	linear ↕	-6.
✓	scrub	0	linear ↕	1.

**client window** (accessible from the **client window** menu item, see above) shows the current **subscriptions widgets list** (ever header name), **priority**, **interpolation** and **data** belonging to subscribed widgets.

A few types of interpolation can be changed:

**off**: no interpolation;

**linear**: Linear interpolation. Presets recalled will be interpolated using a standard linear algorithm.

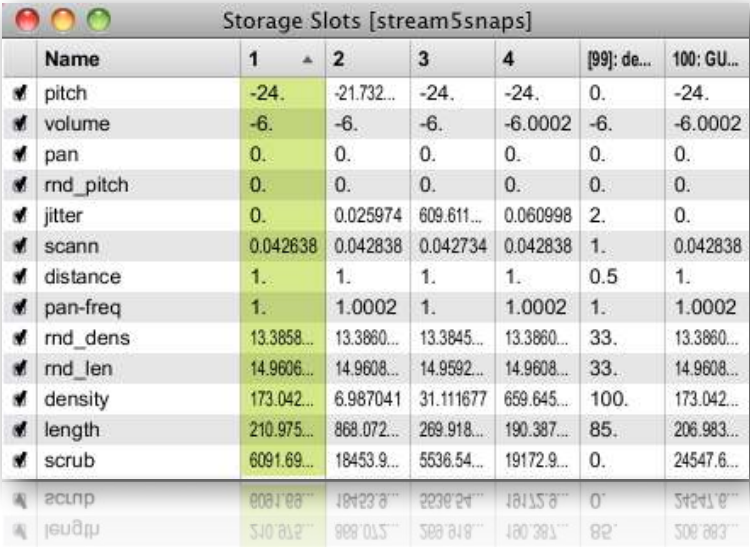
**threshold**: Threshold. Takes optional argument (float), which sets the threshold. Presets recalled will recall data from the first preset specified when the fade amount is below the threshold, and will recall data from the second preset specified when the fade amount is greater than or equal to the threshold. e.g. threshold: '**fade**' < thresh = value a; **fade** >= thresh = value b;

**inverted threshold**: Inverse threshold. Takes optional argument (float), which sets the threshold. Presets recalled will recall data from the first preset specified when the fade amount is greater than or equal to the threshold, and will recall data from the second preset specified when the fade amount is less than the threshold. e.g. inverted thresh: '**fade**' < thresh = value b; **fade** >= thresh = value a;

**exponential curve**: Power curve. Takes an additional argument (float), which sets the exponent to which the fade amount will be raised. Presets recalled will recall data between the two specified presets, along the curve described. Power curves can be used to create faster or slower "attacks" and "decays" for the fade envelope;

**table**: Table-specified curve. Takes optional additional argument (see [transition curve](#) below), which specifies the name of a table to use for curve lookup (**tab1**, **tab2**, **tab3** or **tab4**). Presets recalled will recall data between the two specified presets, along the curve described in the table.

Tables are assumed to contain values between 0 and 100, representing the new fade amount \* 100. If the lookup fade amount does not fall exactly onto a table-specified value, linear interpolation is used to determine the new fade amount. In Pulsaret 2 you have four draw functions to draw your curve, see [transition curve](#) below.



Name	1	2	3	4	[99]: de...	100: GU...
pitch	-24.	-21.732...	-24.	-24.	0.	-24.
volume	-6.	-6.	-6.	-6.0002	-6.	-6.0002
pan	0.	0.	0.	0.	0.	0.
rnd_pitch	0.	0.	0.	0.	0.	0.
jitter	0.	0.025974	609.611...	0.060998	2.	0.
scann	0.042638	0.042838	0.042734	0.042838	1.	0.042838
distance	1.	1.	1.	1.	0.5	1.
pan-freq	1.	1.0002	1.	1.0002	1.	1.0002
rnd_dens	13.3858...	13.3860...	13.3845...	13.3860...	33.	13.3860...
rnd_len	14.9606...	14.9608...	14.9592...	14.9608...	33.	14.9608...
density	173.042...	6.987041	31.111677	659.645...	100.	173.042...
length	210.975...	868.072...	269.918...	190.387...	85.	206.983...
scrub	6091.69...	18453.9...	5536.54...	19172.9...	0.	24547.6...
sculp	008'08"	10423'8"	2230'24"	18115'8"	0'	54241'8"
jeu0j0	510'812"	909'015"	509'818"	180'391"	82'	508'883"

The **storage window** displays any stored presets. The active (recalled) preset is displayed in highlight green. If any client value is changed, are displayed in italics. Eventually, both of these windows will be configurable and editable, so that they can provide display and editing control for clients and storage sets.

**N.B.** When you open storage window, a message box will be displayed:



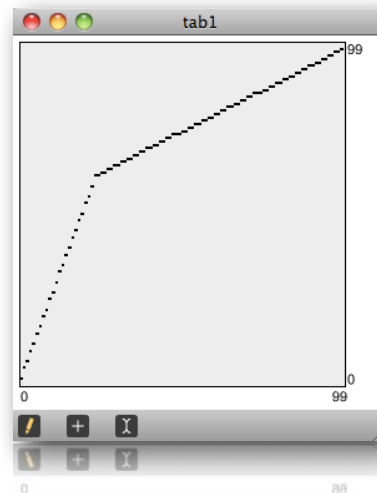
this is to avoid an overhead of CPU when displaying storage window.

## transition curve

All transition by defaults are linearly interpolation. You can superimpose your interpolation curve at one or more subscribed widgets. Four draw functions are available, from toolbar or menù bar. Open one of them

and draw your curve. This feature is **saved** on Pulsaret 2 **device**.

Draw a shape on the function window



In order to maps one or more subscribed clients with one of four functions draw, you need open **clients window** and select **table** from pop-up menù (over **interp** header). Now you must digit the table name, that can be: **tab1**, **tab2**, **tab3** or **tab4**.

	Name	Priority	Interp		Data
<input checked="" type="checkbox"/>	backward	0	threshold	⇅ 0.50	0.
<input checked="" type="checkbox"/>	density	0	table	⇅ tab1	1.
<input checked="" type="checkbox"/>	distance	0	exponential c...	⇅ 2.00	0.590551
<input checked="" type="checkbox"/>	jitter	0	exponential c...	⇅ 0.50	648.818909
<input checked="" type="checkbox"/>	length	0	inverted thres...	⇅ 0.00	63.960632
<input type="checkbox"/>	pan	0	linear	⇅	192.
<input type="checkbox"/>	pitch	0	linear	⇅	-9.543307
<input checked="" type="checkbox"/>	rnd_dens	0	table	⇅ tab1	33.
<input checked="" type="checkbox"/>	rnd_len	0	table	⇅ tab1	15.748032
<input type="checkbox"/>	rnd_pitch	0	linear	⇅	0.
<input type="checkbox"/>	scann	0	linear	⇅	1.
<input checked="" type="checkbox"/>	scrub	0	table	⇅ tab2	141.732285
<input checked="" type="checkbox"/>	volume	0	linear	⇅	-27.26

in the above example, we employ:

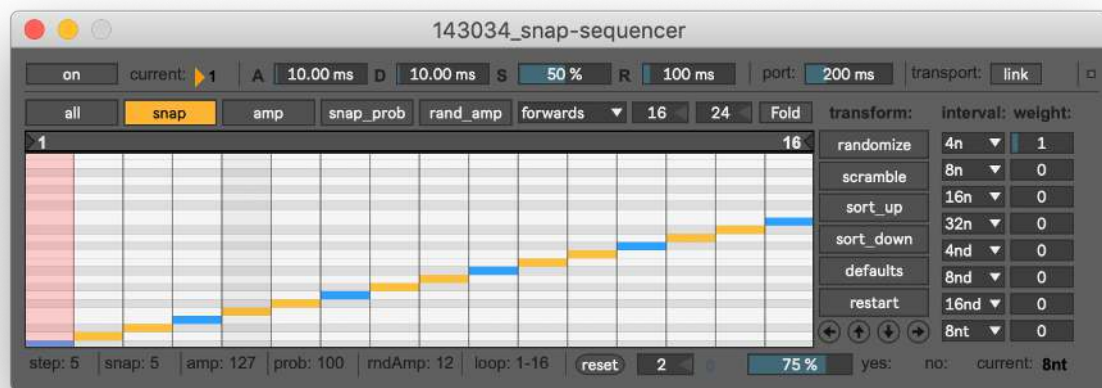
**tab1** like transition curve for **density**, **rnd\_density**, **rnd\_length**;

**tab2** like transition curve for **scrubb**;

**pan**, **pitch**, **rnd\_pitch** and **scann** are unsubscribed from the snapshots transitions system.

See **client window** above for more explanations.

## Snapshots sequencer improviser unit



Snapshots sequencer rhythm improviser unit i.e. the extended name, the aim of this device is to control the snapshot sequence<sup>88</sup> by improvising with different rhythmic pulses. The sequencer shown on vertical the snapshots numbers while in horizontal steps sequence.

### beats cycle

A metronome is synced to Ableton Live global transport time and outputs a trigger on each beat. The beats are counted in what we're defining as a cycle of  $\langle n \rangle$  beats (Cycle sub-patcher). Each time the cycle restarts, we're choosing whether or not we want to change the current time interval. This is set by using a probability factor (ChangeProb sub-patcher). If the program decides to change the time interval, a trig is sent.

### time intervals

Since eight time intervals can be selected. The choice is performed with a random function, according to a weight probability for each possible interval - an interval with a weight of 2 will have twice as many chances to be selected by the random procedure that an interval that has a weight of 1.

### step sequencer

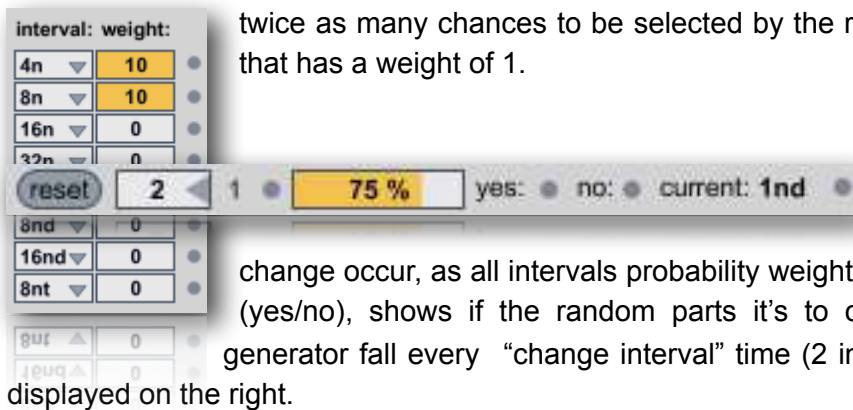
When a given time interval is chosen, it is used as a step rate for driving the step-sequencer. The step-sequencer is synced to Ableton Live global transport time, so the steps are played in accordance with transport tempo. The snap-sequencer has many editing modes that allow for modifying the snapshots content (snap, amp, etc..).

### output

The snap-sequencer outputs a list of data each time a step is triggered. Each data type (snap, amp, amp\_rand...) is interpreted in order to recall a snapshot, with additional possibilities such as: random amplitude of envelope and snap recall probability.

Up to eight time intervals can be selected. The choice is performed with a random function, according to a weight probability for each possible interval - an interval with a weight of 2 will have





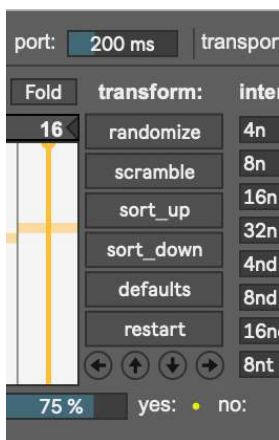
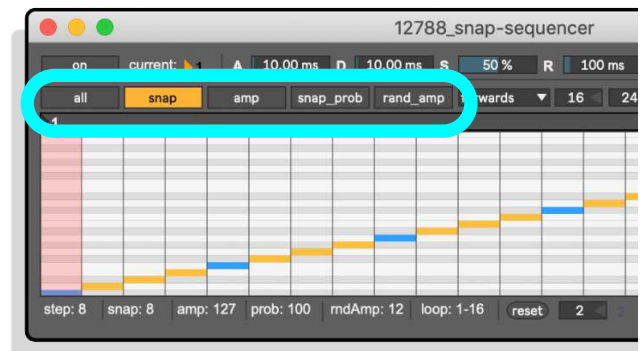
twice as many chances to be selected by the random procedure that an interval that has a weight of 1.

You can set "general changes probability" (75 % in the example) , 0 % no interval

change occur, as all intervals probability weight to 0. Change and no change led (yes/no), shows if the random parts it's to change or not. Random number generator fall every "change interval" time (2 in the example), beat sequence is displayed on the right.

The most left **reset** button, resets sequencer to first step, in permutation mode reinit sequence, no action in random mode.

Edit mode selection is done using this set of tabs, view all displays all of the available data (and allows you to edit the snapshots). **Snap** mode displays and allows editing of the snap only. **Amp** mode allows you to edit the volume of snapshot envelope. **Duration** mode is used to change the step size of each value, while **Probability** mode allows you to manipulate the likelihood of a snapshot.



Sequencer play mode menu read steps in the explained way. You can chose the number of the step (**columns**) and snapshots range (**row**), then **fold/unfold** lets you chose to display all possible snapshots, or only a specific set of snapshots.

Some kinds of transformations are available their names are auto-explicative.

Through the four arrow keys at the bottom, up and down (snap number) or left and right (in time) you can shift the entire sequence.

Snapshots Sequencer is an **ADSR** (Attack Decay Sustain Release) every step envelops the streams (or main) amplitude **audio signal**. You can define a **Attack Decay** time and **Sustain** percentage, at last **Release** time. When Sequencer is turned off, audio signals (streams or main) pass through normally.





## Note and Tick Value

Here is a listing of the note and tick values associated with common note durations. Note value abbreviations that can be used in Pulsaret 2 to specify time are in bold.

**128n** - One-hundred-twenty-eighth note - 15 ticks

**64n** - Sixty-fourth note - 30 ticks

**64nd** - Dotted sixty-fourth note - 45 ticks

**32nt** - thirty-second-note triplet - 40 ticks

**32n** - thirty-second note - 60 ticks

**32nd** - Dotted thirty-second note - 90 ticks

**16nt** - Sixteenth note triplet - 80 ticks

**16n** - Sixteenth note - 120 ticks

**16nd** - Dotted sixteenth note - 180 ticks

**8nt** - Eighth note triplet - 160 ticks

**8n** - Eighth note - 240 ticks

**8nd** - Dotted eighth note - 360 ticks

**4nt** - Quarter note triplet - 320 ticks

**4n** - Quarter note - 480 ticks

**4nd** - Dotted quarter note - 720 ticks

**2nt** - Half note triplet - 640 ticks

**2n** - Half note - 960 ticks

**2nd** - Dotted half note - 1440 ticks

**1nt** - Whole note triplet - 1280 ticks

**1n** - Whole note - 1920 ticks

**1/128**

**1/64**

**1/64D**

**1/32T**

**1/32**

**1/32D**

**1/16T**

**1/16**

**1/16D**

**1/8T**

**1/8**

**1/8D**

**1/4T**

✓ **1/4**

**1/4D**

**1/2T**

**1/2**

**1/2D**

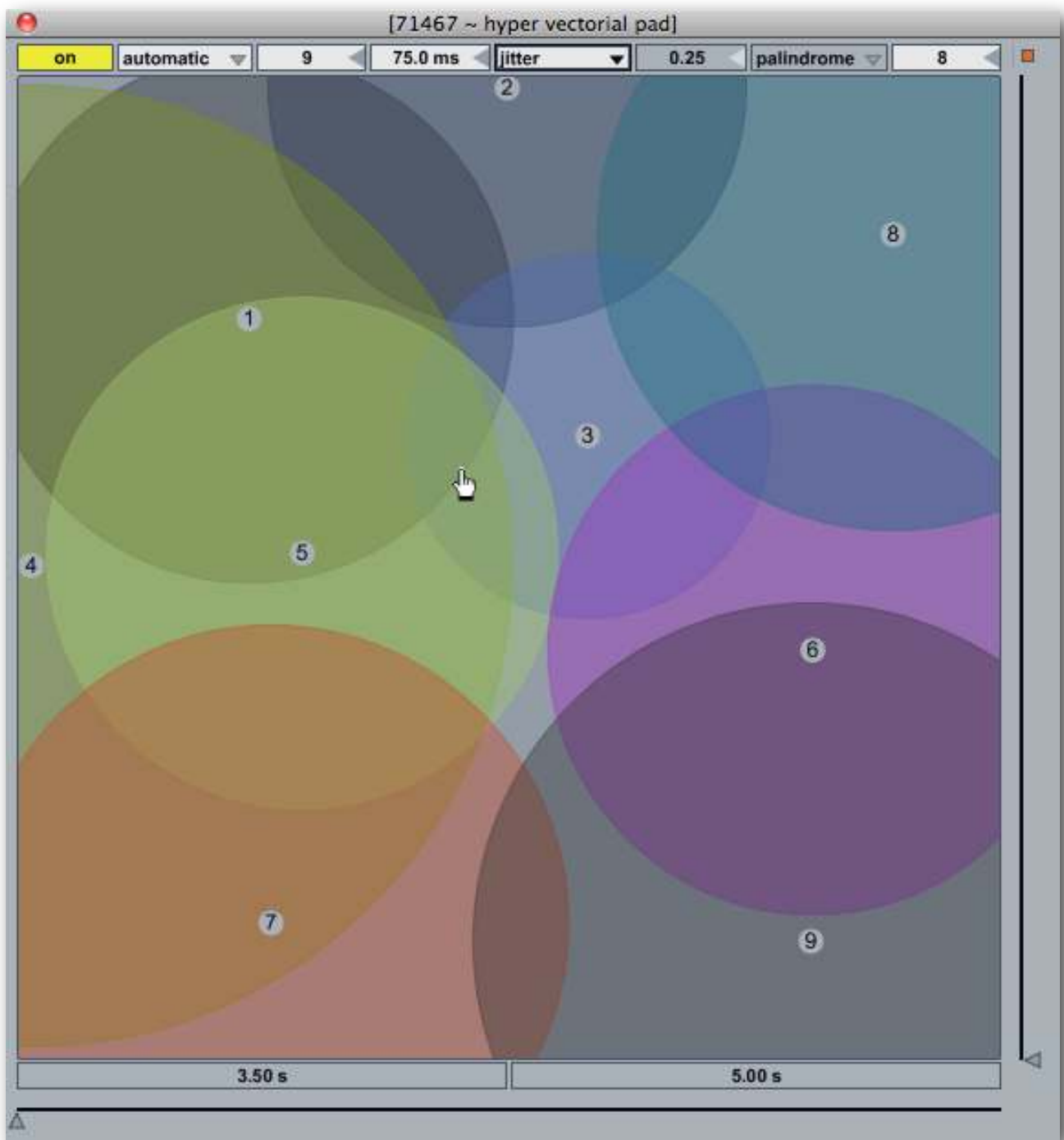
**1/1T**

**1/1**

In Pulsaret 2 you can find also this formula of value.

## Hyper Vectorial Pad<sup>1</sup>

general explanation

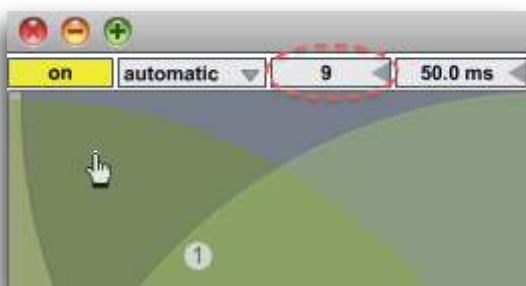


**HV\_pad** (i.e. Hyper Vector Pad) is 2D parameter interpolation user interface. Displays nodes in a 2-dimensional space, and calculates the distance from a point. The distance factor determines the weight between snapshots interpolation. Like for snapshots-module, we have ten **HV\_Pad** grouped in a unique window.

You can open HV\_Pad window from the tool bar button.

<sup>1</sup> See VMCI Virtual Midi Control Interface (<http://www.csounds.com/maldonado/>), thanks Gabriel Maldonado.

**HV\_Pad** extends the functions of micro-pad, interpolating until 24 snapshots together, thus you can modify hundreds of parameters with a single mouse motion, according to a structure configured by the user. This control method is called *Hyper Vectorial Synthesis Control*. Normally the user interacts with these areas with the mouse or through self-scanning exploration, however a remote MIDI control is possible, by assigning the X Y pointers (see [Ableton Live user manual](#)).



In the image below, from number box (red circle) you can set the number of nodes, you can set a specific configuration (system nodes) by **clicking** and **dragging** center of node to move it or **Option Key (alt)** to change node size.

**N.B.** The nodes are numbered progressively, therefore each node (on the pad) must be corresponds by a stored preset snapshot. If you employ an empty snapshot, the interpolation between them will not work properly. Also you cannot disabling completely the nodes (node area = 0), to avoid fall in the same problem.

pad mouse behavior

The pop-up menu allow you to chose mouse behavior:

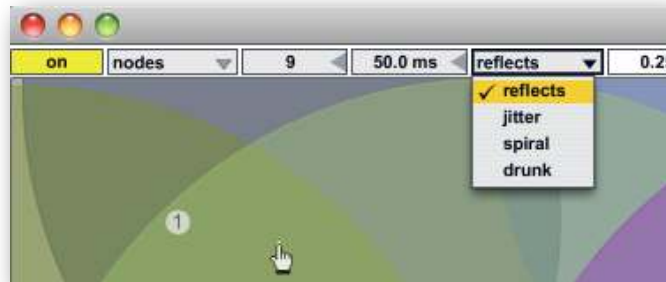


**Automatic:** allows the mouse to change the nodes position, size and slider (pad scrub);

**Nodes:** the mouse can only edit the nodes position and size, **alt + mouse scroll** to change node **size**; **click and drags** on the node number, to **move** it;

**Slider;** the mouse can only changes the slider location (pad scrub).

You can automatize HV\_pads scrub (slider) choosing a scanning mode:



Scanning method:

**reflects** causes a reflection when the cursor reaches the edge;

**jitter** outputting random numbers within a moving specified range around current scrub mouse position;

**drunk** will perform a "drunk" walk, creating unpredictable paths;

**spiral** centripetal, centrifugal or palindrome, with phase deformation parameter;



The **jitter step size** (number box 8) it will be showed only when jitter is selected. This parameter sets the random amount of jittering around of the mouse position during scrub pad.

Automatic exploration in **reflects**, **jitter** and **drunk**, you can interact with the pad. The values of mouse and direction, will be the new coordinates of scanning. Instead in **spiral** you cannot interact with pad

### X/Y time

The X Y indexes scanning ratio, depends by the "frequency" X/Y axes. These are controlled by the two sliders showing value in seconds (i.e. time = 1/frequency).



You can set scanning time for the two axis (2D XY). For example 3.50 sec. for the **X** axe it means that horizontal slider scrub it take 3.50 sec to pass from left edge to right edge of the pad, **Y** = 5.00 sec.

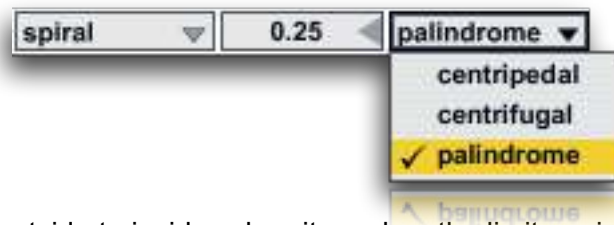
**X Y** parameters, they assumes different means according to selected scanning mode:

in **reflects** is expressed in **time**, is the time it takes to move from **X** = left/right and **Y** = up/down;

in **drunk** is expressed in **time**, is the deviation amount  $\text{rnd}(\mathbf{X})$  and  $\text{rnd}(\mathbf{Y})$ ;

in **spiral**: **X** is the time for each sub-spiral, **Y** is the total duration until reach all pad area;

The **spiral** mode, will enable two related parameters:



**centripetal** moves from outside to inside, when it reaches the limit again, repeat;

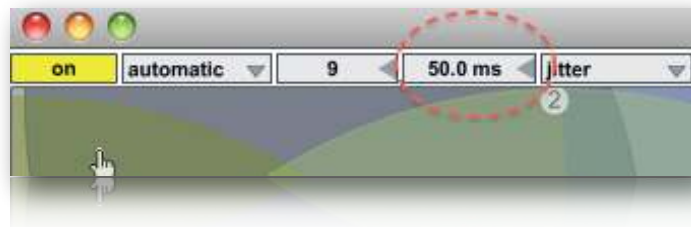
**centrifugal** does the opposite;

**palindrome** is a combination of both;

The **spiral phase deformation** (number box 0.25) 0: no spiral; 0.25 circular; > 0.25 crushed; > 0.5 clockwise rotation

miscellaneous

Auto-exploration is possible only when the Ableton Live transport is enabled (see [Ableton Live user manual](#)).



**on/off** toggle auto-exploration according to selected mode.

The red circled number box, sets the interpolation update time in milliseconds, this is a speed limit to avoid an overhead of cpu with complex interpolations, be careful with short times!

## Matrix

### general explanation

Pulsaret 2 matrix linkage is employed to create a dependency between parameters. Moreover you can rescale, reverse controlled parameter range and/or move it by LFO (Low Frequency Oscillator). This technique is named: Grainlet Synthesis.

### parameter linkage

On the top of the Matrix (horizontal) we have sending parameters, They will be linked with right ones (vertical). moving a send parameter, you will move controlled (linked) parameter.

Every controlled parameter can be mapped on a own range and/or reversed, it means it will be moved on the contrary regarding the motion of the parameter that controls it.

### grainlet/pulsaret mode

In grainlet mode you can interconnects all granular parameters freely, you can also toggle in Pulsaret mode.

A Pulsaret consists in a brief burst of energy. The grain duration contains only one cycle, varying the grain length you change “duty cycle” of pulsaret, while density remain fix and independent.

You can link the two parameters in two ways: **length -> cps** or **cps -> length**.

$$\text{cps} = (1/\text{length}) * \text{cycleN}$$

$$\text{length} = (1/\text{cps}) * \text{cycleN}$$

e.g. 100 Hz = 1/100 = 0.01 ossia 10 ms.

Optionally you can specify the number (numberbox) of cycles for the pulsaret.

e.g.

cycleN = 2

cps = 100 Hz

length = 1/100 = 0.01 \* 2 = **0.02 Sec** (ossia 20 Ms)

for the cps/length linkage:

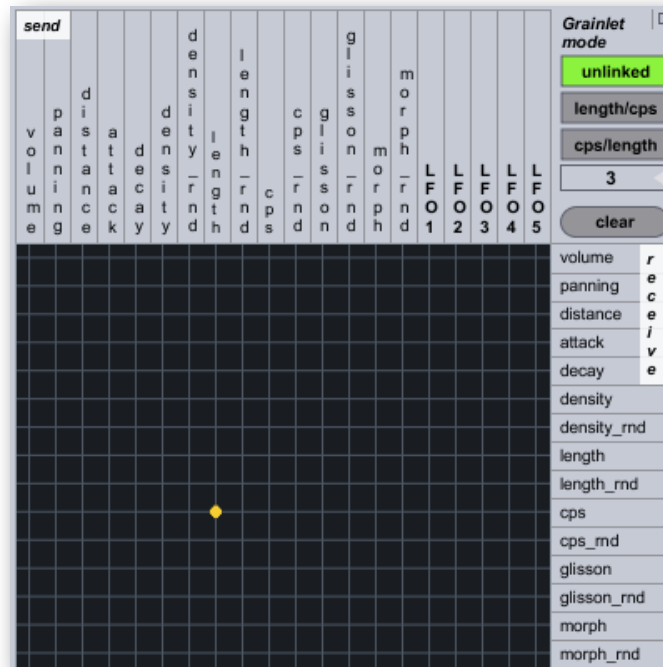
cycleN = 3



length = 45 Ms = 0.045 sec  
 cps =  $1/0.045 * 3 = 66.66$  Hz

parameters rescale

In the below example, **length** and **cps** are linked:



all length range is employ to move cps parameter.

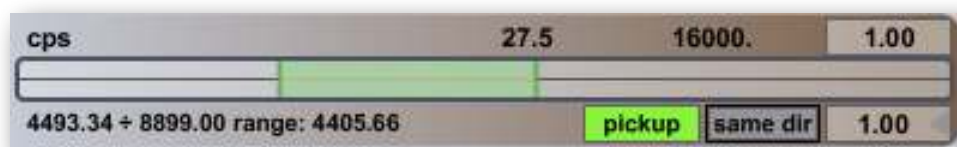
If can maps entire length range on a little part of cps parameters:

You can also enter your values on the numbox (27.5 - 16.000), for changing global range and change exponential curve mapping (numbox):

min value for the selection (click and digit a new value, then press enter)

max value for the selection (click and digit a new value, then press enter)

exponential base value (default 1. = linear)



If pickup is enable, when controlled parameter is out of range, it will no moved until it will be bring inside of the range. it wock in analogous way to MIDI/OSC pickup, see MIDI/OSC for more details.

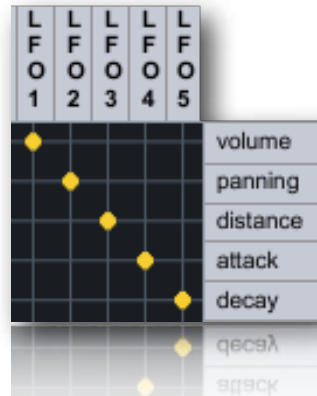
same dir/inverse, make an inverse maps of the controlled parameter.

the last parameter, controlling general rescale output, 1. it means no rescale.



## LFO modulation

Low Frequency Oscillator (LFO) is useful to automate one or more parameters, they will be constantly moved around one middle value depending by shape, frequency and deep of modulation of LFO.



in the above example, LFO1 (of the 5 availables) modulate volume parameter, LFO2 panning etc.... After you have assign matrix linkage, you need enable LFO clicking on the on button.



The LFO params are (from top to bottom):  
on/off LFO (Low Frequency Modulation)

toggle between beat synced and hz; in sync mode, the beat division switches specify the delay time in 16th notes; in time mode, the pan rotation is expressed in hz.

panning rotate frequency in Hz

16 note: each numbered switch represents the time in 16th notes, note that these switches are only active if the time is set to sync; you can change global time by the transport unit.

lfo shape, you can select one of the windowing shapes availables

modulation deep in %

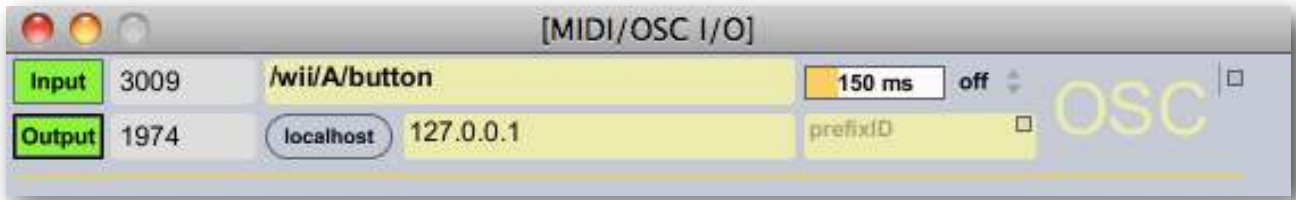
uni/bipolar range, unipolar = 0 to 1 modulation range (with 100% deep); bipolar= -1 +1

change the speedlimiting time parameter update.

## OSC I/O

### header

Open **OSC I/O** from [quick access](#) buttons, you can configure the OSC (Open Sound Control) protocol, through the header,



### OSC:<sup>1</sup>

Dispatch messages through an OpenSound Control address hierarchy with pattern matching. Values are serialized and sent over the network as OSC compatible UDP;

**Input port** (3009) set input port to send messages to at host;

**Output port** (1974) set output port to send messages;

Green left buttons, enable/disable respective OSC I/O.

**Very important!: default values for I/O ports: 30-09-1974 are the author birthday date.**

We need also configure **host** (127.0.0.1) output , entering an **IP** address or **hostname**, clicking on **localhost** to assign default ip value (i.e. local host 127.0.0.1).

**OSC monitor** show OSC I/O data, you can select with mouse the text string address (OSC path) and paste it on widget controller (see below).

You can enter a prefix ID (can be numbers and/or text) for the the OSC output address. When an incoming OSC message is synced on a widget parameter, this read and write values employing the same path address. Some time is useful append an id prefix to differentiate path in order (for instance) to send message to another device .

**N.B. OSCulator**<sup>2</sup> can be employed like MIDI/OSC monitor beyond supplying advanced features.

N.B. All settings will be saved on the project (see [overview](#) for more explanations)

sync

Sync system allow you synchronize software parameters with hardware devices. Thus Pulsaret 2 send out through OSC ports parameters data. Default **sync** is disabled for all widgets, then no OSC data are sent. You can select enable **sync** out for each widget controller or superimpose a common feature at all widgets (see below).

<sup>1</sup> OSC is a CNMAT Max objects, can be found at: <http://www.cnmat.berkeley.edu/MAX>

<sup>2</sup> OSCulator is a software that can be used with many different hardware devices and software. For example, with OSCulator, your Nintendo Wiimote can talk to major MIDI sequencers or your favorite console emulator or even the Kyma sound design workstation. And your [iPad](#), [iPhone](#), [iPod Touch](#) or [Lemur](#) can do just the same... with great ease of use. <http://www.osculator.net>

For example, you can enable sync for all widgets controllers, by selecting the item from header menù.

**Sync** have five outputs mode:

OSC out mode for all widgets:



**off** no OSC outs data;

**on** send OSC data continuously. N.B. this overwrite all settings of the widgets.

Every time you move a Pulsaret 2 parameter on the GUI or when a transition/sequencer/HV\_pad is at work, **sync** values are sent.

To avoid an overhead of outputs data, through the **number box** (150 ms) you can change MIDI/OSC output update rate in milliseconds. For instance during the snapshots transitions, you should limiting sync data in order to preserve a normal cpu overhead: **OSC out update rate**, expressed in milliseconds set the speed limit OSC output data.

## widget mapping

We have two widgets mapping kinds:

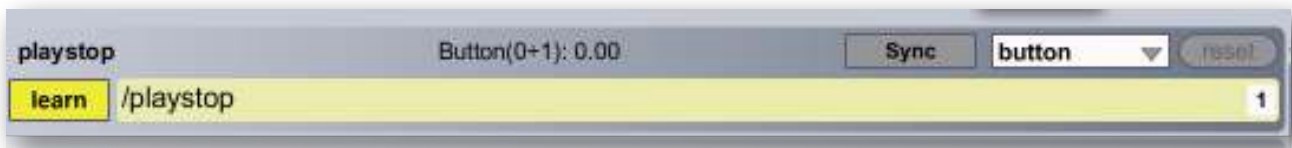
**Cc\_raw**

**Cc\_scale**

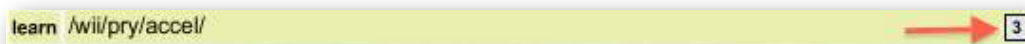
**Cc\_raw** is addressed to mapping Pulsaret 2 buttons/toggle parameters, Cc\_raw read/write OSC returning integer values (between 0 and 127), **Learn** function waits to feel an OSC address, when it happens the the control is set up on the OSC path address.

Now you can controlling Pulsaret 2 parameter via OSC.

Of course you can manually change OSC address.



Toggle to **learn** and send an OSC data in order to assign incoming OSC address. In the example the OSC */play/stop* address are assigned to **play/stop** Pulsaret 2 parameter, if **sync** is enabled Pulsaret 2 also is able to send OSC data. It is possible change OSC path value entering a new one.



An OSC message can be a list, if you want read an element of the list, you can select an OSC input item from the list, (default first element). In the example, Nintendo WII device send the accelerometer values like a three list items, selecting 3 to read only third incoming.

**I/O sync** pop-up menù, set output mode for current widget, see [sync](#) for better understand.

The most right menù (toggle), set the **controller kind**:

**toggle** switch from 1 and 0 every time MIDI/OSC I/O == 0;

**button** send 1 every time MIDI/OSC input != 0;

**incr** and **decr** increment or decrement (regardless of direction) by one step, every time MIDI/OSC input == 0. when values 127, or 0 are reached, the next value is wrapped;

**incr/decr** it counts upward until it reaches 127, then counts down until it reaches 0, then up, then down, and so on;

**controller** mode read raw data from MIDI/OSC input, raw data MIDI between 0 and 127.

**Cc\_scale** extends Cc\_raw functions, you can choose a certain range on which rescale the parameter, for example control **length**, which is associated, covers a range -70 ÷ +6 (76 units).



In the example, incoming OSC data is scaled on a part of the entire parameter range by the orange horizontal selection. You can also change min/max values for the parameter (box numbers -70. and 6.), current range will be reported.

**OSC assumes default values between 0 and 1.** You can also change OSC input range. For example we control **gain1** through **/wii/1/accel/roll** OSC data incoming from OSCulator and Nintendo Wii, we rescale the range for controlling only a middle part of parameter.

**Inv**, invert the I/O range for the widget.

Most right menù, **I/O sync** (off) set the output mode for current widget, see [sync](#) for better understand and

numbox (1.00) on top right set the exponential base value (default 1. = linear). The number is converted according to the following expression:  $y = b e^{-a \log c} \exp \log c$ .

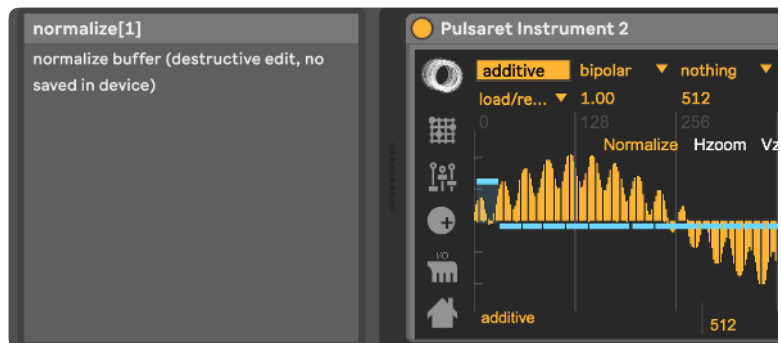
## Overview

### info window

To avoid annoying hint messages when you are working on the GUI, Pulsaret 2 sends all hints in the Ableton Live **Info window**. In the clue window you can read some informations about Pulsaret 2 GUI widget, when the mouse is over it. See [Ableton Live user manual](#).

### float window

Have you noticed all Pulsaret 2 windows contains in a corner (generally top right) a tiny orange button? toggle it to set float/no float window. When toggle float, current window will be always in front. This feature is saved on project like general setup.



### acknowledgments

I would like to thanks Eugenio Giordani and Nicola Casetta, a special thanks to: Renato Alberti, Felix Petrescu and Pasquale Ascione.

Isotonik Studios would like to thank Chaos Culture & Tom Cosm



CNMAT ©  
 OpenSoundControl (OSC-route)  
<http://www.cnmat.berkeley.edu/MAX>  
 All rights reserved

Bibliography:  
 Curtis Roads "Microsound"  
 The MIT Press  
 Cambridge, Massachussets

The Author:  
 Alessandro Petrolati  
 apeSoft  
 All rights reserved © 2010-2019  
[www.apesoft.it](http://www.apesoft.it)