GranuLab

Real-Time Granular Synthesizer

by rasmus ekman

---

contents

#$ [K]What is Granular Synthesis? >>> granny overview

w o t s i t ?

Granular synthesis is the technology of creating complex sound by playing back many short and relatively simple sound fragments with varying parameters.

f o o d f o r c h o p

Just using a sine waveform as base material, and controlling length, pitch and density of grains, several types of sound can be generated.

It is however more exciting to use a concrete sound sample. Good raw material includes speech, instrument or ambient sounds, or indeed synthetic sounds produced with Csound, **GranuLab** or some other program. In this case, granulation means cutting up the sound into short segments and controlling how these are played back.

p r o s a n d c o n s

The advantage of granulation is that you have control over where each grain begins reading in the soundfile. This means that you can do time-stretching and repitching of sounds in real time, without using computationally intensive FFT analysis/resynthesis.

From a formal viewpoint granulation is a bit of a fudge, and sometimes it is impossible to create a clean effect which has neither echoic nor a metallic or dredgy sound.

But in any case which is just the point any sound can be scrambled beyond recognition with very little effort.

$ What is Granular Synthesis?

K [K] What is granular synthesis?

$ K KGranuLab Overview >>> global params >>> grain params

b a s i c o p e r a t i o n

On starting **GranuLab** you'll just see a mass of sliders and buttons. The sliders are grouped in thematic boxes. Each long slider is the main controller of a parameter, beside it are one or two half-length sliders which do random or amplitude-modulation variation of the longer slider's value.

You can load a 16-bit WAV mono or stereo soundfile into **GranuLab** for processing. With stereo files you can decide which channel(s) to use for input to the grain stream (under the Command menu).

There are five basic parameters in **GranuLab**:

Soundfile section - which part of the current sound file is used.

Soundfile playback rate - Controls the rate by which the starting point of each succeeding grain is incremented. This is the key to time-stretching and time-compression.

Grain rate and grain length - These parameters cooperate to determine **grain density**. Low density (less than 1-1.5) means chopped sound. High density allows many echo and filtering effects.

Grain pitch - The internal pitch of each grain. The pitch may be changed during playing time of the grain, by a simple glissando parameter.

Grain envelope - attack and decay time for each grain, in percent of grain length.

Besides the grain parameters, there is also a section for amplitude and panning of the output of each grain stream.

Each basic grain parameter may be modulated, either by random offset from the set value, or by the amplitude of the point in the sound file where the grain is taken from (see amplitude modulation).

The setting of all sliders may be stored as a patch, which may be recalled later.

GranuLab may also be controlled by external MIDI signals.

GranuLab 8 can have up to 8 streams simultaneously.

K GranuLab overview

$ K KGlobal Parameters >>> grain params

using the sliders

All slider changes are subject to portamento (see below), ie gradual move from the current position to the next.

Slider values are used when the next grain is created. The settings for grains currently playing are not changed afterwards. If you set grain length to very long time, you'll just have to live with them.

Commands for all sliders:

SHIFT+click on a slider to restore the default value. Restore all sliders at once by clicking the **Default** patch.

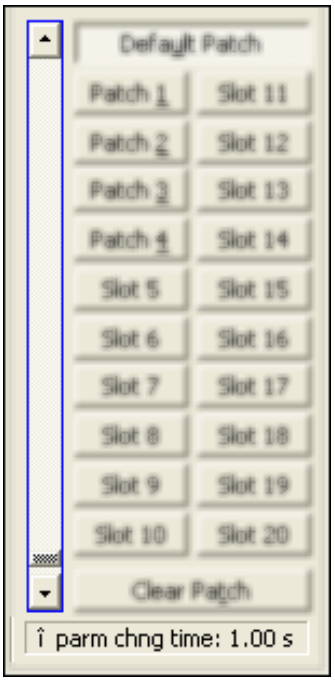
CTRL+drag a slider button to bypass portamento, and set the new value immediately.

Slider output is always shown in the status bar, together with relevant information.

#Kamplitude modulation

Most parameters may be controlled by the amplitude of the input sound. The amplitude envelope of the sound is taken as the file is read in to **GranuLab** (the left channel of stereo files). The amplitude value used for each grain is synchronised with the soundfile at the point where the grain will end its fade-in phase. Some of the amp modulations create quite unique connections in the sound.

the #Kportamento slider



This slider in the Patch banks box (far right) controls the time which it will take for any change of slider values (or patch) to take full effect.

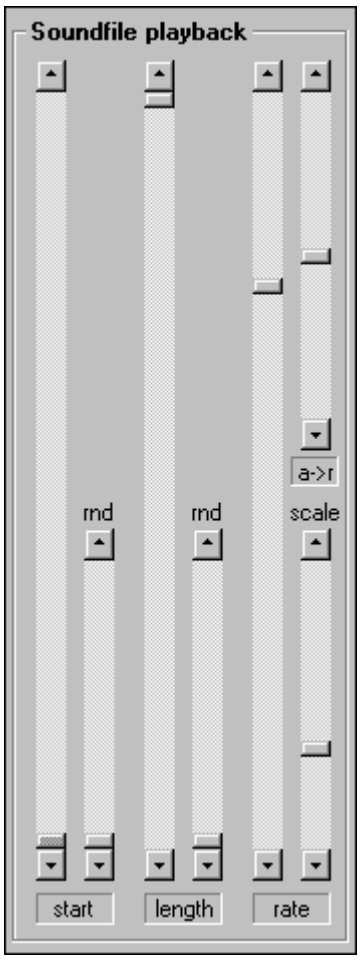
If the time is set to eg 30 seconds, and you click a patch button (or move a slider), the grain generation values will slowly transform from the current values towards the new patch (or slider) value. By changing the portamento time and clicking again on a patch button or slider, you can speed up or slow down the rate of change at any time.

K GranuLab overview

portamento

K K Portamento slider

$ K K Soundfile Playback >>> grain density



(The parameters in this section are not really applicable to the built-in sine wave, so use mainly with loaded sound files.)

Sliders **start** and **length** control which part of the sound is used. The selected part of the sound is looped.

The **rate** group of sliders are the key to time-stretching.

scale: This slider controls the range of the main rate slider. At the default setting, the soundfile playback rate may be varied between +/-2 times normal speed. If the scale is increased, the playback rate range may be up to +/-22 times normal playback.

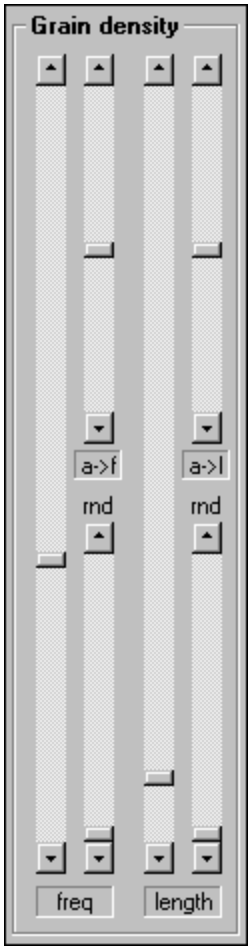
In the maximum range, the precision of the rate setting deteriorates, so it seemed useful to have two controls for playback rate.

a->r: The amplitude of the input sound can be used to control playback rate. This can be used to good effect eg with speech, making vowels relatively longer or shorter.

$ [$] Soundfile Playback

K [K] Time-stretching

$ K # K # K Grain Density >>> grain pitch



The frequency at which new grains are generated (**freq**), multiplied by the length of each grain (**length**), determines grain density. The total density will be displayed in the status bar, rightmost pane. In **GranuLab 8**, the density of all streams follows the density of the current stream.

{bml densdisp.bmp}

Since each grain is faded up and out, density 1.5 - 2 is usually the minimum for smooth non-grainy playback. With thicker density, echo and chorus effects may be created.

A high grain frequency may create a pitch which interferes with the pitch of the sound (for good and for bad).

Warning! You need to be careful with grain **freq** and **length**: Your computer can only manage a certain grain density, and when this is exceeded, the sound will begin chopping. This will at the same time weigh down the processor so that the **GranuLab** program interface (sliders and all) gets no time from Windows for normal operation. This will make it seem to get completely stuck. **In this situation, hit the Q key on your keyboard to stop sound output.**

See also: I/O options.

---

grain_density

$ $ Grain Density

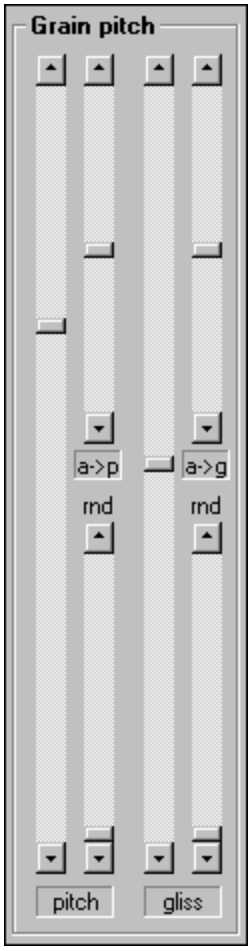
grain_length

K K Grain length

grain_frequency

K K Grain frequency

$ K Grain Pitch >>> envelope



The soundfile playback rate controls where in the soundfile each new grain begins reading.

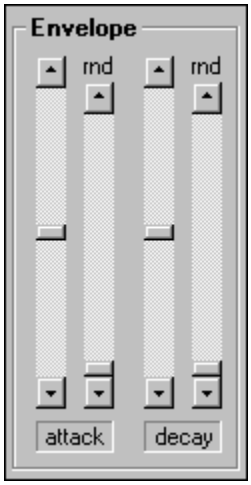
Grain **pitch** decides at which rate the grain will play back its fragment of the sound. This works like a normal sampler, or like interfering with the speed of a gramophone turntable, but since the starting point of each grain is controlled separately, the Donald Duck or tape slow-down effects do not appear with short grains.

a->p: Amplitude modulation of pitch can be used to enhance drum loops, etc.

Grain **gliss** controls pitch change during the play time of each individual grain. The grain pitch may slide up to +/- 8 octaves away from the pitch at the beginning of the grain.

$ $ Grain Pitch

$ K Envelope >>> amplitude & panning

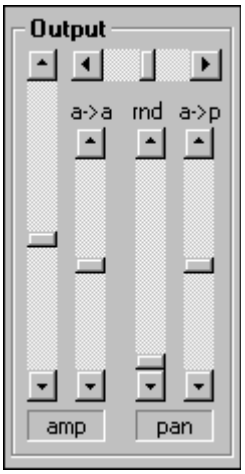


The envelope sliders control how large portion of each grain is spent for fade-in and fade-out, in percents of total grain length.

$ $ Envelope

[#][$][K]Amplitude and [#][K]Panning >>> patch banks

GranuLab 1.5 version



Amplitude:

The leftmost slider sets stream amplitude.

The **a->a** slider gives compression expansion effect, as amplitude of input sound controls the amplitude of grains;

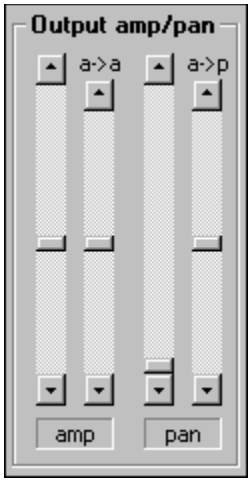
Panning:

The top horizontal slider sets stream panning in output stereo image (less useful with **GranuLab 1.1**)

The **rnd** slider sets the amount of left-right randomisation of grains. This is most useful with mono soundfiles output in stereo.

The slider **a->p** sets the amount of panning controlled by amplitude of input sound.

GranuLab 1.0 version



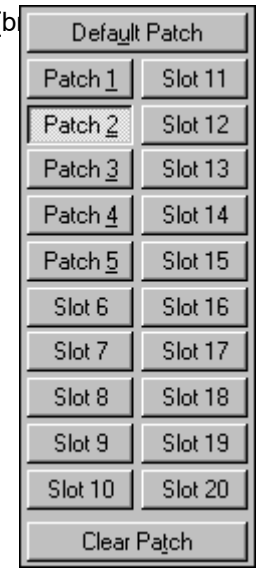
The **GranuLab 1.0** amp and panning controls work similarly to the above, but lacks the stream left-right positioning slider.

$ [$] Amplitude and Panning

$ K # KPatch Banks >>> gesture window

p a t c h e s

{b



A **patch** is the full set of slider values in the **GranuLab** interface. There are 20 patch slots per bank.

Each patch slot holds one full set of slider values. To recall a patch, just click on the button. In detail:

Commands on patch buttons:

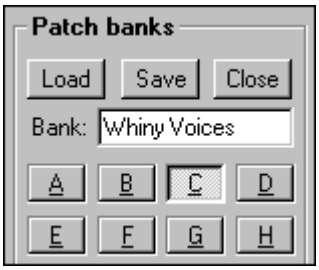
click - set patch values as TARGET for all sliders, subject to portamento. *(invalid on empty slot)*
CTRL+click - set patch values as ACTUAL value for sliders, no portamento. *(invalid on empty slot)*

SHIFT+click - store current TARGET slider values.
SHIFT+CTRL+click - store current ACTUAL values.

Patches are subject to portamento, just as if each slider had been adjusted individually.

The ***Default patch*** slot cannot be changed. It gives (roughly) straight playback of a sound file.

b a n k s



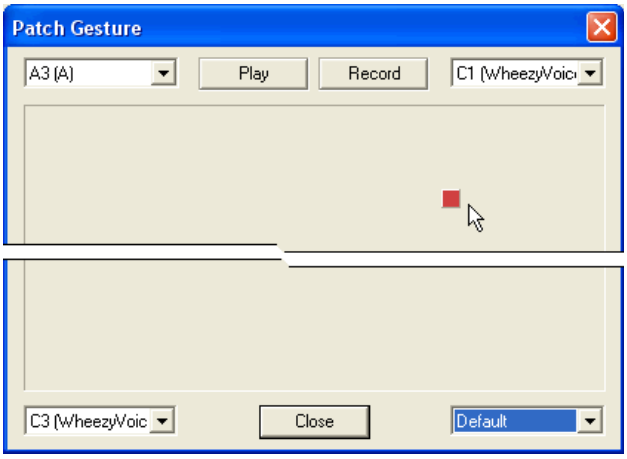
There are eight banks of 20 patches each, accessible from the **A** through **H** buttons.

Each bank can be saved to a file. The name in the text-edit field is used as a file name, so only use valid file-name symbols. The patch bank files are by default all saved to the **\Patches** folder in **GranuLab**'s home folder.

The only way to change the active patch folder is by loading a new patch from another folder. (I wanted minimal use of file dialogs, as they interfere with output sound.)

In **GranuLab 8 Pro**, each user (or project) handle set in Login window gets its own folder under the global patch folder.

$ KKGesture Window >>> granny overview >>> store sound



This is a special window where you may assign a patch to each corner, and then crossfade between the four patches by mouse control. This window is opened from the Command | Open Gesture Window menu. All currently available patches at the time of opening the window appear in the drop-down list boxes in the corners. The list is refreshed when the window is closed and reopened.

SHIFT+click when closing the gesture window to import the patch mix into **GranuLab**'s main window.

The generated gesture patch is applied to the topmost grain stream.

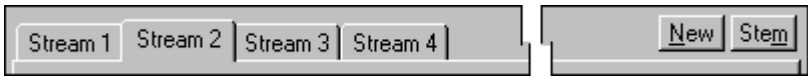
The Gesture Window may be folded up or out by double-clicking in its title bar, to get it out of the way temporarily. When it is folded down again, the set of patches is refreshed.

#$ K KMultiple Streams >>> granny overview >>> versions

s t r e a m s

The whole set of grain parameters applied to a soundfile, outputting sound, is called a *grain stream*.

In **GranuLab 1.5**, there may be up to 8 grain streams. These are accessible from tabs in the interface. Each tab reveals a complete, independent set of grain parameters. These may be applied to the same soundfile, in or out of sync, or to a different soundfile for each stream.

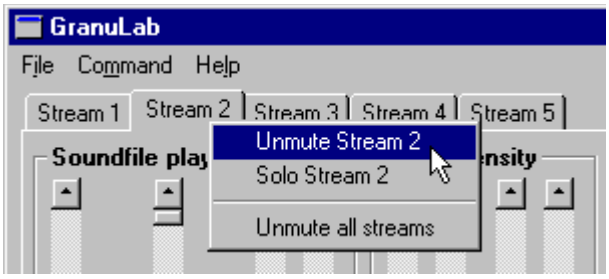


When you hit the **New** button, the current stream is copied to a new stream. (This is the only way of guaranteeing that a soundfile is used by two different streams in sync.)

Stem will remove the currently visible stream.

K Ks o l o a n d m u t e

Right-click a stream tab label to show a popup menu which offers standard solo and mute options. The stream tab does not need to be topmost.



---

multiple_streams

$ Multiple Streams

K K Multiple streams

K GranuLab 8

K Solo stream

K K Mute stream

#$KKLogin Window >>> granny overview >>> versions

(Please note: These features are available in **GranuLab 8 Pro** only.)

See also **GranuLab 8** features.

multiple preference configuration storage

The **Login Window** by default opens at program startup, and lets you log in under any name or text string. All preferences (file paths and output/MIDI options etc) will be saved between program runs separately for each login. This may be convenient in a multi-user environment.

During program run, a new login can be opened or created from the File | New Login... menu.

To delete a login handle, select the entry, then hold down CTRL+SHIFT and click the **Log In** button

---

login_window

$ Login Window

K Login window

K GranuLab 8

$ K Storing Output Sound to File >>> i/o options

Hit the **>>>File** (output to file) button to store the generated output sound to a WAV file.

There are two options for the name of the output file: Either it is always called **granny.wav** and is overwritten every time you hit **>>>File**; or it is called **granny##.wav**, where ## are two digits which are incremented for each new recording. This option is set under menu Command | Use Numbered Out Files. The output file folder can be set by a dialog invoked from menu Command | Output Soundfile Options | Select Soundfile Folder.

You can select output sample rate, mono/stereo etc independently of the format of the input sound file. The input sound is completely resampled, so its sample rate does not affect performance.

t i p :

By recording a file, you can capture very dense grain structures even though your computer cannot keep up with generating the sound.

Make sure that the option **Soft stop** in Command | Grain Generation Options menu is NOT checked. While playing, use the **Q key** on the keyboard to stop sound output, but do not stop recording! As soon as sound output stops, you can click a patch slot button (or move any sliders). The slider values will change to their new settings in file time - according to the state of the portamento slider - rather than in real time.

When you hit the **Play** button to restart sound production, the recorded file will not have the chopped clicky sound that you hear while recording the only problem is timing when to change patches while the output sound is chopped and delayed (since the processor cannot keep up).

---

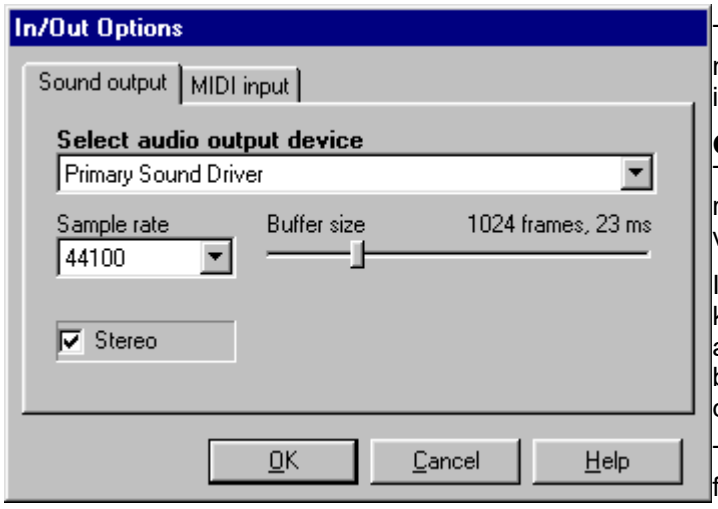
soundfile_output

$ Recording Output Soundfile

K Recording output soundfile

$ K# K K I/O Options >>> store sound >>> midi support

s o u n d o p t i o n s >>> midi options



This dialog is available under the Command menu. Here you select sound card and MIDI input device.

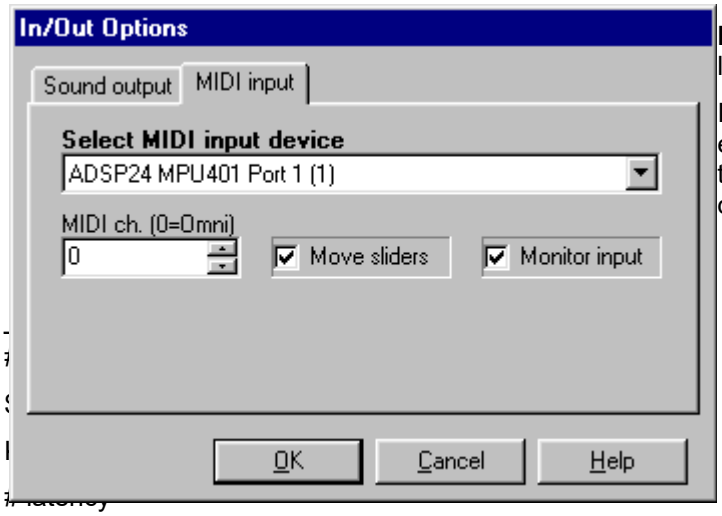
GranuLab supports DirectSound output. This gives around 20-30 milliseconds response time at lowest, but performance will vary.

If you play **GranuLab** from a MIDI keyboard, you'll want as small sound buffer as possible, but this will make the sound break up easily. With lower sampling rate you can also shorten the buffers.

The option **Soft stop** will make **GranuLab** fade out grains gracefully before stopping sound output this is to avoid clicks in the output sound. This is really necessary for MIDI note playing, and preferable at most other times too. **Soft stop** has the following consequences:

1. It will take **GranuLab** the latency time to quit performance. There will then be a release time for MIDI notes, which you can control only by minimising the output buffers.
2. This fade-out will also appear in any recorded sound file, so if you do not want them there, you will have to put up with clicks each time sound output stops. There is currently no way around this.

K M I D I o p t i o n s



Midi channels - selects a midi channel to listen to.

In **Omni mode** (input channel = 0) any MIDI event sent to channels 1-8 will be sent only to the corresponding stream. If the stream doesn't exist, the event is disregarded.

K K Latency

K K Soft stop option

midi_options

omni_mode

K Omni mode (MIDI support)

If MIDI input channel is set to a specific stream 1-16, **GranuLab 8** will only listen to that channel, and send all events to the currently visible stream (see streams).

Move sliders: If this box is checked, Granny will move the sliders visibly along with any MIDI controller input. This used to make a small difference on performance in 1997 (on a 486), but can now be checked always.

Monitor input: If checked, information about all MIDI input is displayed in the status bar. This has never affected program performance measurably, so it is recommended to keep this checked.

See further MIDI support and MIDI controller mapping.

using GranuLab with other MIDI programs

You may want to generate MIDI controller events in another program, and feed this to Granny. This is perfectly feasible, using a "MIDI loopback" program (**Hubi's Loopback** is recommended. It can be found by searching for that string eg in the Google search engine).

Note that any program generating real-time MIDI will be competing with **GranuLab** for CPU time. Many MIDI programs, (like Cakewalk or Cubase), will be quite aggressive about their system priority when playing, so that **GranuLab** is starved for processing time. This will lead to severe chopping of output sound, even for lower densities (on a 200 MHz Pentium MMX the maximum density immediately decreased by two-thirds). You should therefore try to find some fairly simple (or older) program to generate your MIDI events, or preferably use external hardware.

$ K MIDI support >>> controller mapping >>> i/o options

stream selection (GranuLab 8 only)

To send MIDI to a specific granny stream, set MIDI input channel to 0 = Omni in the Commands | "Sound I/O" dialog, "MIDI input" tab.

In Omni mode (MIDI input channel is set to ”channel 0”) any MIDI event sent to channels 1-8 will be sent only to the corresponding stream. If the stream doesn't exist, the event is disregarded.

If MIDI input channel is set to 1-16, Granny will only listen to that channel, and send all events to the currently visible stream (see streams).

NOTE: This is a quick hack, MIDI will still have to be redesigned in **GranuLab²**.

patch selection

Program change messages 0-20 selects the corresponding patch in the current bank. 0 is the default patch (If they appear as 1-21 in your MIDI gear, then default = 1, but I think you'll manage). Select bank **A-H** with program change numbers 30-37 (or 31-38).

GranuLab 8:
Patches are still always applied to the visible stream. This will be changed ASAP.

#MIDI controllers

GranuLab 1.1 and GranuLab 8: Crude controllers (4-32)

Since **GranuLab 8** has one more parameter than Granny 1.0, two more controllers are assigned (no. 32 and 64 for fine control).

Crude controllers work like in Granny 1.0. See MIDI Controller mapping for defaults.

GranuLab 1.1 and GranuLab 8: 14-bit and ”waggle” mode (controllers 36-64)

In **GranuLab** there are two modes of fine control. The old 14-bit mode is like Granny 1.0. The new ”waggle” mode is default, and the recommended mode for fine control.

14-bit mode: Since each MIDI controller has only 128 different values, fine (LSB) controllers can be used for 14-bit precision. Fine controller values are fractions of crude controller: a 127th of a 127th of the full slider length. The fine controller value 0-127 is added to the latest crude controller value. Check 14-bit Mode in the Commands | MIDI Options menu to set this mode. (Default: not selected)

NOTE: If a crude controller value is 127, this sets the max slider value, so its fine co-controller will be ineffective in this case (this is for users who don’t have 14-bit MIDI gear).

Waggle mode: Instead of adding fine controller value to the crude controller, the fine-controller is *added to the current value*. In this mode the fine-controller messages are read as +/-63 instead of 0-127.

This means that you can move over all values of the slider by sending lots of fine controller messages – waggling the controller slider if you have one. Or you can send a crude controller value and then adjust it by scrubbing a fine controller.

Uncheck 14-bit Mode in the Commands | MIDI Options menu to use this mode (This mode is the default).

GranuLab 1.0: Crude and fine controllers (4-31 and 36-63)

The sliders for grain generation may be set by MIDI controllers. Controllers 4-31 are assigned to the sliders from left to right (4 = soundfile loop start; 31 = portamento time). The sliders are just chopped in 127 equal parts. Sending a ”crude” controller message with value *N* will select the *Nth* part of its slider,

---

midi_controllers_desc

counting from bottom of the slider. The initial value of sliders are different numbers (0 for a slider from 0-N, 64 for a slider with range +/-N, etc).

Since each controller has only 128 different values, controllers 36-63 can be used for precision control (14-bit). Only some of the sliders need fine control see MIDI Controller mapping for assignment and default values.

PLEASE NOTE: Mouse slider movement does **not** update the controller value, so mouse and controller movement is not synched this way.

MIDI note input

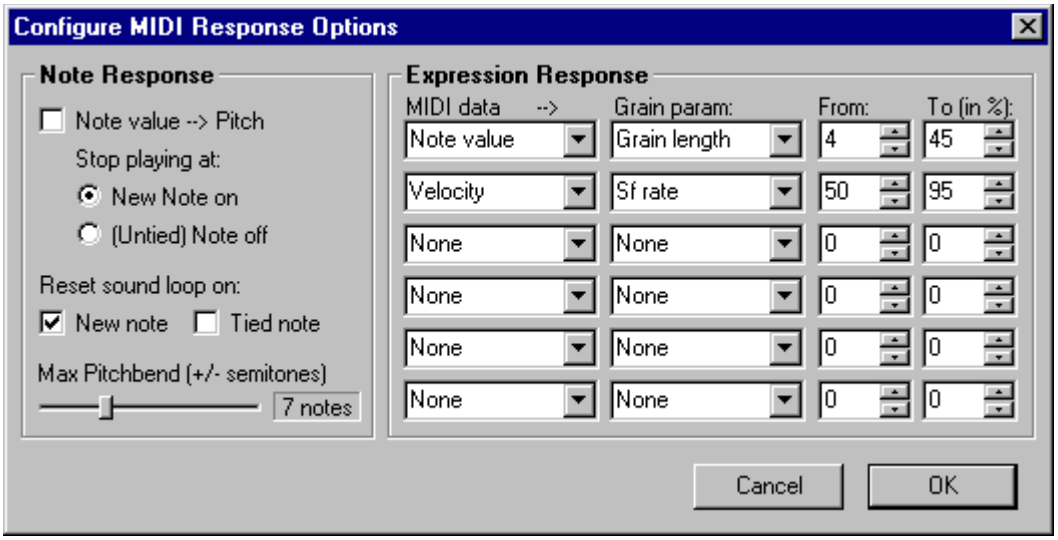
You can use MIDI note input to trigger **GranuLab**. The note values use middle C (Note value 60) for original soundfile pitch. If the pitch slider is in the lower half (reversed grain playback), MIDI notes will also use reversed playback.

GranuLab 8 notes:

In Omni mode, note events on MIDI channels 1-8 are directed to the same stream 1-8. If Granny only listens to one channel, note events on that channel only are sent to the visible stream.

In **GranuLab 8**, If **Note --> Pitch** is selected in the MIDI Response dialog, MIDI note events will start/stop performance of all streams (I couldn't get around that without some deeper refurnishing). Each stream is thus monophonic, ie the stream pitch is set by the latest note value or pitchbend sent to that channel.

K MIDI response



In this dialog you can choose to let common MIDI note data (ie velocity/aftertouch and note/pitchbend) control some of the grain generation parameters. The MIDI data are kept within a range of parameter slider's length, which is given as a percentage. Thus, if you want pitchbend to control grain gliss over the positive half of the grain gliss slider, you set the values From = 50%; To = 100% (or inverse).If you use the expression values to control grain frequency or grain length, it's advisable to use the same MIDI expression type to control the other parameter inversely else you will probably get too high densities.

GranuLab 8 notes:

midi_response

K Midi response dialog

GranuLab 8: If **Note -> Pitch** is selected, MIDI note events will start/stop performance of all streams (I couldn't get around that without some deeper refurnishing). Each stream is thus monophonic, ie the stream pitch is set by the latest note value or pitchbend sent to the stream channel.

GranuLab 8: MIDI Response cannot behave differently for different streams, but it will of course only respond to input on the stream channel! So if you set velocity to control panning, it will do so on every stream.

(This will of course be fixed in later versions but will require bigger internal changes.)

$ K MIDI Controller mapping >>> midi support

Table of **GranuLab** MIDI input controller assignment and default values.

(fine controllers marked with a grey asterisk (*)
will generally not be useful in 14-bit mode,
since they have very small effect on the sound)

Crude (fine)	Slider	Crude default values
Soundfile playback group		
4 (36)	Start position	0
5 (37*)	Start position, random	0
6 (38)	Loop length	127
7 (39*)	Loop length, random	0
8 (40)	Playback rate	96
9 (41*)	Playback rate, scale	0
10 (42*)	Playback rate, amp mod	64
Grain density group		
11 (43)	Grain frequency	48
12 (44*)	Grain frequency, random	0
13 (45*)	Grain frequency, amp mod	64
14 (46)	Grain length	10
15 (47*)	Grain length, random	0
16 (48*)	Grain length, amp mod	64
Grain pitch group		
17 (49)	Pitch	88
18 (50*)	Pitch, random	0
19 (51*)	Pitch, amp mod	64
20 (52)	Glissando	64
21 (53*)	Glissando, random	0
22 (54*)	Glissando, amp mod	64
Envelope group		
23 (55*)	Attack	64
24 (56*)	Attack, random	0
25 (57*)	Decay	64
26 (58*)	Decay, random	0
Output group		
27 (59*)	Output amp	58
28 (60*)	Grain amp, amp mod	64
29 (61*)	Grain random panning	0
30 (62*)	Grain rand pan, amp mod	64
(GranuLab 1.5)		
31 (63)	Panning	64
32 (64)	Parameter change time	2
(GranuLab 1.0)		
31 (63)	Parameter change time	64

In GranuLab 1.0 and earlier versions, panning does not
exist, so Parameter change time uses controller 31.

---

midi_controllers

$ Midi Slider Assignment
K Midi slider assignment

#$KKContact & UnCommercial Info

contact

mail: rasmus.ekman@abc.se

web: https://www.abc.se/~re/Granulab/Granny.html

Below some of the information on the web is reproduced. Note that this may be outdated, so please refer to the above link for current information.

#version information & pricing

GranuLab 1.0 and 1.5:

All versions of GranuLab are freeware since some time.

November 2017

rasmus ekman

rasmus.ekman@abc.se

```
#
            granular_synth_info
# overview
$ GranuLab overview
K GranuLab overview
# global_params
$ Global parameters
K Grain generation parameters
# amplitude_modulation
 K        K Amplitude modulation
# soundfile_playback
 K        K Soundfile playback
 K        K Grain density
# grain_pitch
 K        K Grain pitch
# envelope
 K        K Envelope
# amplitude
 K        K Amplitude
# panning
 K        K Panning
# patch_banks
$ Patch banks
K Patch banks
# patches
 K        K Patches
# patch_gesture_window
$ Gesture Window
K Gesture window
# midi_support
$ Midi Features
K Midi features
# contact
$ Contact & Commercial Info
K Contact
```