Buzz Generators Help

TngKaos

2ndP LoopJump HACK v1.1 Oct18/2001
----
made by 2ndProcess, thanks to P.DOOM for the HACKpointers

This is FREE software.

This is version 1.1, some demos included, parameter descriptions/docs
reworked since v1.0

WARNING1!!!! compatibilty problems may occur with newer versions of Buzz
(which are not likely to be ever available, but who knows...) since it's
HACKED and uses some unofficial pointers!!!!

ANYWAY, I am NOT responsible for damage or anything else of the bad
things that my program may (but is not likely to) cause. IT'S YOUR RISK.

What can I do with this machine?

- easily make songs with pattern lengths not supported by Buzz
  (by skipping the last ticks of each pattern)

- skip or repeat patterns (relative jumps!)

INSTALLATION

Copy 2ndPLoopJumpHACK.dll into the Gear\Generators\ subfolder of you Buzz folder.
Open index.txt (in Gear\) and add this line

```
   2ndPLoopJumpHACK,2ndP LoopJump HACK
```

at Generators/Utilities (or whereever you want).

Parameters

Destination An "address number" in your sequence. You may want to
set this by hand, but usually the "remember" trigger
will be used to set this.

Remember	Writes current song position into "Destination", ie. the next "jump" will go to this place. Mark the first tick of a section to be repeated with this.
Increase/Decrease Destination	Adds/subtracts some patterns from "Destination". You can use this to jump back FROM current position (remember+decrease) or to skip.
Counter	Whenever you jump, this counter is decreased. Set it to how often you want to jump.
Jump	Counts down and triggers the jump if counter is > 0. Put this at the LAST TICK of the LAST PATTERN to be repeated.

NOTES

Have a look at the demo .bmx...

WORKING AT...

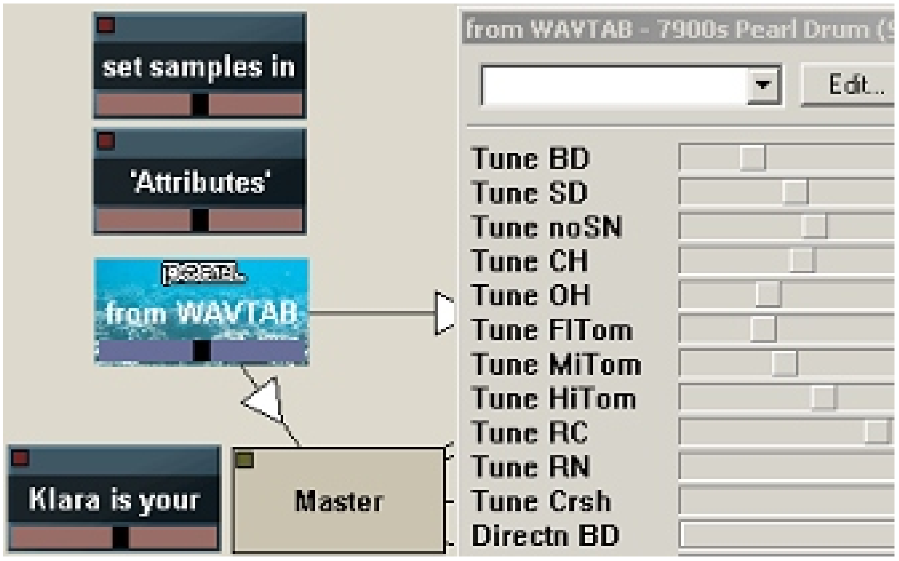
A new control machine for Buzz! Not sure how this will look like...
not sure if I'll ever finish it :(MAYBE with CHORD ENTRY, CONTROLLED
RANDOMIZATION and such fuss...

A low-cpu mixer, just muliple input panning :)

ALL DEVELOPERS, _PLEASE_ READ THIS!!!

Whenever you do machines with ranomizable values... Please allow to set the random seed with a parameter so we can actually COMPOSE with random!

Pearl Drum 1.1 new final release
by 7900 and MvA



features

+ loads wavetable samples, set the wavenumber in the 'attributes' menu (right-click)
   you can use your favorite techno or dnb drumkits
+ original Pearl (rock-) Drums samples as default
+ retrigger 1 = trigger once, 2 = trigger twice in a tick, FF is trigger 255 x in a tick, 0 = off
+ reverse/normal switch per sample
+ tuning per sample
+ delay, feedback per sample
+ panning per sample
+ sleek design :P
+ 100% backw compatible with the old fellow

- stereo waves arent supposed to be used; convert them (they're only played 1/2 way)
- some clicking in the delay
- we wont fix any bugs

Source code not included because it's messy and rather BIG.
If you intent coding for Buzz I recommend taking a look at Whitenoises Looper 2 for
wavetable acces. Leave me a message and I'll send you the stuff anyway.
Same story for bugfixing.
- - - - - - -
Code niet inbegrepen; het is rommelig en nogal veel.
Wil je Buzz plugins schrijven dan naar Whitenoises Looper 2 voor Wavetable toegang. Of stuur een berichtje dan krijgt je alle troep.
Bugfixen zelfde verhaal.

or .. mail me :
(to send me music !)

Pro3 SYNTHTRACKER Version 1.0

What is this? Stereo Tracker **-?effect?-** for **buzz**

Coded by **Arguru gx11488@agger.net (bug + comments + feedback)**

Date **12-July-1999**

Current Version **1.2 First Release**

Type **EFFECT!!!,** yes, only effects can produce stereo sygnal in buzz... but use it as a generator, the only you have to do: connect a dummy real-generator to it, bass3 per example.

NOT done yet!!!:

Xi-support, vibrato effect, anti-clicker, envelope handling and pingpong loop.

Wav Loader

The machine includes a command (press right mouse button on the machine in the editor to view commands on buzz-machines) that will load wav samples into the wavetable.

Handles WAV PCM 8Bit/16Bit/Mono/Stereo samples.

Command List

Each track included...

note/wave col, volume col, pan col (0x00 Left, 0x40 Center, 0x80 Right), and four cols for the effects (Command, Value, Command 2, Value 2). So you can combine two effects per line + volume + panning.

0x01	Pitch Up	Slide Up the pitch
0x02	Pitch Down	Slide Down the pitch
0x03	Set Glide	Set track gliding amount (0x00 - No Gliding, 0x80 - Long gliding)
0x04	Note Retrigger	Retrig Note every 'n' picks (Pick = Tick subdivision)
0x06	Play Mode	Track Playback mode: (0x00 = Normal, 0x01 = BackWard playback)
0x07	Distort Threshold	Set the track threshold distort level
0x08	Distort Clamp	If the sygnal produced by the track is higher than the threshold level, the sygnal will be the clamp
0x0A	Vol. Down	Slides down the volume, like 'Axx' command of Ft2.
0x0B	Vol. Up	Slides up the volume, like 'Axx' command of Ft2.
0x0C	Set CutOff	Set the filter cutoff
0x0D	Set Resonance	Set the filter resonance level.
0x0E	Set Filter Type	Set the filter type. It only works with FILTER ALGO number '0x00' (Makk M3 / Robert Bristow-Johnson) 0x00 = Lowpass, 0x01 = Hipass, 0x02 = Bandpass, 0x03 = Bandreject, 0x04 = Thru (useless)
0x0F	Filter Algorithm	Set track filter algorithm, currently available 2... 0x00 = Multiband Makk M3 / Robert Bristow-Johnson 0x01 = Paul Kellett "2 Pole" Lowpass filter. Maximun value at resonance will produce Auto-Osc. >0x02 = No filter (Improve CPU speed) any ideas?...send them to gx11488@agger.net you can send me the C++ function-code for the filter, anyway, it doesn't needed to be a filter, it could be a echo, reverb chorus, distort, etc... cutoff, resonance and type values could be the parameters... Plugins inside plugins ***:)***

0x10	Cut Off down	Slide down the cutoff
0x11	Cut Off up	Slide up the cutoff
0x12	Note Cut	Note cut after 'n' picks
0x13	Set Flat Dist	Set threshold and clamp at the same level. This produces flat distorsion. Very cool and useable...
0x14	Reset F+D	Reset to default values the filter and the threshold/clamp values (no filter and no distort). The effect value is useless, anyway include it, if you don't it won't work
0x15	Reset All	Reset all track values to default. The same as 'Reset F+D'
0x16	Set Finetune	Set the track finetune. The tracker handles the wavetable rate-value too but, this is better and most friendly. (0x00=-1/2 Halftone, 0x40 = No finetuning, 0x80 = +1/2 Halftone)
0x17	Random Cutoff	This randomizes the filter cutoff. Very cool...
0x20	Volume inertia	To avoid volume changing clicking, the tracker use transitions routines, this value changes the internal track-volume-inertia. Higher values = smoother transitions, Lower = Fast and clicking transitions.
0x21	Pan inertia	The same as volume inertia, but with panning.
0x22	Cutoff inertia	It needed any comments?
0x23	All inertias	...
0x24	Feedback	Set the feedback amount of the built-in-delay fx. (0x00 - No feedback, 0x80 - Full feedback)
0x25	Delay send	Track-To-Delay send amount like Steinberg "Rebirth". (0x00 - No send, 0x80 - Full send)
0x26	Set Delay Time in ticks	Sets the delay time in ticks.
0x27	Set Delay Time in miliseconds	Sets the delay time in miliseconds.
0x28	Set delay type	Delay mode (0x00 = Normal, 0x01 = Cross Delay)

Arguelles Pro4

(a tracker plug-in for Buzz)

known commands:

Commands	Description
00	(no effect)
01	Pitch Down
02	Pitch Up
03	Set Glide
04	Retrigger
05	(not implemented)
06	Set PlayMode
07	Dist. Threshold
08	Dist. Clamp
09	Play Offset
0A	Volume Down
0B	Volume Up
0C	Set Cutoff
0D	Set Resonance
0E	Set Filter Type
0F	(not implemented)
10	Cutoff Down
11	Cutoff Up
12	Note Cut
13	Flat Dist.
14	Reset F+Â

15	Reset Track
16	Track Fine Tune
17	Randomize Cutoff
18	(not implemented)
19	(not implemented)
1A	(not implemented)
1B	(not implemented)
1C	(not implemented)
1D	(not implemented)
1E	(not implemented)
1F	(not implemented)
20	Volume Inertia
21	(not implemented)
22	Cutoff Inertia
23	(not implemented)

BTDSys LiveJumpHACK 1.0

Installation

Unzip all files into your **Gear\Generators** folder.

What is it?

LiveJumpHACK lets you jump to any song position assigned to specific keys on your MIDI or computer keyboard. For example, press C-4 and jump to tick 256, press the 'a' key and jump to tick 512, etc.

Similiar in function, but not identical to performance tools like Abelton Live.

LiveJumpHACK is a "hack" machine based on P. DooM's BUZZHACK with some additions by Ed Powley. Basic setup and operation is simple, right click the LiveJumpHACK machine and click 'settings'.

Settings Dialog

Settings Dialog screenshot

For the most part, this dialog is self-explanatory. To assign a jump to a MIDI key, simply press the key so that it is displayed in the button in the top right corner, choose the tick and jump mode, and click Add. To assign a jump to a key on the computer's keyboard, click the button in the top right corner, press the desired key, and add the jump as before. To remove an assigned jump, highlight it and click Remove. For explanantion of the jump modes (the "Absolute" dropdown in the screenshot), see the parameter descriptions below.

Key capture window

To trigger jumps from the computer keyboard, the key capture window must be active. This is not required for MIDI operation.

The key capture window also displays useful information regarding the state of the machine:

- `ready to rok`: the machine is ready to receive jump commands.
- `press play`: the machine does not do anything if the song is not playing.
- `machine is off`: the On/Off parameter is in the Off position, so the machine is inactive.
- `goto` *`x`* `in` *`y`*: the machine will jump to tick *x* in *y* ticks' time.

Parameters

- **Tick snap:** This allows you to quantise your jumps. All jumps will be delayed until the current song position is divisible by this quantity. For example, with the default setting of 16, if you press a key at 7 ticks into a pattern, it will wait until the 16th tick (9 ticks later) before it jumps.
- **Note trigger:** Entering values for this parameter acts exactly as if the corresponding MIDI key had been pressed, so any assigned jumps are triggered.
- **Tick number trigger:** Allows you to trigger a jump to an arbitrary song position. Used in conjunction with the tick jump mode parameter.
- **Tick jump mode:** Sets the meaning of the number entered for the tick number trigger. This has no effect on jumps triggered by keypresses or the note trigger -- the modes for these are specified in the Settings dialog.
    - *Absolute:* jumps directly to the specified tick, no matter where the jump is triggered from.
    - *Relative back:* jumps backwards the specified number of ticks. Note that this is counted from the point where the jump actually takes effect, not where the jump is triggered.

    - *Relative forward:* as relative back, but jumping forwards by the specified number of ticks.
- **On/Off:** Allows you to completely disable the machine.

Attributes

- **MIDI Channel:** self-explanatory. Note that channel numbering starts at zero, which may be contrary to other machines or hardware.
- **Keep jumping until note off:** If this is set to 1, jumps will be continually triggered for as long as you hold a key down.

Hints

Looping

You can use LiveJumpHACK to set up "loop zones" within your song. This is useful for live performance, as you can effectively assign certain patterns to specific keys.

For example: let's say you want to loop ticks 0 to 32 with key C-4, and ticks 64 to 128 with key D-4.

- Assign C-4 to jump to tick 0, and D-4 to jump to tick 64.
- Create a pattern for LiveJumpHACK. In the first row, enter 0000 for the tick number trigger and 00 for the jump mode. Alternatively, enter C-4 for the note trigger.
- Create another pattern as above, this time entering 0040 for the tick number or D-4 for the note.
- Place these two patterns in the sequence editor, on ticks 16 and 112 respectively (assuming you left the tick snap setting at 16).
- Play your song, and notes C-4 and D-4 should function as desired. Note that to jump between regions, you must press the key after the sequenced jump has been triggered, but before the jump occurs (ie in the last 16 ticks of the loop).

Alternately, if all your loops are the same length, simply set the tick snap parameter to your loop length, set the "keep jumping until note off" attribute to 1, and hold down the keys.

Computer keyboard control

Don't forget, for this to work, the key capture window must be open and active. Depending on demand, a future version may remove this restriction. In contrast, MIDI functionality works all the time.

Thanks

Thanks to mute for ideas and testing. Also thanks to Paul Eye, silicon/silicium, Spark, nool, and anyone else who tested this machine and gave me feedback.

EOF/legal

BTDSys Peer LFO - This is an ALHPA version so be careful!

Installation

Put *BTDSys Peer LFO.dll* in your **Gear\Generators** folder.

Overview

Peer LFO is an LFO (Low Frequency Oscillator) which you can use to modulate any parameter of any other Buzz machine.

- Add Peer LFO to your song, along with some other machine.
- You don't need to connect anything to Peer LFO in the machine view.
- Right click Peer LFO and select Assign Parameter, then click an unassigned track.
- In this dialog, choose the machine and parameter you want to control.
- Click OK, open the Peer LFO parameter window, and switch On/Off to On.
- If you want to control more than one parameter from the same LFO, add more tracks to the machine (Pattern view, Ctrl +). Then assign them as above.

Parameters

Global (machine) parameters

- **On/Off** - activates or deactivates the machine.
- **Manual Phase -** allows you to manually set the phase (position in the cycle) of the wave.
- **Wave Shape -** the type of wave to be used. *Random* sets a new value of the parameter at every time interval specified by the Rate, and *Wander* is similar apart from it travels smoothly to the next random vlaue rather than jumping straight to it. *Wavetable* uses a user-defined wave stored in the Buzz wavetable. The remainder (*Sine, Tri* etc) are simple oscillator waveforms as you may expect.
- **Wave Number -** if Wave Shape is set to Wavetable, this is the wave to use from the Buzz wavetable. If it's a stereo wave, only the left channel will be used.
- **Min Diff -** when using Random or Wander modes, this specifies the minimum difference between consecutive values. This can be used to guarantee a noticable change in the parameter every time interval.
- **Rate -** sets either the period (in Ticks/16 or ms) or the frequency (in Hz/256) of the wave.
- **Rate Unit -** sets the units of the Rate parameter, either 16ths of ticks, milliseconds or 256ths of

Hertz.
- **Inertia** - glide time for Min, Max and Period parameters.

Track parameters

- **Min** and **Max** - the minimum and maximum values of the LFO wave (given as a percentage of the controlled parameter's range)
- **Center** and **Amplitude** - an alternative way of setting the LFO's range.
- **Extent Mode** - sets which of min/max and center/amp is used. Note that from the machine's right-click menu you can synchronise the two pairs of values (so they represent equivalent ranges).
- **Track** - if you assign to a [T]rack parameter, which track will be controlled. If it's set to All, then all the machine's tracks are controlled. If you assign to a [G]lobal parameter, this has no effect.

Attribute **Heed Stop Btn** defines whether the LFO switches itself off when you press the Stop button in Buzz. By default, it is set to 0 (meaning no, it doesn't).

Contact

If you have comments or suggestions, or if you find any bugs please email me.
Also visit my website (even though there's not a lot of interesting stuff on it just yet)

BTDSys Peer LFO

Installation

Put *BTDSys Peer LFO.dll* in your **Gear\Generators** folder.

Overview

Peer LFO is an LFO (Low Frequency Oscillator) which you can use to modulate any parameter of any other Buzz machine.

- Add Peer LFO to your song, along with some other machine.
- You don't need to connect anything to Peer LFO in the machine view.
- Right click Peer LFO and select Assign Parameter, then click an unassigned track.
- In this dialog, choose the machine and parameter you want to control.
- Click OK, open the Peer LFO parameter window, and switch On/Off to On.
- If you want to control more than one parameter from the same LFO, add more tracks to the machine (Pattern view, Ctrl +). Then assign them as above.

Parameters

Global (machine) parameters

- **On/Off -** activates or deactivates the machine.
- **Amp -** global amplitude, affecting the amplitude of every track. Can be used to "fade" the LFO effect in or out.
- **Manual Phase -** allows you to manually set the phase (position in the cycle) of the wave.
- **Wave Shape -** the type of wave to be used.
  - *Random* sets a new value of the parameter at every time interval specified by the Rate.
  - *Wander* is similar to Random, apart from it travels smoothly to the next random vlaue rather than jumping straight to it.
  - *Wavetable* uses a user-defined wave stored in the Buzz wavetable.
  - *Min*, *Max* and *Center* send constant values, the minimum, maximum and center of the defined control range respectively. These allow you to set the range parameters up while watching the changing values in the controlled machine's parameter window.
  - The remainder (*Sine*, *Tri* etc) are simple oscillator waveforms as you may expect.
- **Wave Number -** if Wave Shape is set to Wavetable, this is the wave to use from the Buzz wavetable. If it's a stereo wave, only the left channel will be used.
- **Min Diff -** when using Random or Wander modes, this specifies the minimum difference

between consecutive values. This can be used to guarantee a noticable change in the parameter every time interval.
- **Rate** - sets either the period (in Ticks/16 or ms) or the frequency (in Hz/4096) of the wave.
- **Rate Unit** - sets the units of the Rate parameter, either 16ths of ticks, milliseconds or 4096ths of Hertz.
- **Inertia** - glide time for Amp, Min, Max and Period parameters.

Track parameters

- **Min** and **Max** - the minimum and maximum values of the LFO wave (given as a percentage of the controlled parameter's range)
- **Center** and **Amplitude** - an alternative way of setting the LFO's range.
- **Extent Mode** - sets which of min/max and center/amp is used. The other is simply ignored. Note that from the machine's right-click menu you can synchronise the two pairs of values (so they represent equivalent ranges).
- **Update Freq** - sets how frequently the LFO actually sends its values to a parameter, if you don't want it to every tick.
- **Phase** - sets the relative phase offset compared to the "global" phase.
- **Track** - if you assign to a [T]rack parameter, which track will be controlled. If it's set to All, then all the machine's tracks are controlled. If you assign to a [G]lobal parameter, this has no effect.

Attribute **Heed Stop Btn** defines whether the LFO switches itself off when you press the Stop button in Buzz. By default, it is set to 0 (meaning no, it doesn't).

Other Notes

- PeerLFO can only change the value of a parameter once every tick. This is due to the way Buzz works. If control isn't smooth enough, either increase the TPB value of your song, or use inertia (if the target machine has it) of length 1.0 tick.
- Note that PeerLFO continues to be active, even if it is muted or another machine is in solo mode. However, if it is muted when it is loaded (eg because you added it while another machine was solo, or because this was its state in a loaded song file), it will not initialise itself (ie it will remain inactive) until it is un-muted.
- If you import a song file containing machines whose names are already in use in the current song, the new machines will be renamed by Buzz. However, any Peer LFO machines in the imported song will no longer recognise the renamed machines, and will control the "old" named machines which were already in the song. To work around this, please make sure you give your machines unique names in all your template files, and don't import a file into itself.

Contact

If you have comments or suggestions, or if you find any bugs please email me.
Also visit my website.

Thanks to everyone who alpha tested this machine, suggested features and pointed out bugs, without whom this machine would suck a lot worse

BTDSys PeerADSR

Installation

Put *BTDSys PeerADSR.dll* in your **Gear\Generators** folder.

Overview

PeerADSR is an envelope trigger, used to control other Buzz machines. Envelopes can be set up using ADSR parameters, the Buzz envelope.ocx control, or from wavetable samples.

- Add PeerADSR to your song, along with some other machine.
- You don't need to connect anything to PeerADSR in the machine view.
- Right click PeerADSR and select Assign Parameter, then click an unassigned track.
- In this dialog, choose the machine and parameter you want to control.
- Click OK. Now enter '1's into the trigger column in the pattern view.

About Envelope Types

Three types of envelopes are available in PeerADSR:

- An ADSR envelope, whose shape is specified by the appropriate parameters.
- 'Custom' envelopes, drawn using Buzz's envelope.ocx control (in the machine's Custom Envelopes dialog). The machine may store up to 200 of these at a time.
  Note that if a Sustain point is specified in the envelope, an Off trigger (0 in the Trigger parameter) must be used to end the sustain phase.
- Envelopes based on sample data from the wavetable. You may choose any one of the 200 samples in Buzz's wavetable.

Parameters

Track parameters

- **Trigger** - a value of 1 triggers the envelope from the beginning. A value of 0 forces the currently playing envelope (if any) into its Release phase.
- **Envelope Type** - specifies the type of envelope to be used.
- **Envelope Number** - specifies either the number of the custom envelope, or the wavetable sample number to be used. If ADSR envelope mode is used, this parameter is ignored.

- **Envelope Length** - specifies the total duration of custom or wavetable envelopes. If ADSR envelope mode is used, this parameter is ignored.
- **Min** and **Max** - the minimum and maximum values of the envelope (given as a percentage of the controlled parameter's range)
- **Attack**, **Decay**, **Sustain Level**, **Sustain Time**, **Release** - the characteristics of the ADSR envelope. These are ignored if Custom or Wvaetable envelope mode is used.
  The (poorly done) ASCII diagram below shows the meaning of the parameters (in case you didn't already know):

```
100%      /\    Sus Time
         /  \ <------------>
        /    \_____________
       /   <->             \    ^
      /   Decay             \   | Sus Level
0%   /                       \  v
     <---->                <->
     Attack              Release
```

  Note that if Sustain Time is set to infinite, an Off trigger (0 in the Trigger parameter) must be used to end the sustain phase.
- **Track** - if you assign to a [T]rack parameter, which track will be controlled. If it's set to All, then all the machine's tracks are controlled. If you assign to a [G]lobal parameter, this has no effect.

Attributes

- **Length Max** - sets the maximum possible value of the Length parameters (in ticks). The parameters are mapped linearly with this value at parameter value 0xFFFE, and a value of 0 ticks at parameter value 0x0000 (theoretically).
- **Subtick Control** - sets whether the target machine will be controlled between Buzz's ticks. You may set this to 0 (off) if you experience compatibility problems with some machines, or you specifically want a "stepped" sound.
- **MIDI Channel** - the MIDI channel number the machine will respond to. If this is set to 0, MIDI functionality is disabled. If set to 17, the machine will respond to all channels.
- **MIDI Track** - which track of the PeerADSR machine will be triggered by MIDI notes.
- **MIDI Want Note Offs** - sets whether PeerADSR responds to released notes (note off messages).

The MIDI functionality triggers the envelope in response to all MIDI notes on the specified channel.

Other Notes

- Note that if the PeerADSR machine is muted, it will cease to function properly. Hence do not mute the machine.

- To smooth out the parameter movements on the target machine, you should use inertia if available. Set it to a very short length (0.1 tick or less).

Contact

If you have comments or suggestions, or if you find any bugs please email me.
This machine is DONATIONWARE, so if you like it, send me something cool (eg CDs, hardware, money etc). Email for details of how to get stuff to me.
Also visit my website (y'know, if you're bored).

BTDSys PeerChord *Lite*

Installation

Put *BTDSys PeerChord Lite.dll* in your **Gear\Generators** folder.

Overview

PeerChord is a chord/arpeggio player, which can control any other machine in Buzz (provided the machine supports notes and tracks). Also facilities for 'humanisation' (simulating a human player) are provided.
The *Lite* version has only the features you need to quickly play simple chords and arpeggios. For finer control over all aspects of your chords/arps, use the full version (BTDSys PeerChord).

- Add PeerChord to your song.
- You don't need to connect any other machines to PeerChord.
- Right click PeerChord and select Assign Parameter, then click an unassigned track.
- In this dialog, choose the machine you want to control, and either a note parameter, velocity parameter, or both.
- Click OK.
- Add at least six tracks to the target machine.
- Add a pattern to PeerChord, playing notes.
- If you want to control more than one parameter from the same PeerChord, add more tracks to the machine (Pattern view, Ctrl +). Then assign them as above.

Parameters

- **Root** - sets the root note (lowest note) of the chord.
- **Chord Type** - sets the type of chord to use. For a list of chords and their numbers, right-click the machine and choose About.
- **Velocity** - the velocity (volume level) of the chord.
- **Random Velocity Deviation** - sets how far from the set velocity the notes may stray at random (to simulate the slight variations in velocity of a human player).
- **Delay Mode** - Sets the mode used for delaying.
  - *Off* triggers all notes immediately
  - *Simple* delays all notes by a set amount
  - *Random* delays all notes by different random amounts (as if a keyboardist did not hit all notes at precisely the same time)
  - *Strum* modes simulate a guitarist's strum down or up the strings.

Note that these modes are overridden if any arpeggio mode is set.

- **Delay Length** - Depending on the Delay Mode used, either sets the time to delay notes by (*simple*), the maximum of the random delay (*random*), or the time between consecutive notes (*strum*).
- **Arpeggio Mode** - Sets the type of arpeggio to use.
  - *Off* plays a simple chord (all notes at once)
  - *Up* and *Down* simply loop up or down the notes in the chord.
  - *Up/Down* and *Down/Up* swap directions when the first or last note in the chord is reached.
  - *Random* chooses notes from the chord at random.
  - *'Brownian'* selects the next note based on the previous note. The next note will either be one step higher, one step lower, or the same as the previous note.
- **Arpeggio Steps** - Sets the number of notes to use for the arpeggio. If this is more than the number of notes in the chord, octaves of the notes are used to obtain the required number of notes. Note that this is not necessarily the same as the time taken for one complete cycle of the arpeggio. It is for up and down modes, but for up/down and down/up the length of a complete cycle is actually 2 × (Arpeggio Steps - 1).
- **Arpeggio Step Length** - Specifies the duration between consecutive steps of the apreggio.
- **Arpeggio Reset** - Sets if the arpeggio is reset to its start point every time a new note is played (this applies both to tracked notes and MIDI notes). Note that if no note is currently playing (ie if a note off was played or you lifted your finger from the MIDI key) the arpeggio is reset anyway.
- **First Track** - This sets the first track of the target machine to use. At most 6 tracks will be used, so if you set this to 2 (for example), you should ensure tracks 2, 3, 4, 5, 6 and 7 are available on the target machine.

Attributes/MIDI functionality

The attributes of the machine all relate to MIDI functionality. If you play a note on a MIDI keyboard, PeerChord will play the current chord type with the MIDI note as root note, and with whatever delay/ arpeggio settings are specified.

- **MIDI channel** - the MIDI channel the machine listens to, or 0 disables MIDI functionality.
- **MIDI track** - the track on the PeerChord machine which receives the MIDI notes.
- **MIDI transpose** - allows incoming MIDI data to be transposed up or down by upto 2 octaves. Note that a value of 24 represents no transposition, 0 represents down 24 semitones (2 octaves), and 48 represents up 2 octaves.
- **MIDI velocity** - if this is set to 1, the Velocity of the chord is taken from the velocity of the MIDI key pressed. If it is set to 0, the value of the Velocity parameter is used.
- **MIDI snap to tick** - specifies whether chords are triggered immediately (0), or on the next tick (1). If Arpeggio mode is active, the arpeggio is always triggered on the next tick.

Note that MIDI support on PeerChord is monophonic, ie only the most recently played note is used.

Other notes

- If the PeerChord machine is muted, either directly or by soloing another machine, it may cease to function properly until it is unmuted again.
- PeerChord uses hack methods to trigger notes at times other than on the tick. However, it is possible that this could cause problems with some machines or with future versions of Buzz. Also the Record function in Buzz will not record all of PeerChord's parameter changes into the target machine's patterns.
  If you wish to disable the hack methods, please ensure:
  - Delay Mode is set to Off
  - Arpeggio Step Length represents a whole number of ticks.
  - MIDI snap to tick is set to 1 (on)

Contact

If you have comments or suggestions, or if you find any bugs please email me.
This machine is DONATIONWARE, so if you like it, send me something cool (eg CDs, hardware, money etc). Email for details of how to get stuff to me.
Also visit my website (y'know, if you're bored).

Thanks to everyone who alpha tested this machine for me and suggested features/pointed out bugs - particularly Raúl Reales (who gave me the original idea), Cameron Bonde (Vectrex), Juri Puumala, K. M. Krebs, Jeph Wacheski, Ronny Pries, and whoever else I forgot.

BTDSys PeerChord

Installation

Put *BTDSys PeerChord.dll* in your **Gear\Generators** folder.

Overview

PeerChord is a chord/arpeggio player, which can control any other machine in Buzz (provided the machine supports notes and tracks). Also facilities for 'humanisation' (simulating a human player) are provided.
This is the full version of PeerChord, giving you fine control over all aspects of your chords and arpeggios. If you do not need such flexibility and want a simpler machine to use, try the *Lite* version.

- Add PeerChord to your song.
- You don't need to connect any other machines to PeerChord.
- Right click PeerChord and select Assign Parameter, then click an unassigned track.
- In this dialog, choose the machine you want to control, and either a note parameter, velocity parameter, or both.
- Click OK.
- Add at least six tracks to the target machine.
- Add a pattern to PeerChord, playing notes.
- If you want to control more than one parameter from the same PeerChord, add more tracks to the machine (Pattern view, Ctrl +). Then assign them as above.

Parameters

- **Root** - sets the root note (lowest note) of the chord.
- **Chord Type** - sets the type of chord to use. *None* plays single notes, and *Custom* allows you to define your own chord using the Custom parameters. For a list of chords and their numbers, right-click the machine and choose About.
- **Velocity** - the velocity (volume level) of the chord.
- **Random Velocity Deviation** - sets how far from the set velocity the notes may stray at random (to simulate the slight variations in velocity of a human player).
- **Delay Mode** - Sets the mode used for delaying.
    - *Off* triggers all notes immediately
    - *Simple* delays all notes by a set amount
    - *Random* delays all notes by different random amounts (as if a keyboardist did not hit all notes at precisely the same time)

    - *Strum* modes simulate a guitarist's strum down or up the strings.

  Note that these modes are overridden if any arpeggio mode is set.
- **Delay Length -** Depending on the Delay Mode used, either sets the time to delay notes by (*simple*), the maximum of the random delay (*random*), or the time between consecutive notes (*strum*).
- **Note Cut -** Sets how long notes are sustained (they are stopped by sending note offs). Note that the target machine may cut notes off earlier; try adjusting the other machine's envelope settings if this is the case. If this is set to Infinite, then PeerChord will not attempt to cut notes.
- **Arpeggio Mode -** Sets the type of arpeggio to use.
    - *Off* plays a simple chord (all notes at once)
    - *Up* and *Down* simply loop up or down the notes in the chord.
    - *Up/Down* and *Down/Up* swap directions when the first or last note in the chord is reached.
    - *Random* chooses notes from the chord at random.
    - *'Brownian'* selects the next note based on the previous note. The next note will either be one step higher, one step lower, or the same as the previous note.
- **Arpeggio Steps -** Sets the number of notes to use for the arpeggio. If this is more than the number of notes in the chord, octaves of the notes are used to obtain the required number of notes. Note that this is not necessarily the same as the time taken for one complete cycle of the arpeggio. It is for up and down modes, but for up/down and down/up the length of a complete cycle is actually 2 × (Arpeggio Steps - 1).
- **Arpeggio Step Length -** Specifies the duration between consecutive steps of the apreggio.
- **Arpeggio Reset -** Sets if the arpeggio is reset to its start point every time a new note is played (this applies both to tracked notes and MIDI notes). Note that if no note is currently playing (ie if a note off was played or you lifted your finger from the MIDI key) the arpeggio is reset anyway.
- **Arpeggio Hold -** Allows arpeggio playback to be temporarily halted. A value of 1 will keep triggering the current note over and over, while a value of 2 will stop note triggering altogether. A value of 0 will resume arpeggio playback where it left off.
- **Arpeggio Shuffle Step -** Every nth arpeggio note will be delayed by some amount, where n is the value of this parameter. A value of 0 (off) will prevent any notes from being delayed.
- **Arpeggio Shuffle Length -** The amount every nth note is delayed by.
- **Arpeggio Shuffle Reset -** A value of 1 resets the shuffle counter to 0 (the counter which counts up to every nth note).
- **Octave R,2 / 3,4 / 5,6 -** Allow you to use alternate voicings and inversions, by setting the octave each note of the chord lies in (as offsets from the default octave). The parameters are "doubled up", meaning the first (most significant) hexadecimal digit sets the offset for the first note, and the second digit for the second note. So a value of 0x88 specifies the default octave settings (+0, +0), 0x87 specifies (+0, -1), 0x6A = (-2,+2), etc etc.
- **First Track -** This sets the first track of the target machine to use. At most 6 tracks will be used, so if you set this to 2 (for example), you should ensure tracks 2, 3, 4, 5, 6 and 7 are available on the target machine.
- **Custom Notes 2-6 -** Sets the notes used for the custom chord (if the Chord Type parameter is set to Custom). A chord always contains its root note, and up to 5 other notes above the root. These parameters set the offsets (in semitones) of these notes.

It is normal to specify these notes in ascending order (lowest to highest), but if you are aiming for special effects with the strum or arpeggio modes, you may use a different order.

Attributes

- **MIDI channel** - the MIDI channel the machine listens to, or 0 disables MIDI functionality.
- **MIDI track** - the track on the PeerChord machine which receives the MIDI notes.
- **MIDI transpose** - allows incoming MIDI data to be transposed up or down by upto 2 octaves. Note that a value of 24 represents no transposition, 0 represents down 24 semitones (2 octaves), and 48 represents up 2 octaves.
- **MIDI velocity** - if this is set to 1, the Velocity of the chord is taken from the velocity of the MIDI key pressed. If it is set to 0, the value of the Velocity parameter is used.
- **MIDI snap to tick** - specifies whether chords are triggered immediately (0), or on the next tick (1). If Arpeggio mode is active, the arpeggio is always triggered on the next tick.
- **Velocity compensation** - Lowers the note volumes when a chord with more than one note is played. A value of 0 gives no compensation, while a value of 1024 gives maximum compensation (so notes in a 2-note chord are $^1/_2$ the normal velocity, notes in a 3-note chord are $^1/_3$ the normal velocity, etc). Values between 0 and 1024 are between the two extremes. This applies to both chords played through patterns, and through MIDI (although not to arpeggios).

All but one of the attributes relate to MIDI functionality. If you play a note on a MIDI keyboard, PeerChord will play the current chord type with the MIDI note as root note, and with whatever delay/ arpeggio settings are specified. Note that MIDI support on PeerChord is monophonic, ie only the most recently played note is used.

Other notes

- If the PeerChord machine is muted, either directly or by soloing another machine, it may cease to function properly until it is unmuted again.
- PeerChord uses hack methods to trigger notes at times other than on the tick. However, it is possible that this could cause problems with some machines or with future versions of Buzz. Also the Record function in Buzz will not record all of PeerChord's parameter changes into the target machine's patterns.
  If you wish to disable the hack methods, please ensure:
  - Delay Mode is set to Off
  - Arpeggio Step Length represents a whole number of ticks.
  - MIDI snap to tick is set to 1 (on)
  - Note Cut is set to Infinite.

Contact

If you have comments or suggestions, or if you find any bugs please email me.
This machine is DONATIONWARE, so if you like it, send me something cool (eg CDs, hardware, money etc). Email for details of how to get stuff to me.
Also visit my website (y'know, if you're bored).

Thanks to everyone who alpha tested this machine for me and suggested features/pointed out bugs - particularly RaÇl Reales (who gave me the original idea), Cameron Bonde (Vectrex), Juri Puumala, K. M. Krebs, Jeph Wacheski, Ronny Pries, and whoever else I forgot.

BTDSys PeerCtrl 'Basic' v1.5

This machine is virtually identical to the non-basic version of PeerCtrl, apart from the lack of "slave" parameters (some users find these "extra" parameters counter-intuitive). So click here to read the docs.

EOF/legal

BTDSys PeerCtrl v1.5

What is it?

PeerCtrl is a general-purpose control machine - that is, it controls the parameters of other Buzz machines. The following are just some of its potential uses:

- Control several machines from one machine
- Access parameters which would not normally fit onto the pattern editor screen
- Add inertia to any parameter on any machine
- Tie several parameters to one slider
- Assign non-linear mappings to parameters
- Use endless rotary encoders, such as those on Doepfer's Pocket Dial, to control Buzz
- Control other machines' attributes from patterns
- Use the plugin interface to add new ways of controlling parameters, such as X-Y pads or mixer-style GUIs

Installation

Unzip into your Buzz\Gear\Generators\ folder. Ensure everything in the PeerCtrl directory within the zip file, ends up in Buzz\Gear\Generators\PeerCtrl\ . Add to index.txt if you want, or download an up-to-date index.txt if you're lazy.

Quick-start guide

- Add PeerCtrl to your song, along with some other machine
- You don't need to connect anything to PeerCtrl in the machine view
- Right click PeerCtrl and select Assignment Settings
- In this dialog, choose the machine and parameter you want to control
- Click Update, then OK
- Use PeerCtrl's parameters to control the other machine

Version history

- **v1.0:** First public release.
- **v1.1:** Fixed crashing on song load bug, added control rate attribute.
- **v1.5:** A *lot* of new stuff.

Parameters

Global parameters

- **Inertia** - Sets glide time on the parameters.

Track parameters

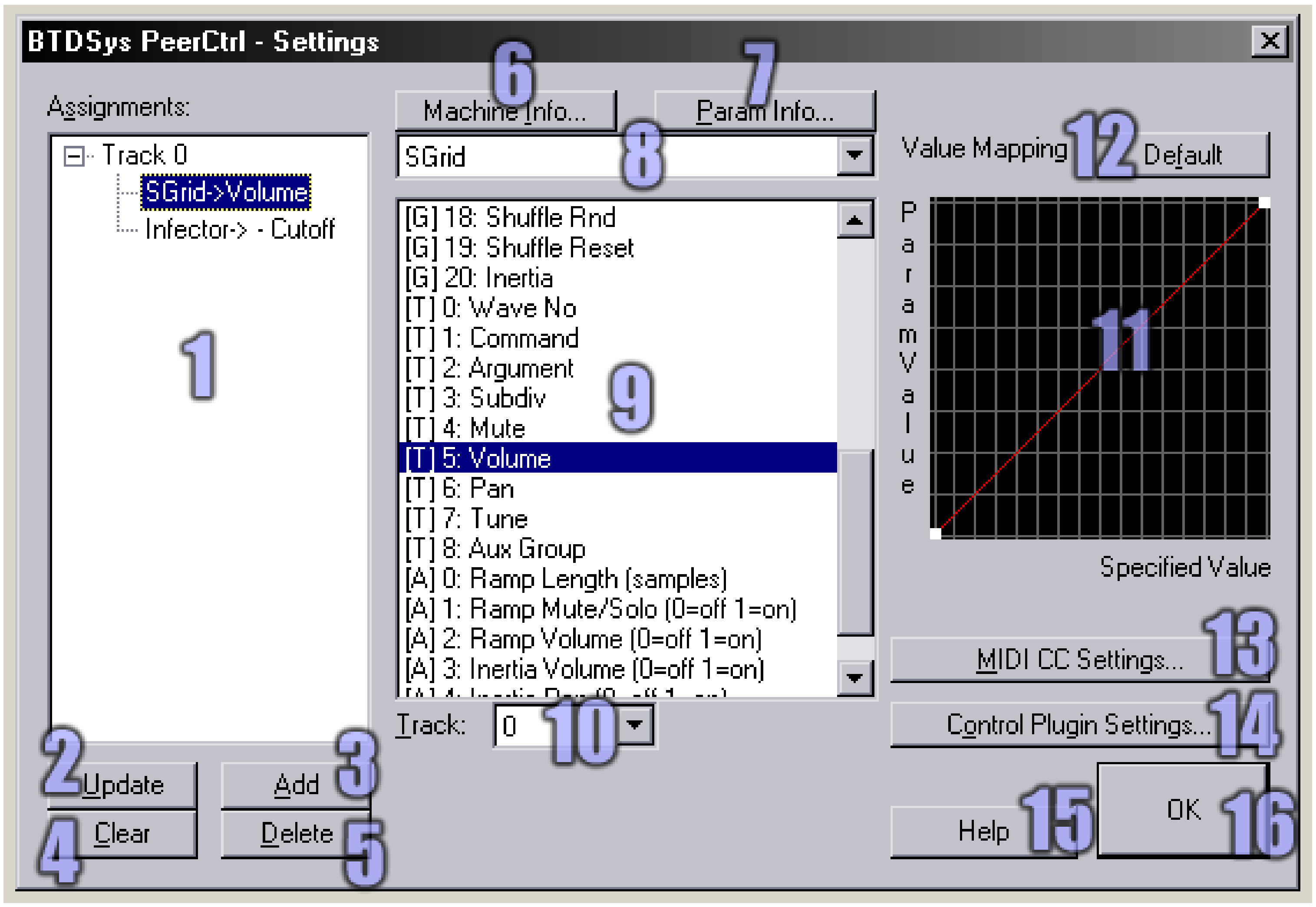
- **Slaved** (not in the 'Basic' version) - If this is switched on, this track is controlled by the slider of the previous track, rather than by its own.
- **Value** - The value to use for the controlled parameter. This is affected by the defined mapping.

Attributes

- **MIDI Inc/Dec Amount** - this sets the sensitivity, when MIDI increment/decrement messages are used.
- **Ctrl Rate** - sets how frequently PeerCtrl performs control changes between ticks. When this attribute has a value *n*, control changes will occur every 256×*n* samples (as well as on every tick). The exception is a value of zero, which causes control changes to occur only on the tick.
  In general, smaller values result in smoother inertia & MIDI control, but place a heavier load on the CPU.

Dialogs

Assignment Settings



Track 0
SGrid->Volume
Infector-> - Cutoff

SGrid

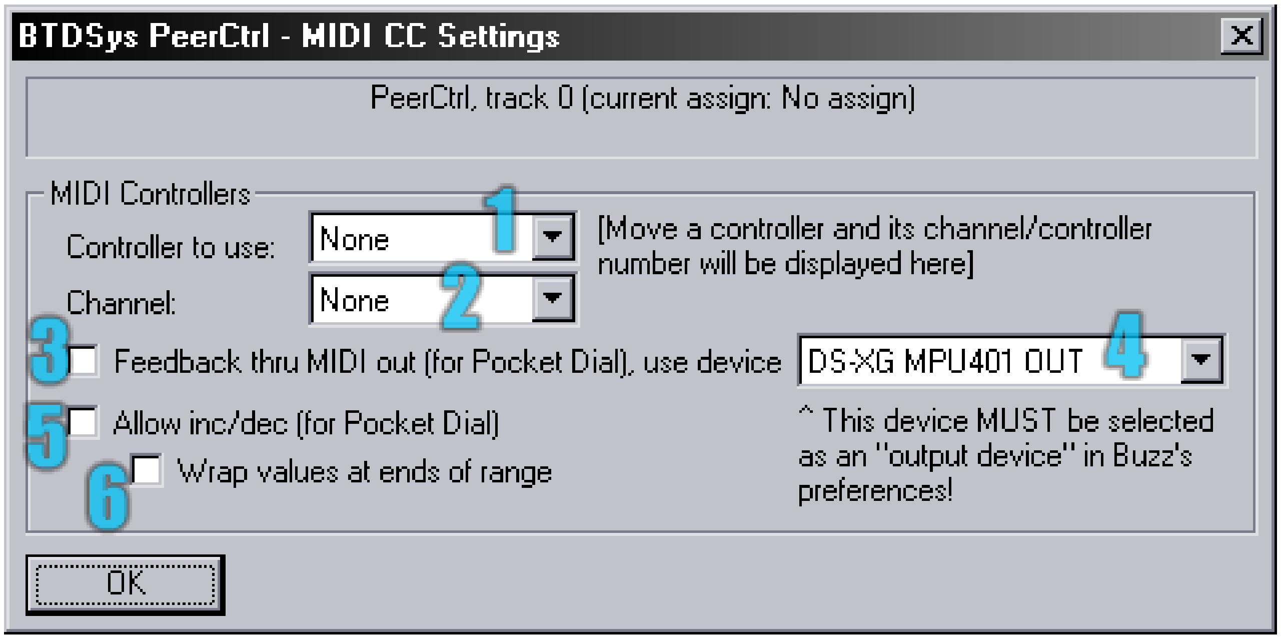
[G] 18: Shuffle Rnd
[G] 19: Shuffle Reset
[G] 20: Inertia
[T] 0: Wave No
[T] 1: Command
[T] 2: Argument
[T] 3: Subdiv
[T] 4: Mute
[T] 5: Volume
[T] 6: Pan
[T] 7: Tune
[T] 8: Aux Group
[A] 0: Ramp Length (samples)
[A] 1: Ramp Mute/Solo (0=off 1=on)
[A] 2: Ramp Volume (0=off 1=on)
[A] 3: Inertia Volume (0=off 1=on)

Track: 0

1. **Assignments list** - Lists all currently assigned parameters on all tracks.
2. **Update** - Save settings by replacing whatever is currently selected in the assignments list.
3. **Add** - Save settings by adding a new assignment entry for the current track.
4. **Clear** - Drop the currently selected assignment, leaving an empty slot.
5. **Delete** - Drop the currently selected assignment, and remove it from the list for the selected track.
6. **Machine info** - Display some very interesting stats on the selected machine.
7. **Param info** - Display equally interesting stats on the selected parameter.
8. **Machine dropdown** - Use this to select the machine you want to control.
9. **Parameter list** - Use this to select the parameter you want to control.
10. **Track dropdown** - Use this to select the track you want to control on the selected machine.
11. **Mapping** - Use this to create a non-linear relationship between the parameter value chosen in PeerCtrl, and the resultant value on the target machine. This works in exactly the same way as the envelope editor in Buzz's wavetable view.
12. **Default mapping** - Restore the default mapping (like that shown in the screenshot).
13. **MIDI CC Settings** - Bring up the MIDI settings dialog for the current track.

14. **Control Plugin Settings** - Bring up the plugin axis settings dialog.
15. **Help** - Bring up this fine document.
16. **OK** - Give it a click once you're done.

MIDI CC Settings


BTDSys PeerCtrl - MIDI CC Settings

PeerCtrl, track 0 (current assign: No assign)

MIDI Controllers

Controller to use: None

Channel: None

[Move a controller and its channel/controller number will be displayed here]

Feedback thru MIDI out (for Pocket Dial), use device: DS-XG MPU401 OUT

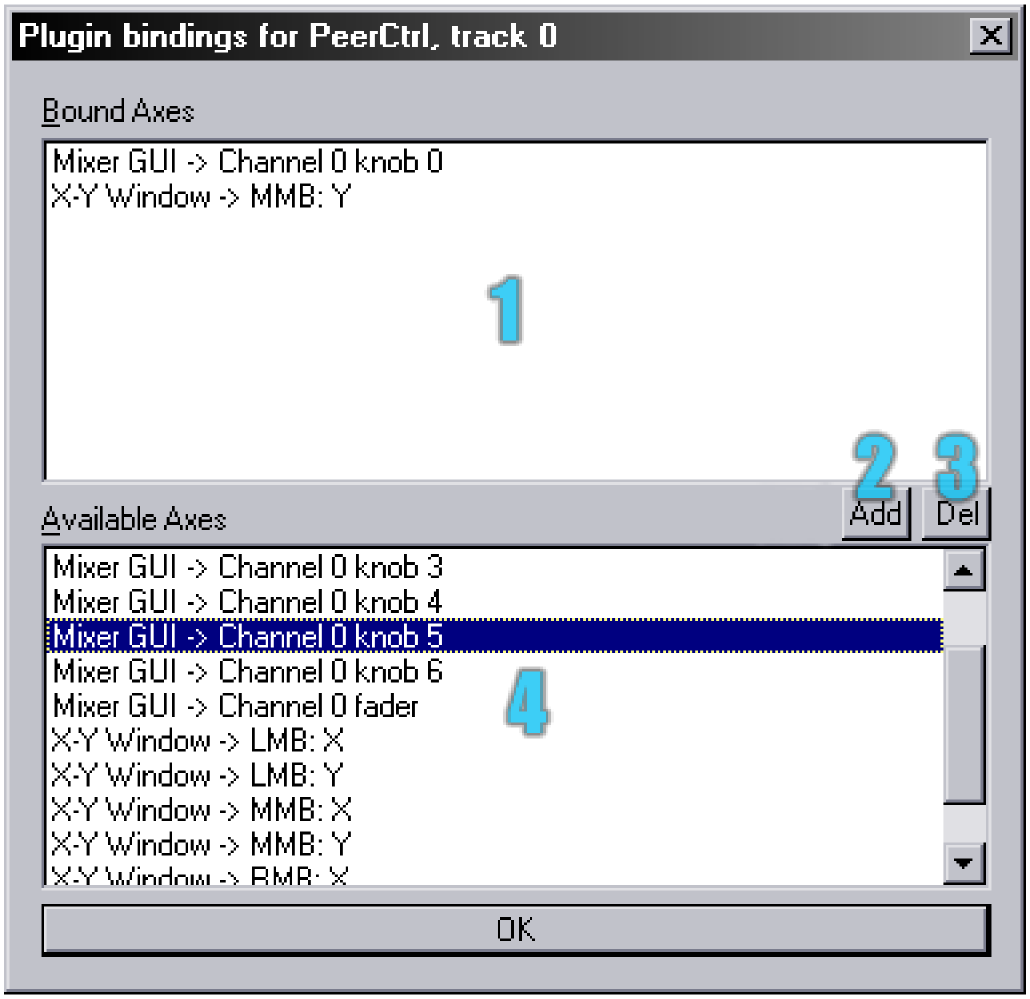
Allow inc/dec (for Pocket Dial)

Wrap values at ends of range

^ This device MUST be selected as an "output device" in Buzz's preferences!

1. **Controller** - Select the MIDI CC number to use. Note that the number of the last moved controller is displayed to the right of this box.
2. **Channel** - The MIDI channel on which PeerCtrl should listen.
3. **Feedback** - Some MIDI controllers require parameter changes are fed back through the MIDI output. Use this option if that is required.
4. **Feedback device** - The device through which to feed back parameter changes. Be sure to configure it in Buzz's options too!
5. **Allow inc/dec** - Some MIDI controllers with endless rotary encoders send CCs 96 and 97 as increment and decrement messages respectively. Use this option to enable this.
6. **Wrap values** - If enabled, an increment message when the parameter is already at its maximum will cause the value to wrap around to the minimum, and similarly at the other end of the scale. If disabled, the parameter will stop at each end of the range.

Plugin Axis Settings


Plugin bindings for PeerCtrl, track 0
Bound Axes
Mixer GUI -> Channel 0 knob 0
X-Y Window -> MMB: Y
1
2
3
Available Axes
Add
Del
Mixer GUI -> Channel 0 knob 3
Mixer GUI -> Channel 0 knob 4
Mixer GUI -> Channel 0 knob 5
Mixer GUI -> Channel 0 knob 6
Mixer GUI -> Channel 0 fader
4
X-Y Window -> LMB: X
X-Y Window -> LMB: Y
X-Y Window -> MMB: X
X-Y Window -> MMB: Y
X-Y Window -> RMB: X
OK

1. **Bound Axes** - Lists the plugin axes already bound to the current track.
2. **Add** - Bind the axis selected in the Available Axes list to the current track.
3. **Del** - Un-bind the axis selected in the Bound Axes list.
4. **Available Axes** - Lists all availalbe axes on all active plugins.

Plugins

PeerCtrl plugins implement features such as new graphical interfaces, or hardware/software interfaces. Plugins take the form of .dll files in your Buzz\Gear\Generators\PeerCtrl\ folder. Programmers can create their own plugins (using Visual C++) -- for more info on that, click here.

Thanks

Thanks to: adrian jonsson, andreas tilliander, cameron bonde, cyanphase, elektrojaenis, erthad, frank potulski, jeph wacheski, juri puumala, km krebs, lee du-caine, magmavander, matt jackson, midi nerd, moon.god, mute, mva, omnihil, oskari, ps, raul reales, ronny pries, thorsten keller, tic-tac shut up, wizenwet, everyone else who gave feedback or encouragement. And thanks to you for actually reading the docs for once. Nice one.

EOF/legal

BTDSys PeerState

What is it?

PeerState allows "snapshots" of the parameter states of machines to be captured and restored easily. You can switch between up to 128 stored parameter values for several machines at once, with only a couple of clicks, one entry in the pattern editor, or one press of a MIDI key.

Installation

Unzip into your Buzz\Gear\Generators\ folder. Add to index.txt if you want, or download an up-to-date index.txt if you're lazy.

Quick-start guide

- Add PeerState to your song, along with some other machines
- You don't need to connect anything to PeerState in the machine view
- Right click PeerState, and select Machines/Parameters
- In this dialog, place checks next to the machines and parameters you want to control
- Click OK
- Double click PeerState, and play around with the resulting dialog

Version history

- **v1.1:** Added snapshot manager window. Fixed crash when loading a song where PeerState controlled another Peer machine. Fixed occasional strange behaviour of inertia. Fixed side-effects (storing/restoring snapshots) when opening dialogs. Added facility for overwriting a snapshot from the 'Simple' dialog without first restoring it. Added "What to open on double click" attribute.
- **v1.0:** First public release.

Parameters

- **Snapshot** - Choose the current snapshot number.
- **Inertia** - Sets the glide time when changing snapshots.

Attributes

- **MIDI channel** - Sets which MIDI channel PeerState will respond to.
- **MIDI note for taking snapshots** - A press of the selected note will cause the current snapshot to be overwritten with current parameter values.
- **Inertia update freq** - Sets the resolution with which inertia takes place. Lower values give smoother glides, but result in higher CPU load.

- **What to open on double click** - Select what will happen upon double clicking PeerState in the machine view:
  0: PeerState's parameter window
  1: The 'Simple' GUI
  2: The 'A/B' GUI

MIDI

The MIDI support in PeerState works as follows:

- When a MIDI note is pressed, the corresponding numbered snapshot is restored and set as current. For example, a press on "Middle C" will call up snapshot number 60.
- In addition, a note may be chosen (in the attributes) to act instead as a "store snapshot" key.

A table of MIDI note numbers is available by clicking here.

Dialogs

There are several dialogs in PeerState, most of which are fairly self-explanatory. But I'll explain them anyway.

Machines/parameters dialog

Here you can choose which parameters you want to include in snapshots. Note that the checkboxes next to machine names indicate at-a-glance whether any of that parameters of that machine are selected—a black tick indicates all parameters are selected, a grey tick indicates some, and no tick indicates none.

'Simple' GUI

This is the dialog brought up by a double-click on the machine. You can use the combo box to call up snapshots, and the big button to store current parameter states. An asterisk next to the snapshot in the combo box indicates there is a snapshot currently stored in that slot.

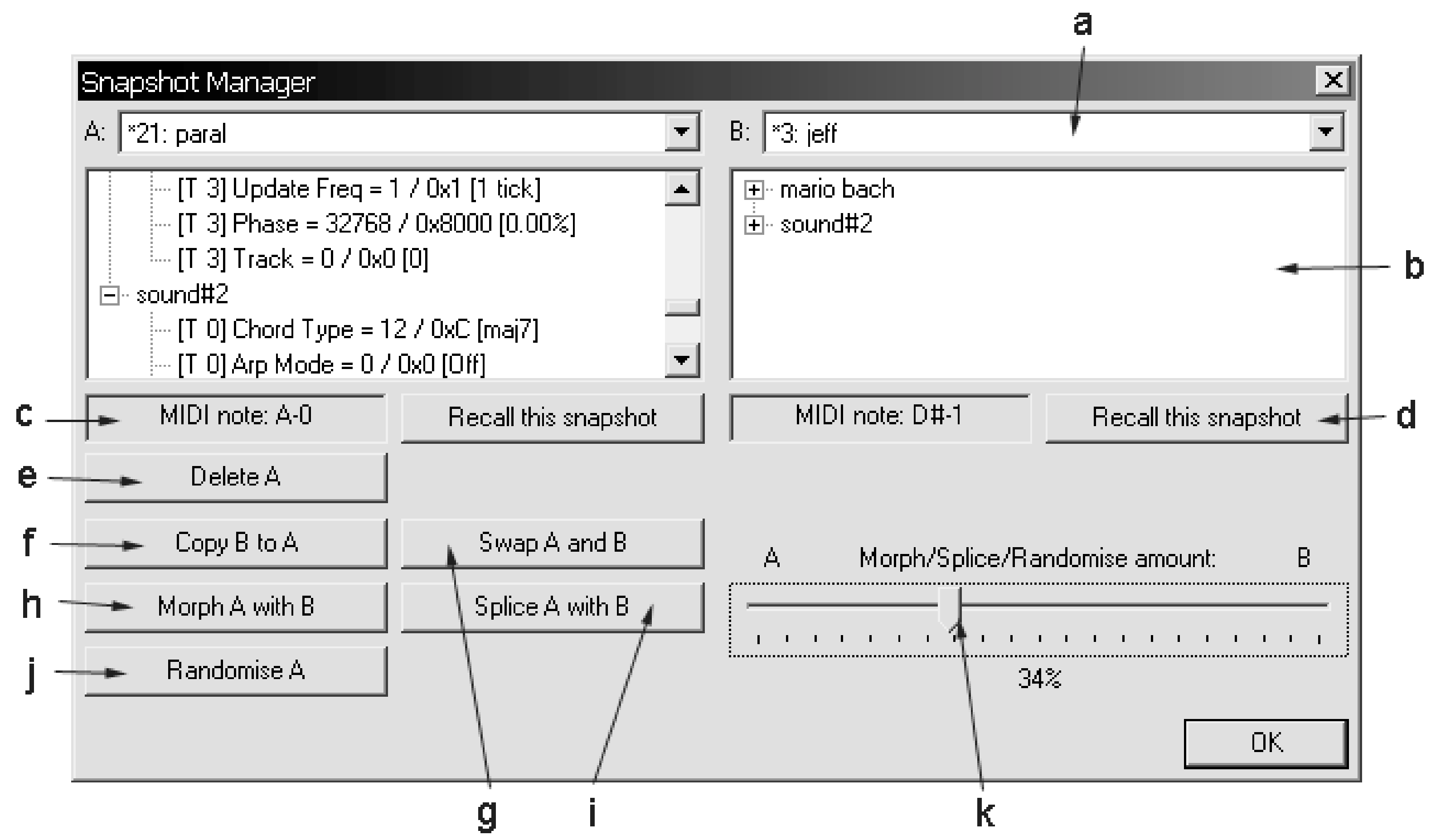
You can use the checkbox in this dialog to prevent a snapshot from being restored when you select it from the combo box. This is useful for overwriting a snapshot without having to restore it and reset all your parameter values first.

'A/B' GUI

This is similar to the 'Simple' GUI, but more suited for quickly comparing and tweaking several snapshots. The big 'Take snapshot' button is replaced with two arrow buttons for selecting snapshots. And whenever you choose a new snapshot number, the old one is automatically overwritten with current parameter values before the new one is restored.

Snapshot manager

This dialog, logically enough, allows you to manage your snapshots. Management tasks include copying/ swapping/deleting, as well as more complex randomisation and genetic functions.



a. You can choose two snapshots (A on the left, and B on the right), and perform editing tasks between them.
b. The various parameter values stored in each snapshot are listed here.
c. Displays the MIDI note which will corresponds to this snapshot. This cannot be changed.
d. Call up the specified snapshot. This works in precisely the same way as the "Snapshot" machine parameter, or the combo boxes in the 'Simple' and 'A/B' GUIs.
e. Deletes snapshot A (ie clears all parameter values from it), leaving snapshot B unchanged.
f. Overwrites snapshot A with the parameter values from snapshot B, leaving snapshot B unchanged.
g. Exchanges all parameter values between snapshots A and B.
h. Does a crossfade between the two snapshots, overwriting snapshot A with the results. The amount slider (k) controls the weighting between the two snapshots: if it is at the far left there is no effect, and if it is at the far right the effect is identical to button f.
i. Performs a randomised "genetic splice" between the two snapshots, overwriting snapshot A with the results. For each parameter in the snapshot, either the value from snapshot A or the value from snapshot B is used. The amount slider (k) controls the probability of choosing parameters from one snapshot or the other: if it is at the far left there is no effect, and if it is at the far right the effect is identical to button f.

j. Applies a random perturbation to the values in snapshot A. Each parameter value is changed by a random amount at most k% of the parameter's range, where k% is the value of slider k. If the slider is at the far left there is no effect, if it is at the far right the effect is similar to the "Random" button in Buzz.
k. Controls the amounts or weightings for buttons h, i and j.

Thanks

Thanks to Ronny Pries for the initial idea. Thanks to Polac for inspiration for the snapshot manager (and for his marvellous VST adaptors). Also thanks to tinga, noolout, mute, thOke, XionD, wizenwet, ideolog, and everyone else who contributed their support and opinions.

EOF/legal

BTDSys Pulsar & PulsarFX

Installation

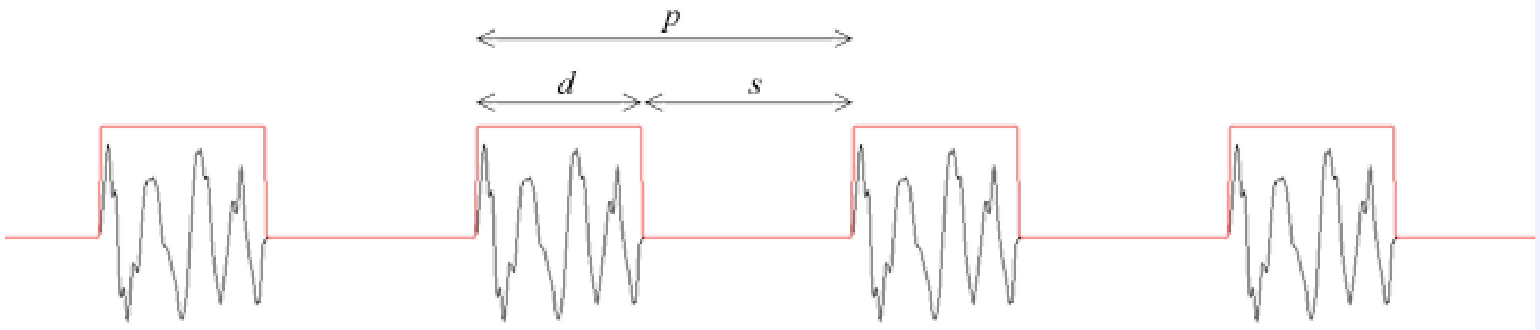
BTDSys Pulsar.dll goes into your **Gear\Generators** folder, BTDSys PulsarFX.dll goes into **Gear\Effects**.

Introduction

This machine implements **Pulsar Synthesis**, which is a variant of granular synthesis named after the spinning neutron stars which emit periodic signals. Pulsar synthesis was originally developed by Curtis Roads and is described and discussed more fully in Microsound.

One of the distinguishing features of pulsar synthesis is its ability to generate both rhythmic and tonal sounds, depending on the pulsar emission frequency used.

Basics of Pulsar Synthesis



Pulsar synthesis generates a train of pulsars, each of which consists of a pulsaret of duration *d* followed by a silent interval of length *s*. The length $p = d + s$ of the pulsar corresponds to the pulsar emission frequency. The pulsaret consists of an arbitrary waveform (shown in black in the diagram above) under an arbitrarily shaped pulsaret envelope (shown in red).

Generally, frequency *p* dictates the fundamental frequency of the generated tone (or the period of the generated rhythm), while frequency *d*, the pulsaret waveform and envelope, and other parameters such as masking contribute to defining the timbre or spectral qualities.

PulsarFX, the effect version of Pulsar, alters this model by continually sampling the pulsaret waveform from its input, rather than using a static waveform from the wavetable.

Parameters

Global Parameters

Inertia: Controls the glide time for certain parameters.

Freq Mode: Toggles between Linear and Logarithmic frequency sliders (*P Freq* and *D Freq*). Logarithmic allows greater resolution at lower frequencies at the expense of less at higher frequencies. Linear has more or less the opposite effect.

Min P Freq: Sets minimum *P Freq* value.

Min D Freq: Sets minimum *D Freq* value.

Oversample: Use oversampling to reduce signal aliasing. Default value is 2x, and can be increased up to 32x. Be fore-warned that this is the most computationally intensive part of Pulsar, and high settings will require a lot of CPU power, especially when used in combination with high *overlap* settings.

DC Removal: Removes any DC offset using a specialized highpass filter to insure maximum dynamic range.

Track Parameters

Synchronise: *(pattern editor only)* Synchronizes! Resets the pulsar generation phase counter, LFO phase counters, burst/rest counters etc. Useful for synchronizing several tracks, or synchronizing the machine to the beat for rhythmic work.

Overlap: Specifies the number of simultanteously overlapping pulsarets. Defaults to 1 (off), which allows only one pulsaret to be generated at a time. Increasing this value will allow multiple pulsarets to overlap. In technical terms: the Duty cycle frequency (*D Freq*) of an individual pulsaret will always complete, regardless if it extends past the pulsar emission frequency (*P Freq*) which determines the rate at which new pulsars are generated.

This setting can have a significant effect on CPU usage, especially in combination with *oversampling*. Furthermore, depending on the source material, high overlap values may cause clipping -- it is suggested that you reduce the volume setting before experimenting.

P Freq: Controls the pulsar emission frequency in hertz. This is the rate that new pulsars are generated. See pulsar diagram for detailed explaination.

P Period: Controls the pulsar emission frequency in ticks. This is the rate that new pulsars are generated. See pulsar diagram for detailed explaination.

P Note: *(pattern editor only)* Sets the note to use for P Freq. *P Param* must be set to *Note* for note data to have any effect.

P Param: Selects the parameter that controls the *P Freq*. Choices are:

- **Freq**: Uses *P Freq* parameter (hertz)
- **Period**: Uses *P Period* parameters (ticks)

- **Note**: Uses a *Note value* (as entered into pattern editor's *P Note* value)
- **Slave**: Slaves to previous track's value. No effect if set on track 0.

D Freq: Controls the Duty cycle frequency in hertz. This determines the length of the pulsaret, the audible part of the pulsar. See diagram for detailed explaination.

Volume: Volume of output.

Pan: Pan position of output.

Burst & **Rest**: "A type of masking that produces a regular pattern of pulsarets that are interrupted at regular intervals. The pattern is denotated by the *b*:*r* ratio, where *b* is the burst length in pulsaret periods and *r* is the rest lenght in pulsaret periods." (Roads, 149)
A *b*:*r* ratio of 4:3 would produce a sequence of four pulsarets and 3 silent periods: 1111000111100011110000....

Stoch Mask: Stochastic masking introduces a probability into the otherwise orderly generation of pulsarets. The value of the stochastic mask represents the chance of the current pulsaret being triggered.

Random Pan: The amount of randomness applied to panning position on a per pulsaret basis. Range is from 0% (no random panning) to 100% (full random panning) Note that this setting is relatitve to the absolute track pan position as set via the *Pan* parameter.

Envelope/Waveform Section

On/Off *(Generator only)*: Turns Pulsar's sound generation on or off.

Freeze *(Effect only)*: When set to ON, freezes the input buffer and repeats whatever is in it until turned off.

Wave *(Generator only)*: Select the wavetable number that Pulsar will use as source material in sound genertion. Values **201** and above use internal waveforms.

Start: Specify the start position (in percent) of the waveform selected in the *Wave* parameter.

End: Specify the end position (in percent) of the waveform selected in the *Wave* parameter.

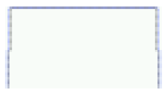
Envelope: Select the envelope to be applied to pulsarets. Default is a rectangular envelope. Other selections are Gausian, Linear Decay, Exponential Decay, Linear Attack, Exponential Attack, or a user supplied envelope in the wavetable. See the pulsaret envelope entry in the terminology section for images.

The Gaussian, Exponential Decay, Exponential Attack, and Wavetable envelopes can be altered using the *Env Param* setting.

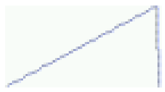
Env Wave: Wavetable slot to use for user defined pulsaret envelope.

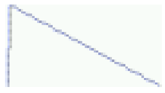
Env Param: Alters the parameters of the pulsaret envelopes. This has different effects depending on the selected envelope:

- **Rectangle** (Note: Not altered by Env Param)

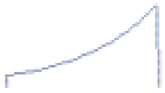

Rectangular Envelope

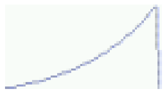
- **Linear Attack / Linear Decay** (Note: Not altered by Env Param)

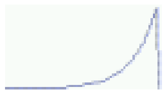

Linear Attack

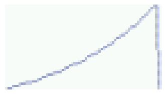

Linear Decay

- **Exp Attack / Exp Decay**: Alters the steepness of the exponential curve.

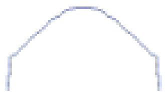

Default Exponential Attack (note discontinuity at onset)

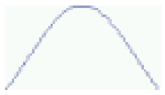

Default Exponential Attack (/w normalization)

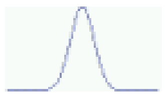

Exponential Attack with env param of 0.15 (/w normalization)

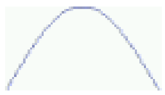

Exponential Attack with env param of 0.7 (/w normalization)

- **Gaussian**: Alters the variance ("steepness" or "width") of the Gaussian function.


Default Gausian (note discontinuities)


Default Gausian (/w normalization)


Gaussian with env param of 0.1 (/w normalization)


Gaussian with env param of 0.7 (/w normalization)

- **Wavetable**: Acts as a DC offset parameter, allowing the center of the envelope to be shifted. A value of 1 centers the wave, while a value of 0 shifts the envelope to maximum depth. Will not have any effect if *Env Norm* is enabled.

Env Norm: Normalizes the pulsaret envelope to the range of 0 to 1. This suppresses discontinuities which sometimes occur at the ends of the Gaussian and Exponential envelopes.

LFO Section (Low Frequency Oscillator)

LFO1 Period: The period (speed), in ticks, of the LFO waveform.

LFO1 Shape: The waveform the LFO generates. All the usual settings: Sine, Square, Triangle, Ramp Up, Ramp Down, Lin Wander, Cos Wander (like linear wander, except with cosine segments, making it smoother), Random, and Wavetable.

LFO1 Wave: Selects the wavetable slot to use for the LFO waveform (provied Wavetable is selected in *LFO Shape*).

LFO1 Target: The parameter that LFO will modulate.

LFO1 Track: Selects which track number of Pulsar that the LFO will modulate. It can select from one track or all.

LFO1 Amount: Controls the depth of the LFO upon the targeted parameter.

LFO2: The settings for LFO2 are nearly identical to LFO1, except that it can target the period and amount of LFO1.

Separators

To aid navigation of the large number of parameters in Pulsar and PulsarFX, separators are used. These show as columns of zeroes in the pattern editor, and as graphical separators in the parameter window (if Cyanphase Overloader 1.5 is installed).

Note that if you do not like the graphical track separator (the purple or red "btdsys" banner), it may be enabled or disabled using the included registry patches, or by manually changing the registry value **HKEY_CURRENT_USER\Software\BTDSys\Pulsar_Buzz\ShowGraphicalSeparator** to 0 (disabled) or 1 (enabled).

Attributes

Off on stop: Sets whether Pulsar stops generating a signal when the stop button is pressed. 0 = No, 1 = Yes.

Note controlled on/off *(Generator only)*: When the P Param is set to Note, this controls whether note on and off events also control the On/Off parameter. 0 = No, 1 = Yes.

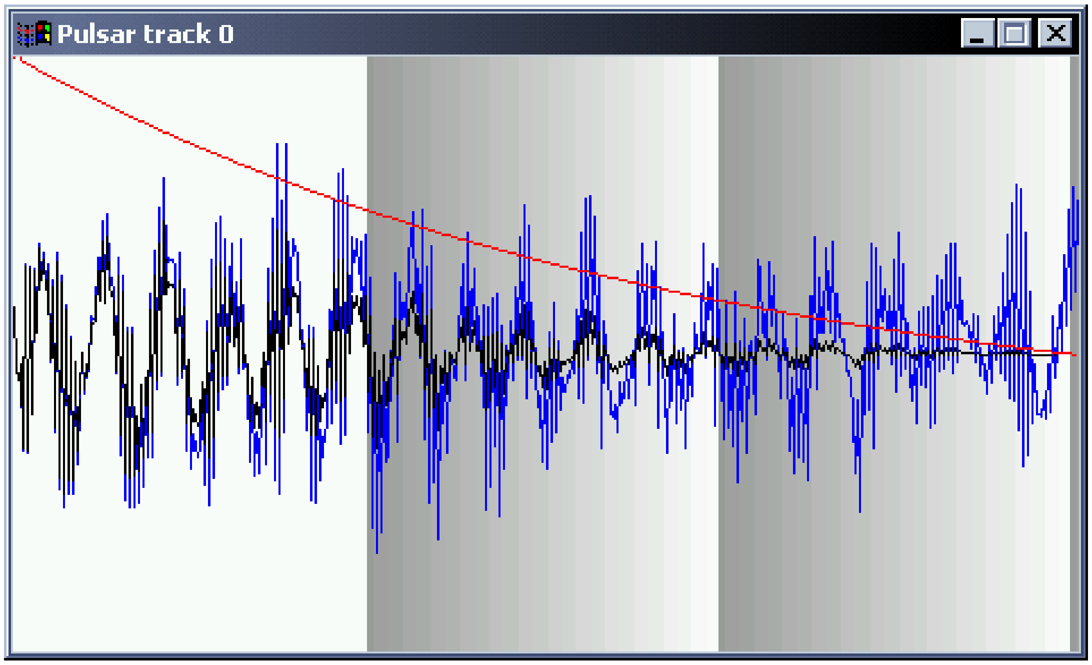
Note controlled freeze *(Effect only)*: As above, but concerning the Freeze parameter.

Oversample quality: Sets the order of the filter used for oversampling. Higher values remove more aliasing artifacts from the signal, at the expense of increased CPU usage.

LFO Amount midpoint: Sets the mapping used for the LFO Amount parameters. Values less than 500 give finer control for lower values at the expense of precision for high values, values greater than 500 have the opposite effect (finer control for high values), and exactly 500 gives a linear scale (same precision for the entire range). Technically, what you are setting here is the value corresponding to the parameter slider's midpoint (in steps of 0.1%), and an appropriate curve is fit around this to give an overall range of 0% to 100%

Input buffer size *(Effect only)*: Sets the size (in samples) of the recording buffer used for sampling the input waves. This should be no less than the longest P period you wish to use, but beware of excessive memory usage resulting from very large values.

Visualization



The visualization window of Pulsar/PulsarFX is an invaluable tool to understand the complexities of the patch you're working on. It can be access by right clicking on the Pulsar machine and selecting "Visualization" followed by the track you want to see. You can have any number of tracks open at one time.

It displays all the information you need to know about the pulsar stream:

- Envelope (in red)
- Waveform (in blue)
- Waveform after application of envelope (in black)
- Overlap (gray-white gradient)

These are the default colours, and can be changed by right clicking on the Pulsar machine and selecting "Customize Visualization Colours". The colours you select here are stored in the registry (under the key **HKEY_CURRENT_USER\Software\BTDSys\Pulsar_Buzz**\), and so are common to all instances of Pulsar and PulsarFX.

Terminology

Aliasing: A type of distortion which occurs in digital recording or synthesis when the signal contains frequency components which exceed the Nyquist frequency (*sampling rate/2*). Because the signal cannot be correctly represented, it becomes **aliased** at a lower frequency, causing 'phantom' frequencies to arise and interfering with the original frequency content. External Links: Aliasing, Nyquist Theorem

Duty cycle frequency: The frequency (*D Freq*) which controls the length of pulsarets. Low values may allow pulsarets to overlap. The maximum density of overlaping can be ontrolled via the overlap parameter.

Masking: The omission (silencing) of pulsarets. Masking can be done in an ordered pattern (burst/rest masking), or randomly/probabilistically (stochastic masking).

Oversampling: A technique used to reduce the severity of aliasing artifacts in digital audio. Oversampling is a complex subject, but the basic theory behind it is to temporarily increase the sampling rate of digitized audio. By increasing the sampling rate, it becomes significantly easier to filter out frequencies that will cross the Nyquist frequencies and cause aliasing.

Pulsar: The term to describe the overall output of pulsar synthesis, taking into account all of the variables. *Also*: A neutron star which rotates at an extremely rapid rate, emitting radiation at a regular interval. These pulses of radio emissions were first detected in 1967, and originally thought to be the product of an extraterrestrial civilization. External Link: Pulsars

Pulsar emission frequency: The frequency (*P Freq*) at which pulsarets are emitted by the generator.

Pulsaret: A discrete pulse of arbitrary sound, emitted at the pulsar emission frequency (*P Freq*) and altered in length by the duty frequency (*D Freq*).

Pulsaret envelope: The amplitude envelope applied to the pulsaret. Different envelopes will affect the spectrum and timbre of the sound, with steeper/discontinuous envelopes generally creating noisier spectra.

Tips and Tricks

Pulsar synthesis can generate intricate, organic and complex sounds. One fascinating trick is running a Pulsar Generator through PulsarFX with slightly different P or D rates.

By assigning LFO2 to modulate LFO1's rate, it is possible to push LFO1 into the audio frequencies. Assign it to Volume for AM effects, P freq or D freq for FM-like effects, or experiment with other combinations.

Although Pulsar does not have any filters (apart from those used for oversampling and DC removal), altering the D rate or the envelope shape can sometimes give a "filter sweep" like sound.

For rhythm work, try using a slow P rate and feeding the output to an LdC Trigger or LnB PeerTrigger. Don't forget to trigger the Synchronise parameter from a pattern to ensure the rhythm stays on the beat.

Be sure to examine the included presets for more ideas. Pulsar synthesis is largely uncharted territory, so don't be afraid to experiment!

Credits

Documentation, waves and suggestions by K.M. Krebs.

Thanks to Ulf Schreiber, Colin Brash, Kwook, and everyone else who helped out with beta testing.

References

Microsound. Curtis Roads, MIT Press, 2001.

BTDSys SampleGrid

Installation

Unzip all files apart from *BTDSys SampleGrid Out.dll* to your **Gear\Generators** folder. *BTDSys SampleGrid Out.dll* goes in **Gear\Effects**.

Most up-to-date index.txts should already contain entries for SampleGrid. If not, copy & paste the following into an appropriate place:

```
/BTDSys SampleGrid
 BTDSys SampleGrid - Byte x 4, Byte x 4
 BTDSys SampleGrid - Byte x 8, Byte x 8
 BTDSys SampleGrid - Byte x 16, Byte x 16
 BTDSys SampleGrid - Switch x 8, Switch x 8
 BTDSys SampleGrid - Switch x 16, Switch x 16
 BTDSys SampleGrid - Switch x 32, Switch x 32
 1,-----
 BTDSys SampleGrid Out, Out
 /..
```

Overview

SampleGrid is a sample player designed especially for rhythm programming. Its main feature is a "grid" of triggers kept together in the pattern view. It also features support for multiple outputs, and MIDI functionality.

Multiple Versions

SampleGrid comes in several "sizes", with different numbers and types of triggers. The versions are functionally identical, apart from the number of triggers available and whether those triggers allow only 1/0 (switch) or numeric (volume/pan/etc) values (byte). This document covers all versions.

Parameters

Global parameters

- **First Wave** - sets the first wave from the Buzz wavetable to be used. By default, SampleGrid uses a number of consecutive waves, so if the first wave is 1, the first trigger uses sample 1, the second 2, third 3, etc.
- **Dividers** *Only in Byte versions* - Just a visual device to help you distinguish the triggers from the rest of the parameters.
- **Triggers** - Actually trigger the drum sounds.
  - *In Switch versions* - a value of 1 triggers the sample on, and 0 off.
  - *In Byte versions* - the sound is triggered, with some characteristic set by the entered value. What this value sets is determined by the Trigger Type parameter. The exception is a value 00, which always offs the current sample.

  Note that the number and type of triggers depends on the version being used. Choose an appropriate version for your needs in each case.
- **Trigger Type** *Only in Byte versions* - Determines what the numeric trigger values will set.
  - *Velocity* - the volume of the sound
  - *Pan* - position in the stereo field
  - *Tune* - pitch of the sound
  - *Probability* - likelihood of the sound being triggered, where FE is certain and 01 is almost certainly not
  - *Command argument* - value is used as argument for the command on this line (if any)

  Note that any values here are unaffected by inertia.
- **Solo Track** - for editing purposes, allows you to solo one track (ie mute the rest). If set to "off" then no tracks are muted.
- **Global Volume/Pan/Tune** - set the overall volume, stereo pan and pitch of all sounds.
- **Shuffle Size** - controls by how much shuffle mode delays beats
- **Shuffle Step** - shuffle mode will delay every nth tick, where n is the value set here.
- **Shuffle Randomness** - how much the delay time for shuffled beats deviates from the specified size.
- **Shuffle Reset** - a 1 in this column will reset the shuffle counter.
- **Inertia** - controls glide time for certain parameters

Track parameters

- **Wave Number** - Allows you to specify the wave number to use. Or if it is set to Auto, the wave will be chosen according to the track number and the First Wave specified for the machine.
- **Command/Argument** - Provide a number of "special effects" for triggering.

Number	Command name	What it does	Argument meaning
00	None	Nothing	None

01	Delay	Delay this hit by some amount	Number of subticks to delay
02	Retrigger	Keep triggering every so often for some length of time	1st digit = interval in subticks 2nd digit = number of ticks to keep retriggering
03	Offset	Play the sample forwards from some point other than the start	Offset from start of sample, where 00=start and FF=end
04	Reverse+offset	Play the sample backwards from some point other than the start	Offset from start of sample, where 00=start and FF=end
05	Probability	Alter the likelihood of playing a hit	Probability of playing sample (where 00="definitely not", 80="50/50 chance", FF="Definitely")
06	Cut	Cut the sample off after some length of time	Number of subticks to cut after
07	Delay+Cut	Like a combination of commands 1 and 6	1st digit = No of subticks to delay 2nd digit = No of subticks to cut after (counted from when the sound is triggered)

- **Tick Subdivide** - how many evenly sized slices to divide each tick into. These slices are used by some commands to specify times.
- **Mute** - allows you to turn the sound of this track on or off.
- **Volume/Pan/Tune** - set the volume, stereo pan and pitch of this sound.
- **Auxbus Group** - which auxbus channels to send this sound over, or use "none" to send through the machine's output.

Attributes

- **Ramp Length** - sets the ramping (anticlick) time in samples. Ramping takes effect when samples are triggered or stopped, and can also affect mute/solo and volume.
- **Ramp Mute/Solo** - sets whether ramping affects muting and soloing of tracks.
- **Ramp Volume** - sets whether ramping affects volume slider changes.
- **Inertia Volume/Pan/Tune** - sets whether the corresponding parameters will be affected by inertia.

AuxBus

This feaure is included for backwards compatibility only. For new songs, use the following:

Multiple Outs

SampleGrid allows use of separate "output" machines to give up to 9 stereo outputs:

- Add a "BTDSys SampleGrid Out" machine (effect) to your song.
- Connect a SampleGrid to the out machine.
- Set the group number on one of SGrid's tracks, and the group parameter of the Out machine, to the same value.
- The volume on the connection between SGrid and the Out machine is unimportant, but it should not be zero.

Now you can add different processing to different sounds from the same machine.

MIDI

SampleGrid supports triggering of samples via a MIDI keyboard. To set this up, right click the machine and choose "MIDI Setup".

Now press a note on your keyboard, and you may use the button to assign that note to a sample. You should also choose the MIDI channel you wish to use (the same channel is used for each sample). You can also specify whether to use the velocity from the keyboard, and whether note offs are used (ie the sample stops when you lift your finger from the key)

Also you can record using MIDI, by checking the appropriate boxes in the MIDI settings dialog. Normal rules apply for MIDI recording in Buzz machines (click Record in Buzz's toolbar, make sure you lay down blank patterns over the part you want to record).

History

v1.0 - initial release

v1.1 (not released) - increased auxbus groups from 4 to 8, fixed a bug with ramping

v1.2 - second release - new non-auxbus multiple out system, fixed some clicking problems

v1.21 - fixed a clicky/buzzy bug with the delay and shuffle functions

Thanks

Oskari, Cyanphase, Jeph Wacheski, Ronny Pries, Frank Potulski, mva, oomek, thOke, moon. god, mute, usr, anyone else I should have thanked.

EOF/legal

BTDSys SeqStep

What is it?

SeqStep saves the Step setting in the Sequence Editor along with your file. It will save the step size when you save the song, and reload it when you next load the song. Also it allows you to program automatic changes of the step value and automatic scrolling of the Sequence Editor view. This is useful if you make a habit of working with strange time signatures or high TPB values.

Installation

Unzip into your Buzz\Gear\Generators\ folder. Add to index.txt if you want, or download an up-to-date index.txt if you're lazy.

Quick-start guide

- Add SeqStep to your song.
- Now when you save your song, the sequence editor step will be automatically saved.
- Add patterns as normal to program the auto step changing and scrolling features.
- For the auto scroll etc to actually have any effect, **you need to switch on your computer keyboard's Scroll Lock**. Normally you won't be able to edit your song while auto scrolling is active.

Version history

- **v1.0:** First public release.

Parameters

Note that both parameters have no effect whatsoever unless your keyboard's Scroll Lock is active.

- **Sequence Editor Step** - Changes the step value, this is useful for using several different time signatures in the same song. Note that the value you enter must be present in the sequence editor's step list for this to work. If the desired value is not present, you can use my Custom Seq Steps utility to add it.
- **Scroll Here** - Enter a 1 in this column, and when it is played the sequence editor will

scroll so that it is at the top of the window. This is useful in conjunction with the Sequence Editor Step parameter, as you can ensure the sequence editor view does not get 'out of step' with the actual programmed sequence, rendering your patterns invisible.

En/disable Record

Have you ever accidentally clicked the Record button instead of Play, overwriting your carefully programmed patterns with parameter tweaks or peer control events? This menu item hides the Record button from the toolbar, completely removing the risk of accidentally activating it. You can still record by pressing F7, but there's less chance of that happening accidentally.

Thanks

Thanks to noolout, thOke, XionD, and everyone else I forgot as usual.

EOF/legal

BTDSys Stutter

Installation

Put *BTDSys Stutter.dll* in your **Gear\Generators** folder.

Notes

- There is an attribute to set whether the End/Length parameter sets the absolute end point, or the loop length (relative to the start point).
- If the end point is before the start point (or length is negative), the loop plays in the opposite direction to "normal".
- Inertia A applies to Start and End/Length, Inertia B applies to Loop Time, Rate and Volume.
- Only one of Loop Time or Rate is used at once (the one chosen by the Rate Par parameter). The other is just ignored.

Contact

If you have comments or suggestions, or if you find any bugs please email me.

Hi there, MAKK is typing again here.

this one is a bugfixed version of the m3 machine. now you can use it with 4 tracks :) thanks to oskari tammelin for his hint! also thanks for the nice responses to my release. I won't release any other version of this machine now, because i start a journey throu europe (5 months) in about 2 hours...
but the source is included so perhaps anyone else can do this job ??? ;))
so have a nice time... bye!

Climox 303

'Acid Man'

Friendly not-so-small-print

Version 2.00 (c) 2000 Joost Diepenmaat - **To be enjoyed and copied freely.**
Without any warranties whatsoever, so use at your own risk.

Description

- The positive description: **Climox 303** is a Roland TB-303 emulator that **really sounds like a Roland TB-303.**
- The negative description: **Climox 303** is a monophonic bass synthesizer that **doesn't sound at all like a acoustic bass.**

About this 303 emulation

This is my second attempt at writing a TB-303 emulation for Buzz.
The first one was never release because it sucked.
This one was because it doesn't.

This emulator is meant to be fast (about 12-16 % CPu time on my P100) and convincing (which I think it is now). But I'd like to point out that the 303 emulation is not perfect yet, especially the filter design, which is now just a rip-off of the standard cookbook formulae LP-filter. Any constructive suggestions or good filter-design algorithms will be appreciated, please send them to my email-address below, or post them on BuzzTrack.

You may find that some features of this buzz machine work **a little different** from other (virtual) analog machines - especially the slide function - that is because I tried to keep close to the original, in both synth and sequencer abilities.

On the other hand, **if you've worked with an original TB-303**, you may be worried that the sequencer in this machine is too easy to program: you can see all the notes in a pattern at once, you can enter slides, rests, notes and accents all at the same time and it does not have decaying push-buttons that sometimes react, and then not and then twice.
For these people I have devised the followin strategy: while entering notes in the pattern-editor, close you eyes, enter the notes, then do the same in the slide and accent rows; then go over the pattern again with the cursor keys, in the mean time hitting INSERT and DELETE and random. You now have a 303 pattern just like the original. :-)

But (a little more) seriously; there are a few thing to keep in mind while programming this machine:

- First of all: the original TB-303 has only 3 octaves to play in (in this machine: C-3 - C-5) plus a tuning button that goes approximately -1 - +1 octave and a transpose function of one octave (0 - +12), which makes the 'usable' notes (with the tuning set to zero) C-2 - C-7 (6 octaves). so keep away from the upper octaves, if you don't want to give yourself away.

- Second: the slide switch keeps the note on as long as the VEG (volume envelope generator) takes it, but after that you have to switch it off in order to hear anything. This is my understanding of the workings of the original, but I could possibly be mistaken. Anyway you can alway make the volume decay longer by editing the VEG Decay attribute.
- Note off ("off") and no note ("...") are equivalent in this machine; if the slide is off there is no note playing; if the slide is on the note is kept on. This is because in the original 303 you don't have notes longer than 1/16th beat, except slid notes.
- Slide and Accent switches are kept in there respective states if they are not set (see also second point).

Commands / sliders in the Climox 303:

Command	Description	Range
Wave:	Select waveform	0 - 1 (square/ saw)
Tuning:	Tune machine (in semitones)	00 - 30 (-24 - +24)
Cutoff:	Filter cutoff frequency	00 - fe (0 - 254)
Resonance:	Filter resonance	00 - fe (0 - 254)
Env Mod:	Filter enviroment fodulation	00 - fe (0 - 254)
Decay:	Environment decay	00 - fe (0 - 254)
Accent:	Accent amount	00 - fe (0 - 254)
Volume:	Main volume	00 - fe (0 - 254)
Note:	Play (or slide to) note	c0 - b9
Slide Switch:	Set slide	0 - 1 (off/on)
Accent Switch:	Set accent	0 - 1 (off/on)

Some people who've been very helpfull:

- I would like to thank Robin Whittle for his excellent texts on the inner workings of the little silver box; I couldn't have done it without his information (I know, I tried ;-)).

Other buzz machines by Climox

At till now I've only released 1 machine; the drumbreaker Climox Breaker.

Upcoming projects

- Coming soon: the **Climox Pulse M** machine - a 6-voice virtual analog ala the Roland SH-101 / JX-3p; 4 filter types, pulse modulation, ENV follow on pulse and cutoff freq, keyboard tracking, slide, hold etc. etc.
  Will be available as soon as the last bugs are eliminated.

Joost 'Climox' Diepenmaat
joost@netlinq.nl
http://thor.prohosting.com/~climox/

Climox 313.dll

Climox 303 v2.10.1 FULL RELEASE!

Before submitting any bug-reports, check for a newer version at my homepage or on buzztrack!

Joost 'Climox' Diepenmaat.
joost@netlinq.nl

http://thor.prohosting.com/~climox/   <-- Check here for new versions/bug fixes.

Changes 2.10 - 2.10.1

Added skin. (Too cool not to release it just for the new look!)

Changes 2.00 - 2.10

Accent and slide now default to OFF. (if you want to keep slide/accent, you now
have to set them on each tick)
Added fine-tuning (-12 - +12 semitones)
Renamed tune to transpose and changed range to -12 - +12 semitones
New waveforms (squarewave is now calculated from saw, leaking included!)
Changed accent-sweep routine (with added sweep_max attribute, this is still not completely
what I'm looking for, but it's getting better)
Polished filter code a bit (less ticks and other noise)
Panic command now resets filter.
Tuned the volume down, so it doesn't clip when using accent (use saturator if you
want clipping).
Added compensation on CUT OFF for ENV MOD.
New about window. ;-)

Note:
This version of Climox 303 is NOT compatible with the v2.00 version.

Todo:
Code new filter (3-pole diode ladder).
Rewrite help-file.

Climox Breaker

'chops your breaks the eazy way'

Friendly not-so-small-print

Version 1.0 (c) 2000 Joost Diepenmaat - **To be enjoyed and copied freely.**
Without any warranties and with some bugs - see **Things to do** section at the bottom.

Description

The **Climox Breaker** can be defined in several ways:

1. The **Climox Breaker** is a buzz generator/drum machine for chopping up long breakbeats.
2. The **Climox Breaker** is the third Buzz Generator by Joost Diepenmaat, but the first to be released because of some finetuning that needs to be done on the other two.
3. The **Climox Breaker** is a quick hack. (But it does what it's supposed to do, so who cares?).

Commands / sliders in the Climox Breaker:

Command	Description	Range
Volume:	Sets main volume	00 - fe (0-99%).
Wave Number:	Selects wave from wavetable	01 - c8.
Particle Size:	Sets particle size in number of samples	01 - ffff.
Tuning:	Adjust playback speed/Tune sample	00 - fffe (50-199%).
Track Volume:	Sets track volume	01 - ff (0-100%).
Play Length:	Number of particles to play	00 (all)/ 01 - fe.
Particle Number:	Trigger play from particle number x	00 - fd. (fe = stop).

Getting started

Here is a quick way to set the particle size to a useful value:

1. Select **Climox Breaker** from the Generator menu.
2. Load your favourite drumbreak into the wavetable.
3. Open the pattern-editor for the **Climox Breaker**, and set the wave number to your sample and the particle to 00.
   First line will look like this:

   ```
   ..  01 .... .... .. .. 00
   ```

4. Loop the pattern, and tune the wave till it plays in the correct speed.
5. Now enter the pattern-editor again and set the particle number to 08 at line number 08.

6. Adjust the particle-size till the break plays correctly again. (Use the pattern editor for best precision.)
7. That's it! All the particles are excactly 1/16th bar now. You can start chopping away.

Things to do

- Remove the bug that crashes buzz when loading waves over the one that is playing in Breaker. |-{ The best way to avoid this is to stop play before loading new waves.
- Add volume ramping to reduce clicks.
- Add some extra tricks like reverse play, some tracker commands etc...

Upcoming projects

- Coming soon: the **Climox Pulse M** machine - a 6-voice virtual analog ala the Roland SH-101 / JX-3p; 4 filter types, pulse modulation, ENV follow on pulse and cutoff freq, keyboard tracking, slide, hold etc. etc.
  Will be available as soon as the last bugs are eliminated.

Joost 'Climox' Diepenmaat
joost@netlinq.nl

cmx HD Player v1.00
(c) 2001 Joost 'climox' Diepenmaat

To be shared and enjoyed free of charge.

If anyone wants to sell this library (for instance
as part of a collection of buzz machines), please
send an email to me asking for permission; chances
are, you'll get it, but I still would like to know
beforehand.

The HD Player plays WAV files directly from the drive,
so you can play very long samples in your songs. This
enables you to use the builtin wave-recorder and play
the recorded part back in your tracks.
It also means that you can now use buzz as a sample
processor for very long recordings, or add external
recordings to your buzz tracks. This is the first
version, so there might be some bugs and very likely
the funtionality will be enhanced, but it is ready for
general consumption.

updates and more machines can be found at:
http://thor.prohosting.com/~climox/

email:
joost@diepenmaat.nl

CyanPhase Bass Pluck is beta, please try not to use it in songs yet, thanks.

CyanPhase BassTrk

This is a preview release of BassTrk,
a pretty good sampler for buzz with some
decent features:

* A decent sampler with a variety of bassline
  options
* includes a multimode filter engine (28 filters),
  distortion, and predistortion
* Resampling is done in CSI interpolation
* The multimode filters include 12dB, 24dB, 36dB,
  48dB, 60dB, 72dB filtering for lowpass, highpass,
  bandpass and bandreject. A moog lowpass filter,
  arguru style 'phat filter', and two vocal wah
  filters is also included.
* This sampler has 1 bassline style VCF.
* Full MIDI implementation (play and record,
  including -24 to +24 semitone transposition) in Buzz

More to come

CyanPhase (blakee@rovoscape.com)

CyanPhase DTMF-1

Version 1.2 - Fixed clicking with analog A-R envelope

My most useful generator yet!..

NEW_ATOM really wanted this type of generator so... anyway
this is a DTMF generator, which means that buzz can sound
like a telephone. So you don't need cooledit to make them
for you :P. the first column allows you to enter the
numbers

This machine comes also with the source code, it demonstrates
how easy it is to calculate and use fast sine coefficients
to generate nice sine waves without a using a buffer or
wavetable.

CYANPHASE SLIDE FLUTE

Type	Generator - Physical Modeling
Author	Edward L. Blake (CyanPhase), Automaton has helped a bit with some stuff
Email	blakee@rovoscape.com
Description	An real-time implementation of a physical model slide flute based on Perry Cook's slide flute. Various settings are available to tweek the flute sound.

PARAMETERS

Breath	Controls how much "breath" (noise) there is. Values are from 0.000 to 0.140.
Flow1: and Flow2:	Controls the ADSR envelopes for the flute, there are two of them, Flow1 controls mostly the breath.
Pressure	Amount of initial air pressure.
Feedback1	Sets the amount of particle feedback that goes through the embouchure part of instrument, higher values sound very resonant however also distorted, lower values sound more "air-ish".
Feedback2	Same as with Feedback1, however this is the amount of particle feedback going through the bore of the flute.
AttackAmt	A bit like a volume but not really, sets the initial "volume" going through the instrument, this is actually part of the ADSR envelopes
SustainAmt	Sets the sustained "volume" going through the instrument, this is actually part of the ADSR envelopes
RealVolume	Sets the output sound gain.

ATTRIBUTES

MIDI Channel	Sets the MIDI channel (0 to 16, 0 being known as omni) to listen to, behaves similarly to all the other Buzz machines that support MIDI.

MIDI Velocity	Sets how MIDI Velocity is treated with the flute: *0 - Don't use MIDI velocity for anything* *1 - Flute pressure (light)* *2 - Flute pressure (medium)* *3 - Flute pressure (high)* *4 - Real volume (light)* *5 - Real volume (medium)* *6 - Real volume (hard)* *7 - Attack Amount (light)* *8 - Attack Amount (medium)* *9 - Attack Amount (hard)* *10 - Sustain Amount (light)* *11 - Sustain Amount (medium)* *12 - Sustain Amount (hard)* *13 - N\A* *14 - N\A* *15 - N\A* Note that the harder settings mean it requires less force on the keyboard to get higher values.
MIDI Transpose (-24..24 smt)	Sets how many semitones to transpose the MIDI note from the normal tuning on the flute. It goes from 24 semitones down to 24 semitones up.
MIDI Recording Enabled	Enables or Disables the use of the record button to record notes into the currently playing pattern
Older Tuning	Use the older slightly off tuning, either for "aesthetic backwards compatibility" or some other reasons.
Vibrato Amount	The amount of Vibrato to apply, smaller numbers gives less vibrato.
Body Ratio	Sets the ratio (normally 49:100 ~ 1:2) of the flute body's delay lines.

STUFF

E-Mail:	blakee@rovoscape.com.
Web site:	cyanphase.tsx.org.

CYANPHASE VIBRASYNTH 1

Type	Generator - Wacky Synth
Author	Edward L. Blake (CyanPhase)
Email	blakee@rovoscape.com
Description	A nasty synth i guess, has a few vibratypes, and creates some rather interesting sounds (check the various included demo songs like the one by canc3r). vibrasynth was made to be sorta abnormal, an experimental synth basically, with some additive, subtractive, FM and ringmod properties.

PARAMETERS

VibraType	The type of "Vibra" you want, the oscillators used means the more detail (and the more CPU used), these do not include the additional custom oscillators: "Quick Vibra" (5 osc) "Vibra I" (5 osc) "Dark Vibra II" (6 osc) "Bright Vibra II" (6 osc) "Phat Vibra X" (6 osc) "Vibra III A" (8 osc) "Vibra III B" (7 osc) "Vibra SK 1" (6 osc, not harmonic) "Vibra SK 2" (6 osc, not harmonic)
WaveType1	Sets the first waveform slot, different waveforms will sound different.
WaveType2	Sets the second waveform slot, different waveforms will sound different.
Tuning	Your not stuck to "standard" tuning, also has a setting to tune to Arguelle's Guru Series and another being "Alt-Tune" (just a random tuning). After the first 3 settings you can set tuning manually.
VCA-Attack	Amount of time that the attack (starting) part of the envelope takes. shorter times makes the sound more stabby.
VCA-Decay	The amount of time after the attack part of the sound envelope
VCA-Release	The amount of time the sound fades out after note-off
AntiClick	Just a basic delta restrictor parameter, it doesnt do much however.
BitSaturate1	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate2	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate3	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher

BitSaturate4	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate5	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate6	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate7	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
BitSaturate8	creates harsh noise, the more the 8 bit saturates are lined up, the less harsher
OSCM1:Vol	Frequency Modulator 1 Oscilator volume.
OSCM1:RFreq	Frequency Modulator 1 Oscilator frequency
OSCM1:Type	Type of frequency modulation that is done on the oscilators (and which ones)
OSCM2:Vol	Frequency Modulator 2 Oscilator volume.
OSCM2:RFreq	Frequency Modulator 2 Oscilator frequency.
OSCM2:Type	Type of frequency modulation that is done on the oscilators (and which ones)
AdditionalOSCs	Additional oscilators that are used as well
OSCs:WaveType	The waveform that the extra oscilators will use
OSC1:Vol	Oscilator 1 Volume
OSC2:Vol	Oscilator 2 Volume
OSC1:RFreq	Oscilator 1 Relative frequency
OSC2:RFreq	Oscilator 2 Relative frequency
Filter:Type	Type of Filter used: *No Filter -* *LP - 12dB/oct Lowpass filter, this filter takes out the high end* *Moog LP - 24dB/oct Moog self-oscilating filter* *HP - 12dB/oct Highpass filter, this filter takes out the low end* *BP - Bandpass filter, this filter creates a peak with both low and high ends taken out* *BR - Bandreject filter, this filter creates a hole in the frequency spectrum*
Filter:CutOff	The frequency where the filter's resonance peak is.
Filter:Resonance	The amount of resonance that is made at the cutoff, high values give more sound.
Inertia	The Inertia of the OSCM part of the synth
SustainVol	Sustained volume of the sound

FltNote:Filter	*No Filter Note LP - 12dB/oct LP filter where the note frequency is the cutoff* *Note LP Moog - 24dB/oct LP filter where the note frequency is the cutoff* *Note HP - 12dB/oct HP filter where the note frequency is the cutoff* *Note BP - BP filter where the note frequency is the cutoff* *Note BR - BR filter where the note frequency is the cutoff* *Notex2 LP - 12dB/oct LP filter where the note frequency x 2 is the cutoff* *Notex2 LP Moog - 24dB/oct LP filter where the note frequency x 2 is the cutoff* *Notex2 HP - 12dB/oct HP filter where the note frequency x 2 is the cutoff* *Notex2 BP - BP filter where the note frequency x 2 is the cutoff* *Notex2 BR - BR filter where the note frequency x 2 is the cutoff*
FltNote:Res	Amount of resonance at the note cutoff point

ATTRIBUTES

MIDI Channel	Sets the MIDI channel (0 to 16, 0 being known as omni) to listen to, behaves similarly to all the other Buzz machines that support MIDI.
MIDI Velocity	Sets how MIDI Velocity is treated with the synth: *0 - Don't use MIDI velocity for anything* *1 - Attack Amount (light)* *2 - Attack Amount (medium)* *3 - Attack Amount (high)* *4 - Decay Amount (light)* *5 - Decay Amount (medium)* *6 - Decay Amount (hard)* *7 - N\A* *8 - N\A* *9 - N\A* *10 - N\A* Note that the harder settings mean it requires less force on the keyboard to get higher values.
MIDI Transpose (-24..24 smt)	Sets how many semitones to transpose the MIDI note from the normal tuning on the synth. It goes from 24 semitones down to 24 semitones up.
MIDI Recording Enabled	Enables or Disables the use of the record button to record notes into the currently playing pattern

STUFF

E-Mail: blakee@rovoscape.com.

Thanks\shoutz	canc3r for the demo song all the ppl who gave suggestions MVA Oskari djlaser Mute #buzz HymaX disasteradio (new atom) rymix 7900 Larsby all the buzz devs discharge waKax Haloform arguru Zephod FSM Whitenoise Apo Kooper levin theEXIT djboo Nool _phi (_khlEr3l) pinkj Socrates setzer clarion\clarotica Buzz-Talk list Wizkid\Coala Uksi Severed Optical Nerve (SevOPT) Vectrex Cervus pyroscott Nathan Snider Bas with the offline buzz archives The many others i forgot

Dave's Delta version 1.0

What is it ?

This is another synth type generator, like the Geonik's Bass.

This machine gives a bit more control, however, with the ability to set the envelopes and pitch glide (cool for some things). You can also detune notes, so you can create some simple synth effects by playing the same note with several slightly detuned versions. An example song is provided. Finally, you can also morph between two waveforms, so you can get smooth transition between a sine and a saw wave for example. I dunno if this will be useful, but I guess it's up to you. If you find any bugs, email me.

Thanks to Rout, since I stole his html for this page. ;)

This is DONATIONWARE. If you want to be kind, you can send me any amount of money (or anything else you think I'd like to have) to the following address. Thank you.

David Wallin
122 Heather Valley Rd.
Holland, Pa
18966
USA

The DS1

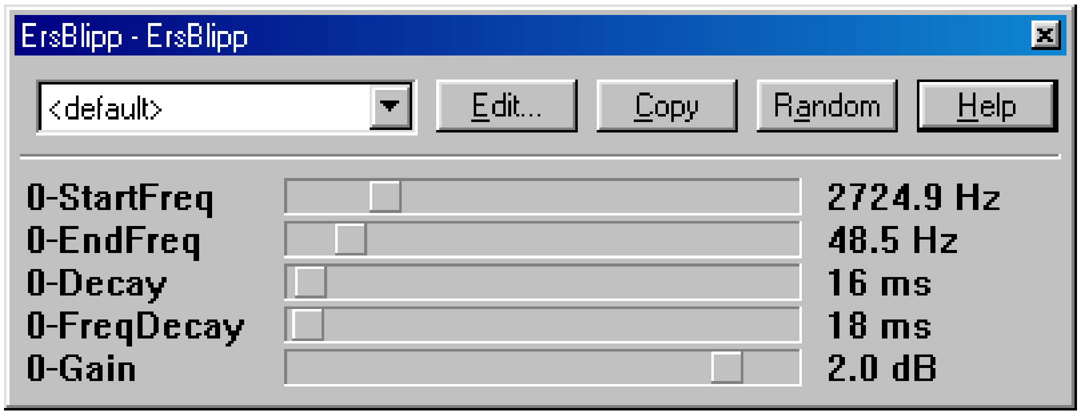
This sampler plays wavetable samples. I made it to trigger
my loops (e.g. made with repeater) in a simple way with my midi keyboard.
Simple means that there is only one slider for the octave (which triggers
the samples) and one for the wavetable-range.
The mode slider controls how the samples are played: once, looped or
holded which means the first note-on starts looping, the second stops it.
The hold mode works only in midi mode.

Volume	Global volume
Midi Channel	midi channel which is used to trigger the samples
Midi Octave	octave which is used to trigger the samples
Wavetable	selects the wavetable range (which samples should be triggerd)
Tempomode	defined how tempo-syncronisation is made (in the version only pitch works)
Mode	defines how (once, looped, looped until midi-note again) the sample is played
Length	sample is syncronized to length rows

Have fun !!!

Holger Zwar
maekflai@aol.com

ErsBlipp (ErsDrums 1.2)

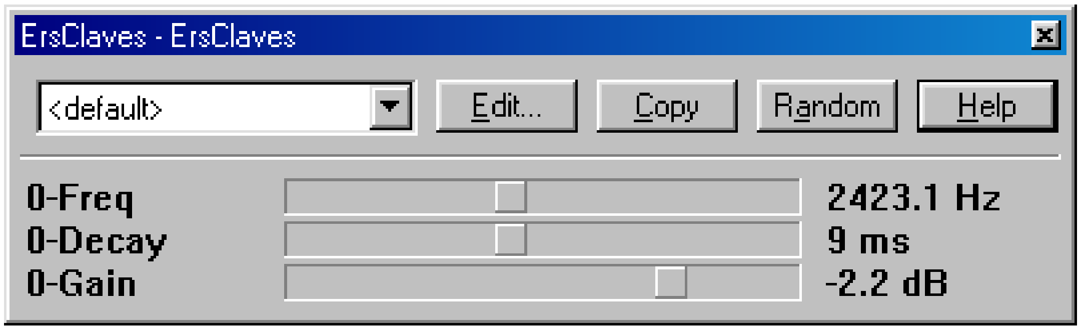


0-StartFreq 2724.9 Hz
0-EndFreq 48.5 Hz
0-Decay 16 ms
0-FreqDecay 18 ms
0-Gain 2.0 dB

The blipp sound consists of a frequency sweep.

- **StartFreq** (*Hz*) is the initial frequency of the sweep.
  Sweep Start Frequency (0000=200 Hz, 4000=6300.1 Hz, 8000=10000 Hz)
- **EndFreq** (*Hz*) is the target frequency of the sweep.
  Sweep End Frequency (0000=20 Hz, 4000=132 Hz, 8000=200 Hz)
- **Decay** (*ms*) is the decay time of the amplitude of the sweep.
  Sweep Amplitude Decay (0000=10 ms, 4000=155 ms, 8000=300 ms)
- **FreqDecay** (*ms*) is the decay time of the frequency of the sweep.
  Sweep Frequency Decay (0000=10 ms, 4000=255 ms, 8000=500 ms)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsClaves (ErsDrums 1.2)

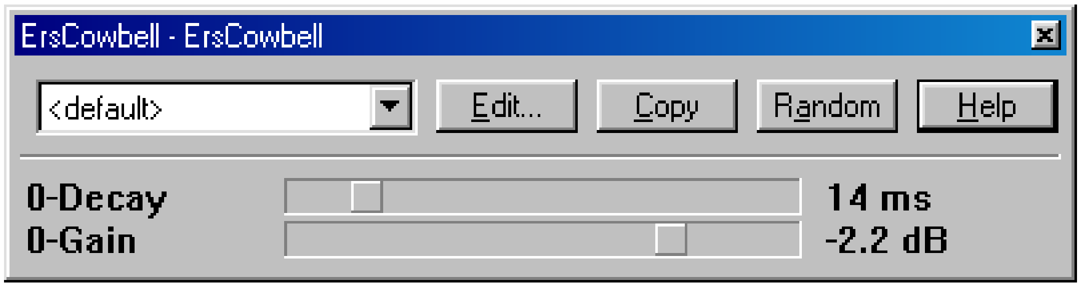


0-Freq 2423.1 Hz
0-Decay 9 ms
0-Gain -2.2 dB

The claves sound consists of a decaying sinewave.

- **Freq** (*Hz*) is the frequency of the thump sound.
  Frequency (0000=400 Hz, 4000=2640.9 Hz, 8000=4000 Hz)
- **Decay** (*ms*) is the decay time of the amplitude of the sound.
  Decay (0000=1 ms, 4000=11 ms, 8000=20 ms)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsCowbell (ErsDrums 1.2)

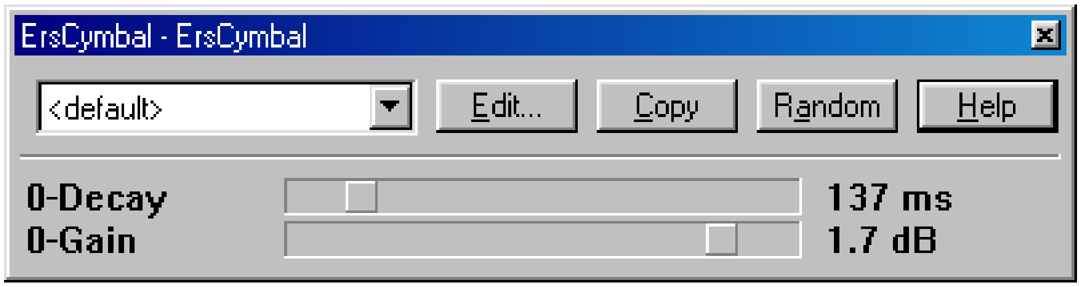


0-Decay 14 ms
0-Gain -2.2 dB

The cowbell sound consists of a decaying metallic sound.

- **Decay** (*ms*) is the decay time of the amplitude of the metallic sound.
  Decay (0000=10 ms, 4000=25 ms, 8000=40 ms)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsCymbal (ErsDrums 1.2)



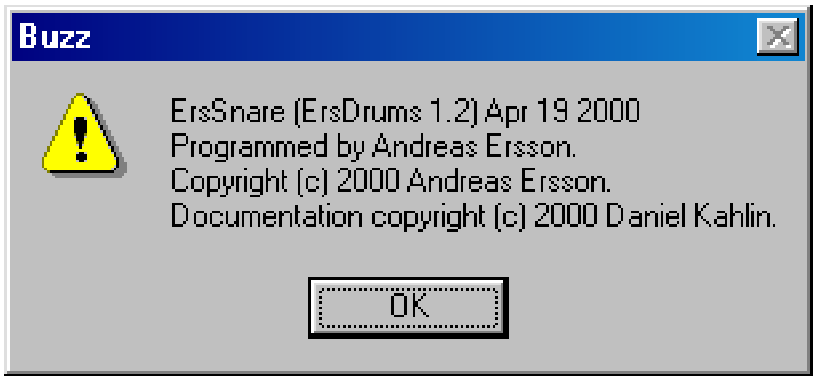
0-Decay 137 ms
0-Gain 1.7 dB

The cymbal sound consists of two decaying highpass filtered versions of a metallic sound. The highpass output with the the lowest cutoff has a shorter decay time than the one with higher cutoff.

- **Decay** (*ms*) is the decay time of the amplitude of the metallic sound.
  Decay (0000=40 ms, 4000=420 ms, 8000=800 ms)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsDrums 1.2

ErsDrums is a set of 7 buzz plugins: ErsKick, ErsSnare, ErsHihat, ErsCymbal, ErsBlipp, ErsClaves and ErsCowbell.
They are digital models of an analogue drum machine, much like the early Roland TR series.
These plugins are always released together in one archive (ErsDrums_1_2.zip for ErsDrums 1.2). The currently installed version can be determined by bringing up the about box of any of the drums.



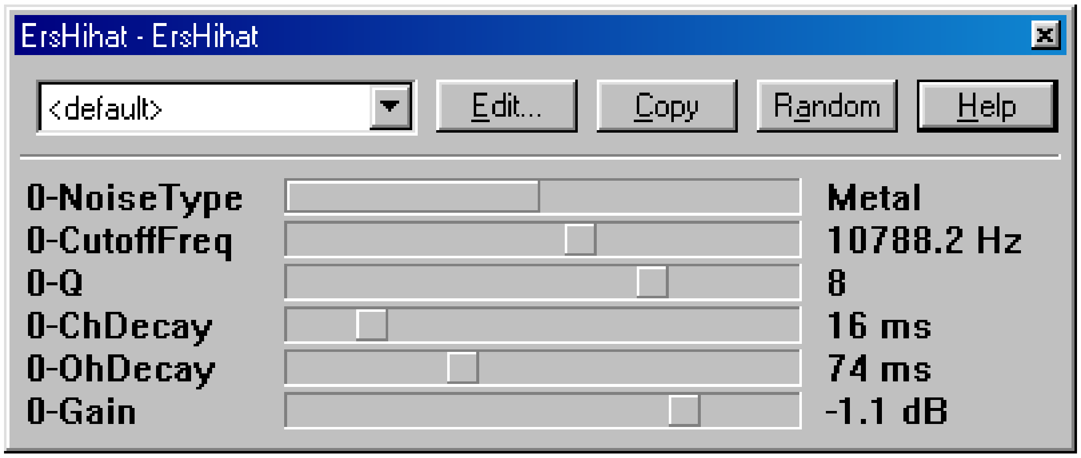
ErsSnare (ErsDrums 1.2) Apr 19 2000
Programmed by Andreas Ersson.
Copyright (c) 2000 Andreas Ersson.
Documentation copyright (c) 2000 Daniel Kahlin.

License

Disclaimer

entire computer, don't blame us!

ErsHihat (ErsDrums 1.2)

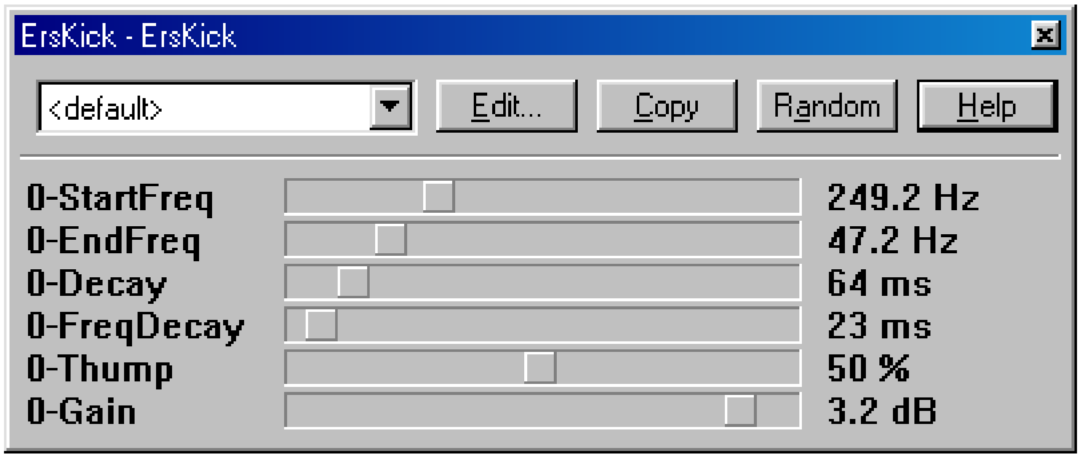


0-NoiseType	Metal
0-CutoffFreq	10788.2 Hz
0-Q	8
0-ChDecay	16 ms
0-OhDecay	74 ms
0-Gain	-1.1 dB

The hihat sound consists of a decaying metallic sound or noise burst passing through a highpass filter.

- **NoiseType** (*metal/noise*) selects if the hihat sound should be based on metallic sound or noise.
  Noise Type (0=metal, 1=noise)
- **Cutoff** (*Hz*) is the cutoff frequency of the highpass filter.
  Cutoff Frequency (0000=1000 Hz, 4000=9714 Hz, 8000=15000 Hz)
- **Q** (*no unit*) is the amount of resonance.
  Q (0000=0.7, 4000=5, 8000=10)
- **ChDecay** (*ms*) is the decay time of the amplitude of the noise burst for the closed hihat sound.
  Ch Decay (0000=10 ms, 4000=30 ms, 8000=50 ms)
- **OhDecay** (*ms*) is the decay time of the amplitude of the noise burst for the open hihat sound.
  Oh Decay (0000=10 ms, 4000=105 ms, 8000=200 ms)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsKick (ErsDrums 1.2)

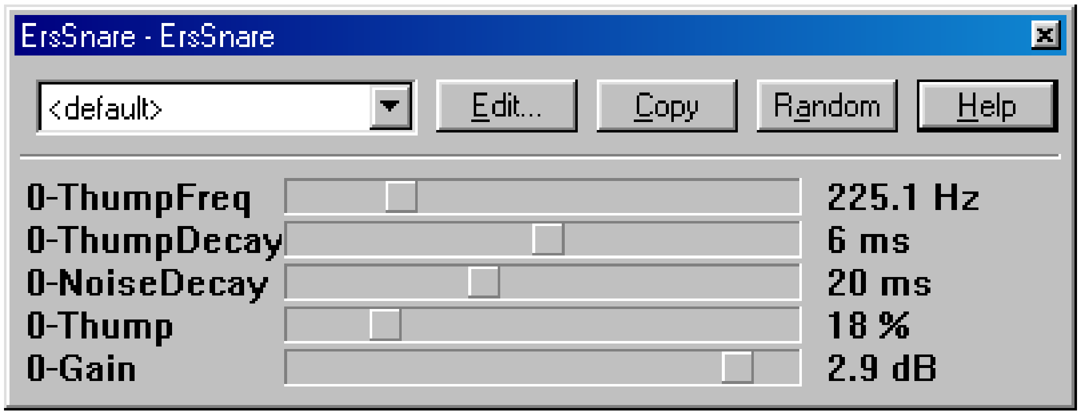


```
0-StartFreq    249.2 Hz
0-EndFreq      47.2 Hz
0-Decay        64 ms
0-FreqDecay    23 ms
0-Thump        50 %
0-Gain         3.2 dB
```

The kick sound consists of two parts, a frequency sweep, and an initial thump.

- **StartFreq** (*Hz*) is the initial frequency of the sweep.
  Sweep Start Frequency (0000=150 Hz, 4000=305.6 Hz, 8000=400 Hz)
- **EndFreq** (*Hz*) is the target frequency of the sweep.
  Sweep End Frequency (0000=20 Hz, 4000=82.25 Hz, 8000=120 Hz)
- **Decay** (*ms*) is the decay time of the amplitude of the sweep.
  Sweep Amplitude Decay (0000=10 ms, 4000=255 ms, 8000=500 ms)
- **FreqDecay** (*ms*) is the decay time of the frequency of the sweep.
  Sweep Frequency Decay (0000=10 ms, 4000=155 ms, 8000=300 ms)
- **Thump** (*%*) is the mix between the thump sound and the sweep.
  Thump (0000=0 %, 4000=50 %, 8000=100 %)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

ErsSnare (ErsDrums 1.2)



0-ThumpFreq 225.1 Hz
0-ThumpDecay 6 ms
0-NoiseDecay 20 ms
0-Thump 18 %
0-Gain 2.9 dB

The snare sound consists of two parts, a decaying noise burst, and an initial thump.

- **ThumpFreq** (*Hz*) is the frequency of the thump sound.
  Thump Frequency (0000=150 Hz, 4000=305.6 Hz, 8000=400 Hz)
- **ThumpDecay** (*ms*) is the decay time of the amplitude of the thump sound.
  Thump Decay (0000=1 ms, 4000=6 ms, 8000=10 ms)
- **NoiseDecay** (*ms*) is the decay time of the amplitude of the noise burst.
  Noise Decay (0000=1 ms, 4000=26 ms, 8000=50 ms)
- **Thump** (*%*) is the mix between the thump sound and the noise burst.
  Thump (0000=0 %, 4000=50 %, 8000=100 %)
- **Gain** (*dB*) is the output gain.
  Gain (00=-30 dB, 6A=0 dB, 7F=6 dB)

FireSledge *RectalAnarchy*

v1.1 - Build 2001/01/09

What is RectalAnarchy ?

RectalAnarchy is a plug-in for Buzz. It's a simple polyphonic synthetizer (generator) based on human voice sounds.

It manages 5 types of voices and can reproduce 5 vowel sounds. The ring parameter system allows all possible combinations and transitions between two settings.

The LFO modulates the Vowel parameter, and an ADSR (Attack-Decay-Sustain-Release) envelope modulates the VCA (amplitude).

Installation and connections

Just copy **FireSledge RectalAnarchy.dll** in your Buzz/Gear/Generators/ directory. Restart Buzz if it was still running.

To put RectalAnarchy into your song, go in the machine view and right-click on the background. Select New > Generator > Synthesis > Other Synthesis > FireSledge RectalAnarchy. If it can't be found here, try in Generator > Unsorted.

Now you just have to connect RectalAnarchy to the Master machine.

Global parameters

Voice Glide Time	Used to smooth transitions between two Voice Type or Vowel parameter values. Big values slowly fade from old to new parameters.
Voice Type	Characterize the singing voice, from soprano down to bass.
Vowel	It is the sound emulated by the synth.
Voice LFO Depth	Amplitude of the oscillations modulating the Vowel parameter.
Voice LFO Period	Period of the LFO oscillations.
Voice LFO waveform	There are 6 possible waveforms : - Sine - Square - Triangle - Ramp up - Ramp down - Brown noise. Use it with very long periods.
Emphasis	Amount of voicing effect.
Glottal Open Time	Parameter for the voice simulation. In practice, long time enhances the bass frequencies but cancels high pitch notes.
VCA Attack Time	Duration of the ADSR attack phase.
VCA Decay Time	Duration of the ADSR decay phase. During the decay phase, VCA level is decreases down to the sustain level.
VCA Sustain Level	Level of the VCA at the begining of the sustain phase.

VCA Sustain Time	Duration of the ADSR sustain phase. The sustain phase is active as long as the note is kept pressed. VCA level decreases during this phase, but it is possible to obtain a constant level and infinite sustain by setting the value to 00.
VCA Release Time	Duration of the ADSR release phase. The enveloppe gets into release state when the note is released.

Track parameters

Velocity	This indicates with how much force the note should be played, and acts mainly on its loudness. Velocity can be changed at any time (kind of "aftertouch" message). Here, velocity uses a quadratic scale instead of a linear one, which is the case in most synthetizers.
Slide time	When a slide time is coming with a note, the previous note isn't stopped, but slides gradually to the new one. This parameter indicates the slide duration.

Change log

2001.01.09	**Version 1.1** - Fixed Note Off problem (occured when two or more tracks played at the same time). - Fixed scale and display of Slide Time parameter. - Fixed Velocity scale, which was much too sensitive.
2001.01.07	**Version 1.0** Initial release.

Miscellaneous infomation - Legal stuff

You enjoy it ? Then please visit **Ohm Force - Designing homestudio**

To contact FireSledge :

Laurent de Soras
92 avenue Albert 1er
92500 Rueil-Malmaison
France
ldesoras@club-internet.fr-or- laurent@ohmforce.com

Frequency UnKnown Buzz Machine

Name: Mr Brown

Type: Generator/Noise

Parameters: Volume, Movement, Attack, Decay, Zero Track

About: This is a Brown Noise generator, it makes noise based on a Brownian Motion algorithm. It can work either as a Brown or White noise generator depending on the settings. Volume sets erm... Movement sets the amount the 'particles' or whatever move each sample. This works partly like a volume control but also changes the tone of the noise. Zero track prevents the algorithm heading off to infinity and also fades between white (0) and Brown (1) noise. Attack and Decay work just like any other generator.

Frequency UnKnown/Aybiss/Aaron Oxford

aybiss@hotmail.com / Aaron.Oxford@studentmail.newcastle.edu.au

members.xoom.com/aybiss

Introduction

Infector is a subtractive synth (also known as analogue synthesizer emulator).

It consists of the following parts:

- Oscillator block
- Filter block (with its own envelope)
- Amp block (ditto)
- 2 independent assignable LFOs

Oscillators

Infector has got an oscillator section, which consists of two regular oscillators (called **OSC1** and **OSC2**) and a sub-oscillator. Both regular oscillators have built in Pulse Width Modulation (PWM) "circuitry", which can adds some "moving" quality to the sound. **OSC2** can be detuned from **OSC1**, for improved "phatness" (transposing and detuning effects). You can also change balance between these oscillators (relative volume of **OSC1** and **OSC2**). Both oscillators have a variety of built-in waveforms (simple ones - like sine or sawtooth wave, or more complex - like SuperSaw). You can also create waveforms yourself - using user waveform slots.

The user waveforms are divided into 4 double slots - A, A', B, B', C, C' and D,D'. For plain PWM use only half of a slot (oscillator waveform set to A, B, C, D). You can also set up two waveforms in the same slot (for example, A and A'), and set the oscillator waveform to mixed mode (AA'). Then the oscillator plays both waveforms at once, with a phase shift controlled by PWM LFO. It might give interesting effects if the waveforms have similar harmonic content.

The sub-oscillator is a simple oscillator tuned always octave down the note frequency. It has no PWM or transpose/detune controls, but you can change its volume and waveform (user waveforms are allowed, too).

Parameter	**Meaning**
OSC1/OSC2 Wave	Waveform of the OSC1/OSC2
PWM Rate	Rate of PWM Low Frequency Oscillator
PWM Depth	Depth of Pulse Width Modulation
PW Offset	Center pulse width (when LFO is zero)
OSC2 Transpose	Number of semitones OSC2 is transposed from OSC1

OSC2 Detune	Number of cents OSC2 is detuned from OSC1
OSC Mix	Balance between OSC1 and OSC2
SubOsc Wave	Waveform of the sub-oscillator
SubOsc Vol	Volume of the sub-oscillator
Glide	Note glide amount (smooth transition between different note pitches)

Filter

The synthesizer is equipped with a single 6-pole multimode filter, with a built-in ADSR envelope generator. The filter can be set to different modes, including 36dB/oct lowpass (**6L**), 24dB/oct lowpass (**4L**), 12dB/oct lowpass (**2L**), triple 6-pole notch (**Notchez**), 36dB/oct highpass (**6H**), 18dB/oct bandpass (**6B**), and mixed filter (**6X**). Except that distinction, some of filter modes have different versions (like **6L** have Multipeak, Separated and HiSquelch variants).

Every filter has two parameters - Cutoff and Resonance. Cutoff always controls the filter's cutoff or center frequency (or frequencies), and resonance is used for different parameters in specific filters (however, it usually controls width or height of resonant peaks). The filter cutoff frequency is controlled by the envelope generator. Two parameters control modulation amount and modulation shape (ie. how envelope generator's output affects the cutoff frequency). Filter cutoff can be also linked to note pitch, with the Keytrack parameter. Keytrack of 0ct means no keytracking, while Keytrack of 100ct means that pitch change of one octave up causes cutoff change of one octave up.

Parameter	**Meaning**
Flt Type	Filter type (like: lowpass, highpass, bandpass...)
Cutoff	Filter cutoff frequency (when no modulation occurs)
Resonance	Amount of filter resonance (meaning changes with filter types)
EnvMod	Amount of cutoff frequency modulation by the envelope generator (how much the envelope generator modulates the cutoff frequency)
Attack	Filter envelope attack time
Decay	Filter envelope decay time
Sustain	Filter envelope sustain level
Release	Filter envelope release time
Mod Shp	Modulation curve shape ("bends" envelope curve in either direction)
Inertia	Inertia amount (controls smoothing of manual parameter changes)
KTrack	Keytrack amount (how much note pitch affects cutoff frequency)

LFO

LFOs in **Infector** are used for controlling filter parameters. **LFO1** is connected to filter cutoff and envelope modulation amount, while **LFO2** is connected to filter cutoff and resonance. While it may seem "suboptimal", it's pretty effective because oscillators' PWM is linked to the separate PWM LFOs, making LFO to PWM path unnecessary (which also frees **LFO1** and **LFO2** from PWM-related duties). Both LFOs have a variety of different shapes, like sine, triangle, saw up and down, square, steps and special sample-and-hold (random) modes. LFO outputs are smoothed out to avoid rapid filter parameter changes (which might cause all those famous clicks and pops).

Parameter	Meaning
LFO rate	Period/frequency of LFO (expressed in Hertz or in ticks)
To Cutoff	Amount of modulation of cutoff frequency by LFO
To Env	Amount of modulation of envelope modulation amount (huh?) by LFO
To Res	Amount of modulation of resonance parameter by LFO
Shape	LFO waveform

Commands

Cmd #	Meaning

`01 aabb`	Start portamento up
	aa = number of ticks/4
	bb = number of semitones

`02 aabb`	Start portamento down
	aa = number of ticks/4
	bb = number of semitones

`03 00aa`	Start portamento to the specified note aa = number of ticks/4

`04 abcd`	Start vibrato
	a = LFO1 speed
	b = LFO1 depth
	c = LFO2 speed
	d = LFO2 depth

`05 aabb`	Protracker-style arpeggio aa = number of semitones up/down in 2 triple bb = number of semitones up/down in 3 triple

`06 aabb`	Play second note (two note arpeggio)
	aa = delay before playing 2nd note (in ticks/12)
	bb = interval between 2nd and 1st note

0C aabb	Reset LFOs aa = new position of LFO1 bb = new position of LFO2
13 000a	Simple shuffle (delay every second note by a/16 ticks)
E5 aaaa	Set channel detune aaaa=8000 - no detune, 8100 - 1 semitone up, 8080 - 1 quartertone up etc.
E9 000a	Retrigger every a/6 ticks
ED 000a	Delay note by a/6 ticks
FD 0000	Reset channel commands (disable vibrato, detune etc)
FE 0000	Reset ALL channels' comands (like FD placed in all channels)

Synth modes

Mode	Meaning
L1 (01)	Restart LFO1 on every note
L2 (02)	Restart LFO2 on every note
FE (04)	Don't restart filter envelope on new note (filter legato)
AE (08)	Don't restart amplifier envelope on new note (amplitude legato)
PQ (10)	Pitch Quantize (all portamentos etc. are quantized to nearest note)
MM (20)	Monosynth mode for MIDI (plays all notes in the same channel)
IK (40)	Inertia for Keytrack (makes cutoff changes related to keytracking smooth)

Fuzzpilz Cheese of the Week

Cheese of the Week is a somewhat oversized 4-oscillator wavetable/PM synth. It's not ***that*** good, to be honest.

Features

- 4 wavetable oscillators able to use any of 29 built-in and 8 custom waveforms, with the ability to phase modulate and sync to one another in a variety of ways. 1 noise source.
- 3 filters with 10 different modes.
- 2 mod envelopes. Custom envelope shapes or AHDDSR parameters.
- 2 LFOs, plus one LFO dedicated to vibrato. 11 built in LFO shapes, 4 custom.
- Crappy unison.
- MIDI input.

Tips and Warnings

- ***Do not*** attempt to copy the parameters and paste them into a pattern. Buzz will crash. I haven't been able to ascertain the cause of this, but I'm 99% sure it isn't my fault. I assume Buzz simply can't deal with so many global parameters; a hardcoded buffer size or something of the sort must be involved.
- Oversampling can be expensive, especially if you're not going to have only one voice playing at a time. Don't turn it up higher than you need.
- Similarly, don't turn down the mod detail parameter unless you're hearing zipper noise of some sort.
- Unison is also expensive, mostly because it works at the voice level – meaning that not only the oscillators are cloned, but also all the envelopes, filters and LFOs.
- MIDI input handling is mostly separate from sequenced notes; input is ***not*** recorded.
- Envelope modulation is relative to the envelope's sustain level – when an envelope reaches sustain, whatever it's set to modulate isn't changed by it.

Phase Modulation?

Many people seem to get confused about this, so here's a bit of explanation on the differences between FM (frequency modulation) and PM. The sounds that can be made by both are the same (and in fact, many "FM" synths actually perform PM) – this is because the only difference is in whether the modulator is added before or after the phase integration. Here's some pseudocode to clarify this:

```
// PM
out = osc(phase + mod);
phase += frequency;
```

```
// amounts to: out = osc(time*frequency + mod[time]);

// FM
out = osc(phase);
phase += frequency + mod;
// amounts to: out = osc(time*frequency + mod[0] + mod[1] + .. +
mod[time]);
```

Some Parameters

Parameter	Explanation
Velo	Global velocity sensitivity – this affects ***only*** the amplitude. In all other places where the velocity is needed, the original value is used.
Note Mode	Behaviour of envelopes on a new note if one is already playing on a track; also, virtual channels per track. • Mono — one voice; the envelopes restart, but their attack starts at the level they had reached during the old note. • Mono Restart — the envelopes are quickly brought down to zero before starting properly. • Mono Legato — if the playing note is releasing, the envelopes are restarted as in Mono mode; otherwise, the envelopes are not restarted. • Virtual 2/4/8 — virtual channels. If all channels are occupied, the oldest is stolen and its envelopes restarted as in Mono mode.
Glide Mode	The way the glide time is interpreted depending on the two notes involved. • constant — the glide time is not affected by the notes. • *octaves — the glide time is multiplied by the number of octaves between the notes. • /semitones — the glide time is divided by the number of semitones between the notes. ...and sums of the above.
Pitch Bend	Global pitch bend. The range and inertia also affect the behaviour of the pitch wheel with MIDI input.
Unison	Number of voices to play at once.
Pitch Spread	Total pitch spread of unison voices – e.g. if you have 5 voices and pitch spread is set to 10 cents, the voices' frequencies are detuned by -10, -5, 0, 5 and 10 cents respectively.
Pan Spread	Total pan spread of unison voices.
Transpose	Global transposition.

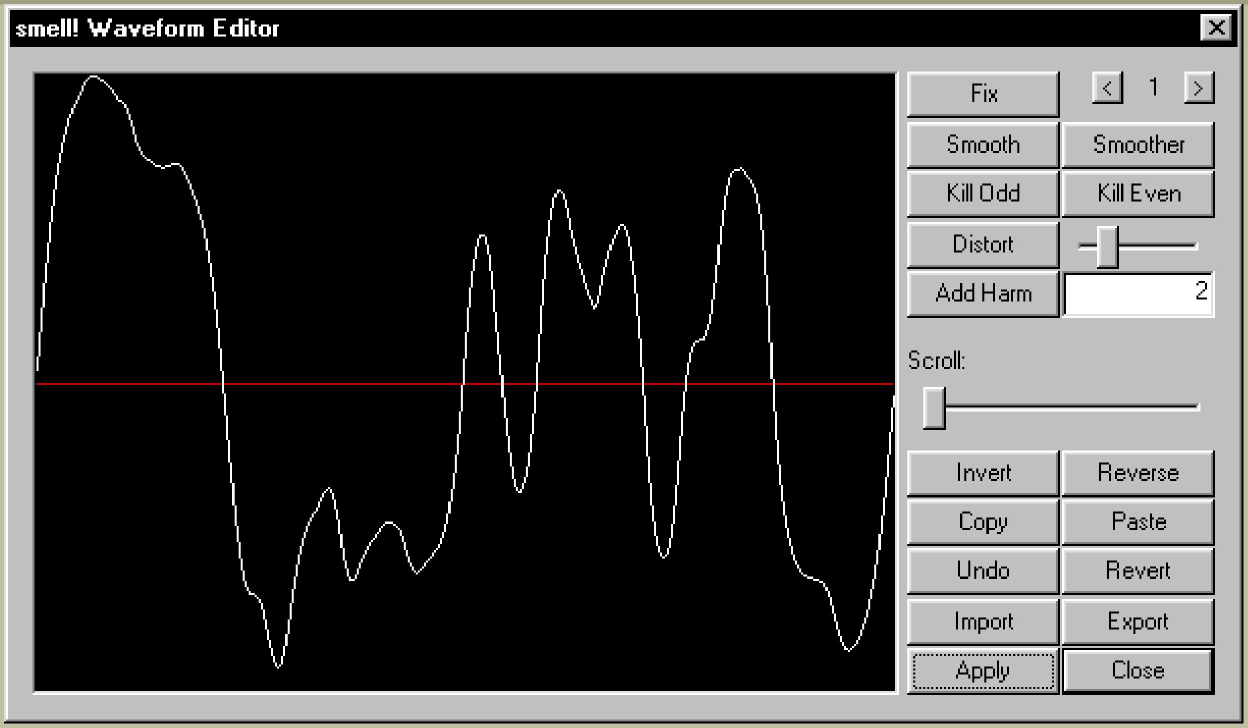
Note Sync	Phase initialization of each oscillator when a note is played. • Zero New — set each oscillator's phase to zero ***only*** if no other note was already using this voice. • Zero Always — set each oscillator's phase to zero whenever a new note starts. • Random New/Always — randomize each oscillator's phase.
Sync Mode	Not related to Note Sync; this controls what happens to a synced oscillator when the other oscillator wraps. • Reset — reset phase to zero. Classic hardsync. • Weak Reset — reset phase to zero only if it already is near zero. • Jump x — immediately increment the phase by x. • Reverse — reverse the direction in which the oscillator goes through the wavetable.
Mode	Oscillator routing. n(x) means that oscillator n is phase modulated by x.
Oversample	Oscillator oversampling.
On Velo	Oscillator n velocity sensitivity.
On Feedback	Oscillator n feedback. The output of the entire oscillator section is used here, not just the appropriate oscillator's.
On Phase	Oscillator n phase offset.
On Env	Each oscillator's amplitude can be additionally modulated by one of the three envelopes.
N Mode	What to do with the noise generator's output. • Add — add it to the oscillators. • Multiply — multiply it and the oscillators' output in various ways. • FM 1 — use it to frequency modulate oscillator 1. • FM 4 — use it to frequency modulate oscillator 4.
Fn Num	Number of filter instances in series.
S Oversample	Saturation oversampling – the saturation is a simple waveshaper that can introduce aliasing. Combat it with this if necessary.
Detail	Modulation detail. Any modulation that needs to be done is done every n samples, which means that lower values are better and use more CPU time.

Vib/Ln Shape	If the shape is set to one of the random modes, the speed parameter sets the time between two random values. • Random D — jump directly from value to value. • Random L — linear interpolation between values. • Random Q — also linear interpolation, but the fractional index moves as x^4 instead of as x. • Random SL/SQ — same as L and Q, but it only takes one fourth of the allotted time to reach the next value.
Old Voices	If the number of MIDI notes held exceeds the number of voices set by the MIDI Voices parameter, older notes have to be interrupted to make room for newer ones. This parameter determines what to do when a voice is freed and keys that have been interrupted are still held. • Forget — disregard them entirely. • Revive Old — play the note corresponding to the inactive key that has been held the longest. • Rejuvenate — same, but pretend the key has only just been pressed. (which can mean that next time a different key may get its turn even if this key is also interrupted before then) • Revive New — play the note corresponding to the inactive key pressed most recently. • Age — same, but consider the key as the oldest from then on.

Track Commands

- 00 — note delay. First byte is tick subdivision, second is delay.
- 01 — adjusts the glide time for one note. The data word is interpreted as a multiplier – 0x0000 is 0, 0x4000 is 0.5, 0x8000 is 1, 0xC000 is 1.5 and that sort of thing.
- 02 — adjusts the glide time for one note as 01 does, and also doesn't retrigger the envelopes for this note.
- 03 — adjusts envelope speed for one note. 0x8000 is *1, 0x8800 is *2, 0x7800 is /2, and so on.
- 1x — resets vibrato and LFOs; the data word is interpreted as the phase to which to set them. To get the correct command for the combination you want, add 1 for vibrato, 2 for LFO 1, and 4 for LFO 2 – thus, for example, 16 resets LFO 1 and LFO 2, but not vibrato.
- 2x — resets envelopes. 1 is the main envelope, 2 is mod envelope 1, 4 is mod envelope 2. If the data word is 0x0001, the envelopes are reset hard, as in Mono Restart mode. If it's 0x0002, they're reset softly. Otherwise they're treated the way they normally would on a new note.

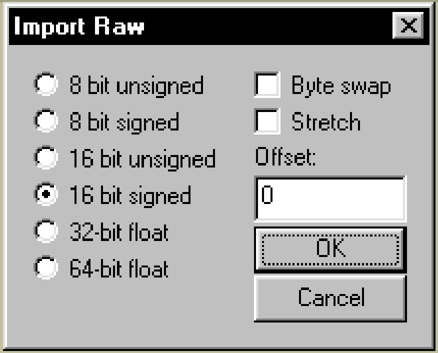
Waveform Editor



Draw with the left mouse button. Shift-click to draw a straight line. Ctrl-shift-click to draw a straight line to the mouse's x position and the wave's y position. Right click to load the built-in waveforms as presets.

Some possibly non-obvious buttons and things:

- Fix — remove DC offset and normalize.
- Kill Odd — remove all odd harmonics.
- Kill Even — remove all even harmonics.
- Distort — just a saturator. Uses the slider to its right to control amount.
- Add Harm — add a pitched-up version of the waveform to itself. Uses the slider and the input box.
- Revert — reverts to the state the waveform was in when the editor was opened.
- Apply — applies changes.
- Close — applies changes and closes the editor. Changes are ***not*** applied if you close the window using the X button.

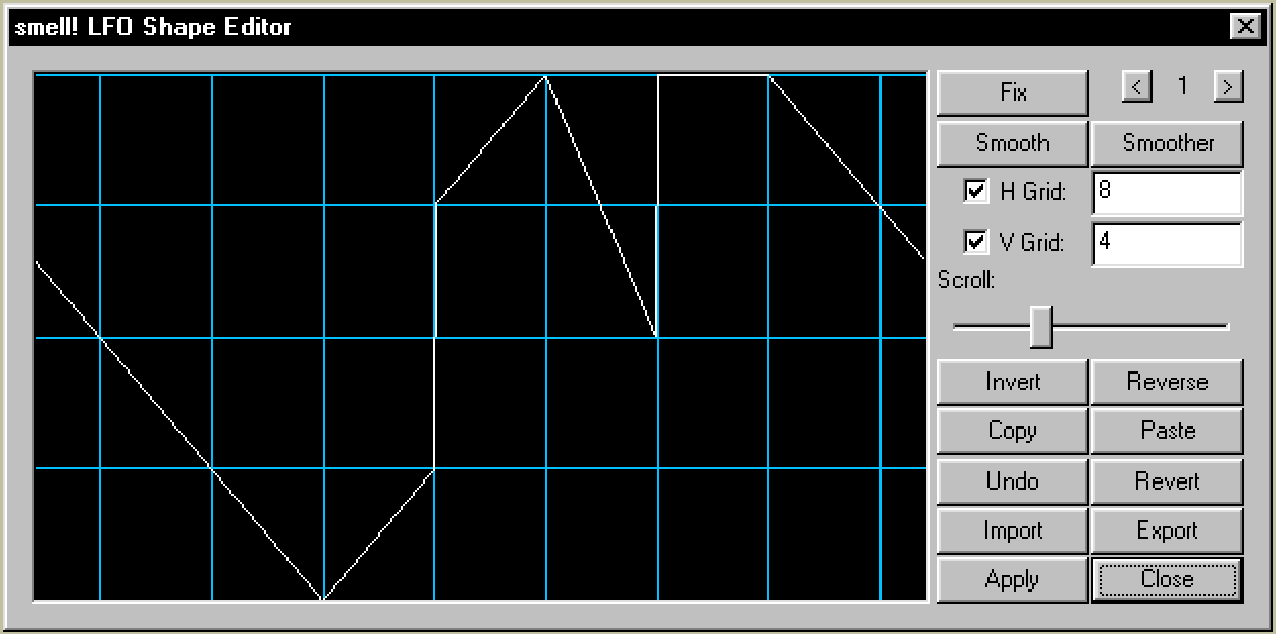

Import Raw

8 bit unsigned
8 bit signed
16 bit unsigned
[✓ 16 bit signed]
32-bit float
64-bit float

Byte swap: unchecked
Stretch: unchecked
Offset: 0

Cheese can import raw data as waveforms. It likes to find 256 samples. If you give it more, the rest is thrown away. If you give it less, it makes do with rubbishy linear interpolation between the last sample and the first, unless you check "Stretch", in which case it stretches what it finds to make it fit.

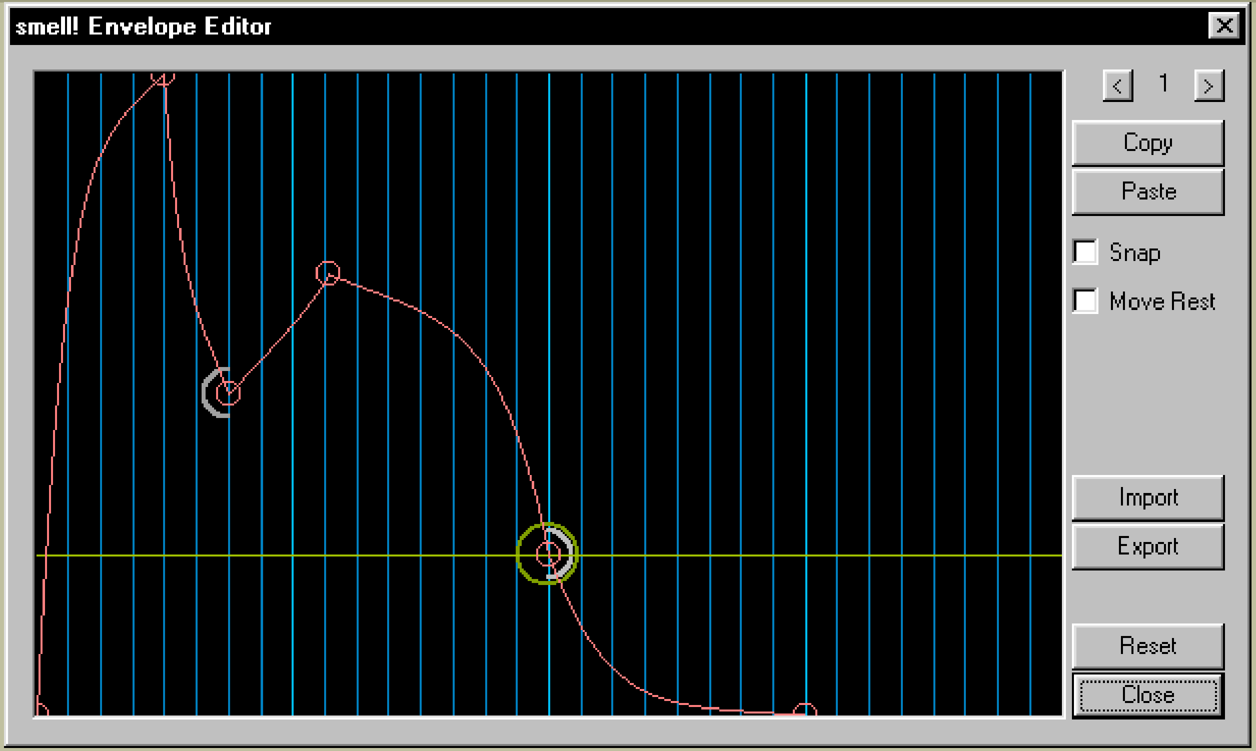
LFO Shape Editor


smell! LFO Shape Editor

H Grid: 8 [✓]
V Grid: 4 [✓]

Draw the same way as in the waveform editor. There's a grid to help you if you want to do rhythmic stuff.

Envelope Editor



Middle click or Ctrl-shift-click to add a new point. Press the left mouse button between two points and drag up and down to adjust the slope. Shift-drag sideways to scroll. Ctrl-drag to zoom. Right click on a point for options regarding that point, right click on the background for envelope options.

The space between the bright blue grid lines represents the duration of one tick (only if the envelope speed parameter is 1, of course). The point with the left grey semicircle around it is the loop start point; the point with the right grey semicircle is the loop end point. The point holding the green circle and line is the sustain point, or sets the reference level if the loop mode isn't sustain. Speaking of which:

Custom envelopes can be set to loop between the loop points, to remain at the sustain point when they reach it, or to just go through the entire shape without halting. Even when the loop mode is not "Sustain", however, the level of the sustain point is used as a reference for modulation if the shape is used for one of the mod envelopes.

And finally, release modes:

- Release param — when the note is released, the rest of the shape is ignored and the envelope simply dies according to the release parameter.
- Param with loop — the envelope keeps looping and more or less slowly fades out, depending on the release parameter.
- Continue — the envelope ignores the loop and continues all the way to the bitter end, where it dies, alone and unloved.
- Jump — tired of its meaningless existence, the envelope first jumps to the loop end point to accelerate its

demise.

Closing Remarks

I hope this wasn't too unclear. I thank various people who tested it, especially everyone who posted in the *Buzzchurch thread*.

Cheese is freeware – do whatever you like with it, but don't try to sell it. It's not donationware either, though if you want to give me money I'm not stopping you. The code is all mine, except the FFT code I use to generate the wavetables, which was written by Laurent de Soras.

— f

=================
Fuzzpilz FuDrum 2
=================

:: What is it?

It's a simple drum synthesizer. While it was originally designed for somewhat ers-ish snares, it can be used for various other sounds as well.

:: How does it work?

There are two oscillators; one for a "thump" and one for white noise, plus one filter.

:: Parameters

```
- Thump Amp:    Amplitude of the thump.
- Thump Wav:    Waveform of the thump.
- Thump Freq:   Frequency of the thump.
- Thump Dec:    Decay of the thump, i.e. the time it
                takes to reach -60 dB. Duh so far.
- TFD Dir:      Direction of the thump frequency decay thingy.
- Thump FreD:   "Frequency decay" of the thump, i.e. the time
                it takes to go one octave up or down, according
                to the setting of the TFD Dir parameter.
- Thump Limit:  Minimum or maximum frequency of the sweep, depending
                on the direction of the frequency decay.
- Noise Amp:    Amplitude of the noise hit.
- Noise Decay:  Its decay.
- Target:       What is filtered, i.e. nothing, thump, noise or both.
- Filter:       Filter type, i.e. lowpass, highpass, bandpass 1 (constant skirt gain),
                bandpass 2 (constant peak gain) or bandreject (notch).
- Cut:          Cutoff frequency of the filter.
- Res:          Resonance of the filter.
- FD Dir:       Direction of the cutoff decay.
- Filter Dec:   Decay of the cutoff, works the same way as the Thump FreD parameter.
- Cut Limit:    Minimum or maximum cutoff frequency.
```

:: Sequencing

The first four columns of each track are what counts here:

1. Subdivide.   Divides the tick into x bits.
2. Delay.   Delay the trigger by y xths.
3. Retrigger.   Repeat the trigger every z xths, after the delay if it is set.
   (or trigger x times in z ticks)
4. Trigger.   You obviously don't have to use any of the others unless you want to.

:: Notes

- You can give the machine a 0 for the retrigger. The resulting sound
  might be useful occasionally, and I like it, so I kept this.
- There's no antialiasing on the thump oscillator. I like
  it aliased.

:: Things added

- Five more waveforms - a cosine saw and four "combo" waveforms. The Combo 2 and 4
  waveforms are somewhat insane and best used with frequency sweeps.
- Thump frequency and filter cutoff limits.
- Other bandpass. (yes, I'm still using the RBJ filters)
- Some optimisations that didn't help much (the performance is still crap), but
  may make a slight difference if you're on a slower machine, especially on the
  cutoff sweeps.

Fuzzpilz FuDrum 3

Description

FuDrum 3 is a drum synthesizer. The 3 means it's the third version, unsurprisingly. New to this one are attack, virtual channels, note support and MIDI input.

How to use it

FuDrum 3 works the same way as the previous versions, basically. There are two sound sources: a wavetable oscillator and a white noise generator. Each of these has its own envelope, but both are sent through the same filter. Notes can be given, but are not necessary.

Parameters and Attributes

Parameter	Description
Note Mode	Sets the way the note (if one is used) influences the sound: <ul><li>Start Freq: Starting frequency of the oscillator.</li><li>Limit Freq: Limiting frequency of the oscillator.</li><li>Start Cut: Starting cutoff frequency of the filter.</li><li>Limit Cut: Limiting cutoff frequency of the filter.</li></ul>
Thump Amp	Amplitude of the oscillator.
Thump Wav	Waveform of the oscillator.
Thump Freq	Starting frequency of the oscillator.
Thump Att	Attack time of the oscillator's envelope.
Thump Dec	Decay time of the oscillator's envelope.
TFD Dir	Direction in which the oscillator's frequency changes.
Thump FreD	The time it takes for the oscillator's frequency to move one octave in the direction set by the above.
Thump Limit	Limit of the oscillator's frequency. It can't get higher/lower than this depending on the direction setting.
Noise Amp	Amplitude of the white noise generator.
Noise Att	Attack time of the noise generator's envelope.
Noise Dec	Decay time of the noise generator's envelope.
Target	Whether to apply the filter to the thump, the noise, both or neither.
Filter	Filter type.

Cut	Initial filter cutoff frequency.
Res	Filter resonance.
FD Dir etc.	These parameters control the movement of the cutoff frequency. They work the same way as their oscillator frequency counterparts above.

Attribute	Description
Virtual Channels	Whether to allow more than one virtual channel per track.
Changes etc.	You would want to turn this off if you needed, for example, to play an extremely long hit, then have the same track play much shorter hits while it's still decaying in the background.
MIDI Channel	Channel for MIDI input.
MIDI Track	The track on which MIDI notes are played.

Stuff

Complain **here**. Check **here** for betas of stuff I'm working on.

===============
Fuzzpilz FuDrum
===============

:: What is it?

It's a simple drum synthesizer. While it was originally designed for somewhat ers-ish snares, it can be used for various other sounds as well.

:: How does it work?

There are two oscillators; one for a "thump" and one for white noise, plus one filter.

:: EXPLAIN THE PARAMETERS, OR I WILL KILL YOU!!!1

No need to get your knife dirty. Here they are, in order of appearance in the parameter window:

```
- Thump Amp:    Amplitude of the thump.
- Thump Wav:    Waveform of the thump.
- Thump Freq:   Frequency of the thump.
- Thump Dec:    Decay of the thump, i.e. the time it
                takes to reach -60 dB. Duh so far.
- TFD Dir:      Direction of the thump frequency decay thingy.
- Thump FreD:   "Frequency decay" of the thump, i.e. the time
                it takes to go one octave up or down, according
                to the setting of the TFD Dir parameter.
- Noise Amp:    Amplitude of the noise hit.
- Noise Decay:  Its decay.
- Target:       What is filtered, i.e. nothing, thump, noise or both.
- Filter:       Filter type, i.e. lowpass, highpass, bandpass or bandreject (notch).
- Cut:          Cutoff frequency of the filter.
- Res:          Resonance of the filter.
- Filter Dec:   Decay of the cutoff, works the same way as the Thump FreD parameter.
- FD Dir:       Direction of the cutoff decay.
```

:: Sequencing

The first four columns of each track are what counts here:
1. Subdivide. Divides the tick into x bits.

2. Delay. Delay the trigger by y xths.
3. Retrigger. Repeat the trigger every z xths, after the delay if it is set.
4. Trigger. You obviously don't have to use any of the others unless you want to.

:: Notes

- You can give the machine a 0 for the retriggererererer,
  which may or may not sound stupid. The resulting sound
  might be useful occasionally, and I like it, so I kept this.
- There's no antialiasing on the thump oscillator. I like
  it aliased, and you can go eat a robot if you don't.
- Stale cola tastes horrible.

Orange Kid 1.0

The Orange Kid is, basically, a subtractive synthesizer. It's named after a character of the old SNES RPG Earthbound; Orange Kid was one of two inventor kids in the game's second town (fittingly named Twoson; the other cities were called Onett, Threed and Fourside). You could invest into both inventors, but while giving Apple Kid your money was very useful (and indeed necessary for the storyline), Orange Kid gave you a completely useless machine and then probably spent the money on clothes or something.

How stuff works:

:: Envelopes
There are two ADSR envelopes. The first one is the normal amp envelope and can also modulate other parameters; the second is a modulation envelope that can, but needn't be applied to the amplitude. The modulation of each parameter works by the equation modulated=normal*(1+modulation*envelope).

:: Filters
Orange Kid has four resonant 2p lowpass/highpass filters. Each parameter of filters 2-4 (i.e. filter type, cutoff, and resonance) can work independently from Filter 1 or follow it in Slave mode. The filter routing parameter explains itself; so does the inertia parameter. The "Cutoff Limit" parameter determines what frequency "100.0%" as cutoff actually means. The note dependent settings for this follow pitch modulation, glide and vibrato.

:: Waveforms
Orange Kid has two oscillators; note that the two envelopes are completely unrelated to this. The parameters "Phase", "Transpose" and "Detune" only apply to the second wave relative to the first. "Noise Seed" is the seed for the random number generator used to make the Noise 1 and Noise 2 waves (especially with Noise 2 you're going to want to get the same wave as last time). Noise 1 turns to quasi-white noise when being shifted in pitch; this was originally a bug, but I left it in because I liked it. If you don't, that's your problem. The "Mix" parameter is self-explanatory. "Phase start" and "Phase end" may need explaining - they allow you to only use a portion of the wavetable and apply to both oscillators.

:: The other parameters
...are easy to understand. Glide time, vibrato depth and rate, global gain (I hate having to use dozens of cheapo amps), and global tuning.

:: Attributes
This does not need much explaining either. You can choose a MIDI channel for input (0 means none), turn MIDI velocity on and off, and record MIDI input, for which you need to be playing an empty pattern in record mode. The Env Mod Detail attribute is the number of samples the machine waits before doing its modulation stuff. Increase for lower CPU usage, decrease for (theoretically) higher quality.

:: Problems
In case it ever happens that the filters choke on something (which would result in the machine producing no sound) you can reset them via right click -> Reset Filters.
Also, sometimes the settings of the machine aren't read properly when you load a song that uses it. I have no idea what could possibly cause this (it looks like some parameters are reset to zero internally before Tick is called the first time; there is no reason at all why this should happen) and can therefore not fix it. Sorry.

Fuzzpilz Otter

Otter is a somewhat organ-like synthesizer. Eight sine oscillators, followed by one saturator each and a four-pole lowpass or highpass filter over the whole thing.

Parameter	**Explanation**
Mix	Whether the oscillators are to be simply added together or multiplied.
Gain i	Gain of oscillator i.
Sat i	Saturation of oscillator i.
Harmonic i	The frequency of oscillator i will be the fundamental frequency times this.
Detune i	Each oscillator can additionally be detuned by –100..100 cents.
G Inertia	Gain inertia - how long it takes the gain of each oscillator to change to a new value.
Attack, Decay, Sustain, Release	Amplitude envelope. Values above 32768 are in ticks.
Filter	Filter type - off, lowpass or highpass.
Max Cutoff	The filter's maximum cutoff frequency.
Cutoff	The filter's cutoff frequency.

Resonance	The filter's resonance.
F Inertia	Filter inertia - how long it takes the filter's cutoff frequency to change to a new value. This does not influence envelope or LFO modulation, but if the cutoff is tied to the note, it will change at this speed on a new note.
F Attack, Decay, Sustain, Release	Filter envelope. Values above 32768 are in ticks.
Mod	Filter cutoff envelope modulation depth.
LFO speed	The LFO's frequency. Values above 2048 set the period in ticks.
LFO mod	Filter cutoff LFO modulation depth.
Gain	Overall gain.

MIDI support does not exist because I'm lazy. Use uMw or Polac's loader if you want MIDI - both are excellent choices. If Otter goes wibbly somehow, right click on it and click Reset. That should take care of it.

I made this because GrubWerm, who needed to use his sustain pedal as a bass drum trigger, requested it. So, how does it work? You simply bind the sustain pedal to the "trigger" parameter and adjust the rest as you needed it. Now, if you press down the sustain pedal, you should hear one click, and nothing more. Before you can trigger it again, you have to pull it back behind the sensitivity value. Send the output through an LdC Trigger, and that's all. It may also be useful for just generating clicks, and you can do interesting things if you combine it with a Jeskola Multiplier.

-- ruckediguh, blut ist im schuh!

This is the beta of a tracker machine that's
sort of nearing completion.

Parameters, in order of appearance:

- Tick subdivide
- Note delay
- Note retrigger - when used without a note,
  the sample starts retriggering from this
  position. When set to 0, the sample returns
  to the pitch and offset where it belongs
  according to the last explicit (note) trigger.
- Offset
- Stretch (to this many ticks)
- Note
- Sample
- Velocity (40=100%)
- Pan (40=center)
- Effect Command 1
- Effect Data 1
- Effect Command 2
- Effect Data 2

Effects are explained below.

Attributes:

The only one worth explaining is "Interpolation".
0 - no interpolation
1 - linear
2 - spline

Effects: (effect data: xxyy)

00: Stop retriggering after xxyy times.
    Is reset the next time a retrigger is started.
01: Retrigger "feedback". Obviously not really
    feedback - the velocity is multiplied by this
    on each retrigger. C000 means one (0 dB if you
    prefer) and this isn't reset until you say so.

02: Multiply the frequency by this on each
  retrigger, 8000 is one.
03: Transpose by xx.yy semitones on each retrigger.
  Centered around 8000.
04: Multiply retrigger time by this after each
  retrigger.
05: Reverse sample direction at this point.
06: Pitch down, xx semitones in yy subticks.
07: Pitch up, xx semitones in yy subticks.
08: Reverse so that the offset specified in the
  offset parameter is reached after xxyy subticks.
09: Probability. This affects note off events.
0A: Change the offset on each retrigger. A value of
  8000 means +-0.
0B: Note cut after xxy.y subticks.
0C: Note release after xxy.y subticks. This has no effect if
  no envelopes are in use.
0D: Multiply note cut/release time by this after each retrigger.
20: Humanize velocity.
21: Humanize pitch with a range of xx.yy semitones.
22: Humanize offset.
23: Humanize pan.
24: Humanize filter cutoff, range of xx.yy semitones.
30: Set filter type:
    0 off
    1 LP
    2 HP
    3 BP with constant skirt gain
    4 BP with constant peak gain
    5 Notch.
31: Set filter cutoff.
32: Set filter resonance.
33: Cutoff down, as in 06.
34: Cutoff up, as in 07.
40: Vibrato speed, its period is xxy.y subticks.
41: Vibrato depth.
42: Vibrato waveform:
    0 Sine
    1 Triangle
    2 Square
    3 Saw
43-45: The same for the filter cutoff LFO.
50: Sync vibrato now, to the offset xxxx in the

waveform.
51: Sync vibrato on notes. No effect data turns this
off.
52-53: Same for LFO.
60: Jump xxyy subticks ahead, disregarding the playback speed
61: Jump xxyy subticks back, disregarding the playback speed
62: Jump xxyy subticks ahead relative to the playback speed
63: Jump xxyy subticks back relative to the playback speed
Ax: Reset, where x is a sum of the following:
1: Retrigger effects.
2: Humanization crap.
4: Filters.
8: LFO and vibrato.
A0 also resets everything.

UnwieldyTracker can currently use wavetable envelopes
for volume, pitch, filter cutoff, filter resonance and panning.

Geoffroy TapTempoHACK

This is a "hack" machine based on P. DooM's BUZZHACK 1.01 library (thanks!).
It can cause problems with older or newer versions of BUZZ (will there ever be ?). BUZZHACK was designed for the Oct. 2000 version of BUZZ.

This buzz generator is suited for live use : it lets you adjust the current Buzz tempo to what you "tap" with your MIDI keyboard.
I'm investigating on how to release a GUI with a button to press for computer addicts :)
Let's see how it works :

Parameters :

- **Activate : activate the machine**
- **Tap Counts : this is the number of taps needed to "guess" the tempo**

Here's an example : the most common use is 5 taps.
You will tap 4 times, just like "one ... two ... three ... four"
The 5th tap will launch Buzz according to the Buzz mode set below

- **Jump Mode : there are four modes :**

- no jump : Buzz will continue playing without any interruption
- tick : Buzz will jump to tick set below with "jump to tick" parameter
- previous : Buzz will jump to the previous "block" of "modulo ticks" ticks.
Here's an explanation : let's say you have a song made of loops of 64 ticks (remember something huh ?)
Set "Modulo Tick" to 64.
If you end to tap the tempo and Buzz is at position 140, then it will rewind to tick 128 (i.e. previous position that is a multiple of 64).
- next : just like "previous", except it will forward to next block.
In the previous example, it would skip to 172.
- closest : it will skip to the closest block (in our example, 128)
- loop start : it will go to Buzz loop start (you set with CTRL-B and CTRL-E)

- **Modulo tick : the size of the block**
- **Manual Tap : this parameter only appears in the pattern editor, it was made to work in cunjunction with BTDSys Trigger for automatic beatmatching. Well, this doesn't work quite well but worth the try.**

Attributes :

- **Midi Channel : to listen to for a key (0 for all channels)**
- **Midi Note Trigger: the key to "tap" the tempo**

for any bug or anything else : coder@minizza.com

thanks go to P.Doom for his Buzzhack library, Mute for his wonderful ideas and help (sorry beatmatching doesn't work quite well), BTDsys for his help during beatmatching debugging, and all buzz enthusiasts
dedicated to Daniel Palomo Vinuesa

Geonik's DirectX Sound Input

by *George Nicolaidis*

Description

A DirectSound recording machine, use your sound card's mixer to select input (audio cd, microphone, line in, etc). You must be using DirectX output drivers for this machine to work.

What's new

Version 1.0	Initial release

Usage

Hit "Configure" in the context menu of the machine to set it up. Everything should be quite straightforward. Channel mix is there because Buzz doens't support stereo generators, so input channels have to be mixed into one.

Notes

If sound gets very clicky and trashed, hitting "Reset driver" may help a bit.

Not tested on NT. Buffer underrun detection not quite good. High latency... in general, I am not sure if it is any better than Wave In :) Tell me if you can think of any fixes, and PLEASE send some bug reports

Donationware

If you like and use Geonik's plugins, you can register them by sending any amount of cash (in any currency) to the following address

George Nicolaidis
31, Agathoupoleos str.
54636 Thessaloniki
Greece

Contact Information

Author	George Nicolaidis aka Geonik
Email	geonik@egnatia.ee.auth.gr
HomePage	http://egnatia.ee.auth.gr/~geonik/home

Geonik's Omega-1

an virtual synth using physical modelling synthesis
by *George Nicolaidis*

Description

The Omega-1

The success of the "Plucked String" clearly demonstrated the need for more quality generators. *Omega-1* is an attempt at regrouping various flavours of the original algorithm and new physical models in one single generator. Each flavour is referenced by its instrument number, making the plug-in work as virtual synthesizer. The machine is extendable and therefore new instruments can be added in the future with no compromise in compatibity with older songs.

What's new

Release 1	Initial release

Usage

G e n e r a l   P a r a m e t e r s

Note	While every note is acceptable, be careful with very low octaves
Volume	Not much to say about this one
Reserved	Do not touch this one ! It may be used in future versions of the plugin
Control 1,2,3	The behaviour of these parameters depends on the currently selected instrument and is described below

A t t r i b u t e s

Dynamic Range	When the sound output by a channel drops below a certain level, the channel stops running to preserve CPU time and becomes available for new notes when dynamic allocation is enabled. This attribute specifies the dynamic range of each channel, ie the distance between the louder and the quieter sound measured in dB.
Default Volume	When the volume for a new note is not specified, this is used.
Dynamic Channels	Setting this to other than zero enables dynamic allocation of channels. It is equal to the maximum number of channels to play at any time, including normal parameter channels. Dynamic allocation means that new notes don't cut the sound of precious ones, as long as there are free channels. When no more channels are available, the quietest channel plays the new note.
Note Off Amplitude	Sending a note-off decreases the amplitude of the currently playing note to this percentage

Instruments

Blank cells mean that the control is ignored.

Instrument	**Description**	**Control 1**	**Control 2**	**Control 3**
1 - Original Ps	The original *Plucked String*, sounding exactly the same as in the omonymous plug-in	Dampening	\	\
2 - Guitar String	Enhanced version of the Original Ps that sports correct tuning for all octaves and filtering to reduce the high frequencies produced by the pluck	Dampening	\	\
3 - Grand Pluck	Two Guitar Strings are played simultaneously at slightly different frequencies to create a chorus effect (set Detune to other than zero)	Dampening	Detune	\
4 - Mandolin	An attempt at reproducing the sound of a mandolin	Dampening	PluckPos	BodySize

Controls

Key	**Control name**	**Description**

Dampening	Dampening Factor	High values make strings decay faster. It relates to the overall friction of the instrument
Detune	Detune	0 is no detune and 128 is max detune. Aplicable when more than one strings are pluck simulteanously
PluckPos	Pluck Posistion	
BodySize	Body Size	

Notes

You must have dynamic channel allocation enabled with dynamic channels set to at least twice the amount of parameter channels to use Grand Pluck.

Credits

The algorithms used are based on code by Perry Cook. The original plucked model is by Karplus-Strong, and the enhanced version by Jaffe-Smith.

Bugs

Since it is new code, there are probably some bugs left. Mail me if you encounter the nasty insect.

Contact Information

Author	George Nicolaidis
Email	geonik@egnatia.ee.auth.gr
HomePage	http://egnatia.ee.auth.gr/~geonik/home

Geonik's Plucked String

an application of Physical Modelling Synthesis
by *George Nicolaidis*

Description

Physical Modelling Synthesis

Up to now, all the synthesis methods have been attempts to reproduce (or model) the *waveform* of a particular instrument or sound. The technique of physical modeling is however fundamentally different. With physical modeling it is the actual physics of the instrument and its playing technique which are modelled by the computer. Typically, a physical model instrument takes the form of a recursive algorithm in which the contents of a delay line (the length of which corresponds to the number of samples required to produce one period of waveform at some desired pitch) are subject to some kind of modification process on each cycle of the recursion. The initial contents of the delay line, and the details of the modification process play a key role in determining the character of the sound produced, and every effort is made to bring these into line with the way the acoustics of a given instrument really operates. This generator takes a look at the one of the earliest and simplest examples of a physical model - the Karplus-Strong "Pluck-String" algorithm.

The Karplus-Strong Plucked String Algorithm

Looking for new and computationally efficient methods of sound synthesis, Alex Strong was the first to find this new synthesis method. Essentially, the algorithm could be described as a variation on wavetable synthesis, an important distinction, however, being that the wavetable data is not kept constant, but subject to modification on each period.

What's new

Release 1	Initial release
Release 2	50% faster, dynamic channel allocation
Release 3	10-20% faster, sound isn't cut when user presses 'Stop'

Usage

P a r a m e t e r s

Note	While every note is acceptable, notes below C-3 produce garbage and notes above C-6 are de-tuned. Be careful !
Volume	Not much to say about this one
Slide to note	When set, the note currently playing slides to the note on the same line
Dampening factor	Increasing this parameter makes the sound shorter

Attributes

Dynamic Range	When the sound output by a channel drops below a certain level, the channel stops running to preserve CPU time and becomes available for new notes when dynamic allocation is enabled. This attribute specifies the dynamic range of each channel, ie the distance between the louder and the quieter sound measured in dB.
Default Volume	When the volume for a new note is not specified, this is used.
Dynamic Channels	Setting this to other than zero enables dynamic allocation of channels. It is equal to the maximum number of channels to play at any time, including normal parameter channels. Dynamic allocation means that new notes don't cut the sound of precious ones, as long as there are free channels. When no more channels are available, the quietest channel plays the new note.

Notes

The Jaffe-Smith extension to the algorithm is not implemented.

Contact Information

Author	George Nicolaidis
Email	geonik@egnatia.ee.auth.gr
HomePage	http://egnatia.ee.auth.gr/~geonik/home

Geonik's PrimiFun

by *George Nicolaidis*

Description

The PrimiFun Synth

The PrimiFun's name and implementation comes from Native-Instruments' *Generator*. Basically it is a pulse oscillator sent to a low-pass filter whose cut-off frequency follows its amplitude, the result being saturated.

Features

- Dynamic channel allocation
- Real ADSR envelope with linear attack and exponential decay
- Volume ramping for click removal
- Full MidiIn support
- Slow :)

What's new

Version 1.0 / Release 1	Initial release
Version 1.1 / Release 2	Dc offset removed
Version 1.2	Midi In support

Usage

P a r a m e t e r s

Volume	Not much to say about this one
Attack Time Decay Time Sustain Level Release Time	The standard ADSR envelope. When the envelope is triggered with a new note event the volume rises during the Attack Time with a linear slope to the Volume value after which it decays exponentially with the Decay Time to the Sustain Level. The output is held at the sustain level until a note-off event after which the volume decays exponentially with the Release Time back to zero

Pulse Width	The pulse shape. A ratio of 1:1 is a square wave
Filter Envelope	The maximum cut-off frequency of the filter
Resonance	Controls the filter's resonance. 0% is no resonance while 100% is maximal resonance, ie self oscillation

A t t r i b u t e s

Dynamic Channels	Setting this to other than zero enables dynamic allocation of channels. It is equal to the maximum number of channels to play at any time, including normal parameter channels. Dynamic allocation means that new notes don't cut the sound of precious ones, as long as there are free channels. When no more channels are available, the quietest channel plays the new note.
Default Volume	When the volume for a new note is not specified, this is used.

Notes

Enjoy. Send bug reports too. And buzz setups with interesting sounds

Donationware

If you like and use Geonik's plugins, you can support the author by sending any amount of cash (in any currency) to the following address

George Nicolaidis
31, Agathoupoleos str.
54636 Thessaloniki
Greece

Contact Information

Author	George Nicolaidis
Email	geonik@egnatia.ee.auth.gr
HomePage	http://egnatia.ee.auth.gr/~geonik/home

Sintetizador Guru5 para Buzz 1.2

Programado por Juan Antonio Arguelles Rius aka "Arguru"

13-Enero-2000

Introduccion

Guru 5 es un sintetizador polifonico de 8 voces [por maquina] de modelado analogico para ser usado junto a buzz como generador. Aunque consume bastante potencia de CPU, es un generador bastante versatil y potente gracias a sus posibilidades de modulacion y su matriz de envios de sus dos LFO's y sus dos ENV's que pueden modular parametros como el ancho de pulso de un Oscilador o el corte de frecuencia del filtro independientemente y en cualquier medida, (desde -100% a +100). Consta de 53 parametros globales y 3 parametros de pista.Debido a la amplia cantidad de parametros, necesitaras un buen monitor que aguante grandes resoluciones para poder editar todos, ya que el editor de parametros de maquinas de buzz no tiene deslizador o "scroll".

Funcionamiento

Osciladores

Guru5 se basa en la sintesis substractiva. Basicamente, este tipo de sintesis "substrae" harmonicos a un tono o timbre base. Esta substraccion de harmonicos se realiza con un filtro [VCF = voltage controlled filter] Los osciladores producen este tono base del sonido. Pueden producir 4 tipos de onda:

Sinusoidal [Sine]: la que menos harmonicos posee. De hecho, en teoria solo posee un harmonico cuya frecuencia corresponde a la de la nota tocada. Usada para tonos mas suaves como flautas, etc...

Diente de Sierra [Sawtooth]: la que mas harmonicos posee, por tanto, la que mas es afectada por el filtro [VCF]. Tipica onda usada en los sonidos sinteticos.

Pulso[pulse]: tren de pulsos. En este tipo de onda, entra en juego el parametro "ancho de pulso" [PW] que puede variarse para conseguir distintos tonos o modularlo para darle caracter al sonido.

Random: genera ruido blanco. Junto al VCF y a las ENV's pueden generar efectos, sonidos percusivos, o el clasico efecto barriendo el filtro paso-bajo: el viento.

Hay 3 osciladores en guru5:

- Principal: base.

- Secundario: adherido en frecuencia al tono, pero puede desafinarse a traves de los controles de desafinacion [OSC2 tune y OSC2 fine]

- SubOscilador: adherido al principal, pero una octava por debajo de este. Se usa para engordar el sonido.

<u>VCF Filtro</u>

El filtro es la parte sin duda que da mas toque analogico al sonido. Este componente substrae harmonicos de la salida producida por los osciladores. El VCF tiene tres respuestas:

- Paso Bajo [Lo Pass]: substrae harmonicos por encima de la frecuencia de cutoff.

- Paso Alto [Hi Pass]: substrae harmonicos por debajo de la frecuencia de cutoff.

- Off : no filtra.

El control Q [Resonancia] es, digamos, la atenuacion del nivel de la banda de la frecuencia cutoff del filtro. Este atenuamiento engorda bastante el sonido. Ajustada al maximo podria llegar a la autooscilacion y podras reventar algun que otro bafle.

<u>Envolventes y LFO's</u>

Guru5 tiene dos envolventes. Una envolvente esta compuesta por 4 etapas, normalmente se aplica al volumen y/o al corte de frecuencia del filtro. Ataque, Decaimiento, Sostenido, Relajacion. Asignables a cualquier parametro.

Ademas posee dos LFO's, utiles para modular ciclicamente parametros como el corte de frecuencia del filtro, tono de algun oscilador [vibrato], ancho de pulso, etc...

Todos los envios de modulacion de parametros [OSC's PW,PITCH,AMPL,VCF Q, VCF CUTOFF, etc...] van desde -100% a 100% [-100% = polaridad negativa]

Contacto:

arguru@vermail.net

2ndP IGroove v1.0

2ndProcess' Groove Aid Machine for Buzz (-> www.jeskola.com/www.buzzmachines.com)

What's different to MGroove...

I **renamed "shuffle" to "swing"** to avoid confusion with Hoester's GrooveBox and normal terminology.

There's still a slider called **"shuffle"** but it's not the same as in MGroove but **like in Hoester's GrooveBox** - it's the **randomization amount**! So now you can emulate a bad live musician :(

More **echo times** in the Info view...

Added **track commands**, which replace the old sync trigger row. You can have up to 4 at once by adding tracks.

IGroove is switched off by default (MGroove couldn't be switched off at all :) You can right-click the machine and click on "switch on/off" to change state or use command **0x11 to switch on** and command **0x10 to switch off**. You can still use it as echo time calculator even when it's switched off.

Old "sync all" trigger is now "0" command, old "0" sync (phase) trigger is now "1"; added a phase sync trigger for the other (faster) phase :)

Installation

Put *IGroove.dll* into the *Buzz\Gear\Generators* folder. Leave the old *MGroove.dll* where it is to be compatible with old songfiles :) but I recommend to hide it in *index.txt* by writing a comma (and nothing else) behind its name. Add a new line containing this: "IGroove,2ndP IGroove" in *index.txt*.

What is it?

An old music tracking trick which adds a more grooving, swinging feel to music without turning to a higher resolution is to change the song speed while the song is played... I thought it would be cool to have this automated, so I did it...

Since now you don't have to compute the swing amount by hand, you can change it in no time and thus write patterns with exactly twisted shuffle amount & type.

You can also switch back to the classic Master pattern editing by recording a master pattern while IGroove is running...

IGroove features an echo time dialog. You see a list of echo tick and beat durations in ms when you select "Echo Times" in the pop-up menu.

IGroove is *not a hack*, at least not in the worse definiton of the word, so you can assume it will work with future versions too.

Global parameters

BPM/TPB: The average speed for your song. Fractional bpm values are to make swinging and shuffling more precise.

BPM Inertia: 0.0001 to 1. 1 means no inertia, ie changes to the BPM slider take effect immediately; 0.0001 means it will take LONG until the new value is reached. The inertia algorithm is (still) DAMN BAD, I admit... :(but it works :)!

Swing: 0 to 16 ticks. how many ticks to wait between swing steps.

Length: 0 means swing is off; high values might give confusing results... You might use 1 or 2 most often in the beginning.

Shuffle: 0 to 75% - each "swing length" ticks, speed is raised then, after again "swing length" ticks, it is decreased (or vice versa????).

Random: 0 to 50% - the amount of "humanization" :)

Track commands (hex)

00: Sync swing (phase+counter) - every song should have this trigger somewhere. it syncs the counter and the phase of the swinging to your song.

01: Set low phase - sets the phase to the lower speed.

02: Set high phase - sets the phase to the higher speed.

04: Beat length -> BPM - sets the bpm slider according to the duration of a beat in milliseconds.

05: Tick length -> BPM - same as 02 but for the tick duration.

10: Switch off - gives control back to the master - switch on with command 11

11: Switch on - switches IGroove on.

12: Read tempo from Master - gets the bpm/tpb settings from Master. usually used when IGroove is switched off.

13: Set tempo in Master - sets the bpm/tpb settings in the Master.

Notes, Hints & Tips

Trigger a "00" command in every song!

You can make more groove modes with the 01 and 02 commands...

You can use IGroove to record patterns for the Master instead of using it directly. You can then remove it from your song and use the Master patterns for tempo and groove control - if you like.

If you don't want to shuffle the whole song, either change the shuffling throughout the song :) or use Hoester's GrooveBox... which is better if you want to shuffle just some tracks playing together with non-shuffled. Interesting effects can be achieved by using both together! 8p

Buzz machines behave differently when tempo is shuffled. E.g. for Ninja Delay, when you set a tick-wise delay while shuffle is = 0, you can then move the shuffle slider without the echo time changing. MTrk has two modes for looping samples, one is synchronized and the other one not.

The inertia feature is nice if you want to slow down/speed up a very fast section that is repeated via 2ndPLoopJumpHACK.

Legal Information

This small bunch of data is FREEWARE. Redistribute it as you like, but don't change it please.

If my software should cause any harm, directly or indirectly, I am NOT RESPONSIBLE. I can't be. Use this at your risk, though I promise you there is none :)

Contact

I can be reached via e-mail: malteschreiber@hotmail.com

19:14 06/12/2004

************************** intoxicat 8n - step sequencer ******************************

Thanks to: btd for peerlib + tutorial, advice and code snippets : usrusr for invaluable advice getting started : jmmcd for including the source code for his machines.

Info: This machine is a peer controller playing a sequence of 8 notes in a loop with various options for triggering and transposing.

Install: Put dll and this txt file in the Generators folder.

Usage:

Right-click on the machine and go to Assignments. Select the target machine note and volume paramter.

Set up a sequence of notes (and volume values) in the param view and trigger this sequence in the pattern view (or just press play in buzz).
The trigger also works as an offset so e.g. 05 will play the pattern starting at step 5.

Trigger options:

00-07 - trigger with sequence offset.
A0 - stops the playing of the sequence until next trigger
B0 - reverses play forcing the next note to be triggered.
C0 - triggers a randomly selected step number
F0 - triggers the next note and puts the sequencer back into forward play.

Add tracks to control more machines.
Each track has an independant trigger, step length, transpose, target track and note lenght.

Other track parameters:

Steplenght (ticks) - sets the number of ticks between each note.

Transpose (semitones) - duh - transposes the sequence for the target machine for this track

Target Track - sets the track to play the sequence on the target machine... Useful in combination with trasnpose for playing chords.

Note Length (ticks) - Sets the lenght of each note - works on all notes in the sequence.

Root Note - works in addition to the other transpose parameter so if you have transpose +2 and play a D4 the total transpose will be +4. Confusing? A little..

Tip: Combine this machine with PeerState to flip between different sequences. Endless fun.

Warning: does not have to be used with 303 clones! :) also the usual its not my fault if this machine breaks your pc... use at your own risk etc. blah blah

Comments, bugs and suggestions to: karl.nilsson at btinternet.com

enjoy!

Intoxicat Asynchronous Cloud Generator.

Type: Generator. (put dll and this htm in Buzz\Gear\Generators\)

This machine brings some of the functionality of asynchronous wavetable granular synthesis to buzz. Could be likened to Audiomulch's Bubble Blower or Curtis Road's Cloud Generator.

Thanks to all helpful devs that make their source codes and advice freely available. Particular thanks to: BTD, Fuzzpilz, Zephod and Cyanphase (whose tutorial I still refer back to.).
If anyone is interested in making their own granular buzzmachines please let me know and I will be happy to share the source for ACloud and give a hand if requested.

I will naturally fix bugs if I can but this could only happen if they are reported. Please use: kjnilsson [at] gmail.com to do so.

Thanks to all those who beta tested as well.

Parameter	Description	SLIDER
Note	Triggers the machine and sets the default playback rate of the wave file(s).	No
Seed	Sets the seed for the random generators. Useful if you want to ensure that the output sounds the same every time you play it.	No
Wave 1	Sets the wave to use for slot 1.	Yes
Offset Index	Sets the offset position in wave 1 to start generating grains from.	Yes
Offset Range	Sets the range of possible index positions that ACloud will use for new grains. The range is calculated from the Offset Index position. Whether it is applied before of after depends on the Skip Mode parameter.	Yes
Wave 2	Same as for 1	Yes
Offset Index	Same as for 1	Yes
Offset Range	Same as for 1	Yes
Slave Offset 2	Sets whether the Offset parameters for Wave to are slaved to those of Wave 1.	Yes

Skip Type	Set the way ACloud skips through the wave file(s) selected. 0 Selects a random position within the offset range. 1 moves forward from the index position. 2 moves backwards from the index position. 1 and 2 can be used for grungy time stretching.	Yes
Skip Rate	Sets the rate at which Skip Types 1 and 2 moves through the wavefile.	Yes
Wave Mix	Sets the likelihood of either wave slots being selected for the next grain.	Yes
Divider	Divider — not used for anything but dividing the wave parameters from the grain parameters.	
Duration	Sets the base duration of grains	Yes
DurationRange	Sets the range of possible grain durations starting from the base duration point. If this is lower than the ‘Duration’ parameter it has no effect.	Yes
Grain Amp	Sets the range of possible amplitudes applied to new grains	Yes
Rate	Sets the rate (pitch) of new grains. Is applied in addition to the base rate supplied but the Note parameter.	Yes
Rate range	Sets the amount of randomness applied to the rate of the new grains in semitones.	Yes
EnvAmount	Sets the steepness of a simple AHD envelope that is applied to new grains. If set to the maximum setting it will apply a triangle shaped envelope.	Yes
EnvSkew	Sets the centre point of the envelope.	Yes
PanL	Sets the leftmost possible pan position of new grains.	Yes
PanR	Sets the rightmost possible pan position of new grains.	Yes
Divider	Another Divider	
Density Mode	0 = Average number of grains per second. 1 = Perceived Density. I.e. the length of the grains will affect the density so that the shorter the grains are the denser the cloud will become in order to keep the perceived density the same.	Yes

Density	Sets the density of the cloud. If Dens Mode is 0, density will set the average number of grains per second. If Mode is 1, it sets the perceived density.	Yes
Max Grains	Sets the maximum number of simultaneous grains.	Yes

Remember this is a BETA... maybe alpha ;)

so, as warned, don't use it in song, or suffer
the consequences *g*. Make sure you do something
before updating to this one. This is still a beta,
just an update for ppl to see what's going on.

Wut's sorta nu?

* Main filter and Note-pitch controlled filters (1 is
  nothing, 2-6 is at note's octave, 7-10 is a octave
  higher than note). I may add a 3rd octave later for
  those who like more treble.
* A second OSCM.
* Much more options with what to do with OSCM, like FMing
  the FM.
* bug fixes, and more bugs *g*
* Hmm, it's not done yet

And some ppl wonder what's the Bitsaturates for... it's
sorta like an aural enhancer dist thing (then again I
just it made it up). with some settings it sounds sorta
more "crystally". hehe.

What's to come?

* Faster... hopefully.
* More filters, like direct ii form and maybe chebyschev
  and whatever else that sounds cool.
* speed...
* Getting rid of the duplicate EvenSines (only EvenSine 2 and
  Inv. EvenSine 2 will stay), adding more waveforms.
* More Vibratypes, once I can make some more up.
* Maybe a DC blocker thing.

IX Peer Accumulator.

Installation

Copy "IX Peer Accumulator.dll" and this document to your Generators folder. That's it. You could add it to your index as well if you like.

What does it do?

It's a trigger style peer controller, mostly intended for adding controllable pseudo-random texture to parameters but you can send fixed values too.

Pseudo-Random?

Each track has it's own independent pseudo-random number generator which will happily spit out seemingly random values all day long. The values are not truly random however but are generated from a seed value, and for a given seed the number sequence will always be the same. You can control the generator's state with various commands that allow you to do things like re-seed the generator or rewind the number sequence to it's start point. When you add a track, the new track's generator is seeded with the current time.

Why "Accumulator"?

It started as a simple machine that would send a peer value only after it had been triggered a certain number of times. It still does that, but I got a bit carried away with the random side of things.

Global Parameters

Command 1	See below.
Argument 1	Value for Command 1. See below.
Command 2	See below.
Argument 2	Value for Command 2. See below.

Track Parameters

Trigger	Trigger, possibly.
Delay	Delay trigger by n subticks.
SubTicks	Sub-tick resolution for trigger delay. Default value is 6 subticks per tick.
Threshold	How many times this track must be triggered before it does something.
Mode	What to do when the track actually triggers.
Val A/B	Value controls.
Track	Which track of the target machine to use.
Command 1	See below.
Argument 1	Value for Command 1. See below.

Command 2	See below.
Argument 2	Value for Command 2. See below.

Triggering Modes

Random	Successful triggers send a random value in the range between A and B.
Use A	Successful triggers will send value A. Useful for tuning the value range.
Use B	Successful triggers will send value B. Also useful for tuning the value range.
Alternate	Successful triggers will alternate between the values of A and B.

Commands

There are several commands you can use to control Accumulator, either on a global or per-track basis. If you use the global command columns, the command will be applied to all existing tracks. Where two commands on the same line conflict, Command 2 has precedence over Command 1 and Track commands override Global commands. Commands are processed before triggering occurs.

00 - Reset	No argument needed. Resets trigger counter, clears all flags and restarts the number generator.
01 - Set Counter	Set trigger counter to argument or reset counter to zero if no argument supplied.
02 - Set Toggle	In Alternate mode, sets toggle phase to A or B. Use zero or no argument for A, non-zero for B.
03 - Set Offset	Sets the random number sequence to the specified step. Use zero or no argument to reset the generator.
04 - Re-seed	Seed random number generator with argument. An argument of zero seeds the generator with the current time. If no argument is supplied, the generator is restarted (same as Set Offset to zero).
05 - Ignore Delay	A non-zero argument causes the track to ignore any specified subtick delay for subsequent triggers. An argument of zero or no argument clears the flag.
06 - Override (IT)	Override (Insert, Temporary). See below.
07 - Override (RT)	Override (Replace, Temporary). See below.
08 - Override (IP)	Override (Insert, Permanent). See below.
09 - Override (RP)	Override (Replace, Permanent). See below.
0A - Clear Override	Clears all override settings for this track. No argument needed.

0B - Invert	Invert values. Random numbers will be inverted (ie. 100% becomes 0%). Non-zero to activate, zero or no argument to deactivate. Override values are not affected.

Override command modes:

The override commands allow you to override the normal value of a trigger. If no argument is supplied, the value is that of the last successful trigger (ie. hold). The override commands behave in the following ways:

Temporary	Overrides only the next successful trigger.
Permanent	Overrides all subsequent triggers until the flag is cleared by either a temporary override, 'Clear Override' or 'Reset' command.
Insert	Inserts the value into the normal value sequence. Subsequent values are shifted along one place ie. the sequence will resume where it was interrupted.
Replace	Replaces the value at this point in the sequence. Subsequent triggers retain their values.

Attributes

MIDI Input Channel	Channel to monitor if MIDI filtering is not enabled. Off by default.
MIDI Trigger Base	The MIDI note that will trigger the first accumulator track.
Reset On Stop	If set, perform a global Reset command when stop button is pressed. Off by default.

Right-click menu

Assign	Manage peer assignments.
Style	Controls how values A and B will be displayed in the parameters window. This is just an aid for setting up values and has no effect on the actual function of the machine. Please note that although it should work correctly for most machines, it is possible that the 'Note' mode may not accurately a machine's interpretation of the values sent by Accumulator.

Acknowledgements

First and foremost, many thanks to Ed Powley for the superb PeerLib upon which this machine is based, and for his seemingly endless patience when faced with stupid noob questions. I have also used the Mersenne Twister random number generator, in the form of MersenneTwister.h by Richard J. Wagner which is in turn based upon code by Makoto Matsumoto, Takuji Nishimura, and Shawn Cokus. Many thanks to those distinguished gentlemen for making their code publicly available to be used by amateurs such as myself. Thanks also to the many developers who frequent The Church whose help, advice and code I have leaned on heavily from day one. Finally, huge thanks to Oskari for making, and now at last re-making, our beautiful, beloved Buzz.

Contact

If you've got any comments, requests, bug reports, whatever, you can find me lurking in The Church (username 'd9') or you can mail me via deenine[at]hotmail[dot]co[dot]uk (but don't expect a quick reply.)

Disclaimer

This is the second machine I've made and I'm still strictly amateur so use it at your own risk. If it kills your computer, take comfort in the fact that it'll probably kill mine too.

- IX

to install jacinth, just put all the files included in the zip into
your buzz\gear\generators directory. some people asked me to include
an installer, but i decided not to do that because i don't feel it's
required. no other machines have an installer either, so why should
jacinth?

gentab.exe generates some internal tables for jacinth. if you're on a
slower computer, you might want to run it and put the generated files
into your buzz\gear\generators directory with the jacinth dll and the
other .jac files. OTOH jacinth isn't that useful on slower computers
because it is so slow already :)
the precalculated files ONLY speed up adding the first instance of
jacinth to your buzz song. the data is only generated or loaded once.

NOTE ABOUT PRESETS: (read this, if nothing else)

when trying out presets, be sure to READ THE PRESET DESCRIPTIONS.
jacinth has a built-in waveform editor, and the waveforms can not
be saved with the presets! if a preset is using custom waveforms,
it should be mentioned in the preset description!
(the description can be viewed by clicking on the Edit... button
in the machine parameter window. that is, the window opened by
double-clicking on the machine.)

so now you've found a preset that is using a custom waveform. how to
load those? easy:
1. right-click on the jacinth machine box in buzz machine view, and
   select Waveform editor...
2. from the File menu, select Load Library...
3. find the correct jacinth waveform library file (name should be in
   the preset description)
4. in the Load Library window, click Default to load all the waveforms
   to the slots they used to be. then click OK.
5. click Close in the main waveform editor window.

now the waveforms in that library are loaded. all the presets using it
should sound as they're supposed to. loading custom waveforms does not
affect jacinth built-in waveforms - those never change.

ALSO:

because of the way jacinth is built, you should always have at least
two or three tracks added before loading presets. otherwise the preset
might not be loaded completely.

 - ld0d

Jeskola Tracker by Oskari Tammelin
Jeskola Tracker UPDATE V2 by Kurt B. Pruenner

01 - Volslide Down
02 - Volslide Up
03 - Porta Down
04 - Porta Up
05 - Tone Porta
06 - Vibrato
0A - Wave Offset
0B - Detune
0C - Note Cut
0D - Note Delay
0E - Retrig

Well, this is yet another update for the Jeskola Tracker, as my earlier fix broke the retrig command, and as there still were pops, clicks and other noise at some times... So just unzip this to Gear/Generators in you BUZZ dir and BUZZ away... :)

(And don't forget to adjust the "Ramp Time" attribute to your needs...)

Coming up next: Extra effects...

One suggestion was an "extended retrig" á la FT2, where the volume is changed with each retrig... Also, I'd say "fine" volume/pitch slides (especially the latter ones) would be nice, no? If you have more suggestions, feel free to mail me...

--
Kurt B. Pruenner   kurt.pruenner@jk.uni-linz.ac.at
Haendelstrasse 17   http://wildsau.idv.uni-linz.ac.at/~k30a2e7/
A-4020 Linz/AUSTRIA Web-Admin of http://www.hermes-gfx.gup.uni-linz.ac.at/

np: Kenji Kawai - Ghosthack (Ghost In The Shell OST)

BUZZ / BUZZ 2 http://www.buzz2.com/
WARP Records   http://www.warp-net.com/
Aphex Twin   http://aphex.base.org/
Underworld   http://www.dirty.org/

¬¬ name
jmmcd Peer Meta-I (copyright © James McDermott 2004)

¬¬ purpose
controlling "layers" of synthesizers; triggering envelope controllers automatically;
controlling several things from one place; triggering pitch-shiftable delay lines.

¬¬ installation
put the .dll and the .html in your gear/generators directory, and the .bmx in your
recycle bin. the rest is for developers.

¬¬ menu options
¬¬ layers: assign the note, and optionally the volume, note-length, and slide-note
parameters of generators. notes entered in the Peer Meta-I patterns will be sent to
these generators. delayed notes can also be sent to these generators. a note-off in a
track in a Peer Meta-I pattern causes a note-off to be sent to the assigned
generators, and any delays currently running in the track to be removed.
¬¬ triggers: assign any parameters of any machines. every note-on in the Peer Meta-I
patterns will cause an on-value (1 by default) to be sent to these trigger parameters.
a note-off will cause a 0 to be sent. an on-value will also, by default, be sent on
delayed notes (but see attributes). this feature was originally intended to send a 1
to (eg) Peer ADSR; but some machines, eg drum-machines, can use volume-values as
triggers - so you can specify the on-value to be sent using the TrigVal parameter.
¬¬ controls: assign any parameters of any machines. these parameters can be controlled
from the Peer Meta-I patterns in the normal way, possibly allowing you to delete some
machine columns in the sequencer view.
¬¬ about: instantly regret the presence of an about box.

¬¬ global parameters
¬¬ length: the length of the delay line. 0 indicates no delay.
¬¬ feedback: the factor by which volume is multiplied at each new delayed note.
¬¬ threshold: the volume level below which the delay ceases to generate new notes.
¬¬ shift: the pitch shift, in semitones, applied to each delayed note.
¬¬ inertia: the time, in ticks, it takes one of the para* parameters to reach a new
value.
¬¬ trigger value: the value to send to trigger parameters on note-on.
¬¬ para*: the values sent to the assigned parameters.

¬¬ track parameters
¬¬ note: the note sent to each assigned generator.
¬¬ volume: the value sent to the volume parameter of each assigned generator.
¬¬ flags: the subtick-delay and -retrigger values. the first byte (0-F) indicates
delay. the second byte (0-E) indicates retriggering. for example, "23" indicates that
the note should be first triggered after 2 subdivisions, and retriggered every 3
subdivisions after that. note that the number of subdivisions per tick is set by an
attribute with default value 12.
¬¬ note length: the length of note to be played. many machines don't have note-length

parameters - in these cases, just don't bother assigning the parameter. note that the target machine is responsible for interpreting the value - so one machine might give you a 4-tick note when sent "4", while another might give a note 4% of the maximum length, or something. if the target machine doesn't have a note-length parameter, but does have a sustain-length parameter, you can control that from here instead.
¬¬ slide note: the note to slide to or from. again, many machines don't have slide-note parameters, and those that do don't all interpret them in the same way.

¬¬ attributes
¬¬ number of subtick divisions: see flags above.
¬¬ trigger on delayed notes: when set to 1, an on-value will be sent to the assigned trigger parameters on every note and every delayed note. when set to 0, nothing will be sent on delayed notes.

¬¬ usage hints
you don't have to use all the features at once! if you just want pitch-shiftable delays, just assign a single layer and set the delay time. if you just want to trigger envelopes with Peer ADSR, just assign a single trigger parameter.

tracks don't work the same way in Peer Meta-I as they do in most peer machines. adding a new track doesn't allow you to control an extra machine. instead, adding an extra track just allows you to play more than one note at the same time - though you have to add extra tracks to the target generators too of course.

when assigning generators, you can assign things to the volume, note-length, and slide-note parameters which aren't volumes, note-lengths, or slide-notes. it's up to the target machine to interpret them!

it's possible to set the delay section up so that some notes from your tracks coincide with notes generated by the delay lines - so that two or more notes try to play through the same generator at the same time. this might be a major source of clicking, non-deterministic melodies (a feature or a bug, depending on your point of view), wasted cpu, and "missing" notes which are cut-off by other, very quiet notes. avoid all these problems by deciding how many delays you want and adjusting the length, feedback and threshold accordingly.

¬¬ known bugs
i don't know anything about midi. does this machine work with midi? should it work differently?

any suggestions or comments are welcome.

¬¬ source code
the source code for this machine is included for educational and collaborative purposes. feel free to work with it. if you make an improvement to the existing machine, please contact me so we can integrate our changes and make an official release. if you want to base a new machine on this code, go ahead (but release the source code for that too). avoid passing off my work as yours, or vice versa.

a small addition was made to btdsys' Peer Library to enable sending note, volume, note-length, and slide-note all at the same time. contact me if you'd like to talk about it.

¬¬ thanks

thanks to oskari, cyanphase, btdsys, and all the devs for making buzz what it is. thanks to btdsys, 7900, usr, and kodream for answering peer questions. thanks to everyone who has released source code. but mostly, especially, many thanks to btdsys for the peer library: it's the best thing that ever happened to buzz.

if you like this machine, and want to thank me for making it, please go and thank other more important developers first. when you're finished, if there is still gratitude in you, you can send me a cd of your music, or a postcard, or an email (jamesmichaelmcdermott@eircom.net), or you can visit www.skynet.ie/~jmmcd, where you'll find out about my band, the moral majority. listen to the songs there and tell me how good they are.

¬¬ name
jmmcd Peer Note-Pool (copyright © James McDermott 2004)

¬¬ purpose
sending note-on messages to generators based on crude probabilistic algorithms.

¬¬ installation
put the .dll and the .html in your gear/generators directory, and the .bmx in your recycle bin. the rest is for developers.

¬¬ menu options
¬¬ generators: assign the note and optionally volume parameters of one generator for each track in the Peer Note-Pool patterns.
¬¬ about: instantly regret the presence of an about box.

¬¬ global parameters
¬¬ prob([C-B]): the *relative* probability that each pitch will be chosen, when the machine is choosing a pitch for you.
¬¬ prob(Off): the note-off message is treated just like any other note.

¬¬ track parameters
¬¬ note: the note sent to each assigned generator.
¬¬ volume: the value sent to the volume parameter of each assigned generator.
¬¬ probability: the probability that a note will be played this tick.
¬¬ centre note: determines the octave in which the notes will be played.
¬¬ octave deviation: determines the amount of randomisation applied to the octave.
¬¬ volume deviation: determines the amount of randomisation applied to the volume.
¬¬ dot-prob: the probability that a note will be played on an empty tick.
¬¬ target track: the generator track to which notes will be sent.
¬¬ on/off: set to on to allow improvisation.

¬¬ attributes
none

¬¬ usage hints
there are a few ways of using this machine. the laziest (and my personal favourite) is to set the note probabilities to correspond to the key i'm working in, turn the dot-prob to a suitable value, and let it play while i work on something else.

if you prefer, you can specify most of your patterns exactly, and allow the machine to improvise only a couple of notes here and there. if you know, for example, that you need a note on tick 20, but don't want to decide what one, set the probability (not the dot-prob) to 100% on that note.

or you can do something in between.

¬¬ known bugs

i don't know anything about midi. does this machine work with midi? should it work differently?

it should be possible to record the notes and volume values sent in the generator's tracks. this *was* working (it should work automatically, without help from the programmer) but it's not now. any ideas?

any suggestions or comments are welcome.

¬¬ source code
the source code for this machine is included for educational and collaborative purposes. feel free to work with it. if you make an improvement to the existing machine, please contact me so we can integrate our changes and make an official release. if you want to base a new machine on this code, go ahead (but release the source code for that too). avoid passing off my work as yours, or vice versa.

a small addition was made to btdsys' Peer Library to enable sending note and volume data at the same time. contact me if you'd like to talk about it.

¬¬ thanks
thanks to oskari, cyanphase, btdsys, and all the devs for making buzz what it is. thanks to btdsys, 7900, usr, and kodream for answering peer questions. thanks to everyone who has released source code. but mostly, especially, many thanks to btdsys for the peer library: it's the best thing that ever happened to buzz.

if you like this machine, and want to thank me for making it, please go and thank other more important developers first. when you're finished, if there is still gratitude in you, you can send me a cd of your music, or a postcard, or an email (jamesmichaelmcdermott@eircom.net), or you can visit www.skynet.ie/~jmmcd, where you'll find out about my band, the moral majority. listen to the songs there and tell me how good they are.

****
* kejo Perlin Noise Generator *
****

programming: Erik Olofsson
build date: 2002-06-19
version: 0.9

****
* Note to user *
****

This is beta software, meaning not completed (probably never will).
Judge this generator from that fact. It is not compatible with 0.8,
due to new parameters.

It is now possible to play with AM/FM functionality and the practical
ADSR-envelope, producing quite fascinating noises.

Have fun, be creative, and don't forget to feedback!

(se 'about...' for contact info)

...

kibibu Doof 1.5 BETA 2

This is a beta, use at your own risk
(if it blows up then let me know though)

This is a newer version than the IRC release one

A couple of points -

There is only one filter, and it is a simple RBJ one (copied from Cyanphase's excellent buzz doc)
To be added are a moogley foogley, HP and something else when i think of it

I am interested in opinions about useability and control.

Issues that WILL be resolved:
- Loading time will be dramatically shorter
  It currently generates waveforms by brute force in a really slow way
  It takes a while on my p4, so it might take extra long on others
  Going to change this to complex exponential

Please email me your thoughts - i'm hardly ever on IRC - and presets too if you make anything funky (or stick em on buzzmachines)

I think the parameters are final now, at least for the functionality this version will have.
The only thing that MAY change is the number of waveforms. I don't know if this breaks compatibility or not.

Advantages over Doof 1.0:
- A filter
- Two waveforms
- Almost NO aliasing! (take that oomek)
- Filter pops

Disadvantages (reasons why old doof is still ok)
- More CPU (due to filter and antialiasing)
- Different waveform - gone is that waveform that just begs to be saturated

You can simulate the old fat waveform by:
 picking the saw
 setting its filter const to 0
 setting its track to something around 2.0
 playing with the res.

Still won't sound the same though

Also, when this one is finished, i'll release the source - as I kinda lost the source to doof 1.0 and I don't want that to happen again (actually, i upgraded it into this one, but i had no backups)

Cheerio!
Cameron Foale (kibibu)

cam@kibibu.com

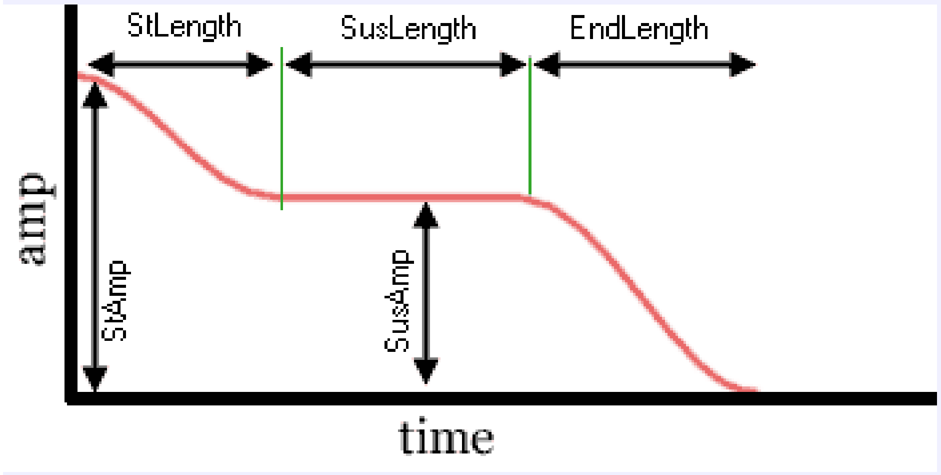
kibibu doof

kick and tweet generator

cameron foale - www.kibibu.com

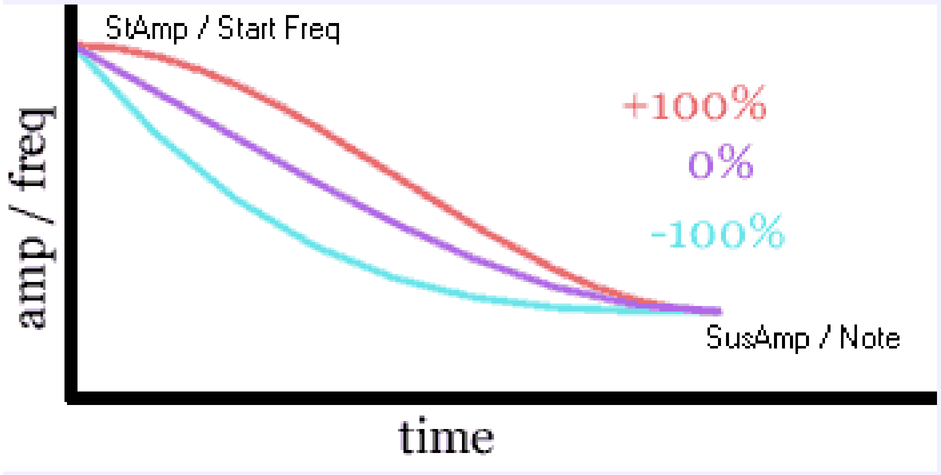
the **kibibu doof** is a generator primarily designed for kick type sounds, but it can also do higher pitched stuff.

It generates a sound in 3 segments, creatively called start, sustain and end. The start and end segments use Bezier "curves" to control both amplitude and frequency of the generated waveform, like this:



This particular envelope is for amplitude. Amplitude is always 0 at the end.

A "sharpness" control changes the slope of the Bezier as follows:



The Sharpness parameters shift the position of the 1st Bezier control point up or down. 100% means the control point is the same value as the start, and -100% is the same as the end of the curve.
In the end segment, the parameter shifts the 2nd control point (ie the one nearest the end), not the 1st.
This is to allow the envelope to join smoothly at either end of the sustain segment.
The machine itself allows sharpness to go to +/- 200%, which lets the envelope go higher or lower than

either of the two ends.

The StartFreqConst and StartFreqTrack control the start frequency. Essentially, the frequency of the note is multiplied by StartFreqTrack and added to StartFreqConst:

```
StartFreq = StartFreqConst + StartFreqTrack
```

Similarly with then end one:

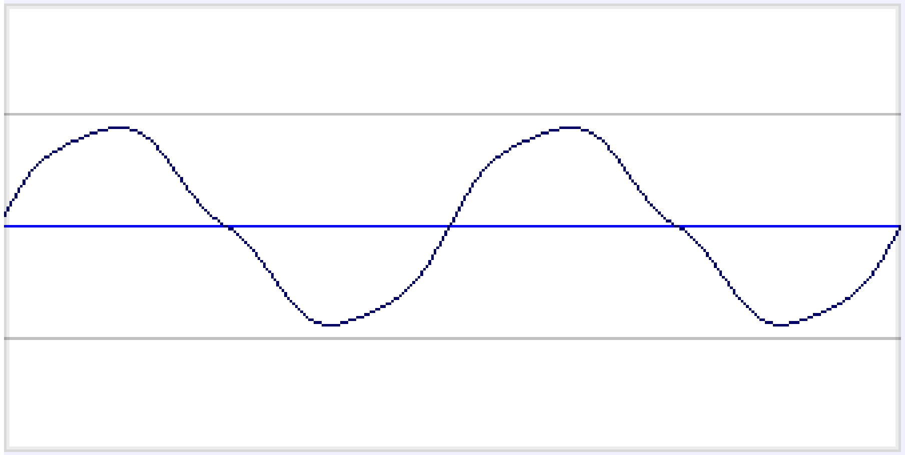
```
EndFreq = EndFreqConst + EndFreqTrack
```

To set either to a fixed frequency, set the track param to 0 and adjust the const. To set them to simply the note freq, set track to 1x and const to 0.

The waveform used is a sauced-up sine wave. Specifically:
(assume freq is multiplied by 2 pi properly)

```
signal(t) = sin(t * f + (sin(t * f * 3) * 0.123 ) + (sin(t * f * 5) *
0.0123 )
```

which is just a sine FMed by two odd harmonic sines. They are harmonics cause the thing is in a wavetable, (which is why in the 9th octave there is some ugly aliasing), and non harmonics would have created saw-wave like effects
It looks like this:



To help prevent clicking, the phase of the wave is not reset and amplitude is filtered when a second doof is triggered while another is still playing. Phase is reset when the signal returns to zero at the end of the end stage.

New in this version:

- **An attribute to control note-off behaviour.**
  This was added as a fix for the bizzare stuff that was going on with the note-off. There are now 4 note off behaviours, which are worth explaining (see the file kibibuDoofNoteOffDemo.bmx for examples)
  0. kibibu Original Style, straight cut, with frayed, aliasy note-offs.
  1. Instant Cut-Off, now with extra clicks.

2. Straight to start of end stage, regardless of current position. Note that this will reset to the start of the end stage *even if you are already in the end stage*. Can be fun to play with
3. The New Default: straight to end stage, unless you are already there.

- **An attribute for future fixing of short (<70ms) note off stages.**
  Doesn't actually work at the moment, but the attribute is there anyway
- **A particularly uninformative "about" box**

kibibu Green Milk

Multi-oscillator Subtractive Synth

Params

Oscillator Properties

Each oscillator has a pitch LFO built in. The combinations of pitch fluctuations for each oscillator leads to a unison-type effect.

Wave1, 2 & 3
: Select the waveforms used by each oscillator. These are interleaved - that is Waveform1 selects the waveform for oscillators 1,4,7,10,13 and 16, Waveform2 selects for oscillators 2,5,8,etc.

UnisonDepth
: controls how much the LFO affects the oscillator frequency

Unison Min S
: First oscillator's LFO speed

Unison Max S
: Last (that is, 16th) oscillators LFO speed. All other LFOs are interpolated between these the first and last

Unison Wave
: The LFO shape for all oscillators

Oscillators
: The number of oscillators to use. More oscillators use more CPU. Try to find the smallest number of oscillators that still achieves the effect you want

Chord Shape
: Each oscillator's pitch can be offset according to a chord shape. In most cases, these are interleaved similarly to the Waveform selection. Chord shapes starting with J use Just Intonation, in the appropriate minor or major scale

Amp

Amp A, Amp D, Amp S, Amp R
: Control the VCA envelope. Lookup ADSR at Wikipedia or something if you don't know what these do. The envelopes used in Green Milk are exponential rather than linear

Filter

The filter is a quad-sampled state-variable filter using an effective 10-tap decimation filter. Post-filter distortion (see later) is handled during the upsampled stage to reduce aliasing.

Filt Cut
: Controls the filter cutoff

Filt Res
: Controls the filter resonance

Filt Env
: Controls how much the envelope affects the cutoff. If the Intelligent Cutoff attribute is set and you drop this into the negatives, the cutoff is increased to compensate

Filt Mode
: There are 2 filters, both of which may be lowpass, highpass, bandpass or notch. The various modes are Single, Dual (serial), Parallel, Separate (2nd filter has 50% cutoff offset) and Wide (2nd filter has 150% cutoff offset)

Filt A, Filt D, Filt S, Filt R
: Filter (cutoff) envelope. If you want to do filter slides, sweep the sustain value here, with inertia controlled by the decay length

Global Timing

Length Scale
: Scales the length of the Amp and Filter envelopes (not LFOs).

Distortion

Both distortions have 4x oversampling. The upsampling is done using a windowed-sinc function. You'll still get some aliasing with the pre-filter distortion, especially in the upper octaves. The post-filter distortion is a little cleaner.

PreF Dist
: Saturating distortion before the filter.

PreF Dist
: Saturating distortion before the filter.

LFOs

Two LFOs that operate per-track.

TLFO Speed
: How quick the LFO goes. Your results may vary (or just suck) at very high frequencies, as the LFOs are updated every 32 samples

TLFO Delay
: How long before the LFO kicks in. As Mute says, "BOOOOOOW...

WOWOWOWOWOWOW"

TLFO Shape

The LFO wave shape. There are a bunch of arpy shapes in here, useful for modulating the pitch. Set TLFO->pitch to +12 or -12 for in-tune results

TLFO->Cutoff

How much the LFO plays with the cutoff

TLFO->Res

How much the LFO plays with the resonance

TLFO->Pitch

How much the LFO pitch shifts all oscillators

Retrig Mode

Which LFOs get reset when a new note occurs

Track Params

Note

The standard 12TET buzz-style note

Velocity

Adjusts amp sustain level

Slide

Slides to the newly entered note over a period of time

Command1, Command2

Commands. Command1 is evaluated first

Cmd1Arg, Cmd2Arg

Arguments to the two command columns

Commands

01	Restart Amp Envelope
02	Restart Filter Envelope
03	Restart Both Envelopes
04	Set Amp Envelope Value
05	Set Filter Envelope Value
0A	Portamento Up (xx = semitones/16, yy = time in same units as slide)
0B	Portamento Down
10	Randomise Unison Oscillator Phase
11	Synchronise Unison Oscillator Phase
12	Randomise Unison LFO Phase

13 Synchronise Unison LFO Phase

14 Randomise Unison Pitch Offset (xx=range, y.y=interval)

15 Set Unison Pitch Offset

16 Set Unison Oscillator Phase

17 Set Unison LFO Phase

18 Set Unison Depth

19 Set Unison Speed (ticks/16)

20 Ignore new global parameters (this tick and this track only)

30 Track LFO 1: Restart (at start of delay)

31 Track LFO 1: Pause

32 Track LFO 1: Resume

33 Track LFO 1: Skip the delay

34 Track LFO 1: Set wave (this track only)

35 Track LFO 2: Set phase (xxxx = percent)

36 Track LFO 2: Set frequency (xxx.x ticks)

40 Track LFO 2: Restart (at start of delay)

41 Track LFO 2: Pause

42 Track LFO 2: Resume

43 Track LFO 2: Skip the delay

44 Track LFO 2: Set wave (this track only)

45 Track LFO 2: Set phase (xxxx = percent)

46 Track LFO 2: Set frequency (xxx.x ticks)

visit http://www.kibibu.com

leave feedback at the church

kibibu PeerTune

Will adjust the tuning of any synth that has individual note tuning attributes.

To use:
- Add a PeerTune as well as any tunable synth (say, kibibu Green Milk...)
- Right click on the PeerTune machine, and select the tunable synth from the "Apply To" menu
  (Only tunable synths will appear)
- Tweak the sliders, or load some presets.
  (The entire scala archive with <= 12 tones is available)

Note that tuning may only apply to synths when the note is triggered, although this behaviour is up to individual synth developers. Green Milk, for example, only calculates tuning when notes are triggered, and will not "retune" an already playing note.

You can target other PeerTunes too, but it will only apply the Detune param to the Detune Offset attribute of the other PeerTune. This isn't a feature you need to bother with unless you want to detune a bunch of (individually tuned) PeerTunes at once.
You can't make cyclic chains (eg where PeerTune1 controls PeerTune2, and PeerTune2 controls PeerTune1) so go nuts!

Tip: If the presets don't make any sense to you, there are comments associated with each one.
     Open up the Edit... button from the parameter view.

--- For devs ---

Its pretty easy to add microtuning to your own synths. PeerTune works with attributes in a kind of "offset from standard note" mode.

Create your tuning attributes to allow values between 0 and 24000, with 12000 meaning "no shift". Call them "C offset", "C# offset" or similar. As long as the first 2 letters represent the note (including the space for single char notes!) you'll be fine.

```
  CMachineAttribute const attrC = { "C offset (cents/10)", 0,24000,12000 };
  CMachineAttribute const attrCs = { "C# offset (cents/10)", 0,24000,12000 };
  etc...
```

Green Milk and Matilde handle this by having an array of 12 "note pitches". The following function sets up note pitches from attributes (if the attributes are in aval.notes[])

```
void mi::AttributesChanged()
{
	// update the built-in pitches
	for(int i = 0; i < 12; ++i)
	{
		// note the -12.0f subtracts the 1 octave offset
		notePitches[i] = float(i) + (float(aval.notes[i]) / 1000.0f) - 12.0f;
	}
}
```

The following functions may be used to convert a buzz note to a "note number"

```
float mi::mapNote(int octave, int note)
{
	return ((octave * 12.0f) + this->notePitches[note]) - 12.0f;
}

float mi::mapNoteNum(byte buzzNote)
{
	byte octave = buzzNote >> 4;
	byte note = (buzzNote & 0x0F) - 1;

	return mapNote(octave,note);
}
```

The float "note number" you get can be converted to a real world frequency (in Hz) by:

```
float frequency = (16.3516f * powf(2.0f, num/12));
```

--- Detune Attribute ---
You can also add another attribute starting with "Detune", which is a single semitone detune.

```
  CMachineAttribute const attrDetune = { "Detune Offset (cents)", 0,2000,1000 };
```

PeerTune will use this rather than detuning all the individual attributes, if it exists.
However, its entirely optional, and peertune will detune your stuff fine without it.
This attribute is how PeerTune adjusts other PeerTunes.

In this case, the mi::AttributesChanged looks more like

```
void mi::AttributesChanges()
{
	for(int i = 0; i < 12; ++i)
	{
```

```
            // note the -13.0f subtracts the 1 octave note offset
            // and the 1 semitone detune offset
            notePitches[i] = float(i) + (float(aval.notes[i] + aval.detune) / 1000.0f) - 13.0f;
      }
}
```

--- REMEMBER! ---
The attributes must start with capital note letters, and MUST have AT LEAST 2-character names
"C " is ok. "C Offset (cents/10)" is better.

The detune attribute, if it exists, MUST start with "Detune" (case is important).

Also note that PeerTune will generate attributes outside of the min/max range you specify.
Don't be surprised with attributes with negative values. The code above doesn't care about ranges,
and will work fine. If the idea of out-of-range attributes scares you, just set the minimum to -12000
and the max to 36000 or thereabouts. Only the default (12000) is important.

Thanks to Btd for allowing others to pick through his code.
This doesn't use the peer control library, but I borrowed Btd's hack code for tweaking attributes.

Cameron Foale (kibibu)
http://www.kibibu.com

KoDream's Improv Generator

The Idea:

Formal languges can be used to make music. A formal language is a language which every sentence can be generated from a set of rules. Each of these lines was a rule, the symbol "->" is pronounced "goes to", and the vertical bar is "or",

S -> verse bridge break chorus break verse

verse -> A B C B A | A B C B A

bridge -> thump thump thump thump

break -> dri ri ri ri ri ri ri ri ri ri ri ri lllll

ri -> ro | ti

chorus -> la la hey

Given the rules above a possible sentence is:

A B C B A thump thump thump thump dri ti ro ti ti ro ro ti ro ti ro ro ro lllll la la hey dri ti ro ro ro ro ti ti ti ti ro ro ro lllll A B C A B

With Improv each of the above symbols would correspond to a number that will be used to control a machine's parameter, and the sentences are generated randomly. This is a type of grammar called an acyclic definite clause grammar, so no cycles(such as S->S you have been warned).

Install:

To install place the .dll in your buzz generators directory. Place the sample grammar in your buzz directory.

Using:

To use the sample grammar. Open buzz, create a new generator, such as M3 or voidlead, then connect the new machine to master. Create an improv machine, in the improv's pattern editor create a pattern with a trigger value of 1 somewhere in it, play the pattern in the sequencer. Now in the machine view, right click on improv, then select Assign Machine Edit Grammar.

This should bring up a pop up window. In the pop up window to load a grammar file from the buzz directory, type its name in the "file:" edit box and click load. To save it to the file name in that edit box click save. Now from the machine combo box select the machine which you wish to generate a melody/rhythm for. Then click assign. Click OK, then play the melody.

Creating your own rules, first select a rule in the rule combo box. The rule "S" will always be the first rule used to produce a sentence. Now type a string of symbols in the production edit box, then click add production. The production will appear in the possible productions for the selected rule. If the symbols have not been defined they will appear in semantics list with an integer next to them. If the symbols have already been defined as rules, they will not apear on the right.

Any unseen symbol is intially assumed to be max int which corresponds to a Note Off or a rest if you will. To change this select the note in the semantics list, and in the note box underneath, type an integer value. Then click update note.

Deleteing a production on the right will delete itself, and any symbols which are not referenced by another rule in the tree of possible sentences.

Credits:

This machine would not be possible withou the code provided by BTDSys and CyanWerks.

Contact:

robin@cdoc.net, for bug reports and suggestions. As always save early and save often.

Enjoy!

KoDream

Kozaluss Angeluss - GenB 1.6

GenB is 4x cascade FM/Div4 multitrack generator.

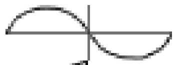
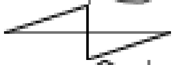
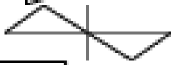
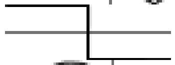
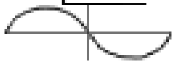
Some parameters description:

- *MODS USED* describes how many modulators should be used.
- *GLIDE* describes the speed of sliding to next note.
- *PGLIDE* describes the speed of sliding to desired pitch.

Attribute *MIDI/pattern NOTE OFF ACTION* tells machine what to do, if NoteOff event happened (either from Midi or from Pattern):

- 0. Does nothing. Just plays note as long, as ADSR is set to.
- 1. If Decay wasn't performing yet, jumps to Decay sequence.
- 2. If Sustain wasn't performed yet, jumps to Sustain sequence.
- 3. If Release wasn't performed yet, jumps to Release sequence.
- 4. Default. Just cuts the note with 'note off' command.

Wave types:

Simple
- Sine - 
- Saw - 
- Triangle - 
- Square - 
- FastSine - 

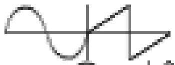
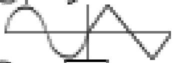
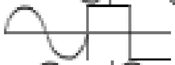
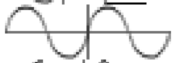
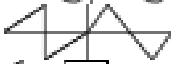
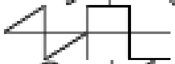
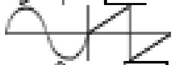
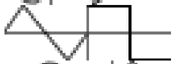
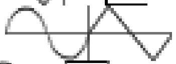
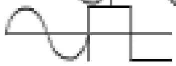
This is approximation of sine, so it's not so smooth as normal sine, but is a little faster.

Mixed (added)
- Sine + Saw
- Sine + Triangle
- Sine + Square
- Sine + FastSine
- Saw + Triangle
- Saw + Square
- Saw + FastSine
- Triangle + Square
- Triangle + FastSine
- Square + FastSine

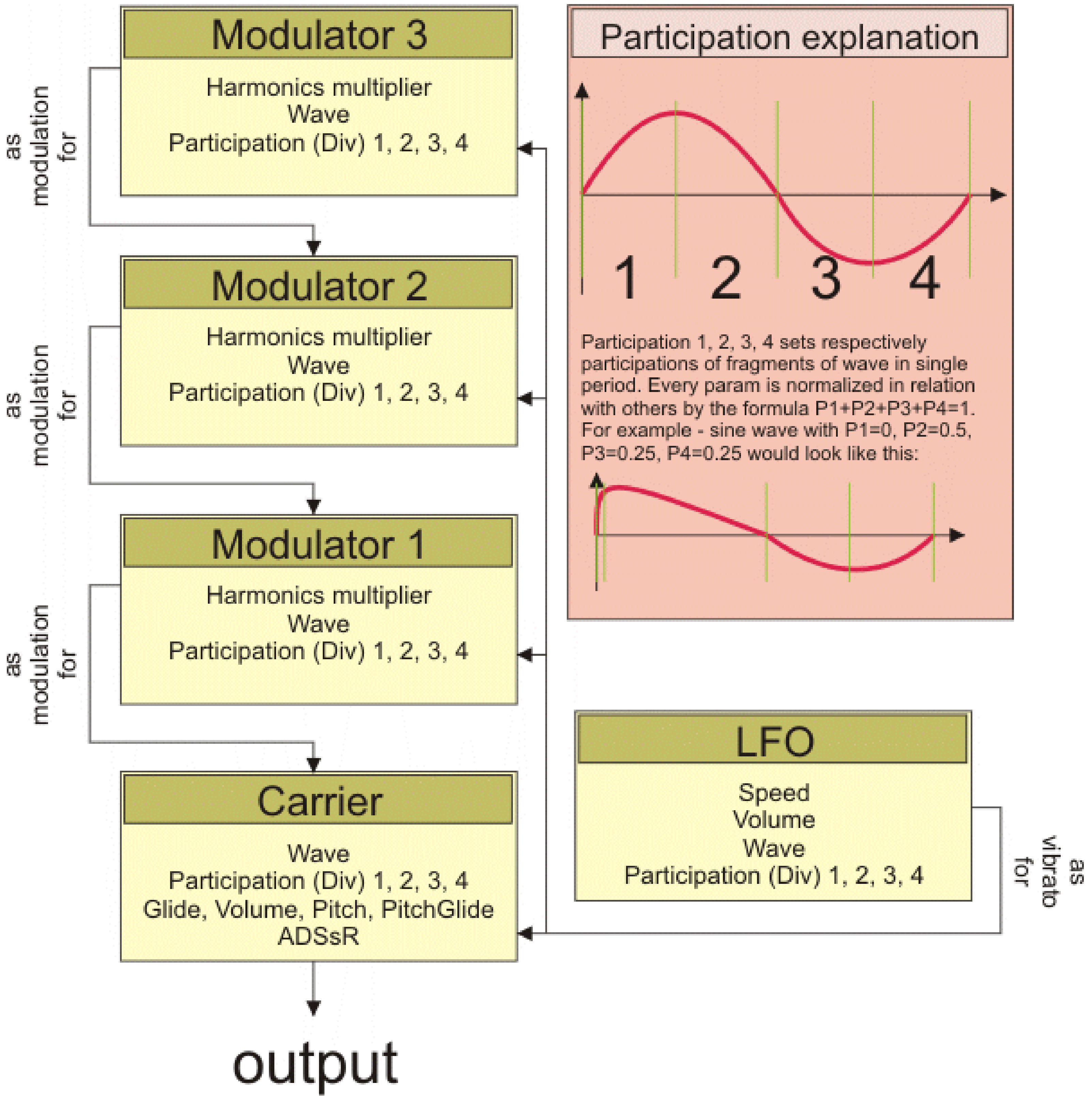
Amplitude modulated (multiplied)

- Sine * Saw
- Sine * Triangle
- Sine * Square
- Sine * FastSine
- Saw * Triangle
- Saw * Square
- Saw * FastSine
- Triangle * Square
- Triangle * FastSine
- Square * FastSine

Joined (Two waves joined, one octave higher)

- Sine | Saw - 
- Sine | Triangle - 
- Sine | Square - 
- Sine | FastSine - 
- Saw | Triangle - 
- Saw | Square - 
- Saw | FastSine - 
- Triangle | Square - 
- Triangle | FastSine - 
- Square | FastSine - 

Here is the scheme as it works with signal:



Modulator 3
Harmonics multiplier
Wave
Participation (Div) 1, 2, 3, 4
as modulation for
Modulator 2
Harmonics multiplier
Wave
Participation (Div) 1, 2, 3, 4
as modulation for
Modulator 1
Harmonics multiplier
Wave
Participation (Div) 1, 2, 3, 4
as modulation for
Carrier
Wave
Participation (Div) 1, 2, 3, 4
Glide, Volume, Pitch, PitchGlide
ADSsR
output
Participation explanation
1
2
3
4
Participation 1, 2, 3, 4 sets respectively participations of fragments of wave in single period. Every param is normalized in relation with others by the formula P1+P2+P3+P4=1. For example - sine wave with P1=0, P2=0.5, P3=0.25, P4=0.25 would look like this:
LFO
Speed
Volume
Wave
Participation (Div) 1, 2, 3, 4
as vibrato for

Few notes:

- Since v1.56 you can freely modify ADSR and LFO-VDecay & SDecay during played sounds. This enables you for example to set first Release to 8 ticks and then after, let's say, 4 ticks change it to 1 without breaking the envelope. The sound will continue to fade as if it were set to 1 tick from start - it means, that the sound will last 1/2 tick from this point (because 4 ticks, that have passed, were

50% of initial 8 ticks setting, and another 50% is left for 1 tick fade).

Kozaluss Angeluss - GenC 1.32

GenC is wave terrain based generator.

Some parameters description:

- *GLIDE* describes the speed of sliding to next note.
- *PGLIDE* describes the speed of sliding to desired pitch.
- #-LFO-RoN - Decides whether LFO should reset phase on new note or not
- Radius Link - Decides whether Radius modulation should be equal to volume modulation (G-ADSR & G-LFO) or it's own modulators (R-ADSR & R-LFO)

Used units guide:

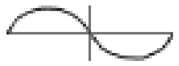
- ADSR time units - *tick* - Length of parts in ticks
- ADSR volume unit - *%* - Percent of maximum volume level
- G-LFO volume unit - *%* - Percent of maximum volume level multiplied with sound volume (for example G-LFO-V=0% => Volume doesn't oscilate, is equal to max, G-LFO-V=100% => volume oscilates between 0 and 2*max)
- R-LFO volume unit - *%* - Percent of maximum amplification of radius oscilator (for example R-LFO-V=0% => Volume doesn't oscilate, is equal to max, G-LFO-V=100% => volume oscilates between 0.5*max and 1.5*max)
- LFO frequency unit - *CpT* - Cycles per tick
- #Shift units - *deg* - Degrees, 0-360
- BRadius units - *deg* - Degrees, 0-180
- #Scale units - *%* - Percent of maximum equal to 360 degrees
- LFO-RP - *deg* - Reset phase (when RoN is on)

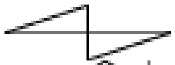
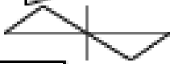
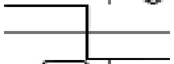
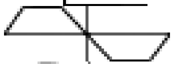
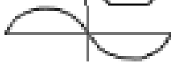
Attribute *MIDI/pattern NOTE OFF ACTION* tells machine what to do, if NoteOff event happened (either from Midi or from Pattern):

- 0. Does nothing. Just plays note as long, as ADSR is set to.
- 1. If Decay wasn't performing yet, jumps to Decay sequence.
- 2. If Sustain wasn't performed yet, jumps to Sustain sequence.
- 3. If Release wasn't performed yet, jumps to Release sequence.
- 4. Default. Just cuts the note with 'note off' command.

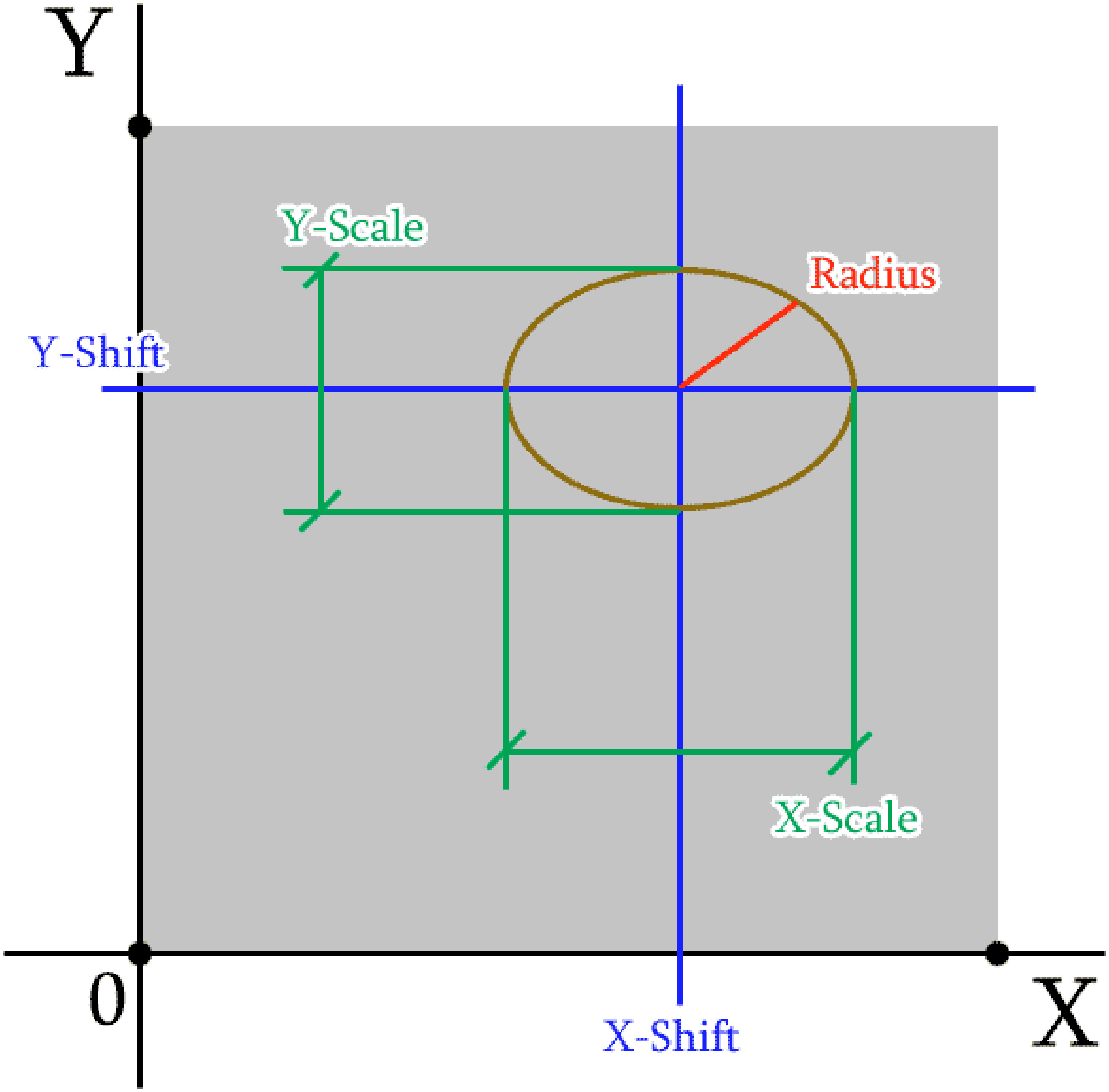
Wave types:

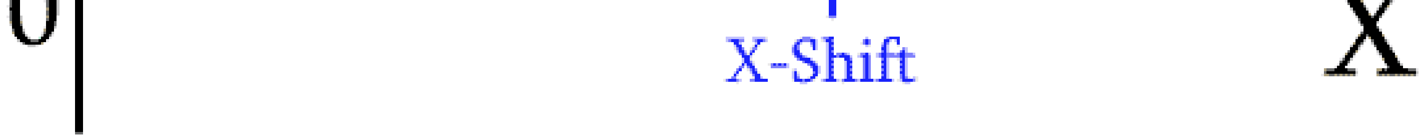
Simple

- Sine - 

- Saw - 
- Triangle - 
- Square - 
- Octagon - 
- FastSine - 
  This is approximation of sine, so it's not so smooth as normal sine, but is a little faster.

Here is a graph that shows what is what in machine's parameters:





Few notes:

- In 1.32 fixed adsr timing bug.
- I've tried to rid of the eventual denormalisation issue with P4 cpus.
- GenC comes in two versions - with inertia (GenC) and without (GenC NI). Version without inertia is 1/3 faster, but the sound quality is worse.
- Since v.1.3. Glide is not reversed and is logarythmic, auto-volume-finder removed as it was causing highs to be louder than lows. Also wavetable was added, but unfortunately it doesn't seem to speed up calculations.
- Since machine relies only on my 'lazy' code, it is extremely slow! (On Athlon1700 one channel consumes 15-20% of CPU power). By 'lazy' code I mean that it is rather written to be easy to use and flexible, than to be ultra-fast. For example - all the parameters are classes in which I've implemented inertia, so every parameter is geometrically inertialised - it makes all changes very smooth.

Kozaluss Angeluss (also Kozaki Soft) at http://www.kozaluss.z.pl/

Kozaluss Angeluss - GenD 1.23

GenD is wave space (extension for wave terrain) based generator.

Some parameters description:

- *GLIDE* describes the speed of sliding to next note.
- *PGLIDE* describes the speed of sliding to desired pitch.
- #-LFO-RoN - Decides whether LFO should reset phase on new note or not
- Radius Link - Decides whether Radius modulation should be equal to volume modulation (G-ADSR & G-LFO) or it's own modulators (R-ADSR & R-LFO)

Used units guide:

- ADSR time units - *tick* - Length of parts in ticks
- ADSR volume unit - *%* - Percent of maximum volume level
- G-LFO volume unit - *%* - Percent of maximum volume level multiplied with sound volume (for example G-LFO-V=0% => Volume doesn't oscilate, is equal to max, G-LFO-V=100% => volume oscilates between 0 and 2*max)
- R-LFO volume unit - *%* - Percent of maximum amplification of radius oscilator (for example R-LFO-V=0% => Volume doesn't oscilate, is equal to max, G-LFO-V=100% => volume oscilates between 0.5*max and 1.5*max)
- LFO frequency unit - *CpT* - Cycles per tick
- #OrbSpeed - *CpMC* - Cycles per main cycle (relative to 'note wave length' length of orbital wave)
- #Shift units - *deg* - Degrees, 0-360
- BRadius units - *deg* - Degrees, 0-180
- #Scale units - *%* - Percent of maximum equal to 360 degrees
- LFO-RP - *deg* - Reset phase (when RoN is on)

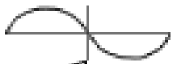
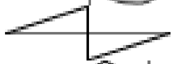
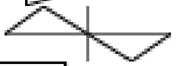
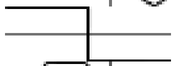
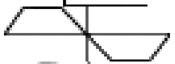
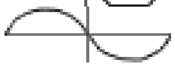
Attribute *MIDI/pattern NOTE OFF ACTION* tells machine what to do, if NoteOff event happened (either from Midi or from Pattern):

- 0. Does nothing. Just plays note as long, as ADSR is set to.
- 1. If Decay wasn't performing yet, jumps to Decay sequence.
- 2. If Sustain wasn't performed yet, jumps to Sustain sequence.
- 3. If Release wasn't performed yet, jumps to Release sequence.

- 4. Default. Just cuts the note with 'note off' command.

Wave types:

Simple

- Sine - 
- Saw - 
- Triangle - 
- Square - 
- Octagon - 
- FastSine - 

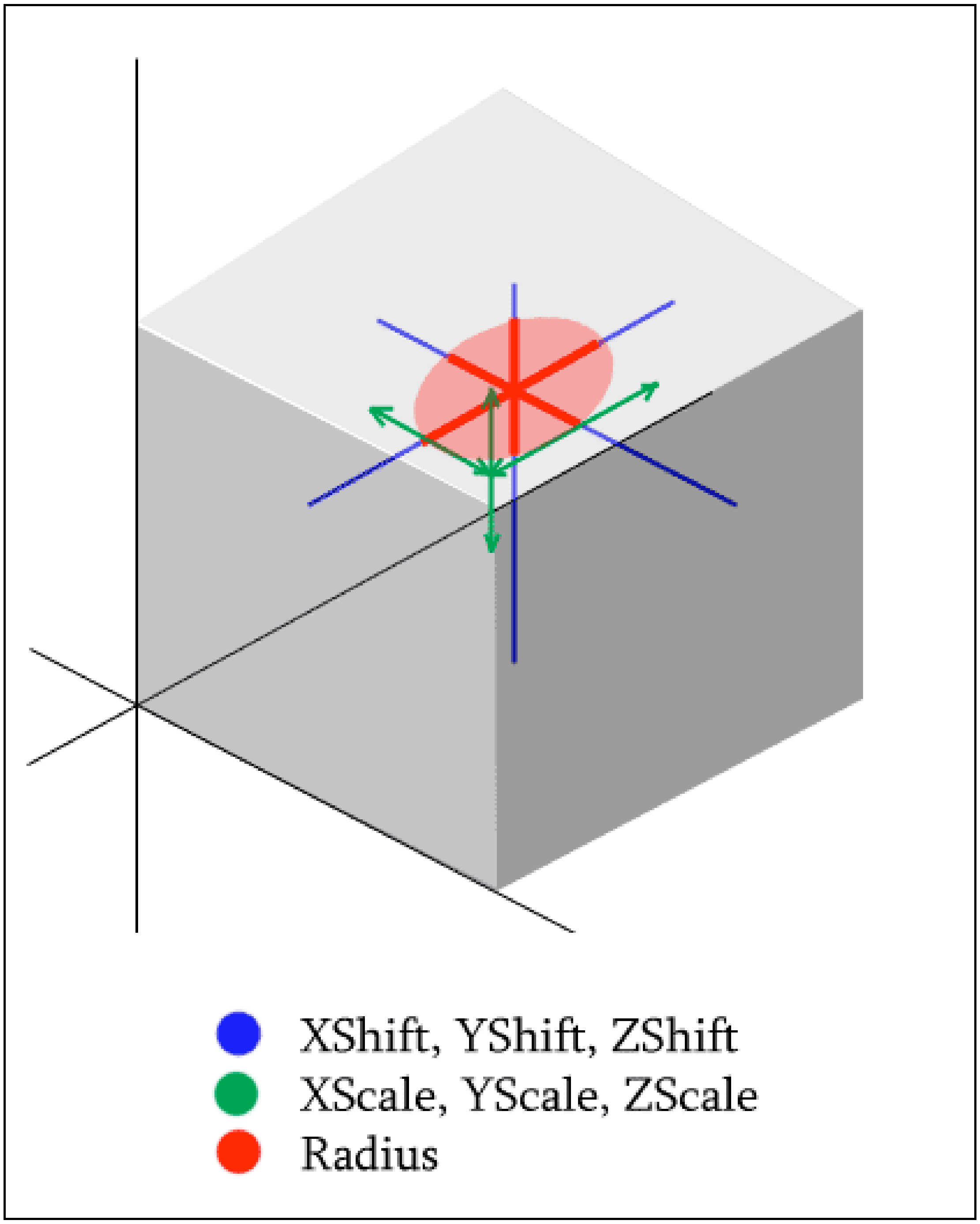
This is approximation of sine, so it's not so smooth as normal sine, but is a little faster.

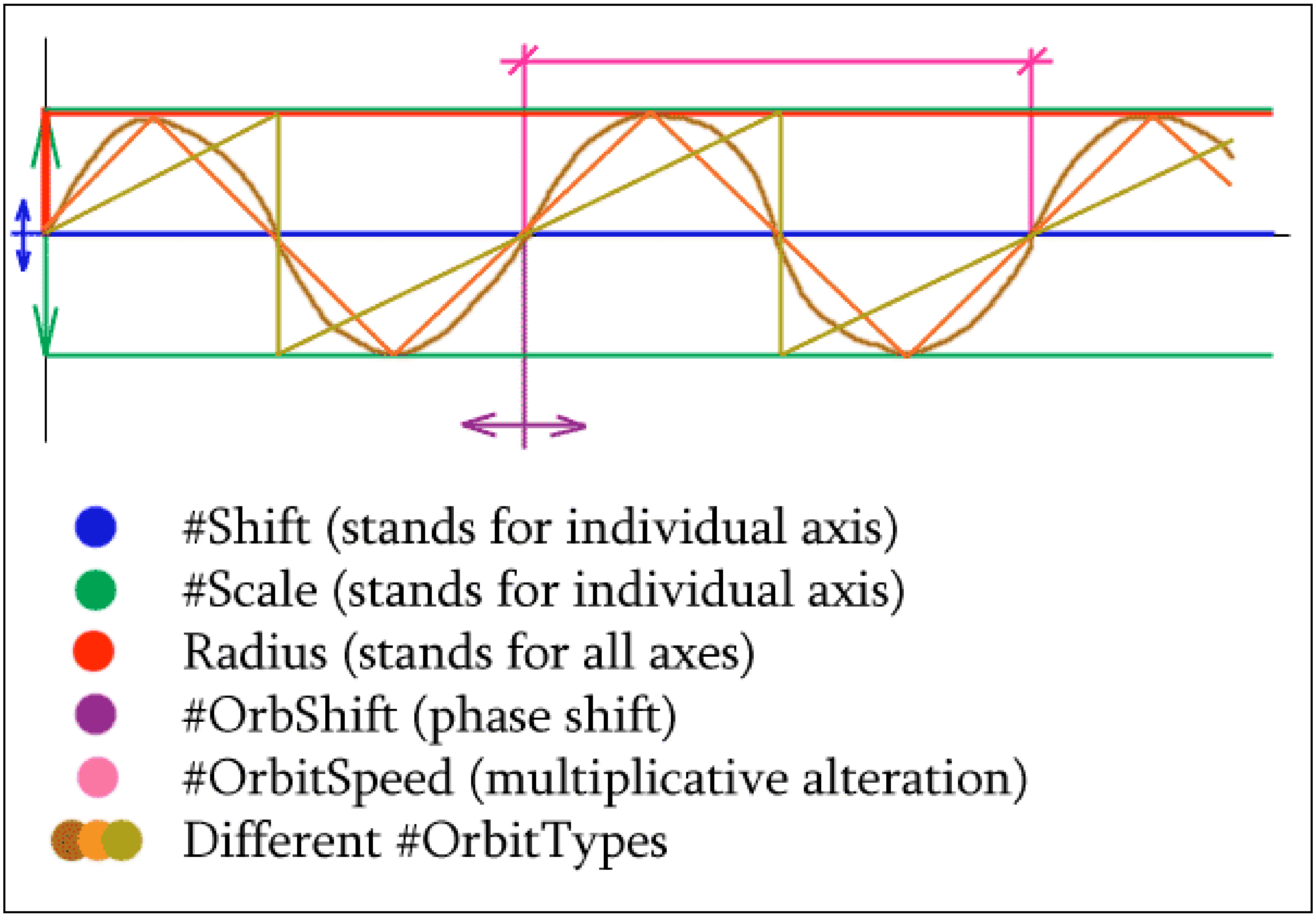
Few words about technology:

- There's an abstract 3D space of functions.
- In this space we select a point (XShift,YShift,ZShift).
- Then we select a box with center in that point, and edges equal to (XScale,YScale,ZScale) *BRadius.
- In this box we run 3 independent oscilators characterised by parameters (#OrbitType,#OrbitSpeed, #OrbShift).
- In this way we have a point flying/oscilating in this box around selected point with certain way.
- Then we got series of arguments (X,Y,Z) that we throw into another oscilators (#Osc) and we get final series of arguments (mX,mY,mZ).
- In last stage we are mixing (mX,mY,mZ) to get one "Mix" value. This is done in few ways:
  0. avg - average (mX+mY+mZ)
  1. mult - multiplication (mX*mY*mZ)
  2. max - select maximum among mX,mY,mZ
  3. min - select minimum among mX,mY,mZ
  4. med - select median among mX,mY,mZ
  5. xy+z - ((mX*mY)+mZ)
  6. x+yz - (mX+(mY*mZ))
  7. zx+y - ((mZ*mX)+mY)
  8. avg-quads - (mX^2+mY^2+mZ^2) (saves sign to fight dc offset)
  9. quad-avg-q - (mX^2)+mY^2+mZ^2)^2 (saves sign to fight dc offset)
  10. avg-mults - (mX*mY+mY*mZ+mZ*mX)
  11. (x>y)?(z):(-z) - if mX is higher than mY, mZ is choosen, otherwise -mZ

12. (y>z)?(x):(-x) - if mY is higher than mZ, mX is choosen, otherwise -mX
13. (z>x)?(y):(-y) - if mZ is higher than mX, mY is choosen, otherwise -mY
14. (x>y)?(x*z):(y*z) - if mX is higher than mY, output is mX*mZ, otherwise mY*mZ
15. (y>z)?(y*x):(z*x) - if mY is higher than mZ, output is mY*mX, otherwise mZ*mX
16. (z>x)?(z*y):(x*y) - if mZ is higher than mX, output is mZ*mY, otherwise mX*mY

Here are few images that should help in understanding how machine works:




#Shift (stands for individual axis)
#Scale (stands for individual axis)
Radius (stands for all axes)
#OrbShift (phase shift)
#OrbitSpeed (multiplicative alteration)
Different #OrbitTypes

Few notes:

- In 1.23 fixed adsr timing bug, made some optimalisation and added few mix-modes.
- I've tried to rid of the eventual denormalisation issue with P4 cpus.
- GenD comes in two versions - with inertia (GenD) and without (GenD NI). Version without inertia is 1/3 faster, but the sound quality is worse.
- Since v.1.2. Glide is not reversed and is logarythmic, auto-volume-finder removed as it was causing highs to be louder than lows. Also wavetable was added, but unfortunately it doesn't seem to speed up calculations (you can check it out in attributes: SynthMode(0=math,1=wavetable), Interpolation(0=nearest,1=linear).
- Since machine relies only on my 'lazy' code, it is extremely slow! (On Athlon1700 one channel consumes 15-20% of CPU power). By 'lazy' code I mean that it is rather written to be easy to use and flexible, than to be ultra-fast. For example - all the parameters are classes in which I've implemented inertia, so every parameter is geometrically inertialised - it makes all changes very smooth.

Kozaluss Angeluss - HSync 1.01

HSync is multitrack stereo sine generator based on Hemi-Sync technology used to syncronize human brainwaves.

Parameters:

- G-Volume - Global volume for all tracks
- Volume - Per-track volume
- BaseFreq - Base frequency
- SyncFreq - Syncrosization freqency
- SlideTime - Time, that will pass before BaseFreq or SyncFreq will be lastly set value
- Reverse - Reverses stereo channels

Few notes:

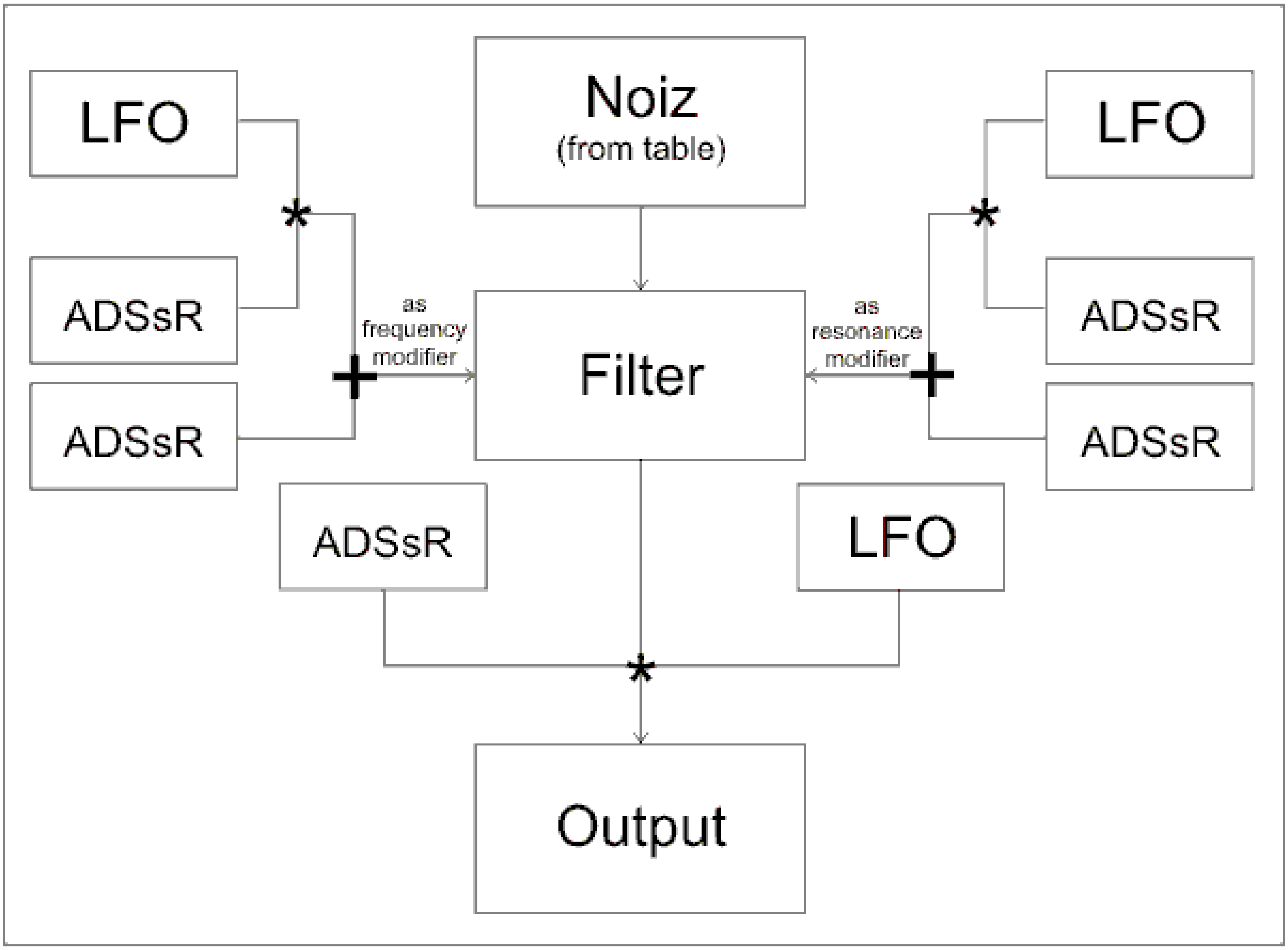
- [Base frequency] goes to the left channel (right - if reversed) and [Base frequency + Sync frequency] goes to the right channel (left - if reversed).
- Channels cannot be blended (if the purpose is syncronization of brain ;)), so do not put any machines that affect stereo after this generator.
- SlideTime is 'smart inertia'. Whenever you set new BaseFreq and/or SyncFreq, it will recount new slide parameters, so you can change them indepedently or together without worrying, that it won't work as expected.
- But - remember to initialize machine first - set SlideTime to 'zero' and trigger it (it may be done with pattern of course). If you won't do it, the first slides will come from default settings.
- More informations about this technology can be found by looking for BrainWave Generator software or Monroe Institute official site.

Kozaluss Angeluss (also Kozaki Soft) at http://www.kozaluss.z.pl/

Kozaluss Angeluss - Noizza 1.5

Noizza is multitrack noise synth generator.

Here is the scheme showing how it works with signal:



Parameters' Prefixes:

- G-... : Global. These are Global Volume [per machine], ADSsR (as envelope) & LFO (as gapper) [per track]
- F-... : Filter. These are base filter parameters
- FF-... : Filter Frequency modifiers
- FQ-... : Filter Q modifiers

ADSsR Parameters:

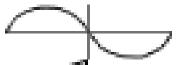
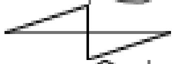
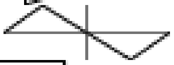
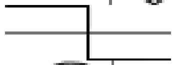
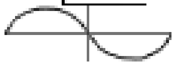
- ADSsR-A : Attack - Time in ticks/32
- ADSsR-D : Decay - Time in ticks/16
- ADSsR-S : Sustain - Time in ticks/16 or Infinity
- ADSsR-s : Sustain Level - Level in percent of Max
- ADSsR-R : Release - Time in ticks/16
- ADSsR-M : Mod - Multiplier

LFO Parameters:

- LFO-WS : WaveShape selection for LFO
- LFO-F : Frequency in cycles per tick/16
- LFO-RP : Reset Phase - Used when RoT=Yes, in radians (0,2PI)
- LFO-RoT : Reset On Trigger - Resets LFO on Global Trigger
- LFO-M : Mod - Multiplier
- LFO-A,D,S,s,R : Same as in ADSsR

LFO Wave types:

Simple

- Sine - 
- Saw - 
- Triangle - 
- Square - 
- FastSine - 

This is approximation of sine, so it's not so smooth as normal sine, but is a little faster.

Mixed (added)

- Sine + Saw
- Sine + Triangle
- Sine + Square
- Sine + FastSine
- Saw + Triangle
- Saw + Square
- Saw + FastSine
- Triangle + Square
- Triangle + FastSine
- Square + FastSine

Amplitude modulated (multiplied)

- Sine * Saw
- Sine * Triangle

- Sine * Square
- Sine * FastSine
- Saw * Triangle
- Saw * Square
- Saw * FastSine
- Triangle * Square
- Triangle * FastSine
- Square * FastSine

Attribute *Trigger-OFF ACTION* tells machine what to do, if TriggerOff event happened:

- 0. Does nothing. Just plays note as long, as ADSR is set to.
- 1. If Decay wasn't performing yet, jumps to Decay sequence.
- 2. If Sustain wasn't performed yet, jumps to Sustain sequence.
- 3. If Release wasn't performed yet, jumps to Release sequence.
- 4. Just cuts the note.

Few notes:

- Parameter F-Gain works only for filters Peaking, LoShelf & HiShelf
- If you experience clicks - try turning off F-RoT option, as it resets filter everytime trigger comes.
- Parameter 'Instant' sets (when 'Yes') all the parameters jump instanty to their destination value when machine is not playing. This is useful, if you want to prepare parameters before machine starts playing. If set to 'No', all the parameters are going to their destinations, but only when playing! This means, that if there's slide in frequency for example, and sound lasts for 1 tick, and then is 3 ticks of silence and then second sound lasting 1 tick, freq will go during this first tick, then will stop and wait during 3 ticks of silence, and then continue to slide during second tick of sound. If you want to prevent this, use 'Yes' in parameter 'Instant'.
- All parameters are inertialized, mostly by blending 0.5:0.5.
- Some filter configurations (Type+Freq+Q) happen to be unstable.
- It was meant to be drum-synth :P
- It can handle from 1 to 8 tracks, but be careful - it kills CPU :(

Kozaluss Angeluss (also Kozaki Soft) at http://www.kozaluss.z.pl/

ld grain 0.94 beta

-1.

(C) Lauri Koponen 2002

Big thanks to Raúl Reales!

These buzz machines use Oskari Tammelin's XDSP Beta 2 library.

0. warnings

THE SOUND IS VERY DEPENDENT ON SAMPLING RATE. CHANGING OUTPUT SAMPLING RATE *WILL* BREAK YOUR SONG!

... and don't set grain density too high, unless you have a fast cpu.

1. what is a grain?

i'm not sure what is the exact definition of a sound grain but in this synth it's a short piece of audio data clipped from a wavetable wave. the piece is very short, usually less than 50ms. each grain has it's own envelope and can be replayed at any speed. a grain can also be extended in length, with it's contents looped. the wavetable wave position that is used as a grain data source can also be advanced at different speeds. also multiple grains can be played at the same time, but this will cause cpu usage to rise.

2. notes on parameters for the grain generator:

inertia
inertia does not affect all parameters.

tuning1
affects waveform playback speed.

tuning2
affects note pitch or waveform playback speed if notetarget is C.

wave
selects waveform index.

glength
grain length in samples.

gdensity
distance between grains, relative to glength.

gdatalen
length of grain data from wavetable in samples.

gposstart
position in wavetable where to start on note-on.

gposdelta
position advancement in wavetable for each grain in samples.

gpitchmod
pitch modulation during grain.

gloopmode
grain data loop mode. grain data (length set with gdatalength) is always looped, this parameter selects forward looping or ping-pong looping.

wloop
enable/disable sample looping.

resetloop
if 'on', sample position will be reset whenever a note-on occurs.

gdenslen
causes gdensity to modify glength too. if gdensity is 50%, also glength is scaled to 50%.

notetarget
selects different note handling mode. different modes allow sound pitch to be modified with note pitch in different situations.

GEnvA/A[M]/D/D[M]/Mdl/Amnt
modify grain envelope.

grain envelope is a bezier curve. A controls envelope attack shape,
D decay shape. A[M] and D[M] allow a bit more control on the shapes.
Mdl is envelope middle position, 50% causes attack and decay length
to be the same. Amnt changes envelope amount on grain volume.
amount 0% = grain volume always at maximum, amount 100% = grain
volume completely controlled by envelope (beginning and end at zero,
middle at maximum).

3. notes on parameters for the grain effect:

the parameters are the same as in the grain generator, except for a few
missing parameters only useful for samples. the latest received input
sound will be used as grain material.

4. extras

there's a built-in dc filter, enable it in the attributes. it's not
very good, though.

5. version history

0.96
- more bugfixes

0.94
- oops! filters work now

0.93
- oskari's xdsp lib
- some bugfixes

0.9
- initial release

- ld0d@kolumbus.fi

ld jacinth Documentation

First of all, this documentation is written very quickly and doesn't describe how different parameters actually sound like. Feel free to experiment.

Currently this documentation is also out of date. Someone want to rewrite it?

Added a small FAQ section, no other updates. **Read it first!**

If you have any questions, please mail ld0d@kolumbus.fi

- ld

1. ## This synth is slow!

   Yes, I know. But to make it less slow, you could try these tips:

   1. Don't use the output saturation unit. Calculating the soft saturation is very slow. You might want to try some distortion machine instead, like Geonik's Saturator.
   2. Switch to mono mode. Stereo mode needs more calculations, and if you don't need independently panned oscillators/tracks you should use the mono mode.
   3. Turn off unused oscillators. If an oscillator's gain parameter is set to 0, the synth spends less time generating the oscillator outputs. Note that using phase or frequency modulation on oscillators 2 and 3 always forces oscillator 1 to generate output.
   4. Set the unison parameter to a lower value. The unison parameter causes the synth to generate all the oscillator outputs multiple times. You can usually achieve the same result with detuning the oscillators slightly and setting unison to a lower value.
   5. Use different filters. Some of the filters are faster to calculate than the others. Especially the moog filter is slow.
   6. Use less envelopes and LFOs. Each active envelope and LFO takes some time to process.

   About random presets:
   Please try to tweak the cpu usage at least a bit if you're doing new sounds by randomizing the parameters.

2. ## FAQ

   ### Waveform editor

   #### Can I play s...

No you can't, it's not a sampler.

Drawing waveforms
The top display in the waveform editor displays an editable waveform frequency response. Changes to this graph will affect the lower display, where you can see how one cycle of the waveform looks like. Upper display is what you hear, lower what you see if you look at the synth output.
Blue bars in frequency graph show octaves of the fundamental frequency.

Waveform Sampling: Zero-crossing finder and cycle fixer
Select the waveform so that the whole cycle is visible in the window, with little extra. Zero-crossing finder will seek the waveform from *left edge rightwards*, cycle fixer will find a loop position from *right edge leftwards*.

Misc.

Where's the noise?
Oscillator 1 waveform B and LFO waveform B. Nowhere else.

It's crashing!
If it crashes in the gui, you'll just have to live with it (or not use the gui). If it doesn't, try to reproduce the crash, and if you can, mail me (ld0d@kolumbus.fi). If you can't reproduce it, I can't really help much. Be sure to be available for more questions if you decide to mail me a bug report :)

It clicks!
Usually the reason is too short attack/decay/release times set in envelopes. Might be also because of the waveform you're using.

Slower controlling for gui knobs
Press and hold the ctrl-button while turning knobs in the gui. That will make them move slower.

Track editing in gui
The lights in the upper left corner of the gui open track parameter windows. If you have 5 tracks, 5 of the 16 lights will be active. Click those.

There's only one envelope and lfo!
Add more tracks and/or read the docs below.

How do I... [lfos, envelopes]
Read the docs below.

3. Operation modes and output post-processing

Modes

The synth can work in three different modes: mono, stereo and simple. Mono and stereo modes are almost the same, except that in mono mode all panning parameters are ignored. In simple mode oscillators 3 and 4, filters 2 and 3 and panning effects are disabled, decreasing cpu usage.

Output saturation

There is a parameter for output gain and a saturation unit that can do "hard" and "soft" clipping. The saturation effect is very slow, especially in stereo mode, so it is preferred that you use some external effect to do saturation.

4. Pitch bend

There's an attribute for the pitch bend depth. The attribute value is the pitch bend parameter value where the pitch is bent one octave up.
That is:
pitchbenddiv = 64 -> pitch bend at +64 = one octave up
pitchbenddiv = 48 -> pitch bend at +48 = one octave up, pitch bend at +4 = one semitone up
pitchbenddiv = 768 -> pitch bend at +64 = one semitone up.

5. Inertia

Inertia affects cutoff and pitch bend speed.

6. Oscillators

Waveforms

The waveforms are a bit highpassed. Try waves 66, 67 and 68, those are generated (they're the same as in ld padsyn) and might be better for some uses (bass sounds?).

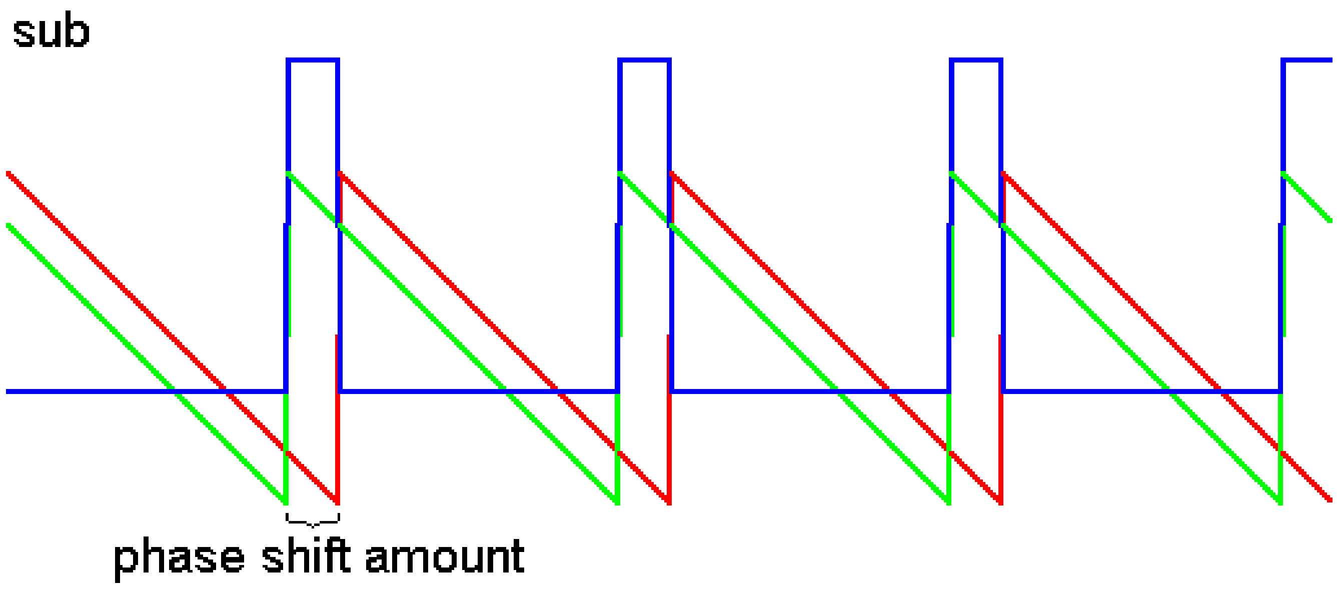
Oscillator 1 wave B and the LFO wave B can be set to generate noise. Osc1 has wave number 69 which is white noise, LFO has waves 69 and 70: sample&hold noise and sample&glide noise.

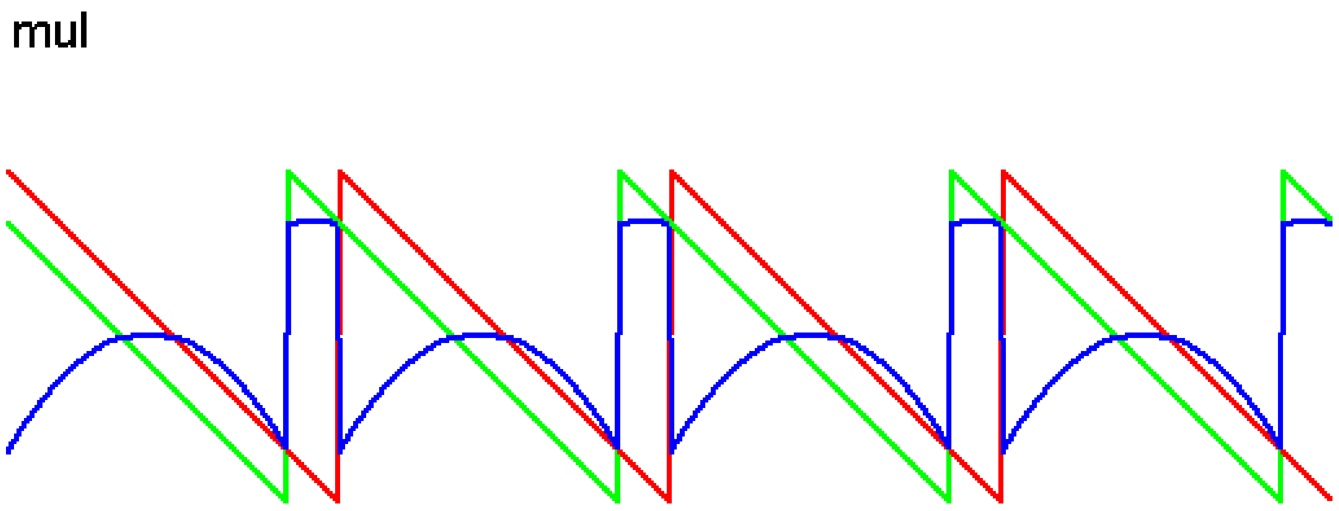
Waveform mixing

The WMix parameters morph output wave smoothly from wave A to wave B.

Phase shift

The PShift parameter just moves the wave in time. But turning the PType parameter to Sub or Mul causes the whole wave to be mixed into a phase-shifted copy of itself:





...So you can easily generate pulse-width modulated pulse wave by using saw wave with subtractive phase shifting. **Remember that the wave phase affects the result of phase modulation and frequency modulation**, even if the PType parameter is set to "Off" position.

Detuning

Detune amount of "12.20" would mean that the oscillator is tuned one octave and 20 cents up.

Gain and Saturation

If the gain parameter is set to >1.0 (>64), there is (asymmetric) distortion applied to the waveform. Also a star (*) appears next to the parameter value when the oscillator output is saturated.

Mixing modes

Oscillators are handled in order from 1 through 4. On oscillators 2-4 you can set the output mode to normal, subtractive or ringmod. This mode affects the output of the oscillator and the combined signal before it (if you set osc3 to ringmod, it will modulate the mixed signal from oscillators 1 and 2).

Phase Modulation

The phase modulation (controlled with parameter PModAmt) is an extension to oscillators 2 and 3, and causes the output of oscillator 1 to be linked to the phase shift amount of oscillator 2 or 3. The PModType parameter can be used to change the source of phase modulation to a sine wave that is synced with the oscillator.

If oscillator 1 waveform is sine wave, phase modulation changes the sound exactly like frequency modulation would.

Frequency Modulation

Frequency modulation source is always oscillator 1. The FM system isn't working too well, and you usually get some "drifting" in the sound. Turning on oscillator syncing often helps.

If you can achieve the same result with phase modulation, use it instead. FM is only added as a curiosity, though it may have some uses.

Oscillator 3 slaving

By using the oscillator 3 mode parameter you can slave it to oscillator 2. This causes it to copy all the parameters of osc2, with only detuning inversed (if osc2 has detune amount of 0.12, osc3 would have detune amount of -0.12). If the mode parameter is set to "Slv/wave", the oscillator 2 wavea, waveb and wavemix parameters are not copied to oscillator 3, but it's own parameters are used instead. If the mode parameter is set to "Slv/pan", the oscillator 2 panning is not copied to oscillator 3.

Syncing

Using the sync parameter oscillators 2 and 3 or 2, 3 and 4 can be synced exactly to oscillator

1. Syncing causes the oscillator phase to be reset when oscillator 1 finishes cycle. If the sync parameter is set to "NoteOn", the oscillator phases will only be reset on note-on.

Unison and the spread parameter

Changing unison parameter duplicates all the oscillators and detunes them slightly, depending on the spread parameter. Great for fat pads. **Note:** the unison parameter will slow the machine down quite a bit. If you play one note with unison set to 6, it will practically play six notes, using six times the cpu time.

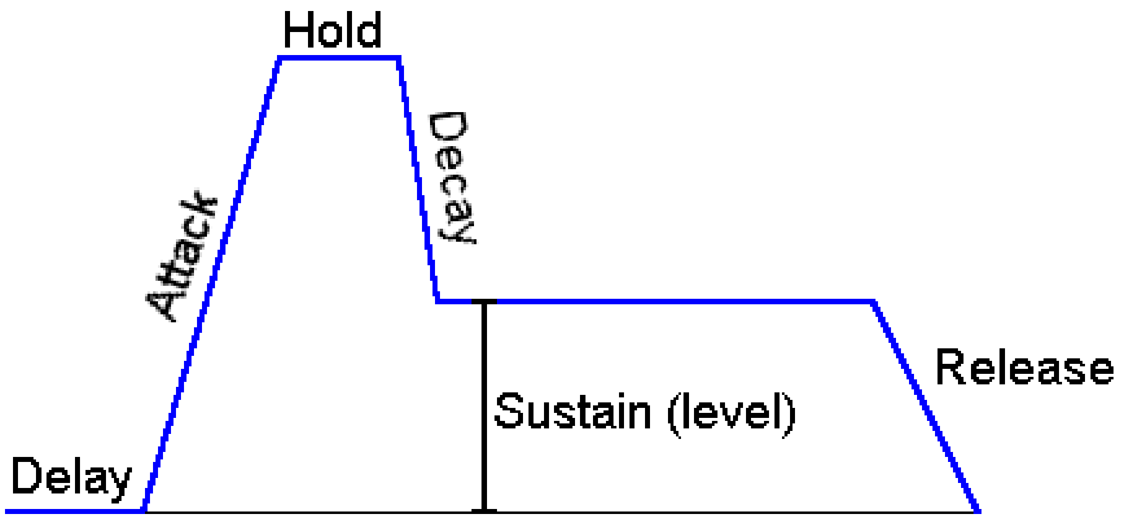
7. Filters

There are three filters in the synth, and there are four different routing modes for them, controlled with the FilterConf parameter. (mode "->(1, 2)->3->" means that filters 1 and 2 are linked in parallel and their output is mixed and put through filter 3.) Filter 1 has a KeyFollow parameter which makes the filter cutoff change according to the notes played (C-4 is considered as the middle note that doesn't affect filter cutoff). Filter 2 and 3 cutoffs can be linked to filter 1 cutoff with the CutoffLink parameter.

8. Envelopes

As you might have noticed, each track on the synth has only one envelope. All of the envelopes are "global" though. If you set the envelope of one track to control filter cutoff, all the tracks are configured to have one envelope controlling that track's cutoff. When one of the tracks is given a note-on command, that track's copy of the envelope is triggered and it changes only that track's cutoff parameter. This feature is shamelessly ripped from Rymix' KyrieSpectra, thank him for it :)

I decided to extend the standard ADSR envelopes a bit.



While in the delay phase, the envelope follows the slope that was there before it was triggered. In the hold phase the envelope output is at maximum level. There is no parameter for sustain time, but instead the envelope moves to release phase when a note-off is received.

If you use the delay parameter on the amp envelope, note that the oscillators change pitch as soon as you enter a new note. If the envelope is delayed, you may or may not, depending on the previous state of the envelope, hear the change of note.

The accent parameter

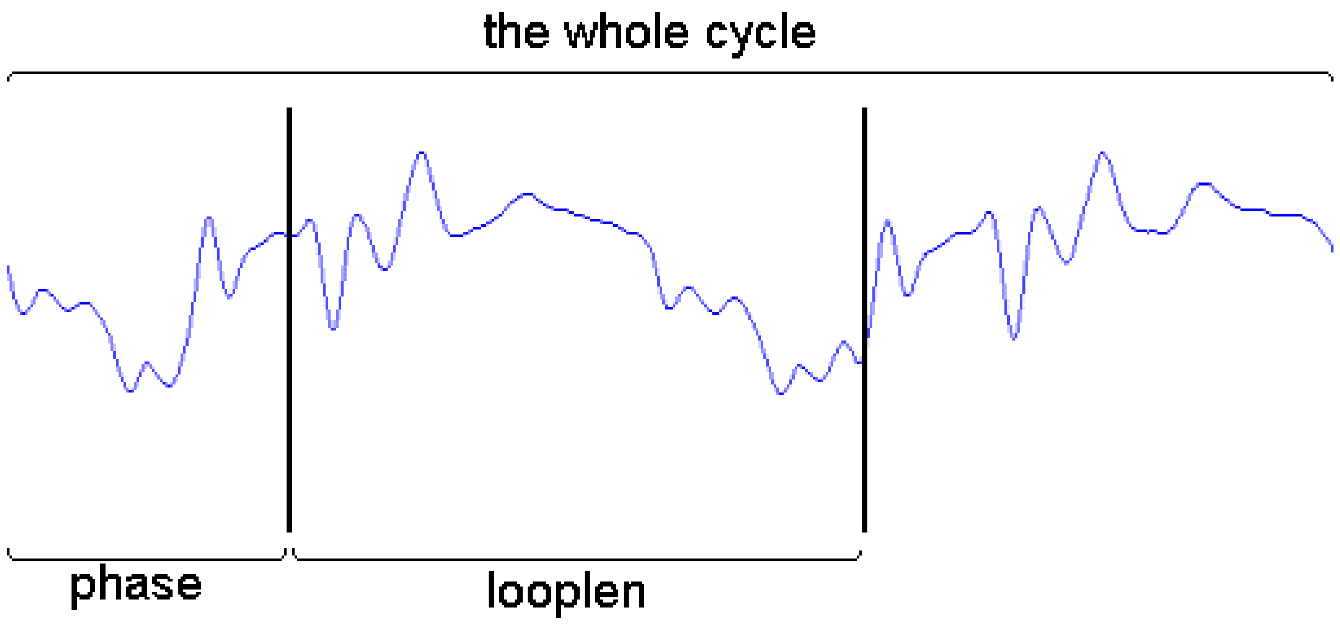
Each track has a volume and an accent parameter. (In the machine attributes you can choose where to route midi velocity messages.) Using the envelope accent parameter you can configure how the track accent parameter affects the envelope amount. There's also DecAcc parameter that changes how much the accent changes decay length.

9. LFOs

The LFOs are also a bit extended from "normal". Like the envelopes, the LFOs are copied to all the tracks. They can also be configured to run as global or track LFOs. Global LFOs are in same phase on all the tracks and aren't affected by note-on commands. Track/TrackOnce LFOs' phase is reset on note-on, and TrackOnce LFOs stop running after one cycle, so they can be used as kind of extended envelopes. The LFO waveforms are set just like oscillators. WaveB can be set also to sample&hold noise or sample&glide noise. The invert parameter negates the wave (changes ramp-up saw wave to ramp-down).

The speedmode parameter affects how the speed parameter is handled, this way you can select the lfo cycle speed in Hertz or ticks.

The phase and looplen parameters can be used to play only a part of a waveform. If looplen is set to 128, the whole cycle is played. If looplen is set to 64, half of the cycle is played: the range [phase, phase + half of wave length].



The LFO outmode parameter changes the lfo from unipolar to bipolar (unipolar goes from 0 to lfoamp, bipolar goes from -lfoamp to +lfoamp).

ld padsyn v1.08 (C) 2001 Lauri Koponen
subtractive synth

0. Version history

v1.03
   fixed bad tuning (many thanks to dj digitalis!)
   added tuning attributes
   added note-synced LFOs
   renamed "readme.txt" to "ld padsyn.txt"

v1.02
   fixed LFOs while machine not generating output
   fixed syncing
   fixed/added some texts

v1.01
   fixed subosc waveform

v1.0
   initial release

1. General chat

i know there are quite a few bugs in the machine, but if it
crashes, mail me and i'll try to fix it. the bugs concerning
then FM algos are old news, consider those as "features". ;)
generating the waveforms takes a while on slower computers,
go for a cup of coffee while waiting it to initialize.

feel free to mail me if you like or hate the machine, or
if you have any ideas for additions.. all feedback is welcome!

2. Very short tutorial about the phasemod system

Both main oscillators (osc1 and osc2) have support for simple
"phase modulated waveform mixing". By setting the O(1/2)PType
parameter to 'Sub' or 'Mul', the synth generates two waveforms

instead of one, with phase difference of O(1/2)PShift. If
OxPType is 'Sub', output is wave1 - wave2, and if OxPType is
'Mul', output is wave1 * wave2. Try this with some oscilloscope
plugin (Geonik's Visualization) to see how the waveform
is affected.

To generate PWM pulse wave, set osc1 or osc2 waveform to Saw,
PType to Sub, and alter pulse width with PShift. PShift can
be controlled also with the LFOs.

Osc2 phase shift affects the osc2 phase even if O2PType is set
to 'None'. This can be used with for example FM to get different
waveforms.

3. FM and FM modes

Osc2 always modulates Osc1 frequency.

There are a few FM modes. I think 'Sub' is the way Reason (by
Propellerheads) does FM, but 'Mul' is a "more correct" FM algo.
'Add' and 'Sub' just change osc1 phase, while 'Mul' actually
changes the oscillator frequency.

There are known stability problems with high FM values with
high frequency signals.

4. About detuning

The global tuning parameters have been added as attributes in v1.03.
It seems that what my synth uses as 'cents' are actually 1.5 cents,
so... better get used to it ;)

- ld0d at kolumbus dot fi

jacinth 1.05 upgrade
8.12.2002

This is an update to the original jacinth, modified to use Oskari
Tammelin's XDSP Beta 2 library.

This version should be perfectly compatible with original jacinth,
but songs and presets that use the new features are NOT compatible
with OLD jacinth!

Installing this update:

FIRST DOWNLOAD AND INSTALL THE ORIGINAL JACINTH PACKAGE - THIS IS ONLY
AN UPGRADE!

It's also helpful if you make sure that original jacinth works on your
computer before trying to install this update. Make sure you READ
the documentation, ESPECIALLY THE README FILES that come with the
original ZIP! You have been warned!

Ok, so you have jacinth up and running. Then just close buzz and
overwrite the old "gear\generators\ld jacinth.dll" with the one in
this zip. Be sure to check you got it right by looking at the about box:
load up jacinth in your song, right click on it and select "About..." in
the popup menu. The version should be 1.05.

Changes to original jacinth:
----
 * LFOs and envelopes can now affect LFO speed
 * Accent can be linked to Volume
 * Pitchbend hack: parameter value 127 means 128 ("63" is now "64"),
   allowing full range of pitchin'.

About that strange accent parameter:

Accent values 0..128 (0x00..0x80 hex) work just as they did before,
but values in range 129..228 (0x81 to 0xE4 hex) cause accent to be linked
to volume at a percentage, 1%..100%, according to the parameter value.

That is:
Accent = 120 -> accent is set to 120, just like before.
Accent = 180 = 128 + 58 -> accent is set to 58% of volume!

If you're playing with midi and you want to use the new accent % feature,
you have to set it for every track separately before playing notes!

This feature is not supported through the GUI.

Thanks to Vectrex for the ideas for these new features and mva and
HamsterAlliance for testing the XDSP version!

- ld0d

LdC SpeedLoader 0.9

This plugin is for use with Jeskola Buzz 1.2 - It may not work with Buzzle and may even make it crash.

Place in your buzz/gear/generators/ directory.

Overview

SpeedLoader is a utility plug-in that can make your BMXs load/save faster by disabling audio during these processes.

How To Use

Just load it and leave it - no need to connect to anything.

How It Works

SpeedLoader disables Buzz audio while your BMX is loading or saving, then re-enables it when you hit play or use the unmute slider/parameter. This is especially important if you use VST/DX (eg. with PolacVST(i)), as some of them are "always on" (and hence it's likely that any subsequent machines in the chain leading to Master are also "on"), and will be sapping your CPU during loading/saving, meaning it takes even longer to load. Buzz samples are compressed, and CPU is needed for compressing when saving and decompression upon loading, which makes the problem even worse. So if you are working with very "heavy" BMXs, with lots of machines/VST and big samples, this machine will probably make loading/saving faster for you.

NOTE: SpeedLoader doesn't actually disable the Buzz audio system and it doesn't switch off the driver either. It just sets the "Volume" parameter of the Buzz Master machine to the minimum during saving/loading, which disables all Buzz audio processing. You could achieve the same thing by manually setting the Volume parameter of the Buzz Master machine to the minimum every time you save your BMX/BMW (and hence every time you load it in again).

Parameters

* Unmute - Flip this to manually unmute the machine. Useful if you are using Buzz for signal processing stuff and don't want to play the BMX you just loaded to get the signal happening.

* Load - Enables/Disables SpeedLoader for loading - in case you don't want the audio to go off when you load something.

* Save - Same as above but for saving.

Known Bugs

* Not really a bug, but something that is a little annoying: SpeedLoader always resets the Master Volume to the maximum... I tried to make it read the previous volume and then restore it (by "hacking"), but I couldn't get it working nicely. I decided to release this anyway as version 0.9. You shouldn't be using the Buzz Volume thing anyway, just leave it at maximum and use something else to control the final output volume (eg. use LdC Automax to get the loudest possible signal without clipping).

Bug Reporting

Any problems, let me know: lee_ducaine@hotmail.com

Lipid IT Loader *by Austin Luminais*

This machine is used to load samples from an .IT module file into the wavetable.

Usage

In the context menu that appears when you right click on the machine is the command "Import Module Samples". This will allow you to choose an .IT file, and then all of the samples in that module will be loaded into the wavetable. Any waves currently in the wavetable will not be overwritten. When you are finished loading samples you can remove the machine.

Important Note: This currently does not support IT 2.14+ format files that use compressed samples. If you attempt to import a file saved in this format, no samples will be loaded. If you have information on the format of such files, please send it to me.

This plug-in is freeware.

E-mail: **lipid@68k.org**

O h m - L i v e s l i c e

New version: 1.5

After a small break, I've started buzzin' again. So I decided to release an updated version of LiveSlice. This version is crash-free - I've fixed some bugs and memory leaks to ensure much greater stability. If there are still issues there's a chance you can get hold of me by visiting www.livelab.dk and locating my email address, or if you are on KVRaudio.com, send a PM to the user ohm.

LiveSlice is now running in Buzé - seemed to be the most promising buzz clone on windows when I wrote this.

I've also added support for 24 and 32 bit wave files and updated the waveform display so it looks more like what you hear. Other features are better support for slices of varying duration. If you have a snare hit that is twice the duration of the other slices, simply drag the slice cutting it in half and create an empty (zero duration) slice - extremely short slices will not be played back.

I didn't make major changes to the playback engine to ensure 100% backwards compatibility. If you need more slicer features, go try the VST version. Screenshots at the bottom of this page (if you are online).

Quickstart guide

- Double click the machine to bring up the GUI
- Click the open button to loat a wave file from disk (44.1 kHz 16 bit uncompressed wavefiles only) or click the buzz wavetable icon to import a loop from the wavetable (the wavetable slot can be changed by typing in a different number in the box)
- The upper waveform is the original sound, the lower one is the current arrangement
- Rearrange slices by dragging them (you can drag from both waveforms)
- Playback starts when the play botton of buzz is pressed, and continues to loop until buzz is stopped.
  To use liveslice with the buzz sequencer set the attribute labeled "wait for trigger parameter" to 1. Now create a pattern with the value 1 in the trigger column, an place it in your song.
- Set the number of slices by clicking on the boxes next to the "slices:" label
- Set the number of measures by clicking the "measures" box
- Try randomizing the slices using the spacebar to toggle between your current arrangement and a random one, or pressing the "dice"

icon.

- You can always revert to the default (linear) arrangement using the "X" button next to the dice
- Save a preset by right clicking on any of the 32 buttons in the bottom half of the window
- You load a preset by left clicking or by using the keyboard shortcuts (see below)
  Presets can be exported and imported to / from .sli files using the load / save buttons above the presets

Sample Import / Export

import
press the "open file" button to select a file from your **harddrive**. Samples must be 16 bit uncompressed. The selected sample is previewed - this can be turned off using a machine attribute.

press the **wavetable** button to load a file from the buzz wavetable. Select the slot by typing in a number in the box next to the button. If you loaded a sample from disk you can transfer it to the wavetable by right clicking this button. Useful if you want to share your bmx file.

export
the save button exports the re-sliced loop and all individual slices to the buzz wavetable. The loop is saved as it sounds - with envelope modifications, but without any attached peer effects - use the CnG recorder if you need those.

Parameter envelopes

There are four available parameter envelopes:

- Volume
- Panning
- Pitch
- Decay / Length
- Slice - an alternative to the drag'n' drop editing of slices
- Probability - 0: slice will never play 100: slice will always play
- Two PEER controllers
  (can be configured to control any buzz machine.)

To activate, say the volume envelope click the button labeled "Volume". Notice that the button changes color to indicate that the green

volume envelope is editable. Clicking the button again will let the volume envelope remain visible but not editable (so that you can view the volume envelope while editing the panning envelope)
Peer controllers are configured by right clicking the machine and using the assignments menu (this procedure might change slightly)

Beneath the envelope selection buttons you find a bunch of icons. The lock is for locking all sliders in the envelopes (make them move together), the rest of them can be used to apply common envelopes (rising, falling, random, etc.) the slider below the icons define the bottom and top values (it's a double slider).

Fine tuning your slices

You can change slice points by dragging them. Move the pointer to one of the lines (upper waveform) and drag with the left mouse button. There is an auto quantize feature that will move the slicepoints to the closest edge in the sound.
If you set the tempo or pitch to low values, slices will overlap - if this gets in your way you can use CTRL + left mouse button to change the decay of a slice, or use the DECAY envelope

Slice Preview

Click the right mouse button to preview a slice. If you click in the top loop, the sound will be played in the original tempo, and playback will loop the entire file.
Click in the lower loop to preview a slice with all length / volume / pitch / pan modifications

Wavetable Import / Export

Import
Type in the ID of the wave you want to import and click the wavetable icon. ID's are in hex - be aware that buzz sometimes displays the wavetable ID's in decimal - all ID's revert to hex when a file is reloaded however :-)
Note that you can move a loop that has been loaded as a file, to the wavetable by RIGHT-clicking the wavetable button - useful if you loaded from file, and want to move your bmx file to another computer.
Export
pressing the save-button with the little waveform label will export the slice arrangement (what you see in the bottom most waveform display) as well as all individual slices to wavetable slots starting from position 40(hex). The start position can be configured as an attribute - note that attributes are always decimal, but I guess most buzz users are quite skilled in hex / decimal conversions...

The slices are exported without any envelope modifications (pitch, volume, pan, etc.) - if you want each slice as they sound, re-import the exported loop into liveslice and do another export.

Buzz syncronization

By default LiveSlice will sync to the buzz tempo. However buzz does not report it's exact playback position to the machines so LiveSlice needs a little help to tighten the synchronization. To do this create a pattern that resets the playback position to slice 0 at regular intervals (there is a trigger parameter for that).
If you use alternating TPB values to control the groove, you need to create patterns with the "sliceposition" parameter, ranging from 0 the number of slices you are using.
If you change the bpm often, you need to turn off the BPM synchronization of live slice by setting the "Sync to bpm" parameter to zero.

Recording

There are two recording modes in LiveSlice.

- When you press record in buzz, all preset changes and mute/ unmute changes will be recorded to the current buzz pattern. This is useful for arranging your presets. First create a number of different grooves (presets) straight - stop-time - break - variations - silent, and then play them using the keyboard, and record them to a pattern.
- When the record button in liveslice is pressed, everything you hear will be recorded to presets 25-32 (the last row of presets). The recording continues, overwrites existing presets, until the record button is pressed again.

Keyboard shortcuts

Preset control
the preset shortcuts are layd out as on the screen:

the **first row of keys** (1-8) selects the first row of presets (1-8)
the **second row of keys** (a-k) selects the second row of presets (9-16)
etc.

Save preset: hold down SHIFT while pressing the corresponding key to

Slide display mode:	TAB
Toggle Mute:	Return
Toggle Randomize:	SPACE
Toggle Recording:	[.] (the dot)
Envelope values:	Arrow keys (up/down changes value, left/right selects slice)

Parameter desription

In addition to the GUI, Live Slice can be controlled by various buzz parameters

- Sync to bpm
  Can be used to disable sync to buzz tempo
- Playback pos
  Forces live slice to play the slice at a particular position in current arrangement
- Current preset
  Switch to one of the 16 presets
- Manual
  Override the preset in the GUI (tells live slice not to read from the preset)
- Play slice
  Manually play a particular slice, overriding the preset
- Reverse slice
  reverses the slice about to be played
- Wavetable slot
  defines the wavetable slot for the import function
- Import wave
  from selected wavetable slot
- Randomize
  randomiz the slices in the GUI
- Revert
  go back to linear ordering of slices
- Global Pitch
  A note that controls the global pitch (C-4 is the default)

- Inertia
  not yet implemented
- Reset position
  starts playing back from position 0 in the slice arrangement
- Mute
  mutes the sound

Track parameters (commands)

Each track contains a caommand / value pair. Possible commands are:

cmd	description	value
00	load wavetable	slot
01	num. measures	1-4
02	num. slices	4-32
03	quantize	sensitivity (0-254)
10	randomize envelope	envelope 1-8 (1: vol. 2: pan, etc)
11	reset envelope	1-8
12	set evelope to "rising"	1-8
13	set evelope to "falling"	1-8
14	set evelope to "gap1"	1-8
15	set evelope to "gap2"	1-8
16	edit envelope	1-8
17	show envelope	1-8
18	hide envelope	1-8
1A	set minimum envelope value	0-254
1B	set maximum envelope value	0-254

Attributes:

Internal tempo
Pr. default liveslice will follow the buzz tempo, and will do some calculations on bpm changes - this is not good if you use bpm changing groovemachines. Therefore you can use this to disable bpm sync and have liveslice define it's own bpm

Crossfade length
Each slice is faded in and out to avoid clicks in the sound. The slices are faded in before they actually start, to avoid getting slow attacks (on snares for instance), but maintain accurate timing. This attribute defines the lenght of these fades

Pitch envelope range
In semitones - up to 240 semitones ! good to turn everything into clicks

Interpolation
0 = none (cool lofi sound)
1 = simple linear interpolation

First wavetable slot for export:
defines the first slot for the export function

Play preview:
remember to disable previews if you want to load waves in a live situation

Disable double buffering:
saves CPU, display might flicker a bit - use this if you have to use the MME driver and / or your computer is slow

Cut slices on end:
If the pitch is lowered one octave, all slices are twice as long - this can be changed using the length envelope, or by setting this attribute to 1 and force all slices to be muted before the next slice starts.

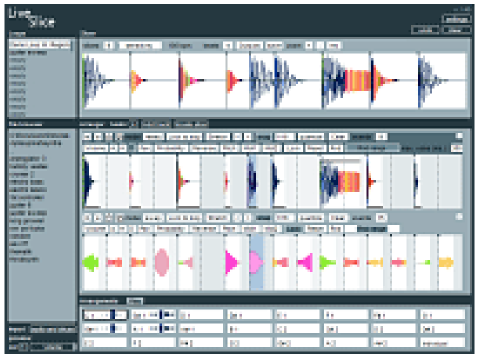
Need more LiveSlice?

Why not try LiveSlice VST - the next step in the slicer evolution. It has tons of new features:

- a full sequencer with multiple tracks
- time stretching
- auto slice and wav. editing
- multiple outputs
- audio recording
- imports industry standard loops

all this without sacrificing the speed of the workflow and the ease of use that you like from the buzz version.

All this comes at a price of course, but it's a very reasonable one :) I'm not in the habit of ripping off poor musicians like myself, and the money does indeed help to encourage future development (as you can see from the new feature set).



Live v0.1 (beta) - Very short tutorial
----

1. What is Live?

Live is a Global Controller machine. It's similar to Jeskola C1 but different in many ways. I designed it as a tool for live performances. The main function is to define programs which represent the parameter settings for every used machine. Theese programms can be activated by midi note. But thats not everything...

2. Installation

Live is a generator. Copy Live.dll and Live.txt into the Generators folder. It requires the latest Buzz.exe. Get it at www.jeskola.com/beta. In the index.txt it should be found under the section 'Global Controler'.

3. Setup in Buzz

Insert Live like any other machine. Connect the output direct to the Master.
A double-click on the machine opens the GUI.

4. The GUI elements

The windows is separated in different groups:

Machines	contains all used machines
Parameters	contains all parameters of the selected (white) machine
Sliders	visualized parameter values for data entry
Program	a list of the 128 possible programs
Song	contains the program-no. for each of the 256 song steps

A left click selects an item. Scroll a list with a right button and moving up/down. The sliders are set with left click. With a right click the can be deactivated (not used, not sended).

At the bottom of the window are several buttons:

Setup	opens the setup-window for midi-channel and keyboard-transpose

X (Param.)	cuts all used values of the selected machine into the clipboard
C (Param.)	copies all used values of the selected machine into the clipboard
V (Param.)	inserts the clipboard values and sends them

1	sets all steps as used (only in sequencer mode)
2	sets every second as used (only in sequencer mode)
4	sets every fourth step as used (only in sequencer mode)
8	sets every eighth step as used (only in sequencer mode)
RS	sets steps used by random (only in sequencer mode)
RV	sets values for the used steps by random (only in sequencer mode)

Run	activates playing the defined sequences

X (Prog.)	cuts the selected program into clipboard
C (Prog.)	copies the selected program into clipboard
V (Prog.)	inserts the clipboard into the selected program and sends it

...	opens window to rename the current program

Set	sets the selected program to the current song step
-	decrements the counter (to repeat a song step)
+	increments the counter (to repeat a song step)
Ins	inserts program 0 at current song step
Del	deletes the current song step
Run	activates song playing

5. How to work with Live?

Make your setup by inserting and connecting the machines/templates you like. You need no patterns (F2) or sequence (F4). Insert Live and connect it direct to the master. Open the GUI. Try selecting the machines and play around with the sliders. Changes are directly (every tick) send to Buzz. To save a program select another one. Programs are sended by selecting them. You can recall any program with your midi keyboard. To enter the step-sequencer for a parameter click it twice (it turns red). Now you can draw the sequence.

Have fun
Holger Zwar (maekflai@aol.com) bug-reports and ideas welcome

Looper Generator

Looper allows you to add loops to your songs easily (like Acid) because it stretches them to be the proper time according to the tempo. It does NOT pay attention to loop points (instead it loops the whole sample. Make sure your sample is trimmed properly.

Feedback, suggestions, bugs:

dwallin@planetquake.com

coming soon: www.buzztrack.com

M3 - a BUZZ Plugin

This is a short explanation of the M3 buzz plugin. It consists of 4 main sections: the oscillator section, the amplifier section, the filter section and the LFO section.

The oscillator section consists of 2 oscillators and a sub oscillator. With the wave controllers you set their wave forms. the random options sets every new note a new waveform except noise (was uncool ;). The pulse width sliders are not only for the pulse (square) waves but for all (except noise of course). They simply move the centers of the waves to left/right. Oscillator 2 is detuneable with 2 sliders (semi and fine) and syncable to oscillator 1 (master). The sub oscillator oscillates always one octave below oscillator 1.
With the mix slider you control the balance between both. The mixtype fader is for controlling the kind of mixture. Normally you use the simple 'add', but the others sound somehow cool, i think. The Pitch Envelope Generator (PEG) with its simple linear AR-curve bears upon all 3 oscillators (1/2/sub). The Glide controller is the speed of portamento (0 = off).

The amplifier section consists of a linear ASR curve. The sustain slider is NOT the sustain level but the sustain time. The level is controlled by the volume parameter.

In the filter section you can choose between lowpass/highpass with resonance adjustment and bandpass/bandreject(notch) with bandwidth controlling. You also have a ASR curve for modulating the cutoff/center frequency.

Finally you have 2 LFOs (low frequency oscillators) for modulating different parameters of the oscillators, filter and amplifier. The waveforms are the normal ones, but if you choose noise the LFO holds one random value for the time of one cycle. You can modulate different parameters by 1 LFO at the same time but only with the same amount.

Hope that was enough. Sorry for bad english. Have fun

MAKK makk@gmx.de

M4 documentation

The M4 machine consists of four main sections:
the oscillator section,
the amplifier section,
the filter section and
the Low Frequency Oscillator (LFO) section.

Oscillator section

You have here two main oscillators (Osc1 and Osc2) and a sub oscillator (SubOsc).
The sub oscillator always oscillates one octave (12 halfnotes) below oscillator 1. Its level you can control with the SubOsc Vol controller.
Oscillator 2 is detuneable with the Osc2 SemiDet and Osc2 FineDet sliders and can be synced by oscillator 1 (Osc2 Sync).
For each oscillator you can choose their wave form with the Wave parameter (Osc1 Wave, Osc2 Wave and SubOsc Wave).
The pulse width controllers (Osc1 PW and Osc2 PW) move the centers of the waves to left/right. Below you see a sine wave with pulse width 50:50 and with pulse width about 70:30. Hope you understand the principle.

With the Osc Mix slider you control the balance between oscillator 1 and oscillator 2. The Osc MixType fader is for controlling the kind of mixture.
The oscillator section contains a Pitch Envelope. The attack and decay time can be adjusted with the Pitch Attack and Pitch Decay faders, the strength of the envelope modulation with Pitch EnvMod. Negative envelope modulation is also possible.
With the Pitch Glide slider you control the speed of portamento. A value of 0 means no portamento, 127 is a very slow glide effect.

Amplifier section

The amplifier has a linear Attack-Sustain-Release envelope curve. The Amp Attack, Amp Sustain and Amp Release faders control the attack, sustain and release time of the amplitude enevelope. The volume (sustain level) is a non global parameter and is controlled per track.

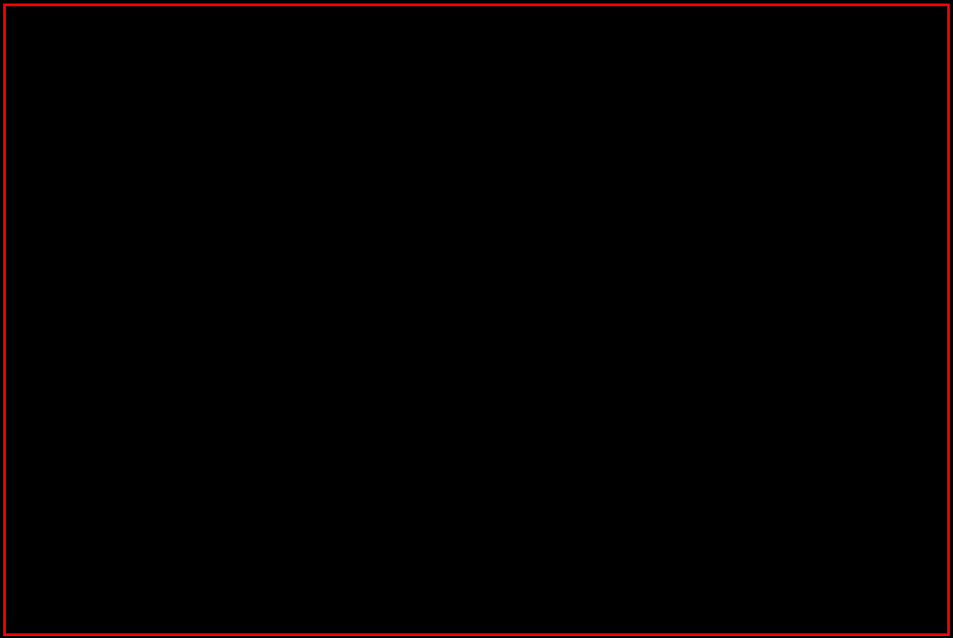
Filter section

In the filter section you have lowpass filter with -12db, -18db and -24db/octave and highpass filter with resonance adjustment and bandpass/bandreject(notch) filter with bandwidth controlling.
Choose one of them with Filter Type. The cutoff/center frequency you control with the Filter Cutoff slider, the resonance or bandwidth with Filter Q/BW (Q=Quality=Resonance, BW=BandWidth).
The filter has an envelope for the cutoff/center frequency. Set the attack, sustsain and release time with Filter Attack, Filter Sustain and Filter Release, the strength of modulation with Filter EnvMod. Negative

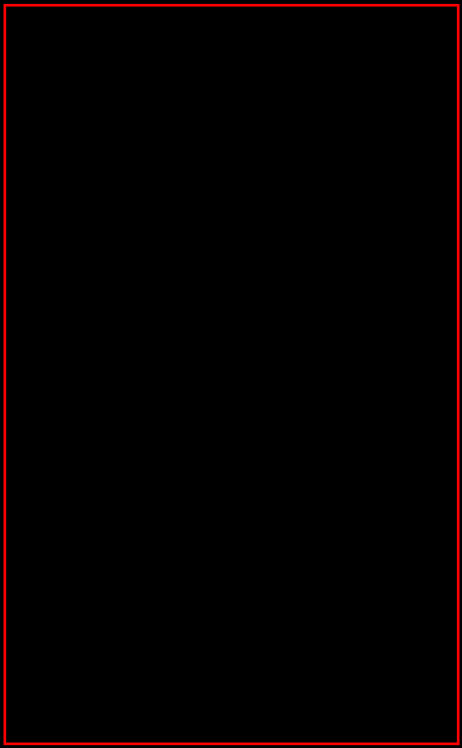
envelopes are possible.

Low Frequency Oscillator section

The M4 machine has two low frequency oscillators (LFO1 and LFO2).
A LFO is a very slow oscillator which can modulate different parameters of the oscillators, amplifier and filter.
LFO1 Dest and LFO2 Dest (Dest = Destination) specify, which parameter(s) the LFOs modulate.
LFO1 can modulate the frequency of oscillator 1, the pulse width of oscillator 1, the volume and the filter cutoff/center frequency.
LFO2 can modulate the frequency of oscillator 2, the pulse width of oscillator 2, the oscillator mix and the filter resonance/bandwidth.
LFO1 Wave and LFO2 Wave specify the waveform of the LFOs. If random is selected, the LFO holds one random value for the time of one cycle (sample and hold).
With LFO1 Freq and LFO2 Freq you set their frequencies in Hertz (0-116) or in Ticks (117-127).
The strenght of the modulation of the selected parameters by the LFOs you set with the LFO1 Amount and LFO2 Amount sliders.
In the Attributes window you can scale the amounts for the different parameters. from 0% (0) to 100% (127).



With this feature you can for example modulate the pulse width of oscillator 1 only very slightly (LFO1 PulseWidth1 Scale = 13 (about 10%)) and the filter cutoff frequency very strong (LFO1 Cutoff Scale = 127 (100%)) with the same LFO.
With the LFO1 Ph Diff and LFO2 Ph Diff you can get very nice sounding results. Ph Diff means Phase Difference, so what these parameters do is setting different LFO phases for every track.
Below you see a picture of a sine LFO wave for three tracks with a phase difference of 90°, which means 90° difference between track 0 and 1, 90° difference between track 1 and 2. If this LFO modulates the cutoff frequency for example, you have different cutoff frequencies for each track. Hope you understand.



I've spend really much time to write this doc, so I hope it helps you.
happy sounding!

Makk (makk@gmx.de)

Magicfish IT Sample Loader v1.1

by Austin Luminais

This machine is used to load samples from an .IT module file into the wavetable. It replaces "Lipid IT Loader", which no longer works in newer versions of Buzz.

In the context menu that appears when you right click on the machine is the command "Import IT Samples". This will allow you to choose an .IT file, and then all of the samples in that module will be loaded into the wavetable. The samples will be appended; any waves currently in the wavetable will not be overwritten. When you are finished loading samples you can remove the machine.

New in version 1.1: Compressed samples are now supported!

This plug-in is freeware.

E-mail: **lipid@magicfish.net**

Website: ***http://www.magicfish.net***

Matilde Tracker v1.8 for Buzz

Introduction

Matilde Tracker is a tracker machine for Buzz which behaves more like Protracker than Jeskola Tracker.

All Protracker effects that make sense in Buzz are implemented and behave in a similar fashion to their Protracker cousins, so you'll feel right at home. The E/xy effects have been renamed to Ex/yy, read on.

Additionally you now have two effects per note, life is sweet!

The machine is a stereo generator, so you need Buzz 1.2 or later to use it. In fact, you'll probably need the latest beta too. If you think you need the newest version of Buzz, go to www.buzzmachines.com. You'll also auxbus.dll. If stereo is annoying, there's a mono version you can use instead.

Volume, pitch and panning envelopes are supported, they take 64 ticks to complete. If this value is undesirable, it can be changed in the machine's attributes.

Have fun,
Carsten Sørensen

Known bugs - in order of severity, worst first:

- Playing with samples that are not present in the wavetable might crash buzz. This also affects peer stuff badly.
- Slider positions are not initialized correctly on loading new module, unless you progam their settings in the pattern.
- Note delay (ED) cannot be combined with certain effects.

Requirements for going open-source

- The stand-alone resampler part must be kept as a replaceable stand-alone component, because SurfSmurf has future plans with this.
- MTrk must also be functional without the Overloader (even tho the absense of Overloader-specific features is acceptable)
- Stability must have priority over adding features in the development of MTrk.

What's new in v1.8?

(v1.8 additions by Kibibu and Joachim Michaelis)

- Seperate tuning attributes for each note
- Volume ramp now 10x more precise. Default value is now 0ms instead of 1ms. Increase to 10 or 20 if you encounter clicks.

What's new in v1.7beta?

(All v1.7 additions by Joachim Michaelis)

- Attribute "Filter Mode" is now 2 by default instead of 1
- Changed NO_VALUE ass suggested by zeffii

What's new in v1.6beta?

(All v1.6 additions by Joachim Michaelis)

- Changed name to avoid messing up old Buzz songs
- Volume now behaves more logaritmic - like analog gear
- Slider: Amplitude decay
- Slider: Sample offset
- Slider: Offset mode (quantized or free percussion-suitable)
- Slider: Tuning
- Command: 2F long loop fit
- Command: 30 probability without note off
- Attribute: Long loop fit factor
- Attribute: Offset volume gain

What was new in v1.5?

- Virtual channels! Read more below.
- New combined note delay and cut command (18)
- New sustain pedal command (19)
- New note cut (DC) command that releases note instead of setting volume to zero
- Note delay (ED) command can now delay note off's
- Bug fixed with looping stereo samples
- About window made bigger
- Bug fixed with some effects causing bi-di loops to mess up
- Fixed CPU use problem with very volumes (<= -100dB)

What was new in v1.3?

- Small bug fixed regarding volume envelopes whose first point was zero
- Bi-directional loops, works when Buzz 1.2 is released
- Wavetable screen wave playing supported
- Wavetable screen envelope cursor supported
- Randomize volume command (14)
- Random delay command (15)
- Randomize pitch command (16)
- Harmonic play command (17)
- Spline interpolation (check attributes)
- Envelopes for pan and pitch
- Attribute for pitch envelope depth

Improved MIDI support:

- Attribute for MIDI wave to use
- MIDI now uses full polyphony **or** free tracks only, yet another attribute
  V2 TODO: Two virtual channels per track, selectable via attributes. V2 TODO: WAV harddisk streaming V2 MAYBE: Timestretch

What was new in v1.2?

- **Stereo wave support added.** You need the "latest" Buzz to use it.
- Yet another reloading-waves-while-playing crash fixed.
- Continued arpeggio problem fixed
- Subdivision precision improved
- Loop points are now set the same way as Jeskola Tracker
- Default name changed by popular request
- MIDI support added
- Shuffle command added (command 13)
- Loop fit command added (commands 11 and 12)
- F/00 crash fixed
- Command 2 crash fixed
- WM errors fixed

What was new in v1.1?

- Probability command (10)
- Sustain point and note off handled correctly
- Reloading-waves-while-playing crash fixed

Virtual Channels

Matilde Tracker supports "virtual channels." It's a bit like NNAs in Impulse Tracker, but on the other hand, it's not. To use them, you must enable them for the tracker machine in question - this is done through an attribute (see below.)

When enabled, the tracker supports 64 note polyphony, in much the same way a hardware sampler handles it. What it does exactly, is when you play a new note the last one is allowed to finish. This is great for chord riffs, plucked strings and a bunch of other stuff. So now, instead of going through the hassle of using more tracks, you can enable virtual channels instead.

Sometimes the note will be cut anyway to make sure no samples sit there and take up all the channels. This happens when you have a looping sample with no suitable volume envelope. This is a safety measure and completely intentional.

Using MIDI

Setting up MIDI input is now very easy - everything is controlled from the attributes.

You'll have to set up the attributes to the right channel. Now, the machine will respond to MIDI in events.
The tracks play the note with the last instrument used in the track, so a track will have to have played something before any sound is made. This is normally not what you want, using an attribute you can select the wave MIDI events will use instead.

The default behaviour for MIDI events is to use all available tracks in succession from left to right, including all unused tracks. You can change the behaviour using an attribute to use unused tracks **only** so you can jam along with your track.

For more information, refer to the attribute descriptions below.

Machine attributes

Attribute	Description
Volume Ramp	The time in milliseconds micro volume-ramping takes. This is to avoid clicks.
Volume Envelope Span	The number of ticks a volume envelope spans from left to right.
MIDI Channel	The MIDI channel the machine responds to. 0 means it doesn't respond.
MIDI Velocity Sensitivity	How sensitive the machine is to MIDI velocity. 0 means it doesn't care, 256 means it cares a lot ;)
MIDI Wave	MIDI notes will use this waveform.
MIDI Uses Free Tracks	If 1 - MIDI notes will only use free tracks. If not, all tracks.
Filter	Filtering mode when playing waveforms. 0=none, 1=linear, 2=spline. Spline is much slower than linear, so only use it if you can handle it.
Pitch Envelope Depth	Pitch envelope will range between +- this attribute.
Enable Virtual Channels	0=normal behaviour, 1=virtual channels enabled
Long loop fit factor	Adjust the length of the units of the 2F command. If you specify 1, the 2F command will be just like the 12 command. If you specify 2, it will loop fit twice as long breakbeats etc.
Offset volume gain	Drum sounds ususally fade out towards the end, so when you use the offset slider, the sound will attenuate. This attribute can resolve this problem by raising the volume correspondingly. The default value of 10 will cause a slight increase in volume, when you use the Offset slider in "percussion" mode. 0=volume is unchanged regardless of the Offset slider's position.

Pattern editor

Column	Description
1	The note to play
2	The current wave. If empty, the previously selected wave will be used
3	Volume. 0=mute, 40=half, 80=full, FE=about double the volume
4 and 6	Effect to use, see below for description
5 and 7	Argument to the effect

Obsolete/changed/ignored Protracker/FT2 effects

Command	Description	Reason
5/xy	Keep tone portamento'ing and do volume slide with argument	Obsolete, there's two effects columns
6/xy	Keep vibrato'ing and do volume slide with argument	Obsolete, there's two effects columns
B/xx	Break to position	Not possible, machines have no control over the songposition
C/xx	Set volume	Replaced by the volume parameter
D/xx	Break to next position, step *xx*	Not possible, machines have no control over the songposition
E/0x	Set filter state	Not implemented, use a filter machine instead
E/3x	Set glissando	Not implemented
E/43, E/47	Set vibrato waveform to noise	Not implemented
E/6x	Pattern loop	Not possible, machines have no control over the songposition
E/73, E/77	Set tremolo waveform to noise	Not implemented
E/Ex	Patterndelay	Not possible, machines have no control over the songposition
E/Fx	Invert loop	Not implemented

Implemented effects

Command	Argument	Description	Notes
00	xy	Arpeggio On subdivision step 0, the original note will be played. Step one will play the original note+*x* halftones. Step two will play the original note+*y* halftones. Step three will restart the process	
01	xx	Slide up *xx* notches	
02	xx	Slide down *xx* notches	
03	xx	Tone portamento If *xx* is zero, keep portamento'ing	
04	xy	Vibrato *x* - speed *y* - depth If either is zero, use previous value	
05	xy	Slide panning *x* - amount to slide panning left *y* - amount to slide panning right	
06	xy	Autopan *x* - speed *y* - depth If either is zero, use previous value	

07	xy	Tremolo *x* - speed *y* - depth If either is zero, use previous value	
08	xx	Set panning position 0=left, 80=middle, FF=right	
09	xx	Sample offset *xx* - offset into sample. Unlike Protracker this is not an absolute offset but scales to the whole length of the sample, ie a value of 80 will start from the middle of the sample. If there's no argument, the sample offset will be set right at the end, useful for E8/01.	Use with a note
0A	xy	Volume slide *x* - amount to slide volume up *y* - amount to slide volume down	
0F	xx	Subdivide amount This is the same as the Protracker Fxx command, except it doesn't actually change the speed of the song, only the speed of the track's effects. If the subdivide amount is higher, effects will be updated more often, making them run faster. The default value is 6.	
10	xx	Probability (with note-off) *xx* - Probability for sample being played. 01=will almost certainly not be played, 80=50%, FF=almost certain. When notes are not played, the previous wave is interrupted. (see also command 30)	Use with a note
11	xx	Loop fit *xx* - Number of ticks the waveform's loop should take to complete. Changes the frequency of the waveform.	Use with a note
12	xx	Loop fit *xx* - Same as 11, but tracks song speed changes and adjust the frequency accordingly.	Use with a note
13	xy	Auto shuffle *x* - Ticks to shuffle. 2 shuffles every other step, 3 every third step and so on. y=Shuffle amount. 0=none, F=almost a full tick. change the subdivision amount to, say, 10 for greater precision.	
14	xx	Randomize volume *xx* - Maximum amount the volume will be randomized	
15	xx	Random delay *xx* - Maximum number of subdivision steps the note will be delayed	Use with a note
16	xx	Randomize pitch *xx* - Maximum number of notches the pitch will be randomized	
17	xx	Harmonic play *xx* - The base frequency will be multiplied by xx	

18	xy	Combined note delay and cut *x* - The subdivision step to trigger the note *y* - The subdivision step to release the note	Use with a note
19	xy	Sustain pedal *y* - Subdivision step to trigger command *x*=1 - Depress sustain pedal *x*=2 - Release sustain pedal	
2F	xx	Long loop fit *xx* - Same as 12, but multiplied by 128	Use with a note
30	xx	Probability (without note-off) *xx* - Probability for sample being played. 01=will almost certainly not be played, 80=50%, FF=almost certain	Use with a note
DC	xx	Note cutoff, releases note	
E1	xx	Fine slide up *xx* notches	
E2	xx	Fine slide down *xx* notches	
E4	0x	Set vibrato type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
E5	xx	Set finetune for track 00 = -1/2 halfnote, 80 = 0, FF = ~+1/2 halfnote	
E6	0x	Set panning type See E4/0x for parameter	
E7	0x	Set tremolo type See E4/0x for parameter	
E8	01	Reverse direction of sample being played	
E9	xx	Retrig sample *xx* - subdivision count at which sample is retriggered	Use with a note
EA	xx	Fine volume slide up	
EB	xx	Fine volume slide down	
EC	xx	Note cutoff, set volume to zero	
ED	xx	Note delay Delay samplestart for *xx* subdivision steps	Use with a note
EE	xx	Fine panning slide left	
EF	xx	Fine panning slide right	

Matilde Tracker v1.5 for Buzz

Introduction

Matilde Tracker is a tracker machine for Buzz which behaves more like Protracker than Jeskola Tracker.

All Protracker effects that make sense in Buzz are implemented and behave in a similar fashion to their Protracker cousins, so you'll feel right at home. The E/xy effects have been renamed to Ex/yy, read on.

Additionally you now have two effects per note, life is sweet!

The machine is a stereo generator, so you need Buzz 1.2 or later to use it. In fact, you'll probably need the latest beta too. Replace your buzz.exe with the one here. You'll also auxbus.dll. If stereo is annoying, there's a mono version you can use instead.

Volume, pitch and panning envelopes are supported, they take 64 ticks to complete. If this value is undesirable, it can be changed in the machine's attributes.

Have fun,
Carsten Sørensen

What's new in v1.5?

- Filter, yay!
- Virtual channels! Read more below.
- New combined note delay and cut command (18)
- New sustain pedal command (19)
- New note cut (DC) command that releases note instead of setting volume to zero
- Note delay (ED) command can now delay note off's
- Bug fixed with looping stereo samples
- About window made bigger
- Bug fixed with some effects causing bi-di loops to mess up
- Fixed CPU use problem with very volumes (<= -100dB)

What was new in v1.3?

- Small bug fixed regarding volume envelopes whose first point was zero
- Bi-directional loops, works when Buzz 1.2 is released
- Wavetable screen wave playing supported
- Wavetable screen envelope cursor supported
- Randomize volume command (14)
- Random delay command (15)

- Randomize pitch command (16)
- Harmonic play command (17)
- Spline interpolation (check attributes)
- Envelopes for pan and pitch
- Attribute for pitch envelope depth

Improved MIDI support:

- Attribute for MIDI wave to use
- MIDI now uses full polyphony **or** free tracks only, yet another attribute
  V2 TODO: Two virtual channels per track, selectable via attributes. V2 TODO: WAV harddisk streaming
  V2 MAYBE: Timestretch

What was new in v1.2?

- **Stereo wave support added.** You need the "latest" Buzz to use it.
- Yet another reloading-waves-while-playing crash fixed.
- Continued arpeggio problem fixed
- Subdivision precision improved
- Loop points are now set the same way as Jeskola Tracker
- Default name changed by popular request
- MIDI support added
- Shuffle command added (command 13)
- Loop fit command added (commands 11 and 12)
- F/00 crash fixed
- Command 2 crash fixed
- WM errors fixed

What was new in v1.1?

- Probability command (10)
- Sustain point and note off handled correctly
- Reloading-waves-while-playing crash fixed

Virtual Channels

Matilde Tracker supports "virtual channels." It's a bit like NNAs in Impulse Tracker, but on the other hand, it's not. To use them, you must enable them for the tracker machine in question - this is done through an attribute (see below.)

When enabled, the tracker supports 64 note polyphony, in much the same way a hardware sampler handles it. What it does exactly, is when you play a new note the last one is allowed to finish. This is great for chord riffs, plucked strings and a bunch of other stuff. So now, instead of going through the hassle of using more tracks, you can enable virtual channels instead.

Sometimes the note will be cut anyway to make sure no samples sit there and take up all the channels. This happens when you have a looping sample with no suitable volume envelope. This is a safety measure and completely intentional.

Using MIDI

Setting up MIDI input is now very easy - everything is controlled from the attributes.

You'll have to set up the attributes to the right channel. Now, the machine will respond to MIDI in events.
The tracks play the note with the last instrument used in the track, so a track will have to have played something before any sound is made. This is normally not what you want, using an attribute you can select the wave MIDI events will use instead.

The default behaviour for MIDI events is to use all available tracks in succession from left to right, including all unused tracks. You can change the behaviour using an attribute to use unused tracks **only** so you can jam along with your track.

For more information, refer to the attribute descriptions below.

Machine attributes

Attribute	Description
Volume Ramp	The time in milliseconds micro volume-ramping takes. This is to avoid clicks.
Volume Envelope Span	The number of ticks a volume envelope spans from left to right.
MIDI Channel	The MIDI channel the machine responds to. 0 means it doesn't respond.
MIDI Velocity Sensitivity	How sensitive the machine is to MIDI velocity. 0 means it doesn't care, 256 means it cares a lot ;)
MIDI Wave	MIDI notes will use this waveform.
MIDI Uses Free Tracks	If 1 - MIDI notes will only use free tracks. If not, all tracks.
Filter	Filtering mode when playing waveforms. 0=none, 1=linear, 2=spline. Spline is much slower than linear, so only use it if you can handle it.
Pitch Envelope Depth	Pitch envelope will range between +- this attribute.
Enable Virtual Channels	0=normal behaviour, 1=virtual channels enabled

Pattern editor

Column	Description

1	The note to play
2	The current wave. If empty, the previously selected wave will be used
3	Volume. 0=mute, 40=half, 80=full, FE=about double the volume
4 and 6	Effect to use, see below for description
5 and 7	Argument to the effect

Obsolete/changed/ignored Protracker/FT2 effects

Command	Description	Reason
5/xy	Keep tone portamento'ing and do volume slide with argument	Obsolete, there's two effects columns
6/xy	Keep vibrato'ing and do volume slide with argument	Obsolete, there's two effects columns
B/xx	Break to position	Not possible, machines have no control over the songposition
C/xx	Set volume	Replaced by the volume parameter
D/xx	Break to next position, step *xx*	Not possible, machines have no control over the songposition
E/3x	Set glissando	Not implemented
E/43, E/47	Set vibrato waveform to noise	Not implemented
E/6x	Pattern loop	Not possible, machines have no control over the songposition
E/73, E/77	Set tremolo waveform to noise	Not implemented
E/Ex	Patterndelay	Not possible, machines have no control over the songposition
E/Fx	Invert loop	Not implemented

Implemented effects

Command	Argument	Description	Notes
00	xy	Arpeggio On subdivision step 0, the original note will be played. Step one will play the original note+*x* halftones. Step two will play the original note+*y* halftones. Step three will restart the process	

01	xx	Slide up *xx* notches	
02	xx	Slide down *xx* notches	
03	xx	Tone portamento If *xx* is zero, keep portamento'ing	
04	xy	Vibrato *x* - speed *y* - depth If either is zero, use previous value	
05	xy	Slide panning *x* - amount to slide panning left *y* - amount to slide panning right	
06	xy	Autopan *x* - speed *y* - depth If either is zero, use previous value	
07	xy	Tremolo *x* - speed *y* - depth If either is zero, use previous value	
08	xx	Set panning position 0=left, 80=middle, FF=right	
09	xx	Sample offset *xx* - offset into sample. Unlike Protracker this is not an absolute offset but scales to the whole length of the sample, ie a value of 80 will start from the middle of the sample. If there's no argument, the sample offset will be set right at the end, useful for E8/01.	Use with a note
0A	xy	Volume slide *x* - amount to slide volume up *y* - amount to slide volume down	
0F	xx	Subdivide amount This is the same as the Protracker Fxx command, except it doesn't actually change the speed of the song, only the speed of the track's effects. If the subdivide amount is higher, effects will be updated more often, making them run faster. The default value is 6.	
10	xx	Probability *xx* - Probability for sample being played. 01=will almost certainly not be played, 80=50%, FF=almost certain	Use with a note
11	xx	Loop fit *xx* - Number of ticks the waveform's loop should take to complete. Changes the frequency of the waveform.	Use with a note

12	xx	Loop fit *xx* - Same as 11, but tracks song speed changes and adjust the frequency accordingly.	Use with a note
13	xy	Auto shuffle *x* - Ticks to shuffle. 2 shuffles every other step, 3 every third step and so on. y=Shuffle amount. 0=none, F=almost a full tick. change the subdivision amount to, say, 10 for greater precision.	
14	xx	Randomize volume *xx* - Maximum amount the volume will be randomized	
15	xx	Random delay *xx* - Maximum number of subdivision steps the note will be delayed	Use with a note
16	xx	Randomize pitch *xx* - Maximum number of notches the pitch will be randomized	
17	xx	Harmonic play *xx* - The base frequency will be multiplied by xx	
18	xy	Combined note delay and cut *x* - The subdivision step to trigger the note *y* - The subdivision step to release the note	Use with a note
19	xy	Sustain pedal *y* - Subdivision step to trigger command *x*=1 - Depress sustain pedal *x*=2 - Release sustain pedal	
20	xx	Filter cutoff	
21	xx	Slide filter cutoff up *xx* notches	
22	xx	Slide filter cutoff down *xx* notches	
23	0x	Set filter cutoff LFO type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
24	xy	Filter cutoff LFO *x* - speed *y* - depth If either is zero, use previous value	
25	xx	Fine slide filter cutoff up *xx* notches	
26	xx	Fine slide filter cutoff down *xx* notches	
28	xx	Filter resonance	

29	xx	Slide filter resonance up *xx* notches	
2A	xx	Slide filter resonance down *xx* notches	
2B	0x	Set filter resonance LFO type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
2C	xy	Filter resonance LFO *x* - speed *y* - depth If either is zero, use previous value	
2D	xx	Fine slide filter resonance up *xx* notches	
2E	xx	Fine slide filter resonance down *xx* notches	
DC	xx	Note cutoff, releases note	
E0	xx	Set filter type for track. *xx* - 1 - filter disabled *x* - 2 - 4p filter, lowpass *x* - 3 - 4p filter, highpass	
E1	xx	Fine slide up *xx* notches	
E2	xx	Fine slide down *xx* notches	
E4	0x	Set vibrato type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
E5	xx	Set finetune for track 00 = -1/2 halfnote, 80 = 0, FF = ~+1/2 halfnote	
E6	0x	Set panning type See E4/0x for parameter	
E7	0x	Set tremolo type See E4/0x for parameter	
E8	01	Reverse direction of sample being played	
E9	xx	Retrig sample *xx* - subdivision count at which sample is retriggered	Use with a note
EA	xx	Fine volume slide up	

EB	xx	Fine volume slide down	
EC	xx	Note cutoff, set volume to zero	
ED	xx	Note delay Delay samplestart for *xx* subdivision steps	Use with a note
EE	xx	Fine panning slide left	
EF	xx	Fine panning slide right	

Matilde Tracker v1.5 for Buzz

Introduction

Matilde Tracker is a tracker machine for Buzz which behaves more like Protracker than Jeskola Tracker.

All Protracker effects that make sense in Buzz are implemented and behave in a similar fashion to their Protracker cousins, so you'll feel right at home. The E/xy effects have been renamed to Ex/yy, read on.

Additionally you now have two effects per note, life is sweet!

The machine is a stereo generator, so you need Buzz 1.2 or later to use it. In fact, you'll probably need the latest beta too. Replace your buzz.exe with the one here. You'll also auxbus.dll. If stereo is annoying, there's a mono version you can use instead.

Volume, pitch and panning envelopes are supported, they take 64 ticks to complete. If this value is undesirable, it can be changed in the machine's attributes.

Have fun,
Carsten Sørensen

What's new in v1.5?

- Filter, yay!
- Virtual channels! Read more below.
- New combined note delay and cut command (18)
- New sustain pedal command (19)
- New note cut (DC) command that releases note instead of setting volume to zero
- Note delay (ED) command can now delay note off's
- Bug fixed with looping stereo samples
- About window made bigger
- Bug fixed with some effects causing bi-di loops to mess up
- Fixed CPU use problem with very volumes (<= -100dB)

What was new in v1.3?

- Small bug fixed regarding volume envelopes whose first point was zero
- Bi-directional loops, works when Buzz 1.2 is released
- Wavetable screen wave playing supported
- Wavetable screen envelope cursor supported
- Randomize volume command (14)
- Random delay command (15)

- Randomize pitch command (16)
- Harmonic play command (17)
- Spline interpolation (check attributes)
- Envelopes for pan and pitch
- Attribute for pitch envelope depth

Improved MIDI support:

- Attribute for MIDI wave to use
- MIDI now uses full polyphony **or** free tracks only, yet another attribute
  V2 TODO: Two virtual channels per track, selectable via attributes. V2 TODO: WAV harddisk streaming
  V2 MAYBE: Timestretch

What was new in v1.2?

- **Stereo wave support added.** You need the "latest" Buzz to use it.
- Yet another reloading-waves-while-playing crash fixed.
- Continued arpeggio problem fixed
- Subdivision precision improved
- Loop points are now set the same way as Jeskola Tracker
- Default name changed by popular request
- MIDI support added
- Shuffle command added (command 13)
- Loop fit command added (commands 11 and 12)
- F/00 crash fixed
- Command 2 crash fixed
- WM errors fixed

What was new in v1.1?

- Probability command (10)
- Sustain point and note off handled correctly
- Reloading-waves-while-playing crash fixed

Virtual Channels

Matilde Tracker supports "virtual channels." It's a bit like NNAs in Impulse Tracker, but on the other hand, it's not. To use them, you must enable them for the tracker machine in question - this is done through an attribute (see below.)

When enabled, the tracker supports 64 note polyphony, in much the same way a hardware sampler handles it. What it does exactly, is when you play a new note the last one is allowed to finish. This is great for chord riffs, plucked strings and a bunch of other stuff. So now, instead of going through the hassle of using more tracks, you can enable virtual channels instead.

Sometimes the note will be cut anyway to make sure no samples sit there and take up all the channels. This happens when you have a looping sample with no suitable volume envelope. This is a safety measure and completely intentional.

Using MIDI

Setting up MIDI input is now very easy - everything is controlled from the attributes.

You'll have to set up the attributes to the right channel. Now, the machine will respond to MIDI in events.
The tracks play the note with the last instrument used in the track, so a track will have to have played something before any sound is made. This is normally not what you want, using an attribute you can select the wave MIDI events will use instead.

The default behaviour for MIDI events is to use all available tracks in succession from left to right, including all unused tracks. You can change the behaviour using an attribute to use unused tracks **only** so you can jam along with your track.

For more information, refer to the attribute descriptions below.

Machine attributes

Attribute	Description
Volume Ramp	The time in milliseconds micro volume-ramping takes. This is to avoid clicks.
Volume Envelope Span	The number of ticks a volume envelope spans from left to right.
MIDI Channel	The MIDI channel the machine responds to. 0 means it doesn't respond.
MIDI Velocity Sensitivity	How sensitive the machine is to MIDI velocity. 0 means it doesn't care, 256 means it cares a lot ;)
MIDI Wave	MIDI notes will use this waveform.
MIDI Uses Free Tracks	If 1 - MIDI notes will only use free tracks. If not, all tracks.
Filter	Filtering mode when playing waveforms. 0=none, 1=linear, 2=spline. Spline is much slower than linear, so only use it if you can handle it.
Pitch Envelope Depth	Pitch envelope will range between +- this attribute.
Enable Virtual Channels	0=normal behaviour, 1=virtual channels enabled

Pattern editor

Column	Description

1	The note to play
2	The current wave. If empty, the previously selected wave will be used
3	Volume. 0=mute, 40=half, 80=full, FE=about double the volume
4 and 6	Effect to use, see below for description
5 and 7	Argument to the effect

Obsolete/changed/ignored Protracker/FT2 effects

Command	Description	Reason
5/xy	Keep tone portamento'ing and do volume slide with argument	Obsolete, there's two effects columns
6/xy	Keep vibrato'ing and do volume slide with argument	Obsolete, there's two effects columns
B/xx	Break to position	Not possible, machines have no control over the songposition
C/xx	Set volume	Replaced by the volume parameter
D/xx	Break to next position, step *xx*	Not possible, machines have no control over the songposition
E/3x	Set glissando	Not implemented
E/43, E/47	Set vibrato waveform to noise	Not implemented
E/6x	Pattern loop	Not possible, machines have no control over the songposition
E/73, E/77	Set tremolo waveform to noise	Not implemented
E/Ex	Patterndelay	Not possible, machines have no control over the songposition
E/Fx	Invert loop	Not implemented

Implemented effects

Command	Argument	Description	Notes
00	xy	Arpeggio On subdivision step 0, the original note will be played. Step one will play the original note+*x* halftones. Step two will play the original note+*y* halftones. Step three will restart the process	

01	xx	Slide up *xx* notches	
02	xx	Slide down *xx* notches	
03	xx	Tone portamento If *xx* is zero, keep portamento'ing	
04	xy	Vibrato *x* - speed *y* - depth If either is zero, use previous value	
05	xy	Slide panning *x* - amount to slide panning left *y* - amount to slide panning right	
06	xy	Autopan *x* - speed *y* - depth If either is zero, use previous value	
07	xy	Tremolo *x* - speed *y* - depth If either is zero, use previous value	
08	xx	Set panning position 0=left, 80=middle, FF=right	
09	xx	Sample offset *xx* - offset into sample. Unlike Protracker this is not an absolute offset but scales to the whole length of the sample, ie a value of 80 will start from the middle of the sample. If there's no argument, the sample offset will be set right at the end, useful for E8/01.	Use with a note
0A	xy	Volume slide *x* - amount to slide volume up *y* - amount to slide volume down	
0F	xx	Subdivide amount This is the same as the Protracker Fxx command, except it doesn't actually change the speed of the song, only the speed of the track's effects. If the subdivide amount is higher, effects will be updated more often, making them run faster. The default value is 6.	
10	xx	Probability *xx* - Probability for sample being played. 01=will almost certainly not be played, 80=50%, FF=almost certain	Use with a note
11	xx	Loop fit *xx* - Number of ticks the waveform's loop should take to complete. Changes the frequency of the waveform.	Use with a note

12	xx	Loop fit *xx* - Same as 11, but tracks song speed changes and adjust the frequency accordingly.	Use with a note
13	xy	Auto shuffle *x* - Ticks to shuffle. 2 shuffles every other step, 3 every third step and so on. y=Shuffle amount. 0=none, F=almost a full tick. change the subdivision amount to, say, 10 for greater precision.	
14	xx	Randomize volume *xx* - Maximum amount the volume will be randomized	
15	xx	Random delay *xx* - Maximum number of subdivision steps the note will be delayed	Use with a note
16	xx	Randomize pitch *xx* - Maximum number of notches the pitch will be randomized	
17	xx	Harmonic play *xx* - The base frequency will be multiplied by xx	
18	xy	Combined note delay and cut *x* - The subdivision step to trigger the note *y* - The subdivision step to release the note	Use with a note
19	xy	Sustain pedal *y* - Subdivision step to trigger command *x*=1 - Depress sustain pedal *x*=2 - Release sustain pedal	
20	xx	Filter cutoff	
21	xx	Slide filter cutoff up *xx* notches	
22	xx	Slide filter cutoff down *xx* notches	
23	0x	Set filter cutoff LFO type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
24	xy	Filter cutoff LFO *x* - speed *y* - depth If either is zero, use previous value	
25	xx	Fine slide filter cutoff up *xx* notches	
26	xx	Fine slide filter cutoff down *xx* notches	
28	xx	Filter resonance	

29	xx	Slide filter resonance up *xx* notches	
2A	xx	Slide filter resonance down *xx* notches	
2B	0x	Set filter resonance LFO type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
2C	xy	Filter resonance LFO *x* - speed *y* - depth If either is zero, use previous value	
2D	xx	Fine slide filter resonance up *xx* notches	
2E	xx	Fine slide filter resonance down *xx* notches	
DC	xx	Note cutoff, releases note	
E0	xx	Set filter type for track. *xx* - 1 - filter disabled *x* - 2 - 4p filter, lowpass *x* - 3 - 4p filter, highpass	
E1	xx	Fine slide up *xx* notches	
E2	xx	Fine slide down *xx* notches	
E4	0x	Set vibrato type *x* - 0 - sine, retrig waveform at samplestart *x* - 1 - saw, retrig waveform at samplestart *x* - 2 - square, retrig waveform at samplestart *x* - 4 - sine, don't retrig waveform at samplestart *x* - 5 - saw, don't retrig waveform at samplestart *x* - 6 - square, don't retrig waveform at samplestart	
E5	xx	Set finetune for track 00 = -1/2 halfnote, 80 = 0, FF = ~+1/2 halfnote	
E6	0x	Set panning type See E4/0x for parameter	
E7	0x	Set tremolo type See E4/0x for parameter	
E8	01	Reverse direction of sample being played	
E9	xx	Retrig sample *xx* - subdivision count at which sample is retriggered	Use with a note
EA	xx	Fine volume slide up	

EB	xx	Fine volume slide down	
EC	xx	Note cutoff, set volume to zero	
ED	xx	Note delay Delay samplestart for *xx* subdivision steps	Use with a note
EE	xx	Fine panning slide left	
EF	xx	Fine panning slide right	

```
// MersenneTwister.h
// Mersenne Twister random number generator -- a C++ class MTRand
// Based on code by Makoto Matsumoto, Takuji Nishimura, and Shawn Cokus
// Richard J. Wagner  v1.0  15 May 2003  rjwagner@writeme.com

// The Mersenne Twister is an algorithm for generating random numbers.  It
// was designed with consideration of the flaws in various other generators.
// The period, 2^19937-1, and the order of equidistribution, 623 dimensions,
// are far greater.  The generator is also fast; it avoids multiplication and
// division, and it benefits from caches and pipelines.  For more information
// see the inventors' web page at http://www.math.keio.ac.jp/~matumoto/emt.html

// Reference
// M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally
// Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on
// Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

// Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
// Copyright (C) 2000 - 2003, Richard J. Wagner
// All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//
//   1. Redistributions of source code must retain the above copyright
//      notice, this list of conditions and the following disclaimer.
//
//   2. Redistributions in binary form must reproduce the above copyright
//      notice, this list of conditions and the following disclaimer in the
//      documentation and/or other materials provided with the distribution.
//
//   3. The names of its contributors may not be used to endorse or promote
//      products derived from this software without specific prior written
//      permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
// EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
// PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
// LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
// NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
// SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

// The original code included the following notice:
//
//     When you use this, send an email to: matumoto@math.keio.ac.jp
//     with an appropriate reference to your work.
//
// It would be nice to CC: rjwagner@writeme.com and Cokus@math.washington.edu
// when you write.

#ifndef MERSENNETWISTER_H
#define MERSENNETWISTER_H

// Not thread safe (unless auto-initialization is avoided and each thread has
// its own MTRand object)

#include
#include
#include
#include
#include

class MTRand {
// Data
public:
        typedef unsigned long uint32;  // unsigned integer type, at least 32 bits

        enum { N = 624 };       // length of state vector
        enum { SAVE = N + 1 };  // length of array for save()

protected:
        enum { M = 397 };  // period parameter

        uint32 state[N];   // internal state
        uint32 *pNext;     // next value to get from state
        int left;          // number of values left before reload needed


//Methods
public:
        MTRand( const uint32& oneSeed );  // initialize with a simple uint32
        MTRand( uint32 *const bigSeed, uint32 const seedLength = N );  // or an array
        MTRand();  // auto-initialize with /dev/urandom or time() and clock()

        // Do NOT use for CRYPTOGRAPHY without securely hashing several returned
        // values together, otherwise the generator state can be learned after
        // reading 624 consecutive values.

        // Access to 32-bit random numbers
        double rand();                          // real number in [0,1]
        double rand( const double& n );         // real number in [0,n]
        double randExc();                       // real number in [0,1)
        double randExc( const double& n );      // real number in [0,n)
        double randDblExc();                    // real number in (0,1)
        double randDblExc( const double& n );   // real number in (0,n)
        uint32 randInt();                       // integer in [0,2^32-1]
        uint32 randInt( const uint32& n );      // integer in [0,n] for n < 2^32
        double operator()() { return rand(); }  // same as rand()
        // Access to 53-bit random numbers (capacity of IEEE double precision)
        double rand53();  // real number in [0,1)

        // Access to nonuniform random number distributions
        double randNorm( const double& mean = 0.0, const double& variance = 0.0 );

        // Re-seeding functions with same behavior as initializers
        void seed( const uint32 oneSeed );
        void seed( uint32 *const bigSeed, const uint32 seedLength = N );
        void seed();

        // Saving and loading generator state
        void save( uint32* saveArray ) const;  // to array of size SAVE
        void load( uint32 *const loadArray );  // from such array
        friend std::ostream& operator<<( std::ostream& os, const MTRand& mtrand );
        friend std::istream& operator>>( std::istream& is, MTRand& mtrand );

protected:
        void initialize( const uint32 oneSeed );
        void reload();
        uint32 hiBit( const uint32& u ) const { return u & 0x80000000UL; }
        uint32 loBit( const uint32& u ) const { return u & 0x00000001UL; }
        uint32 loBits( const uint32& u ) const { return u & 0x7fffffffUL; }
        uint32 mixBits( const uint32& u, const uint32& v ) const
                { return hiBit(u) | loBits(v); }
        uint32 twist( const uint32& m, const uint32& s0, const uint32& s1 ) const
                { return m ^ (mixBits(s0,s1)>>1) ^ (-loBit(s1) & 0x9908b0dfUL); }
        static uint32 hash( time_t t, clock_t c );
};


inline MTRand::MTRand( const uint32& oneSeed )
        { seed(oneSeed); }

inline MTRand::MTRand( uint32 *const bigSeed, const uint32 seedLength )
        { seed(bigSeed,seedLength); }

inline MTRand::MTRand()
        { seed(); }

inline double MTRand::rand()
        { return double(randInt()) * (1.0/4294967295.0); }

inline double MTRand::rand( const double& n )
        { return rand() * n; }

inline double MTRand::randExc()
        { return double(randInt()) * (1.0/4294967296.0); }

inline double MTRand::randExc( const double& n )
        { return randExc() * n; }
```

```
inline double MTRand::randDblExc()
	{ return ( double(randInt()) + 0.5 ) * (1.0/4294967296.0); }

inline double MTRand::randDblExc( const double& n )
	{ return randDblExc() * n; }

inline double MTRand::rand53()
{
	uint32 a = randInt() >> 5, b = randInt() >> 6;
	return ( a * 67108864.0 + b ) * (1.0/9007199254740992.0);  // by Isaku Wada
}

inline double MTRand::randNorm( const double& mean, const double& variance )
{
	// Return a real number from a normal (Gaussian) distribution with given
	// mean and variance by Box-Muller method
	double r = sqrt( -2.0 * log( 1.0-randDblExc()) ) * variance;
	double phi = 2.0 * 3.14159265358979323846264338328 * randExc();
	return mean + r * cos(phi);
}

inline MTRand::uint32 MTRand::randInt()
{
	// Pull a 32-bit integer from the generator state
	// Every other access function simply transforms the numbers extracted here

	if( left == 0 ) reload();
	--left;

	register uint32 s1;
	s1 = *pNext++;
	s1 ^= (s1 >> 11);
	s1 ^= (s1 <<  7) & 0x9d2c5680UL;
	s1 ^= (s1 << 15) & 0xefc60000UL;
	return ( s1 ^ (s1 >> 18) );
}

inline MTRand::uint32 MTRand::randInt( const uint32& n )
{
	// Find which bits are used in n
	// Optimized by Magnus Jonsson (magnus@smartelectronix.com)
	uint32 used = n;
	used |= used >> 1;
	used |= used >> 2;
	used |= used >> 4;
	used |= used >> 8;
	used |= used >> 16;

	// Draw numbers until one is found in [0,n]
	uint32 i;
	do
		i = randInt() & used;  // toss unused bits to shorten search
```

```
        while( i > n );
        return i;
}


inline void MTRand::seed( const uint32 oneSeed )
{
        // Seed the generator with a simple uint32
        initialize(oneSeed);
        reload();
}


inline void MTRand::seed( uint32 *const bigSeed, const uint32 seedLength )
{
        // Seed the generator with an array of uint32's
        // There are 2^19937-1 possible initial states.  This function allows
        // all of those to be accessed by providing at least 19937 bits (with a
        // default seed length of N = 624 uint32's).  Any bits above the lower 32
        // in each element are discarded.
        // Just call seed() if you want to get array from /dev/urandom
        initialize(19650218UL);
        register int i = 1;
        register uint32 j = 0;
        register int k = ( N > seedLength ? N : seedLength );
        for( ; k; --k )
        {
                state[i] =
                        state[i] ^ ( (state[i-1] ^ (state[i-1] >> 30)) * 1664525UL );
                state[i] += ( bigSeed[j] & 0xffffffffUL ) + j;
                state[i] &= 0xffffffffUL;
                ++i;  ++j;
                if( i >= N ) { state[0] = state[N-1];  i = 1; }
                if( j >= seedLength ) j = 0;
        }
        for( k = N - 1; k; --k )
        {
                state[i] =
                        state[i] ^ ( (state[i-1] ^ (state[i-1] >> 30)) *
1566083941UL );
                state[i] -= i;
                state[i] &= 0xffffffffUL;
                ++i;
                if( i >= N ) { state[0] = state[N-1];  i = 1; }
        }
        state[0] = 0x80000000UL;  // MSB is 1, assuring non-zero initial array
        reload();
}


inline void MTRand::seed()
{
        // Seed the generator with an array from /dev/urandom if available
        // Otherwise use a hash of time() and clock() values

        // First try getting an array from /dev/urandom
        FILE* urandom = fopen( "/dev/urandom", "rb" );
        if( urandom )
        {
                uint32 bigSeed[N];
                register uint32 *s = bigSeed;
                register int i = N;
                register bool success = true;
                while( success && i-- )
                        success = fread( s++, sizeof(uint32), 1, urandom ) != 0;
                fclose(urandom);
                if( success ) { seed( bigSeed, N );  return; }
        }

        // Was not successful, so use time() and clock() instead
        seed( hash( time(NULL), clock() ) );
}


inline void MTRand::initialize( const uint32 seed )
{
        // Initialize generator state with seed
        // See Knuth TAOCP Vol 2, 3rd Ed, p.106 for multiplier.
        // In previous versions, most significant bits (MSBs) of the seed affect
        // only MSBs of the state array.  Modified 9 Jan 2002 by Makoto Matsumoto.
        register uint32 *s = state;
        register uint32 *r = state;
        register int i = 1;
        *s++ = seed & 0xffffffffUL;
        for( ; i < N; ++i )
        {
                *s++ = ( 1812433253UL * ( *r ^ (*r >> 30) ) + i ) & 0xffffffffUL;
                r++;
        }
}


inline void MTRand::reload()
{
        // Generate N new values in state
        // Made clearer and faster by Matthew Bellew (matthew.bellew@home.com)
        register uint32 *p = state;
        register int i;
        for( i = N - M; i--; ++p )
                *p = twist( p[M], p[0], p[1] );
        for( i = M; --i; ++p )
                *p = twist( p[M-N], p[0], p[1] );
        *p = twist( p[M-N], p[0], state[0] );

        left = N, pNext = state;
```

```
}


inline MTRand::uint32 MTRand::hash( time_t t, clock_t c )
{
	// Get a uint32 from t and c
	// Better than uint32(x) in case x is floating point in [0,1]
	// Based on code by Lawrence Kirby (fred@genesis.demon.co.uk)

	static uint32 differ = 0;  // guarantee time-based seeds will change

	uint32 h1 = 0;
	unsigned char *p = (unsigned char *) &t;
	for( size_t i = 0; i < sizeof(t); ++i )
	{
		h1 *= UCHAR_MAX + 2U;
		h1 += p[i];
	}
	uint32 h2 = 0;
	p = (unsigned char *) &c;
	for( size_t j = 0; j < sizeof(c); ++j )
	{
		h2 *= UCHAR_MAX + 2U;
		h2 += p[j];
	}
	return ( h1 + differ++ ) ^ h2;
}


inline void MTRand::save( uint32* saveArray ) const
{
	register uint32 *sa = saveArray;
	register const uint32 *s = state;
	register int i = N;
	for( ; i--; *sa++ = *s++ ) {}
	*sa = left;
}


inline void MTRand::load( uint32 *const loadArray )
{
	register uint32 *s = state;
	register uint32 *la = loadArray;
	register int i = N;
	for( ; i--; *s++ = *la++ ) {}
	left = *la;
	pNext = &state[N-left];
}


inline std::ostream& operator<<( std::ostream& os, const MTRand& mtrand )
{
	register const MTRand::uint32 *s = mtrand.state;
	register int i = mtrand.N;
	for( ; i--; os << *s++ << "\t" ) {}
	return os << mtrand.left;
}


inline std::istream& operator>>( std::istream& is, MTRand& mtrand )
{
	register MTRand::uint32 *s = mtrand.state;
	register int i = mtrand.N;
	for( ; i--; is >> *s++ ) {}
	is >> mtrand.left;
	mtrand.pNext = &mtrand.state[mtrand.N-mtrand.left];
	return is;
}

#endif  // MERSENNETWISTER_H

// Change log:
//
// v0.1 - First release on 15 May 2000
//      - Based on code by Makoto Matsumoto, Takuji Nishimura, and Shawn Cokus
//      - Translated from C to C++
//      - Made completely ANSI compliant
//      - Designed convenient interface for initialization, seeding, and
//        obtaining numbers in default or user-defined ranges
//      - Added automatic seeding from /dev/urandom or time() and clock()
//      - Provided functions for saving and loading generator state
//
// v0.2 - Fixed bug which reloaded generator one step too late
//
// v0.3 - Switched to clearer, faster reload() code from Matthew Bellew
//
// v0.4 - Removed trailing newline in saved generator format to be consistent
//        with output format of built-in types
//
// v0.5 - Improved portability by replacing static const int's with enum's and
//        clarifying return values in seed(); suggested by Eric Heimburg
//      - Removed MAXINT constant; use 0xffffffffUL instead
//
// v0.6 - Eliminated seed overflow when uint32 is larger than 32 bits
//      - Changed integer [0,n] generator to give better uniformity
//
// v0.7 - Fixed operator precedence ambiguity in reload()
//      - Added access for real numbers in (0,1) and (0,n)
//
// v0.8 - Included time.h header to properly support time_t and clock_t
//
// v1.0 - Revised seeding to match 26 Jan 2002 update of Nishimura and Matsumoto
//      - Allowed for seeding with arrays of any length
//      - Added access for real numbers in [0,1) with 53-bit resolution
//      - Added access for real numbers from normal (Gaussian) distributions
//       - Increased overall speed by optimizing twist()
//       - Doubled speed of integer [0,n] generation
//       - Fixed out-of-range number generation on 64-bit machines
//       - Improved portability by substituting literal constants for long enum's
//       - Changed license from GNU LGPL to BSD
```

2ndP MGroove v1.0   Oct 26 2001

2ndProcess' Groove Aid Machine for Buzz (-> www.jeskola.com)

WELCOME TO THE NEW GROOVE DIMENSION....

Hi there, welcome to my second machine! It's actually become better than
I expected... (of course there's still room for improvement...)!

With this machine, the time till someone makes a new funky Buzz-Plugin-host
may pass a little less painful :) ...cause now we have a shuffle-_slider_!

You can get 1000's of variations of one rhyhtm just by tweaking
MGroove's sliders, leaving the other patterns and sliders of your
song alone.

You can EASILY add shuffle/swing to a song without worrying about your song
tempo changing... you can even write a pattern for MGroove with the _shuffle_
changing!

As you see, there is no need for external groove calculators anymore...
You won't need echot timers as well as the Info dialog shows the length
of 1 tick in milliseconds!

No need to edit Master patterns by hand to achieve shuffle fx...
but you still can record the shuffle values produced by MGroove
to a Master pattern!

Upwards compability SHOULD be given as long as "Master" is always
called "Master"... which is not likely to ever change I think.

installation
 - - - - - -
Put MGroove.dll into the Gear\Generators subfolder of your Buzz folder.
Update index.txt with this line:

```
        MGroove,2ndP MGroove
```

parameter description
- - - - - - - - - - -

BPM   The BPM value for your song.

TPB   - TPB -   -   -   -

BPM Inertia 0.0001 to 1. 1 means no inertia, ie changes to the
BPM slider take effect immediately; 0.0001 means it
will take LONG until the new value is reached.
The inertia algorithm is DAMN BAD, I admit... :(but it works :)!

Shuffle   0 to 16 ticks. how many ticks to wait between shuffle steps.
Length   0 means shuffle is off; high values might not be practical...
try 1 or 2.

Shuffle   0 to 75% - try 25%, 12.5%,....

Sync   In most songs you will have just 1 "1" trigger at the beginning
of the song, which syncs the shuffle count & phase to your
song. "0" resets only the phase; Be aware that this might
change the pattern length (in time), which might be what
you want...

NOTES
- - -
You can use MGroove to record patterns for the Master instead of using
it directly. You can then remove it and use the patterns for tempo and
groove control.

If you don't want to shuffle the whole song, either change the shuffling
throughout the song :) or use Hoester's GrooveBox... which is better if
you want to shuffle just some tracks playing together with non-shuffled.
Interesting effects can be achieved by using both together! 8p

Buzz machines behave differently when tempo is shuffled. E.g. for Ninja
Delay, when you set a tick-wise delay while shuffle is = 0, you can
then move the shuffle slider without the echo time changing. MTrks
has two modes for looping samples, one is synchronized and the other one
not.

The inertia feature is nice if you want to slow down/speed up a section
that is repeated via 2ndPLoopJumpHACK.

Have fun and a look at the demo bmx.

CONTACT/LEGAL INFORMATION
- - - - - - - - - - - - -

I can be reached via e-mail: malteschreiber@hotmail.com

```
   ______                 _____  __
  /      \  ____________ ____\   \/ /
 /  \ /  \/  _ \_  _ \ __ \    /
/    Y    \ <_> ) | \/ ___/    \
\____|__  /____/|__|  \_______/\  \
        \/                   \_/

      \    / _ ._ _| | _
       \/\/ (_)| (_| || |
```

Morex Word In 1.0 Readme

Description

This is a stereo Wave In hard disk sample reader. It reads 32-bit floating point
Wav files at a variety of sample rates up to 96 kHz.

Uses

1) As an audio input from some other software.

2) As an audio input from buzz. You may, for example, need to output from Buzz to Cubase or
Cool Edit to do some post processing, then take your mix back into Buzz for final
mastering (with the Jupiter2, for example). With this machine, you can do this without
any loss of audio quality.

2) As a 32-bit audio sequencer/mixer. You can have multiple instances of WordIn playing
multiple files simultaneously, and you can mix them however you like with Buzz.

3) To save processor power!!! For instance, if you are writing a piece with multiple
machines, and your audio is stuttering, you may be saturating your processor (you can check
with the CPU Monitor in the Buzz View menu). To remove the clicks, select some machines that
don't change frequently (such as your drum generators), and record your track with only these
machines playing. Then, mute these machines and play them back through WordIn to release
the processor power, et voilla, you have more CPU cycles to play with for the rest of your
sounds.

Installation

Unzip to \buzz\gear, restart Buzz if it's running.

Usage

1) Make sure the effect is connected to the Master.

2) Select the wave file to input by right clicking on the machine and
   selecting the menu option.

3) Set the Play parameter to On or Off to start and stop playing.

4) The Ticks parameter sets the number of ticks into the sample at which it will start playing.

5) The Offset parameter is a number of samples which is added to the Ticks parameter when the
sample starts playing, allowing fine control of sample positioning.

6) This zip file contains an audio track template (in the +Mixers folder), containing correct values
for the Tick parameter for approx. 320 bars of song. If you use this template, you can move the Start
and End sequence terminators without worrying about where your sample will play from (Highly Recommended!).

Notes

If you're recording from buzz for playback through WordIn, then for best results, use CyanPhase's Overloader Extended Hard Disk Recorder. Select the CyanPhase Wav Recorder plug-in, and from the Config button, select 32-bit IEEE Float (type 3). Please note that on my system, the hard disk recorder adds 4571 blank samples to anything you record, so you may need to compensate for this with the Offset parameter (set it to 11DB).

Distribution

This code is free; please include this readme and the template in any distribution.

Version 0.2

NEW! Stereo mp3s now supported

NEW! Ability to load into any wavetable slot in buzz!

Version 0.1

mp3loader.... a machine that decompresses mp3s (VBR ones as well!!) into a spare wavetable slot in buzz

1) Put dll in Buzz\gear\generators

2) Put the mp3loader into the machine editor in Buzz

3) Right click and select "Load an mp3.."

4) Select your mp3...
tip: if you got winamp installed then right click any mp3 and select "play in winamp" to preview.

5) Edit the title to go in the wavetable (if you hate the default one)

6) Hit OK

7) If crash then report (fukinace@yahoo.co.uk).. include mp3 in email (if its < 500K)

use this mp3loader at your own risk...its very alpha...
I am not to blame if it does anything bad to your machine or any of the files stored on it!!!!

MarC
fukinace@yahoo.co.uk

http://marcnet.virtualave.net/

(by the way, this machine aint on there yet!! Im still developing MarCNet 2 ..)

Ninereeds Broadcast

By Steve Horne, 21 July 1999

This generator does not generate an output signal at all. Instead, it provides a convenient way to transmit commands over the 'broadcast interface'.

The broadcast interface is a way of communicating between Buzz machines. The reasons for this are...

1. To provide a centralised form of control, so that a command defined in one place can affect the output of a number of machines simultaneously.
2. To provide a way of communicating between machines in Buzz and some external application.
3. To provide a way of sending additional information to a machine that has outgrown its parameters, without losing backward compatability.

Unfortunately, commands cannot be sent to just any machine. The machine must be written to support this capability, and must be configured to listen to the right channel.

Communication with external applications is not yet supported. Communication with a parent application that is using BuzzPlay.DLL may be possible, but synchronization is not yet supported.

History

???
: First Release

Global Parameters

The global parameters are...

Channel
: This parameter specifies which channels the command is to be sent over. This is constructed by adding the following values in hex...
    - 01 : Include channel 0
    - 02 : Include channel 1

- 04 : Include channel 2
- 08 : Include channel 3
- 10 : Include channel 4
- 20 : Include channel 5
- 40 : Include channel 6
- 80 : Include channel 7

A value of 0F will therefore send on channels 0, 1, 2 and 3 simultaneously.

The idea is that machines should be grouped, with each group listening to a particular channel. Commands can the be sent to one particular group, or to a number of groups, at the same time.

The default is to send on channel 0 only.

Time

This indicates when the command should be handled by the receiving machines...

- 0000 : Now
- 0080 : In half a tick
- 0100 : In one tick
- 1000 : In sixteen ticks

Command

This indicates what command to send, in the range 01 to FF. The following convention for command numbers should be followed by all machines...

01 .. 3F

Commands in this range have standard meanings implemented by the machine.

40 .. 7F

Commands in this range are reserved for machine-specific user-configurable macros.

80 - BF

Commands in this range can be used for machine-specific non-standardised tasks.

C0 - FF

Commands in this range have standard meanings, which are partially or fully implemented by the broadcast library itself.

Byte Params 1..4

These parameters specify byte data, the meaning of which depends on the command. The values are in the range 00 to FE.

Word Params 1..4

These parameters specify word data, the meaning of which depends on the command. The values are in the range 0000 to FFFE.

The currently defined commands are...

Select Program

Selects a particular instrument program from a bank of memorised settings.

```
Command      : 01
Byte Param 1 : Program ID
```

All Notes Off

Generates a note off command in all channels simultaneously, overriding the pattern contents.

```
Command      : C0
```

Ignore All Notes

Ignores all note start commands in all channels until command C2 is received. Note off commands are still accepted.

```
Command      : C1
```

Cancel Ignore All Notes

Cancel Ignore All Notes

```
Command      : C2
```

Transpose

Modifies all note start commands to transpose the notes up or down by a set offset, giving an easy way to handle key changes without needing additional patterns to be defined.

Any note that goes out of range is converted to a note off.

```
Command      : C3
Byte Param 1 : High digit = number of octaves to raise
             : Low digit  = number of semitones to raise
Byte Param 2 : High digit = number of octaves to lower
             : Low digit  = number of semitones to lower
```

Set or Clear Controls

Sets the values of up to four controls. Control IDs range from 0 to 7, and are intended to control things like filter cutoffs. They will usually map simply to normal global or track parameters.

If a control ID is set but no value specified, that control is cleared - any current control slide is disabled, but no new value is given. This can be used to give control back to the pattern immediately.

```
Command      : C4
Byte Param 1 : Control ID 1, range 0..7
Byte Param 2 : Control ID 2, range 0..7
Byte Param 3 : Control ID 3, range 0..7
Byte Param 4 : Control ID 4, range 0..7
Word Param 1 : Control Value 1, range 0000..FFFE
Word Param 2 : Control Value 2, range 0000..FFFE
Word Param 3 : Control Value 3, range 0000..FFFE
Word Param 4 : Control Value 4, range 0000..FFFE
```

Slide Controls

Sets value slides for either one or two controls. These are the same controls as handled by command C5, but this command allows start and end values and durations to be set - the control is automatically stepped from the start to the end values every tick.

```
Command      : C5
Byte Param 1 : Control ID 1, range 0..7
Byte Param 2 : Time 1, in ticks
Byte Param 3 : Control ID 2, range 0..7
Byte Param 4 : Time 2, in ticks
Word Param 1 : Start Value 1, range 0000..FFFE
Word Param 2 : End   Value 1, range 0000..FFFE
Word Param 3 : Start Value 2, range 0000..FFFE
Word Param 4 : End   Value 3, range 0000..FFFE
```

The following machines are known to support additional commands, in the range 80 to BF...

- Generators
  - Ninereeds NRS04
- Effects
  - None at present

Track Parameters

This machine does not have any track parameters.

Multiple commands can be effectively sent simultaneously by...

- Using more than one Broadcast machine.
- Defining one of the commands early, and specifying a delay.

Attributes

This machine does not have any attributes.

Edit Dialog

This machine does not have an edit dialog.

Ninereeds LFO

By Steve Horne, 21 May 2000

This generator outputs a signal made up of a DC component and a number of low frequency signals.

The Ninereeds LFO is intended to control other machines, perhaps using AuxBus. Good examples would include vibrato or tremelo effects using a Ninereeds NRS04 or NRS05, or simply using a Jeskola Multipier as a kind of enveloper.

I hope to write some filters soon which can use an external LFO via AuxBus to control the cutoff and/or resonance.

History

21 May 2000
: First release? - some work to do yet, I think.

Global Parameters

The global parameters are...

DC Fade Speed
: Sets the rate of change in the DC offset, as a half-life in ms.

DC Level Initialise
: Immediately sets a specific value for the DC offset.

DC Level Target
: Sets a new target value for the DC offset - the time taken to reach this value depends on the fade speed.

Track Parameters

The track parameters are...

Waveform
: Sets the waveform of the LFO. Normally this will be left at 01 for a sine wave.

Frq Fade Speed
: Sets the rate of change of the frequency of the LFO, as a half-life in ms.

Frq Initialise
: Immediately sets a specific frequency for the LFO.

Frq Target
: Sets a new target frequency for the LFO.

AC Fade Speed
: Sets the rate of change in the amplitude of the LFO.

AC Level Initialise
: Immediately sets a specific value for the amplitude of the LFO.

AC Level Target
: Sets a new target value for the amplitude of the LFO.

Attributes

This machine does not have any attributes.

Edit Dialog

To access the edit dialog, go to the Machine Editor screen in Buzz, right click on the appropriate machine and select 'Edit Ninereeds LFO' on the context menu.

Broadcast Commands Accepted

The following fixed broadcast commands are understood by this machine...

???

If you have any comments or suggestions, please e-mail them to steve@lurking.demon.co.uk.

Ninereeds DSPLIB.dll
nblib.dll

These files need to be copied into the root Buzz Directory. All Ninereeds machines require these two files. Without these files, all Ninereeds machines will NOT work.

http://www.lurking.demon.co.uk/
http://www.buzzmachines.com/

Ninereeds NRS04

By Steve Horne, 10 July 1999

This generator is a synthesizer, with the following features...

- ADSR envelope, with linear attack and exponential decay and release.
- Pitch bend.
- Thirteen basic waveforms.
- Fractal distortion of the basic waveform that varies with the envelope.
- Ability to produce a 'twin' note, detuned from the original.
- Support for ring modulation, amplitude modulation, frequency modulation and pulse width (or waveform centre) modulation based on an auxiliary input signal.
- Ability to store and choose from a number of 'programs' - essentially named instrument configurations.
- Support for partial external control - listens to 'command broadcast channels'.

A feature that some people may find novel is that this machine actually uses note-off commands. I may add some more precise timing controls to a future machine, but I find using note-offs much more convenient than providing calculated durations for every note.

However, if the sustain level is set to zero, and the release rate is set the same as the decay rate, there is no need to worry about note-off commands.

History

September 3, 1999
: I've left this so long, I'm ashamed of myself. However, this version has...
    - A bug fix that initialises note volumes correctly.
    - A bug fix that allows the output to be sent to more than one effect - I really don't understand the mi::Work functions mode parameter, but ignoring it seems to have the required effect.
    - A bug fix that means square waveforms are now square - sorry if this breaks old tunes.
    - Optional optimisation of fractals and envelopes using precalculated lookup tables - warning - these do affect the end result, especially the fractal optimisation.
    - An internal modulator for FM, AM and PWM effects - allows modulator frequency to be relative to the note frequency - no modulator envelope yet, nor any controls to 'equalise

the brightness over a range of notes'.

July 18, 1999

Added some major facilities, including...

- Ability to recieve from an AuxBus channel.
- Ability to perform various modulations using the AuxBus channel.
- Support for external control - listens for broadcast commands.
- A program bank storing named instrument configuartions.

This is by no means complete...

- The modulations are only really good for special effects - some secondary oscilators and/or LFOs are needed.
- Program changes and other externally driven changes to the sound are very abrupt, and tend to create clicks. Some kind of smooth transition is needed.
- The ability to save and load program banks as readable text files, and possibly directly exchange them between machines, could be useful.

July 13, 1999

Fixed some very silly defaults, added an amplitude control for the twin note, and added an extra waveform (based on double triangle, but adjusted to give it some even harmonics).

Nothing major, no compatability issues, but fewer irritations.

July 10, 1999

First Release

Global Parameters

This machine does not have any global parameters.

Track Parameters

The track parameters are...

Note

Triggers new notes. A note off here stops the note, by triggering the release phase.

Bend Note

Starts a pitch bend, specifying the target note. A note off here stops a pitch bend, so that the note continues at whatever pitch it reached.

Volume

How loud to play the note.

Attributes

This machine does not have any attributes.

Edit Dialog

To access the edit dialog, go to the Machine Editor screen in Buzz, right click on the appropriate machine and select 'Edit Ninereeds NRS04' on the context menu.

A property sheet will be displayed, which has three pages...

Waveform

- Allows the basic waveform to be selected.
- Allows the fractal distortion parameters to be edited.
- Allows the 'twin' note to be enabled, and its detuning and amplitude to be edited.

> Of the commonly known simple waveforms, the only one that has even harmonics is the ramp (sometimes called sawtooth). While waveforms without even harmonics are useful, they often sound beepy or obviously synthesized, no matter how much they are processed - both string and brass acoustic instruments definitely produce even harmonics, and I'm sure this is pretty normal for real instruments.
>
> If you find the ramp waveform too harsh, try 'Altered Bumps 2', 'Double Bumps 2' or 'Double Triangle 2' instead - and don't forget to play with the fractal settings.
>
> Finally, if the Twin note is enabled using the checkbox it is always calculated, even if its amplitude is zero. The only effect in this case is to halve the amplitude of the main note. Using a twin note almost doubles the CPU usage, so turn it off if you can't hear the difference.

Envelope

- Allows the ADSR envelope to be edited.
- Allows the pitch bend rate to be edited.

AuxBus Effects

- Connect to an AuxBus channel.
- Disconnect from AuxBus.
- Configure ring modulation.

- Configure frequency modulation.
- Configure amplitude modulation.
- Configure pulse width / waveform centre modulation.

Waveform centre modulation basically distorts a waveform such that the normal centre point is reached slightly earlier or slightly later than normal. In the case where the original waveform is a square this gives pulse width modulation. The advantage is simply that it can be applied to any waveform.

Broadcast Rx

- Allows broadcast command listening to be enabled and a channel to be selected.
- Allows programs to be configured in the program bank.
- Allows a mapping between program IDs and named programs to be established.

About

My web and email address, and other pointless stuff.

The parameters on the dialog are not meant to be edited during a live performance. There should be no possibility of a crash, so you can adjust the parameters while a tune is playing to get the sound you want, but there is no attempt to avoid clicks or other strange effects.

Broadcast Commands Accepted

The following fixed broadcast commands are understood by this machine...

Select Program

```
Command      : 01
Byte Param 1 : Program ID
```

Set Waveform

```
Command      : 80
Byte Param 1 : Waveform ID
```

Set Fractal

```
Command      : 81
Byte Param 1 : Fractal Depth
```

```
     Word Param 1 : Fractal Effect Low
     Word Param 2 : Fractal Effect High
```

Set Envelope

```
     Command      : 82
     Word Param 1 : Attack
     Word Param 2 : Decay
     Word Param 3 : Sustain
     Word Param 4 : Release
```

Set Pitch Bend Rate

```
     Command      : 83
     Word Param 1 : Pitch Bend Rate
```

Set Twin Note

```
     Command      : 84
     Byte Param 1 : Enable Twin Note
     Byte Param 2 : Apply Fractal Before Mixing
     Word Param 1 : Detune - not yet implemented
     Word Param 2 : Twin Note Amplitude
```

Set Ring Modulation

```
     Command      : 85
     Byte Param 1 : Enable on Main Note
     Byte Param 2 : Enable on Twin Note
```

Set Frequency Modulation

```
     Command      : 86
     Byte Param 1 : Enable Frequency Modulation
     Word Param 1 : Modulation Level
```

Set Amplitude Modulation

```
     Command      : 87
     Byte Param 1 : Enable Amplitude Modulation
     Word Param 1 : Modulation Level
```

Set Pulse Width (Waveform Centre) Modulation

```
Command      : 88
Byte Param 1 : Enable Pulse Width Modulation
Word Param 1 : Modulation Level
```

How does that fractal thing work?

Most controls for this machine are fairly self-explanatory, or at least fairly standard. The fractal distortion, however, is my own invention and some explanation is in order.

The fractal distortion uses an iterative function. That is, once you get the result of the function, you feed it back in again. It is heavily based on an equation for modelling population fluctuations that was first documented in “Simple Mathematical Models with Very Complicated Dynamics” (Nature, No. 26, 1976), by R. M. May.

A sound signal has very little to do with population fluctuations, so I designed a cubic function to use instead. As with the May equation, there is a parameter that controls how 'chaotic' it is. I have called this the fractal effect. In addition, the fractal depth is the number of times the function is applied (the equivalent of the number of generations ahead to predict in the May equation).

When the fractal depth is set to zero, or the fractal effect is set to one, the function has no effect on the input signal. If the fractal depth is zero, the fractal is never called.

When the fractal effect is set to a high sufficiently high value, the function becomes chaotic. With a sufficiently high fractal depth, it is quite simple to produce white noise. With more moderate values for the effect and depth parameters, the fractal distortion will add moderate levels of high harmonics.

The reason there are two 'fractal effect' controls is simply that the fractal effect is varied as the note is played. The effect value actually used will be between the 'high' effect and the 'low' effect, depending on the current envelope level.

Finally, there is a checkbox in the 'Twin Note' group which allows the fractal to be applied either before or after the twin note is mixed in with the original note. If the fractal is applied before the mixing, then it needs to be calculated separately for both the original and the twin note. Unfortunately, this is normally necessary - applying the fractal after mixing gives a rather odd effect. I left it in as an option in case someone really needs an odd effect.

What's all this broadcast command listening stuff

I have created a small DLL which provides an additional means of communication between machines. It uses similar principles to AuxBus, but instead of communication signals it allow commands to be sent between machines.

The command format is based on the kinds of things that can be done in the pattern editor, and the main way of sending these commands will be using a machine that stores these commands in its pattern editor - Ninereeds Broadcast.

The inspiration was a recent message to the buzz developers mailing list, which suggested the possibility of a tune reacting to externally generated events - specifically, to locations and situations in a game.

The end result supports a number of requirements...

- Central control of all machines, a bit like a conductor. For example, all machines that support this could be switched to an 'emphasis' configuration in response to a single command by one machine.
- Pattern-driven control of a machine in ways that weren't anticipated when it was first created - ie extension beyond the limits of existing parameters.
- External control as described above.
- External listeners that detect events within the song - most likely reaching certain points in a song.

If you have any comments or suggestions, please e-mail them to steve@lurking.demon.co.uk.

Ninereeds NRS05

By Steve Horne, 30 November 1999

This generator is a synthesizer, with the following capabilities...

- Simple synthesis using an ADSR envelope and a choice of 14 waveforms.
- Synthesis with fractal distortion.
- A variety of modulations (frequency, amplitude, pulse width etc) using either an internal modulator or an AuxBus input.
- Physical modelling style synthesis.
- I addition to changing it's own controls, the NRS05 can both send and recieve commands over the broadcast library.

History

November 30, 1999

First release (apart from the beta).

Not complete by any means. Some planned extensions are...

- User interface changes.
- More modulators.
- Extensions to the physical modelling section.

Global Parameters

This machine does not have any global parameters.

Track Parameters

The track parameters are...

Note

  Triggers new notes. A note off here stops the note, by triggering the release phase.

Bend Note

  Starts a pitch bend, specifying the target note. A note off here stops a pitch bend, so that the note continues at whatever pitch it reached.

Volume

  How loud to play the note.

Attributes

This machine does not have any attributes.

Edit Dialog

To access the edit dialog, go to the Machine Editor screen in Buzz, right click on the appropriate machine and select 'Edit Ninereeds NRS04' on the context menu.

A property sheet will be displayed, which has three pages...

Waveform

- Allows the basic waveform to be selected.
- Allows the fractal distortion parameters to be edited.
- Allows the 'twin' note to be enabled, and its detuning and amplitude to be edited.

Of the commonly known simple waveforms, the only one that has even harmonics is the ramp (sometimes called sawtooth). While waveforms without even harmonics are useful, they often sound beepy or obviously synthesized, no matter how much they are processed - both string and brass acoustic instruments definitely produce even harmonics, and I'm sure this is pretty normal for real instruments.

If you find the ramp waveform too harsh, try 'Altered Bumps 2', 'Double Bumps 2' or 'Double Triangle 2' instead - and don't forget to play with the fractal settings.

Finally, if the Twin note is enabled using the checkbox it is always calculated, even if its amplitude is zero. The only effect in this case is to halve the amplitude of the main note. Using a twin note almost doubles the CPU usage, so turn it off if you can't hear the difference.

Envelope

- Allows the ADSR envelope to be edited.
- Allows the pitch bend rate to be edited.

AuxBus Effects

- Connect to an AuxBus channel.
- Disconnect from AuxBus.
- Configure ring modulation.
- Configure frequency modulation.
- Configure amplitude modulation.
- Configure pulse width / waveform centre modulation.

Waveform centre modulation basically distorts a waveform such that the normal centre point is reached slightly earlier or slightly later than normal. In the case where the original waveform is a square this gives pulse width modulation. The advantage is simply that it can be applied to any waveform.

Broadcast Rx

- Allows broadcast command listening to be enabled and a channel to be selected.
- Allows programs to be configured in the program bank.
- Allows a mapping between program IDs and named programs to be established.

About

My web and email address, and other pointless stuff.

The parameters on the dialog are not meant to be edited during a live performance. There should be no possibility of a crash, so you can adjust the parameters while a tune is playing to get the sound you want, but there is no attempt to avoid clicks or other strange effects.

Broadcast Commands Accepted

The following fixed broadcast commands are understood by this machine...

Select Program

```
Command      : 01
Byte Param 1 : Program ID
```

Set Waveform

```
     Command       : 80
     Byte Param 1 : Waveform ID
```

Set Fractal

```
     Command       : 81
     Byte Param 1 : Fractal Depth
     Word Param 1 : Fractal Effect Low
     Word Param 2 : Fractal Effect High
```

Set Envelope

```
     Command       : 82
     Word Param 1 : Attack
     Word Param 2 : Decay
     Word Param 3 : Sustain
     Word Param 4 : Release
```

Set Pitch Bend Rate

```
     Command       : 83
     Word Param 1 : Pitch Bend Rate
```

Set Twin Note

```
     Command       : 84
     Byte Param 1 : Enable Twin Note
     Byte Param 2 : Apply Fractal Before Mixing
     Word Param 1 : Detune - not yet implemented
     Word Param 2 : Twin Note Amplitude
```

Set Ring Modulation

```
     Command       : 85
     Byte Param 1 : Enable on Main Note
     Byte Param 2 : Enable on Twin Note
```

Set Frequency Modulation

```
     Command       : 86
     Byte Param 1 : Enable Frequency Modulation
     Word Param 1 : Modulation Level
```

Set Amplitude Modulation

```
     Command      : 87
     Byte Param 1 : Enable Amplitude Modulation
     Word Param 1 : Modulation Level
```

Set Pulse Width (Waveform Centre) Modulation

```
     Command      : 88
     Byte Param 1 : Enable Pulse Width Modulation
     Word Param 1 : Modulation Level
```

How does that fractal thing work?

Most controls for this machine are fairly self-explanatory, or at least fairly standard. The fractal distortion, however, is my own invention and some explanation is in order.

The fractal distortion uses an iterative function. That is, once you get the result of the function, you feed it back in again. It is heavily based on an equation for modelling population fluctuations that was first documented in “Simple Mathematical Models with Very Complicated Dynamics” (Nature, No. 26, 1976), by R. M. May.

A sound signal has very little to do with population fluctuations, so I designed a cubic function to use instead. As with the May equation, there is a parameter that controls how 'chaotic' it is. I have called this the fractal effect. In addition, the fractal depth is the number of times the function is applied (the equivalent of the number of generations ahead to predict in the May equation).

When the fractal depth is set to zero, or the fractal effect is set to one, the function has no effect on the input signal. If the fractal depth is zero, the fractal is never called.

When the fractal effect is set to a high sufficiently high value, the function becomes chaotic. With a sufficiently high fractal depth, it is quite simple to produce white noise. With more moderate values for the effect and depth parameters, the fractal distortion will add moderate levels of high harmonics.

The reason there are two 'fractal effect' controls is simply that the fractal effect is varied as the note is played. The effect value actually used will be between the 'high' effect and the 'low' effect, depending on the current envelope level.

Finally, there is a checkbox in the 'Twin Note' group which allows the fractal to be applied either before or after the twin note is mixed in with the original note. If the fractal is applied before the mixing, then it needs to be calculated separately for both the original and the twin note. Unfortunately, this is normally necessary - applying the fractal after mixing gives a rather odd effect. I left it in as an option in case someone really needs an odd effect.

What's all this broadcast command listening stuff

I have created a small DLL which provides an additional means of communication between machines. It uses similar principles to AuxBus, but instead of communication signals it allow commands to be sent between machines.

The command format is based on the kinds of things that can be done in the pattern editor, and the main way of sending these commands will be using a machine that stores these commands in its pattern editor - Ninereeds Broadcast.

The inspiration was a recent message to the buzz developers mailing list, which suggested the possibility of a tune reacting to externally generated events - specifically, to locations and situations in a game.

The end result supports a number of requirements...

- Central control of all machines, a bit like a conductor. For example, all machines that support this could be switched to an 'emphasis' configuration in response to a single command by one machine.
- Pattern-driven control of a machine in ways that weren't anticipated when it was first created - ie extension beyond the limits of existing parameters.
- External control as described above.
- External listeners that detect events within the song - most likely reaching certain points in a song.

If you have any comments or suggestions, please e-mail them to steve@lurking.demon.co.uk.

Place the Overloader dll in your Gear folder, not effects or generators
And place the ini file into your main buzz folder (not Gear)

Thx

Version Beta 0.0001

CyanPhase

Credits\Shoutz

Mute, discharge, Hym and djlaser for the major ideas and testing

especially Mute since he dragged me into this in the
first place hehe.

P. DooM's HACK Jump

What it is

A BUZZ **Generator**. It doesn't generate anything, but it lets you jump around in a BUZZ song. You can set the song position pointer to any value you like!

How to use

- Tick: Destination tick. Remember: This is a HEX value!
- Trigger: 1, to enable a jump

Notes

The jump will occur just **after** the tick! So if you set a jump at tick 15, it will still play all notes of tick 15 and jump afterwards.

HACK

HACK means: This machine accesses BUZZ internal variables directly. And this means for you: this machine can't be used with any other BUZZ machine host than BUZZ itself, and it probably can't be used with older versions of BUZZ. If there will be a new version of BUZZ one day, it'll perhaps **not be compatible anymore!**

Have fun!
P. DooM, April 2001
kaufmann@pop.agri.ch
My Homepage

P. DooM's HACK Msync 1.1

What is HACK Msync?

A BUZZ **Generator**. In fact, it doesn't generate anything. It's a program that can **sync BUZZ to a midi clock source**. It's implemented as a BUZZ machine because a) this makes it easy to use and b) it can get better control over BUZZ this way.
The main use of this machine is: you can sync BUZZ to a sequencer, such as cakewalk pro audio or steinberg Cubase VST (see next topics). But **you can also use it to sync two instances of BUZZ!** Just use a sequencer and send the same MIDI clock signal to both of them - they will play the same part, at the same time and in sync. This works **far better and is more accurate than the old BUZZync program**.

The main features of the machine are:

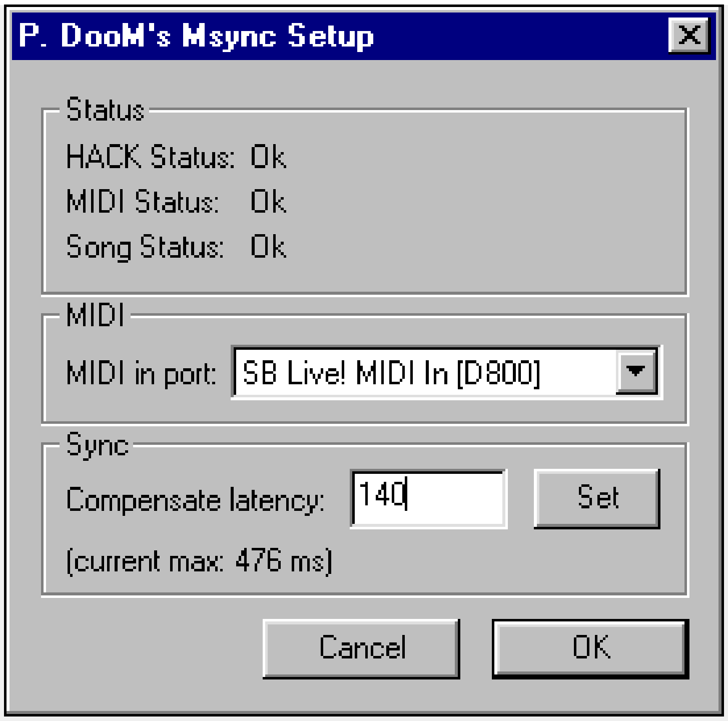
- sync to midi clock
- receive start/stop/continue midi signals
- receive midi SPP (song position pointer) messages
- automatic BPM adjustments
- **latency compensation**
- timing accuracy of about +/-5 ms (estimated)

Changes to version 1.0:

- now allows loops and jumps in your BUZZ track
- disabled an unnecessary HACK-error message
- corrected latency compensation. Note: **You have to use lower latency compensation settings in this version!**

How to set up Msync machine

After you've 'installed' the machine (by copying all files to your *buzz/gear/generators/*-directory), just insert the machine in your song. There's no need to connect it to the master, as it doesn't generate output. All parameters are accessible through the **Setup menu** (right click on the machine).


P. DooM's Msync Setup

Status

HACK Status: Ok

MIDI Status: Ok

Song Status: Ok

MIDI

MIDI in port: SB Live! MIDI In [D800]

Sync

Compensate latency: 140

(current max: 476 ms)

Explanation:

Status

Some information about the operation of Msync. If something doesn't work the way it should, check this out first!

- **HACK Status:** Shows if all BUZZ internal variables can be accessed. If this doesn't say 'Ok', you better update to the latest BUZZ version! Or drop me a mail if you can't get the error-message away.
- **MIDI Status:** Says 'Ok' if the MIDI port was opened successfully. If it doesn't, make sure you selected the right MIDI port (see below), and make sure that the port isn't in use by some other application (or used by BUZZ itself!).
- **Song Status:** Says 'Ok' if Msync works with the current song. If it doesn't, you very probably got a bad TPB setting. **TPB must be: 4, 6, 8, 12 or 24!**

MIDI

- **MIDI in port:** Select the in port here. You better use just one MIDI port especially for syncing (with just the clock signal and no other MIDI data like notes/controllers on it).

Sync

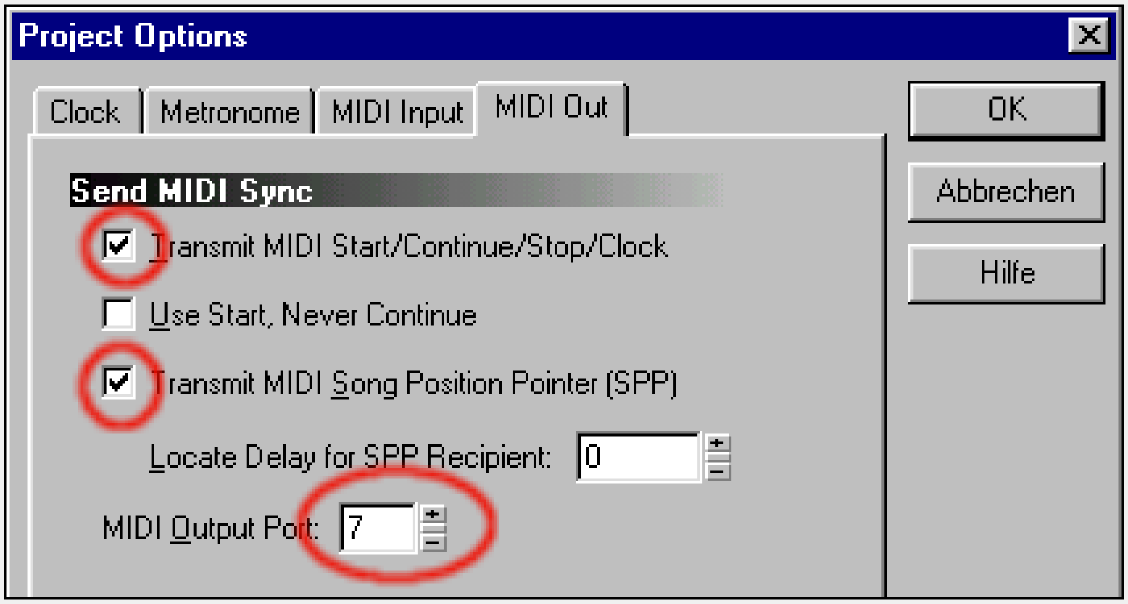
- **Compensate latency:** BUZZ has always some latency. This means if the song position of BUZZ matches the MIDI song position perfectly, BUZZ will still be late

because it takes some time to generate and output sound.
With this setting you can compensate this. It'll make BUZZ play ahead of the MIDI clock, thus getting the audible output in sync! **The value you have to enter here might differ a little from the latency of your soundcard (ASIO or directx drivers). The best setting also depends on the current song tempo (BPM) and TPB!** So you better play around with this a bit to find the perfect value for your configuration.
Rule of thumb: A higher value will make BUZZ play earlier, at a low value BUZZ will be too late.

- **Set:** apply the new latency. You can keep the Setup window open while playing, just change latency and press *Set* to easily try out different latency values.
- **current max:** The maximum *compensate latency* value you can enter. This depends on the current tempo (BPM) and TPB.

Settings for cakewalk pro audio

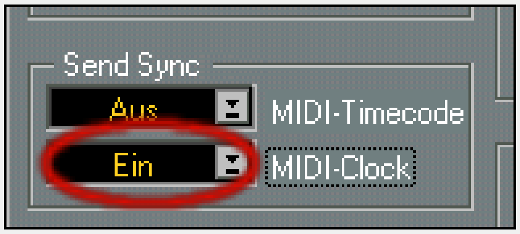
Go to *Options-Project* and select the *Midi Out*-tab. You can see the proper settings in the screenshot below. Note that the number for *Midi Output Port* corresponds to the number of the MIDI-device in pro audio (the number next to the MIDI Port in track view), and may be different on your system!
Note also that cakewalk is a pretty greedy cpu grabber. It might be that you can't use BUZZ with cakewalk on the same pc. It sometimes helps to bring BUZZ to the foreground while playing.



Send MIDI Sync
[✓] Transmit MIDI Start/Continue/Stop/Clock
[ ] Use Start, Never Continue
[✓] Transmit MIDI Song Position Pointer (SPP)
Locate Delay for SPP Recipient: 0
MIDI Output Port: 7

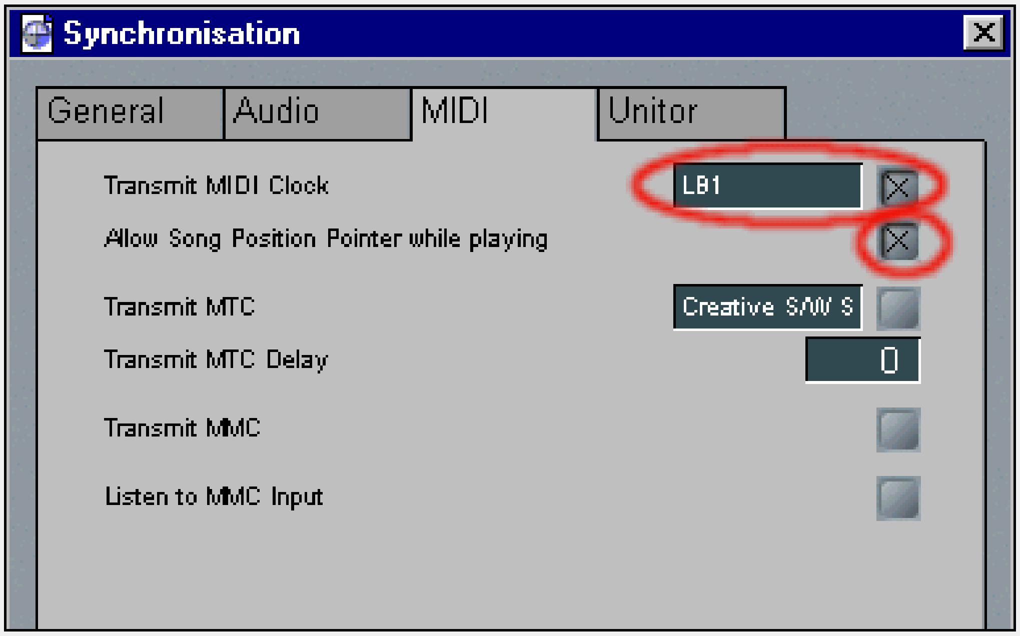
Settings for Cubase VST

Here, the important setting is hidden in *Options-Synchronisation*. Just select the MIDI device for MIDI-Clock, and all important MIDI timing/positioning signals will be sent to that port (btw: *Ein* means *On* in english :)


Send Sync
Aus MIDI-Timecode
Ein MIDI-Clock

Settings for Emagic Logic Audio

Go to *Options-Settings-Synchronisation Settings* and select the *MIDI*-tab. You can see the settings you should use for Msync in the screenshot below (of course the MIDI port may be different on your system!). **Hint:** disable *Auto enable external Sync* in the *General*-tab to get rid of some problems that occur when using a loopback (such as Hubis Loopback) for transmitting the MIDI clock signal.

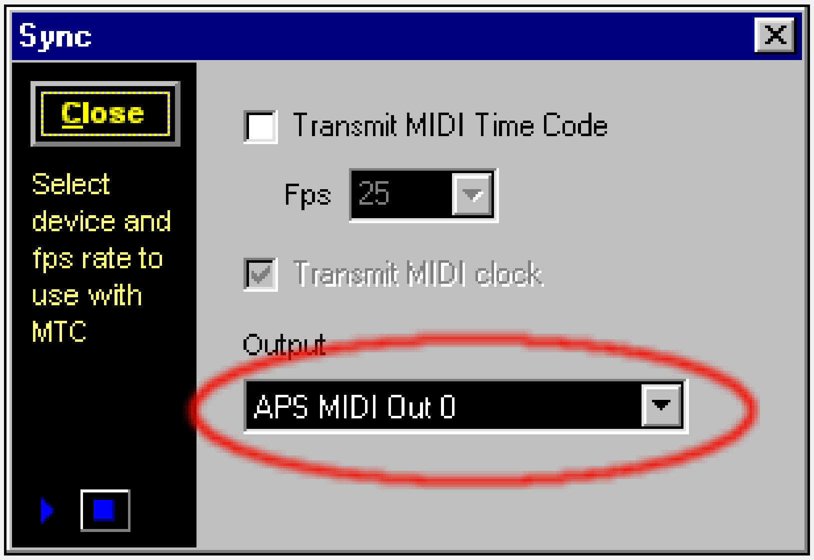

Synchronisation
General | Audio | MIDI | Unitor

Transmit MIDI Clock: LB1 [✓]
Allow Song Position Pointer while playing [✓]
Transmit MTC: Creative S/W S [ ]
Transmit MTC Delay: 0
Transmit MMC [ ]
Listen to MMC Input [ ]

Settings for Massiva

Also very easy to set up. Just go to *File-Sync* and select the MIDI output device. As always, this setting may differ on your system (I'm just sending it to *APS MIDI Out* here).
Massiva is a nice **freeware sequencer** by Jørgen Aase. You can get it **here**. Greets to Vectrex at

this place!



Transmit MIDI Time Code

Fps 25

Transmit MIDI clock

Output

APS MIDI Out 0

Important notes, please read!

- TPB must be one of these: 4, 6, 8, 12 or 24!
- Set the tempo of your BUZZ track to the correct tempo. (Same as the tempo in your sequencer).
- Put some empty bars at the beginning of your song. This will give Msync some time to catch the tempo/position.
- With this version of Msync, **you can use loops and jumps** in your BUZZ track. But remember that when you set a song position in your sequencer, Msync will just play from the corresponding position in BUZZ, which might be outside the looppoints. This will not crash BUZZ, but it'll get it out of sync.
- You can use the *loop start* pointer in BUZZ (set with ***Ctrl+B***) to define the starting point for Msync. So when you start the song in your sequencer, **Msync will play from that position**.
- Do not use tempo changes **(do not change BPM or TPB in the master track)** in your BUZZ tracks! This will make it impossible for Msync to work correctly. You can control the tempo of BUZZ from your sequencer. You can also change the song tempo gradually in your sequencer during the song, if these tempo changes don't occur too fast.
  Note that shuffling (by changing the BUZZ tempo) will not be possible anymore with Msync!
- Try to get a low latency in BUZZ. (Talking about BUZZ-latency now, not *latency compensation* in Msync!) Use directx-drivers, or if you have them, ASIO! I recommend to set latency under 50ms. The lower the better. Remember: BUZZ's latency can be compensated with the latency compensation option in Msync, but anyway, **a low latency in BUZZ will make the timing more stable**.

Troubleshooting

- If something doesn't work the way it should, have a look at the Setup (see above). The source of the problem might be indicated there.
- **When I start the song in my sequencer, BUZZ (Msync) always misses the beginning. Only after some seconds it is in sync with the sequencer.**
  Msync takes some time to catch the tempo and song position and apply these values to BUZZ (approx. 32 ticks). The rather complicated latency-elimination algorithm makes this kind of 'boot up' problem unavoidable. What you can do to get around this is:
  - start the song some bars before the part you want to hear. E.g. if you want to play from bar 16, set the song position to bar 13 (in your sequencer) and hit play. When the sequencer arrives at bar 16, Msync will (hopefully) have had enough time to 'catch the beat'.
  - insert some empty bars at the beginning of the song. This will give Msync some time to get in sync.
  - use TPB 8 instead of TPB 4, if you can change your track easily. This will half the time needed to start syncing.
- **I want to use MIDI port XYZ with Msync, but it doesn't work! Setup says 'CANNOT OPEN MIDI DEVICE'.**
  The MIDI port is in use by some other MIDI application. Make sure the port is not used by your sequencer (Start BUZZ first, then start the sequencer).
  Also make sure that BUZZ *doesn't* use the same MIDI port! (Have a look at *View-Preferences-MIDI Input* in BUZZ). The *MIDI input device* of BUZZ **must be different** to the port used by Msync!
- **At the beginning of the song, BUZZ is in sync, but after some minutes of playing it gets gradually out of sync.**
  Perhaps your MIDI connection is not reliable enough. The sequencer sends a MIDI tick every 1/24 beats, so if one of these ticks is skipped BUZZ will be late 1/24 beat. If a lot of skips occur, this may put BUZZ out of sync.
- **In Logic Audio, I can't play my song anymore. It stops or the cursor hangs.**
  You got a MIDI feedback problem. See *Settings for Emagic Logic Audio* above.

What is HACK?

HACK means: This machine accesses BUZZ internal variables directly. And this means for you: this machine can't be used with any other BUZZ machine host than BUZZ itself, and it probably can't be used with older versions of BUZZ. If there will be a new version of BUZZ one day, it'll perhaps **not be compatible anymore!**

Note

Making this machine for you was hard work, I had to hack into BUZZ first and then work out the mind-bogging formulas to sync BUZZ to something else, only by changing the tempo and estimating how fast BUZZ goes. Not to speak of the long programming, testing and debugging phase. However, I think it's quite usable now and will hopefully integrate BUZZ into pro studios more! If you like the machine, spread it, put my name on your CD covers, tell me that I rule or send me some money or a postcard :)
Have fun!

Peter *P. DooM* Kaufmann, April 2001

kaufmann@pop.agri.ch
P. DooM's Homepage

Donate

Polac VST Loaders for Jeskola Buzz

Version: 1.1.9.9.3 (01/09/09)
Author: Frank Potulski
Contact: **polac_and_gmx_bot_de**
Website: **http://www.xlutop.com/buzz**

Download

What's New

- option to skip a hanging plugin while vst scanning
- bugfix: some plugins did not initialize properly on song load
- bugfix: problems with 'sync to host' vsts if 'accurate bpm' mode was enabled in buzz
- bugfix: possible note hangs on incoming sustain events(cc64)

Installation

Since even some more experienced Buzz users might be getting in trouble with the installation of all .dll's "shipping" within the package, the files need to be placed like this:

Buzz\vstscan.exe
Buzz\dxscan.exe
Buzz\dxvst.dll [DX/DXi loader]
Buzz\pbuzz.dll [BUZZ loader]
Buzz\ppsy.dll [Psycle loader]

Buzz\Gear\Machines.dll

Buzz\Gear\Generators\Polac VSTi 1.1.dll [VST Instrument Loader]
Buzz\Gear\Generators\Polac MIDI Bus II.dll [->send midi from vst to other buzz machines]

Buzz\Gear\Effects\Polac VST 1.1.dll [VST Effect Loader]
Buzz\Gear\Effects\Polac Out II.dll [for multiple output support]
Buzz\Gear\Effects\Polac MIDI In.dll [->for additional midi in devices]

Buzz\Gear\Vst\midivst.dll [MIDI Out support, ASIO waveout required]
Buzz\Gear\Vst\dmovst.dll [DMO loader]
Buzz\Gear\Vst\ladspavst.dll [Ladspa loader]

Important: without the Machines.dll the loaders won't work properly!

You've got to modify your Buzz\Gear\index.txt in order to access VST Plugins from the right-mouse click menu.

*Polac VSTi 1.1, [PVSTi]VST Instruments
*Polac VST 1.1, [PVST]VST Effects

The Polac Out II.dll and Polac MIDI In doesn't need a specific index.txt entry in order to work.

Once you're done with this procedure, you're almost done. The only thing left to do is telling *BOTH* Polac Machines (VST & VSTi) which directories are scanned for vst plugins. Just place an instance of BOTH loaders into your Buzz setup, right-click them and choose "Preferences" from the menu. A new window will open and ask you to specify your VST plugin resources. The loaders will scan the choosen paths recursive (invoking subdirectories) once you're done with setting things up and press the "OK" button. The procedure stated above needs to be done ONCE for BOTH loaders. If you succeeded you'll now be able to load your VST plugin of choice from within the usual right-click menu in the machine view.

Congrats! You're done with the installation.

Parameters and Attributes

A short description of all parameters and attributes. Many of these parameters and attributes can also be accessed in the settings view(just open the loader window).

Global Parameters	
Dry Out	If you are using the VST Loader, this parameter controls the dry output volume.
Wet Out	This parameter controls the wet output volume.
Panning	Sets the panning position of the Wet Out signal.

Global Command	Nine global commands available: • 00: Transpose • 01: Tempo Sync • 02: Trigger Seq • 03: Bypass • 04: Morph: Prg A • 05: Morph: Prg B • 06: Global Inertia • 07: Vel Commands • 08: Dly Commands
Global Command Value	Sets the global command value. You can find a detailed global command description below.
Program	Selects a program of a VST(i). Only visible in pattern view.

Track Parameters	
Note	Note.
Note Velocity	Sets the velocity of a note.
Note Delay	Position of the note in a row/tick.
Note Cut	Cuts the note between two ticks.
Track Command	19 track commands are available: • 00: Retrigger • 01: ArpUp • 02: ArpDown • 03: Legato Mode • 04: Chord • 05: Slide Up/Down • 06: LFO Trigger • 07: LFO • 08: LFO Settings • 09: MIDI Message • 0A: Mute Track • 0B: Inertia • 0C: All Notes Off • 0D: Program Change • 0E: Bank Select • 0F: Ornament • 10: Automation Delay • 11: Psycle Note Command • Cx: Global Commands
Track Command Value	Sets the track command value. You can find a detailed track command description below.
Track Parameter	Sets the VST parameter you want to control. Also Pitchbend, MIDI CC, Mono/Polyaftertouch can be selected.
Track Parameter Value	Sets the value of the controlled parameter.
MIDI Channel	MIDI Channel selection.

Attributes	
MIDI In Channel	Sets the receiving MIDI Channel(0=all).
MIDI Out Channel	Specifies transmitting MIDI Channel(0=all).
MIDI Velocity Slope	Velocity sensitivity.
MIDI Velocity Min	Sets the velocity low value.
MIDI Velocity Max	Sets the velocity high value.
MIDI Transpose	Transposes the incoming MIDI notes.
MIDI: Receive Note	Enables/disables MIDI note receipt.
MIDI: ReceiveCC	Enables/disables MIDI CC receipt.

Quantisation	Note quantisation while recording. 0=OFF 1=1 2=1/2 3=1/3 4=1/4 5=1/5 6=1/6 7=1/8 8=1/12 9=1/16 10=1/32 11=1/64
Quantisation Note Off	Enables/disables the note off quantisation. Quantisation must be set before, otherwise enabling note off quantisation will affect nothing.
Record Note Off	Enables/disables note off recording.
Keyzone Lo	Keyzone Lo.
Keyzone Hi	Keyzone Hi.
Velocityzone Lo	Velocityzone Lo.
Velocityzone Zone Hi	Velocityzone Zone Hi.

Global Commands	
Pattern Transpose	**00 00xx** xx(0-30): Transpose in semitones (x-24).
Tempo Sync	**01 xxxx** xxxx(0-200): 0: Loader sends Buzz tempo to VST(i) 1-200: Loader sends selected tempo to VST (i)
Trigger Seq	**02 xxxx** xxxx(0-FFFE): 0: Stops sequence playback. 1-FFFE: Triggers sequence on measure x-1. Note: This note effect is useful for a couple like Reaktor or Fruityloops. You can jump to a certain song position with this command.
Bypass	**03 000x** x(0-1): 0: Bypass disabled 1: Bypass enabled
Morph: Prg A	**04 xxxx** xxxx(0-FFFE): Sets program A(for program morphing purpose). You can morph programs in the track parameter column(30FB: Morph Programs).
Morph: Prg B	**05 xxxx** xxxx(0-FFFE): Sets program B(for program morphing purpose). You can morph programs in the track parameter column(30FB: Morph Programs).
Global Inertia	**06 xxxx** xxxx(0-FFFE): Sets the inertia for all tracks(x/256 ticks).

Vel Commands	**07 0xyy** x(0-2): 0: Humanize Velocity yy(0-7F): controls humanize amount 1: Quantize Velocity yy(0-0F): quantises to velocity/y 2: Constant Velocity yy(0-7F): sets velocity constant to y
Dly Commands	**08 xxyy** xx(00,01,02,04,1x,2x): 00: Humanize Note Delay yy(0-7F): controls humanize amount 01: Quantize Note Delay yy(0-0F): quantises to tick/y 02: Constant Note Delay yy(0-FF): delays notes for y/16 tick(s) 04: Tick Affinity yy(0-FF): controls the note delay affinity 1x: Shuffle x(0-F): shuffle step x*2 tick(s) yy(0-FF): shuffle amount 2x: Constant Note Delay II xyy(0-FFF): delays notes for y/32 tick(s)

Track Commands	
Retrigger	**00 xy0z** x(0-F): Retriggerlength y(0-1): 0: Retriggerlength 1/(x+1) Ticks 1: Retriggerlength x+1 Ticks z(0-F): Number of Retriggers Note cut still influences here the note length, relative to retriggerlength/number of retriggers.
ArpUp	**01 abyz** a(0-F): Arpeggiolength a+1 Ticks b(0-F): Number of steps y(0-F): Second Note z(0-F): Third Note Note Cut still influences here the note length, relative to arpeggiolength/number of steps.
ArpDown	**02 abyz** a(0-F): Arpeggiolength a+1 Ticks b(0-F): Number of steps y(0-F): Second Note z(0-F): Third Note Note Cut still influences here the note length, relative to arpeggiolength/number of steps.

Legato Mode	**03 xxxx** xxxx(0-FFFE): Note Overlap Useful for VSTi's with glide/portamento mode. If this Track Command is set, the previous note glides to the current.
Chord	**04 wxyz** w(0-F): Second chord note(unused if 0). x(0-F): Third chord note(unused if 0). y(0-F): Fourth chord note(unused if 0). z(0-F): Fifth chord note(unused if 0). Quite useful for some lazy people.
Slide Up/Down	**05 xxyy aaaa bbbb** xx(0-7F): slide start point. yy(0-7F): slide end point. aaaa(0-30FF): the param you want to slide bbbb(0-7FFF): slide speed b/256 tick. Slides the selected track parameter.
LFO Trigger	**06 xxyy** xx(0-FF): LFO Rate yy(0-FF): LFO Depth Starts the LFO,resets LFO.
LFO(free run)	**07 xxyy** xx(0-FF): LFO Rate yy(0-FF): LFO Depth Starts the LFO without resetting.
LFO Settings	**08 0xyz** x(0-f): LFO Phase/Offset y(0-1): LFO Mode 0: Hz 1: Tick z(0-5): LFO Shape 0: Sine 1: Square 2: Saw Up 3: Saw Dowm 4: Triangle 5: LFO Shape: Random Track-dependant LFO Settings.
MIDI Message	**09 xxyy** xx(0-FF): MIDI Message # 00-7F: CC 0-7F 80-FD: user-defined MIDI Message FE: Pitch Bend Range FF: Pitch Bend yy(0-FF): Value The MIDI Messages can be edited: ->Default Valus->Midi Messages.

Mute Track	**0A 000x** x(0-1): mute/unmute track Mutes/unmutes the track, Mute track will only disable the note playback on the track, the parameter automation is not influenced by this track command.
Inertia	**0B xxxx** xxxx(0-FFFE): Track-dependant Inertia, x/256 Ticks.
All Notes Off	**0C 000x** x(0-1): Sends a note off for all active pattern notes.
Program Change	**0D 00yy** yy(0-7F): Program Change
Bank Select	**0E xxyy** xx(0-7F): MSB yy(0-7F): LSB
Ornament	**0F x0yz** x(0-1): Mode selection y(0-F): Length tick / (2 << y) z(0-F): Relative note
Automation Delay	**10 xxxx** xxxx(0-FFFE): Sets the automation delay position in a tick, if set to 0 the parameter automation syncs to the note delay. Note that the parameter inertia does not work if you use this command.
Psycle Note Command	**11 xxyy** Special note command for psycle plugins only: xx(0-FF): psycle command yy(0-FF): psycle argument
Global Commands	**Cx yyyy** Cx(0-9): Global command selection. yyyy(0-FFFE): See above how to use the respective global command. Useful if you want to use more than one global command in a row.

Shortcuts

There are some shortcuts defined.

Shortcuts	
`	Switch between the buzz mainwindow and the previously active window.
CTRL+SHIFT	Center loader window.
CTRL+D	Stores the current program to favorites.
Left/Right	Next/previous program in bank.
CTRL+Left/Right	Next/previous program via midi program change.
CTRL+Up/Down	Midi bank select.
/,*	Midi Transpose: Octave up/down.
+,-	Midi Transpose: Semitone up/down.
CTRL+LDBLCLICK	On machine: Opens GUI view.
SHIFT+LDBLCLICK	On machine: Opens settings view.
CTRL+SHIFT+LDBLCLICK	On machine: Opens parameter view.
ALT+LDBLCLICK	On machine: Opens buzz parameters.
LDBLCLICK	Minimizes loader window.
RDBLCLICK	Parameter list: Fits parameter list to screen.
CTRL+SPACE	Closes all loader windows except the active one.
ESCAPE	Closes active loader window.

SHIFT	Parameter list: enables the parameter fine adjustment.
TAB	Windows styled cycling through all opened loader instances.
CTRL+TAB	Cycles through the views of a loader window.
CTRL+Return	Activates/Deactivates the overwrite mode.
F2	GUI view.
F3	Paramlist view.
F4	Settings view.
CTRL+F8	Bypass On/Off.
F9	Enables/disables MIDI CC receipt.
F10	Enables/disables MIDI note receipt.
F11	Show/Hide all loader windows.
F12	All Notes Off(MIDI Reset).

Loader Window and Menus

If you have successfully loaded a plugin you can open the loader window. Three views are available: GUI view, the paramlist view and the settings view. There are several menus:

Machine Menu	
Open...	Loads a plugin via open dialog.
Plugins>	All vst plugins which could be found in the selected directories.
Editor...	Opens the loader window.
MIDI Send...	There are a couple of plugins which are midi only, which means they are producing midi events. These events can be sent to other machines to control them with these events (Trollo, QuickKeys, MoCon...-plugins).
Preferences...	Default settings for attributes and a couple of other things. Here you can configure the loader startup values.
About...	Blabla...

Presets Menu	
Load...	Loads a complete bank or a single program. You can import presets from Cubase, Logic as well as Orion.
Save...	Saves a complete bank or a single program. You can save the presets as FXP or FXB.
Rename...	Renames the active program.
Copy	Makes a copy of the active program.
Paste	Pastes a previously copied program to the active program.
Preset Factory...	You're too lazy, stupid or not just in the mood to create a complex preset yourself? You don't need to - let the Preset Factory handle that for you!
Show Preset Numbers	Bank preset numbering on/off.
Prefer Builtin/Favorites	Defines what is displayed in the program menu by default. If set to 'builtin' you'll see the default bank of a plugin, otherwise the programs of your favorites(Organize...).
Organize...	Opens a preset manager where you can manage your favorite programs.
Store...	Stores the selected program to the favorites (Shortcut: CTRL+D).
Programs	All programs of the bank or favorites are listed here, depending on what is set as default. A couple of plugins will have its own program management, therefore these programs cannot be displayed here.

Learn Menu	
Auto Learn	If enabled, all touched controls in the vst gui get assigned to buzz sliders - track by track, one after another (until the last available track slider has been reached).
Learn MIDI CC	Necessary to make the loader recognize vst plugins which use the Midi CC method for controller automation.

Clear Params	Wipes all learned parameters.
Save as Default	Writes an .dat containing all learned parameters of a plugin. These settings will be automatically used if you load the plugin.
Restore Default	Reverts all changes to the saved defaults.
Learned Parameter List	Shows the list of all available tracks and their learned VST parameters.

View Menu	
GUI	Shows the VST plugins' gui (default view).
Parameter List	A schematic listing of all available VST plugin parameters.
Settings	All loader settings (as found in the machine attributes) can be found here as well besides a number of track dependant things (track selector, LFO...).
Plugin Information	Opens dialog containing some informations about a plugin.
MIDI: Receive Note	Enables/disables the MIDI note receipt.
MIDI: Receive CC	Enables/disables the MIDI CC receipt.
Bypass	Bypass on/off.
All Notes Off	Sends a "all notes off" to plugin.

GUI View

The default view once a plugin is loaded into the Polac machines. A double-right-click on a VST or VSTi instance will open it.

Parameter View

A list view showing all parameters of the loaded VST plugin. All parameters can be edited here as well. Use SHIFT+Leftmouse for fine tuning values. To the left of each paraemeter you'll find a button. You can select/deselect here a parameter for randomizing or morphing purpose(preset factory).

Settings View

Surely the most important section of the loader besides the vst gui. There are a couple of sections here:

Track Settings:

- Track Selector: Here you can select a track for record, mute a track.
  -Leftclick: (de)selects track for recording
  -DoubleLeftclick: (de)selects all tracks
  -CTRL+Leftclick: (un)mutes track.
  -CTRL+DoubleLeftclick: (un)mutes all tracks
  -Rightclick: selects active track(track settings)
- Overwrite: When enabled, the pattern data will be overwritten during recording. You can optionally keep the automation or non-automation data(rightclick the overwrite button). Overwrite can be optionally toggled via CC64, quite useful if using a foot switch.
- Inertia: Used in order to smooth parameter automation changes. 0.00 ticks might occur steppy (depending on the plugin), above 1 isn't very responsible for live purposes but can be used for making long parameter movements easier. A good, responsive default is 0.50 ticks.
- LFO Shape: Available waveforms as LFO Shape:
  -Sine
  -Square
  -Saw Up
  -Saw Down
  -Triangle
  -Random
- LFO Mode: The track LFO's can operate with a HZ or tick related measure.
- LFO Phase: Sets the starting point for each LFO.
- Assigned Parameter/CC: Shows the list of all available vst plugin parameters. Can be used to assign vst parameter to a track parameter by simply selecting it. The value slider can be directly linked to a midi cc as well as its value range can be customized. A right-mouse click on it opens the regarding menu.
- Midi Channel: Specifies the Midi Channel that is used for each track (!**ATTENTION**! Not to be confused with the Global Midi Channels!).

MIDI Settings:

- In Channel: MIDI Channel that is beeing used by Loader instance to listen for incoming MIDI events.
- Out Channel: MIDI Channel that is used by Loader to sent MIDI events.
- Transpose: Transposition of incoming notes by value(note).

- Velocity Slope: Velocity sensitivity.
- Velocity Min: Sets the minimum velocity.
- Velocity Max: Sets the maximum velocity.
- Quantize: Determines the quantisation raster of incoming midi events for recording and ONLY for recording. Changes made after recording don't have an effect.
- Quantize NoteOff: If this switch is enabled, NoteOff events are quantized as well.
- Record NoteOff: Enables/disables note off recording.
- MIDI Mode: Mode A can be referred as "standard" mode. Incoming MIDI events will be used by all VST Loader instances which have Midi enabled. Mode B will only enable Midi for the highlighted VST Loader instance.

Global Settings:

- Dry Out(PVST only)
- Wet Out
- Panning
- Transpose: Global transposition of all notes for playback.
- Tempo Sync: If Buzz is enabled, Buzz' master tempo will be used for the VST loader tempo syncronisation. In case you're working with tempo changes (e.g. shuffle using master tempo changes) it's quite handy to lock the loaders' tempo manually.
- Link Dry/Wet(PVST only): If enabled, dry out and wet out can be controlled with only one slider.

Miscellaneous Settings:

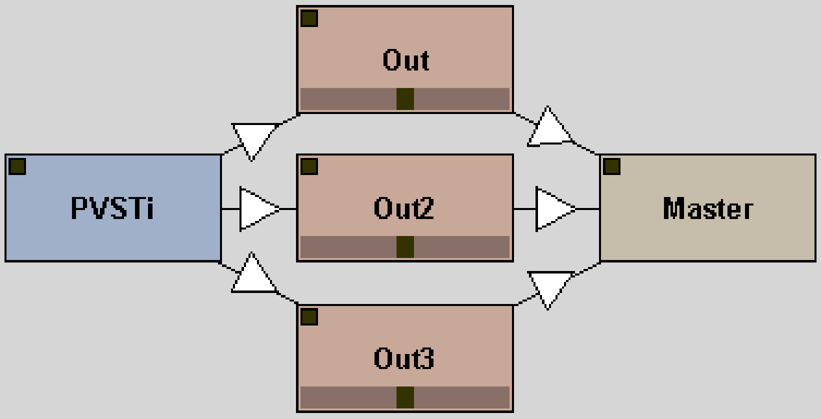
- Automation Mode:
  -Normal: if activated, the loader records automation data as long as you move the fader/knob.
  -Touchfader: if activated, the loader records automation data as long as the fader/knob is held, even if you stop the movement.
- Virtual Keyboard: If enabled, the pc keyboard can be used to send midi notes to the vst plugin.
- MIDI Mode: Mode "User-Defined" can be referred as standard mode. Incoming MIDI events will be used by all VST Loader instances which have Midi enabled. Mode "Active Editor" will only enable Midi for the highlighted VST Loader instance. Mode "All Editors" will enable Midi for all vst's with opened editors.
- Refresh Button: use this to update the settings view.
- Morph: Here you can select the programs for morph purpose. If some programs are selected you can select "Morph Prgs" in the "Track Parameter" combo box. The slider movement can be recorded to track parameter colums now.

How To Use...

Some hints:

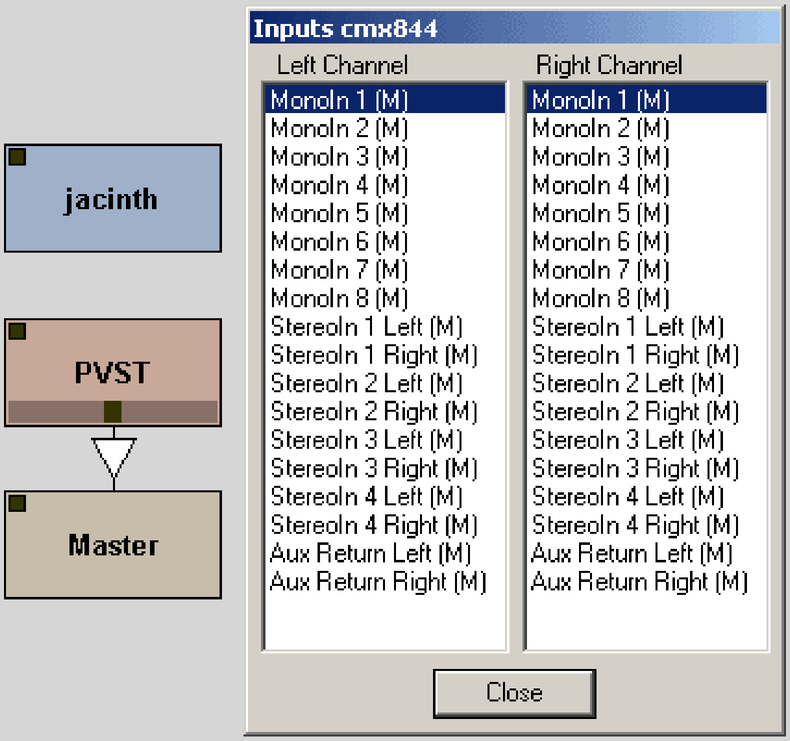
Multi-out VST's

If a VST has multiple outputs, you can use the Polac Out II to access these outputs. The picture below shows how you have to connect the machines. Polac Out II will detect the connected machine. Doubleclicking this machine will open a dialog showing up all available outputs of the connected VST.



Multi-in VST's

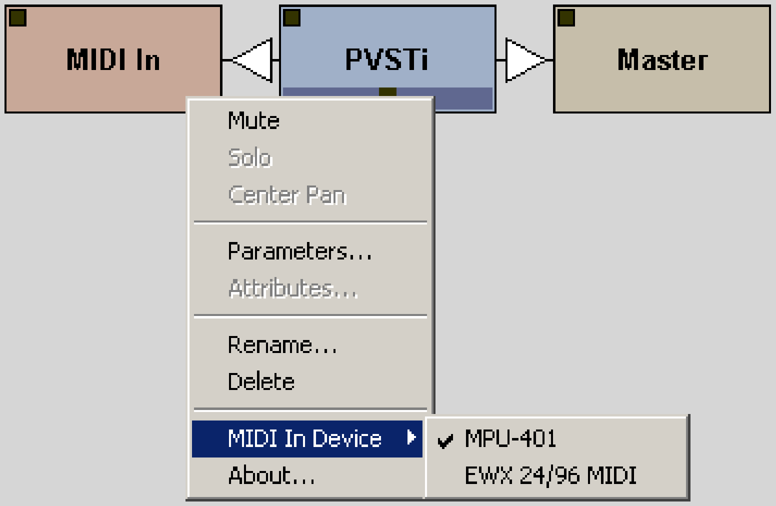
If a VST has more than two inputs, a dialog will automatically pop up when a new machine is connected to the loader. All available inputs are listed in this dialog, you only have to choose the appropiate inputs.



Left Channel	Right Channel
MonoIn 1 (M)	MonoIn 1 (M)
MonoIn 2 (M)	MonoIn 2 (M)
MonoIn 3 (M)	MonoIn 3 (M)
MonoIn 4 (M)	MonoIn 4 (M)
MonoIn 5 (M)	MonoIn 5 (M)
MonoIn 6 (M)	MonoIn 6 (M)
MonoIn 7 (M)	MonoIn 7 (M)
MonoIn 8 (M)	MonoIn 8 (M)
StereoIn 1 Left (M)	StereoIn 1 Left (M)
StereoIn 1 Right (M)	StereoIn 1 Right (M)
StereoIn 2 Left (M)	StereoIn 2 Left (M)
StereoIn 2 Right (M)	StereoIn 2 Right (M)
StereoIn 3 Left (M)	StereoIn 3 Left (M)
StereoIn 3 Right (M)	StereoIn 3 Right (M)
StereoIn 4 Left (M)	StereoIn 4 Left (M)
StereoIn 4 Right (M)	StereoIn 4 Right (M)
Aux Return Left (M)	Aux Return Left (M)
Aux Return Right (M)	Aux Return Right (M)

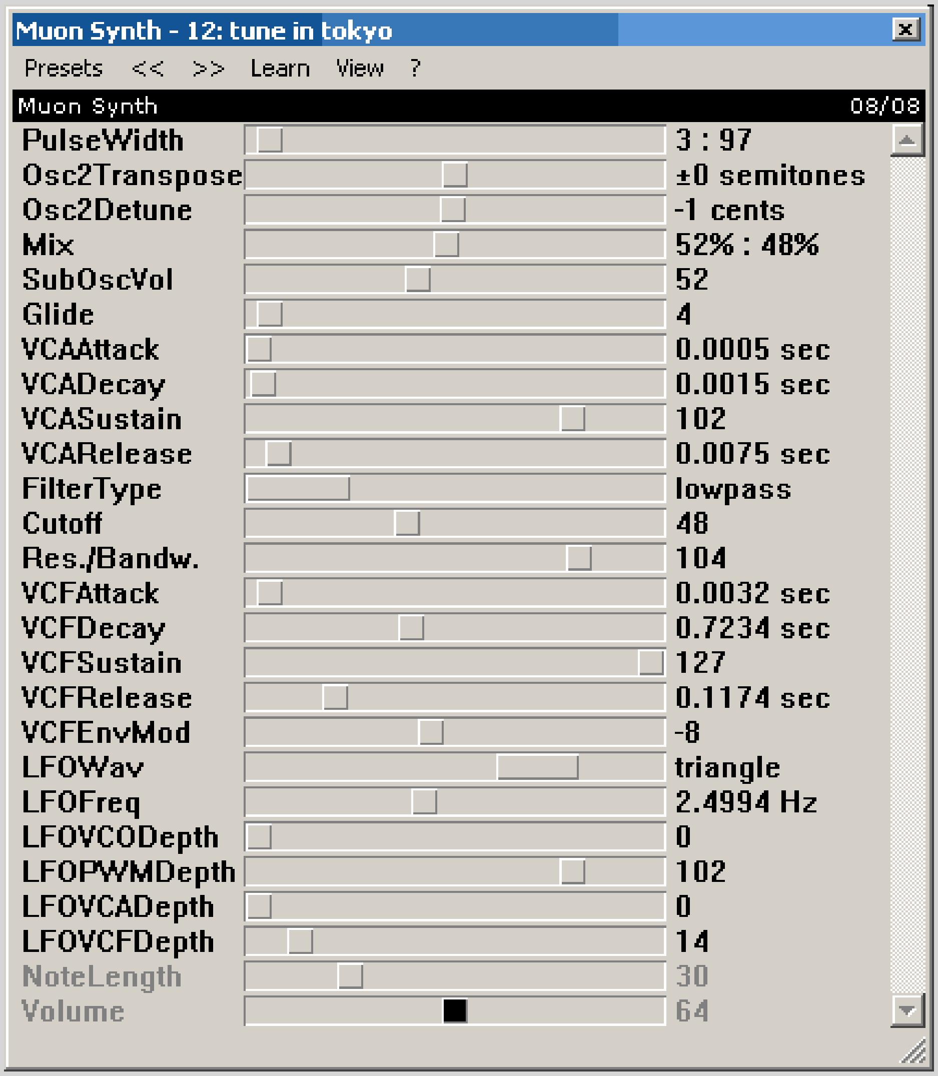
Additional MIDI In Devices

You can use the Polac MIDI In machine to make other machines able to receive midi data also from other midi devices than the buzz default midi device. See below how to use this machine:

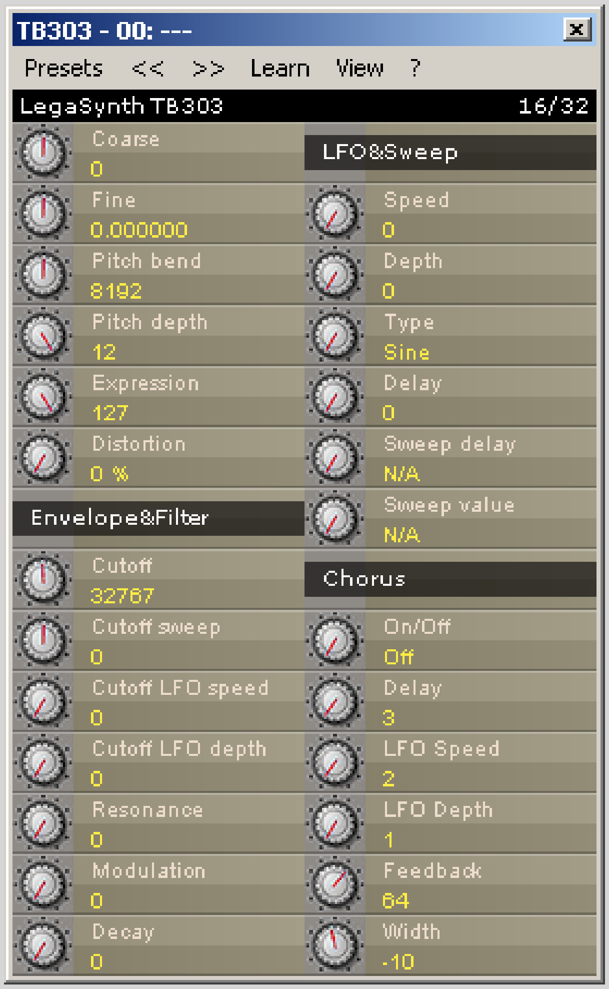


Buzz/Psycle wrappers

To use them inside the loader simply place some buzz or psycle machines in one of your vst directories. Alternatively you can create a shortcut of a machine via explorer and move the shortcut to a vst directory. The machines will be detected by the loader and can be selected via pluign menu later on.



PulseWidth	3 : 97
Osc2Transpose	±0 semitones
Osc2Detune	-1 cents
Mix	52% : 48%
SubOscVol	52
Glide	4
VCAAttack	0.0005 sec
VCADecay	0.0015 sec
VCASustain	102
VCARelease	0.0075 sec
FilterType	lowpass
Cutoff	48
Res./Bandw.	104
VCFAttack	0.0032 sec
VCFDecay	0.7234 sec
VCFSustain	127
VCFRelease	0.1174 sec
VCFEnvMod	-8
LFOWav	triangle
LFOFreq	2.4994 Hz
LFOVCODepth	0
LFOPWMDepth	102
LFOVCADepth	0
LFOVCFDepth	14
NoteLength	30
Volume	64

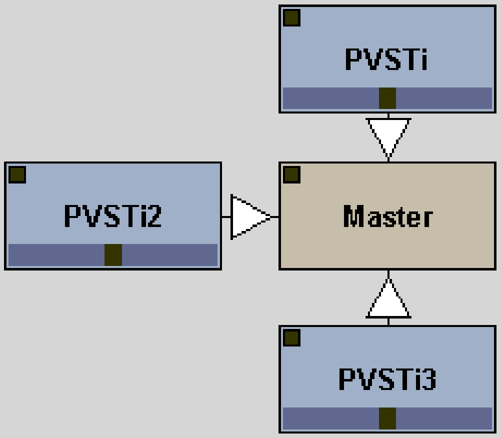

TB303 - 00: ---

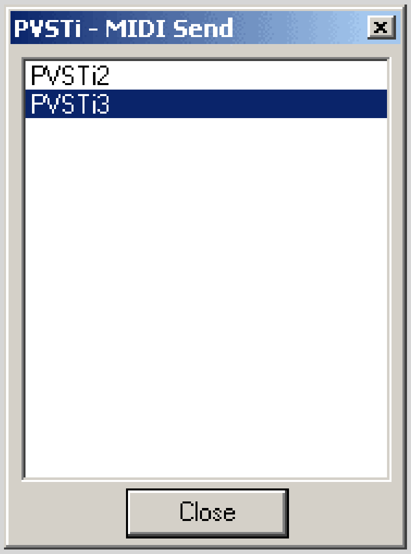
LegaSynth TB303 16/32

Coarse 0	LFO&Sweep
Fine 0.000000	Speed 0
Pitch bend 8192	Depth 0
Pitch depth 12	Type Sine
Expression 127	Delay 0
Distortion 0 %	Sweep delay N/A
Envelope&Filter	Sweep value N/A
Cutoff 32767	Chorus
Cutoff sweep 0	On/Off Off
Cutoff LFO speed 0	Delay 3
Cutoff LFO depth 0	LFO Speed 2
Resonance 0	LFO Depth 1
Modulation 0	Feedback 64
Decay 0	Width -10

MIDI-generating VST's (Midi Send...)

In the loader rightclick menu is an entry called "MIDI Send". A couple of plugins are generating midi. These midi events can be directly sent to other loader instances to control them with these midi events. Only if a plugin is a "midi generator", the midi send dialog will shop up other loader instances where to send the events.





Parameter Automation

Up to 128 VST parameters can be controlled simultanously, depending on the number of tracks. VST parameters/MIDI CCs can be assigned in this way:

1. Open the GUI/Params window.
2. (a) Open the Learn menu and select a track you want to assign with a VST parameter/MIDI CC. If you want to learn a MIDI CC also enable "Learn MIDI CC".
   (b) Open the Learn menu and select Auto Learn. If you want to learn a MIDI CC also enable "Learn MIDI CC".
3. Now you only have to move a Fader/Knob/Button, in most cases the parameter is learned now. Alternatively you can move a Controller/Pitchbend of your masterkeyboard or faderbox to learn them.

Several Notes:

- there are a couple of plugins that cannot be assigned to a track in this way. So you have assign the VST parameters manually(=>Parameters...).
- you can unbind a parameter by rightglicking on a learned parameter in the parameter list.

Midi Message Editing

Midi messages can be used via track command 07. Predefined midi messages are:
00-7F: MIDI CC
FE: Pitch Bend Range
FF: Pitch Bend

Messages [80] to [FD] are undefined. To edit the midi messages, right-click the loader, open the Default Values... dialog, click the midi message button. This will open the midi message.ini in your texteditor. A short example how to edit the midi message.ini:

```
All is numbered from [00]-[FF], here we are editing midi message [FE].

[FE] Bn 64 00,Bn 65 00,Bn 06 xx //Pitch Bend Range

n: A byte with n will be combined with the track midichannel
xx: the variable value
//: Comment.

Each byte has to be separated from the other bytes with either a space, tab or comma.
Three bytes will give one midi event.
```

Preset Factory

You're too lazy, stupid or not just in the mood to create a complex preset yourself? You don't need to - let the Preset Factory handle that for you!

The Preset Factory allows you to create new sounds in 3 ways:

1. Morphing:
   The morph slider will, generally spoken, take preset A, B and linearly blend between both. If there wasn't the Mode setting above. "ALL" will really just do the blend-job, but then it get's interesting again. "Selected" will only touch the parameters which are selected in the parameter list (F3). "Not selected" does the same with all non-selected parameters. "Learned" only affects the vst parameters that have been assigned to buzz parameters. "Not learned" vice versa.

2. Genetics ("Breed"):
   The genetics slider determines the amount of vst parameters that will be taken from both presets. It doesn't change the preset unless you press the "Breed!" button. Once it's pressed, it takes the specified amount of (random!) vst parameters to create a new sound. Of course, you can point out which parameters may be used for this procedure using the Mode as in the morphing.

   To make it easier for you:
   E.g. your plugin has got 10 vst parameters and you don't move the genetics slider, 5 parameters from both selected presets will remain in the new preset. Which exact parameters are used is Breed's choice (random).

3. Randomize:
   Good old-fashioned patch randomizing. You can specify the amount of randomization by enabling the % range. The Mode is enabled for usage with randomizing as stated above.

Preferences

Here you can configure the loader preferences, which means that the loader will use these settings each time you start a new loader instance. Also the vst directories can be configured here. The preferences dialog can be accessed via the machine menu(rightclick on loader). There are three tabs, two for global settings which will have effect for all plugins and one for settings done per plugin:

Global I:

Here one can set the default values for the loader attributes, also some other things can be defined here such as inertia, the number of tracks, midi or automation mode etc...

Global II:

Select your vst directories and some settings related to the dx/buzz loader, also an option to select the PVST data folder, 'My Documents' is set by default.

Plugin:

Some plugin-specific settings:

- VST Keys: If enabled the loader sends keystrokes to the plugin.
- Fixed Length Processing: The loader processes samples with a constant blocksize, note that this will delay the output for 256 samples. Only a couple of plugins will need this enabled (Spinaudio/PPG).
- Save Bank To Song: All Programs of a VST(i) are saved to the buzzsong.
- Save Prg To Song: Only the selected program of a VST(i) is saved to the buzzsong. A couple of plugins will cause problems on bank saving(Delay Lama,FFX4), so this has to be enabled.
- Use sizeWindow(): Disable this if a gui does not resize correctly.
- Stereo Fix: some plugins will remain mono if this one is not enabled(Cyanide).
- Disable shortcuts/virtual keyboard. Disable if a plugin has its own keyboard handling.

Links

More infos about these loaders can be found here:

PVST FAQ
buzzmachines.com message board posts about PVST

Some buzz-related sites:

buzzmachines.com - latest infos on effects/generators
buzzchurch.com - a new buzz-related forum
the new buzz beta - oskari's latest buzz binaries
buzzmusic - site hosting buzz-generated music
buzzformat.com - buzz overloader
cyanwerks.com - buzzdev archive here
noolmusic - lots of infos about buzz here
buzé - remake of buzz, open source
aldrin - linux/windows buzzclone, open source

Buzz Devs:

Btdsys
Fuzzpilz
ld0d

Various links:

kvraudio.com
rjkole.com
ronnypries.de

Thanks To...

Thomas Potulski - for programming help and tips

Ronny Pries - for lots of ideas and help, help on this manual, beta testing, for being sort of project manager ;)

oskari - for buzz and newbuzz help

CyanPhase - for the great buzz tutorials

usr - for suggestions and help with the machines.dll

Cameron Bonde - for tons of feature suggestions and beta testing

CyanPhase - for the great buzz tutorials

ld0d - for some help on the vst menus in the machine menu

djlaser - for tips and suggestions

P. DooM - for the buzzhack sourcecode

Also big thanks to all on the beta list:

.tOm, ps, Wayfinder, soma, WakaX, mute, btd, Rolf Kohl, Mva, Boneyhands and the rest I forgot here.

Legal Notice

Donate

Polac VST Loaders for Jeskola Buzz

Version: 1.1.9.9.3 (01/09/09)
Author: Frank Potulski
Contact: **polac_and_gmx_bot_de**
Website: **http://www.xlutop.com/buzz**

Download

What's New

- option to skip a hanging plugin while vst scanning
- bugfix: some plugins did not initialize properly on song load
- bugfix: problems with 'sync to host' vsts if 'accurate bpm' mode was enabled in buzz
- bugfix: possible note hangs on incoming sustain events(cc64)

Installation

Since even some more experienced Buzz users might be getting in trouble with the installation of all .dll's "shipping" within the package, the files need to be placed like this:

Buzz\vstscan.exe
Buzz\dxscan.exe
Buzz\dxvst.dll [DX/DXi loader]
Buzz\pbuzz.dll [BUZZ loader]
Buzz\ppsy.dll [Psycle loader]

Buzz\Gear\Machines.dll

Buzz\Gear\Generators\Polac VSTi 1.1.dll [VST Instrument Loader]
Buzz\Gear\Generators\Polac MIDI Bus II.dll [->send midi from vst to other buzz machines]

Buzz\Gear\Effects\Polac VST 1.1.dll [VST Effect Loader]
Buzz\Gear\Effects\Polac Out II.dll [for multiple output support]
Buzz\Gear\Effects\Polac MIDI In.dll [->for additional midi in devices]

Buzz\Gear\Vst\midivst.dll [MIDI Out support, ASIO waveout required]
Buzz\Gear\Vst\dmovst.dll [DMO loader]
Buzz\Gear\Vst\ladspavst.dll [Ladspa loader]

Important: without the Machines.dll the loaders won't work properly!

You've got to modify your Buzz\Gear\index.txt in order to access VST Plugins from the right-mouse click menu.

*Polac VSTi 1.1, [PVSTi]VST Instruments
*Polac VST 1.1, [PVST]VST Effects

The Polac Out II.dll and Polac MIDI In doesn't need a specific index.txt entry in order to work.

Once you're done with this procedure, you're almost done. The only thing left to do is telling *BOTH* Polac Machines (VST & VSTi) which directories are scanned for vst plugins. Just place an instance of BOTH loaders into your Buzz setup, right-click them and choose "Preferences" from the menu. A new window will open and ask you to specify your VST plugin resources. The loaders will scan the choosen paths recursive (invoking subdirectories) once you're done with setting things up and press the "OK" button. The procedure stated above needs to be done ONCE for BOTH loaders. If you succeeded you'll now be able to load your VST plugin of choice from within the usual right-click menu in the machine view.

Congrats! You're done with the installation.

Parameters and Attributes

A short description of all parameters and attributes. Many of these parameters and attributes can also be accessed in the settings view(just open the loader window).

Global Parameters	
Dry Out	If you are using the VST Loader, this parameter controls the dry output volume.
Wet Out	This parameter controls the wet output volume.
Panning	Sets the panning position of the Wet Out signal.

Global Command	Nine global commands available: • 00: Transpose • 01: Tempo Sync • 02: Trigger Seq • 03: Bypass • 04: Morph: Prg A • 05: Morph: Prg B • 06: Global Inertia • 07: Vel Commands • 08: Dly Commands
Global Command Value	Sets the global command value. You can find a detailed global command description below.
Program	Selects a program of a VST(i). Only visible in pattern view.

Track Parameters	
Note	Note.
Note Velocity	Sets the velocity of a note.
Note Delay	Position of the note in a row/tick.
Note Cut	Cuts the note between two ticks.
Track Command	19 track commands are available: • 00: Retrigger • 01: ArpUp • 02: ArpDown • 03: Legato Mode • 04: Chord • 05: Slide Up/Down • 06: LFO Trigger • 07: LFO • 08: LFO Settings • 09: MIDI Message • 0A: Mute Track • 0B: Inertia • 0C: All Notes Off • 0D: Program Change • 0E: Bank Select • 0F: Ornament • 10: Automation Delay • 11: Psycle Note Command • Cx: Global Commands
Track Command Value	Sets the track command value. You can find a detailed track command description below.
Track Parameter	Sets the VST parameter you want to control. Also Pitchbend, MIDI CC, Mono/Polyaftertouch can be selected.
Track Parameter Value	Sets the value of the controlled parameter.
MIDI Channel	MIDI Channel selection.

Attributes	
MIDI In Channel	Sets the receiving MIDI Channel(0=all).
MIDI Out Channel	Specifies transmitting MIDI Channel(0=all).
MIDI Velocity Slope	Velocity sensitivity.
MIDI Velocity Min	Sets the velocity low value.
MIDI Velocity Max	Sets the velocity high value.
MIDI Transpose	Transposes the incoming MIDI notes.
MIDI: Receive Note	Enables/disables MIDI note receipt.
MIDI: ReceiveCC	Enables/disables MIDI CC receipt.

Quantisation	Note quantisation while recording. 0=OFF 1=1 2=1/2 3=1/3 4=1/4 5=1/5 6=1/6 7=1/8 8=1/12 9=1/16 10=1/32 11=1/64
Quantisation Note Off	Enables/disables the note off quantisation. Quantisation must be set before, otherwise enabling note off quantisation will affect nothing.
Record Note Off	Enables/disables note off recording.
Keyzone Lo	Keyzone Lo.
Keyzone Hi	Keyzone Hi.
Velocityzone Lo	Velocityzone Lo.
Velocityzone Zone Hi	Velocityzone Zone Hi.

Global Commands	
Pattern Transpose	**00 00xx** xx(0-30): Transpose in semitones (x-24).
Tempo Sync	**01 xxxx** xxxx(0-200): 0: Loader sends Buzz tempo to VST(i) 1-200: Loader sends selected tempo to VST (i)
Trigger Seq	**02 xxxx** xxxx(0-FFFE): 0: Stops sequence playback. 1-FFFE: Triggers sequence on measure x-1. Note: This note effect is useful for a couple like Reaktor or Fruityloops. You can jump to a certain song position with this command.
Bypass	**03 000x** x(0-1): 0: Bypass disabled 1: Bypass enabled
Morph: Prg A	**04 xxxx** xxxx(0-FFFE): Sets program A(for program morphing purpose). You can morph programs in the track parameter column(30FB: Morph Programs).
Morph: Prg B	**05 xxxx** xxxx(0-FFFE): Sets program B(for program morphing purpose). You can morph programs in the track parameter column(30FB: Morph Programs).
Global Inertia	**06 xxxx** xxxx(0-FFFE): Sets the inertia for all tracks(x/256 ticks).

Vel Commands	**07 0xyy** x(0-2): 0: Humanize Velocity yy(0-7F): controls humanize amount 1: Quantize Velocity yy(0-0F): quantises to velocity/y 2: Constant Velocity yy(0-7F): sets velocity constant to y
Dly Commands	**08 xxyy** xx(00,01,02,04,1x,2x): 00: Humanize Note Delay yy(0-7F): controls humanize amount 01: Quantize Note Delay yy(0-0F): quantises to tick/y 02: Constant Note Delay yy(0-FF): delays notes for y/16 tick(s) 04: Tick Affinity yy(0-FF): controls the note delay affinity 1x: Shuffle x(0-F): shuffle step x*2 tick(s) yy(0-FF): shuffle amount 2x: Constant Note Delay II xyy(0-FFF): delays notes for y/32 tick(s)

Track Commands	
Retrigger	**00 xy0z** x(0-F): Retriggerlength y(0-1): 0: Retriggerlength 1/(x+1) Ticks 1: Retriggerlength x+1 Ticks z(0-F): Number of Retriggers Note cut still influences here the note length, relative to retriggerlength/number of retriggers.
ArpUp	**01 abyz** a(0-F): Arpeggiolength a+1 Ticks b(0-F): Number of steps y(0-F): Second Note z(0-F): Third Note Note Cut still influences here the note length, relative to arpeggiolength/number of steps.
ArpDown	**02 abyz** a(0-F): Arpeggiolength a+1 Ticks b(0-F): Number of steps y(0-F): Second Note z(0-F): Third Note Note Cut still influences here the note length, relative to arpeggiolength/number of steps.

Legato Mode	**03 xxxx** xxxx(0-FFFE): Note Overlap Useful for VSTi's with glide/portamento mode. If this Track Command is set, the previous note glides to the current.
Chord	**04 wxyz** w(0-F): Second chord note(unused if 0). x(0-F): Third chord note(unused if 0). y(0-F): Fourth chord note(unused if 0). z(0-F): Fifth chord note(unused if 0). Quite useful for some lazy people.
Slide Up/Down	**05 xxyy aaaa bbbb** xx(0-7F): slide start point. yy(0-7F): slide end point. aaaa(0-30FF): the param you want to slide bbbb(0-7FFF): slide speed b/256 tick. Slides the selected track parameter.
LFO Trigger	**06 xxyy** xx(0-FF): LFO Rate yy(0-FF): LFO Depth Starts the LFO,resets LFO.
LFO(free run)	**07 xxyy** xx(0-FF): LFO Rate yy(0-FF): LFO Depth Starts the LFO without resetting.
LFO Settings	**08 0xyz** x(0-f): LFO Phase/Offset y(0-1): LFO Mode 0: Hz 1: Tick z(0-5): LFO Shape 0: Sine 1: Square 2: Saw Up 3: Saw Dowm 4: Triangle 5: LFO Shape: Random Track-dependant LFO Settings.
MIDI Message	**09 xxyy** xx(0-FF): MIDI Message # 00-7F: CC 0-7F 80-FD: user-defined MIDI Message FE: Pitch Bend Range FF: Pitch Bend yy(0-FF): Value The MIDI Messages can be edited: ->Default Valus->Midi Messages.

Mute Track	**0A 000x** x(0-1): mute/unmute track Mutes/unmutes the track, Mute track will only disable the note playback on the track, the parameter automation is not influenced by this track command.
Inertia	**0B xxxx** xxxx(0-FFFE): Track-dependant Inertia, x/256 Ticks.
All Notes Off	**0C 000x** x(0-1): Sends a note off for all active pattern notes.
Program Change	**0D 00yy** yy(0-7F): Program Change
Bank Select	**0E xxyy** xx(0-7F): MSB yy(0-7F): LSB
Ornament	**0F x0yz** x(0-1): Mode selection y(0-F): Length tick / (2 << y) z(0-F): Relative note
Automation Delay	**10 xxxx** xxxx(0-FFFE): Sets the automation delay position in a tick, if set to 0 the parameter automation syncs to the note delay. Note that the parameter inertia does not work if you use this command.
Psycle Note Command	**11 xxyy** Special note command for psycle plugins only: xx(0-FF): psycle command yy(0-FF): psycle argument
Global Commands	**Cx yyyy** Cx(0-9): Global command selection. yyyy(0-FFFE): See above how to use the respective global command. Useful if you want to use more than one global command in a row.

Shortcuts

There are some shortcuts defined.

Shortcuts	
`	Switch between the buzz mainwindow and the previously active window.
CTRL+SHIFT	Center loader window.
CTRL+D	Stores the current program to favorites.
Left/Right	Next/previous program in bank.
CTRL+Left/Right	Next/previous program via midi program change.
CTRL+Up/Down	Midi bank select.
/,*	Midi Transpose: Octave up/down.
+,-	Midi Transpose: Semitone up/down.
CTRL+LDBLCLICK	On machine: Opens GUI view.
SHIFT+LDBLCLICK	On machine: Opens settings view.
CTRL+SHIFT+LDBLCLICK	On machine: Opens parameter view.
ALT+LDBLCLICK	On machine: Opens buzz parameters.
LDBLCLICK	Minimizes loader window.
RDBLCLICK	Parameter list: Fits parameter list to screen.
CTRL+SPACE	Closes all loader windows except the active one.
ESCAPE	Closes active loader window.

SHIFT	Parameter list: enables the parameter fine adjustment.
TAB	Windows styled cycling through all opened loader instances.
CTRL+TAB	Cycles through the views of a loader window.
CTRL+Return	Activates/Deactivates the overwrite mode.
F2	GUI view.
F3	Paramlist view.
F4	Settings view.
CTRL+F8	Bypass On/Off.
F9	Enables/disables MIDI CC receipt.
F10	Enables/disables MIDI note receipt.
F11	Show/Hide all loader windows.
F12	All Notes Off(MIDI Reset).

Loader Window and Menus

If you have successfully loaded a plugin you can open the loader window. Three views are available: GUI view, the paramlist view and the settings view. There are several menus:

Machine Menu	
Open...	Loads a plugin via open dialog.
Plugins>	All vst plugins which could be found in the selected directories.
Editor...	Opens the loader window.
MIDI Send...	There are a couple of plugins which are midi only, which means they are producing midi events. These events can be sent to other machines to control them with these events (Trollo, QuickKeys, MoCon...-plugins).
Preferences...	Default settings for attributes and a couple of other things. Here you can configure the loader startup values.
About...	Blabla...

Presets Menu	
Load...	Loads a complete bank or a single program. You can import presets from Cubase, Logic as well as Orion.
Save...	Saves a complete bank or a single program. You can save the presets as FXP or FXB.
Rename...	Renames the active program.
Copy	Makes a copy of the active program.
Paste	Pastes a previously copied program to the active program.
Preset Factory...	You're too lazy, stupid or not just in the mood to create a complex preset yourself? You don't need to - let the Preset Factory handle that for you!
Show Preset Numbers	Bank preset numbering on/off.
Prefer Builtin/Favorites	Defines what is displayed in the program menu by default. If set to 'builtin' you'll see the default bank of a plugin, otherwise the programs of your favorites(Organize...).
Organize...	Opens a preset manager where you can manage your favorite programs.
Store...	Stores the selected program to the favorites (Shortcut: CTRL+D).
Programs	All programs of the bank or favorites are listed here, depending on what is set as default. A couple of plugins will have its own program management, therefore these programs cannot be displayed here.

Learn Menu	
Auto Learn	If enabled, all touched controls in the vst gui get assigned to buzz sliders - track by track, one after another (until the last available track slider has been reached).
Learn MIDI CC	Necessary to make the loader recognize vst plugins which use the Midi CC method for controller automation.

Clear Params	Wipes all learned parameters.
Save as Default	Writes an .dat containing all learned parameters of a plugin. These settings will be automatically used if you load the plugin.
Restore Default	Reverts all changes to the saved defaults.
Learned Parameter List	Shows the list of all available tracks and their learned VST parameters.

View Menu	
GUI	Shows the VST plugins' gui (default view).
Parameter List	A schematic listing of all available VST plugin parameters.
Settings	All loader settings (as found in the machine attributes) can be found here as well besides a number of track dependant things (track selector, LFO...).
Plugin Information	Opens dialog containing some informations about a plugin.
MIDI: Receive Note	Enables/disables the MIDI note receipt.
MIDI: Receive CC	Enables/disables the MIDI CC receipt.
Bypass	Bypass on/off.
All Notes Off	Sends a "all notes off" to plugin.

GUI View

The default view once a plugin is loaded into the Polac machines. A double-right-click on a VST or VSTi instance will open it.

Parameter View

A list view showing all parameters of the loaded VST plugin. All parameters can be edited here as well. Use SHIFT+Leftmouse for fine tuning values. To the left of each paraemeter you'll find a button. You can select/deselect here a parameter for randomizing or morphing purpose(preset factory).

Settings View

Surely the most important section of the loader besides the vst gui. There are a couple of sections here:

Track Settings:

- Track Selector: Here you can select a track for record, mute a track.
  -Leftclick: (de)selects track for recording
  -DoubleLeftclick: (de)selects all tracks
  -CTRL+Leftclick: (un)mutes track.
  -CTRL+DoubleLeftclick: (un)mutes all tracks
  -Rightclick: selects active track(track settings)
- Overwrite: When enabled, the pattern data will be overwritten during recording. You can optionally keep the automation or non-automation data(rightclick the overwrite button). Overwrite can be optionally toggled via CC64, quite useful if using a foot switch.
- Inertia: Used in order to smooth parameter automation changes. 0.00 ticks might occur steppy (depending on the plugin), above 1 isn't very responsible for live purposes but can be used for making long parameter movements easier. A good, responsive default is 0.50 ticks.
- LFO Shape: Available waveforms as LFO Shape:
  -Sine
  -Square
  -Saw Up
  -Saw Down
  -Triangle
  -Random
- LFO Mode: The track LFO's can operate with a HZ or tick related measure.
- LFO Phase: Sets the starting point for each LFO.
- Assigned Parameter/CC: Shows the list of all available vst plugin parameters. Can be used to assign vst parameter to a track parameter by simply selecting it. The value slider can be directly linked to a midi cc as well as its value range can be customized. A right-mouse click on it opens the regarding menu.
- Midi Channel: Specifies the Midi Channel that is used for each track (!**ATTENTION**! Not to be confused with the Global Midi Channels!).

MIDI Settings:

- In Channel: MIDI Channel that is beeing used by Loader instance to listen for incoming MIDI events.
- Out Channel: MIDI Channel that is used by Loader to sent MIDI events.
- Transpose: Transposition of incoming notes by value(note).

- Velocity Slope: Velocity sensitivity.
- Velocity Min: Sets the minimum velocity.
- Velocity Max: Sets the maximum velocity.
- Quantize: Determines the quantisation raster of incoming midi events for recording and ONLY for recording. Changes made after recording don't have an effect.
- Quantize NoteOff: If this switch is enabled, NoteOff events are quantized as well.
- Record NoteOff: Enables/disables note off recording.
- MIDI Mode: Mode A can be referred as "standard" mode. Incoming MIDI events will be used by all VST Loader instances which have Midi enabled. Mode B will only enable Midi for the highlighted VST Loader instance.

Global Settings:

- Dry Out(PVST only)
- Wet Out
- Panning
- Transpose: Global transposition of all notes for playback.
- Tempo Sync: If Buzz is enabled, Buzz' master tempo will be used for the VST loader tempo syncronisation. In case you're working with tempo changes (e.g. shuffle using master tempo changes) it's quite handy to lock the loaders' tempo manually.
- Link Dry/Wet(PVST only): If enabled, dry out and wet out can be controlled with only one slider.

Miscellaneous Settings:

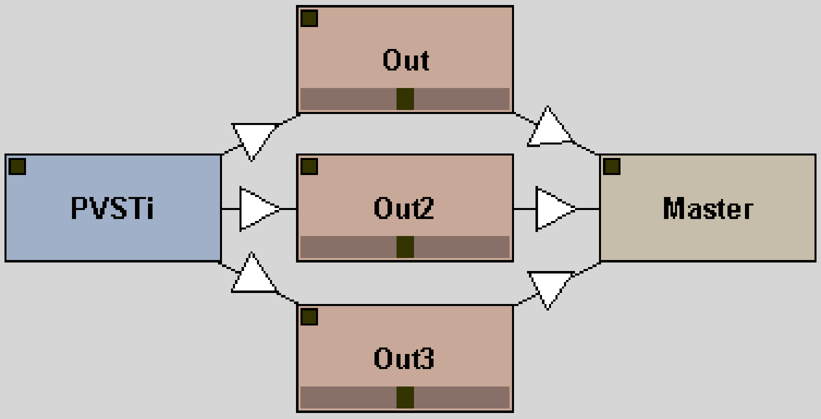
- Automation Mode:
  -Normal: if activated, the loader records automation data as long as you move the fader/knob.
  -Touchfader: if activated, the loader records automation data as long as the fader/knob is held, even if you stop the movement.
- Virtual Keyboard: If enabled, the pc keyboard can be used to send midi notes to the vst plugin.
- MIDI Mode: Mode "User-Defined" can be referred as standard mode. Incoming MIDI events will be used by all VST Loader instances which have Midi enabled. Mode "Active Editor" will only enable Midi for the highlighted VST Loader instance. Mode "All Editors" will enable Midi for all vst's with opened editors.
- Refresh Button: use this to update the settings view.
- Morph: Here you can select the programs for morph purpose. If some programs are selected you can select "Morph Prgs" in the "Track Parameter" combo box. The slider movement can be recorded to track parameter colums now.

How To Use...

Some hints:

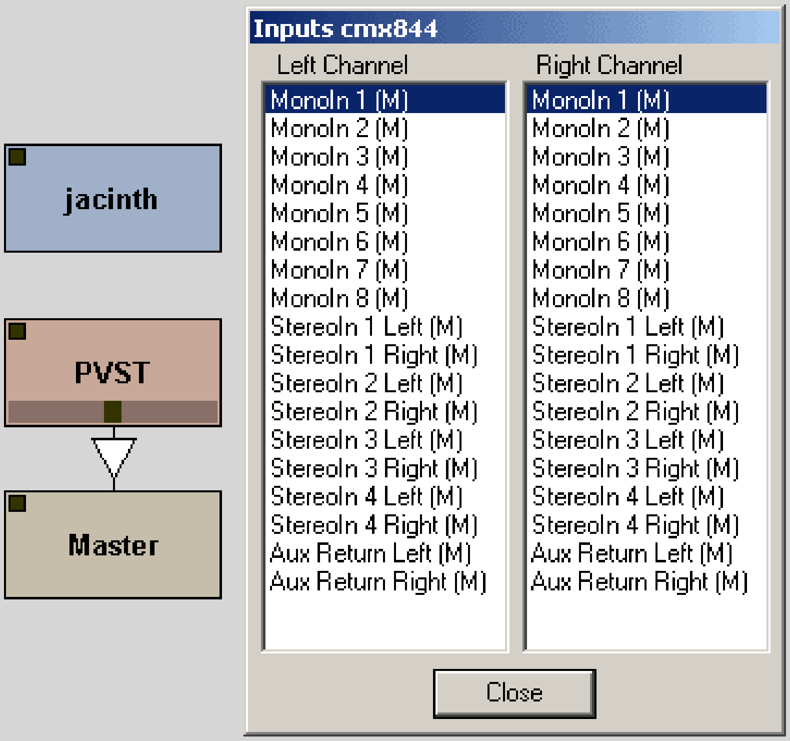
Multi-out VST's

If a VST has multiple outputs, you can use the Polac Out II to access these outputs. The picture below shows how you have to connect the machines. Polac Out II will detect the connected machine. Doubleclicking this machine will open a dialog showing up all available outputs of the connected VST.



Multi-in VST's

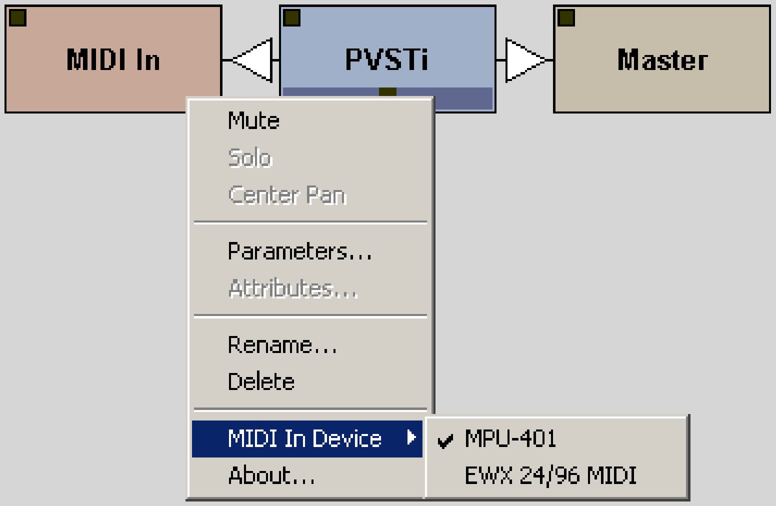
If a VST has more than two inputs, a dialog will automatically pop up when a new machine is connected to the loader. All available inputs are listed in this dialog, you only have to choose the appropiate inputs.



Left Channel	Right Channel
MonoIn 1 (M)	MonoIn 1 (M)
MonoIn 2 (M)	MonoIn 2 (M)
MonoIn 3 (M)	MonoIn 3 (M)
MonoIn 4 (M)	MonoIn 4 (M)
MonoIn 5 (M)	MonoIn 5 (M)
MonoIn 6 (M)	MonoIn 6 (M)
MonoIn 7 (M)	MonoIn 7 (M)
MonoIn 8 (M)	MonoIn 8 (M)
StereoIn 1 Left (M)	StereoIn 1 Left (M)
StereoIn 1 Right (M)	StereoIn 1 Right (M)
StereoIn 2 Left (M)	StereoIn 2 Left (M)
StereoIn 2 Right (M)	StereoIn 2 Right (M)
StereoIn 3 Left (M)	StereoIn 3 Left (M)
StereoIn 3 Right (M)	StereoIn 3 Right (M)
StereoIn 4 Left (M)	StereoIn 4 Left (M)
StereoIn 4 Right (M)	StereoIn 4 Right (M)
Aux Return Left (M)	Aux Return Left (M)
Aux Return Right (M)	Aux Return Right (M)

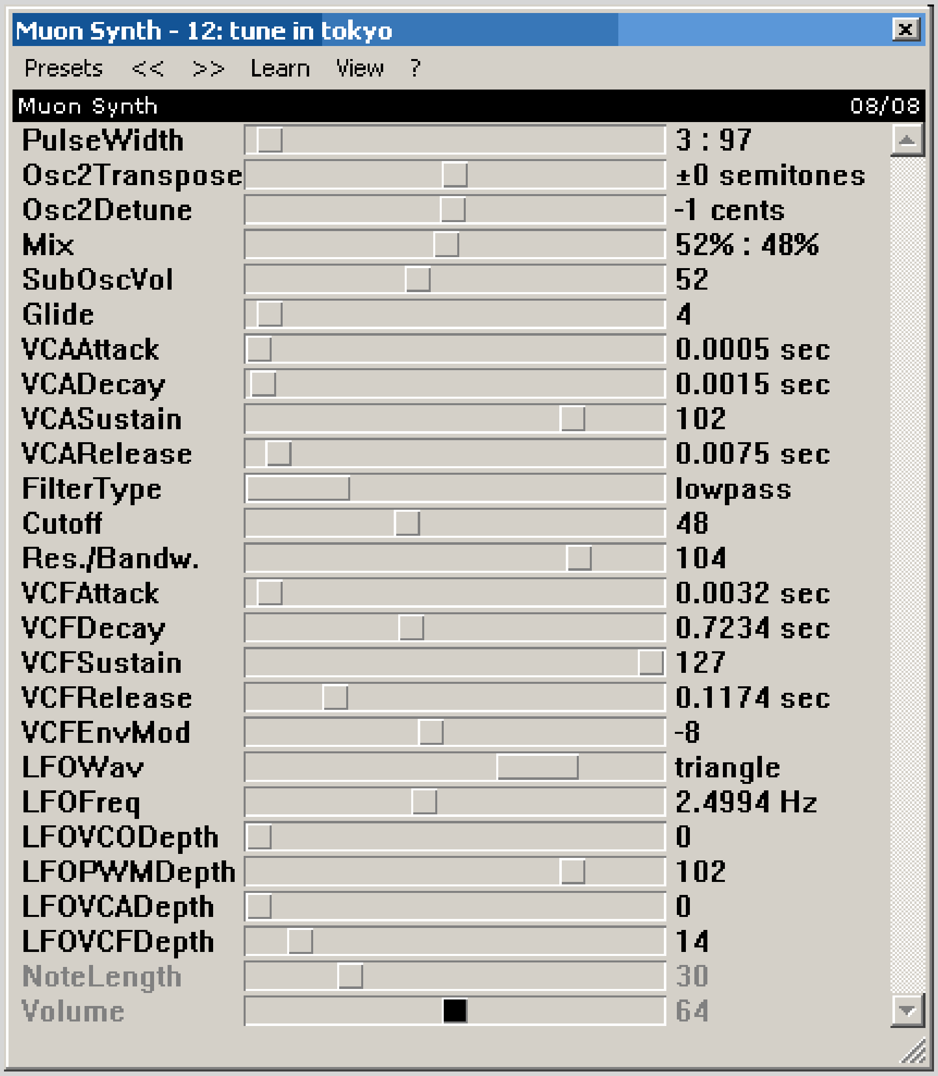
Additional MIDI In Devices

You can use the Polac MIDI In machine to make other machines able to receive midi data also from other midi devices than the buzz default midi device. See below how to use this machine:



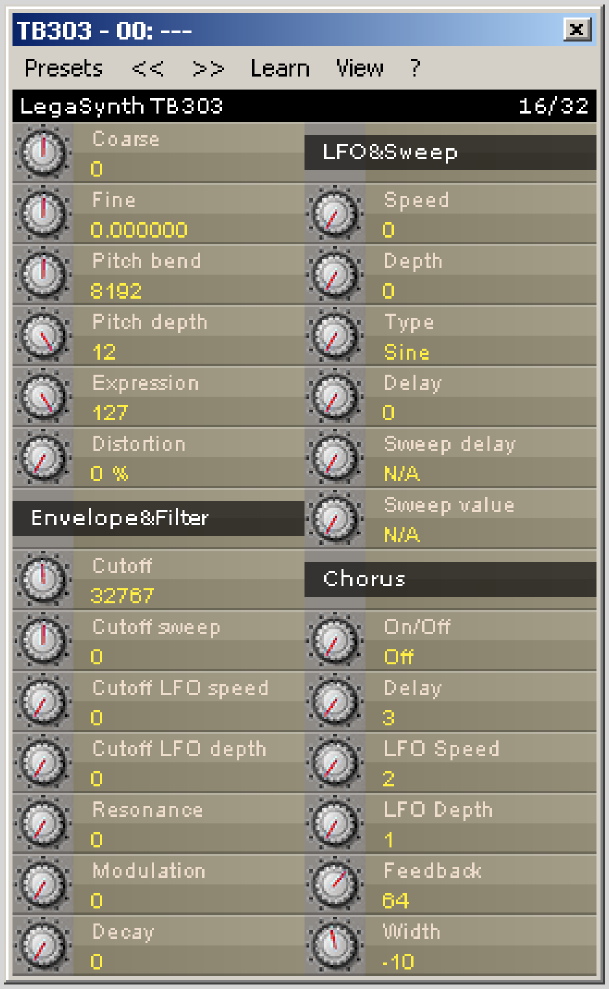
Buzz/Psycle wrappers

To use them inside the loader simply place some buzz or psycle machines in one of your vst directories. Alternatively you can create a shortcut of a machine via explorer and move the shortcut to a vst directory. The machines will be detected by the loader and can be selected via pluign menu later on.


Muon Synth - 12: tune in tokyo

Muon Synth 08/08

PulseWidth	3 : 97
Osc2Transpose	±0 semitones
Osc2Detune	-1 cents
Mix	52% : 48%
SubOscVol	52
Glide	4
VCAAttack	0.0005 sec
VCADecay	0.0015 sec
VCASustain	102
VCARelease	0.0075 sec
FilterType	lowpass
Cutoff	48
Res./Bandw.	104
VCFAttack	0.0032 sec
VCFDecay	0.7234 sec
VCFSustain	127
VCFRelease	0.1174 sec
VCFEnvMod	-8
LFOWav	triangle
LFOFreq	2.4994 Hz
LFOVCODepth	0
LFOPWMDepth	102
LFOVCADepth	0
LFOVCFDepth	14
NoteLength	30
Volume	64

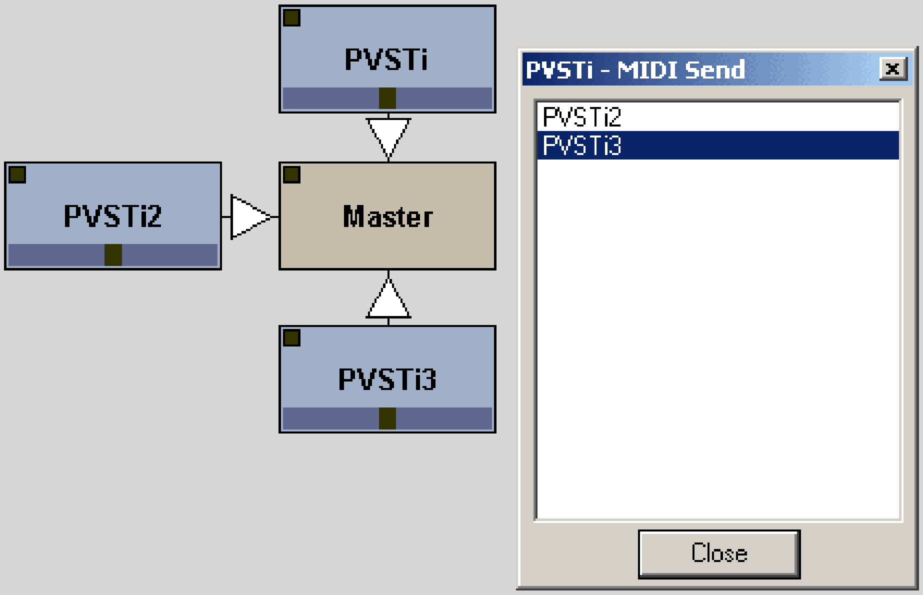

TB303 - 00: ---

LegaSynth TB303 16/32

Coarse	0
Fine	0.000000
Pitch bend	8192
Pitch depth	12
Expression	127
Distortion	0 %
Envelope&Filter	
Cutoff	32767
Cutoff sweep	0
Cutoff LFO speed	0
Cutoff LFO depth	0
Resonance	0
Modulation	0
Decay	0
LFO&Sweep	
Speed	0
Depth	0
Type	Sine
Delay	0
Sweep delay	N/A
Sweep value	N/A
Chorus	
On/Off	Off
Delay	3
LFO Speed	2
LFO Depth	1
Feedback	64
Width	-10

MIDI-generating VST's (Midi Send...)

In the loader rightclick menu is an entry called "MIDI Send". A couple of plugins are generating midi. These midi events can be directly sent to other loader instances to control them with these midi events. Only if a plugin is a "midi generator", the midi send dialog will shop up other loader instances where to send the events.



Parameter Automation

Up to 128 VST parameters can be controlled simultanously, depending on the number of tracks. VST parameters/MIDI CCs can be assigned in this way:

1. Open the GUI/Params window.
2. (a) Open the Learn menu and select a track you want to assign with a VST parameter/MIDI CC. If you want to learn a MIDI CC also enable "Learn MIDI CC".
   (b) Open the Learn menu and select Auto Learn. If you want to learn a MIDI CC also enable "Learn MIDI CC".
3. Now you only have to move a Fader/Knob/Button, in most cases the parameter is learned now. Alternatively you can move a Controller/Pitchbend of your masterkeyboard or faderbox to learn them.

Several Notes:

- there are a couple of plugins that cannot be assigned to a track in this way. So you have assign the VST parameters manually(=>Parameters...).
- you can unbind a parameter by rightglicking on a learned parameter in the parameter list.

Midi Message Editing

Midi messages can be used via track command 07. Predefined midi messages are:
00-7F: MIDI CC
FE: Pitch Bend Range
FF: Pitch Bend

Messages [80] to [FD] are undefined. To edit the midi messages, right-click the loader, open the Default Values... dialog, click the midi message button. This will open the midi message.ini in your texteditor. A short example how to edit the midi message.ini:

```
All is numbered from [00]-[FF], here we are editing midi message [FE].

[FE] Bn 64 00,Bn 65 00,Bn 06 xx //Pitch Bend Range

n: A byte with n will be combined with the track midichannel
xx: the variable value
//: Comment.

Each byte has to be separated from the other bytes with either a space, tab or comma.
Three bytes will give one midi event.
```

Preset Factory

You're too lazy, stupid or not just in the mood to create a complex preset yourself? You don't need to - let the Preset Factory handle that for you!

The Preset Factory allows you to create new sounds in 3 ways:

1. Morphing:
   The morph slider will, generally spoken, take preset A, B and linearly blend between both. If there wasn't the Mode setting above. "ALL" will really just do the blend-job, but then it get's interesting again. "Selected" will only touch the parameters which are selected in the parameter list (F3). "Not selected" does the same with all non-selected parameters. "Learned" only affects the vst parameters that have been assigned to buzz parameters. "Not learned" vice versa.

2. Genetics ("Breed"):
   The genetics slider determines the amount of vst parameters that will be taken from both presets. It doesn't change the preset unless you press the "Breed!" button. Once it's pressed, it takes the specified amount of (random!) vst parameters to create a new sound. Of course, you can point out which parameters may be used for this procedure using the Mode as in the morphing.

   To make it easier for you:
   E.g. your plugin has got 10 vst parameters and you don't move the genetics slider, 5 parameters from both selected presets will remain in the new preset. Which exact parameters are used is Breed's choice (random).

3. Randomize:
   Good old-fashioned patch randomizing. You can specify the amount of randomization by enabling the % range. The Mode is enabled for usage with randomizing as stated above.

Preferences

Here you can configure the loader preferences, which means that the loader will use these settings each time you start a new loader instance. Also the vst directories can be configured here. The preferences dialog can be accessed via the machine menu(rightclick on loader). There are three tabs, two for global settings which will have effect for all plugins and one for settings done per plugin:

Global I:

Here one can set the default values for the loader attributes, also some other things can be defined here such as inertia, the number of tracks, midi or automation mode etc...

Global II:

Select your vst directories and some settings related to the dx/buzz loader, also an option to select the PVST data folder, 'My Documents' is set by default.

Plugin:

Some plugin-specific settings:

- VST Keys: If enabled the loader sends keystrokes to the plugin.
- Fixed Length Processing: The loader processes samples with a constant blocksize, note that this will delay the output for 256 samples. Only a couple of plugins will need this enabled (Spinaudio/PPG).
- Save Bank To Song: All Programs of a VST(i) are saved to the buzzsong.
- Save Prg To Song: Only the selected program of a VST(i) is saved to the buzzsong. A couple of plugins will cause problems on bank saving(Delay Lama,FFX4), so this has to be enabled.
- Use sizeWindow(): Disable this if a gui does not resize correctly.
- Stereo Fix: some plugins will remain mono if this one is not enabled(Cyanide).
- Disable shortcuts/virtual keyboard. Disable if a plugin has its own keyboard handling.

Links

More infos about these loaders can be found here:

PVST FAQ
buzzmachines.com message board posts about PVST

Some buzz-related sites:

buzzmachines.com - latest infos on effects/generators
buzzchurch.com - a new buzz-related forum
the new buzz beta - oskari's latest buzz binaries
buzzmusic - site hosting buzz-generated music
buzzformat.com - buzz overloader
cyanwerks.com - buzzdev archive here
noolmusic - lots of infos about buzz here
buzé - remake of buzz, open source
aldrin - linux/windows buzzclone, open source

Buzz Devs:

Btdsys
Fuzzpilz
Id0d

Various links:

kvraudio.com
rjkole.com
ronnypries.de

Thanks To...

Thomas Potulski - for programming help and tips

Ronny Pries - for lots of ideas and help, help on this manual, beta testing, for being sort of project manager ;)

oskari - for buzz and newbuzz help

CyanPhase - for the great buzz tutorials

usr - for suggestions and help with the machines.dll

Cameron Bonde - for tons of feature suggestions and beta testing

CyanPhase - for the great buzz tutorials

ld0d - for some help on the vst menus in the machine menu

djlaser - for tips and suggestions

P. DooM - for the buzzhack sourcecode

Also big thanks to all on the beta list:

.tOm, ps, Wayfinder, soma, WakaX, mute, btd, Rolf Kohl, Mva, Boneyhands and the rest I forgot here.

Legal Notice

Polac VST(i) Loader for Jeskola Buzz

Version: 1.0 *BETA*
Author: Frank Potulski
Contact: **polac@gmx.de**
Website: **http://www.xlutop.com/buzz/**

About VST(i) Loader

VSTi Loader and VST Loader are machines for Jeskola Buzz which make it possible to run VST Effects and Instruments in Buzz.

Installation

Just copy the "Polac VSTi.dll" to the Buzz/Gear/Generators/ directory and the "Polac VST.dll" to the Buzz/Gear/Effects/ directory. It is recommended to install the latest Buzz version as well as the latest Overloader version.

Parameters and Attributes

A short description of all parameters and attributes. Many of these parameters and attributes can also be accessed over the popup menu "Settings"(just open the GUI or Params window).

Global Parameters	
Dry Out	If you are using the VST Loader, this parameter controls the dry output volume.
Wet Out	This parameter controls the wet output volume.
Panning	Sets the panning position of the Wet Out signal.
Program	Selects a program of a VST(i).

Track Parameters	
Note	Switch to pattern mode and insert some notes. :) **Important for those who want to use plugins with an internal sequencer (Reaktor, Scuzzphut5 etc.):** You can trigger a sequence with a B-9. A corresponding note off will stop the sequence.
Velocity	Sets the velocity of a note.
Delay	Position of the note in a row/tick.
Note Cut	Sets a note off in a row. A note must be triggered too in the same row to hear a result.
Parameter Name	Sets the VST parameter you want to control. Also Pitchbend and MIDI CC can be selected.
Parameter Value	Sets the value of the controlled parameter.
Parameter Inertia	Parameter movement smoothing.

Attributes	
MIDI In Channel	Sets the receiving MIDI Channel(0=all).
MIDI Out Channel	Specifies transmitting MIDI Channel.
MIDI Velocity	Settings for note on velocity(0=ON, 1-127=velocity fixed to value).
MIDI Transpose	Transposes the incoming MIDI notes.
MIDI Enable	Enables/disables MIDI.

Quantisation	sort of tick quantisation. While recording, it rounds the delay of a note to a certain value: 0=OFF 1=1 Tick 2=1/2 Tick 3=1/3 Tick 4=1/4 Tick 5=1/8 Tick . . 8=1/64 Tick
Quantisation Note Off	Enables/disables the note off quantisation. Quantisation must be set before, otherwise enabling note off quantisation will affect nothing.
Hotkeys	Enables/disables hotkeys.

Open a VST(i) Plugin

Rightclicking the VST(i) Loader symbol in the machine view of Buzz will open a popup:

Open...	Loads a plugin via an open dialog.
Plugins>	A list all VST(i) plugins available.
GUI...	Opens the built-in editor of a VST(i).
Params...	The parameter list editor.
Add Pluginpath...	Opens a dialog where you can select your VST(i) plugin directories.

GUI/Params Window

If you have successfully loaded a plugin you can open either the GUI or Params window of the plugin. Some plugins will have no own GUI, so a left-click on GUI will open the Params window. There are several menus:

Presets Menu	
Load...	Loads a complete bank or a single program. You can import presets from Cubase, Logic as well as Orion.
Save...	Saves a complete bank or a single program. You can save the presets as FXP or FXB.
Rename...	Renames the active program.
Copy	Makes a copy of the active program.
Paste	Pastes a copied program to the active program.
Programs	All programs of a bank are listed here. A couple of plugins will have its own program management, therefore these programs cannot be displayed here.

Settings Menu	
Output Settings...	Settings for Dry Out, Wet Out and Panning.
MIDI In Channel>	Sets the receiving MIDI Channel(0=all).
MIDI Out Channel>	Specifies transmitting MIDI Channel.
MIDI Velocity...	Turn on/off the MIDI velocity. If MIDI velocity is set to off, the velocity is fixed to a certain value.
MIDI Transpose...	Transposes the incoming midi notes.
MIDI Record Settings...	Sets record quantisation. You can also choose your record tracks.
MIDI Enabled	Enables/disables MIDI.
MIDI Reset	Sends a MIDI reset.
Learn MIDI CC	Must be selected if you want to learn a MIDI controller.

Learn Param 1-8	You may check one of these menu items if you want to learn a VST parameter or MIDI CC.
Hotkeys Enabled	Enables/disables hotkeys.

Learning Parameters

Up to eight VST parameters can be controlled simultanously. There is the possibility to chain a VST parameter/MIDI CC automatically to one of the eight parameter sliders:

1. Open the GUI/Params window.
2. Open the Settings menu and select a parameter(Learn Param 1-8) you want to chain with a VST parameter/MIDI CC. If you want to learn a MIDI CC also enable "Learn MIDI CC".
3. Now you only have to move a Fader/Knob/Button, in most cases the parameter is learned now. Alternatively you can move a Controller/Pitchbend of your masterkeyboard or faderbox to learn the MIDI Controller belonging to it.

Notes:
There are a couple of plugins that cannot be chained in this way. So you have assign the VST parameters manually(=>Parameters...).

Hotkeys

There are some hotkeys defined, which can be disabled if you feel annoyed by them. Make sure that either the GUI or Params window has the focus.

Hotkeys	
F1	Enables/disables MIDI.
ESC	MIDI Reset.
NUM0	Learn MIDI Controller.
NUM1-8	Learn parameter 1-8.
NUM9	Clear all learned parameters.
LEFT	Previous program.
RIGHT	Next program.
SPACE	Minimizes the GUI and Params window, so that only caption and menu are visible. Doubleclicking the caption area of each window will affect the same.

Known Plugin Problems

- Tobybear VSTChess refuses to work
- some versions Jørgen Aase's Audiosynth will cause problems
- Rumpelrausch Täips Crazy Diamonds occasionally explodes(but not only with this host)
- with xverb it's quite the same thing

If you find another plugin which isn't working properly, please let me know. Perhaps I'm able to fix this problem(**polac@gmx.de**).

Credits

Big thanks to Thomas Potulski for his programming tips and help. Without his support this project hadn't been possible. Also thanks to Cyanphase for his Buzz programming tutorials. :)

Legal Notice

pooplog's swims synth

pooplog666@hotmail.com

type: synths

The SWIMS is a unique semi-complex synth, but it is also super fast. There are a million of them so that you can select the one with only the options that you want-to keep it super fast. I never intended these for public use, which explains the cryptic interface. I originally designed all of my machines to use less than 3% cpu on a p233. Buzz crashes constantly on my athlon (I can't even get the audio card config menu to come up), so I probably will stop developing machines for Buzz and move on to another better supported platform.

first: what are these funny names?

for example
8-swims-de is a:
8 stage width modulation synth - dual - envelope

the **DUAL** means it has a carrier signal as well
and the **ENVELOPE** means that it has an envelope control

Let me explain all the parameters, perhaps that might help you understand how it works:

ADSR: This is your envelope. It is designated by the -E in the synth name.
ATTACK
DECAY
SUSTAIN
RELEASE

SWIMS SECTION:
Fine Tune - Well, that's kinda obvious as well
Width 1:2 - this is where it gets interesting. This is the width ratio of the waveform. you've heard of pulse modulation, well, this lets you modulate the width
of any type of waveform. It is the ratio between the first and second
half-cycle of the wave.
WFS 1 - this is the wave form shape of the first half cycle. the other neat thing about swims is that you get to pick a waveform for each half-cycle. generally you would
like to pick a positive wave shape.
WFS 2 - same as above but the second half cycle. generally you should pick a negative. these parameters repeat depending on the number of desired stages in your total waveform.
DUAL Section: Carrier waveform. It is designated by the -D in the synth name. This is

another waveform that gets mixed with your waveform that is generated in
the SWIMS section. Why? Sometimes you want to add a sine that is an octave lower
to add fatness, for example.
C transpose - the transpose of the carrier in tones relative to the swims synth
C Fine tune - kinda obvious
Carrier A:B - this works the same as Width 1:2 but just for the carrier
CS A - same as WFS 1 but just for the carrier
CS B - same as WFS 2 but just for the carrier
Mix WF:C - this is the mix between the SWIMS wavefore and the Carrier waveform

GOOD LUCK!
pooplog

pooplog's pwm synth

pooplog666@hotmail.com

type: synth

The fast-pwm is a simple square wave synth with variable pulse width, but it is also super fast. I never intended these for public use, which explains the cryptic interface. I originally designed all of my machines to use less than 3% cpu on a p233. Buzz crashes constantly on my athlon (I can't even get the audio card config menu to come up), so I probably will stop developing machines for Buzz and move on to another better supported platform.

first: what are these funny names?

for example
fast-pwm-e is a:
fast pulse width modulation synth - envelope

and the **ENVELOPE** means that it has an envelope control

Let me explain all the parameters, perhaps that might help you understand how it works:

ADSR: This is your envelope. It is designated by the -E in the synth name.
ATTACK
DECAY
SUSTAIN
RELEASE

PWM Section:
Fine Tune - Well, that's kinda obvious as well
Pulse WIdth - This is the width ratio of the waveform. (Known as Wave Duty to some folk...)

GOOD LUCK!
pooplog

PSI Corp's DrumAnd(b)Ass

Release 1 (1999-03-31)

by: Per-Olov Jernberg aka Illuminator/PSI::Corp

Purpose:

my friend (and our musician) H-Esc thought it would be nice with a drummachine with some nice drum&bass sounds on, so ... here it is..
the plugin has been made forward compatible, so new effects and perhaps new samples will be available in newer versions.

Features:

- 30 Different samples 7 Basses, 13 Hats and 10 Snares
- Effects like DelayNote, ReTrig, ReTrig Twice.

History:

- Release 1 - initial release, 7basses, 11hats, 10snares, 3effects

Effects:

example patterns:

```
.. 1 .. .. ....    <-- trig sample as usual. (once)

.. 1 .. 01 0050    <-- wait 0x50 ms then trigger.

.. 1 .. 02 0060    <-- trigger and then after 0x60 milliseconds, trigger again.

.. 1 .. 03 0060    <-- trigger and then after 0x60 milliseconds, trigger again. and again after another 0x60
milliseconds.
```

Parameters are: Sample, Trigger, Volume, Effect, Effect parameter (ms)

Samples:

a list of samplenames, composed by H-Ecs.

```
1. Wet deep Basedrum
2. Donk Basedrum
3. Base-Basedrum
4. Long Deep Basedrum
5. Industria Flash-Basedrum/Snare
6. Short Dry Basedrum
7. Normal "909"-basedrum

8. Hat with a small kick
9. Side of Snare
10. Side of Snare2
11. Ride Cymbal
12. hihi-hat
13. Industrial "ztuu.."
14. Disted hat with a kick
15. Industrial Side of Snare
16. Disted Clap
17. Tamburine
18. Closed "909"-hat
19. Closed Industrial hat
20. Normal Cymbal

21. Short snare
22. "Bong"
23. Strange Clap mixed with a closed hat
24. Strange Industrial Snare
```

```
25. Snare mixed with a Cymbal
26. Snare mixed with a Kick
27. Short Snare with a Little hat
28. Industrial supercool long asskicking snare
29. Normal "909"-snare
30. Short Snare/hat
```

---

as usual, if you find any bugs - mail the author :)

PSI Corp's WaveAss

Release 1 (1999-03-25)

by: Per-Olov Jernberg aka Illuminator/PSI::Corp

Purpose:

To provide a simple synth generator wich can produce complex waveforms

Features:

- Three customizable waves (can be drawn)
- Morphing (between the three waves)

History:

- Release 1 - initial release

Morphing:

when in the range 0-64 (00-40): morphing between wave 1 and wave 2 (a value of 32 is 50% wave 1 and 50% wave2)
when in the range 64-128 (40-80): wave 2 and wave 3 is used for morphing
and when in the range of 128-192 (80-C0): wave 3 and wave 1 is used for morphing

---

as usual, if you find any bugs - mail the author :)

PSI Drum 2

aka PSI Corp's Drum&Ass 2
programmed by: Illuminator aka Per-Olov Jernberg

E F F E C T S

```
T = time in ticks/1000 (milliticks, eg. 0001 = 0.001 tick)
% = percent (eg 0=0%, 80=100%, FE=~254%)
F = fade speed (1/32 per sample)
```

# Nr	Name/Description	Effect Parameters	Parameter Description
01	Delay Trig	TTTT	T = Time until triggering
02	Re(peated)Trig	NTTT	N = Number of times to retrig T = Time before next triggering
03	Temporary Volume Shift	%%%%	% = Amount of original volume
04	Volume Up (fade)	FFFF	F = Fade up speed
05	Volume Down (fade)	FFFF	F = Fade down speed
06	Temporary Pitch Shift	%%%%	% = Amount of original pitch
07	Pitch Up (fade)	FFFF	F = Fade up speed
08	Pitch Down (fade)	FFFF	F = Fade down speed
09	Backward / Reverse	none	
0A	Stop NOW	none	Stops sample playback on channel

0B	Stop After...	TTTT	T = Time until samples are stopped
0C	Sample Offset	PPPP	P = Sample to jump to (divided by 8)
0D	Nudge Sample	DDDD	D = Samples to nudge, 0000 = -100%, 4000 = 0%, 8000 = +100%
0E	Reverse Play Direction	none	Flips playing direction,
10	Rush	NNTT	N = Number of ticks to rush T = TriggerCount*16 per Tick
21	"Dahlström 2"	TTTT	T = BackStep value *Samples are played as usual, after T ticks* *it goes back T/2 ticks and repeats...* *(weird effect)*

D R U M K I T S

the drumkit format is pretty easy, it is simply a header telling us how many drums
there are and each drum has its own little header telling us wich class it should
go under, and then the wavedata follows in signed short (2bytes) data...
(16Bit, 44100Hz, Mono)

if you are more interested in the drumkit format, if you´re going to write a fancy
rebirth-ripper or something, throw me an email

an AXS drumkit converter will be available on my homepage a couple of weeks after
releasing this.

if you have made your own drumkit wich you´re going to use in a song you´re about
to distribute, simply check the "store drumkit in song"-checkbox and the drumkit will be saved
in your buzz file, then whoever downloaded your song can simply choose "Save Drumkit" in the
menu and *kabam* he/she has the same drumkit :)

the drumkit manager dont check if the .wav file is a real wavefile it just skips the header and
stores the wavedata (wow!) so be sure your wavefiles are uncompressed 16bit mono files.

and please, try not to use too strange characters in the filename or drumkit name!

K N O W N   B U G S

buzz MAY crash if you is loading a sample into a slot that is playing in the wavetable section...
buzz MIGHT get greedy and take all cpu, then it goes down, this bug might be fixed.
the effect parameter setup may suck, but as soon as you learn them they are quite nice...

M I S C E L L A N E O U S

this must be the worst helpfile ever written.
visit me: http://www.demoparty.com/~possan
visit psi: http://www.demoparty.com/~torsten
visit d99: http://www.demoparty.com now!
the drumkit manager is VERY horrible to use... (save often)
be aware of PhatAss - our new synth...

Rebirth MIDI

Alpha Version 0.1
(c) 1998 Chilled Dreams
This Buzz-Plugin is freeware. Dedicated to ya'all Rebirth lovers.

Here is a table of the MIDI-Commands for Rebirth 2.0 (Rebirth standard mapping). I provide this because the original documentation is wrong on various points. Moreover you have to enter hexadecinmal values in Buzz but the Rebirth documentation presents the controller-IDs in decimal values. I think it's easier to read than to convert every value with a calculator.

MIDI Controller messages

Ctrl.	Function
7	Master Volume
0b	Synth 1 Level
0c	Synth 1 Pan
0d	Synth 1 Delay Amount
0e	Synth 2 Level
0f	Synth 2 Pan
10	Synth 2 Delay Amount
11	808 Level
12	808 Pan
13	808 Delay Amount
14	909 Level
15	909 Pan
16	909 Delay Amount
17	Synth 1 Waveform
18	Synth 1 Tune
19	Synth 1 Cuttoff
1a	Synth 1 Reso
1b	Synth 1 Env Mod

1c	Synth 1 Decay
1d	Synth 1 Accent
1e	Synth 2 Waveform
1f	Synth 2 Tune
20	Synth 2 Cuttoff
21	Synth 2 Reso
22	Synth 2 Env Mod
23	Synth 2 Decay
24	Synth 2 Accent
25	808 AC Level
26	808 BD Level
27	808 BD Tone
28	808 BD Decay
29	808 SD Level
2a	808 SD Tone
2b	808 SD Snappy
2c	808 LT Level
2d	808 LT Tune
2e	808 LT Switch on/off
2f	808 MT Level
30	808 MT Tune
31	808 MT Switch on/off
32	808 HT Level
33	808 HT Tune
34	808 HT Switch on/off
35	808 RS Level
36	808 RS Switch on/off
37	808 CP Level
38	808 CP Switch on/off
39	808 CB Level
3a	808 CY Level
3b	808 CY Tone

3c	808 CY Decay
3d	808 OH Level
3e	808 OH Decay
3f	808 CH Level
40	808 Instrument Selection
41	909 AC Level
42	909 BD Level
43	909 BD Tune
44	909 BD Attack
45	909 BD Decay
46	909 SD Level
47	909 SD Tune
48	909 SD Tone
49	909 SD Snappy
4a	909 LT Level
4b	909 LT Tune
4c	909 LT Decay
4d	909 MT Level
4e	909 MT Tune
4f	909 MT Decay
50	909 HT Level
51	909 HT Tune
52	909 HT Decay
53	909 HiHat Level
54	909 RS Level
55	909 CP Level
56	909 CH Decay
57	909 OH Decay
58	909 CC Level
59	909 CC Tune
5a	909 RC Level
5b	909 RC Tune

5c	909 Flam amount
5d	909 Instrument Selection
5e	PCF Pattern
5f	PCF Mode (LP/BP) (trigger at 40h)
60	PCF Freq
61	PCF Q
62	PCF Amount
63	PCF Decay
64	Delay Steps (1-32)
65	Delay Triplet mode (trigger at 40h)
66	Delay Pan
67	Delay Feedback
68	Dist Shape
69	Dist Amount
6a	Compressor Ratio
6b	Compressor Threshold

MIDI Keyboard events

All modes

Note	Function
E-4	Select Pattern/Program Synth
F-4	Focus to Synth1
F#4	Focus to Synth2
G-4	Focus to 808
G#4	Focus to 909
A-4	Transport Play
A#4	Transport Stop
B-4	Transport Record
C-5	Transport Bar -

C#5	Transport Bar +
D-5	PCF enable
D#5	Delay enable
E-5	Dist enable
F-5	Comp enable
F#5	Synth1 Mix on/off
G-5	Synth1 Dist on/off
G#5	Synth1 PCF on/off
A-5	Synth1 Comp on/off
A#5	Synth2 Mix on/off
B-5	Synth2 Dist on/off
C-6	Synth2 PCF on/off
C#6	Synth2 Comp on/off
D-6	808 Mix on/off
D#6	808 Dist on/off
E-6	808 PCF on/off
F-6	808 Comp on/off
F#6	909 Mix on/off
G-6	909 Dist on/off
G#6	909 PCF on/off
A-6	909 Comp on/off
A#6	Master Comp on/off (doesn't work)
B-6	Programm Synths from Keyboard
C-7	Select Patterns from Keyboard

Select patterns from keyboard mode

Note	Function
C-0	Synth1 - Pattern on/off
C#0	Synth1 - Bank A
D-0	Synth1 - Bank B

D#0	Synth1 - Bank C
E-0	Synth1 - Bank D
F-0	Synth1 - Pattern 1
F#0	Synth1 - Pattern 2
G-0	Synth1 - Pattern 3
G#0	Synth1 - Pattern 4
A-0	Synth1 - Pattern 5
A#0	Synth1 - Pattern 6
B-0	Synth1 - Pattern 7
C-1	Synth1 - Pattern 8
C#1	Synth2 - Pattern on/off
D-1	Synth2 - Bank A
D#1	Synth2 - Bank B
E-1	Synth2 - Bank C
F-1	Synth2 - Bank D
F#1	Synth2 - Pattern 1
G-1	Synth2 - Pattern 2
G#1	Synth2 - Pattern 3
A-1	Synth2 - Pattern 4
A#1	Synth2 - Pattern 5
B-1	Synth2 - Pattern 6
C-2	Synth2 - Pattern 7
C#2	Synth2 - Pattern 8
D-2	808 - Pattern on/off
D#2	808 - Bank A
E-2	808 - Bank B
F-2	808 - Bank C
F#2	808 - Bank D
G-2	808 - Pattern 1
G#2	808 - Pattern 2
A-2	808 - Pattern 3
A#2	808 - Pattern 4

B-2	808 - Pattern 5
C-3	808 - Pattern 6
C#3	808 - Pattern 7
D-3	808 - Pattern 8
D#3	909 - Pattern on/off
E-3	909 - Bank A
F-3	909 - Bank B
F#3	909 - Bank C
G-3	909 - Bank D
G#3	909 - Pattern 1
A-3	909 - Pattern 2
A#3	909 - Pattern 3
B-3	909 - Pattern 4
C-4	909 - Pattern 5
C#4	909 - Pattern 6
D-4	909 - Pattern 7
D#4	909 - Pattern 8

Program Synth from Keyboard - Focus on Synth

Note	Function
C-0... C-1	Note C ... Note C
C#1	Octave down
D-1	Octave up
D#1	Accent
E-1	Slide
F-1	Note/Pause
F#1	Back
G-1	Step
G#1	Pitch Mode on/off

Program Synth from Keyboard - Focus on 808

Note	Function
C-0	Step 1 - On/Off
C#0	Step 2 - On/Off
D-0	Step 3 - On/Off
D#0	Step 4 - On/Off
E-0	Step 5 - On/Off
F-0	Step 6 - On/Off
F#0	Step 7 - On/Off
G-0	Step 8 - On/Off
G#0	Step 9 - On/Off
A-0	Step 10 - On/Off
A#0	Step 11 - On/Off
B-0	Step 12 - On/Off
C-1	Step 13 - On/Off
C#1	Step 14 - On/Off
D-1	Step 15 - On/Off
D#1	Step 16 - On/Off
E-1	Instrument - AC
F-1	Instrument - BD
F#1	Instrument - SD
G-1	Instrument - LT
G#1	Instrument - MT
A-1	Instrument - HT
A#1	Instrument - RS
B-1	Instrument - CP
C-2	Instrument - CB
C#2	Instrument - CY
D-2	Instrument - OH

D#2	Instrument - CH

Program Synth from Keyboard - Focus on 909

Note	Function
C-0	Step 1 - On/Off
C#0	Step 2 - On/Off
D-0	Step 3 - On/Off
D#0	Step 4 - On/Off
E-0	Step 5 - On/Off
F-0	Step 6 - On/Off
F#0	Step 7 - On/Off
G-0	Step 8 - On/Off
G#0	Step 9 - On/Off
A-0	Step 10 - On/Off
A#0	Step 11 - On/Off
B-0	Step 12 - On/Off
C-1	Step 13 - On/Off
C#1	Step 14 - On/Off
D-1	Step 15 - On/Off
D#1	Step 16 - On/Off
E-1	Instrument - AC
F-1	Instrument - BD
F#1	Instrument - SD
G-1	Instrument - LT
G#1	Instrument - MT
A-1	Instrument - HT
A#1	Instrument - RS
B-1	Instrument - CP
C-2	Instrument - CH
C#2	Instrument - OH

D-2	Instrument - CC
D#2	Instrument - RC

remarks: I don't know how to switch the 909 steps to the bright-lightening state (where all sounds are more powerfull)

Rebirth MIDI 2

(c) 1999 Chilled Dreams
This Buzz-Plugin is freeware. Dedicated to ya'all Rebirth lovers.

What does this thing

This Buzz machine is made to sync MIDI-Device with Buzz and send some MIDI-controller messages. My intention was to sync Rebirth with Buzz and so the MIDI features are limited to MIDI sync, controller sends and Note-On (with immediate Note-Off). That's all what I need for remote controlling Rebirth.

Limitations

Because of the limited features in Buzz regarding to sequencing infos, song position, start/stop and so on it's something complicated to let the sync work.
When hitting the Play-Button on Buzz no automatic MIDI Start would be send. This is because Buzz don't tells the machines if the Song is runnig. Instead you must enter a start command in a Pattern. The same is true for MIDI Continue and Song Position. Stop is handled automatically (Buzz tells the machines when you stop the song).

Installation / Settings

Copy the machine into the gear\generators directory. The Machine is called "Rebirth MIDI 2".
Right-click on the machine and choose "Settings" for setting up your MIDI device and other things:

MIDI Out Device	The device for MIDI Out Since this version I'm using the MIDI-functions of Buzz. So I am not opening a separate device like in the version before. Don't forget to enable the selected MIDI device in the preferences dialog of Buzz.
MIDI channel	The MIDI channel used to send controller commands to ReBirth
Sync Adjust	Adjusts the starting time of the MIDI-Syncs. Play around with this value if you have some problems. Normally this should be zero.

Song position unit	Selects what a song position actually means: MIDI beats (The standard for MIDI) or ReBirth Bars. It's easier to set to "ReBirth Bars" when using the machine in connection with ReBirth.
Start on Play Button	If enabled the machine tries to find the Buzz's Play-Button window. Then it hooks the button. If the Play Button is hit the machine starts to sends a MIDI start/continue and the sync clocks. So you don't have to write the start/ continue commands in all your Rebirth MIDI 2 patterns.

If you're using both programs (Rebirth and Buzz) on one computer you need a MIDI-Loopback-Device. Try MIDI-Yoke or HLD (contained on the Rebirth CD). This takes the MIDI-Output of one program and sends it to another.

Commands in the Patterns

First column - Start / Continue / Song pos Commands

0 - MIDI Start	Send a MIDI Start. After this the MIDI timeclocks are sent. A MIDI Start lets a song playing from it's beginning.
1 - MIDI Continue	Send a MIDI Continue and then the timing clocks. So a connected MIDI slave continues playing the song from where it was stopped before or from the position you've cued to.
2 - Song Position	Send a MIDI Song Position. Buzz don't tells me the current song position. So if you wanna use this, you must enter this manually. A Song Position is given in "MIDI-Beats". A MIDI-Beat is on Step or Row (on 4/4-Beat). **This feature is currently untested**

Timing clocks are sent until you hit the "Pause"-Button in Buzz. When playin' again the connected slave starts to work on the next Continue or Start Command (that you've entered in a pattern).

2nd column - Song position value

<value>	The song position value for command 1 (continue) and 2 (song pos) For Command 1 (continue) this value only counts if it's nonzero and the song is not currently playing! Otherwise it's ignored.

column 3 to 10 - Rebirth Pattern switch

Here you can easily send some pattern switch commands to ReBirth. See the statusline for description. You can change Rebirths pattern bank and the pattern for every machine. **Note**: ReBirth should be in Pattern mode for meaningfull using this feature.

Commands for a Track

In every track you can send a Controller and a Note-On. The statusline shows a little help for the controller values. Rebirth 2.0 standard mapping is assumed.
See the MIDI-Tables for the right controller values and MIDI-Notes!

Guide to use

Create a pattern named Start (or whatever). At the beginning in the pattern enter the command "0" (Start).
Create a pattern named Continue (or what you like). Here use the command "1" (Continue).
In the song sequencer you should set the Start pattern at the beginning. The continue patterns can follow if you need to continue the song on every position.
Sometimes Rebirth needs a short while to sync correctly after a Start. So it could be useful to add an empty startup pattern before the real song begin on both programs.

Performance problems

I think you need a fast computer if your're rocking Rebirth and Buzz on a single station. My 233 MHz AMD is suckin' up when the music contains more than some drums. On a 300 MHz PII it's running well. Also the timing is more stable on a faster machine.

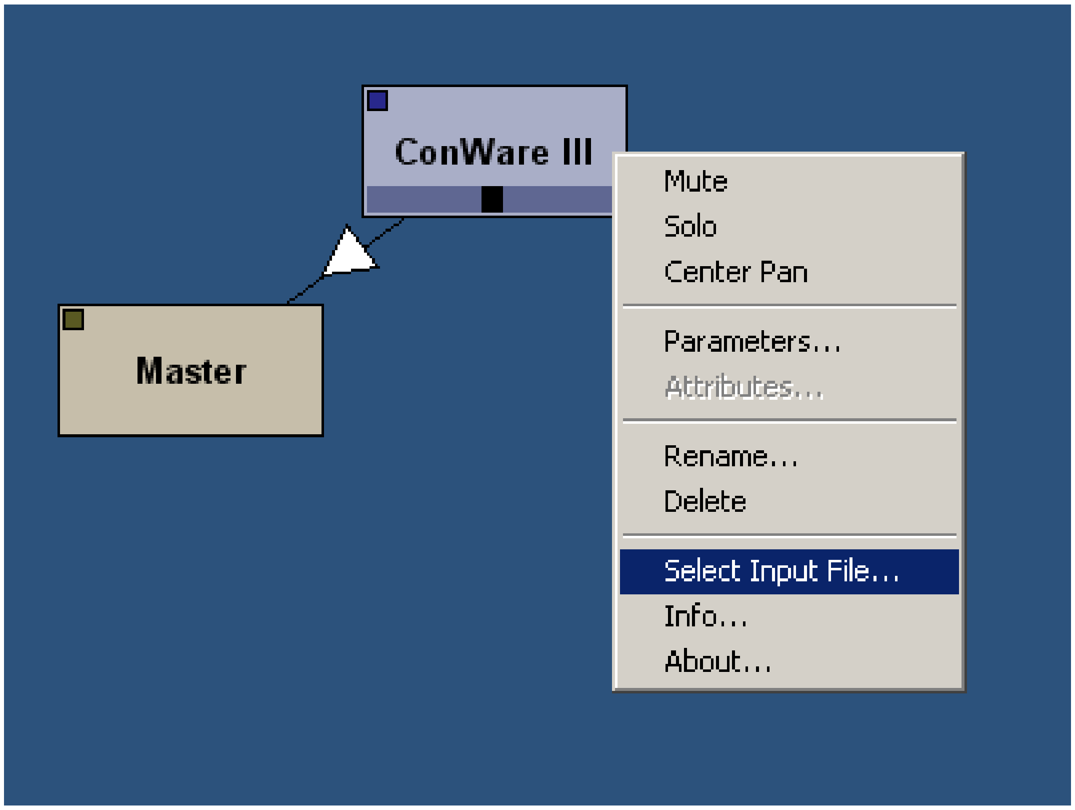
Timing problems

Buzz uses a more or less large wave buffer for preparing the sound. This means that a sound is not played immediately when created but with a short delay. Because of that every MIDI-Clock would comes too early. I'm calculating a short delay, which is exactly the difference between wave creation and wave playing (assuming Buzz tells no lies). So the timing is correct for Buzz's side. On Rebirth there's also a wave buffer. I suggest to play around with Rebirths "Adjust Sync" setting to let the beats roll tight.

Bug-Reports, flames etc:
eMail: chilleddreams@chilleddreams.de
Homepage: http://www.chilleddreams.de/

rDev Software Continous Wave Reader III

Final Version

What it does	Do you have a band and you have recently recorded some songs with your favourite multitrack recording program? Well, that's fine but now you have to do the mixing work. Of course, the program of your choice is the buzz tracker. ;-) However, all those wave files are too big to fit into memory, so using wavetable is not an option. Luckyly, there are HD Player generators out there, so you grab one and it works just fine. However, each time you change something, you have to start the whole buzz-song again because you can't resume playing at the right position. This is exactly what the "**ConWare III**" generator fixes!
How to use it	First of all, add a "**ConWare III**" generator to your project. After that, you have to select an input file for it. Click onto the box representing the generator with your right mouse button and choose "Select input file" from the menu.  ConWare III Master Mute Solo Center Pan Parameters... Attributes... Rename... Delete Select Input File... Info... About... In the following dialog box you can choose your favourite file. You can only use "simple" 16-bit wave files, whether stereo or mono. It is also neccessary that the wave file has the same sample rate as the output of the Buzz Tracker because the **ConWare** generator does not do any sample rate conversion! **Warning:** If you have opened the selected file in another application, opening the wave file may fail. When you save the song to disk, the generator will only write the

filename to disk, not the sample data! So, don't move your wave files around, otherwise you will get an error message when loading a song from disk. However, if this occurs, a error message will pop up during loading of the song. Once the song is loaded, you can use the "Select Input File" menu item again and tell the generator the new location. You will not lose any settings!

Ok, now we have selected our input file. Now we have to set some "reentrance marks" to make sure that the **ConWare** reader knows where to play from no matter where we start within the song. First of all, we have to create a pattern which holds our "reentrance marks". It should contain the following data:

```
0  | 0001 .... .
1  | 0002 .... .
2  | 0003 .... .
3  | 0004 .... .
4  | 0005 .... .
5  | 0006 .... .
6  | 0007 .... .
7  | 0008 .... .
8  | 0009 .... .
9  | 000A .... .
10 | 000B .... .
11 | 000C .... .
12 | 000D .... .
13 | 000E .... .
14 | 000F .... .
15 | 0010 .... .
usw.
```

There is a mark in every line so that the generator always knows what the current position in this song is. However, there may be empty lines in the pattern, but a few seconds after the generator gets the last mark, it stops playing. In most cases, it is nesseccary to split the marks into multiple patterns, but make sure that you add them to the sequence in the correct order and don't forget: The marks are in hexadecimal numbert!

Now we can start playing and....voilà! It works fine.

Advanced use	If you add a "0000" as "reentrance mark", the generator stops playing (warning: no fadeout!) With the second parameter, you can finetune the playing position. Normally, you don't change it during playing of the song. For example, if you set the value to "10000" (decimal), the generator plays 10000 samples ahead (at a sample rate of 44.100 khz, this would be about ~226 ms). Using the third parameter, you can choose whether to use the left or the right channel of a stereo input file. The generator always outputs mono a signal. This parameter does not have an effect with mono files. To get exact results, the sample_rate should be exactly dividable by the amount of samples per tick. To get to know if this is the case, you can choose "Info..." from the generators popup menu and look out for the line "Exact match". If the samplerate multiplied by 60 is exactly dividable by the amount of BPM multiplied by the amount of ticks per beat, it matches exactly. For example, the setting "120 BPM, 2 Ticks" fits, because (44.100 *khz* * 60 *sec/min*) / (120 *beats/min* * 2 *ticks/beat*) = 11025.0 *1/tick*
License	This plugin is freeware.
Author	If you have any suggestions or comments, feel free to drop me a mail. E-Mail: POMBBEHTXMWX@spammotel.com Homepage: http://www.rdev-software.de.vu

HI,

this is 11-Ox - my ?th attempt to make an effect with that you can build
new fm-synths. but i'm not so happy with it ;-(. anyway it's an interesting
machine, cause it uses the public dll memory to communicate with other
Ox's. i added some small demos that could help to understand. the most
important sliders (if using more Ox's) are ID and Source. Every Ox should
have it's own ID! If you connect one Ox to another it will get a note signal
of the destination, if it's source is equal to the destination's ID.
The frequency is controlled via the Pitch. In Carrier-Mode the notes are
shifted. As a Modulator the frequencys are multiples of the frequency of then
source Ox. If you want FM you have to increase the FM-Amount and FM-Depth of
the Carrier.

HAVE FUN
z.war

(if someone's goog in grafics, tell me - i'need good looking skins ;-)

SLIDERS

Mixer:

Osc	Internal Oscillator Amount
Thru	Amount of input (Additive)
Ring	Amount of Osc * Thru (Ring Modulation)
FM	Ammount of the FM-Signal (modulated by the amplitude of the input)

Amplitude Envelope:

Attack	Time
Decay	Time
Sustain	Level (so use Note-off)
Release	Time

Oscillator:

Waveform	Sin,Tri,Saw,Pulse
Sync	Carrier or Modulator (how to calculate the frequency)
Pitch	Osc-Pitch (depending on Sync)
Detune	best use for Additive-Synths
Glide	Portamento

FM:

Depth	Sets how hard the input-amp. modulates the Osc-Freq.
Fine	like Depth but finer ;-)
Self	Sets how hard the FM-signal modulates itself
Self Fine	like Self but finer ;-))

Operator:

ID	ID of every Ox
Source	describes which Ox is taken as source

ATTRIBUTES

Reserved	(for Midi)
Amount Inertia	Ramp-Time for Mixer-changes (to avoid clicks)
Attack Range	Ramp-Time for Attack (decrease for very fast periods)
Decay Range	Ramp-Time for Decay
Release Range	Ramp-Time for Release
Glide Speed	Ramp-Time for Glide

DEMOS
-----

demo1	using Ox as a simple Sine-generator, playing a chord
demo2	a simple 2-Operator FM-Synth
demo3	a simple 2-Oscillator virtual-analog
demo4	a mixture of 3op-FM with 2-Carriers (attempt for pad)

KNOWN BUGS

-If using more tracks sometimes clicks/noise is heared. You can avoid this by
 changing the envelope.

HINTS

-If using more tracks, the modulators should have the same number of tracks.

Rebirth MIDI 2

What does this thing

This Buzz machine is made to sync MIDI-Device with Buzz and send some MIDI-controller messages. My intention was to sync Rebirth with Buzz and so the MIDI features are limited to MIDI sync, controller sends and Note-On (with immediate Note-Off). That's all what I need for remote controlling Rebirth.

Limitations

Because of the limited features in Buzz regarding to sequencing infos, song position, start/stop and so on it's something complicated to let the sync work.
When hitting the Play-Button on Buzz no automatic MIDI Start would be send. This is because Buzz don't tells the machines if the Song is runnig. Instead you must enter a start command in a Pattern. The same is true for MIDI Continue and Song Position. Stop is handled automatically (Buzz tells the machines when you stop the song).

Installation / Settings

Copy the machine into the gear\generators directory. The Machine is called "Rebirth MIDI 2".
Right-click on the machine and choose "Settings" for setting up your MIDI device and other things:

MIDI Out Device	The device for MIDI Out Since this version I'm using the MIDI-functions of Buzz. So I am not opening a separate device like in the version before. Don't forget to enable the selected MIDI device in the preferences dialog of Buzz.
MIDI channel	The MIDI channel used to send controller commands to ReBirth

Sync Adjust	Adjusts the starting time of the MIDI-Syncs. Play around with this value if you have some problems. Normally this should be zero.
Song position unit	Selects what a song position actually means: MIDI beats (The standard for MIDI) or ReBirth Bars. It's easier to set to "ReBirth Bars" when using the machine in connection with ReBirth.
Start on Play Button	If enabled the machine tries to find the Buzz's Play-Button window. Then it hooks the button. If the Play Button is hit the machine starts to sends a MIDI start/ continue and the sync clocks. So you don't have to write the start/continue commands in all your Rebirth MIDI 2 patterns.

If you're using both programs (Rebirth and Buzz) on one computer you need a MIDI-Loopback-Device. Try MIDI-Yoke or HLD (contained on the Rebirth CD). This takes the MIDI-Output of one program and sends it to another.

Commands in the Patterns

First column - Start / Continue / Song pos Commands

0 - MIDI Start	Send a MIDI Start. After this the MIDI timeclocks are sent. A MIDI Start lets a song playing from it's beginning.
1 - MIDI Continue	Send a MIDI Continue and then the timing clocks. So a connected MIDI slave continues playing the song from where it was stopped before or from the position you've cued to.
2 - Song Position	Send a MIDI Song Position. Buzz don't tells me the current song position. So if you wanna use this, you must enter this manually. A Song Position is given in "MIDI-Beats". A MIDI-Beat is on Step or Row (on 4/4-Beat). **This feature is currently untested**

Timing clocks are sent until you hit the "Pause"-Button in Buzz. When playin' again the connected slave starts to work on the next Continue or Start Command (that you've entered in a pattern).

2nd column - Song position value

<value>	The song position value for command 1 (continue) and 2 (song pos) For Command 1 (continue) this value only counts if it's nonzero and the song is not currently playing! Otherwise it's ignored.

column 3 to 10 - Rebirth Pattern switch

Here you can easily send some pattern switch commands to ReBirth. See the statusline for description. You can change Rebirths pattern bank and the pattern for every machine. **Note**: ReBirth should be in Pattern mode for meaningfull using this feature.

Commands for a Track

In every track you can send a Controller and a Note-On. The statusline shows a little help for the controller values. Rebirth 2.0 standard mapping is assumed.
See the MIDI-Tables for the right controller values and MIDI-Notes!

Guide to use

Create a pattern named Start (or whatever). At the beginning in the pattern enter the command "0" (Start).
Create a pattern named Continue (or what you like). Here use the command "1" (Continue).
In the song sequencer you should set the Start pattern at the beginning. The continue patterns can follow if you need to continue the song on every position.
Sometimes Rebirth needs a short while to sync correctly after a Start. So it could be useful to add an empty startup pattern before the real song begin on both programs.

Performance problems

I think you need a fast computer if your're rocking Rebirth and Buzz on a single station.
My 233 MHz AMD is suckin' up when the music contains more than

some drums. On a 300 MHz PII it's running well. Also the timing is more stable on a faster machine.

Timing problems

Buzz uses a more or less large wave buffer for preparing the sound. This means that a sound is not played immediately when created but with a short delay. Because of that every MIDI-Clock would comes too early. I'm calculating a short delay, which is exactly the difference between wave creation and wave playing (assuming Buzz tells no lies). So the timing is correct for Buzz's side. On Rebirth there's also a wave buffer. I suggest to play around with Rebirths "Adjust Sync" setting to let the beats roll tight.

Bug-Reports, flames etc:
eMail: chilleddreams@chilleddreams.de
Homepage: http://www.chilleddreams.de/

rp|Farbrausch again ...

This host has been ported from Psycle to Buzz by Cyanphase and edited by Ryg.
There are surely still bugs and features missing but you're invited to test
and use it. It may work better with some VSTi's than the other Adapters, and worse
in other cases. This needs to be experienced.

Thanks to Cyanphase for providing the code and taking care.
Thanks again, Ryg for beeing coorperative (He had no choice ;),
Tom for testing & the usual Buzz suspects for beeing alive and kickin' !

Feel free to contact Cyanphase at blakee@rovoscape.com, Ryg at fg@farb-rausch.de or
me at rp@farb-rausch.de

Bye,
Ronny

RnZnAnFnCnRnL VST host

[Rout, Zephod, Arguru, FSM, Cyanphase, Ryg, Ld0d] v0.8.1.2

What's new?

A lot of bugfixes and new amazing features have been added:

-Parameter Automation
-Parameter Learn
-Parameter movement smoothing (Inertia)
-Midi Learn, Record, Tempo Sync
-Individual track columns for Note Delay
-Note effects
-Total parameter recall

How can I get the buzz parameter window instead of the VST GUI?

Right click on machine, then select parameters...

Why does a vst plugin.dll not appear in the menu?

First off all, was Buzz running when you placed it there? The VST Folder will be scanned every time you launch Buzz. Make sure you copied the plugin.dll (with all files it might need!) into the VST Folder you specified
in the Extended Options

Controller assignment

There are several ways to assign plugin knobs to buzz parameter sliders. Most plugins send their parameters to the host that you can choose the parameter using the (n)-Param Num sliders to select a controller. (n)-Param Val will send the set value to the controller choosen above. The movement of (n) Param-Val sliders can be smoothed with the (n)-ParamInert slider to avoid heavy jumps whilst moving the slider.

Plugins which work with Midi CC's to control knobs (such as Quadrasid or Reaktor) mostly don't send any parameter information. You can mostly define which Gui element of function shall listen to which Midi CC. The host supports Midi CC's as well. You can find them in the right range of the (n)-Param Val sliders (Midi CC0 - MidiCC255). Sending the values works as stated above.

Pitchbend can be found as own controller next to None, as MIDICC256. It's number 3071 or 0BFF.

Parameter learn

Finding the correct VST Plugin Parameter can be a tricky thing if you think about synths like FM7 which are stuffed with hundrets of knobs/sliders/buttons... To make it easier for you, the adapters can "learn" which VST GUI element you want to control. Open the VST plugins' gui and go to the 'Parameter' menu, select the Parameter slider you want to assign the Gui element to (e.g. 'Learn 0'). Once you moved a VST gui element after that, the regarding buzz parameter slider (here 0-Param Num) will be set to the choosen VST Knob/Slider...

Same works with plugins that listen Midi Controllers, you only have got to enable "Midi Learn" before choosing the slider number.

'Clear All' should be self explaining.

Track Commands and their Values

00 xxxx: Note Cut (scaled in 1/256 ticks)

01 00xy: Retrigger
x = delay between triggers, delaytime = (x + 1) / 32 ticks
y = number of triggers - 1 (note)
e.g. 01 00F0 => retrigger once exactly at the middle of the tick
e.g. 01 0072 => retrigger three times on quarter-ticks, 0/4:normal note, 1/4:first, 2/4:second, 3/4:third

02 00xy: Arpeggio
x = first pitch shift in halftones
y = second pitch shift in halftones
e.g. 02 0037 => first play normal note, then at 1/3 tick note 3 halftones up from normal, then at 2/3 tick note 7 halftones up from normal. (minor chord)

Attributes

Midi Channel [1/16] - sets Midi Channel to be used for incoming events, 0 disabled
Midi Velocity [0/1] - switches Velocity on/off
Midi Transpose [0/48] - transposes incoming Midi Notes by given value (24=no transpose)
Midi Recording Enabled [0/1] - turns record from Midi Input on / off
Nodelay/Quantize [0/1] - turns recording of Note Delay values on (0) and off (1, considered as sorta quantizing)
Nocut/Quantize [0/1] - turns recording of Note Cut values on (0) and off (1, dito)
Humanize Delay Amount [0/128] - adds a random factor to note delay, strenght adjusted by attribute
Humanize Velocity Amount [0/128]- adds a random factor to note velocity, strenght adjusted by attribute

Midi Sync Issues

The host is able to send tempo to the vst plugin.. It's yet not possible to include the Song position (for plugins which use an internal sequencer). In case you use Reaktor and ensembles which feature internal sequencers you might face the situation that those sequencers do not reset their position when buzz stops. There is nothing that we can do hostside but you can check the ensemble for the "Start" module. Its Reset event output is responsible for resetting sequencers' positions.

We will try to provide modified Versions of such Reaktor ensembles in the future. Stay tuned!

Known VST Plugin Issues

-Mda DX10 neglects to work
-AKAI Professional Quadcomp won't work

Known Host Issues

-Might crash if vst plugin dll used in song is not found
-Saving Presets isn't implemented
-Changing the program may work with some plugins and not with others.

A few words about these fine VST Adapters from rp|fr

These 2 Buzz .dll's have been started by Cyanphase (blakee@rovoscape.com), who ported the Adapters from Psycle and fixed out many issues as well as implemented things. Having no time to carry them on for a while, i begged him for the sources which he provided (still biggest props for that!). I took the sources to Ryg (fg@farb-rausch.de), who also fixed a few bugs as well as implemented the missing playback feature in the VSTi Loader. In return, Cyanphase fixed even more things... I finally asked ld0d (ld0d@kolumus.fi) for Parameter Automation, but he pushed it far beyond that. The latest version i received from him is superb compatible and features a lot of things you can also read below.

Not to forget Arguru here who helped to fix a few compability problems as well as Rout, Zephod & FSM who started all that mess to load plugins into a plugin.
And of course DjLaser (djlaser@buzztrack.com) who is helping with this text!

Cheers,
Ronny (rp@farb-rausch.de)

This is rp|farbrausch writing a few words about this version...

The RnZnAnFnR VST Instrument Adapter.dll is based on the latest Version (0.35) of
FSM's VSTi Adapter. Even if this one still tells you it's 0.35, it contains a few
but remarkable bugfixes by Ryg^Farbrausch.

So, wtf changed?

Former Versions always crashed when you tried to load a VSTi with more than 1 stereo
output. This Version let's you use those plugins (such as tc mercury-1, quadrasid,
even reaktor [which hasn't crashed here anymore using this version...]).

Thanks to FSM for sharing his code and his codework!
Thanks to Ryg for fixing this bug and not killing me for tripping on his nerves ;)
Thanks Tom for testing.
Also thanks to Rout, Zephod and Arguru for working on this adapter and making it
happen!

Contact Ryg at fg@farb-rausch.de and me at rp@farb-rausch.de

Cheers,
Ronny

Rout 808 version 1.02

What is it ?

This is a drum machine. It may sound a little like the Roland TR-808.

The idea of this machine was to allow 'Buzzers' to create songs that take less disk space [and less download time...] (there's no need to include all the samples into each song).

History

version 1.0 (01.24.99)
First release.

version 1.02 (03.06.99)
I've fixed a bug that slowed down dramatically the computer when one of them was connected with an EQ-3 machine.

Author

Mathieu Routhier (mrouthier@cyberdude.com)
web page: members.xoom.com / Rout

Thanks to Richard Hoffmann for his lowpass filter code. Thanks to Oskari Tammelin too for his help concerning lowpass filters.

This is DONATIONWARE. If you want to be kind, you can send me any amount of money (or anything else you think I'd like to have) to the following address. Thank you.

Mathieu Routhier
3431 de la dauversière
Ste-Foy, Qué
G1X 2H6
Canada

Rout SoundFont(R) Loader version 0.9b

What is it ?

This is a plugin to use with Buzz (www.buzz2.com). It allows you to load a SoundFont(R) into buzz and to use its instruments. It doesn't support all the features of the SoundFont 2.0 specification. In fact, it only supports volume envelope, loop points and multi-layering of instruments. See for yourself.

Your soundfonts will not sound as richer as if they were played directly from your SB Live (or any soundfont compatible sound card). Some soundfonts will sound good and some will suck. To improve their sound, I suggest using a reverb machine or a delay machine.

It is possible to access drumkits from a soundfont. Just select Bank 128.

This is a stereo generator. So you need Buzz version 1.2 to use it.

Don't email me saying "This soundfont sounds bad".

Thanks goes to Ninereeds and Jeskola who helped me on this project.

History

version 0.9b (01.04.2000)
Beta release.

version 1.0 (03.02.2000)
First stable release.

- **Support for stereo soundfonts added.**
- **No more crash on loading when machine is playing.**
- **Multi-track bug fixed.**

version 1.1 (03.03.2000)
Does not click anymore. I added something that fades the stopping note in a half-tick.
Added a pretty little skin ! (you need the latest buzz 1.2 to see it)

Author

Mathieu Routhier (mrouthier@mail.com)
web page: members.xoom.com / Rout

This is DONATIONWARE. To support me and encourage the creation of other Buzz machines, you can send me any amount of money (or anything else you think I'd like to have [suggestion: CDs]) to the following address. Thank you.

Mathieu Routhier
3431 de la dauversière
Ste-Foy, Qué
G1X 2H6
Canada

SPECCY v2.0

In brief

- **Type** Generator
- **Autor** Ruff
- **E-mail** lrs@cg.ukrtel.net
- **Description** Multi-purpose stereo generator

Ïìäòpèìaé âiò÷èçíÿíoão âèpoáíèka !

List of parameters:

Symbol "/" at the begining shows that the parameter has corresponding inertia.
Parameters in "<>" are amplitudes. "--()--" meens 3D location that is combination of panning and lo-pass filtering with different cutoffs for L and R.

```
                GLOBAL PARAMETERS

        Command            Description

        Waveform:          Voice1 waveform (sine-saw-square-triangle)
        Unison:            Voice2 detune
        unison MIX:        Way of mixing voice1 with voice2
        FM/PM scale:       Amount of modulating if previous parameter if PM or F
        /voice1 --()--:    Voice1 3d location
        /voice1 res.:      Voice1 resonance
        /voice2 --()--:    Voice2 3d location
        /voice2 res.:      Voice2 resonance
        /voice2 pedist.:   Voice2 cutoff pedistall (positive or negative cutoff
offset)
        /resonance:        overall resonance
        /pedistal:         overall cutoff pedistall (positive or negative cutoff
offset)
        LocCutoffEnv:      shows how 3d location parameters influence on assimetric L
or R cutoff
        LocPanEnv:         shows how 3d location parameters influence on assimetric L
or R amplitude
        Am gain:          _is used to gain signal after AM mixing (if it takes place
anywhere)
        Attack:            \
        Decay:              \
        Sustain:             ADSR-envelope parameters
        SustainLevel:       /
        Release:          _/
        fb delay:          feedback line delay
        fb PM scale:       Amount that delayed signal (feedback line) is modulated by
original one
                                  (values more then 10 are used to produce noize,
```

drums and hats)

way1 amp:	Original signal amplitude for mixing into feedback line
way23 amp:	Way23 signal amplitude for mixing into feedback line to reach feedback effect
way23ballance:	Way23 signal consists of delayed signal and delayed signal mixed with original signal in a special way (see final MIX)
fb MIX:	Way1-way2-way3 mix formula
dry amp:	Original signal amplitude (see final MIX)
final MIX:	Way of mixing original signal with the signal from feedback line (delayed signal)
Arp.length:	Time to play 1 arpeggio note
Arp.smooth:	Time to morph between arpeggio notes
loc. inertia	3d location inertia
res. inertia	resonance inertia
pedist. inertia	pedistal inertia

TRACK PARAMETERS

Command	Description
Volume:	Max. volume that the sound reaches when attack is complete
/--()--:	Overall 3d location
Ornament:	Ornament number

Slide trigger is used to slide from one note to another (like 303). It is done as smooth, as it is set in Arp.smooth parameter.

Details

This machine may be used to create everything you want - pads, acid sounds, zx-beeps, fm bells, noize (lots of noize ;-)), hats, ... You don't need to process it through delay, reverb or room simulation effects, because using phase modulated feedback with different 3d locations and pedistals you can simulate this effects in a rather funny way or create something special. I have saved all SPECCY I features in SPECCY II generator, so it contains the same 34 ornaments, among them the most popular on Speccy are: OctaveAttack, Slideup, PerfectOct, DoubleOct, Kvint-Octave, MinorChord, MajorChord.

Thanks and greetz

- **jiSm**. Tnx 4 skinz, cool demotune and presets
- **kixx**/randomajestiq (imp/dms ;->). SPECCY is always watching you
- **sleepy** 4 moral support
- everybody who reads this

Stuff

Speccy II was delayed for such a long period only because some bad guys stole my PC. Of cause, it sucked, but in spite of this S2 has been released. Be

happy ;-> If you have found this machine very useful and even wrote a song using it, of course, I won't mind to hear it. But if you don't feel any nostalgie when you hear triangle wave changing its pitch 50 times per second and you consider that this machine suxx and I'm a lamer, I would be very pleased to receive from you some texts on DSP programming, where I could find some formulae.
So, if you have any suggestions or you have found some bugs or if you just want to contact, mail to:

lrs@cg.ukrtel.net

TRIP-HOP 4ever !
keep buzzzzzzing !

SPECCY v1.1

In brief

- **Type** Generator
- **Autor** Ruff
- **E-mail** lrs@cg.ukrtel.net
- **Description** The only machine that can produce ornamented ZX-Spectrum sound without clicks. In addition it has some features that has nothing to do with ZX-Speccy.

Ïiàòpèìaé âiò÷èçíÿíoão âèpoáíèka ! ;->

What's new in v1.1 ?

1. Slightly bugfixed ;->
2. Totally declicked (Yeah !!! I've done it !!!)
3. Reeled And Skinned ((RedSnapper4ever;-))
4. Antialiased with 24db lopass
5. Attributed (antialiasing on/off, work always on/off)
6. 'About' dlg has been added ;->

By default WorkAlways attribute is 1 (on), because there are some non-correctly written effects (like DexFiltah with low cutoff and high resonanse) that calculates nothing if WM is write only. But if you use only correctly written effects, then set WorkAlways to 0. It'll save your CPU resources in some cases.

List of parameters:

```
        Command                 Description

        Waveform:               Master osc. waveform (sine-saw-square-triangle)
        Unison:                _Slave osc. detune
        Attack:                 \
        Decay:                   \
        Sustain:                  ADSR-envelope parameters
        SustainLevel:            /
        Release:               _/
        Arp.length:             Time to play 1 arpeggio note
        Arp.smooth:             Time to morph between arpeggio notes

        Volume:                 Max. volume that the sound reaches when attack is
complete
        Ornament:               Ornament number
```

Details

I have made 34 ornaments, among them the most popular on Speccy are: OctaveAttack, Slideup, PerfectOct, DoubleOct, Kvint-Octave, MinorChord, MajorChord.

- This machine is **not** AY-emulator. It does not support AM and noise (yet), but due to the fact that original AY has rather flat sound, I have added some non-AY features, such as chorus (to get chorus effect set **Unison** to 3..10, or to get phaser effect set it to 1..2), different waveforms (it would be rather stupid not to add this feature :-)).
- And, what about the original Speccy sound, if you care only about it's originality, set **Waveform** to triangle, **Unison** = 0, **Release** = the less the better, **ArpLength** = 20.000 ms, **ArpSmooth** = 2.000 ms.

Thanks follow

- to jiSm 4 skinning, finding the main bug, the BEST demotune and lots of moral support
- to ladproject 4 some presets and moral support
- to Jochem van der Lubbe for some info (thnx alot)
- to Claudio Bustos for bugtesting (sorry, I can't find a PC that crashes on SPECCY to fix the bug), info and moral support
- to Bert Hulshoff (sorry...)
- to Swetlana Semenowa 4 moral (and may be some other ;->) support
- to koses p. and Wannja 4 greetinxx ;-> and moral support
- to James Counihan 4 moral support
- to Feliciano GuimarÖes 4 moral support
- to Oskari Tammelin for the best tracker on PC platform.

Stuff

If you have found this machine very useful and even wrote a song using it, of course, I won't mind to hear it. But if you don't feel any nostalgie when you hear triangle wave changing its pitch 50 times per second and you consider that this machine suxx and I'm a lamer, I would be very pleased to receive from you some texts on DSP programming, where I could find some formulae to make some cool stuff.
I'll be glad to find smth at:

lrs@cg.ukrtel.net

TRIP-HOP 4ever !
Thousand of greetinxx from ex-USSR !
keep buzzzzzzing !

Scales.txt file for BTDSys PeerScale
Source: http://www.xs4all.nl/~huygensf/scala/
(via http://zabumba.free.fr/zikzip/modz.html)

Ditonic

5 7 Honchoshi: Japan

7 5 Niagari: Japan

10 2 Warao ditonic: South America

Tritonic

3 4 5 Peruvian tritonic 2

3 7 2 Ute tritonic

4 3 5 Peruvian tritonic 1, Raga Malasri

5 2 5 Sarvasri Raga, Warao tritonic: South America

5 5 2 Sansagari: Japan

6 1 5 Ongkari Raga

Tetratonic

1 5 1 5 Messiaen truncated mode 5

5 1 5 1 Messiaen truncated mode 5 inverse

2 4 2 4 Messiaen truncated mode 6

4 2 4 2 Messiaen truncated mode 6 inverse

1 4 3 4 Lavangi Raga

2 1 7 2 Warao tetratonic: South America

2 2 3 5 Eskimo tetratonic (Alaska: Bethel)

2 3 2 5 Genus primum

2 3 4 3 Bhavani Raga

2 4 5 1 Sumukam Raga

4 3 3 2 Mahathi Raga

3 4 3 2 Bi Yu: China

5 2 3 2 Genus primum inverse

Pentatonic

2 3 2 1 4 Han-kumoi: Japan, Raga Shobhavari

2 1 4 1 4 Hira-joshi, Kata-kumoi, Yona Nuki mineur: Japan

1 4 2 1 4 Hon-kumoi-joshi, Sakura, Akebono II: Japan, Olympos Enharmonic, Raga Salanganata, Saveri, Gunakri (Gunakali), Latantapriya

1 4 2 3 2 Kokin-joshi, Miyakobushi, Han-Iwato, In Sen: Japan, Raga Vibhavari (Revati), Bairagi, Lasaki

1 4 1 4 2 Iwato: Japan

2 3 2 2 3 Ritusen, Ritsu (Gagaku): Japan, Zhi, Zheng: China, Raga Devakriya, Durga, Suddha Saveri, Arabhi, Scottish Pentatonic, Ujo, P'yongjo: Korea, Major complement

2 2 3 2 3 Major Pentatonic, Ryosen, Yona Nuki majeur: Japan, Man Jue, Gong: China, Raga Bhopali (Bhup), Mohanam, Deskar, Bilahari, Kokila, Jait Kalyan, Peruvian Pentatonic 1, Ghana pent.2

2 3 2 3 2 Yo: Japan, Suspended Pentatonic, Raga Madhyamavati, Madhmat Sarang, Egyptian, Shang, Rui Bin, Jin Yu, Qing Yu: China

2 3 3 2 2 Chaio: China

2 2 2 3 3 Kung: China

1 4 2 2 3 Altered Pentatonic, Raga Manaranjani II

2 1 2 4 3 Abhogi Raga

4 2 1 4 1 Amritavarshini Raga, Malashri, Shilangi

2 1 2 3 4 Audav Tukhari Raga

4 1 4 2 1 Bhinna Shadja Raga, Kaushikdhvani, Hindolita

1 2 4 1 4 Balinese Pelog, Raga Bhupalam, Bhupala Todi, Bibhas

2 2 3 1 4 Bhupeshwari Raga, Janasammodini

3 2 4 2 1 Chandrakauns Raga (modern), Marga Hindola, Rajeshwari

3 2 4 1 2 Chandrakauns (kafi) Raga, Surya, Varamu

3 2 3 3 1 Chandrakauns (kiravani) Raga

1 2 3 2 4 Chhaya Todi Raga

2 3 2 4 1 Desh Raga

1 6 1 3 1 Deshgaur Raga

5 2 1 3 1 Devaranjani (Devaranji) Raga

3 2 2 3 2 Minor Pentatonic, Raga Dhani (Suddha Dhanyasi), Abheri, Udhayaravi Chandrika, Qing Shang, Gu Xian, Jia Zhong, Yu: China, P'yongjo-kyemyonjo: Korea, Minyo: Japan, Peruvian Pentatonic 2, Blues Pentatonic

4 2 1 2 3 Dhavalashri Raga

4 1 2 4 1 Gambhiranata, Ryukyu: Japan, Raga

1 4 2 4 1 Gauri Raga

4 1 3 3 1 Bacovia: Romania, Raga Girija

2 2 3 4 1 Hamsadhvani Raga (Hansadhvani)

3 3 2 2 2 Harikauns Raga, Chin: China

3 2 1 4 2 Jayakauns Raga

4 1 4 1 2 Khamaji Durga Raga

3 2 2 1 4 Kokil Pancham Raga

1 4 3 3 1 Kshanika Raga

2 2 2 5 1 Kumurdaki (Kumudki) Raga

5 2 2 1 2 Kuntvarali Raga

3 2 3 2 2 Malkauns (Malakosh) Raga, Raga Hindola, Man Gong, Quan Ming, Yi Ze, Jiao: China

4 3 2 2 1 Mamata Raga

1 3 3 3 2 Manaranjani I Raga

2 5 2 1 2 Matha Kokila (Matkokil) Raga

1 3 1 3 4 Megharanjani Raga, Syrian pentatonic

1 3 1 6 1 Megharanj Raga

3 1 3 2 3 Mohanangi Raga

3 3 1 4 1 Multani Raga

1 1 4 1 5 Nabhomani Raga

4 1 2 2 3 Nagasvaravali Raga, Raga Mand

3 2 2 4 1 Nata Raga, Udayaravicandrika, Madhuranjani

2 2 5 2 1 Neroshta Raga

4 2 3 2 1 Hindol Raga (Sunada Vinodini), Sanjh ka Hindol

2 3 3 3 1 Priyadharshini Raga

2 1 2 2 5 Purnalalita Raga, Chad Gadyo: Jewish, Ghana Pentatonic 1, Nando-kyemyonjo: Korea

5 2 2 2 1 Puruhutika Raga, Purvaholika

1 3 3 2 3 Rasika Ranjani Raga, Vibhas (marva), Scriabin

2 3 4 2 1 Rasranjani Raga

1 3 3 1 4 Reva Raga, Revagupti, Ramkali, Vibhas (bhairava)

3 3 1 3 2 Samudhra Priya Raga, Madhukauns (pentatonic)

1 5 1 1 4 Saugandhini Raga, Yashranjani

2 1 4 2 3 Sivaranjini (Shivranjani) Raga, Akebono I: Japan, Dorian Pentatonic

3 4 1 2 2 Shailaja Raga

2 4 1 2 3 Shri Kalyan Raga

2 4 1 4 1 Vaijayanti Raga, Hamsanada

4 3 2 1 2 Valaji Raga

4 1 2 1 4 Zilaf Raga

2 2 3 3 2 Dominant Pentatonic

4 1 2 3 2 Mixolydian Pentatonic, Raga Savethri

3 2 2 2 3 Minor added sixth Pentatonic, Kyemyonjo: Korea

Hexatonic

2 2 2 2 2 2 Whole-tone, Messiaen mode 1, Raga Gopriya, Anhemitonic

2 3 2 2 1 2 P'yongjo: Korea, Yosen: Japan, Raga Darbar, Narayani, Suposhini, Andolika, Gorakh Kalyan, Mixolydian Hexatonic

2 2 2 3 1 2 Prometheus, Raga Barbara

1 3 2 3 1 2 Prometheus Neapolitan

3 2 1 1 3 2 Blues scale I, Raga Nileshwari

1 2 2 3 2 2 Ritsu: Japan, Raga Suddha Todi

2 2 1 2 2 3 Arezzo Major Diatonic Hexachord, Raga Kambhoji, Devarangini, Scottish Hexatonic

2 1 2 2 3 2 Minor Hexatonic, Raga Palasi, Manirangu, Nayaki, Yo: Japan, Eskimo Hexatonic 1 (Alaska: King Island)

2 2 2 2 3 1 Eskimo Hexatonic 2 (Alaska: Point Hope)

2 1 4 2 2 1 Hawaiian

2 1 2 1 3 3 Pyramid Hexatonic

1 2 2 1 3 3 Double-Phrygian Hexatonic

2 1 3 1 4 1 Raga Amarasenapriya

2 1 2 4 1 2 Bagesri Raga, Sriranjani, Kapijingla

1 3 3 1 3 1 Bauli Raga

3 1 1 2 3 2 Bhanumanjari Raga, Jog

1 2 3 2 2 2 Bhavani Raga

2 3 2 1 3 1 Bhinna Pancama Raga

2 2 2 1 4 1 Caturangini Raga

1 1 4 1 2 3 Chandrajyoti Raga

1 3 2 1 1 4 Dhavalangam Raga

2 2 1 1 1 5 Dipak Raga

1 2 2 2 3 2 Gandharavam Raga

1 3 1 2 4 1 Gaula Raga

2 1 2 3 3 1 Ghantana Raga, Kaushiranjani (Kaishikiranjani)

3 2 2 1 2 2 Gopikavasantam Raga, Desya Todi, Phrygian Hexatonic

1 2 3 2 3 1 Gurjari Raga Todi

1 3 2 3 2 1 Hamsanandi Raga, Marva, Pancama, Puriya

2 2 1 4 2 1 Hamsa Vinodini Raga

4 1 2 2 2 1 Hari Nata Raga, Genus secundum

1 3 2 2 1 3 Hejjajji Raga

2 4 1 1 2 2 Jaganmohanam Raga

1 4 2 2 2 1 Jivantika Raga

3 3 1 3 1 1 Jivantini Raga, Gaurikriya

4 2 1 1 2 2 Jyoti Raga

1 3 3 1 1 3 Kalagada Raga

1 4 2 1 1 3 Kalakanthi Raga

1 3 1 2 2 3 Kalavati Raga, Ragamalini

4 1 2 1 2 2 Kamalamanohari Raga

4 1 2 2 1 2 Khamas Raga, Baduhari

2 2 3 2 2 1 Kumud Raga, Sankara (Shankara), Lydian Hexatonic

1 3 1 3 3 1 Lalita Raga, Sohini, Hamsanandi, Lalit Bhairav

2 2 3 1 3 1 Latika Raga

3 3 1 2 1 2 Madhukauns Raga (hexatonic)

2 4 1 3 1 1 Malarani Raga(Hamsanada)

1 3 3 2 1 2 Malayamarutam Raga

2 1 4 2 1 2 Manavi Raga

1 3 2 1 4 1 Mandari Raga, Gamakakriya, Hamsanarayani

3 2 2 2 1 2 Manohari Raga

2 2 2 3 2 1 Mruganandana Raga

2 3 2 2 2 1 Nagagandhari Raga

2 2 1 2 4 1 Nalinakanti Raga, Kedaram

2 3 2 1 2 2 Navamanohari Raga

1 4 2 1 3 1 Padi Raga

4 1 2 1 3 1 Paraju Raga, Ramamanohari, Simhavahini, Sindhu Ramakriya, Kamalamanohari

1 4 2 1 2 2 Phenadyuti Raga, Insen, Honchoshi, Niagari: Japan

1 2 2 1 4 2 Honchoshi plagal form: Japan

1 3 1 2 1 4 Purna Pancama Raga, Malahari, Geyahejjajji, Kannadabangala

2 2 1 4 1 2 Rageshri (Rageshwari) Raga, Nattaikurinji

2 1 3 3 2 1 Ranjani Raga, Rangini

3 1 2 1 4 1 Rasamanjari Raga

1 4 2 2 1 2 Rasavali Raga

1 3 1 4 1 2 Rudra Pancama Raga

1 2 4 2 1 2 Salagavarali Raga

2 3 2 3 1 1 Brindabani Sarang Raga, Megh (Megh Malhar)

2 2 1 3 3 1 Sarasanana Raga

2 4 1 2 1 2 Sarasvati Raga

4 1 2 1 1 3 Saravati (Sharavati) Raga

2 1 3 1 3 2 Simharava Raga (Sinharavam)

2 1 2 2 4 1 Sindhura Kafi Raga

2 2 1 2 3 2 Siva Kambhoji Raga, Vivardhini

2 1 2 2 2 3 Suddha Bangala Raga, Gauri Velavali

1 1 3 3 1 3 Raga Suddha Mukhari

1 2 2 2 1 4 Raga Suddha Simantini

2 1 3 1 1 4 Syamalam Raga

3 2 2 1 3 1 Takka Raga

4 1 2 3 1 1 Tilang Raga, Savitri, Brindabani Tilang

2 1 4 1 2 2 Trimurti Raga

1 3 1 4 2 1 Vasanta Raga, Chayavati

1 3 1 3 2 2 Vasantabhairavi Raga

2 1 3 1 2 3 Vijayanagari Raga

1 1 4 1 4 1 Vijayasri Raga

4 2 1 3 1 1 Vijayavasanta Raga

1 2 2 3 3 1 Viyogavarali Raga

4 2 1 2 1 2 Vutari Raga

2 2 2 1 2 3 Yamuna Kalyani Raga, Kalyani Keseri, Ancient Chinese

1 1 4 1 1 4 Messiaen mode 5

4 1 1 4 1 1 Messiaen mode 5 inverse

1 3 1 3 1 3 Messiaen truncated mode 3, Prometheus (Liszt)

3 1 3 1 3 1 Messiaen truncated mode 3 inverse, Major Augmented, Genus tertium

1 2 3 1 2 3 Messiaen truncated mode 2

1 3 2 1 3 2 Messiaen truncated mode 2

2 1 3 2 3 1 Takemitsu Tree Line mode 1

2 1 3 2 2 2 Takemitsu Tree Line mode 2

Heptatonic

1 2 2 1 2 2 2 Locrian G.Mixolydian, G.Hyperdorian, M.Hypophrygian, M.Locrian, Pien chih: China, Makam Lami, Yishtabach: Jewish

2 2 1 2 2 2 1 Major G.Lydian, M.Ionian, M.Hypolydian, Bilaval That, Mela Shankarabharanam, Ghana Heptatonic, Peruvian major, 4th plagal Byzantine, Ararai: Ethiopian, Makam Cargah, Ajam Ashiran, Dastgah Mahur, Dastgah Rast Panjgah

2 1 2 2 2 1 2 Dorian G.Phrygian, M.Dorian, M.Hypomixolydian, Kafi That, Mela Kharaharapriya, Raga Bageshri, Sriraga, Bhimpalasi, Mischung 5, Gregorian nr.8, Eskimo Heptatonic, Yu: China, Hyojo, Oshikicho, Banshikicho: Japan

1 2 2 2 1 2 2 Phrygian G.Dorian, M.Phrygian, G.M.Hypoaeolian, Bhairavi That, Mela Hanumatodi, Raga Asavari (Asaveri), Raga Bilashkhani Todi, In: Japan, Makam Kurd, Gregorian nr.3, Ousak: Greece, Major inverse

2 2 2 1 2 2 1 Lydian, G.Hypolydian, M.Lydian, Kalyan That (Yaman), Mela Mecakalyani, Raga Shuddh Kalyan, Ping, Kung: China

2 2 1 2 2 1 2 Mixolydian G.Hypophrygian, G.Ionian (Iastian), M.Mixolydian, G.M.Hypoionian, Khamaj That, Mela Harikambhoji, Raga Harini, Janjhuti, Surati, Mischung 3, Ching, Shang: China, Gregorian nr.7

2 1 2 2 1 2 2 minor-nat G.M.Hypodorian, G.M.Aeolian, G.Hyperphrygian, Natural Minor, Asavari That, Mela Natabhairavi, Raga Jaunpuri, Raga Adana, Se, Chiao: China, Gregorian nr.2, Makam Buselik, Nihavend, Peruvian minor, Geez, Ezel: Ethiopian, Kiourdi descending: Greece

1 1 3 1 1 3 2 Chromatic Mixolydian

1 3 1 1 3 2 1 Chromatic Lydian, Raga Lalit, Bhankar

3 1 1 3 2 1 1 Chromatic Phrygian

1 1 3 2 1 1 3 Chromatic Dorian, Mela Kanakangi, Raga Kanakambari

1 3 2 1 1 3 1 Chromatic Hypolydian, Purvi That, Mela Kamavardhani, Raga Shri, Pantuvarali, Basant, Kasiramakriya, Suddharamakriya, Puriya Dhanashri, Dhipaka, Pireotikos: Greece

3 2 1 1 3 1 1 Chromatic Hypophrygian, Blues scale III

2 1 1 3 1 1 3 Chromatic Hypodorian, Raga Dvigandharabushini

2 3 1 1 3 1 1 Chromatic Mixolydian inverse

1 1 2 3 1 1 3 Chromatic Phrygian inverse

1 1 3 1 1 2 3 Chromatic Hypophrygian inverse

3 1 1 3 1 1 2 Chromatic Hypodorian inverse

2 1 2 2 2 2 1 Jazz Minor, Melodic Minor ascending,Minor-Major, Mela Gaurimanohari, Raga Patdip, Velavali, Deshi(2), Mischung 1, Hawaiian

2 1 2 2 1 3 1 Mischung 4, Harmonic Minor, Spanish gipsy, Pilu That, Mela Kiravani, Raga Kiranavali, Kirvani, Kalyana Vasantha, Deshi(3), Maqam Bayat-e-Esfahan, Sultani Yakah, Zhalibny Minor

1 3 1 2 2 1 2 Zanjaran, Harmonic Minor inverse, Mela Cakravaka, Raga Ahir Bhairav, Bindumalini, Vegavahini, Makam Hicaz

2 2 1 2 1 3 1 Ethiopian, Harmonic Major, Mela Sarasangi, Raga Haripriya, Mischung 2, Tabahaniotiko: Greece

1 2 1 3 1 2 2 Huzzam Makam, Maqam Saba Zamzam, Phrygian flat 4

2 2 1 3 1 2 1 Ionian sharp 5

2 2 2 2 1 2 1 Lydian Augmented, Lydian sharp 5

1 2 2 1 3 1 2 Locrian natural 6, Maqam Tarznauyn

2 2 1 1 2 2 2 Major Locrian

2 1 2 1 2 2 2 Minor Locrian, Half Diminished, Locrian sharp 2, Minor flat 5

1 2 1 2 2 2 2 Ravel, Super Locrian, Altered, Diminished Whole-tone, Locrian flat 4, Pomeroy

2 1 2 1 2 3 1 Locrian no.2

1 2 1 2 2 1 3 Ultra Locrian, Mixolydian sharp 1

1 2 2 1 2 1 3 Locrian double-flat 7

2 3 1 2 1 2 1 Nohkan Flute scale: Japan

2 1 1 3 1 2 2 Sabach (Sambah): Greece

2 1 2 1 3 1 2 Makam Karcigar, Maqam Nahawand Murassah, Dorian flat 5, Kiourdi ascending, Kartzihiar: Greece

2 1 2 1 3 2 1 Jeths' mode

1 1 3 2 1 2 2 Ratnangi Mela, Raga Phenadyuti

1 1 3 2 1 3 1 Ganamurti Mela, Raga Ganasamavarali

1 1 3 2 2 1 2 Vanaspati Mela, Raga Bhanumati

1 1 3 2 2 2 1 Manavati Mela, Raga Manoranjani

1 1 3 2 3 1 1 Tanarupi Mela, Raga Tanukirti

1 2 2 2 1 1 3 Senavati Mela, Raga Senagrani, Malini

1 2 2 2 1 3 1 Neapolitan Minor, Mela Dhenuka, Raga Dhunibinnashadjam, Maqam Shahnaz Kurdi

1 2 2 2 2 2 1 Neapolitan Major, Phrygian Major, Mela Kokilapriya, Raga Kokilaravam

1 2 2 2 2 1 2 Jazz Minor inverse, Phrygian-Mixolydian, Raga Natabharanam, Ahiri Todi, Mela Natakapriya,

1 2 2 2 3 1 1 Rupavati Mela

1 3 1 1 2 3 1 Lalita Raga, Chromatic Hypolydian inverse, Raga Suddha Pancama

1 3 1 2 1 1 3 Gipsy Hexatonic, Mela Gayakapriya, Raga Kalakanti

1 3 1 2 1 2 2 Dorico Flamenco: Spain, Mela Vakulabharanam, Alhijaz: Arabic, Raga Jogiya, Vativasantabhairavi, Ahava Rabba: Jewish, Maqam Humayun, Hitzaz: Greece, Harmonic Major inverse, Phrygian Dominant

1 3 1 2 1 3 1 Persian, Major Gipsy, Double Harmonic Major, Bhairav That, Mela Mayamalavagaula, Raga Paraj, Kalingada, Byzantine Liturgical Chromatic, Hitzazkiar: Greece, Maqam Zengule, Hijaz Kar, Suzidil

1 3 1 2 2 2 1 Suryakanta Mela, Bhairubahar That, Raga Supradhipam, Sowrashtram

1 3 1 2 3 1 1 Hatakambari Mela, Raga Jeyasuddhamalavi

2 1 2 1 1 3 2 Blues Modified

2 1 2 2 1 1 3 Mela Jhankaradhvani, Raga Jhankara Bhramavi

2 1 2 2 3 1 1 Mela Varunapriya, Viravasantham

2 2 1 2 1 1 3 Keseri Raga, Mela Mararanjani

2 2 1 2 1 2 2 Mischung 6, Mixolydian flat 6, Major-Minor, Mela Carukesi, Raga Charukeshi, Tarangini

2 2 1 2 3 1 1 Samanta, Mela Naganandini, Raga Nagabharanam

3 1 1 2 1 1 3 Yagapriya Mela, Raga Kalahamsa

3 1 1 2 1 2 2 Mela Ragavardhani, Raga Cudamani

3 1 1 2 1 3 1 Sengiach: Greece, Mela Gangeyabhusani, Raga Gangatarangini

3 1 1 2 2 1 2 Bluesy R&R, Mela Vagadhisvari, Raga Bhogachayanata, Nandkauns, Ganavaridhi

3 1 1 2 2 2 1 Houzam: Greece, Mela Sulini, Raga Sailadesakshi, Raga Trishuli

3 1 1 2 3 1 1 None Raga, Mela Calanata, Chromatic Dorian inverse

1 1 4 1 1 1 3 Salaga Mela

1 1 4 1 1 2 2 Jalarnava Mela

1 1 4 1 1 3 1 Varali Raga, Mela Jhalavarali, Jinavali

1 1 4 1 2 1 2 Navanitam Mela, Raga Nabhomani

1 1 4 1 2 2 1 Pavani Mela, Raga Kumbhini

1 1 4 1 3 1 1 Mela Raghupriya, Raga Ravikriya, Ghandarva

1 2 3 1 1 1 3 Girvani Raga, Mela Gavambodhi

1 2 3 1 1 2 2 Bhavapriya Mela, Raga Bhavani, Kalamurti

1 2 3 1 1 3 1 Todi That, Mela Shubhapantuvarali, Raga Multani, Gamakasamantam, Chromatic Lydian inverse

1 2 3 1 2 1 2 Mela Sadvidhamargini, Raga Sthavarajam, Tivravahini

1 2 3 1 2 2 1 Sauviram,Raga, Mela Suvarnangi

1 2 3 1 3 1 1 Divyamani Mela

1 3 2 1 1 1 3 Foulds' Mantra of Will scale, Mela Dhavalambari

1 3 2 1 1 2 2 Namanarayani Mela, Raga Narmada, Pratapa

1 3 2 1 2 1 2 Petrushka chord, Mela Ramapriya, Raga Ramamanohari, Romanian Major

1 3 2 1 2 2 1 Marva That, Mela Gamanasrama, Raga Partiravam, Puriya, Puriya Kalyan, Sohani, Peiraiotikos: Greece

1 3 2 1 3 1 1 Visvambhari Mela, Raga Vamsavathi

1 3 2 2 2 1 1 Verdi's Enigmatic ascending

1 3 1 3 2 1 1 Verdi's Enigmatic descending

2 1 3 1 1 1 3 Mela Syamalangi, Raga Shyamalam

2 1 3 1 1 2 2 Camara Raga, Mela Sanmukhapriya, Chinthamani

2 1 3 1 1 3 1 Minor Gipsy, Mela Simhendramadhyama, Raga Madhava Manohari, Maqam Nawa Athar, Hisar, Double Harmonic Minor, Hungarian Minor, Niavent: Greece

2 1 3 1 2 1 2 Nigriz: Greece, Mela Hemavati, Raga Desisimharavam, Maqam Nakriz, Tunisian, Hedjaz, Misheberekh: Jewish, Souzinak (Peiraiotiko Minore), Dorian sharp 4, Kaffa, Gnossiennes

2 1 3 1 2 2 1 Lydian Diminished, Mela Dharmavati, Raga Arunajualita, Dumyaraga, Madhuvanti, Ambika

2 1 3 1 3 1 1 Nitimati Mela, Raga Nisada, Kaikavasi

2 2 2 1 1 1 3 Kantamani Mela, Raga Kuntala, Srutiranjani

2 2 2 1 1 2 2 Lydian Minor, Mela Risabhapriya, Raga Ratipriya

2 2 2 1 1 3 1 Latangi Mela, Raga Gitapriya, Hamsalata

2 2 2 1 2 1 2 Bartok, Lydian Dominant, Mela Vacaspati, Raga Bhusavati, Overtone, Lydian-Mixolydian

2 2 2 1 3 1 1 Mela Citrambari, Raga Chaturangini

2 2 1 4 1 1 1 Ragesri Raga

2 3 2 2 1 1 1 Sorati Raga, Sur Malhar

3 1 2 1 1 1 3 Sucaritra Mela, Raga Santanamanjari

3 1 2 1 1 2 2 Mela Jyotisvarupini, Raga Jotismatti

3 1 2 1 1 3 1 Mela Dhatuvardhani, Raga Dhauta Pancama, Devarashtra

3 1 2 1 2 1 2 Hungarian Major, Mela Nasikabhusani, Raga Nasamani

3 1 2 1 2 2 1 Kosalam Mela, Raga Kusumakaram, Lydian sharp 2

3 1 2 1 3 1 1 Mela Rasikapriya, Raga Rasamanjari, Hamsagiri

3 1 2 2 1 2 1 Aeolian flat 1

4 1 2 2 1 1 1 Madhuri Raga

1 3 1 1 3 1 2 Oriental, Raga Ahira-Lalita, Minor Gipsy inverse, Tsinganikos: Greece

2 2 2 2 2 1 1 Leading Whole-tone

Octatonic

1 3 1 1 1 2 2 1 Bhatiyar Raga

2 1 3 1 1 1 1 2 Cintamani Raga

2 1 2 2 2 1 1 1 Mian Ki Malhar Raga, Bahar, Sindhura

2 1 2 2 1 1 1 2 Mukhari Raga, Anandabhairavi, Deshi, Gregorian nr.1, Dorian/Aeolian mixed

1 3 1 1 1 1 3 1 Ramkali Raga (Ramakri)

1 3 1 2 1 1 2 1 Saurashtra Raga

2 1 1 1 2 2 1 2 Minor Bebop, Raga Zilla, Mixolydian/Dorian mixed

2 2 1 2 2 1 1 1 Genus diatonicum, Dominant Bebop, Raga Khamaj, Desh Malhar, Alhaiya Bilaval, Maqam Shawq Awir, Gregorian nr.6, Major/Mixolydian mixed, Chinese Eight-Tone, Rast: Greece

2 2 1 2 1 1 2 1 Bebop Major, Altered Mixolydian

2 1 2 1 1 2 1 2 Blues scale II

2 1 2 1 1 1 3 1 Algerian

1 2 1 1 2 1 2 2 Spanish Phrygian

1 2 1 1 1 2 2 2 Espla's scale, Eight-tone Spanish

1 2 2 1 1 1 3 1 Half-diminished Bebop

1 2 3 1 1 2 1 1 Neveseri: Greece

2 1 2 1 2 1 2 1 Diminished, Modus conjunctus, Messiaen mode 2 inverse, Whole-Half step scale

2 2 1 1 1 2 2 1 Ishikotsucho: Japan, Raga Yaman Kalyan, Chayanat, Bihag, Hamir Kalyani, Kedar, Gaud Sarang, Genus diatonicum veterum correctum, Gregorian nr.5, Kubilai's Mongol scale, Major/Lydian mixed

1 3 1 1 2 2 1 1 Verdi's Scala enigmatica

2 1 2 2 1 1 2 1 Zirafkend: Arabic

1 1 1 2 2 2 1 2 Adonai Malakh: Jewish

1 2 1 2 2 1 2 1 Magen Abot: Jewish

2 1 2 2 1 2 1 1 Gregorian nr.4, Maqam Nahawand, Raga Suha (Suha Kanada), Utility Minor

1 1 1 2 2 1 3 1 Harmonic and Neapolitan Minor mixed

1 3 1 2 1 2 1 1 Hijaz Maqam

1 2 1 1 1 3 1 2 Shadd'araban Maqam

1 2 1 2 1 2 1 2 Octatonic, Messiaen mode 2, Dominant Diminished, Diminished Blues, Half-Whole step scale

1 1 1 3 1 1 1 3 Messiaen mode 4

3 1 1 1 3 1 1 1 Messiaen mode 4 inverse

1 1 2 2 1 1 2 2 Messiaen mode 6

2 2 1 1 2 2 1 1 Messiaen mode 6 inverse

1 1 1 2 2 1 2 2 Phrygian/Aeolian mixed

1 2 2 1 1 1 2 2 Phrygian/Locrian mixed

1 2 2 1 1 2 2 1 Van der Horst Octatonic

1 2 2 1 2 2 1 1 Prokofiev

1 2 1 2 1 2 2 1 Shostakovich

Enneatonic

1 1 2 1 1 2 1 1 2 Messiaen mode 3, Tsjerepnin

2 1 1 2 1 1 2 1 1 Messiaen mode 3 inverse

2 1 2 2 1 1 1 1 1 Pilu Raga, Full Minor

2 1 1 1 2 2 1 1 1 Malgunji Raga, Ramdasi Malhar, Major/Dorian mixed

2 2 1 2 1 1 1 1 1 Pahadi Raga

2 1 1 1 1 1 2 1 2 Blues Enneatonic

2 1 2 1 1 1 1 1 2 Kiourdi: Greece

2 2 1 1 1 2 1 1 1 Taishikicho: Japan, Ryo: Japan, Raga Chayanat, Lydian/Mixolydian mixed

1 2 1 1 2 1 1 2 1 Genus chromaticum

1 2 1 1 2 1 2 1 1 Moorish Phrygian

1 1 2 1 1 1 2 1 2 Youlan scale: China

1 1 1 2 2 1 1 1 2 Chromatic and Diatonic Dorian mixed

1 1 2 1 2 1 1 2 1 Chromatic and Permuted Diatonic Dorian mixed

2 1 1 1 2 1 1 1 2 Houseini: Greece, Modes of Major Pentatonic mixed

---

Decatonic

1 1 1 1 2 1 1 1 1 2 Messiaen mode 7

2 1 1 1 1 2 1 1 1 1 Messiaen mode 7 inverse

2 1 1 1 2 1 1 1 1 1 m/M_mix, Major/Minor mixed

2 1 1 1 1 1 2 1 1 1 Minor Pentatonic with leading tones

1 1 1 1 1 2 1 2 1 1 Raga Sindhi-Bhairavi

1 1 2 1 1 1 1 2 1 1 Symmetrical Decatonic

Chromatic

1 1 1 1 1 1 1 1 1 1 1 1 Twelve-tone Chromatic

---

adds tng 05/11/2006_16/03/2007

1 11 semi

2 10 second

3 9 min-third

4 8 Maj-third

5 7 fourth Honchoshi: Japan

6 6 augmented_fourth

7 5 fift Niagari: Japan

8 2 sixte-minor

9 3 sixte

10 2 seventh

11 1 maj7

---

tritonics_2

4 3 5 Maj1

3 5 4 m3-m6

5 4 3 Fourth-6

3 4 5 min1

4 5 3 M3-6

5 3 4 Fourth-m6

7 2 3 Fift-6

3 7 2 m3-m7

2 3 7 sus2-4

2 7 3 sus2-6

7 3 2 Fift-m7

3 2 7 m3-sus4

7 1 4 Fift-aug1

1 4 7 b2-sus4

4 7 1 M3-M7

7 4 1 Fift-M7

4 1 7 M3-sus4

1 7 4 m2-m6

6 3 3 dim1

3 3 6 dim2

3 6 3 dim3

6 4 2 b5-m7

4 2 6 M3-b5

2 6 4 m2-m6

6 2 4 b5-m6

2 4 6 m2-b5

4 6 2 M3-m7

6 5 1 b5-M7

5 1 6 sus4-b5

1 6 5 m2-5

6 1 5 #4-5

1 5 6 m2-b5

5 6 1 sus4-M7

8 2 2 aug1

2 2 8 sus2-M3

2 8 2 sus2-M7

8 3 1 #5-M7

3 1 8 m3-M3

1 8 3 m2-6

8 1 3 #5-6

1 3 8 m2-M3

3 8 1 m3-M7

9 2 1 Sixth-M7

1 9 2 m2 6

9 2 1 sixte M7

2 9 1 M2 M7

9 1 2 sixte m7

1 2 9 m2 m3

10 1 1 seven2

1 10 1 m2 M7

4 4 4 aug

1 1 10 chroma

---

tetra_2

4 3 4 1 Major7

3 4 1 4 Major7-r1

4 1 4 3 Major7-r2

1 4 3 4 Major7-r3

3 4 3 2 minor7

4 3 2 3 minor7-r1

3 2 3 4 minor7-r2

2 3 4 3 minor7-r3

4 3 3 2 dominant

3 3 2 4 dominant-r1

3 2 4 3 dominant-r2

2 4 3 3 dominant-r3

3 4 4 1 m7M

4 4 1 3 m7M-r1

4 1 3 4 m7M-r2

1 3 4 4 m7M-r3

3 3 4 2 m7_5b

3 4 2 3 m7_5b-r1

4 2 3 3 m7_5b-r2

2 3 3 4 m7_5b-r3

3 3 3 3 dim

2 4 4 2 tng_mirror1

4 2 2 4 tng_mirror1a

2 2 4 4 tng_2244

4 4 2 2 tng_4422

5 1 1 5 tng_mirror2

1 5 5 1 tng_mirror2a

5 5 1 1 tng_5511

1 1 5 5 tng_1155

9 1 1 1 sixte+chroma

1 9 1 1 chroma2

1 1 9 1 chroma3

1 1 1 9 chroma4

8 1 2 1 augsym

1 2 1 8 symaug

6 1 4 1 tritonix

1 4 1 6 tritonix2

4 1 6 1 tritonix3

6 1 3 2 tritonax

1 3 2 6 tritonax2

3 2 6 1 tritonax3

2 6 3 1 tritonax4

6 1 1 4 tritonox

1 1 4 6 tritonox2

1 4 6 1 tritonox3

4 6 1 1 tritonox4

6 2 2 2 tritonux

2 2 2 6 tritonux2

2 2 6 2 tritonux3

2 6 2 2 tritonux4

4 1 1 6 M3chroma

7 2 1 2 tetra1

7 3 1 1 tetra2

7 1 3 1 tetra3

1 7 1 3 tetra3a

---

try??

1 2 3 4 2 suite

```
5 1 2 1 2 1 sym

1 1 2 2 3 3 tng_1

2 2 3 3 1 1 tng_1A

3 3 1 1 2 2 tng_1B

2 2 1 1 3 3 tng_2

1 1 3 3 2 2 tng_2A

3 3 2 2 1 1 tng_2B

3 2 1 1 2 3 mirroir

1 11 diton_T

1 1 10 triton_T

1 1 1 9 tetra_T

1 1 1 1 8 penta_T

1 1 1 1 1 7 hexa_T

1 1 1 1 1 1 6 hepta_T

1 1 1 1 1 1 1 5 octa_T

1 1 1 1 1 1 1 1 4 ennea_T

1 1 1 1 1 1 1 1 1 3 deca_T

1 1 1 1 1 1 1 1 1 1 2 eleven_T
```

Sgorpi's Mean Bassdrum Machine is a simple generator (thus goes into Gear/Generators)
with 2 oscillators and a pitch envelope...
Most parameters should speak for themselves, but for a reference:

```
Max frequency   - Top frequency of the envelope
Min frequency   - Bottom frequency of the envelope
Length          - Length of the sound
Waveform        - Waveform of the main oscillator
Doubletone      - Pitch multiplycation of second oscillator (like if osc1 is
                  playing at 50Hz, doubletone is 2.000, osc2 will play 100Hz)
Double start    - Start volume of the double tone (linear progression to
                  end volume)
Double end      - End volume of the double tone
Dbl Waveform    - doubletone waveform
Dbl start phase - starting phase of the doubletone wave

Spline curve (from the rightclick menu on the envelope)
                - draw a nice curved line between the points in stead of
                  a linear one
```

The envelopes can be loaded and saved to files, the .BEF format of the envelope.ocx control in buzz can be loaded too.
Left click to add an envelope point, rightclick on a point to choose 'delete', rightclick outside a point to access other envelope functions (like loading and saving and spline curves).
The vertical grid lines indicate tick positions (possibly not a 100% accurate). The horizontal gridlines indicate a doubling of frequency from the Min Frequency to the Max Frequency. (A doubling means an octave increase, thus a logarithmic scale.)

Happy music making!

Hedde 'Sgorpi' aka 'DJ Tex-nd' Bosman

sgorpi@totalegekte.com / sgorpi@gmail.com

ps: add in your index.txt:

```
 /Generators
  /Kick
  sgorpi_mbm, Sgorpi's Mean Bass Machine
```

Soft Tone Generator 1

By Steve Horne, 3 March 1999

This generator is a simple synthesizer, aimed at producing 'soft' instrument sounds, similar to wind instruments. Its key feature is that the amplitude never makes sudden jumps - even when a new note is played - but always follows a natural decay curve.

The output waveform can be distorted using a fractal algorithm similar to that used in my Fractal Effect machine. The key difference is that the distortion is applied to the raw waveform before the amplitude scaling is done. Thus the distortion remains the same whatever amplitude a note is played at. It is therefore more of a waveform reshaping than a true distortion.

Version Record

30 March 1999

Original version was built on the assumption that middle C is 440 Hz, which is wrong. 440 Hz is in fact Middle A.

Furthermore, I had not realised that Buzz separates the note numbers into an octave and semitone, stored in separate nibbles. I had assumed a MIDI-like linear simitone sequence.

This version fixes both of these errors.

3 March 1999

Initial Release

Global Parameters

Waveform

This selects one of the following waveforms...

- Sine
- Triangle
- Double Triangle
- Hex
- Square
- Ramp
- Alternating Sawtooth

The precise shape of each of these is less important than its sound, so I won't bother with any graphics.

Attack

The rate at which the amplitude increases to the target amplitude when a note is on, specified as a half life in milliseconds.

Note - if the target amplitude is decreased while a note is playing, the note amplitude will decay to the new amplitude but the decay will occur at the 'attack' rate.

Decay

The rate at which the amplitude decays to zero when a note is off, specified as a half life in milliseconds.

Because all notes are sustained indefinitely, the note off command (the '1' key) will frequently be needed.

Effect

This is a coninuous value, which (once scaled) ranges from 0.0 to 9.0.

If this parameter is set to 1.0, the output will follow the input exactly.

If this parameter is set below 1.0, the function tends to 'fatten' input waveforms - triangles become sine waves (or square waves if Depth is high enough).

If this parameter is set to a high value, the function becomes chaotic. At 9.0 (with sufficiently high Depth), the output becomes white noise.

Depth

This parameter is the number of times that the distortion is applied.

If Depth is zero, the input is not changed.

If Depth has a low value, the algorithm reshapes the waveforms.

As Depth is increased (with a sufficiently high Effect value), more high harmonics appear until the signal eventually becomes white noise.

The fractal depth is limited to 10 (rather than the 32 supported by the fractal effect) because of the possibility of multiple tracks being handled at once.

Track Parameters

Note

Triggers notes.

If a new note is started while the previous note is still playing, the pitch changes immediately. This can sound similar to a wind instrument changing notes without stopping between - an effect such as Geonik's 'expression 2' should be used to make it more realistic.

Volume

This provides a target amplitude for the note being played. It is effectively the sustain level of the note. It can be changed at any time, and the note will smoothly follow the target amplitude.

This is my first generator, and I have treated it as a learning exercise rather than a serious project. There is considerable room for optimisation if there is any demand. However, as I mentioned for the Fractal Effect machine, the fractal algorithm I have used is inherently slow.

If you have any comments, please e-mail them to steve@lurking.demon.co.uk.

Soft Tone Generator 2

By Steve Horne, 26 March 1999

This generator is a simple synthesizer, which is heavily based on my previous Soft Tone Generator 1. Its main enhancement is that the 'fractal effect' parameter is now specified as a high and low range. The actual value used in synthesizing a tone is selected from in this range, based on the current amplitude.

The result is that the waveform varies with the amplitude of the tone. In the early attack and the late decay stages of a note, the waveform can differ significantly from the sustain stage. Furthermore, notes played quietly will also have different waveforms.

Global Parameters

Waveform

This selects one of the following waveforms...

1. Sine
2. Triangle
3. Double Triangle
4. Hex
5. Square
6. Ramp
7. Alternating Sawtooth

The precise shape of each of these is less important than its sound, so I won't bother with any graphics.

Attack

The rate at which the amplitude increases to the target amplitude when a note is on, specified as a half life in milliseconds.

Note - if the target amplitude is decreased while a note is playing, the note amplitude will decay to the new amplitude but the decay will occur at the 'attack' rate.

Decay

The rate at which the amplitude decays to zero when a note is off, specified as a half life in milliseconds.

Because all notes are sustained indefinitely, the note off command (the '1' key) will frequently be needed.

Effect Low

This is a coninuous value, which (once scaled) ranges from 0.0 to 9.0.

If this parameter is set to 1.0, the output will follow the input exactly.

If this parameter is set below 1.0, the function tends to 'fatten' input waveforms - triangles become sine waves (or square waves if Depth is high enough).

If this parameter is set to a high value, the function becomes chaotic. At 9.0 (with sufficiently high Depth), the output becomes white noise.

The low effect value is only used directly when the amplitude is zero. The effect value used also depends on the Effect High parameter and the current note amplitude.

Effect High

This is the same as Effect Low, but specifies the value to use when the amplitude of the note is at its maximum.

Depth

This parameter is the number of times that the distortion is applied.

If Depth is zero, the input is not changed.

If Depth has a low value, the algorithm reshapes the waveforms.

As Depth is increased (with a sufficiently high Effect value), more high harmonics appear until the signal eventually becomes white noise.

The fractal depth is limited to 10 (rather than the 32 supported by the fractal effect) because of the possibility of multiple tracks being handled at once.

Track Parameters

Note

Triggers notes.

If a new note is started while the previous note is still playing, the pitch changes immediately. This can sound similar to a wind instrument changing notes without stopping between - an effect such as Geonik's 'expression 2' should be used to make it more realistic.

Volume

This provides a target amplitude for the note being played. It is effectively the sustain level of the note. It can be changed at any time, and the note will smoothly follow the target amplitude.

If you have any comments, please e-mail them to steve@lurking.demon.co.uk.

Synthrom Sinus v2

The changes

Version 2 supports western 12tone based notes, is thus incompatible to version 1 (obsolete by now), however you may still create "odd" frequencies, see below for details. Another new feature is custom waveform support, you may even exchange custom waveforms (*.ssw) with other buzz users.

The theory

This buzz-machine is a waveform generator from a mathematical view.
You select one of the given functions with a certain frequency, and then select more functions (total maximum is 10)
to mix with the first one (in a custom way and with custom influence percentages). This also means that additional tracks are
not to be seen as separate signals but as stages of manipulation (not commutable, changing their order might cause other sounds) that can be triggered as you wish.

knowledge is power

The parameters

Waveform:	choose from the following: **sin** the base of everything ;) **sin.square** a square wave that starts at 0 by default **saw.1**a wave that raises smooth from base (0) to 1 and then immidately drops to base again /I **saw.2** dito, but the base is -1 **tri.1** formed like a pyramid, base is 0 **tri.2** dito, but the base is -1 **noise** creates a frequency-dependend - well, ehrm - noise
Offset:	selects the angle at which the function's calculation will start, eg, selecting sine with offset 90° will effectively result in a cos-waveform, as the sine of 90° is 1 and the playback will start there
Filter:	manipulates the function's values

invert multiplies the value with -1
pos cuts all negative values
neg cuts all positive values
invert pos multiplies all positive values with -1

invert pos for a basic sine

invert neg multiplies all negative values with -1

Volume: adjusts the amplitude, thus also important for the mixing:
imagine two "div"-mixed waves, with the second being louder - the result will be quieter.

Frequency: ranges from 40 to 11,110Hz; what you type is what you get
and it's correct, even if you use buzz at an other sampling rate than 44khz.
The righmost value (FFFF) enables note support. With every other value, the note row of a track simply works as a trigger.

Sweep: this value is constantly (read again and again) being added to the frequency, leading to "sweeps"
can also be used to generate percussions, try e.g. an 80hz square with sweep -1 :)

MixType: sets the combination mode for this track's and the previous function value (thus obsolete for track 1)
max the bigger of the two values wins
avg both values are added and divided by two
add both values are added
mul both values are multiplied (gets loud!)
div value one is divided by value two
and boolean operation, each bit of the new wave is and(oldbit,thistrackbit) (only 1 if both are 1)
or boolean operation, each bit of the new wave is or(oldbit,thistrackbit) (only 0 if both are 0)

MixValue: multiplies one value with 0<=x<=1, e.g. to add with 50%:100% means new=0.5*old +thistrack

Attack: the time [0-11888ms] this track's waveform needs to reach its set volume, read fade-in

Sustain: the time [0-11889ms] this track's waveform is to be played
if you e.g. have a nice sound made with 4 tracks, all 4 sustain values should be the same...

Release: the time [0-11888ms] this track's waveform needs to reach level 0 from its set volume, read fade-out
note that 0ms will eventually lead to a click-sound at the end of the sustain-time, also setting sustain longer than the time between two triggers will cause clicks as the release then doesn't even start

The edit window

there you go...

Remember to use the "Accept" button to refresh synthesis and hear your changes.
If you find it kinda hard to draw a waveform in here you may use the "Load Bitmap" feature. The bmp-resolution does not matter, as long as it is vertically centered. Our aim with the "Load/ Save Settings" function was to enable waveform swapping and maybe setting up a global standard supported by other future synths...a how-to-implement text is included in this release.

request

We're totally not into midi. We'd be glad if anybody could mail us how to implement necessary features (midi channel attribute?) and beta-test the result.

credits

coding, additional ideas: 1mT (homepage)
idea, betatesting, helpfile: Geroyche (homepage)

keep buzz-updated:

the latest program files
the latest machines
another nice machine page
an up-to-date index.txt
share problems, tips, visions

Synthrom´s Sinus

The theory

This buzz-machine is a waveform generator from a mathematical view.
You select one of the given functions with a certain frequency, and then select more functions (total maximum is 10)
to mix with the first one (in a custom way and with custom influence percentages). This also means that additional tracks are
not to be seen as separate signals but as stages of manipulation (not commutable, changing their order might cause other sounds) that can be triggered as you wish.

knowledge is power

The parameters

Waveform:	choose from the following: **sin** the base of everything ;) **sin.square** a square wave that starts at 0 by default **saw.1**a wave that raises smooth from base (0) to 1 and then immidately drops to base again /I **saw.2** dito, but the base is -1 **tri.1** formed like a pyramid, base is 0 **tri.2** dito, but the base is -1 **noise** creates a frequency-dependend - well, ehrm - noise
Begin:	selects the angle at which the function's calculation will start, eg, selecting sine with begin at 90° will effectively result in a cos-waveform, as the sine of 90° is 1 and the playback will start there
Filter:	manipulates the function's values **invert** multiplies the value with -1 **pos** cuts all negative values **neg** cuts all positive values **invert pos** multiplies all positive values with -1 invert pos for a basic sine **invert neg** multiplies all negative values with -1
Volume:	adjusts the amplitude, thus also important for the mixing: imagine two "div"-mixed waves, with the second being louder - the result will be quieter.

Frequency:	ranges from 40 to 11,111Hz; what you type is what you get and it's correct, even if you use buzz at an other sampling rate than 44khz
Sweep:	this value is constantly (read again and again) being added to the frequency, leading to "sweeps" can also be used to generate percussions, try e.g. an 80hz square with sweep -1 :)
MixType:	sets the combination mode for this track's and the previous function value (thus obsolete for track 1) **max** the bigger of the two values wins **avg** both values are added and divided by two **add** both values are added **mul** both values are multiplied (gets loud!) **div** value one is divided by value two **and** boolean operation, each bit of the new wave is and(oldbit,thistrackbit) (only 1 if both are 1) **or** boolean operation, each bit of the new wave is or(oldbit,thistrackbit) (only 0 if both are 0)
MixValue:	multiplies one value with 0<=x<=1, e.g. to add with 50%:100% means new=0.5*old +thistrack
Attack:	the time [0-11888ms] this track's waveform needs to reach its set volume, read fade-in
Sustain:	the time [0-11889ms] this track's waveform is to be played if you e.g. have a nice sound made with 4 tracks, all 4 sustain values should be the same...
Release:	the time [0-11888ms] this track's waveform needs to reach level 0 from its set volume, read fade-out note that 0ms will eventually lead to a click-sound at the end of the sustain-time, also setting sustain longer than the time between two triggers will cause clicks as the release then doesn't even start

credits

coding, additional ideas: 1mT (homepage)
idea, betatesting, helpfile, demo: Geroyche (homepage)

keep buzz-updated:

the latest program files
the latest machines
an up-to-date index.txt
share problems, tips, visions

theese are the original dx7 algorithms. * means feedback. more will be coded soon.

```
    6*       6           6
    |*       |           |
    5        5           5    3   6*   3   6*          6*          6           6
    |        |           |    |   |*   |   |           |*          |           |
2   4    2*  4    2   3*  4    2   5    2   5    2   4   5    2   4*  5    2*  4   5
|   |    |*  |    +---+*--+    |   |    |   |    |   +-+-+    |   +*+-+    |*  +-+-+
1   3    1   3        1        1   4    1   4    1     3      1     3      1     3
+---+    +---+        +        +---+    +---+*   +-----+      +-----+      +-----+
Alg#1    Alg#2      Alg#18    Alg#3    Alg#4     Alg#7       Alg#8       Alg#9

3*          3                5   6*      5   6        4   6*      4   6    3
|*          |              +-+-+*      +-+-+        |   |*      |   |    |
2   5   6   2   5   6*   2     4     2*    4      2   3   5   2*  3   5    2     6*
|   +-+-+   |   +-+-+*   |     |     |*    |      +---+---+   +*--+---+    |   +-+*+
1     4     1     4      1     3     1     3          1           1        1   4   5
+-----+     +-----+      +-----+     +-----+          +           +        +---+---+
 Alg#10      Alg#11       Alg#14      Alg#15          Alg#16      Alg#17      Alg#19

    5*              5*
    |*              |*
2   4               4       2   4   6*  2   4   6*   2*  4   5   6   2   4   5   6*
|   |               |       |   |   |*  |   |   |    |*  +---+---+   |   +---+---+*
1   3   6   1   2   3   6   1   3   5   1   3   5    1       3       1       3
+---+---+   +---+---+---+   +---+---+   +---+---+*   +-------+       +-------+
  Alg#28       Alg#30        Alg#5       Alg#6        Alg#12          Alg#13

  3*    5   6      3*      6      2       6*          3     6*              6*
+-+*+   +-+-+    +-+*+   +-+-+    |    +---+*--+       |   +-+*+         +---+*--+
1   2     4      1   2   4   5    1   3   4   5    1   2   4   5   1   2   3   4   5
+---+-----+      +---+---+---+    +---+---+---+    +---+---+---+   +---+---+---+---+
  Alg#20           Alg#21           Alg#22           Alg#23          Alg#24

          6*         3   5   6*       3*  5   6           4   6*
        +-+*+        |   +-+-+*       |*  +-+-+           |   |*
1   2   3   4   5    1   2     4      1   2     4     1   2   3   5
+---+---+---+---+    +---+-----+      +---+-----+     +---+---+---+
    Alg#25             Alg#26           Alg#27          Alg#29

            6*
            |*
```

```
1   2   3   4   5    1   2   3   4   5   6*
+---+---+---+---+    +---+---+---+---+---+*
    Alg#31               Alg#32
```

usr n+1 peer controller

Intro:

This is a peer controller for those tasks that need precision, like selecting waveforms, transposing etc. Not a replacement for the original peerCtrl by BTDsys - whose peerlib i used of course, thanks Ed, you're the man, please, everyone, check out his music, although he will get famous anyway!

Installation:

As usual with peer controllers, unzip the whole package to your gear\generators folder.

Parameters:
----

global:
 - Type: Selects between the value parameters with different resolutions.

 - Value 16: Value parameter, resolution: 0..15.

 - Value 32: Value parameter, resolution: 0..31.

 - Value 255: Value parameter, resolution: 0..254.

 - Value 65535: Value parameter, resolution: 0..65534.

track:
 - Target Track: Selects which track in the controllee to control, if a track parameter is selected.

 - Multiplier: The selected value is multiplied by this before it is applied to the controlled parameter.

 - Offset: Offset (and the minimum value of the controlled parameter) is added to the multiplied value before it is applied to the controlled parameter.

This is the third beta release of Vibrasynth, things are starting to take shape, parameters will not change from now on. But features will be added in the empty parameter spaces.

I'll have a final release of it real soon too.

:)

CyanPhase
blakee@rovoscape.com

vII MidiOut Help/Documentation (version 1h)

Contents

1.Introduction/Purpose

- This is a first implementation of a MidiOut Machine for BUZZ. It is to regard as beta and will maybe heavily changed in the future, although I already tried to prepare all needed parameters and designed vMidiOut to be fit for everything else than SysX (for all still missing things look at the future plans).
- I guess a MIDI out machine generally isn't that suited for distributed songs, because it will behave and sound different on every computer.
- The main purposes may be using BUZZ to make music with all your MIDI-equipment (and mixing all together by other means than BUZZ itself) and the possiblility to sync another device or programm to BUZZ (BUZZ itself currently can't adapt to the timing of another timing source/ master).

Some remarks:

- *Synchronizing with a MIDI sequencer and external devices should work fine now, also it may be possible to share a DirectX wavedevice with softsynths like Rebirth.*
- *I doubt the Wavein machine already is adequate for the mixing task (to get any external sounds back into a final mix), but haven't tried it yet. I don't know how the VirtualAudioCable fits in yet, but Hubis MidiLoopback Device and MidiCable sure help a lot.*

More exotic usages would be

- *controlling BUZZ parameters via MIDI, either by the MidiOut machine itself or with a (synchronized) sequencer, the latter would finally allow to change parameters live and record this with the sequencer ;)*
- *Concerning the MIDI conversion and merging options of m2buzz there is now a possibility to synchronize a sequencer with BUZZ (either with MID clocks or MTC) playing some rhythm loops or backing parts of a song , record something live with the sequencer and convert and merge this together with the rest of the song.*
- *Another fine usage is the recently discovered MIDI playability of the ES-9, so you could really record the played notes with a synchronized sequencer and convert the stuff to ES-9. (This applies even better to the new version of Geonik's PrimiFun).*
- *Since m2buzz v3e you can also just convert MIDI files for use with the MIDIout machine, although controllers and tempo changes aren't processed yet.*

2.Basic Usage

To get vMidiOut working:

- put it into the **Gear/Generators** directory and
- choose the devices you want to use in the **BUZZ Preferences->Midioutput** (don't forget this!), they will be opened by BUZZ and stay this until you terminate BUZZ or uncheck them.
- Set the device for a vMidiOut machine in the options dialog, since version 1d you can just do this by name without remembering the device number (which will nonetheless be stored as normal BUZZ attribute of the machine). To adress several devices you need more MidiOut machines (since version 1b).
- You also have to connect every machine to the master and set a volume greater zero, although it doesn't produce any sound, because otherwise the machine won't be activated by BUZZ (at least with the method attribute set to 4 or 5). Don't worry, CPU times seem to be very low (not always: sync messages and slides need some more power).

Use DirectSound or Silent output, not the default WaveOut driver of BUZZ !
After being able to use the standard WaveOut driver of BUZZ again, it showed me something terrible: using it as output device makes the vMidiOut timing VERY random, regardless of the settings. So I only can recommend the DirectSound output of BUZZ (maybe with settings similar to mine used for testing: latency 60...100ms, resolution 4, timing 5ms) or the Silent one (giving a big gap between the visible song position and the outgoing events, but quite usable for MIDI only). Probably the best settings will vary between the systems.
After changing driver settings or the machine method I'd recommend to save and reload the song to avoid some misbehaviours.

I'm aware that this machine seems a bit complicated, but keep in mind that in difference to other generators it is used to control very different kinds of MIDI equipment, so it has to have this lot of parameters. And BUZZ 1 doesn't seem best adapted for MIDI integration, so I had to go around several corners to get some things done, but finally I really can use this machine. And if you just start with the simple things in the next chapter, you will be able to use this as well ;)
There is another MIDI machine by Hagen in preparation, especially for controlling Rebirth. Have a look at his work for this purpose or as alternative to my machine.

2.1 Getting Started
After you have set up the machine and BUZZ for using one of your MIDI devices, you can use the notes in the first column like you would do with other BUZZ generators. If you're using a General MIDI device (like intern synthesizers of most sound cards) chances are great that you're listening to some piano sounds. You can change these sounds in the 5th column (MIDI patch) or with the slider in the parameters window.

The second column (velocity) is what volume is for other generators, only that the range is from 0...127 (0x7f) with default 64(0x40) instead of double these values, sorry for this, I just used the MIDI standard here.

If you only want to let BUZZ play some melody, you don't need to deal with the channel value in the 4th column. But for using several voices or effects this is necessary: MIDI associates most events with one of 16 (logical) channels: most effects apply to all notes on one channel, and every channel has always only one active instrument. This instrument can be switched with every note you play, but it is more convenient to use different channels for different instruments.

This can be handled very easy with vMidiOut by also using a different channel for every BUZZ track. In this case you have to put the values for the channel and the instrument (=MIDI patch or MIDI program) only once: at the start of your song.

With General MIDI the channel number 10 (0x0a) has a special function: it's the drum channel, and the notes you play on this channel don't change the pitch of one instrument but trigger different drums.

The most useful effects to start with are probably the MIDI controllers 0x000a (volume) and 0x0007 (pan) as the BUZZ sliders don't work for this machine :(The pitch wheel (0x0101) and modulation wheel (0x0001) come handy as well. Of the things I implemented additionally to the MIDI standard I'm most excited about the cut/retrigger command (0x0120), because you won't find this in any sequencer. But with this we are already in the advanced usage topics I'll deal with below. The same goes for the delay, effect slides and synchronizing.

2.2 Menu Commands

The menu just has some reset functions (where all note-off functions can also be configured to sent with the BUZZ pause button) and the option settings.

If you've got a lot of hanging notes (especially if working without checked "monophone tracks" option), you can kill them with the following commands:
All notes off just sends MIDI controller 123 to the device on channel 1, which is very fast but doesn't work with some devices,
Stop all notes really sends a Note-Off for all 128 notes on all 16 channels, this is the PANIC button, but may several secondes on some devices.
If neither this doesn't help, you could try to disable the nasty device for a short time in the BUZZ preferences.
Reset controllers sends MIDI controller 121 to the device. It depends on the MIDI device if it really resets everything.

2.3 Option settings

In the options dialog you can change the following parameters:

- **MIDI device:** as explained above, don't forget to activate the device in the BUZZ MIDI out preferences.
- **Help mode:** BUZZ allows to show a short description for machine parameters, in case of MIDI controllers (effect commands 0x0000-0x007f) I give the standard General MIDI meaning here as default, but for controlling Rebirth it's more convenient to show the special meaning of those controllers for this program. Help modes for other programs/ devices may follow. Changing the help mode doesn't affect anything else of the behaviour of vMidiOut.
- **monophone tracks:** You can switch off the default behaviour of the machine, to send a note-off for every new note it finds on a track. In this case every note-off just cancels the last note sent on this track, all others have to be switched off explicitely with a velocity off zero, so you'll probably soon end with a lot of hanging notes ;)
- **MIDI play/channel:** not very useful for a MIDI machine, if checked the machine listens to the selected MIDI channel (0-15) of the device you choosed as BUZZ MIDI input and repeats every note on the output MIDI device with the same channel, note value and velocity. This is like MIDI through, but for note messages only.

Some options dealing with the BUZZ stop button and a following continue of BUZZ playback, especially for syncing
purposes:

- **BUZZ pause options:** You can configure the messages, the machine will send, if the BUZZ pause button is pressed. The "all note off message" would be controller 123 on channel 1, which is great for getting rid of the hanging notes, if your MIDI device supports it. If not, choose "all note-offs" here.

The "sync stop message" depends on the device/program you want to sync with BUZZ. The sync messages (0xf8) will be stopped in every case, but some programs explicitely need the stop messages to stop their timer, while my sequencer already pauses completely without sync messages, but would leave the "extern MIDI sync mode" with a stop message.
The BUZZ continue options apply only if you have stopped a running sync and it get's restarted by the "sync refresh" commands (0x0110 0x0000 or 0x0113 0x0000)

- **smart sync restart:** If this is checked (default), sync start commands 0x0110 with given song position are only processed after a BUZZ pause, if no sync was running or if the song position is less than the last one set (good for loops):

  advantage:Just start start every pattern with the appropriate 0x0110 + song position command (yes, this is some work, a song position command for the BUZZ machine interface would be very handy) and your sync slave is always where you want it, even with the usual looping of a part of the song for editing purposes.
  disadvantage:You have always to press the pause button after position changes (at least for setting it forward). You also shouldn't use the *refresh sync*command 0x0110 0x0000 with this, as this makes the sync running again and prevents the use of the correct restarting position at the next pattern.

Some probably rarely used **extended options**:

- **MIDI delay:** If the method attribute of the machine is set to 4 or 5 all MIDI output can be delayed in the range from 0 (no delay) to 0x1ff (delay ca. 2 Rows). For method 5 you probably should leave this at zero, while for method 4 you can use this to correct the gap between MIDI and wave output.
- **slide resolution** (methods 4, 5 and 6 only): At 125BPM / 4TPB MIDI controllers would be sent about 20-30 times per row for sliding with the maximal resolution of 1. This fine resolution probably isn't needed and just causes a high CPU usage and unecessary MIDI messages, so this is set to 2 as default (=half message rate). Keep in mind, that the actual amount of steps in row (=tick subdivision) is decreasing with higher BPM rate, as the used BUZZ wave routines have a constant time grid.
- **allow timer** You should try the sync command 0x0110 instead of the timer sync command 0x0113, it seems to be better and doesn't use an extra timer. But to use method 7, the MIDI time code command 0x0115 or the old timer sync command 0x0113 you have to explicitely allow timer usage for a machine with this option. Keep in mind that for every machine you use this command one of a limited number of timer-resources is needed. You also should have this checked for one machine in the song only. The **timer resolution** of 5ms generally shouldn't be changed..

Using different scales (this is completely experimental, I'm not even certain that the resulting pitches are correct): Scale 0 always is the default chromatic scale, for all other scales every note is adapted to the active scale and preceeded by a pitchwheel command (even for incomming notes via MIDI play). Since the pitchwheel is common for all notes on a MIDI channel, only one single note can be correct, in case of chords there will be errors. Another thing often occuring with playing these scales by an extern keyboard is erasing a new note with the old one: assume you're playing two neighbour notes, which will be mapped to the same MIDI note, but with different pitchwheel values. If you release the old note after triggering the new one, both will be cancelled and you hear nothing :(

You can either chose one of the predefined scales (there may be more in the future) or load a scale as user scale (number 1).
The format is very simple: comment lines have to start with an '!', the first non-comment line is a description string, the second giving the number of tones in the scale (for instance 12 for the default chromatic). Now the given number of lines has to follow, containing the pitch values in comparision to the scale root note, either as proportion a/b or in cents (the root note must not be contained in the file!).
Here a simple example for the chromatic scale :

```
!this is the scale file for the chromatic scale
standard chromatic scale with 12 tones
12
!now the notes follow, the first with proportion 1/1 or 0.000 cents
100 cents
200 cents
300 cents
400 cents
500 cents
600 cents
700 cents
800 cents
900 cents
1000 cents
1100 cents
2/1
!the last line also could have been 1200 cents
```

After chosing a scale file for loading you will be asked for the MIDI root note, this will be the note number of the first note for the scale, all notes equal or less than this number will have the same pitch value: the scale root. This one is given in cent, compared to C0. The defaults are 0 for both: keyboard or BUZZ   C0 will just be mapped to the real C0 pitch. To transpose everything up 2 octaves you could set the scale root to 2400.0 (=C2). Beginning with this the scaling intervalls are calculated
up to MIDI note 127. For the example this would be:
C-0 -> 0 cents, C#0 -> 100 cents, D-0 -> 200 cents, ... B-0 ->1100 cents, C-1 -> 1200 cents, now the scale is full once,
and we start again with C#1 -> 1200 cents + 100 cents = 1300 cents,...

3.Advanced usage information

The machine allows 16 tracks, default for every track is MIDI channel 1 at startup. You can change the channels in the corresponding column of the track as you like, several tracks can access one MIDI channel, so it's possible to apply different controllers together. Drum channel generally is 10 (0x0a). This has to be set only once (at the start of the song) and will be remembered later.

If you choose an instrument number a program change message will be sent to the actual channel, so you also only should do this at the start of the song or if you want to change the actual patch. Now every note will be transmitted, default velocity is 64 (0x40), highest possible 127 (0x7f). If you use the default of "monophone tracks", an already running note will be canceled. If not and you only have one running note, a note-off or just a zero in the velocity column will stop it, otherwise only the last one stops and all other have to explicitely switched off with the note-number and a zero velocity.

To reduce the amount of typing into the BUZZ track columns, you have to set only one of the three effect columns. The others are replaced by the last value you've put in there or some defaults (for the effect-slide byte this is to do no sliding at all, and the sync commands generally behave different with or without data values). So if you're working with one effect per track only (during some rows or patters at least), you have to set the effect kind only once together with a starting value for the data. Later only set the data or just the "Slide-To value". After a slide this value will be used as default data instead of the last data value.

3.1 The different methods

If I had a method to solve all timing things, I'd make it the only one and nobody had to care about them. But since I don't have this final solution (yet?), I only can give a summary of the advantages and disadvantages of the different methods. You can set them as attribute in the machine menu:

- **method 0: simple timing** <--- use this, if nothing else works :(

  +very few CPU usage
  +compatible, should work on every PC
  -not realtime, is a bit early (up to one row), especially no sync between MIDI and WAVE output, midIdelay can't be used to correct this
  -timing resolution is once per row
  ->only timer sync
  ->no delay, slide, retrigger
- **method 3: event based timing** <--- gives the best timing for undelayed events

  +very few CPU usage
  +compatible, should work on every PC
  +sync between MIDI and WAVE output (tolerance 5 ms?) ,
  (-) but maybe disturbed by high CPU usage

-(was?) a bit unstable, gave some crashes for me
-timing resolution is once per row
->only timer sync
->no delay (including mididelay), slide, retrigger

- **method 4: wave based timing** <--- largest range of adjustable mididelay

  +delay, slide, retrigger
  +new wave based sync without timer (but why is this incorrect again?)
  +sync between MIDI and WAVE output, (-) has to be adjusted with mididelay
  -not tested much yet (was just a step on the way to method 5, but may still have some advantages)
  -maybe dependent on system + driver + BPM settings settings (for me it works with 125 BPM and a latency of about 100 ms, but doesn't with <50ms or >110 ms) (is this still the case?)
  -timing resolution limited by Wave-functions (for 125BPM this is about 25 per row)
- **method 5: event+wave based timing** <--- default, best one for synchronizing (command 0x0110)

  +delay, slide, retrigger
  +new wave based sync
  +sync between MIDI and WAVE output (automatical)
  -system dependency? (similar method 4)
  -timing resolution limited by Wave-functions, especially disturbing for start of undelayed events
- **method 6: mixture of methods 3+5** <-- still in testing phase, only slightly worse than method 3

  +delay, slide, retrigger
  +new wave based sync
  +sync between MIDI and WAVE output (automatical)
  +undelayed events should have a timing as method 3, others are limited by Wave functions
  -system dependency? (similar method 4,5)
- **method 7: timer based** <-- still in testing phase, seems no very big improvement over method 3

  -needs timer (you have to activate it in the options dialog)
  -> no other timer commands (timer sync 0x0113, MTC 0x0115) allowed currently
  +timing resolution is adjustable (options dialog), but not saved currently!
  +sync between MIDI and WAVE output, but still depends on BUZZ driver capabilities (seems best with DirectSound output)
  -no delay (including mididelay), slide, retrigger implemented yet

The obsolete methods 1 and 2 are removed from the machine because they weren't usable or had no advantages over method 3. I still use method 5 as default, the new method 6 may give some better results for undelayed events, still allowing delays/slides and retriggers. But without these, method 3 should give the best possible timing without using an extra timer. Method 4 could be used to adjust the timing in both directions, allowing compensation of delays of your MIDI devices, but this isn't worked out in detail yet. I hope to have fixed most of the reasons method 3 crashed before and also the system dependency of methods 4, 5 and 6 should be gone now. Even method 7 showed to be not the final one: everything seems to depend on the wave driver settings with DirectSound giving by far better results than the multimedia waveout driver.

3.2 Delays and Effect slides

The delay and effect-change bytes are used only with the method attribute of the machine set to 4, 5 or 6. The delay byte can be in the range from 0...0xff (255), the latter gives approximately a delay of a whole row. Currently it may happen, that high delays prevent the event from being triggered at all. If you've choosen a general mididelay for the machine, values are just added together.
The time resolution for the delay and the effect slides of course isn't 255/row, but system, driver and BPM dependent. The actual value is shown in the description field for the effect slide byte, or as minimal difference for the delays in the description for the delay columns. On my system (P200, AWE32) for instance it's about 30/row for 125 BPM with the DirectX driver at a resolution of 4 and a latency of 100ms and gets lower with a higher BPM rate or less latency. The resolution for effect slides additionally is divided by the number you have as **slide resolution** in the options dialog (default is 2).
Effect slides are currently implemented for continuous MIDI controllers 0x0001...0x0061, all NRP's and the pitchwheel. You just put the MSB (most significant byte = higher byte) of the effect data value you wish to slide to until the next row into the effect change column (the last one). This would look like

```
 Note  Vel Del Chan  Patch  Effect   EData  Slide

  C-4  ..  ..  ..     10     0007    fffe    80

  ..   ..  ..  ..     ..      ..     8000    01

  ..   ..  ..  ..     ..      ..     0000    ..
```

for a volume slide to zero in rows. The last zero just makes sure the effect data is really at the target, the slide itself may stop a bit earlier. The second data value (0x8000) can be omitted.
For 7Bit controller and NRP data you can slide trough the whole possible range this way, as a slide value of 01 would be exactly the same as an effect value of 0000.
For 14Bit controllers (not yet implemented) and the pitchwheel this is different: the slide value of 0x01...0xff really just gives effect values in the range 0x0100...0xff00, so you not only get just 8Bit resolution, you also miss a small range at the start and the end. This will probably not fixed without releasing a completely new version 2 with a 16Bit effect slide value (and a normal volume range instead of the velocity) although I already had plans to abuse the (only rarely needed) channel byte for the LSB...

3.3 Cut and Retrigger

The lower byte of the data for the cut/retrigger command (0x0120) is a delay in the same format as the mididelay or the delay column. The higher byte gives a counter for the events to happen, that means 0 just toggles note off after the delay(=cut), 1 starts it again immediately after the cut (=Retrigger), 2 cuts again after another delay and so on.
This way every even number let's you with the note cut off, every odd one with a note-on. If there's an event on the next row the process of retriggering is stopped after reaching it and again the delay values near 0xff aren't reached for sure. If there's no new event, the retrigger process may last about several rows.

3.4 Effects/MIDI controllers
You can apply a lot of effects, which generally depend on your MIDI-equipment. If you just use the effect data column the last applied effect is used. You can try to **reset all controllers** with the command in the machines menu, it just sends controller 121. Since I wanted to have a full 16Bit range for some of the effects, but also wanted to allow using the BUZZ parameter sliders, I expanded the data ranges of continuous controllers to 16Bit (with version 1b). Some mesages need exact values, so I tried to keep them (controller 0,32 and 98-127 currently). There probably will be more changes here in the future :]
For most commands a (very much shortened down) description is shown on the bottom line of the pattern editor, you can also switch to the Rebirth help mode in the options dialog, to get some information about the usage of the controllers for Rebirth, which isn't given here in the table.

vMidiOut Effects	**MIDI events**	**MIDI Data Range**	**BUZZ/vMidiOut Data Range (if different)**
0x00xx	**MIDI controller**	*	*
0x0000-0x007f	**(7Bit) Midi Controller xx (0-0x7f)** (Ctrl 0-31 are MSB and 32-63 LSB of 14Bit ctrl)	0-0x7f	**ctrl 1-0x1f,0x21-0x61**: 0-0xfffe (shifted 9Bit left) **ctrl 0,0x20,0x62-0x7f:** 0-0x007f (=MIDI)
	00:Bank change	(MSB of) bank number 0-0x7f	0-0x7f

01:Modulation wheel 02:Breath controller 04:Foot Pedal 05:Portamento Time 06: Data Entry (Registered Parameter) 07:Volume 08:Balance 0x0a (10):Pan 0x0b (11):Expression		0-0xfffe (shifted 9Bit left)
0x0c (12):Effect control 1 0x0d (13):Effect control 2		0-0xfffe (shifted 9Bit left)
0x10 (16):Slider 1 0x11 (17):Slider 2 0x12 (18):Slider 3 0x13 (19):Slider 4		0-0xfffe (shifted 9Bit left)
0x20-0x3f (32-63):least significant bytes of controllers 0-31 for 14Bit precision 0x26: Data Entry (NRP)		
0x40-0x5f (64-69):Switches: 0x40 (64):Sustain 0x41 (65):Portamento 0x42 (66):Sustenuto 0x43 (67):Soft 0x44 (68):Legato 0x45 (69):Hold 2 Pedal	0: off, 0x7f: on	0: off, 0xfffe: on
70 Sound Variation 71 Sound Timbre 72 Sound Release Time 73 Sound Attack Time 74 Sound Brightness 75 Sound Control 6 76 Sound Control 7 77 Sound Control 8 78 Sound Control 9 79 Sound Control 10		0-0xfffe (shifted 9Bit left)

	80 General Purpose Button 1 81 General Purpose Button 2 82 General Purpose Button 3 83 General Purpose Button 4	0: off, 0x7f: on	0: off, 0xfffe: on
	0x5b (91):Reverb/Effects level 0x5c (92):Tremolo level 0x5d (93):Chorus level 0x5e (94):Celeste level 0x5f (95):Phaser level		0-0xfffe (shifted 9Bit left)
	0x60 (96):Data Button increment 0x61 (97):Data Button decrement		
	controller combinations (NRP/ RP, see below) 0x62 (98):NRP controller (data at 0x26) 0x63 (99):NRP preselect (MSB?) 0x64 (100):RP LSB (data at 0x06) 0x65 (101):RP MSB		
	0x78-0x7f (120-127) Mode messages		
	0x78(120):All sound off	0	0
	0x79(121):Reset all controllers	0	0
	0x79(122):Local Mode off/on	0 or 0x7f	0 or 0x7f
	0x7b(123):All notes off	0	0
	0x7c (124): Omni Mode on 0x7d (125): Omni Mode off 0x7e (126): Mono Mode 0x7f (127): Poly Mode		
0x0080-0x00ff	**14 Bit Midi Controller 0x80+xx** (not implemented yet)	0-0x03fff	0-0xfffe (shifted 2Bit left)
0x010x	**other MIDI messages**	*	*
0x0101	Pitchwheel	0-0x3fff (default:0x2000)	0-0xfffe (shifted 2Bit left) (default:0x8000)
0x0102	Channel pressure (Aftertouch)	0-0x7f	0-0xfffe (shifted 9Bit left)

0x0103	Key pressure (Aftertouch)	note: 0-0x7f +pressure:0-0x7f	0-0x7fffe: high byte: MIDI note 0-0x7f low byte: 0-0xff (shifted 1Bit)
0x011x	**Sync macros**	*	*
	0x0110: start wave sync (methods 4,5,6 only!)	(song position 0...0x3fff in beats)	....(no value): start (without setting songposition) 0000: refresh/correct sync 1...0x3fff: starting position 0x4000: start position 0 (in general not needed because equal ....) 0x8000+xxxx: continue at position xxxx 0xfffe: continue (without setting songposition)
	0x0111: stop syncing (for commands 0x0110,0x0113,0x0115)		....(no value): stop + sync stop message 0xfc 0000: dont send 0xfc
	0x0112: wave sync (!obsolete)		
	0x0113: start timer sync (activate timer in options dialog!, don't use with method 7)		....(no value): get timerrate from BPM 0000: refresh sync xxxx: set timer rate (in ms)

	0x0115: start MIDI Time Code sync (activate timer in options dialog!, don't use with method 7), only 25 frames/sec implemented		....(no value): start MTC at offset 0 0000: refresh sync xxxx: set offset (in sec) (set additional frame offset in effect slide column)
0x12x...0x14x	Special vMidiOut commands	*	*
0x12x	Cut/Retrigger	*	*
0x0120	cut/retrigger		0xAABB: A=count, B=delay even A cuts, odd A retriggers
0x13x	Transposing	*	*
0x130	transpose machine (drum channel 0x0a will not be transposed)		0 or .... (no value) no/reset transpose two formats are possible: 0xAB00: A octave: 0..7 or 8...f (-8....-1) B halftone 0....c 0x00CC: halftones: 1...0x7f or 0x80...0xff (-128...-1)
0x131	tranpose current track only		as above (command 0x130)
0x132	reset all transposing (for machine and all tracks)		-
0x14x	Chords	*	*

0x140	chords (from "library")		chord numbers are compatible to Argüelles TB3003: 0: no chord 1: maj 4, 7 2: min 3, 7 3: min 4, 8 4: maj 3, 8 5: min 5, 8 6: maj 5, 9 7: min 5,10 8: maj 4, 7,11 9: min 3, 7,10 a: maj 3, 7, 9 others may follow...
0x141	build chord (up)		0xABCD: every 4bit number A,B, C or D (0..0xf) gives the halftone difference to the base note, a value of 0 will be ignored, for example chord maj 4,7 would be 0x4700 or 0x4007 or 0x0047
0x142	build chord (down and up)		0xABCD: as above, but A and B give tones less than the base note 1-> -1, 2-> -2...
0x143	build chord (down)		0xABCD: all chord tones will be less than the base note 1-> -1,2->-2,...
0x144	build chord (down transpose and up)		as 0x142, but A,B are taken positive with transposing one octave down 1-> -11, 2-> -10...

0x145	buid chord (down transpose)		as 0x143, but all numbers are taken as positive with transposing one octave down 1-> -11, 2-> -10
0x01fx	System message fx	*	*
	f0:SysX start (no way to implement SysX here)		
	f1: MIDI time code quarter frame (use 0x0115 instead)		
	f2:song position (in beats)	0-0x03fff	
	f3:song select	0-0x7f	
	f6: tune request	none	
	f7: SysX end		
*	realtime messages (use the sync commands 0x011x instead of 0x01f8-0x1fc)	none	*
	f8:timing clock (to be sent 6 times per Beat)		
	fa:start		
	fb:continue		
	fc:stop		
	fe:active sense		
	ff:system reset		
0x02xx	7Bit NRP xx (short cut, see NRP) (signification given here is for SB AWE 32/64)	0-0x07f	0-0xfffe (shift 9Bit left)
	00: LFO1 delay		
	01: LFO1 frequency		
	02: LFO2 delay		
	03: LFO2 frequency		
	04: Envelope1 delay		

05: Envelope1 attack
06: Envelope1 hold
07: Envelope1 decay
08: Envelope1 sustain
09: Envelope1 release
0x0a (10) Envelope2 delay
0x0b (11) Envelope2 attack
0x0c (12) Envelope2 hold
0x0d (13) Envelope2 decay
0x0e (14) Envelope2 sustain
0x0f (15) Envelope2 release
0x10 (16) Initial pitch
0x11 (17) LFO1 to pitch
0x12 (18) LFO2 to pitch
0x13 (19) Envelope 1 to pitch
0x14 (20) LFO1 to volume
0x15 (21) Initial Filter Cutoff
0x16 (22) Initial Resonance (Filter-Q)
0x17 (23) LFO1 to Filter Cutoff
0x18 (24) Envelope 1 to Filter
0x19 (25) Chorus
0x1a (26) Reverb

3.5 Some explanations about MIDI data
I'm not that MIDI expert and just tried to apply the information I collected from several sources and bring them into an useful form for the MidiOut machine. MIDI is a protocoll for communication between musical devices, which basically consists of a stream of bytes (8Bit), where the single bits are transmitted sequentially, like serial communication (mouse,modem) does, but with a higher bitrate. If the highest bit is set, the byte is a so called status byte, which means some kind of command, while without the highest bit there are only 7 bits remaining for data, that's just the range 0-0x7f (0-127) in the table. To complicate things, different commands/statusbytes need/allow different data, some combining two (7Bit) data bytes to a 14Bit value (range 0-0x03fff). For controllers 0-31 this get's even worse: you can decide for yourself, if 7Bit resolution are enough and just send one data byte, or if you need some finer adjustments sending the second (least significant byte:LSB) as controller xx+32 (32-63). Also the

meaning of the controllers is only rarely standard with General MIDI just giving a start. I don't know much about GS and XG.

3.6 NRP for AWE32/64 (and RP)
The NRP (non registered parameter) stuff is especially useful for the AWE, at least with this values (don't know if there's some kind of standard; GS and XG seem to use NRP too, but with other values). For instance if you use an effect 0x215, a value of 127 for controller 99 and a value 0x15 (21) for controller 98 will be sent for the current channel. Finally the effect data is transmitted on controller 0x26 (38).
Since version 1e all 3 controllers are sent every time you use a NRP command, so you can apply different NRP's from several BUZZ tracks. In case of sliding the two first controllers are only sent, if the same vMidiOut machine has sent a different NRP command to the same MIDI channel in the meantime. So you can do slide for more than one NRP, but you have to do this with the same machine.

Registered Parameter (RP) as part of the General MIDI standard
I've not added a shortcut for the RP's yet, but I also have found only few used ones:
To set the pitch bend range for a channel, set controller 0x64 (100) (RP LSB) to a value of 0, and controller 0x65 (101) (RP MSB) to a value of 0, then set controller 6 to the desired range in semitones (up to 12).
RP's 1 and 2 are fine and coarse tune, no idea about the parameters.

3.7 Timing/Synchronizing/MIDI TimeCode

Basic Ideas
To sync a sequencer with BUZZ, you would select some option like "MIDI Sync" there, connect BUZZ and the sequencer with Hubi's LoopBack device (or any other of that kind) and BUZZ would have to send a sync start message (=effect 0x01fa), a stop (= 0x1fc) and between a lot of sync timer messages (=0x1f8).
But only one sync message per row isn't enough, you need 6 of them, and this with as equal distance as possible. This was the thing, which should be easy to achieve (BUZZ manages several times 44100 samples the second, shouldn't it manage these 6 syncs in a row?), but nonetheless was rather hard to implement, and even now has (still?) some drawbacks.

One simple solution is to set the TBP rate to 6*4=24 and use only every 6th line for your song (but you don't have to restrict to these) and every line for a sync message 0x01f8. This should get you going, but the resulting patterns are hard to deal with.

To get timing more correct I used the event mechanism of the BUZZ machine interface, so at least a synchronisation between Wave and MIDI output can be achieved (method attribute set to 3 or 5).

Hubis Loopback Setup
Most of the problems I had with synchronizing were caused by the fuse mechanism of Hubis Loopback

device (versions >2.4). This shall prevent system crashes caused by a circular loop of MIDI messages by inhibiting the same message to occur more than a certain amount in a certain time. The default values are o.k. for normal MIDI usage, even for equidistant sync messages, but for the correction mechanism I use, some sync messages were swallowed. So go to your control panel -> multimedia->extended and choose one of your Hubi's Loopback port (default names are LB1,...) and change the "duplicates" setting to a higher value (for instance 20). This may prevent the fuse mechanism from working, but lets all sync messages of vMidiOut pass. Other programs similar to Hubis Loopback Device may have a similar fuse mechanism, take care to adapt it to the needs of vMidiOut.

Timer Sync (command 0x0113)
Further I experimented with an own timer, which is based on the BUZZ BPM and TBP settings, but else works independent from the BUZZ timing (sync command 0x0113). This came close, but had a lot of disadvantages:
-limited timer resources
-distortion of BUZZ timer ?
-even best resolution of 1ms matches only distinct BPM rates, for instance 119 BPM <-> 21ms, 125 BPM <-> 20ms, 131 BPM <-> 19ms, the formula is

```
time_ms=60000/ BeatsPerMin/6/TicksPerBeat;
```

(This is also of concern for doing the sync messages with an extern programm, like Vellocets vmidisync, and BUZZ sending only start and stop messages (effects 0x01fa and 0x01fc): get vmidisync together with other useful programs (vmidijoy and c2nrp) at Vellocets homepage .

Starting the timer with effect 0x0113 and a non-zero data just does the same as Vellocets vmidisync, with the advantage to let the BUZZ pause button send a configurable stop message and pausing the sync messages. So 0x0113 0x00014 just would give the BPM rate of 125 (unfortunately not exact), without any further adjustments (=not synced to BUZZ).
Starting the timer with no data-value (0x0113 ....) calculates the rate nearest to the actual BPM setting of BUZZ and starts the timer. Now you have to put a command 0x0113 0x0000 (or just .... 0x0000) in EVERY row. This does two things: it corrects the mismatch between the BUZZ BPM setting and the timer rate by sending additional or inhibiting sync messages if necessary, and further it allows to restart the timer after you pressed the pause button in BUZZ (I really found no other way, strange).

Using the timer requires to check the "allow timer" option of the machine.

Wave based sync (command 0x0110)
The last implemented method is based on the wave based timing of the methods 4, 5 or 6 and basically just tries to always send 6 sync messages between two BUZZ rows (for 4TPB). This doesn't need an additional timer, but the timing between the single sync messages may a bit vary.

To use this you would make 3 patterns for vMidiOut. In the start pattern place a command 0x0110 .... in the first row, followed by all other lines set to .... 0000. The second pattern, which just refreshs the sync and has to be repeated the whole sequence, just consists of .... 0000 (or maybe a 0x0110 0000 in the first row). The last pattern to stop the sync messages has a single 0x0111 in it (either a 0x0111 .... to send a final stop message or a 0x0111 0000 to leave this out).

Since version 1e other TPB values than 4 should also work, but recommended are still the values 4,6,8,12 and 24.

Since version 1f you can use a starting song position as argument. It is measured in MIDI beats, which are just the BUZZ rows for 4 TPB, and can be in the range from 1...0x3fff.

MIDI Time Code (command 0x0115)
Synchronizing via MIDI clocks only controlls the flow of time: every 6 MIDI clocks a beat passes, if the clock messages are sent faster, the resulting BPM rate increases (or vice versa: if the BUZZ BPM rate increases, the sync messages have to be sent faster, this is done automatically for the 0x0110 commnad, but for the 0x0113 command the timer has to be restarted).

MIDI Time Code in contrary is used to transmit the absolute time (in seconds, minutes and hours) and not only some kind of clock tick. This way an additional control mechanism is given to correct the timing. A second is partitioned in several frames, where 24, 25 and 30 frames per second are usual values. The smallest time unit is a quarter frame, for the value of 25 frames per second this are just 10 ms. This is the only framerate I've implemented in vMidiOut. It causes a higher activity of the machine compared to the sync clock messages every 20 ms at 125 BPM, so the CPU usage of vMidiOut is higher with MTC. The actual frame position is corrected via the refresh MTC sync command (0x0115 0000) in every BUZZ row by comparing the ellapsed time with the BUZZ sample position divided by the sample rate.

The command 0x0115 is used as the other sync commands: 0x0115 .... starts sending quarter frame messages with an offset of 0, so if you place this command at the start of your BUZZ song, the ellapsed time in BUZZ and in your sequencer should be the same all the time. The command 0x0115 0000 corrects the actual frame position and also is used to restart synchronizing after a break. Starting MTC with another value gives an offset in seconds, an additional frame offset can be given in the EffectSlide column. Using a value of 0xfffe doesn't change the offset at all, if no offset was given before, it's value is undefined.

A sequencer synchronized via MTC usually needs 1 or 2 seconds to adapt to the given timing (in difference to synchronisation via MIDI clocks), this should be taken into consideration. Also you need to allow the usage of the timer in the options dialog.

---

4. History

version 1a:
-Menu commands for Note-off, Controller reset

version 1b:
-**Data incompatibilities!** (at least it should not crash with old ones)
-MIDI device is now machine attribute
-delay byte on place of former device byte and new effect change byte as last byte (both still unused)
-changed parameter ranges for continuous controllers, NRP and pitchwheel to 0...0xfffe (ctrl 0, 98-127 stay 0..0x7f)
-assumed 7Bit for all controllers (14Bit maybe come later as extra effects)
-Bugfix with NRP causing System reset (0xff) message
-implemented event method by Jeskola with correct timing (Attribute method=1), still causes some crashs :(
-process program change, controllers and notes in this order (seems more logical)
-two new event based methods (2 and 3), but you should use method 0 or 3 only
-two new **experimental** sync macros:

- -0x0110 sends a start message and begins a series of event based sync messages (set effectdata to 1), (these are still too slow)
- -0x0112 bases it's timing on the BUZZ wave routines, but very imperfect currently :(you'll get a BPM independent, very fast and inconstant rate for effect data 1, you can divide the rate by raising the data value.

-Effect 0x0111 stops these syncmessages for both and sends a stop message (don't use 0x01fc only!).
-Unfortunately all these messages don't stop with BUZZ pausing :(

version 1c:
-added option for playing machine with Midi (very useful for a MIDI machine :]
-BUZZ stop now ends sync messages and sends ctrl. 123 :)
-**timer sync** command 0x0113 :)
-**configuration dialog**
-stop command 0x0111 0x0000 doesn't send stop message

version 1d:
-option for BUZZ-stop to send all note-offs (finally getting rid of all hanging notes :)
-removed some obsololete things: event sync (0x0110) and methods 1 and 2
-timer sync must be enabled in options dialog (this way you won't have several timers without thinking about it ;)
-option to have **monophon BUZZ tracks**
-fixed event based timing of method 3
-some people say, my machine is rather hard to understand: it will get worse ;)
-added method 4 with **wave based timing** (delta time =method 0) --> slides and delays should now be

possible
-added method 5 with **event + wave based timing** (delta time =method 3)
-added general **MIDIout delay**, works only with methods 4+5 (with method 5 this should usually set to 0, with method 4 this shall be used to correct the gap between wave and MIDI)
-added **event delay** for methods 4+5 (still in "work"-units)
-changed code to use several (2) event cues to have correct delay for method 5
-added **effect slides** (still system/bpm dependent) for 7Bit Ctrl, NRPN and pitch wheel
-added a lot of descriptions for the pattern and parameter view, for the idea and instruments thanks to Rout
-all time units are now (more or less) system independent (delay 0...0xff for one row)
-added **new sync** command based on methods 4 or 5 as 0x0110 (it's as poor as 0x0113 :(but without timer :)
-added **cut/retrigger** command :)
-changed defaults to method 5 and monophone tracks
-changed option dialog a lot, especially **choose device by name** now
-fixed some bugs with synchronizing
-use 3 event cues now (to access old data for sync and may be other purposes)
-fixed crashes during machine startup and avoided them for shutdown
-added **sync finetune**: finally some positive results with this
-use slide-end value as new default data value

version 1e:
-implemented **Midi Time Code** (MTC, command 0x0115), finally something works without finetune ;)
-improved method 3 handling to allow faster BPM rates
-special thanks to Hagen: He's working on a MIDI machine especially for controlling Rebirth; we finally didn't join our machines because of different aims, but from his contributions resulted the **Rebirth help mode** (gives meaning of controllers for Rebirth instead of GMIDI), some synchronizing corrections (multi threading) and the timer sync command 0x0114 (unfinished and removed again because of the next point ;) I recommend to test his machine if you just want to control Rebirth from BUZZ, it's easier to understand/manage and has an additional Rebirth buffer correction.
-detected wrong setup of Hubis Midi Loopback -> **removed sync finetune** ;)
-added **method 6** to fix the limited resolution for undelayed events in method 5
-added **MTC offset** (in seconds, additional frame offset in Effect Slide column)
-correct **14Bit songposition** command (0x01f2)
-tried to fix malfunctioning sync with newer Cakewalk versions (added delay value for 0x0110 command), partial success only?
-fixed/enhanced method 4: mididelay maybe up to 0x1ff (=2 rows) now
-increased cue number to 10! -> this allows BPM rates up to 500 at 4TPB for methods 3, 4 and 5
-fixed missing MIDI events for higher BPM rates and method 5
-fixed some casually missing sync clocks for sync start
-fixed missing sync clocks for higher BPM rates
-expanded **sync messages for all TPB values** (but best for 4,6,8,12 and 24)
-increased cue number to 20-> now 150BPM at 24 TPB are possible (interesting for converted MIDI

files), but don't raise the BPM rate much higher (for high TPB values only of course), BUZZ may crah sometimes
-**improved NRP handling**: all 3 necessary controllers are sent once every row but for slides only if another NRP (in another track, but with the same machine) was sent to the same MIDI channel -> you can do slides with more than one NRP on the same channel now, but they have to be done with the same machine
-added option **slide_resolution** to reduce CPU usage and MIDI traffic, default is 2 (for version 1d this was 1)

version 1f:
-some additional controller descriptions
-changed channel pressure range -> 0-0xfffe, key pressure 0-0x7ffe (MIDI note + pressure 0-0xfe)
-added **song position** as argument to command 0x0110
-slightly changed (improved?) timing behaviour for methods 3 and 5 (use time at start of tick processing as reference)
-new experimental timing method **7 with own timer** running at constant resolution (you must additionally enable the timer in the options menu), but without much improvement ?, it also doesn't allow delay, slide, retrigger, sync and MTC
-timer resolution is now saved with the song
-added **smart sync restart** option (greetings to Rikard and Hagen for this one), sync start commands with given song position are only processed after a BUZZ pause, if no sync was running or if song position decreases (to allow looping).

version 1g:
some funny things inspired by the buzz-talk community:
-track and machine **transposing** (commands 0x0130,0x0131 and 0x0132)
-using different **scales** (implemented via pitchwheel, so monophone channels are needed, a pitchwheel reange of 2 halftones up/down is assumed), a few hard coded scales (taken from VAZ) and >1000 loadable from a scale library I had on my HD.
Thanks to these people for the work, read the text file for details
>These scales were brought together mostly by John Chalmers
>(non12@cyber.net) and Manuel Op de Coul (coul@ezh.nl).
The user scale will be saved with the song. And finally MIDI playability is useful a bit ;)
-**chords** command 0x0140: the chord values are completely taken from Argüelles TB3003, I hope you don't mind. Of course this doesn't work with any non-standard scales.

version 1h:
-pitchwheel reset in machine menu
-additional chord commands 0x141-0x145 to build almost any chord

---

possible plans/future (and still missing things):

- any way of further improving timer sync or new wave based sync?
- cue-switching per track? -> more convenient retrigger if longer than 1 row
- correct handling of 14Bit controllers and NRP's
- "personal sync" for m2buzz: send a note on one channel every row, volume set to zero (hopefully obsolete now ;)
- easy slides option? (for systems with high latency only):

  probably not, it's easy enough now ;)
- retrigger commands with varying pitch and/or delay (using the Effect slide byte)
- another version with velocity [0...0x7f] ->volume [0...0xfe] and 16Bit Effect Change ?:

  probably not before everything else is working nice enough
- SysX-Realtime: MTC full frame messages
- SysX: MasterVolume
- allow/handle several timers ?

---

contact me: (vII) or look at my Homepage

WhiteNoise Additive v1.0
By David Wallin (c) 2002

Contents

1.1 Synth Overview and requirements

Additive Synth offers many possibilities not found in traditional analog synths. Traditional subtractive methods provide you with a few waveforms and to get different timbres you must filter out certain parts in order to get the sounds you want. Additive synthesis lets you build the sound from the ground up, choosing what parts you want and allowing you to do tonal variations which are not possible in other synths. The Kawai K5000 was an inspiration for this synth, but there are many differences in what you can do. Where the K5000 allows you to specify envelopes for each harmonic, this version allows you to make a table of harmonic changes. Like a the Waldorf Wave synths, you can traverse the table by using the envelopes or LFO's to change the table index. So, this synth is more like an "Additive Wavetable" synth. Additive wave sequences are stored as simple 128x128 images (128 Harmonics x 128 Time indexes). Because of this system, you can quickly and visually edit the harmonics in a paint program and load them into the synth. You can also do more wild tonal variations using this method. Like the K5000, I also have included a 128 band Formant Filter mode. The formant filter allows you to imitate accoustic instrument body resonances, vocal vowel sounds, analog filter sounds and even unimagined, unnatural timbre variations. As an added bonus, the formant filter can also be wave-sequenced allowing you to draw filter sweeps, vocalizations and other changes. Since LFO's can be tempo synced, the ability exists arpeggio-like patterns and even rythmic formant changes, if waves are drawn properly. Additive synth fall short of the k5000 in many areas, namely by limiting to only two oscillators as opposed to the k5000's max of 6, and also in less polyphony possible (though this will depend on CPU speeds). Though, in it's defense, the K5000 only allowed 64 harmonics per oscillator, so technically each oscillator of Additive synth does as much as 2 K5000 oscillators.

1.2 What is Additive Synthesis

A good resource is here, and there are many other resources on the net. Please check them out. Basically, all sounds can be expressed by the adding up of individual sine-wave tones at different frequencies. A pitched sound is usually made up of individual sinewaves, called "partials" whose frequency is some ratio of the main pitch, or the fundamental. For instance, a square wave is made up of only odd partials, or partials that have the ratios: 1 * frequency, 3 * frequency, 5 * frequency, etc. Since these partials have whole number ratios to the fundamental, we call them *harmonics*.. By specifying the volumes of each of these harmonics, we can create many kinds of waveforms. Conceivably, a sound in nature can have an infinite number of partials (and even non-harmonic ones). On a digital system, the number is finite. The maximum frequency that can be expressed is limited to 1/2 of the sample rate (also known as the *nyquist* frequency). Additive synth will let you specify up to 128 harmonics, though only some of these will be heard on middle octave and higher octave notes. On lower notes, chances are that you will hear all 128 harmonics. Each oscillator is limited to creating harmonic sounds, however, with the ring modulation Mix Type and by detuning the second oscillator, we can create some non-harmonic sounds. Bell sounds are one example of non-harmonic sounds.

1.3 Parts of Additive Synth

The first part is the Oscillator section. We have two osc's to work with. You can load your own sounds by right clicking on the machine, picking "load wave" and selecting a raw file to load. Some example files are provided, and can be found in your generators directory. See the section on "Creating your own waves" for more info on how to create your own. The oscillator section allows you to control the wave sequence index, and also the phase of the oscillators. The next part is the Mix section which controls how the two oscillators are used in conjunction with each other. Next we have the amplitude envelope which controls how the sound gets louder or softer in response to your playing. Attack is the amount of time it takes before a sound reaches full volume, decay is the time to drop down to the sustain level, and release is the time it takes to fade out. The same goes for the next section called "ENV1". This is an assignable envelope that can be used to control many things, including pitch and the oscillator indexes. LFO1 and LFO2 (meaning Low Frequency Oscillator) are oscillators that can be assigned to control different parameters as well. Then we have the Assignable section which actually controls which things get routed to what and how much they modulate them.

2.1 The Oscillator Section

As we said, each oscillator is represented by a 128 x 128 block. This block, we have been calling a "Wave Table" or "Wave Sequence". <INSERT PIC>

Each row on the block represents all 128 harmonics that are played at one time. We have 128 of these "time indexes" and you can set which index is being played by changing the values of "OSC1Index" and "OSC2Index". These parameters can also be modulated with the LFO's and ENV1. By modulating the index we can create changes in the timbre or tone of the sound. It's recommended that you try different waves and experiment to see what I mean. Some wave tables, such as the Saw and Square wave contain no changes over time and simply represent a static tone. Feel free to create new versions of these with your own changes - Saw and Square make good bases for editing. By default, the saw wave is loaded. Phase does exactly what you expect by allowing you to initialize the phase of the oscillator to a set value. By setting the Limit parameter, you can lower the number of harmonics actually rendered - this lowers CPU usage. You should do this if you are dealing with sounds that aren't very bright and don't need the extra harmonics.

3.1 The Mix Section

In the Add and Subtract modes, the output of the two oscillators is summed. The MixBalance parameter allows you to set the Mix between the two. In the RingMod mode, OSC1 is multiplied by OSC2. This can be used to create metallic and harsh sounds. In this mode, mix balance does nothing. In Formant mode, OSC2 is not rendered as usual, but instead it's wave table is used as a Formant sequence. So, you can still set OSC2's index and modulate the index just as you would with an oscillator. You can also use the OSC2 Semi and Detune parameters to shift the formant filter up and down however, at this time, OSC2 pitch modulation has no effect on this. MixBalance also does nothing in this mode.

3.2 The formant mode

Another reference from the same site as before. Formants are frequency zones that are boosted or cut in accoustic instruments and voices because of the shape of the instrument. this is how our human throats can form different vowel sounds - by changing in shape and creating different formants. You can play an instrument at different notes but the formants stay fixed. So, formants are very important for creating realistic accoustic instrument sounds or vocal sounds. We can also use the formant filter to imitate analog synth sounds. There are 128 bands to the formant filter. The frequencies of the bands go linearly from 0 to 5000 hz. Many examples are included (all the waves marked as FF_) to get you started for making vocal type sounds and basic filter sounds. Please check these out.

4.1 The Mod Section

With the mod section, you can route the LFO's and ENV1 to modulate various things, such as Oscillator Pitch, amplitude and OSC indexes. Simply set the source to either LFO1, LFO2 or ENV1. Destination can be set to OSC1 Index, OSC2 Index, OSC1 or OSC2 pitch, or Amp. By modulating the index you can create interesting timbral changes.

5.1 Tricks and Tips

- To make a Square wave with changeable pulse width, load Saw.Raw for osc1 and osc2. Set the Mixmode to subtract. This will give you a square wave. By changing the phase, you can change the pulsewidth of the square wave
- The Saw and Square waves have no changes in their wavesequence. You can use these as a base to build other sounds. Also notice that these waves are not very bright, but they sound good. Keep that in mind when making your own waves - the first 16 or so harmonics are the most important for determining the sound.
- Always turn off modulation routings to save CPU if they are not being used

5.2 Making Arpeggios and rythmic sequences

Basic kinds of arpeggios can be made using the Arp waves for the LFO shapes and by setting the LFO speed to be based on Ticks (the high values of the speed parameter will read N Ticks). Also, turn on LFO Sync so that the sequence can start on a note. By setting up your Wave sequences in a rythmic manner and using this property, you can also get rythmic timbral changes. IE, have changes in the .RAW appear at intervals like 16, 32, 48, 64, etc.

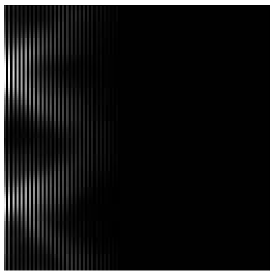
5.3 Making your own wavesequences

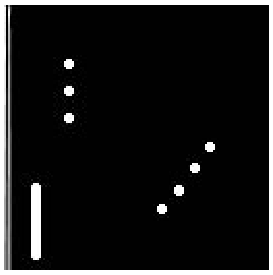
.RAW files follow a very simple format and can be loaded easily in Photoshop. Simply load them as 128x128 raw files, 8 bit, and no header. Once you load a .RAW in a song, it is saved in the .BMX file, so any changes to the .raw on disk will not affect your song. The columns in the waveform are the harmonics, going from 1 to 128. The rows are the time indexes.

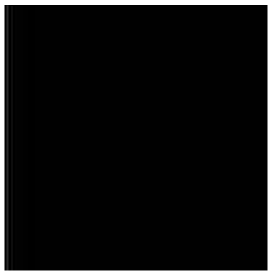
5.4 A note about presets

Since presets only save parameter data, you must load the .RAW files yoursef. Click 'edit' on the parameter window and check out the comments for each preset. On each preset, I've tried to mention what waves should be loaded to get the intended sound.

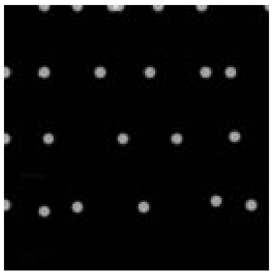
Appendix A - Sample Waves

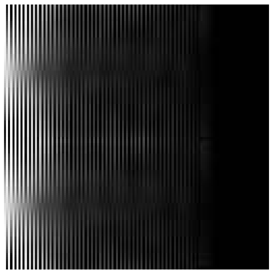

Animated_sqr

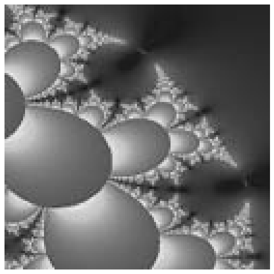

Bleeps


Dark Pulse

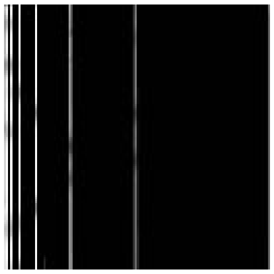
Dark Square

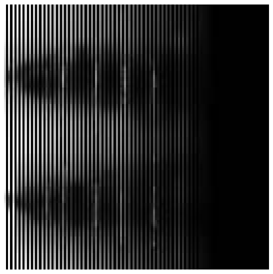

Dotz


Evenodd


Fractal1

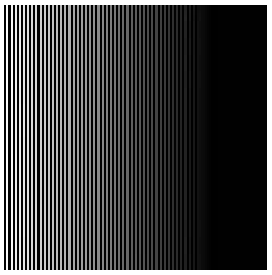
Harmonica

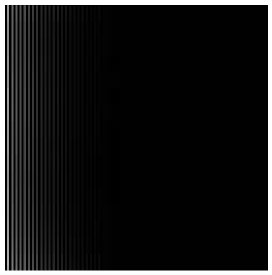

Organ


PWM sqr

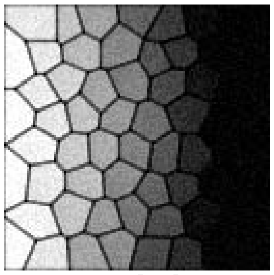
Saw

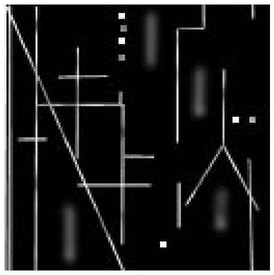
Saw fade

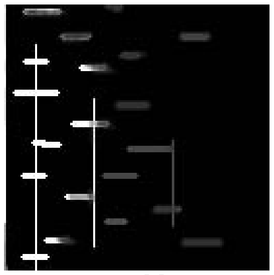

Sqr

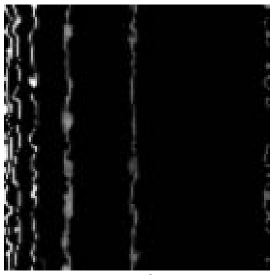

Sqr2

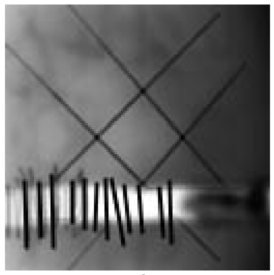
Square

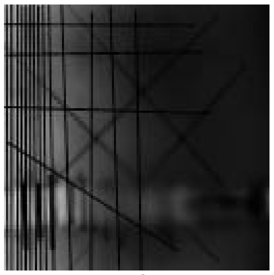

Stained Glass

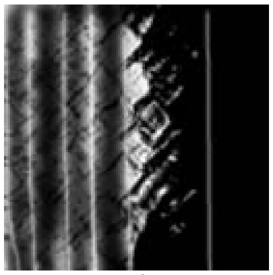

Techno

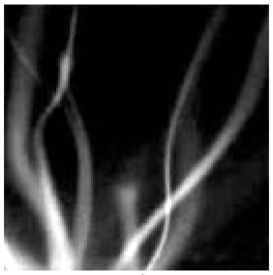

Twinkles


Waveform

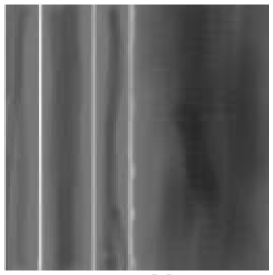

Waveform2

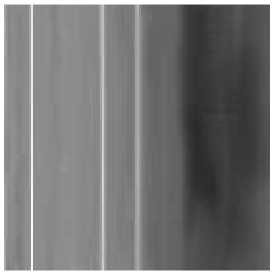

Waveform3

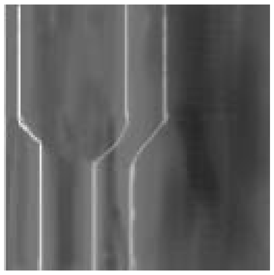

Waveform3

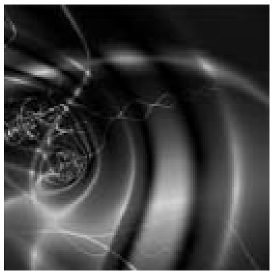

Waveform4

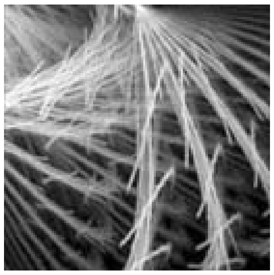
Formant Waves


FF_ahh

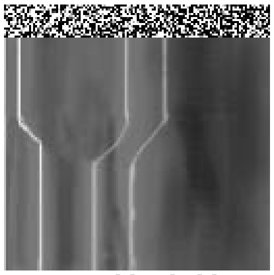

FF_ee

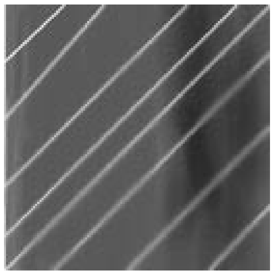

FF_eehahh


FF_Fract1

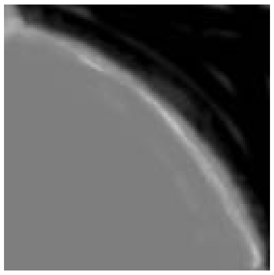

FF_Fract2

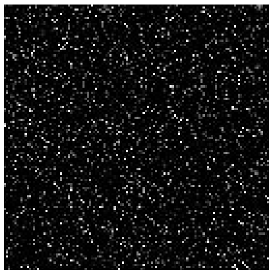
FF_Hp

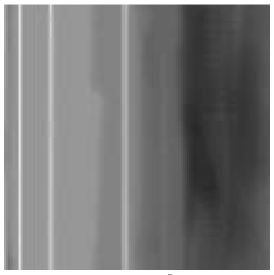

FF_Kkkeehahh


FF_Linez

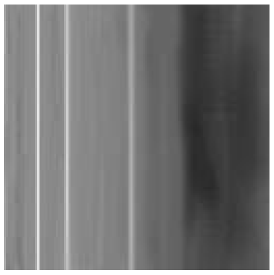
FF_Lp

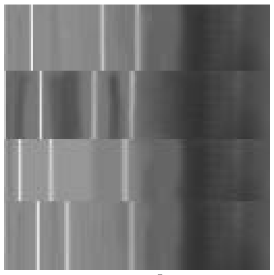

FF_LP_lowres

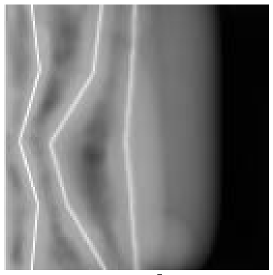

FF_Noise

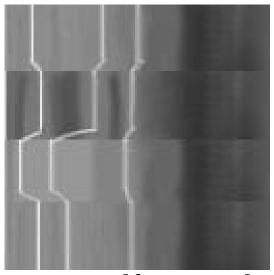

FF_ooh

FF_Thickband


FF_U


FF_Vocal_Fun


FF_VocalFun2


FF_VocalfunSmooth

Dave's Delta version 2.0

What is it ?

This is another synth type generator, like the Geonik's Bass.

This machine gives a bit more control, however, with the ability to set the envelopes and pitch glide (cool for some things). You can also detune notes, so you can create some simple synth effects by playing the same note with several slightly detuned versions. An example song is provided. Finally, you can also morph between two waveforms, so you can get smooth transition between a sine and a saw wave for example. I dunno if this will be useful, but I guess it's up to you. If you find any bugs, email me.
New to this version:
- Fixed bugs with notes being incorrect (oops). If they are still off, please tell me. It sounds correct to me
- Better tune control
- better demo song
- slight gliding for morph and tune values (automatic)
- fixed some clicks/pops

Thanks to Rout, since I stole his html for this page. ;)

This is DONATIONWARE. If you want to be kind, you can send me any amount of money (or anything else you think I'd like to have) to the following address. Thank you.

David Wallin
122 Heather Valley Rd.
Holland, Pa
18966
USA

WhiteNoise's Drummer version 1.0

What is it ?

This machine is specifically made to handle drum sounds (so it's a drum tracker really). You can't specify a note, just trigger the sounds, and it's supposed to be limited to one wave per traack (you could concievably change waves in a track, but that would defeat the purpose of using this machine). Other features in this drum machine - You can set an offset in the wave so the sounds start at a different spot. This gives the sound a slightly different attack, and thus a different tone. You can use this to add some realism and variation to your drums instead of having them sound totally the same the whole time. Also you can control the rate which the samples are played. This is essentially the same as changing the note if you were playing a sample in the tracker, but this gives you better control, the rate is glided, plus, you can play samples backwards! This meachine features ramping for when samples start, so you shouldn't get any clicks. There are some odd problems with it - if a sound doesn't play, usually the fix is to set the 'offset' to something (experiment). One other thing of note - when setting a negative rate, the offset is handled as from the end of the wave, rather than the begining. If you have a lot of extra space on the end of the wave, you might not hear anything. Use the offset to account for that.

Thanks to Rout, since I stole his html for this page. ;)

This is DONATIONWARE. If you want to be kind, you can send me any amount of money (or anything else you think I'd like to have) to the following address. Thank you.

David Wallin
122 Heather Valley Rd.
Holland, Pa
18966
USA

WhiteNoise's Square version 1.0

What is it ?

A cheese-tastic C64-style Square wave generator.

This machine was designed with customizability in mind. Basically, it makes a square wave and the width of the pulse can be varied for interesting effects. You can set the LFO pattern used to modulate the pulse, the speed it modulates, and the values modulated between. In addition to this, you can do Frequency modulation (for vibrato), Amplitude modulation (for tremolo) and even 0ldsK007 appregiator like effects, reminiscient of SID tunes. And of course, you can pitch bend and glide notes. Check out the demo song for a smattering of sounds that you can make with this machine (it's even good at making square noise).
If you find any bugs, or have any comments or suggestions for new machines, email me.

Some Notes:
- In AM and FM modes, the PulseWidth variables are used to control the frequencies or volumes respectively.
- Phat mode adds two octave waveforms and detunes them slightly.
- OldsK007 is the appregiator mode
- LFO speed is in times/second (hz)
- In Oldsk007, LFO speed sets the appregiator rate, and PulseWidthA sets the ratio between the sections that are raised 2 octaves and the dry section.

This is DONATIONWARE. If you want to be kind, you can send me any amount of money (or anything else you think I'd like to have) to the following address. Thank you.

David Wallin
122 Heather Valley Rd.
Holland, Pa
18966
USA

Suppahelp!

(Ynzn's newest machines)

by *Christiaan Janssen*

Click'n'pop Generator

This machine generates random pulses. You can control the amplitude, period and length of these pulses. There is also a couple other controls.

P a r a m e t e r s

On/Off switch	Maintained by historical purposes. Even when the song is stopped, if you turn this on the c'n'p will start working, and when you turn this off it will stop (easy, isn't it?). When you press the Buzz stop button it is turned off automatically. I should remove this parameter, but maybe somebody finds it useful.
Trigger	It forces the c'n'p internal timer to restart, and turns the machine automatically on. If you're writing drums or similar things with great delays (high values for the other parameters) it lets you sinchronize the c'n'p with the rest of the song.
Remove DC	At first sight it isn't very useful. It was supposed to cancel the zero-frequency component of the signal, to avoid problems with other effects based on integration. I discovered that it can be an amazing source of new sounds and new sensations.
Medium Value	For each of the three controllable parameters, this value is its "normal" value. If you turn variation to 1, all pulses will have this value. So turning all three variations to 1 you get a regular square signal.
Variation	The variable that defines each of the three controllable values of the pulse is a random uniform variable from MedVal-Variation to MedVal+Variation. I wanted it to be a normal Gaussian random variable, but it was too much work :). Maybe some day...

Values

3Dizer

It's a stereo effect. You tell it your position and the position of the source of sound and it calculates the delays and attenuations of each ear. If you change the parameters in real time it simulates the Doppler effect, shifting the pitch of the sound up or down, depending on the direction and speed of the movement.

P a r a m e t e r s

L/R Separation	The virtual separation of your ears. The higher this value is, the thougher the effect will be. More separation means more delay between ears and higher difference in the attenuations. There are no units, this distance determines the impression that will cause the other values to you.
Angle	The position of the soundsource is defined in polar coordinates. This angle is given in degrees. 0 degrees means directly at your right. 90 degrees your front. 180 degrees your left. 270 degrees is behind you. 360 degrees is the same as 0 degrees. In fact there is no difference between in front and behind.
Distance	The distance of the sound source from you. The radius in polar coordinates. The higher the distance is, the higher the attenuation will be. Also, high distances mean high delays. Realistic, isn't it?
Gain	If you want a great great delay, the easiest way is to place the sound source far away. But then maybe you can't hear it due to de great attenuation. With this parameter (in dBs) you can amplify the sound before it begins the trip to your ears. It is sometimes fun to set a huge L/R separation, place the souce just over one of the ears, and set a great gain. You will hear a lot of saturation in this ear and the other one normal and delayed. :)

Isn't it stupid?

RemComp + RemGate

A compressor and a noise gate. If you don't know what a compressor or a gate is look Geonik's comp/gate documentation. This ones have the special feature that they analize the sound that comes from the auxbus and affect the one that goes through them (I'm not sure if this is clear enough).

P a r a m e t e r s

Threshold	The effect works only if the auxbus signal is above or below this value (it's given in dB). If it works above or below depends on which of the two effects are you working with.
Ratio	If the effect is working, the input signal (not the one that comes through the auxbus) is attenuated. The attenuation ratio is given by this parameter.

Attack	To avoid clicking, the effect follows an A-R curve when working. To explain this I will use an example: Imagine you are testing the gate. If the signal that comes from the auxbus is under the threshold, the gate is "closed". This means that it's working: The input signal is attenuated. When the auxbus signal becomes loud, the gate starts "opening". The attenuation ratio decreases to 1:1, but instead of doing it automatically, it takes the time defined by this parameter.
Release	When the auxbus signal becomes quiet again, the gate begins to "close". But it needs the release time to completely close. This A-R curve is used both to avoid clicking and strange distortions.

No pictures this time

The 'mediana' filter

I think it's all explained in the web. Anyway, here it is the description of this effect: A non-linear filter (this means you can't calculate a frequency response, the response depends on the shape of the signal) which picks n samples, sorts them in order of amplitude, and returns the one in the middle. The effect over low-freq sinusoids is a little unheardable deformation in the peaks (the same as a compressor). The effect over short clicks is its complete erasing. The effect over sawteeth is a strange distortion. The effect over squares is a little delay (n/2). The effect over voice samples... well, look for yourself.

P a r a m e t e r s

NumSamples	The number of samples it uses each time. The higher this number is, the heavier on CPU usage will this effect become. Try different values to see different effects.
On/off switch	It was in the Interpolator sourcecode, and I didn't remove it. Useless, it switches between thru mode (with calculations) and normal mode (calculations+output).

ChirpFilter

A filter whose cutoff freq and ressonance changes over time. Something that sounds like chuip-chuip. It also has an ADSR envelope. I love its sound, did I say so before?

P a r a m e t e r s

trigger	The name says it all. Triggering the chirpfilter restarts the internal timer (to control the ADSR), the cutoff and the ressonance.

type	Lowpass, highpass or bandpass. The basic Butterworth filters that come with Buzz.
frequency	The starting cutoff frequency when the chirpfilter is triggered.
ressonance	The starting ressonance when the chirpfilter is triggered. In the bandpass mode it controls the Q of the filter.
frequency speed	The speed in which the cutoff changes. Values over 512 mean that the cutoff increases linearly over time (high values mean higher changes). Values under 512 mean that it decreases over time (low values mean higher changes). If it's 512 the cutoff stays unchanged (as a normal filter).
rez speed	The same as above, but instead of affecting the cutoff freq it affects the ressonance (or Q in the bandpass mode).
attack	These next four parameters let you define the ADSR curve that is followed each time the chirpfilter is triggered. They are given in milliseconds. Attack is the attack time, the time that it takes to reach to the maximum volume (0 dB), beginning in -infinite dB.
decay	The time it takes to reach to the sustain value, beggining in the moment it reachs 0 dB.
sustain	The sustain level. 128 means 100% (0 dB), 0 means 0% (-infinite dB). 64 means 50% (-6 dB).
release	The release time. It's the time it takes to reach the -infinite dB, counting from the sustain point.

ADSR curve

Author	Christiaan Janssen "Ynzn"
Email	cjan5813@alu-etsetb.upc.es
Web	http://members.xoom.com/Ynzzn/Buzz/mainpeix.html

Zephod & Thevider VKR granular

Introduction

This machine is based on the principles of granular synthesis. Granular synthesis simply chops a sample in to little bits, and rearranges the bits realtime. When you choose proper rearrangements you can do anything with the sound you want: play it backward, timestretch it, pitchshift it, etc.

If you really like this machine, or use it in a commercial production (eg: you make money with it), please send the author a donation. A donation can be anything, ranging from a postcard to computer hardware, or maybe even money. Remember: even machinedevelopers need to eat sometimes :-)

Features

- Realtime generation of grains from any sample of the wavetable
- Multiple tracks
- Pitchshifting
- Timestretching
- Sample offset

Basic Usage

Step 1: load a sample in your wavetable.
Step 2: make a pattern for the VKR, and choose your wave in the 'wave' column.
Step 3: enter some notes, and add the pattern to the seq. editor.
Step 4: hit play and play with the sliders!

Advanced usage

There are some notes:
1. The Grainspace value can couse clicks when set too low. This is -not- a bug, but the envelope between the grains is too fast to be smooth then.
2. The same for really short grainlengths.
3. The attack value is the amount of time to get to full volume from the current volume.
4. The decay value is the amount of time to go from the current volume to 0 -after a note-off-

Contact Info

1. IRC EFNet #buzz	Offical download and support
2. **Feedback**	Direct feedback, bug reports and suggestions to stijn@hot.a2000.nl
3. My adress (for donations) is: Stijn Kuipers 2e Passeerdersdwarsstraat 7 1016 XE Amsterdam The Netherlands	

Buzz Links

1. **Buzz Home**	Offical Buzz site
2. **Laser Productions**	Good source of information and Buzz files

BuzzTrack.com Links

1. **New Machines**	Download the newest Machines
2. **Machine Presets**	Download user created Presets
3. **Drum Kits**	Download user created Drum Kits
4. **Buzz Tutorials**	Read various tutorials from BuzzTrack.com
5. **Buzz Songs**	Download Buzz Songs
6. **Buzz Software**	Download Buzz Related Software

Zephod CSynth Synthesizer for Buzz

Introduction

Zephod CSynth is a polyphonic analog modeling **16-track** synthesizer made to be used as a buzz generator. It uses a lot of CPU, but has great features like two LFOs and two envelopes, so you can modulate things like an oscillator´s phase distortion or the cutoff frequency (from -100% to +100%)

Zephod CSynth has **53 global parameters** and **2 track parameters**. Because of the huge amount of the parameters, you will need a high resolution screen (like 1280x1024) to see all the machine´s parameters (buzz doesn´t has a vertical scroll for machines yet :)

You can also load the **GUI**, righ clicking the machine in **Machine View** and selecting **Edit**

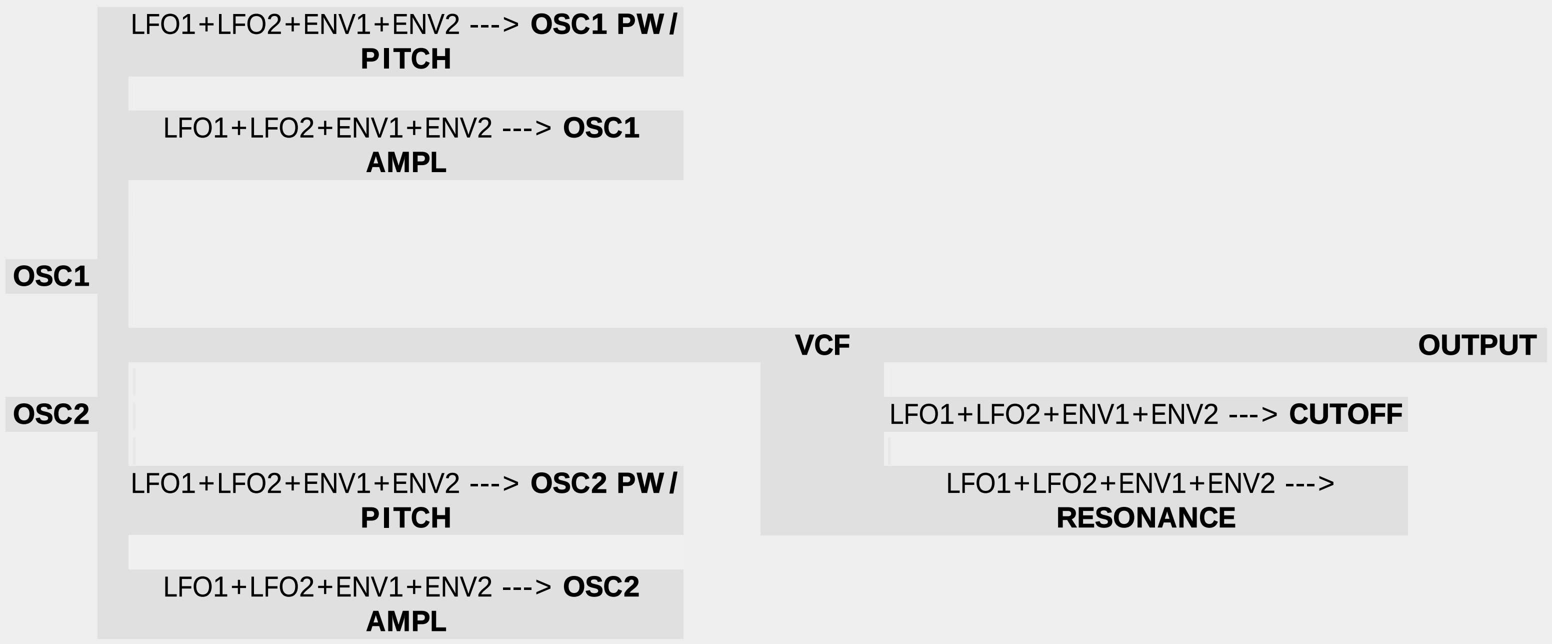
How does it work

Zephod CSynth is based on substractive synthesis. This kind of synthesis "substracts" harmonics to a tone. A **VCF** (Voltage Controlled Filter) is used to do this. Zephod CSynth can produce **4 types** of **waveforms**

Sine	This kind of waveform only has one harmonic. Its frequency is the same as the note played and is used to play smooth tones, such as flutes, bass sounds, etc.
Sawtooth	This waveform is very rich in harmonics and the VCF can affect it drastically. This waveform is the most common for synthetic sounds.
Pulse	On this kind of waveform, the Phase Distotion (PD) can get more dynamic sounds or more character if its modulated.
Random	Generates White Noise. It´s used with VCF and Envelopes to get percussive sounds or the classic "wind effect" with a low-pass filter

The Zephod CSynth Diagram

Here you can see how teh sound is distributed in the Zephod CSynth


LFO1+LFO2+ENV1+ENV2 ---> OSC1 PW / PITCH
LFO1+LFO2+ENV1+ENV2 ---> OSC1 AMPL
OSC1
VCF
OUTPUT
OSC2
LFO1+LFO2+ENV1+ENV2 ---> CUTOFF
LFO1+LFO2+ENV1+ENV2 ---> OSC2 PW / PITCH
LFO1+LFO2+ENV1+ENV2 ---> RESONANCE
LFO1+LFO2+ENV1+ENV2 ---> OSC2 AMPL

Oscillators

There are **3 oscillators** in Zephod CSynth

Main (OSC1)	Base oscillator
Secondary (OSC2)	It has the same tone as the first oscillator, but you can detune it thru the parameters (OSC2 tune and OSC2 finetune) to get a richer sound
SubOscillator	This oscillator is one octave detuned with the OSC1. Its used to get fatness to the sound

VCF (Voltage Controlled Filter)

The Voltage Controlled Filter is the part wich gives the analog feel to the sound. It subtracs harmonics to the output produced in the oscillators. There are three types of VCF

Low Pass	substract harmonics behind the cutoff frequency
Hi Pass	substract harmonics below the cutoff frequency
Off	Deactivates the VCF

Q Control (Resonance)

Q is the cutoff filter´s banwidth attenuation. This gives a fatter sound. Setting Q to the maximum value gives **autooscillation**.

Envelopes

Zephod CSynth has two **ADSR** (attack, decay, sustain and release) envelopes, wich can be assigned to almost every parameter.

Normally it is used with **Volume** or **VCF**

LFOs

The **Low Frequency Oscillators** are used to modulate parameters as the cuttoff frequency, oscillator´s tone (vibrato), pulse width, etc

All the LFO **values** (OSC's PW, PITCH, AMPL, VCF Q, VCF CUTOFF, etc) goes from **-100%** to **100%** (where -100% = negative polarity)

Importing .nti presets

Zephod CSynth can import .nti files (Noisetrekker´ Csynth presets)

To import them, right click the machine in **Machine View** and selecting **Import *.nti**

The import *.nti is available on the GUI too

Contact			
Zephod	stijn@hot.a2000.nl	http://www.buzztrack.com	Code and GUI
DJLaser	djlaser@buzztrack	http://zap.to/djlaser	Help File
New Atom Mine	newatommine@telebot.net	http://www.zylex.net.nz/atom	Preset file (.nti to .prs convertion)

Zephod MX7

By Zephod:
Stijn Kuipers
2e Passeerdersdwarsstraat 7
1016 XE Amsterdam
The Netherlands
stijn@hot.a2000.nl

This machine is -donationware-. If you like this machine, or use it a lot, or even make money with it, please remember that the buzzdevelopers are doing this without being paid. So please be kind, and send me something (doesn't matter what: money, computer hardware, CD's with your music, postcards, whatever) If people send the buzzdevelopers things, they will be more motivated to continue to deliver pro-quality software at around zero cost.

Machine Details

Full Name	Zephods MX7
Buzz Name	MX7
Author	Zephod
Description	FM synthesizer

Global Parameters

To learn more about the meaning of the parameters, I suggest taking a look at yahoo.com and seacrh for FM synthesis. M1, M2, and M3 refer to Operator 1, 2 and 3.

	Min.	Max.	Default	Description
Attack	1	2000	60	Attack in ms
Decay	1	2000	100	Decay in ms
Sustain	0	128	60	Sustain
Release	1	2000	200	Release in ms
M1 Attack	1	2000	60	M1 Attack in ms
M1 Decay	1	2000	100	M1 Decay in ms
M1 Sustain	0	128	60	M1 Sustain
M1 Release	1	2001	200	M1 Release in ms
M1 amount	0	128	60	M1 amount
M1 detune	0	2000	1000	M1 detune

M2 Attack	1	2000	60	M2 Attack in ms
M2 Decay	1	2000	100	M2 Decay in ms
M2 Sustain	0	128	60	M2 Sustain
M2 Release	1	2001	200	M2 Release in ms
M2 amount	0	128	60	M2 amount
M2 detune	0	2000	1000	M2 detune
M3 Attack	1	2000	60	M3 Attack in ms
M3 Decay	1	2000	100	M3 Decay in ms
M3 Sustain	0	128	60	M3 Sustain
M3 Release	1	2001	200	M3 Release in ms
M3 amount	0	128	60	M3 amount
M3 detune	0	2000	1000	M3 detune
Oscillator Wave	0	4	0	Oscillator Waveform
Modulator Waveform	0	4	0	Modulator Waveform
Modulatorroute	0	3	0	Modulatorroute

Track Parameters

	Min.	**Max.**	**Default**	**Description**
Attack	1	2000	60	Attack in ms
Decay	1	2000	100	Decay in ms