Buzé Documentation

Welcome to Buzé!

Buzé is a full featured, portable modular music tracker that works on Windows and Linux (using Wine). Check out the introduction for a brief list of features.

Topics for the documentation:

Please note: This documentation is a work in progress. Some information may be out of date and not entirely correct.

Getting started

Read the FAQ for installation tips, troubleshooting startup problems and more. Also printing out these cheatsheets might be a big help.

Learn how to create and connect plugins in the machine view.

Learn about patterns and pattern formats under basic pattern concepts.

Learn how to create patterns, enter notes and automate plugin parameters in the pattern editor.

Learn how to create pattern formats in the pattern format view.

Learn how to alter the sound of synths and effects in the parameter view.

Review and learn from some general usage tips that have been collected over time.

Tips

Get inspired

Print the whole manual + cheatsheats, and carry it around it while you are on the bus, train or just on a holiday. It pays off, seriously.

Use the Lunar plugins

Buze is shipped with many Lunar plugins (Reverb/Compressor/Gain/Distortion/etc). Use them as much as possible to promote sharing and portability of songs.

Use Note Generators (with) VST plugins

Get used to connecting Note Generators/MIDI Trackers to plugins. A VST plugin is just an effect or an generator, so it needs something extra to have notes send to it.

Learn the keys

Know what the F-keys do. Also Alt-backspace on samples or machines always show properties. ESC always goes back (to the previously state/focused window).

(Multiple) screen layouts

Hold CTRL and drag a tab into another tab. This way you can combine views with one another. Don't forget to store it as a screenlayout, then you can recall the views later using the numpad buttons.

Rename your machines

Select a machine -> Alt-backspace -> (keydown) -> enter name -> ESC

Show hide the paramviews

If you doubleclick a machine, it opens the paramview. Instead of ending up in a neverending open/close clicky-madness, just leave your paramviews open (and hit F12 to hide/unhide them).

Combined machineview

Hold CTRL and click a tab of a parameterview, and release it on top of another parameterview. This way you can combine them.

Templates: Build your typical-studio

Many audioengineers use specific combinations of effects over and over again. The great thing about buze, is that it's modular, but not perse. Once you have a great combination of effects, just select these machines, and rightclick: 'Save Selection As'. Enter for example 'mysuperreverb.armz' and you saved it as a preset. When entering 'mysuperreverb' in the machineview, you will have it under your fingertips.

Remember your favorite plugins/instruments

If you have 1000 soundfonts/presets/synth, and at one day you find the-best-sounding-string-ever, or the-best-preset-ever, you can just rightclick the plugin -> 'Save Selection As' -> 'superstring.armz'. When entering 'superstring' in the machineview, you will have it under your fingertips.

Smart MIDI mapping

Of course you can map everything to MIDI using rightclicks on sliders in the parameterview. But, if you have particular wishes, write your own script in the scripting view. There are example scripts their which demonstrate scripted MIDI-mappings.

Recycle your sound

Any machine can be rightclicked, and its output can be rendered to the wavetable. This is handy in situations of too CPU-intensive songs, or just to refactor yourself into simplicity.

Help what does this machine do?

When opening the parameterview of a machine, press the help icon. It will then try to open a .html, .txt, .pdf file carrying the same name of the machine (without dll). (So if you install plugins, it can help to rename the docs) Also, in any case press F1 in case of doubt.

Keep machineview tidy

For machines which are small details, rightclick it, and press 'minimize'.

Separate notes & automation in 1 pattern

It is handy to have 'note view' and a 'envelope view'. So you can switch between the musical data, and the automations of effects etc.

1. Create envelope columns where necessary (hit CTRL-t on non-note columns)
2. Optionally select an envelope-type using the rightclick contextmenu
3. Hide all non-note related columns using CTRL-k
4. Select all: CTRL-| (several times)
5. Hit CTRL-k to toggle between views..voila!

Reroute notes

Did you know you can reroute notes? If you use Note Generators, you can send its output to other Note Generators, or MIDI Trackers. Also MIDI-plugins can be usefull in this situation.

Send MIDI to your external gear

1. Make sure you have configured at least one MIDI-output in the preferences screen.
2. Now add an 'MIDI Out' Machine and optionally a 'MIDI Tracker' machine.
3. Rightclick the 'MIDI Out' Machine and select a midi-output.
4. Connect the 'MIDI Tracker' to the 'MIDI Out' machine.
5. Click the 'MIDI Tracker' machine and hit some keyboardkeys.
You will now hear some notes being send to your external gear.

Frequently Asked Questions

Answers to questions and solutions to problems that have come up.

Q: Where can I get (human) help?

A: Goto #buze of EFNET with your IRCclient (like mIRC), we will be there .

Q: What are the keyboard shortcuts?

A:Hit F1 as much as possible in any situation. You will get your keyboardshortcuts overview there. Also for a full overview see the keyboard shortcuts manualsection here.

Q: How to get not confused by pattern(formats)?

A: Remember, hitting ESC will always take you back to the main/default pattern.

Q: How can I nest patterns like in Buzz?

A: Add a 'Pattern Player'-machine. Go to the patternformat editor (shift-F2), and tick the 'Trigger' checkbox of the 'Pattern Player'-machine. A column appears, and if you enter '00', a nested patternblock will appear. Hit enter to enter the pattern, and ESC to leave. Once you are inside your nested patternblock, you can hit shift-F2 again to add columns to it.

Q: How do I render audio?

A: Rightclick the master (or any machine), and choose 'Render To'. From there you can render the machine's output to a file, or a wave in the wavetable.

Q: My VST instrument does not make sound?

A: You forgot to connect a 'Note Generator'-machine to it. By doing so, you will also be able to send notes to your plugin in the patterneditor. Optionally a 'MIDI Tracker' can be used. Remember it once, and you're done.

Q: Can Buzé co-exist with Buzz?

A: Yes, theoretically, although the programs have grown apart to such as a degree it may be considered beneficial to maintain two installations.

Q: How to install?

A: Unpack the zip, add plugins and start buze.exe. Check out the documentation for the individual plugin wrappers on how and where to install plugins.

Q: Where can I get Psycle and LADSPA plugins?

- Psycle comes with 58 plugins. Install and copy the contents of PsyclePlugins into Gear\Psycle.
- There are 90 precompiled LADSPA plugins for Windows on the Audacity download page. Install to Gear\Ladspa.

Q: I get this error message when Buzé starts: MI.H: GetInfo() called

A: This message appears when Paniq's Plutonium is initialized. While it is a harmless message, this machine is defunct and can safely be deleted.

Q: I get the error message above (or some other error message), but I cannot press OK or X...

A: You might have (an older version of) Gear/Generators/CyanPhase Buzz Overloader.dll. Please delete it - Buzé will not work when it is there.

Q: What is the deal with the text mode console window that opens on startup?

A: Unless you started Buzé with a /debug parameter, this window appears when Paniq's Pybuzz is initialized. Whenever Pybuzz is used, it will print debug messages and other output to this window.

Q: Buzé crashes on startup...

A: Buzz and Buzé normally crashes on startup for the same reason: unstable machines. While we have attempted to put the most well-known broken machines in blacklist.txt, you will occasionally need to track down unstable machines by yourself. Use a tool such as Dependency Walker or buze.exe /debug to help finding problem DLLs. If you encounter a machine that you think ought to be placed in the blacklist, please let us know by logging a bug report.

Known startup problems:

- The presence of Gear/Generators/CyanPhase Buzz Overloader.dll crashes on startup (solution: delete file)
- Drivers for Samson C01U USB Mic are reported to cause startup crashes (solution: uninstall driver)
- Device named "ASIO DirectX Full Duplex Driver" is known to cause problems, and may crash on startup unless audio driver buffer size is set to 512 (solution: use another driver)
- Audio driver initialization may halt if there are non-existent (USB) ASIO devices in the registry. (solution: delete unused ASIO device entries in the registry under HKEY_LOCAL_MACHINE\SOFTWARE\ASIO)

Q: I get noise bursts...

A: You are apparently using a broken machine. Don't do that. Find out which machine causes the noise and delete it. If you encounter a machine that causes noise bursts, please let us know by logging a bug report.

Q: Can Buzé use VSTs?

A: Yes. Buzé now supports VST and VSTi natively, with access to most parameters and features directly. By default, plugins are loaded from Gear/VST, but this can be changed in Preferences -> Plugins -> VST Paths.

Buzé also supports VSTs through the Buzz wrapper: The Polac VST plugins enable both VST and VSTI. Make sure you have latest version though.

Q: I have problems with a machine I use a lot in Buzz...

First make sure you have all DLLs the machine needs. Some machines use runtime libraries from Visual Studio, such as msvcp60.dll, msvcrt.dll and these must exist on your system, either in Buzés program folder or in c:\windows\system32.

Polac VST & Polac VSTi:

- The index.txt entries are not expanded with subdirectories like in original Buzz.
- When selecting a VST instrument from its right click menu, the machine changes its name in Buzz. This hack has not been successfully implemented in Buzé.
- If PVST crashes when you connect it to the master, make sure you are using the latest version of both PVST and Buzé.

All WaveOut/WaveIn machines, Jeskola/Polac ASIO In/Out machines:
These machines depend on functionality in Buzz' native wave output drivers, which are not available in libzzub. Use the built-in Audio Input and Audio Output plugins instead.

BTDSys PeerCtrl, BTDSys PeerCtrl 'Basic', BTDSys LiveJumpHACK:
These machines use known offset hacks and were attempted patched, but caused the wrapper to fail altogether. Therefore, they are currently blacklisted as they will cause trouble no matter what.

ld mixer:
Please see the separate ld mixer 1.03 checklist page.

Q: I have problems with a VST I use a lot in another sequencer...

Please report any problematic VSTs in our bugtracker. Our VST wrapper is a work in progress.

Q: Does Buzé crash if I use any of the machines utilizing hacks?

A: No, most likely not. The Buzz machine wrapper patches several known hacks to prohibit crashes.

For a list of the currently patched plugins, have a look at Gear/Native/buzz2zzub.ini, which controls patching options per DLL. The .ini-file has some documentation, which shortly explains the purpose of each patch option. Patched plugins are probably safe to use, although if their core functionality is based on hacks they may not (yet) function properly.

If you are a machine author using hacks in your Buzz machine (not recommended), make sure the supported patches are enabled in buzz2zzub.ini for it to work as expected in Buzz. The wrapper does not support any direct GUI-hacks, such as responding to correct window messages, but instead focuses on the memory offset hacks used by several popular plugins.

Q: How can I report a bug?

A: The one place to keep bug reports and feature requests is in the ticket database. To enter tickets, you will need a Sourceforge account.

Q: Did you buy the Buzzlib license from Oskari Tammelin and is it possible to use libzzub in commercial projects?

A: The libzzub/Buzé project owns a Buzzlib license, which was aqcuired mainly to distribute binaries legally. None of the buzzlib code has been used in libzzub or Buzé. Unfortunately the overall licensing scheme is quite messy. Buzé is released under the same license as WTL. libzzub is LGPL, which means you can use the replayer engine in a commercial project as long as any changes you make to libzzub are returned to the community.

libzzub uses a Buzz-wrapper to provide support for original Buzz machines. The wrapper is GPL-licensed, except Oskaris original headers which have the following notice:

```
// This header file may be used to write _freeware_ DLL
"machines" for Buzz
// Using it for anything else is not allowed without a
permission from the author
```

I am not a lawyer, but I think you should talk to Oskari if you want to use libzzubs Buzz-wrapper commercially.

Q: What's the difference/purpose/relationship of all this Buzé, BuzzRMX, Aldrin, Neil, Armstrong, libzzub, libneil-stuff?

A: Aldrin, Buzé, Buzz RMX and Neil are different Buzz clones with entirely different design goals. All four are based on different versions of the same replayer engine.

libzzub is the name of the replayer engine originally split off from an early version of Buzé. Later libzzub went through a major rewrite and renamed to Armstrong. Armtrong has gone through yet another major rewrite since then.

Aldrin is a successor of Jeskola Buzz developed mainly on and for the Linux operating system.

Buzé is a modern remake of Jeskola Buzz for Windows.

Buzz RMX is a 1:1 remake of Jeskola Buzz for Windows. It is forked from an an earlier version of Aldrin and is intended as a pick-up point for developers that want to tweak and extend the original Buzz user interface.

Armstrong is an open source platform independent C/C++ music tracking and sequencing library.

Neil is an active open source fork of Aldrin.

libzzub is the former name of Armstrong. The "zzub" prefix is often seen in the source code still.

Q: How do I pronounce Buzé?

A: It's pronounced with a french accent, preferably in a slightly gay tone. Try yourself: Buh-zeh, buh-zeh, buh-zeh.

Cheatsheets

Of course slight changes can occur from version to version. But here are two cheatsheets which explain a bit of the basics.

MACHINEVIEW

cheatsheet based on Buze v0.7.3, go for info or download buze @ http://www.batman.no/buze

F4 opens the machineview

ALT-BACKSPACE (opens properties, where you can rename your machine etc).

CTRL-TAB or ESC will move your cursor back to the previous window.

Rightclick lets you add your favourite machines/plugins (defined in Gear/index.txt)

Typing on your keyboard will launch the completionbox (to add plugins/templates)

Play a VST/Plugin by playing on your keyboard using Note generators

Clicking 'Midi Focus' will enable you to always trigger notes on a selected plugin using your pc- or midikeyboard

Rightclicking a machine enables you to render the output of a machine to a wav-file or the internal wavetable (so you can re-use it)

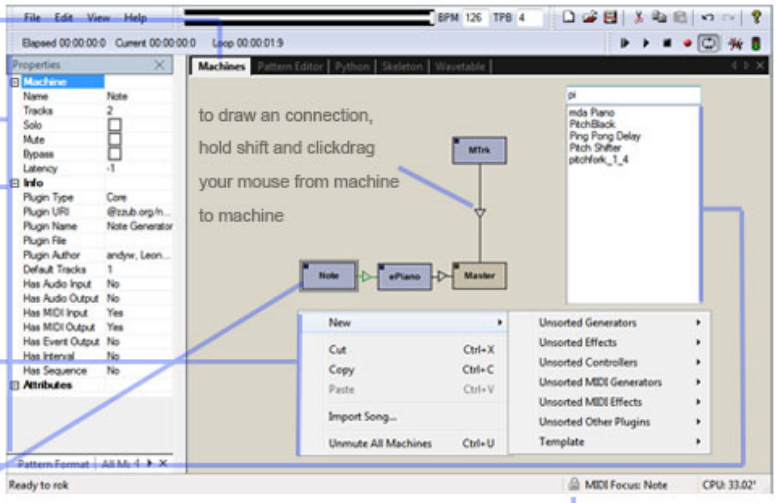
'Save Selection To File' allows you to save selected machine(s) to one 'machine'. (This is called a template, and can be launched from this menu)

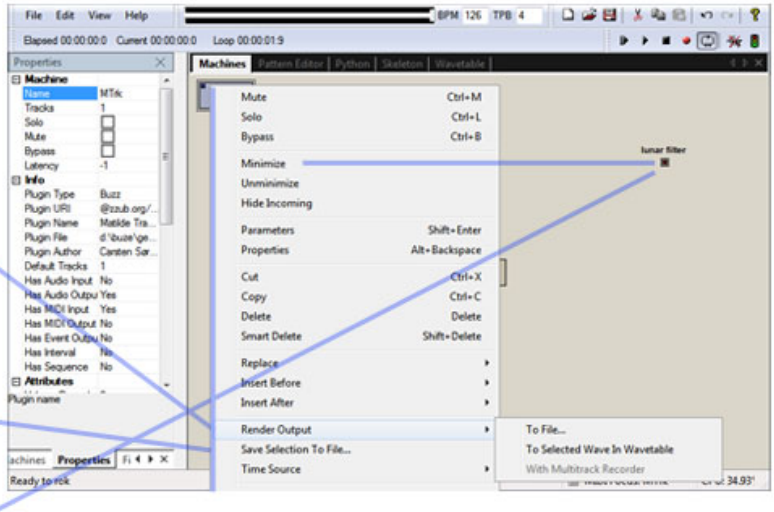
Using the same rightclickmenu, you can minimize plugins to create some order in times of 'pluginchaos'

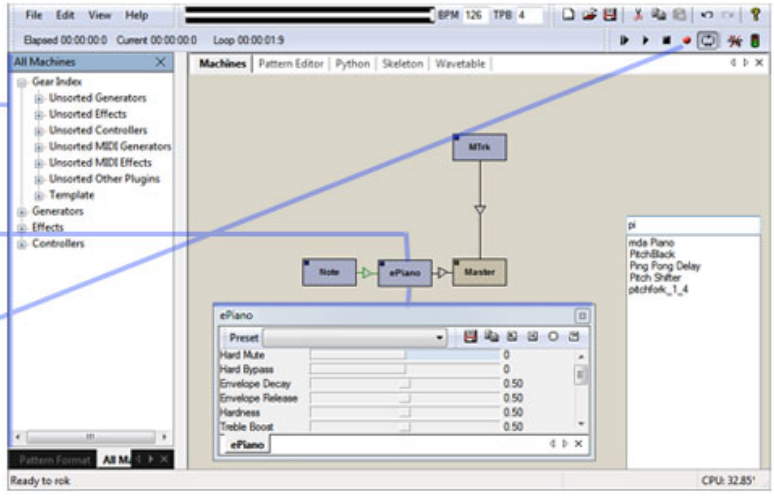
SHIFT-F4 will popup all machines which can be used in buze.

SHIFT-ENTER or doubleclick on a machine will open the parameterview. Move the sliders with your mouse, bind them to your midi controller using rigthclick, and/or press record to record your movements/notes in the pattern editor.

CTRL-Z will always undo your actions in almost any view.







PATTERNVIEW & WAVETABLE

cheatsheet based on Buze v0.7.3, go for info or download buze @ http://www.batman.no/buze

F2 opens the pattern editor

SHIFT-F2 opens the patternformat editor.
Here you can 'design' or define your pattern.
In other words, which pluginparameters
do you want to sequence?
Just tick them using the mouse, or use enter,
and the arrow-keys.
ESC always takes you back to
the pattern editor.

CTRL-T lets you change the view of a
column (sliders,envelopes, notes,etc)

CTRL-Z undos...always

CTRL-B (begin loop) and CTRL-E (end loop)
define the loop region.
Note that 'loop' should be enabled.

If you prefer to 'chain' patterns like oldschool
patterns, use the orderlist.
Here you can insert patterns (after creating
them using the rightclickmenu) using the
arrow up/down keys.

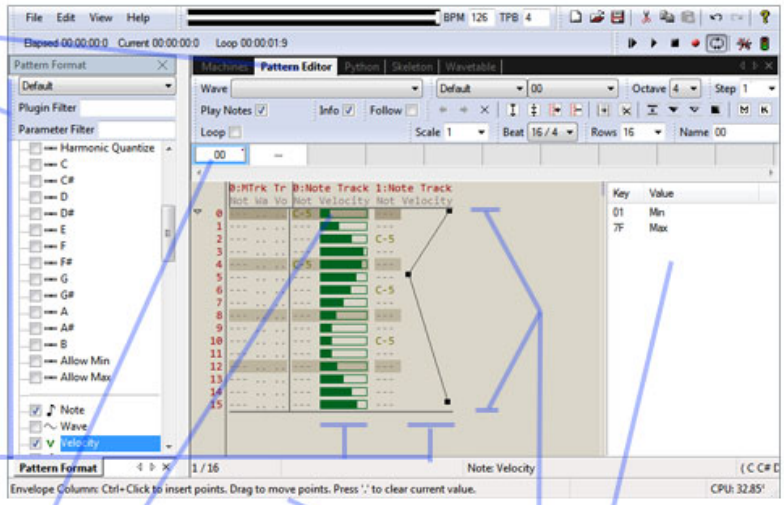
pressing CTRL-space on a column inserts
the current values of all the parameter of its
plugin. A great 'capture' feature.

F9 - In the wavetable you can store samples,
which you can use with for example the
matilde2 plugin.

SHIFT-F9 opens the filebrowser, where you
can load samples. by dragging them to the
wavetable.

Also here you can store shortcuts paths.
You can apply effects on the wave which
is currently selected.
CTRL-Z will always undo.

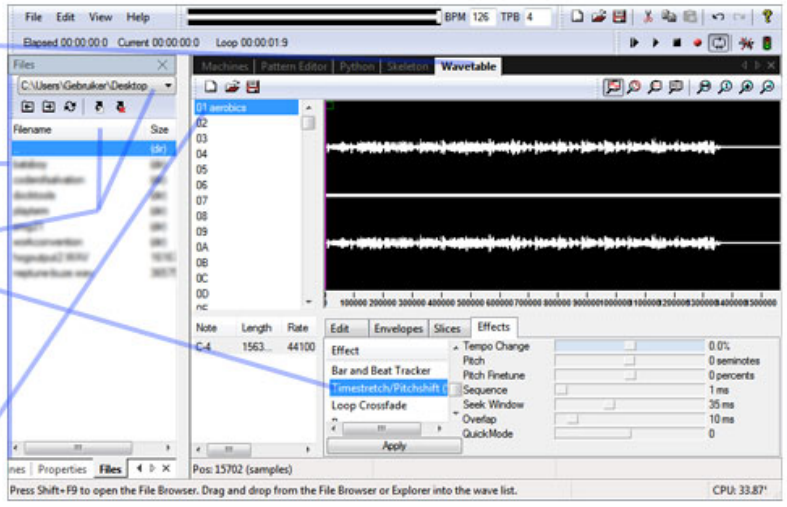
Rightclick lets you configure/launch external
programs to edit samples. (Configure your
waveeditor in the preferences screen)



reading the info can help!

For Ableton/Logic/Fruityloops/Cubase-users: you can just work in big pattern.
Just make sure you hit CTRL-G a couple of times to
make the length of the pattern longer.

For Diehard Trackers: did you know that you can even have multiple patterns with multiple layouts?
Instead of working with one patternformat (the 'Default' layout), you can add
several patternformats, and even nest them!
(by adding the 'Pattern Player' machine in the machineview & patternview)



Global

Buze is built on a framework which resembles Visual Studio in regards to using docks and tabs to organize its views.

Working with tabs, views and docks

- Press F1 in most views to bring up a view with all related keyboard, mouse and MIDI shortcuts. Press F1 twice to show the global help.
- Press Ctrl+Tab and Ctrl+Shift+Tab to cycle through the recently displayed views.
- Views can be undocked or moved by dragging the tab label. An undocked window can be docked by holding down CTRL while dragging the window title.
- The screenset feature helps organizing up to ten different view and dock layouts.

Main toolbars

[TODO: Screenshot + description of every button on all main toolbars]

Main menu

File

New: Clears everything and creates a blank document.

Open: Opens an existing song from disk. Supports .ARMZ, .BMX, .MOD, .IT, .S3M.

Save / Save As...: Saves the current project in .ARMZ file format.

Recent files: Quick access to a list of recent projects.

Save / Save As...: Saves the current project in .ARMZ file format.

Edit

Undo/redo: Steps through history.

Cut/Copy/Paste: Work with the clipboard.

View

Toolbar: Toggles main toolbar visibility.

Status Bar: Toggles status bar visibility.

Main Toolbars: Toggles visibility of other toolbars in the main frame.

Pattern Toolbars: Toggles visibility of toolbars in the pattern editor.

Lock Toolbars: Freeze the toolbars. When enabled, toolbars can not dragged.

Recall screenset: Restores views, windows, sizes and positions from a predefined screeenset.

Store screenset: Saves the position and size of all currently open views and windows to one of the ten available screenset slots.

Pattern Editor: Sets focus to, or opens the pattern editor.

New Pattern Editor: Opens a new pattern editor.

Machines: Sets focus to, or opens the machine view.

Wavetable: Sets focus to, or opens the wavetable and wave editor.

Comment: Sets focus to, or opens the song comment editor.

Pattern Format Editor: Sets focus to, or opens the pattern format editor.

Pattern List: Sets focus to, or opens the pattern list.

Analyzer: Sets focus to, or opens the signal analyzer.

All Machines: Sets focus to, or opens the machine browser.

Files: Sets focus to, or opens the file browser.

CPU Meter: Sets focus to, or opens the CPU Meter.

History: Sets focus to, or opens the history view.

Themes: Changes the current theme

Preferences: Sets focus to, or opens the preferences view.

The status bar

The two last columns of the status bar display the currently selected MIDI-plugin and current CPU usage.

Machine View

The machine view helps creating and structuring plugins in a graph.

Working with plugins

- Click a plugin for single plugin selection.
- Hold ctrl and left click to toggle selection state for single plugins.
- Left click+drag in the background to create a selection rectangle with a new selection.
- Hold shift and left click+drag in the background to create a selection rectangle that extends the selection.
- Zoom with ctrl+mousewheel.
- Left ctrl+drag the background to offset all plugins.
- Use ctrl+arrows and ctrl+shift+arrows to move the selected plugins.
- Middle click a connection to disconnect.
- Left click an event connection to edit its parameter bindings.
- Double click a plugin to open its parameter view - unless overriden by the plugin itself. Press shift+enter to always see the parameter view.
- Refer to the plugins properties to set polyphony/number of tracks and custom attributes.
- Plugins listed in the All Machines View and the machine view right click context menu are read from index.txt. The "Unsorted" entries contain plugins which are not listed in index.txt.
- Click+drag on an audio connection triangle to alter the connection volume.

Working with plugin groups

- To create a group, choose "Create New Group" from the background context menu.
- Or select one or more plugins and/or groups, choose "Create Group From Selection" from the context menu.
- To move a plugin or group between existing groups, use the context menu option "Move Plugin(s) To Group".
- Double click a group to display its contents. Press Escape to go back to the parent group.
- Copy/cut/paste of plugins and plugin groups works as expected

Creating plugins

There are several ways to add new plugins to a song.

1. Go to the All Machines view by pressing shift+F3, or from the main menu, View -> All Machines. Plugins listed in this hierarchy can be dragged into the machine view.
2. Right click in the machine view background and select a plugin from "New" in the popup menu.
3. Right click an existing effect plugin and select a plugin from either the "Replace", "Insert Before" or "Insert After" popup menus.
4. Right click in a connection triangle and select a plugin from "Insert" in the popup menu.
5. Click somewhere in the background and start typing, or click on a connection and start typing to insert.

Connecting plugins

To connect any two plugins, hold down shift and click and drag from the plugin you want to create a connection. While dragging and holding the mouse over a plugin, a popup menu shows which types of connections can be created.

The types of connections are:

Connection type	Connection data	Connection requirements
Audio connection	Multi-channel, 32-bit floating point audio data	Source plugin has zzub_plugin_flag_has_audio_output. Target plugin has zzub_plugin_flag_has_audio_input. Click and drag on the connection triangle to adjust the volume on all channels.
MIDI connection	MIDI messages	Source plugin has zzub_plugin_flag_has_midi_output. Target plugin has zzub_plugin_flag_has_midi_input and exposes at least one virtual MIDI device.
Event connection	Parameter values mapped 1:1 from a list of parameter bindings	Source plugin has zzub_plugin_flag_has_event_output and at least one controller output parameter. Target plugin needs no special flags. Only one or more parameters. Click on the connection triangle to edit the current event parameter bindings.
Note connection	Note messages consisting of note, wave index and amp	Source plugin has zzub_plugin_flag_has_note_output. Target plugin needs no special flags. Only one or more note parameter.

Background context menu

Right+click somewhere in the background to bring up a context menu:

New...: Creates a new plugin or plugins from a template. The plugin and template hierarchy displayed in the "New"-submenu is parsed from Gear/Templates/*.armz and Gear/index.txt. Plugins not listed in index.txt are placed in separate menus for "Unsorted Generators", "Unsorted Effects", "Unsorted Controllers", "Unsorted MIDI Effects", "Unsorted MIDI Generators" and "Unsorted Other Plugins".

Cut/Copy/Paste: Works with plugin selections on the clipboard.

Create new group: Creates a plugin group with stereo input/output.

Import song...: Import an existing .ARMZ from disk into the current project.

Unmute all machines: Yes.

Plugin context menu

Right+click a plugin or plugin selection to bring up its context menu:

Mute/Solo/Bypass: Toggles muting options.

Minimize/Unminimuze: Toggles plugin display size.

Hide Incoming: Toggles whether to hide incoming connection wires. Can help keeping the graph more tidy.

Parameters: Shows the plugin parameter view.

Properties: Shows plugin properties.

Cut/Copy: Works with plugin selections on the clipboard.

Delete: Deletes the plugin.

Smart Delete: Deletes the plugin and keeps existing connections between plugins before and after the deleted plugin.

Replace: Replaces the machine with another machine **Insert Before/Insert After**: Creates a new machine on the connection before or after the selected plugin.

Render Output: Render the output of the selected plugin to disk or wavetable as fast as possible. Fails if the song has looping patterns causing the song to run infinitely.

Save Selection To File: Saves the selected plugins to an .ARMZ on disk. Similar to "Copy", but saves to disk instead of the clipboard.

Time Source: Sets the source of BPM/TPB and other timing info for the selected plugin. This is useful when a project plays patterns in different tempos at the same time. Plugins know only about a single tempo, so this setting decides what tempo a plugin should track for its internal calculations.

Create Full Pattern Format: Creates a new pattern format with all parameters from the selected plugin.

Create Simple Pattern Format: Creates a new pattern format with only note and velocity parameters from the selected plugin.

Connection context menu

Right+click on a connection triangle to bring up a context menu:

Disconnect <type>.: Disconnects the specific connection type.

Copy/Paste: Works with the audio connections volumes.

Insert...: Inserts a plugin here.

Properties: Shows connection properties. For setting up the range of audio channels transfered between the source and target plugins.

Parameter View

Typically, double clicking a machine in the machine view opens its parameter view. Some machines may override the double click action and provide its own user interface instead. In that case, right click the machine and select "Machine Parameters".

VSTs open their user interface embedded in the parameter view, hiding the built-in Buzé-sliders. Press the circle-shaped button on the toolbar to toggle the various parameter view display modes.

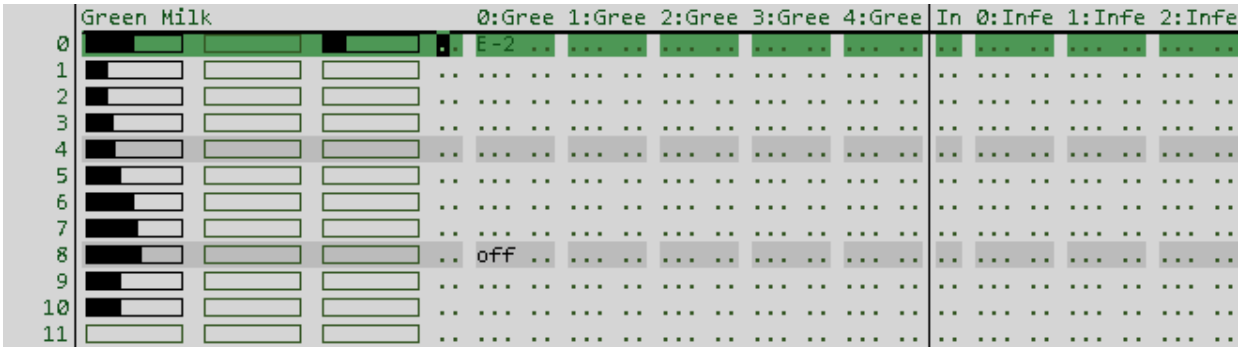
The parameter view can also be opened for audio connections: right click a connection and select "Audio Connection Parameters".

Working with parameters

- Click a parameter name to "gray it out". The randomize function only applies to non-grayed parameters
- Use the mouse and arrow keys to select and change the parameter value
- Use Ctrl+Left/Right and Shift+Left/Right for faster slider movements
- Press ENTER to type a numeric parameter value into a textbox
- Ctrl+Up/Down selects previous/next preset from the preset toolbar

Pattern Editor

The pattern editor is basically a tool to edit numbers in a grid. The numbers are organized in distinct columns refering to plugin parameters, repeated by rows for each step of time. Columns can be rendered in different ways, depending on the type of parameter.



Part of a pattern with sliders, raw values and notes columns.

Working with patterns

- Use Enter and Esc to navigate into and out of subpatterns. Pressing Esc in the root pattern focuses the orderlist.
- Press Shift+F2 or choose "Pattern Format Editor" from the context menu to add or remove columns in the current pattern format.
- Press Alt+Backspace to change pattern properties such as name, length, resolution and looping
- Press Ctrl+H to toggle horizontal (like IT) or vertical (like FT) editing modes
- Press Alt+| to cycle automatic column entry modes for note editing. Toggles between none/either/both of wave and volume.
- Press Ctrl+Num plus/Ctrl+Num minus to add/remove tracks (voices). Note that this modifies the pattern format and will affect all patterns of the same pattern format
- Press Ctrl+F5 to play the current pattern only
- Press Ctrl+F6 to set the replay row - indicated by the little arrow pointing downwards next to the row number - and play the current pattern only from the current position
- Press Ctrl+Shift+F5 to play the current pattern from the replay row
- Press Alt+Enter to bring up the parameter view for the plugin under the cursor
- Colors can be modified in theme files.

Working with trigger columns

- Make sure the Infopane is enabled in the pattern editor toolbar to display the embedded pattern list while the cursor is on a trigger column.
- To navigate in the embedded infopane, use Alt+Up/Down/Page Up/Page Down to move the cursor.
- When a pattern format is selected in the infopane, press space to insert a new pattern at the cursor location. Or make a selection first to create the pattern of a given length.
- When a pattern is selected in the infopane, press space to insert the selected pattern at the cursor location. Or make a selection first to insert the pattern repeatedly throughout the selection.
- Press Ctrl+Num plus/Ctrl+Num minus to add/remove trigger columns tracks (voices).
- Trigger columns can be added to any pattern format by adding the "Trigger" parameter of a Pattern plugin to the pattern format.

Trigger columns are accommodated by the right pane showing a list of all formats and patterns that can be triggered, including itself. This means a pattern could trigger itself - don't!

Working with the pianoroll

- To enable the per-plugin pianoroll, add a "Note Meta" parameter from a plugin with notes to a pattern format.
- Voices for the pianoroll must be allocated manually; add at least one note parameter to the pattern format "as usual", and use "Add track" in the pattern for more voices.
- Click and drag in the pianoroll background to paint new notes.
- Click the top or bottom of a note to stretch the duration of a note.
- Click and drag a note to change its pitch and time.
- Shift+Click to select and work with more than one note.
- Note that pianoroll editing will cause reorganization of the underlying pattern data, and may not always lead to the desirable results.

Working with envelopes

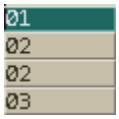
- Any numeric pattern column can be rendered as and have its values interpolated linearly in realtime:
- To enable interpolation for a column, choose "Column Interpolation" -> "Envelope (Linear)" from the pattern editor context menu.
- To enable the envelope editor for a column, choose "Column Editor" -> "Envelope" from the pattern editor context menu.

Basic keyboard and mouse operations

Operation	**Action**
Arrows	Move cursor
Arrow up/down+Arrow left/right	Move cursor diagonally
Page Up	Move cursor up 16 rows
Page Down	Move cursor down 16 rows
TAB	Move cursor to next track
Shift+TAB	Move cursor to previous track
Home	Move to beginning of line
Ctrl+Home	Move to beginning of loop
End	Move to last column
Ctrl+End	Move to end of loop
(keyboard notes)	Edit notes
Shift+(keyboard notes)	Edit notes, chord mode
Ctrl+Scroll wheel	Change pattern display resolution
Shift+Scroll wheel	Change font size
Drag+Drop selection	Move selected pattern data
Drag+Ctrl+Drop selection	Copy selected pattern data
Drag selection+Right click	Stamp selected pattern data

Column types

Pattern columns are usually rendered as value columns or trigger columns by default. Select different rendering modes via right click -> Column Editor or Ctrl+T. Any column can be collapsed with Ctrl+K, and uncollapsed with Ctrl+Shift+K.

Column type	Parameter types		Special actions
Value column	Note, switch, byte, word		Space toggles note meta on notes, or pastes the current parameter value. Ctrl+Drag mouse to slide the value.
Trigger column	Any column with pattern trigger flag		Space pastes pattern from pattern list. Special2 clones the pattern under the cursor. Special3 expands/collapses. Special5 goes to the pattern under the cursor. Special6 clones the pattern under the cursor and goes to the new pattern.
Slider column	Byte, word		Ctrl+Click to draw slider values.
Button column	Switch		Ctrl+Click or space toggles the button state
Envelope column	Byte, word		Ctrl+Click to insert/move envelope points. This column type is only a visual representation for the underlying numbers, linear column interpolation must be enabled for playback.
Pianoroll	Note meta		Click+drag to insert new note, or click to select+move single notes. Shift+click to select+move multiple notes. Drag from near the note edges to move the start/end of the selected notes.
Note Matrix	Note meta		Click to toggle single-hit notes.

Pattern properties

Press Alt-Backspace to display and edit properties of the current pattern.

Name: Changes the pattern name

Length: Changes the pattern length

Resolution: Changes the pattern resolution. Also known as rows per tick, or lines per row. Determines how fast the pattern is played back.

Looping: Toggles pattern looping. Looping patterns should not be put in the order list as this would make the song of infinite length, causing hard disk recorders to fail.

Loop Begin: Sets pattern loop begin row.

Loop End: Sets pattern loop end row.

Track parameter: Name: Sets the label used to describe this track in the pattern editor.

Track parameter: Mute: Stops playing pattern events in this track.

Pattern editor toolbars

Right click in the pattern editor toolbar area to toggle visibility of individual toolbars.

Wave toolbar: Changes the currently selected waveform

Format/Pattern toolbar: Two dropdowns: the first lists all pattern formats, the other lists patterns based on the selected pattern format. For navigating other patterns.

Octave toolbar: Changes the current octave for note editing.

Step toolbar: Changes the number of rows the cursor will skip when entering a value.

Play Notes toolbar: Toggles whether to play notes when a note is entered in a note column.

Info toolbar: Toggles whether to display the info pane.

Follow toolbar: Toggles whether the primary pattern editor should follow the order list.

Pattern Loop toolbar: Toggles looping of the current pattern.

Pattern Scale toolbar: Hides rows in the current pattern.

Pattern Beat toolbar: Sets the current pattern beat coloring.

Pattern Rows toolbar: Changes the current pattern length.

Pattern Name toolbar: Renames the current pattern.

Pattern context menu

Right+click on a pattern to bring up the context menu:

Pattern Create: Creates a new pattern.

Pattern Clone: Creates a new pattern as a copy of the current.

Pattern Delete: Deletes current pattern.

Pattern Properties: Displays editable pattern properties for length, resolution, track names, etc.

Pattern List: Opens the pattern list view.

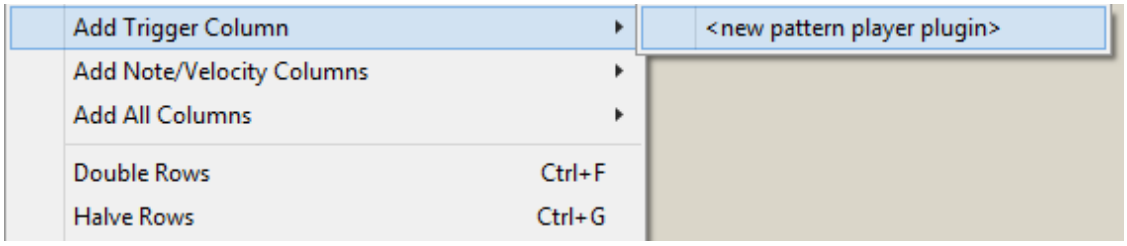
Pattern Format Create: Create a new, blank pattern format.

Pattern Format Clone: Create a new pattern format based on the current.

Pattern Format Delete: Delete the current pattern format and all patterns.

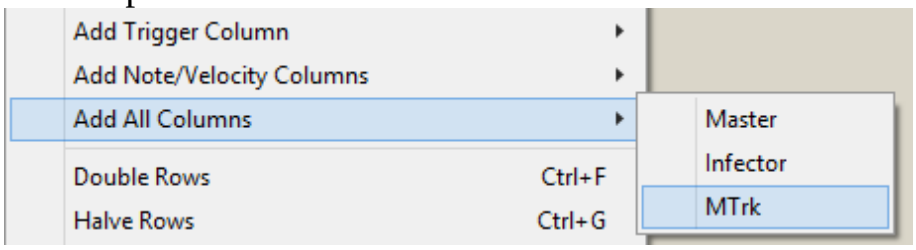
Pattern Format Editor: Add or remove columns in the current pattern format.

Add Trigger Column: Inserts a column for triggering sub patterns from the current pattern format. Choose to create a new pattern player and/or a new track, or reuse an existing.



Add Note/Velocity Columns: Inserts only note and velocity parameters from a specific plugin into the current pattern format.

Add All Columns: Inserts all parameters from a specific plugin into the current pattern format.



Double Rows: Inserts a blank row after every row.

Halve Rows: Deletes every second row.

Double Length: Makes the pattern twice as long.

Halve Length: Cuts off half of the pattern.

Editor in New Pattern Editor: Opens a new pattern editor **Link Scrollbars To**: Select another open pattern editor which will have its scrollbars locked against this pattern editor.

Column Editor: Toggles column rendering mode. See the column type table above.

Column Interpolation: Toggles column interpolation mode. Determines how the pattern player interprets parameter values in patterns: Normal (Absolute) is the default and means pattern values are effective immediately. Normal (Inertia) slides towards the current pattern value value over 4 rows. Linear (Envelope) interpolates the value between the two nearest pattern events before and after the current position. The column interpolation update frequency is the same as the current pattern resolution.

Machine Parameters: Shows the machine parameter view for the plugin under the cursor.

Show Orderlist: Only in the primary pattern editor.

Transform context menu

Press Ctrl+Right Click to bring up the transform context menu. Quick rundown on the transforms:

Random From -- randomizes the selection from the set of all values existing in that column currently. So easy to set a constraint, just type some values in. But if you type one value in twice, it will be twice as likely to appear.

Shuffle -- takes all values in selection and randomizes which time-event has that value. The positions dont change, just the values

Gradiate fills in between all points in the selection existing.

Thin does the "window blinds" effect. deletes stuff at a modulo **Repeat** is like "Echo" stuff repeats at a phase, each new event becomes the repeater

Unique removes repeats

Scale takes a "In Range" and maps it to an "Out range"

Fade does the obvious.. needs a toolbar thingy for some of this stuff to select if it works on volume[greens] or not **Rotate** rows moves N chunks to the top and wraps around. rotate rhythms rotates the set of distances between each event. so if you hold down the hotkey for it, it will give all rhythmic permutations possible given those distances, and will wrap around eventually depending on how many columns it is **All To First** sets all values in the selection to be the value at the top of the selection **First To Last** replaces all instances of the value at the top of the selection, with the value at the bottom of the selection. IF the value at the top of the selection is a novalue.. nothing there, then all blank values get filled in by the value at the bottom of the selection **Remove First** deletes all of one at top of selection. If you have multiple columns selected, the previous three will skip all columns that dont have the same [pluginid, group, column]. because the "top" value only applies to same column types.

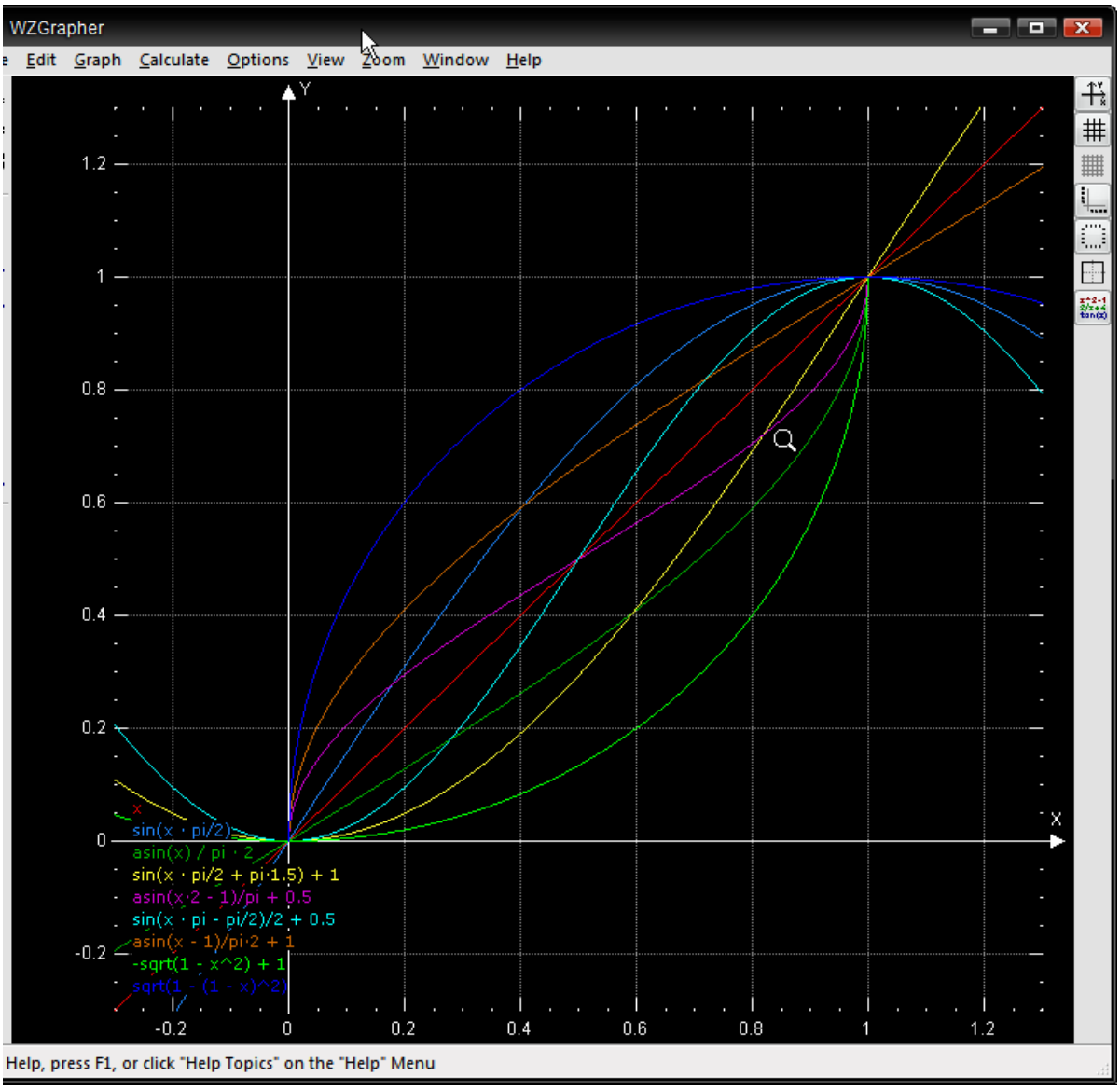
Replace waves uses the wave from the toolbar.

Notelengths will clip stuff... min, or max even

Track swap has some special behaviour and its gonna be real useful. If you make a selection and track swap it, only columns that are shared in both the leftmost and rightmost track, get swapped so if you selected in MTrk... starting on track 1's note column, ending on track 4's volume column then three columns would be swapped between track 1 and 4 note, wave, volume. works in reverse too. like if you selected all of track 4 and only part of track 1. so for trigger cols, same .. just select the edges.

Row swap .. swaps the row at top of the selection with the bottom.

Curvemap is hardcoded .. you dont enter strings (im not sure if anyone is crazy enough to want to even) , but the 8 presets should be alright. The curve index # in the dlg is from top to bottom same as screenshot:

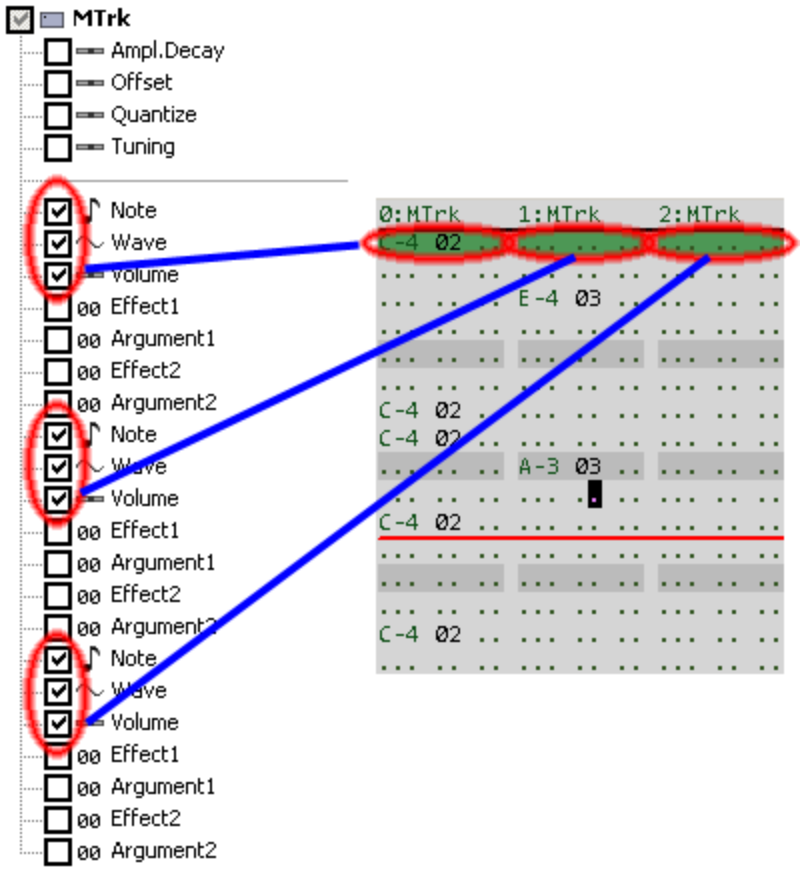


The K transposer

```
< zeffii_> also i have no idea how to use the K tranposer :)
< Megz> there is 2 modes you can be in
< Megz> chromatic and harmonic
< Megz> if the checkbox is unchecked, you are in chromatic
< Megz> if it's checked you're in harmonic
< Megz> when it is unchecked, the key always has 12 tones
< Megz> so if its unchecked and you choose a key signature, it
loads that signature then inserts any missing degrees
< Megz> from a default 12-tone set called "chromatic keyed"
< Megz> when it is Checked, you can have less than 12 tones
selected
< Megz> the Rightmost radial group which has no label means an
inactive tone
```

< Megz> when in harmonic mode and you transpose, each note in the set moves to the next note in the set
< Megz> so to do a simple test
< Megz> check the box, and set it to "C", "Ionian (Major)"
< Megz> then transpose C E G. will become D F A.
< Megz> notice that when it's unchecked you cannot select inactive tones
< Megz> cause their radials are grayed out

Pattern Format View



A pattern format relates plugin parameters with pattern columns.

The pattern format view has a few elements:

The pattern format dropdown shows a list of all pattern formats in the current project.

Type the beginning of the name of a plugin and/or parameter in the **filter textboxes** to show only the interesting parameters.

The plugin parameter list shows list of all the plugins and parameters with a checkbox next to each parameter. The checkbox toggles visibility of the parameter in the currently selected pattern format.

Working with pattern formats

- Any plugin parameter can be part of a pattern format.
- Unchecking a pattern format column will remove the column and its values from all patterns.
- To create a pattern format with nested patterns, check the Trigger and/or Transpose parameter on instances of the Pattern Player plugin.
- Bring up the pattern format view from the pattern editor with shift+F2, or right+click inside the pattern editor and choose "Pattern Format Editor" or from the main menu View -> "Pattern Format Editor".
- Track parameters are automatically added and removed from the current format when changing the number of tracks (voices) from the pattern editor.

Order List

The order list contains patterns to play in sequence in the current project. It exists as a widget inside the primary pattern editor, where it is toggled from the pattern editor context menu -> "Show orderlist".

Working with the orderlist

- Press Enter to ascend into the selected pattern for editing. Focus is returned to the orderlist by pressing Esc from a root pattern.
- The "Follow"-toolbar and the associated checkbox in the pattern editor controls whether the primary pattern editor should always display the currently selected pattern in the orderlist.
- There should not be any looping patterns in the orderlist; pattern loops override the orderlist.
- Patterns are played through entirely before advancing to the next pattern in the orderlist. Remember to set the pattern length accordingly.

Orderlist context menu

Right+click on a pattern to bring up the context menu:

Insert Order: **Clone Pattern**: **New Pattern**: **Remove Order**: **Delete Pattern**: **Select Pattern**: **Cut/Copy/Paste Order**: **Pattern Properties**: **Render to Wave**: **Queue**:

Set Loop Begin: **Set Loop End**: **Play**:

Wave Table

The wavetable gives access to an internal cache of waveforms available to tracker plugins.

There are four sections in the wavetable view:

The instrument list has 200 slots for sample-based instruments.

The wavelevel list contains samples in an instrument.

The wave editor allows for basic editing of samples.

The edit, envelope and slice tabs.

Working with the wavetable

- Use a tracker plugin such as Matilde Tracker and FuzzPilz UnwieldyTracker to play samples from the wavetable in a project.
- Drag files from Windows' Explorer, or open the File Browser inside Buze and drag waveforms from there.
- Preview samples by double clicking or playing notes on the keyboard. The File Browser can also preview samples directly from disk.

The wavetable toolbar

Clear wave: Clears the currently selected instrument and all samples in it.

Save wave: Saves the first sample of the currently selected instrument as a WAV to disk.

Scale: Samples: Tells the wave editor to display the scales in samples.

Scale: Time: Tells the wave editor to display the scales in minutes and seconds.

Scale: Ticks: Tells the wave editor to display the scales in ticks, based on the global BPM/TPB.

Scale: Hex: Tells the wave editor to display the scales in hex-percent, compatible with Matilde and UnwieldyTracker's "sample offset" commands. The hex scale always goes from 0-FFFF, relative to the sample length. The last two digits can be discarded on Matilde Tracker. I.e, for the scale xxyy, Mtrk: 09/xx, and Utrk: xxyy

The instrument context menu

Right click in the context menu to show the context menu:

Properties: Opens the instrument properties.

Export And Open With External Editor: Saves the current instrument to disk and opens it with the wave editor specified under Preferences.

Re-Import: Used with the Export-feature above after editing in the external editor.

The wavelevel context menu

Right click in the context menu to show the context menu:

Properties: Opens the wavelevel properties. For setting looping points, base note, etc.

Delete: Deletes the wavelevel.

Add blank level: Adds a new blank wavelevel.

File Browser

The file browser shows audio files and archives of supported formats on the computer, supports live preview playback and can load samples into the wavetable.

The file browser can be opened with Shift+F9. Or press ENTER when a slot in the wavetable has focus.

Supported file formats

The file browser uses the stream plugins in Armstrong for live preview. The stream plugins use libsndfile or libmad to load and play files directly from disk. libsndfile supports WAV, AIFF, FLAC, AU and many more formats, whereas libmad is a library for playing back MP3.

The file browser uses the sample import API in Armstrong for browsing sample archives and loading sample data into the project. The import API supports most regular wave formats via libsndfile and libmad, and also (non-previewable) samples from .mod/.s3m/.it/.xm and .drumkit files as archives.

Supported archive file formats can be browsed like any directory. In order to load a sample from disk or from an archive, just drag the file into a slot in the wavetable. Or press ENTER to load the sample into the currently selected wavetable slot.

The file browser toolbar

Previous: Goes to the previously visited path.

Next: Goes to the next visited path.

Refresh: Refreshes the contents of the current directory.

Add path: Adds the current directory to the shortcuts dropdown list.

Remove path: Removes the currently selected path from the shortcuts dropdown list.

All Machines View

Plugins listed in the All Machines View and the machine view right click context menu are read from index.txt. The "Unsorted" entries contain plugins which are not listed in index.txt.

Machines can be dragged from the All Machines View into the Machine View.

Preferences

Audio preferences	
Mixer Threads	Number of worker threads to use for multithreaded mixing. When this is set to 1, a simpler single-threaded algorithm is used for mixing.
Output Device	Currently selected output/playback device.
Mixing Rate	Samplerate
Latency	Buffer size used for mixing. 4 buffers are used in total.
Master Output Channel	For output devices with more outputs than a stereo channel. The master sends its output to this stereo channel pair.
Input Device	Currently selected input/recording device.
MIDI preferences	
Enable MIDI Output Devices	Enable or disable detected MIDI output devices on the system. These devices are accessible e.g via the MIDI Output plugin.
Enable MIDI Input Devices	Enable or disable detected MIDI input devices on the system. These devices are accessible e.g via the MIDI Input plugin.
GUI preferences - Global	
VU Meter Speed	Sets the global VU falloff speed.
VU Update Rate	Sets the global VU update rate.
Show Accelerators	Injects the current keyboard bindings into all toolbars, menus and context menus.
GUI preferences - Machine View	

Default Zoom	
Disable Machine Skins	
Scale by Window Size	
Machines Default Minimized	
GUI preferences - Pattern Editor	
Font	Changes the fixed-width font used in the pattern editor.
Font Size	
Default Pattern Length	Default length of new patterns.
Default Value Entry Mode	Horizontal = Cursor moves to the right, down at the last digit Vertical = Cursor moves down
Horizontal Scroll Mode	
Vertical Scroll Mode	
Sticky Selection	
Note-Off String	
Note-Cut String	
Background for a Note	
Background for a Byte	
Background for a Switch	
Background for a Word	
Trigger Column Width	Number of characters reserved for a pattern trigger column.

Format/Pattern Creation Mode	
Default Scroller Width	
Pattern Naming Mode	
Right Click Mode	
Subrow Naming Mode	
GUI preferences - Wavetable	
Default Wave Editor	

Press "Apply" in the preferences view to enable any changes.

Plugins

Plugins are key to get the most out of Buzé. Internal plugins expose unique features of Armstrong, and external plugin formats such as VST and LADSPA open a world of thousands of synths, effects, MIDI-utilities, analyzers and more.

Types of plugins

The broad categories of plugins could be arranged as following:

- Built-in core plugins
    - Audio device plugins
    - MIDI plugins
    - Streaming plugins
    - Recording plugins
    - Connection plugins
    - Pattern player plugins
    - Controller plugins
- Plugins wrappers
    - VST
    - Buzz
    - Psycle
    - LADSPA

Refer to the plugin references for more details about the specific built-in plugins.

Types of plugins - by feature

A plugin author/wrapper will decorate his plugin(s) a combination of flags to tell the engine about its features. Among other things, the flags indicate what other plugins they can connect to and the type of data flowing between the plugins. There are four distinct types of connections: Audio, MIDI, events and notes.

Because a plugin can perform almost any combination of audio/MIDI/event processing, it could be useful to understand the meaning of the individual flags and how the user interface responds to these flags.

Flag	**User interface response**
has_audio_input	The plugin accepts incoming audio signals. Multi channel input is allowed. When creating an audio connection, the user must choose which range of channels to connect to.
has_audio_output	The plugin can generate or alter audio, and sends its audio to connected plugins marked has_audio_input. Multi channel output is allowed. When creating an audio connection, the user must choose which channel range to connect from.
has_midi_input	The plugin accepts incoming MIDI connections. A plugin with this flag can expose more than one virtual MIDI device, and the user must decide which device to connect to when creating the connection from another MIDI plugin. All Buzz machines have this flag set, forwarding any incoming MIDI signals to a fixed "Buzz MIDI Device" that represents the Buzz machines' MIDI capabilities. F.ex it is possible to connect a MIDI tracker to any Buzz machine, and also VSTs and Psycle plugins support incoming MIDI connections. Not all Buzz machines implement MIDI-support, and some (most?) plugins need to have MIDI support enabled through Properties.
has_midi_output	The plugin sends outgoing MIDI signals. Can be connected to MIDI devices on plugins with the has_midi_input flag. A plugin that combines has_midi_input and has_midi_output could do MIDI filtering, or perform other types of MIDI altering on the fly.
has_event_output	

	The plugin sends parameter changes through event connections. Plugins with this flag are peer plugins, in a natively supported way. This type of plugin can expose one or more "hidden" event-parameters, in addition to its public parameters seen in the parameter view. This hidden parameter comes in to play when making event connections. F.ex an LFO-plugin that combines this flag with the has_interval-flag would expose such a hidden value-parameter. When bound to another parameter on a plugin through an event connection, the second parameter would be modified at a given interval. However, a transpose-plugin could also use this flag to expose a series of hidden parameters, each acting as a connectable modifier for altering notes on-the-fly according to its public parameters. BTD's Peer machines for Buzz use a different technique for its peering capabilities, and should not confused with this kind of native peer support. This is a one-way type of connection, in the sense there is no has_event_input-flag, since an event connection could be made to any plugin that has parameters - which most plugins do.
has_note_output	The plugin sends notes through note connections to any plugins with a note parameter.
is_connection	Internally, a connection is implemented as a plugin. Connection plugins are not rendered as boxes in the machine view, but they still appear e.g in the Pattern Format View, and other plugin-lists. Connection plugins are usually named such as "Audio1", "Midi2", "Event3", and have parameters which can be automated by MIDI and/or added to pattern formats.
is_sequence	The plugin implements its own tempo, and can be used as a tempo source by other plugins.
has_interval	This flag allows plugins to determine the internal processing chunk size. It is used by plugins that need to interrupt the graph processing to play notes or change parameters with sample exact precision. Native peer LFO's use this flag to determine the interval of parameter updates.
is_stream	The plugin plays streams, e.g from disk or the wavetable. The flag triggers special processing during song seeking in order to resume the stream correctly.
mono_to_stereo	Less used, legacy, obscure, internal. Subtle or no impact on the user interface

plays_waves uses_lib_interface uses_instruments does_input_mixing no_output control_plugin auxiliary is_root offline	experience.

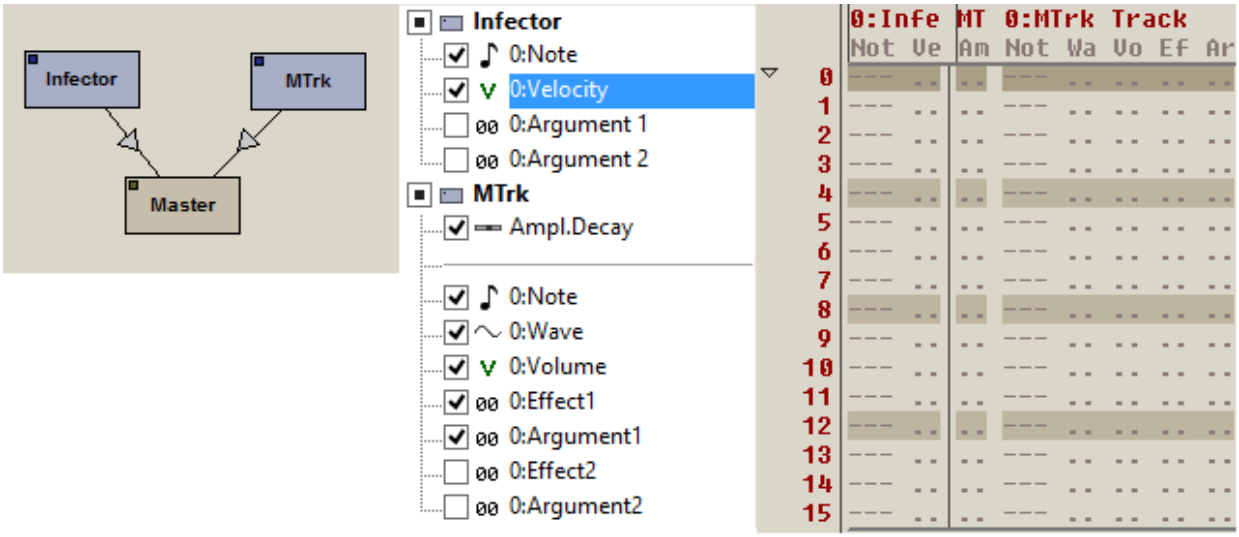
Plugin parameters

A plugin has 5 parameter groups, or sections of parameters which are usually refered to by their group index number.

Virtual parameters	group 0	Parameters defined by the engine on all plugins for toggling global plugin states: Mute, Soft Mute, Bypass and Soft Bypass. The "Hard" versions mute/bypass permanently until toggled manually. The "Soft" variants mute/bypass until a parameter change.
Global parameters	group 1	Global plugin parameters defined by the plugin author.
Track parameters	group 2	Track plugin parameters defined by the plugin author. Track paramerers are repeated for the number of plugin tracks.
Controller parameters	group 3	Not visible from the parameter view, and neither user controllable in the regular sense. Only event plugins have controller parameters, which can be connected to parameters on controlled plugins.
Meta parameters	group 4	Parameters defined by the engine on certain plugins. These parameters do not affect the audio in any way, but offer a method for the user interface to add special "handles" in the song data to support a richer user experience. A "Note Meta"-parameter is added on plugins with note parameters. Buze uses this to enable the inline piano roll or note matrix. A "Wave Meta"-parameter is added on plugins with wave parameters. Buze plans to use this to enable an inline wave editor and recording facility.

Patterns

Patterns are the primary means to both automate plugin parameters and sequence sub patterns. Every pattern is based on a template called a "Pattern Format". The pattern format specifies which plugins and parameters can be automated in patterns based on it. Pattern formats can be edited at any time, and any change made in a template will immediately affect all depending patterns.



After using the machine view to create some plugins, use the pattern editor to create and edit patterns. Use the pattern format editor to adjust in detail which columns should appear in the patterns.

When starting a new project, the pattern editor shows an empty pattern based on an empty pattern format. The quickest way to add some columns is using the options for "Add Trigger Column", "Add Note/Velocity Columns", "Add All Columns" in the pattern editor context menu. Keep in mind these operations operate on the pattern format, such that if the project contains other patterns based on the same format, added columns will appear in them as well.

Track columns

Some plugins have "track parameters": a group of parameters that can be repeated/duplicated and mapped to some internal concept of tracks in the plugin. By changing the number of tracks in the Plugin Properties, more or less parameters will become available in the pattern format editor.

Once some track parameters have been added to a pattern format, it is possible to quickly add or remove entire tracks from the current pattern format by pressing Ctrl+Num plus or Ctrl+Num minus.

Matilde Tracker uses tracks to play different notes/samples at the same time. FSM Infector uses tracks for polyphony. Pattern Player plugin uses tracks to play different patterns at the same time.

Trigger columns

Trigger columns are based on plugin parameters as any other columns. However, there is only a single plugin that has a trigger typed parameter: the builtin Pattern Player plugin.

The pattern player plugin interfaces with the world through its parameters, and Buzé implements a ton of specialized features to help working with the particular trigger parameter.

The default project

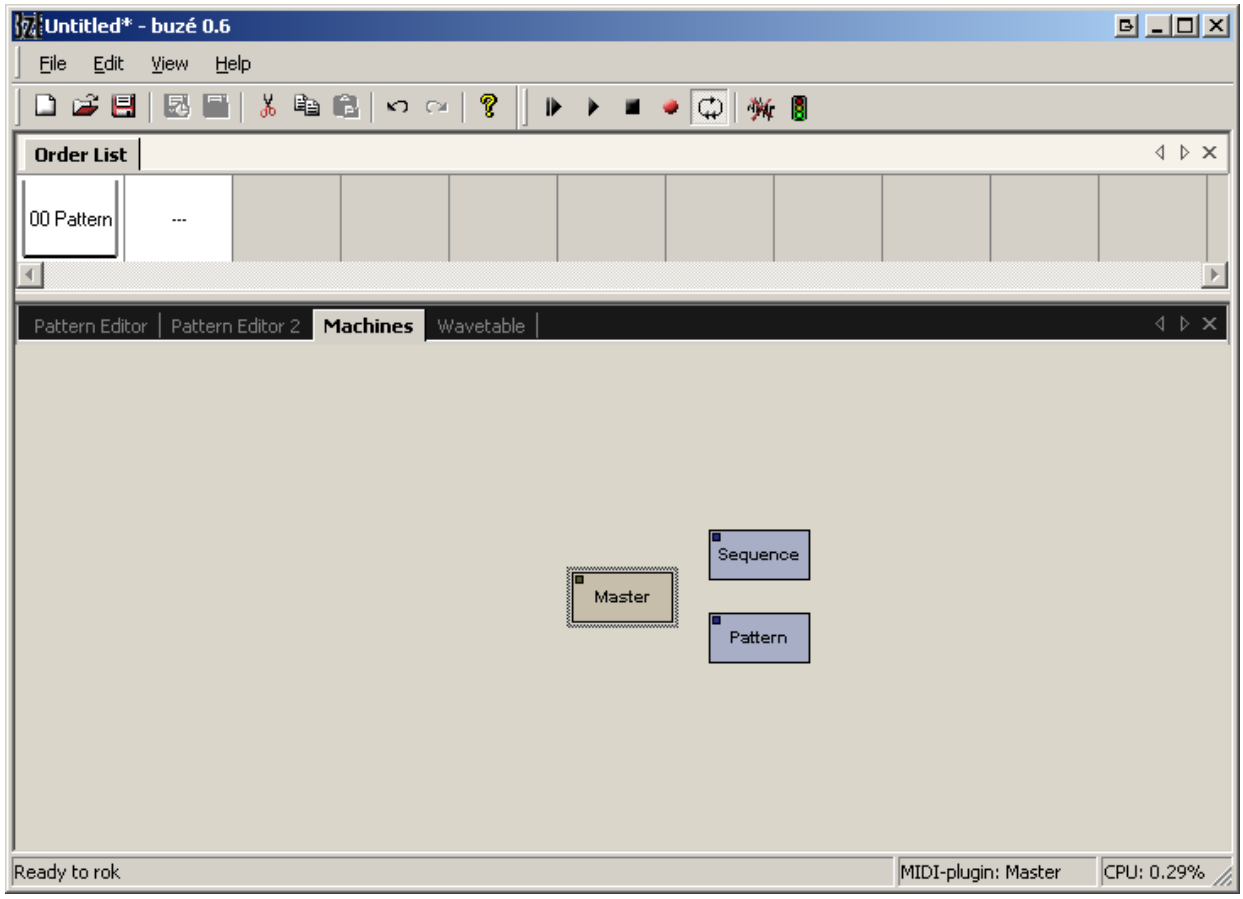
When starting a new project, either upon program startup or selecting File -> New from the menu, an empty, default song is created from a template. The template chosen for the default document depends on the Preferences setting for "Create Default Pattern Player".

The default startup template: No default pattern player

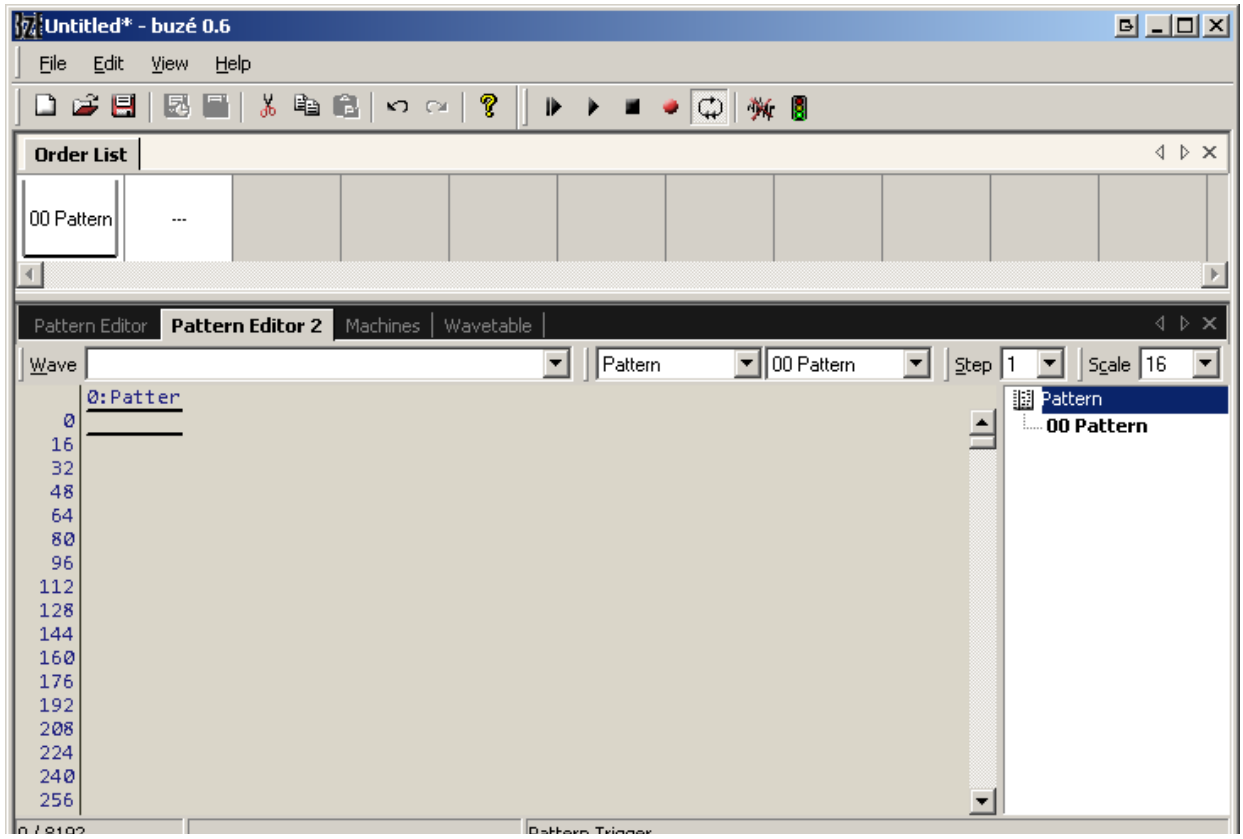
The default setting for "Create Default Pattern Player" is "Off", which creates the simplest possible setup: a Master plugin, an empty pattern format and pattern of length 16. This mode is suitable if there is no need or desire for immediate pattern nesting, and/or maximum control. Naturally, the user can create pattern players manually when the need to use nested patterns arises.

The advanced startup template: With a default pattern player

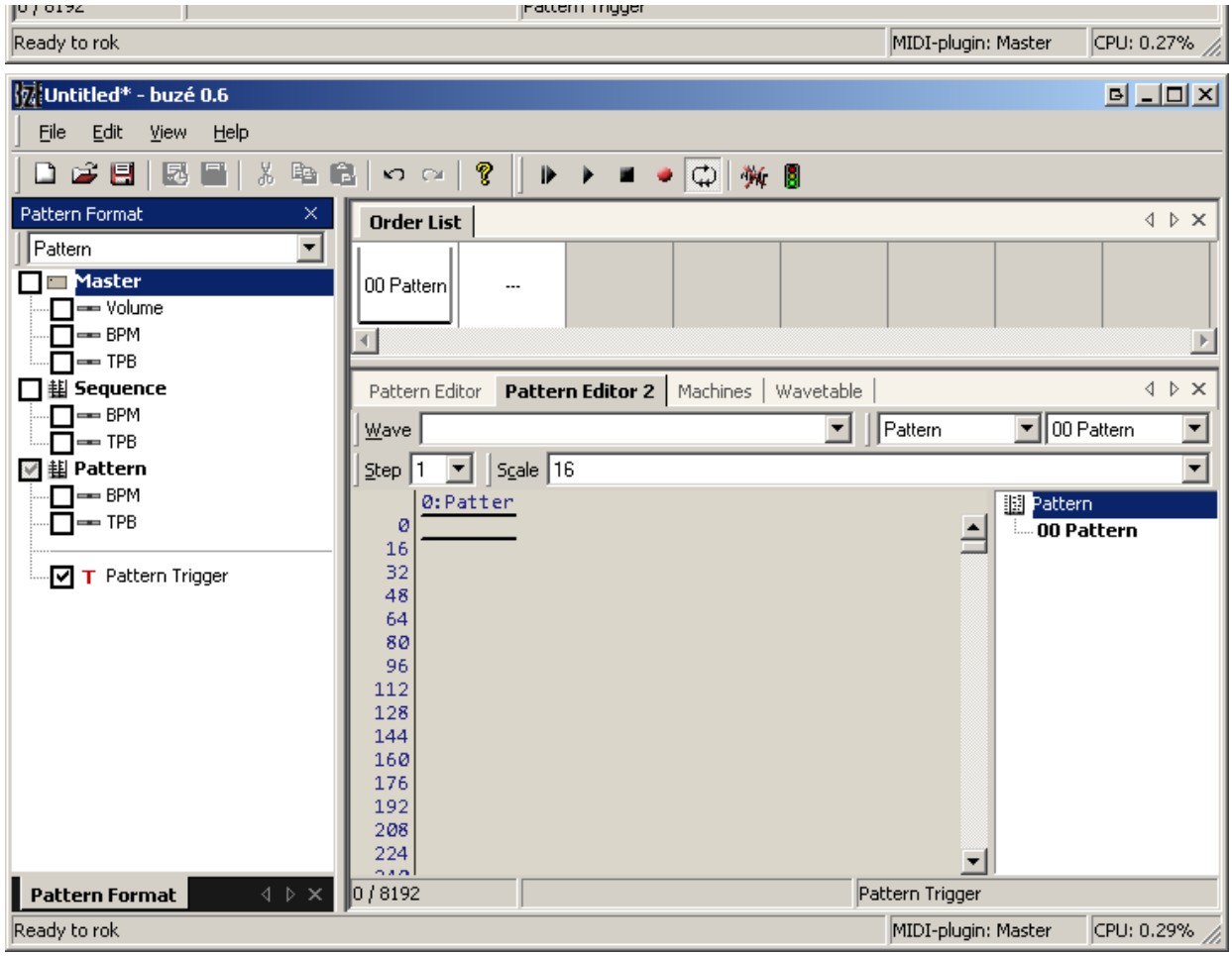
When "Create Default Pattern Player" is turned on, the default document is prepared for using nested patterns. In addition to the master, a Pattern Player plugin is created, its trigger parameter is added on the default pattern format, and the default pattern length is set to 8192. This default pattern is ready for nested pattern sequencing.


Untitled* - buzé 0.6
File Edit View Help
Order List
00 Pattern

Pattern Editor
Pattern Editor 2
Machines
Wavetable
Sequence
Master
Pattern
Ready to rok
MIDI-plugin: Master
CPU: 0.29%


Untitled* - buzé 0.6
File Edit View Help
Order List
00 Pattern

Pattern Editor
Pattern Editor 2
Machines
Wavetable
Wave
Pattern
00 Pattern
Step 1
Scale 16
0:Patter
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
256
Pattern
00 Pattern


Ready to rok
MIDI-plugin: Master
CPU: 0.27%
Untitled* - buzé 0.6
File
Edit
View
Help
Pattern Format
Pattern
Master
Volume
BPM
TPB
Sequence
BPM
TPB
Pattern
BPM
TPB
Pattern Trigger
Order List
00 Pattern

Pattern Editor
Pattern Editor 2
Machines
Wavetable
Wave
Pattern
00 Pattern
Step
1
Scale
16
0:Patter
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
Pattern
00 Pattern
Pattern Format
0 / 8192
Pattern Trigger
Ready to rok
MIDI-plugin: Master
CPU: 0.29%

Pattern formats and patterns

A pattern format is, in its simplest form, a list of selected parameters from any of the plugins in the project. This list defines what columns will be visible in the pattern editor. There can be any number of pattern formats in a project, and any number of patterns based on each of the pattern formats. The user decides length and resolution per pattern.

Patterns can be triggered from the order list, or from trigger-columns in other patterns.

The order list

The order list contains a list of patterns to play in order, with looping points.

The Sequence plugin

The sequence plugin is a user interface "handle" to control the tempo of patterns playing in the order list. There should never be more than one Sequence plugin in a project.

Pattern Player plugins

A Pattern Player plugin has global parameters for tempo and track (voice) parameters for pattern triggers and transposing. Each track (voice) can play and transpose a pattern. By adding more tracks (voices), it can play more patterns.

By default, a pattern player inherits its tempo from the global sequence tempo, but offers parameters for setting BPM and TPB independent of the global tempo. There can be more than one pattern player in a project, each with optionally different tempo.

Using MIDI

In order to use external MIDI input or output devices, they must be enabled via the MIDI settings in Preferences. Enabled MIDI devices can be accessed by using the MIDI Input or MIDI Output plugins.

Bind MIDI controllers to plugin parameters:

MIDI controllers can be assigned to individual plugin parameters from the parameter view. From the machine view, double click a plugin, or press Shift+Enter, or select "Parameters" from the plugin context menu to show the parameter view. Or from the pattern view, press Shift+Enter on a column which belongs to the plugin whose parameters you want to see. Then right-click a parameter and choose "Bind to MIDI controller..." to show the MIDI binding dialog. To create the binding, simply move your MIDI controller and see that the values in the dialog updates and press OK.

Recording live MIDI notes and control changes into patterns

'Step' mode

In this mode, the midi notes will be inserted at the cursor.

- Add a miditracker to the machineview, and connect it to a VST instrument.
- In the pattern view (F2) check the miditracker's note-column in the pattern format window (SHIFT-F2)
- additionaly add some midichannels by pressing CTRL-+

'Live' mode

This mode is also called 'overdub', and inserts notes at the pattern's current play-position.

- Select a plugin in the Machine View to set MIDI focus.
- Enable the recording-button on the main toolbar and press play.
- Notes played through the machine view will be recorded into the first pattern with note columns belonging to the MIDI focus plugin.

MIDI plugins and routing in the machine view

MIDI Tracker and Note Generator

The MIDI Tracker and Note Generator plugins can be used to generates MIDI signals.

MFX and VST MIDI plugins

MFX is a plugin format designed to generate or process MIDI signals. Similarly, VST also supports plugins capable of generating or processing MIDI signals. It is possible to create MIDI connections between MIDI-capable plugins and route signals as desired.

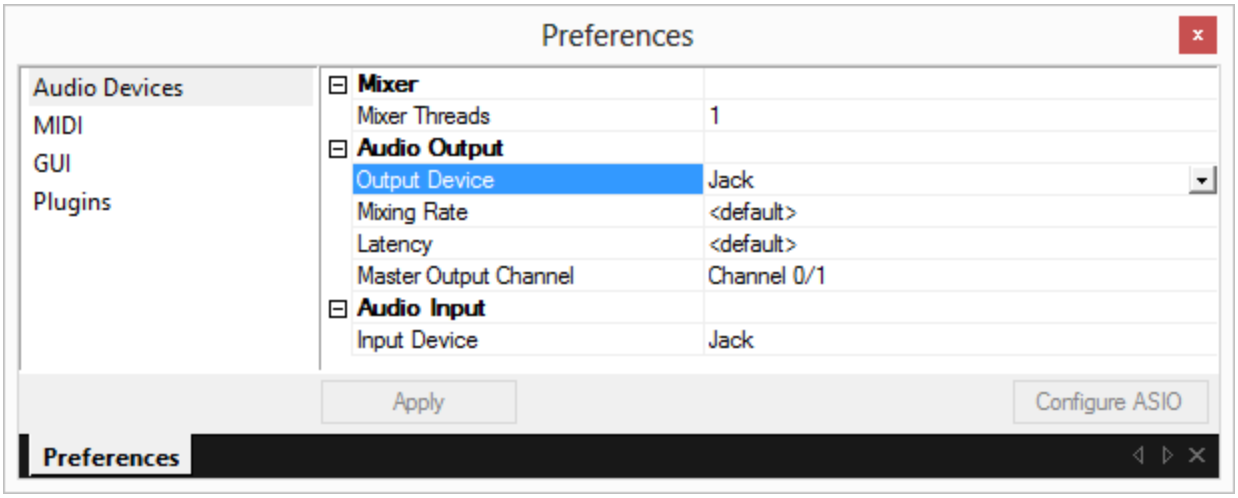
hw2zzub

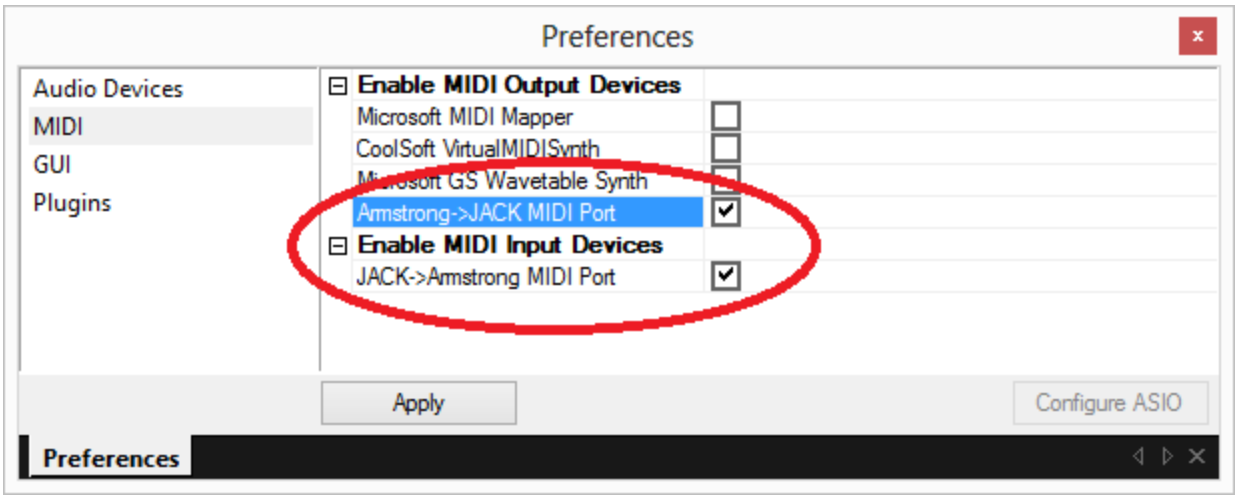
hw2zzub is a native plugin wrapper for hardware MIDI devices. hw2zzub reads profiles for physical devices described in a special JSON-format, which includes mappings for CC-to-parameters and audio device input channels.

Using JACK

JACK is a mechanism to route audio and MIDI in real-time between applications from different vendors. This solves the same problem as Rewire.

If JACK is installed, it can be selected in Preferences -> Audio, and also from Preferences -> MIDI. The MIDI devices work only if the JACK audio driver is selected.



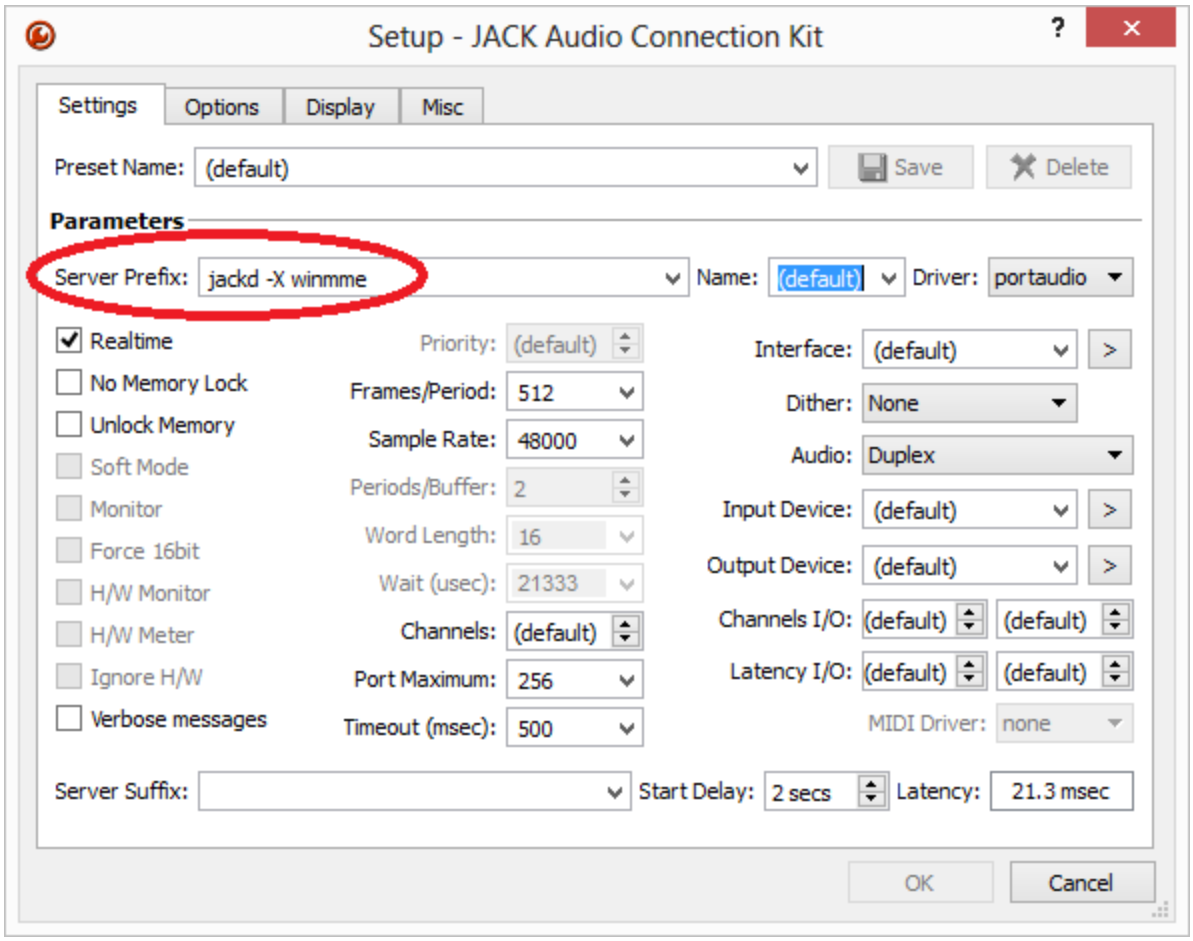


Buzé initializes JACK with 16 audio output channels and 16 audio input channels (16 + 16 audio ports). The channels are automatically routed to the physical JACK audio ports, and leaving any remaining channels for aux.

Buzé initializes JACK with a MIDI input port and a MIDI output port. Note these ports are not routed in JACK by default.

MIDI in JACK for Windows

Make sure to "-X winmme" on the jackd startup options:



JACK in other applications

On a basic level, an application needs only to support ASIO in order to work with JACK: When installing JACK on Windows, it installs a new virtual JackRouter ASIO device by default. Any applications using this audio driver, can send and receive audio from other applications connected to JACK.

The JackRouter ASIO device by itself does not deal with MIDI and synchronization, and will require virtual MIDI cables or a JACK-enabled VST.

Using VST instruments

(Beware Buzz-users; Buze supports PVST, but also implements its own native VST-wrapper which is described here.) VST instruments rely on MIDI messages to play notes, and therefore depends on another plugin to generate MIDI notes.

For example, one could use another VST, such as a VST implementing a pianoroll to generate MIDI notes. Or, as we'll discuss here, use one of the built-in MIDI generators.

To see this in action, first create an instance of a VST plugin. Let's use Joachims TB4005 as an example.

Right click in the Machine View background and find "TB4005" from the context menu. Then right click in the Machine View background and find "MIDI Tracker" from the context menu. Then connect the MIDI Tracker to the TB4005. Notice how the color of the wire is blue, indicating a MIDI wire instead of the black audio wire.

Finally connect the TB4005 to the Master with an audio wire. Click on the MIDI Tracker plugin and play some notes on your keyboard to hear the TB4005.

Many Buzz plugins also support MIDI. It is possible to connect the MIDI Tracker to both the Infector as well as the TB4005 in order to make them play the same notes. Note however for Infector, you need to set up Infectors MIDI channel properties. To see these properties, right click the plugin in the Machine View and select "Machine Properties".

Consider using the more light weight Note Generator plugin instead of the MIDI Tracker.

Peer plugins

"Peer"-technology refers to the capability of controlling plugin parameters from other plugins, as pioneered by BTDSys' contributions to Jeskola Buzz. Armstrong and thus Buze offers a native replacement for Buzz' peers: Internally termed "controller plugins", with data passed via "event connections".

The most basic controller plugin is the Value Generator. While simple, it has a lot of power:

- Control multiple parameters at once
- Can be chained with other controller plugins
- Apply simple arithmetic transforms on chained value generators
- Random-generator

Note that the Value Generator is not used for notes. Instead, the Note Generator plugin is more suitable.

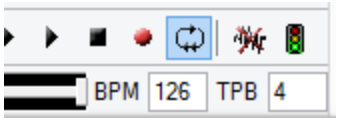
To use this: Create a controller plugin, connect it to another plugin and click on the connection triangle to open a dialog to select parameter bindings. In the dialog, press Enter to accept, or Esc to cancel.

There are several other controller plugins available: LFO Generator, Signal Generator, ADSR Generator, Value Mapper.

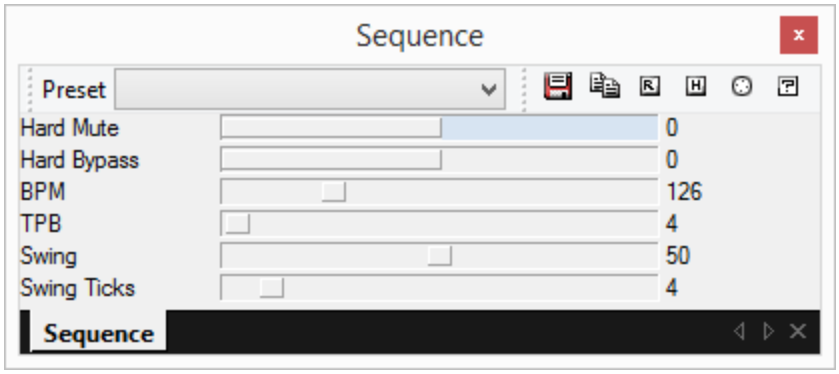
Tempo, time signatures, shuffle/swing

Global tempo

The BPM and TPB textboxes in the main toolbar specify the global tempo.



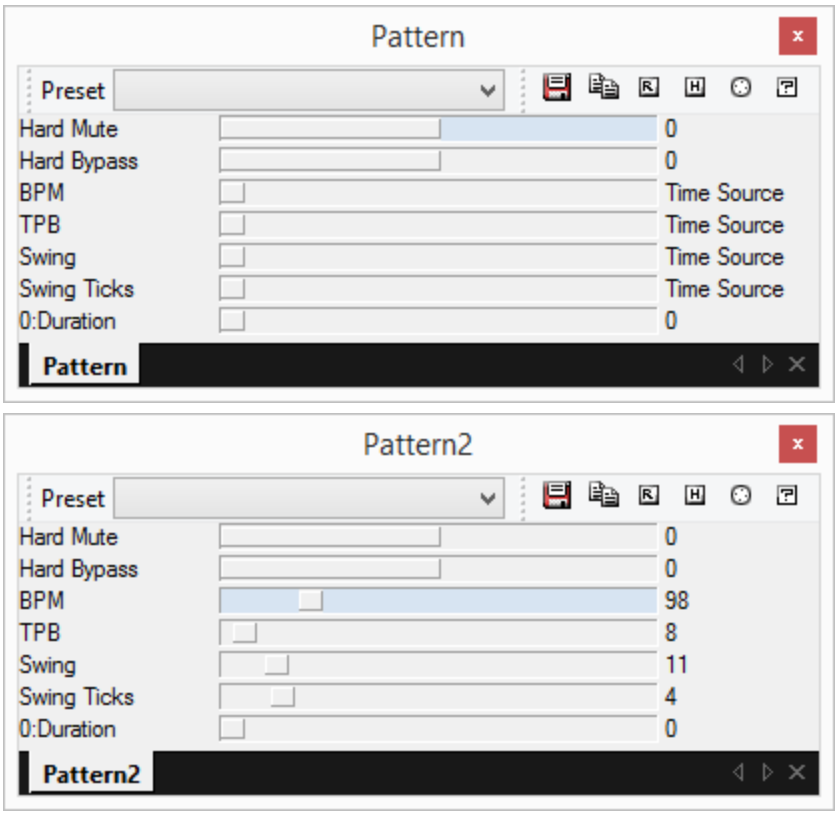
The main toolbar provides access to global BPM and TPB only. If you wish to change the global swing amount/length, you need to create an instance of the Sequence plugin and set its parameters. Likewise, in order to automate the global tempo parameters in a pattern, you need to create a Sequence plugin instance as well. There can only be one Sequence plugin in a project, and it always corresponds to the global tempo.



The global tempo affects primarily the currently playing pattern in the order list, although it is usually inherited in sub patterns by using Pattern Player plugins.

Alternative time signatures

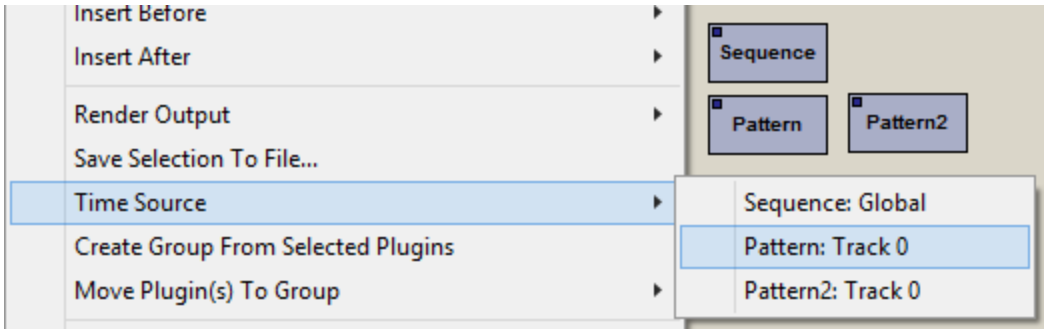
It is possible to override parts of the tempo signature for individual patterns using the Pattern Player plugin.



By default, the tempo parameters are set to "Time Source", meaning the Pattern Player will use tempo information usually derived from the global tempo. The "Time source" usually refers to the global tempo, but in some cases could be set explicitly to refer another Pattern Player plugin.

Using time sources

Time sources allow exposing different tempo signatures to particular plugins. For example, delay and arpeggiators plugins could use tempo information exposed by the host to determine tick lengths. Changing the time source "tricks" these plugins into seeing a different tempo without affecting the host playback tempo.



Time sources can be set on any plugins, although not all plugins use this information. It is possible to chain Pattern Player plugins this way (and create "infinite time loops" - don't!).

Shuffle/swing

Traditionally, to create a swing/shuffle effect one had to automate the tempo to swing the entire sequence, or use a MIDI plugin to swing the timestamps of MIDI notes and control changes.

A new, built-in shuffle/swing feature is also available via the Sequence and Pattern Player plugins through the "Swing" and "Swing Ticks" parameters.

The Swing parameter specifies where the swing occurs, relative to each beat. The default swing value of 50% means the swing occurs in the middle of the beat = no swing. A low swing value will swing early, and a high value will swing late.

The number of rows to swing in a beat is determined by the swing ticks parameter.

When swinging patterns played at an even number of swing ticks (2, 4, 6 etc), the number of rows per beat can always be split in two equally long integer size chunks. At odd number of swing ticks (1, 3, 5), the beat cannot be split into two equally long chunks: theres will always be extra row which falls "outside" of the swing. In these cases the the last row of the beat is played at the target tempo so that the total time of the beat parts matches the "outer beat".

Pattern resolution and TPB

The "Resolution" property on patterns is a TPB multiplier per pattern and affects swing etc during playback.

Customizing Buzé

Buzé is configurable via several external resources.

Keyboard bindings

Most keyboard bindings can be changed by editing a text file called hotkeys.json in the application directory. Please refer to this file for more information. The keyboard bindings are parsed on program startup, so remember to restart Buze after editing hotkeys.json.

Machine index hierarchy (index.txt)

- Gear/index.txt
- Gear/index.plur
- Gear/Include_native_ladspa_psycle.txt

Displayed in the All Machines-view and in the Machine View's "right click -> New" context menu.

Buzé is compatible with Buzz' and Overloader's extended index.txt. Additionally, Buzé's index-parser supports including external files.

(TODO: examples and information about how to write traditional and overloaded index.txt's. add links to popular indexes)

Song templates in the machine index hierarchy

Gear/Templates/*.armz

Create a directory called Gear/Templates and place songs here. They will show up in under Template in the machine view right-click menu after restarting Buze.

Themes

Themes/*

Buzé supports exactly the same theme files as Buzz, with some extensions.

Theme Parameters

Buzz machine compatibility

buzz2zzub.ini

Controls blacklisting and dynamic patching of hacked plugins. Please refer to buzz2zzub.ini directly for detailed information.

MIDI output device aliases

Gear/midi_aliases.txt

GUI layout

Gear/gui.xml

The state of the windowing dock-tab-framework is saved to this path on exit. Not really intended to be edited by users.

Fonts

Download and install alternative monospaced fonts for increased readability in the pattern editor.

- Andalé Mono
- Consolas

Everything else: registry

Audio device settings, MIDI settings, GUI tweaks, toolbar positions. All registry settings should be editable in Preferences.

HKEY_CURRENT_USER\Software\zzub\buze

Master plugin

Global parameters

Parameter name	Type	Description
Volume	word	Master Volume (0=0 dB, 4000=-80 dB)

Track parameters

Parameter name	Type	Description
none		

Description

The Master plugin receives audio from incoming audio connections and passes the audio on to the audio device. The Master plugin also exposes all active MIDI output devices, and passes on MIDI messages from incoming MIDI connections.

The master can optionally be deleted, and users can choose to use separate plugins for its audio and MIDI capabilities: The Audio Output

or Audio Output16 plugins are capable of sending audio to the current audio device. The MIDI Output plugin exposes all active MIDI output devices in the same way the Master does.

Audio Output

The Audio Output plugin lets you send audio to available outputs on the current audio driver.

Global parameters

Parameter name	Type	Description
none		

Track parameters

Parameter name	Type	Description
none		

Plugin attributes

Attribute name	Description
Output Channel	Set a stereo channel on the audio device to send output to

Audio Input

The Audio Input plugin lets you record audio from available inputs on the current audio driver. For example recording from a microphone, or receiving audio from an external MIDI device via ASIO.

Global parameters

Parameter name	Type	Description
none		

Track parameters

Parameter name	Type	Description
none		

Plugin attributes

Attribute name	Description
Record Channel	Select a stereo channel on the audio device to forward input from

File Recorder

The file recorder machine records its input to a stereo WAV file.

To set a file for recording, go to the plugins properties, and use the file selector from the plugins "Stream Source" property.

Global parameters

Parameter name	Type	Description
Enable	switch	Turn recording on/off
Record Mode	byte	Record mode (0=wait for play/stop, 1=continuous)
Format	byte	Output format (0=16bit, 1=32bit float, 2=32bit integer, 3=24bit)

Wavetable Recorder

The wavetable recorder machine records its input to a stereo wavetable slot.

Global parameters

Parameter name	Type	Description
Enable	switch	Turn recording on/off
Record Mode	byte	Record mode (0=wait for play/stop, 1=continuous)
Format	byte	Output format (0=16bit, 1=32bit float, 2=32bit integer, 3=24bit)
Instrument	wavetable index	Wavetable instrument slot to use (01-C8)

MIDI Tracker

The miditracker is modeled after Polac VST and sends MIDI messages via MIDI connections.

Global parameters

Parameter name	Type	Description
Global Command	byte	Global Command
Command Value	word	Global Command Value
Program	word	Change MIDI Program

Track parameters

Parameter name	Type	Description
Note	note	Note to play
Velocity	byte	Velocity of the played note
Note Delay	byte	Delay before note triggering (ticks?)
Note Cut	byte	Delay before note release (ticks?)
Command	byte	Track Command (details below)
Command Value	word	Command value
Parameter	word	Track parameter (details below)
Parameter Value	word	Parameter value
MIDI Channel	byte	MIDI channel

MIDI Tracker Commands

Name	Value	Description
MIDI Message	09	Send a MIDI Message (details below)

MIDI Message

The MIDI Message word is of the form xxyy, where "xx" is the message and "yy" is its value.
Here are the available messages:

Name	Message	Value Range	Description
CC	00-7F	00-7F	Sends a MIDI Control Change (CC)
Pitch Bend Range	FE	00-FF	Defines range of pitch bends
Pitch Bend	FF	00-FF	Sends a MIDI Pitch Bend

MIDI Tracker Parameters

Name	Value	Description
None	30ff	Does nothing
Pitch Bend	30fe	MIDI pitch bend
CC	3000...30fa	MIDI Control Change (CC)

MIDI Input

Forwards MIDI messages from open MIDI input devices (e.g a USB MIDI keyboard) via MIDI connections.

Global parameters

Parameter name	Type	Description
none		

Track parameters

Parameter name	Type	Description
none		

MIDI Output

Sends MIDI messages to an open MIDI output device (e.g an external MIDI synth) via MIDI connections.

Global parameters

Parameter name	Type	Description
none		

Track parameters

Parameter name	Type	Description
none		

MIDI CC

The midicc machine sends MIDI Control Change (CC) messages.

Global parameters

Parameter name	Type	Description
Smooth	switch	Smooth changes
Auto Learn	switch	Auto learn controllers

Track parameters

Parameter name	Type	Description
Channel	byte	MIDI Channel to send CC to
CC	byte	Control Change (CC)
Value	byte	Control Change (CC) value

MIDI Time

The miditime machine sends MIDI Sync messages from Buzé to MIDI instruments.

Stream Plugins

The stream plugins support a wide range of wave formats thanks to libmad and libsndfile.

These plugins share the same parameters, in order to let hosts implement previewing features.

The stream plugins synchronize the currently playing stream to the song playback position during seeks and jumps.

To select a file for playback, go to the plugins properties, and use the file selector from the plugins "Stream Source" property. Note that the wavetable stream plugin takes a specially formed filename, and is currently not possible to set up from the user interface.

Types of stream plugins

- WAV/AIFF/etc via libsndfile
- MP3 via libmad
- Streaming from the internal wavetable

Global parameters

Parameter name	Type	Description
Offset Low/High	word+word	Trigger offset
Length Low/High	word+word	Number of samples to play

Track parameters

Parameter name	Type	Description
Channel	byte	MIDI Channel to send CC to
CC	byte	Control Change (CC)
Value	byte	Control Change (CC) value

Value Generators

The Value Generator filters, transforms and sends values to other plugins' parameters.

Value generators can be used to control multiple plugin parameters at once, and/or connected in chains to perform complex operations on values before they reach their targets.

The value generator is also capable of generating random numbers using the Mersenne Twister algorithm using a specified seed.

Global parameters

Parameter name	Type	Description
Value	word	Input value
Seed	word	Random number generator seed. Used when operator = random. Setting the seed resets the random number generator.
Threshold	byte	Skips n values for every outputted value.
Allow min	word	Min value in allowed range. Tells the plugin to ignore values below this value.
Allow max	word	Max value in allowed range. Tells the plugin to ignore values above this value.

Track parameters

Parameter name	Type	Description
Operator	byte	0=add, 1=sub, 2=mul, 3=div, 4=mod, 5=neg, 6=random, 7=scale, 8=min, 9=max
Operator value	word	Value for operation. Ignored when operator is neg or random.

The track operations are applied in order. Division by zero becomes zero. The random operator destroys the input value.

Note Generator

The Note Generator transforms and sends notes to other plugins. Supports both note connections and MIDI connections.

Global parameters

Parameter name	Type	Description
Global Octave	byte	-5..5
Global Note	byte	-6..6
Harmonic Quantize	byte	0=custom, 1=major, 2=minor
C / C# / D / D# / E / F / F# / G / G# / A / A# / B	byte	Transpose single notes

Track parameters

Parameter name	Type	Description
Note	note	Note to process
Wave	byte	Wave index. Unused in MIDI output
Volume	byte	Volume

LFO Value Generators

The LFO Value Generator emits LFO values to other plugins' parameters at specified intervals.

Global parameters

Parameter name	Type	Description
Interval type	byte	The type of interval. 0=ticks, 1=ticks/256, 2=sec/16
Interval length	byte	The interval length in units set by the interval type.
Frequency	byte	Valid range ~0-8hz
Amplitude	word	Output value scale
Minimum	word	Output value minimum/center
LFO Type	byte	0=sine, 1=square, 2=saw, 3=triangle, 4=random
Seed	word	Seeds the random number generator when the LFO type is random.

ADSR Value Generators

The ADSR Value Generator emits ADSR values to other plugins' parameters at specified intervals.

Global parameters

Parameter name	Type	Description
Interval type	byte	The type of interval. 0=ticks, 1=ticks/256, 2=sec/16
Interval length	byte	The interval length in units set by the interval type.
Attack	word	0-1 sec
Decay	word	0-1 sec
Sustain	word	0-100%
Release	word	0-1 sec
Trigger	switch	1=start, 0=release

Signal Value Generators

The Signal Value Generator takes audio input and re-emits the signal as values to other plugins' parameters at specified intervals.

Global parameters

Parameter name	Type	Description
Interval type	byte	The type of interval. 0=ticks, 1=ticks/256, 2=sec/16
Mode	byte	0=envelope, 1=immediate, 2=absolute

Sequence

The Sequence plugin maps the global pattern player primitive, which plays patterns in the order list and controls the global tempo.

Global parameters

Parameter name	Type	Description
BPM	word	Beats per minute
TPB	byte	Ticks per beat

Track parameters

Parameter name	Type	Description
none		

The Sequence plugin is a singleton, which means there can be only one in a project, and maps directly to the internal "root pattern player". Its parameters control the global tempo.

The global tempo is decides the tempo of patterns in the order list.

Pattern Player

The pattern player plugin plays patterns in desired BPM/TPB.

Global parameters

Parameter name	Type	Description
BPM	word	Beats per minute, 0 = use timesource
TPB	byte	Ticks per beat, 0 = use timesource

Track parameters

Parameter name	Type	Description
Pattern Trigger	word	Pattern ID to play
Transpose	note	Transpose pattern notes relative to C-4

A Pattern Player plugin has global parameters for tempo and track parameters for pattern to play and transpose. By adding more tracks, it can play more patterns.

By default, a pattern player inherits its tempo from the sequence plugin, but it offers parameters for setting BPM and TPB independently of the global tempo.

To simultaneously play patterns in a different tempo, add another Pattern Player plugin to the project.

// Volume Column commands #define VOLCMD_NONE 0

#define VOLCMD_VOLUME 1

#define VOLCMD_PANNING 2

#define VOLCMD_VOLSLIDEUP 3

#define VOLCMD_VOLSLIDEDOWN 4

#define VOLCMD_FINEVOLUP 5

#define VOLCMD_FINEVOLDOWN 6

#define VOLCMD_VIBRATOSPEED 7

#define VOLCMD_VIBRATODEPTH 8

#define VOLCMD_PANSLIDELEFT 9

#define VOLCMD_PANSLIDERIGHT 10

#define VOLCMD_TONEPORTAMENTO 11

#define VOLCMD_PORTAUP 12

#define VOLCMD_PORTADOWN 13

#define VOLCMD_VELOCITY 14 //rewbs.velocity #define VOLCMD_OFFSET 15 //rewbs.volOff #define MAX_VOLCMDS 16

// Effect column commands #define CMD_NONE 0

#define CMD_ARPEGGIO 1

#define CMD_PORTAMENTOUP 2

#define CMD_PORTAMENTODOWN 3

#define CMD_TONEPORTAMENTO 4

#define CMD_VIBRATO 5

#define CMD_TONEPORTAVOL 6

#define CMD_VIBRATOVOL 7

#define CMD_TREMOLO 8

#define CMD_PANNING8 9

#define CMD_OFFSET 10

#define CMD_VOLUMESLIDE 11

#define CMD_POSITIONJUMP 12

#define CMD_VOLUME 13

#define CMD_PATTERNBREAK 14

#define CMD_RETRIG 15

#define CMD_SPEED 16

#define CMD_TEMPO 17

#define CMD_TREMOR 18

#define CMD_MODCMDEX 19

#define CMD_S3MCMDEX 20

#define CMD_CHANNELVOLUME 21

#define CMD_CHANNELVOLSLIDE 22

#define CMD_GLOBALVOLUME 23

#define CMD_GLOBALVOLSLIDE 24

#define CMD_KEYOFF 25

#define CMD_FINEVIBRATO 26

#define CMD_PANBRELLO 27

#define CMD_XFINEPORTAUPDOWN 28

#define CMD_PANNINGSLIDE 29

#define CMD_SETENVPOSITION 30

#define CMD_MIDI 31

#define CMD_SMOOTHMIDI 32 //rewbs.smoothVST

#define CMD_VELOCITY 33 //rewbs.velocity #define CMD_XPARAM 34 // -> CODE#0010 -> DESC="add extended

parameter mechanism to pattern effects" -! NEW_FEATURE#0010

#define CMD_NOTESLIDEUP 35 // IMF Gxy #define CMD_NOTESLIDEDOWN 36 // IMF Hxy #define MAX_EFFECTS 37

Buzz Wrapper

Refer to http://www.buzzmachines.com/ for more information and plugins download.

General

All Buzz plugins can receive MIDI notes and control changes via MIDI connections.

The wrapper fully supports machines that use hacks. Hacks are enabled per-machine in buzz2zzub.ini and should be updated regularily, either manually or by downloading a newer version from the Buze website.

The wrapper has been updated with partial support of some "new Buzz" features which have been added to the original Buzz since 2008.

If buzz.exe and its DLL dependencies are present in the Buze program directory, the wrapper will also attempt to load built-in machines directly from the executable binary. This allows for using Jeskola Qsamo on Windows XP (and Wine?) without .NET 4.0. However, it is also an experimental and unsupported feature, and may break at any time, as it makes certain assumptions about the code layout and linker switches used when compiling the Buzz executable.

Installing new plugins

Copy new plugin DLLs to Gear/Generators or Gear/Effects. New plugins will be detected on program startup.

VST(i) Wrapper

Wrapped VST-plugins does not have note columns. In order to play notes on a VST-plugin, create plugin capable of sending MIDI notes, such as a Note Generator or MidiTracker and connect it to the VST using a MIDI-connection.

Installing new plugins

Copy new plugin DLLs to Gear/VST. Optionally change the VST Path in Preferences to point at one or more directories containing VST plugins. New plugins will be detected on program startup.

Lunar Wrapper

The Lunar plugin format was developed specifically for Buzé audio engine Armstrong.

Installing new plugins

Copy new plugin directories to Gear/Lunar/fx.

Psycle Wrapper

Refer to http://psycle.pastnotecut.org/ for more information and plugins download.

Installing new plugins

Copy new plugin DLLs to Gear/Psycle. New plugins will be detected on program startup.

LADSPA Wrapper

Audacity has a set of 90 LADSPA-plugins available on their download page. Refer to http://audacity.sourceforge.net/download/plugins for more information and plugins download.

Installing new plugins

Copy new plugin DLLs to Gear/LADSPA. New plugins will be detected on program startup.

VAMP Wrapper

VAMP plugins are used for offline wave analysis, and are used to generate slices based on their output.

Refer to http://www.vamp-plugins.org/ for more information and plugins download.

Installing new plugins

Copy new plugins to %PROGRAMFILES%/VAMP Plugins. New plugins will be detected on program startup.

Writing plugins

Our general attitude is: find a compatible plugin wrapper SDK that offers the capabilities you need and go with it.

No one plugin format offers every capability of Armstrong, except in the very rare cases a plugin belongs in the core plugin set.

So as to picking a plugin format, there are the following things to consider:

VST

- Custom GUI allowed
- Multi-IO (VST 2.x, not 3)
- Receive MIDI
- Allows saving custom data in presets
- Parameters support byte values in the range 0..127

Buzz

- Custom GUI allowed
- Multi-IO (Newbuzz)
- Custom pattern editors (newbuzz)
- Does input mixing
- Receive MIDI

Psycle and LADSPA

- Cross platform - no GUI
- Stereo only
- No MIDI capabilities

Lunar

- Cross platform
- Multi-IO
- Interval, value and note controllers (native peer)
- Send/receive MIDI
- Supports OpenGL for GUI
- Minimal environment and libraries support

Native plugins

- Native plugins support everything supported by Armstrong and the host OS
- Should not depend on any specific GUI toolkits, beyond supporting wrapped plugin GUIs
- Compiled directly into the Armstrong shared library, no external plugin DLLs supported
- Should be essential and/or versatile - getting included in the core should be hard

Buze API browser

Please choose a class from the panel on the left side.

Tools used in the build process

The build process uses several tools to generate source code. This document deals with the internal tools developed specifically for building the Buzé/Armstrong projects (but which can be and are used in other projects as well).

Zidl tool - Zzub Interface Description Language

Zidl reads a public interface file as input, and produces source code in various programming languages as output.

Interface files are text files in zidl language, which supports enums, namespaces, structs, classes, methods and callbacks used to interoperate with a library.

The output source file can be a C header, Lua and Python bindings, XML documentation and more.

History

Zidl was originally developed by Paniq as a means to use the zzub C++ library from a Python application. The original zidl was written in Python. Later, zidl was ported to C++ by clvn using the re2c lexer and lemon parser generator. The new version later added support for more features and languages. zzub was also renamed to Armstrong some time along the way, but the z was stuck

Using zidl

Zidl uses C as the "lowest common denominator" for cross-language interopability, as most scripting languages can use libraries in C. Zidl can be used in C/C++ projects to expose desired functionality to scripting engines. It is likely also usable from other languages capable of producing shared libraries with C calling conventions (f.ex Delphi).

A library typically generates its public interface header using zidl, and implements the interface methods using C calling conventions.

Given the zidl file and the installed shared library, users can then generate Lua and Python bindings which call into the final library binary exposing the C interface.

Download and Source Code

Get the source code and cross platform project files from the following Subversion repository:

```
svn://anders-e.com/zidl/trunk/zidl
```

Latest binaries for Windows exist in the win32deps directory of the Buzé repository.

Documentgen tool - C++ code generator for SQLite

documentgen reads a database description from a JSON file and produces C++ code to assist working with a SQLite database.

The output C++ code contains:

- Datastructures using boost::mpl to enable traversing database tables, fields, types and relations at compile-time
- Strings with CREATE DATABASE and CREATE TABLE queries
- Triggers and callbacks for before/after INSERT/UPDATE/DELETE operations
- Code supporting undo/redo

History

The documentgen tool was developed to automate and simplify the process of making changes to the file format used by Buzé. Buzé uses SQLite internally to contain song data, and also as a file format when saving projects to disk. The tool has been generalized to be used in any suitable SQLite/C++ project.

Download and Source Code

Get the source code and cross platform project files from the following Subversion repository:

```
svn://anders-e.com/dbgenpp/trunk/dbgenpp
```

Latest binaries for Windows exist in the win32deps directory of the Buzé repository.

Introduction

Buzé is a full featured, portable modular music tracker that works on Windows and Linux (using Wine). People using it vary from studio bedroom producers to sound designers.

Buzé is a multi-plugin DAW, with support for many file formats. A notable distinction of the Buze is its ableton-like envelope. You also have a number of workflow options including piano rolls, you can design custom pattern layouts, you have dynamic midi and audio routing and you can combine different pattern-resolutions. Like a standard DAW, you can use Buze for wave audio editing, midi controlling and more! You can utilize the multifaaceted Buze as a sampler, software synth, audio editor or sequencer.

Buzé sports a no non-sense interface. Its easy to learn and will help you increase your musical productivity. The best thing about Buze is that its free, and it supports a gazillion of free downloadable instrument/effects. All you have to do is download it, run it, get machines, get samples, and enjoy your free portable software studio.

Features

- Portable studio, no installer-software
- 100% pattern-based sequencer with envelopes and pianorolls
- Very productive since almost everything can be done using shortcuts
- All plugins* can be switched between GUI or native Buzé-look, so you will not be distracted by massive GUI plugins
- Supports a wide range of plugins for generating and processing audio, MIDI, value and note signals. 30+ built-in plugins
- Full undo/redo, configurable keyboard shortcuts and window layouts
- Scripting engine, the whole audio/midi-engine + plugins can be customized and extended by your own scripts

Natively supported file formats

- Supports VST, Buzz, Psycle, LADSPA and Lunar plugins natively
- Imports ARMZ, BMX, IT, S3M, MOD, MID, SID song file formats
- Wavetable can import WAV, MP3, FLAC, IFF, Drumkit and many more file formats. Features a basic wave editor.

Buzé user documentation

Topics for the user documentation:

Other

- Practical Usage Tips
- Frequently Asked Questions
- Buze Cheatsheets
- ld mixer checklist

Buzé developer documentation

General

- Armstrong notes
- Creating plugins for Armstrong
- Armstrong API browser
- Buzé API browser
- Building the source code
- Tools used in the build process

Plugins

The built-in plugins are:

- Master
- Audio Output and Audio Output16
- Audio Input
- File Recorder (.WAV output)
- Wavetable Recorder
- MIDI Tracker
- MIDI Input
- MIDI Output
- MIDI Control Change
- MIDI Time
- Streaming plugins (.WAV, .MP3 input)
- Value Generator
- Note Generator
- LFO Value Generator
- ADSR Value Generator
- Signal Value Generator
- Sequence plugin
- Pattern Player plugin
- Modplug plugin

The built-in wrapper plugins are:

- VST Adapter
- Buzz Adapter
- Psycle Adapter
- LADSPA Adapter
- VAMP Adapter
- MIDI Fx (MFX) Adapter
- MIDI Hardware Adapter

ld mixer 1.03 checklist

First of all make sure you have latest version from ld's site:
http://koponen.home.xs4all.nl/beta/ld_mixer_v1_21.zip and this:
http://koponen.home.xs4all.nl/beta/ld_mixer_extra_files.zip.
If you replace any existing files, take backups first.

Won't appear or any menus without these files:

- Gear\Effects\ld mixer.dll
- auxbus.dll
- MSVCP60.dll
- MSVCRT.dll (already exists on Windows XP)

Crashes as soon as you connect anything to it without these files:

- overloader extbuzz.dll
- Gear\Machines.dll

Crashes when you connect anything to it when running multiple sessions of Buzé or Buzz from different folders at the same time.

Crashes when you connect anything to it when running from a directory where the full path has a space in it, e.g C:\Program Files\Buze. (fixed in Buzé 0.4.8!)

Skin doesn't work without Gear\Effects\ld mixer.gfx.

Theme Parameters

Below are all the understood theme parameters and their descriptions.

Buzé View	Parameter Name	Description
Machine View	MV Amp BG	
	MV Amp Handle	
	MV Arrow Volume High	
	MV Arrow	
	MV Arrow Volume Low	
	MV Background	
	MV Effect	
	MV Effect LED Off	
	MV Effect LED On	
	MV Effect Mute	
	MV Effect Pan BG	
	MV Generator	
	MV Generator LED Off	
	MV Generator LED On	
	MV Generator Mute	
	MV Generator Pan BG	
	MV Machine Border	
	MV Machine Text	
	MV Master	
	MV Master LED Off	
	MV Master LED On	
	MV Pan Handle	
	MV Line	
	MV MIDI Line	
	MV Event Line	

Pattern Editor	PE BG	
	PE BG Dark	
	PE BG Very Dark	
	PE Sel BG	
	PE Text Value	
	PE Text Note	
	PE Text Note Off	
	PE Text Wave	
	PE Text Trigger	
	PE Trigger Highlight	
	PE Trigger	
	PE Trigger Shadow	
	PE Text Shade	
	PE Text Rows	
	PE Text Headers	
	PE Loop Points	
	PE Loop Points Disabled	
	PE Playback Pos	
	PE Divider	
	PE Hidden	
	PE Control	
	PE Selection	
	PE Cursor	
Notes	PE Text Note 1	
	PE Text Note 2	
	PE Text Note 3	
	PE Text Note 4	
	PE Text Note 5	
	PE Text Note 6	

	PE Text Note 7	
	PE Text Note 8	
	PE Text Note 9	
	PE Text Note 10	
	PE Text Note 11	
	PE Text Note 12	
Signal Analysis	SA Amp BG	
	SA Amp Line	
	SA Freq BG	
	SA Freq Line	

Windows and views

Read about the individual windows and views:

- Global
- Machine View
- Parameter View
- Pattern Editor
- Pattern Format View
- Order List
- Wave Table
- File Browser
- All Machines View
- Preferences

Basic concepts

Topics on basic concepts:

- Plugins
- Patterns
- The default project
- Using MIDI
- Using JACK for Windows
- Using VST plugins
- Using peer plugins
- Tempo, time signatures, shuffle/swing
- Customizing Buze

```
#include <zzub/zzub.h>
#include <iostream>

using namespace std;

int main() {
    zzub_player_t* player = zzub_player_create();
    zzub_audiodriver_t* driver = zzub_audiodriver_create();
    zzub_audiodriver_create_device(player, -1, -1);

    zzub_player_initialize(player);

    zzub_audiodriver_enable(driver, 1);

    zzub_player_load_armz(player, "test.armz");
    zzub_player_history_commit(player, 0, 0, "Loaded song");
    zzub_player_set_state(player, zzub_player_state_playing);
```

```
    cout << "Press ENTER to quit" << endl;

    cin.getline();


    zzub_player_set_state(player, zzub_player_state_stopped);

    zzub_player_destroy(player);


    zzub_audiodriver_enable(driver, 0);

    zzub_audiodriver_destroy(driver);


    return 0;

}
```

Flow of a Armstrong call that modifies the song

Many public C API methods operate on the storage component, using methods that primarly generate and execute SQL commands. Lets see what happens when we try to rename a plugin. This call flow is similar for most calls that modify the storage database.

1. A client (such as Buze) calls one of the public C methods, let us say zzub_plugin_set_name()

2. The C method modifies storage::plugin::name and calls storage::plugin::update(), which executes an SQL UPDATE statement on the plugin table
3. SQL triggers in the database generate and save undo SQL statements in the (temporary) history table
4. SQL triggers generate storage events by calling storage::document::notify_listeners()
5. storage::notify_listeners() calls player::update_document(), which is a registered storage event listener
6. player::update_document() parses the event and calls player::on_update_plugin()
7. player::on_update_plugin() calls mixer::update_plugin()
8. mixer::update_plugin() creates a copy of the updated metaplugin and adds it to a queue of objects to be swapped in the audio thread later
9. ... execution returns to the client, who can make more changes to the document, or commit the changes when done...

10. To commit changes, the client calls zzub_player_history_commit() which calls storage::document::barrier()
11. storage::document::barrier() creates an undo step, and generates a storage event by calling storage::document::notify_listeners()
12. storage::notify_listeners() calls player::update_document(), which is a registered storage event listener
13. player::update_document() parses the event and calls player::on_barrier()
14. player::on_barrier() calls mixer::commit()
15. mixer::commit() generates a user->audio event by calling mixer::invoke_audio_event()

16. ... execution returns to the client. At this point, the audio thread will begin handling the audio event:

17. In the audio thread, the mixer parses the event in mixer::process_audio_event_queue() and calls mixer::on_barrier()
18. mixer::on_barrier() swaps in the new metaplugin

When the client wants to undo, it calls zzub_player_undo(), which calls document::undo(). Simplified, this executes the undo SQL query saved in step 3, and then resumes at step 4.

Storage

Armstrong uses an SQLite database for storing song state and temporary files for wave data. All operations on a song are ultimately executed as SQL statements on the database. The storage library provides convenience methods and classes for most operations. Parts of the storage library is autogenerated by the documentgen-program.

Undo/Redo

The basic concept for undo/redo with an SQL database is described on the SQLite wiki.

Armstrong extends the technique in the article with support for multiple INSERT/UPDATE/DELETE per undo step, notification callbacks and the option to temporarily disable undo buffering.

The ability to temporarily disable undo buffering is important when the host wants to create and destroy plugins transparently. For example: during mixdown, a recorder plugin can be created and used to record to disk. Or, the analyzer view can create a recorder plugin for streaming output to the display. Or, for previewing samples from disk or the wavetable, a temporary stream plugin can be used. This kind of "jacking the undo buffer" can lead to a broken undo buffer, and leaves a lot of responsibility on the host developer.

The .armz file format

Armstrong saves to a new file format - .armz - which is a zipped archive containing the SQLite database file (song.armdb) and all waveforms (wavelevel_*.raw).

Song versioning

The storage version number is stored in the version field in the song table. Upon loading, the version field is checked, and if the version number is lower than the current, a series of upgrade scripts are executed. The upgrade scripts are kept as an array of hard coded SQL statements in document.cpp, and is maintained as the .armz database schema changes over time. This approach has limitations, but has worked out nicely so far.

SQL Extensions

Armstrong adds several helper-functions to the embedded SQLite engine for use in its internal SQL-queries.

Function name	**Description**
noteutil_buzz_to_midi_note	Converts a Buzz note to a linear MIDI note (because notes are stored as Buzz notes)
noteutil_midi_to_buzz_note	Converts a MIDI note to a Buzz note
undoredo_enabled_callback	Returns 1 if undo is enabled
wavelevel_insert_samples	Intended for internal use only. Raw sample data helper
wavelevel_replace_samples	Intended for internal use only. Raw sample data helper

wavelevel_delete_samples	Intended for internal use only. Raw sample data helper
wavelevel_delete_file	Intended for internal use only. Raw sample data helper
XXX_notify_callback	Intended for internal use only. Used in INSERT/UPDATE/DELETE-triggers. Invokes document::notify_listeners() with row id and an event id

Database schema

```
CREATE TABLE attribute (id integer primary key,
plugin_id integer, attrindex integer, value integer);
CREATE TABLE attributeinfo (id integer primary key,
plugininfo_id integer, attrindex integer, name
varchar(64), minvalue integer, maxvalue integer,
defaultvalue integer);
CREATE TABLE connection (id integer primary key,
from_plugin_id integer, to_plugin_id integer, type
integer);
CREATE TABLE envelope (id integer primary key,
wave_id integer, attack integer, decay integer,
sustain integer, release integer, subdivision
integer, flags integer, disabled integer);
CREATE TABLE envelopepoint (id integer primary key,
envelope_id integer, x integer, y integer, flags
integer);
CREATE TABLE eventconnectionbinding (id integer
primary key, connection_id integer, sourceindex
integer, targetparamgroup integer, targetparamtrack
integer, targetparamcolumn integer);
CREATE TABLE midiconnection (id integer primary key,
connection_id integer, mididevice varchar(512));
CREATE TABLE midimapping (id integer primary key,
```

```
plugin_id integer, paramgroup integer, paramtrack
integer, paramcolumn integer, midichannel integer,
midicontroller integer);
CREATE TABLE parameterinfo (id integer primary key,
plugininfo_id integer, paramgroup integer, paramtrack
integer, paramcolumn integer, name varchar(64),
description varchar(128), flags integer, type
integer, minvalue integer, maxvalue integer, novalue
integer, defaultvalue integer);
CREATE TABLE pattern (id integer primary key, song_id
integer, name varchar(64), length integer, resolution
integer, display_resolution integer,
display_verydark_row integer, display_dark_row
integer, patternformat_id integer);
CREATE TABLE patternevent (id integer primary key,
pattern_id integer, time integer, plugin_id integer,
paramgroup integer, paramtrack integer, paramcolumn
integer, value integer);
CREATE TABLE patternformat (id integer primary key,
song_id integer, name varchar(64));
CREATE TABLE patternformatcolumn (id integer primary
key, patternformat_id integer, plugin_id integer,
paramgroup integer, paramtrack integer, paramcolumn
integer);
CREATE TABLE plugin (id integer primary key, flags
integer, song_id integer, name varchar(64), data
blob, trackcount integer, x real, y real,
streamsource varchar(64), is_muted integer,
is_bypassed integer, is_solo integer, is_minimized
integer, plugininfo_id integer);
CREATE TABLE plugininfo (id integer primary key,
song_id integer, uri varchar(64), name varchar(64),
short_name varchar(64), author varchar(64), mintracks
integer, maxtracks integer);
CREATE TABLE pluginparameter (id integer primary key,
plugin_id integer, paramgroup integer, paramtrack
integer, paramcolumn integer, value integer);
CREATE TABLE sequence (id integer primary key,
plugin_id integer, pattern_id integer, position
integer, width integer);
```

```
CREATE TABLE song (id integer primary key, version
integer, title varchar(64), comment blob, songbegin
integer, songend integer, loopbegin integer, loopend
integer, loopenabled integer);
CREATE TABLE wave (id integer primary key, song_id
integer, name varchar(64), filename varchar(64),
flags integer, volume real);
CREATE TABLE wavelevel (id integer primary key,
wave_id integer, basenote integer, samplerate
integer, samplecount integer, beginloop integer,
endloop integer, format integer, filename
varchar(64));
```

Mixing

Multithread mixing

During mixing, Armstrong distributes the work load across a user-defined number of threads, executed by the operating system on any available CPUs. The number of worker threads must be one or more. When a single worker thread is specified, the mixer runs in "single-thread" mode, falling back to mixing on the audio thread.

Plugins in the graph are considered tasks, where connections define the dependencies. The dependencies are counted, and stored with each task.

The distributed mixer adds tasks on a lock free queue which is polled by the worker threads. Only tasks with a dependency count of zero are added to the queue. When a task is done processing, it decreases the dependency counter of all of its dependent tasks, allowing the mixer to schedule new tasks. The task counter and dependency counts are stored as atomic<int>s, ensuring lock free operation throughout the process.

Plugin processing order

During processing, Armstrong uses a non-recursive loop to traverse the plugins. Every time the graph changes (a plugin or connection was inserted or deleted), the process order is updated. The following steps determine the final processing order:

1. Create a graph with plugins as vertices and connections as edges in a boost::adjancency_list.
2. Run a depth_first_search to determine back edges. I.e which connections are used in feedback loops.
3. Remove the back edges from the graph.
4. Find roots in the graph. I.e plugins which do not send their output to any other plugins
5. Perform a topological sort for each root.
6. Results from each topological sort are prepended to the final work order, except the result containing the master; which is added at the end.

Message passing in the mixer

The mixer uses five ringbuffers for message passing between the threads.

- user_event_queue - for audio->user thread events
- audio_event_queue -> for immediate user->audio thread events
- commit_event_queue -> for delayed user->audio thread events
- encoder_user_event_queue -> for encoder->user thread events
- encoder_event_queue -> for audio->encoder thread events

Audio to user thread events

The following types of messages originate in the audio thread, and are forwarded to the user thread via mixer::user_event_queue:

- Parameter changes

- State changes (e.g at the end of a song when looping is disabled)
- MIDI control changes

User messages are polled by calling mixer::process_user_event_queue(). The equivalent C method is zzub_player_handle_events().

Immediate user to audio thread events

The following types of events originate in the user thread, and are passed to the audio thread as fast as possible via mixer::audio_event_queue:

- Start/stop state changes
- Play note
- Song position changes
- MIDI plugin changes
- Parameter changes
- Editing barrier
- Plugin process events

Delayed user to audio thread events

Delayed events are sent upon calling mixer::barrier(). A barrier indicates all the latest changes should be to updated to the running graph. The following events originate in the user thread, and are passed to the audio thread via mixer::commit_event_queue:

- Parameter changes
- Plugin process events
- Plugin state format changes

- Graph changes

Encoder to user thread events

Encoder plugins could generate user events, usually for passing audio and slices to the wavetable.

Audio to encoder thread events

For passing audio to encoders.

Tickless processing

The mixer knows little of tempo or ticks, and instead provides a mechanism where plugins decide when to process plugin events.

There are two modes for which a plugin can intercept processing, which is specified through a plugin flag:

zzub_plugin_flag_is_sequence	Effectively marks the plugin as a time source, which maintains its own tempo by associating with and using one or more pattern players.
zzub_plugin_flag_has_interval	Used by plugins that want to intercept the processing at fixed intervals. The engine calls plugin::process_sequence() to determine the number of samples to process before calling plugin::process_sequence() again.

Player

The player implements listener-interfaces for both the mixer and the storage and routes events internally.

Language bindings

The Armstrong API is described in a spesial interface description language called zidl (Zzub IDL). The zidl-tool supports generating language bindings for Python. It can also generate a C header file, a .def file for linking on Windows and HTML documentation.

The Zidl tool is currently undergoing a rewrite to accomodate for future requirements in a more satisfying manner.

Armstrong API browser

Please choose a class from the panel on the left side.

Build instructions

Required tools

- Visual Studio Express 2012 for Desktop with at least Update 1. Non-express and newer versions are assumed to work
- TortoiseSVN or your preferred Subversion client

VS2010 works, but only after editing the project files to use the correct platform toolset. Project files for VS2008 and older are no longer maintained.

Install Boost

Download and install precompiled 32 bit library binaries of latest Boost from the Boost download page. For VS2012, choose the msvc-11.0-32 build.

Armstrong requires Boost version 1.53 or newer! Armstrong depends on boost::lockfree and boost::atomic which were adopted in Boost version 1.53.

All other dependencies on Windows except Boost are kept in the repository and must be copied into the working directory as described below.

Download source codes and other dependencies from the repository

Using the command line - note the very important steps where win32deps is **exported** from trunk and manually **copied** into the working directory: C:\Users\clvn\Code> md buzesrc C:\Users\clvn\Code> cd buzesrc C:\Users\clvn\Code\buzesrc> svn co svn://anders-e.com/buze/trunk/buze buze ... C:\Users\clvn\Code\buzesrc> svn export svn://anders-e.com/buze/trunk/win32deps depfiles ... C:\Users\clvn\Code\buzesrc> xcopy depfiles buze /s ... Optionally checkout the docs, website and installer files:

```
C:\Users\clvn\Code\buzesrc> svn co svn://anders-
e.com/buze/trunk/docs docs
        ...
        C:\Users\clvn\Code\buzesrc> svn co
svn://anders-e.com/buze/trunk/installer installer
        ...
```

Configure Boost paths in Visual Studio

If Boost was installed in the step above, Visual Studio needs to know where to locate its headers and libraries.

First start Visual Studio and open buzesrc\buze\buze.sln. Go to the Property Manager and view the properties of "Microsoft.Cpp.Win32.user" under one of the project configurations.

In VS Express, this is in the menu under View -> Other Windows -> Property Manager. Also note that in VS Express the Property Manager is hidden by default. It can be enabled via Tools -> Setting -> Expert settings. (here and here) The following paths should be updated to match where Boost was installed. This example assumes Boost version 1.55 with prebuilt binaries for VS2012: **Include Directories:** c:\boost_1_55_0\

Library Directories:
c:\boost_1_55_0\lib32-msvc-11.0

Build with Visual Studio

With all the source code and dependencies in place, building should be a matter of opening buzesrc\buze\buze.sln in Visual Studio and choose Build -> Build Solution.

If you get build errors on the first attempt, first check under Build -> Configuration Manager and make sure the "Build" checkboxes are checked for all projects.

After successful compilation, buzesrc\buze contains new binaries for a core Buzé install with all standard GUI plugins.

MFX (Midi FX) Wrapper

Mfx plugins are used for processing and generating MIDI signals.

Refer to http://www.midiplugins.com/ for more information and plugins download.

Installing new plugins

Mfx plugins must be registered with regsvr32. Please refer to any instructions included with the individual plugins. New plugins will be detected on program startup.