A Yoshimi User Manual

Chris Ahlstrom
(ahlstromcj@gmail.com)

September 4, 2015

Contents

List of Figures

List of Tables

1 Introduction

1.1 Yoshimi Versus ZynAddSUbFX

This document describes how to use *Yoshimi* [16], the software synthesizer derived from the great *ZynAddSubFx* [18] software synthesizer. Because of their similarities, much of this document also applies to *ZynAddSubFx* and depends upon *ZynAddSubFX* documentation and diagrams.

Yoshimi is a MIDI software synthesizer for Linux. It synthesizes in real time, can run polyphonic or monophonic in multiple simultaneous patches in one or more MIDI channels, has broad microtonal capability, and much more. It includes extensive addititive, subtractive, and PAD synth capabilities which can be run simultaneously within the same patch. It also has comprehensive effects capabilities.

Originally based on the 2.4.0 version of *ZynAddSubFX* (Copyright 2002-2009 Nasca Octavian Paul), development of *Yoshimi* has continued for quite a while now in its own directions. These include major optimizations for audio and MIDI performance, and more recently progressive development of user-level access to all controls. At the same time refinement continues, both visually and within the code.

What are the advantages of *Yoshimi* versus *ZynAddSubFX*? Well, this information may be outdated by now, but *Yoshimi* is supposed to have better JACK support than does *ZynAddSubFX*. *Yoshimi* also has a cleaner and neater graphical user interface. On the other hand, *ZynAddSubFX* has a few features that *Yoshimi* does not, such as being able to easily record the output waveforms.

1.2 Document Structure

The structure of this document is a struggle. No matter which route is taken, there's no way to avoid jumping all over this document to adequately cover a topic. Therefore, the sections are basically provided in the order their contents appear in the user interface of *Yoshimi*. To help the reader jump around this document, multiple links and references are supplied.

Usage tips for each of the functions provided in *Yoshimi* are sprinkled throughout this document. Each tip occurs in a section beginning with "Tip:". Each tip is provided with an entry in the Index, under the main topic "tips".

Bug notes for some of the oddities found in *Yoshimi* are sprinkled throughout this document. Each bug occurs in a sentence beginning with "Bug:". Each bug is provided with an entry in the Index, under the main topic "bugs".

TODO items are also present, in the same vein. This document currently has a lot of them!

1.3 Let's Get Started!

Let us run *Yoshimi*, but run it without using *JACK*, which complicates the discussion of *Yoshimi*. The first thing to do is make sure one has no other sound application running (unless one wants to risk blocking *Yoshimi* or hearing two sounds simultaneously, depending on one's sound card and ALSA setup). Then start *Yoshimi* so that it uses ALSA for audio and ALSA for MIDI. Also use the "&" character so that we get back to the command-line prompt. See section 14 ("Yoshimi Man Page") on page 164.

```
$ yoshimi -a -A &
```

Figure 1: Yoshimi Splash Screen!

One sees a brief message, and then the splash screen. We show the splash screen, figure 1, here because it goes away too fast when one runs *Yoshimi*! What fun is that?

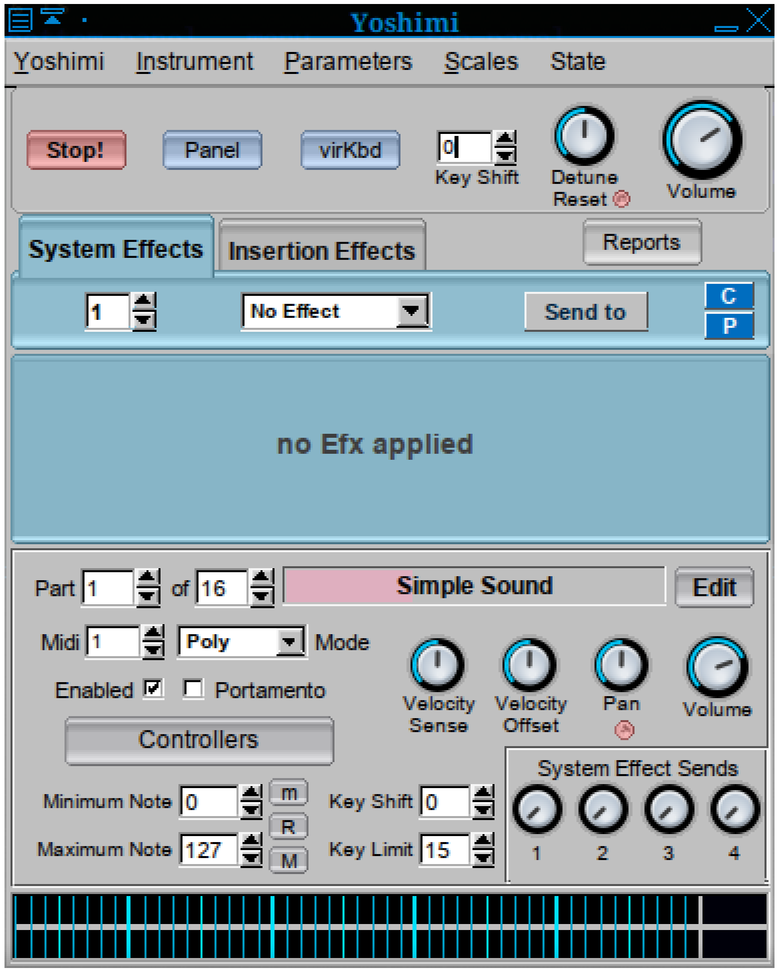

Figure 2: Yoshimi Main Screen, 1.3.5

Then the *Yoshimi* main window appears, as shown in figure 2 ("Yoshimi Main Screen, 1.3.5") on page 11.

For this manual, we describe the main window as being composed of the following sections:

1. **Menu**.
2. **Top Panel**.
3. **Effects Panel**.
4. **Bottom Panel**. Includes the VU-meter at the bottom.

There's a lot going on with *Yoshimi*, and there's no way to describe it in linear order. This manual will describe how to do useful things in each of the sections noted above, while leaving some of the details to be described in later sections, to which reference can be made for the details. This document depends heavily on index entries and references.

2 Concepts

Before we start with the user-interface, let's cover some concepts. *Yoshimi* requires the user to understand many concepts. Understanding these concepts makes it easier to configure *Yoshimi* and to drive it from a sequencer application.

Significant portions of this section are shamelessly copied (and tweaked) from Paul Nasca's original *ZynAddSubFX* manual [23] or [24]. One can discern such sections by the usage of the term *ZynAddSubFX* instead of *Yoshimi*. However, even the *Yoshimi* developers sometimes refer to *ZynAddSubFX* or *Zyn*.

Note that there are some audio/electrical concepts discussed in greater detail in section 4 ("Stock Settings Elements") on page 54. Maybe they belong in this "concepts" section, but they are directly tied to user-interface items.

2.1 Concepts / Terms

This section doesn't provide comprehensive coverage of terms. It covers mainly terms that puzzled the author at first.

2.1.1 Concepts / Terms / cent

The **cent** is a logarithmic unit of measure used for musical intervals. Twelve-tone equal temperament divides the octave into 12 semitones of 100 cents each. Typically, cents are used to measure extremely small finite intervals, or to compare the sizes of comparable intervals in different tuning systems. The interval of one cent is much too small to be heard between successive notes.

2.1.2 Concepts / Terms / instrument

In *Yoshimi*, an *instrument* is a complex sound that can be constructed using ADDsynth, SUBsynth, PADsynth, and kits. Each instrument is loaded into a *part* (see section 2.1.3).

In our documentation, we will sometimes use the terms "instrument", "part", "patch", and even "program" interchangeably and loosely.

2.1.3 Concepts / Terms / part

In *Yoshimi*, a *part* is one of 16 slots into which one can load an instrument (see section 2.1.2). Each part can be enabled or disabled, and assigned to a particular MIDI channel.

2.1.4 Concepts / Terms / patch

In MIDI jargon, a *patch* is a one of 16 channels in a MIDI device. Many synthesizers can handle several waveforms per patch, mixing different instruments together to create synthetic sounds. Each waveform counts as a MIDI voice. Some sound cards can support two or more waveforms per patch. *Yoshimi* has some ability to combine waveforms ("voices") into one instrument (section 2.1.2), which can then be loaded into a *Yoshimi* part (section 2.1.3).

Before General MIDI, which standardized patches, MIDI vendors assigned patch numbers to their synthesizer products in an arbitrary manner.

2.1.5 Concepts / Terms / preset

In MIDI jargon, a *preset* is an instrument that can be easily loaded. It is also called a *program* (2.1.6). A program is selection via a "program-change" message.

In *Yoshimi*, a *preset* is any collection of settings that can be saved to the clipboard or to a file, for later loading elsewhere.

2.1.6 Concepts / Terms / program

In MIDI jargon, a *program* is the same as a *preset* (**??**).

2.1.7 Concepts / Terms / voice

In MIDI jargon, a *voice* is the same as a *preset* or a a *program*.

In *Yoshimi*, a *voice* is a single configurable waveform that is just one of up to eight waveforms in an ADDsynth setting. Such voices can also be used as modulators for other voices.

2.2 Concepts / ALSA Versus JACK

Some discusson from the *Yoshimi* wiki. Here for eventual clarification.

A bit of a question mark was raised over ALSA MIDI support. A lot of people seem to be giving this up and relying on bridges like *a2jmidi* for legacy software and hardware inputs. JACK MIDI is already synchronous so should be jitter-free whereas ALSA MIDI runs on a 'best effort' basis. Added to which JACK is available for OS X and Windows so concentrating on this could make a possible port to other platforms more attractive – not to me I (Will J. Godfrey) hasten to add!

Seq24 (a nice, if old, sequencer) uses ALSA MIDI. To connect applications that exclusively support JACK MIDI, *a2jmidid* will do the translation. (Jack v. 1 has this integrated in recent versions, apparently).

2.3 Concepts / Sessions

As with most applications, *Yoshimi* and *ZynAddSubFX* allow for one to save one's work and reload it. Here are the file extensions used for saving the data:

- `.xmz` A Session. Everything. It's format is either XML or compressed XML, as explained below. See section 3.1.4.1 ("Menu / Yoshimi / Settings / Main Settings") on page 25.
- `.config` Sometimes one will see the extension `.config` used in the `$HOME/.config/yoshimi` directory. These files contain information to translate between bank directory names and bank ID values.
- `.state` Sometimes one will see the extension `.state` used in the `$HOME/.config/yoshimi` directory. These files contain a lot more information, that needed to duplicate a *Yoshimi* session that was saved. Might be the same as an `.xmz` file.
- `.xiz` An Instrument. These files can have two formats, compressed and uncompressed.
- `.xsz` Scale Settings. These files stored microtonal settings that *Yoshimi* can use to produce non-standard musical scales.

- `.xpz` Presets. A preset is canned version of a *Yoshimi* sub-setting. Presets can be copied and pasted using the **C** and **P** user-interface dialogs associated with many of the *Yoshimi* dialog windows. They make it easy to save portions of the current settings for later use. For example, resonance settings can be saved.

The Unix `file` command indicates that these files are one of two types:

- *exported SGML document, ASCII text*. These files are unindented XML data with an encoding of UTF-8 and a DOCTYPE of "ZynAddSubFX-data".
- *gzip compressed data, from Unix*. These files can be renamed to end in ".gz", and then run through the `gunzip` program to yield the XML file (but without an `.xml` extension).

The format probably depends on the "XML compression level" option discussed in section 3.1.4.1 ("Menu / Yoshimi / Settings / Main Settings") on page 25.

At some point in the future we may add a discussion of the contents of these files. In general, the contents are structured a lot like the user-interface elements that are used to set them.

2.3.1 Concepts / Sessions / All

One of the simplest ways to save one's work is to save the entire session. This can be done through the **Yoshimi** menu (the **File** menu in *ZynAddSubFX*) and will result in the creation of an `.xmz` file. Once created, this file will hold the settings for all settings within that session, such as microtonal tunings, all patches, system effects, insertion effects, etc.

2.3.2 Concepts / Sessions / Parts

In *Yoshimi*, a *part* is one of 16 slots into which one can load an instrument (a patch or program).

In many cases saving everything is not what is desired. Saving a patch later on in an editing session is one such example. In order to save a patch, one can either save it from the **Instruments** menu in section 3.2.3 ("Menu / Instrument / Save Instrument..") on page 37, or through the **Bank** window in section 3.2.5 ("Menu / Instrument / Show Banks..") on page 43.

With the **Instrument** menu, one can just save the file to any given location with the `.xiz` extension.

With the **Banks** menu, one can assign a patch to a given slot with a bank. This instrument will remain in that slot for future use until it is deleted. To see the physical location of the `.xiz` file, one should check the "Yoshimi / Settings / Banks / Root Dirs" (*FileSettingsBank_Root_Dirs*) window to see the paths for banks.

Note that one needs to have write permissions to add instruments to the bank. Banks are more thoroughly described in section 2.4 ("Concepts / Banks and Roots") on page 14.

2.4 Concepts / Banks and Roots

An important concept in *Yoshimi* is *banks*. Instruments can be stored in banks. These are loaded and saved automatically by the program. On program start, the last used bank is loaded. A single bank can store up to 128 instruments.

The file `Banks.txt` in the *Yoshimi* source-code bundle makes an important point about a transition (in newer versions) to tagging roots (directories) and banks with an ID code:

> One no longer has the concept of a default root directory, but a current one. This can by changed at any time without requiring a re-start, so there is now no longer a need to display the (confusing) contents of all roots. Also, roots now have ID numbers associated with them, but no changes have been made to the actual directores to achieve this. Instead the IDs are stored in the config file. The same ID system is used for banks, again without making any file system changes.
>
> At first run (and whenever new root directories are set) unknown roots and banks are given these IDs. Once set they will not change no matter how many more roots and banks are later added. One can, however, manually change root directory IDs in 'settings'. Also, there is a Banks window so that these can be set up, moved and renamed in exactly the same way as instruments can. With these IDs, roots and banks can be grouped/ordered by function instead of alphabetically. When using the GUI one will always know exactly which root and bank one fetches an instrument from.
>
> The significance of all this is that one's MIDI sequencer can now reliably use these ID numbers to select roots, banks and (already available) instruments. That Rosegarden or Muse file one saves today will be just as valid in the future, unless one makes the deliberate choice to change some IDs. Indeed, one can now start with an 'empty' Yoshimi, and via MIDI, set roots, banks and load instruments into parts (enabling the parts as one does so) swapping banks and roots as necessary. While the MIDI file runs it can silently pull instruments from any root/bank into any non-sounding part without disturbing the playing ones.
>
> In "Yoshimi / Settings / CC" one can enable or disable all these MIDI features, and can define which CCs one wants to use. Bank can be either MSB or LSB (as before). Root can be any non-reserved CC but including the one not in use for Bank. Also, Extended Program Change now has the same restrictions as Root, and these three are all cross-checked against each other. As an example, one might set Bank to LSB and Root to 0 (MSB), effectively giving one extended bank control compatible with all sequencers.
>
> Also, different instances have their own config files so that one can have (say) the main instance with current root(9), bank(23) while instance 4 has current root(2), bank(6). One can call up instances by number and thus access saved settings for that instance. As each instance has its own MIDI and audio ports, they can behave more-or-less independently.
>
> In doing all of this we have completely changed the way we manage the structure internally, resulting in much greater efficiency, at the cost of only a slightly slower startup. Swapping roots performs *no* file operations. Swapping banks only fetches the directory list of the newly selected bank. Changing an instrument doesn't have to search for a file, only load from its already known location.

"CC" is a "continuous controller". A MIDI bank change is usually a CC#0 value of 0, followed by a CC#32 value of X, where X is the desired bank number from 0 to whatever. (However, in some cases it may simply be a CC#0 on its own with a value of X). Many synths also require that you send a program change after the bank change, to select the program within the bank.

Some minor instructions are provided in section 3.2.6 ("Menu / Instrument / Show Root Paths..") on page 45.

Also note that, as well as bank and program changes, there is the ability to set a MIDI CC to access the voices from 129 to 160 (numbered re 1). All the Bank controls are contained in a tab in the main **Settings** window, and take immediate effect.

Bank root directories are identified with IDs that can be changed by the user in the user-interface. This feature is also made available for selection over MIDI. MIDI only sees banks in the *current* root directory, but all banks are accessible to the user-interface.

2.4.1 Concepts / Sessions / Presets

Have a favorite setting for an envelope, or a difficult-to-reproduce oscillator? Then presets are for you! Presets allow for one to save the settings for any of the components which support copy/paste operations. This is done with preset files (`.xpz`), which get stored in the folders indicated by *FileSettingsPreset_Root_Dirs*.

2.5 Concepts / Basic Synthesis

This section describes some of the basic principles of synthesis, and contains suggestions on how to make instruments that sound like they have been made with professional equipment. This applies to *Yoshimi* or to any synthesizer (even if one wrote it oneself with a few lines of code). All the ideas from *Yoshimi* are derived from from the principles outlined below.

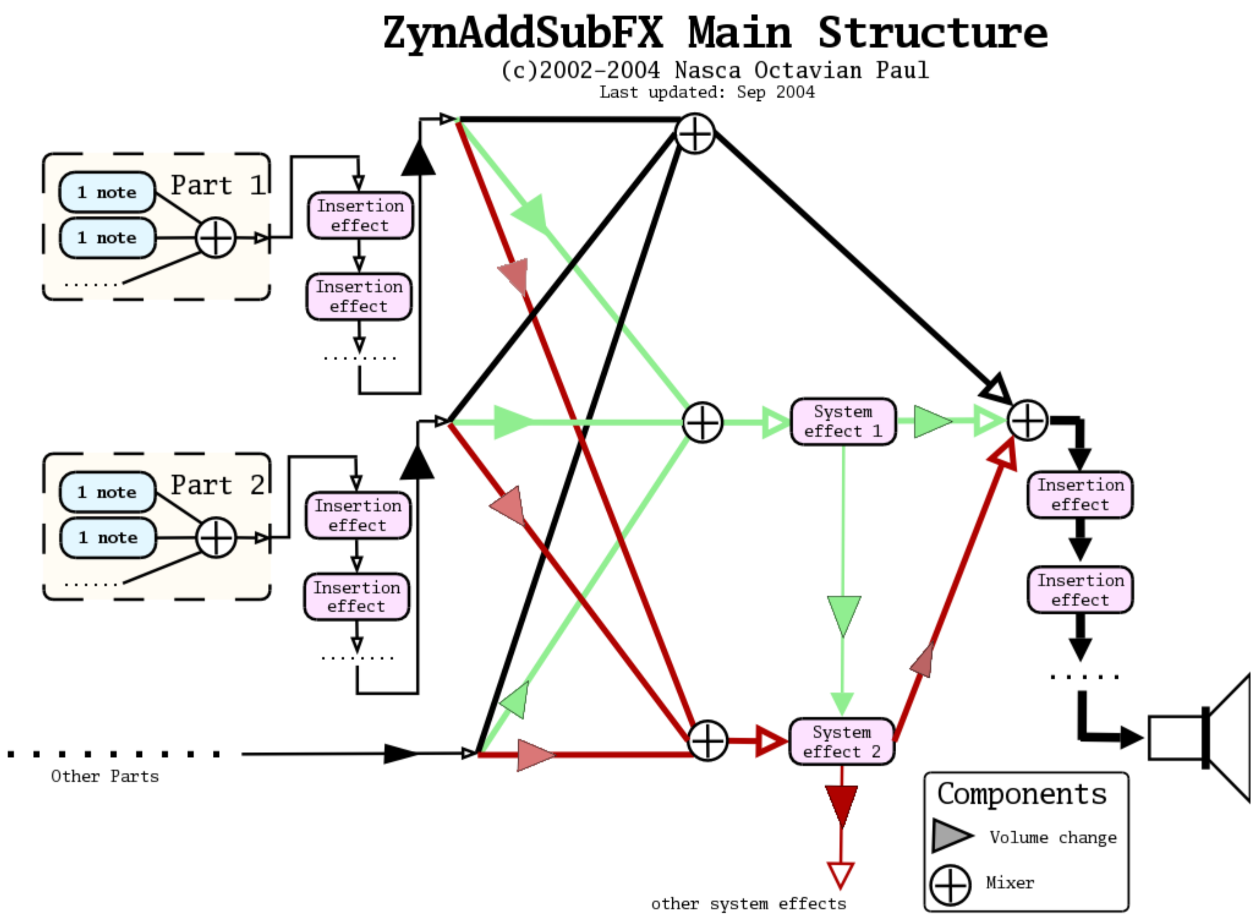

Figure 3: ZynAddSubFX/Yoshimi Main Structure

For a given part, the synthesizer first creates a note. Each notes waveform (for example, in a chord) is summed (mixed). This complex waveform is then send to the series of Insertion effects (if any) that are defined. Each part is then sent to a System effect and (depending on the wetness of the mix) directly to a mixer. Additional Insertion effects (if any) are then applied. The result is the final output of the synthesizer.

The synthesizer has three major types of parameters:

1. **Master settings/parameters**. Contains all parameters (including effects and instruments).
2. **Instrument parameters**. Contains ADDnote/SUBnote/PADnote parameters for a part.
3. **Scale settings**. Contains the settings of scales (*Yoshimi* is a micro-tonal synth) and few other parameters related to tunings.

2.5.1 Concepts / Basic Synthesis / Panning

Pan lets one apply panning, which means that the sound source can move to the right or left. Set it to 0.0 to only hear output on the right side, or to the maximum value to only hear output on the left side.

2.5.2 Concepts / Basic Synthesis / Wetness

Wetness determines the mix of the results of the effect and its input. This mix is made the effects output. If an effect is wet, it means that none of the input signal is bypassing the effect. If it is dry, then the effect is bypassed completely, and has no effect.

2.5.3 Concepts / Basic Synthesis / Single Note

The idea of this synthesis model is from another synthesizer Paul Nasca wrote years ago, released on the Internet as "Paul's Sound Designer". The new model is more advanced than that project (adding SUBsynth, more LFO's/Envelopes, etc.), but the idea is the same.

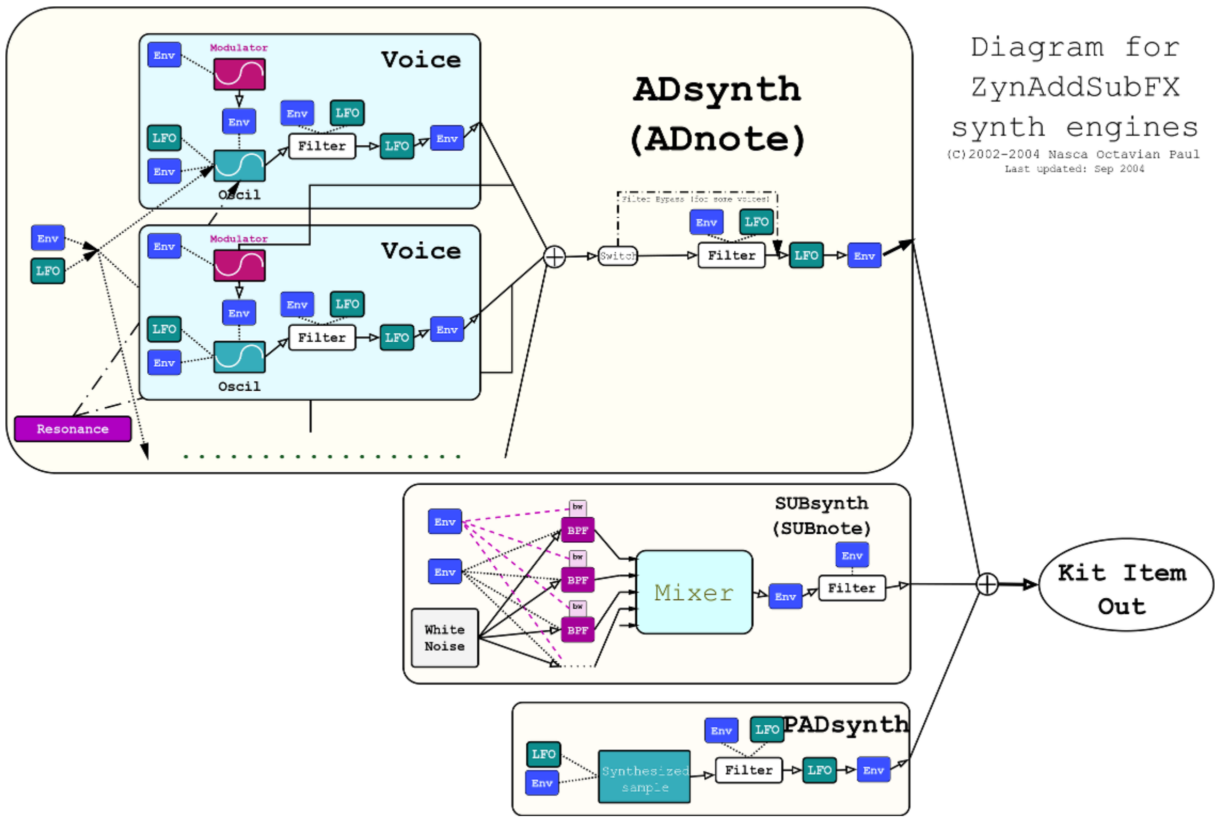

Figure 4: ZynAddSubFX/Yoshimi Note Generation

The figure represents the synthesizer module components. The continuous lines are the signal routing, and the dotted lines are frequency controlling signals (they controls the frequency of something). The dashed lines controls the bandwidths of bandpass filters. "Env" are the envelopes, "LFO" the Low Frequency Oscillators, "BPF" are band pass filters, "bw" are the bandwidth of the BPF's. If one uses instrument kits, the "note out" represents the output of the kit item.

2.5.4 Concepts / Basic Synthesis / Harmonics

Harmonics are sine waves that are multiple of the base frequency of a note. *Yoshimi* and *ZynAddSUbFX* introduce the concept of increasing the bandwidth of a harmonic so that it is not quite a sine wave.

2.5.4.1 Harmonic Bandwidth

"Harmonic bandwidth" does not refer to sample-rate, it refers to the frequency "spread" of each harmonic. This is the most important principle of making instruments that sound good. Unfortunately there is very little documentation about it.

Often it is believed that the pitched sounds (like piano, organ, choir, etc.) for a single note have a frequency, but it's actually harmonics and nothing more. Many people try to synthesize a sound using an exact frequency plus the harmonics, and observe that the result sounds too "artificial". They might try to modify the harmonic content, add a vibrato, tremolo, but even that doesn't sound "warm" enough. The reason is that the natural sounds don't produce an exact periodic; their sounds are quasi-periodic. Please notice that not all quasi-periodic sounds are "warm" or pleasant. (Nasca's discussion of periodic vs. quasi-periodic, and the figures he shows, are not included here.) Basically, by slightly increasing the bandwidth of a periodic sound, it is possible to make it quasi-periodic.

A very important thing about bandwidth and natural sounds is that the bandwidth has to be increased if one increases the frequency of the harmonic. If the fundamental frequency is 440 Hz and the bandwidth is 10 Hz (that means that the frequencies are spread from 435 to 445 Hz), the bandwidth of the second harmonics (880Hz) must be 20 Hz. A simple formula to compute the bandwidth of each harmonic if one knows the bandwidth of the fundamental frequency is $BWn = nbw1$, where `n` is the order of the harmonic, `bw1` is the bandwidth of fundamental frequency and `BWn` is the bandwidth of the n'th harmonic. If one does not increase the bandwidth according the frequency, the resulting instrument will (usually) sound too 'artificial' or 'ugly'. There are at least three methods of making good sounds with the above considerations:

1. **Detuning**. By adding slightly detuned sounds (in *Yoshimi* it is called "ADDsynth"). The idea is not new: it has been used for thousands of years in choirs and ensembles. That's why choirs sound so beautiful.
2. **Noise sculpting**. By generating white noise, subtracting all harmonics with band-pass filters and adding the results (in *Yoshimi* it is called "SUBsynth").
3. **Generation by spectrum**. By "drawing" the above graph that represents the frequency amplitudes on a large array, put random phases and do a single IFFT for the whole sample.

2.5.4.2 Harmonic Amplitude

An important principle of natural harmonics is to decrease the amplitude of higher harmonics on low-velocity notes.

All natural notes have this property, because on low-velocity notes there is not enough energy to spread to higher harmonics. On artificial synthesis one can do this by using a low-pass filter that lowers the cutoff frequency on notes with low velocities or, if one uses FM (frequency modulation), by lowering the modulator index. The spectrum of the sound should be almost the same according to the frequencies and not the harmonics.

This means that, for example, the higher the pitch is, the smaller the number of harmonics it will contain. This happens in a natural instrument because of the resonance. In this case there are many instruments that don't obey this, but sound quite good (example: synth organ). If one records the C-2 note from a piano and one plays it at a very high speed (8 times), the result will not sound like the C-5 key from the piano. The pitch is C-5, but the timbre is very different. This is because the harmonic content is preserved (the n-th harmonic will have the same amplitude in both cases) and not the spectrum (eg. the amplitudes of the harmonics around 1000 Hz are too different from one case to another).

In artificial synthesis one can use filters to add resonance or FM synthesis that varies the index according to the frequency. In *Yoshimi* one can add the resonance:

1. **ADDsynth**: Use the Resonance, a high harmonics sound content, and filters or FM.
2. **SUBsynth**: Add some harmonics and use the Global Filter.

2.5.5 Concepts / Basic Synthesis / Randomness

The main reason why the digital synthesis sounds too "cold" is because the same recorded sample is played over and over on each key-press. There is no difference between a note played the first time and second time. Exceptions may be the filtering and some effects, but these are not enough. In natural or analogue instruments this doesn't happen because it is impossible to reproduce exactly the same conditions for each note. To make a warm instrument one must make sure that it sounds slightly different each time. In *Yoshimi* one can do this:

1. **ADDsynth**: Set the "Randomness" function from Oscillator Editor to a value different than 0, or change the start phase of the LFO to the leftmost value.
2. **SUBsynth**: All notes already have randomness because the starting sound is white noise.
3. **PADSynth**: The engine starts the sample from random positions on each keystroke.

In setting the randomness of the oscillator output, there are two types of randomness. The first is *group randomness*, where the oscillator starts at a random position. The second is *phase randomness*: from -64 (max) to -1 (min) and each harmonic (the oscillator is phase distorted) is from 1 (min) to 63 (max). 0 is no randomness. One could use this parameter for making warm sounds like analogue synthesizers.

See the ADDSynth oscillator editor for this kind of control, named **Ph.rnd** or **rnd**.

There is now the possibility to add a 'naturalising' random pitch element to a part. This is found in the part edit window. The settings are not currently saved, but will be once the control values are settled, and there has been enough experience to decide whether it should be a part or voice setting.

2.5.6 Concepts / Basic Synthesis / Components

Important: All indexes of MIDI Channels, Parts, Effects starts from 0, so, for example, the first Part is 0.

Yoshimi components:

1. **Parts**. They receive the note messages from MIDI Channels. One may assign a part to any channel. A part can store only one instrument. "Add.S" represents ADDsynth and "Sub.S" is SUBsynth.
2. **Insertion Effect**. This effect applies only to one part; one can have any number of insertion effects for one part, but the number of these cannot be bigger than NUM.INS.EFX.
3. **Part Mixer**. Mixes all parts.
4. **System Effects**. Applied to all parts, one can set how much signal is routed through a system effect.
5. **Master mixer**. Mixes all outputs of Parts Mixers and System Effects.

2.5.7 Concepts / Basic Synthesis / Filters

Yoshimi offers several different types of filters, which can be used to shape the spectrum of a signal. The primary parameters that affect the characteristics of the filter are the cutoff, resonance, filter stages, and the filter type.

Cutoff: This value determines which frequency marks the changing point for the filter. In a low pass filter, this value marks the point where higher frequencies are attenuated.

Resonance: The resonance of a filter determines how much excess energy is present at the cutoff frequency. In *Yoshimi*, this is represented by the Q-factor, which is defined to be the cutoff frequency divided by the bandwidth. In other words higher Q values result in a much more narrow resonant spike.

Stages: The number of stages in a given filter describes how sharply it is able to make changes in the frequency response. The affect of the order of the filter is roughly synonymous with the number of stages of the filter. For more complex patches it is important to realize that the extra sharpness in the filter does not come for free as it requires many more calculations being performed. This phenomena is the most visible in SUBsynth, where it is easy to need several hundred filter stages to produce a given note.

The **Q**: value of a filter affects how concentrated the signals energy is at the cutoff frequency. For many classical analog sounds, high Q values were used on sweeping filters. A simple high Q low pass filter modulated by a strong envelope is usually sufficient to get a good sound.

Filter Type: There are different types of filters. The number of poles define what will happen at a given frequency. Mathematically, the filters are functions which have poles that correspond to that frequency. Usually, two poles mean that the function has more "steepness", and that one can set the exact value of the function at the poles by defining the "resonance value". Filters with two poles are also often referenced as Butterworth filters.

For the interested reader, functions having *poles* means that we are given a quotient of polynomials. The denominator has degree 1 or 2, depending on the filter having one or two poles. In the file `DSP/AnalogFilter.cpp`, `AnalogFilter::computefiltercoefs()` sets the coefficients (depending on the filter type), and `AnalogFilter::singlefilterout()` shows the whole polynomial (in a formula where no quotient is needed).

Filters are thoroughly described in section 4.2 ("Filter Settings") on page 56.

2.5.8 Concepts / Basic Synthesis / Envelopes

Envelopes are long-period wave forms that are applied to frequency, amplitude, or filters. Envelopes generate effects such as tremolo and vibrato, as well as effects that occur when a sound-generating physical component changes shape. Envelopes are thoroughly described in section 4.4 ("Envelope Settings") on page 66.

2.6 Concepts / MIDI

It is useful to discuss some of the details of MIDI in order to understand *Yoshimi*. Obviously, we assume some knowledge already, or one wouldn't be running *Yoshimi*.

2.6.1 Concepts / MIDI / Messages

Yoshimi responds to the following MIDI messages.

For the controllers that are not defined in GM:

- **Bandwidth** control (75) increases or decreases the bandwidth of instruments. The default value of this parameter is 64.
- **Modulation amplitude** (76) decreases the amplitude of modulators on ADDsynth. The default value of this parameter is 127.
- **Resonance Center Frequency** control (77) changes the center frequency of the resonance.
- **Resonance Bandwidth** control (78) changes the bandwidth of the resonance.

The Program Change (192) also provides user selectable CC for voices 128-160. There is an option to make Program Change enable a part if it's currently disabled.

User selectable CC for Bank Root Path change. For more details of bank changes see section 2.4 ("Concepts / Banks and Roots") on page 14.

Instruments inside banks should *always* have file-names that begin with four digits, followed by a hyphen. Otherwise the results can be rather unpredicatable.

Table 1: ZynAddSubFX/Yoshimi MIDI Messages

0 or **32**	Bank Change (user selectable, does *not* force a program change)
1	Modulation Wheel
7	Volume
10	Panning
11	Expression
64	Sustain pedal
65	Portamento On/Off
71	Filter Q (Sound Timbre)
74	Filter Cutoff (Brightness)
75	BandWidth (different from GM spec)
76	FM amplitude (different from GM spec)
77	Resonance Center Frequency (different from GM spec)
78	Resonance Bandwith (different from GM spec)
120	All Sounds OFF
121	Reset All Controllers
123	All Notes OFF
192	Program Change (voices 1-128)
224	Pitch Bend

2.6.2 Concepts / MIDI / NRPN

NRPN stands for "Non Registered Parameters Number". NRPNs can control all System and Insertion effect parameters. Using NRPNs, *Yoshimi* can now directly set some part values regardless of what channel that part is connected to. For example, one may change the reverb time when playing to keyboard, or change the flanger's LFO frequency.

NRPNs are described in greater detail in section section 13 ("Non-Registered Parameter Numbers") on page 158.

2.6.2.1 Concepts / MIDI / NRPN / Vector Control

Vector control is a way to control more than one part with the controllers. Vector control is only possible if one has 32 or 64 parts active in *Yoshimi*. In vector mode parts will still play together but the vector controls can change their volume, pan, filter cutoff in pairs, controlled by user defined CCs set up with NRPNs.

Vector control is described in greater detail in section section 13.2 ("NRPN / Vector Control") on page 160.

2.6.2.2 Concepts / MIDI / NRPN / Effects Control

NRPNs are very useful in modifying the parameters of the *Yoshimi* effects.

Effects control is described in greater detail in section section 13.3 ("NRPN / Effects Control") on page 161.

2.7 Concepts / LV2 Plugin

Yoshimi now runs as an LV2 plugin.

Supported features:

1. Sample-accurate midi timing.
2. State save/restore support via LV2_State_Interface.
3. Working UI support via LV2_External_UI_Widget.
4. Programs interface support via LV2_Programs_Interface.
5. Multi channel audio output. 'outl' and 'outr' have LV2 index 2 and 3. All individual ports numbers start at 4.

Planned feature: Controls automation support. This will be a part of a common controls interface.

Download and build the source code found at the *Yoshimi* site [16], and one will find a file named `LV2_Plugin/yoshimi_lv2.so`

The LV2 *Yoshimi* interface can be run in hosts such as Ardour 3, Carla, and QTractor.

Apparently, *ZynAddSubFX* can also be used as an LV2 plugin with the help of the carla-lv2 project, by drag-and-dropping an `.xiz` or `.xmz` file into it.

At some point we hope to document the process of setting up and using the *Yoshimi* LV2 plugin.

3 Menu

The *Yoshimi* menu, as seen at the top of figure 2 ("Yoshimi Main Screen, 1.3.5") on page 11, is fairly simple, but it is important to understand the structure of the menu entries.

3.1 Menu / Yoshimi

The *Yoshimi* menu entry contains the sub-items shown in figure 5 ("Yoshimi Menu Items") on page 23. The next few sub-sections discuss the sub-items in the *Yoshimi* sub-menu. (Note that, in *ZynAddSubFX*, this menu is called the *File* menu.)

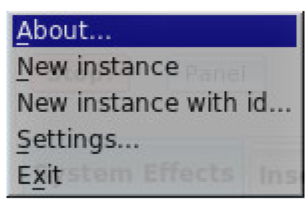

Figure 5: Yoshimi Menu Items

Bug: There seems to be a bug in that the expected menu hot-keys (Alt-Y, Alt-I, Alt-P, and Alt-S) do not work (Yoshimi 1.3.5).

3.1.1 Menu / Yoshimi / About...

There is no "Help" menu in *Yoshimi*. Therefore, the "About" dialog appears in the "Yoshimi" menu, as shown in figure 6 ("Yoshimi Menu, About Dialog") on page 24. These guys need some acknowledgment for their hard work! And they acknowledge the massive groundwork laid by the *ZynAddSubFX* project.

Figure 6: Yoshimi Menu, About Dialog

3.1.2 Menu / Yoshimi / New instance

Creates a new instance of *Yoshimi*. We're not quite sure what this one does, really. Does it create a new run of *Yoshimi* with a random `--name-tag` value? In our basic investigation, it simply finds that the previous *Yoshimi* instance has grabbed audio access, when JACK is not being used:

```
Yay! We're up and running :-)
failed to open alsa audio device:default: Device or resource busy
AlsaClient audio open failed
Failed to open MusicClient
Yoshimi stages a strategic retreat :-(
```

Now, if JACK is running, then this feature will work. Start a normal (JACK-using) instance of *Yoshimi*. Then use this menu entry. *Yoshimi* will start another instance of itself, with an ID of 1. This instance can be verified by running a JACK session manager such as QJackCtl.

It is important to note that each instance of *Yoshimi* has its own configuration file. Each also has its own MIDI and audio ports. Thus, these instances are independent of each other.

3.1.3 Menu / Yoshimi / New instance with id...

Creates a new instance of *Yoshimi* with an ID that is a number. See figure 7 ("Yoshimi Menu, Instance Dialog") on page 25. It tries to open a *Yoshimi* instance based on the configuration found in the file `/.config/yoshimi/yoshimi.configXX`, where *XX* is the ID one supplied.

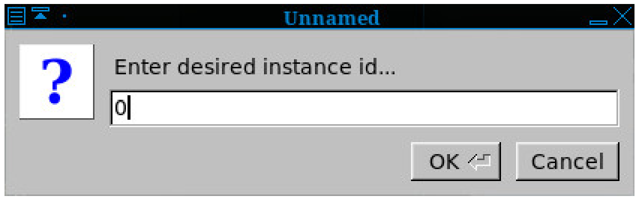

Figure 7: Yoshimi Menu, Instance Dialog

Useful when connecting devices with JACK. Start a normal (JACK-using) instance of *Yoshimi*. Then use this menu entry, supply a number as an ID. *Yoshimi* will start another instance of itself, with an ID of whatever number one specified. This instance can be verified by running a JACK session manager such as QJackCtl.

Again, though, in a non-JACK setup it simply fails.

3.1.4 Menu / Yoshimi / Settings...

The *Yoshimi Settings* dialog contains five tabs that control the major and overall settings of *Yoshimi*.

3.1.4.1 Menu / Yoshimi / Settings / Main Settings

The Main Settings tab controls the main configuration items that follow, which apply to all patches/parts/instruments. The main settings are shown in figure 8 ("Yoshimi Main Settings Tab") on page 25below.

All these settings only take effect after restarting the synthesizer.

The settings dialogs are quite different between *ZynAddSubFX* and *Yoshimi*.

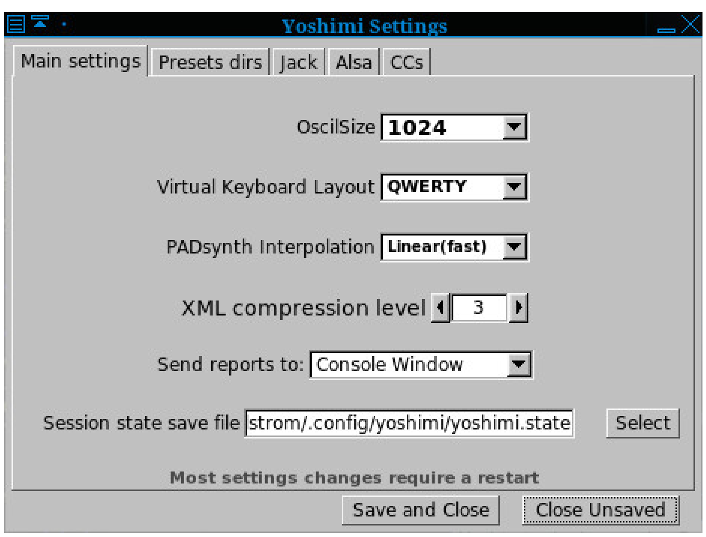

Figure 8: Yoshimi Main Settings Tab

The following settings exist in the *Main settings* tab:

1. **OscilSize**
2. **Virtual Keyboard Layout**

3. **PADsynth interpolation**
4. **XML compression level**
5. **Send reports to**
6. **Session state save file**
7. **Select**
8. **Save and Close**
9. **Close Unsaved**

1. OscilSize. ADDsynth Oscillator Size (in samples).

Values: `128, 256, 512+, 1024*, 2048, 4096, 8192, 16384`

Sets the number of the points of the ADDsynth oscillator. The bigger is better, but it takes more CPU time on the start of any note, and it may add latency to some processes.

The default value for *Yoshimi* is shown marked with an asterisk, and the default value for *ZynAddSubFX* is 512. (This asterisk/plus-sign convention is used throughout this manual). See figure 9 ("OscilSize Values") on page 26below for the OscilSize dropdown element.

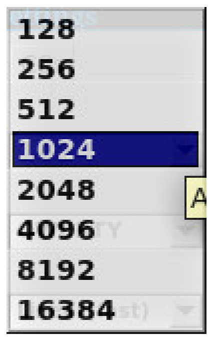

Figure 9: Adsynth Oscillator Size (samples)

2. Virtual Keyboard Layout. The virtual keyboard is useful, but it is difficult to move the mouse rapidly to the next key on the virtual keyboard. Therefore, *Yoshimi* supports using the computer keyboard to produce notes.

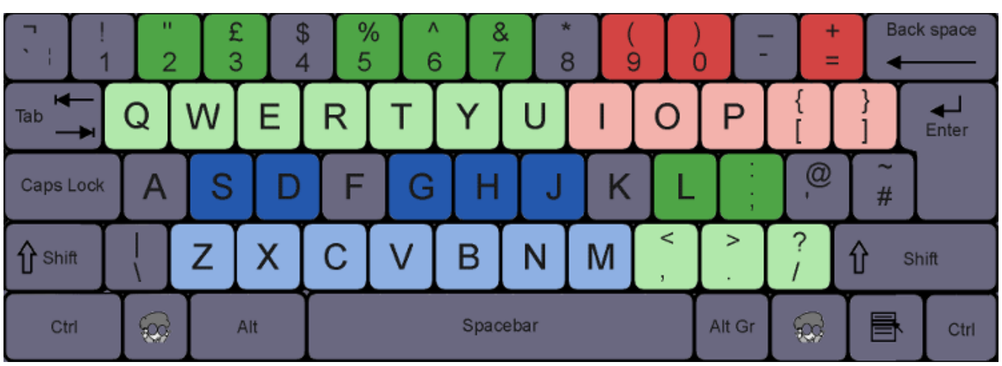

Figure 10: QWERTY Virtual Keyboard

See figure 10 ("QWERTY Virtual Keyboard") on page 26, for the mapping of the computer keyboard to the virtual keyboard. Three octaves (blue, green, and red) are available, with the dark keys of each color representing the "black" keys. Note that this is a QWERTY layout. *Yoshimi* also supports other keyboard layouts. See figure 11 ("Virtual Keyboard Layout") on page 27, for the virtual keyboard layout settings dropdown.

Values: `Dvorak, QWERTY*, AZERTY`

Figure 11: Virtual Keyboard Layout Values

3. PADsynth interpolation.

Values: `Linear(fast)*, Cubic(slow)`

See figure 12 ("PADSynth Interpolation") on page 27below for the interpolation values.

Figure 12: PADSynth Interpolation Values

4. XML compression level. Compression level of *Yoshimi* XML files.

Values: `0 to 9, 3*`

The settings and instruments of *Yoshimi* are preserved in XML files. The value of 0 indicates that the XML file is uncompressed. In general, 0 is probably the best setting. Setting this option makes the XML files a bit larger, perhaps larger by a factor of more than 10, making a 10K file into a 180K file. For a little "wasted" space, one can view the XML file in a text/programmer's editor. But, if ones system is tight on disk space, higher levels of compression can be specified.

5. Send reports to.

Values: `stderr, Console Window*`

Notices and error messages can be sent to the standard error log of the terminal in which *Yoshimi* can be run, or, more usefully, to an output console window. See figure 13 ("Send Reports") on page 27. It provides a depiction of the selection dropdown.

Figure 13: Send Reports To

6. Session state save file. Main Settings Session State Save File. Enter the name of the desired session state file here, including the path to it. Example: `/home/myself/.config/yoshimi/yoshimi.state`

7. Select. Select Saved-State File. See figure 14 ("Session Save State") on page 28. It provides a depiction of this dialog, which lets one pick an existing file as the *Yoshimi* state file.

Values: `/.config/yoshimi/yoshimi.state`

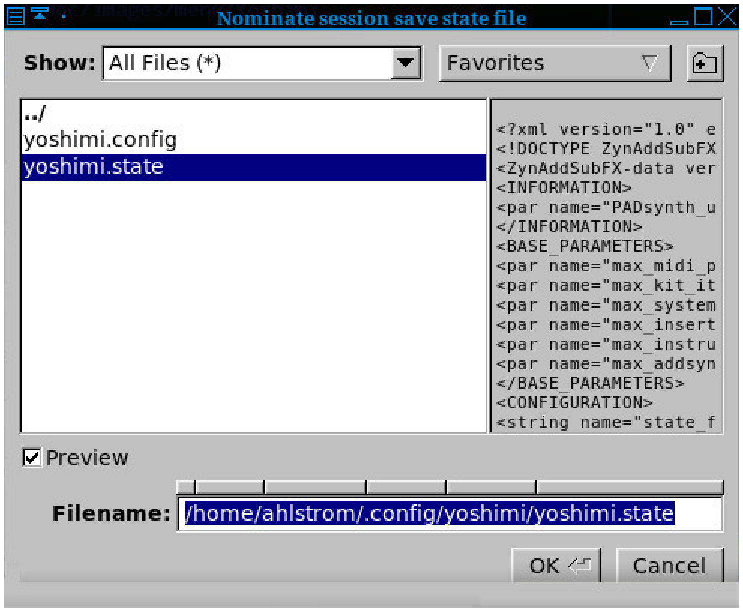

Figure 14: Session Save State File

8. Save and Close. This selection saves and closes the *Yoshimi* settings dialog.

9. Close Unsaved. Close Unsaved, Main Settings.

This selection closes the *Yoshimi* settings dialog. However, it may actually leave the changes preserved (a bug). This needs to be checked, we've seen this in other dialogs. **Bug:** And yes, indeed, it saves the changes. Please watch for this effect in all such dialogs; we're indexing such defects under the topic of "bugs".

3.1.4.2 Menu / Yoshimi / Settings / Preset dirs

The *Yoshimi* preset directories are the locations where presets can be found. When first installed, the system preset directory is

```
/usr/share/yoshimi/presets
```

The user can provide additional directories for the presets. These directories are useful for containing copies of the system presets that one can modify safely, and for providing custom presets designed by the user.

The following items are provided by the preset directory settings:

1. **Preset list**
2. **Add preset directory...**
3. **Remove preset directory...**
4. **Make default**
5. **Save and Close**
6. **Close Unsaved**

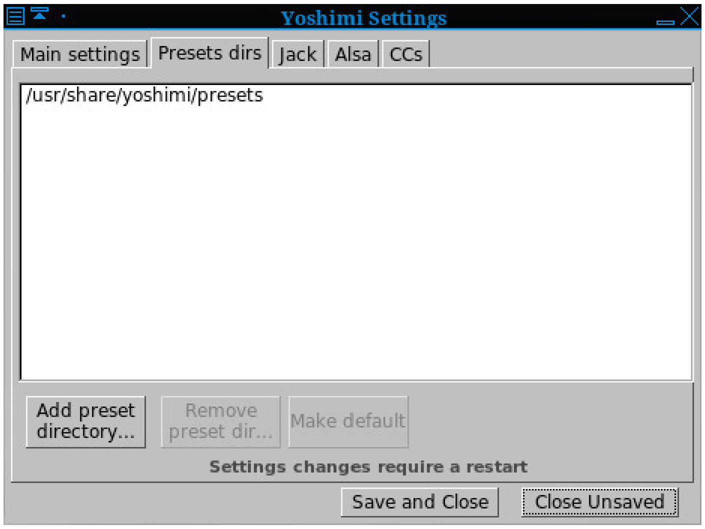

Figure 15: Yoshimi Preset Dirs Dialog

1. Preset list. This interface element contains a list of preset directories. By default, the only directory present is the installed preset directory. For example, `/usr/share/yoshimi/presets`. Another example would be this project; let YOSHIMI-DOC be the directory where this project is stored. Then one can add `YOSHIMI-DOC/config/yoshimi/presets` to this list, using the button described next.

2. Add preset directory.... Use this button and dialog to add a preset directory to the list, for easy access.

Press the **Add preset directory...** button, revealing the following dialog.

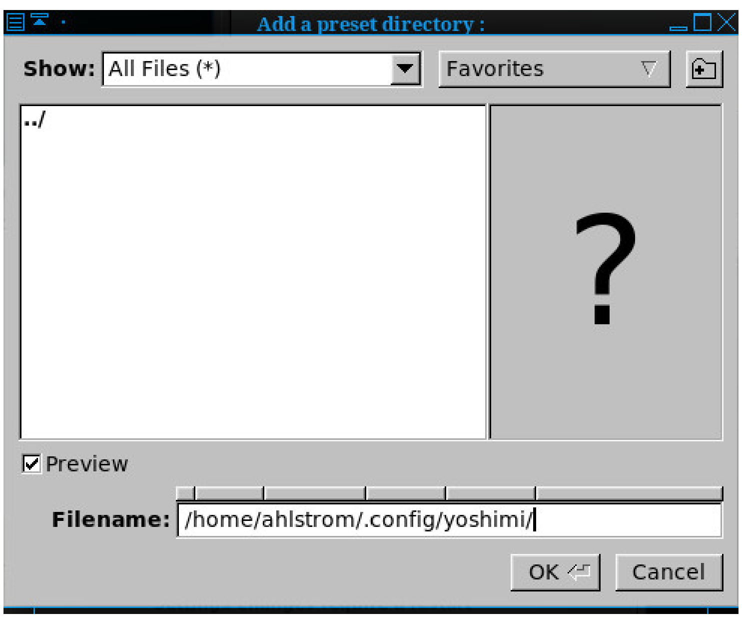

Figure 16: Add a Preset Directory

Navigate to the desired directory, select it, and press the **Ok** button. The press the **Save and Close** button. IMPORTANT: Then restart **Yoshimi**.

3. Remove preset directory.... Select one of the preset directories in the preset list, then press this button to remove the preset directory from the list of preset directories.

4. Make default presets. Make Default Presets Directory. Select one of the preset directories in the preset list, then press this button to make the preset directory the default preset directory.

5. Save and Close presets. Press this button to save the changes.

6. Close Unsaved presets. Close Unsaved, Presets.

Bug: This button doesn't seem to work, in that the changes seem to remain in place when the dialog is reopened.

3.1.4.3 Menu / Yoshimi / Settings / Jack

JACK is the "Jack Audio Connection Kit", useful increasing audio performance and configurability.

When using the JACK audio backend, parts can be individually routed and sent to the main L/R outputs. This is controlled from the panel window, section 5.1 ("Mixer Panel Window") on page 77, and the settings are saved with all the other parameters.

Direct part outputs carry the Part and Insertion effects, but not the System ones.

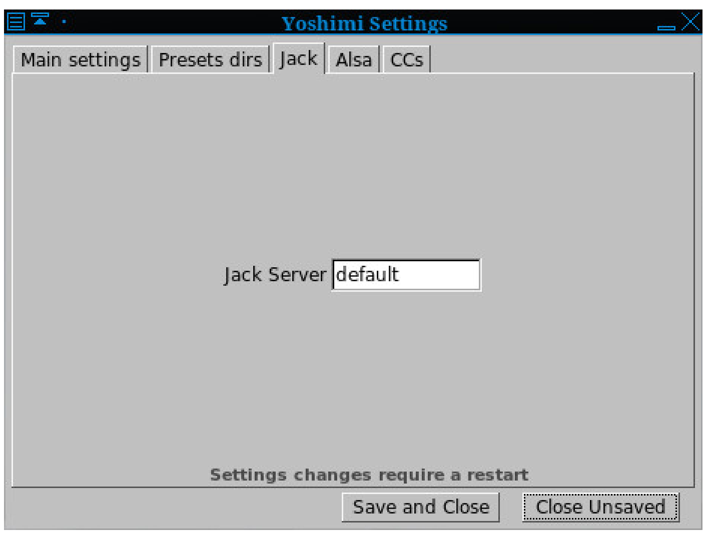

Figure 17: JACK Settings Dialog

1. Jack Server. Jack Server Name. It is possible to have more than one JACK server running. This option tells this instance of *Yoshimi* which server to use.

Values: `default*, name` name, as in "jackd –name"

2. Save and Close. Save and Close, JACK Settings.

3. Close Unsaved Jack. Close Unsaved, JACK Settings.

Bug: This button allows changes to be saved.

3.1.4.4 Menu / Yoshimi / Settings / Alsa

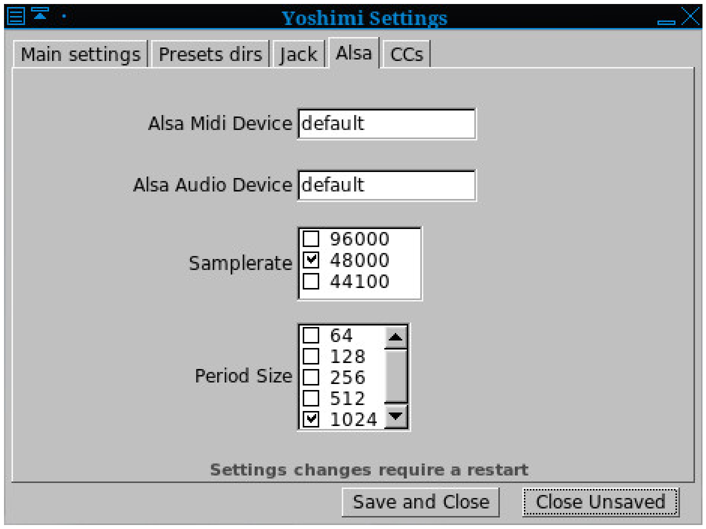

Figure 18: ALSA Settings Dialog

1. Alsa Midi Device. ALSA MIDI Device.

Values: `default*`

TODO: What can we change this value to?

2. Alsa Audio Device. ALSA Audio Device.

Values: `default*`

TODO: What can we change this value to? "hw:0"

3. Samplerate. Sample Rate. Sets the quality of the sound, higher is better, but it uses more CPU. One can select from a list.

ZynAddSubFX: if one wants a sample-rate that is not in the list, select "Custom" and change the value from the right. Default is 44100.

Values: `96000, 48000*, 44100`

4. Period Size. Buffer Size. Sets the granularity of the sound. The Default is 256 (TODO: 1024?) samples. To find out the internal delay in milliseconds, divide the Period Size value by the Sample Rate and multiply the result by 1000 (eg.: 256/44100*1000=5.8 ms).

Values: `64, 128, 256, 512, 1024*`

5. Save and Close. Save and Close, ALSA Settings.

6. Close Unsaved Alsa. Close Unsaved, ALSA Settings.

Bug: This button allows changes to be saved.

3.1.4.5 Menu / Yoshimi / Settings / CCs

This dialog, shown in figure 19 ("MIDI CC Preferences") on page 32, presents MIDI "continuous controller" (CC) preferences.

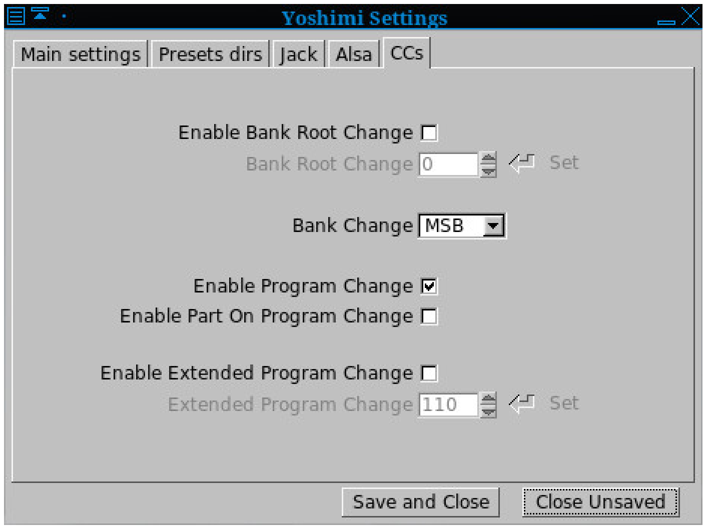

Figure 19: MIDI Continuous Controller (CC) Preferences

The concepts of banks and roots is very useful. See section 2.4 ("Concepts / Banks and Roots") on page 14.

1. Enable Bank Root Change. Enable Bank Root Change.

Values: `Off*, On`

2. Bank Root Change.

Values: `0*, to 127`

If enabled, a new reddish button, **Pending**, appears. Once the change has been made in the scroll list, click this button to set the change.

It is not clear if this change persists after *Yoshimi* closes, or if the session or state must be saved. But apparently this change can be made without a restart being required.

3. Bank Change. Bank Change. Defines which continuous controllers one wants to use. Note that MIDI Controller 0 = CC0 = Bank Select MSB, and MIDI Controller 32 = CC32 = Bank Select LSB. When combined, these Bank Select messages provide

$$128 * 128 = 16384$$

banks.

But note that all a Bank Select does is selects the bank for the next Program Change event. The program doesn't change after changing a bank, until a Program Change is sent.

Bank changes can be completely disabled - some hardware synths don't play nice with banks.

Values: `LSB, MSB*, Off`

4. Enable Program Change.

Values: `Off*, On`

Enables/disables MIDI program change. Program changes can be completely disabled, but some hardware synths don't play nice!

5. Enable Part On Program Change.

Values: `Off*, On`

The part is enabled if the MIDI program was changed on this part.

6. Enable Extended Program Change.

Values: `Off*, On`

7. Extended Program Change. If enabled, a new reddish button, **Pending**, appears. Once the change has been made in the scroll list, click this button to set the change.

Values: `0-127, 110*`

8. Save and Close. Save and Close, Continuous Constrollers Settings.

9. Close Unsaved CC. Close Unsaved, Continuous Controllers Settings.

Bug: This button allows changes to be saved.

3.1.5 Menu / Yoshimi / Exit

Simply exits from *Yoshimi*. The user is prompted if unsaved changes exist, as shown in figure 20 ("Yoshimi Menu, Exit") on page 33.

Figure 20: Yoshimi Menu, Exit

3.2 Menu / Instrument

The *Yoshimi* Instrument menu lets one select instruments and work with banks of instruments. *Yoshimi* stamps instrument XML files with its own major and minor version numbers so it is possible to tell which version created the files, or whether they were created by *ZynAddSubFX*.

When opening an instrument bank one can now tell exactly which synth engines are used by each instrument. This is represented by three pale background colours:

- Red: ADDsynth
- Blue: SUBsynth
- Green: PADsynth

If the instruments are kits they scanned to find out if *any* member of the kit contains each engine. This scanning is duplicated in the current part, the mixer panel for the currently loaded instruments, and in the Instrument Edit window the same colors highlight the engine names when they are enabled with the check boxes.

The following sub-menus are provided, as shown in figure 21 ("Yoshimi Menu, Instrument") on page 34.

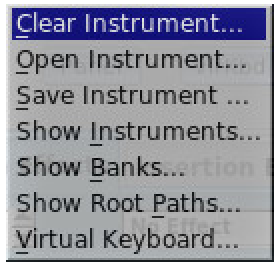

Figure 21: Yoshimi Menu, Instrument

1. **Clear Instrument...**
2. **Open Instrument...**
3. **Save Instrument...**
4. **Show Instruments...**
5. **Show Banks...**
6. **Show Root Paths...**

3.2.1 Menu / Instrument / Clear Instrument...

This menu entry brings up a prompt to clear the parameters of the instrument.

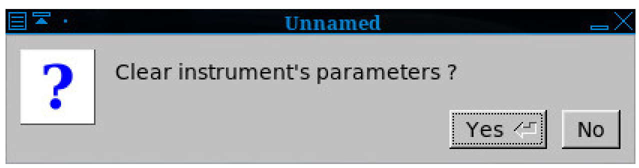

Figure 22: Clear Instrument Dialog

Bug: Sometime it seems that one needs to clear the instrument if one is loading a new instrument to test it out, because some settings seem to remain from the previous instrument. Don't quote us on that.

3.2.2 Menu / Instrument / Open Instrument...

This menu entry brings up a prompt to open a new instrument.

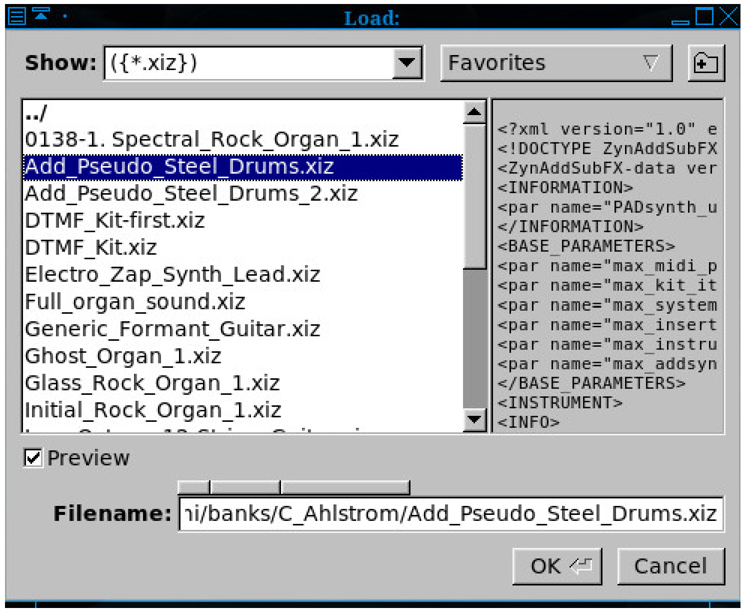

Figure 23: Open Instrument Dialog

This dialog has a number of user-interface elements to note.

1. **Show**
2. **Favorites**
3. **Create a new diretory**
4. **Instrument List**
5. **XML Preview**
6. **Preview**
7. **Show hidden files**
8. **Directory Bar**
9. **Filename**
10. **OK**
11. **Cancel**

1. Show. Show types of files. This item shows a file filter for selecting instrument files. The types of filters are as follows (screen shot not available):

1. **({*.xiz})** (compressed XML files)
2. **All Files (*)**
3. **Custom Filter**

2. Favorites. Favorite directories. Provides a list of options and favorite directories in which to find instrument files.

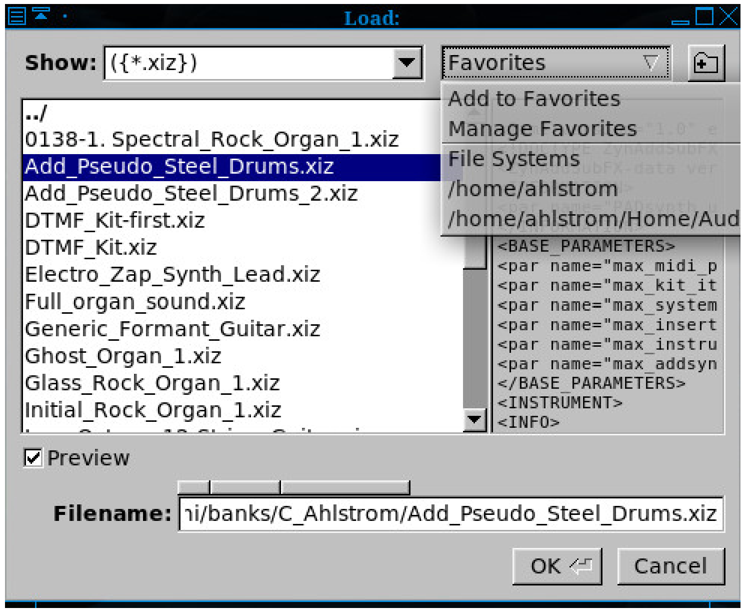

Figure 24: Favorites Dropdown

1. **Add to Favorites**
2. **Manage Favorites**
3. **File Systems**
4. **(Additional favorite directories)**

Add to Favorites simply adds the currently selected directory shown in the instrument list to the list of favorites.

To add Favorites in the file dialog, navigate to the desired directory. Then click **Favorites**, and select **Add to Favorites**.

Once one has a number of favorites set up, there is a **Manage Favorites** that can be used. For example, if one needs to get rid of a directory, one can use the **Manage Favorites** dialog, shown in figure 25 ("Favorites Dropdown") on page 36below, to do that.

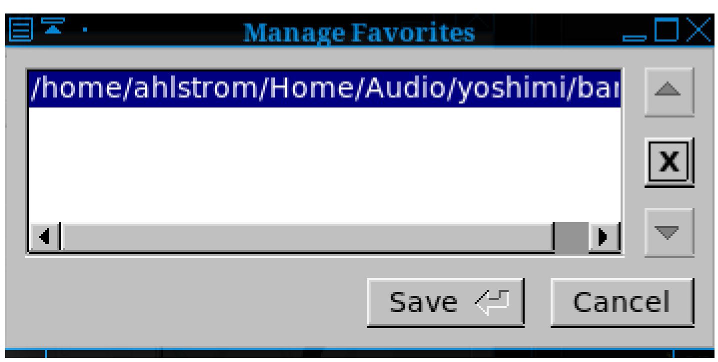

Figure 25: Favorites Dropdown

File Systems Provides a list of all file systems starting at root ("/"). This list can be pretty confusing, with a lot of entries. But note that one navigates to ("/"), and from there to `/usr/share/yoshimi/banks` to get easy access to all the instruments that are preinstalled with *Yoshimi*. Unfortunately, one can only add instrument files in this manner, and the dropdown would quickly become unmanageable. So generally, one will want to use only **Add to Favorites** and **Manage Favorites**.

Note that *banks* are an important concept in *Yoshimi*, as we will see in section 3.2.4 ("Menu / Instrument / Show Instruments..") on page 37.

3. Create Directory. Creates a New Directory. This little symbol options a small "New Directory?" dialog (not shown here, it is very simple and stock) into which one can type a directory name to be added to the current directory of the instrument list.

4. Instrument List. Provides a list of the instrument files available in the current directory. Also shown are sub-directories (if available) that might contain more instruments, and a ("`../`") entry to navigate to the parent directory.

5. Preview. If one thinks the preview feature is not useful, uncheck this check-box. so that one doesn't see the preview window. As a bonus, one can see more of the instrument file-name.

6. Preview pane. XML Preview. This box can show the beginning of the XML data of an instrument file. **Bug:** It seems to show the XML only if the XML is not compressed.

7. Show hidden files. Shows file that are hidden. Not sure how useful this feature is; who would hide a *Yoshimi* instrument file?

8. Directory Bar. Provides an alternate way to move up through the directory structure.

9. Filename. File Name. Provides the full path to the instrument file.

10. OK/Cancel. We don't really need to discuss the **OK** and **Cancel** buttons, do we?

3.2.3 Menu / Instrument / Save Instrument...

This menu entry brings up a prompt to save a new instrument. It has all of the user-interface elements of the "Open Instrument" dialog shown in figure 23 ("Open Instrument Dialog") on page 35in section 3.2.2 ("Menu / Instrument / Open Instrument..") on page 34.

However, if nothing has changed, then a "Nothing to Save!" prompt (not pictured) is shown.

3.2.4 Menu / Instrument / Show Instruments...

The instruments can be stored in banks. These are loaded/saved automatically by the program, so one doesn't have to worry about saving the banks before the program exits. On program start, the last used bank is loaded. A single bank can store up to 128 instruments. However, there is space for a number of additional instruments in the bank.

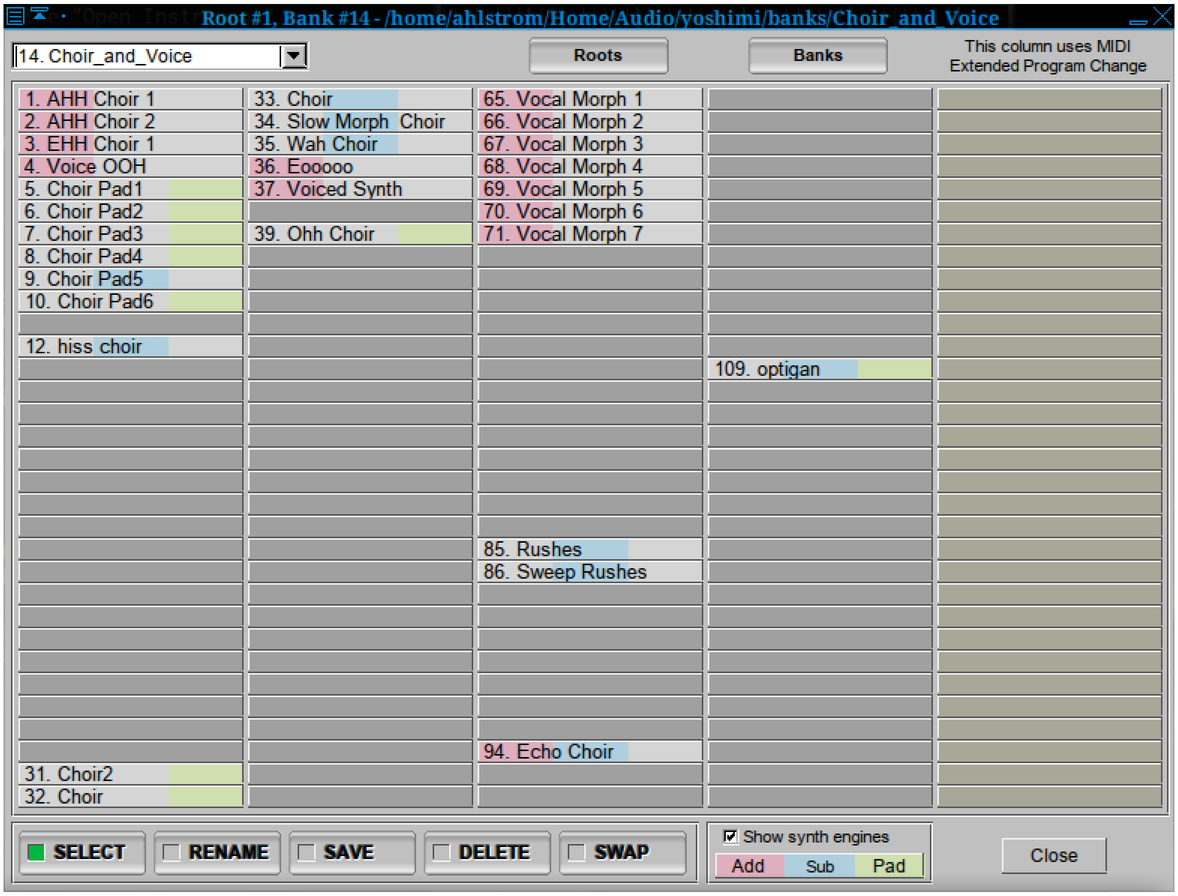

Figure 26: Show Instruments, 1.3.5 and Above

As figure 26 ("Show Instruments") on page 38shows, this is a very complex dialog with a lot of options. Note how *Yoshimi* now shows the color codings for the synth-sections use: red for ADDsynth, blue for SUBsynth, and green for PADsynth.

Also note how the numbers at the beginning of the filenames are used as an "instrument" or "program" number.

Learning how to use this dialog is an important way to make instruments easier to manage, and so this will be a long section.

An important pair of concepts in *Yoshimi* are *banks* and *roots*. These concepts are described in section 2.4 ("Concepts / Banks and Roots") on page 14.

A bank has 3 modes in *ZynAddSubFX*:

1. **READ**. The instrument is loaded from the bank to the current part.
2. **WRITE**. The instrument is written to the bank.
3. **CLEAR**. The instrument from the bank is cleared (removed).

Pressing the left mouse button on a slot reads/writes/clears the instrument from/to it (according to the current mode).

Pressing the right mouse button on a slot changes its name.

The setup in *Yoshimi* is a bit different than in *ZynAddSubFX*. Observe figure 27 ("Show CA's Instruments") on page 39. It shows a bank loaded from a directory containing customs banks from one of the authors of this document. Note that this dialog has been modified in recent versions of *Yoshimi*.

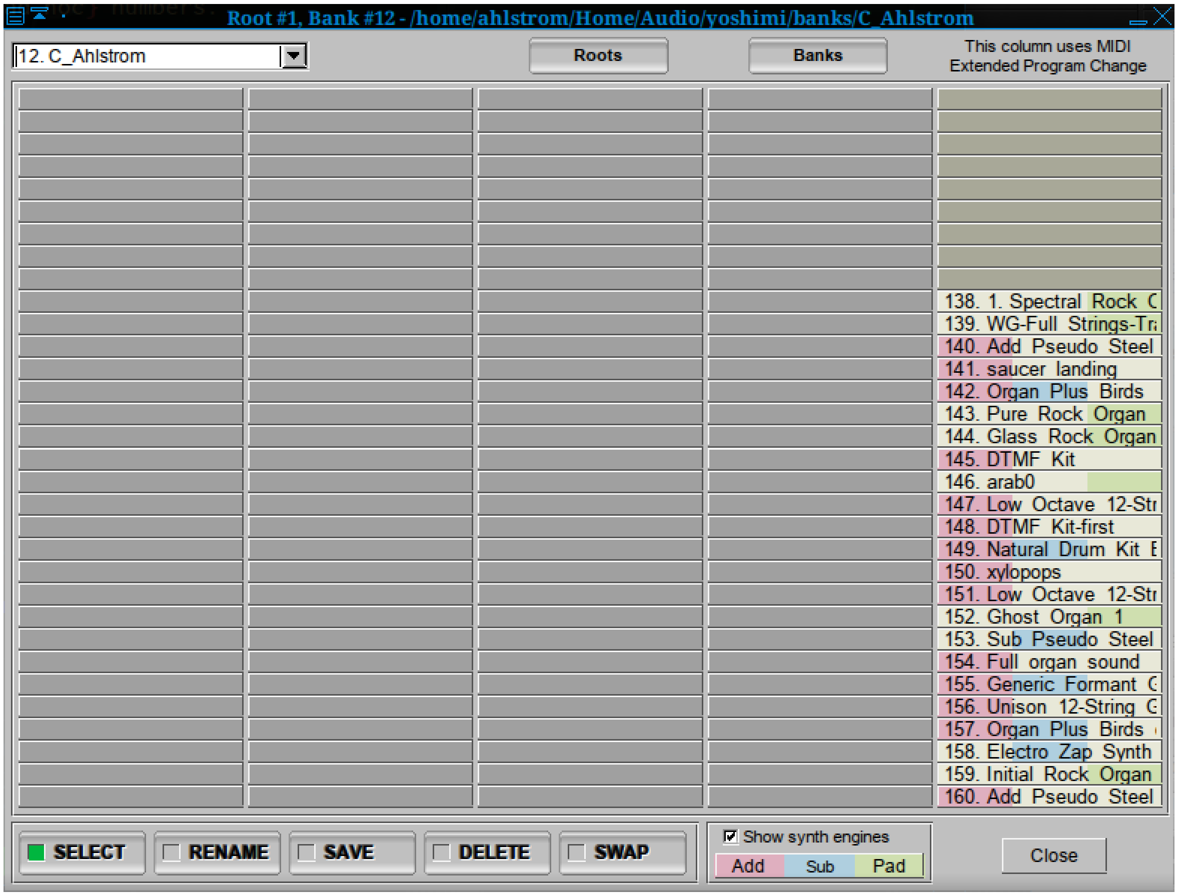

Figure 27: Show My Instruments

The interesting feature of this figure is that all of the file-names are not numbered, and therefore they show up in the extended (rightmost) column of the dialog, prepended with *ad hoc* numbers that *Yoshimi* generates.

Here is a list of the user-interface items in the instruments/banks dialog. Please refer to figure 26 ("Show Instruments") on page 38as well.

1. **Bank Name**
2. **Refresh banks** (OBSOLETE)
3. **Roots**
4. **Banks**
5. **New Bank** (OBSOLETE)
6. **Instrument and Bank Matrix**
7. **SELECT**
8. **RENAME**
9. **SAVE**
10. **DELETE**
11. **SWAP**
12. **Show synth engines** (was **Show PADsynth status**)
13. **Close**

1. Bank Name. Instruments Bank Name. Basically, each bank is a directory name, with a number prepended. The banks are found under a current "root", which is a also a directory name, and is the name of the parent directory of a set of banks. Here is the dropdown list for "my" setup:

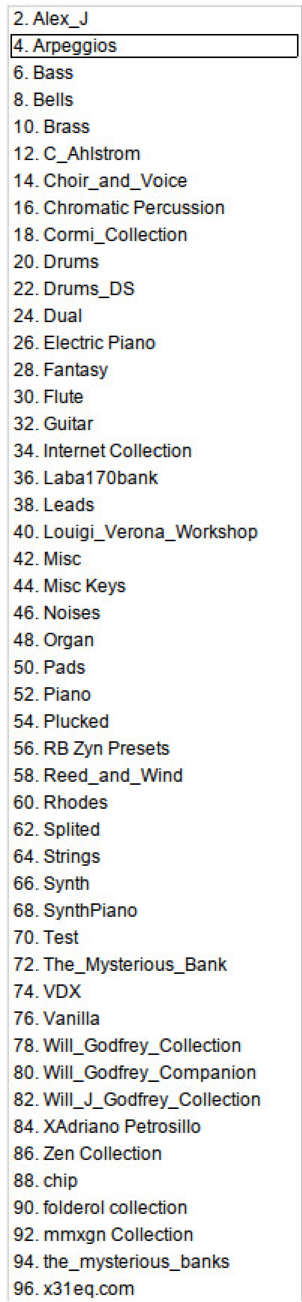

Figure 28: A Sample Bank List

And here is the directory listing associated with it, in the order produced by the UNIX/Linux "ls -1" (list single-column) command (shown in two columns to save space):

```
Alex_J                         Noises
Arpeggios                      Organ
```

```
Bass                      Pads
Bells                     Piano
Brass                     Plucked
C_Ahlstrom                RB Zyn Presets
chip                      README
Choir_and_Voice           Reed_and_Wind
Chromatic Percussion      Rhodes
Cormi_Collection          Splited
Drums                     Strings
Drums_DS                  Synth
Dual                      SynthPiano
Electric Piano            Test
Fantasy                   The_Mysterious_Bank
Flute                     the_mysterious_banks
folderol collection       Vanilla
Guitar                    VDX
Internet Collection       Will_Godfrey_Collection
Laba170bank               Will_Godfrey_Companion
Leads                     Will_J_Godfrey_Collection
Louigi_Verona_Workshop    x31eq.com
Misc                      XAdriano Petrosillo
Misc Keys                 Zen Collection
mmxgn Collection
```

The first thing to note is that there are only 128 *Yoshimi* banks supported in a *Yoshimi* root. The list above takes up about half of the available slots, so it might be time to move some of those banks to a new root directory.

The numbers in the dropdown list are generated by *Yoshimi* the first time it sees a new root path or a new bank within the root path. Once set, these numbers will never change unless you actually move them around (using the **SWAP** button).

The bank number is also the MIDI ID; one can be sure that it will alway be there for bank changes no matter how many banks are added later. *Yoshimi* always lists the banks in ID order, not alphabetical order, so one can group them sensibly and permanently. However, at first-time creation *Yoshimi* sets the IDs in alphabetical order and tries to space them evenly over the range to provide wiggle room.

Selecting one of the items in this dropdown list selects the bank and loads it into the Banks dialog. Using a right mouse click often produces modified or alternate results, especially in roots/banks/instruments.

2. Refresh banks. Instruments Refresh Banks. This item is *not present* in the newest *Yoshimi*.

3. Roots. Instruments Roots Button. Shows a list of directories that can serve as "root" directories. The "Bank Root Paths" dialog shown in figure 31 ("Show Root Paths") on page 45shows the system root (e.g. `/usr/share/yoshimi/banks`) and a user's home location for his/her banks and roots.

4. Banks. Instruments Banks Button. The dialog brought up by this button (see figure 30 ("Show Banks") on page 44) is different view of the dialog shown in figure 28 ("A Sample Bank List") on page 40. Again, we're not yet sure where the numbers in the list come from. They seem to be autogenerated.

5. New Bank. Instruments New Bank. This item is *not present* in the newest *yoshimi*. The **ADD** button can be used instead.

6. Instrument and Bank Matrix. Instruments Bank Matrix. Shows the instruments that are in the currently selected bank (directory).

7. Instruments. Instruments Button. This seems to be present in the newest version of *Yoshimi*, and brings up the instruments in the current bank.

8. SELECT. Instruments SELECT. When this button is selected, then clicking on a bank brings up a very similar dialog that shows the instruments in that bank, laid out with the numbers that are prepended to the filename of each instrument in that bank.

9. RENAME. Instruments RENAME. When this button is selected, then clicking on a bank brings up a small dialog to rename the clicked-on bank.

10. SAVE. Instruments SAVE. Saves the instruments as currently configured.

11. DELETE. Instruments DELETE. Selecting this button and clicking an empty bank entry does nothing. Selecting this button and clicking an existing bank entry brings up a small dialog asking one if this bank is really to be deleted.

12. SWAP. Instruments SWAP. Selecting this button, then selecting one bank, and then another, swaps the numbering and postion of the selected banks.

13. Show synth engines. If enabled, then the usage of each of the *Yoshimi* synthesis engines is indicated by color coding, as shown in the figure above.

14. Close. Closes the window.

Here is a more conventional view of instruments, supplied with *Yoshimi*, shown in figure 29 ("Show Pads Instruments") on page 43.

Figure 29: Show Pads Instruments

3.2.5 Menu / Instrument / Show Banks...

As shown in figure 30 ("Show Banks") on page 44, the banks dialog uses most of the same user-interface elements as the instruments dialog shown in figure 27 ("Show CA's Instruments") on page 39.

Figure 30: Show Banks

This figure illustrates a setup where the installed banks were combined with banks downloaded from various web sites. The following list shows that the interface elements in the banks dialog are slightly different from the instruments dialog.

1. **Roots**
2. **Current Bank** (passive display element)
3. **Instruments**
4. **SELECT**
5. **RENAME**
6. **ADD**
7. **DELETE**
8. **SWAP**
9. **Close**

1. Roots. Banks Roots. "Roots" button. Shows a list of directories that can serve as "root" directories.

2. current bank. Current Bank. Simply indicates the current bank via color-highlighting. Note that one can left-click on a bank in this dialog to make it the current bank. This setting is saved across *Yoshimi* restarts.

3. Instruments. Banks Instruments. Brings up a banks dialog that shows the instruments in the current bank.

4. SELECT. Banks SELECT. When this button is selected, then clicking on a bank brings up a very similar dialog that shows the instruments in that bank, laid out with the numbers that are prepended to the filename of each instrument in that bank. Although we don't show a figure for it, note that some banks provide instruments with numbers in the extended program-change range (above 0127) prepended to the file-names.

5. RENAME. Banks RENAME. When this button is selected, then clicking on a bank brings up a small dialog to rename the clicked-on bank.

6. ADD. Banks ADD. Selecting this button and clicking an empty bank entry brings up a small dialog to create a new empty bank name for that entry. If one clicks on an existing bank entry, then a small dialog comes up stating that the bank number selected is already in use.

7. DELETE. Banks DELETE. Selecting this button and clicking an empty bank entry does nothing. Selecting this button and clicking an existing bank entry brings up a small dialog asking one if this bank is really to be deleted.

8. SWAP. Banks SWAP. Selecting this button, then selecting one bank, and then another, swaps the numbering and postion of the selected banks.

3.2.6 Menu / Instrument / Show Root Paths...

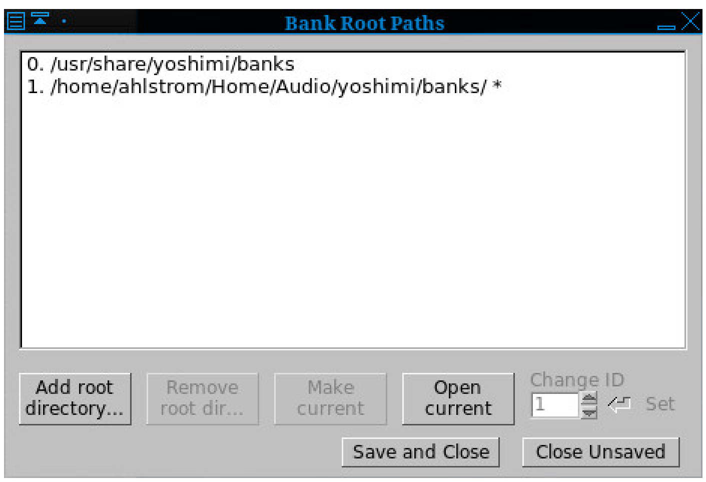

Figure 31: Show Root Paths

1. Add root directory.... Show Root Paths Add Root Directory. To add a bank root path:

Yoshimi (as installed by Debian Linux) provides a default bank at `/usr/share/yoshimi/banks`. To add one's own directory, navigate to "Yoshimi / Instrument / Show Root Paths ...". Then click on "Add root directory...".

Once selected, one will see that `/usr/share/yoshimi/banks` is marked with an asterisk. One can select the new root directory, and make it current by clicking the "Make current" button.

Then the Banks dialog will show all the banks in that directory, one bank per subdirectory (each subdirectory "is" a bank). The current bank will be shown, with all of the instruments it contains. All of the files with filenames starting with 4-digit numbers will be shown in the slot corresponding number. Those without numbers will start with numbers at 129 or above ("extended program change"). One should give them numbers by renaming them outside of *Yoshimi*, then reloading the bank.

But note that MIDI CC can be set to access voices from 129 to 160.

Figure 32: Add Root Directory

2. Remove root directory.... Show Root Paths Remove Root Directory. If a path is selected, then this button is active, and can be used to delete the selected path from the "root paths" list.

3. Make current. Show Root Paths Make Current. This button marks the currently-selected path as the "current root" path.

4. Open current. Show Root Paths Open Current. This button opens the current root path.

5. Change ID. Show Root Paths Change ID.

Values: `0* to 127`

We need to know more about how this ID can be used. Is it a way to make the path selectable via an extended MIDI control, or some other automation method?

6. Save and Close. Show Root Paths Save and Close.

7. Close Unsaved Root Paths. Show Root Paths Close Unsaved.

Bug: Doesn't seem to work! Make a change, and click this button, then reopen the dialog, and the change is preserved.

3.3 Menu / Parameters

Yoshimi stamps its parameter XML files with its own major and minor version numbers so it is possible to tell which version created the files, or whether they were created by *ZynAddSubFX*.

Figure 33: Yoshimi Menu, Parameters

1. **Recent**
2. **Open**
3. **Save**
4. **Clear**

3.3.1 Menu / Parameters / Recent

This menu entry provides access to parameter settings files that have been recently used.

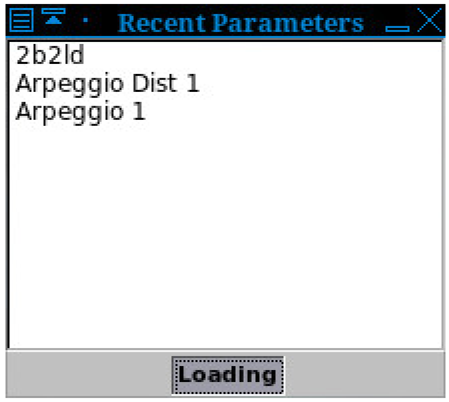

Figure 34: Yoshimi Menu, Recent Parameters

Selecting one of the items in this list causes it to be loaded. All the settings, including effects and instruments, are loaded.

3.3.2 Menu / Parameters / Open

Opens a standard *Yoshimi* dialog for selecting a `*.xmz` file. It is similar to figure 23 ("Open Instrument Dialog") on page 35, as can be seen in the next figure.

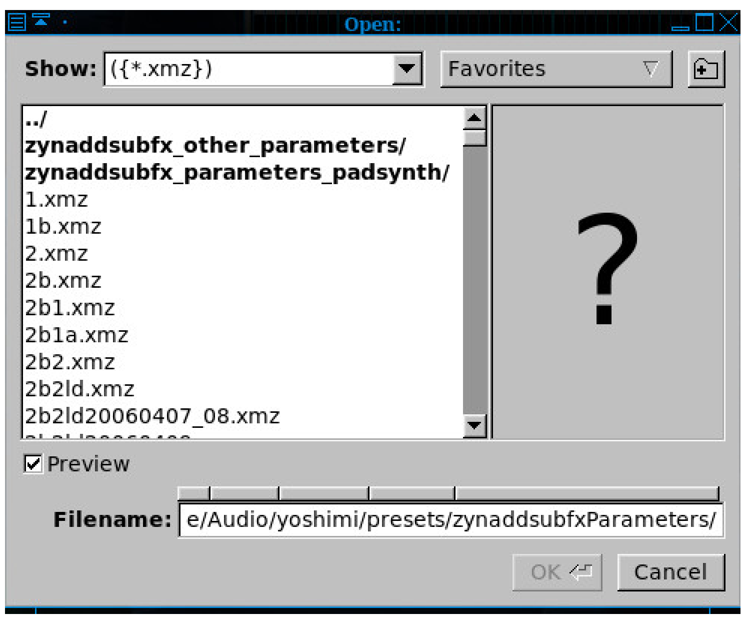

Figure 35: Yoshimi Menu, Open Parameters

3.3.3 Menu / Parameters / Save

This menu entry provides a way to save parameter settings in a file.

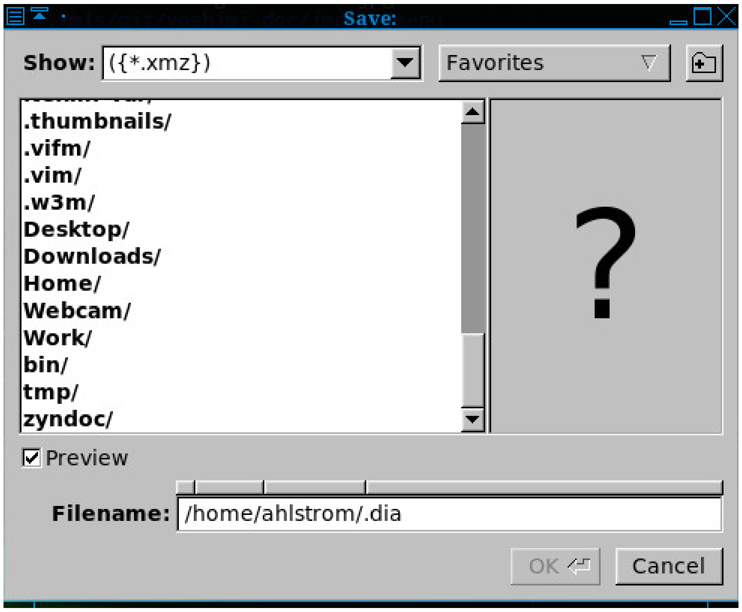

Figure 36: Yoshimi Menu, Save Parameters

TODO: What is the full extent of parameters saved?

If nothing has changed, then the following dialog is shown.

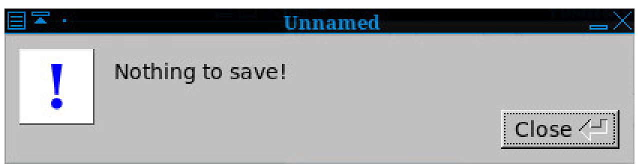

Figure 37: Yoshimi Menu, Nothing to Save

3.3.4 Menu / Parameters / Clear

Using this button brings up the following dialog. Once clicked, *Yoshimi* seems to revert to its default "Simple Sound" setup.

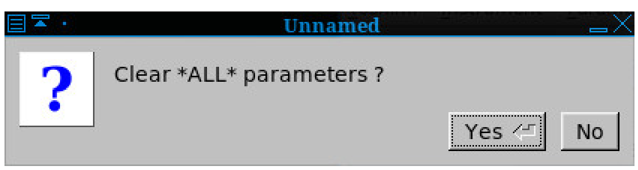

Figure 38: Yoshimi Menu, Clear Parameters

TODO: What is the full extent of parameters cleared?

3.4 Menu / Scales

Yoshimi is a microtonal synthesizer, and is capable of a wide range of microtonal scales.

At present, we're not too experienced with this feature.

Figure 39: Yoshimi Menu, Scales

1. **Load Scale Settings...**
2. **Save Scale Settings...**
3. **Show Scale Settings...**

3.4.1 Menu / Scales / Load

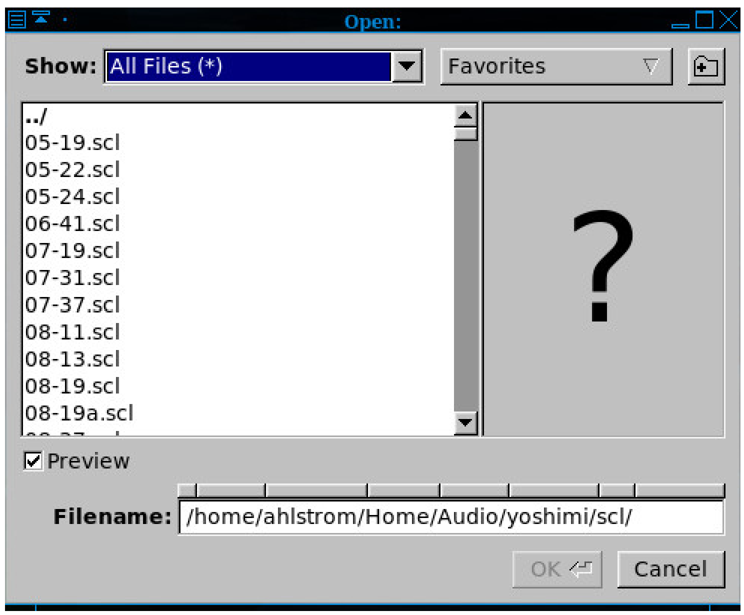

Figure 40: Yoshimi Menu, Open Scales

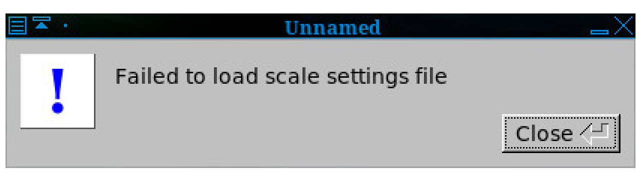

Figure 41: Yoshimi Menu, Failed to Load Scales

3.4.2 Menu / Scales / Save

This dialog opens a stock file-dialog to allow the saving of `*.xsz` files.

3.4.3 Menu / Scales / Show

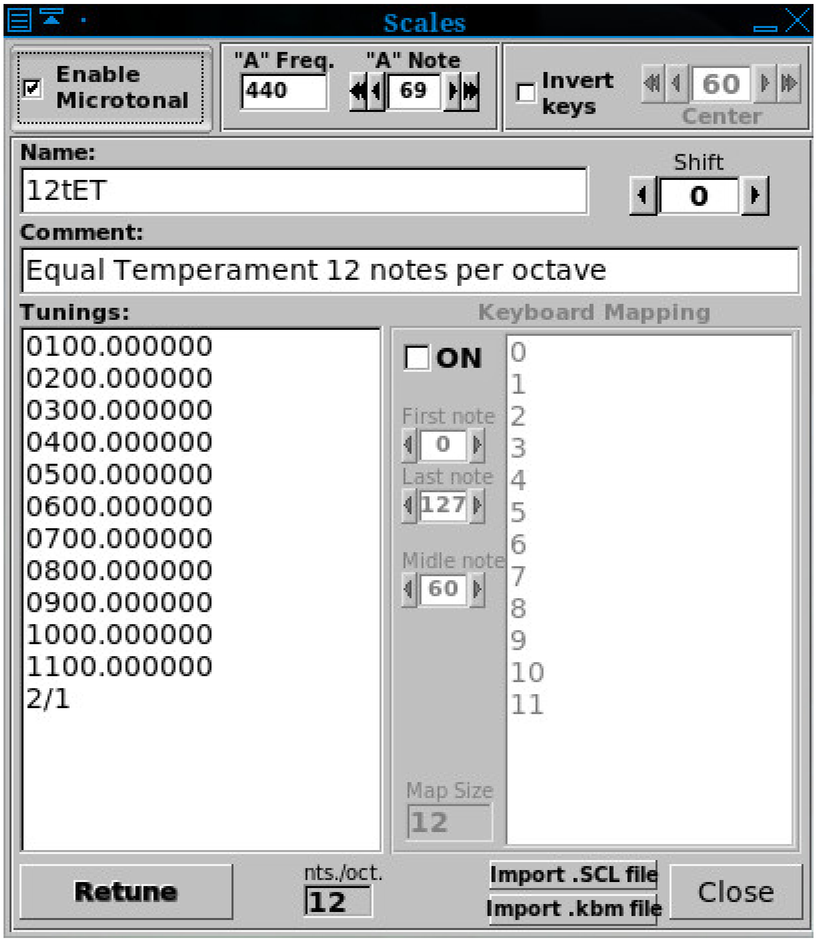

Figure 42: Yoshimi Menu, Scales Settings

3.4.3.1 Scales Basic Settings

This item controls the micro-tonal capabilities of *Yoshimi* and some other settings related to tuning. The last entry in the tunings list represents one octave. All other notes are deduced from these settings.

1. Microtonal. Enable Microtonal Scales. When disabled, the synthesizer will use equal-temperament, 12 notes per octave. Otherwise, one can input any scale one desires.

Values: `Off*, On`

2. "A" Freq.. Frequency of the "A" Note. Sets the frequency of the "A" key. The standard is 440.0 Hz.

Values: `440*`

3. "A" Note. Sets the MIDI Value of the "A" Note.

Values: `0 to 127, 69*`

4. Invert Keys. Allows the keys to be inverted, so that higher-valued keys play lower notes.

Values: `Off*, On`

5. Center. Center for Inverted Keys. This is the center where the notes frequencies are turned upside-down if **Invert keys** is enabled. If the center is 60, the note 59 will become 61, 58 will become 62, 61 will become 59, and so on.

Values: `0 to 127, 60*`

6. Name. Name of the Mapping. For example, the default mapping is called "12tET".

7. Shift. Key Shift. Shift the scale. If the scale is tuned to A, one can easily tune it to another key.

Values: `-63 to 64, 0*`

8. Comment. Comment for Key Mapping. Provides a comment or a description of the scale. By default, this is "Equal Temperament 12 notes per octave.

9. Tunings. Tunings. Here one can input a scale by entering all the tunings for one octave. One can enter the tunings in two ways:

1. As the number of cents (1200 cents=1 octave) as a float number like "100.0", "123.234"
2. As a proportion like "2/1" which represents one octave, "3/2" a perfect fifth, "5734/6561". "2/1" is equal to "1200.0" cents.

The default is a series of values: `0100.0, 0200.0, ..., 1100.0, 2/1`.

10. Retune. Retune. TODO: What does this button do?

11. nts./oct.. Notes Per Octave.

Values: `12*` (range not yet known)

12. Import .SCL file. Import Scala files. Scala is a powerful application for experimentation with musical tunings (intonation scales, micro-tonal,...etc.). From its home page [12], one can download more than 2800 scales which one can import directly into *Yoshimi*. Note that the zip file *must* be unzipped with the `-aa` ("autoconvert") option.

```
$ unzip -aa scales.zip
```

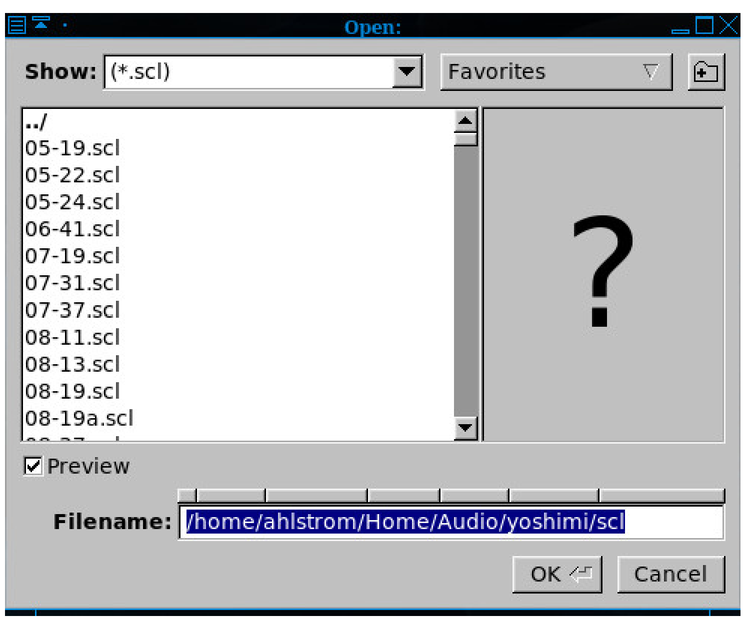

Figure 43: Yoshimi Menu, Scales, Import File

13. Import .scl file. This item is a standard file dialog for reading a `*.scl` file.

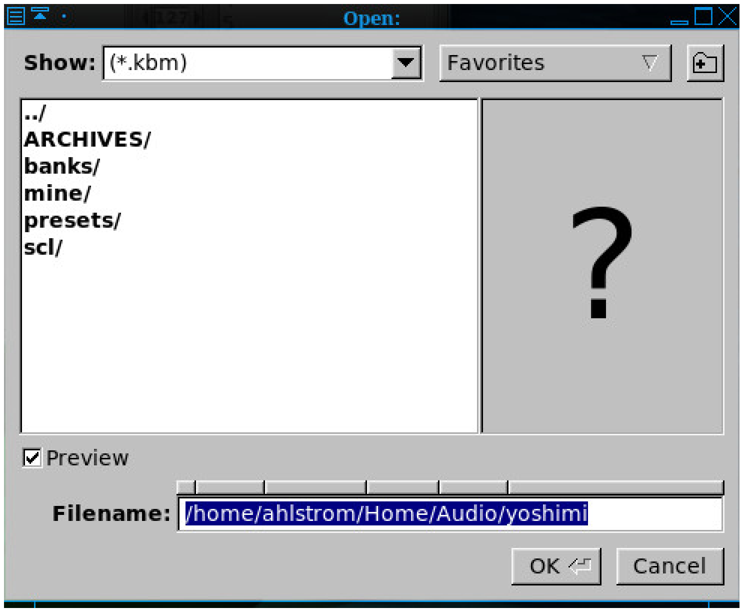

Figure 44: Yoshimi Menu, Scales, Import Keyboard Map

14. Import .kbm file. This item is a standard file dialog for reading a `*.kbm` file.

15. Close, Scales Dialog.

The items related to the **Keyboard Mapping** are discussed separately in the next section.

3.4.3.2 Keyboard Mapping

One can set the MIDI keyboard mapping to scale-degree mapping. This is used if the scale has more or less than 12 notes/octave. One can enable the mapping by pressing the **ON** check-box.

1. **ON**
2. **First Note**
3. **Last Note**
4. **Midle Note**
5. **Map**
6. **Map Size**

1. Scales!ON.

Values: `Off*, On`

2. Scales!First Note. First MIDI Note Number. Keys below this value are ignored.

Values: `0* to 127`

3. Scales!Last Note. Last MIDI Note Number. Keys above this value are ignored.

Values: `0 to 127*`

4. Scales!Middle Note. Middle note where scale-degree 0 is mapped to; the middle note represents the note where the formal octave starts. Note the misspelling of "middle".

Values: `0 to 127*`

5. Scales!Map. Scales map. This is the input field where the mappings are entered. The numbers represent the order (degree) entered on **Tunings Input** field, with the first value being 0. This number must be less than the number of notes per octave (since the values start at 0). If one doesn't want a key to be mapped, one enters an "x" instead of a number.

Values: `0 to 11`

6. Scales!Map Size. Provides the size of the scale-map.

Values: `12`

3.5 Menu / State

Yoshimi state is saved in files with the extension `.state`. These files are also XML files.

TODO: What is the difference between "state" and "parameters"? Which one is all-inclusive? What items are saved in each?

1. **Save**
2. **Load**

As the following figures show, state files are normally stored in the user's `.config/yoshimi/yoshimi.state` file.

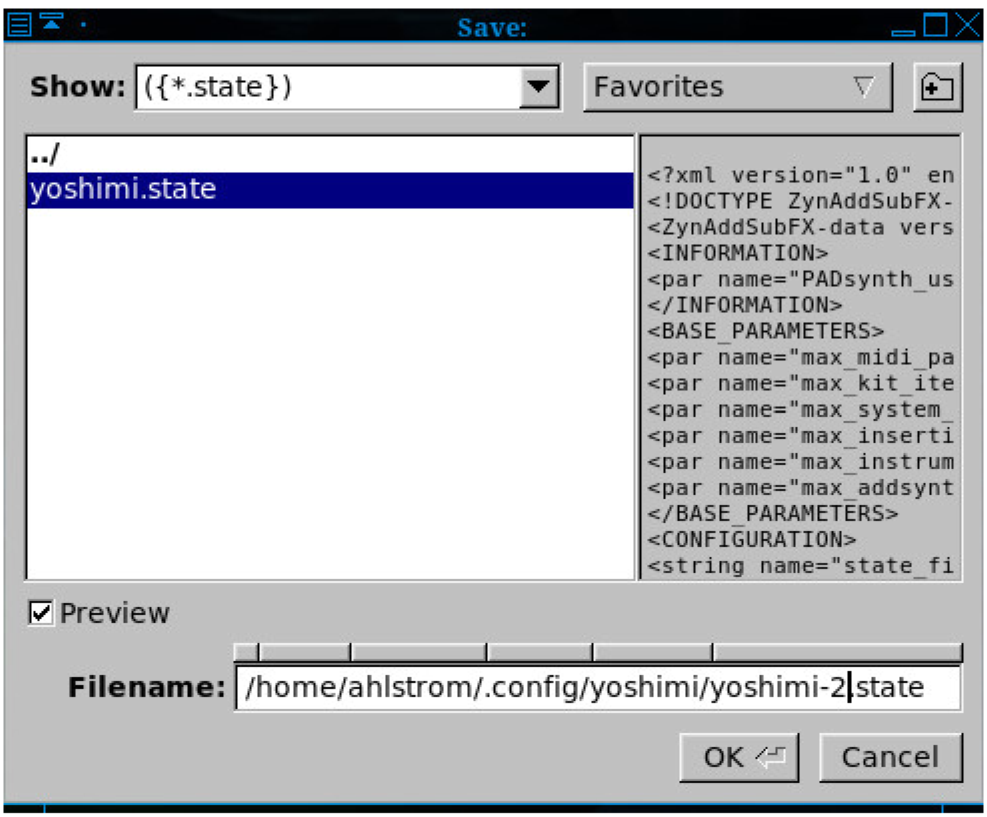

Figure 45: Yoshimi Menu, State Save

This item is a standard *Yoshimi* file dialog.

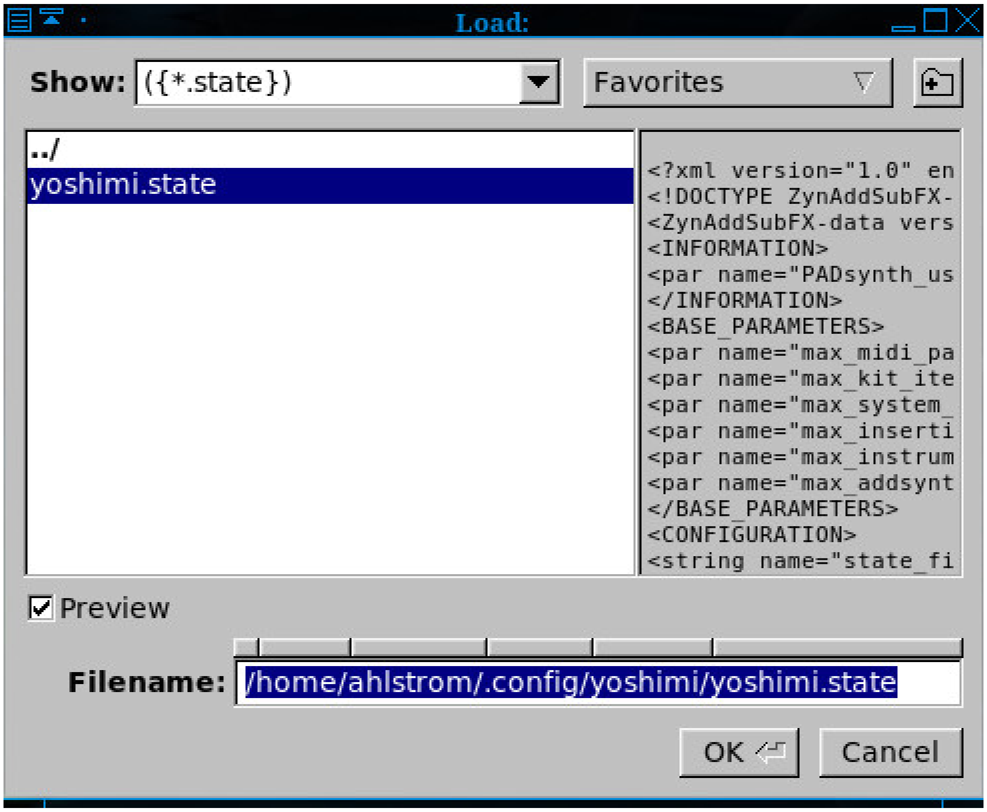

Figure 46: Yoshimi Menu, State Load

This item is a standard *Yoshimi* file dialog.

4 Stock Settings Elements

This section collects all of the setting values one will find for audio parameters in the *Yoshimi* GUI. Sometimes the labels and tool-tips in the application are a bit too brief to understand. One will find their meanings in this section.

This section also covers the sub-panels that provide the settings. By describing these deep details here, we can refer to them when describing how to set up specific sounds in *Yoshimi*.

Much of this material comes from `http://sourceforge.net/zynaddsubfx/Doc` and has been reorganized in minor ways.

4.1 Settings Features

This section notes some minor interface and synthesizer features that may be seen thoughout *Yoshimi*.

4.1.1 Title Bars

The title bars of all editing windows display both the part number and the current name of the instrument one are working on. In the ADDsynth Oscillator Editor, one also sees the voice number of the oscillator one is editing.

4.1.2 Color Coding

A GUI enhancement for *Yoshimi 1.3.5* is color-coded identification of an instrument's use of Add, Sub, and Pad synth engines, no matter where in the instrument's kit they may be. This can be enabled/disabled in the mixer panel. It does slow down *Yoshimi*'s startup, but due to the banks reorganisation (done some time ago) it causes no delay in changing banks/instruments once *Yoshimi* is up and running. Some saved instruments seem to have had their Info section corrupted. Yoshimi can detect this and step over it to find the true status. Also, if one resaves the instrument, not only will the PADsynth status be restored, but ADDsynth and SUBsynth will be included, allowing a faster scan next time.

4.1.3 Knobs

Visual knobs are used for modifying numerical parameters. Horizontal, as well as vertical, mouse movements will adjust the knob. Holding down Ctrl provides finer adjustment. One can also use the mouse scroll wheel to adjust rotary controls.

4.1.4 Automation

In *Yoshimi 1.3.5*, a number of existing, as well as new features have come together to give much greater flexibility (especially for automation) using standard MIDI messages. These are:

1. **NRPNs**
2. **ZynAddSubFX controls**
3. **Independent part control**
4. **16, 32 or 64 parts**
5. **Vector Control**
6. **Direct part stereo audio output**

1. NRPNs. NRPNs can handle individual bytes appearing in either order, and usually the same with the data bytes. Increment and decrement is also supported as graduated values for both data LSB and MSB. Additionally, the ALSA sequencer's 14-bit NRPN blocks are supported.

2. ZynAddSubFx controls. System and Insertion Effect controls are fully supported, with extensions to allow one to set the effect type and (for insertion effects) the destination part number.

3. Part control. Independent part control enables one to change instrument, volume, pan, or indeed any other available control of just that part, without affecting any others that are receiving the same MIDI channel. This can be particularly interesting with multiply layered sounds. There are more extensions planned.

4. 16/32/64 Parts. With 32 and 64 parts, it helps to think of 2 or 4 rows of 16. When one saves a parameter block, the number of parts is also saved, and will be restored when one reloads. By default each *column* has the same MIDI channel number, but these can be independently switched around, and by setting (say) number 17 taken right out of normal access.

In tests, *compiling* for 64 parts compared with 16 parts increased processor load by a very small amount when *Yoshimi* was idling, but this becomes virtually undetectable once one has 8 or more instruments actually generating output. In normal use, selecting the different formats makes no detectable difference, but using the default 16 reduces clutter when one doesn't need the extras.

5. Vector control. Vector control is based on these parts columns, giving one either 2 (X only) or 4 (X + Y) instruments in this channel. Currently the vector CCs one set up can (as inverse pairs) vary any combination of volume, pan, and filter cut-off. More will be added. To keep the processor load reasonable it pays to use fairly simple instruments, but if one has sufficient processing power, it would be theoretically possible to set up all 16 channels with quite independent vector behavior!

6. Direct part audio. Direct part audio is JACK-specific, and allows one to apply further processing to just the defined part's audio output (which can still output to the main L+R if one wants). This setting is saved with parameter blocks. Currently it is only set in the mixer panel window, but it will also eventually come under MIDI direct part control. Again, to reduce unnecessary clutter, part ports are only registered with JACK if they are both enabled, and set for direct output. However, once set they will remain in place for the session to avoid disrupting other applications that may have seen them.

4.2 Filter Settings

This section describes filtering at a high level, in terms of frequency responses and other concepts of filtering. The end of this section covers a user interface used in filter settings. It is a stock-panel re-used in other user-interface elements. See section 4.2.5 ("Filter Parameters User Interface") on page 59if one is in a hurry.

Yoshimi offers several different types of filters, which can be used to shape the spectrum of a signal. The primary parameters that affect the characteristics of the filter are the cutoff, resonance, filter stages, and the filter type.

Filter stages are the number of times that this filter is applied in series. So, if this number is 1, one simply has this filter. If it is two, the sound first passes the filter, and the results then pass the same filter again. In *ZynAddSubFX*, the wetness is applied after all stages were passed.

4.2.1 Filter Type

A filter removes or attenuates frequency elements or tones from a signal. Filtering changes the character of a signal.

The basic analog filters that *Yoshimi* and *ZynAddSubFX* offer are shown in figure 47 ("Basic Filter Types") on page 57, with the center frequency being marked by the red line. The state variable filters should look quite similar.

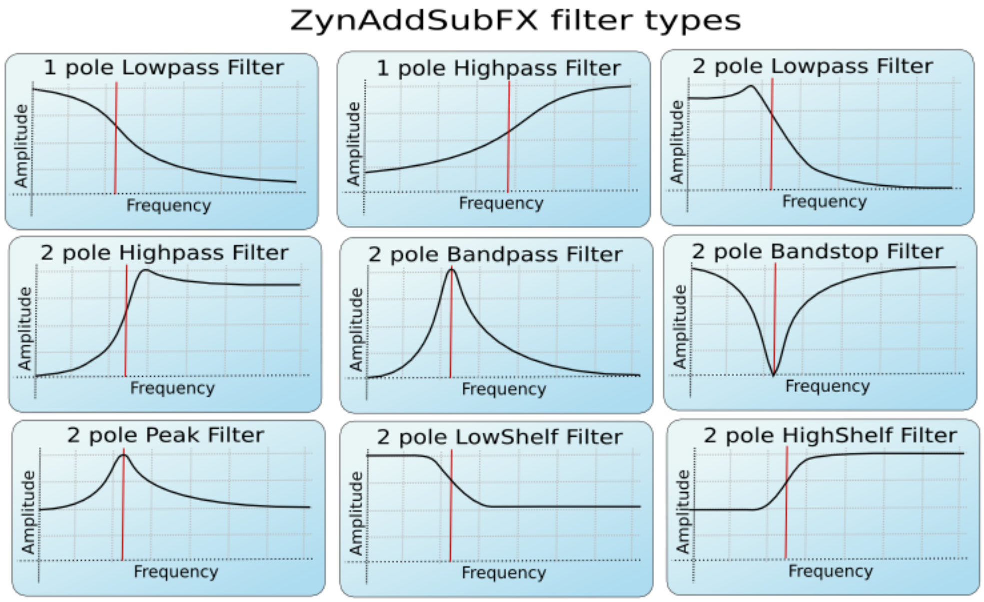

Figure 47: Filter Types, Yoshimi/ZynAddSubFX

1. A **low-pass** filter makes the sound more muffled.
2. A **band-pass** filter makes the sound more tone-like, and sometimes more penetrating, if the total energy in the passband is preserved as the bandwidth decreases.
3. A **high-pass** filter makes the sound seem sharper or more strident.

4.2.2 Filter Cutoff

The filter cutoff value determines which frequency marks the changing point for the filter. In a low pass filter, this value marks the point where higher frequencies begin to be attenuated.

4.2.3 Filter Resonance

The resonance of a filter determines how much excess energy is present at the cutoff frequency. In *Yoshimi* and *ZynAddSubFX*, this is represented by the Q-factor, which is defined to be the cutoff frequency divided by the bandwidth. In other words higher Q values result in a much more narrow resonant spike.

The Q value of a filter affects how concentrated the signals energy is at the cutoff frequency. The result of differing Q values are shown in figure 48 on page 58. For many classical analog sounds, high Q values were used on sweeping filters. A simple high Q low pass filter modulated by a strong envelope is usually sufficient to get a good sound.

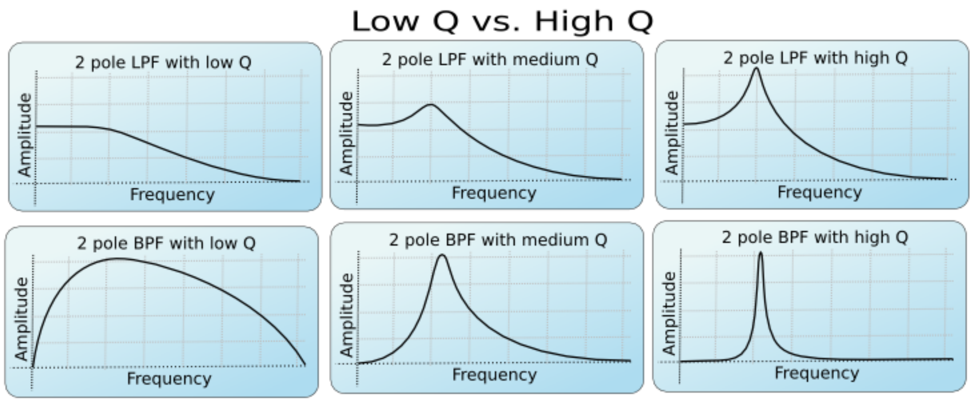

Figure 48: The Effect of the Q Value

4.2.4 Filter Stages

The number of stages in a given filter describes how sharply it is able to make changes in the frequency response. The more stages, the sharper the filter. However, each added stage increases the processor time needed to make the filter calculation.

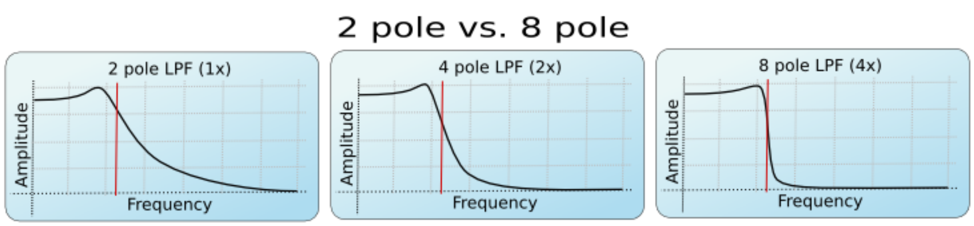

Figure 49: The Effect of the Order of a Filter

The affect of the order of the filter can be seen in the figure above. This is roughly synonymous with the number of stages of the filter. For more complex patches, it is important to realize that the extra sharpness in the filter does not come for free, as it requires many more calculations being performed. This phenomena is the most visible in SUBsynth, where it is easy to need several *hundred* filter stages to produce a given note.

There are different types of filters. The number of poles define what will happen at a given frequency. Mathematically, the filters are functions which have poles that correspond to that frequency. Usually, two poles mean that the function has more "steepness", and that one can set the exact value of the function at the poles by defining the "resonance value". Filters with two poles are also often referred to as *Butterworth Filters*.

For the interested, functions having poles means that we are given a quotient of polynomials. The denominator has degree 1 or 2, depending on the filter having one or two poles. In the file `DSP/AnalogFilter.cpp`, `AnalogFilter :: computefiltercoefs()` sets the coefficients (depending on the filter type), and `AnalogFilter :: singlefilterout()` shows the whole polynomial (in a formula where no quotient is needed).

4.2.5 Filter Parameters User Interface

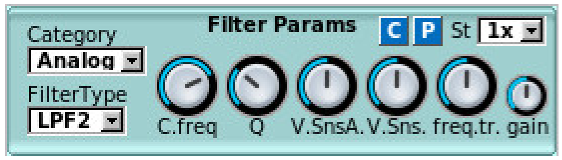

Figure 50: Stock Filter Parameters Sub-Panel

The user interface for filter parameters is a small stock sub-panel that is re-used in a number of larger dialog boxes, as shown in the figure above. Let's describe each item of this sub-panel.

Figure 51: Filter Categories, Dropdown Box

1. **Category**
2. **Filter Type**
3. **C.freq**
4. **Q**
5. **V.SnsA**
6. **freq.tr**
7. **gain**
8. **St**
9. **C**
10. **P**

1. Category. Determines the category of filter to be used. There are three categories of filters (as shown in the dropdown element shown in figure 51 ("Filter Categories Dropdown") on page 59).

1. **Analog** (the default)
2. **Formant**
3. **StVarF**

An **analog** filter is one that approximates a filter that is based on a network of resistors, capacitors, and inductors.

A **formant** filter is a more complex kind of filter that acts a lot like the human vocal tract, allowing for sounds that are a bit like human voices.

A **state variable** ("StVarF") filter is a type of active filter. The frequency of operation and the Q factor can be varied independently. This and the ability to switch between different filter responses make the state-variable filter widely used in analogue synthesizers.

Values: `Analog*, Formant, StVarF`

2. Filter Type. Selects the type of filter to be used, such as high-pass, low-pass, and band-pass. See the dropdown element in figure 52 ("Filter Type Dropdown") on page 60.

Figure 52: Type of Filter Passband, Dropdown Box

Values: `LPF1, HPF1, LPF2*, HPF2, BPF2, NF2, PkF2, LSh2, HSh2`

3. C.freq. Cutoff frequency or center frequency. This items has various definitions in the literature. Usually it refers to the frequency at which the level drops to 3 Db below the maximum level. In various dialogs, this value is the center frequency of the filter or the base position in a vowel's sequence.

Values: `0 to 127, 90*`

4. Q. The level of resonance for the filter. It indicates a measure of the sharpness of a filter. The higher the Q, the sharper the filter. Generally, a higher Q value leads to a louder, more tonal affect for the filter. Note that some filter types might ignore this parameter.

5. V.SnsA. Velocity sensing amount for filter cutoff. Velocity sensing amount of the filter.

TODO.

Values: `0 to 127, 64*`

6. V.Sns. Velocity sensing function of the filter. Set the amplitude of the velocity sensing.

Values: `0 to 127, 64*`

7. freq.tr. Filter Frequency Tracking Amount. When this parameter is positive, higher note frequencies shift the filters cutoff frequency higher. For the filter frequency tracking knob, left is negative, middle is zero, and right is positive.

Values: `0 to 127, 64*`

8. gain. Filter gain. Additional gain/attenuation for a filter. Also described as the filter output gain/damping factor.

Values: `0 to 127, 64*`

9. St. Filter stages. The more filter stages applied to a signal, the stronger (in general) the filtering. It is the number of additional times the filter will be applied (in order to create a very steep roll-off, such as 48 dB/octave). This dropdown element is shown in figure 53 ("Filter Stage Dropdown") on page 60. Obviously, the more stages used, the more calculation-intensive the filter will be. This should also increase the latency (lag) of the filter.

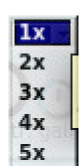

Figure 53: Filter Stage Dropdown

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog.

4.3 LFO Settings

Yoshimi provides LFOs for it amplitude, frequency, and filtering functions. "LFO" means Low Frequency Oscillator. These oscillators are not used to make sounds by themselves, but they change parameters cyclically as a sound plays.

LFOs are, as the name says, oscillators with, compared to the frequency of the sound, low frequency. They often appear in order to control the effect.

4.3.1 LFO Basic Parameters

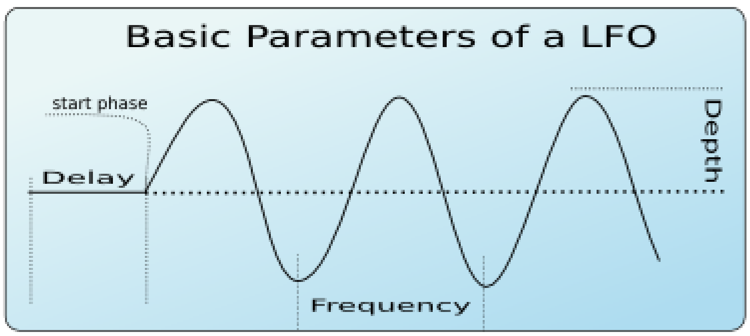

Figure 54: Basic LFO Parameters

1. **Delay**.
2. **Start Phase**.
3. **Frequency**.
4. **Depth**.

The LFOs has some basic parameters (see figure 54 ("Basic LFO Parameters") on page 61.

1. Delay. LFO Delay. This parameter sets how much time takes since the start of the note to start the cycling of the LFO. When the LFO starts, it has a certain position called "start phase".

2. Start Phase. LFO Start Phase. The angular position at which a LFO waveform will start.

3. Frequency. LFO Frequency. How fast the LFO is (i.e. how fast the parameter controlled by the LFO changes.)

4. Depth. LFO Depth. The amplitude of the LFO (i.e. how much the parameter is controlled by the LFO changes.)

4.3.2 LFO Function

Another important additional LFO parameter is the shape or type of the LFO. There are many LFO Types that vary according to the function used to generate the LFO. *Yoshimi* supports the LFO shapes shown in figure 55 ("LFO Functions") on page 62.

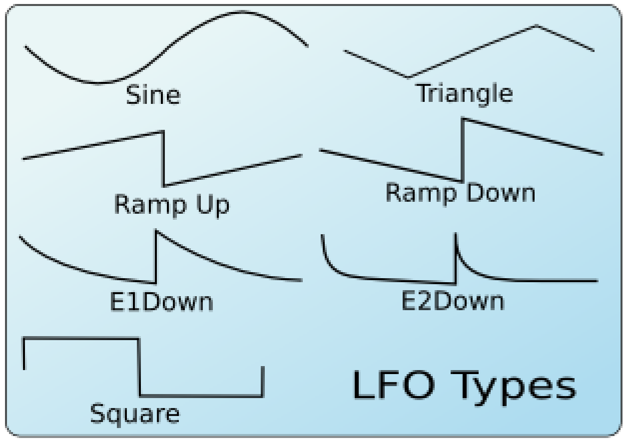

Figure 55: LFO Types, Shapes, or Functions

4.3.3 LFO Randomness

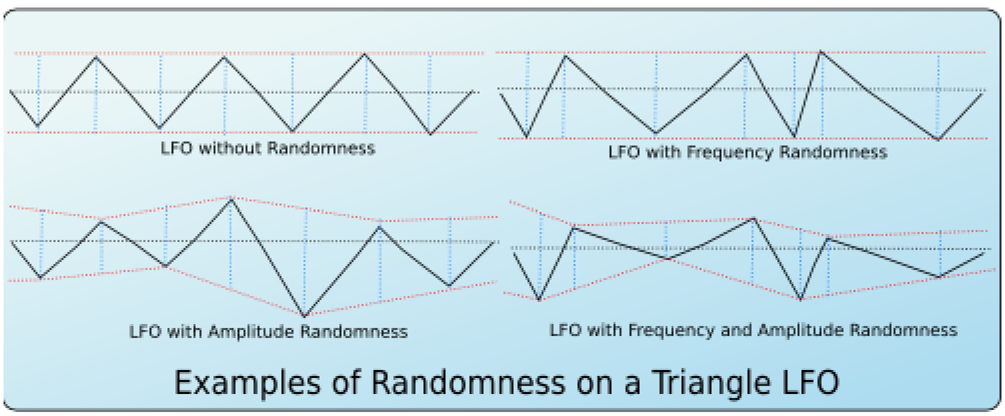

Figure 56: LFO Randomization

Another parameter is the LFO Randomness. It modifies the LFO amplitude or the LFO frequency at random. In *Yoshimi* one can choose how much the LFO frequency or LFO amplitude changes by this parameter. Observe figure 56 ("LFO Randomization") on page 62. It shows some examples of randomness and how it changes the shape of a triangle LFO.

4.3.4 LFO, More Settings

Other settings are available as well.

Continous mode: If this mode is used, the LFO will not start from "zero" on each new note, but it will be continuous. This is very useful if one applies on filters to make interesting sweeps.

Stretch: It controls how much the LFO frequency changes according to the notes frequency. It can vary from negative stretch (the LFO frequency is decreased on higher notes) to zero (the LFO frequency will be the same on all notes) to positive stretch (the LFO frequency will be increased on higher notes).

4.3.5 LFO User Interface Panels

Figure 57: Amplitude LFO Sub-Panel

In *Yoshimi*, LFO parameters are available for amplitude, filters, and frequency. They all have essentially the same interface elements. Note that figure 57 ("Amplitude LFO Sub-Panel") on page 63shows an example of an LFO stock sub-panel.

These parameters are:

1. **Freq**
2. **Depth**
3. **Start**
4. **Delay**
5. **A.R**
6. **F.R**
7. **C** or **C.**
8. **Str**
9. **Type**
10. **C** (copy)
11. **P** (paste)

1. Freq. LFO Frequency. This parameter varies from 0 to 1. TODO: We still need to figure out what that scale means, however.

Values: `0 to 1, 0.63*`

2. Depth. LFO Depth. Also called "LFO Amount".

Values: `0* to 127`

3. Start. LFO Start Phase. If this knob is at the lowest value, the LFO Start Phase will be random.

Values: `0 = random to 127, 64*`

4. Delay. LFO Delay.

Values: `0* to 127`

5. A.R. LFO Amplitude Randomness.

Values: `0* to 127`

6. F.R. LFO Frequency Randomness.

Values: `0* to 127`

7. C. LFO Continous Mode.

Values: `Off*, On`

8. Str. LFO Stretch. See the image in figure 57 ("Amplitude LFO Sub-Panel") on page 63. It shows the LFO stretch is set to zero, though the tooltip would show it to be 64.

Values: `0 to 127, 64*`

9. Type. LFO Function.

Values: `SINE*, TRI, SQR, R.up, R.dn, E1dn, E2dn`

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog.

Figure 58: LFO Function Type Dropdown Element

10. Type. LFO Type (or Shape, or Function). The various shapes of LFO functions are shown in figure 55 ("LFO Functions") on page 62. The values that can be selected are shown in figure 58 ("LFO Type Dropdown") on page 64.

Values: `SINE*, TRI, SQR, R.up, R.dn, E1dn, E2dn`

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog.

For reference, figure 59 ("Filter LFO Sub-Panel") on page 64shows the LFO sub-panel for a filter, and figure 61 ("Frequency LFO Sub-Panel") on page 65shows the LFO sub-panel for frequency.

4.3.6 Filter LFO Sub-panel

Figure 59: Filter LFO Sub-Panel

1. **Enable** (present on some versions of this sub-panel).
2. **Freq.**
3. **Depth**
4. **Start**
5. **Delay**
6. **Str.**
7. **C.**
8. **A.R.**
9. **F.R.**
10. **Type**
11. **C**
12. **P**

1. Enable. Enable the panel. (Present on some versions of this sub-panel).

2. Freq.. LFO Frequency.

Values: `0 to 1, 0.64*`

3. Depth. LFO Amount.

Values: `0* to 127`

4. Start. LFO Startphase (leftmost is random).

Values: `0 to 127, 64*`

5. Delay. LFO Delay.

Values: `0* to 127`

6. Str.. LFO Stretch.

Values: `0 to 127, 64*`

7. C.. Continuous LFO.

Values: `Off*, On`

8. A.R.. LFO Amplitude Randomness.

Values: `0* to 127`

9. F.R.. LFO Frequency Randomness.

Values: `0* to 127`

10. Type. LFO Type.

Figure 60: LFO Function Type Dropdown

Values: `SINE*, TRI, SQR, R.up, R.dn, E1dn, E2dn`

11. C. Copy to Clipboard/Preset.

12. P. Paste from Clipboard/Preset.

4.3.7 Frequency LFO Sub-panel

Figure 61: Frequency LFO Sub-Panel

This panel is basically identical to the Filter LFO panel described in the previous section.

4.4 Envelope Settings

Envelopes control how the amplitude, the frequency, or the filter changes over time. The general envelope generator has four sections:

1. **Attack**. The attack is the initial envelope response. It begins when the key for the note is first held down (at Note On). The volume starts at 0, and rises fast or slowly until a peak value. In *Yoshimi*, the attack is always linear.
2. **Decay** When the attack is at its highest value, it immediately begins to decay to the sustain value. The decay can be fast or slow. The attack and decay together can be used to produce something like horn blips, for example.
3. **Sustain** This is the level at which the parameter stays while the key is held down, i.e. until a Note Off occurs.
4. **Release** When the key is released, the sound decays, either fast or slowly, until it is off (the volume is 0).

The ADSR envelope generally controls the amplitude of the sound. In *Yoshimi*, amplitude envelopes can be linear or logarithmic.

Together, these values are called "ADSR".

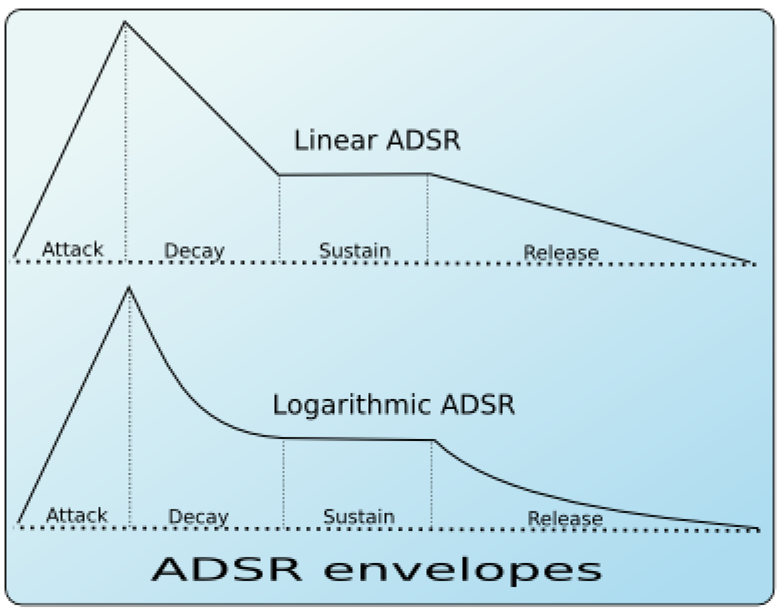

Figure 62: ADSR Envelope (Amplitude)

Figure 62 on page 66 shows a depiction of an ADSR envelope. The ADSR is mostly applied to amplitude envelopes.

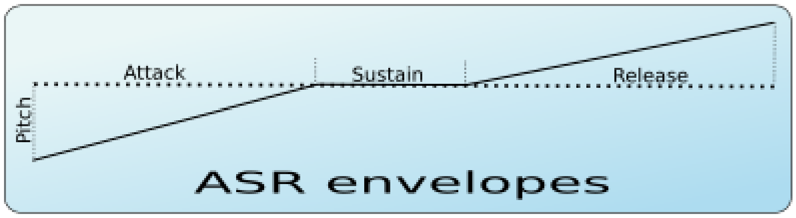

Figure 63: ASR Envelope, Frequency

Frequency envelopes control the frequency (more exactly, the pitch) of the oscillators. The following image depicts the stages of these envelopes.

For frequency envelopes, a simpler form of envelope is used. This envelope is an ASR envelope, shown in figure 63 ("ASR Envelope, Frequency") on page 67. The dotted line represents the real pitch of the sound without the envelope. The frequency envelopes are divided into 3 stages:

1. Attack. It begins at the Note On. The frequency starts from a certain value and glides to the real frequency of the note.
2. Sustain. The frequency stays the same during the sustain period.
3. Release. This stage begins on Note Off and glides the frequency of the note to a certain value.

4.4.1 Amplitude Envelope Sub-Panel

Figure 64: Amplitude Envelope Sub-Panel

1. **A.dt**
2. **D.dt**
3. **S.val**
4. **R.dt**
5. **Str**
6. **L**
7. **frcR**
8. **C**
9. **P**
10. **E**

1. A.dt. Attack duration, attack time. TODO: determine the units of time at play for ADSR durations.

Values: `0* to 127`

2. D.dt. Decay duration, decay time.

Values: `0 to 127, 44*`

3. S.val. Sustain value. This is the (relative?) level at which the envelope will settle while the note is held down. The only stage that always remains defined is the Sustain, where the envelopes freezes until a Note Off event.

Values: `0 to 127*`

4. R.dt. Release time.

Values: `0 to 127, 25*`

5. Str. Stretch. How the envelope is stretched according the note. Envelope Stretch means that, on lower notes, the envelope will be longer. On the higher notes the envelopes are shorter than lower notes. In the leftmost value, the stretch is zero. The rightmost use a stretch of 200%; this means that the envelope is stretched about 4 times peroctave.

Values: `0 to 127, 64*`

6. L. Linear envelope. If this option is set, the envelope is linear, otherwise, it will be logarithmic.

Values: `Off*, On`

7. frcR. Forced release. This means that if this option is turned on, the release will go to the final value, even if the sustain stage is not reached. Usually, this must be set.

Values: `Off, On*`

If this option is turned on, the release will go to the final value, even if the sustain level is not reached.

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog.

8. C. opy to Clipboard/Preset.

9. P. aste from Clipboard/Preset.

10. E. mplitude Envelope Window.

4.4.2 Envelope Settings

Amplitude Envelope Window.

Figure 65: Amplitude/Filter/Frequency Envelope Editor

1. **Graph Window**
2. **FreeMode**
3. **C**
4. **P**
5. **Close**

11. FreeMode. Freemode Enable.

Values: `Off*, On`

4.4.3 Freemode Envelope Settings

The envelopes are parts that control a parameter (frequencies) of a sound.

For all envelopes, there is a mode that allows the user to set an arbitrary number of stages and control points. This mode is called Freemode. The only stage that always remains defined is the Sustain, where the envelopes freezes until a Note Off event. The Freemode envelope editor has a separate window to set the parameters and controls.

The main concept of the freemode editor window is the *control point*. One can move the points using the mouse. In the right on the window, it shows the total duration of the envelope. If the mouse button is pressed on a control point, it will be shown the duration of the stage where the point is.

figure 66 ("Amplitude/Filter/Frequency Envelope Freemode Editor") on page 69shows an example of the stock freemode envelope editor, with freemode enabled.

Figure 66: Amplitude/Filter/Frequency Envelope Freemode Editor

All of the envelope editors have some common controls.

1. **Graph Window**
2. **Add point**
3. **E**
4. **FreeMode**
5. **Add point**
6. **Delete point**
7. **Sust**
8. **Stretch**
9. **L**
10. **frcR**
11. **Close**
12. **C**
13. **P**

1. E. Editor. Graph Window. Shows a window with the real envelope shape and the option to convert to freemode to edit it. The envelope editor shows a window in which one can view and modify the detailed envelope shape, or convert it to "freemode" to edit it almost without restriction. By default, only the *Freemode* button/checkbox is visible.

If an envelope has the FreeMode mode enabled, it allows one to edit the graph of the envelope directly. Select a point from the graph and move it. Notice that *only the line before the currently edited point of the envelope* changes its duration.

If a point is being dragged, the text on the right shows the duration of the line before it. Otherwise, the text shows the total duration of the envelope.

If the envelope doesn't have the FreeMode mode enabled, it doesn't allow one to move the points; the envelope window is then useful only to see what happens if one changes the ADSR settings.

2. FreeMode. FreeMode. Provides a mode where completely arbitrary envelopes may be drawn.

Values: `Off*, On`

Actually, the envelopes aren't completely arbitrary, as the sustain section is always flat, and its duration corresponds with the duration the note is held down. When this mode is enabled, the rest of the controls shown in figure 66 ("Amplitude/Filter/Frequency Envelope Freemode Editor") on page 69appear, and are described in the following paragraphs.

3. Add point. Add point. Provides a way to add a data point to the Freemode envelope. It adds the point after the currently-selected point. One can select a point by clicking on it.

4. Delete point. Delete point. Provides a way to delete the current data point from the Freemode envelope.

5. Sust. Sustain point. Sets the sustain point. The sustain point is shown using the yellow line. If the point is at 0, then sustain is disabled.

Values: `0, 1, 2*`

1. 0 means that sustain is disabled, and the envelope immediately starts dying, even if the note is held.
2. 1 seems to mean the sustain curve follows its course while the note is held.
3. 2 seems to mean that extra sustain kicks in after the note is released.

It is difficult to determine the difference between 1 and 2.

6. Stretch. Envelope Stretch. How the envelope is stretched according the note. On the higher notes the envelopes are shorter than lower notes. At the leftmost value, the stretch is zero. The rightmost sets a stretch of 200%; this means that the envelope is stretched about four times/octave.

7. L. Envelope Linear. This setting is only available in the amplitude envelope. If enabled, the envelope is linear. If not enabled, the envelope is logarithmic (dB).

Values: `Off*, On`

8. frcR. Forced Release. This means that if this option is turned on, the release will go to the final value, even if the sustain stage is not reached. Usually, this must be set. When the key is released, the position of the envelope jumps directly to the point after the release point. If the release is disabled, the envelope position jumps to the last point on release.

Values: `Off*, On`

9. Close. Close Dialog.

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog, as well as a button to bring up the editor window.

4.4.4 Envelope Settings, Frequency

These envelopes controls the frequency (more exactly, the pitch) of the oscillators. Observe figure 63 ("ASR Envelope, Frequency") on page 67. It depicts the stages of these envelopes. The dotted line represents the real pitch of the sound without the envelope.

The frequency envelopes are divided into 3 stages: attack (see 1.); sustain (see 3.); and release (see 4.).

One question to answer is: can the attack and release go in the opposite directions, or do the knob ranges prohibit this?

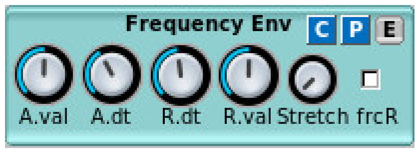

Figure 67: Frequency Envelope Sub-Panel

1. **Enable** (present on some versions of this sub-panel).
2. **A.value** or **A.val**
3. **A.dt**
4. **R.dt**
5. **R.val** (present on some versions of this sub-panel).
6. **Stretch**
7. **frcR**
8. **C**
9. **P**
10. **E**

For Frequency Envelopes the interface has the following parameters:

1. Enable. Enable the panel. (Present on some versions of this sub-panel).

2. A.val. Attack value. We need to figure out what this means.

Values: `0 to 127, 64*`

3. A.dt. Attack duration. Attack time.

Values: `0 to 127, 40*`

4. R.dt. Release time.

Values: `0 to 127, 60*`

5. R.val. Release Value. Actually present only on the Frequency Env sub-panel.

Values: `0 to 127, 64*`

6. Stretch. Envelope Stretch. Envelope Stretch (on lower notes make the envelope longer).

Values: `0 to 127, 64*`

7. frcR. Forced release.

Values: `Off, On*`

If this option is turned on, the release will go to the final value, even if the sustain level is not reached.

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog, as well as a button to bring up the editor window.

4.4.5 Envelope Settings for Filter

This envelope controls the cutoff frequency of the filters. The filter envelopes are divided into 4 stages:

1. Attack. It begins at the Note On. The cutoff frequency starts from a certain value and glides to another value.
2. Decay. The cutoff frequency continues to glide to the real cutoff frequency value of the filter (dotted line).
3. Sustain. The cutoff frequency stays the same during the sustain period (dotted line).
4. Release. This stage begins on Note Off and glides the filter cutoff frequency of the note to a certain value.

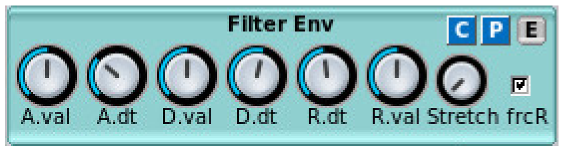

Figure 68: Filter Envelope Sub-Panel

1. **A.value**
2. **A.dt**
3. **D.val**
4. **D.dt**
5. **R.dt**
6. **Stretch**
7. **frcR**
8. **L**

Filter Envelopes has the following parameters:

1. A.value. Attack Value. Starting Value. We need to figure out what this means.

Values: `0 to 127, 64*`

2. A.dt. Attack Duration. Attack Time.

Values: `0 to 127, 40*`

3. D.val. Decay Value.

Values: `0 to 127, 64*`

4. D.dt. Decay Duration. Decay Time.

Values: `0 to 127, 70*`

5. R.dt. Release time.

Values: `0 to 127, 60*`

6. Stretch. Stretch. Envelope Stretch (on lower notes make the envelope longer).

Values: `0 to 127, 64*`

7. frcR. Forced Release.

Values: `Off, On*`

If this option is turned on, the release will go to the final value, even if the sustain level is not reached.

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog, as well as a button that bring up the editor window.

Addition picture and GUI items for ADDsynth version?

Figure: bottom-panel/instrument-edit/ADD/ADDsynth-filter-envelope.jpg

8. L. If this option is set, the envelope is linear, otherwise, it will be logarithmic.

Values: `Off*, On`

4.4.6 Formant Filter Settings

This window allows one to change most of the parameters of the formant filter.

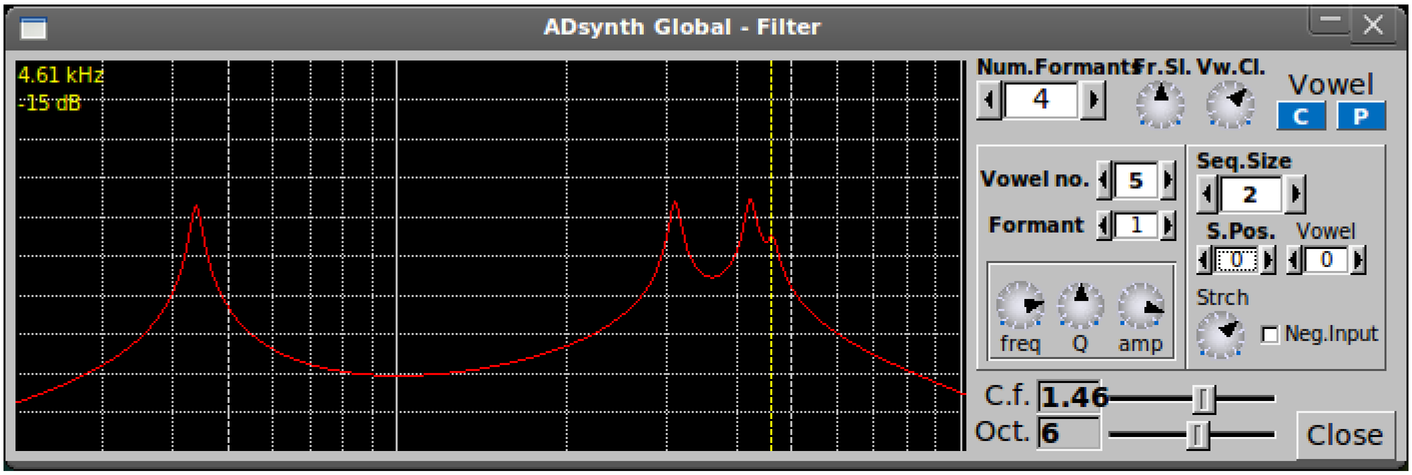

Figure 69: Formant Filter Editor Dialog

1. **Category**
2. **Num.Formants**
3. **Fr.Sl.**
4. **Vw.Cl.**
5. **C.f.**
6. **Oct.**
7. **Vowel no**
8. **Formant**
9. **freq**
10. **Q**
11. **amp**
12. **Seq Size**
13. **S.Pos**
14. **Vowel**
15. **Strch**
16. **Neg Input**

4.4.6.1 Formant Parameters

9. Num.Formants. Number of Formants Used.

Values: `0 to xxx?`

10. Fr.Sl.. Formant Slowness.

Values: `0 to xxx?`

This parameters prevents too-fast morphing between vowels.

11. Vw.Cl.. Vowel "Clearness".

Values: `0 to xxx?`

Sets how much the vowels are kept "clear", that is, how much the "mixed" vowels are avoided.

12. C.f.. Center Frequency.

Values: `0 to xxx?`

The center frequency of the graph.

13. Oct.. Number of Octaves.

Values: `0 to xxx?`

The number of octaves in the graph.

4.4.6.2 Formant Vowel Parameters

14. Vowel no. The number of the current vowel. This number means what?

Values: `0 to xxx?`

15. Formant. The current formant.

Values: `0 to xxx?`

16. freq. The frequency of the current formant.

Values: `0 to xxx?`

17. Q. The Q (resonance depth or bandwidth) of the current formant.

Values: `0 to xxx?`

18. amp. Amplitude of the current formant.

Values: `0 to xxx?`

4.4.6.3 Formant Sequence Parameters

The sequence represents what vowel is selected to sound according to the input from the filter envelopes and LFO's.

19. Seq Size. Sequence Size. The number of vowels in the sequence.

Values: `0 to xxx?`

20. S.Pos. Sequence Position. The current position of the sequence.

Values: `0 to xxx?`

21. Vowel. The vowel from the current position.

Values: `0 to xxx?`

22. Strtch. How the sequence is stretched. This number means what?

Values: `0 to xxx?`

23. Neg Input. Negative Input. If enabled, the sequence is reversed.

Values: `0 to xxx?`

4.4.7 Controller Settings

TODO.

4.5 Clipboard Presets

In many of the settings panels, there are buttons labelled **C**, **P**, and **E**, **E** is the editor window, discussed in section 4.4.3 **C** and **P** are the clipboard/present copy and paste dialogs, respectively.

4.5.1 Clipboard/Preset Copy

Note that figure 70 ("Copy to Clipboard") on page 75shows an example of the copying dialog for the clipboard.

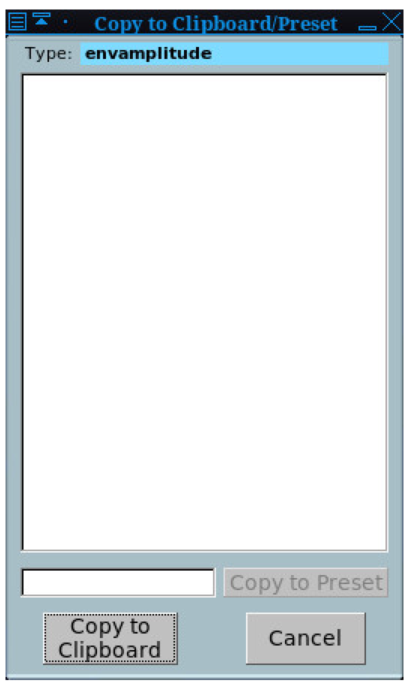

Figure 70: Copy to Clipboard/Presets

1. Type. Clipboard type for copying. This field indicates the context (e.g h. "envamplitude") or name of the clipboard to which the data will be copied.

2. Clipboard list. Clipboard list.

3. Copy to Preset. Clipboard to preset. Provides a way to specify the preset to which this data should be copied.

To save to a preset, type the desired name of the setting. This entry will enable this button. When the button is pressed, the preset will be saved to the default directory, as specified per paragraph 3.1.4.2 ("Menu / Yoshimi / Settings / Preset dirs") on page 28. (Be sure to set up a default directory where ordinary users have write permissions!) The file-name of the of the preset will be a hidden file such as

```
.ADnoteParameters.xpz
```

The main part of this name is shown near the top of the preset dialog, as a cue. There is no way in *Yoshimi* to change this name. One must do it using file system commands. However, changing the name won't do much good. Only the name shown above will ever be visible in *Yoshimi*. (And yet it ships with a large number of non-hidden `.xpz` files!)

One other question about the preset file, not yet answered, is if all the presets ever copied remain in that file, so that the user doesn't have to keep making copies of the hidden preset file.

4. Copy to Clipboard. Preset to Clipboard.

4.5.2 Clipboard/Preset Paste

Observe figure 71 ("Paste from Clipboard") on page 76. It shows an example of the pasting dialog for the clipboard.

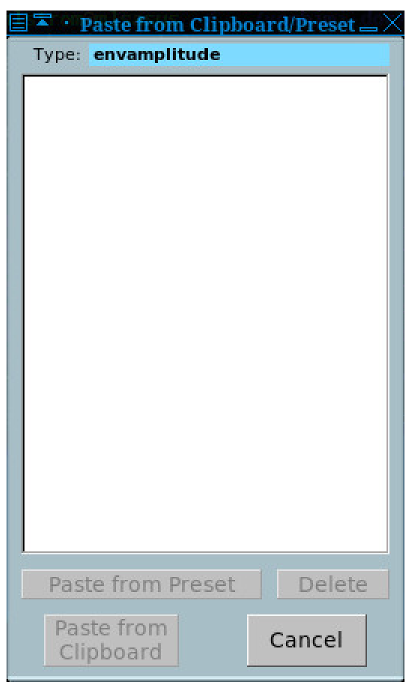

Figure 71: Paste from Clipboard/Presets

1. **Add point**
2. **Type**
3. **Clipboard list**
4. **Paste from Preset**

5. **Paste from Clipboard**

1. Type. Clipboard type for pasting. This field indicates the context (e.g h. "envamplitude") or name of the clipboard to which the data will be copied.

2. Clipboard list. Clipboard list.

3. Paste from Preset. Paste from preset. Provides a way to specify the preset to which this data should be copied.

4. Paste from Clipboard. Clipboard to preset.

5 Top Panel

The *Yoshimi* top panel provides quick access to some major features of the application. The top panel is shown in figure 2 ("Yoshimi Main Screen, 1.3.5") on page 11.

Here are the major elements of the top panel.

1. **Stop!**
2. **Panel**
3. **VirKbd**
4. **Key Shift**
5. **Detune**
6. **Reset Detune**
7. **Volume**

1. Stop!. Stop! This button causes *Yoshimi* to "Cease all sound immediately!"

2. Panel. This button brings up a panel that shows a "mixer" view of all of the parts that have been created in the current state of *Yoshimi*.

For the details of this panel, see section 5.1 ("Mixer Panel Window") on page 77.

3. VirKbd. This button brings up the virtual keyboard, which is a way to enter MIDI information without a real MIDI keyboard. It also provides a way to use the computer keyboard for faster playing. See section 5.2 ("Virtual Keyboard") on page 79.

4. Key Shift. Master Key Shift. This is the key-shift (transpose) that applies to all parts.

Values: `-12 to 12, 0*`

5. Detune. Detune. Provides a global fine detune functionality. The fine detune mapping to the knob values shown below is -64 to 63 cents.

Values: `0 to 127, 64*` (float)

6. Reset, Detune. Reset detune. Resets the overall detuning functionality of *Yoshimi* off. Resets the global fine detune to 0.

7. Volume. Volume, Master Volume. Controls the overall volume of all sounds generated by *Yoshimi*.

5.1 Mixer Panel Window

The *Panel* button opens the mixer panel window. The mixer panel window provides a global view of the most important adjustable parameters of all of the defined parts. There are two views, a 2x8 view and a 2x16 view. Figure 72 on page 78 shows the 2x8 view.

The Panel Window allows one to edit some important part parameters (instrument/volume/panning/etc..) and it acts like a mixer. Also, this window shows VU-meters for each part. To change an instrument, click on the Edit button for that instrument. Sometimes, if one edits the parameters of the part in the main window, one needs to refresh the panel by clicking the Refresh button.

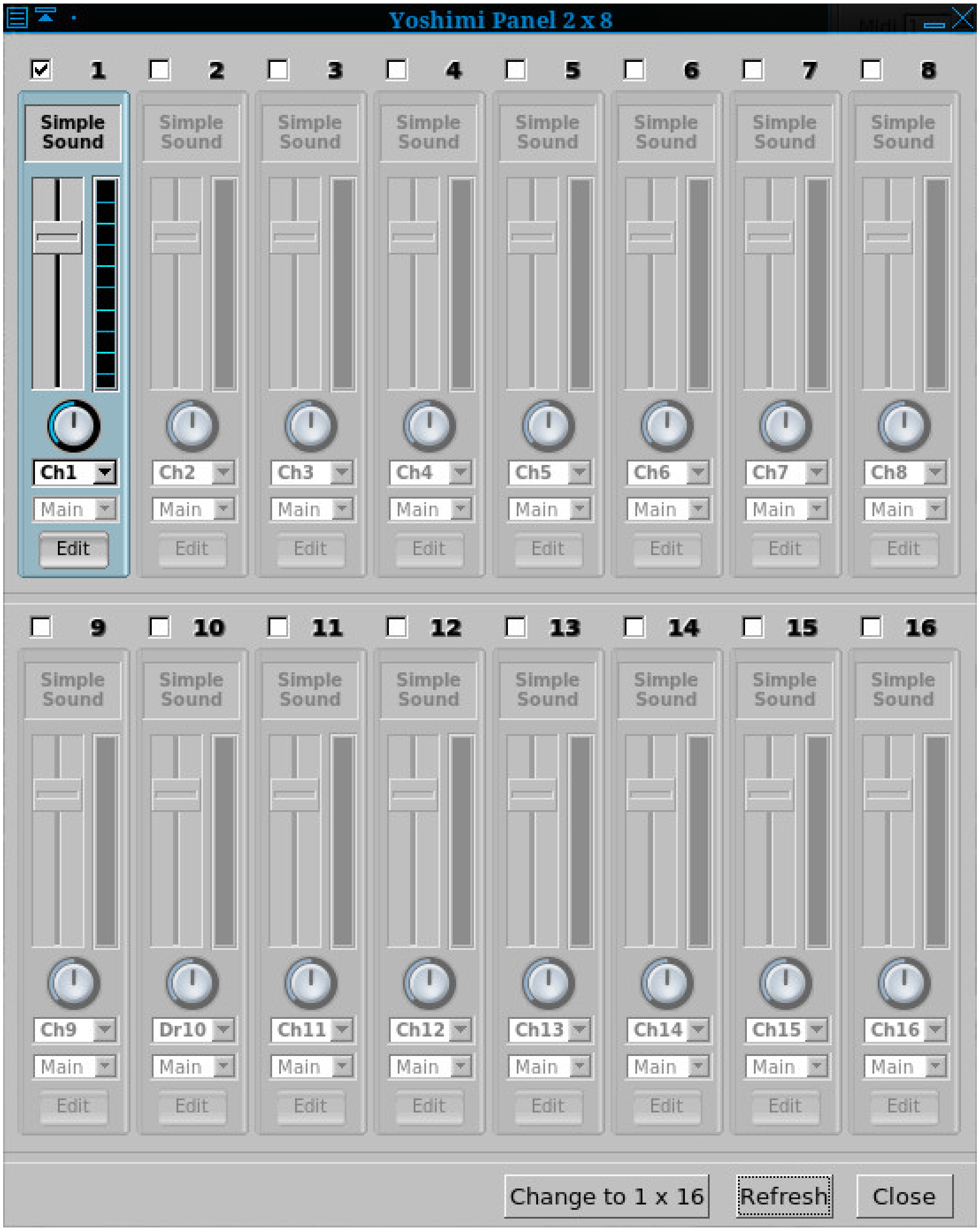

Figure 72: Yoshimi Part Panel, 2x8

1. **Part Summary.** Parts View or Summary.

2. Enable part. Enable/Disable the part. The check-boxes enable/disable the part.

Values: `Off*, On`

3. Part name. Instrument name. Click on this box to change the instrument.

4. Volume Slider. Volume Bar. Changes the volume of the part.

5. VU-meter display. Shows the level of the part when playing.

6. Panning Knob. Panning Dial-Button. Changes the panning of the part.

TODO.

7. Channel. Receive from MIDI channel. Changes the MIDI channel assigned to the part.

Values: `Ch1*, Ch2, ..., Ch16`

8. Main. Set Audio Destination. TODO.

Values: `xxxxxx`

9. Edit. Left mouse button: Part select. Right mouse button: Instrument edit.

Does not work!

TODO.

10. Parts Layout. Changes the layout of the panel.

11. Refresh. Refresh Edit. If one edits the parameters of the part in the main window, one may need to refresh the panel by clicking the Refresh button.

12. Close. Close the window.

5.2 Virtual Keyboard

This section describes the detailed usage of the *Yoshimi* virtual keyboard. The virtual keyboard lets one play notes using the keyboard/mouse. There is no MIDI requirement.

Using the keyboard. The keyboard is split into two "octaves"(in fact it is more than 1 octave). It may happen that the keys will not trigger any note-on This is because another widget than the keyboard itself is selected. In order to continue playing using the keyboard, click with the mouse on some keys on the virtual keyboard.

Using the mouse. One can use the mouse too, to play. If one presses the shift key while pressing the mouse button, the keys will be not released when the mouse button is released. If one presses the "Panic" or "Stop!" button from the ZynAddSubFX/Yoshimi main window, all keys will be released.

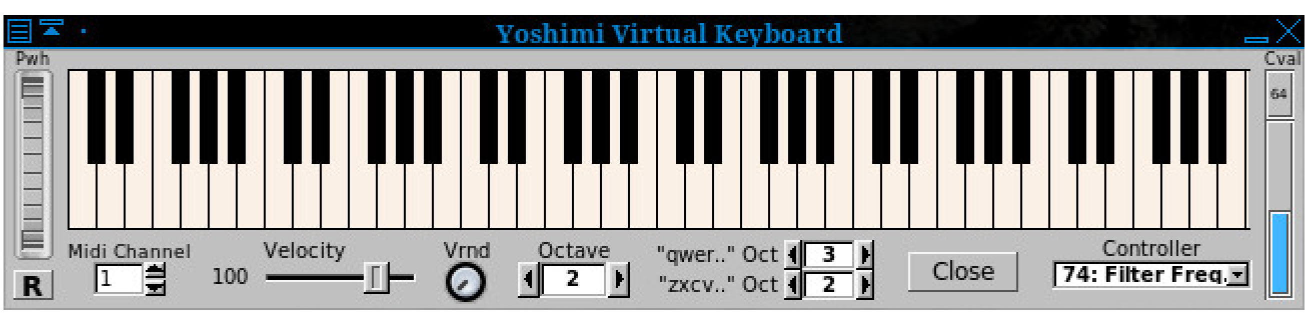

Figure 73: Yoshimi Virtual Keyboard

5.2.1 Virtual Keyboard, Basics

1. **Pwh**
2. **R**
3. **Midi Channel**
4. **Velocity**
5. **Velocity**
6. **Octave**
7. **"qwer.." Oct**
8. **"zxcv.." Oct**
9. **Controller**
10. **Cval**
11. **Close**

13. Pwh. Pitch bend knob. Pitch wheel. Press the **R** button to reset it.

14. R. Reset Pitch Bend.

15. Midi Channel. MIDI Channel. Sets the MIDI channel for the virtual keyboard.

Values: `1* to 16`

16. Velocity. Velocity of Notes. Sets the note-on velocity for the virtual keyboard.

Values: `1 to 127, 100*`

17. Velocity. Velocity Randomness.

Values: `0* to 127`

18. Octave. Transposes all of the virtual keyboard notes by the given number of octaves.

Values: `1, 2*, 3, 4, 5`

19. "qwer.." Oct. q2w3e4r5t6y Octave. Transposes the upper keys ("qwert"); the range of these keys is from C-4 to A-5 (replace the '5' with the octave).

20. "qwer.." Oct. zsxdcfvgbh Octave. Transposes the lower keys ("zxcvb"); the range of these keys is from C-3 to E-4 (replace the '4' with the octave).

Values: `1, 2*, 3, 4, 5`

21. Controller. Keyboard Controller.

Values: `01:Mod.Wheel, 07:Volume, 10:Panning, 11:Expression, 64:Sustain, 65:Portamento, 71:Filter Q, 74:Filter Freq*, 75:Bandwidth, 76:FM Gain, 77:Res.c.freq, 78:Res.bw.`

Sets the controller to be changed according to Cval. See section 5.2.3 ("Virtual Keyboard, Controllers") on page 81.

22. Cval. Controller value. Changes the controller value. Note that the Cval might not reflect the internal value of the controller when one changes the controller.

Values: `1 to 127, 96*`

23. Close. Close button.

5.2.2 Virtual Keyboard, ASCII Mapping

In addition to this virtual keyboard, the QWERTY (or Dvorak, or AZERTY) keyboards can be used to produce notes. The computer keyboard layout is shown in figure 10 ("QWERTY Virtual Keyboard") on page 26, From lowest octave to highest, the colors are blue, then green, then red. The "white" keys are the light colors, and the "black" keys are the deeper colors. The range of the keys on the "zxcvb..." row is C3 to E4. The range of the keys on the "qwert..." row is C4 to A5. These octave ranges can be adjusted.

The computer keyboard will produce notes only when the virtual keyboard is active.

TODO: Note that there may be some other keys that serve a purpose with the QWERTY keyboard.

Also note that we replaced the monopoly symbol with the monopolist symbol. On X11 systems, this key is known as the "Super" key.

5.2.3 Virtual Keyboard, Controllers

1. **Mod. Wheel**
2. **Volume**
3. **Panning**
4. **Expression**
5. **Sustain**
6. **Portamento**
7. **Filter Q**
8. **Filter Freq.**
9. **Bandwidth**
10. **FM Gain**
11. **Res. c. freq**
12. **Res. bw.**

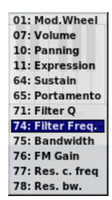

Figure 74: Virtual Keyboard Controllers

1. Mod. Wheel.

TODO.

2. Volume.

TODO.

3. Panning.

TODO.

4. Expression.

TODO.

5. Sustain.

TODO.

6. Portamento.

TODO.

7. Filter Q.

TODO.

8. Filter Freq..

TODO.

9. Bandwidth.

TODO.

10. FM Gain.

TODO.

11. Res. c. freq.

TODO.

12. Res. bw..

TODO.

6 Effects

The *Yoshimi* Effects panel provides a number of special effects that can be applied to parts. Effects are, generally, blackboxes that transform audio signals in a specified way. More exactly, the only input data for an effect in *ZynAddSubFX* is an array of samples, which is read on line. The output is the transformed array of samples.

As described, effects have no information about anything else. For example, key presses are not recognized. Therefore, pressing a key does not initiate the LFO. Phase knobs will always be relative to a global LFO, which is only dependent on the system time.

Wetness determines the mix of the results of the effect and its input. This mix is made at the effects output. If an effect is wet, it means that nothing of the input signal is bypassing the effect. If it is dry, then the effect has no effect.

The Effects panel is shown in figure 2 (”Yoshimi Main Screen, 1.3.5”) on page 11.

Note that these effects have been incorporated into a separate guitar-effects project called *Rakkarrak* [10].

There are two types of effects: System effects and Insertion effects. The System effects apply to all parts and allows one to set the amount of effect that applies to each part. Also, it is possible to send the output of one system effect to another system effect. In the user interface this is shown as ”source -<destination”. eg. ”0 -<1” means how much of the system effect 0 is sent to system effect 1.

Insertion effects are described in section 6.1.2 (”Effects / Panel Types / Insertion”) on page 86.

6.1 Effects / Panel Types

There are three variations of Effects sub-panels:

- **System Effects**.
- **Insertion Effects**.
- **Part/Instrument Effects**.

Here are the major elements of the main effects panel, which shows the System and Insertion effects tabs.

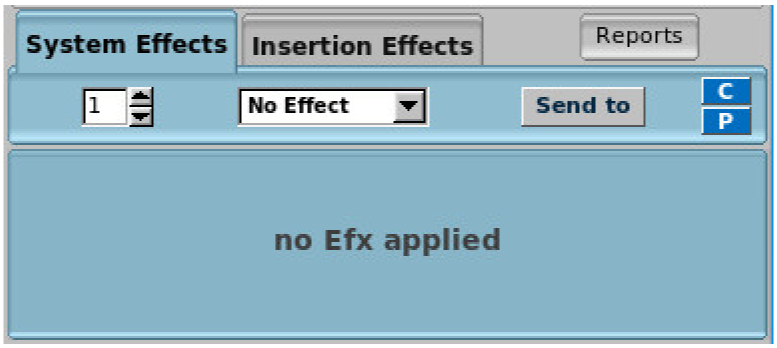

Figure 75: System Effects Dialog

1. **System Effects Tab**
2. **Effect Number**
3. **Effect Name**
4. **Send to**
5. **C**
6. **P**
7. **Effects Panel**
8. **Insertion Effects Tab**
9. **Reports**

1. System Effects Tab. System Effects Tab. The items in this tab are described in the next few paragraphs.

2. Effect Number. Effect Number. Up to 8 effects can be supported at one time by one part.

3. Effect Name. Effect Name.

Values: `No Effect*`, `Reverb`, `Echo`, `Chorus`, `Phaser`, `AlienWah`, `Distortion`, `EQ`, `DynFilter`

Figure 76: Effects Names

4. Send to. Effects Send To.

TODO: Document how effects/send-to works.

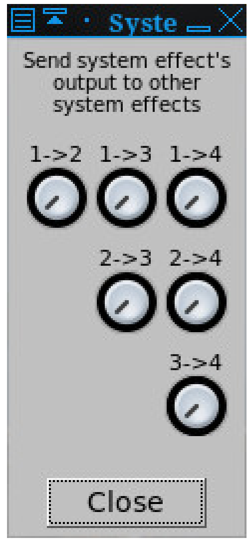

Figure 77: Effects, Send To

5. C. Copy-to-clipboard Dialog.

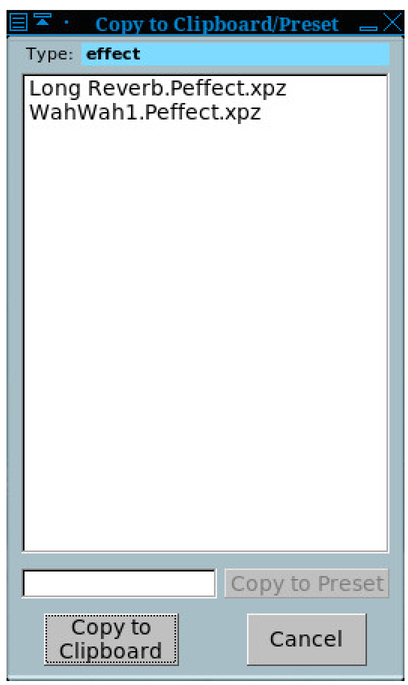

Figure 78: Effects / Copy To Clipboard

6. P. Paste-from-clipboard Dialog.

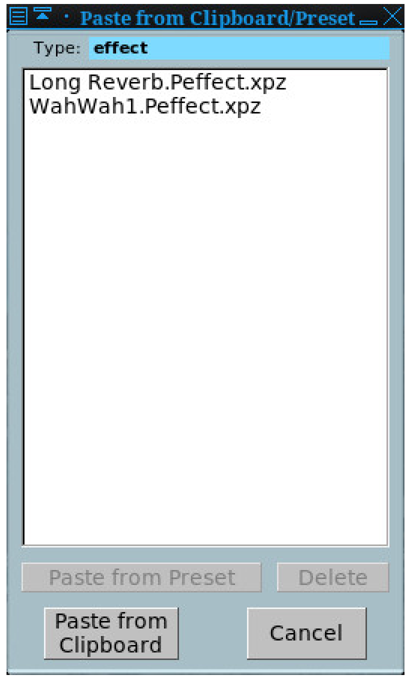

Figure 79: Effects / Paste From Clipboard

7. Effects Panel. Effects Panel. This area is filled by the controls for the selected effect.

8. Insertion Effects Tab. Insertion Effects Tab. The items in this tab are described below, in the 6.1.2 sub-section.

9. Reports. Effects Reports.

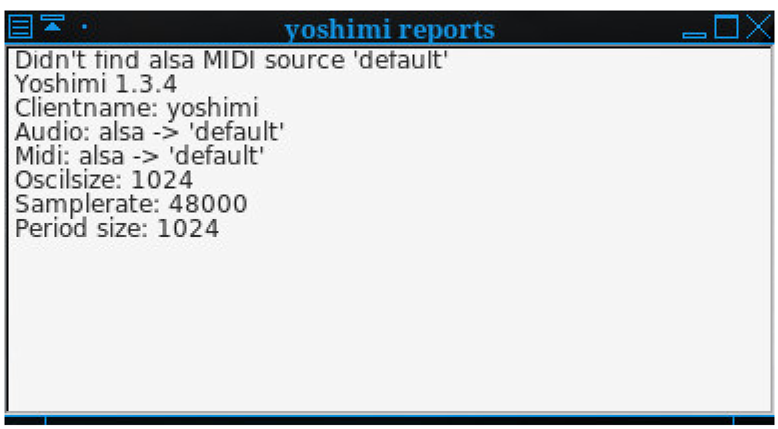

Figure 80: Effects / Reports

The next sub-sections show the variations on the effects panels, using the DynFilter effect as the subject effects panel.

6.1.1 Effects / Panel Types / System

The first variation appears when you enable an effect in the **System Effects** panel of the main *Yoshimi* dialog. It contains the standard controls for the given effect, plus the following interface items.

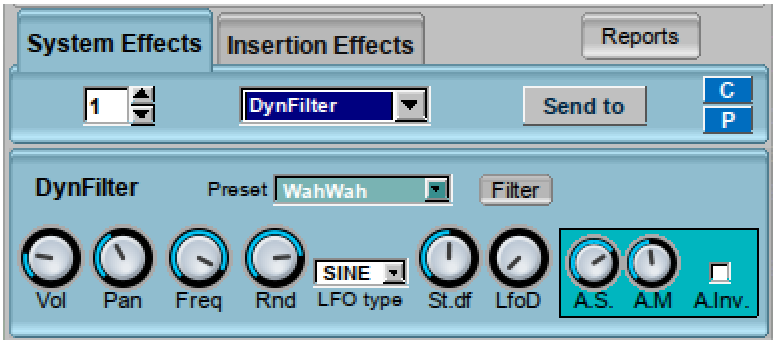

Figure 81: Sample System Effects Dialog

1. **Effect number**
2. **Effect selection**
3. **Effect Filter**
4. **C**
5. **P**

6.1.2 Effects / Panel Types / Insertion

The second effects variation appears when you enable an effect in the **Insertion Effects** panel of the main *Yoshimi* dialog. It contains the standard controls for the given effect, plus the following interface items.

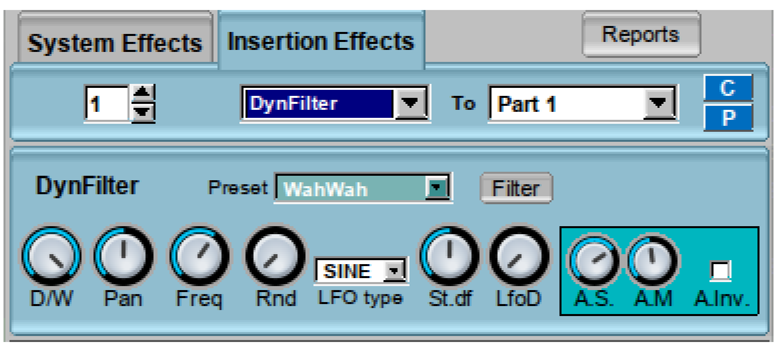

Figure 82: Sample Insertions Effects Dialog

1. **Effect number**
2. **Effect selection**
3. **To**
4. **C**
5. **P**

The insertion effects apply to one part or to master out. One may use more than one insertion effect for one part or master out. If one does so the effects with smaller indexes will be applied first (eg. first insertion effect no.0, than no.1, ...). If the part selected for insertion effect is "-1" then the effect will be disabled; if the part is "-2" the effect will be applied to Master Out.

1. To. Send the Effect To.

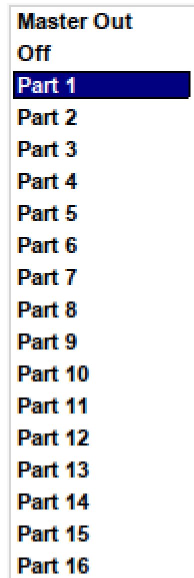

Figure 83: Part Selection Dropdown

6.1.3 Effects / Panel Types / Instrument

There is also a "part" or "instrument" effects window which is accessed by going to the main window, clicking the **Edit** button in the bottom panel to open the edit dialog, and then clicking the **Effects** button there. The part effects window has almost the same layout as System and Insertion effects; it is now almost identical to Insertion effects.

It contains the standard controls for the given effect, plus the following interface items.

1. **Effect number**
2. **Effect selection**
3. **To** (part-selection-dropdown.png)
4. **C**
5. **P**
6. **Bypass**
7. **Close**

"To" values:

Values: `Master Out, Off, Part 1, Part 2, ..., Part 16`

Figure 84: Sample Instrument Effects Dialog

Note the extra **Bypass** check-box, which presumably can be used to bypass this effects box in a given part.

6.2 Effects / None

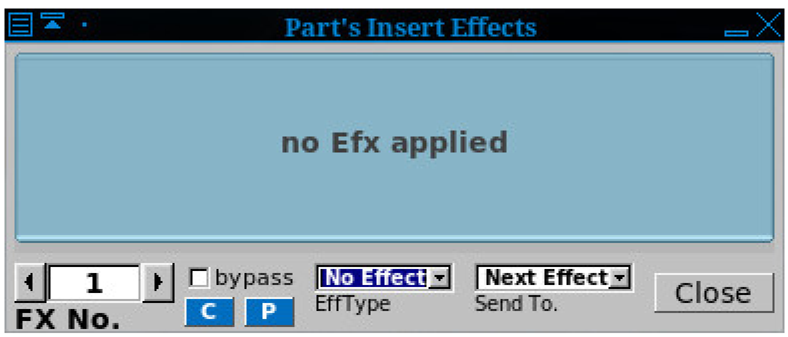

Figure 85: Effects Edit, None

This dialog form may be obsolete with the latest *Yoshimi*. It was captured around April of 2015. We have a lot of others in the same style that may need to be recaptured for use in new versions of the figures.

6.3 Effects / DynFilter

A dynamic filter is, as the name says, a filter which changes its parameters dynamically, dependent on the input and current time. In ZynAddSubFX, frequency is the only variable parameter. It can be used as an "envelope following filter" (sometimes referenced "Auto Wah" or simply "envelope filter").

6.3.1 Effects / DynFilter / Circuit

Though this filter might look a bit complicated, it is actually easy. We divide the parameters into two classes:

Filter Parameters are the ones obtained when one clicks on Filter. They give the filter its basic settings.

Effect Parameters are the other ones that control how the filter changes.

The filter basically works like this: The input signal is passed through a filter which dynamically changes its frequency. The frequency is an additive of:

- The filters base frequency.
- An LFO from the effect parameters.
- The "amplitude" of the input wave.

The amplitude of the input wave is not the current amplitude, but the so called "Root Mean Square (RMS)" value. This means that we build a mean on the current amplitude and the past values. How much the new amplitude takes influence is determined by the Amplitude Smoothness (see below).

RMS value plays an important role in the term loudness. A fully distorted signal can sound 20 db louder due to its higher RMS value. This filter takes this into account, depending on the smoothness.

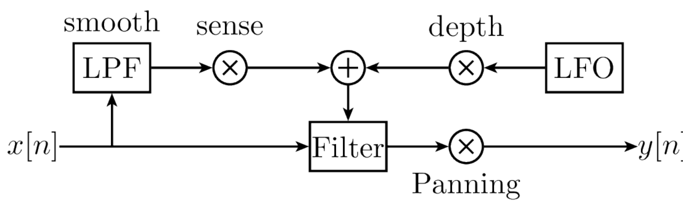

Figure 86: Dynamic Filter Circuit Diagram

6.3.2 Effects / DynFilter / User Interface

Figure 87: Effects Edit, DynFilter

This figure shows the Part/Instrument variation of the DynFilter sub-panel. The System/Insertion variation has the following elements.

1. **Preset**
2. **Filter**
3. **Vol** (system/insertion) or **D/W** (part/instrument)
4. **Pan**
5. **Freq**
6. **Rnd**
7. **LFO Type**
8. **St.df**
9. **LfoD**

10. **A.S.**
11. **A.M.**
12. **A.Inv.**

The 4 knobs in the middle (Freq, Rnd, LFO Type, St.df) control the LFO.

Let's start with the user-interface elements present in the System/Insertion variation of this effect.

1. Preset. DynFilter Preset.

Figure 88: DynFilter Presets

Values: `WahWah, AutoWah, Sweep, VocalMorph1, VocalMorph2`

2. Filter. DynFilter Filter.

This small button brings up Filter Params stock sub-panel item. This stock user-interface item is shown and described in section 4.2.5 ("Filter Parameters User Interface") on page 59.

3. Vol. DynFilter Volume.

Values: *0 to 127*

If the effect is used as a System effect, then this control appears.

4. D/W. DynFilter Dry/Wet Mix Setting.

Values: *0 to 127*

If the effect is used as an Insertion effect, then this control appears. "Dry" means the unprocessed signal and "wet" means the processed signal.

5. Pan. DynFilter Panning.

Values: *0 to 127*

After the input signal has passed through the filter, Pan can apply panning.

6. Freq. DynFilter LFO Frequency.

Values: *0 to 127*

7. Rnd. DynFilter LFO Randomness.

Values: *0 to 127*

8. LFO Type. DynFilter LFO Type.

9. St.df. DynFilter LFO ??????. TODO Left/right channel phase shift.

10. LfoD. DynFilter LFO Depth. This control is one that helps define the mix of the LFO and the amplitude.

11. A.S.. DynFilter A.S. TODO This control is one that helps define the mix of the LFO and the amplitude. A.S sets the Amplitude Sensing (i.e. how much influence the amplitude shall have).

12. A.M.. DynFilter A.M. One of two knobs let one control the way how the RMS value of the amplitudes is measured. A.M sets the Amplitude Smoothness (this is described above). The higher one sets this value, the more slowly will the filter react.

13. A.Inv.. DynFilter A.Inv. One of two knobs let one control the way how the RMS value of the amplitudes is measured. A.Inv., if set, negates the (absolute) RMS value. This will lower the filter frequency instead of increasing it. Note that this will not have much effect if the effects input is not very loud.

6.3.3 Effects / DynFilter / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the DynFilter effect.

For more information on NRPN, see section 2.6.2 ("Concepts / MIDI / NRPN") on page 22.

6.4 Effects / AlienWah

AlienWah is a nice effect done by Paul Nasca. It resembles a vocal morpher or wahwah a bit, but it is more strange. That's why he called it "AlienWah" The effect is a feedback delay with complex numbers.

The AlienWah effect is a special, dynamic formant filter (TODO: is this true?). Paul Nasca named it AlienWah because it sounded "a bit like wahwah, but more strange". The result of the filter is a sound varying between the vocals "Ahhhhh" (or "Uhhhhh") and "Eeeeee".

6.4.1 Effects / AlienWah / Circuit

No diagram, just a description of AlienWah.

Hint: Keep in mind that Effects that can be controlled by LFO can also be controlled arbitrarily: Set the LFO depth to zero and manipulate the phase knob (e.g. with NRPNs or maybe via OSC in the future).

The way that the filter moves between the two vocals is mainly described by an LFO. A bit easified, Paul Nasca has stated the formula (for i2 = -1 and R <1) as

$$fb = R * (cos() + i * sin())$$

$$yn = yn - delay * R * (cos() + i * sin()) + xn * (1 - R).$$

The input xn has the real part of the samples from the wavefile and the imaginary part is zero. The output of this effect is the real part of yn. is the phase.

6.4.2 Effects / AlienWah / User Interface

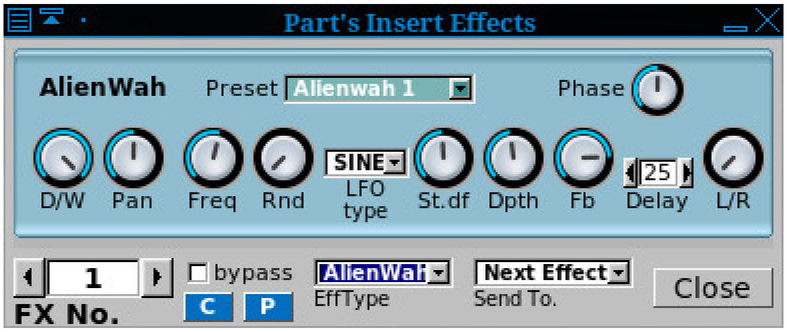

Figure 89: Effects Edit, AlienWah

1. **Preset**
2. **Phase**
3. **Vol** or **D/W**
4. **Pan**
5. **Freq**
6. **Rnd**
7. **LFO type**
8. **St.df.**
9. **Dpth**
10. **Fb.**
11. **Delay**
12. **L/R**

1. Preset. AlienWah Preset.

Values: `AlienWah 1, AlienWah 2, AlienWah 3, AlienWah 4`

2. Phase. The phase of the AlienWah. See in the above formula. This lets one set where the vocal is between "Ahhhhh" and "Eeeeee".

3. Vol. AlienWah Volume.

Values: `0 to 127`

The volume control is present is this effect is used as an insertion effect.

4. D/W. AlienWah Dry/Wet.

Values: `0 to 127`

The **Vol** control is replaced by this control if the effect is used as an Insertion effect.

5. Freq. LFO Frequency.

Values: `0 to 127`

Determines the LFOs frequency in relative units.

6. Rnd. LFO Amplitude Randomness.

Values: `0 to 127`

Part of the LFO definition.

7. LFO type. Set the LFO shape.

Values: `SINE, TRI`

Part of the LFO definition. Note that the LFO in other contexts has ramps and exponential shapes that are not present here.

8. St.df.. AlienWah Left/Right Chanell Phase Difference.

Values: `0 to 127`

Part of the LFO definition. Sets the phase difference between LFO for left/right channels. **St.df** lets one determine how much left and right LFO are phase shifted. 64.0 means stereo, higher values increase the right LFO relatively to the left one.

9. Dpth. LFO depth.

Values: `0 to 127`

Dpth is a multiplier to the LFO. Thus, it determines the LFO's amplitude and its influence.

10. Delay. Amount of delay before the feedback.

Values: `1 to 100`

If this value is low, the sound is turned more into a "wah-wah"-effect.

11. Fb.. AlienWah Feedback.

Values: `0 to 127`

TODO: What is the effect of the AlienWah feedback setting?

12. L/R. Determines how the left/right channels are routed to output:

- *Leftmost/0*. Left to left and right to right.
- *Middle/64*. Left+right to mono.
- *Rightmost/127*. Left to right, and right to left.

L/R applies crossover at the end of every stage. This is currently not implemented for the Analog Phaser.

13. Subtract. The output is inversed (inverted?)

6.4.3 Effects / AlienWah / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the AlienWah effect.

6.5 Effects / Chorus

In a chorus, many people sing together. Even if each of them sings at exactly the same frequency, all their voices usually sound different. We say they have a different timbre. Timbre is the way we perceive sound and makes us differ between different music instruments. This is, physically, achieved by varying both the amplitude envelope and the frequency spectrum. Multiple sounds with slightly different timbres make a sound more shimmering, or powerful. This is called the chorus effect.

The chorus effect can be achieved by multiple people singing together. In a concert, there are many instruments, resulting in the same effect. When making electronic music, we only have an input wave and need to generate these different timbres by ourselves. ZynAddSubFX therefore simply plays the sound, pitch modulated by an LFO, and adds this to the original sound. This explains the diagram below: The multiple pitches are generated by a delayed version of the input. This version is being pitched by an LFO. More detailed, this pitch is generated by varying the reading speed of the delayed sound; the variation amount is controlled by an LFO.

Related effects to Chorus are Flangers. Flangers can be described as Chorus with very short LFO delay and little LFO depth. One can imagine a flanger as two copies of a sound playing at almost the same time. This leads to interference, which can be clearly heared. It is popular to apply flangers to guitars, giving them more "character".

6.5.1 Effects / Chorus / Circuit

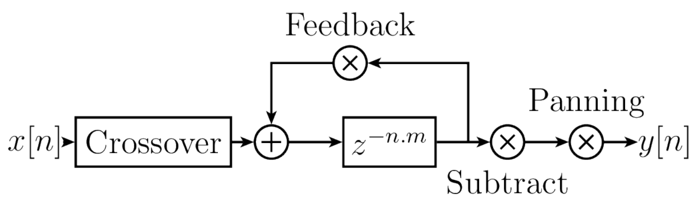

Figure 90: Chorus Circuit Diagram

First, crossover is applied.

The Freq, Rnd, LFO Type, St.df, Depth knobs control the LFO for the pitch. If the depth is set to zero, the pitch will not be changed at all.

Delay is the time that the delayed sound is delayed "on average". Note that the delay also depends on the current pitch.

After the correct element of the sound buffer is found using the LFO, the Fb knob lets one set how loud it shall be played. This is mostly redundant to the D/W knob, but we have not applied panning and substraction yet.

Next, the signal can be negated. If the Substract Checkbox is activated, the amplitude is multiplied by -1.

Finally, Pan lets one apply panning.

6.5.2 Effects / Chorus / User Interface

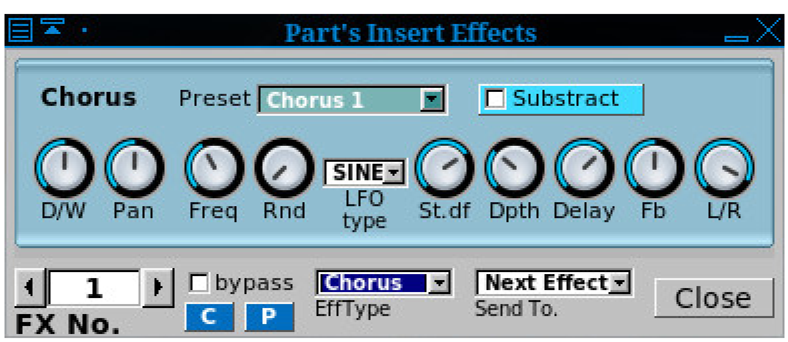

Figure 91: Effects Edit, Chorus

1. **Freq**
2. **Rnd**
3. **LFO type**
4. **St.df.**
5. **Dpth**
6. **Delay**
7. **Fb.**
8. **L/R**
9. **Subtract**

1. Freq. Chorus LFO Frequency.

2. Rnd. Chorus LFO randomness.

3. LFO type. Set the LFO shape.

4. St.df.. The phase difference between LFO for left/right channels .

5. Dpth. Chorus LFO depth.

6. Delay. Delay of the chorus. If one uses low delays and LFO depths, this will result in a flanger effect.

7. Fb.. Chorus Feedback.

8. L/R. How the left/right channels are routed to output:

1. leftmost. Left to left and right to right.
2. middle. Left+right to mono.
3. rightmost. Left to right, and right to left.

9. Subtract. The Chorus output is inversed (inverted?)

6.5.3 Effects / Chorus / NRPN Values

Effects can be controlled via ”non-registered parameter numbers”, or NRPNs. This section details the value supported by the Chorus effect.

6.6 Effects / Distortion

Distortion means, in general, altering a signal. Natural instruments usually produce sine-like waves. A wave is transformed in an unnatural way when distortion is used. The most distorted waves are usually pulse waves. It is typical for distortion to add overtones to a sound. Distortion often increases the power and the loudness of a signal, while the dB level is not increased. This is an important topic in the Loudness War.

As distortion increases loudness, distorted music can cause ear damage at lower volume levels. Thus, one might want to use it carefully. Distortion can happen in many situations when working with audio. Often, this is not wanted. In classical music, for example, distortion does not occur naturally. However, distortion can also be a wanted effect. It is typical for Rock guitars, but also present in electronic music, mostly in Dubstep and DrumNBass.

The basic components of distortion are mainly

- A preamplifier.

- The waveshaping function.
- Filters.

Preamplification changes the volume before the wave is shaped, and is indeed the amount of distortion. For example, if one clips a signal, the louder the input gets, the more distortion one will get. This can have different meanings for different types of distortions, as described below.

The filters are practical. A reason for using them afterwards is that distortion can lead to waves with undesired high frequency parts. Those can be filtered out using the LPF. A reason for using filters before applying is to achieve multiband distortion. ZynAddSubFX has no "real" multiband distortion by now, however.

The topic of types of distortion is completely discussed in the Oscillator Section.

(FIND THE REFERENCE)

Note that one can use the Oscillator editor in order to find out what the distortion effect does. Also note that while the Oscillator editors distortion is limited to some oscillators one can produce in the Oscillator editor, the distortion effect can be used on every wave that one can generate with ZynAddSubFX.

6.6.1 Effects / Distortion / Circuit

We explain the functionality in a diagram and list the components below.

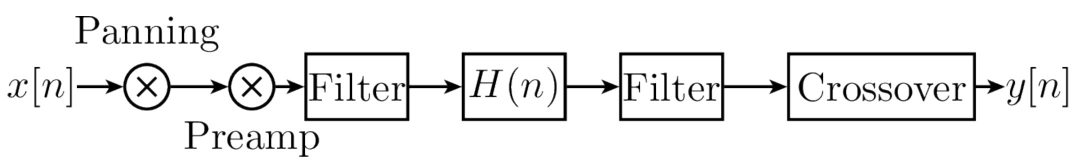

Figure 92: Distortion Circuit Diagram

Negation is the first thing to happen. If the Neg Checkbox is activated, the amplitude is multiplied by -1.

Panning is applied. Note, however, that one must activate the Stereo Checkbox, labelled St, before.

Pre-amplification is done next. The amount can be changed using the Drive nob. Indeed, this is the amount of distortion. For example, if one clips a signal, the louder the input gets, the more distortion one will get. This can have different meanings for different types of distortion, as described above.

HPF and LPF are filters with 2 poles. Whether they are used before or after the waveshape, depends on the checkbox labeled PF.

The next step is the wave shape. This defines how the wave is actually modified. The Type ComboBox lets one define how. We will discuss some types below.

After the wave shape, we scale the level again. This is called output amplification. One can change the value using the Level knob.

Crossover is the last step. This is controlled by the knob LR Mix and means that afterwards, a percentage of the left side is applied to the right side, and, synchronously, the other way round. It is a kind of interpolation between left and right. If one sets the LR Mix to 0.0, one will always have a stereo output.

6.6.2 Effects / Distortion / User Interface

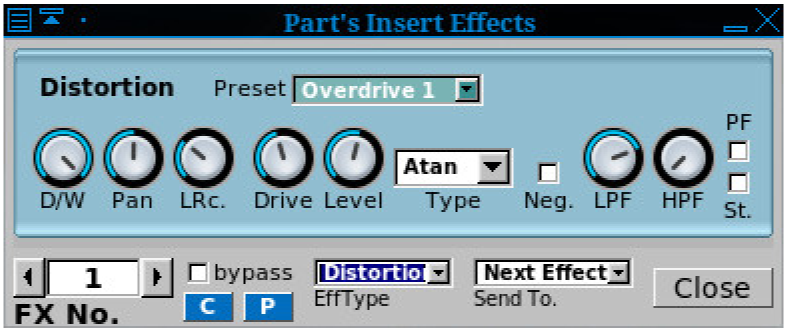

Figure 93: Effects Edit, Distortion

1. **Drive**
2. **Level**
3. **Type**
4. **Neg.**
5. **LPF**
6. **HPF**
7. **St.**

1. Drive. Set the amount of distortion.

2. Level. Amplify or reduce the signal after distortion.

3. Type. Set the function of the distortion (like arctangent, sine).

4. Neg.. Negates the amplitude (invert the signal).

5. LPF. Low Pass Filter.

6. HPF. High Pass Filter.

7. St.. Set the distortion mode (stereo or mono, checked is stereo).

6.6.3 Effects / Distortion / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the Distortion effect.

6.7 Effects / Echo

The echo effect, also known as delay effect, simulates the natural reflection of a sound. The listener can hear the sound multiple times, usually decreasing in volume. Echos can be useful to fill empty parts of songs with.

6.7.1 Effects / Echo / Circuit

The good circuit diagram is shown in an old printout we have, but the current version of the Echo description at http://zynaddsubfx.sourceforge.net/Doc/ shows a junk file. So Paul Nasca's description will have to suffice.

In ZynAddSubFX, the echo is basically implemented as the addition of the current sound and a delayed version of it. The delay is implemented as in the picture below. First, we add the delayed signal to the effect input. Then, they pass an LP1. This shall simulate the effect of dampening, which means that low and especially high frequencies get lost earlier over distance than middle frequencies do. Next, the sound is delayed, and then it will be output and added to the input.

The exact formula in the source code for the dampening effect is as follows:

$$Y(t) := (1 - d)X(t) + dY(t - 1)$$

where t be the time index for the input buffer, d be the dampening amount and X,Y be the input, respective the output of the dampening. This solves to

$$Y(z) = Z(Y(t)) = (1 - d)X(z) + dY(z)z - 1H(z) := Y(z)X(z) = 1 - d1 - dz - 1$$

which is used in $Y(z) = H(z)X(z)$. So H(z) is indeed a filter, and by looking at it, we see that it is an LP1. Note that infinite looping for d=1 is impossible.

6.7.2 Effects / Echo / User Interface

Figure 94: Effects Edit, Echo

TODO (yoshimi): Pan lets one apply panning of the input.

1. **Delay**
2. **LRdl.**
3. **LRc.**
4. **Fb.**
5. **Damp**

1. Delay. The delay time of one echo.

2. LRdl.. Left-Right-Delay. The delay between left/right channels. If it is set to the middle, then both sides are delayed equally. If not, then the left echo comes earlier and the right echo comes (the same amount) later than the average echo; or the other way round. Set the knob to 0 to hear on the right first.

3. LRc.. Echo Crossover. The "crossing" between left/right channels.

4. Fb.. Echo feedback. Feedback describes how much of the delay is added back to the input. Set Fb. to the maximum to hear an infinite echo, or to the minimum to just hear a single repeat.

5. Damp. Echo damping. How high frequencies are damped in the Echo effect. The Damp value lets the LP1 reject higher frequencies earlier if increased.

6.7.3 Effects / Echo / NRPN Values

An equalizer is a filter effect that applies different volume to different frequencies of the input signal. This can, for example, be used to "filter out" unwanted frequencies. ZynAddSubFXs implementations follow the "Cookbook formulae for audio EQ" by Robert Bristow-Johnson. (NEED A REFERENCE)

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the Echo effect.

TODO TODO TODO TODO

6.8 Effects / EQ

EQ is a parametric equalizer. On the equalizer graph there are 3 white vertical bars for 100Hz, 1kHz, 10kHz.

6.8.1 Effects / EQ / Circuit

6.8.2 Effects / EQ / User Interface

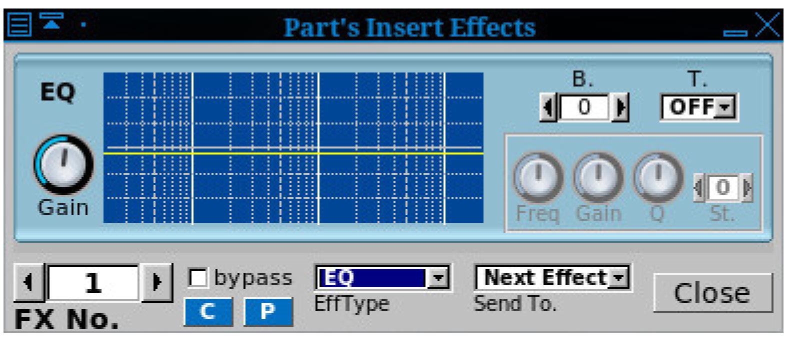

Figure 95: Effects Edit, EQ

We describe all parts of the GUI here. The term passband (or often just "band") refers to the amount of frequencies which are not significantly attenuated by the filter.

1. **Gain**
2. **B**
3. **T**
4. **Freq**
5. **Gain**
6. **Q**
7. **St**

Global:

1. Gain. Amplifies or reduce the signal that passes through EQ.

2. B. Set the current frequency band (or filter). B lets one choose the passband number. Multiple passbands define one filter. This is important if one wants multiple filters to be called after each other. Note that filters are commutative.

Bands:

3. T. Set the type of the filter.

4. Freq. The frequency of the filter. Freq describes the frequencies where the filter has its poles. For some filters, this is called the "cutoff" frequency. Note, however, that a bandpass filter has two cutoff frequencies.

5. Gain. The gain of the filter. Gain is only active for some filters and sets the amount of a special peak these filters have. Note that for those filters, using the predefined gain makes them effectless.

6. Q. The Q (resonance, or bandwidth) of the filter. Resonance lets one describe a peak at the given frequency for filters with 2 poles. This can be compared to real physical objects that have more gain at their resonance frequency.

7. St. Number of additional times the filter will be applied (in order to do very steep roll-off - eg. 48 dB/octave). St. lets one define multiple filter stages. This is equivalent to having multiple copies of the same filter in sequence.

6.8.3 Effects / EQ / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the EQ effect.

6.9 Effects / Phaser

The Phaser is a special dynamic filter. The result is a sweeping sound, which is often used on instruments with a large frequency band, like guitars or strings. This makes it typical for genres like rock or funk, where it is often modulated with a pedal, but also for giving strings a warm, relaxing character.

6.9.1 Effects / Phaser / Circuit

No circuit diagram, just a picture of the results of the phaser effect in the form of a spectrogram.

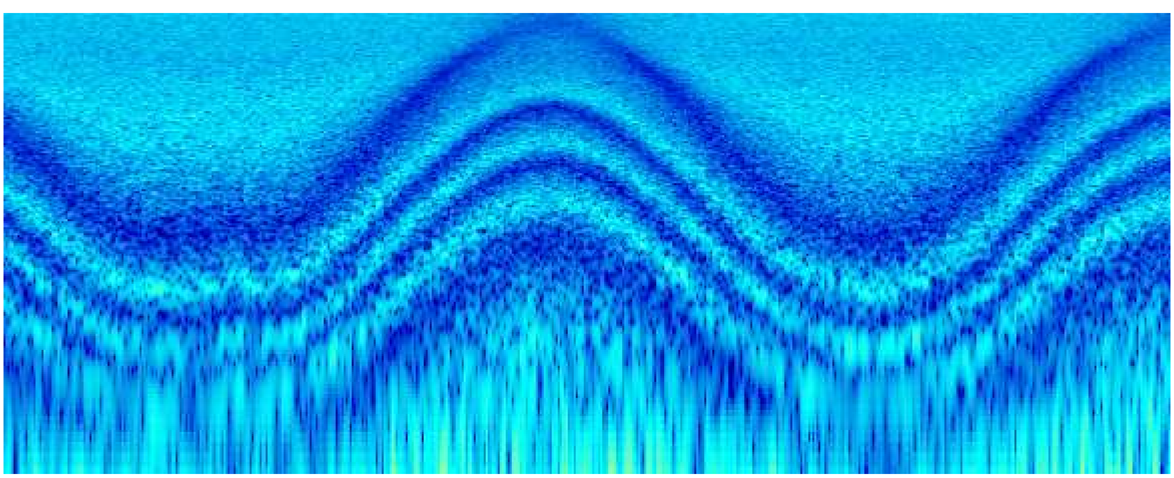

Figure 96: Phaser Circuit Diagram

The audio signal is split into two paths. One path remains unchanged. The other one is sent to a delay line. The delay time (the so called phase) is made dependent on the frequency. Therefore, an all-pass filter is applied to the signal, which preserves the amplitude, but determines the delay time. In the end, both paths are added.

The following picture describes how this works on white noise. Light blue signalises that the frequency is not present at the current time, and dark blue signalises the opposite. The dark blue peaks appear if the delay time is very short, because then, the second path almost equals the first one, which results in duplication of the signal. If the delay line is very long, then it is – in the case of white noise – totally at random whether the delayed signal currently duplicates the unchanged path, or whether it cancels it out to zero. This random effect results in white noise between the clear blue structures.

ZynAddSubFX offers different types of phasers:

- Analog and "normal" phasers. Analog phasers are more complicated. They sound punchier, while normal phasers sound more fluently. However, analog filters usually need more filter stages to reach a characteristic sound.
- Sine and triangle filters. Note that an analog triangle filter with many poles is a barber pole filter and can be used to generate Shepard Tones, i.e. tones that seem to increase or decrease with time, but do not really.
- The LFO function can be squared. This converts the triangle wave into a hyper sine wave. The sine squared is simply a faster sine wave.

TODO: Barber is deactivated, since PLFOtype is only 0 or 1?

For the normal phaser:

1. First, the LFO is generated. There are 4 controls (Freq,Rnd,LFO tpye,St.df) that define the LFO.
2. Phase and Depth are applied afterwards in the usual way (TODO: I dont understand the code here for the normal phase). For the analog phaser, Phase is not implemented, yet.
3. If hyp is being set, then the LFO function is being squared.
4. Next, the input is being used.
5. Analog decides whether the phaser is analog or "normal".
6. First, Pan applies panning to the original input in every loop.
7. Next, barber pole phasing is being applied (Analog only).
8. Fb applies feedback. The last sound buffer element is (after phasing) multiplied by this value and then added to the current one. For normal filter, the value is added before, for analog after the first phasing stage.
9. Now, Stages phasing stages are being applied. dist sets the distortion for when applying the phasing stages. This has only effect for analog phasers.
10. The feedback is performed next.
11. In the end, Substract inverts the signal, multiplying it by -1.

6.9.2 Effects / Phaser / User Interface

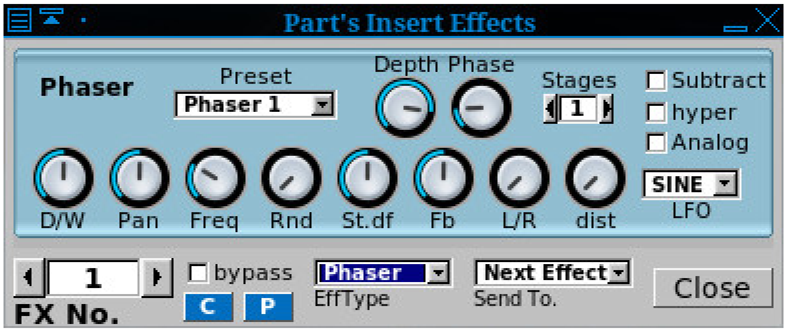

Figure 97: Effects Edit, Phaser

TODO. Include the item-paragraphs for each GUI element.

1. **Preset.**
2. **Depth**
3. **Phase**
4. **Stages**
5. **Subtract**
6. **hyper**
7. **Analog**
8. **D/W**
9. **Pan**
10. **Freq**
11. **Rnd**
12. **St.df**
13. **Fb**
14. **L/R**
15. **dist**
16. **LFO**

The extra fields that are shown if the effect is an insertion effect are not shown. They are described in section TODO TODO TODO.

8. Preset. Phaser Presets.

TODO: need a diagram of the dropdown

Values: `Phaser 1, ...` TODO.

9. Depth. Phaser Depth. Phaser LFO Depth?

Values: `TODO`

10. Phase. Phaser Phase.

Values: `TODO`

11. Stages. Phaser Stages.

Values: `TODO`

12. Subtract. Phaser Subtract.

Values: `Off*, On`

13. hyper. Phaser Hyper.

Values: `Off*, On`

14. Analog. Phaser Analog.

Values: `Off*, On`

15. D/W. Phaser Dry/Wet.

16. Pan. Phaser Panning.

17. Freq. Phaser Freq.

18. Rnd. Phaser Randomness.

19. St.df. The phase difference between LFO for left/right channels.

20. Fb. Phaser Feedback.

21. L/R. L/R. How the left/right channels are routed to output:

1. leftmost. Left to left and right to right.
2. middle. Left+right to mono.
3. rightmost. Left to right, and right to left.

22. dist. Phaser Distortion? TODO

23. LFO. Phaser LFO Type.

6.9.3 Effects / Phaser / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the value supported by the Phaser effect.

6.10 Effects / Reverb

A Reverberation actually expresses the effect of many echoes being played at the same time. This can happen in an enclosed room, where the sound can be reflected in different angles. Also, in nature, thunders approximate reverbs, because the sound is reflected in many different ways, arriving at the listener at different times.

In music, reverbs are popular in many ways. Reverbs with large room size can be used to emulate sounds like in live concerts. This is useful for voices, pads, and hand claps. A small room size can simulate the sound board of string instruments, like guitars or pianos.

6.10.1 Effects / Reverb / Circuit

As mentioned, a reverb consists of permanent echo. The reverb in ZynAddSubFX is more complex than the echo. After the delaying, comb filters and then allpass filters are being applied. These make the resulting sound more realistic. The parameters for these filters depend on the roomsize. For details, consider the information about Freeverb.

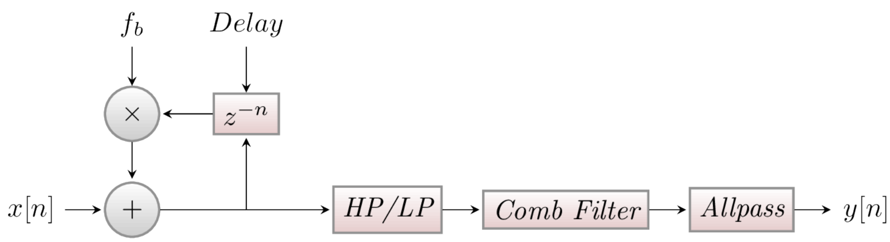

Figure 98: Reverb Circuit Diagram

6.10.2 Effects / Reverb / User Interface

The user-interface for the Reverb effect depends on whether it is used as a System effect or an Insertion effect. Observr figure 99 ("Effects Edit, Reverb") on page 104, where the Insertion mode is shown. In the System mode, only the light-blue portion of the user-interface appears.

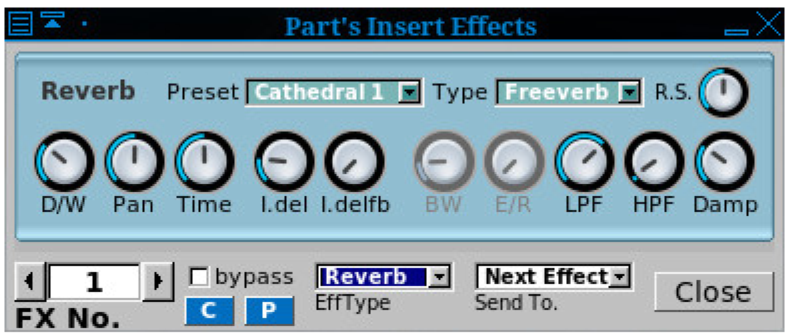

Figure 99: Effects Edit, Reverb

1. **Preset**
2. **Type**
3. **R.S.**
4. **D/W**
5. **Pan**
6. **Time**
7. **I.del**
8. **I.delfb**
9. **BW**
10. **E/R**
11. **LPF**
12. **HPF**
13. **Damp**
14. **FX No.**
15. **bypass**
16. **EffType**
17. **Send To**
18. **C**
19. **P**
20. **Close**

1. Preset. Reverb Preset.

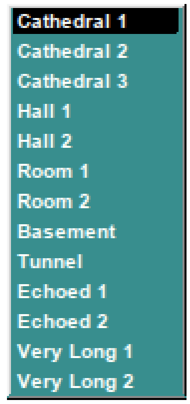

Figure 100: Reverb Preset Dropdown

Values: `Cathedral 1, Cathedral 2, Cathedral 3, Half 1, Half 2, Room 1, Room 2, Basement, Tunnel, Echoed 1, Echoed 2, Very Long 1, Very Long 2`

2. Type. Reverb Type. The combobox lets one select a reverb type.

Figure 101: Reverb Type Dropdown

- Freeverb is a preset. It was proposed by Jezar at Dreampoint.
- Bandwidth has the same parameters for the comb and allpass filters, but it applies a unison before the LPF/HPF. The unisons bandwidth can be set using BW.
- Random chooses a random layout for comb and allpass each time the type or the roomsize is being changed.

Values: `Random, Freeverb, Bandwidth`

3. R.S.. Reverb Room Size. The room size defines parameters only for the comb and allpass filters.

4. D/W. Reverb Dry/Wet Setting. This setting controls much of the original signal is mixed with the reverb effect.

5. Pan. Reverb Panning. Pan lets one apply panning. This is the last process to happen.

6. Time. Reverb Time. Set the duration of late reverb. Time controls how long the whole reverb shall take, including how slow the volume is decreased.

7. I.del. Reverb Initial Delay. The initial delay (I.del) is the time which the sounds need at least to return to the user.

8. I.delfb. Reverb Initial Delay Feedback. Sets the initial delay feedback. The initial delay feedback (I.delfb) says how much of the delayed sound is added to the input. It is not recommended to use this setting together with low initial delays).

9. BW. Reverb Bandwidth.

10. E/R. Reverb E/R. Echo Reflection? TODO!

11. LPF. Reverb Lowpass Filter. This filter is applied before the comb filters.

12. HPF. Reverb Highpass Filter. This filter is applied before the comb filters.

13. Damp. Reverb Damp. Damp determines how high frequencies are damped during the reverberation. The dampening control (Damp) currently only allows to damp low frequencies. Its parameters are used by the comb and allpass filters.

14. FX No.. Reverb FX Number.

Values: `1 to 8?`

15. bypass. Reverb FX Bypass.

Values: `Off*, On`

16. EffType. Reverb Effect Type.

Values: `Reverb, EQ, Echo, etc.  TODO`

17. Send To. Reverb Send To.

Values: `Next Effect, ...  TODO`

18. C. Reverb Copy.

19. P. Reverb Paste.

20. Close. Close Window.

6.10.3 Effects / Reverb / NRPN Values

Effects can be controlled via "non-registered parameter numbers", or NRPNs. This section details the values supported by the Reverb effect.

TODO: detail the values supported by the Reverb effect.

7 Bottom Panel

7.1 Bottom Panel Controls

The *Yoshimi* bottom panel provides quick access to some major features of the application. The bottom panel is shown in figure 2 ("Yoshimi Main Screen, 1.3.5") on page 11.

Here are the major elements of the bottom panel.

1. **Part**
2. **Part of**
3. **Instrument Name**
4. **Edit** (Instrument Edit Button)
5. **Midi**
6. **Mode**
7. **Enabled**
8. **Portamento**
9. **Velocity Sens**
10. **Velocity Offset**

11. **Pan**
12. **Pan Reset Button**
13. **Volume**
14. **Controllers**
15. **Minimum Note**
16. **Maximum Note**
17. **m**
18. **R**
19. **M**
20. **Key Shift**
21. **Key Limit**
22. **System Effect Sends 1**
23. **System Effect Sends 2**
24. **System Effect Sends 3**
25. **System Effect Sends 4**
26. **Sound Meter**

1. Part. Part Number.

Values: `1 to 16; 1 to 32; 1 to 64`

Show and set current part. The maximum number of values depends on the **Part of** selection.

2. Part of. Maximum Number of Parts.

Values: `16*, 32, 64`

Yoshimi now has up to 64 parts in blocks of 16. One can now decide how many one wants to have available using this user-interface item.

By default, all the upper parts (numbers greater than 16) are mapped to the same MIDI channel numbers as the lowest ones, but have independent voice and parameter settings. They can not normally receive independent note or control messages. However, vector control will intelligently work with however many one has set, as will all the NRPN direct part controls. See section 13.2 ("NRPN / Vector Control") on page 160.

This item is a fairly new feature of *Yoshimi* (as of version 1.3.5).

3. Instrument Name. Instrument Name. Left-click to open the Bank window. Right-click to change the name of the current instrument. If you change the name of the instrument, be sure to select **Menu / Instrument / Save Instrument** to preserve that change.

The name now has color-coding to indicate the instrument's use of ADDsynth, SUBsynth, or PADsynth. One can see the "red" color for ADDsynth in the figure for the bottom panel. Blue would indicate SUBsynth, and green would indicate PADsynth.

4. Edit. Instrument Edit button. This button brings up the instrument-edit dialog shown in figure 103 ("Instrument Edit Dialog") on page 112.

This dialog provides a very broad overview of the instrument, and provides access to far more detailed dialogs to edit the instrument. This dialog is explained in detail in section 7.3 ("Bottom Panel Instrument Edit") on page 112.

5. Midi. MIDI Channel.

Values: `1 to 16`

6. Mode. Mode. Poly. Sets the mode (polyphonic/monophonic/legato). In polyphonic mode, multiple simultaneous notes are supported. (How many at maximum?). In monophonic mode, only one note is supported. In legato mode, the sound flows smoothly from note to note without any breaks.

Values: `Poly, Mono, Legato`

7. Enabled. Enable the part. If the Part is disabled it doesn't use CPU time.

Values: `Off*, On`

8. Portamento. Enable/disable the portamento. One can set the duration and other parameters by opening the Controllers window.

Values: `Off*, On`

9. Velocity Sens. Velocity Sensing Function.

Values: `0 to 127, 64*`

10. Velocity Offset. Velocity Offset.

Values: `0 to 127, 64*`

11. Pan. Pan.

Values: `0 to 127, 64*`

12. Pan (reset). Reset Pan to Middle (64).

13. Volume. Instrument Volume.

Values: `0 to 127, 64*` True???

14. Minimum Note. Minimum note the part receives.

Values: `0* to 127`

15. Maximum Note. Maximum note the part receives.

Values: `0 to 127*`

16. m. Minimum Note Capture Button.

Set minimum note to last note played.

17. R. Minimum and Maximum Note Reset Button.

Reset the minimum key to 0 and the maximum key to 127.

18. M. Maximum Note Capture Button.

Set maximum note to last pressed key.

19. Key Shift. Key Shift.

Values: `-12 to 12, 0*`

20. Key Limit. Maximum keys for this part.

Values: `0 to 55, 15*`

21. System Effect Sends 1, 2, 3, and 4.

TODO: Describe how these sends work.

Values: `0 to 127*`

22. Sound Meter. VU Meter. Sound Meter.

This discussion of "Audio Output and Levels" comes from `Output Levels.txt`.

At the bottom of the main window there is a pair of horizontal grids representing a bargraph type display. The upper one is for the left hand channel and the lower one for the right hand one. The grid divisions each represent 1dB, and the brighter divisions are therefore 5dB. The thicker bright divisions therefore being 10dB. The overall scale range is -48dB to 0dB.

As the output level rises pale blue strips will light up in these grids. These fast responding bars are the peak levels and should never be allowed to go above 0dB, otherwise the output is likely to be clipped and distorted. There is also a pair of boxes on the end of these grids which will show the highest peak level seen. If clipping has happened the box background will change from black to red.

To clear clip and peak level indication click on this area.

As well as the peak level, the display shows a much slower responding RMS level, as a yellow line on top of the blue bar. This gives and indication of the apparent accoustic power.

If one opens the panel window one will see vertical bargraphs for each individual part. On these, the faint bars are 5dB steps and the bright ones 10dB. The peak leve isn't shown numerically, but if one exceeds 0dB a thick red line will appear at the top of the bargraph. This is also cleared from the box in the main window.

7.1.0.1 Tip: Using the VU Meter

The VU meter topic is very interesting, because one of the problems is a tendency to overdrive by way of sustain pedal. At the last test it showed up in the output before it showed up in the VU meter, so the VU meter should help a lot in analysis.

One way to avoid overdrive is to keep polyphony to 20 on each patch (two or three patches per *Yoshimi* instance, with two or three *Yoshimi* simultaneous instances depending on the patch).

Another item which helps a lot is compression (for example, the Calf multiband compressor is amazingly good.

7.2 Bottom Panel / Controllers

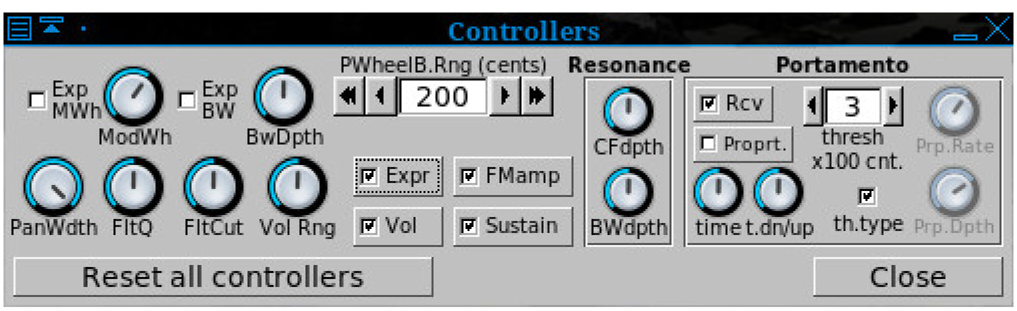

Figure 102: Controllers Dialog

1. **Exp MWh**
2. **ModWh**
3. **Exp BW**
4. **BwDepth**
5. **PanWdth**
6. **FltQ**

7. **FitCut**
8. **Vol Rng**
9. **PWheelB.Rng**
10. **Expr**
11. **FMamp**
12. **Vol**
13. **Sustain**
14. **Resonance** (section)
15. **PortaMento** (section)
16. **Reset all controllers**
17. **Close**

1. Exp MWh. Exponential Modulation Wheel. Changes the modulation scale to exponential.

Values: `Off*, On`

2. ModWh. Modulation Wheel Depth.

Values: `0 to 127, 80*`

3. Exp BW. Exponential Bandwidth Controller. Changes the bandwidth scale to exponential.

Values: `Off*, On`

4. BwDepth. Bandwidth Depth.

Values: `0 to 127, 64*`

5. Exp BW. Exponental Bandwidth. Changes the bandwidth scale to exponential.

Values: `0 to 127, 64*`

6. PanDpth. Panning Depth.

Values: `0 to 64*`

7. FltQ. Filter Q (resonance) Depth.

Values: `0 to 127, 64*`

8. FltCut. Filter Cutoff Frequency Depth.

Values: `0 to 127, 64*`

9. Vol Rng. Volume Range.

Values: `64 to 127, 64*`

10. PWheelB.Rng. Pitch Wheel Bend Range (cents). 100 cents = 1 halftone.

Values: `-6400 to 6400, 200*`

11. Expr. Expression Enable. Enable/disable expression.

Values: `Off, On*`

12. FMamp. FM Amplitude Enable. Enable/disable receiving Modulation Amplitude controller (76).

Values: `Off, On*`

13. Vol. Volume Enable.

Values: `Off, On*`

Enable/disable receiving volume controller. Sensitivity to MIDI volume change (CC7) is now variable in 'Controllers' in the same way as pan width etc. The nnumeric range is 64 to 127; the default at 96 gives the same sensitivity as before at -12dB relative to the GUI controls. 127 gives 0dB and 64 gives -26dB

14. Sustain. Sustain Pedal Enable. Enable/disable sustain pedal.

Values: `Off, On*`

15. Reset all controllers. Reset All Controllers.

16. Close. Close Window.

7.2.1 Bottom Panel / Controllers / Resonance

1. CFdepth. Resonance Center Frequency Depth, Center Frequency Controller Depth.

Values: `0 to 127, 64*`

2. BWdepth. Resonance Bandwidth Depth, Resonance Bandwidth Controller Depth.

Values: `0 to 127, 64*`

7.2.2 Bottom Panel / Controllers / Portamento

1. Rcv. Portamento Receive, Receive Portamento Controllers. Determines if the part receives Portamento On/Off (65) controller.

Values: `Off, On*`

2. Proprt.. Portamento Proportional, Enable Proportional Portamento (over fixed portamento).

Values: `Off*, On`

3. time. Portamento time. The duration of the portamento.

Values: `0 to 127, 64*`

4. t.dn/up. Portamento Time Stretch (up/down).

Values: `0 to 127, 64*`

5. threshx100 cnt.. Threshold of the Portamento.

Values: `0 to 127, 3*`

Minimum or maximum difference of notes in order to do the portamento (x 100 cents). It represents the minimum or the maximum number of halftones (or hundred cents) required to start the portamento. The difference is computed between the last note and current note.

The threshold refers to the frequencies and not to MIDI notes (one should consider this if one uses microtonal scales).

6. th.type. Threshold Type (min/max). Checked means that the portamento activates when the difference of frequencies is above the threshold ("thresh"); not checked is for below the threshold.

Values: `Off, On*`

7. Propt.. Proportional Portamento. If set, the portamento is proportional to ratio of frequencies.

Values: `Off, On*`

8. Prp.Rate. Distance required to double change from nonproportional portamento time. The ratio needed to double the time of portamento.

Values: `0 to 127, 80*`, requires **Proprt.** = `On`

9. Prp.Depth. The difference from nonproportional portamento.

Values: `0 to 127, 90*`, requires **Proprt.** = `On`

7.3 Bottom Panel Instrument Edit

The main instrument-editing dialog is relatively simple, and provides for editing information that identifies the instrument, and buttons to access the more complex dialogs of the ADDsynth, SUBsynth, PADsynth, Kit Edit, and Effects components.

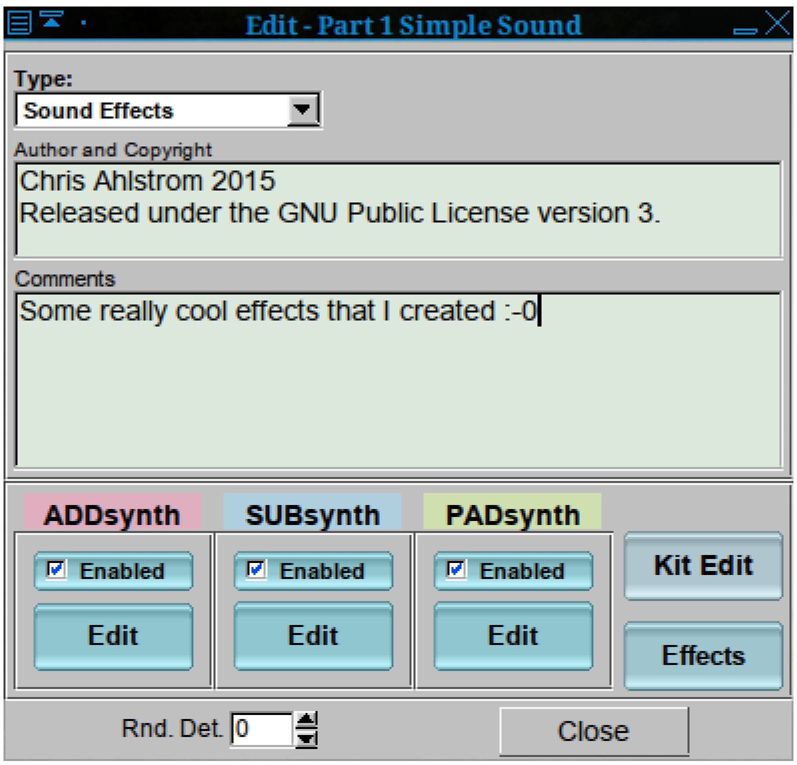

Figure 103: Instrument Edit Dialog

This dialog provides a very broad overview of the instrument, and provides access to far more detailed dialogs to edit the instrument. This dialog is called up by the **Edit** button on the bottom panel of the main *Yoshimi* main screen.

1. **Type**
2. **Author and Copyright**
3. **Comments**
4. **ADDsynth**
   1. **Enabled**
   2. **Edit**
5. **SUBsynth**
   1. **Enabled**
   2. **Edit**
6. **PADsynth**

   1. **Enabled**
   2. **Edit**
7. **Kit Edit**
8. **Effects**
9. **Rnd. Det.**
10. **Close**

The ADDsynth, SUBsynth, PADsynth, Kit Edit, and Effects dialogs are detailed in separated sections, as they are all very complex dialogs with many sub-dialogs.

10. Type. Instrument Type. Instrument Category.

This dropdown dialog allows one to tag the type of instrument, to indicate what category of instruments it fits into. The following figure shows the types.

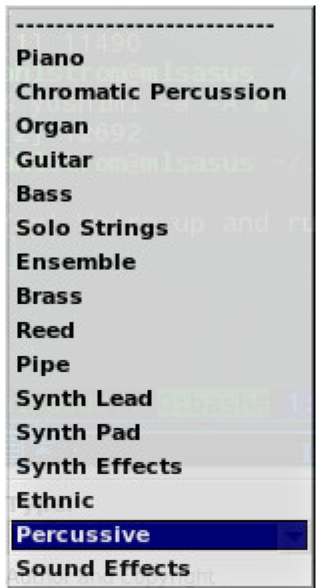

Figure 104: Instrument Type Dropdown

Values: `Piano, Chromatic Percussion, Organ, Guitar, Bass, Solo Strings, Ensemble, Brass, Reed, Pipe, Synth Lead, Synth Pad, Synth Effects, Ethnic, Percussive, Sound Effects`

11. Author and Copyright. This field provides space for identifying the author, copyright, and license for the part.

12. Comments. Allows free-form comments and notes to be entered.

13. ADDsynth.

1. **Enabled**. Enables this synth type to be used in the part/instrument. When enabled, its marker color, red, is shown.
2. **Edit**. Brings up the editing dialog presented in figure 105 ("ADDsynth Edit/Global Dialog") on page 115. There one will find a full discussion of that dialog.

14. SUBsynth.

1. **Enabled**. Enables this synth type to be used in the part/instrument. When enabled, its marker color, blue, is shown.

2. **Edit**. Brings up the editing dialog presented in figure 138 ("SUBsynth Edit Dialog") on page 149. There one will find a full discussion of that dialog.

15. PADsynth.

1. **Enabled**. Enables this synth type to be used in the part/instrument. When enabled, its marker color, green, is shown.
2. **Edit**. Brings up the editing dialog presented in figure 116 ("PADsynth Edit Dialog") on page 135. There one will find a full discussion of that dialog.

16. Kit Edit. Brings up the editing dialog presented in figure 142 ("Kit Edit Dialog") on page 154. There one will find a full discussion of that dialog.

17. Effects. Brings up the editing dialog presented in figure 85 ("Effects Edit, None") on page 88There one will find a full discussion of that dialog.

18. Rnd. Det.. Small Random Detune.

Values: `0* to 20`

This value is an experimental feature. It would lend some complexity or piquancy to the part/instrument.

19. Close. Closes the Edit window.

8 ADDsynth

The *Yoshimi* ADDsynth (also spelled "ADsynth") dialog is a complex dialog for creating an instrument. This is the most complex, most advanced and most sophisticated part of the synthesizer and allows one to edit the parameters that apply to all the voices of ADDsynth.

ADDsynth, a primarily additive synthesis engine, is one of the three major synthesis engines available in ZynAddSubFX. The basic concept of this engine is the summation of a collection of voices, each of which consists of oscillators.

"ADDsynth" (sometimes spelled "ADsynth") or "ADnote" is a complex engine which makes sounds by adding a number of voices. Each one has filters, envelopes, LFOs, morphing, modulation, resonance, etc. Each voice includes a very powerful waveform generator with up to 128 sine/non-sine harmonics. One can use Fourier synthesis, or if one doesn't like it, one can use wave-shaping/filtering of functions. This engine includes anti-aliasing. Modulation includes ring modulation, phase modulation, and more. The modulators can have any shape. [22]

The sum of the voices are passed through filters and amplification to produce the final sound. This could lead one to think that ADDsynth is just a bunch of minor post-processing, and at this level much of the sound generation is hidden.

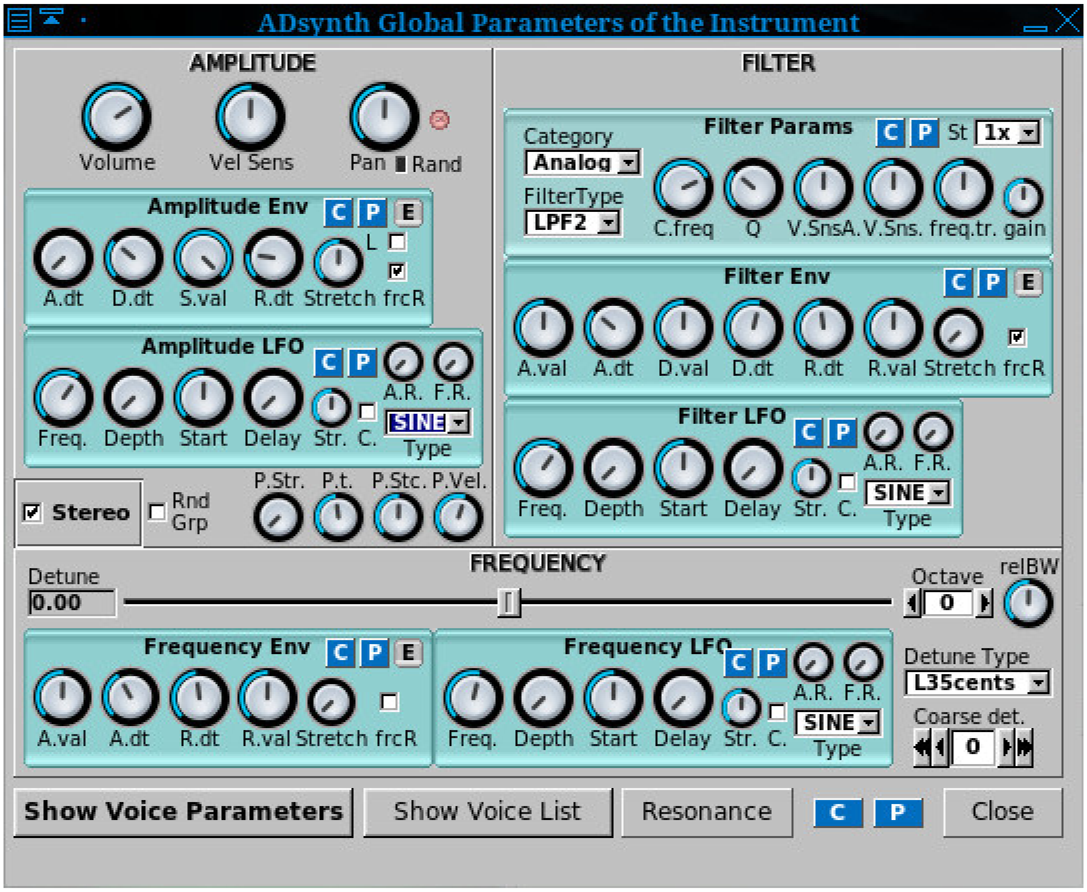

Figure 105: ADDsynth Edit/Global Dialog

The major sections of this dialog are listed:

1. **AMPLITIUDE** (stock section)
2. **FILTER** (stock section)
3. **FREQUENCY** (stock section)
4. **Show Voice Parameters** (section)
5. **Show Voice List** (section)
6. **Resonance** (stock section)
7. **C**
8. **P**
9. **Close**

This complex dialog is best described section by section. Many of the sub-sections are stock sub-panels described elsewhere in this document.

8.1 ADDsynth / AMPLITUDE

1. **Volume**
2. **Vel Sens**
3. **Pan**
4. **Rand**

5. **Reset (panning)** (red button)
6. **Amplitude Env** The Amplitude Env panel is described in detail in section 4.4.1 ("Amplitude Envelope Sub-Panel") on page 67.
7. **Amplitude LFO** The Amplitude LFO panel is described in detail in section 4.3.5 ("LFO User Interface Panels") on page 63.
8. **Stereo**
9. **Rnd Grp**
10. **P.Str.**
11. **P.t**
12. **P.Stc.**
13. **P.Vel.**

Note the two sub-panels, mentioned above, that are described elsewhere. They will not be discussed in detail below.

1. Volume. ADDsynth Volume.

Values: `1 to 127, 64*`

Sets the overall/relative volume of the instrument.

2. Vel Sens. ADDsynth Velocity Sensing function. Turn the knob rightmost/maximum to disable this function.

Values: `1 to 127, 64*`

Velocity sensing is simply an exponental transformation from the notes velocity to some parameter change. Observe figure 106 ("Velocity Sensing Function") on page 116. It shows how the velocity sensing controls affects the translation of a parameter over the whole range of possible note velocities.

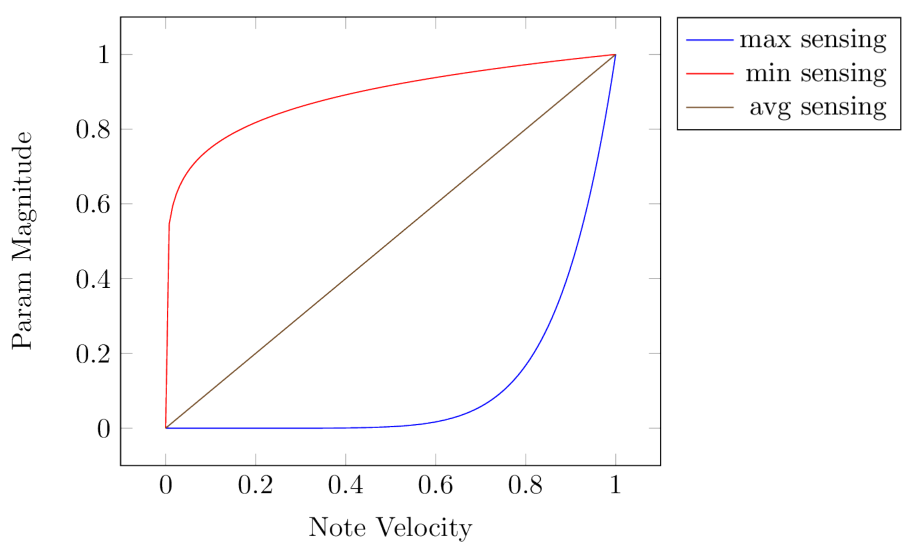

Figure 106: Velocity Sensing Function

3. Pan. ADDsynth Global Panning.

Values: `0, 1 to 127, 64*`

Dialing the knob to leftmost or zero gives random panning.

4. Rand. ADDsynth Random Panning Indicator. A red fill-in color provides an indicator for the activation of random panning in this control.

5. Reset (panning). ADDsynth Reset Panning (red button). Clicking this red button changes the panning value to 64 (centered).

Next, we skip the **Amplitude Env** and **Amplitude LFO** panels, which are described elsewhere, as noted above.

6. Stereo. ADDsynth Stereo.

Values: `Off, On*`

Stereo can be enabled. When disabled, all the voices will also have panning disabled.

7. Rnd Grp. ADDsynth Random Group.

Values: `Off*, On`

How the harmonic amplitude is applied to voices that use the same oscillator.

TODO: Get a more detailed explanation of what this setting means.

8. P.Str.. ADDsynth Punch Strength.

Values: `0* to 127`

The punch strength of a note in ADDsynth is a constant amplification to the output at the start of the note, with its length determined by the punch time and stretch and the amplitude being determined by the punch strength and velocity sensing. The **relBW** control in the frequency panel is effectively a multiplier for detuning all voices within an ADnote.

9. P.t. ADDsynth Punch Time (duration).

Values: `0 to 127, 64*`

Sets the punch effect duration (from 0.1 ms to 100 ms on an A note, 440Hz).

10. P.Stc.. ADDsynth Punch Stretch.

Values: `0 to 127, 64*`

Sets the punch effect stretch according to frequency. On lower-frequency notes, punch stretch makes the punch effect last longer.

11. P.Vel.. ADDsynth Punch Velocity Sensing.

Values: `0 to 127, 72*`

The higher this value, the higher the effect of velocity on the punch of the note.

8.2 ADDsynth / FILTER

The ADDsynth FILTER block consists solely of sub-panels described in detail in the sections noted below. The sub-panels of the FILTER section are:

1. **Filter Params**
2. **Filter Env**
3. **Filter LFO**

1. Filter Params. ADDsynth Filter Parameters. The Filter Params panel is described in detail in section 4.2.5 ("Filter Parameters User Interface") on page 59.

2. Filter Env. ADDsynth Filter Envelope. The Filter Env panel is described in detail in section 4.4.5 ("Envelope Settings for Filter") on page 72.

3. Filter LFO. The Filter LFO panel is described in detail in section 4.3.5 ("LFO User Interface Panels") on page 63.

8.3 ADDsynth / FREQUENCY

1. **Detune**
2. **FREQUENCY slider**
3. **Octave**
4. **RelBW**
5. **Frequency Env**. A stock sub-panel described in section 4.4.4 ("Envelope Settings, Frequency") on page 71.
6. **Frequency LFO** A stock sub-panel described in section 4.3.7 ("Frequency LFO Sub-panel") on page 65.
7. **Detune Type**
8. **Coarse det.**

1. Detune. ADDsynth Detune Value. This display box shows the value of the detune as selected by the frequency slider described below.

Figure 107: ADDsynth Frequency Detune Type

Values: `Default*, L35cents, L10cents, E100cents, E1200cents`

This value defines the number of cents that define the range of the **FREQUENCY** slider, that is 35 cents, 10 cents, 100 cents (one semitone), or 1200 cents (1 octave), below and above the main frequency. The default is 35 cents. The 1200-cents setting provides a whole octave of detuning in either direction.

The "L" probably stands for "linear", and the "E" for "exponential", to describe how the detune slider acts.

2. FREQUENCY slider. ADDsynth Fine Detune (cents), a slider control.

Values: `-35.00 to 35.00`, `-10.00 to 10.00`, `-100.00 to 100.00`, `-1200.00 to 1200.00`

While the detune type dropdown and the octave selection provide a coarse selection of detune, the slider allows for a finer selection of detune, up to roughly one-third of a semitone.

3. Octave. ADDSynth Octave.

Values: `-8 to 7, 0*`

The octave setting changes the frequency by octaves.

4. RelBW. ADDSynth Relative Bandwidth.

Values: `0 to 127, 64*`

Bandwidth: how the relative fine detune of the voice is changed.

5. Frequency Env. ADDsynth Frequency Envelope. The Frequency Env panel is described in detail in section 4.4.4 ("Envelope Settings, Frequency") on page 71.

6. Frequency LFO. The Frequency LFO panel is described in detail in section 4.3.5 ("LFO User Interface Panels") on page 63

7. Detune Type. Frequency Detune Type.

Values: `L35cents, L10cents, E100cents, E1200cents`

This setting provides a coarse detuning. We would welcome an explanation of exactly is meant by the numbers and the "E" versus "L" designation.

8. Coarse det.. Coarse Detune, "C.detune".

Values: `-64 to 63, 0*`

The one-arrow buttons change the value by one. The two-arrow buttons change the value by ten.

Again, we need a way to explain the interactions of the slider, the octave setting, the detune type, and the coarse detune settings.

9. Show Voice Parameters. ADDsynth Show Voice Parameters. This button brings up the "voice parameters" dialog discussed in the next section. Again, this dialog is built from some stock sections and stock sub-panels, plus additional elements.

8.4 ADDsynth / Voice Parameters

Each *Yoshimi* ADDsynth instrument consists of up to 8 voices. This dialog provides a way to define each of the 8 voices in great detail.

By default, an instrument consists of one voice, voice 1.

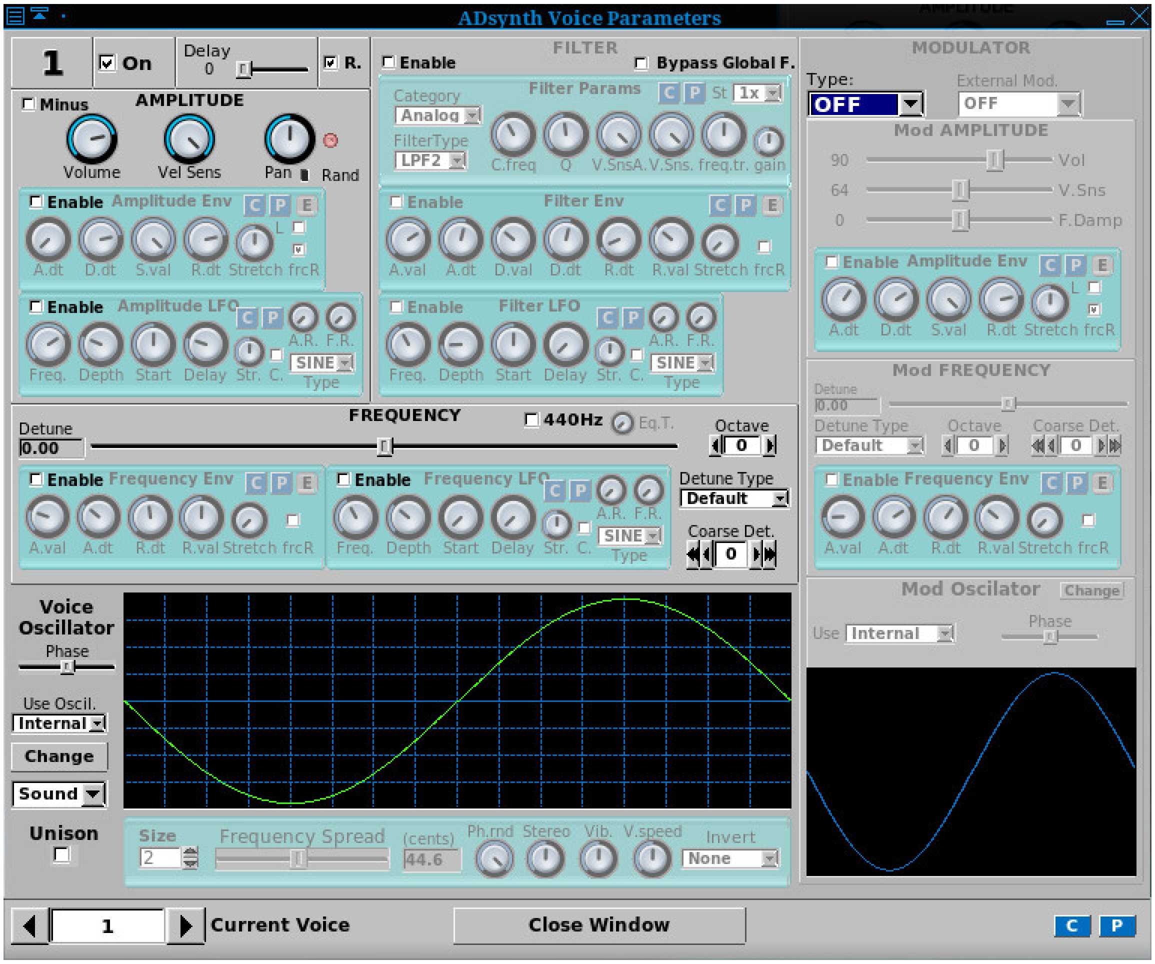

Figure 108: ADDsynth Voice Parameters Dialog

This dialog consists of a few extra settings, plus a number of stock dialog sections. Take some time to compare figure 105 ("ADDsynth Edit/Global Dialog") on page 115, which covers the overall instrument, with figure 108 ("ADDsynth Voice Parameters Dialog") on page 120, which covers each of the voices. The stock sections in the former cover the whole instrument as one, while the very similar stock sections in the latter cover only the voice they configure. Obviously, the combinations of settings are essentially endless.

Each voice can be amplitude-controlled, filter-controlled, and frequency-controlled. Each voice can also be modulated by an internal modulator. Another property of the voice is that one can tell *Yoshimi* to modulate a given voice with any of the voices that are numbered less than that voice.

1. **Voice Number**
2. **On**
3. **Delay**
4. **R.**
5. **AMPLITUDE** (see section below)
6. **FILTER** (see section below)
7. **MODULATOR** (see section below)
8. **FREQUENCY** (see section below)

1. Voice Number. ADDsynth Voice Number.

Values: `1* to 8`

This display element shows the voice number represented by the settings in this dialog. Each *Yoshimi* part/instrument can consist of up to eight voices.

The voice being worked on can be selected using the **Current Voice** selector.

2. On. ADDsynth Voice On/Off.

Values: `Off, On`

Enables this voice in the part/instrument.

3. Delay. ADDsynth Voice Delay.

Values: `0* to 127`

TODO: We still need to determine what the units of the delay are.

Bug: The tooltip for this setting says "Volume".

4. R.. ADDsynth Voice Resonance On/Off.

Values: `Off, On*`

The rest of the GUI elements (AMPLITUDE, FILTER, MODULATOR, FREQUENCY, and Voice Oscillator) are more detailed, and discussed in the sections that follow.

8.4.1 ADDsynth / Voice Parameters / AMPLITUDE

This section of the voice parameters dialog also includes a couple of stock sub-panels that have an additional "Enable" control.

1. **Minus**
2. **Volume**
3. **Vel Sens**
4. **Pan**
5. **Pan randomness indicator**
6. **Pan reset** (red button)
7. **Amplitude Env, Stock + Enable**
8. **Amplitude LFO, Stock + Enable**

1. Minus. ADDsynth Amplitude Minus.

Values: `Off*, On`

This setting enables the inversion of the volume control action. It enables negative values for the volume control of the voice.

2. Volume. ADDsynth Voice Volume.

Values: `0 to 127, 90?`

Sets the (relative) volume of this voice in the part/instrument.

3. Vel Sens. ADDsynth Voice velocity-sensing function; setting to rightmost/max disables this function.

Values: `0 to 127*`

4. Pan. ADDsynth Voice panning; setting to leftmost/0 gives random panning.

Values: `0 to 127, 64*`

5. Pan randomness indicator. ADDsynth Voice random panning On/Off. Fills in red to indicate that random panning is in force.

6. Pan reset (red button). ADDsynth Center Panning.

Clicking this small red button resets the panning to center.

7. Amplitude Env, Stock + Enable. ADDsynth Amplitude Envelope Sub-panel. See section 4.4.1 ("Amplitude Envelope Sub-Panel") on page 67. Additionally, the **Enable** checkbox allows the enabling of this component.

8. Amplitude LFO, Stock + Enable. ADDsynth Amplitude LFO Sub-panel. See section 4.3.5 ("LFO User Interface Panels") on page 63. Additionally, the **Enable** checkbox allows the enabling of this component.

8.4.2 ADDsynth / Voice Parameters / FILTER

This section of the voice parameters dialog also includes a couple of stock sub-panels that have an additional "Enable" control.

1. **Enable**
2. **Bypass Global F.**
3. **Filter Params, Stock**
4. **Filter Env, Stock + Enable**
5. **Filter LFO, Stock + Enable**

1. Enable. ADDsynth Voice Enable Filter.

Values: `Off*, On`

This value enables the whole FILTER dialog section.

2. Bypass Global F.. ADDsynth Voice Bypass Global Filter.

Values: `Off*, On`

The voice signal bypasses the global filter.

TODO: Make sure there is a discussion of the global filter.

3. Filter Params, Stock. See section 4.2.5 ("Filter Parameters User Interface") on page 59.

4. Filter Env, Stock + Enable. See section 4.4.5 ("Envelope Settings for Filter") on page 72.

5. Filter LFO, Stock + Enable. See section 4.3.6 ("Filter LFO Sub-panel") on page 64.

8.4.3 ADDsynth / Voice Parameters / MODULATOR

1. **Type:**
2. **External Mod.**
3. **Mod AMPLITUDE**
4. **Mod FREQUENCY**
5. **Mod Oscillator**

1. Type:. ADDsynth Modulator Type.

Values: `OFF(, MORPH, RING, PM, FM, PITCH`

Figure 109: Voice Modulator Type

1. **OFF**. This setting turns off the modulator.
2. **MORPH** The morph modulator works by combining the output of two oscillators into one, with the amplitude envelope translating between one waveform and the other.
3. **RING** The ring modulator is useful for making bell-like sounds and some weird effects. The ring modulator works by multiplying two waveforms together, producing a signal that is the sum and difference of the frequencies present in the waveforms. The ins-and-outs of the ring modulator are explained in detail in paragraph 8.4.3.1.
4. **PM** The PM (pulse modulation) modulator works by using a modulator envelope to change the pulse width of a pulse waveform. Generally, set **F.Damp** to zero, so that the modulation amount doesn't depend on the note number.
5. **FM** The (frequency modulation) morph modulator works by modulator the frequency. Examples can be heard in the "Ethereal" and "Steel Wire" instruments.
6. **PITCH** The pitch modulator works by... we're not sure... TODO: is this pitch shifting?

2. External Mod.. External Oscillator. External Modulator.

Values: `OFF*, Other voice numbers?`

External Oscillator. This feature allows one of the voices (of the up to 8 allowed in a single ADDsynth instrument) to be used as a modulator or external oscillator for another voice in the instrument. This option specifies to use the oscillator of another voice, or -1 for the *internal* oscillator.

The parameters must be lower than the voice index; one cannot use the oscillator from a voice with a bigger index (e.g. one can't use the oscillator of voice 8 for voice 4). This is very useful because, if one uses many voices with the same oscillator settings, one can use only one oscillator and select other voices to use this, and if one changes a parameter of this oscillator, all voices using this oscillator will be affected.

External Modulator. Use another voice as a modulator instead of the modulator of the internal voice. One can make a modulation "stack". The modulator of the voice is disabled.

External. Uses the oscillator as the modulator of another voice. It behaves like **Ext. Oscil**, except that it works on the *modulator*. Please notice the difference between this parameter and **Ext. Mod**. See below.

3. Mod AMPLITUDE. Modulator Amplitude.

1. **Vol** Volume. Values: `0 to 127, 90*`
2. **V.Sns** Velocity Sensing Function; set to rightmost/max to disable. Values: `0 to 127, 64*`
3. **F.Damp** Modulator Damp at higher frequency. How the modulator intensity is lowered according to lower/higher note frequencies. Values: `0 to 127, 90*`

4. **Amplitude Env, Stock + Enable** See section 4.4.1 ("Amplitude Envelope Sub-Panel") on page 67.

4. Mod FREQUENCY. Modulator Frequency.

1. **Detune slider** Fine Detune (cents). Values: `-35.00 to 35.00, 0*`
2. **Detune Type** Fine Detune (cents). Values: `L35cents, L10cents, E100cents, E1200cents` See figure 107 ("ADDsynth Frequency Detune Type") on page 118.
3. **Octave** Octave. Values: `-8 to 7, 0*`
4. **Coarse Det.** Coarse Detune. Values: `-64 to 63, 0*`
5. **Filter Env, Stock + Enable** See section 4.4.5 ("Envelope Settings for Filter") on page 72.

5. Mod Oscillator. Modulator Oscillator.

Bug: The word "oscillator" is mispelled in the application.

1. **Change** ADDsynth Oscillator Editor.
2. **Use** Oscillator to Use. See the paragraph below. Values: `Internal*, Other oscillators?`
3. **Phase** Oscillator Phase. Values: `0 to 360 (0 to 2PI)`
4. **Waveform graph** Waveform graph.

As far as we can tell, one has the choice between **Internal**, which in this case means a completely independent modulator oscillator per voice (extra change button), or **External**, which refers to the modulation oscillators one has already defined for the voices with a lower index. This means one can make one modulation oscillator for voice 1, and reuse it in voices 2 and 3. This is the same system used for the normal oscillators.

8.4.3.1 Tip: Using the Ring Modulator

This section is derived from one of the short text files in the *Yoshimi* source-code bundle ([16] or [17]). It notes that "Some people have been confused about how to use an 'external' Mod Oscillator", and provides usage notes that we will elaborate on here. Here is the way to use the ring modulator:

1. Open the ADDsynth editing window. Then open **Show Voice Parameters**.
2. For **Type**, select the **RING** value. This selection will activate the **Mod Oscillator**.
3. In the **Mod Oscillator**, click on **Change** to open the **ADDsynth Oscillator Editor**.
4. Set the wave-shape to **Triangle**.
5. Switch to voice number 2 and enable it.
6. Again, for **Type**, select the **RING** value. However, feel free to select one of the other modulators, if one wishes.
7. One can now use **Internal** for voice 2, or select **ext.m** 1, to use the first voice as in internal modulator.
8. Change the internal voice to, for example, **Square**.
9. Do the same setup for voice 3. One will find that one can use its **Internal** or either of the two previous ones.

Now the joker in the pack is that one can disable both the previous voices but *still* use their Mod Oscillators.

What is going on here? Need to explain better!!!!!

In a newsgroup ([11], the following note is found.

> Say I want the A tone ring-modulated by 880Hz. A is 440 Hz, the ring modulation setting lets me choose the modulation frequency relative to the frequency of the tone. So I choose

octave 1 and let the detune at zero. If I move the detune, it'll shift the modulation frequency a bit, which will make a disharmonic effect.

Wet/dry setting is controlled by volume in "modulation amplitude". The modulation frequency can further be multiplied or several modulations can be simulated by changing the oscillator waveform.

One huge letdown is that it is only available for Adsynth. PadSynth does not seem to have ring modulation option, so the coolest sounds stay out of question for massive lead tones. :-(

8.4.4 ADDsynth / Voice Parameters / FREQUENCY

This frequency section is almost a stock part. It is similar to the ADDsynth Edit's **FREQUENCY** section.

1. **Detune**
2. **FREQUENCY slider**
3. **440Hz**
4. **Eq.T.**
5. **Octave**
6. **Detune Type**
7. **Coarse det.**
8. **Frequency Env, Stock + Enable**
9. **Frequency LFO, Stock + Enable**
10. **Voice Oscillator**

1. Detune. Voice Parameters Detune. Shows the value selected by the frequency slider.

2. FREQUENCY slider. Frequency Slider. Provides fine detune, in cents.

Values: `-35.00 to 35.00, 0*`

Note that 35 cents is roughly one-third of a semitone.

3. 440Hz. 440 Hz Selection.

Values: `Off*, On`

Fixes the voice base frequency to 440 Hz. One can adjust this with the detune settings. No matter what key is played on the keyboard, this voice will emit only 440 Hz. This is useful for defining a constant frequency to use as a modulator for the other voices in the part.

For example, one can define voice 1 to be a tone, then define voice 2 to be 440 Hz. The two voices will mix, but only voice 1 will change frequencies as different keys are played.

4. Eq.T.. Equal Temperament?

Values: `0 to 127?`

This item is enabled if the **440Hz** check-box is enabled. It determines how the frequency varies according to the keyboard. If set to its leftmost (0) value, then the frequency is fixed.

5. Octave. Voice Parameters Octave.

Values: `-8 to 7, 0*`

6. Detune Type. Detune Type.

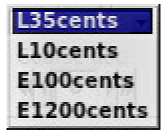

Figure 110: Frequency Detune Type

Values: `L35cents, L10cents, E100cents, E1200cents`

7. Coarse det.. Coarse Detune.

Values: `-64 to 63, 0*`

TODO: Is this setting in units of semitones?

8. Frequency Env, Stock + Enable. Frequency Envelope. See section 4.4.4 ("Envelope Settings, Frequency") on page 71.

9. Frequency LFO, Stock + Enable. Frequency LFO. See section 4.3.7 ("Frequency LFO Subpanel") on page 65.

10. Voice Oscillator. Voice Parameters Oscillator. See the next section.

8.4.5 ADDsynth / Voice Parameters / Voice Oscillator

1. **Phase**
2. **Use**
3. **Waveform graph**
4. **Change**
5. **Sound**
6. **Unison**
7. **Current Voice**
8. **C**
9. **P**
10. **Close Window**

1. Phase. Voice Oscillator Phase.

Values: `0 to 360 (0 to 2*pi)`

2. Use Oscil.. Use Oscillator.

Values: `Internal*, Other oscillators`

If the **Current Voice** is set to a value greater than 1, meaning that one is editing additional voices, then this dropdown item also includes the values of all oscillators less then this one, marked as "External". For example, if one is currently editing current voice 3, then the dropdown list includes **Internal**, **Ext. 1**, and **Ext. 2**.

3. Waveform graph. Waveform Graph. Shows a period of the currently configured oscillator.

4. Change. Voice Oscillator Change.

This button brings up the ADDsynth Oscillator Editor dialog.

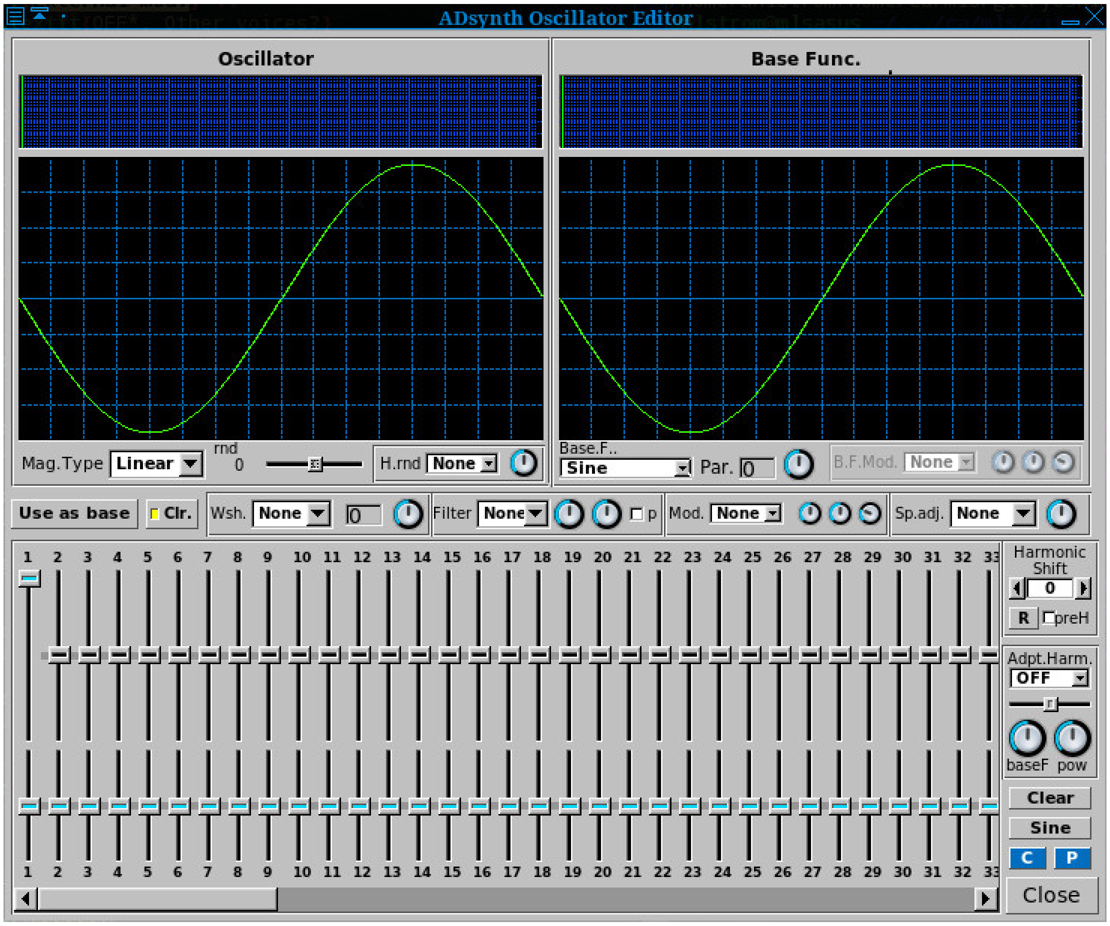

Figure 111: ADDsynth Oscillator Editor

This item is nearly identical to the PADsynth harmonic editor depicted in figure 129 ("Harmonic Content Editor") on page 141. Obviously, it is a topic unto itself! Let us continue with rest of the minor controls.

5. Sound. Oscillator Type (sound/noise). Sound/Noise choice. Select the mode of the oscillator (sound versus white noise).

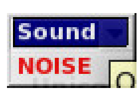

Figure 112: Voice Oscillator Choices

Values: `Sound*, NOISE`

If **NOISE** is selected, then the waveform graph simply announces "White Noise" (not shown here). Also, the **Unison** control is disabled.

6. Unison. Unison is useful in creating the chorus like sound of many simultaneous oscillators.

Values: `Off*, On`

Enabling this item causes the following Unison-related items to become enabled.

1. **Size**

2. **Frequency Spread**
3. **Ph.rnd**
4. **Stereo**
5. **Vibrato**
6. **V.speed**
7. **Invert**

1. Unison Size. Sets the number of unison sub-voices.

Values: `2* to 50`

2. Unison Frequency Spread. Frequency spread of the unison (cents).

Values: `0 to 200, 44.6*`

3. Phase Randomness. Unison Phase Randomness.

Values: `0 to 127*`

4. Stereo Spread. Unison Stereo Spread.

Values: `0 to 127, 64*`

5. Unison Vibrato. Unison Vibrato.

Values: `0 to 127, 64*`

6. Vibrato Speed. Unison Vibrato Average Speed.

Values: `0 to 127, 64*`

7. Phase Invert. Unison Phase Invert. Values: `None*, Random, 50%, 33%, 25%, 20%`

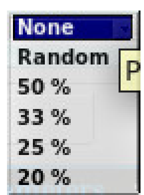

Figure 113: Phase Invert Dropdown

Finally, at the bottom of the ADDsynth Voice Part dialog, we find the last few controls.

1. Current Voice. Current Voice.

Values: `1* to 8`

2. C. Copy D note parameters ("DnoteParameters").

3. P. Paste D note parameters ("DnoteParameters").

4. Close Window. Close.

8.5 ADDsynth / Voice List

The ADDsynth Voices List shows a summary of voices 1 to 8, and allows some overall control of them, almost like a simple mixer. It is brought on-screen via the **Show Voice List** button of the ADDsynth global part editor.

Figure 114: ADDsynth Voices List

1. **No. (1 to 8)**
2. **Waveform**
3. **Vol**
4. **Pan**
5. **R.**
6. **Detune**
7. **Vib. Depth**
8. **Hide Voice List**

1. No. (1 to 8). Voice List Number.

Values: `Off, On`

This check-box enables or disables a given voice in the current part.

2. Waveform Icon. Waveform Icon. The waveform icon shows a rough rendering of the actual shape of the voice waveform, or the letter **N** is the voice is constructed from white noise. Note that this picture isn't updated, if the voice is edited, until the voice list is closed and reopened.

3. Vol. Voice Volume.

Values: `0 to 127, 100*`

This slider controls the relative volume of a given voice in the current part.

4. Pan. Voice Panning (0/leftmost is Random).

Values: `0 to 127, 64*`

This slider controls the panning of a given voice in the current part.

5. R.. Resonance On/Off.

Values: `Off, On*`

Enable/disable the resonance effect of a voice. Note that the resonance is configured in by the Resonance dialog brought up by the **Resonance** button at the bottom of the ADDsynth main dialog. The resonance dialog has its own Enable button, as well. These seem to be independent settings.

6. Detune Value. This read-only text-box shows the current value of detune as selected by its slider.

7. Detune. Fine Detune (cents).

Values: `-35 to 35, 0*`

8. Vib. Depth. Frequency LFO Amount/Depth.

Values: `0 to 127, 40*`

This setting can be very useful because, with the detune settings, one can create very good sounding instruments.

9. Hide Voice List. Hide Voice List. A Close button, really, and that is what it is in the latest version of *Yoshimi*.

8.6 ADDsynth / Resonance

The ADDsynth resonance editor is brought on-screen via the **Resonance** button of the ADDsynth global part editor.

The resonance effect acts as a "resonance box" or a filter with arbitrary frequency response. This produces very realistic sounds. The cursor location is shown below the graph (the frequency, kHz, and the amplitude, dB).

Paul Nasca has a video on YouTube that includes a demonstration of how the resonance dialog works and affects the sound, if you care to look for it.

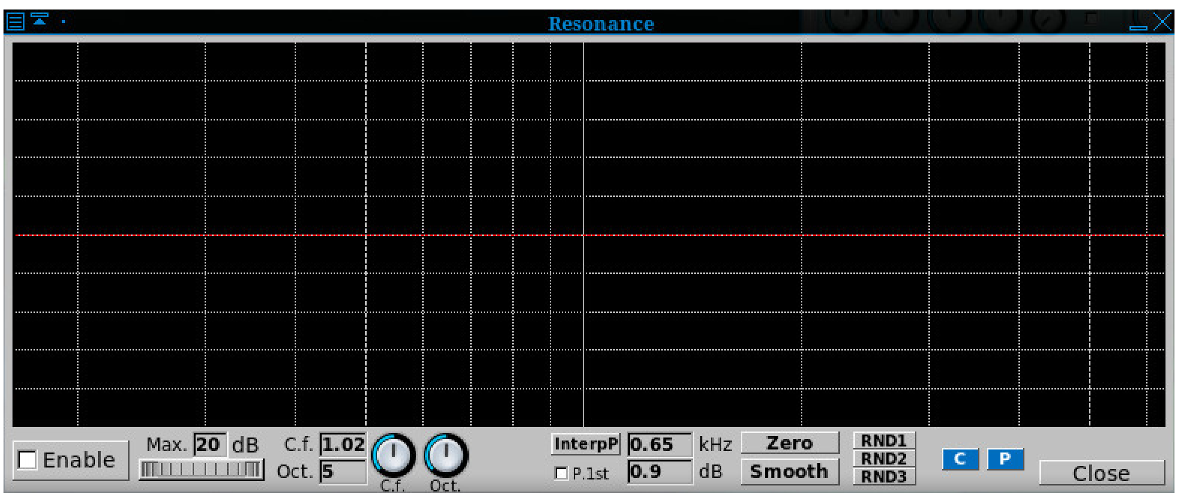

Figure 115: ADDsynth Resonance

1. **Graph Window**
2. **Enable**
3. **Max dB (wheel)**
4. **C.f. (knob)**
5. **Oct.**
6. **P.1st**
7. **InterpP**
8. **KHz**
9. **dB**

10. **Zero**
11. **Smooth**
12. **RND1**
13. **RND2**
14. **RND3**
15. **C**
16. **P**
17. **Close**

1. Graph Window. Resonance Graph Window. Lets one draw the resonance frequency response in "freehand" mode.

2. Enable. Resonance Enable.

Values: `Off*, On`

Turn the Resonance effect on.

3. Max dB (wheel). The Maximum Amplitude (dB) wheel.

Values: `1 to 90, 20*`

Sets the amount of resonance: lower values have little effect. Use the roller below to set it.

4. C.f. (knob). Center Frequency (kHz).

Values: `0 to 127, 64*` for `0.10 to 10.0, 1.0*`

Sets the center frequency of the graph. The value is shown in the read-only text-box to the left.

5. Oct.. Number of Octaves.

Values: `0 to 127, 64*` for `0 to 10, 5*`

Sets the number of octaves the graph represents. The value is shown in the read-only text-box to the left.

6. P.1st. Protect the fundamental Frequency.

Values: `Off, On`

Do not damp the first harmonic.

TODO: What does this mean, what affect does it have?

7. InterpP. Interpolate the peaks. This setting is a weird one where mouse movement affects it, but also affects the next field as well. Oh, kHz and dB.

This setting allows one to make resonance functions very easily. To use it effectively, first, clear the graph using the **Zero** button. Click the left button on a position on the graph. Click the **InterpP** button. It will interpolate automatically between the positions pointed to (or drew). Also one can clear a part of the graph by dragging with the right mouse button. In fact, the **interpP** button interpolates between non-zero values. If one presses the **InterpP** with the right mouse button, the interpolation will be linear, and if one uses the left button, the interpolation will be smooth.

8. KHz. The current frequency on graph.

9. dB. The current level on graph window.

Values: `-90 to +90`

10. Zero. Clear the resonance function. Zero. Clear the graph.

Amplification - how the output signal is amplified (WHERE?)

11. Smooth. Smooth the resonance function. Smooth the graph.

TODO: What is the exact nature of this smoothing?

12. RND1. Randomize the resonance function, 1. RND1, RND2, RND3 are used to create random resonance functions.

TODO: Why three buttons?

13. RND2. Randomize the resonance function, 2.

14. RND3. Randomize the resonance function, 3.

15. C. Copy Dialog.

16. P. Paste Dialog.

17. Close. Close.

9 PADsynth

The *Yoshimi* PADsynth dialog is a complex dialog for creating a pad instrument, "PADsynth" or "PADnote" is engine that makes very beautiful pads and other instruments. (These instruments can be exported for use with other programs too).

The PADsynth dialog consists of two major tabs, "Harmonic Structure" and "EnvelopesLFOs". Each of these tabs is fairly complex, so the discussion will break the tabs down by sub-sections.

9.1 PADsynth / Algorithm

9.1.1 PADsynth / Algorithm / General

This algorithm generates very beautiful sounds, even if its idea is much simpler than other algorithms. It generates a perfectly looped wave-table sample which can be used in instruments. It easily generates sounds of ensembles, choirs, metallic sounds (bells) and many other types of sound. Paul Nasca wanted to make this algorithm known, and everyone is welcome to learn and use this algorithm into one's projects or products (non-commercial or commercial).

Quote [22]:

> You will not be disappointed by this algorithm.
>
> I hope that this algorithm will be implemented in many software/hardware synthesizers. Use it, spread it, write about it, create beautiful instruments with it. If your synthesizer uses plenty of samples, you can use this algorithm to generate many ready-to-use samples.
>
> This algorithm, this page, the images, the implementations from this page, the audio examples, the parameter files from this page are released under Public Domain by Nasca Octavian Paul. e-mail: zynaddsubfx AT yahoo DOT com

In order to understand how this algorithm works, one needs to be familiar with howto think about musical instruments. Please read an introduction for the description of the meaning and the importance of bandwidth of each harmonic and randomness.

This algorithm generates some large wave-tables that can be played at different speeds to get the desired sound. This algorithm describes only how these wave-tables are generated. The result is a perfectly looped wave-table. Unlike other synthesis methods, which use the Inverse Fast Fourier Transform, this one does not use overlap/add methods and there is only one IFFT for the whole sample.

The basic steps are:

1. Make a very large array that represents the amplitude spectrum of the sound (all default values are zero).
2. Generate the distribution of each harmonic in the frequency spectrum and add it to the array.
3. Put random phases to each frequency of the spectrum.
4. Do a single Inverse Fourier Transform of the whole spectrum. There is no need of any overlapping windows, because there is only one single IFFT for the whole sample.

The output is a sample which can be used as a wave-table In the next image, the steps are represented graphically:

TODO: A GRAPHIC

9.1.2 PADsynth / Algorithm / Harmonic Bandwidth

We consider one harmonic (overtone) as being composed of many frequencies. These sine components of one harmonic are spread over a certain band of frequencies. Higher harmonics have a wider bandwidth. In natural choirs/ensembles the bandwidth is proportional to the frequency of the harmonic.

Here is an example of a spectrum of an instrument generated by ZynAddSubFX:

TODO: A GRAPHIC, full spectrum, closeup of the spectrum

The harmonics becomes wider and wider, until a certain frequency, where they may merge into a noise band (as in the full spectrum image from above shows). This is a normal thing and we recommend to not avoid this by limiting the bandwidth of the harmonics.

The frequency distribution of one harmonic/overtone (or the harmonic profile).

This describes the function of the spread of the harmonic. Here are some examples of how they can be spread:

a) A special case is where there is only a single sine component inside the harmonic In this case, the harmonic and the "sine component" are the same thing.

b) Detuned. In this case there are two sine components which are detuned.

c) Evenly spread inside the harmonic (all components has the same amplitude)

d) Normal (Gaussian) distribution. The sine components amplitude are bell shaped. The largest amplitude is in the center of the band. This distribution gives the most natural sounds (it simulates a very, very large ensemble).

Of course, one can use many other profiles of the harmonic. ZynAddSubFX's PADsynth module offers many ways to generate the harmonic profile. Also, it's very important that the harmonic must have the same amplitude, regardless of the profile functions/parameters and the bandwidth.

For many more details of this algorithm, see Paul Nasca's document [22].

9.1.2.1 Tip: Using the PADsynth

Keep in mind that the resulting wave-tables are perfectly looped. When using the wave-tables for instruments, on each NoteOn, start from a random position and not from the start. This avoids hearing the same sound on each keystroke.

One can use the same wave-table for generating stereo sounds, by playing the same wave-table at different positions for left and right. The best is to create a difference between left right of N/2.

Generate different wave-tables for different pitches and use the one that is closest to the desired pitch.

Upsample or downsample the amplitude array of the harmonic before running the algorithm, according to the fundamental frequency. In this case we need to set a parameter "base_frequency" which represents the frequency where the array is left unchanged.

Example: We have A_orig[]=1,2,1,3,0,0,1,0 and base_frequency is equal to 440 Hz Here are some cases:

A[] for 440 Hz: is the same as A_orig[]

A[] for 220 Hz: is the A_orig[] upsampled by factor of 2

so: A[]=1, 1, 1.5, 2, 1.5, 1, 2, 3, 1.5, 0, 0, 0, 0.5, 1, 0.5, 0

(the original A_orig amplitudes are shown as bold)

A[] for 880 Hz: the A_orig[] is downsampled by a factor of 2

so: A[]=1.5, 2, 0, 0.5

A[] for F Hz: the A_orig[] is scaled by a factor of 440/F.

Even if this idea is very simple, the resulting sounds are very natural, because it keeps the spectrum constant according to the frequency of the harmonic and not to the number of the harmonic. This follows the point 4 from the document where I described some principles regarding synthesis.

9.2 PADsynth / Harmonic Structure

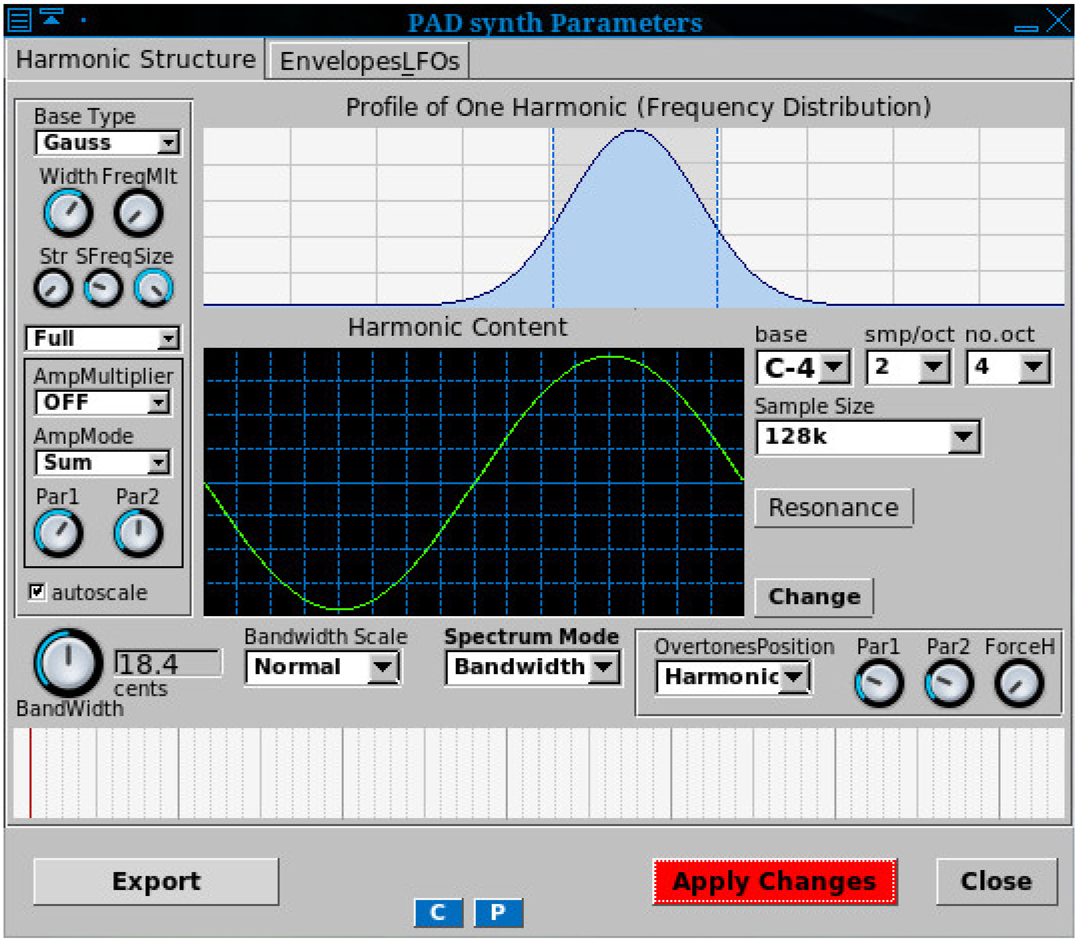

Figure 116: PADsynth Edit Dialog

1. **Basics** (section)
2. **Harmonic** (section)
3. **Resonance** (section)
4. **Change** (section)
5. **Bandwidth and Position** (section)
6. **Export** (section)
7. **C**
8. **P**
9. **Apply Changes**
10. **Close**

9.2.1 PADsynth / Harmonic Structure / Basics

1. **BaseType**
2. **Width**
3. **FreqMlt**

4. **Str**
5. **SFreq**
6. **Size**
7. **Full/Upper/Lower**
8. **AmpMultiplier**
9. **AmpMode**
10. **Par1**
11. **Par2**

1. BaseType. Base Type of Harmonic.

Figure 117: Base Type of Harmonic

Values: `Gauss*, Square, DoubleExp`

2. Width. Width of Harmonic.

Values: `1 to 127?`

3. FreqMlt. Frequency Multiplier.

Values: `1 to 127?`

4. Str. Stretch.

5. SFreq. Harmonic Sfreq?

6. Size. Harmonic Size.

7. Full/Upper/Lower. Harmonic Spread???

Figure 118: PADsynth Full/Upper/Lower Harmonics

Values: `Full*, Upper Half, Lower Half`

8. AmpMultiplier. Amplitude Multiplier.

Figure 119: PADsynth Amplitude Multiplier

Values: `OFF*, Gauss, Sine, Flat`

9. AmpMode. Amplitude Mode.

Figure 120: PADsynth Amplitude Mode

Values: `Sum*, Mult, Div1, Div2`

10. Par1. Harmonic Parameter 1?

Values: `0 to 127?`

11. Par2. Harmonic Parameter 2?

Values: `0 to 127?`

9.2.2 PADsynth / Harmonic Structure / Harmonic

1. **Profile of One Harmonic**
2. **Harmonic Content Window**
3. **base**
4. **smp/oct**
5. **no.oct**
6. **Sample Size**
7. **Resonance** (section)
8. **Change** (section)

1. Profile of One Harmonic. Profile of One Harmonic (Frequency Distribution).

2. Harmonic Content Window. Harmonic Content Window.

3. base.

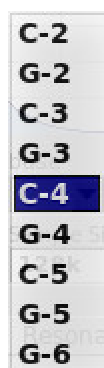

Figure 121: Harmonic Base Dropdown

Values: `C-2, G-2, C-3, G-3, C-4*, G-4, C-5, G-5, G-6`

4. smp/oct. Harmonic Samples Per Octave?

Figure 122: Harmonic Samples Per Octave

5. no.oct. Number of Octaves of Harmonic.

Figure 123: Harmonic Number of Octaves

Values: `1, 2, 3, 4*, 5, 6, 7, 8`

6. Sample Size. Harmonic Sample Size.

Figure 124: Harmonic Sample Size Dropdown

Values: `16k (Tiny), 32k, 64k (Small), 128k*, 256k (Normal), 512k, 1M (Big)`

9.2.3 PADsynth / Harmonic Structure / Bandwidth and Position

1. **BandWidth**
2. **cents**
3. **Bandwidth Scale**
4. **Spectrum Mode**
5. **OvertonesPosition**
6. **Par1**
7. **Par2**
8. **ForceH**

9. **Harmonics Plot**

1. BandWidth. Harmonics Bandwidth.

Values: `0 to 127?`

2. cents. Bandwidth Reading (cents).

3. Bandwidth Scale. Bandwidth Scale.

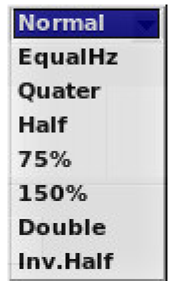

Figure 125: Harmonics Bandwidth Scale.

Values: `Normal, EqualHz, Quater, Half, 75%, 150%, Double, Inv. Half`

4. Spectrum Mode. Harmonics Spectrum Mode.

Figure 126: PADsynth Harmonics Spectrum Mode

Values: `Bandwidth*, Discrete, Continuous`

5. OvertonesPosition. Overtones Position.

Figure 127: PADsynth Overtones Position

Values: `Harmonic*, ShiftU, ShiftL, PowerU, PowerL, Sine, Power`

6. Par1. PADSynth Bandwidth Parameters 1?

7. Par2. PADSynth Bandwidth Parameters 2?

8. ForceH. PADSynth Bandwidth ForceH.

9. Harmonics Plot. PADSynth Harmonics Plot.

9.2.4 PADsynth / Harmonic Structure / Export

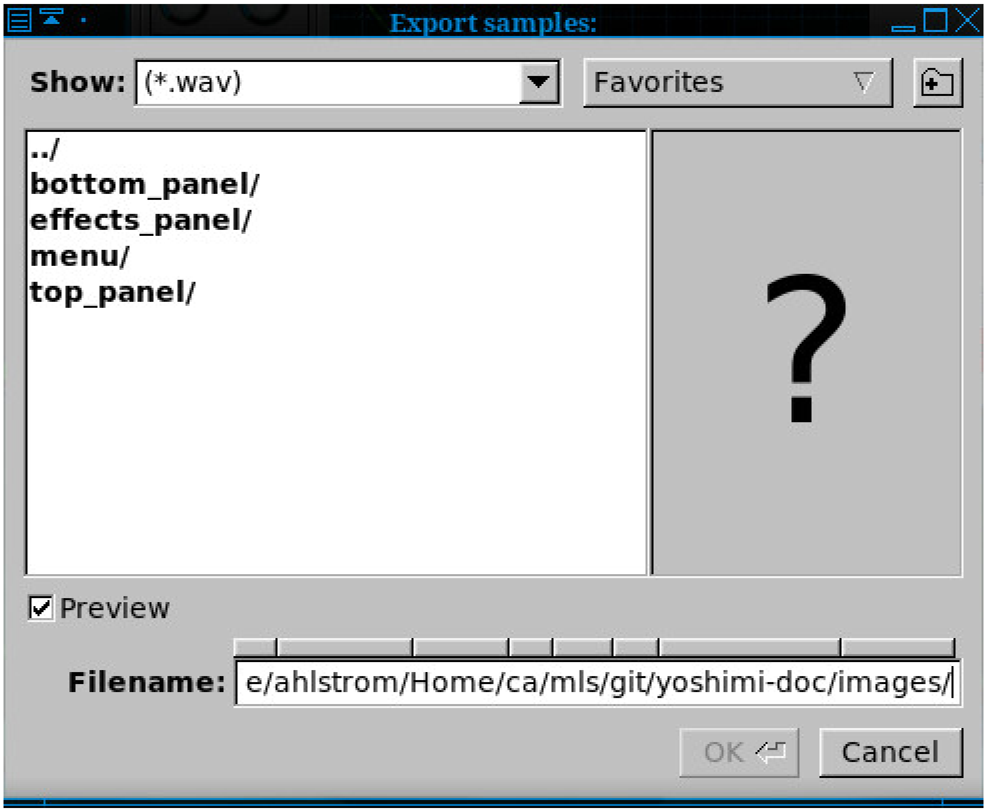

Figure 128: Harmonics Structure Export Dialog

This export dialog is a file dialog similar to other file dialogs, such as that shown in section ....

9.2.5 PADsynth / Harmonic Structure / Resonance

The PADsynth Harmonics resonance dialog is identical to the resonance dialog described in section ....

Also see this image file: images/bottom-panel/instrument-edit/PAD/resonance.jpg. It shows something that the ADDsynth version doesn't... an "Apply" button.

9.2.6 PADsynth / Harmonic Structure / Change

Harmonic Content Editor. Another complex dialog. Like figure 111 ("ADDsynth Oscillator Editor") on page 127, it allows one to create an unlimited number of oscillators.

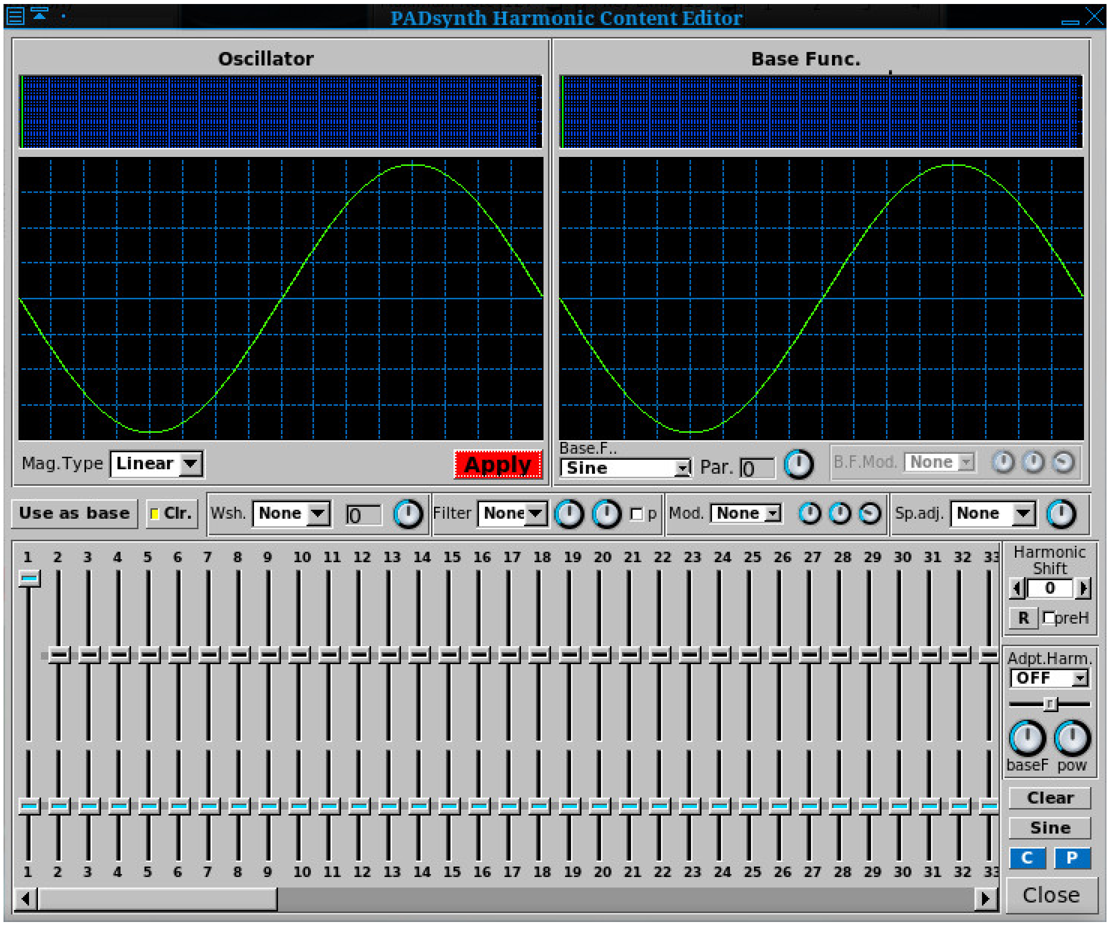

Figure 129: Harmonic Content Editor

This dialog is complex enough that it makes sense to break it down into sub-sections.

1. **Oscillator** (section)
2. **Base Function** (section)
3. **Middle** (section)
4. **Harmonic** (section)

9.2.6.1 PADsynth / Harmonic Structure / Change / Oscillator

1. **Oscillator Spectrum Graph**
2. **Oscillator Waveform Graph**
3. **Mag.Type**
4. **rnd** (ADDsynth Oscillator Editor only)
5. **H.rnd** (ADDsynth Oscillator Editor only)
6. **H.rnd knob** (ADDsynth Oscillator Editor only)
7. **Apply** (not present in ADDsynth Oscillator Editor)

TODO: Describe the 3 ADDsynth elements noted above.

rnd - Set the randomness of the oscillator output. There are 2 types of randomnesses, first is group randomness(the oscillator starts at random position), second is from -64(max) to -1 (min) and each harmonic (the oscillator is phase distorted) is from 1(min) to 63 (max). 0 is no randomness. One could use this parameter to make warm sounds like analogue synthesizers.

1. Oscillator Spectrum Graph. Oscillator Spectrum Graph.

2. Oscillator Waveform Graph. Oscillator Waveform Graph.

3. Mag.Type. Oscillator Magnitude Type. Sets how the magnitudes from the user interface behave. See the values below.

Figure 130: PADsynth Harmonic Content Mag Type

Values: `Linear*, -40dB, -60db, -80dB, -100dB`

4. Apply. PADsynth Harmonic Content Editor Apply Button.

9.2.6.2 PADsynth / Harmonic Structure / Change / Base Function

1. **Base Func. Spectrum Graph**
2. **Base Func. Waveform Graph**
3. **Base F..**
4. **Par. Value**
5. **Par. Wheel**
6. **B.F.Mod.**
7. **Wheel 1**
8. **Wheel 2**
9. **Wheel 3**

1. Base Func. Spectrum Graph. Harmonic Base Function Spectrum Graph.

2. Base Func. Waveform Graph. Harmonic Base Function Waveform Graph.

3. Base F... Harmonic Base Function. Sets what function to use as the harmonics base function. One can use any base function as harmonics.

Figure 131: PADsynth Harmonic Content Base Function

Values: `Sine*, Triangle, Pulse, Saw, Power, Gauss, Diode, AbsSine, PulseSine, StrchSine, Chirp, AbsStrSine, Chebyshev, Sqr, Spike, Circle`

4. Par. Value. PADsynth Parameter Value.

5. Par. Wheel. PADsynth Parameter Wheel. Change the parameter of the base function.

6. B.F.Mod.. PADSynth Base Frequency Mod.

7. Wheel 1. PADsynth Wheel 1.

8. Wheel 2. PADsynth Wheel 2.

9. Wheel 3. PADsynth Wheel 3.

9.2.6.3 PADsynth / Harmonic Structure / Change / Middle

1. **Use as base**
2. **Clr.**
3. **Wsh.**
4. **Wsh Value**
5. **Wsh Wheel**
6. **Filter**
7. **Filter Wheel 1**
8. **Filter Wheel 2**
9. **Filter p**
10. **Mod.**
11. **Mod. Wheel 1**
12. **Mod. Wheel 2**
13. **Mod. Wheel 3**
14. **Sp.adj.**
15. **Sp.adj. Wheel**

1. Use as base. Use as Base. Convert the oscillator output to a base function. Changing the Base function or its parameter will erase the converted base function.

2. Clr.. Clear. Clear the settings and make the oscillator equal to a base function. If this is cleared, one can click the **Use as base** button to make multiple conversions to base functions.

3. Wsh.. Harmonic Editor Wave-shaping, "W.sh".

Wave shaping function that applies to the oscillator. It has one parameter that fine-tunes the wave-shaping function.

4. Wsh Value. Harmonic Editor Wave-shaping Value.

Figure 132: PADsynth Harmonic Content Editor Wave-Shaping Function

Values: `None*, Atan, Asym1, Pow, Sine, Qnts, Zigzg, Lmt, LmtU, LmtL, ILmt, Clip, Asym2, Pow2, Sgm`

The type of wave-shaping distortion has much influence on how the overtones are being placed. Sometimes, one gets a "fat" bass, and sometimes, high frequencies are added, making the sound "crystal clear".

Atan & Sigmoid. This is the default setting. It is an easy way to apply loudness to a wave without getting undesired high overtones. Thus, it can be used both for making instruments that sound like "real" ones, but also for electronic music. The transformation turns, roughly said, every amplitude into a square amplitude. Thus, sine, power, pulse and triangle turn into a usual square wave, while a saw turns into a phased square wave. A chirp wave turns into a kind of phase modulated square wave.

Quants ("Qnts") Quantization adds high overtones early. It can be seen as an unnatural effect, which is often used for electronic music. The transformation is a bit similar to building the lower sum of a wave, mathematically said. This means that the transformation effect turns an "endless high" sampled wave into only a few samples. The more distortion one applies, the fewer samples will be used. Indeed, this is equivalent to say that more input amplification is used. To see this, here is a small sample of code, where "ws" is the (correctly scaled amount of input amplification, and "n" the number of original samples.

If one turns on quantisation very high, one might be confused that, especially high notes, make no sound. The reason: High frequencies are "forgotten" if one samples with only few samples. Also, the sign of an amplitude can be forgotten. This behaviour might make some quantisations a bit unexpected.

Limiting ("Lmt*" and "Clip") Limiting usually means that for a signal, the amplitude is modified because it exceeds its maximum value. Overdrive, as often used for guitars, is often achieved by limiting: It happens because an amplifier "overdrives" the maximum amplitude it can deliver.

ZynAddSubFX has two types of limiting. Soft limiting, here as Lmt, means that the sound may not exceed a certain value. If the amplitude does so, it will simply be reduced to the limiting value. The overtones are generated in the lower frequencies first.

Hard limiting, is also called clipping and abbreviated Clip. This means that if the maximum is exceeded, instead of being constant at the limiting value, the original signal still has some influence on the output signal. Still, it does not exceed the limiting value. For ZynAddSubFX, a signal exceeding the limiting value will continue to grow "in the negative". This leads to overtones being generated on the full frequency band.

5. Wsh Wheel. Harmonic Editor Wave-shaping Wheel?

6. Filter. Harmonic Editor Filter. Sets the type of the harmonic filter.

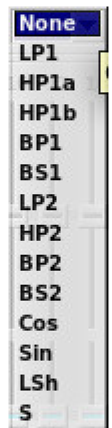

Figure 133: PADsynth Harmonic Content Filter

Values: `None*, LP1, HP1a, HP1b, BP1, BS1, LP2, HP2, BP2, BS2, Cos, Sin, LSh, S`

7. Filter Wheel 1. Harmonic Editor Filter, Wheel 1.

8. Filter Wheel 2. Harmonic Editor Filter, Wheel 2. The knob in the right sets the filter parameter (frequency).

9. Filter p. Harmonic Editor Filter, p?

10. Mod.. Harmonic Editor Modulation.

Figure 134: PADsynth Harmonic Content Editor Modulation

Values: `None*, Rev, Sine, Pow`

11. Mod. Wheel 1. Harmonic Editor Modulation Wheel 1?

12. Mod. Wheel 2. Harmonic Editor Modulation Wheel 2?

13. Mod. Wheel 3. Harmonic Editor Modulation Wheel 3?

14. Sp.adj.. Harmonic Editor Spectrum Adjust. Adjust the spectrum of the waveform.

MORE FROM ZYN:

RMS normalize. Enables the RMS normalization method (recommended); this keeps the same loudness regardless the harmonic content.

Below are the harmonics and their phases. One can use them to add to oscillator harmonics that has the waveform of the base function. Increasing the number of harmonics has virtually no effect on CPU usage. Right click to set a harmonic/phase to the default value.

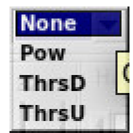

Figure 135: PADsynth Harmonic Content Editor Spectrum Adjust

Values: `None*, Pow, ThrsD, ThrsU`

15. Sp.adj. Wheel. Harmonic Editor Spectrum Adjust Wheel?

9.2.6.4 PADsynth / Harmonic Structure / Change / Harmonic

1. **Harmonics Amplitude**
2. **Harmonics Bandwidth**
3. **Harmonics Scrollbar**
4. **Harmonic Shift**
5. **Harmonic Shift R** (dialog?)
6. **Harmonic Shift preH**
7. **Adpt.Harm.**
8. **Adpt.Harm. Slider**
9. **Adpt.Harm. baseF**
10. **Adpt.Harm. pow**
11. **Clear**
12. **Sine**
13. **C**
14. **P**
15. **Close**

16. Harmonics Amplitude. Harmonics Amplitude. Provides 128? sliders for the amplitude of harmonics.

17. Harmonics Bandwidth. Harmonics Bandwidth. Provides 128? sliders for the bandwidth of harmonics.

18. Harmonics Scrollbar. Harmonics Scrollbar.

19. Harmonic Shift. Harmonics Shift.

Values: `-x to 0 to x?`

20. Harmonic Shift R. Harmonics Shift R?.

21. Harmonic Shift preH. Harmonics Shift preH? preF in Zyn?

preF. Set the order of doing the filter and wave-shaper (uncheck to filter after wave-shaping, check to wave-shape after filtering).

OKAY?

22. Adpt.Harm.. Adaptive Harmonics?

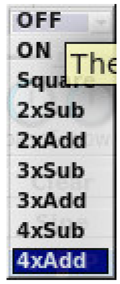

Figure 136: PADsynth Adaptive Harmonic Type

Values: `OFF*, ON, Square, 2xSub, 2xAdd, 3xSub, 3xAdd, 4xSub, 4xAdd`

23. Adpt.Harm. Slider. Adaptive Harmonics Slider?

24. Adpt.Harm. baseF. Adaptive Harmonics Base Frequency?

25. Adpt.Harm. pow. Adaptive Harmonics Power?

26. Clear. Harmonics Clear. Clears the harmonics settings.

27. Sine. Harmonics Sine. The user is prompted to "Convert to sine?" This seems to reset everything to the state where it has not been modified, but that's not certain.

28. C. Harmonics Copy.

29. P. Harmonics Paste.

30. Close. Harmonics Close.

9.3 PADsynth / Envelopes and LFOs

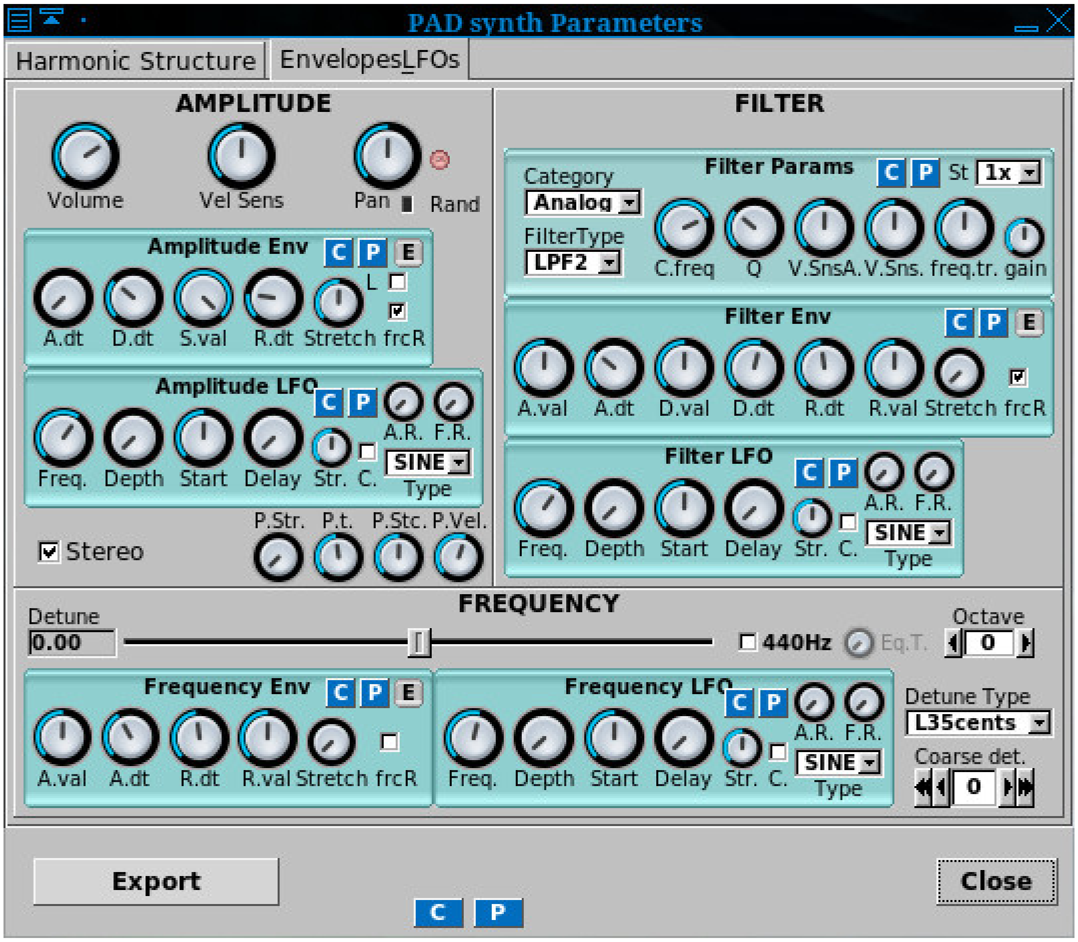

Figure 137: PADSynth Parameters, Envelopes and LFOs

1. **AMPLITUDE**
2. **FILTER** (section)
3. **FREQUENCY** (section)
4. **Export**
5. **C**
6. **P**
7. **Close**

31. AMPLITUDE. See section 8.1 ("ADDsynth / AMPLITUDE") on page 115. This stock dialog section provide volume, velocity sensing, panning, an amplitude envelope sub-panel, and an amplitude LFO sub-panel.

32. FILTER. See section 8.2 ("ADDsynth / FILTER") on page 117.

33. FREQUENCY. See section 8.3 ("ADDsynth / FREQUENCY") on page 118.

34. Export. Very similar to figure 128 ("Harmonics Structure Export Dialog") on page 140.

35. C. The stock copy dialog.

36. P. The stock paste dialog.

37. Close. Close.

10 SUBsynth

The *Yoshimi* SUBsynth dialog is a complex dialog for creating a subtractive-synthesis instrument, "SUB-synth" or "SUBnote" is a simple engine which makes sounds through subtraction of harmonics from white noise. [22]

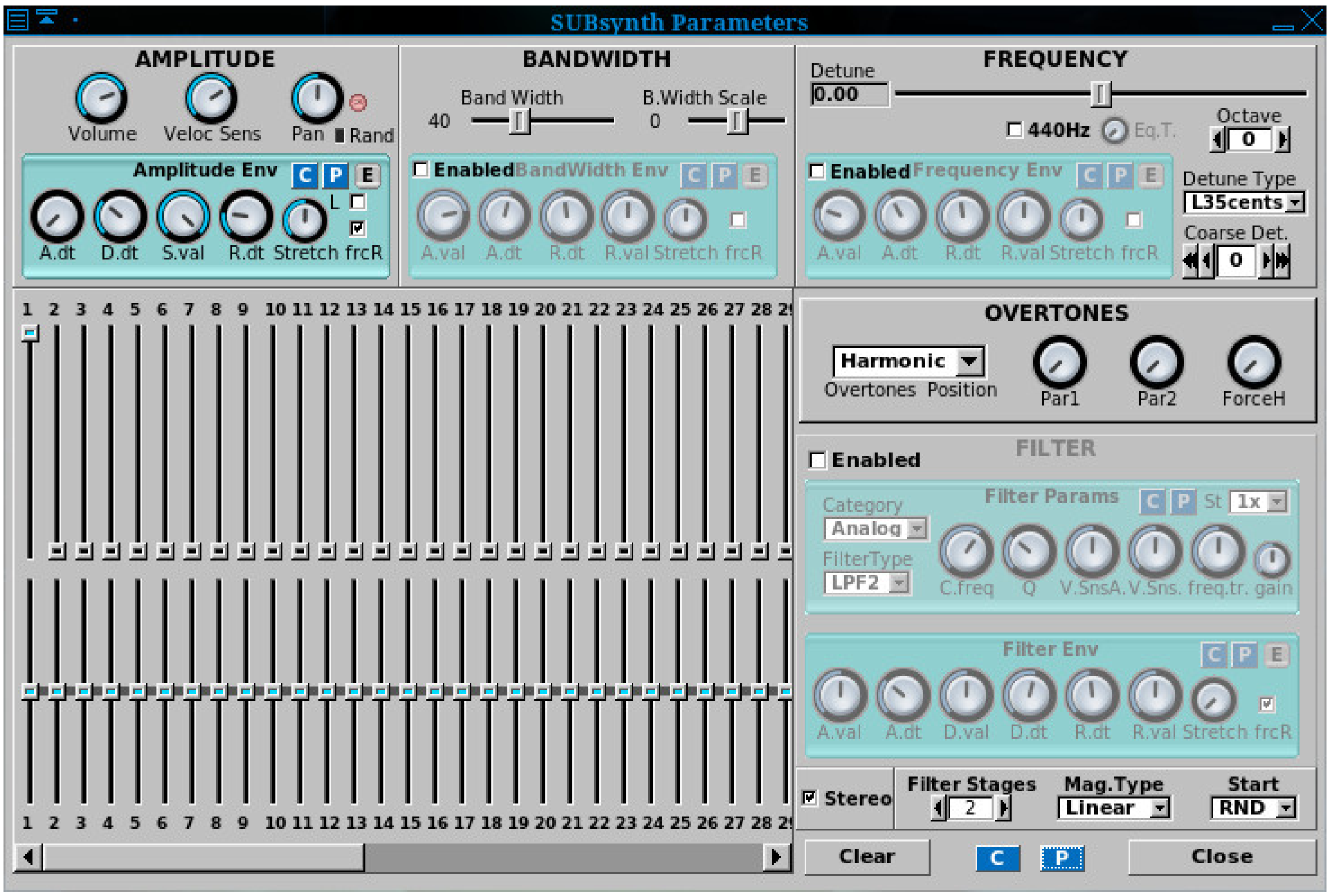

Figure 138: SUBsynth Edit Dialog

1. **AMPLITUDE** (section)
2. **BANDWIDTH** (section)
3. **FREQUENCY** (section)
4. **OVERTONES** (section)
5. **FILTER** (section)
6. **Harmonics** (section)
7. **Clear**
8. **C**
9. **P**
10. **Close**

10.1 SUBsynth / AMPLITUDE

1. **Volume**
2. **Vel Sens**
3. **Pan**
4. **Rand**
5. **Reset (panning)** (red button)
6. **Amplitude Env** (stock sub-panel)

1. Volume. SUBsynth Volume.

Values: `1 to 127, 64*`

2. Vel Sens. Velocity Sensing function, rightmost/max to disable.

Values: `1 to 127, 64*`

3. Pan. Global panning, leftmost/zero gives random panning.

Values: `1 to 127, 64*`

4. Rand. Indicator for activation of random panning.

5. Reset (panning). Reset Panning.

6. Amplitude Env. Amplitude Envelope. See section ...... for this stock sub-panel.

10.2 SUBsynth / BANDWIDTH

1. **BandWidth**
2. **B.Width Scale**
3. **Bandwidth Env**

1. BandWidth. SUBsynth Bandwidth. Sets the bandwidth of each harmonic.

Values: `1 to 127, 40*`

2. B.Width Scale. SUBsynth Bandwidth Scale. Sets how the bandwidth of each harmonic is increased according to the frequency. The default (0) increases the bandwidth linearly according to the frequency.

Values: `0 to 127???`

3. Bandwidth Env. SUBsynth Bandwidth.

1. **Enabled**
2. **A.val**
3. **A.dt**
4. **R.dt**
5. **R.val**
6. **Stretch**
7. **frcR**
8. **C**
9. **P**
10. **E**

1. Enabled. Enable the panel.

2. A.val. Attack value. We need to figure out what this means.

Values: `0 to 127, 64*`

3. A.dt. Attack duration. Attack time.

Values: `0 to 127, 40*`

4. R.dt. Release time.

Values: `0 to 127, 60*`

5. R.val. Release Value. Actually present only on the Frequency Env sub-panel.

Values: `0 to 127, 64*`

6. Stretch. Bandwidth Stretch. On lower notes make the bandwidth lower.

Values: `0 to 127, 64*`

7. frcR. Forced release.

Values: `Off, On*`

If this option is turned on, the release will go to the final value, even if the sustain level is not reached.

Also present in this sub-panel are the usual **C**opy and **P**aste buttons that call up a copy-parameters or paste-parameters dialog, as well as a button to bring up the editor window.

10.3 SUBsynth / FREQUENCY

1. **Detune**
2. **FREQUENCY Slider**
3. **440Hz**
4. **Eq.T**
5. **Octave**
6. **Detune Type**
7. **Coarse Det.**
8. **Frequency Env**

Category - Filter category: Analog/Formant/SVF ????

1. Detune. Frequency Detune. Fine detune?

2. FREQUENCY Slider. Frequency Slider.

3. 440Hz. Frequency 440Hz. 440Hz. Fix the base frequency to 440Hz. One can adjust it with detune settings.

4. Eq.T. Frequency Equalize Time?.

5. Octave. Frequency Octave. Octave Shift.

6. Detune Type. Frequency Detune Type. Sets the "Detune" and "Coarse Detune" behavior

7. Coarse Det.. Frequency Coarse Detune, "C.Detune".

8. Frequency Env. Frequency Envelope Stock Sub-Panel.

1. **Enable**
2. **A.value** or **A.val**

3. **A.dt**
4. **R.dt**
5. **R.val**
6. **Stretch**
7. **frcR**
8. **C**
9. **P**
10. **E**

See section ....

10.4 SUBsynth / OVERTONES

The harmonics settings controls the harmonic intensities/relative bandwidth. Moving the sliders upwards increases the relative bandwidth. Please note that, if one increases the number of harmonics, the CPU usage increases. Right click to set the parameters to default values.

1. **Overtones Position**
2. **Par1**
3. **Par2**
4. **ForceH**

1. Overtones Position. Subsynth Overtones Position.

Values: `Harmonic, ShiftU, ShiftL, PowerU, PowerL, Sine, Power, Shift`

Figure 139: Harmonic Type Dropdown

2. Par1. Subsynth Overtones Par1.

Values: `0 to 127`

3. Par2. Subsynth Overtones Par2.

Values: `0 to 127`

4. ForceH. Subsynth Overtones ForceH.

Values: `0 to 127`

10.5 SUBsynth / FILTER

1. **Enabled**
2. **Filter Params** (stock sub-panel)

3. **Filter Env** (stock sub-panel)
4. **Stereo**
5. **Filter Stages**
6. **Mag. Type**
7. **Start**

1. Enabled. SUBsynth Filter Enabled.

2. Filter Params. Filter Params. See section xxxxx for this stock sub-panel.

3. Filter Env. Filter Params. See section xxxxx for this stock sub-panel.

4. Stereo. SUBsynth Stereo. Make the instrument stereo. The CPU usage goes up about 2 times. Is this really a FILTER item?

5. Filter Stages. Filter Stages. Filter Order. Sets the number of filter stages applied to white noise. This parameter affects the CPU usage.

Values: `0, 1, 2*, 3, 4, 5???`

6. Mag. Type. Magnitude Type. Type of magnitude settings (Linear/dBs)

Figure 140: SUBSynth Magnitude Type Dropdown

Values: `Linear, -40dB, -60dB, -80dB, -100dB`

7. Start. Start Type. How to start the filters.

Figure 141: SUBsynth Start Type

Values: `Zero, RND, Max.`

10.6 SUBsynth / Harmonics

This section consists of 64 sliders to control the amplitude of the narrow noise band at a given harmonic, and 64 sliders to control the bandwidth of each band.

TODO.

11 Kit Edit

The *Yoshimi* Kit dialog is a dialog for creating a set of drums or layered instruments. It provides a way to use individual voices and synth blocks to create drumlike sounds, or complex layered sounds. Within this window one can create drum kits, layered instruments, or one can combine more instruments into one instrument.

Is this true of *Yoshimi*?:

Item 0 is a special type: it cannot be disabled (but it can be muted), to edit it one must use "ADs edit" or "SUBs edit" from the part window.

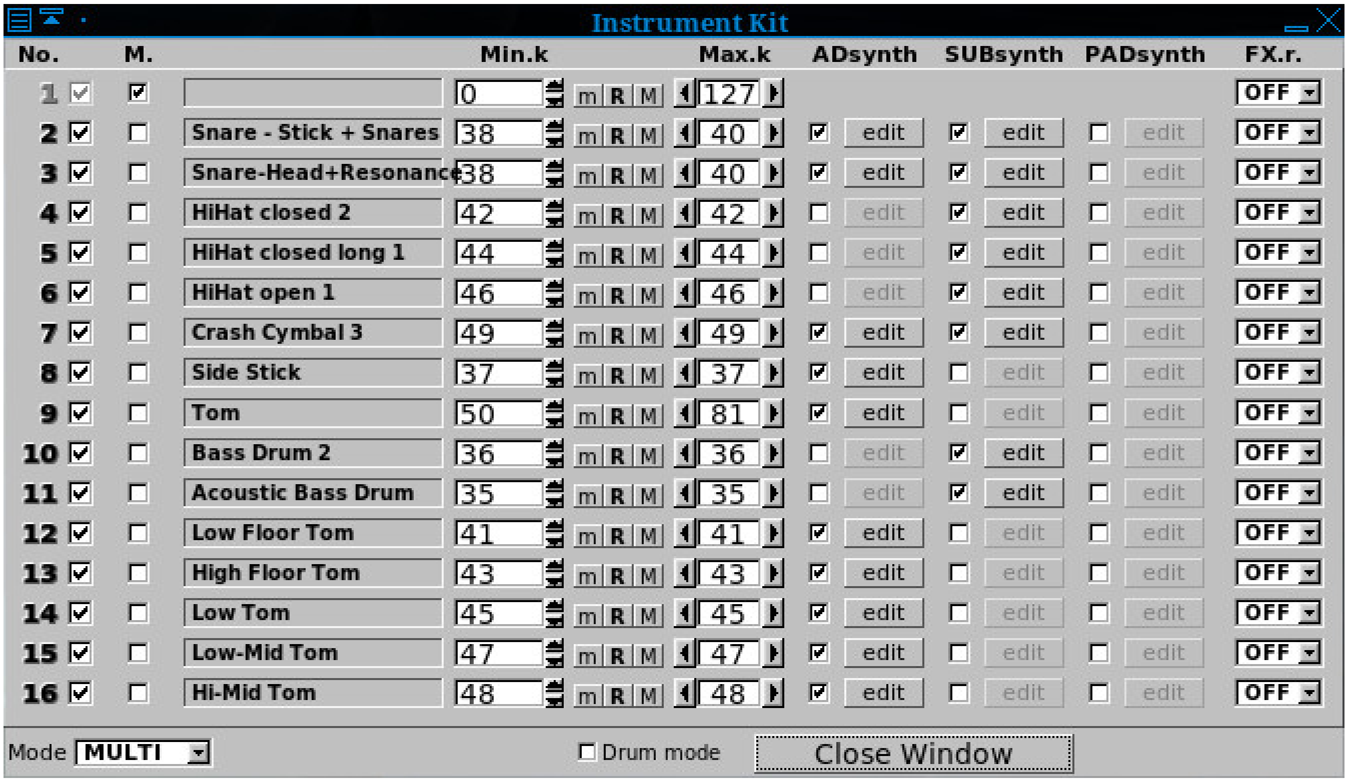

Figure 142: Kit Edit Dialog

1. **Rows 1 to 16.** This dialog contains 16 identical rows containing the following elements, in the order given:
    1. **No.**
    2. **Enable**
    3. **M.**
    4. **Instrument Name**
    5. **Min.k**
    6. **m** (set minimum note)
    7. **R** (reset default note range)
    8. **M** (set maximum note)
    9. **Max.k**
    10. **ADsynth**
        1. **Enable**
        2. **edit**
    11. **SUBsynth**
        1. **Enable**
        2. **edit**

    12. **PADsynth**
        1. **Enable**
        2. **edit**
    13. **FX.r**
2. **Mode**
3. **Drum mode**
4. **Close Window**

Some items described in ZynAddSubFX that aren't seen in any diagrams:

1. Kit Mode. Enable the kit mode.
2. Protect the kit. when loading an instrument, only item 0 will be changed, Other items will remain untouched. This allows one to combine more instruments. If one wants to add more instruments to the kit, one must copy the item 0 to another item, because the item 0 will be replaced. If one loads master settings or clearx the instrument/master setting, the kit is cleared .
3. Swap/Copy. Swap two items or copy a item to other item.

1. No.. Kit Row Number. Kit Item Number. A simple label to indicate the instrument number in the kit.

2. Enable. Kit Row Enable.

Value: `Off*, On`

3. M.. Kit Row "M". Mute an item of the kit.

4. Instrument Name. Kit Instrument Name.

5. Min.k. Kit Instrument Minimum Key. Sets the minimum key of the item of the kit.

6. m. Sets the minimum note of this instrument to value of the last note pressed.

7. R. Resets the minimum and maximum notes to their default values.

8. M. Sets the maximum note of this instrument to value of the last note pressed.

9. Max.k. Kit Instrument Maximum Key. Sets the maximum key of the item of the kit.

10. ADsynth. Kit ADDsynth. A checkbox is provided to enable/disable this synth component, and an edit button is provided to edit the component.

11. SUBsynth. Kit SUBsynth. A checkbox is provided to enable/disable this synth component, and an edit button is provided to edit the component.

12. PADsynth. Kit PADsynth. A checkbox is provided to enable/disable this synth component, and an edit button is provided to edit the component.

13. FX.r. Kit Effect. Chooses the Part Effect (PartFX) to process the item (OFF means that is unprocessed).

Values: `OFF, FX1, FX2, FX3`

14. Mode. Kit Mode.

- **OFF** means no kit is enabled, so one only has the Add, Sub, and Pad sounds in the Instrument Edit window.
- **MULTI** means all the kit items will sound together regardless of their note ranges.
- **SINGLE** means only the lowest numbered item will sound in a given note range. There will be no overlap.

For example: Item 0 has **Min.k** set to 0 and **Max.k** set to 60, and Item 1 has **Min.k** set to 40 and **Max.k** set to 127.

In **SINGLE mode**, only Item 0 will sound in the note range 0 to 60, and Item 1 will sound in the range 61 to 127.

In **MULTI** mode, only Item 0 will sound in the range 0 to 40, both items will sound from 41 to 60, and only Item 1 will sound from 61 to 127.

Values: `OFF*, MULTI, SINGLE`.

15. Drum mode. Kit Drum Mode. If drum-mode is set, then microtonal tuning is ignored for this kit, otherwise it could make drum sounds very unpredictable!

16. Close Window. Close.

12 Banks Collection

In this section, we attempt to collect and summarize all of the existing banks for *Yoshimi* and *ZynAddSubFX* that we can find. Many of them are supplied by the two projects.

Between all of the collections, there is a large amount of duplication. There is also semi-duplication, with slight variations on the same basic instrument. Various Linux distributions which package *ZynAddSubFX* and *Yoshimi* might add some banks to their versions of these packages. Thus, there are far more sound settings than we can discuss and categorize.

One thing we're looking for is a good General MIDI (GM) bank for *Yoshimi*. As part of our *Yoshimi Cookbook* [2], we include a basic General MIDI bank for. However, there are number of patches with no good implementation in it.

12.1 Yoshimi Banks

Yoshimi comes with the following banks, which may be found in `/usr/share/yoshimi/banks` as installed by the installer. In this case, it is the Debian installer.

1. **Arpeggios**. Also in ZynAddSubFX.
2. **Bass**. Also in ZynAddSubFX.
3. **Brass**. Also in ZynAddSubFX.
4. **chip**.
5. **Choir_and_Voice** Also in ZynAddSubFX, slightly different bank name.
6. **Drums**. Also in ZynAddSubFX, but with only one drum kit included.
7. **Dual**. Also in ZynAddSubFX.
8. **Fantasy**. Also in ZynAddSubFX.
9. **Guitar**. Also in ZynAddSubFX.
10. **Misc**. Also in ZynAddSubFX.
11. **Noises**. Also in ZynAddSubFX.
12. **Organ**. Also in ZynAddSubFX.
13. **Pads**. Also in ZynAddSubFX.
14. **Plucked**. Also in ZynAddSubFX.
15. **Reed_and_Wind**. Also in ZynAddSubFX, slightly different bank name.
16. **Rhodes**. Also in ZynAddSubFX.
17. **Splited**. Also in ZynAddSubFX, slightly different bank name.

18. **Strings**. Also in ZynAddSubFX.
19. **Synth**. Also in ZynAddSubFX.
20. **SynthPiano**. Also in ZynAddSubFX.
21. **The_Mysterious_Bank**. Also in ZynAddSubFX, slightly different bank name. ZynAddSubFx has three more mysterious banks (see next section).
22. **Will_Godfrey_Collection**.
23. **Will_Godfrey_Companion**.

12.2 Additional ZynAddSubFX Banks

ZynAddSubFX comes with the following banks, which may be found in the source code [25] or installation packages of this project. *ZynAddSubFX* has some of the same banks (as far as we can tell) as *Yoshimi*, but with the following additions:

1. **Companion**.
2. **Cormi_Noise and Cormi_Sound** [3].
3. **Laba170bank**.
4. **olivers-100**. Some very good instruments, including sitar and steel drums.
5. **the_mysterious_bank**.
6. **the_mysterious_bank_2**.
7. **the_mysterious_bank_3**.
8. **the_mysterious_bank_4**.

12.3 Additional Banks

Here are some additional banks we have found, or have built ourselves. It often happens that, later on, a site is no longer available. Or we forget from whence we got the banks. In these cases, the banks are stored in the `contrib/banks` directory of this project.

1. **Alex_J** The site seems to be gone/expired. So one will find these in the "contrib/banks" directory for safekeeping.
2. **Bells** We have no idea where we got this one. Lost track of that information.
3. **C_Ahlstrom** These are mine, but not yet made into a systematic bank. They are included with this document.
4. **Chromatic Percussion** Not sure where we got this at this time.
5. **Drums_DS** Not sure where we got this at this time.
6. **Electric Piano** Not sure where we got this at this time.
7. **Flute** Not sure where we got this at this time.
8. **folderol collection** [5], also found at [21].
9. **Internet Collection** Not sure where we got this at this time.
10. **Leads** Not sure where we got this at this time.
11. **Louigi_Verona_Workshop** The site seems to be gone/expired. So one will find these in the "contrib/banks" directory for safekeeping.
12. **Misc Keys** Not sure where we got this at this time.
13. **mmxgn Collection** [9]
14. **Piano** Not sure where we got this at this time.
15. **RB Zyn Presets** Not sure where we got this at this time.
16. **Vanilla** See [21] for this bank, and for some demonstration files of *ZynAddSubFX* sounds, and some other nice links.

17. **VDX** Not sure where we got this at this time.
18. **x31eq.com** [15]
19. **XAdriano Petrosillo** Not sure where we got this at this time.
20. **Zen Collection** Not sure where we got this at this time.

13 Non-Registered Parameter Numbers

This section comes from the source-code documentation file `Zyn nrpn.txt` or the *ZynAddSubFx* online manual [22] and the *Using_NRPNS.txt* document that accompanies the *Yoshimi* source code.

Yoshimi implements System and Insertion effects control in a manner compatible with *ZynAddSubFX*. As with all *Yoshimi*'s NRPNs, the controls can be sent on any MIDI channel.

13.1 NRPN / Basics

NRPN stands for "Non Registered Parameters Number". NRPNs can control all System and Insertion effect parameters. Using NRPNs, *Yoshimi* can now directly set some part values regardless of what channel that part is connected to. For example, one may change the reverb time when playing to keyboard, or change the flanger's LFO frequency. The controls can be sent on any MIDI channel (the MIDI channels numbers are ignored).

The parameters are:

- **NRPN MSB** (coarse) (99 or 0x63) sets the system/insertion effects (4 for system effects or 8 for insertion effects). We abbreviate this value as `Nhigh`.
- **NRPN LSB** (fine) (98 or 0x62) sets the number of the effect (first effect is 0). We abbreviate this value as `Nlow`.
- **Data entry MSB** (coarse) (6) sets the parameter number of effect to change (see below). We abbreviate this value as `Dhigh`.
- **Data entry LSB** (fine) (26) sets the parameter of the effect. We abbreviate this value as `Dlow`.

If the effect/parameter doesn't exists or is set to none, then the NRPN is ignored.

One must send NRPN coarse/fine before sending Data entry coarse/fine. If the effect/parameter doesn't exists or is set to none, then the NRPN is ignored. It's generally advisable to set NRPN MSB before LSB However, once MSB has been set one can set a chain of LSBs if they share the same MSB.

The data CCs associated with these are 6 for MSB and 38 for LSB.

Only when an NRPN has been established can the data values be entered (they will be ignored otherwise).

If a supported control is identified, these data values will be stored locally (if needed) so that other NRPNs can be set.

Whenever either byte of the NRPN is changed, the data values will be cleared (but stored settings will not be affected).

If either NRPN byte is set to 127, all data values are ignored again.

In *Yoshimi* NRPNs are not themselves channel-sensitive, but the final results will often be sent to whichever is the current channel.

Yoshimi also supports the curious 14-bit NRPNs, but this shouldn't be noticable to the user. In order to deal with this, and also some variations in the way sequencers present NRPNs generally, if a complete NRPN is set (i.e. `Nhigh`, `Nlow`, `Dhigh`, `Dlow`), then the data bytes can be in either order, but must follow `Nhigh` and `Nlow`.

After this, for running values, once `Dhigh` and `Dlow` have been set if one changes either of these, the other will be assumed. For example, starting with `Dhigh` = 6 and `Dlow` = 20:

Change `Dlow` to 15 and *Yoshimi* will regard this as a command `Dhigh` 6 + `Dlow` 16 Alternatively change `Dhigh` to 2 and *Yoshimi* will regard this as a command `Dhigh` 2 + `Dlow` 20. This can be useful but may have unintended consequences! If in doubt change either of the NRPN bytes and both data bytes will be cleared.

Additionally there is 96 for data increment and 97 for decrement.

Data increment and decrement operation enables one to directly change the data LSB by between 0 and 63. To change the MSB add 64 to cover the same range. Setting zero might seem pointless, but it gives an alternative way to make an initial setting if one's sequencer doesn't play nice.

Although data increment and decrement are only active if a valid NRPN has been set, they are otherwise quite independent single CCs. For example:

```
Start Value       Command value   Result
----- -----       --------------  ------
LSB     5           inc 20      25
MSB     7           inc 68      11
LSB     128(off)    inc 1       1
MSB     126         dec 74      116
MSB     128(off)    dec 65      127
```

A small example (all values in this example are hex):

```
B0 63 08 // Select the insertion effects
B0 62 01 // Select the second effect (remember: the first is 00 and not 01)
B0 06 00 // Select the effect parameter 00
B0 26 7F // Change the parameter of effect to the value 7F (127)
```

WARNING: Changing of some of the effect parameters produces clicks when sounds passes thru these effects. We advise one to change only when the sound volume that passes thru the effect to be very low (or silence). Some parameters produce clicks when are changed very fast.

Here are the effects parameter number (for Data entry, coarse). The parameters that produces clicks are written in red and have (AC) after their entry (always clicks). The parameter that produces clicks only when they are changed fast are written in blue and have a (FC) after the entry (Fast Clicks). Most parameters have the range from 0 to 127. When parameters have another range, it is written as "(low...high)".

Here are the basic formats:

1. Send NRPN:

- MSB = 64 (same as for vectors)
- LSB = 0

2. Send Data MSB (6); all value ranges start from zero, not 1.
   - 0 : data LSB = part number
   - 1 : data LSB = program number
   - 2 : data LSB = controller number
   - 3 : data LSB = controller value
   - 4 : data LSB = part's channel number (15 to 127 disconnects the part from any channel)
   - 7 : data LSB = main volume (not yet implemented)
   - 35 (0x23) : data LSB = controller LSB value (not yet implemented)
   - 39 (0x27) : data LSB = main volume LSB (not yet implemented)

Other values are currently ignored by *Yoshimi*.

NOTE: THE PARAGRAPHS THAT FOLLOW COULD BE MOVED TO THE EFFECTS SECTION.

13.2 NRPN / Vector Control

Vector control is a way to control more than one part with the controllers. It is a little bit reminiscent of the "vector" control knob on the Yamaha PSS-790 consumer MIDI synthesizer. Vector control is only possible if one has 32 or 64 parts active.

In vector mode parts will still play together but the vector controls can change their volume, pan, filter cutoff in pairs, controlled by user defined CCs set up with NRPNs.

One must set the X axis CC before Y, but if one doesn't set Y at all one can run just a single axis. If yuo only have 32 parts active Y settings are ignored.

For example: parts 1 and 17 can be set as x1 & x2 (volume only) while parts 33 and 49 can be y1 & y2 (pan only)

Independently of this Parts 2 & 18 could use filter and pan from another CC.

Setting up vector control is currently done as follows.

In the required channel send:

- NRPN MSB (99) set to 64
- NRPN LSB (98) 1
- Data MSB (6) set mode
  - 0 = X CC
  - 1 = Y CC
  - 2 = X features
  - 3 = Y features
  - 4 = x1 instrument (optional)
  - 5 = x2 instrument (optional)
  - 6 = y1 instrument (optional)
  - 7 = y2 instrument (optional)

Setting CC for X enables vector control any value oustide the above list disables it.

Data LSB (38) value to set features:

- 1 = Volume
- 2 = Pan
- 4 = Filter Cutoff (Brightness)

An Example:

From channel 1, send the following CCs

```
CC      Value
99       64
98        1
 6        0
38       14
 6        1
38       15
 6        2
38        1
 6        3
38        2
```

This sequence will set up CC14 as the X axis incoming controller and CC15 as Y, with X set to volume control and Y set to pan control.

One can either go on with the NRPNs to set the instruments (this will load and enable instruments from the current bank), or enable and load them by hand. For channel 1 this would be part 1 and 17 for X and part 33 and 49 for Y.

13.3 NRPN / Effects Control

13.3.0.1 Reverb

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)
- 02 - Reverb Time
- 03 - Initial Delay (FC)
- 04 - Initial Delay Feedback
- 05 - reserved
- 06 - reserved
- 07 - Low Pass
- 08 - High Pass
- 09 - High Frequency Damping (64..127) 64=no damping
- 10 - Reverb Type (0..1) 0-Random, 1-Freeverb (AC)
- 11 - Room Size (AC)

13.3.0.2 Echo

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)

- 02 - Delay (AC)
- 03 - Delay between left and right (AC)
- 04 - Left/Right Crossing (FC)
- 05 - Feedback
- 06 - High Frequency Damp

13.3.0.3 Chorus

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)
- 02 - LFO Frequency
- 03 - LFO Randomness
- 04 - LFO Type (0..1)
- 05 - LFO Stereo Difference
- 06 - LFO Depth
- 07 - Delay
- 08 - Feedback
- 09 - Left/Right Crossing (FC)
- 10 - reserved
- 11 - Mode (0..1) (0=add, 1=subtract) (AC)

13.3.0.4 Phaser

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)
- 02 - LFO Frequency
- 03 - LFO Randomness
- 04 - LFO Type (0..1)
- 05 - LFO Stereo Difference
- 06 - LFO Depth
- 07 - Feedback
- 08 - Number of stages (0..11) (AC)
- 09 - Let/Right Crossing (FC)
- 10 - Mode (0..1) (0=add, 1=subtract) (AC)
- 11 - Phase

13.3.0.5 AlienWah

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)
- 02 - LFO Frequency
- 03 - LFO Randomness
- 04 - LFO Type (0..1)
- 05 - LFO Stereo Difference
- 06 - LFO Depth
- 07 - Feedback
- 08 - Delay (0..100)
- 09 - Left/Right Crossing (FC)
- 10 - Phase

13.3.0.6 Distortion

- 00 - Volume or Dry/Wet (FC)
- 01 - Pan (FC)
- 02 - Left/Right Crossing
- 03 - Drive (FC)
- 04 - Level (FC)
- 05 - Type (0..11)
- 06 - Invert the signal (negate) (0..1)
- 07 - Low Pass
- 08 - High Pass
- 09 - Mode (0..1) (0=mono,1=stereo)

13.3.0.7 EQ

- 00 - Gain (FC)

All other settings of the EQ are shown in a different way. The N represent the band (”B.” setting in the UI) and the first band is 0 (and not 1), like it is shown in the UI. Change the ”N” with the band one likes. If one wants to change a band that doesn’t exist, the NRPN will be ignored.

- 10+N*5 - Change the mode of the filter (0..9) (AC)
- 11+N*5 - Band’s filter frequency
- 12+N*5 - Band’s filter gain
- 13+N*5 - Band’s filter Q (bandwidth or resonance)
- 14+N*5 - reserved

Example of setting the gain on the second band in the EQ module:

- The bands start counting from 0, so the second band is 1 =<N=1.
- The formula is 12+N*5 =<12+1*5=17, so the number of effect parameter (for Data entry coarse) is 17.

13.3.0.8 DynFilter

- 0 - Volume
- 1 - Pan
- 2 - LFO Frequency
- 3 - LFO Randomness
- 4 - LFO Type
- 5 - LFO Stereo Difference
- 6 - LFO Depth
- 7 - Filter Amplitude
- 8 - Fitler Amplitude Rate Change
- 9 - Invert the signal (negate) (0..1)

Click behaviour of DynFilter has not yet been tested.

13.3.0.9 Yoshimi Extensions

If the Data MSB bit 6 is set (64) then Data LSB sets the effect type instead of a parameter number. This must be set before making a parameter change.

- 0 - Reverb
- 1 - Echo
- 2 - Chorus
- 3 - Phaser
- 4 - AlienWah
- 5 - Distortion
- 6 - EQ
- 7 - DynFilter

For Insert effects, if the Data MSB bit 5 is set (32) then Data LSB sets the destination part number. 127 is off and 126 is the Master Output.

A complete example:

- 99 - 8   insert effects
- 98 - 3   number 4 (as displayed)
- 6 - 32   set destination
- 38 - 126   Master Out
- 99 - 8 *
- 98 - 3 *
- 6 - 64   change effect
- 38 - 4   Alienwah
- 99 - 8 *
- 98 - 3 *
- 6 - 0   Dry/Wet
- 38 - 30   value

Notes (*): these repeats are not needed as far as *Yoshimi* is concerned, but some sequencers get unhappy without them.

Change just a parameter on an exisiting system effect:

- 99 - 4   system effects
- 98 - 0   the first effect
- 6 - 1   Pan
- 38 - 75   value

14 Yoshimi Man Page

The *Yoshimi* man page is actually the output of the `yoshimi --help` command, which prints out the command-line that are discussed in this section.

Yoshimi 1.3.4, a derivative of ZynAddSubFX - Copyright 2002-2009 Nasca Octavian Paul and others, Copyright 2009-2011 Alan Calvert

`-a --alsa-midi[=device]`
Use ALSA MIDI input. From the command line, as well as autoconnecting the main L & R outputs to JACK, with ALSA MIDI one can now auto-connect to a known source.

```
./yoshimi -K --alsa-midi="Virtual Keyboard"
```

`-A --alsa-audio[=device]`
Use ALSA audio output.

`-b --buffersize=size`
Set ALSA audio buffer size.

`-c --show-console`
Show console on startup.

`-i --no-gui`
Do not show the GUI.

`-j --jack-midi[=device]`
Use JACK MIDI input. From the command line, as well as autoconnecting the main L & R outputs to JACK, with JACK MIDI one can now auto-connect to a known source.

```
./yoshimi -K --jack-midi="jack-keyboard:midi_out"
```

`-J --jack-audio[=server]`
Use JACK audio output.

`-k --autostart-jack`
Auto-start the JACK server.

`-K --auto-connect`
Auto-connect JACK audio.

`-l --load=file`
Load a `.xmz` file.

`-L --load-instrument=file`
Load an `.xiz` file

`-N --name-tag=tag`
Add tag to client-name.

`-o --oscilsize=size`
Set OscilSize from command-line.

`-R --samplerate=rate`
Set ALSA audio sample rate.

`-S --state[=file]`
Load state from file, where the file defaults to `$HOME/.config/yoshimi/yoshimi.state`

`-? --help`
Give this help list.

`--usage` P
 rovide a short usage message.

`-V --version`
 Print program version.

Mandatory or optional arguments to long options are also mandatory or optional for any corresponding short options.

From the command line, as well as autoconnecting the main L & R outputs to JACK, with either JACK or ALSA MIDI one can now auto-connect to a known source.

ALSA can often manage with just the client name, but JACK needs the port as well. These commands are case sensitive, and quite fussy about spaces etc. so it's wise to use quotes for the source name, even if they don't seem to be needed.

15 Building Yoshimi

This section describes building and debugging *Yoshimi*. Building *Yoshimi* requires getting the source code, making sure all of the necessary dependencies are installed, and using CMake to set up the build.

The source-code is located at its main location ([16]) or its alternate location ([17]

Like *ZynAddSubFX*, *Yoshimi* uses CMake as its build system [20]. CMake is a preprocessor that can generate project build setups for Visual Studio, UNIX make, and Xcode.

15.1 Yoshimi Source Code

Get the source code version you want from SourceForge (`http://sourceforge.net/projects/yoshimi/files/1.3/`). Download the desired tar-ball and unpack it in your work area.

Since SourceForge has had some issues, the *Yoshimi* team has wisely hosted the source code at another site as well, `https://github.com/abrolag/yoshimi`. One can grab the whole git repository there using the following command in your work area:

```
$ git clone https://github.com/abrolag/yoshimi.git
```

15.2 Yoshimi Dependencies

To save some wasted time, make sure the *development versions* of the following packages have been installed using your Linux distribution's package manager:

- `pkg-config`
- `libz`
- `fftw3f`
- `mxml`
- `ALSA` (libasound)
- `JACK`
- `Boost`

- fontconfig
- libcairo
- FLTK
- lv2

15.3 Build It

The following instructions are for an in-source build. An in-source build is simpler if you just want to build and install *Yoshimi*.

We will also show how to set up for an out-source-build, which keeps the build products out of the way.

The location of `CMakeList.txt` does not appear to be standard, so these instructions differ in details from the build instructions of *ZynAddSubFX*. Basically, the build is based in the project's `src` directory, instead of its root directory.

1. Enter the source directory where the code was unpacked.
2. Generate the project build-files:

```
$ cd src
$ cmake .
```

3. Build the code and install it (as root):

```
$ make
# make install
```

Here is how to make an out-of-source debug. Despite what cmake documentation (and Googling) says, using a command like the following *does not work* unless you have `ccmake` installed.

```
$ cmake -DCMAKE_BUILD_TYPE=Debug ..
$ ccmake
```

In Debian Linux, install the `cmake-curses-gui` package to get access to `ccmake`. Or use the shorter `cmake -DBuildForDebug=on ..` command below.

1. Enter the source directory where the code was unpacked.
2. Create a "Debug" or "Release" directory for an out-of-source build:

```
$ cd src
$ mkdir Debug
```

3. Generate the project build-files in the `Debug` directory.

```
$ cd Debug
$ cmake -DBuildForDebug=on ..
$ make
```

The output file, and executable name `yoshimi` is now ready to run (and be debugged).

Here is a debugging use case we used in *Yoshimi 1.3.5.1* and slightly earlier versions. Here is how to verify the bug:

1. Run the following command:

```
$ yoshimi -a -A
```

2. Navigate the following command path: Menu / Instrument / Show Banks
3. Select the **RENAME** button.
4. Select the bank (e.g. Arpeggios).
5. In the file prompt that comes up, click **Cancel**.
6. Observe a "Segmentation fault".

To avoid a lot of debugging, let *valgrind* find the bug for you. Install *valgrind*. Then, in the `src/Debug` directory, run:

```
$ valgrind --log-file=yoshvalgrind.log ./yoshimi -a -A
```

In the log file, one sees that the last good call was in the `Bank :: readOnlyBank()` function. That would be a good place to put a breakpoint.

However, even without *valgrind*, this particular bug is easy to find under the *debugger*. The steps are simple:

```
$ cd src/Debug
$ gdb ./yoshimi
(gdb) r -a -A
```

Then repeat the steps above that trigger the bug. One sees

```
Program received signal SIGSEGV, Segmentation fault.
```

Issue the command "backtrace" at the `(gdb)` prompt. There will be a list of stack frames starting at 0. Frame 1 is in *Yoshimi*, so issue the command "frame 1", a see

```
if (strlen(tmp) > 2) ...
```

`tmp` is a null pointer here; we need to add an initial check for the null pointer there to avoid triggering the crash.

16 Summary

In summary, we can say that you will absolutely love *Yoshimi*.

There are some topics that this document does not yet treat:

- Changing the colors and style of the GUI, as seen in some versions of *ZynAddSubFX*. This does not yet seem possible, because the synth part is not truly separate from the GUI, and the *Yoshimi* developers have not found it easy to recode the FLTK GUI.

17 References

The *Yoshimi* references.

References

[1] Will J. Godfrey *A discussion of making Bank/Root specifications more regular.* http://sourceforge.net/p/yoshimi/mailman/message/33200765/ 2014.

[2] Chris Ahlstrom *A Yoshimi Cookbook.* https://github.com/ahlstromcj/yoshimi-cookbook/ 2015.

[3] Cormi *Cormi Collection* https://www.freesound.org/people/cormi/ His cormi57.wordpress.com site has a lot of other nice sound material, as well. 2015.

[4] Straulino *A collection of instruments* http://www.straulino.ch/zynaddsubfx/ http://www.straulino.ch/zynaddsubfx/Drums_DS_v2012-12-08.zip http://www.straulino.ch/zynaddsubfx/ZynAddSubFX_Natural_Drum_Kit_Demo_v2012-06-22.ogg

[5] "folderol" *A collection of instruments* http://www.kara-moon.com/forum/index.php?topic=762.0

[6] Will Godfrey *Will Godfrey's Music* http://www.musically.me.uk See the "Bits 'n Stuff" link especially.

[7] Will J. Godfrey *A discussion of licensing issues with Youshimi and ZynAddSubFX components, and FLTK library versions.* http://sourceforge.net/p/yoshimi/mailman/yoshimi-devel/ 2015.

[8] caonoize *ZynAddSubFX by paulnasca: Downloads, Banks, Patches, etc.* http://www.kvraudio.com/product/zynaddsubfx_by_paulnasca/downloads 2015.

[9] mmxgn *Instruments made with zynaddsubfx/yoshimi* https://github.com/mmxgn/instruments-zyn 2011.

[10] Rakarrack team *The download site for the Rakarrack software effects engine.* http://rakarrack.sourceforge.net/ 2015.

[11] LinuxMusicians newsgroup *Ring Modulation in ZynAddSubFX* http://linuxmusicians.com/viewtopic.php?f=1&t=8178 2012.

[12] Manuel Op de Coul ¡coul@huygens-fokker.org¿ *The Scala Musical Tuning Application.* http://www.huygens-fokker.org/scala/ Scala is a powerful software tool for experimentation with musical tunings, such as just intonation scales, equal and historical temperaments, microtonal and macrotonal scales, and non-Western scales. 2014.

[13] Sharphall *How to create drum sounds in ZynAddSubFX or Yoshimi, Part 1* http://sharphall.org/docs/zynaddsubfx_yoshimi_drum_tutorial.php Never got continued, unfortunately.

[14] Gordon Reid *Synth Secrets: Creative Synthesis* http://www.soundonsound.com/sos/allsynthsecrets.htm 1999-2004.

[15] Graham *A small musical website* http://x31eq.com and specifically http://x31eq.com/zasf

[16] Yoshimi team abrolag@users.sourceforge.net *The download site for the Yoshimi software synthesizer.* http://yoshimi.sourceforge.net/ 2015.

[17] Yoshimi team *The alternate location for the Yoshimi source-code.* https://github.com/abrolag/yoshiminow. 2015.

[18] Mark McCurry, Paul Nasca (ZynAddSubFX team) *The download site for the ZynAddSubFX software synthesizer.* http://zynaddsubfx.sourceforge.net/ 2015.

[19] Paul Nasca? *The download site for the ZynAddSubFX banks, instruments, parameters, and demos* http://zynaddsubfx.sourceforge.net/doc/instruments/ 2009.

[20] ZynAddSubFX team. *Building ZynAddSubFX* http://zynaddsubfx.sourceforge.net/Doc/#_appendix_b_building_zynaddsubfx 2009.

[21] AMSynth team. *Provides a number of OGG demos of ZynAddSubFX sounds. It also includes the author's own "Vanilla" bank, and links to additional patch collections and demonstration videos.* http://www.amsynth.com/zynaddsubfx.html

[22] Mark McCurry, Paul Nasca *ZynAddSubFX online manual.* http://zynaddsubfx.sourceforge.net/Doc 2015.

[23] Paul Nasca *Original ZynAddSubFX manual, ODT format.* http://linux.autostatic.com/docs/zynaddsubfx_manual-v0.1.odt 2011.

[24] Paul Nasca *Original ZynAddSubFX manual, PDF format.* http://linux.autostatic.com/docs/zynaddsubfx_manual-v0.1.pdf 2011.

[25] Paul Nasca, Mark Murray *The download package for the ZynAddSubFX source-code* http://downloads.sourceforge.net/project/zynaddsubfx/zynaddsubfx/2.5.0/zynaddsubfx-2.5.0.tar.gz

Index