A Yoshimi Cookbook

Chris Ahlstrom
(ahlstromcj@gmail.com)

October 30, 2020

Contents

1 Introduction **3**
1.1 Project Structure . . . 3
1.2 XML File Compression . . . 3
1.3 What Game Shall We Play Today? . . . 3

2 Concepts **4**
2.1 Concepts / Terms . . . 4
2.1.1 Concepts / Terms / cent . . . 4
2.1.2 Concepts / Terms / ring modulation . . . 5

3 Installing Cookbook Resources **5**
3.1 Resources / Paths . . . 5

4 Fun With Presets **5**
4.1 Presets / Paths . . . 5
4.2 Presets / Pick a Patch . . . 6
4.3 Presets / Pick a Bank . . . 8

5 Creating Instruments **8**
5.1 Bells . . . 9
5.1.1 Bells by Voices . . . 9
5.1.2 Ring Modulation with 440 Hz Tone . . . 11
5.1.3 Complex Bells by Ring Modulation . . . 12
5.2 Ethnic . . . 13
5.2.1 Ethnic / Steel Drums . . . 13
5.3 Drums . . . 14
5.4 Chromatic Percussion . . . 17
5.5 Effects . . . 19
5.5.1 Effects / Dial Tones . . . 19
5.6 Piano . . . 21
5.7 Leads . . . 21
5.8 Guitar . . . 21
5.8.1 Guitar / 12-String . . . 21
5.9 Strings . . . 22
5.10 Bass . . . 22

5.11 Saxophones . . . 22

6 Banks and General MIDI **22**
6.1 Creating a Basic GM Bank . . . 23
6.2 Root Paths . . . 26
6.3 Using a Basic GM Bank . . . 28

7 Converting Tunes for Yoshimi Playback **29**
7.1 Tune / "Before You Accuse Me" . . . 29

8 Usage Notes **31**
8.1 Notes / Instrument Design . . . 32

9 Sequencers **32**
9.1 Sequencers / Sequencer64 . . . 32
9.1.1 Sequencers / Sequencer64 / ALSA . . . 33
9.1.2 Sequencers / Sequencer64 / JACK . . . 34
9.1.3 Sequencers / Sequencer64 / "Out There" . . . 36
9.1.3.1 Initial Installation . . . 36
9.1.3.2 Start from Scratch . . . 36
9.1.3.3 Load Example XMZ File . . . 37
9.1.3.4 Playing Out_There . . . 37
9.1.4 Sequencers / Sequencer64 / Adapting a Tune . . . 38
9.1.5 Sequencers / Sequencer64 / Basic Composing . . . 38

10 Summary **38**

11 References **38**

List of Figures

1 Navigating to a Preset . . . 6
2 The Penvamplitude Preset . . . 8
3 Typical Steel Drum Spectrum . . . 13
4 Steel Drum SUBsynth Configuration . . . 14
5 Natural Drum Kit from DS 2 . . . 15
6 Natural Drum Kit Keyboard Layout . . . 16
7 General MIDI Drum Kit Keyboard Layout . . . 17
8 Copy to Clipboard Preset . . . 18
9 Kit Edit Dialog for DTMF Kit . . . 20
10 DTMF Layout on the Keyboard . . . 21
11 Bank Root Paths . . . 27
12 Two Banks, Demo and Basic GM . . . 27
13 A General MIDI Basic Bank . . . 28
14 Yoshimi Audio Connection, JACK . . . 35
15 Yoshimi MIDI Connection, JACK . . . 36

List of Tables

1 Simple Bell Tones 9
2 Ring Modulation Bell Tones 12
3 DTMF Frequencies Table 19
4 GM Basic Files 23

1 Introduction

This document is a follow-on to the author's "A Yoshimi User Manual" [15]. The user manual attempts complete coverage of the user-interface and concepts behind *Yoshimi*. This cookbook attempts to provide recipes to solve some common problems in getting *Yoshimi* to perform at its best for the user.

1.1 Project Structure

The "Yoshimi Cookbook" project consists of two parts:

- The source material for this document.
- A self contained "yoshimi" configuration and data section to support the examples in this cookbook.

The documentation source-files are provided in the `tex` directory. They are use to create the cookbook via Makefiles and the external "latexmk" project. The result of a "make" is a new PDF of the cookbook in the `pdf` directory. The latest PDF is always provided there so that one does not have to install the external projects needed to create it.

The configuration, banks, presets, and instrument files can be used to supplement or replaces the user's own configuration and data files.

1.2 XML File Compression

One thing we recommend for following this cookbook is to set the *Yoshimi* compression level to **0**. This makes it a lot easier to text-edit the file to read its contents and understand them. To make this setting:

1. Navigate to **Menu / Yoshimi / Settings.../ Main settings**.
2. Click on the left arrow of **XML compression level** until the value of 0 appears.
3. Click on the **Save and Close** button.
4. Restart *Yoshimi*, then navigate again to this dialog to verify that the setting has been saved.

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. But these days, that should not be a problem. Maybe if one is running *Yoshimi* on an old *Raspberry Pi* device. On the other hand, for normal usage, it is best to leave the compression setting at 3, since decompression is faster than reading larger amounts of data from a file.

1.3 What Game Shall We Play Today?

There are a number of recipes that are hinted at in the user manual, but that solve problems that the author has encountered while using *Yoshimi*.

- **Banks and MIDI**. *Yoshimi* has had recent modifications to support bank-switching and using program-change messages to make *Yoshimi* a more flexible MIDI playback tool.
- **General MIDI**. It should be possible to set up one or more banks that are General MIDI compliant.
- **Usage of Modulators**. *Yoshimi* provides a number of modulation setups, but it isn't clear how to use them, especially the ring modulator.
- **Creation of Special Instruments**. There are some instruments that don't seem to have decent *ZynAddSubFX*/*Yoshimi* instrument files. Some examples, based on our desires: sitar, koto, bagpipes, steel drums, telephone tones, middle-eastern pipes, Japanese instruments, that steel-whip percussion sound heard in many songs....
- **More!**.

This document explains how to do some of the above tasks.

2 Concepts

This section, like its counterpart in our *Yoshimi User Manual*, presents some useful concepts, while keeping them out of the way.

2.1 Concepts / Terms

This section doesn't provide comprehensive coverage of terms. It covers mainly terms that puzzled the author at first or that are necessary to understand the recipes.

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.

Since the detuning provided in *Yoshimi* is based primarily on *cents* (and octaves), it pays to understand cents. If a given frequency `f`' is offset from another frequency `f`, the relationships between them in *semitones* (`s`) are:

$$f' = f * 2^{s/12}$$

$$s = 12log(f'/f)/log2$$

Note that the log operator is the base-10 log. In cents, these relationships become:

$$f' = f * 2^{s/1200}$$

$$c = 1200log(f'/f)/log2$$

where `c` is cents. These relationships hold whether `f`' is less than or greater than `f`. They provide an easy way to determine how much to detune a frequency in *Yoshimi*.

2.1.2 Concepts / Terms / ring modulation

Ring modulation is the multiplication (heterodyning) of two signals, and is named for the ring-like circuit that can produce it. When two tones, `f1` and `f2`, with `f1` <`f2`, are multiplied, the spectrum changes from {f1, f2} to {f2-f1, f2+f1}. Depending on the ratio of `f1` and `f2`, the sounds can be bell-like or very discordant. *Yoshimi* provides ring modulation, as well as other forms of modulation.

3 Installing Cookbook Resources

This section explains how to set up the *Yoshimi Cookbook* files into one's "home" directory so that they are easily accessible and easily modifiable.

First, let's get the files copied to the home directory. Currently, the *Yoshimi Cookbook* resources are not installable. One simply unpacks this project, or git-clones it to the desired directory. Let's call it `yoshimi-cookbook`. This name represents where one has cloned the project.

```
cd .../yoshimi-cookbook
cd ..
cp -r yoshimi-cookbook ~/.config
```

These commands end up "installing" the cookbook resources to `~/.config/yoshimi-cookbook`.

3.1 Resources / Paths

How do we copy the yoshimi-cookbook stuff to our "home" directory?

Menu / Paths / Bank Root Dirs

Push the button **Add root directory...**.

Select:

`/home/username/.config/yoshimi-cookbook/banks`

4 Fun With Presets

This section expands on its counterpart in the *Yoshimi User Manual*, showing how presets can be employed to re-use blocks of settings. In this short tutorial, we will grab and save a number of presets, and then use them to partly reconstitute the sound.

4.1 Presets / Paths

First, one needs to make sure to have one's `~/.config/yoshimi/presets` directory in the set of preset directories. Navigate to the **Menu / Paths / Preset Dirs...** menu entry and see if it is there. Add it if necessary.

Second, make sure that there are some root directories available. Navigate to the **Menu / Paths / Bank Root Dirs...** menu entry and make sure at least the first of these directories are present:

```
/usr/share/yoshimi/banks
/usr/share/zynaddsubfx/banks
```

Replace `/usr` with `/usr/local` if *Yoshimi* was built from source code. Also note that the second directory will not be present if *ZynAddSubFX* is not installed.

4.2 Presets / Pick a Patch

Now navigate to **Main window / Patch Sets / Show Patch Banks...**. Pick (left-click) item **60. Pads** (it might be associated with a different number in one's own installation of *Yoshimi*). In the **Pads** bank window that opens, pick **8. Resonance Pad1**. This instrument will now be active in the main window for the current **Part**. In the main window, click the **Edit** button to the right of the part name. In the **Instrument Edit** dialog that comes up, click the **Edit** button for the desired part of the instrument (e.g. **ADDsynth**). The following figure shows a set of windows that could be open in a *Yoshimi* instrument-editing session:

Figure 1: Navigating to a Preset

Take a moment to note each of the windows that are open now. Play the virtual keyboard for awhile, until the sound of this patch/instrument is memorized (we want to reconstruct this sound later).

Now let's save some of the instrument's presets. Click the *Yoshimi* main window's **Edit** button. Oddly, this Pad instrument uses only the ADDsynth engine. Press the ADDsynth item's **Edit** button. Up comes the complex, multi-panelled window for the ADDsynth setup. Note the blue **C** and **P** buttons in each of the following sub-panels:

- **Amplitude Env**
- **Amplitude LFO**
- **Filter Params**
- **Filter Env**
- **Filter LFO**
- **Frequency Env**
- **Frequency LFO**
- The whole **ADDsynth Window**

Let's save them all! Click on each **C** button, then fill in the **Copy to Preset** text field with the name *"demo"*, and then press the **Copy to Preset** button. All of the file-names that result will start with `demo`.

Then click on the **Show Voice Parameters** button, then click on the **C** button to save the one enabled sub-panel for that instrument's voice, **Frequency LFO**. Note that the *demo* is already saved in this preset of type *Plfofrequency*. So, if we want to save this, we'd use a name other than *demo*. Click **Cancel**.

Now click on the **C** button for the whole ADDsynth voice window. Go ahead and do **Copy to Preset** named "demo".

Now look in "`~/.config/yoshimi/presets`", and see all the saved files:

```
$ ls ~/.config/yoshimi/presets
demo.ADnoteParametersn.xpz (whole ADDsynth preset?)
demo.ADnoteParameters.xpz (whole ADDsynth Voice preset?)
demo.Penvamplitude.xpz
demo.Penvfilter.xpz
demo.Penvfrequency.xpz
demo.Pfilter.xpz
demo.Plfoamplitude.xpz
demo.Plfofilter.xpz
demo.Plfofrequency.xpz
```

Two of them are confounded with the same type, but slightly different file-names. How?

Now all of these presets are available for loading. Let's use them to take a basic sine wave, and build up its settings until we have something approximating the **8. Resonance Pad1** sound. Let's try to reconstruct the sound.

Load the *Yoshimi* **Simple Sound**. This sound can also be obtained via the **Instruments / Clear** command. Listen to **Simple Sound** via the virtual keyboard, and note how pure and tone-like it is. Then click the main window **Edit** button, then the ADDsynth **Edit** button. Now load all of the presets

above by clicking on each **P** button and then selecting the `demo` entry, as shown in the screen-shot (which shows the Penvamplitude preset).

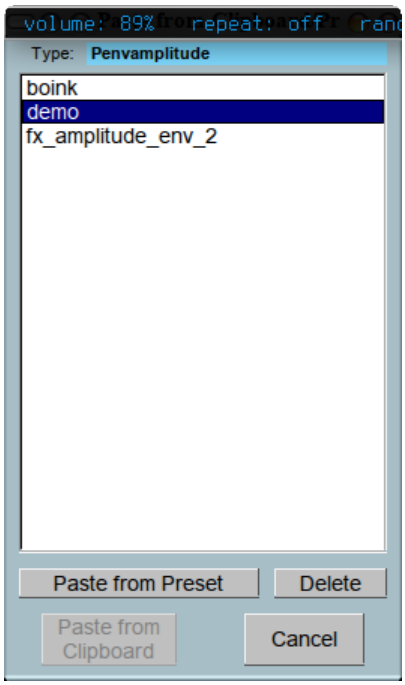

Figure 2: The Penvamplitude Preset

Then click the **Paste from Preset** button. After all this is done, the **Resonance Pad1** sound should be pretty much recovered. Pasting the ADDsynth overall preset seems to have the most effect. One must play a lot with the presets to fully understand how they work.

In summary, the presets are essentially keyed to the sub-panels. In naming them, you need to use a name that represents the sound that the preset contributes, via the settings in its sub-panels.

4.3 Presets / Pick a Bank

Now let's see if we can save an entire bank as a preset.

How do we copy the yoshimi-cookbook stuff to our "home" directory?

Menu / Paths / Bank Root Dirs

Push the button **Add root directory...**.

Select:

`/home/username/.config/yoshimi-cookbook/banks`

5 Creating Instruments

One of our goals in using *Yoshimi* is to support *General MIDI* (*GM*) to the greatest extent possible.

However, no banks have been created with GM in mind. And many of the instruments, though given names that indicate what they are intended to be, fall well short of being recognizable per their name;

they should be doable with a complex synthesizer like *Yoshimi*, but perhaps have not been done yet, or we have not found them.

It is true that there are a vast number of *Yoshimi* and **ZynAddSubFX** patches/parts/instruments out there. The author had attempted a survey of them, and the task was all but impossible, since there are so many of them. Still, many candidates have been identified. Other candidates might be suitable with a little tweaking.

Here are a number of categories of instruments for which we want to assemble an improved set of instruments:

1. **Bells**
2. **Ethnic**
3. **Drums**
4. **Effects**
5. **Piano**
6. **Leads**
7. **Guitar**
8. **Strings** (individual and ensemble)
9. **Bass**
10. **Saxophones**

These recipes will be found in this cookbook project; the `banks` directories are found in the following directories:

```
yoshimi/banks
yoshimi/banks/demo
```

5.1 Bells

The bells patches we've heard so far are nice, but a bit anemic.

Good bell patches are easier with ring modulation, done right. We're not sure if there are any such patches extant; please send them to us if have some in hand.

In the meantime, creating bells is a good excuse to master *Yoshimi*'s ring modulator. However, we will first learn how to create a reasonable, clangy bell using just a few voices in an ADDsynth part, and no need for modulation.

5.1.1 Bells by Voices

The following table comes from a tutorial ([2]). Along with a spectrum shown in reference [3], it allows us to recreate a simple, but realistic bell. In this table, `F` represents the fundamental frequency, i.e. the note being played.

Table 1: Simple Bell Tones

Wave Number	Frequency	Cents Offset	Relative Amplitude
1	0.56F	-1000	0.5
2	0.92F	-140	1.0
3	1.19F	+300	0.5

4	1.71F	+930	0.25
5	2.00F	+1200	0.125
6	2.74F	+1745	0.125
7	3.00F	+1901	0.125
8	3.76F	+2290	0.125
9	4.00F	+2400	

Note that the frequencies are relative to the fundamental frequency (F). Also note that wave 2 (close to F) can be missing, and the sound still is bell-like.

The file `yoshimi/banks/demo/Bells-simple-addsynth.xiz` is the result of following the steps below. We start, as usual, with a newly-started *Yoshimi* instance.

1. Open the ADDsynth editing window by clicking the **Edit** button in the bottom panel, and then clicking the ADDsynth **Edit** button in the edit window.
2. Click on the **Show Voice Parameters** button. Note that it is **Current Voice 1**, and it should be enabled.
3. For voice 1, make the following settings:
    1. **Octave**: Set to 0.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to -1000 approximately.
4. Go to voice 2, and make the following settings:
    1. **Octave**: Set to 0.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to -140 approximately.
5. Go to voice 3, and make the following settings:
    1. **Octave**: Set to 0.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to 300 approximately.
6. Go to voice 4, and make the following settings:
    1. **Octave**: Set to 0.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to 930 approximately.
7. Go to voice 5, and make the following settings:
    1. **Octave**: Set to 1.
    2. **Detune Type**: Set to Default.
    3. **FREQUENCY Detune**: Set to 0.
8. Go to voice 6, and make the following settings:
    1. **Octave**: Set to 1.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to 545 approximately.
9. Go to voice 7, and make the following settings:
    1. **Octave**: Set to 1.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to 700 approximately.
10. Go to voice 8, and make the following settings:
    1. **Octave**: Set to 1.
    2. **Detune Type**: Set to E1200cents.
    3. **FREQUENCY Detune**: Set to 1090 approximately.

These settings then end up roughly matching the settings of the first 8 waves in table 5.1.1 ("Bells by Voices") on page 9. This instrument isn't perfect. It's not quite equally tempered, though close. The character of the tone changes a bit as the notes get higher. One can fiddle with the relative amplitudes of the various voices to change the character of this sound.

5.1.2 Ring Modulation with 440 Hz Tone

Now for an initial demonstration of ring modulation. This demonstration does not quite create a bell tone, but does show the sound of modulation.

Start with a fresh *Yoshimi* and a cleared instrument ("Simple Sound"). Open the virtual keyboard using the **virKbd** button. Click a key and verify that you can hear a tone. We'll use the middle C key (the "comma" on the PC keyboard) as a reference. We will call it the "C" note.

The following steps will set up two tones, voice 1 and voice 2, and voice 2 will use voice 1 as an external modulator. Note that you can accomplish most of these steps by loading the project file `yoshimi/banks/demo/Bells-440-ring-modulation.xiz`, but use that only as a last resort.

1. Open the ADDsynth editing window by clicking the **Edit** button in the bottom panel, and then clicking the ADDsynth **Edit** button in the edit window.
2. In the **Amplitude Env** sub-panel, increase the **D.dt** and **R.dt** to give the current sound a nice slow decay.
3. Click on the **Show Voice Parameters** button. Note that it is **Current Voice 1**, and it should be enabled.
4. Switch to **Current Voice** number 2 and enable it. Play the "C" note, and observe that it is the same frequency, but louder.
5. Move the **FREQUENCY Detune** slider a bit, and play the "C" note. It should sound the same as before, but change slowly in amplitude, as heard and as seen on the **VU meter**. Try to set the detune back to 0; this is easier if you highlight the tuning knob and use the left or right arrow keys.
6. In the **MODULATOR** section of voice 2, for **Type**, select the **RING** value. (However, feel free to select one of the other modulators, to experiment, once you've mastered the ring modulator.) Press the "C" key again, and notice that the tone character changes a bit. This is due to the internal modulator.
7. For **External Mod.** for voice 2, select **Ext.M** 1, to use the voice 1 as the internal modulator. The "C" note may change in character, but only slightly. Apparently the default internal modulator is the same as the default external voice 1 waveform.
8. To actually hear some modulation, we have to separate the frequencies of voice 1 and voice 2. Click the **440Hz** check-box in the **FREQUENCY** section of voice 1. Press the "C" key and verify hearing a two-tone signal, somewhat like a phone tone.
9. Now go back to voice 2 and click the **Change** button to bring up the ADDsynth oscillator dialog.
10. Move the slider to maximum for harmonic 10. Press the "C" key and verify the new sound (a bit like a car horn). Set the sliders back to 0, and "C" will be a single tone again.
11. Change the **Octave** values of voice 2 in its **FREQUENCY** section and listen to the effects.

Now we need to see if we can apply modulation across instruments. Sadly, this does not seem to be possible.

Increase the **D.dt** and **R.dt** values of the main **Amplitude Env** to give this sound the onset and decay of a bell, and it then sounds less abstract, and more like a bell. Of course, this kind of bell is even less tunable than the simple bell of the previous section.

Another thing to try with this setup is to simply change voice 2 to use different types of modulators besides **RING**. **MORPH** sounds basically identical to **RING**. **PM** seems to expose higher harmonics, making the sound louder and brighter. **FM** sounds similar to PM, but softer and smoother. **PITCH** is disabled.

Another experiment is to disable the modulator (voice 1 here) and see how that changes the sound; all it should do is drop voice 1 from the spectrum – voice 1 will still be used as the modulator.

Finally, by adding a slow decay to this sound, it becomes amazingly more bell-like.

5.1.3 Complex Bells by Ring Modulation

The next step is to make the bells more complex, by combining the methods of the previous two sections. Recall table 5.1.1 ("Bells by Voices") on page 9. It shows the 9 frequencies in the simple bell spectrum, though we could define only 8 of them. How can we best add extra frequencies? We can ring-modulate the higher frequencies against one of the lower frequencies.

Table 2: Ring Modulation Bell Tones

Wave Number	Frequency	Mod Frequency	f2-f1	f2+f1
1	0.56F	—	—	—
2	0.92F	0.56F	0.36F	1.48F
3	1.19F	0.56F	0.63F*	1.75F*
4	1.71F	0.56F	1.15F*	2.27F
5	2.00F	0.56F	1.44F	2.56F
6	2.74F	0.56F	2.18F	3.30F
7	3.00F	0.56F	2.44F	3.56F
8	3.76F	0.56F	3.20F	4.32F

The asterisk marks frequencies that are close to existing frequencies. Luckily, there are only three of them, so our modulation should add a good number of frequencies.

1. Load the file `yoshimi/banks/demo/Bells-simple-addsynth.xiz` to save a lot of steps. The next steps add voice 1 as a ring modulator for voices 2 through 8.
2. Open the ADDsynth editing window by clicking the **Edit** button in the bottom panel, and then clicking the ADDsynth **Edit** button in the edit window.
3. Click on the **Show Voice Parameters** button. Note that it is **Current Voice 1**, and it should be enabled.
4. Go to voice 2 and do the following steps:
    1. In the **MODULATOR** section (greyed out), change the **Type** from **OFF** to **RING**.
    2. Changes the **External Mod.** dropdown from **Off** to **ExtMod. 1**.
5. Go to voice 3 and repeat those steps. Note how all the voices below the current voice become available as modulators.

We saved the result in the file `yoshimi/banks/demo/Bells-ringmod-addsynth.xiz` for safe-keeping.

QUESTION: If one loads and instrument and tinkers with it, but do not save it, does *Yoshimi* save it on exit anyway?

5.2 Ethnic

We found a simple steel drum instrument, but think we might do better, creating one using ADDsynth and one using PADsynth.

Instruments we have not found, and would like to synthesize, are: bagpipes and arabic pipes.

5.2.1 Ethnic / Steel Drums

There is a steel-drum instrument that ships with *Yoshimi*:

`/usr/share/yoshimi/banks/The_Mysterious_Bank/0122-pseudo_steeldrums.xiz`.

It is an ADDsynth module comprised of three voices:

1. A **Unison**-enabled voice of **Size** = 10 and a **Frequency Spread** of 44.6 cents.
2. Another voice that is exactly the same as the first, except that it has its **Amplitude Env** sub-panel enabled, to add more volume and character to the instrument, it is stronger on the right, and, most importantly, an octave higher.
3. Another voice that is exactly the same as the second, except it is an octave lower than voice 0.

If voice 2 and 3 are disabled, the instrument still sounds reminiscent of steel drums, so obviously the overall amplitude envelope is important.

Can we do better? Well, the instrument above sounds too pristine. We should be able to add some "dirt" to the instrument to make it sound more lively.

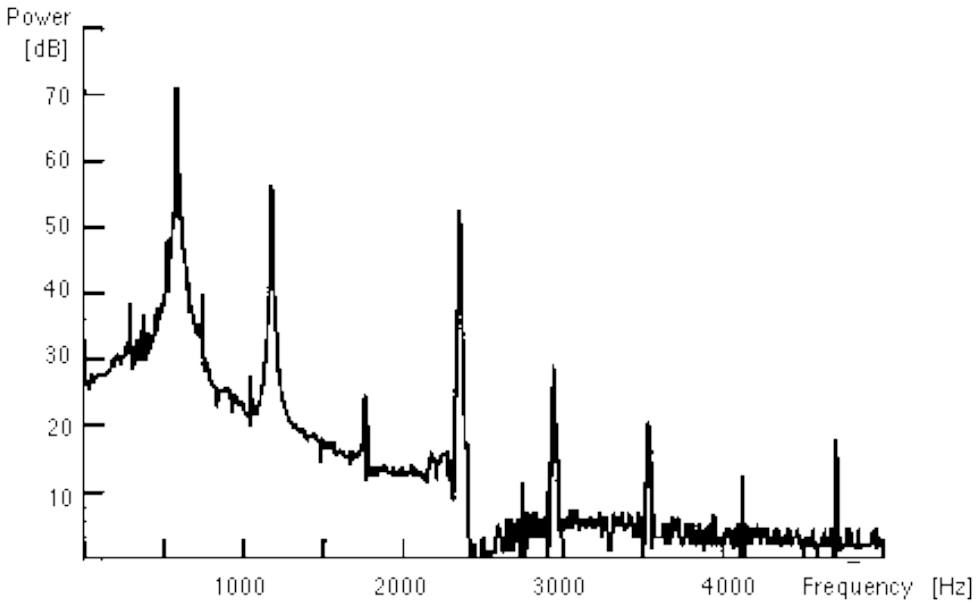

Figure 3: Typical Steel Drum Spectrum

Taking a cue from this figure, our steel drums extend the original by adding a couple more tones at octave intervals. Also, some slight detuning was introduced to add to the flavor. We could probably add a couple more, and carefully contour their amplitude levels to match the spectrum levels shown above.

To hear the ADDsynth steel drum sound, load the file `yoshimi/banks/demo/Add_Pseudo_Steel_Drums.xiz`.

Not content with that, with our hands behind our back, we pull SUBsynth from a hat. The SUBsynth settings for a steel drum are shown in the following figure:

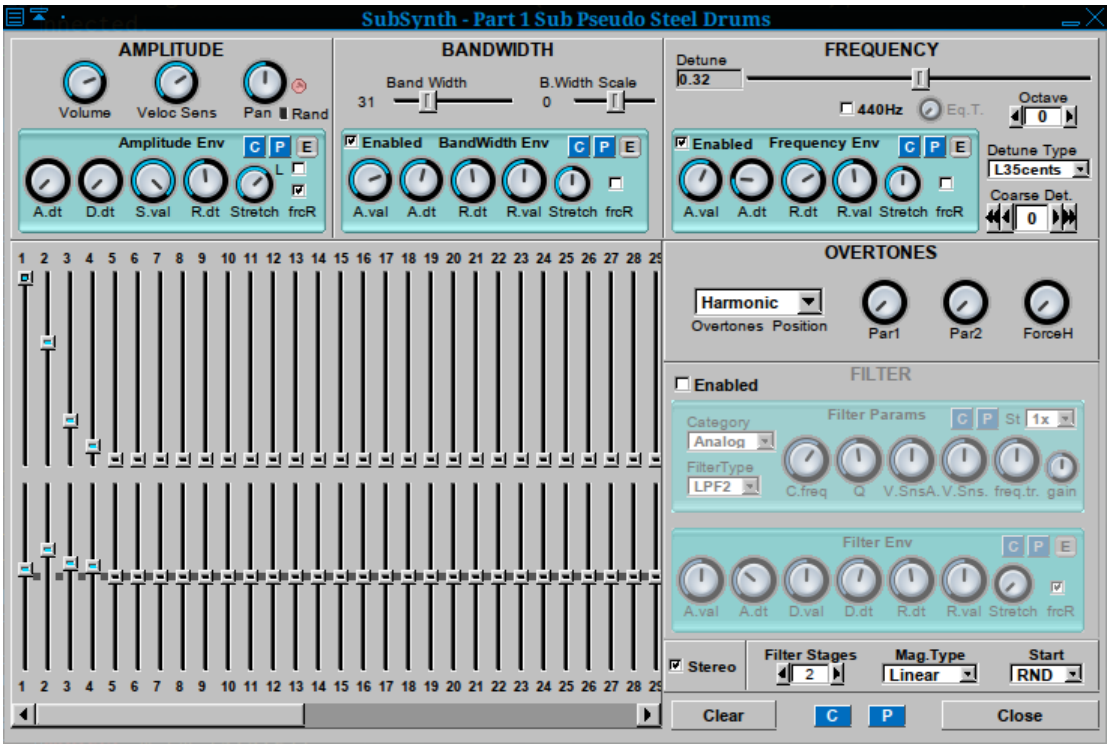

Figure 4: Steel Drum SUBsynth Configuration

Note the top box of slider controls. It sets the amplitudes of the harmonics, and should vaguely resemble the spectrum diagram. The lower box of slider controls sets the bandwidth of each harmonic, with the fundamental frequency being very narrow.

To hear the SUBsynth steel drum sound, load the file `yoshimi/banks/demo/Sub_Pseudo_Steel_Drums.xiz`. It's pretty nasty.

However, the best steel drum patch we've encountered so far is the `banks/olivers-100/0029-Steel Drums.xiz` from the installed banks of *ZynAddSubFx*, and that's the one we now copy to our cookbook project's `yoshimi/banks/gm-basic/0115-Steel Drums.xiz` GM slot.

5.3 Drums

We want a decent drum kit that attempts to fill in the gaps for a GM-compliant drum kit with solid sounds, with the help of an existing kit.

It turns out that a "Natural Drum Kit", which we'd found separately on the Internet (from Dario Straulino) a long time ago, is now part of the instruments installed with *Yoshimi*. But long ago we used some of the sounds from various kits to create our own "natural drum kit", and extended some of the sounds across more (pitched) keys so that any MIDI drum note would produce *some* sound. We also made sure the sounds were laid out in GM format as much as possible.

Fire up *Yoshimi* and load the instrument stored in `yoshimi/banks/demo/Natural_Drum_Kit_Basic.xiz`, and we'll walk through it. Click the **Edit** button in the bottom panel, and then click the **Kit Edit**

button.

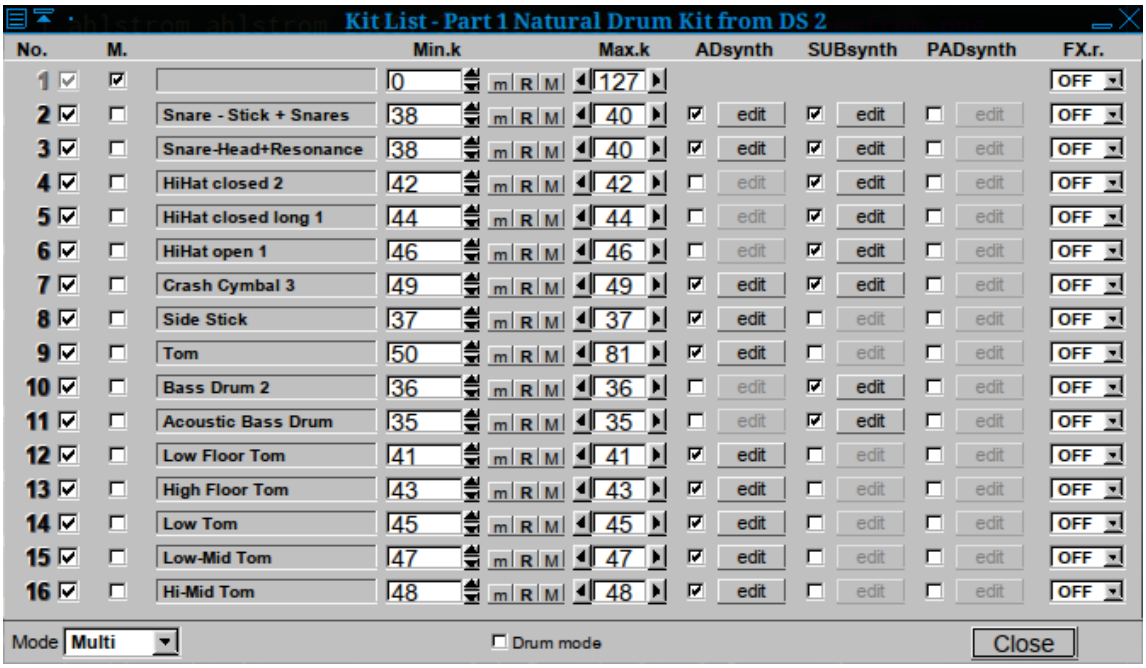

Figure 5: Natural Drum Kit from DS 2

Item 1 is the master control for the whole kit, determining the range of keys that it covers and the effect (if any) it goes through.

Item 2 and **Item 3** provide the two parts of the "natural" snare drum. Both parts are composed of an ADDsynth and a SUBsynth section, and they provide 3 pitches of snare.

One distinguishing feature of the "Snare - Stick + Snares" item is it's only ADDsynth voice. Clickthe ADDsynth **Edit** button for item 2 in the kit editor. Then Click on the **Show Voice Parameters** button. Note that the **Amplitude Env** sub-panel is enabled, and uses a Freemode envelope. The voice is locked on **440Hz**, but detuned to be a lot lower than that. The **Voice Oscillator** (click on the **Change** button) uses a **Base F.** (function) set to **Power** to generate a spike pulse. There doesn't seem to be a way to temporarily mute an item, so we cannot hear what this item would sound like on its own.

The SUBsynth part of the "Snare - Stick + Snares" item provide an **Amplitude Env**, **Bandwidth Envelope**, and a rich set of alternate harmonics of very narrow bandwidth.

The "Snare-Head+Resonance" item's ADDsynth settings provide Freemode **Amplitude Env** and **Frequency Envelope** settings with the voice locked on **440Hz**, but detuned to be a lot lower than that. The **Voice Oscillator** is a sine wave. Voice 2 is set to **NOISE**, but is disabled. Something to try out? Indeed, we changed voice 2 to a sine wave, moved it up an octave, and now the the Snare keys have slightly different qualities, a barely noticeable pitch. A keeper!

The "Snare-Head+Resonance" item's SUBsynth settings are a set of harmonics with a **Bandwidth Env**.

Item 4 is the "HiHat closed 2" instrument, a SUBsynth-only item. It provides an **Amplitude Env**, a **Bandwidth Env**, and a **Frequency Env** centered around 440 Hz, offset downward 2 octaves. A number of harmonics are provided, obviously taken from some spectral diagram of a cymbal.

Item 5 is the "HiHat closed long 1" instrument, a SUBsynth-only item. It is the same as the "HiHat closed 2" instrument, but with a longer amplitude envelope.

Item 6 is the "HiHat open 1" instrument, a SUBsynth-only item. Much like the other hi-hats, but with a Freemode **Bandwidth Envelope**.

Item 7 provides the "Crash Cymbal 3" item. It is composed of an ADDsynth and a SUBsynth section. The ADDsynth provides a **Voice Oscillator** that, like **Item1**, uses a **Base F.** (function) set to **Power** to generate a spike pulse.

The SUBsynth section is similar to that of the "Snare-Head+Resonance" (**Item 3**), a set of harmonics with a **Bandwidth Env**.

Item 8 is the "Side Stick" instrument, an ADDsynth-only item. It's only voice, voice 1, is filtered white noise. One can increase the **Q** value of the filter to make it a bit metallic. One can lower the **C.freq** value to make it sound like a bigger stick.

Item 9 is the "Tom". It is just a tone with an AR frequency envelope. The puzzling thing is that it is locked to 440 Hz plus 2 octaves, but it can run the full length of keys 50 to 81 (so we can get "Extra toms") and gradually rise in pitch. How? Unlocking voice 1 seems to change nothing!

Item 10 is "Bass Drum 2", a SUBsynth patch. Like the SUBsynth part of the "Snare - Stick + Snares" item, it provides a Freemode **Amplitude Env** and **Bandwidth Envelope**, but only a small set of low harmonics, and a -3 **Octave** offset, one octave lower than the snare.

Item 11 is "Acoustic Bass Drum". It merely adds some more harmonics to the "Bass Drum 2" patch.

Item 12 is the "Low Floor Tom", an ADDsynth patch. Voice 1, like **Item 9**, is a pitched tone with a frequency envelope. Voice 2 is added to provide a noisy component.

Item 13 through **16** are also noisy toms. At some point, we'll try to use the XML files to determine how their pitches are determined. To difficult to compare the two GUIs without taking screen shots.

Okay, so this drum kit is a little tom-heavy!

Now, without defining more than one drum kit, we have only about 15 "drums" available to us in *Yoshimi*. So we filled in the missing drums with more "toms", just so some sound will be made. The frequencies of the upper toms get pretty crazy! Here's a diagram of the keyboard layout. Correlate it with figure 5 ("Natural Drum Kit from DS 2") on page 15, to understand the abbreviations.

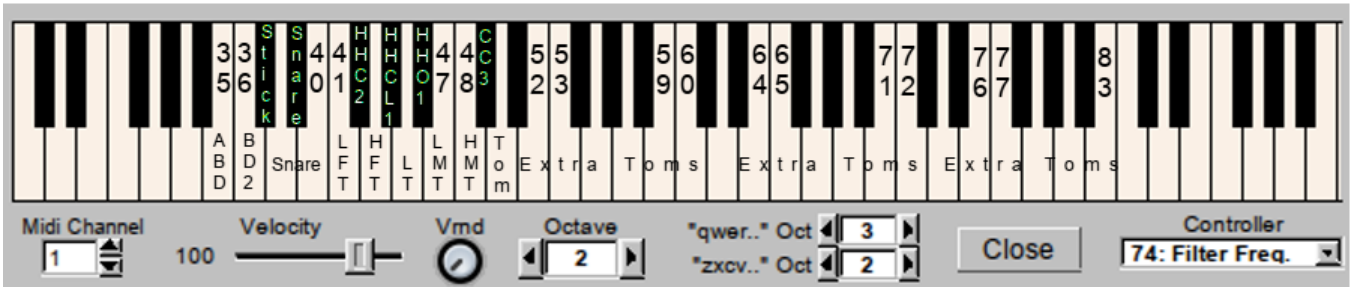

Figure 6: Natural Drum Kit Keyboard Layout

For your reference, here is the full GM drum layout. The diagram is taken from WikiMedia.org.

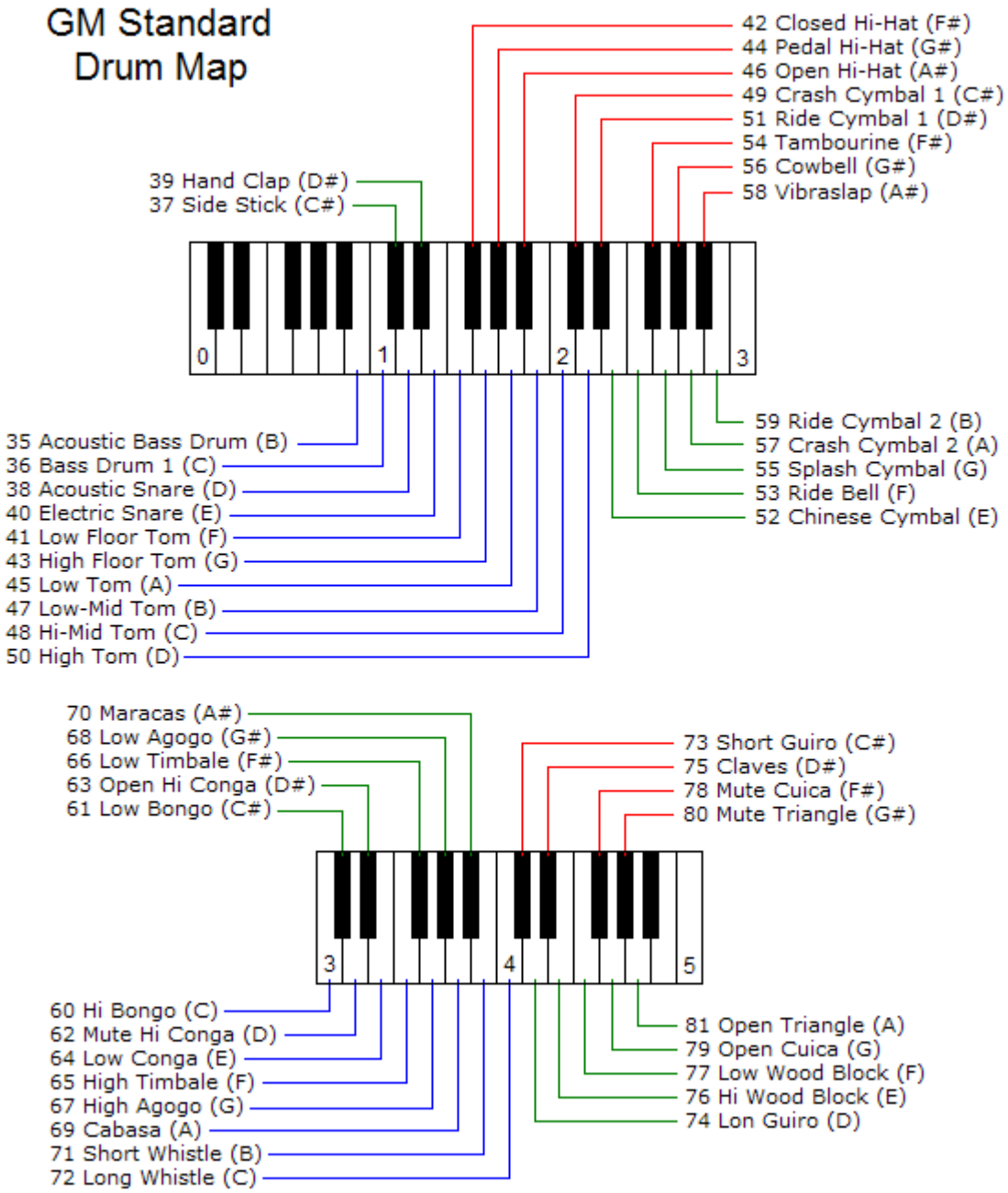

Figure 7: General MIDI Drum Kit Keyboard Layout

5.4 Chromatic Percussion

One of the big blanks in the gm-basic sound set are the various stand-alone percussion instruments: woodblock, taiko drum, melodic tom, synth drum, and reverse cymbal. These occupy GM slots 116 through 120.

One easy way to obtain these instruments is to find them in an existing drum kit, and extract each into

its own instrument file. Here is how we did it to obtain instrument 118, the Melodic Tom.

First, set up a presets directory to hold any presets one creates, and make it the default presets directory. For example, one can set this project's `yoshimi/presets` directory as the default location for preset files.

1. Navigate to **Menu / Settings... / Preset Dirs** (a tab in the settings dialog).
2. Click the **Add preset directory** button.
3. Navigate to the desired directory, select it, click **OK**.
4. Select the new preset directory, then click the **Make default** button.
5. Press the **Save and Close** button.
6. Restart *Yoshimi*.

Next, extract the Tom drum from the Natural Drum Kit for use as Melodic Tom GM patch:

1. Load the "Natural Drum Kit" instrument.
2. Click **Edit** to bring up the overall editor dialog.
3. Then click the **Kit Edit** button.
4. Move to item 9, "Tom", and click the **Edit** button to access its ADDsynth settings (the Global dialog).
5. Press the blue **C** button in the ADDsynth Global dialog.
6. Next to the grayed-out **Copy to Preset** button, give the setting (to be saved) a name: "Nat Drum ADDsynth Tom Part". See the figure below.
7. Press the now active **Copy to Preset** button.
8. Depending on what dialog one is in when pressing the **C** button, the file will be save in the default preset directory under a name like `demo.ADnoteParameters.xpz`.

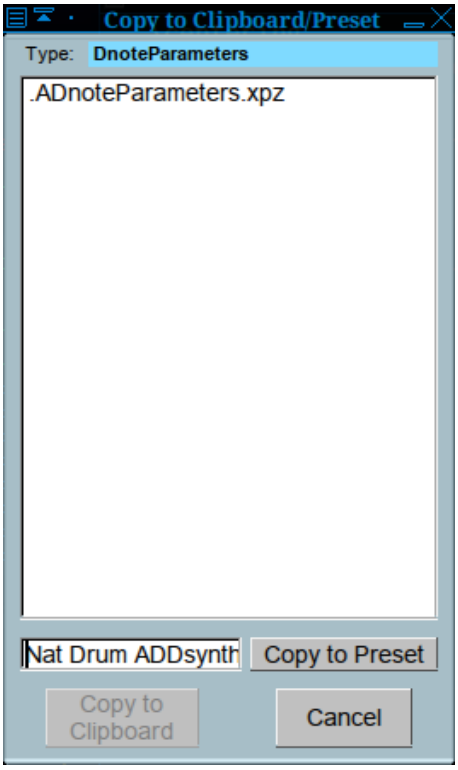

Figure 8: Copy to Clipboard Preset

One could then copy the **demo.ADnoteParameters.xpz** file to a file with a more descriptive name. Finally, the preset can then be loaded into a new instrument as follows:

1. Navigate to **Menu / Instrument / Clear Instrument**, and click **Yes** at the prompt.
2. Navigate to the desired settings dialog. In this case, it will involve clicking **Edit**, then the ADDsynth **Edit** button.
3. In the **ADDsynth Global** dialog, press the blue **P** button in that dialog, to open the **Paste From Clipboard** dialog.
4. Select the desired preset (that matches the current settings dialog). In this case, it will be **demo.ADnoteParam**
5. Press the **Paste from Preset** button.
6. Open the virtual keyboard using the **virKbd** button, and verify that the instrument sounds like the proper Tom kit item.
7. Go back to the Main edit dialog and change the **Type** to the best matching value; here, it would be **Chromatic Percussion**. Then fill in the **Author and Copyright** and **Comments** fields.
8. In the main *Yoshimi* window, right-click on the part name. In the prompt for **Instrument name**, type the name of the instrument/part; in this case, "Melodic Tom".
9. Navigate to **Menu / Instrument / Save instrument...**. Navigate to the desired directory, append the name (e.g. "Melodic Tom.xiz"), and save it.

It is wise to restart *Yoshimi* and verify that the new instrument can be loaded and played.

5.5 Effects

This section documents the various "effects" instruments we've created.

5.5.1 Effects / Dial Tones

We've created a nice dial-tone effect that we'll describe here. Dial tones consist of two notes, as shown by the **Low F** and **High F** columns in the following table.

Table 3: DTMF Frequencies Table

Tag	DTMF	Kit#	MIDI#	Low	Low F	Actual F	High	High F	Actual F
1	1	5	53	F3	697	705	F5	1209	1245
2	2	6	77	F5	697	705	F5	1336	1337
3	3	7	89	F6	697	698	F5	1477	1468
4	4	8	55	G3	770	770	G5 -	1209	1236
5	5	9	79	G5	770	776	G5 -	1336	1334
6	6	10	91	G6	770	773	G5 -	1477	1462
7	7	11	57	A3	852	855	G#5 +	1209	1245
8	8	12	81	A5	852	868	G#5 +	1336	1327
9	9	13	93	A6	852	866	G#5 +	1477	1480
	*	14	59	B3	941	948	A#5 +	1209	1257
0	0	15	83	B5	941	968	A#5 +	1336	1281
#	#	16	95	B6	941	950	A#5 +	1477	1480
A	A	–	–	A2	697	—	F5	1633	—
B	B	–	–	B2	770	—	G5 -	1633	—
C	C	–	–	C2	852	—	G#5 +	1633	—
D	D	–	–	D2	941	—	A#5 +	1633	—

b	busy	2	71	B4	480	472	B4 -	620	622
r	ringback	3	69	A4	440	440	A4	480	480
d	dialtone	4	65	F4	350	350	F4	440	440

This table is implemented in a *Yoshimi kit*. Each note in the kit is created by making an ADDsynth instrument with two voices. The lower voice generally corresponds to the note being play, with an offset, if needed, to achieve close to the proper frequency for the lower note of the DTMF tone. The second voice corresponds to the other note, and it is detune appropriately to achieve close to the proper frequency for the upper note of the DTMF tone.

The following figure shows the kit dialog:

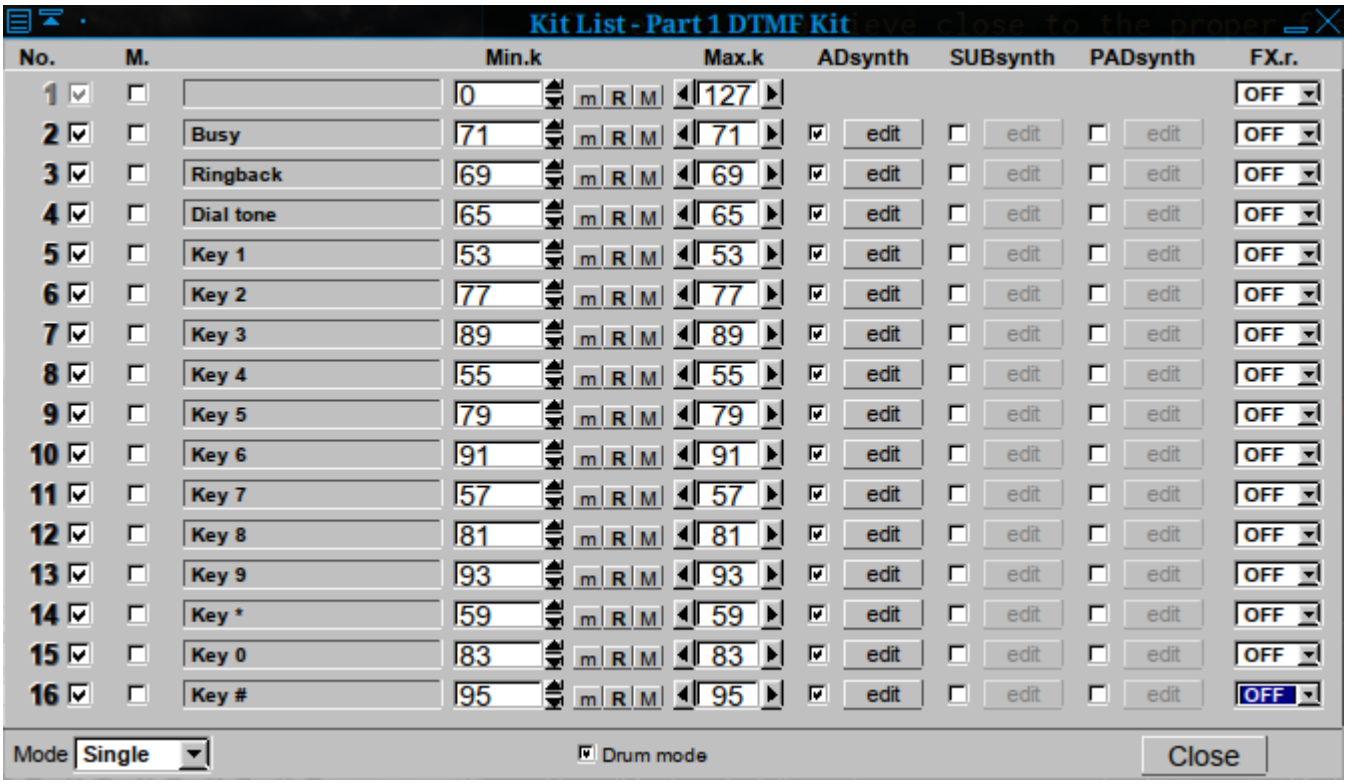

Figure 9: Kit Edit Dialog for DTMF Kit

To edit the kit, follow the steps below. If desired, open the instrument file `yoshimi/banks/demo/DTMF_Kit.xiz` to save some work.

1. Open the kit editing window by clicking the **Edit** button in the bottom panel, and then clicking the **Kit Edit** button in the edit window.
2. Make sure that the **Mode** is set to **Single**.
3. Make sure that the **Drum mode** is enabled.
4. For all 16 kit items, make sure that the **FX.r** selections are set to **OFF**.
5. For kit items 2 to 16, enable the the **ADsynth** check-box.
6. For kit items 2 to 16, perform the following procedure to set up the two frequencies correctly as per the table above:
    1. In the kit editor, click the **Name** field and enter the name of the DTMF tone item being edited.

2. In the kit editor, set **Min.k** and **Max.k** to the value of the note that is less than or equal to the lower note of the item listed in the table.
3. Click the ADDsynth **edit** button.
4. Click on the **Show Voice Parameters** button. Note that it is **Current Voice 1**, and it should be enabled.
5. Given the frequency for the note being edited, detune voice 1 to achieve the desired lower frequency.
6. Change to voice 2, and enable it.
7. Given the frequency for the note being edited, detune voice 1 to achieve the desired higher frequency.

The "Actual F" values were verifed using 24-Hz resolution (at 1200 Hz) in the spectrum analyzer built into Audacity. Sometimes it took a few tries to get the best possible frequency. We could list the detuning values in a table; for now, you can see the values we ended up using.

The following figure shows the layout on the *Yoshimi* virtual keyboard:

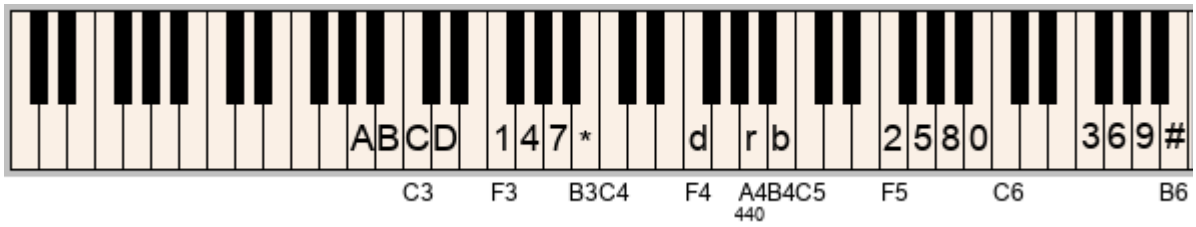

Figure 10: DTMF Layout on the Keyboard

5.6 Piano

TODO.

5.7 Leads

TODO.

5.8 Guitar

Yoshimi ships with some mediocre acoustic and steel guitar patches, but some amazing distortion guitars. We decided to try our hand at a 12-string guitar.

5.8.1 Guitar / 12-String

A 12-string guitar has pairs of strings, with each member of a pair an octave apart in frequency, for all strings but one. We've provided two versions of an attempt at a 12-string.

The first 12-string simulation is stored in `yoshimi/banks/demo/Low_Octave_12-String_Guitar_Harms.xiz`.

This patch provides one voice only, but the **Voice Oscillator** provides the fundamental and two harmonics. In this oscillator, **H.md** is set to **Pow**. **Base F.** is set to **Pulse**. Voice 1 also enables the **Filter**

Params to get a more acoustic sound. Still sounds a bit harpsichordy. Probably both strings need to be more complex.

The second 12-string simulation is stored in `yoshimi/banks/demo/Unison_12-String_Guitar_Harms.xiz`. This version is similar to the previous 12-string, but, in voice 1, **Unison** is enabled, and vibrator and phase randomness are added. This version sounds significantly richer.

5.9 Strings

(individual and ensemble)

TODO.

5.10 Bass

TODO.

5.11 Saxophones

TODO.

6 Banks and General MIDI

Banks are discussed quite heavily in the user manual [15]. Banks have evolved quite a bit in *Yoshimi*, and are a powerful way to manage instruments, and more amenable to automation than ever.

In this section, we will attempt to set up a basic bank that is compliant with the General MIDI specification. In order to do so, we will cherry pick instruments from the package that is provided when *Yoshimi* is installed, renaming them as needed to fit into the appropriate General MIDI slot, and add our own instruments and some others we have found.

One problem with *Yoshimi* instruments is the selection of the *best* instrument for a given General MIDI program number. There are simply too many to be able to evaluate them all.

For this recipe, the `banks` and `presets` will be stored in the following directories of this project:

```
yoshimi/banks
yoshimi/banks/demo
yoshimi/banks/gm-basic
yoshimi/presets
```

The user should set up these directories using the *Yoshimi* directory menu entries.

6.1 Creating a Basic GM Bank

Creating even a basic General MIDI bank is beset with issues, even if one has at hand a large number of pre-built instruments.

First, what is the purpose of the General MIDI specification? To provide a dependable set of instruments so that tunes will sound basically similar on different GM-compliant synthesizers. That's about it. It doesn't guarantee that the sounds are consistent, nor does it guarantee that they are all of high quality. The "FX", "Lead", and "Pad" instruments provide ambiguous descriptions that a wide range of sounds might fit. Getting a complete and high-quality set of sounds is extremely difficult.

Second, evaluating a large number of pre-built *Yoshimi* and *ZynAddSubFX* instruments takes a lot of work. We'd done some of this work for another project, and never finished. Nor is the naming of such instruments all that helpful; many of the file-names are misleading. Finding decent matches for a GM instrument takes time.

Third, there are many GM instruments for which we've been able to find no good *ZynAddSubFX*/*Yoshimi* counterpart. The only options are to pick a tolerable match, build a tolerable match oneself, or just plug in any old sound and wait for others to step up.

Nonetheless, let's forge ahead. The project file `contrib/instrument.ods` is a *LibreOffice* spreadsheet that represents some research in finding GM-compatible instruments. It's pretty bad; maybe 50% useful. We converted it to `contrib/gmcopy` to copy the files into the project directory `yoshimi/banks/gm/basic`. We show the banks in table 6.1 for convenience.

Table 4: GM Basic Files

General MIDI Instrument	Yoshimi Instrument Used
0001-Acoustic Grand Piano	SynthPiano/0033-Analog Piano 1
0002-Bright Acoustic Piano	SynthPiano/0034-Analog Piano 2
0003-Electric Grand Piano	SynthPiano/0143-Space Piano
0004-Honky-tonk Piano	SynthPiano/0068-Synth Piano 3 fat
0005-Electric Piano 1	Rhodes/0002-DX Rhodes 2
0006-Electric Piano 2	Rhodes/0007-Dig Rhodes
0007-Harpsichord	olivers100/0026-Harpsichord
0008-Clavinet	Misc Keys/0060-Clavinet 1
0009-Celesta	Bells/0002-Music_Box
0010-Glockenspiel	Bells/0011-Glass bells
0011-Music Box	Bells/0013-Tiny bells
0012-Vibraphone	Chromatic Percussion/0045-Vibes no_trem
0013-Marimba	Chromatic Percussion/0056-FM marimba
0014-Xylophone	Will_Godfrey_Collection/0001-Xylophone
0015-Tubular Bells	Chromatic Percussion/0097-Marimba 3
0016-Dulcimer	Plucked/0004-Plucked 4
0017-Drawbar Organ	Organ/0001-Organ 1
0018-Percussive Organ	Organ/0012-Organ 12
0019-Rock Organ	Organ/0068-Square Organ
0020-Church Organ	Organ/0061-Great Organ
0021-Reed Organ	Reed_and_Wind/0039-Reed 7
0022-Accordion	Organ/0097-Accordion Pad 1

Continued next page

General MIDI Instrument	Yoshimi Instrument Used
0023-Harmonica	Reed_and_Wind/0099-Sharp Reed
0024-Tango Accordion	Organ/0101-Accordion 1
0025-Acoustic Guitar nylon	Piano/0144-Soft Piano1
0026-Acoustic Guitar steel	Guitar/0065-Clean Guitar1
0027-Electric Guitar jazz	Guitar/0066-Electric Guitar
0028-Electric Guitar clean	Guitar/0133-Smooth Guitar
0029-Electric Guitar muted	Guitar/0035-Short
0030-Overdriven Guitar	Guitar/0042-Trash Guitar 3
0031-Distortion Guitar	Guitar/0005-Dist Guitar 5
0032-Guitar harmonics	Laba170bank/0028-PianoBell
0033-Acoustic Bass	Will_Godfrey_Collection/0045-Steel Bass
0034-Electric Bass finger	Bass/0009-Electric bass 1
0035-Electric Bass pick	Bass/0041-Electric_Bass
0036-Fretless Bass	Bass/0050-Fretless Bass
0037-Slap Bass 1	Rhodes/0042-Hard Rhodes1
0038-Slap Bass 2	olivers-100/0018-Bass Guitar - Slap
0039-Synth Bass 1	Bass/0013-FM rubber bass
0040-Synth Bass 2	Bass/0024-Moog bass
0041-Violin	Strings/0051-Synth Violin 2 Fat
0042-Viola	Strings/0051-Synth Violin 2 Fat
0043-Cello	Strings/0051-Synth Violin 2 Fat
0044-Contrabass	Bass/0005-Bass 5
0045-Tremolo Strings	Strings/0001-Saw Strings 1
0046-Pizzicato Strings	Strings/0003-Saw Strings 3
0047-Orchestral Harp	Pads/0065-Soft Pad
0048-Timpani	Noises/0018-Gun
0049-String Ensemble 1	VDX/0065-Strings
0050-String Ensemble 2	folderol collection/0029-Full Strings
0051-Synth Strings 1	Strings/0010-Strings Pad1
0052-Synth Strings 2	Strings/0014-Strings Pad5
0053-Choir Aahs	Choir_and_Voice/0001-AHH Choir 1
0054-Voice Oohs	Choir_and_Voice/0004-Voice OOH
0055-Synth Voice	Choir_and_Voice/0005-Choir Pad1
0056-Orchestra Hit	Misc/0010-Industrial orchestra
0057-Trumpet	Leads/0027-Prophet horn 2
0058-Trombone	Brass/0033-Analog Brass 1
0059-Tuba	Brass/0001-FM Thrumpet
0060-Muted Trumpet	Synth/0001-Soft Synth 1
0061-French Horn	Brass/0034-Analog Brass 2
0062-Brass Section	Brass/0007-Synth Brass 5
0063-Synth Brass 1	Brass/0003-Synth Brazz 1
0064-Synth Brass 2	Brass/0004-Synth Brazz 2
0065-Soprano Sax	Reed_and_Wind/0066-Fat Reed2
0066-Alto Sax	Reed_and_Wind/0065-Fat Reed1
0067-Tenor Sax	Reed_and_Wind/0037-Reed 5
0068-Baritone Sax	Reed_and_Wind/0099-Sharp Reed

Continued next page

General MIDI Instrument	Yoshimi Instrument Used
0069-Oboe	Reed_and_Wind/0040-Reed 8
0070-English Horn	Brass/0034-Analog Brass 2
0071-Bassoon	Will_Godfrey_Collection/0102-Bassoon
0072-Clarinet	Reed_and_Wind/0006-Clarinet
0073-Piccolo	Will_Godfrey_Collection/0071-Ocarina
0074-Flute	Will_Godfrey_Collection/0057-Soft Flute
0075-Recorder	Will_Godfrey_Collection/0059-Ocarina
0076-Pan Flute	Will_Godfrey_Collection/0127-Pan Pipe
0077-Blown Bottle	Will_Godfrey_Collection/0125-Bottle
0078-Shakuhachi	Will_Godfrey_Collection/0125-Pan Pipe 32
0079-Whistle	Will_Godfrey_Collection/0027-Ghost Whistle
0080-Ocarina	Flute/0071-Ocarina
0081-Lead 1 square	Leads/0022-Square lead
0082-Lead 2 sawtooth	Louigi_Verona_Workshop/0008-saw-lead
0083-Lead 3 calliope	Leads/0018-Sine lead
0084-Lead 4 chiff	chip/0018-Chiffer_Chip
0085-Lead 5 charang	Louigi_Verona_Workshop/0001-progressive-lead-1
0086-Lead 6 voice	Choir_and_Voice/0067-Vocal Morph 3
0087-Lead 7 fifths	chip/0017-SuperSquare1
0088-Lead 8 bass lead	Strings/0157-Dual Strings Oct2
0089-Pad 1 new age	Pads/0028-Ethereal
0090-Pad 2 warm	Will_Godfrey_Companion/0019-Warm Square Swell
0091-Pad 3 polysynth	Dual/0065-Dream of the Saw
0092-Pad 4 choir	Alex_J/0100-Choir Pad
0093-Pad 5 bowed	The_Mysterious_Bank/0004-trance_strings_pad
0094-Pad 6 metallic	The_Mysterious_Bank/0011-dreaming_bells
0095-Pad 7 halo	Alex_J/RadioPulsePad
0096-Pad 8 sweep	Pads/0011-lightbeam
0097-FX 1 rain	The_Mysterious_Bank/0037-the_rain
0098-FX 2 soundtrack	The_Mysterious_Bank/0038-falling_stars
0099-FX 3 crystal	Will_Godfrey_Companion/0006-Tinkle Bell
0100-FX 4 atmosphere	The_Mysterious_Bank/0038-the_starting_machine
0101-FX 5 brightness	Noises/0014-droplets for chords
0102-FX 6 goblins	Noises/0002-Ioioioioioi
0103-FX 7 echoes	Noises/0072-Cave Gates
0104-FX 8 sci-fi	The_Mysterious_Bank/0031-etrange_sound
0105-Sitar	olivers-100/0019-FM Sitar
0106-Banjo	olivers-100/0016-Banjoey
0107-Shamisen	Plucked/0034-Plucked String2
0108-Koto	Plucked/0003-Plucked 3
0109-Kalimba	Plucked/0001-Plucked 1
0110-Bagpipe	Reed_and_Wind/0033-Reed 1
0111-Fiddle	Laba170bank/0055-DevilsFiddle2
0112-Shanai	Reed_and_Wind/0035-Reed 3
0113-Tinkle Bell	Bells/0011-Glass bells
0114-Agogo	The_Mysterious_Bank/0028-snare

Continued next page

General MIDI Instrument	Yoshimi Instrument Used
0115-Steel Drums	olivers-100/0029-Steel Drums
0116-Woodblock	The_Mysterious_Bank/0028-snare
0117-Taiko Drum	The_Mysterious_Bank/0028-snare
0118-Melodic Tom	demo/Melodic_Tom_from_Nat_drum_kit
0119-Synth Drum	The_Mysterious_Bank/0028-snare
0120-Reverse Cymbal	The_Mysterious_Bank/0028-snare
0121-Guitar Fret Noise	The_Mysterious_Bank/0028-snare
0122-Breath Noise	The_Mysterious_Bank/0028-snare
0123-Seashore	Noises/0008-Wind and Surf
0124-Bird Tweet	The_Mysterious_Bank/0028-snare
0125-Telephone Ring	The_Mysterious_Bank/0028-snare
0126-Helicopter	The_Mysterious_Bank/0028-snare
0127-Applause	The_Mysterious_Bank/0028-snare
0128-Gunshot	Noises/0018-Gun
0129-Drum Kit	C_Ahlstrom/Natural_Drum_Kit_Basic
End of table	

Presumably, this basic bank could be improved enough to be useful for most music. Alternative (and better) banks could be created, as well.

One thing to note about the instruments. When copied from the original directory to the `gm-basic` directory, each instrument copied retains its name, of course. To make it less confusing to use in *Yoshimi*, then, one should rename each instrument to its GM name. To do that, right click on the name, change it, and then save the instrument to the same file from which it was loaded. Be careful! It is easy to make a mistake! And we have not yet renamed any of them!

6.2 Root Paths

The first thing to do is to add the yoshimi-cookbook `banks` directory to your setup. Run *Yoshimi*, and navigate the following user-interface path: *Menu / Instrument / Show Root Paths ...* Then click **Add root directory...**. Navigate to where the yoshimi-cookbook project is stored and add the `yoshimi/banks` directory. The result should be something like the following:

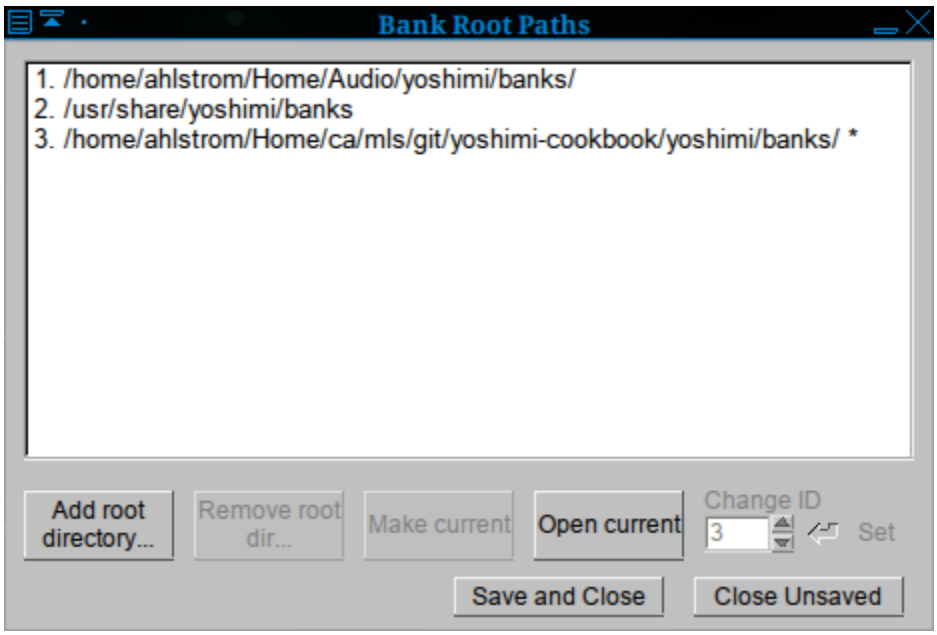

Figure 11: Bank Root Paths

Click on the new directory. It has ID = 3 in that diagram. We will refer to this value as the "banks path". Now click the **Make current** button. Verify that it now has the asterisk. Click the **Save and Close** button.

Now let's open the "gm-basic" bank. Run *Yoshimi*, and navigate the following user-interface path: *Menu / Instrument / Show Banks ...*

In the matrix of banks, you should see "gm-basic" somewhere. (We also have a "demo" bank in place.)

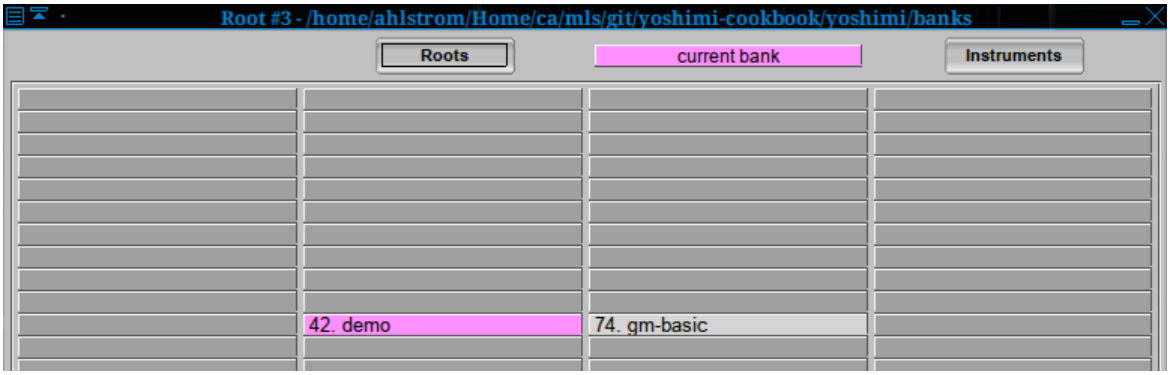

Figure 12: Two Banks, Demo and Basic GM

Click on the "gm-basic" bank. The larger dialog below will be shown. Note that this action also makes "gm-basic" the current bank. This setting will be preserved across a restart of *Yoshimi*.

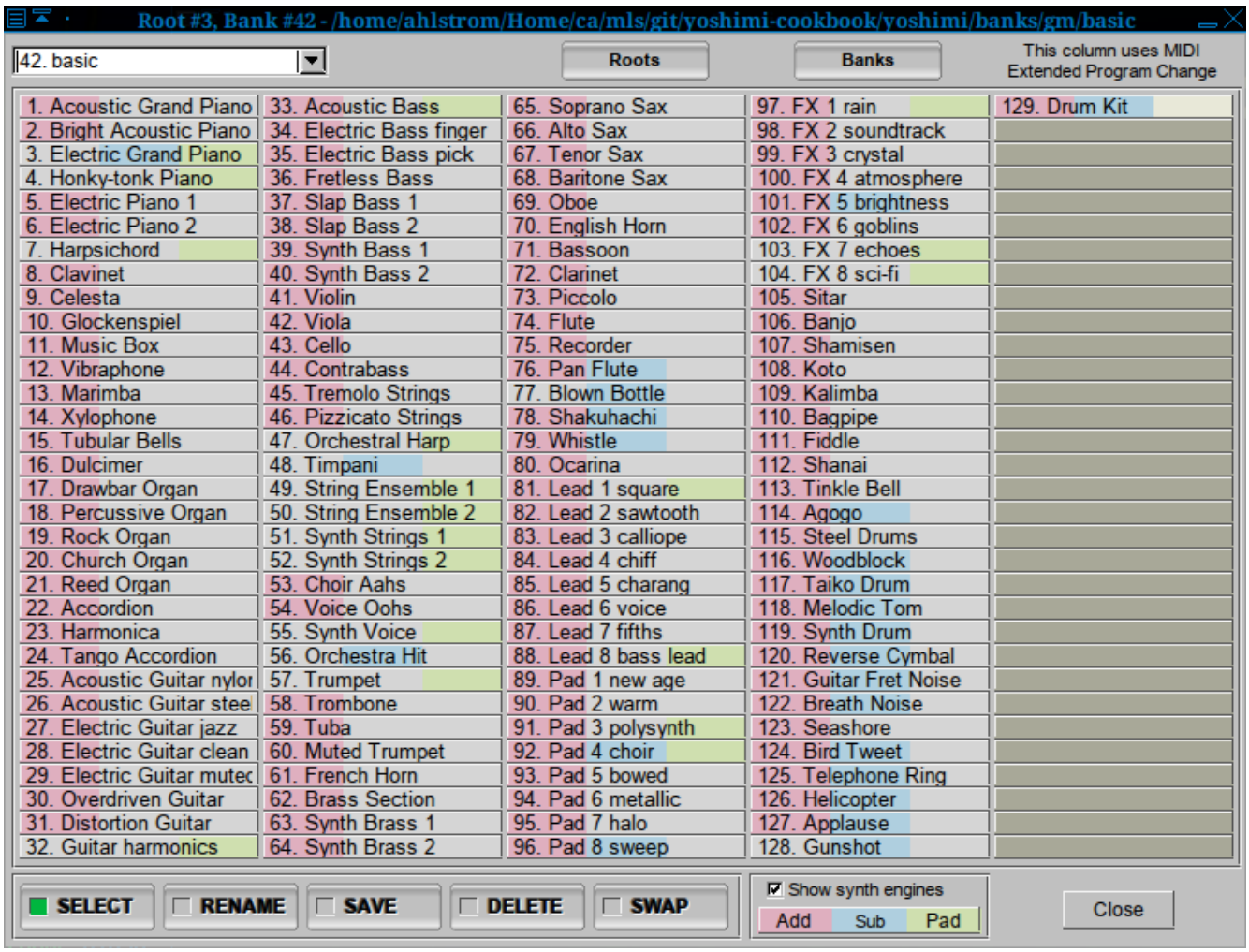

Figure 13: A General MIDI Basic Bank

Remember that these banks have GM names; the original files used to create each one are listed in the spreadsheet mentioned in the previous section.

Also note the drum kit, with an ID of 129. Normally, drum kits might be stored in a bank of drum kits, but here we have a GM-compliant drum kit, compliant in the sense that most of the keys are mapped correctly, and all of the keys will play *something*.

6.3 Using a Basic GM Bank

Once the current root path has been set (e.g. to `.../yoshimi-cookbook/yoshimi/banks`) and the current bank has been set (e.g. to `gm-basic`), then a MIDI file can select instruments 1 to 128 from the current bank using a Program Change event followed by the desired instrument's number re 0.

The bank and root paths can be saved in the *Yoshimi* state. But the MIDI file can also start with bank-selection events, and change the current bank using Bank Select LSB (CC32) and Bank Select MSB (CC0) events.

7 Converting Tunes for Yoshimi Playback

This section explains various methods for converting tunes to play in *Yoshimi*, plus some concrete methods for configuration *Yoshimi* and taking advantage of its newer capabilities.

We will subdivide this section by tunes converted.

7.1 Tune / "Before You Accuse Me"

This file describes the importing an old (circa 1996) Cakewalk transcription, a rendering of Eric Clapton and Robert Cray from a transcription that someone gave to me to try out. It was originally transcribed to fit the Yahama PSS-790 [5] consumer-level keyboard, as we recall (it was years and years ago.)

First, let us run the `b4uacuse.mid` MIDI file as is through the TiMidity++ [12] software synthesizer:

```
$ timidity b4uacuse.mid
Format: 1  Tracks: 14  Divisions: 120
Track name: PSS-790 Patchin
Track name: Guitar 1 (E.C.)
Track name: Guitar 2 (R.C.)
Track name: Vocal
Track name: Rhythm (Chords)
Track name: Bass Guitar
Track name: Drums
Track name: Chris Ahlstrom
Track name: (GEnie: KICKAHA
Track name: "Before You
Track name: Accuse Me" by
Track name: Eric Clapton &
Track name: Robert Cray
No instrument mapped to tone bank 0, program 12 -
    this instrument will not be heard
No instrument mapped to tone bank 0, program 87 -
    this instrument will not be heard
```

Our `/etc/timidity/timidity.cfg` file specifies the `/etc/timidity/freepats.cfg` file. In this file, we see that bank 0 does not define the three patch numbers that are noted as missing above and below: 12, 62, and 87. Patch 12 on the PSS-790 is *Marimba*, but is mapped to GM *Jazz Guitar* (*Elec. Guitar jazz*). In *TiMidity++* freepats, this is patch 26. Patch 0 is mapped to patch 62, which is *Synth Brass 1* in GM; the closest in *TiMidity++* freepats is patch 61 *Brass Section*. Patch 87 on the PSS-790 is *Lead 8 (bass+lead)* which gets mapped to *Flugelhorn (*English Horn). The tune sounds reasonable, but obviously is not scored correctly. Let us use *midicvtpp* [4] to convert it to good (we hope) GM format:

```
$ midicvtpp --m2m GM_PSS-790_Multi.ini -i b4uacuse.mid -o b4uacuse-gm-1.mid
```

or, for more output information:

```
$ midicvtpp --m2m GM_PSS-790_Multi.ini \
      --summarize b4uacuse.mid b4uacuse-gm-1.mid 2> summary.txt
```

The command creates a file that has only one "error" message and plays pretty well in Timidity:

```
No instrument mapped to tone bank 0, program 62 -
    this instrument will not be heard
```

Again, note that patch 62 is a *Synth Brass 1* patch.

We cannot tell what the target voices of the PSS-790 were, and what they were converted to. That sounds like a feature to add to the *midicvtpp* program! Done, in version 0.4.0! We can go forward with the GM MIDI bank we have created for *Yoshimi*. Take a look at the file `sequencer64/b4uacuse/summary.txt` in this project. It is used in the patch descriptions here.

The next step is to see how these patches map to our preliminary *Yoshimi* GM bank. We have three ways to go for patching:

1. Keep the program-change track(s) and use them to select instruments.
2. Move each program-change to the appropriate track.
3. Remove all program-change events and rely on a Yoshimi state file to set up.

The disadvantage of keeping the program changes:

1. They are currently spread out in three different tracks, as can be seen by opening the b4uacuse-gm-1.mid file in Sequencer64 and viewing the "PSS-790 Patch", "Chris Ahlstro", and "(GEnie: KICKA" tracks using the Event Editor.
2. They make it difficult to remix the song for playback on various synthesizers other than Yoshimi.
3. At least in *Yoshimi*, it becomes more difficult to change the instruments used.
4. They take away a chance to learn how *Yoshimi* state works.

On the other hand, it will be interesting to add program changes to the tune and see how *Yoshimi* handles it these days. Later.

So let us take `b4uacuse-gm-1.mid`, trim out the patch tracks and empty tracks, recreate the layout of the song editor, and save it as `b4uacuse-gm-patchless.midi` (a *Sequencer64* format). Here are the tracks, and their putative GM instruments as numbered in our GM bank:

```
Track 0: Guitar 1 (E.C.)        #26 Acou. Guitar (steel)
Track 1: Guitar 2 (R.C.)        #27 Elec. Guitar (jazz)
Track 2: Bass Guitar            #34 Elec. Bass (finger)
Track 3: Drums                  Drums
Track 4: Vocal                  #70 English Horn
Track 5: Rhythm (Chords)        #4 Honky-tonk Piano
```

See `summary.txt` to see what conversions were made by *midicvtpp*. We go to **Yoshimi / Menu / Patch Sets / Show Patch Banks...** and look at the banks. We select *116. gm-basic*. This bank has passable (in most cases) equivalents to GM instruments, plus a drum kit at program number 129 (an extended program number).

We will set the channels to "track + 1" (channels are counted from 1), except for drums, which go to channel 10. We enable each *Yoshimi* Part. We note the name (in quotes) of the corresponding Yoshimi patch.

Yoshimi parts are taken from the *gm-basi* c bank ("116. gm-basic"):

```
Track 0, Ch.  1: Guitar 1 (E.C.) #26 Acoustic Guitar (steel) ("Clean Guitar 1")
Track 1, Ch.  2: Guitar 2 (R.C.) #27 Electric Guitar (jazz) ("Electric Guitar")
Track 2, Ch.  3: Bass Guitar     #34 Electric Bass (finger) ("Electric bass 1")
Track 3, Ch. 10: Drums           #129 Drum Kit ("Natural Drum Kit from DS 2")
Track 4, Ch.  5: Vocal           #70 English Horn ("Analog Brass 2")
Track 5, Ch.  6: Rhythm (Chords) #4 Honky-tonk Piano ("Synth Piano 3 fat")
```

Note that The bank must be installed in `~/.config/yoshimi-cookbook/banks` or in a root path setup as in section 6.2 ("Root Paths") on page 26.

After setting up each Track/Part/Channel, we played the tune, using ALSA, and adjusted the volume of some of the parts while playing it through buss 5 (where *Yoshimi* sets up on our system) via the command

```
$ sequencer64 --bus 5 b4uacuse-gm-patchless.midi
```

We then saved the entire state of *Yoshimi* to `/.config/yoshimi/yoshimi-b4uacuse-gm.state` via **Menu / State / Save ...**. We reload this file via

```
$ yoshimi -a -A --state=yoshimi-b4uacuse-gm.state
```

to verify that it still works. We ended up switching to JACK playback for recording the audio:

```
$ sequencer64 -J b4uacuse-gm-patchless.midi
$ yoshimi -j -J --state=yoshimi-b4uacuse-gm.state
```

and *avconv* (*ffmpeg*) for converting it to an MP3 file that one can listen to without loading up all this software.

8 Usage Notes

This section contains some notes about usage and settings we've collected that don't fit anywhere else. These notes come from http://www.freelists.org/list/yoshimi. To post to the list, email to: yoshimi@freelists.org The archive of the old SourceForge mailing list is found at: https://www.freelists.org/archive/yoshimi.

8.1 Notes / Instrument Design

From user "Ichthyostega":

> I find it difficult to balance or voice an instrument design such that it "runs well" when used in composition over various octaves. With some designs, the descant dominates, while in other cases the bass dominates. When I play a chord over several octaves, or have several lines at the same time, they show a different weight and are not well balanced out. Of course, I can somewhat fix that by giving the individual notes different velocities or use several MIDI channels with different volume settings. But I rather consider this the job of instrument design: it should come out well balanced out of the box.

Will answers:

> First off, the way the sound engines work involves a degree of modelling whereby sound energy tends to decrease with increasing frequency. This was a deliberate choice made by Paul when he designed them, intending to more-or-less emulate real instruments. It's been a minor nuisance to me occasionally, but to a degree you can get round it by fiddling the **Resonance** graph (not SubSynth). You can also set filters to **LPF1** with **Q** to zero and **C.freq** to max.
>
> Any waveform with little harmonic content will tend to sound quieter at lower frequencies, while the amplitude is actually pretty constant. *Simple Sound* is a classic example of that.
>
> Fades across a instruments in a kit has been mentioned before. I had a look at it some time ago, and it is actually an *extremely* difficult problem. If there were only two possible kit items it would 'just' be hard – across potentially 16 is in the nightmare realm!

9 Sequencers

This section discusses the basic usage of some sequencers with *Yoshimi*. Some sequencers support ALSA MIDI, some support JACK MIDI, and some support both. This section provides some use cases and pointers that cover a range of sequencers. Note that this section depends on having one's Banks/Roots directories set up properly. It is important to make sure that there is a default "presets" directory present.

So far, we demonstrate only *Sequencer64*. Others to try in the near future: *Rosegarden*, *Qtractor*, *Ardour*, *MuSE*, *LMMS*.

9.1 Sequencers / Sequencer64

Sequencer64 is our fork/reboot of the *Seq24* project [7]. It currently fixes some bugs in *Seq24*, adds a few useful features, a little more color, and refactors the code. Much of the advice in this section will apply to *Seq24*, with some minor differences, such as the locations and names of the configuration files. However, *Seq24* doesn't have a buss-override option.

Note that the two *Sequencer64* configuration files are:

```
~/.config/sequencer64/sequencer64.rc
~/.config/sequencer64/sequencer64.usr
```

The simplest thing to do for this cookbook is

- Make sure these files do not exist; save the old versions somewhere and then delete the two files.
- Then start *Sequencer64*.

- Then immediately exit it.

New versions of those files will appear, and we can start with a clean slate. We will edit those files, as that is a bit more foolproof than using command-line options. The "rc" is the most important file, but there are configuration items in the "usr" ("user") file that can make *Sequencer* prettier, and a couple options that can affect playback. We'll point out the necessary values when needed.

9.1.1 Sequencers / Sequencer64 / ALSA

Skip this section if all that matters for one's setup is JACK support. Otherwise, first make sure that JACK is not running. Next, remove the "rc" and "usr" files from the configuration directory, as noted in the previous section. Then open up a console window (terminal) and run *Yoshimi* in ALSA audio and ALSA MIDI modes:

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

Now, run *Sequencer64*, and check the new versions of those files, as noted in the previous section. On our computer, with Timidity installed, we see the following ALSA entries in the `sequencer64.rc` file:

```
# Output buss name: [0] 14:0 Midi Through Port-0
0 0  # buss number, clock status
# Output buss name: [1] 128:0 TiMidity port 0
1 0  # buss number, clock status
# Output buss name: [2] 128:1 TiMidity port 1
2 0  # buss number, clock status
# Output buss name: [3] 128:2 TiMidity port 2
3 0  # buss number, clock status
# Output buss name: [4] 128:3 TiMidity port 3
4 0  # buss number, clock status
# Output buss name: [5] 129:0 input
5 0  # buss number, clock status
```

The last entry in that list is "129:0 input", which corresponds to *Yoshimi*. Note that the *Sequencer64* application will find the actual name of the device (e.g. "129:0 Yoshimi Port 0").

While we're in the "rc" file, let's make sure of the following settings:

```
1     # show sequence numbers (1 = true / 0 = false)

[manual-alsa-ports]
0     # flag for manual ALSA ports
```

The first setting enables showing the sequence number in empty slots in the main window. (This setting really belongs in the "user" file!) The second setting disables manual setup of the ALSA MIDI ports. *Sequencer64* will determine the ports that exist on the system.

In the "user" file, we want to set the buss number to 5, to match our current ALSA setup. It may differ on your system. If it is 0, you don't have to do the next step. Nor is it needed for running with JACK (discussed later).

Open the "user" file and make sure the following setting matches the location of *Yoshimi*'s MIDI input port on your system (it is 5 on ours):

```
5       # midi_buss_override
```

Now open a MIDI file, verify that the correct buss value is shown, and that the file plays and is audible. If this is the case, one is done. If not, please read the *Sequencer64* user's manual ([8]).

Note that, generally, the "midi_buss_override" value should be set to "-1". We use it here as a convenience for our demonstration. Also, one can load a preset/patch-set file from the command line, as shown here:

```
$ yoshimi -a -A --load=/home/user/.../yoshimi-cookbook/yoshimi/examples/Out_There.xmz
Yoshimi is starting
ConfigFile /home/user/.config/yoshimi/yoshimi.config not found,
      will use default settings
Missing bank file
Scanning for banks
Missing history file
March little endian = 1
Format = Signed Little Endian 32 Bit 2 Channel
Using alsa_audio for audio and alsa_midi for midi
pLoaded Out_There parameters
Yoshimi 1.3.9 rc3
Clientname: yoshimi
Config: Audio: alsa -> 'default'
Midi: alsa -> 'default'
Oscilsize: 512
Samplerate: 48000
Period size: 256

Yay! We're up and running :-)
```

Now go to section 9.1.3 ("Sequencers / Sequencer64 / "Out There"") on page 36. Load up a sample song and patch-set, and play it.

9.1.2 Sequencers / Sequencer64 / JACK

Now let's repeat our setup using JACK. However, note that, while *Sequencer64* supports JACK synchronization and can serve as a JACK Master, it's MIDI support is purely ALSA-based. Therefore, an ALSA-to-JACK bridge program must be run to expose the MIDI ports to JACK.

Before continuing, make sure the *Sequencer64* "rc" and "usr" files reflect the following settings, which are different than those described in the ALSA section above. First, the "rc" file:

```
      1     # show sequence numbers (1 = true / 0 = false); ignored in legacy
[manual-alsa-ports]
0     # flag for manual ALSA ports (--manual-alsa-ports)
```

Manual ALSA ports can be turned on, but that results in 16 additional *Sequencer64* MIDI ports being created; the MIDI Through port will be enough for now, since we're using only one synthesizer, *Yoshimi*, for sound generation. Next, the "usr" file:

```
-1      # midi_buss_override (--bus n)
```

(If the buss is set properly, as above, in the `~/.config/sequencer64/sequencer64.usr` file, there is no need for the `--bus` option on the command line.)

Now run the following series of commands, from a console window:

```
$ qjackctl &
$ a2jmidid --export-hw &
$ sequencer64 &
$ yoshimi -j -J
```

In *qjacktrl*, in the **Audio** tab, connect "yoshimi" on the left with "system" on the right.

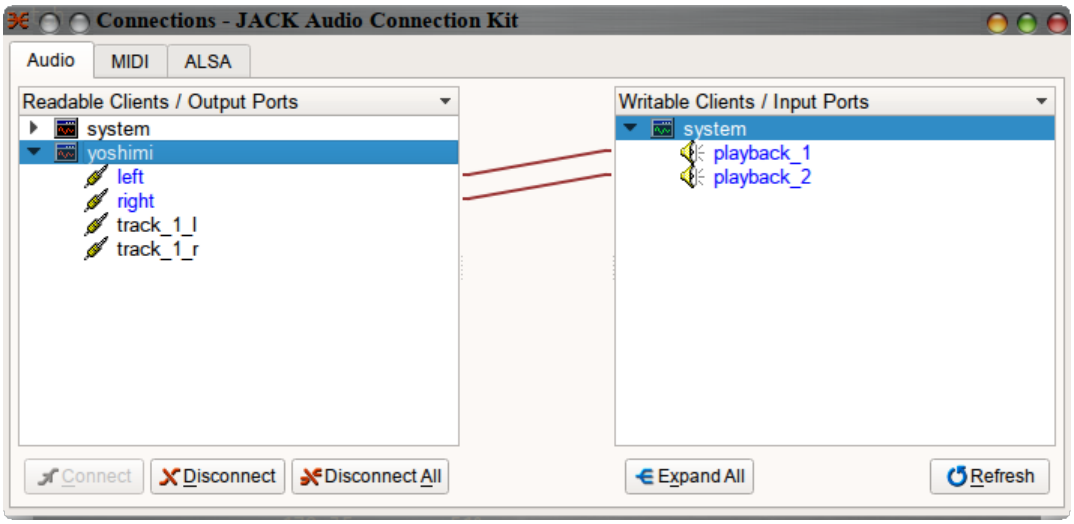

Figure 14: Yoshimi Audio Connection, JACK

In the **MIDI** tab, connect a2j's "Midi Through [14] (capture)..." port on the left to yoshimi-01's "midi in" port on the right.

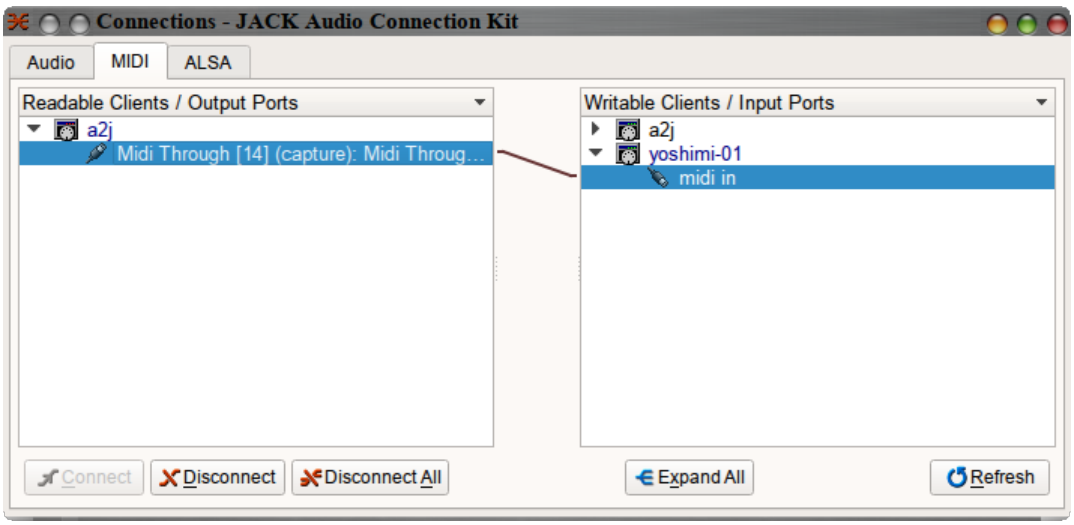

Figure 15: Yoshimi MIDI Connection, JACK

Now go to section 9.1.3 ("Sequencers / Sequencer64 / "Out There"") on page 36. Load up a sample song and patch-set, and play it.

9.1.3 Sequencers / Sequencer64 / "Out There"

In this section, we set up and run a demo file and patch-set file provided with *Yoshimi*. Let's start from scratch.

9.1.3.1 Initial Installation

1. Install *Yoshimi* from a package manager or from source code.
2. Install *ZynAddSubFx* from a package manager or from source code. This action gets us more instrument banks.
3. Clone the yoshimi-cookbook [16] project from GitHub.
4. Clone the sequencer64 [7] project from GitHub.

9.1.3.2 Start from Scratch

Remove any existing *Yoshimi* configuration.

```
$ cd ~/.config
$ mv yoshimi/ yoshimi-orig
```

Start Yoshimi:

```
$ yoshimi --version
Yoshimi is starting
ConfigFile /home/user/.config/yoshimi/yoshimi.config not found,
      will use default settings
```

```
Missing bank file
Scanning for banks
Missing history file
Yoshimi 1.3.9
```

This "first start" results in the creation of `~/.config/yoshimi` and an empty `~/.config/yoshimi/presets` directory.

9.1.3.3 Load Example XMZ File

The next step in this demonstration is to load a parameters (patch-set) file. We pick the parameters file provided by the *Yoshimi* source-code package for the "Out There" project. We show the command-line *Yoshimi* in ALSA mode for simplicity:

```
$ yoshimi -a -A --load=/home/user/.../yoshimi-cookbook/yoshimi/examples/Out_There.xmz
```

Remember that *Yoshimi* cannot find the parameters file if `/home/user` part is replaced with `~`. This action results in additional files created in `~/.config/yoshimi`: `yoshimi.history`, `yoshimi.banks`, `yoshimi.windows`.

The history file contains a reference to the full path of the `Out_There.xmz` file.

The `yoshimi.banks` file is a gzip-compressed XML file that references the default bank-root (bank-root 0): `/usr/share/yoshimi/banks` or `/usr/local/share/yoshimi/banks`. It also defines prefix number for all of the banks (bank directories) that ship with *Yoshimi*. It also references the *ZynAddSubFX* bank location (if installed): `/usr/share/zynaddsubfx/bank` (bank-root 1). It defines bank numbers for all of the banks that ship with *ZynAddSubFX*.

The `yoshimi.windows` file contains the locations of about 20 windows, such as "master", "panel", "instruments", "SUBnote", "ADDnote", and many more.

The patch-set can also be loaded through the *Yoshimi* user-interface. In the **Yoshimi / Patch Sets / Load External...** menu, load the example `Out_There.xmz` patch-set file file, and observe that the "Angel Piano" instrument appears as **Part 1**.

9.1.3.4 Playing Out_There

Run *Yoshimi* as above for either ALSA or JACK. Then run *Sequencer64* as above for either ALSA or JACK.

Sequencer64 converts the SFM 0 file, `Out_There.mid`, to SMF 1 as it reads in the file. It also deposits the original SMF 0 track into the 16th slot, highlighted in a dark-cyan color. Delete this SMF 0 track by right-clicking on it and selecting the **Cut** menu entry.

Arm (unmute) all of the tracks in the main window view of Sequencer64. This is done by simply left-clicking once on each slot. They are then shown with a black background.

Play! Either click the play button or hit the spacebar. This setup can loop endlessly, and the differing lengths of the parts make it vary as it plays, forever. (*Sequencer64* is a live-looping sequencer.)

When done, don't save either the *Yoshimi* parameters or the converted tune.

```
yoshimi> exit
System config has been changed.  Still exit N/y? y
```

9.1.4 Sequencers / Sequencer64 / Adapting a Tune

We discuss the "WPB" song here. This song had a patch-set that was no longer valid.

Add /.config/yoshimi-cookbook/banks and make it the current bank.

9.1.5 Sequencers / Sequencer64 / Basic Composing

Finally, we are ready to add our own music, starting with a loop. In *Sequencer64*, right click on a pattern slot and select **New**. On the bottom of the new pattern window, select the left-most icon button, which shows a blue box pointing to a MIDI port (tooltip: "Sequence dumps data to MIDI bus."). Press the Space bar while in the main window or the pattern window to start the loop, and add notes.

> **Peculiarities of Sequencer64**. First, if the pattern is empty the progress bar will not move. Second, to add a note, one must press the right button (the pointer changes to a pencil) and, while holding it, press the left mouse button. Or click in the pattern editor, press "p" to select the pencil mode, then click/drag to add notes. Press "x" to "eXit" from that mode. Also remember that notes are drawn only with the length selected by the "notes" button near the top of the pattern window. Again, see the Sequencer64 user manual for the gory details.

MORE TO COME!

10 Summary

In summary, we can say that you will absolutely love cooking with *Yoshimi*.

11 References

The *Yoshimi* cookbook reference list.

References

[1] Will J. Godfrey *A discussion of making Bank/Root specifications more regular.* http://sourceforge.net/p/yoshimi/mailman/message/33200765/ 2014.

[2] Blair School of Music, Vanderbilt University. *Creating a Simple Bell* http://computermusicresource.com/Simple.bell.tutorial.html 2012?

[3] FM 8 Tutorials *Spectrum of a Simple Bell* http://www.fm8tutorials.com/wp-content/uploads/2012/03/Bell-spectrum.png 2012?

[4] Chris Ahlstrom *midicvt/midicvtpp: MIDI Conversion Commands* https://github.com/ahlstromcj/midicvt/ A consolidation of a number of MIDI-to-ASCII conversion programs, plus a new MIDI-to-MIDI conversion mode useful for converting old device-specific MIDI files to General MIDI format. 2016.

[5] Ashley Pomeroy *PSS-790 At a Glance* http://www.sonicstate.com/synth/yamaha_pss-790/ A brief note about this consumer-level synthesizer. Undated.

[6] LinuxMusicians newsgroup *Ring Modulation in ZynAddSubFX* http://linuxmusicians.com/viewtopic.php?f=1&t=8178 2012.

[7] Chris Ahlstrom. *Sequencer64* https://github.com/ahlstromcj/sequencer64/ Sequencer64 is a reboot and extension of the Seq24 live sequencer. It has been significantly refactored and upgraded, with many new features, bug fixes, and even better JACK support. 2016.

[8] Chris Ahlstrom. *The Sequencer64 User Manual.* https://github.com/ahlstromcj/sequencer64-doc/ Comprehensive documentation for Sequencer64. 2016.

[9] 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.

[10] 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.

[11] Gordon Reid *Synth Secrets: Creative Synthesis* http://www.soundonsound.com/sos/allsynthsecrets.htm 1999-2004.

[12] Masanao Izumo et al. *TiMidity++ Software Synthesizer* http://timidity.sourceforge.net/ 2004.

[13] Yoshimi team abrolag@users.sourceforge.net *The download site for the Yoshimi software synthesizer.* http://yoshimi.sourceforge.net/ 2016.

[14] Yoshimi team *The alternate location for the Yoshimi source-code.* https://github.com/abrolag/yoshimi/ 2016.

[15] Chris Ahlstrom *A Yoshimi User Manual.* https://github.com/ahlstromcj/yoshimi-doc/ 2016.

[16] Chris Ahlstrom *A Yoshimi Cookbook.* https://github.com/ahlstromcj/yoshimi-cookbook/ 2016.

[17] Barney Holmes (djbarney) *Generating synthesised drums and percussion in Linux using Yoshimi or ZynAddSubFX.* https://djbarney.wordpress.com/2013/10/27/generating-synthesised-drums-and-percussion-in-linux-using-yoshimi/ 2013.

Index

ALSA, 33
Audacity, 21

bank
    GM, 27
banks, 22
    demo, 27
    gm-basic, 27
bells, 9, 11, 12
    spectrum, 9

C note, 11
cent, 4
comma key, 11
current bank, 27, 28
current root, 28

directories
    demo bank, 9, 22
    demo presets, 22
    GM basic bank, 22, 27
drum kit, 14, 28

effects
    dial tone, 19

files
    12-string, 21, 22
    bells 440, 11
    bells addsynth, 10, 12
    bells ring mod, 12
    demo.ADnoteParameters.xpz, 7, 18, 19
    DTMF kit, 20
    natural drums, 14
    steel drums, 13, 14

General MIDI, 23
GM, 23
    bank, 27
    drum kit, 14, 28
    spreadsheet, 23
    table, 23
guitar
    12-string, 21

JACK, 34

modulation
    ring, 5

natural drum
    bass drums, 16
    cymbal, 16
    hihats, 15
    pitched toms, 16
    side stick, 16
    snare, 15
    toms, 16

power function, 15, 16
pulse, 15, 16

ring modulation, 5, 9
root, 26
root path, 26

Seq24, 32
Sequencer64, 32
sequencer64.rc, 32
sequencer64.usr, 32
steel drums, 13

XML compression, 3