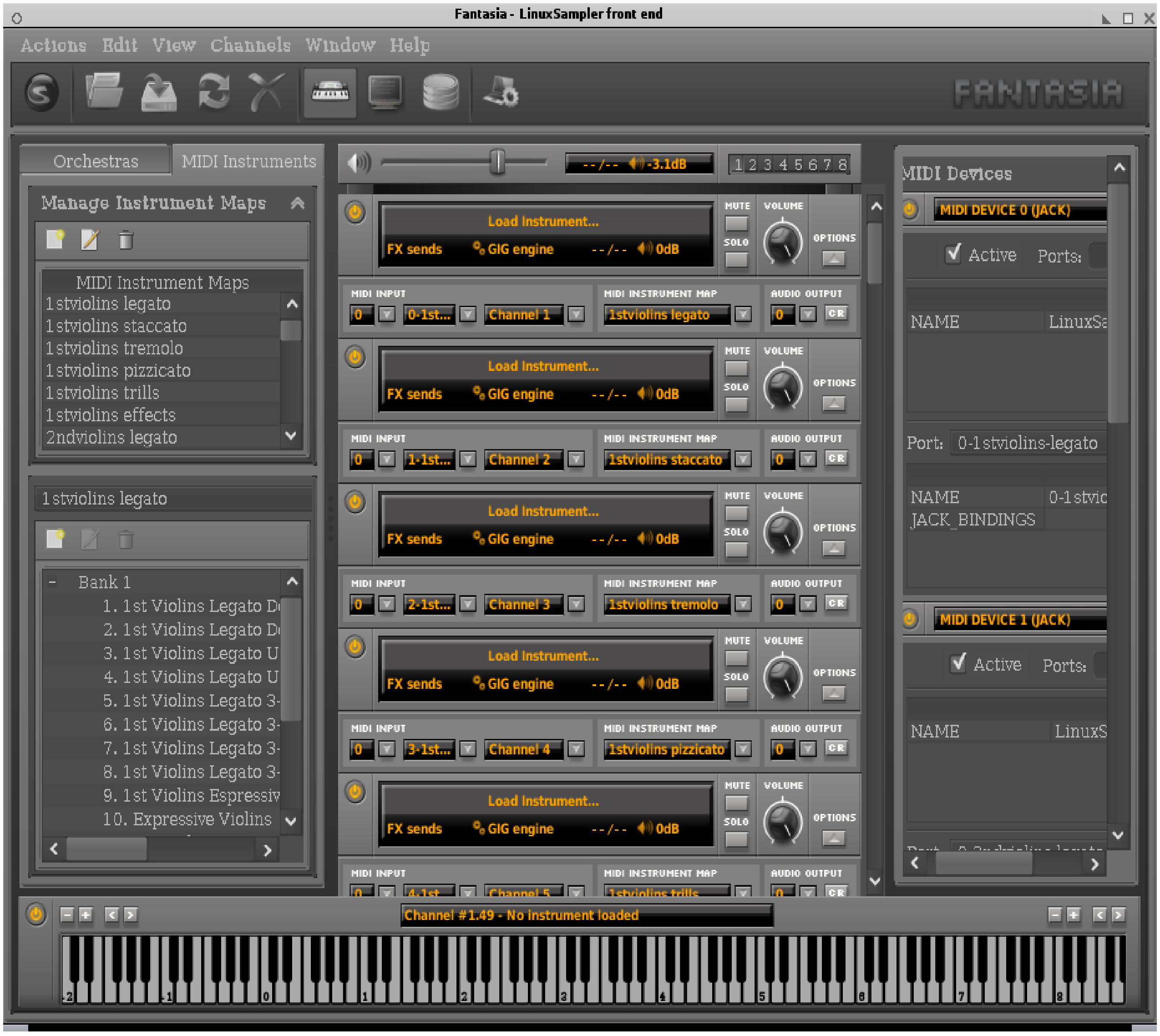
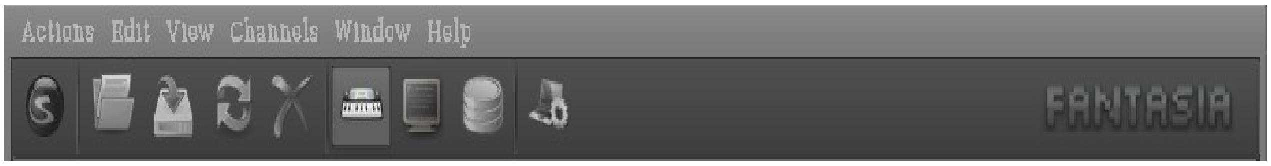
Linuxsampler

Here at the OpenOctaveProject, an important part of of our professional workflow pipeline is Linuxsampler (LS), an opensource engine for Gig, SFZ, SF2 format files. Because we value LS so highly, and it's importance, we've created this quick guide to using LS. It's by no means complete or comprehensive, but may help new users to get on their way, using this excellent software.
For a more detailed guide go to the LS website at:
http://www.linuxsampler.org/
and browse to documentation.

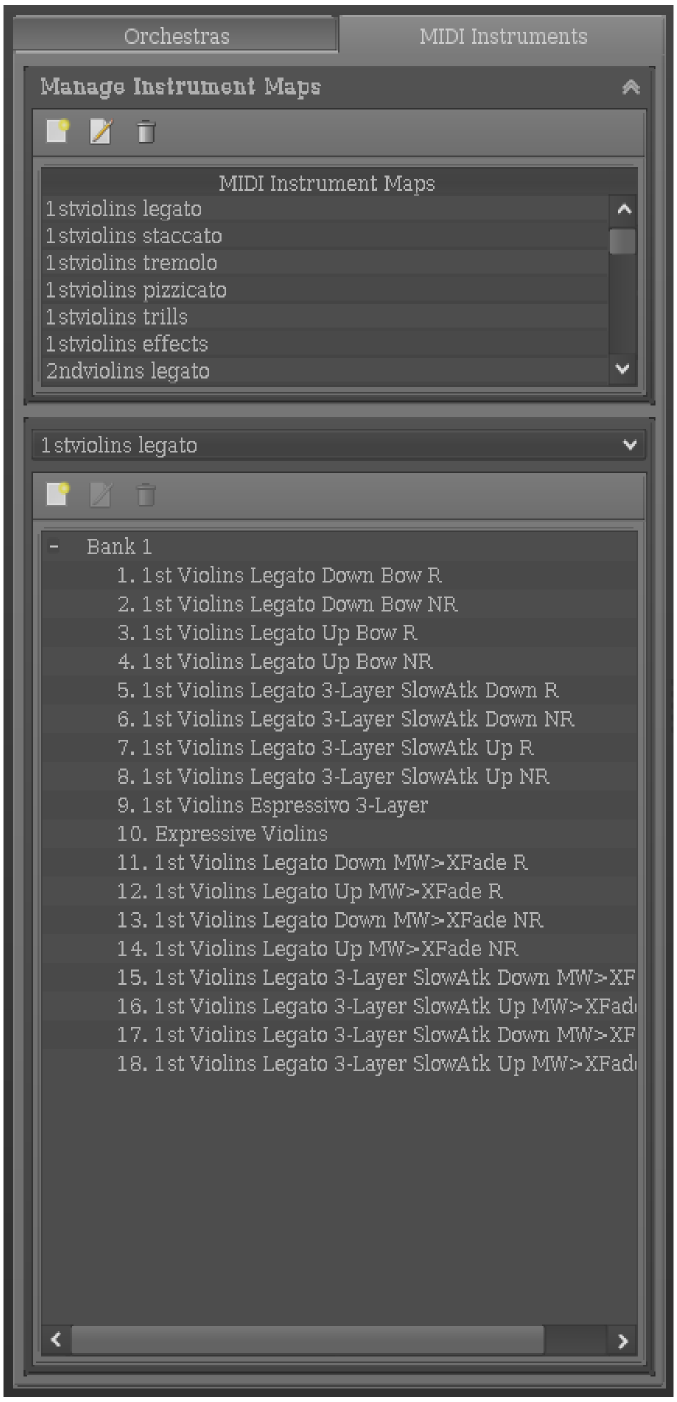


If OOM2 is the musician's pallette, on which he paints his musical ideas, and creates his masterpieces, then Linuxsampler is the powerful engine, driving and managing large sample libs, or small ensembles equally well. There are a number of use cases and workflow preferences we could describe here, so we'll give a couple that may present the intent of a wide range of users. LS comes in 2 parts, the backend, which runs in the background on your computer, and the frontend, or user interface, that gives the user access to the backend, and enables him or her to edit LS parameters. For an example of a user interface, and there are more than 1, we're using Fantasia, an excellent java interface for LS, written by Grigor Iliev.

The base UI components of Fantasia are:



This is the LS toolbar, containing menus and icons with which you can manage your LS configuration, load .lscp files, save midi-instrument maps, and many other functions.

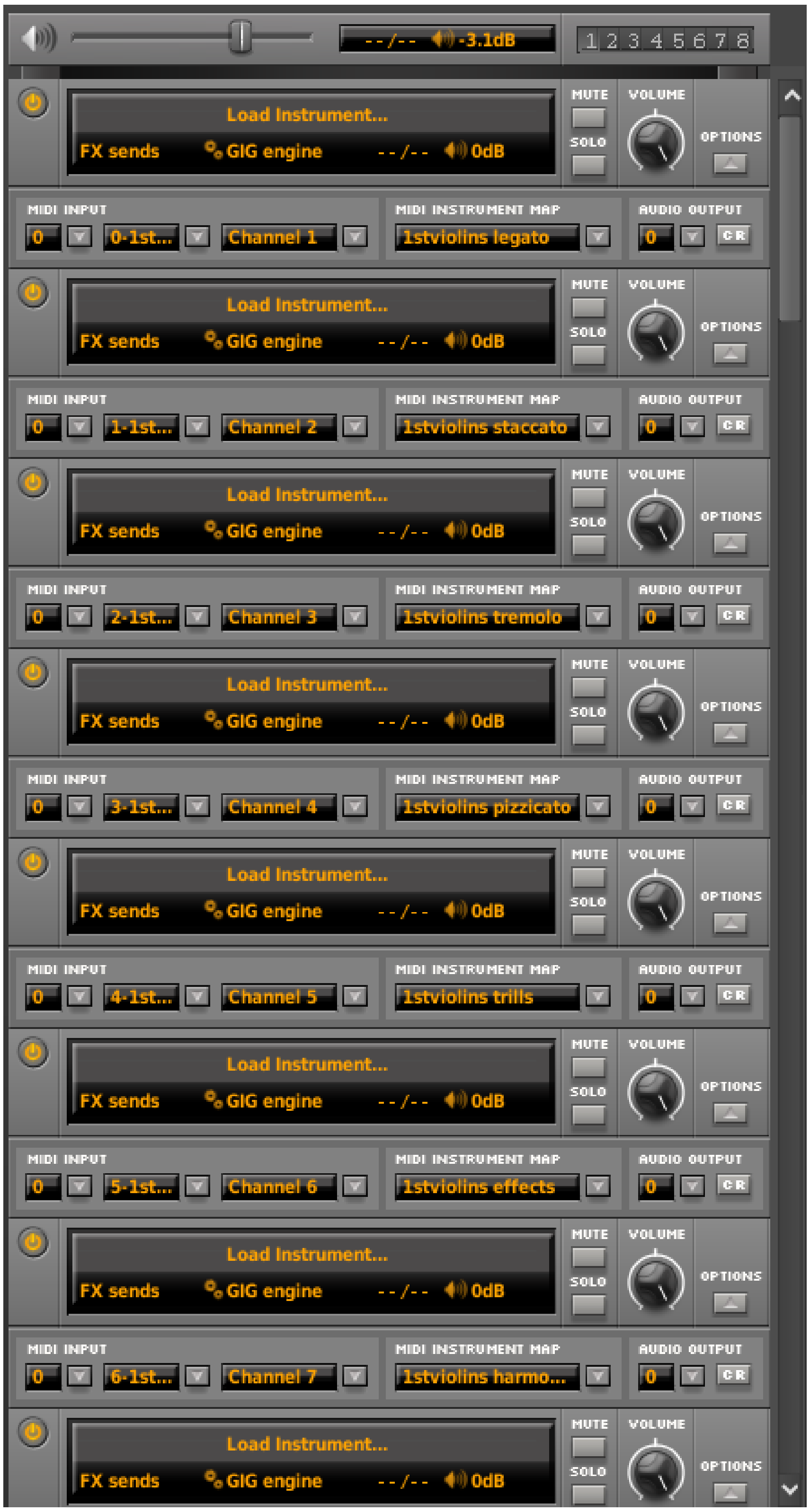


MIDI Instrument Maps
1stviolins legato
1stviolins staccato
1stviolins tremolo
1stviolins pizzicato
1stviolins trills
1stviolins effects
2ndviolins legato

1stviolins legato

- Bank 1
  1. 1st Violins Legato Down Bow R
  2. 1st Violins Legato Down Bow NR
  3. 1st Violins Legato Up Bow R
  4. 1st Violins Legato Up Bow NR
  5. 1st Violins Legato 3-Layer SlowAtk Down R
  6. 1st Violins Legato 3-Layer SlowAtk Down NR
  7. 1st Violins Legato 3-Layer SlowAtk Up R
  8. 1st Violins Legato 3-Layer SlowAtk Up NR
  9. 1st Violins Espressivo 3-Layer
  10. Expressive Violins
  11. 1st Violins Legato Down MW>XFade R
  12. 1st Violins Legato Up MW>XFade R
  13. 1st Violins Legato Down MW>XFade NR
  14. 1st Violins Legato Up MW>XFade NR
  15. 1st Violins Legato 3-Layer SlowAtk Down MW>XF
  16. 1st Violins Legato 3-Layer SlowAtk Up MW>XFad
  17. 1st Violins Legato 3-Layer SlowAtk Down MW>XF
  18. 1st Violins Legato 3-Layer SlowAtk Up MW>XFad

This section is where you build your midi maps, or create an orchestra of your required gig files,or both.



--/-- -3.1dB

1 2 3 4 5 6 7 8

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 0-1st... Channel 1
MIDI INSTRUMENT MAP 1stviolins legato
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 1-1st... Channel 2
MIDI INSTRUMENT MAP 1stviolins staccato
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 2-1st... Channel 3
MIDI INSTRUMENT MAP 1stviolins tremolo
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 3-1st... Channel 4
MIDI INSTRUMENT MAP 1stviolins pizzicato
AUDIO OUTPUT 0 CR

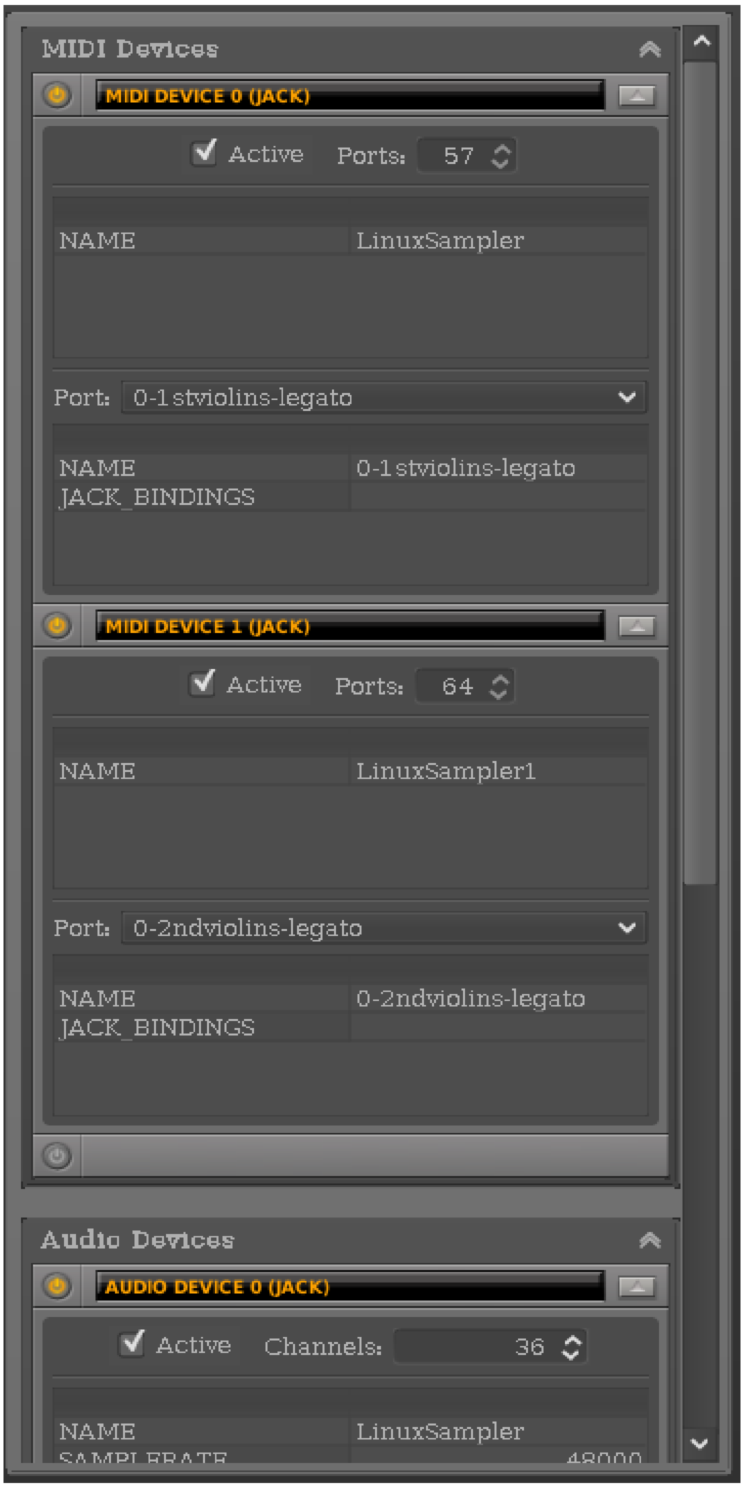
Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 4-1st... Channel 5
MIDI INSTRUMENT MAP 1stviolins trills
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 5-1st... Channel 6
MIDI INSTRUMENT MAP 1stviolins effects
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS
MIDI INPUT 0 6-1st... Channel 7
MIDI INSTRUMENT MAP 1stviolins harmo...
AUDIO OUTPUT 0 CR

Load Instrument...
FX sends GIG engine --/-- 0dB
MUTE SOLO VOLUME OPTIONS

Here in the centre of LS are your channel strip lanes. You have 8 to choose from, and can organise your created channels as you require.



MIDI Devices

MIDI DEVICE 0 (JACK)

✓ Active Ports: 57

NAME	LinuxSampler

Port: 0-1stviolins-legato

NAME	0-1stviolins-legato
JACK_BINDINGS	

MIDI DEVICE 1 (JACK)

✓ Active Ports: 64

NAME	LinuxSampler1

Port: 0-2ndviolins-legato

NAME	0-2ndviolins-legato
JACK_BINDINGS	

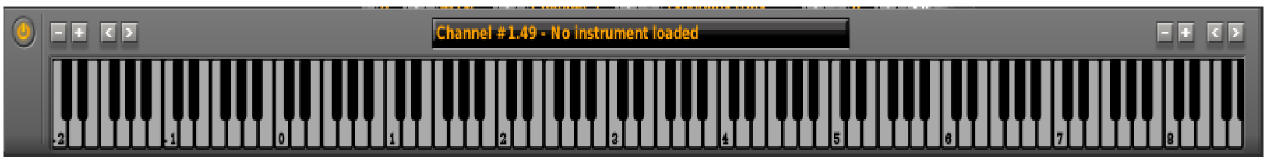
Audio Devices

AUDIO DEVICE 0 (JACK)

✓ Active Channels: 36

NAME	LinuxSampler
SAMPLERATE	48000

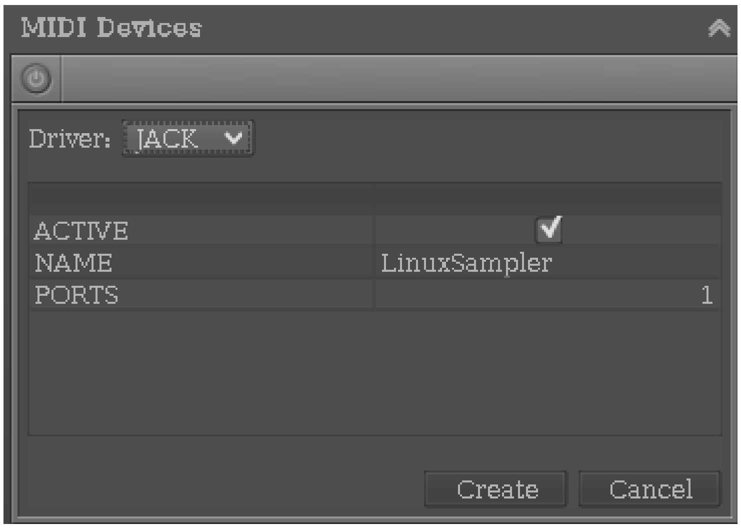
This is the LS devices section, where you create devices, in which you add the number of ports you require. LS has no arbitrary limits on how many you create, so your project can be as small or large as you want.


Channel #1.49 - No instrument loaded

This is the keyboard, which can be shown or hidden from view. It's a useful tool for those who may not use a hardware midi keyboard, or simply wish to test gig files in channels, to give just 2 examples. Click on the keys with the mouse, or use your midi keyboard as an input device.

New Configuration
Starting with a blank canvas, a configuration file, which is an *.lscp file in LS, can be created in Fantasia. The user begins with creating the required number of devices and ports he will require. These can be modified at any time, so a general idea of you want can be a good place to start.
New Midi Device

To create a midi device, press the "power" button icon in the top left of the midi section.
The window will open into a dialog, in which you can set your parameters for the new midi device.



Driver: JACK

ACTIVE	✓
NAME	LinuxSampler
PORTS	1

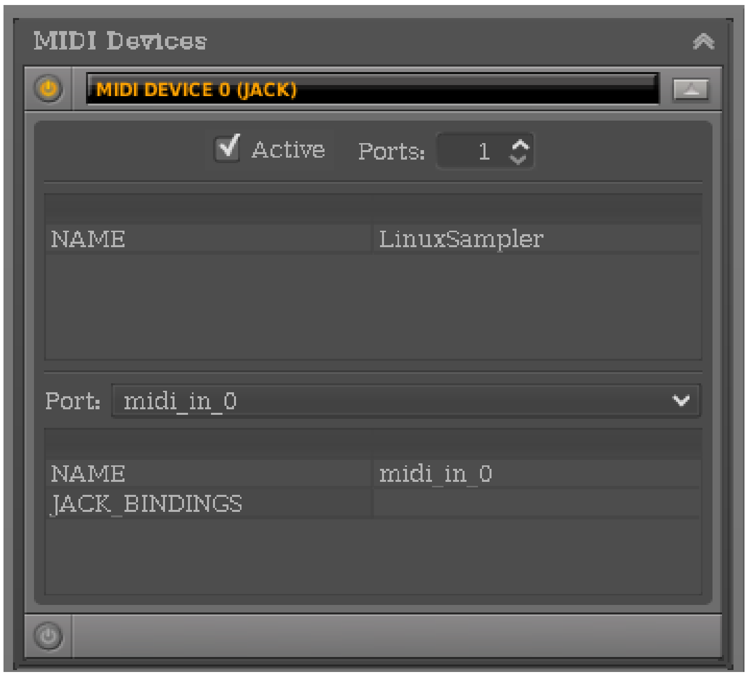
From the top, we can see the **DRIVER** dropdown list, and in this example, we've chosen Jack as the midi format. Depending on how you compiled LS in Linux, you can also choose from Alsa and Arts.

Next is the checkbox to make the device **ACTIVE**. We obviously want to tick this to use the device.

Next is the **NAME** of the midi device. You can choose your own, and in the examples to follow, we've created 2 midi devices, entitled *Linuxsampler* and *Linuxsampler1* . You can add as many devices as you want, and name them anything you like provided no two devices have the same name.

Under the name, is the **PORTS** configuration. In keeping with LS's design of no arbitrary limits where possible, you can add as many ports to the devices as you want, within any restrictions that might be present based on your hardware capability.

And finally, two buttons, labelled **Create** and **Cancel**. Select create to create the device you've built, or Cancel to quit the device configuration.



✓ Active Ports: 1

NAME	LinuxSampler

Port: midi_in_0

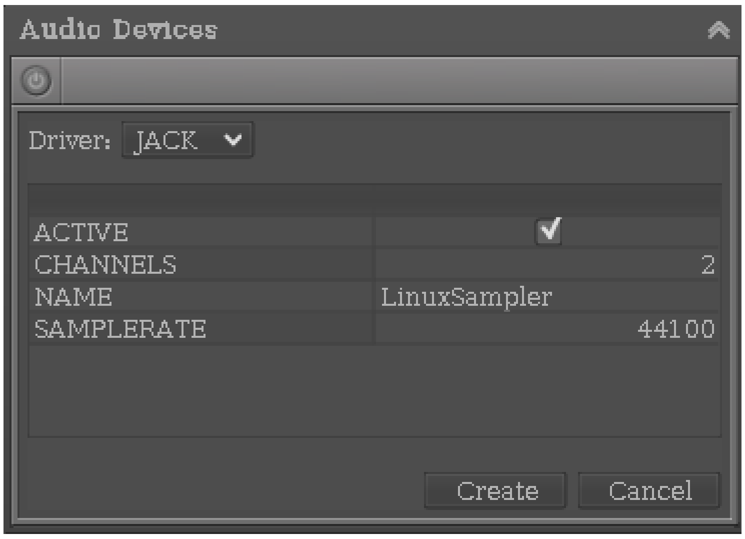
NAME	midi_in_0
JACK_BINDINGS	

Here's our newly created device, showing the configuration of the new device, entitled "Linuxsampler". Should you required more ports, simply increase the number in the value box. The port we've created can be renamed to anything you like, by double clicking on the port name itself in the **NAME** line.

Note that a new line has appeared, which is **JACK_BINDINGS**. For those who have a static template.lscp of some sort, and are using the same devices all the time, but not neccessarily using OOM2, you can bind midi ports direct to any currently running midi device, a list of which you'll see when you click on the box.

Be aware that if you bind multiple ports to the same jackmidi input device, then any midi data you feed into LS will play on all those ports at once.

New Audio Device



Driver: JACK

ACTIVE	✓
CHANNELS	2
NAME	LinuxSampler
SAMPLERATE	44100

The same procedure follows for creating an audio device. Press the power button, and a dialog will appear enabling the configuration of your audio device, and once again, you can create as many devices as your hardware will handle.

At the top of the dialog is the **DRIVER** dropdown list, and here we've chosen Jack.

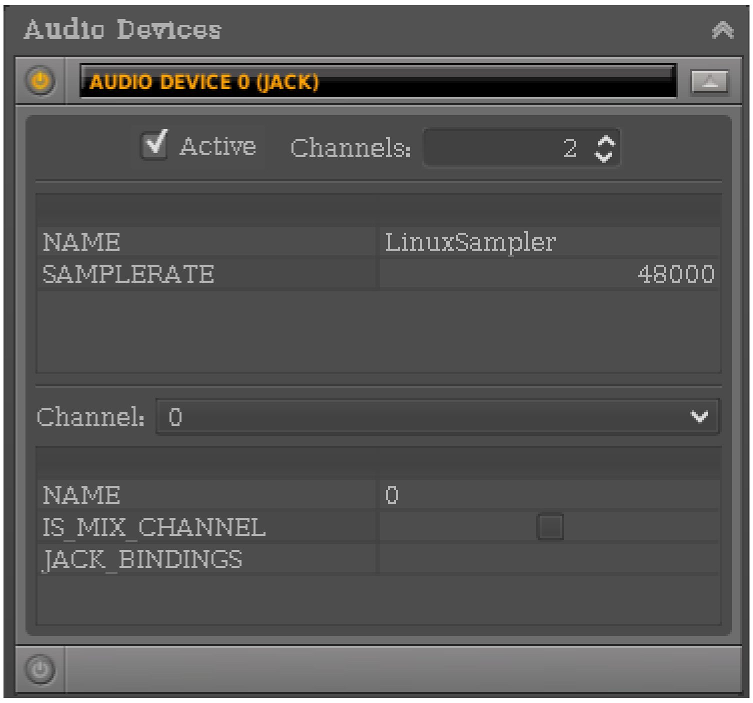
Next is the **ACTIVE** checkbox.

After that is the number of **CHANNELS** you can enable for the device. Channel strips in LS can be mono or stereo, so take some time to figure out what you're likely to need, then define the number of channels for your device. *There is no left and right when deciding how many channels you want, only the total number*.

Next is the **NAME** line, and as for midi devices, you can decide what to name your audio device.

Next is the **SAMPLERATE** line, which you can set to your requirements. When using LS with jack, it's often a good idea to match samplerates where possible, reducing the resampling on the fly required. Many soundcards in Linux have a default samplerate of 48000, so check your local soundcard parameters, and see what samplerate you're currently running.

Now you're ready to create your audio device, or cancel it. Click on **create** or **cancel** .



✓ Active Channels: 2

NAME	LinuxSampler
SAMPLERATE	48000

Channel: 0

NAME	0
IS_MIX_CHANNEL	☐
JACK_BINDINGS	

Your device is created, and you can see what you've chosen as the configuration.

The Name of the device and the samplerate were set during creation, and can't be changed unless you destroy the device and build another one, but the number of channels at the top of the device dialog can be changed at will.

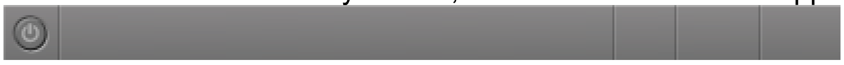
At any time you can change the channel name, in the **NAME** line to something you'll recognise easily. This is more than useful when you have a lot of channels.

Next you'll see the **IS_MIX_CHANNEL** checkbox.

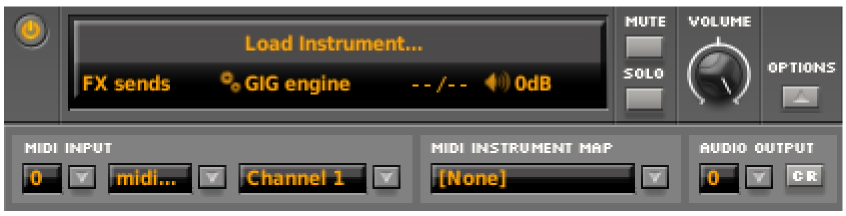
Finally, we have the **JACK_BINDINGS** dropdown list, in which you'll see currently running inputs to which you can bind LS channel outputs, per channel, when you click on the box.

At this point we diverge into two directions. The first is using LS as a simple sampler where GIG, SFZ, or SF2 files are loaded directly into channel strips, and the second is a more complex configuration using midi maps to create a bank and patch structure with which channels can load different files when triggered with bank and program change messages from OOM2. (or your midi keyboard, if it has this capability)

Case 1, The Simple Sampler
LS serves a strong role as a simple sampler, where GIG, SFZ, and SF2 files are loaded directly into channels. If the user has the hardware capability, with sufficient RAM, fast hard drives, and multiple core CPU's, a simple sampler structure can be significant in size, and with multiple ports, and channel strips covering a wide range of usage. A typical setup, with a dual core AMD, 4GB of RAM, and 2 x 7,200 rpm HD dedicated to sample libs, with 64mb of cache each, can run a large template effortlessly. (As tested here at the OpenOctaveProject) More powerful, and faster hardware, will yield even better results in a decently tuned system, where the "bloat" of multiple GUI heavy apps and Window Managers, are reduced where possible, leaving the hardware resource to more efficiently run LS, and other audio related apps.



To create a new channel, press the power button, in the centre vertical column in Fantasia. A new channel strip will appear.

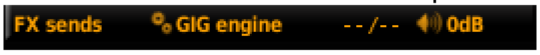


The channel strip has been created, and the component parts are as follows:

This button when lit means the channel is active. When you click on a lit button, you will delete the channel strip, after being prompted by a popup dialog box, asking you if this is really what you intend.

This is where you load your instrument into the channel. click on the strip and a dialog will open up asking you how you want to load the instrument, from your prebuilt orchestra, the database, or from a file directory.

In this section of the channel strip are 4 settings.



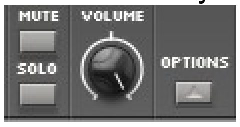
FX sends this opens a dialog where you can create FX sends for the channel strip, that can be controlled by user defined midi CC numbers.

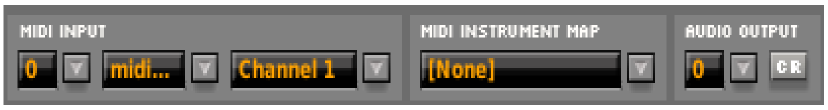
Gig engine This dropdown selection box lists the engine types you can use for the strip, GIG, SFZ, or SF2. You can only use one engine type for a channel strip.

--/-- This component shows the number of voices you're using for the strip, when a loaded instrument is playing. **--/--** means the strip is not playing anything at the moment.

0dB This function shows the dB value set for the channel strip.

In this section you have **mute** and **solo** , which locally affect the state of the channel strip, the **volume** knob, with which you can set the volume for the channel strip, and the **options** button, which shows or hides additional midi and audio options.





This is the options section of the channel strip where you define midi and audio settings.

Midi Input This section deals with midi input parameters. From the left is:
0 The device number with a drop down box to select another device if you've created more than one.

midi... The midi port name, with a drop down box listing the ports you've created for that device.

Channel 1 The midi channel number for the channel strip, which can be 1-16, and All.

Midi Instrument Map This value and drop down box deals with assigning a midi map to the channel strip, and will be covered more thoroughly in the second of our use cases.

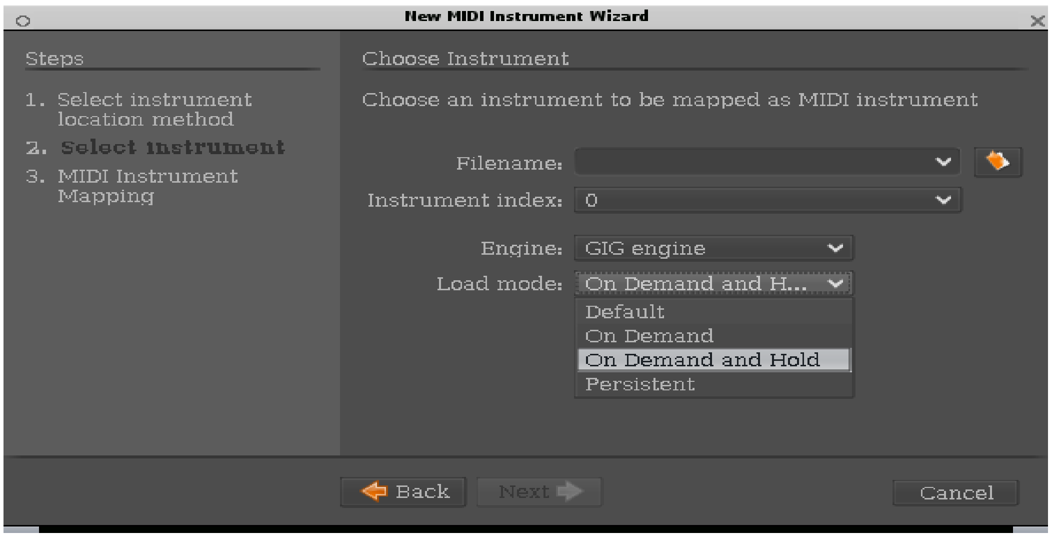
Audio Output
It's here that you'll assign audio outputs for the strip.

0 followed by an arrow. This is the audio device assigned to the track.

CR Press this button and a dialog will appear in which you can define audio output for the channel strip. Note that the default is stereo, meaning you may see something like 0 and 1. If you want the strip to be mono, assign the same number to both outputs.

Case 2, The Complex Sampler

As well as the simple configuration described in Case 1, LS is more than capable of taking on a complex role, using bank and patch changes to switch out loaded gig files from channels and replacing them with new ones. In order to understand this further, it's important to note the following:
When creating a midi map, and laying out the instruments, their load state, and how they behave, can be one of 4 definitions. The user creates a new midi map with a name in the top left section of Fantasia, using the **MIDI Instruments** tab. Below that is the working area for creating the maps themselves, and using the new addition button, the user opens
the **Midi Instrument Wizard** dialog.


New MIDI Instrument Wizard

Steps
1. Select instrument location method
2. Select instrument
3. MIDI Instrument Mapping

Choose Instrument
Choose an instrument to be mapped as MIDI instrument

Filename:
Instrument index: 0
Engine: GIG engine
Load mode: On Demand and H...

The components are relatively self-explanatory, with file loading, an index list for gig files that contain more than one instrument/articulation, the engine type, and the **Load Mode** . It's here in the load mode that we can see the 4 definitions of loaded sample behaviour.

The load mode functions are as follows:

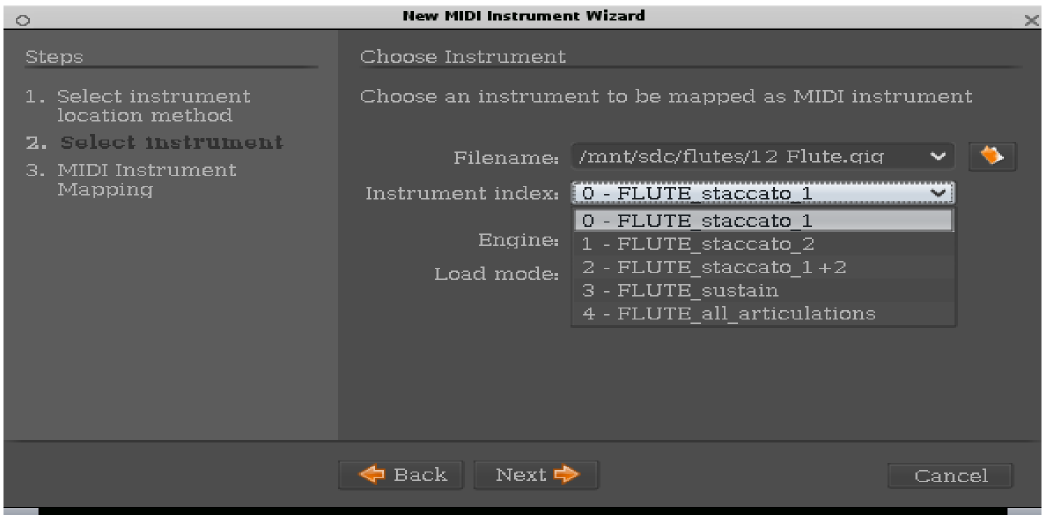
Persistent This is the state for loading a sample that is not a default, but is persistent in it's determination to stay loaded in the channel.

Default This is the state for loading a default gig file/index that gets loaded by default when the LS *.lscp file is loaded (The * defines "some" file name)

ON_DEMAND This is, in essence, a non-live state, where a sample will only be loaded when called upon. When another sample is requested for the same channel, the previous sample is unloaded, not only from the channel, but it's pre-sample portion is unloaded from RAM as well, freeing up resource. For sequencer based bank and patch changes sent to an ON_DEMAND gig file, there is an obviously delay while the old sample is unloaded, and the new sample is loaded, and this may prove unsuitable for accurate playback, where quick switching between bank and patch changes means the gig file doesn't have time to load before the next one is called. ON_DEMAND is ideal for those users who record from a sampler directly into an OOM audio track, using the midi keyboard input to switch banks and patches, but only recording audio, and not midi. The time it takes to load a gig file is fairly short, but with enough delay that the user can prepare himself for recording straight audio from his live playing.

ON_DEMAND_HOLD This is, one could say, the *sequencer* mode, where samples are loaded, and stay loaded, that is, their pre-sample portion that is loaded in RAM stays there, so the user can make multiple bank and patch changes in the same channel once all the gigs have been loaded, and any bank and patch changes from there are instant, with no delay in file playback even when issued with closely stacked bank and program changes. So the user can operate many gig files from the same track in OOM, switching banks and patches, and the playback is accurate, with every event being played as it should. For big scores, and lots of tracks, the user should note that even if multiple articulations are called from a single gig file that's loaded, LS only concerns itself with loaded the pre-sample for the file, so does so as a single instance, no matter how many times that file is used, in multiple channels if requested. The user should also note that as ON_DEMAND_HOLD retains the pre-sample portion, if the user continues to load different gig files, then the RAM usage will increase accordingly, so it's worth keeping an eye on the RAM in something like Htop, just to make sure you keep within a reasonable usage amount. At the OpenOctaveProject, we've tested LS with some sizable configurations, using lots of gig files, and LS remains remarkably efficient. RAM resources are finite, and if you keep loading gig files, you will eventually run out.

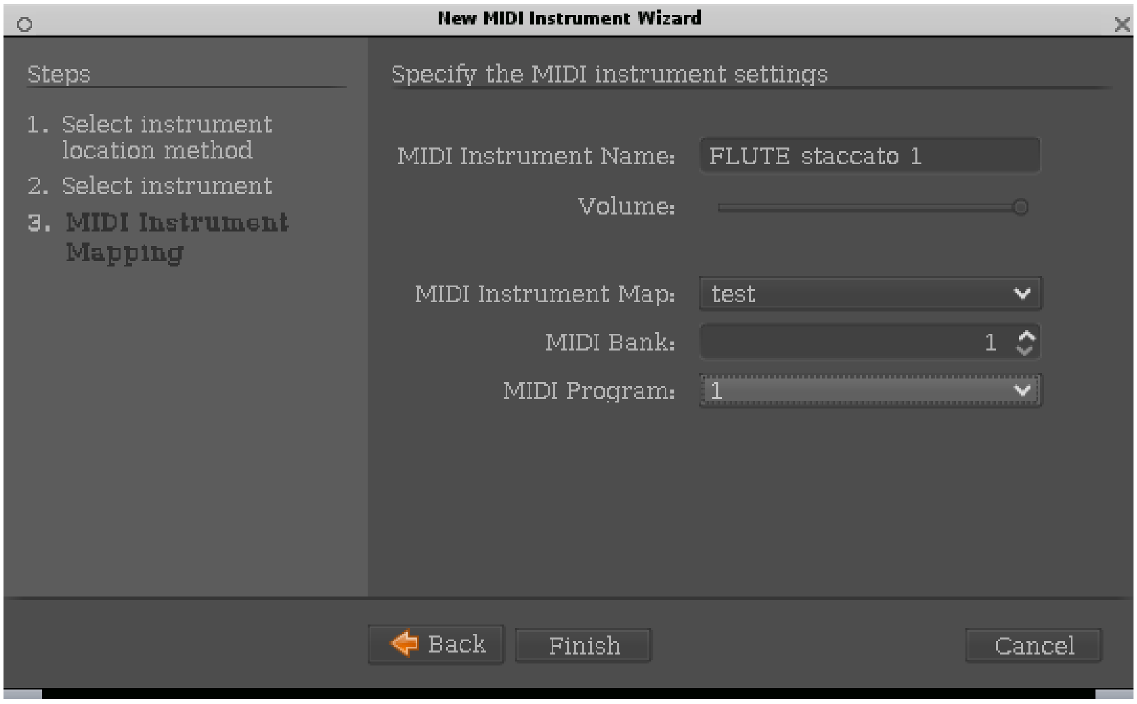
Now load mode has been explained, we'll continue with creating a new bank and patch for the midi map.


New MIDI Instrument Wizard
Steps
1. Select instrument location method
2. Select instrument
3. MIDI Instrument Mapping
Choose Instrument
Choose an instrument to be mapped as MIDI instrument
Back
Next
Cancel

Filename: /mnt/sdc/flutes/12 Flute.gig
Instrument index: 0 - FLUTE_staccato_1
Engine:
Load mode:

0 - FLUTE_staccato_1
1 - FLUTE_staccato_2
2 - FLUTE_staccato_1+2
3 - FLUTE_sustain
4 - FLUTE_all_articulations

I
n the picture, we can see a flute gig file has been selected from our sample directory, and that gig file contains multiple indexes. We choose the one we want, check we have the right engine (gig in this case), check the load mode, and then go to the next window to define a bank and patch number for this instrument.


New MIDI Instrument Wizard

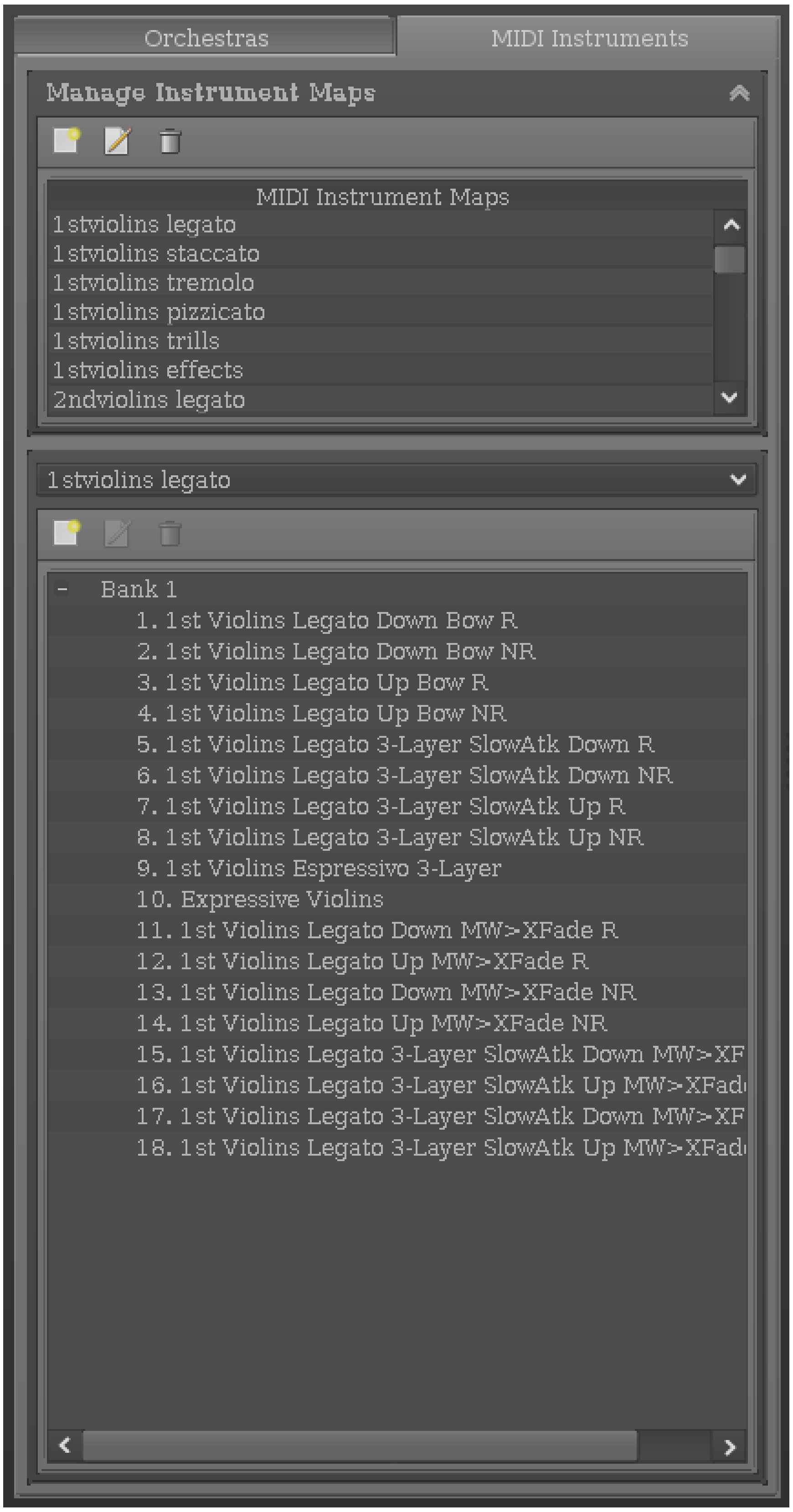
Steps
1. Select instrument location method
2. Select instrument
3. MIDI Instrument Mapping

Specify the MIDI instrument settings

MIDI Instrument Name: FLUTE staccato 1
Volume:
MIDI Instrument Map: test
MIDI Bank: 1
MIDI Program: 1

In the final dialog for creating a new bank and patch, we can see the name of our selected instrument, the volume slider for that instrument, should we wish to adjust it, the name of the map we're assigning the patch to, the bank value box, and the program value box. when we're sure we have it right, we press the fnish button, and our new bank and patch is created. For subsequent patches which we want to add to the same bank, in the same running session of Fantasia, the patch numbers will increase incrementally for each new patch we add. Be aware that if you mess one up, and delete it, the incremental function will not automatically step back one patch number, so check this carefully when you continue on.

It's usually tough enough building large bank and patch maps, without having to do it over and again. In LS, in the Actions menu at the top left of Fantasia, you'll see the export command, and a list appears to the right of that in which you'll see the item, **Midi Instrument maps** . Select it, and save your complete midi map as an *.lscp file without the added devices. This is your backup, and it would make sense (it does here) to save this file in more than one location, so you don't have tobuild the complete map again.

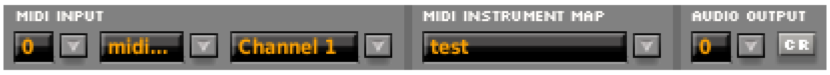

Orchestras
MIDI Instruments
Manage Instrument Maps

MIDI Instrument Maps
1stviolins legato
1stviolins staccato
1stviolins tremolo
1stviolins pizzicato
1stviolins trills
1stviolins effects
2ndviolins legato

1stviolins legato

- Bank 1
  1. 1st Violins Legato Down Bow R
  2. 1st Violins Legato Down Bow NR
  3. 1st Violins Legato Up Bow R
  4. 1st Violins Legato Up Bow NR
  5. 1st Violins Legato 3-Layer SlowAtk Down R
  6. 1st Violins Legato 3-Layer SlowAtk Down NR
  7. 1st Violins Legato 3-Layer SlowAtk Up R
  8. 1st Violins Legato 3-Layer SlowAtk Up NR
  9. 1st Violins Espressivo 3-Layer
  10. Expressive Violins
  11. 1st Violins Legato Down MW>XFade R
  12. 1st Violins Legato Up MW>XFade R
  13. 1st Violins Legato Down MW>XFade NR
  14. 1st Violins Legato Up MW>XFade NR
  15. 1st Violins Legato 3-Layer SlowAtk Down MW>XF
  16. 1st Violins Legato 3-Layer SlowAtk Up MW>XFad
  17. 1st Violins Legato 3-Layer SlowAtk Down MW>XF
  18. 1st Violins Legato 3-Layer SlowAtk Up MW>XFad

You can now go on with creating the rest of your map, and as you can see in this picture, the midi map has many entries, each one structured with banks and patches.



In the above picture, in the Midi Instrument Map section, we've loaded a midi map called test. When OOM2 sends bank and patch change data to this channel, the patches listed in the map will respond, and change to a new patch.

There's one more task to do here, and that's save not only our midimap in a separate file, but save our new test.lscp file as well. Using the Hard Drive icon at the top left of the app, we can save the project into the directory of our choice. As our project for examples is called *test*, we're going to save this as *test.lscp*, in a directory/folder we've created called templates. .lscp files are just data, and have no significant size, so using a templates directory/folder on out home drive is of little concern. Be sure to regularly save this folder somewhere else, as a backup.

Running your LS template from a terminal

As we've created our LS template successfully, we can now take advantage of the backend/GUI structure of Linuxsampler, and leave the GUI out, running our test.lscp file from a terminal, and save valuable CPU cycles in the process of doing so.
This is a simple process requiring just 2 terminals. First, and after you have JACK up and running, open a terminal, and type:

```
linuxsampler
```

This starts the LS backend.

Now open the 2nd terminal, and using test.lsp as our example, we use the cat command to read firstly the path to our test.lscp file, then issue a netcat command to LS. My test.lscp is in my home directory, in a folder called templates, so the cat command in the terminal will look like this:

```
cat /home/myname/templates/test.lscp | nc localhost 8888
```

The nc stands for netcat, and localhost represents the computer/ip linuxsampler is installed on. If i had linuxsampler installed on another machine inside my LAN, with an ip, for example, of 127.0.0.10, then my command would look like this:

```
cat /home/myname/templates/test.lscp | nc 127.0.0.10 8888
```

The linuxsampler backend can be run from anywhere as long as we can issue netcat commands to it, so LS can load the contents of the test.lscp file. This is particularly useful in a LAN where the user seeks to spread the load of running a DAW environment across his LAN, or to use a modern phrase, his "local cloud".

work in progress......