# Linux Midi Orchestration

*Peter Schaffter*

## Contents

# List of Figures

# ***Preface***

The ability to write orchestral scores and generate playback from them is one of the great gifts of the digital age. With the cost of symphony concerts out of reach for most people and the number of professional orchestras dwindling, composers stand very little chance of having new works performed. Community and student orchestras, too, are falling by the wayside, leaving neophyte orchestrators few opportunities to perfect their craft.

Midi orchestration, by which is meant digital audio performances rendered by a virtual orchestra from computer-generated scores, ensures that composers have a means to get their music get heard, and students have an invaluable tool for learning orchestration.

Western art music, particularly orchestral music from the mid-eighteenth century onward, is one of the crowning artistic and intellectual achievements of human kind. It must not be allowed to fall victim to the forces presently impoverishing everyone, culturally as well as economically. Midi orchestration is a candle in the darkness of a world beset by commercial musical interests, fostering the composition and performance, albeit digital, of music that continues to delight the ears, stir the heart, challenge the mind, and nourish the soul.

The goal of midi orchestration isn't perfect realism. Such is not possible, even with the finest software. The goal is a simulacrum, the way a statue is a simulacrum of a person: all the features are accurately rendered but there is no mistaking it for a human being. Just the same, the word realism fits our purpose because it is our model and our guide. A statue can be beautiful in its own right. So, too, can midi orchestration.

This paper deals specifically with midi orchestration on a Linux system. Commercial applications and sample libraries exist for Windows and Mac, but the vendors refuse to release their products for Linux. What's more, the software can run to thousands of dollars, out of reach for many composers and students. Free and open source software on Linux may be the only option they have.

Furthermore, the paper addresses itself primarily to composers and students with sufficient training to understand the issues involved. A basic knowledge of instrumentology and score writing is expected, as well as some familiarity with midi, acoustics, and audio production. The intent is not a shortcut for producing scores that sound like Rimsky-Korsakov or John Williams without study.

Lastly, a good working knowledge of the MuseScore notation program is essential. No other graphical notation program under Linux produces scores of comparable excellence, making it the only option for score-driven midi orchestration if the scores and parts are to be printed and distributed.

Setting up the virtual orchestra is an enormous task. If one is not familiar with the software, the learning curve is vertiginous. The complexity of the audio chain results in unpredictable and destructive crashes. The dearth of high-quality, non-commercial soundfonts means long hours of scouring the Web, and even longer hours auditioning the pickings. A library of virtual instruments must be created. Selecting the best instruments, balancing them, and placing them correctly in a virtual hall requires a knowledge of acoustics and experience in orchestration. Be prepared to spend several weeks to several months on the project.

It is my hope that, in time, the work will not be necessary, or, if it remains so, that the components of the audio chain become more polished and integrate better. Presently, crashes resulting in unrecoverable losses are the most frustrating aspect of setting up a virtual orchestra under Linux.

The biggest hurdle, however, is knowledge of the software involved, without which it is impossible to proceed. The scope of this paper does not leave room for tutorials, so some familiarity with the software listed in the first chapter is expected.

# Preliminaries

## Software prerequisites

- **JACK**
  JACK is the *de facto* Linux audio and midi connection kit.
- **Cadence**
  Cadence provides configuration and control tools for JACK. Online documentation is available at the Cadence Documentation website. Cadence should be kept loaded at all times.
- **Carla**
  Carla provides a JACK patchbay for connecting audio and midi ports. It also provides a plugin rack for audio effects and filters. Carla looks deceptively like a good candidate for an all-in-one orchestral setup because it also provides sampling, however it cannot handle the quantity of soundfonts and plugins required for a full virtual orchestra.
- **MuseScore**
  MuseScore is a music notation program, or score editor, with midi capabilities. As far as notation goes it's excellent, but the midi is rudimentary. It is to be hoped that this situation will improve since no other open-source WYSIWYG score writer is as polished as MuseScore. It is, in fact, the only practical choice for midi orchestration on a Linux system.
- **LinuxSampler**
  Linuxsampler holds the soundfonts that make up a virtual orchestra. There are other samplers and sampler-capable DAWs (digital audio workstations), but none perform as reliably as LinuxSampler, which, moreover, natively plays soundfonts in all three major formats.
- **LMMS**
  Like Carla, LMMS looks like a good solution to all-in-one setups but is unusable for virtual orchestras. The concept is sound and the interface a joy—one of the few Linux audio applications about which that can be said—but playback from MuseScore over the a2jmidi bridge, which is required, is appallingly laggy and unsynchronized. LMMS is, however, a good tool for previewing soundfonts before setting up an orchestra.
- **Non-session-manager**
  A session manager is essential for audio setups, keeping track of which

programs are needed and their JACK connections. Claudia, a session manager from KXStudio, is more polished and reliable, and furthermore has its own JACK patchbay. It cannot, however, manage all the connections needed for a full virtual orchestra.

- **Non-mixer**
  Non-mixer not only mixes, but acts as a plugin host. It features ambisonic placement (left/right, front/back) for the instruments of a virtual orchestra and integrates well with Carla.
- **JAMIN**
  JAMIN provides parametric equalization, look-ahead limiting, 3-band compression, and 3-band stereo width control.
- **the command line**
  Some operations, such as making backups, are accomplished more reliably and more efficiently at the command line than with GUI tools.
- **a text editor**
  The xml template for an orchestral score has to be edited by hand to add essential functionality.

JACK, MuseScore, LMMS, LinuxSampler, Non-session-manager, Non-mixer, and JAMIN are available as packages from most Linux distributions. Carla and Cadence must be acquired from KXStudio.

## Score template

The first step in setting up a virtual orchestra is to create a template score in MuseScore. The template will necessarily be large since it must include all the instruments typically found in a modern orchestra. For a standard orchestra with winds in pairs, staves are needed for all of the instruments listed in Figure 1. Additional percussion and miscellaneous instruments may be added. Figure 2 shows shows a typical winds-in-pairs score template with standard nineteenth-century percussion.

An orchestra with winds in threes requires additional staves, as listed in Figure 3. Similar to the trombones in the winds-in-pairs template, the winds-in-threes template accommodates the notation of any combination of players within a woodwind department. For example, if a piccolo and two flutes are needed, the piccolo is notated on the Piccolo staff and the two flutes on the first or both of the Flutes staves. If all three flutes are needed,

| **Woodwinds** | **Brass** | |
|---|---|---|
| Piccolo | French Horns I.II. | |
| Flutes I.II. (II—piccolo) | French Horns III.IV. | |
| Oboes I.II. (II—English horn) | Trumpets I.(II.) | |
| English Horn | Trumpets (II.)III. | |
| Clarinets I.II. (II—Bass Clarinet) | Trombones I.(II.) | |
| Bass Clarinet | Trombones (II.)III. (III—Bass Trombone) | |
| Bassoons I.II. (II—Contrabassoon) | Bass Trombone | |
| Contrabassoon | Tuba | |
| **Timpani and Percussion** | **Miscellaneous** | **Strings** |
| Timpani | Harp I | Violins 1 |
| Snare Drum | Harp II | Violins 2 |
| Bass Drum | Piano | Violas |
| Cymbals | Voice | Cellos |
| Tubular Bells | | Basses |

**Fig. 1.** *Staves needed for winds in pairs*

they can be spread out over the two Flutes staves, either I and II on the first and III on the second, or I on the first and II and III on the second.

It is crucial to think through the template carefully, making sure it includes all the needed instruments. Adding an instrument after the virtual orchestra is set up requires reassigning midi channel numbers and redoing the connections for every instrument that falls below the addition. The purpose of the template is to avoid this sort of work as much as possible.

Unlike adding instruments, subtracting them does not entail extra work. Instruments not needed for a project are removed from the score by unchecking **Visible** in MuseScore's **Edit→Instruments...** dialogue.[1] The staves vanish and the score spaces itself as if they weren't there. No further work is required. If, later on, an instrument needs to be restored, checking **Visible** returns it.

It is wise to attend to the visual appearance of the template at the same time as selecting the instruments. Staves should be labelled correctly, brackets and accolades added where required, and default settings for various ele-

[1]The **Visible** checkbox is not present when adding an instrument to the instrument list. Once the instrument has been added and the **Edit→Instruments...** dialogue closed, **Visible** subsequently appears whenever the dialogue is re-opened.

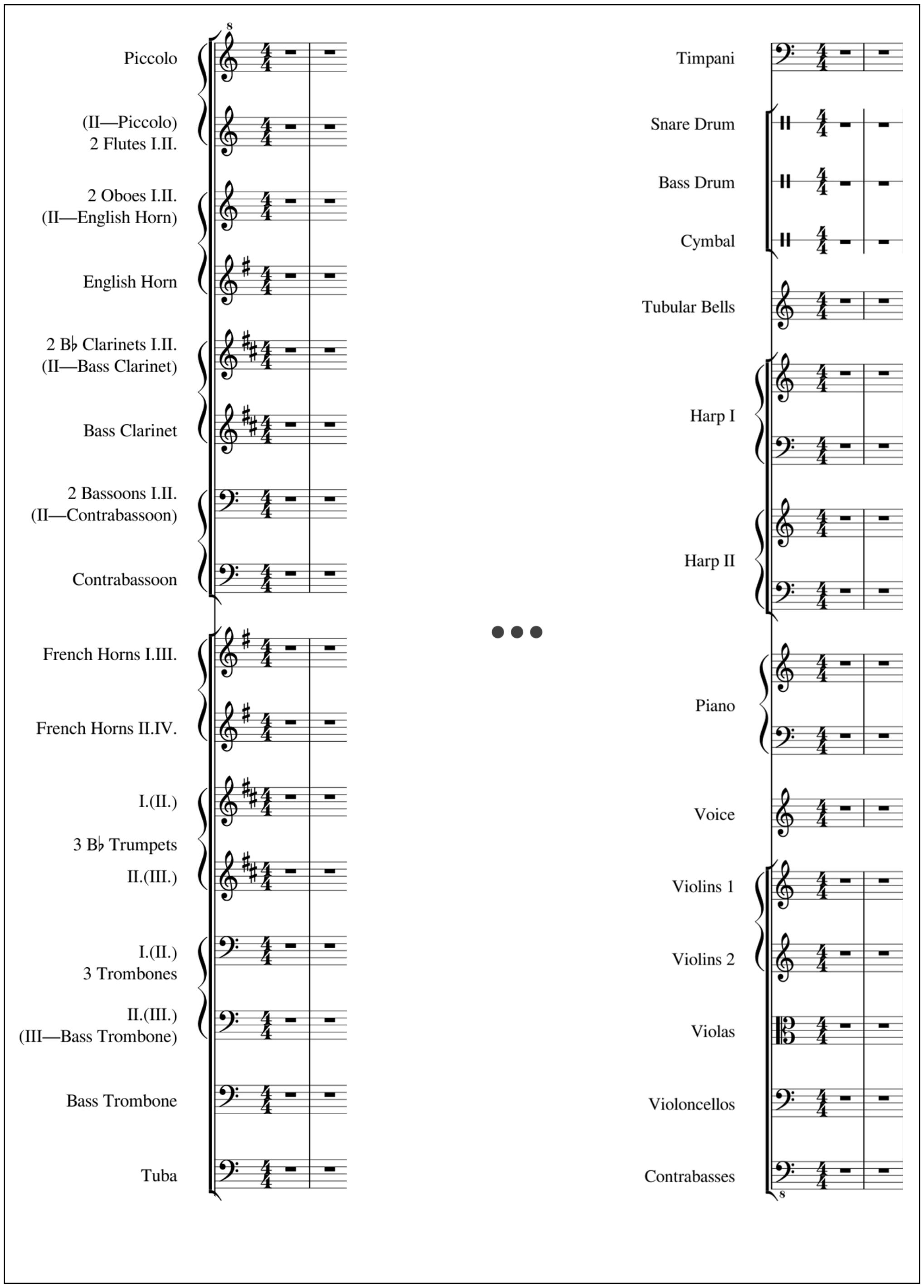


**Fig. 2.** *Score template, winds in pairs*

ments of the score adjusted in the **Style→General...** and **Style→Text...** dialogues. Again, the purpose of the template is to avoid repetitive work, so it is well to attend to these details at the outset.

When the template is complete, it must be saved as an .mscx file.

Piccolo
Flutes I.(II.)
Flutes (II.)III. (III—piccolo)
Oboes I.(II.)
Oboes (II.)III. (III—English horn)
English Horn
Clarinets I.(II.)
Clarinets (II.)III. (III—Bass Clarinet)
Bass Clarinet
Bassoons I.(II.)
Bassoons (II.)III. (III—Contrabassoon)
Contrabassoon

**Fig. 3.** *Staves needed for winds in threes*

## Adding extra channels to staves

Work on the template is not finished when it is saved. Understanding what remains to be done, and why, requires some explanation.

In MuseScore, with two exceptions, every staff is assigned a single midi channel, which, in a virtual orchestra, is connected to a single soundfont. Notes on the Flutes staff go to a flute soundfont, notes on the Trombones staff go to a trombone soundfont, and so on.

But is a single soundfont adequate for the range of sounds expected from a department of the orchestra? In almost all cases, the answer is no. A fortissimo horn sounds very different from a pianissimo horn, not merely louder. An oboe playing solo and two or three playing in unison have distinctly different timbres.

To accommodate the significant timbral differences expected from any department of the orchestra, it is necessary to have more than one channel per staff. Surprisingly, the MuseScore GUI does not provide a facility to increase the number of channels associated with a staff. Adding the extra channels requires editing the .mscx source file directly.

Before editing the source file, due consideration must be given to how many channels are needed for which instruments. Failure to allot an optimal number means unnecessary rewiring later on. The following guidelines are useful.

> ***Winds***
> Winds need an additional **section** channel (*sect.*), plus one or two **alternative channels** (*alt.*) to deal with unforeseen circumstances and special requirements. Diverse soundfonts are easily assigned to *alt.* channels on a project-by-project basis.
>
> ***Brass***
> French horns, trumpets, and trombones need *sect.* and *alt.* channels. French horns furthermore need a *stopped* channel, while trumpets and trombones may need channels for different types of mutes (straight, cup, Harmon, etc.).
>
> ***Timpani, percussion, miscellaneous***
> Timpani need an extra *roll* channel. Percussion and miscellaneous instruments may not need additional channels; it depends on the instrument.
>
> ***Strings***
> Strings already come with three channels, normal, pizzicato, and tremolo, but this is insufficient for the variety of sounds a string section is called upon to produce. Strings benefit from having at least two *alt.* channels—more, if desired—which can be used for different purposes depending on the needs of a project.

## Editing the template's .mscx source file

The template's .mscx file is an .xml file. **`<Channel>`** stanzas are nested within **`<Instrument>`** stanzas, just before the closing tag. Figure 4 shows a truncated **`<Instrument>`** stanza with the default, single **`<Channel>`** stanza.

New channels are added just above the closing **`</Instrument>`** tag, and take the form

```
<Channel name="id"></Channel>
```

where **`id`** is the name the channel will be given in the **Staff Text Properties...** channel selection dialogue in MuseScore, e.g. "sect." or "alt."

By default, the first channel—the one already there—has the name "normal." The *sect.* and *alt.* channels are added with

```
<Channel name="sect."></Channel>
<Channel name="alt. 1"></Channel>
<Channel name="alt. 2"></Channel>
```

```
<Instrument>
  <longName>Flute</longName>
  <shortName>Fl.</shortName>
  <trackName>Flute</trackName>
  ...
  <Channel>
    <program value="73"/>
    <synti>Fluid</synti>
  </Channel>
</Instrument>
```

**Fig. 4.** *Truncated .mscx instrument stanza*

Because the midi output from MuseScore will be going to an external sampler, there is no need for the `<program>` and `<synti>` tags.

Proceed through the .mscx file and add the desired number of channels, appropriately named, to each `<Instrument>` stanza. A good text editor with regexp substitution can speed up the process.

## CREATING A PORT/CHANNEL MAPPING FILE

MuseScore follows the General MIDI standard. Every midi output port (midi-out) carries sixteen channels, with channel 10 reserved for non-pitched percussion.[2] Midi-outs are created automatically by MuseScore whenever the number of channels in a score exceeds a multiple of sixteen. A score with sixty-four channels has four MuseScore midi-outs.

Confusingly, MuseScore's midi-outs, which are labelled "mscore-midi-*n*" in the **Carla** patchbay, begin at "1," whereas LinuxSampler's midi-ins begin at "0." Adding to the confusion, **LinuxSampler**'s audio output channels ("audio-outs") are not grouped by ports, but numbered sequentially, starting at "0." A score with sixty-four channels therefore has audio-outs numbered 0 – 63. The number is the only label given to audio-outs in the patchbay, so it is impossible to see which audio-out carries the signal for which instrument, as Figure 5 shows.

[2]Timpani, marimba, xylophone, and other pitched percussion instruments do not go on channel 10.

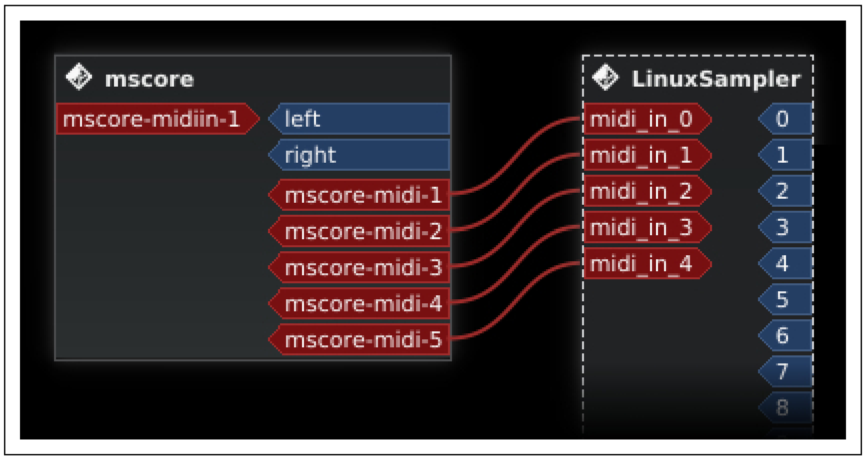


**Fig. 5.** *Carla patchbay, MuseScore and LinuxSampler*

What's more, LinuxSampler soundfonts must be told which port and channel to listen on, and which audio output channel will carry the signal.

Without a guide to the port/channel mappings from MuseScore and the audio-outs from LinuxSampler, setup and changes to a virtual orchestra become insurmountably complex. It is essential to keep a reference file that holds the mapping information. The document type—plain text, PDF, word processing document—is unimportant. Figure 6 demonstrates a possible layout, based on the score template for winds in pairs. Note that LinuxSampler's audio out "0" is not used.

The italic rows show the MuseScore and LinuxSampler ports that will be connected in Carla. The **midi_in** and **audio out** channels are set in LinuxSampler, which also requires the *midi_in* port number from the italic rows. Notice that **audio out** continues to increment after a port change, while **midi_in** begins again at 1.

As noted, channel 10 of every midi input port is reserved for non-pitched percussion. Upon reaching channel 9 of any port, as with the English horn in Figure 4, the next available channel—in this case, for an alternative English horn—is 11, not 10. Failure to remember this knocks the remainder of the mapping file out of whack, defeating its purpose.

Of equal importance is remembering that MuseScore pre-allots three channels to each department of the string section: normal, pizzicato, and tremolo. Assuming two *alt.* channels and an intervening channel 10, the mapping for Violins 1 might look like Figure 7.

| midi_in | audio out | Instrument name |
|---|---|---|
| *mscore-midi-1 → LinuxSampler midi_in_0* | | |
| 1 | 1 | Piccolo |
| 2 | 2 | Piccolo (alt.) |
| 3 | 3 | Flute |
| 4 | 4 | Flutes (sect.) |
| 5 | 5 | Flute (alt.) |
| 6 | 6 | Oboe |
| 7 | 7 | Oboes (sect.) |
| 8 | 8 | Oboe (alt.) |
| 9 | 9 | English Horn |
| 10 | 10 | ---reserved--- (Snare Drum) |
| 11 | 11 | English Horn (alt.) |
| 12 | 12 | Clarinet |
| 13 | 13 | Clarinets (sect.) |
| 14 | 14 | Clarinet (alt.) |
| 15 | 15 | Bass Clarinet |
| 16 | 16 | Bass Clarinet (alt.) |
| *mscore-midi-2 → LinuxSampler midi_in_1* | | |
| 1 | 17 | Bassoon |
| 2 | 18 | Bassoons (sect.) |
| 3 | 19 | Bassoon (alt.) |
| ... | ... | ... |

**Fig. 6.** *Example port/channel mapping file*

Again, notice that **audio out** has continued to increment while **midi_in** has been resetting to "1" every sixteen channels.

**Instrument name** becomes important when setting up and connecting Non-mixer.

## ACQUIRING SOUNDFONTS

After the score template and mapping file are complete, soundfonts are needed for the instruments that make up the virtual orchestra.

| midi_in | audio out | Instrument name |
|---|---|---|
| 8 | 53 | Vlns. 1 normal |
| 9 | 54 | Vlns. 1 pizz. |
| 10 | 55 | ---reserved--- (Triangle) |
| 11 | 56 | Vlns. 1 trem. |
| 12 | 57 | Vlns. 1 alt. 1 |
| 13 | 58 | Vlns. 1 alt. 2 |

**Fig. 7.** *Mapping file, Violins 1*

Soundfonts are composed of audio samples of notes played on a real instrument. The samples are collated in a soundfont editor, the range of notes each sample covers is established, and various tweaks are performed. The ideal is a virtual instrument that displays the same timbral characteristics and dynamic capabilities as the real thing throughout its entire range.

The three major soundfont formats are .sf2, .sfz, and .gig. By far, the commonest is .sf2. For convenience, this paper assumes .sf2, although LinuxSampler can play all three.

Soundfonts may be grouped together into soundbanks, usually arranged according to the General Midi specification. These one-size-fits-all collections are never adequate for serious midi orchestration.

A Google search for "soundfont" or "sf2" reveals a wealth of freely-available soundfonts, but the quantity is not matched correspondingly by quality. Even the most promising soundfonts invariably reveal flaws. Extensive previewing is unavoidable.

A good program for previewing is LMMS, which requires minimal setup, has an onscreen keyboard, and can easily add reverb. Reverb is vital to realism in a virtual orchestra, so previewing with reverb is essential. Any other program or audio setup that provides reverb may be used instead.

## What to look for in a soundfont

The perfect orchestral soundfont has the following characteristics:

- neutrality with respect to
  - *timbre*
  - *attack*

  - *vibrato*
  - *swell*
- looped samples
- normalized samples
- does not use velocity layers
- has no reverb
- is mono, not stereo

## *Neutrality*

### *Timbre*

Neutrality of timbre means that a sustained note at moderate volume exhibits the characteristic timbre of the instrument *as it would sound if played by a professional in an ensemble passage.* Most soundfonts are divas. Finding ones that are comfortable being orchestral players is exceptionally difficult, particularly in the brass section.

### *Attack*

The ideal attack is neither too crisp nor too gentle. For example, too much chiff on the flute means it will never play a decent legato, and trills will wind up sounding like machine gun fire. On the other hand, not enough chiff results in flabby phrases and the impression of a player who never takes a breath.

Strings with a satisfactory attack are particularly difficult to find. Usually, the attack is too aggressively *martelé*, or so slow that notes may not reach full volume even at moderate tempi. A compromise between the two is essential, and well worth holding out for.

### *Vibrato*

Vibrato is a terrible problem, especially in soundfonts for winds. Brass is sometimes affected, too. Where vibrato is concerned, none is better than some, and some should be minimal and unobtrusive. In a virtual orchestra, the only instrument that benefits from a modest vibrato is the oboe.

### *Swell*

"Swell" means that notes begin softly and grow to full volume. While this can add expressivity to a solo soundfont, it is impossible to work with in orchestral music. As with slow strings, long notes don't reach full volume until well after they're sounded, and shorter notes don't have time to reach full volume at all.

### *Looped samples*

If an instrument is capable of producing a sustained tone, the soundfont's samples should be looped, which means that when a note is sounded, it can continue indefinitely. Looping has critics, sticklers who feel that samples shouldn't exceed the length of a sustained tone as playable by a live performer. While that may be true for "solo" soundfonts, it's poppycock for orchestral soundfonts. The illusion of a perfectly sustained note over many bars is stock in trade for orchestral players.

### *Normalized samples*

Another flaw present in many soundfonts is a lack of consistency in the volume of the samples. A correctly-designed soundfont begins with normalized samples, i.e. samples whose volumes are the same throughout the whole range of the instrument. When normalization is skipped, some notes are noticeably louder or softer than their neighbours.

The fault is corrected by adjusting the offending notes' velocities in the score, or by normalizing the samples' volumes and rebuilding the soundfont. Both are time-consuming, but one or the other is unavoidable.

### *Velocity layers*

Velocity layers give rise to another pitfall. Almost every instrument exhibits timbral differences when the instrument is played louder or softer. Velocity layers attempt to reproduce this by switching to a different set of samples when the dynamic level of the score goes up or down. Attempt, but fail. Velocity layers kick in when a specific velocity is reached, but if the switch is not desired, there's no way to get rid of it.

The symptom of velocity layers is that increasing a note's velocity by as little as 1 or 2 (velocities are on a scale from 0 – 126) results in a note that's much louder than expected.

In a virtual orchestra, a better solution to timbral differences resulting from changes in dynamics is to have separate soundfonts for different dynamic levels; "*alt.*" channels are ideal for this.

### *No reverb*

More than an effect, reverb is practically an instrument to itself. In real life, reverb exposes the full beauty of an instrument's tone. In midi orchestration, reverb creates a "hall" for the virtual orchestra and permits control over the apparent distance of various instruments from the audience. Orchestras are not merely arranged left to right on stage, but front to back as well. The correct application of reverb permits placing the winds behind the strings, the brass behind the winds, and the timpani and percussion at the back of the stage.

To be used effectively, reverb must be user-controllable for all instruments and sections, which means the soundfonts in a virtual orchestra should start off "dry." Many soundfonts add their own reverb and have to be discarded no matter how ideal they are in other respects.

### *Mono vs. Stereo*

No instrument or instrument section spans the entire stage in real life, but that's the effect of stereo in a soundfont. For midi orchestration, mono soundfonts are preferable because they allow panning instruments to their correct location in the sound field.

Stereo soundfonts don't have to be avoided altogether; the output can be mixed to mono before it goes down the audio chain.

## The string section

Strings, the heart of an orchestra, present the greatest challenge to the midi orchestrator in search of soundfonts.

The General Midi specification allots four slots for the string section. Contrary to expectation, the slots are not for violins, violas, celli and basses, but rather Strings, Slow Strings, Pizzicato Strings, and Tremolo Strings. In other words, the GM specification treats the string section as a single, homogeneous group and dispenses with the significant timbral differences that exist between the departments.

This thinking is entrenched in the world of midi, with the result that there are very few soundfonts dedicated to the separate string departments. Some are to be found in soundbanks, but the quality usually varies so much from department to department that none is usable as a complete package.

There are two workarounds for this problem, neither ideal. One is to find "Strings" or "Slow Strings" soundfonts that, when playing in the typical register of a particular string department, most closely approximate the timbre of that department. This usually means four separate Strings soundfonts, one for each department. Finding ones that match in terms of attack and articulation is nearly impossible.

The other workaround is to find a Strings soundfont whose attack and articulation are satisfactory, then use that same soundfont for all departments, applying equalization to mimic the correct timbres.

Of the two workarounds, the second yields the better result.

Other than the canned sound that afflicts Strings soundfonts generally, their biggest failing is in the matter of attack. Generic GM "Strings" should, or so one would think, have an attack resembling normal, détaché bowing. "Slow strings" should approximate a decent legato at slow tempi. Neither is usually true. "Strings" frequently have an attack so aggressive the only place they could be used is the opening of Beethoven's Fifth. Conversely, "Slow strings" are so languorous they can only be used as pads in orchestral pop.

A good, general purpose strings soundfont is one that

- is timbrally realistic at varying dynamics throughout its entire range;
- has a sufficiently short attack that notes reach full volume (the "sustain" phase of the envelope) almost immediately;
- sound approximately right when differing articulations and bowing styles are applied (staccato, spiccato, louré, etc.).

From the above, it can be inferred that selecting strings soundfonts entails compromise. No one soundfont can perfectly mimic all the articulations and shades of colour of a real string section. Rather than focusing on excellence in one area of general purpose strings, one should listen for "close enough" in all areas.

It is useful to have several soundfonts available for each string department. A general purpose Strings soundfont on the "normal" channel can, for example, lose brilliance and crispness in passages of short staccato; having an alternate soundfont with a brighter timbre and faster attack/delay may improve the realism of such passages. The same goes for significant colour changes that may be requested in a score: *sul tasto, sul ponticello,* harmonics, etc. The additional soundfonts can be allotted to *alt.* channels.

### Chorused woodwinds

Two or more of the same instrument playing in unison produces a "chorus effect," a faintly shimmery quality resulting from minute differences in intonation, tone quality, and amplitude. In midi orchestration, it is essential that this effect be reproduced when a woodwind department is playing in unison.

Sadly, there are almost no freely-available "section" soundfonts for winds. The effect must be applied artificially with a chorusing plugin attached to Non-mixer or loaded in Carla. The soundfonts to be used for chorusing go on the *sect.* channels of the mapping file.

### Brass sections

Artificial chorusing, while useful for woodwinds, is unsatisfactory for brass. In scoring, brass departments in unison is generally reserved for forte passages, and brass department soundfonts ("Trumpet Section," "Heroic Horns") usually use samples of the department playing forte, so they work quite well.[3]

## Building a library

The quantity of soundfonts needed for a virtual orchestra, and their occasionally whimsical names, makes accessing and keeping track of them difficult. A well-organized library is essential. A "Soundfonts" directory should be created, with subdirectories for the various instrument types (Figure 8).

Within the subdirectories, it is advisable to rename the soundfonts selected during the preview process using a meaningful naming scheme. The goal is to ensure that instruments of the same type are grouped together in file choosers and browsers, which is not trivial. A good naming scheme is

**`Section-Instrument-<id>.sf2`**

where

**`Section`** is the same as the directory: **`Woodwinds`**, **`Brass`**...
**`Instrument`** is the name of the instrument: **`Flute`**, **`Trumpet`**...
**`<id>`** is numeric (**`00`**, **`01`**,...) or descriptive (**`bright`**, **`mellow`**,...)

thus

[3]A more neutral, less brassy soundfont should still be on the primary channel for any brass instrument.

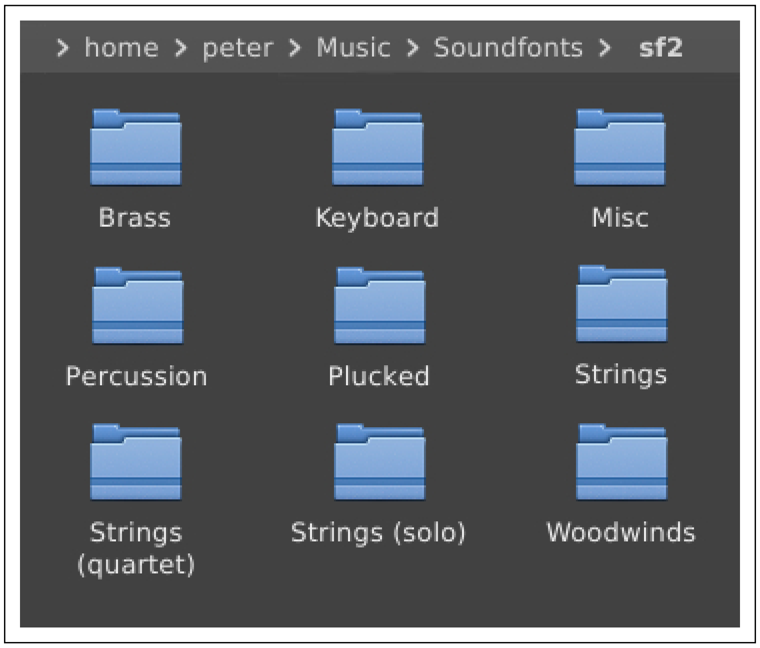


**Fig. 8.** *Soundfont library, top-level directory*

```
Winds-Clarinet-00.sf2       Winds-Clarinet-bright.sf2
                       or
Winds-Clarinet-01.sf2       Winds-Clarinet-mellow.sf2
```

A file should be kept detailing the qualities of the soundfonts, including any defects, both those that are correctable ("G♭5 – D6 very soft") and those that are not ("uses velocity layers").

## Useful tools

A number of fairly good quality soundfonts reside inside GM soundbanks. Adding them to a soundfont library requires extracting them, which can be done with a tool called **sf2splitter**.

Some soundfonts are compressed into .sfark and .sfpack files. Neither format gained much acceptance—zip works just as well—but there are enough files of both types on the Web to make **sfark** and **sfpack** essential.

**Sf2splitter**, **sfark**, and **sfpack** are freely available. There are no Linux versions, but they run well under **WINE**, the Linux Windows emulator.

# Setting up the orchestra

## Non-session-manager

The use of a session manager is not optional. In addition to loading the programs needed for midi orchestration, it keeps track of all the connections made in the Carla patchbay, where, in a completed setup, the number of virtual cables runs to well over a hundred.

Non-session-manager (NSM), as noted earlier, is much less polished than other session managers, but it can handle the extensive requirements of a virtual orchestra where the others cannot.

The online Non Session Manager User Manual covers usage adequately, but insufficient emphasis is placed on the significance of the **Abort** button. The complexity of a virtual orchestra can, and does, lead to periodic crashes in one or another of its components. Occasionally when this happens, it is not possible to re-launch the application without first shutting down NSM. If the **Close** button is used, NSM will save its current state before quitting, meaning that connections lost due to the crash will have vanished when the session is re-launched. **Abort** prevents NSM from trying to save the current state.

Because crashes are a genuine concern during setup of the virtual orchestra, it is advisable to make very frequent backups of the NSM session. Use of the **Duplicate** button is strongly discouraged for this purpose. Copy the directory holding the session at the command line instead, giving it a conventional ".bak" extension.

A session is created in NSM by clicking **New** and giving the session a name. Clients are added by clicking **Add Client to Session**. The virtual orchestra described herein needs the following clients:

- JACKPatch (described in the NSM User Manual)
- Carla
- MuseScore
- LinuxSampler
- Non-mixer
- JAMIN

NSM sessions typically begin with JACKPatch, but otherwise the order of loading doesn't matter. Except for LinuxSampler, clients are added simply by typing the program name in the **Enter program name** dialogue.

LinuxSampler, which in this paper uses the JSampler frontend called "Fantasia," requires entering the full path to the "Fantasia.jar" executable. Information about downloading, installing, and running JSampler is available at www.linuxsampler.org/jsampler/manual/html/jsampler.html.

NSM sessions save the configuration and state of JACKPatch, Non-mixer, Carla, and MuseScore whenever **Save** is clicked in the menubar, or when a session is closed.[1]

Sessions are opened by clicking on the name in the **Sessions** list. Note that upon opening a session, MuseScore itself will, in addition, ask whether to restore its previous session.

NSM does not save the JAMIN configuration active at the time of closing, which means the configuration must be saved from within the program and loaded manually at the start of each session. If the LinuxSampler backend is not running when a session starts, it, too, must have its desired configuration loaded manually.

## LinuxSampler

LinuxSampler holds the soundfonts used in a virtual orchestra. It is next in line to MuseScore in the audio chain: MuseScore sends its midi signals to LinuxSampler, which plays them with the appropriate soundfont.

LinuxSampler is a backend application, like a server. It has two frontends: **QSampler**, a Qt frontend, and **JSampler**, which is written in Java. Neither is very polished. QSampler is marginally more so, but is less stable and lacks some functionality. JSampler, on the other hand, gobbles system resources.

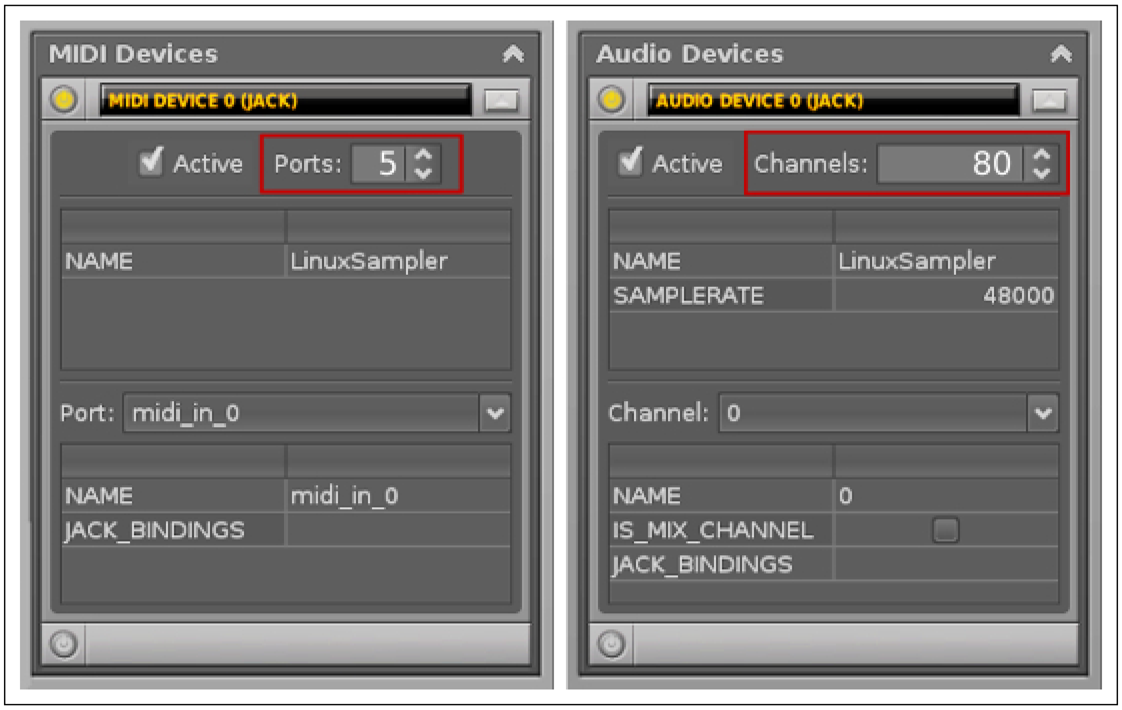
For the purposes of this paper, the JSampler "Fantasia" frontend is assumed. Further references to LinuxSampler should be taken to mean this frontend.

Of particular assistance for newcomers to LinuxSampler is the MuseScore tutorial *MuseScore with LinuxSampler*, beginning at the section Configuring LinuxSampler. Information prior to this section deals with setting up JACK using **qjackctl**. It may provide some insight, but can safely be ignored since

[1] The same may be saved individually by clicking **Save** beside their names.

Cadence and Carla are the preferred tools for configuring JACK and making connections.

Before loading and configuring soundfonts in LinuxSampler, MIDI and Audio devices must be created. The MuseScore tutorial cited above gives information on how to accomplish this. The MIDI device needs at least 5 ports. The Audio needs at least 80 channels ($5 \times 16$).[2] See Figure 9.



| | |
|---|---|
| Active | ✓ |
| Ports: | 5 |
| NAME | LinuxSampler |
| Port: | midi_in_0 |
| NAME | midi_in_0 |
| JACK_BINDINGS | |

| | |
|---|---|
| Active | ✓ |
| Channels: | 80 |
| NAME | LinuxSampler |
| SAMPLERATE | 48000 |
| Channel: | 0 |
| NAME | 0 |
| IS_MIX_CHANNEL | ☐ |
| JACK_BINDINGS | |

**Fig. 9.** *LinuxSampler MIDI and Audio devices*

It is a good idea to save the configuration immediately with a meaningful filename, e.g. sampler-orchestra. LinuxSampler configurations have the extension ".lscp." Saving is done by clicking the **Export current sampler configuration** button in the toolbar.

## Mapping instruments

Loading instruments into LinuxSampler is covered in the Loading instruments in LinuxSampler section of the MuseScore/LinuxSampler tutorial. The mapping file comes into play at this stage since it is essential to know which midi ports and channels carry which instruments, and which audio-out channels should be assigned to them. The top of the example mapping file will assist in demonstrating:

[2]The number of ports and channels may be greater.

| midi_in | audio out | Instrument name |
|---|---|---|
| *mscore-midi-1 → LinuxSampler midi_in_0* | | |
| 1 | 1 | Piccolo |
| 2 | 2 | Piccolo (alt.) |
| 3 | 3 | Flute |
| 4 | 4 | Flutes (sect.) |
| 5 | 5 | Flute (alt.) |
| 6 | 6 | Oboe |
| 7 | 7 | Oboes (sect.) |
| 8 | 8 | Oboe (alt.) |
| ... | ... | ... |

The flute department, including piccolo, needs a total of five soundfonts. Since it is usually best to put *alt.* instruments at the bottom of LinuxSampler where they are easy to find, the arrangement of instruments begins with a piccolo soundfont and two identical flute soundfonts, the first for solo flute, the second for chorused flutes. The oboes are managed similarly. Figure 10 shows the instruments loaded into LinuxSampler.[3]

Clicking on the Options arrow expands the instrument slots to allow setting port and channel numbers. The arrows in the highlighted regions (Figure 11) reveal dropdowns. The leftmost one tells LinuxSampler which *LinuxSampler_midi_in* port to listen on. Beside it is the **midi_in** channel number. At the far right, the **CR** (Channel Routing) button calls up a dialogue where the **audio out** is set.

Consulting the mapping file, we see the piccolo from MuseScore will be coming into LinuxSampler on the *midi_in_0* port. The port is set by selecting an entry from the **midi...** dropdown (Figure 12). The channel in the **midi_in** column of the mapping file is "1," and is set in LinuxSampler by selecting an entry from the **Channel** dropdown (see Figure 13).

The **audio out** channel is also "1." Clicking on the **CR** button, highlighted at the right in Figure 11, calls up a dialogue with two columns: **Audio In**

[3]The instrument names are the names embedded in the soundfonts themselves and cannot be changed in LinuxSampler.

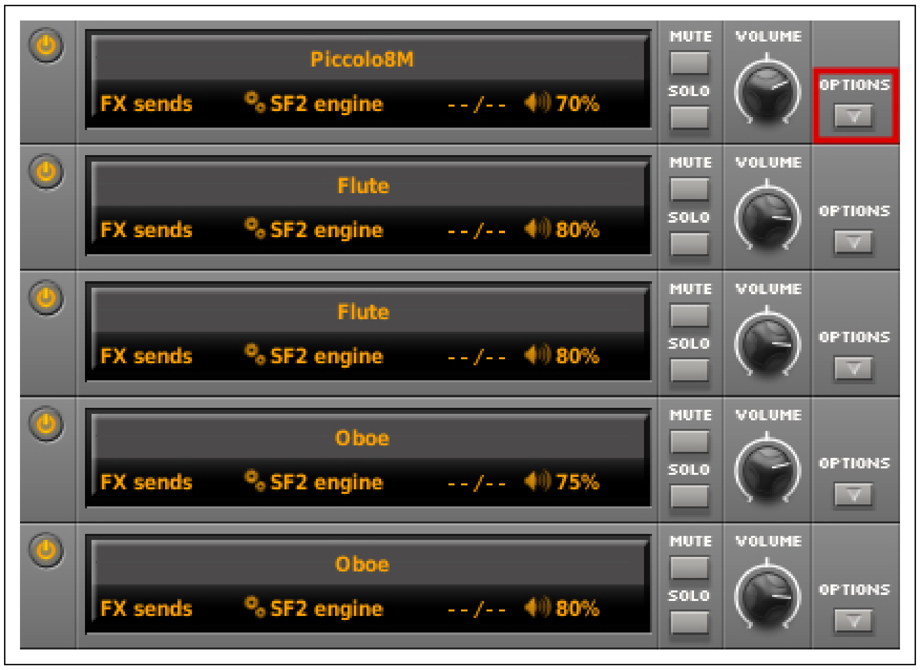


**Fig. 10.** *LinuxSampler instrument slots*

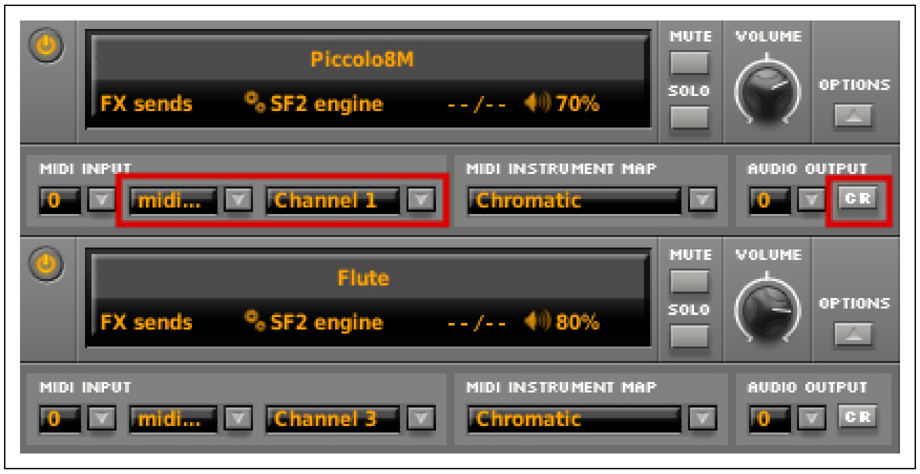


**Fig. 11.** *Slots expanded to show options*

and **Audio Out**. Only Audio Out requires setting. Both entries in the column should be set to the same channel number, which downmixes stereo soundfonts. to mono.[4]

The next instrument in the mapping file mapping file is “Piccolo (alt.).” Its **midi_in** and **audio out** are “2,” but it is to be skipped for now.

[4]Note that the cursor must be placed over the appropriate entry, as shown in Figure 14, before clicking reveals the dropdown selection list.

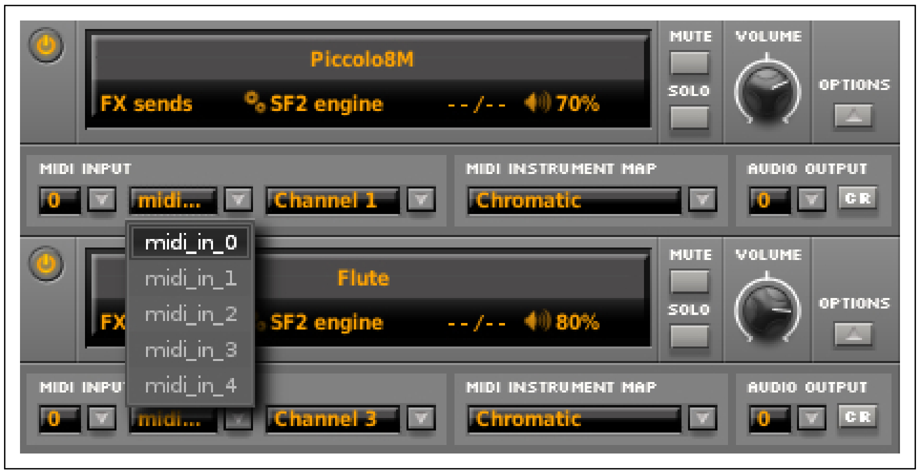


**Fig. 12.** *Midi port dropdown*

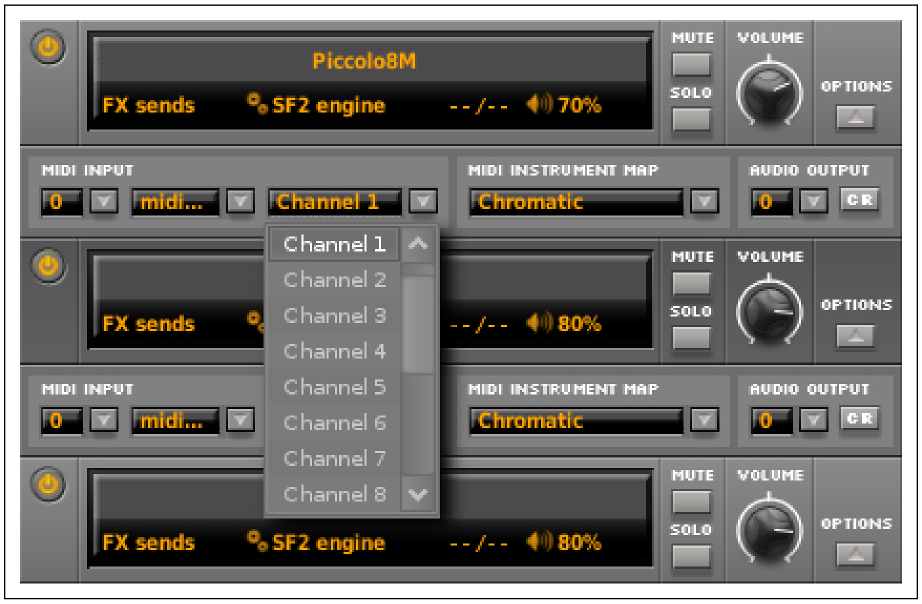


**Fig. 13.** *Midi channel dropdown*

Following “Piccolo (alt.)” is “Flute,” which will be used any time the flutes are not playing in unison. The **midi_in** and **audio out** channels are “3,” and are set the same way as the piccolo. The midi and audio channels for “Flutes (sect.)”, which will be chorused further down the audio chain, are set to “4.”

With a mapping file, one has merely to continue setting ports and channels according to what it says. Without one, confusion over midi port numbers, midi channel numbers, and audio-outs is certain to lead to errors.

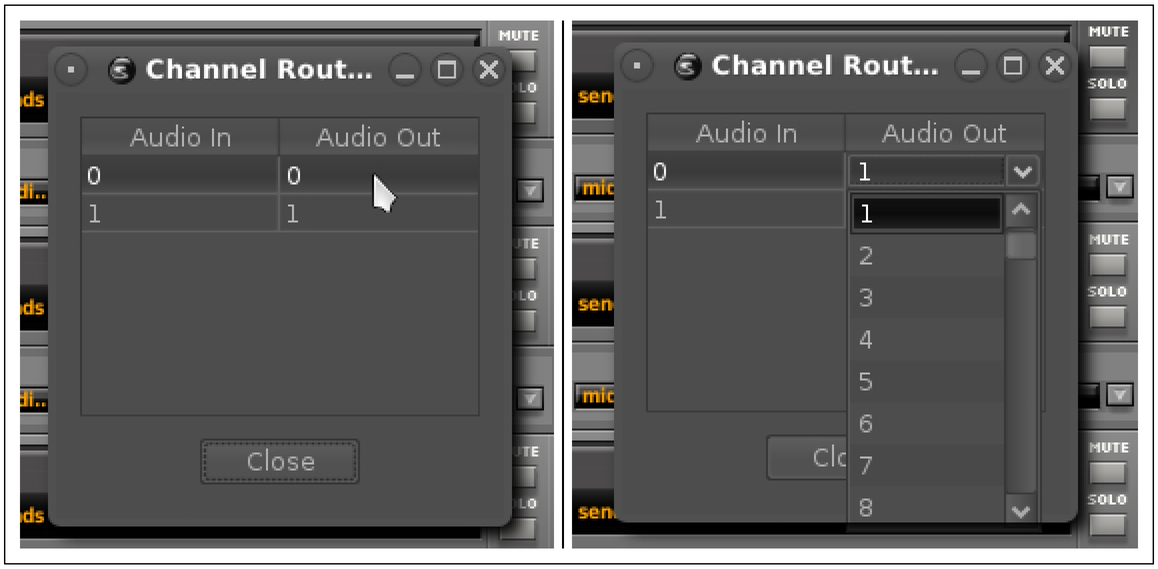


**Fig. 14.** *Audio channel routing selector and dropdown*

### *Percussion channels*

Regardless of the port number, channel 10 of the sixteen **midi_in** channels available to the port is reserved for non-pitched percussion:

| midi_in | audio out | Instrument name |
|---|---|---|
| *mscore-midi-1 → LinuxSampler **midi_in_0*** | | |
| 9 | 9 | *(instrument name)* |
| **10** | **10** | ---reserved--- (Snare Drum) |
| 11 | 11 | *(instrument name)* |
| ... | ... | ... |
| *mscore-midi-2 → LinuxSampler **midi_in_1*** | | |
| 9 | 25 | *(instrument name)* |
| **10** | **26** | ---reserved--- (Bass Drum) |
| 11 | 27 | *(instrument name)* |
| ... | ... | ... |

The suggested LinuxSampler setup (Figure 9) uses five ports, limiting the number of percussion channels to five. Further non-pitched percussion may be added by increasing the number of ports and the corresponding number of channels, which must be increased by sixteen for every additional port.[5]

The important thing to remember is that non-pitched percussion cannot go on any other channel than 10, and must be spread out across the available ports. For convenience, the soundfonts may be grouped together in Linux-Sampler; the order of loading is unimportant. What matters is setting the correct port and channel numbers.

### Setting levels

Instrument levels are a crucial part of midi orchestration. The issue must be dealt with methodically given the plethora of choices where levels can be set. Non-mixer, which stands next to LinuxSampler in the audio chain, is the logical place to do this, but to be used effectively, a baseline needs to be set on the signal going into Non-mixer.

It is generally best to set the overall gain of LinuxSampler quite low, between 15% and 25%. Soundfonts should initially have their gain set to around 80%, which leaves a 20% margin for increase. MuseScore mixer settings should be left at their default. The actual balancing and fine-tuning of instrument levels is covered in Levels, panning, and reverb.

## Non-mixer

If MuseScore and LinuxSampler are the musical heart of midi orchestration, Non-mixer is the heart of audio processing.

The online Non-mixer manual provides a guide to usage.

### Instrument strips

Setting up Non-mixer requires a number of steps, beginning with adding a set of "strips" for every instrument in the virtual orchestra, including *sect.* strips for woodwinds and brass. The purpose of this first set of strips is fine-tuning the levels of individual instruments. Strips for *alt.* channels are not needed for winds and brass—*alt.* soundfonts are typically fed into the same strip as the solo instrument—but they are needed for the five departments of the string section.

[5]As will be seen when connecting the setup in **Carla**, this increases the depth of the already very long LinuxSampler module, the top portion of which may have to be moved off the patchbay canvas in order to find the additional **audio out** channels receiving input from their respective channel 10s.

Figure 15 shows a strip that has just been added. Setting the **Instrument name** and **Orchestra section** (group dsp) are indispensable for the Carla patchbay. The number of channels should be left at "1."[6] Figure 16 shows how the Flute and Vlns. 2 strips look after setting the names.[7]

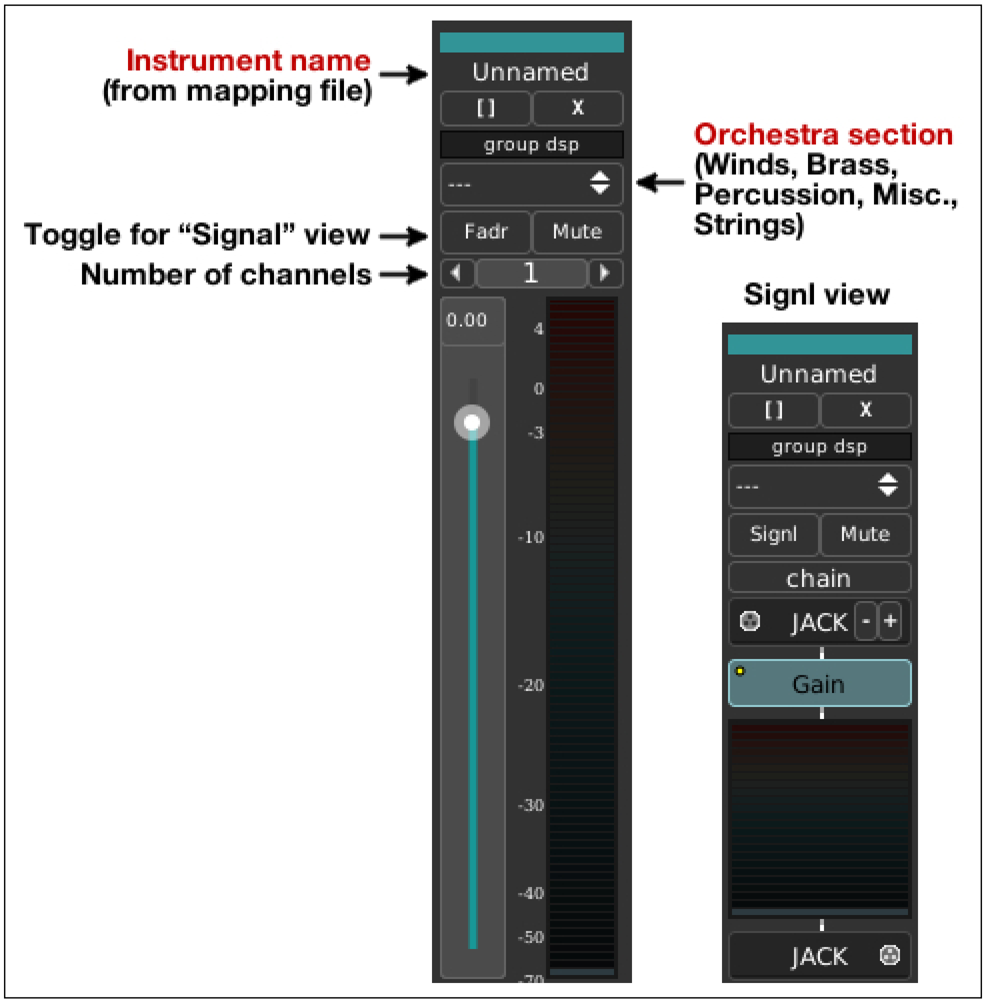


**Fig. 15.** *Non-mixer strip*

## Department strips

The second set of strips groups instruments by department (e.g. Flutes, which comprises Flutes I., II., and piccolo). The strips need two plugins to manage panning and reverb, which, as can be inferred, is done by department, not instrument. Nothing is gained by panning/reverbing every instrument in a department separately.

[6]Note that the number of channels and is only visible in the **Fadr** view, and the **Gain** module is only visible in **Signl** view.
[7]The expected *tremolo* channel after *pizz.* in Fig. 16 has been replaced by *alt. 1*.

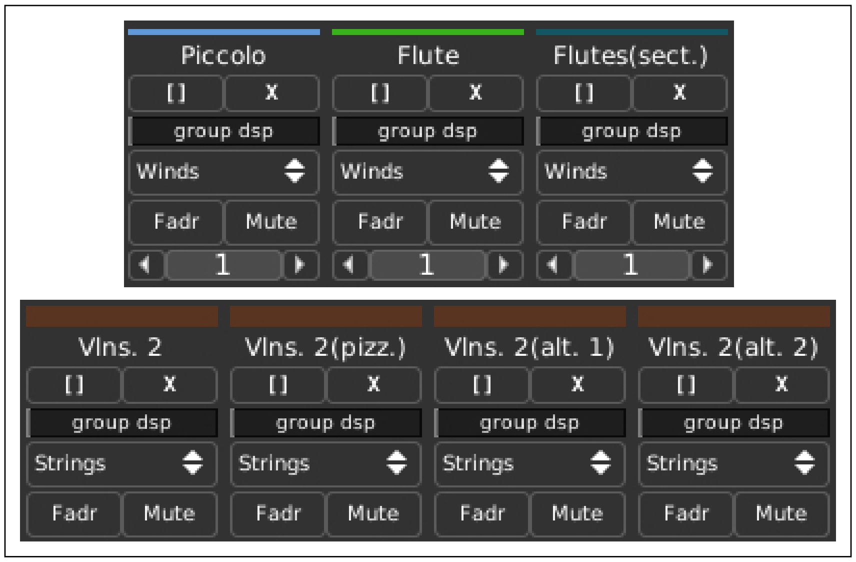


**Fig. 16.** *Non-mixer instrument strips for Flutes and Violins 2*

Like the strips for individual instruments, department strips need to be named and given a **group dsp** identification. A useful convention is to name them simply Flutes, Oboes, Clarinets, etc., and to set the group dsp to the same name. A certain amount of creativity may be needed to avoid conflicts with already extant instrument and group dsp names.

The two plugins needed on department strips are **Aux** and an **ambisonic panner**. The use of Aux is covered in Levels, panning, and reverb. Toggling to Signl view and right clicking on **Gain** reveals a context menu from which Aux is selected directly. **Plugin**, in the same menu, calls up a menu from which the panner can be chosen. It should be **AMB order 1,1 mono panner**. Figure 17 shows the department strips for woodwinds.

For consistency, it is a good idea to create individual department strips with Aux and panner plugins for the timpani and each of the Miscellaneous instruments as well.[8]

If equalization is being used to simulate the different timbres of the string departments, an equalization plugin is also required on each string department strip, as seen in Figure 18. Several equalizers are available from the Plugin menu. The ten-band **C*Eq10** equalizer is a good choice. The equalizer must be inserted before Aux.

[8]The Aux and panner plugins could be applied to the instrument strip instead.

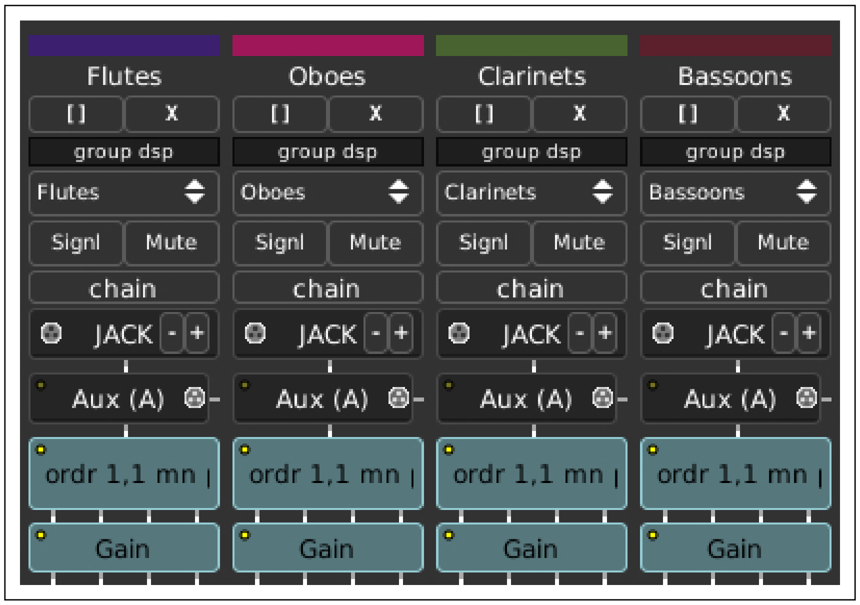


**Fig. 17.** *Non-mixer department strips, woodwinds section*

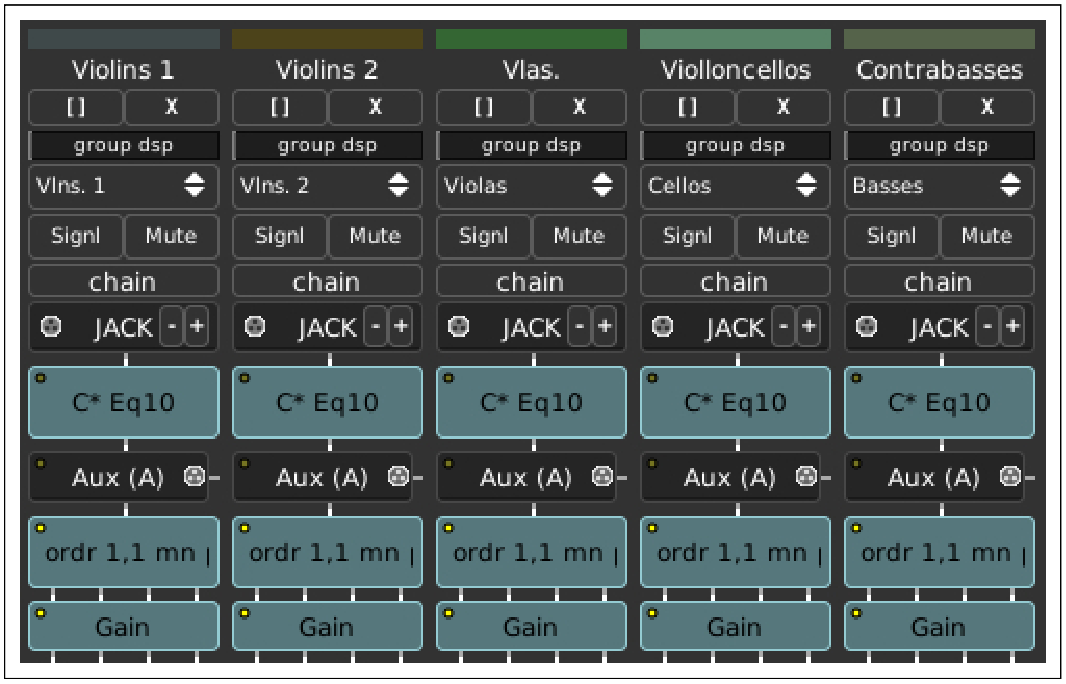


**Fig. 18.** *String departments with EQ plugin*

## Section strips

After the instrument and department strips come the section strips. These are used to balance the levels of entire sections, for example to make the overall woodwinds more, or less, prominent.

Strips are needed for Winds, Brass, Percussion, Harps, Strings, and any individual instruments—usually from Miscellaneous—that may benefit from being treated as a section, e.g. the piano. None requires a plugin. The **group dsp** should be set to "Sections" for all, and the number of channels must be set to "4" (Figure 19).

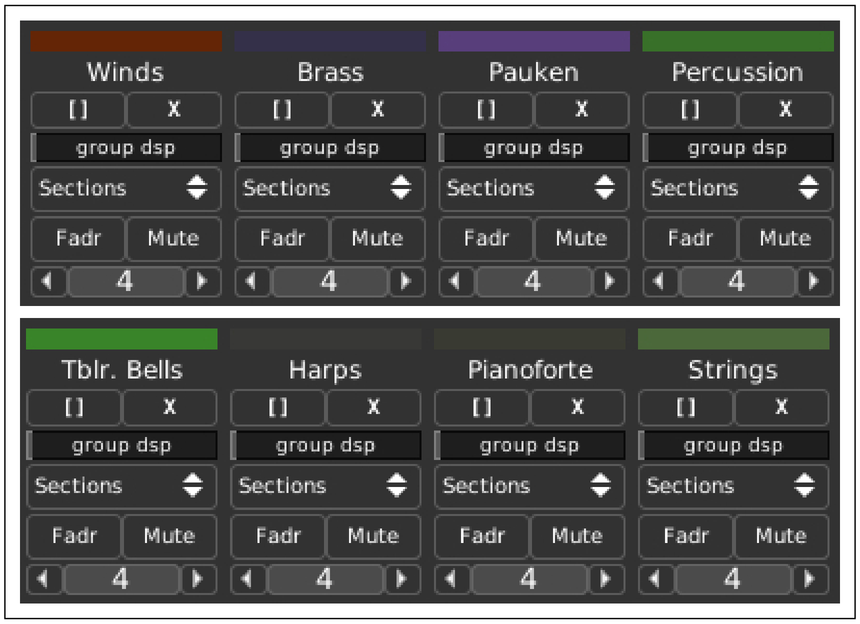


**Fig. 19.** *Non-mixer* "Sections" *strips*

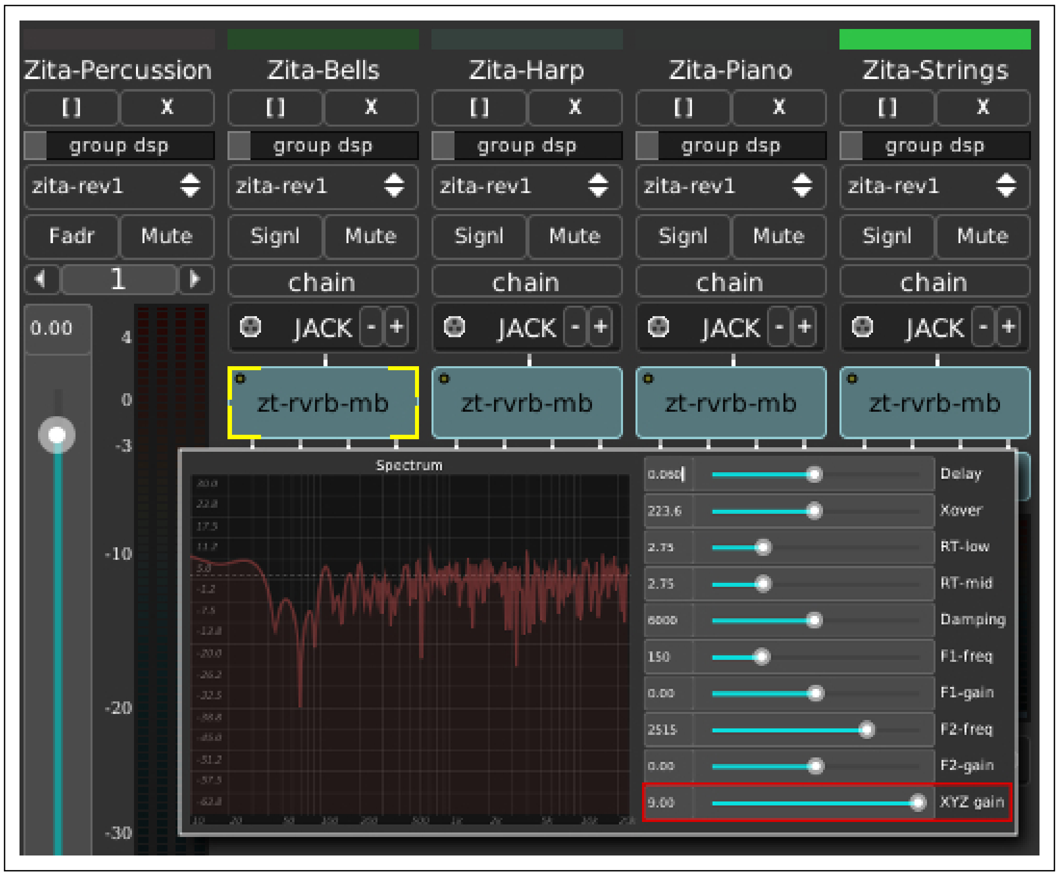
## Reverb strips

Non-mixer offers the possibility of both algorithmic and convolution reverb. The setup described thus far is for implementing algorithmic reverb. The reverberator plugin to be used is Fons Andriaensen's **zita-reverb-amb**. It goes on the same number of strips as the sections. For convenience, the strips should be named similarly, with the prefix "Zita-," thus Zita-Winds, Zita-Brass, etc. The **group dsp** should be "zita-rev1," and the number of channels set to "1" (Figure 20, far left strip.)

It has not yet been necessary to adjust the controls of any group of strips, but zita-rev1 must have its **XYZ-gain** set to maximum. Clicking on the plugin calls up the controls. The XYZ slider is at the bottom (Figure 20).

## Reverb Master and Master Out strips

The final two strips to be added to Non-mixer are the **Reverb** master and **Master Out**.



| | |
|---|---|
| 0.060 | Delay |
| 223.6 | Xover |
| 2.75 | RT-low |
| 2.75 | RT-mid |
| 6000 | Damping |
| 150 | F1-freq |
| 0.00 | F1-gain |
| 2515 | F2-freq |
| 0.00 | F2-gain |
| 9.00 | XYZ gain |

**Fig. 20.** *Zita-rev1 controls, XYZ Gain*

The **Reverb** master controls the amount of overall reverb, effectively establishing the distance of the orchestra from the listener. **Master Out** controls the volume of the output from Non-mixer to the system speakers.

The Reverb strip needs four channels and doesn't require a group dsp name. The Master Out strip also needs four channels, and the ambisonic decoder plugin, **Ambisonic Decoder (B-Format to Stereo)** (Figure 21). A group dsp for Master Out is optional.

## Carla

The majority of work setting up a virtual orchestra takes place in Carla, where virtual cables are used to connect the components. The process is tedious, but if care has been taken with the mapping file and loading instruments into LinuxSampler, the tedium is not compounded by confusion.

Partial documentation for Carla can be found at the KXStudio site. The documentation does not cover the patchbay, but its usage is self-explanatory.

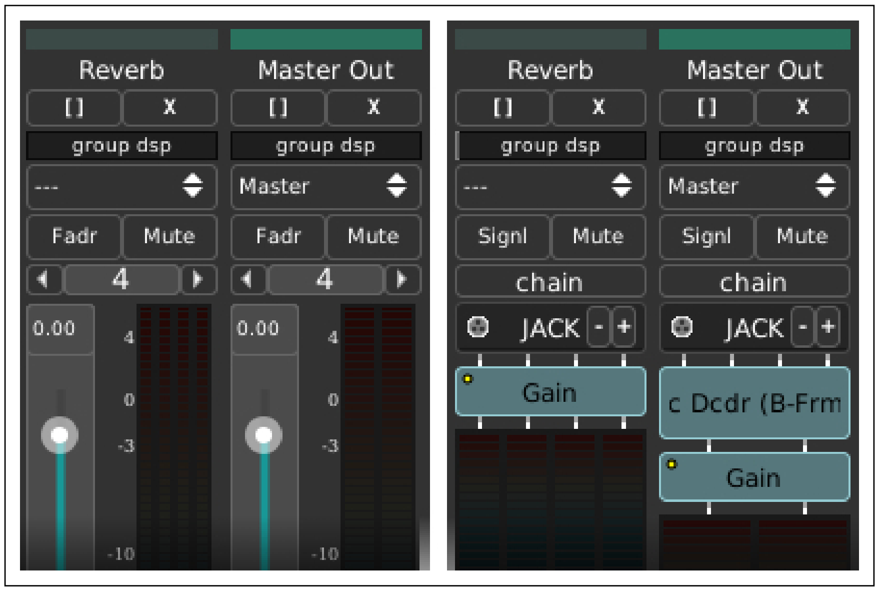


**Fig. 21.** *Reverb and Master Out strips,* **Fadr** *and* **Signl** *views*

## Plugin rack — add Calf Multi Chorus

Besides providing a JACK patchbay, Carla also acts as a plugin host, which is useful because Non-mixer does not support Calf plugins, and Calf's Multi Chorus is superior to the chorus plugins available in Non-mixer.[9]

Calf Multi Chorus is loaded by clicking **Add Plugin** and typing "Calf Multi Chorus" in the search bar.[10] Four instances are needed, one for each woodwind department. Right-clicking on the plugins brings up a context menu where they can be renamed to Chorus-Flutes, Chorus-Oboes, Chorus-Clarinets, and Chorus-Bassoons.

Controls are called up by clicking on **GUI** or **Edit** at the left of the plugin. **GUI** brings up a fancy set of knob adjusters. **Edit** brings up a cleaner and more useful set of slider controls.

There is no magic formula for chorusing winds. Some experimentation is necessary to find the right settings for each department. It is generally best to leave the settings at their default and adjust them when the instrument levels are being set.

[9]Calf plugins are packaged for most major Linux distributions.
[10]The **LV2** and **Effects** checkboxes must be active for Calf Multi Chorus to show up.

A setting that *should* be changed immediately is the number of voices in the **Parameters (1)** tab of the Edit controls. The default is "4," and must be reduced to "2" or "3" depending on the size of the department. The examples throughout this paper are based on an orchestra with winds in pairs, which presupposes a setting of "2."

## The patchbay

Assuming a running NSM session with all the virtual instruments loaded in LinuxSampler, all the strips added to Non-mixer, and the template score open in MuseScore, Carla's patchbay at first appears chaotic, with modules scattered over the whole canvas, often overlapping or occluding each other. It is impossible to continue until the mess is organized.

It is best to begin by locating and moving the MuseScore module (**mscore**) to the top left of the canvas and placing LinuxSampler beside it to the right (Figure 5).

The orchestra "Section" modules are then placed to the right of LinuxSampler, top-aligned, in score order: Winds, Brass, Percussion, Miscellaneous, Strings (Figure 22). Orchestra section modules are the ones that group departments together (Flute+Piccolo=flute section, Oboe+English Horn=oboe section, etc.).

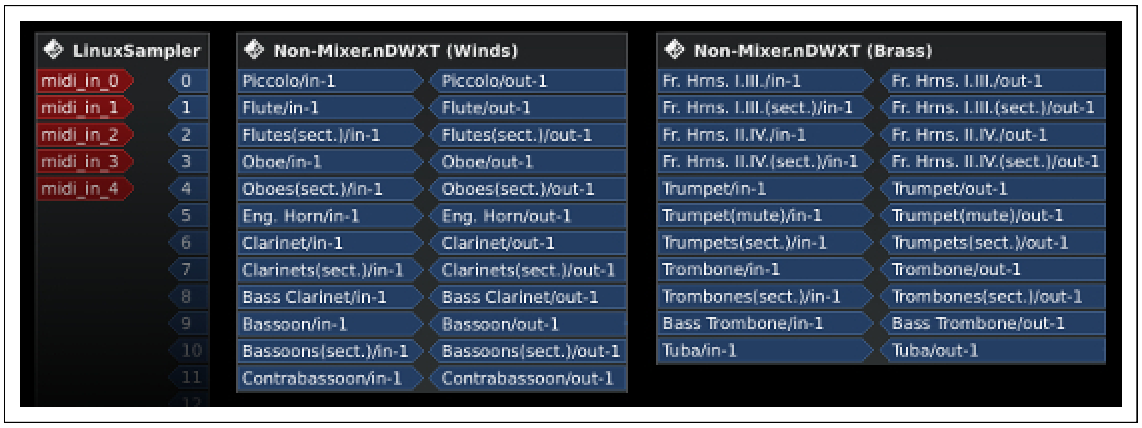


**Fig. 22.** *LinuxSampler, Winds, and Brass modules*

Underneath each orchestra section are positioned the appropriate department modules (second set of Non-mixer strips), e.g. under **Winds** go *Flutes*, *Oboes*, etc., under **Brass** go *French Horns*, *Trumpets*, etc. The Carla module holding the chorus plugins should be tucked under the bassoons.

To the right of everything go the modules **Sections** and **zita-rev1**, side-by-side, top-aligned. The remainder—**Reverb**, **Master**, **jamin**, **system**, **alsa2jack**—are small and may be stacked under the zita-rev1 module.

An example of the complete, organized patchbay is not possible because the canvas is too large to be reduced legibly. The schematic in Figure 23 may be of assistance in visualizing.

## Wiring the orchestra together

At this stage, the components of the virtual orchestra have been set up but are not yet connected. With over a hundred connections to be made, an understanding of the audio processing chain and how it simulates the experience of live music in a real hall is essential.

### The virtual hall

Midi realism depends as much on creating a sense of space as it does on having good soundfonts. For the Linux orchestrator working only with freely-available soundfonts, it could be argued that spatial realism is even more important. Little can be done to improve less-than-ideal virtual instruments, but a multitude of sins can be mitigated by positioning them correctly.

In a real concert hall, the orchestra is arranged in an arc around the conductor. At the front are the strings; typically, from left to right: first violins, second violins, violas, celli, and contrabasses. Behind them, grouped compactly centre stage, are the woodwinds. Behind them are the brass, usually in a single row. At the very back are the timpani and percussion sections. Miscellaneous instruments (harp, piano, celesta, etc.) are generally placed behind the strings on the left. Figure 24 shows a typical seating plan.

The arrangement is pleasing to the ears because, in addition to achieving a good blend without sacrificing clarity in the departments, it also possesses a spaciousness that contributes to the magic of the music.

The same acoustical principles apply to midi orchestration as to live performance. When virtual instruments are panned to positions approximating those of a live orchestra, they blend idiomatically while retaining their individuality. The bassoons, for example, are close enough to the celli and contrabasses to reinforce bass lines, but far enough away from the flutes that they retain their characteristic timbre in passages for wind choir.

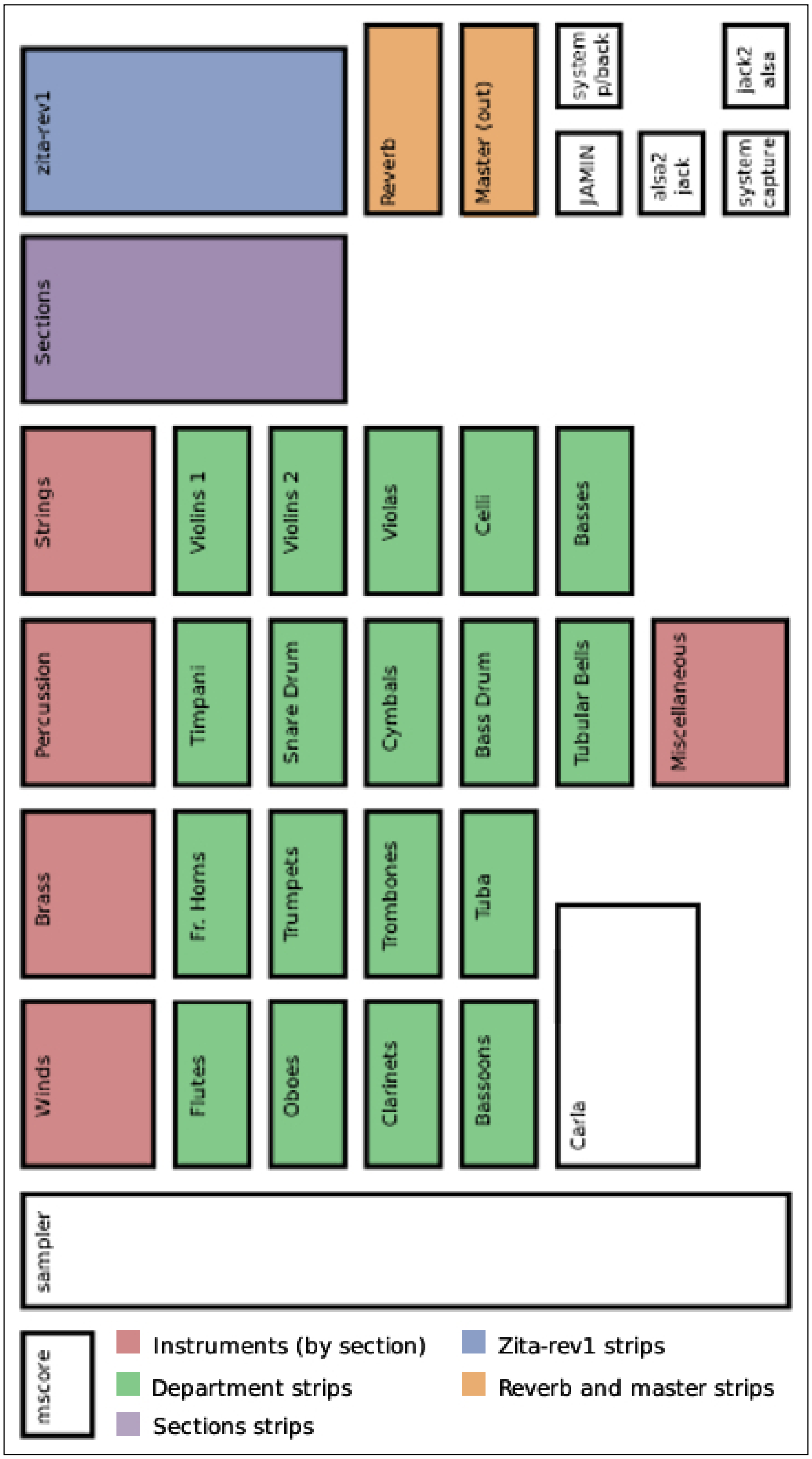


**Fig. 23.** *Carla patchbay layout*

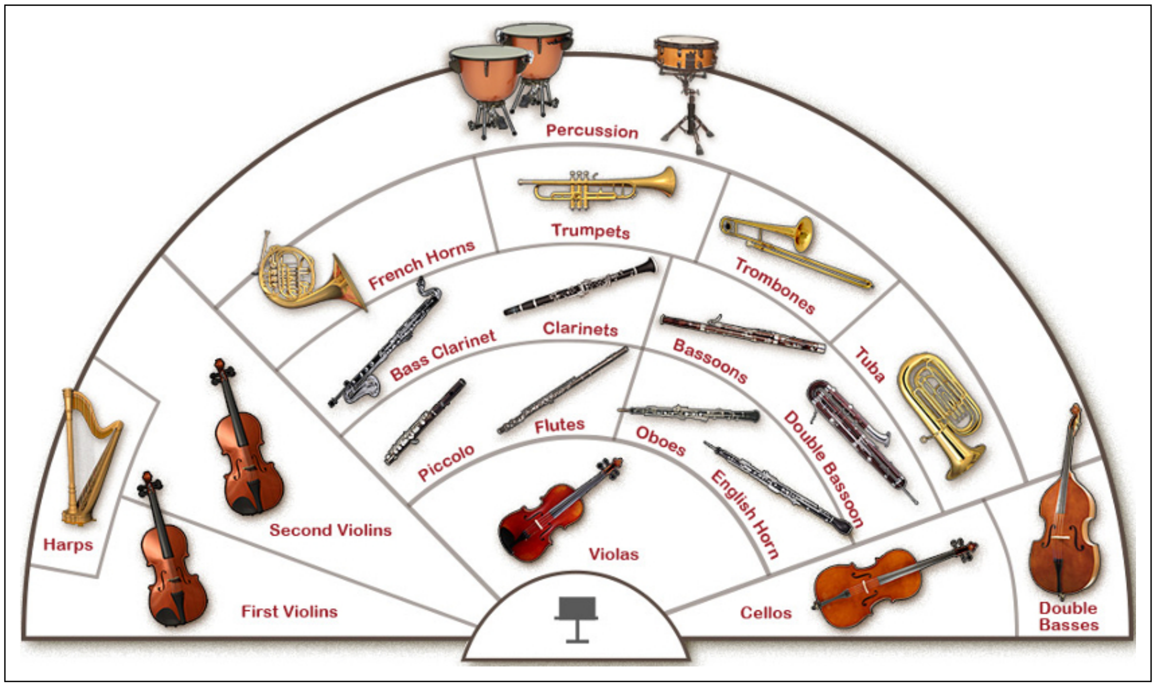


**Fig. 24.** *Typical arrangement of orchestra sections*

### *Z-axis panning*

Left and right panning is only one half of instrument positioning because an orchestra fills the stage from front to back as well.

The perception of distance in a hall, real or virtual, is a result of reverb. The farther away a sound is, the greater the echo that reaches the ears. Conversely, the closer the sound, the less the echo. The perception of nearness and distance is not related to the loudness of a sound, but to the amplitude and duration of its echoes.

In midi orchestration, Z-axis panning—placing instruments nearer and farther away—is trickier than Y-axis (left and right). The principle is simple—more reverb means farther away, less means closer—however the implementation is less straightforward. Reverb for a department or section can only be increased so far before the instruments sound as if they're playing in another room. Equally, too little reverb leads to violins playing an inch from the listener's nose. It is therefore necessary to have control over the dry signal, independent of the reverb, to compensate. Realistic Z-axis panning requires that the two be carefully balanced.

The department strips in Non-mixer provide everything needed for positioning the departments of the virtual orchestra left to right, front to back.

The ambisonic panner controls left and right panning, the Aux plugin controls the amount of reverb signal, and the Gain plugin sets the level for the dry signal.

Using a single flute channel from LinuxSampler as an example, the flowchart in Figure 23 shows how the signal is processed.

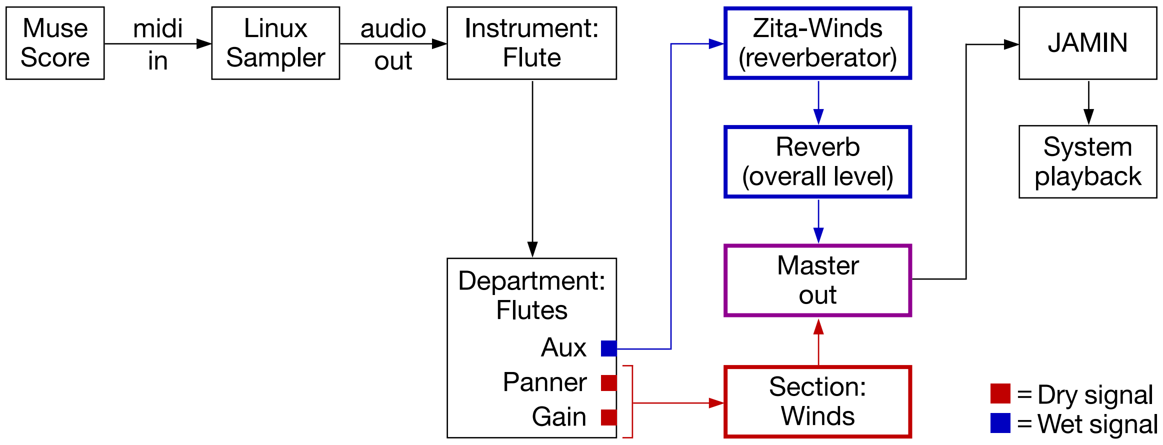


**Fig. 23.** *Signal processing flowchart*

The midi signal from MuseScore playback is fed into the sampler, which produces an audio signal—in this case, the sound of a naked flute. The audio signal goes directly to the Flute instrument strip in Non-mixer, where its volume is adjusted.

The signal is then sent to the Flutes department strip, where the signal is cloned.[11] **Aux** takes the signal and feeds it into the Winds reverberator; the amount of reverb for Flutes is therefore controlled by adjusting the **Aux** level. The wet signal further goes to the master **Reverb** strip that controls the overall reverb of the virtual hall.

**Gain** passes the unmodified (dry) signal into the Winds section strip, subject to left and right placement via the panner. The Flutes dry signal is therefore raised and lowered by adjusting the Flutes department Gain.

Finally, the wet and dry signals are mixed together in the **Master out** strip and sent to JAMIN for mastering.

[11]In a completed setup, the Flutes department strip has signal coming in from the piccolo and all the flute channels (normal, sect., alt.), these together forming the complete flute department.

The use of the department strips' Aux and Gain plugins is covered in Levels, panning, and reverb.

## Connections 1: MuseScore to LinuxSampler

Connecting MuseScore to LinuxSampler is straightforward. Each of MuseScore's **mscore-mid-*n*** ports is connected to the corresponding LinuxSampler **midi_in_n**, as seen in Figure 5.

> **Note:** Whenever MuseScore is launched, it automatically connects itself to **system_playback** in the patchbay and must be disconnected at the start of every session unless the MuseScore synthesizer is cleared of all soundbanks and soundfonts.

## Connections 2: LinuxSampler to Non-mixer

If not already grasped, the need for a port/channel mapping file becomes evident at this step. Every one of the very long list of numbers at the right of the LinuxSampler module represents an audio-out that must be connected to the correct instrument strip in Non-mixer. Without perfect recall, it is impossible to know which number corresponds to which instrument.

The first instrument in the mapping file is the primary piccolo on audio-out "**1**," so audio-out "**1**" of Carla's LinuxSampler module is connected to **Piccolo/in-1** of the **Winds** module. The piccolo soundfont is now connected to Non-mixer's Piccolo strip.

The next instrument is an *alt.* piccolo, on LinuxSampler audio-out "**2**." Because *alt.* instruments generally go on the same Non-mixer strip as the primary, audio-out "**2**" is also connected to **Piccolo/in-1**. Figure 25 demonstrates.

After piccolos in the mapping file come "Flute," "Flutes *(sect.)*," and "Flute *(alt.)*" on channels 3, 4, and 5. "**3**" and "**5**" (normal and *alt.*) are connected to **Flute/in-1**, as with the piccolo (Figure 26).

"Flute (*sect.*)" takes a detour to the Carla module where chorusing for *sect.* woodwinds takes place. LinuxSampler audio-out "**4**" must be connected to both input ports (**Chorus-Flutes:in L** and **Chorus-Flutes:in R** in the Carla module), and both output ports (**Chorus-Flutes:out L** and **Chorus-Flutes:out R**) must be connected to **Flutes(sect.)/in-1** (Figure 27).

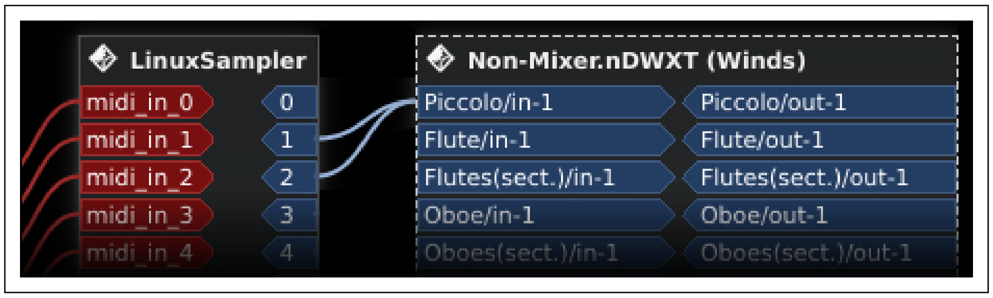


**Fig. 25.** *Piccolos connected to Non-mixer*

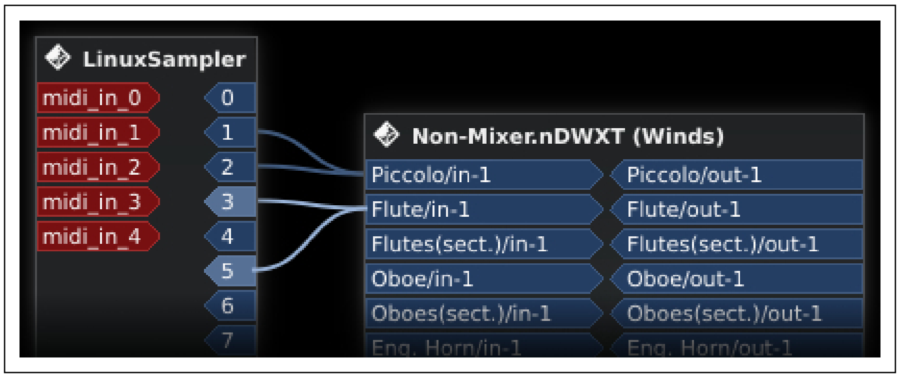


**Fig. 26.** *Flute (normal and* alt.*) connected to Non-mixer*

Connections for the remainder of the woodwinds follow the Flute model, with the normal and *alt.* audio-outs from LinuxSampler going into a single Non-mixer channel, and the *sect.* audio-outs being routed through the chorus plugin before being sent to the appropriate **...(sect.)/in-1** channel of Non-mixer.

Brass, timpani, percussion, and miscellaneous instruments are connected similarly to the winds, except that none require chorus plugins.

Strings differ from winds and brass in that each audio-out from Linux-Sampler is connected to a unique Non-mixer strip. In other words, *alt.* string department soundfonts do not share a Non-mixer channel with the primary department soundfont. Given the dissimilarities in timbre, articulation, and volume likely to be encountered in strings soundfonts, this allows for setting their levels independently.

Assuming a mapping file that places 1st Violins on LinuxSampler audio-out "**53**", with three additional channels afterwards (*pizz.*, *alt.1*, *alt.2*), the connections would look as they do in Figure 28.

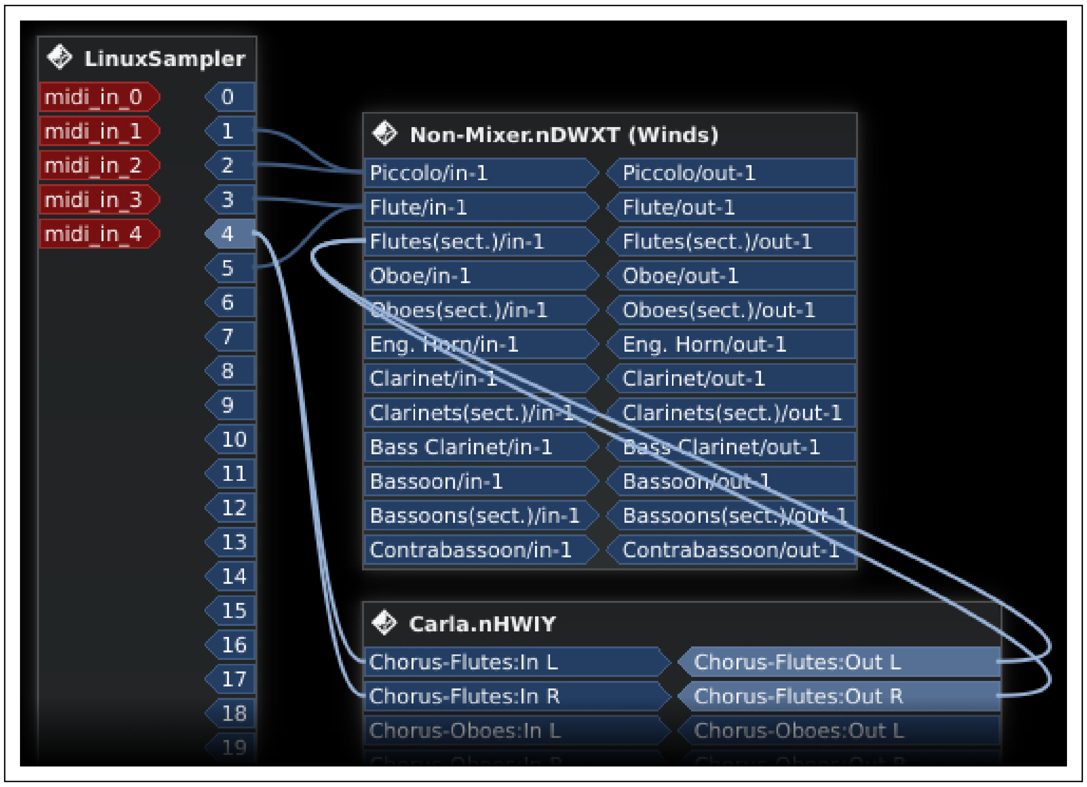


**Fig. 27.** *Flute* “sect.” *connections*

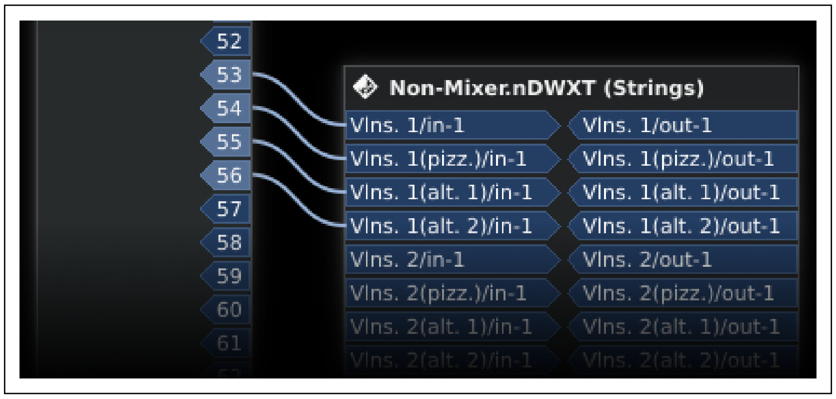


**Fig. 28.** *1st Violins connections*

## Connections 3:
## Instrument strips to department strips

Once LinuxSampler audio-outs are connected to the Instrument strips in Non-mixer, the instruments need to be connected to their respective Department strips, where panning and z-axis placement are controlled. Using Flutes as an example, Figure 29 shows how this is done. In an orchestra, the flute department is comprised of piccolos and flutes, so all of the piccolo and flute outputs on the right of the “Winds” module are connected to **Flutes/in-1** of the “Flutes” module, effectively combining them into a single “department” entity.

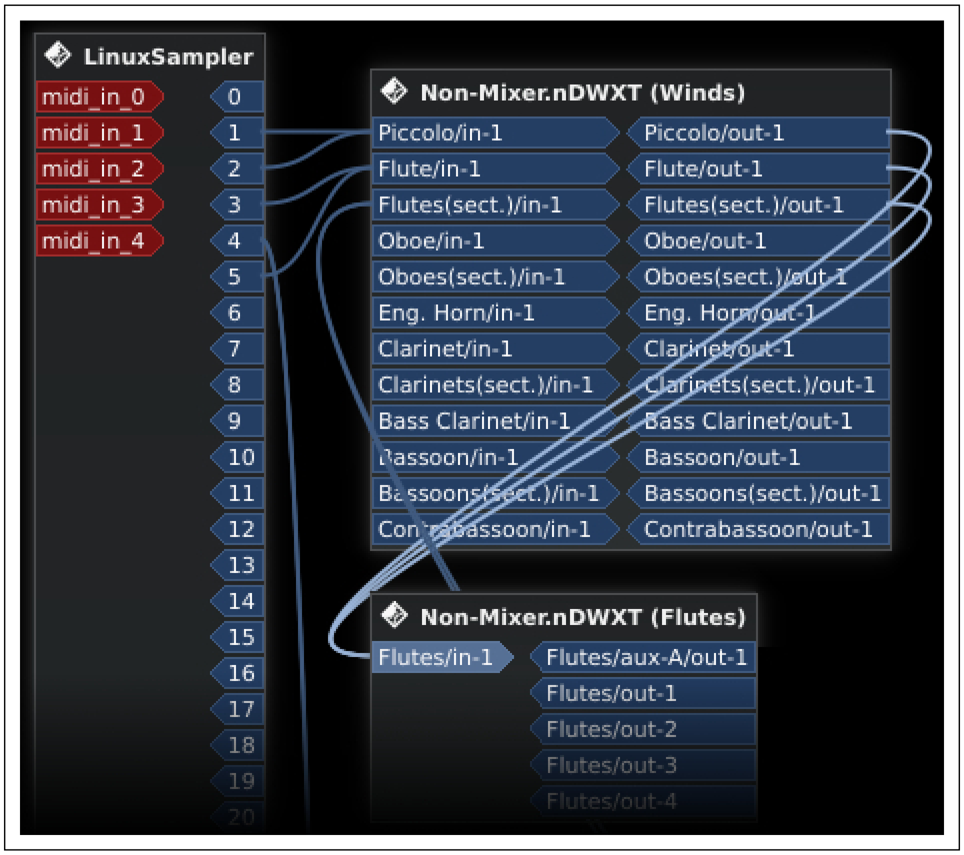


**Fig. 29.** *Flutes department connections*

The same procedure is applied throughout the patchbay, with the output of all the instruments that form any one department being fed into the single input channel of the appropriate department module.

## Connections 4: Departments to Sections & zita-rev1 strips

Output from the department modules comprises two parts: **aux-A/out-1**, and four ambisonic outputs labelled **out-1** to **out-4**. **Aux-A/out-1** carries the signal that goes to the reverberator. Reverb is applied by section, with each department determining the amount of signal the reverb receives via the **Aux** gain, so each department’s **aux-A/out-1** is connected to the appropriate **Zita-<section>** input of the “zita-rev1” module. Flutes, Oboes, Clarinets, and Bassoons, for example, all go to the **Zita-Winds** input.

The dry signal from each department's ambisonic panner is carried on **out-1**...**out-4**, which correspond to the **<section>/in-1** inputs of the "Sections" module. All four inputs and outputs must be connected. Figure 30 shows the Flutes department connected to the "zita-rev1" and "Sections" modules.

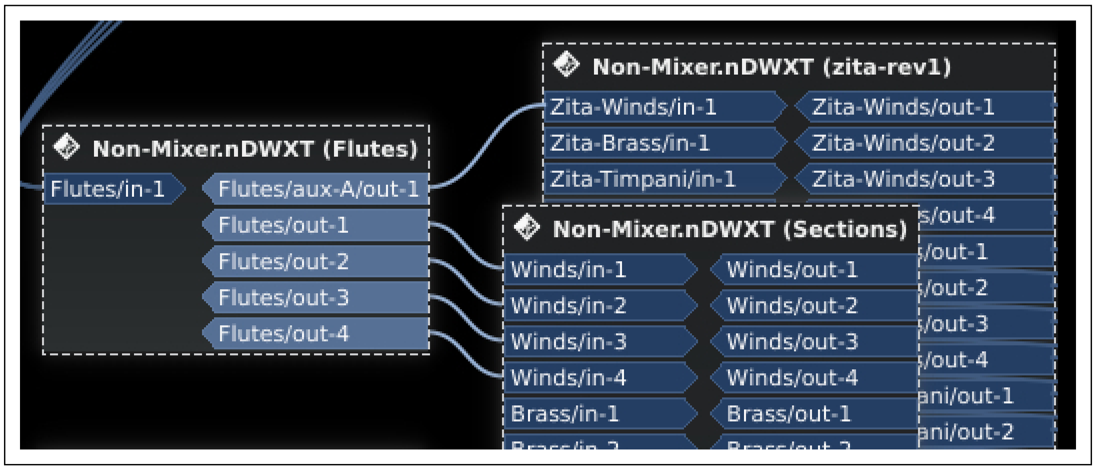


**Fig. 30.** *Flutes department reverb and dry connections*

Connections are made similarly for every department in the patchbay.

## Connections 5: Sections and zita-rev1 strips to masters

Both the dry and wet (reverb) signals need to be connected to the "Master" module. The dry input goes directly to "Master" from the "Sections" module, with each of the **out-1**...**out-4** outputs connected to the corresponding **in-1**...**in-4** inputs of the "Master" module.

The wet signal from the "zita-rev1" module must first be connected to the "Reverb" module before going into "Master". Each of the four outputs for each section in "zita-rev1" is connected to the corresponding "Reverb" input, with the corresponding "Reverb" outputs going to the corresponding "Master" inputs. Figure 31 shows the connection path for the Winds section.

## Connections 6: Master to jamin→system(playback)

The last step in wiring the orchestra together is connecting the "Master" outputs to **jamin**'s inputs, and **jamin**'s outputs to **system**(playback), as shown in Figure 32.

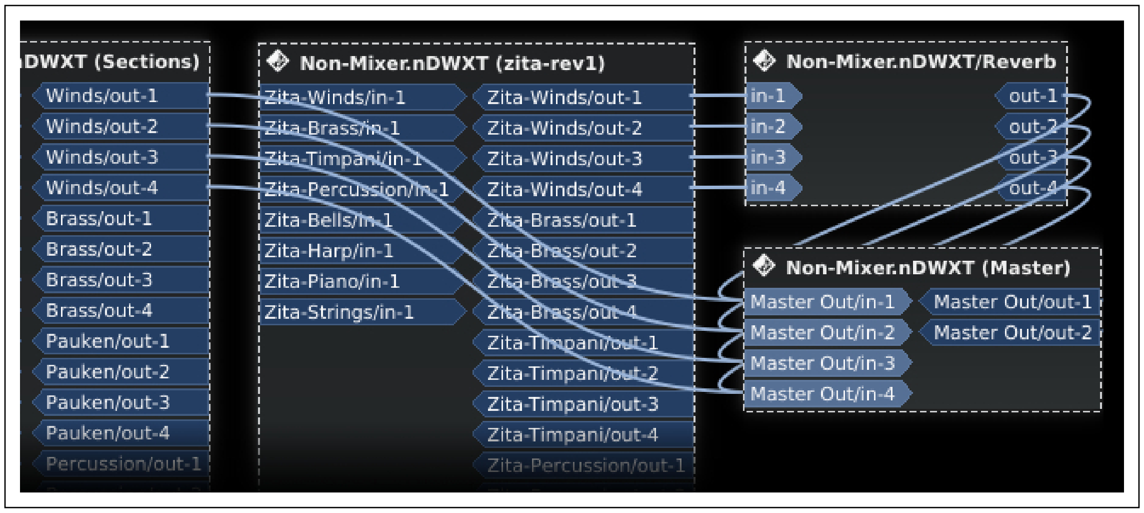


**Fig. 31.** *Dry and reverb connections to* **Master**

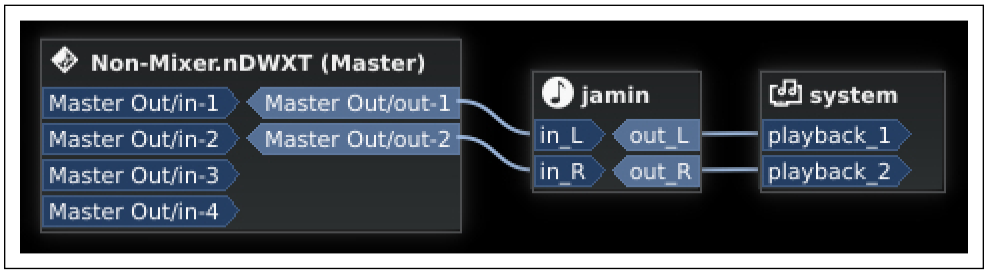


**Fig. 32.** **Master** *connected to* **jamin→system(playback)**

At this point the virtual orchestra is completely set up and ready for adjusting levels, panning, and reverb.

# Levels, panning, and reverb

In conventional audio production, levels, panning, and reverb are adjusted at the end, after all the tracks have been recorded. In midi orchestration, they must be adjusted even before commencing on a score, otherwise it is impossible to assess the effectiveness of the orchestration or to determine the correct dynamics. No two projects are ever identical, but it is essential to have a "baseline" orchestra to begin with where the instrument levels are normalized and all the players in their correct positions.

## Setting instrument levels

Every instrument in the score template has a corresponding instrument strip in **Non-mixer**. These strips are used to normalize the volume of each instrument such that a mezzo-forte in the flutes is approximately the same as a mezzo-forte in the violins.

The most effective way to set instrument levels is to open the score template in MuseScore and save it as `levels.mscz` . For every pitched instrument except timpani, write a two-octave scale in quarter notes in the instrument's most typical range (e.g. flutes A4 – A6, oboes D4 – D6, clarinets B♭3 – B♭5, etc.). The scales should be written sequentially, one instrument after the other, rather than stacking them.

> For reasons as yet undiscovered, Musescore sometimes plays recently opened files at twice the expected volume. It is recommended, therefore, that you save, close, and re-open `levels.mscz` before proceeding.

Strings form the heart of the symphony orchestra, so it is best to set their levels first. Select the two bars containing the Violins 1 scale, click the loop-playback button in MuseScore's toolbar (🔁) and start playback. Raise the Non-mixer window and scroll to the Violins 1 instrument strip. The peak meter should be responding to each note.

If it is not already active, switch to **Fadr view**. Using the slider, adjust the gain until what you hear is a first violin section playing mezzo-forte. Ideally, the meter's peak should be around -5dB. If you hear mezzo-forte but the peak isn't around -5dB, or if the peak is correct but you don't hear a mezzo-forte, adjust the Violins 1 output gain in **LinuxSampler** until you achieve the ideal.

Using the Violins 1 peak as a standard, proceed similarly through every instrument in the `levels.mscz` score, always aiming for a similar mezzo-forte around -5dB.

French Horns require special treatment because their carrying power is less than that of other brass. In *tutti* passages, it is established orchestration practice either to double the horn parts or to mark them one dynamic level higher than the rest of the orchestra. For now, their levels should be set to match the other instruments. The difference in carrying power often takes care of itself when the horns are situated properly in the virtual orchestra.

Be attentive to sharp spikes or drops on certain notes, or in certain parts of the two-octave scale. These indicate problems in the soundfont itself, and may require choosing another soundfont. If none is available, the problems must be be dealt with in MuseScore on a note-by-note basis.[1]

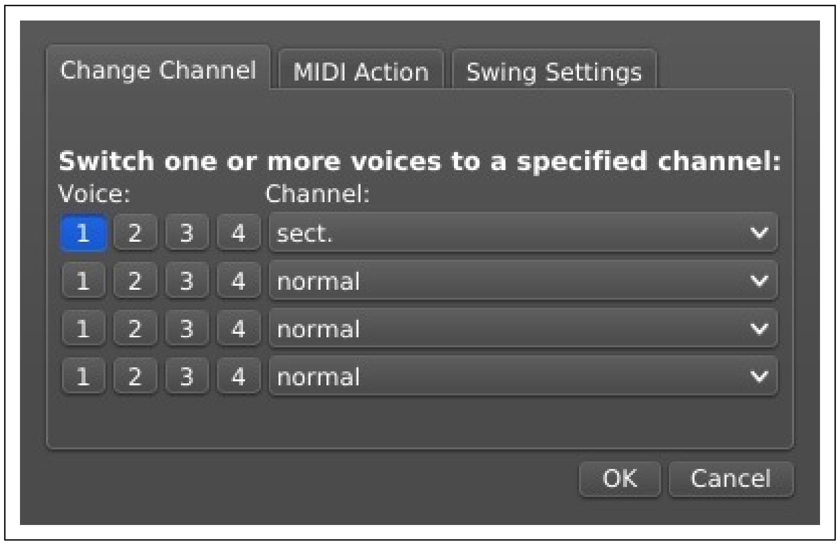
Bear in mind that the goal at this stage is uniformity of input and output signal. Refinements that take into account the natural carrying power and loudness of different instruments are made after putting the orchestra sections into position.

## Applying chorusing to wind departments

Wind departments playing in unison, indicated with the score marking *a due* or **I.II.**, require their own test scales, which must be assigned to the *sect.* channels. Switching to *sect.* channels is done by

- clicking on the first note of the scale
- hitting **Ctrl-t** to add Staff Text
- entering some text (e.g. *a due* or **I.II.**)
- hitting **Esc**, which leaves the text highlighted
- right-clicking the highlighted text to bring up the **Staff Text** dialogue
- selecting **Staff Text Properties**
- clicking on the desired voice (1) to highlight it
- selecting the desired channel ("sect.") from the dropdown beside the selected voice (Figure 33).

[1]Alternatively, the soundfont can be repaired using a soundfont editor, e.g. swami or polyphone.



Switch one or more voices to a specified channel:

| Voice: | Channel: |
|---|---|
| 1 2 3 4 | sect. |
| 1 2 3 4 | normal |
| 1 2 3 4 | normal |
| 1 2 3 4 | normal |

**Fig. 33.** *MuseScore* Staff Text Properties *channel selection dialogue*

**Caveat:** The voice number must always be highlighted when changing channels otherwise the channel switch will have no effect. This is particularly important when switching back to the "normal" channel. Failure to highlight the affected voice(s) will result in no change taking place.

Once scales have been assigned to the winds *sect.* channels, the amount of chorusing needs to be established before the levels can be set. Carla's "Rack" tab holds the chorus plugins. Clicking on either the **GUI** or **Edit** buttons of the plugins brings up controls.

Other than making sure the number of instruments to be chorused is correct, no advice can be given on the best settings to use. Only the ears can determine what sounds most like two oboes or two clarinets playing in unison.

Do not raise the level of *sect.* instruments higher than that of solo instruments on the assumption that two instruments sound louder than one. Two flutes in unison should exhibit a difference in timbre, not volume.

## Applying equalization to string departments

As with chorusing winds, if equalization is being applied to the same strings soundfont to simulate the timbres of the different departments, that, too, must be taken care of before setting the levels. Separate test scales need to be created and the channels set appropriately (i.e. *pizz.*, alt. 1, etc.).

The equalizers are located on the strings' department strips (Fig. 18). Clicking on the module brings up the controls. As with chorusing, no practical advice can be given as to which settings are best. In a very general way, the one should aim for:

**Violins 1:** bright without becoming shrill in the upper register.

**Violins 2:** the same as Violins 1; the slight difference in timbre between Violins 1 and Violins 2 resulting from the second violins being a smaller section to the right of and partially behind the firsts takes care of itself when the section is positioned correctly.

**Violas:** fuller but duller than the violins, with a faintly nasal quality and a suggestion of hiss; violas are difficult to reproduce with a general purpose strings soundfont.

**Cellos:** full and resonant in the lower register, with the same carrying power as the violins; clear, passionate and "chesty" in the upper register.

**Basses:** similar to cellos, but with a coarser tone and greater resonance in the low register.

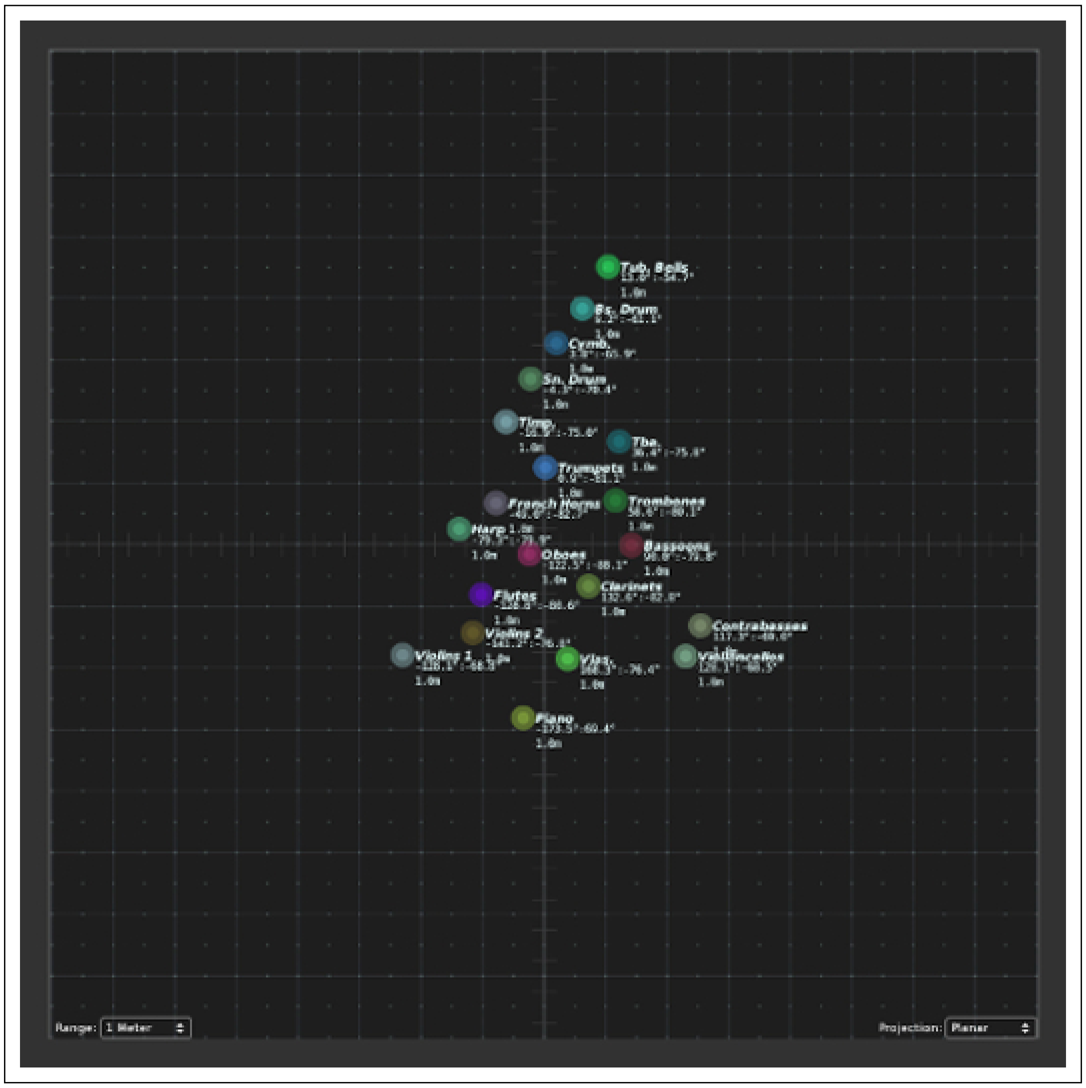
It will be noticed that all of the channels for each string department (normal, pizz., alt. 1, alt. 2) go to just one equalizer. Experience shows this to be preferable to equalizing each channel individually, although it is possible to do so by putting equalizers on each instrument strip and removing the one governing the whole department.

Care must be taken to preserve "presence" in the string departments. Trimming upper frequencies in the violins to remove shrillness, for example, may result in a canned sound when they are playing in their medium and low registers. It is better to deal with problems like screechiness or boominess by applying shelf filters in **JAMIN**, i.e. at the end of the audio chain.

## PANNING: LEFT / RIGHT PLACEMENT OF DEPARTMENTS

The ambisonic panners on the Sections strips make possible the use of Non-mixer's convenient **Spatialization Console** for left/right placement of orchestral departments (see Figure 34). The console is opened by pressing the computer keyboard's **F8** key. Departments are panned to the desired location within the console simply by dragging them.

The console is marked off in a grid. The Y-axis represents left-to-right placement relative to the grid's centre. The X-axis—actually the Z-axis—



Range: 1 Meter

Projection: Planar

**Fig. 34.** *Spatialization console*

represents front-to-back placement but is disabled for the setup described herein.[2] The placement of orchestral departments along the X-axis in Figure 34 is to prevent overprinting and has no effect on front-to-back placement. It is recommended that the **Range** (bottom left) be set to 1 meter to prevent overprinting as well.

The orchestral seating chart in Figure 24 may be used as a guide to positioning departments left-to-right. In addition, Mattias Westlund's online article,

[2]Full left-right/front-back spatialization via the console requires the use of Non-mixer's spatializer plugin on the section strips rather than the ambisonic panner. The plugin is meant for use with convolution reverb, which is set up and wired differently. See Appendix.

*Orchestral Positioning: Panning*, provides an introduction to the subject with good general guidelines.

Since the string section spans the whole virtual stage, it is best to begin by getting the string departments in place. Using the scales in the **`levels.mscz`** score, start with the first violins, on the "normal" channel. After dragging them to a suitable location on the left side of the virtual stage, situate the cellos at an approximately equivalent position on the right. Listen to the scales sequentially and check for overall balance of positions, then stack them and play the first violins and cellos simultaneously, listening for blend. If either the balance of positions or the blend seems off, experiment until both are satisfactory.

Proceed to the second violins, which should be slightly to the right of the firsts. The distinction ought to be subtle, more as if the seconds are filling a hole between the first violins and stage centre than simply being to the right of the firsts.

Violas are tricky. They occupy the centre position, but mustn't come across as an extension of the violins; they should sound audibly *between* the violins and the cellos, acting as an aural bridge. This usually means panning them a little right of centre to begin with, and adjusting the final position after situating the cellos and basses.

When positioning the basses, blend trumps position. Stack the cello and bass scales—the basses playing an octave lower than the cellos—and drag the basses slightly to the left, through, and to the right of the cellos while the scale loops. A noticeable sweet spot should become evident, where the two sections bloom into a resonant, unified whole.

It is advisable at this stage to write or transcribe a short passage for strings in the mezzo-forte dynamic range and use it to assess the strings' panning. Fine-tuning will almost certainly be necessary, especially the violas.

The guidelines in Westlund's article may be followed for positioning winds, brass, and percussion. Essentially, the winds are grouped closely in the centre—from left to right: flutes, oboes, clarinets, bassoons.[3] The brass occupies

[3]Winds may also be in two rows, flutes and oboes in front, clarinets and bassoons behind. In this arrangement, the two higher winds are panned slightly left of centre, the lower two slightly right. The front/back positions are adjusted with Z-axis panning.

the middle two-thirds of the virtual stage, approximately equidistant: horns on the left, trumpets in the middle, trombones/tuba on the right. Timpani and percussion, which require that appropriate test passages be created, should be slightly left of centre. Harps go to the left of the horns. If a piano is needed as part of the orchestra, it should go a little left of the harps.

After all the departments have been positioned one at a time, the blend should be checked. Panning has a noticeable effect on how well instruments sound together. Short test passages in chorale style for the winds and the brass—just a few bars of textbook four-part harmony for both—is usually sufficient to reveal flaws.

Be aware that no matter how satisfactory the initial placements sound, they will almost certainly need slight adjustments at the start or during the course of every new project.

Save the settings frequently, using **Non-session-manager**'s **Save** rather than Non-mixer's. If some part of the audio setup crashes and requires a session to be aborted, the loss of work will be slight.

## REVERB: FRONT / BACK PLACEMENT OF DEPARTMENTS

Placing departments front-to-back, known as Z-axis panning, is trickier than positioning them left-and-right. The illusion of stage depth is just that: an illusion. Successful Z-axis panning relies on well-developed aural discernment and familiarity with the sound of a live orchestra. Westlund's series of articles beginning with *Orchestral Positioning: Reverb in theory* provide valuable insights.

Two factors influence how far back a department sounds: the amount of dry signal, and the amount of reverb mixed in. Overall reverb in the virtual hall is controlled by the master **Reverb** strip, while the reverb generated by the various sections is controlled by adjusting the "Department strips"' Aux signals. The dry signal is controlled with the strips' Gain sliders.

It is best to establish the overall hall reverb first. Start by zeroing the master Reverb gain (slider all the way to the bottom). Then, using the **`levels.mscz`** score, loop a portion of the first violins material. While the loop plays, raise the Reverb gain until a desirable level is attained. Compare the first violins with the cellos and find a compromise if one or the other sounds too wet or too dry. For the purposes of Z-axis panning, these two departments of strings are closest to the front of the virtual stage; the remaining string departments and the other sections of the orchestra are all behind them to a greater or lesser degree.

The amount of overall reverb can be adjusted once the departments are in place, so it is not essential for the initial **Reverb** setting to be the ideal.

An advantage to using algorithmic reverb like zita-rev1 is that the reverb's profile can be altered by adjusting the delay, the low and mid range reverb lengths, and the equalization. Making adjustments at this stage is not recommended, though; they are better done on a project-by-project basis.

Initial Z-axis panning is not done with the reverb strips (Zita-Winds, Zita-Brass, etc.), but with the **Aux** modules on the "Departments" strips. Clicking on the module brings up a gain slider, which controls the amount of signal the section will be sending to the reverberator.

The most reliable way to establish Z-axis panning is to use a previously-written score that calls upon all the instruments. Such a score provides the proper context for assessing front-to-back placement. When a score of this type is not available, scales in unison (at the appropriate octave) should be stacked, from Piccolo to Contrabass, in **`levels.mscz`** and MuseScore's mixer[4] used to "solo" diverse combinations, e.g. the string choir with the flutes in order to position the flutes behind the violins.

The procedure for Z-axis panning is the same for all departments.

1. Loop the appropriate section of a test score (or **`levels.mscz`**) and start playback.
2. Call up the gain control for the department's **Aux** module.
3. Raise the Aux gain until the department is audibly further back in the virtual hall.
4. Compare the department's front/back placement with the strings section and fine-tune.
5. Compare the placement with other departments in the same section and correct as necessary.

If moving a department back results in noticeable weakening or thinning of the sound, switch to **Fadr** view and raise the strip's overall gain by small

[4]Other than providing a convenient way to mute and solo instruments, MuseScore's mixer serves no purpose.

increments to increase the amount of dry signal. Return to the Aux control and adjust the gain for more reverb if increasing the dry signal noticeably pulls the department forward.

### Special cases: 2nd violins, violas, French horns

#### *2nd violins*

In a live orchestra, second violins are slightly back of the firsts and extend deeper into the stage—again only slightly. If an identical passage is played in sequence by Violins 1 and 2, the seconds should sound like a good first generation copy: marginally softer and marginally more diffuse. The effect must be subtle but discernible.

#### *Violas*

The violas are further back than the violins but occupy stage centre, which increases their "presence." Give them a little more reverb than the second violins, then increase the dry signal to pull them forward. If increasing the dry signal makes them too loud in relation to the violins, decrease their gain slightly in the viola instrument strips.

#### *French horns*

In French, French Horns are called *cors d'harmonie*, harmony horns, which describes their principle role in orchestration. Horns are frequently used to fill in or reinforce harmony because they blend magically with all the other sections of the orchestra. This is due both to their neutral timbre at mezzo-forte or softer, and to the manner of their playing, with the left hand in the bell and the bells themselves pointing backwards. The sound is diffuse, almost directionless, seeming to come from a distance. The reverb, therefore, needs to be greater than that of other brass departments, and the dry signal a little less. The resultant weakening of carrying power is desirable since it requires doubling the horns or increasing their dynamic level by one in those places in the score where it would be required by a real orchestra.

## Balancing the ensemble

Once the departments are positioned, the sections need to be balanced. For example, the brass as a whole may need to be less prominent, or the winds moved further back on the virtual stage.

The Sections strips control the overall gain of the sections' dry signals, while the zita-rev1 strips determine the amount of reverb. Increasing or decreasing the dry signal alters a section's prominence, but may also affect its Z-axis placement; this is corrected by adjusting the section's reverb gain in the appropriate Zita-*section* strip.

It is next to impossible to determine the overall balance of sections without working from a completed score, which, moreover, is likely to have different demands from other scores. Balancing the ensemble, therefore, is most effectively done on a score-by-score basis rather than trying to establish a one-size-fits-all blend of sections. One of the reasons for separating instruments, departments, and sections in Non-mixer is that it makes re-balancing the ensemble a relatively simple matter: one has merely to adjust the sections' dry and reverb signals.

It is well worth remembering that no matter how carefully one sets up the virtual orchestra, every composition has different needs. This can come as a shock at the start of new projects, where the reaction may well be, "Why does that sound so wrong?" What's wrong is the same "wrong" faced by a conductor or audio engineer: music cannot be rendered sensitively by algorithms or presets, or even described completely by the score. Every piece is different and occupies its own unique sonic universe. The setup described herein is a template, a standard that simplifies creating that universe, not a recipe for hands-free perfection.

# Pre-mastering

The output from Non-mixer does not go directly to system playback (speakers or headphones). After the dry/wet signals from the sections strips have been shunted to the Master strip for final volume adjustment, the output is routed through **JAMIN** (see Signal Flowchart).

JAMIN is a mastering tool where equalization, compression, stereo-width, and look-ahead limiting are applied to the whole virtual orchestra. Lampros Liontos' excellent online article, *Mastering with JAMIN*, covers JAMIN usage in clear, non-technical language and is invaluable for getting the most out of the program.

Mastering is the final step in audio processing, so it may seem odd for the virtual orchestra to be connected to JAMIN at the outset. In truth, JAMIN should be doing very little signal processing while a project is being scored. Having it permanently attached to the audio chain is largely a matter of convenience. There are, however, several settings that it is well to take care of from the outset.

## JAMIN SETTINGS

### Makeup gain

Makeup gain is a compressor control that boosts signal in the low, middle, and high bands, chiefly to compensate for attenuation that results from equalization and compression. In the case of midi orchestration, it provides necessary *oomph* to an orchestra composed entirely of synthetic instruments and prevents it from sounding thin or anaemic. Figure 35 shows a portion of JAMIN's window with the makeup gain sliders highlighted.

Leaving them at their defaults (04.0) is usually sufficient. Alternatively, one may click **Global bypass** at the bottom right of the JAMIN window, which mitigates the attenuation. Experience shows, however, that soundfonts, generally, benefit from the boost, producing a more realistic sound that facilitates orchestration.

### Equalization and compression

It is well beyond the scope of this paper to discuss compression and equalization. The topics are complex, and opinions differ concerning their

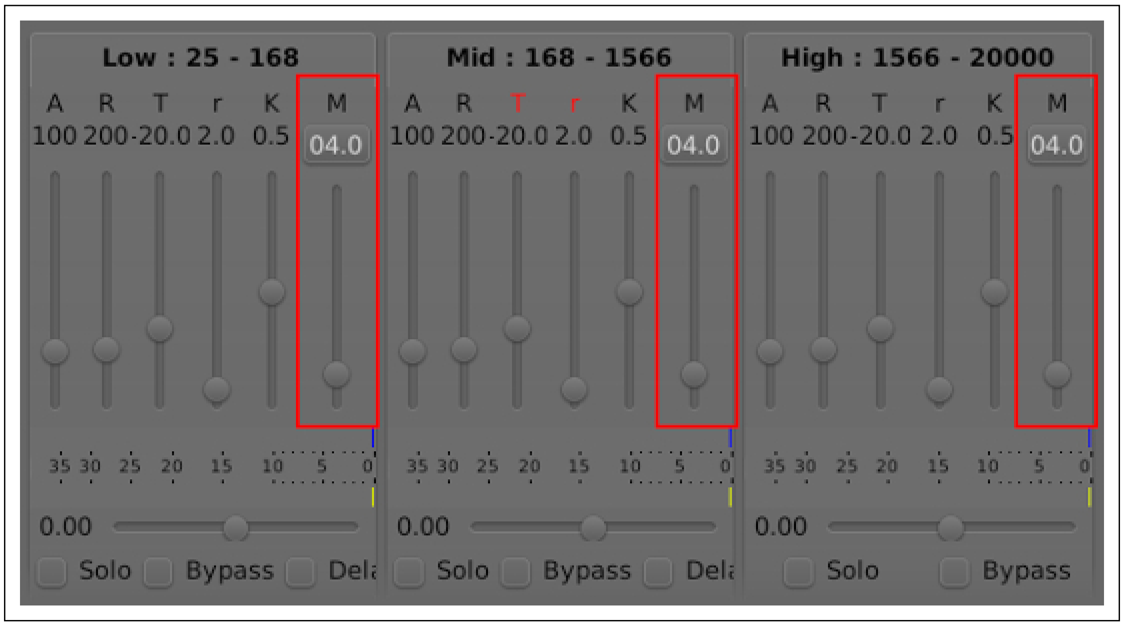


**Fig. 35.** JAMIN *compressors, makeup gain*

application to orchestral music. Compression boosts weak signals and clamps down on loud, narrowing the dynamic range—generally undesirable in music that routinely runs the gamut from pianissimo to blazing triple-forte. Careful use of equalization can correct some problems (shrillness, muddiness), but more often than not results in the canned, artificial sound that good midi orchestration seeks to avoid.

Generally speaking, adjustments to JAMIN should be made after scoring is complete, on a project-by-project basis, in preparation for recording. The settings should be saved in JAMIN itself since **Non-session-manager** does not store the active JAMIN configuration when a session is closed.

The only changes to JAMIN's defaults that may be desirable while scoring a composition are the application of shelf filters at the high and low ends of the equalization curve, and ensuring no that compression takes place.

## Shelf filters

Shelf filters cut or boost all frequencies below (low shelf) or above (high shelf) the shelf's frequency. A low shelf of -4 to -6dB at 50 – 60Hz removes rumble and the noise associated with electrical interference; this is unlikely to be a problem with the virtual orchestra, but it does no harm to set the shelf anyway. A high shelf above 0dB at 10kHz can be used to add brilliance to the orchestra. Conversely, the same shelf, set below 0dB, can be used to remove shrillness and upper frequency noise introduced by the reverb. It is usually in the latter capacity that a high shelf filter is most effective in midi orchestration. Figure 36 shows typical high and low shelves in JAMIN, which are set by dragging the yellow oblongs.

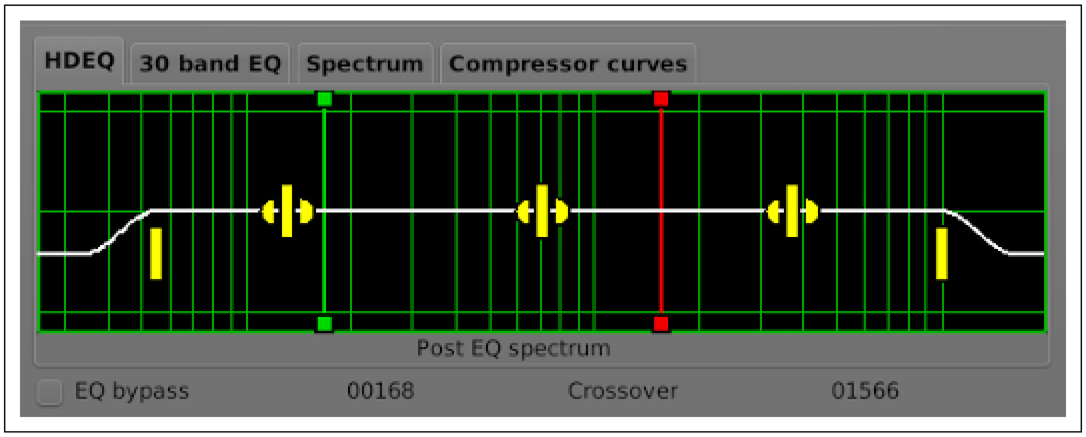


**Fig. 36.** *High and low shelves in JAMIN*

Regardless of where, when, or how much equalization is applied, caution is advised. It is a well-known phenomenon that new always sounds better when equalizing. After adjustments have been made, returning to the original settings often reveals them to have been preferable.

## Bypassing compression

The final adjustment you may want to make to JAMIN as part of the default setup is bypassing compression without disabling the makeup gain. This is done by raising JAMIN's three compression thresholds (the "**T**" sliders in Figure 35) all the way to the top. Compression is then applied only if the signal rises above 0dB, the point at which clipping (audible distortion of the signal due to excess gain) would occur.

Other adjustments in JAMIN that may be desirable after a project is completed should suggest themselves upon perusal of *Mastering with JAMIN* and Westlund's *Mixing and Processing a Virtual Orchestra*.

# Scoring for the midi orchestra with MuseScore

The goal of midi orchestration is a polished, printed score that observes all the conventions and standards of professional engraving while at the same time generating realistic playback—a score that performs itself. Playback must therefore reflect the timbre, dynamics, instrument blend, and balance that a reading of the same score by live musicians would produce. Only if this requirement is met is it possible to orchestrate with confidence; otherwise, one ends up writing to make the virtual orchestra sound good, with disappointing to ghastly results when the music is performed live.

It is possible, under Linux, to achieve the goal by different means than those described herein. One could, for example, write the score by hand, format it for printing with lilypond, then export the midi and edit it in a sequencer like rosegarden. Such a procedure, however, adds steps to a process that needs to be kept as simple as possible in order to foster experimentation, learning, and mastery of the orchestra.

MuseScore is the only graphical score writing software available for Linux capable of truly polished scores. The developers have grasped that beautiful engraving requires manual control down to the smallest detail so one is never stuck with inalterable, algorithmically-derived solutions to formatting challenges. Though not for orchestra, the score of *K'un*, for piano solo (meterless, with additive rhythms and four-staff systems), admirably demonstrates MuseScore's sophistication. It can be viewed on YouTube at

https://www.youtube.com/watch?v=qEWPldWLbLs

At the time of this writing, however, it must be acknowledged that MuseScore's midi handling is not nearly as sophisticated as its engraving.[1] Coaxing a musically sensitive performance from sampled instruments is a painstaking task regardless of the software, but MuseScore's midi deficiencies add significantly to the labour.[2] That said, it remains the best available

[1]Version 2.0.2 is the current, non-development release.

[2]The developers are quick to point out that MuseScore is a notation program, not a

choice of WYSIWYG programs for engraving orchestral scores in a Linux environment. Overcoming the midi limitations forms the bulk of what follows, and indeed, much of what has come before.

For the remainder of this paper, it is assumed the reader is proficient in MuseScore usage. The MuseScore Handbook is complete and well-written, complemented by developer Marc Sabatella's in-depth *Mastering MuseScore*. Questions about usage may be posted to the MuseScore forum and are answered quickly.

## Starting a new project

New projects begin with cloning the NSM session holding the orchestra. Rather than firing up NSM and using the Duplicate button, which can be very slow, it is better to navigate to the NSM directory/folder and copy the orchestra to a new name. After launching MSN, clicking on the new session name opens it.

Once the session is fully loaded, raise the MuseScore window and open the score template file. Using **Save As...**, save the template to an appropriate new name, which closes the template, leaving it unaltered, and switches to the newly named file.

Next, raise the LinuxSampler window and load the appropriate .lscp script.

If no changes are to be made to the LinuxSampler setup, it is safe to close LinuxSampler and remove it from the NSM session entirely—advantageous because the java front-end ("Fantasia") loads slowly and gobbles system resources. Quitting the front end and removing it from the session leaves the server running in the background with the current configuration.

A good habit to get into is to start the server from the command line, along with an .lscp script, before launching NSM and opening a session.

```
linuxsampler &
cat <path-to-script>/<script>.lscp | netcat localhost 8888
```

Thereafter, only the second command is needed to change to another LinuxSampler configuration, for example to work on a different project.

DAW front end, and that notation takes precedence over midi in the development cycle. It is to be hoped that, in time, they will change their tune and redress the imbalance. MuseScore is too good at what it does to stay hobbled by what it doesn't.

Lastly, raise JAMIN and load the default settings for the orchestra.

## Setting up the score

Score setup begins with hiding staves that are not required. MuseScore respaces the score accordingly, and if this yields the desired layout and instrumentation, nothing more need be done except perhaps reducing the page length, leaving room to expose one or two hidden staves. Other adjustments might include staff spacing (**Layout→Page Settings**), and the various settings under **Style→General→Page**.

### *Adding instruments*

If the orchestration calls for instruments not present in the template, adding them demands a bit of planning. Since additional staves incur the penalty of changing the port/channel/audio-out assignments of all the staves that come afterwards and re-wiring the Carla patchbay, the template should be studied for ways to avoid the extra work. It usually can be, if one remembers that any staff or staff channel can be made to play a different instrument simply by changing the soundfont in LinuxSampler.

Using the score template for winds in pairs as an example, if an alto flute is needed in the score but not the piccolo, Flutes I and II can be made to occupy the top staff (Piccolo), leaving the original Flutes staff available for the alto flute. The procedure would be as follows:

**MuseScore** —

- rename the piccolo staff to **Flutes I.II.**; the piccolo *alt.* channel can be used for *sect.* flutes
- rename the flutes staff to **Alto Flute**

**LinuxSampler** (front-end) —

- exchange the piccolo soundfont for the flute soundfont
- exchange the flute soundfont for the alto flute
- change the channel assignment of the soundfont used for *sect.* flutes to that of the *alt.* piccolo; assuming the setup described in the preceding chapters, the *alt.* piccolo is on port 0, channel 2
- export (save) the new configuration to a meaningful name

**Non-mixer** —

- change the gain of the Piccolo instrument strip to match that of the Flute

- set the gain of the Flute instrument strip (now Alto Flute) to the desired level

**Carla** —

- in the patchbay, re-route the LinuxSampler audio-out for piccolo *alt.* (channel 2) to the Chorus-Flutes plugin
- in the plugin rack, adjust the volume of the Chorus-Flutes plugin as necessary

It's quite a few steps, but if an Alto Flute staff were simply added under **Flutes I.II.** instead, all of the subsequent LinuxSampler audio-outs would have to be re-wired in Carla, or all the port/channel assignments in LinuxSampler changed. Both are prone to slip-ups and require considerable effort. When adding instruments, it is always good practice to use the template creatively to keep re-wiring to a minimum.

## Workflow

As stated, the goal of midi orchestration is a correctly-formatted score that plays itself. Three separate but related activities are involved: entering notes (composing), making them sound good (interpreting), and making them look good (engraving). The distinction between the first two is not hard and fast; inspiration often derives from experimentation, which demands getting some things "right," sonically, in order to hear where the music wants to go.

It is best to think in terms of creating two scores, a performance score and a printed score. The performance score is the working copy where composition takes place without regard to aesthetics and layout conventions. As work on a project proceeds, the performance score becomes filled with notes that have been colourized, "revealed" hidden dynamics and articulations, and other clutter needed for a musically-sensitive midi performance.

When composition is complete and the playback deemed satisfactory, the score is cloned and a properly-formatted print copy prepared. Layout issues are addressed, notes are de-colourized, non-essentials hidden, slurs and phrasing tweaked, etc. This is what MuseScore does best and most efficiently. The ample documentation should be consulted for methods, tips, and tricks that assist composers in creating the best possible scores. Figure 37 shows a section of score in working copy and print versions.

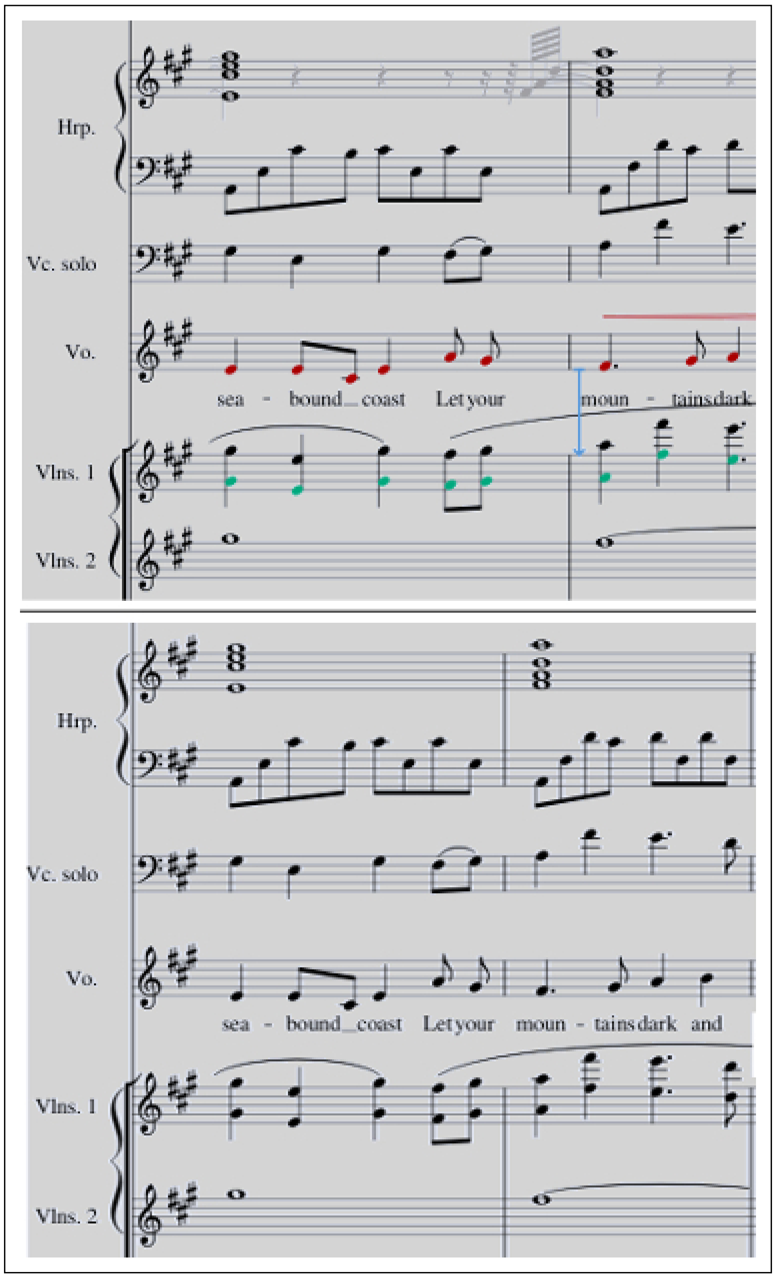

Hrp.
Vc. solo
Vo.
sea - bound_coast Let your moun - tains dark
Vlns. 1
Vlns. 2
Hrp.
Vc. solo
Vo.
sea - bound_coast Let your moun - tains dark and
Vlns. 1
Vlns. 2

**Fig. 37.** *Score in working copy and print versions*

## Midi editing controls

The MIDI specification includes controls for modifying channel volume (dynamics), individual note velocity (loudness relative to channel volume), swell (crescendo or diminuendo through notes), gate time (sounded duration in relation to notated duration), and other parameters. For the sanity of the composer and to encourage excellence in playback, these controls should be adjustable from the score.

Other methods of editing midi than working from the score exist—piano roll editors, for example—but they do not allow composers to grasp the harmonic context of notes, nor to identify quickly the role a note plays in the instrumental fabric. Creating a score for midi performance, one becomes, in effect, both composer and conductor. Few conductors would undertake a work written in piano roll notation; it lacks the necessary musical information from which to make interpretive judgments. The same applies to midi "conducting." It must be done from the score.

While the ideal of all controls adjustable from the score may be asking for the world, *these five are indispensable:*

- volume
- velocity
- swell
- gate time
- tempo

The orchestra is not about highly expressive, individual performances, but about players working together to communicate a composer's intentions. These five controls, though few, are enough to draw musically satisfying performances from a virtual orchestra.

MuseScore provides only volume, velocity, tempo, and, to a very limited extent, gate time.[3]

## Tools

### *The Inspector*

MuseScore's Inspector is a hideable panel conveniently located to the right of the score. When a note, dynamic, hairpin, or articulation is selected on the score, the Inspector reveals these editable midi parameters:

- **Tuning**: for use with MuseScore's internal sampler only
- **Play**: (un)mute notes, execute or ignore articulations and ornaments
- **Velocity**: note or dynamic loudness
- **Velocity type**: whether note velocity is absolute ("User") or relative to the current dynamic ("Offset")

[3]From the score itself, the latter is available only when a note is decorated with an articulation; gate times for articulations can be edited in the .mscx file. MuseScore's piano roll editor, which should serve for shortening gate times, is too beta to be useful.

- **Velocity change**: hairpins
- **Time stretch**: articulations

The visual aspect of notes and other score elements is also controlled from the Inspector. Two operations—changing the colour and toggling an element's visibility—form a routine part of midi orchestration. Toggling visibility can be done with a keyboard shortcut, but changing colour must be done from the Inspector.

### *Staff Text Properties*

When text directions are added to a score (**Ctrl-t** on a selected note), the **Staff Text Properties...** dialogue allows mapping up to four staff voices to different midi channels (Figure 33), subject to the channels having been added to the score's .mscx file. Primarily for switching channels to implement directions like *pizz.* or *a due*, it can be used in conjunction with hidden voices for layering and other useful tricks.

### *Tempo markings*

Tempo markings in a score are observed during MuseScore playback, either literally, if a "note value=beats-per-minute" is given (e.g.♩ **= 80**), or via a beats-per-minute setting in the Inspector. In addition to setting the overall tempo, tempo markings provide the means to control accelerando, ritardando, and rubato.

### *Articulations*

Like tempo markings, articulations in the score are observed during playback. Articulations have two controllable midi properties: gate time and time stretch. The former determines how short a note decorated with an articulation sounds in relation to its full value, the latter how long. Both properties permit articulations to be used for non-obvious purposes discussed later in the chapter.

### *The Mixer*

With Non-mixer taking care of levels, panning, and reverb, there is no need for MuseScore's mixer other than to mute or solo instruments. In this capacity, it is more convenient than performing the same operations in LinuxSampler, since it can remain visible without occluding the score.[4] Soloing is helpful when adjusting note velocities for individual instruments.

The mixer also permits soloing combinations of instruments, so it can be used to verify balance and blend between or within sections as well.

## SCORING TECHNIQUES

### Tempo

The overall tempo of a piece is set using **Tempo Marking**, which can be accessed from the menubar (**Add→Tempo Marking**) or from the keyboard with **Alt-t**. The default is ♩ **= 80**. Double clicking on the marking when it's selected allows editing. The note value can be changed by replacing the quarter-note with a different note value from the **Insert special characters** palette (**F2**), and the tempo set by erasing "80" and entering a different beats-per-minute (BPM).

The Inspector contains two settable items for the tempo marking, **Follow text** and **Tempo**. If **Follow text** is checked, the tempo will be as marked. If unchecked, the **Tempo** field allows setting the BPM to an arbitrary metronome speed—useful if your **Tempo Marking** does not include a "note-value=*n*," e.g. says only "Allegro" or "Adagio."

### Dynamics

Dynamics are how composers indicate the overall loudness of a passage, and, by extension, how the various departments should adjust to achieve it. Except in the case of French horns, it is rarely necessary to specify different dynamic levels for different instruments. In a passage marked forte for all instruments, the players make adjustments so the *orchestra* is forte, not they.

In midi orchestration, these adjustments need to be explicit.

#### *Dynamic markings*

Dynamic markings have a default velocity (loudness/volume) assigned to them that can be seen in the Inspector when the dynamic is clicked: ***pp***=39, ***mf***=80, ***ff***=112, and so on. The range of values is from "0" (silence) to "126" (maximum volume).

With a well set up virtual orchestra, these default values are adequate to

[4] If the LinuxSampler front end is not running, the mixer is, in fact, the only way to mute and solo. Non-mixer cannot serve the purpose since its strips can only be muted, not soloed.

determine if a passage should be marked, for example, ***mf*** or ***f***, but within the passage, some of the instruments may need to be louder or softer.

For example, if the first and second violins are playing in unison, a ***mf*** marking in both parts will likely produce unbalanced midi dynamics, the combined mezzo-fortes resulting in violins that predominate by volume rather than by the increased fullness of tone expected from such a unison. Reducing the velocity of the ***mf*** for one or both parts is therefore required to balance the violins with the rest of the orchestra.

> **Tip:** A good habit to adopt is colorizing dynamics, notes, and hairpains whose velocities have been changed from the default. A hot colour, like red, indicating "increase," and a cool colour, like blue, indicating "decrease." make it easy to see what kinds of changes have been made, and where. Restoring everything to black for the final, printed score can be accomplished in a single, global operation.

### *Hidden dynamics*

Often, as an instrument changes register, less or more velocity is needed to achieve the impression of a consistent dynamic. Clarinets that start off mezzo-forte in the clarino register can sound as if they're playing forte when a passage climbs into the altissimo register. While this effect can be mitigated by selectively reducing individual note velocities, it is often simpler to add a hidden dynamic that covers all the offending notes.

A similar use for hidden dynamics is adjusting the balance of a section to simulate what live players would do, either reflexively or on instruction from the conductor. Figure 38 shows hidden ***mp***s (grey) in the strings immediately after the visible ones. In this score, their velocity was reduced slightly for the start of a phrase featuring winds, also marked ***mp***—the same adjustment live players would make.

Hidden dynamics can also raise the volume of staccato notes, a particular problem with strings, where the soundfonts may not have time to reach full volume because of the short gate time. A passage marked ***p*** that begins legato and ends staccato may need a hidden ***mp*** (or greater) to bring the staccato up to piano. Figure 39 demonstrates.[5]

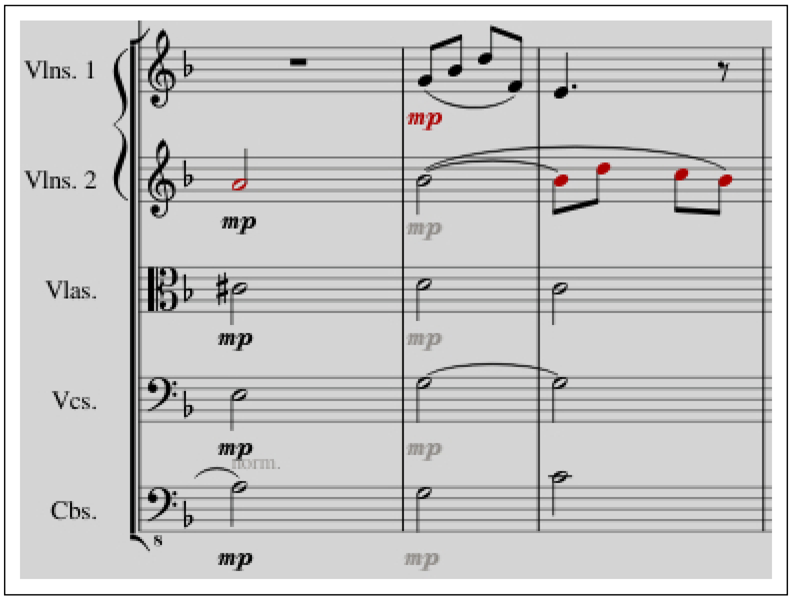


**Fig. 38.** *Hidden dynamics correcting string section balance*

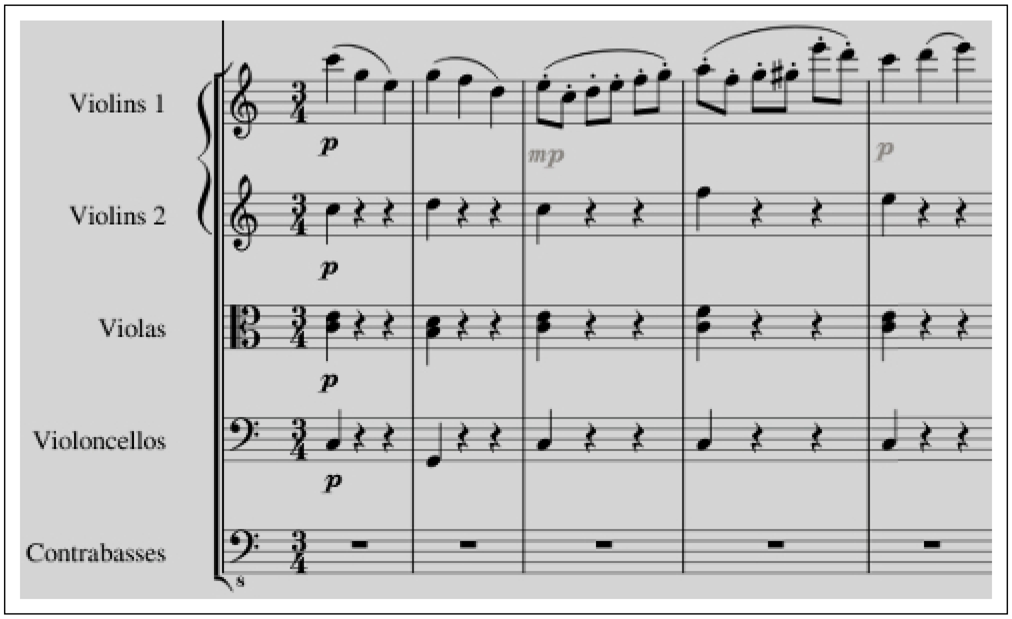


**Fig. 39.** *Hidden dynamics correcting staccato violins*

***Doubling notes*** It is sometimes necessary to coax more volume from an instrument than the upper velocity limit of "126" allows. This is a common problem with brass, horns especially, where a neutral soundfont, chosen for

[5]Alternatively, a hidden ***p*** could be used, its velocity raised to an appropriate level. The dynamic mark selected for hidden dynamics is unimportant since any one can be set to a velocity from 0 to 126.

blend, prevents them from reaching a satisfactory fortissimo. The solution is to double the notes in another voice and hide them. The velocity of the hidden notes can then be adjusted to produce the desired loudness.

Be aware that MuseScore does not provide a convenient way to select notes that are underneath other notes at the same position on the staff. The top voice has to be moved out of the way, as shown in Figure 40. In the first bar, an A4 has been entered and a dynamic assigned. Following it, an A4 has been entered in Voice 2 and the top note (Voice 1) selected. The third bar shows the Voice 1 note scooted up an octave to reveal Voice 2. The Voice 2 note and its stem are hidden in the next bar. Finally, Voice 1 is returned to its original position and colourized to indicate a note receiving special treatment.[6]

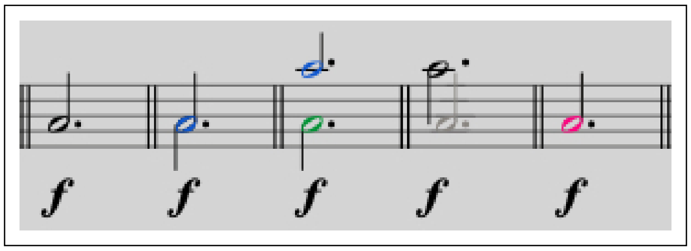


**Fig. 40.** *Hiding doubled notes*

### *Sforzando*

Sforzando and its ilk (***fp***, ***sfp***, etc.) can be simulated with hidden, short accent notes.

Owing to a bug in MuseScore's handling of midi polyphony, the effect is not possible with a hidden staff voice (Figure 40); the gate time of the shorter note applies to both. The workaround is is to put the accent notes on an unused (hidden) staff whose soundfont in LinuxSampler has been swapped for the needed one.[7]

Figure 41 shows a how a typical ***sfp*** is created. The upper staff, which will be visible in the printed score, has a full-value note whose ***sfp*** marking has been set to ***p*** (49). The lower staff, which will be hidden in the print version, holds the short, forte note that provides the accent.

[6]A "Hide Voice *n*" staff function would simply the procedure considerably.
[7]Which staff is unimportant; any unneeded one can serve the purpose.

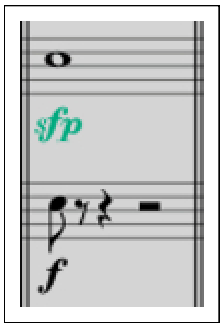


**Fig. 41.** *A typical sforzando-piano*

### *Crescendo/diminuendo*

Crescendo and diminuendo are accomplished with hairpins anchored to the first and last affected notes.

> **Caveat:** MuseScore's handling of crescendo and diminuendo is only effective from one note to the next. A whole note under a hairpin will not get louder or softer, only the note afterwards. See Swell.

Hairpins bracketed by dynamics transition from the first dynamic to the second. Depending on how the score is to appear infinal form, it may be necessary to hide one or both. Example 1 in Figure 42 shows an explicit ***f*** diminuendoing to an explicit ***p***. Example 2 shows a passage marked***f*** several bars beforehand diminuendoing to ***p***. The***f*** here is hidden.

Example 3 shows an ambiguous diminuendo. Although the second dynamic is ***p***, it is hidden from the printed score because the composer wants the phrase to get softer without arriving at an explicit dynamic. In performance, the conductor would normally decide how much diminuendo was needed. A “midi conductor” must be told exactly what to do.

Notice that the hairpins in examples 2 and 3 appear to start on the quarter-note D4. They are, in fact, anchored to the same eighth note as example 1, as Figure 43 shows. Hairpins can be extended visually in either direction without the anchors moving.[8]

Hairpins that are not bracketed by dynamics can use **Velocity change** in the Inspector to set the amount of crescendo or diminuendo. Bracketed

[8]Visual adjustments of this kind are best made after the working copy is complete.

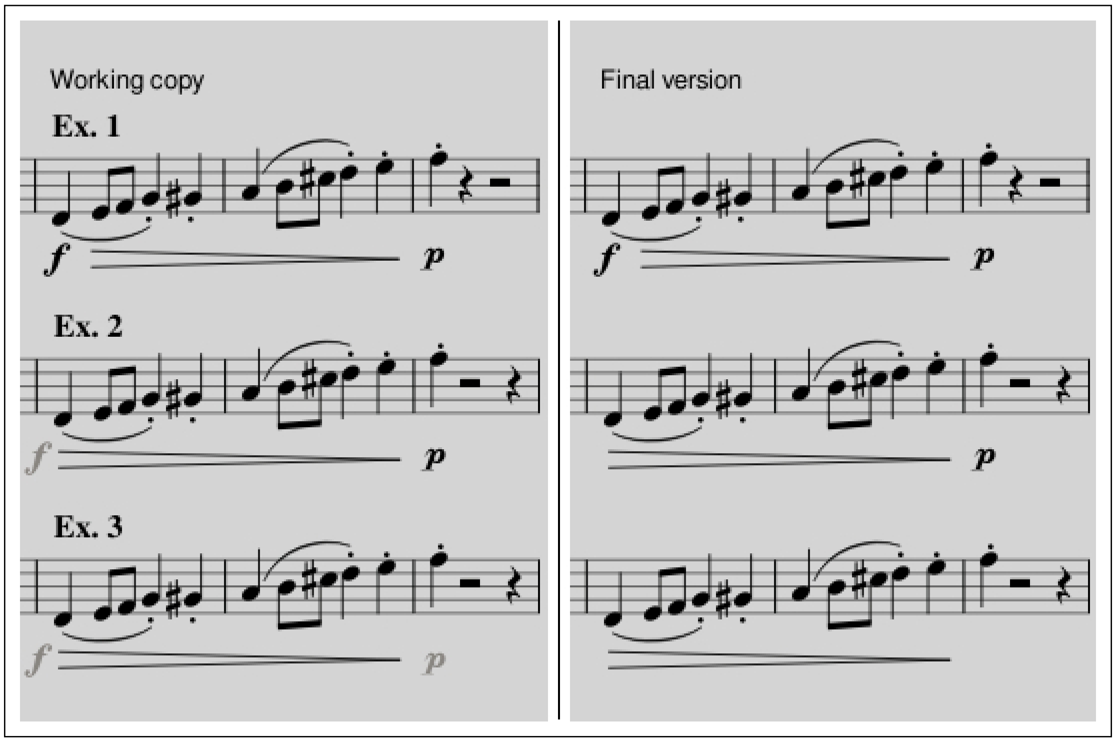


**Fig. 42.** *Hairpins between dynamics*

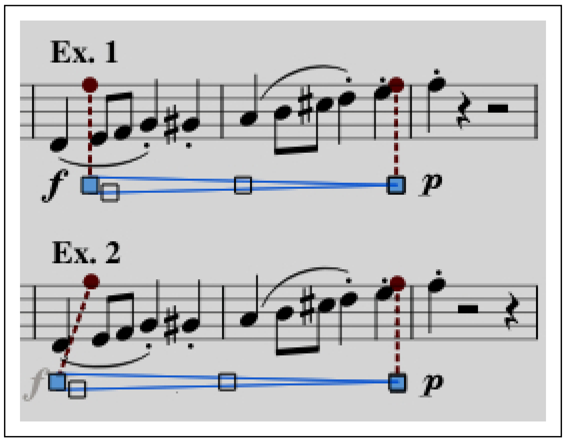


**Fig. 43.** *Hairpin anchors*

hairpins can also make use of **Velocity change** if the default dynamic gradation proves insufficient or excessive. Some individual notes under hairpins may need their velocities adjusted since *crescendi* and *diminuendi* rarely proceed in uniform steps.

Hidden, winged hairpins (<‍> ) can be added to expressive phrases to impart a characteristic dynamic arch. The technique should be applied to select instruments only and used with restraint lest the orchestra begin to sound like a squeezebox.

#### *Swell*

MuseScore 2.x does not support crescendo or diminuendo through long notes and there is no way of tricking it into doing so. Careful scoring of dynamic changes can mitigate the limitation, but for now, MuseScore lacks one of the five indispensable tools for midi orchestration.

### Phrasing

The biggest liability of midi is its dreadful uniformity. Without intervention, every note in a phrase sounds identical to every other with respect to loudness and articulation. Only the pitch and duration change. There are no strong and weak beats, no articulated breaks between phrases, no expressive shaping of the line, no rubato.

Phrasing is an art. Except in cases of prodigious talent, it takes years of study and practice to master. The elusive adjective, "musicianship," most often refers to a player's ability to shape phrases sensitively. The skill is indispensable for realistic midi orchestration.

Midi phrasing entails manipulating three basic elements: the velocity of individual notes in relation to their surroundings, the articulations within a phrase, and the delineation between contiguous phrases.

#### *Note velocity*

No task is more important for achieving midi realism—nor more painstaking—than adjusting note velocities. In good score-driven midi, it is not unusual for hundreds of such adjustments to be made.

Velocity controls a note's loudness. In MuseScore's Inspector, the value can be absolute (**Velocity type** "User"), or relative to the current dynamic ("Offset"). "Offset" is most useful: at a dynamic of ***p*** with a value of "49," velocity "10" raises a note's velocity to "59." Negative values are allowed. Note that the sum of the dynamic plus the offset can never be greater than "126." If more volume is needed, the note may have to be doubled in another voice.

Velocity adjustments influence two crucial aspects of midi orchestration: blend (vertical sonority) and expression (horizontal sonority). In a live orchestra, up to eighty players or more achieve blend and expression by making tiny adjustments to the volume of every note they play, supported by the rest of the orchestra, all of whom are doing the same thing. It follows that the more time spent getting each note perfect in a midi score, the more realistic the result.

So many factors go into getting note velocities correct that they cannot be reduced to rules or procedures. Only the ears can determine what works and what doesn't. Every phrase of a score needs to be auditioned critically. Every note that jars, or is weak, or doesn't sit well in the orchestration needs to be adjusted. Every one. The process is at the heart of good midi orchestration, and there are no shortcuts.[9]

Figure 44 shows a passage for winds in which all the notes whose velocities have been adjusted are colourized: red for louder, aquamarine for softer. It gives some idea of how laborious the process can be.[10]

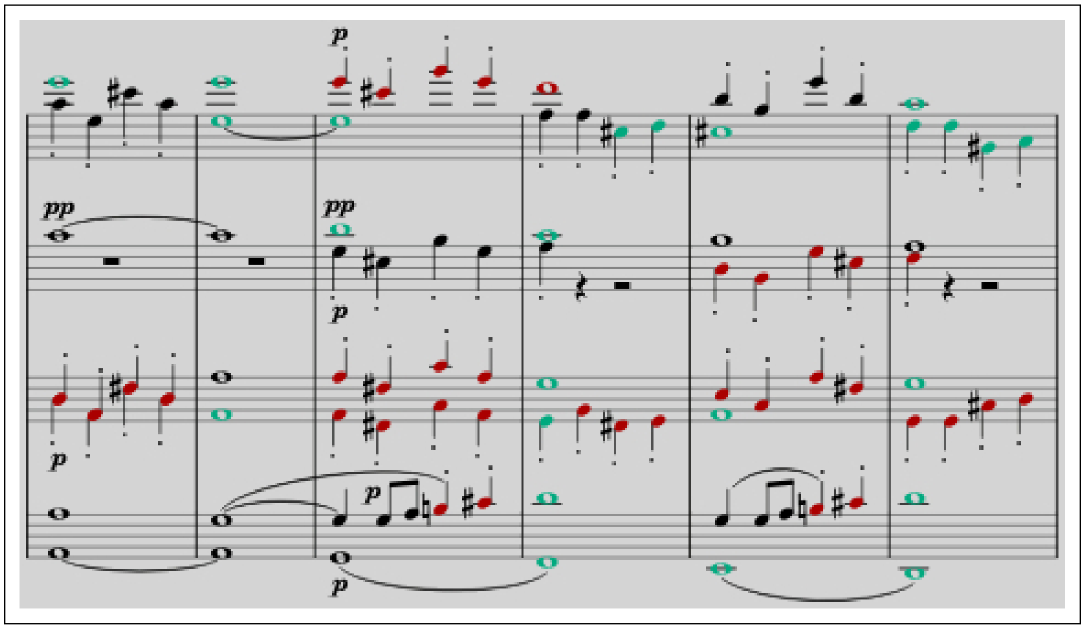


**Fig. 44.** *Velocity changes indicated by colour*

***Problem notes*** Blend and expression are not the only reasons for adjusting note velocities. Soundfonts collected from the Web are often marred by

[9]The section "Adjusting note velocities" in the online article *Improving Playback from MuseScore 2.0 Piano Scores* discusses the matter in detail.
[10]A handy feature in MuseScore would be for velocity-changed notes to change colour automatically. At present, every velocity change requires its own separate colour change.

notes that sound noticeably louder or softer than others. Velocity layers may be the culprit, in which case adjusting the velocity doesn't correct the problem. The more usual reason is carelessness, or lack of expertise, on the part of the soundfont's creator.

Fixing problem notes is a routine annoyance of Linux midi orchestration. If there aren't too many, it's a good idea to keep track of what those notes are for any given instrument, along with the changes needed to normalize their volume. That way, the velocities can be adjusted when the notes are entered, saving a little time.

### *Articulations*

In the .mscx source file of a score, every instrument is separately assigned six articulations with editable gate times: *sforzato*, *staccato*, *staccatissimo*, *tenuto*, *portato* and *marcato*, corresponding to the articulations in the **Articulations & ornaments** section of the MuseScore **Palette**. (Figure 45).

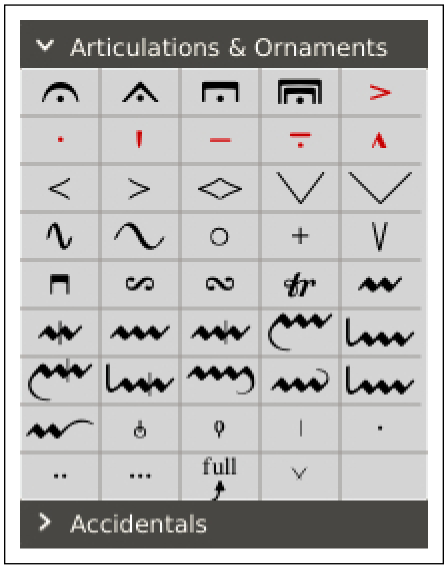


**Fig. 45.** *MuseScore* **Palette**, *articulations*

The gate times determine how long a note decorated with a particular articulation lasts, expressed as a percentage of notated value. For example, an articulation stanza like this

```
<Articulation name="staccato">
  <velocity>100</velocity>
  <gateTime>50</gateTime>
  </Articulation>
```

in the Oboes section of the file would mean that oboe staccato notes are half as long as their notated value.

The liabilities inherent in this are threefold. Firstly, there is no agreement in the musical world about the length of staccato, marcato, tenuto, and the like. Their interpretation is intentionally left to the performer and cannot be fixed by a rule.

Secondly, because the gate times are embedded in the .mscx file, they can't be changed from the GUI, and thus persist throughout the entire score.

Thirdly, using staccato as an example, if the gate time is "50," (fifty percent of notated value), a staccato quarter-note simply becomes a full-length eighth note. Were that a composer's intention, it would be notated as an eighth note with a rest afterwards, not as a staccato quarter, the meaning of which is "short," not "half value." Furthermore, the staccato becomes noticeably shorter or longer when the tempo increases or decreases; in performance, staccato is *coloured* by tempo, not *determined* by it. All MuseScore articulations are similarly encumbered by being a fixed percentage of notated value.

***Changing articulation gate times*** The gate times of articulation can only be changed by editing a score's .mscx file. The defaults assigned to each instrument's six articulations are somewhat arbitrary and, depending on the soundfont, tempo, reverb, and other factors, may not be optimal. Strings generally need the most attention.

> When scoring a piece, it is good practice to amend the score template .mscx file whenever an instrument's articulations are corrected so they become the new default for future compositions.

***Improving staccato*** When an articulation is selected, checking or unchecking **Play** in the Inspector determines whether or not the articulation is observed during playback. This, along with the fact that articulations can be hidden (press "**v**" or uncheck **Visible**), provides one way of overcoming fixed percentage gate times for staccato.

Generally speaking, longer notes (e.g. quarters) decorated with staccato need a shorter gate time than shorter notes (e.g. eighths). Consider the phrase in Figure 46. The version at top, unmodified, will exhibit an audible difference in the staccato quarters and the staccato eighths. The eighths will sound crisp enough, but the quarters will seem sluggish. In the version underneath, with **Play** unchecked for the staccato, hidden *staccatissimi* correctly impart the same staccato feel to every note.

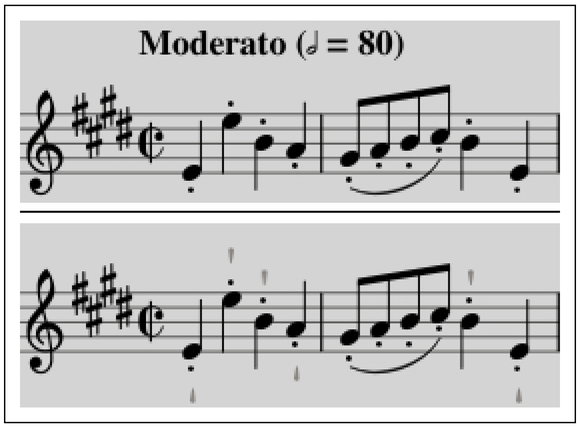


**Fig. 46.** *Hidden* staccatissimi *shortening quarter note* staccati

A cumbersome but potentially more satisfactory solution is to use a hidden voice to write out the staccato as short notes (sixteenths or smaller, as needed) with rests between. Figure 47 demonstrates. With **Play** unchecked for the visible notes and their staccati, the hidden notes of short value simulate the passage as it would be executed by a live player. The advantage to this method is that the staccato can be precisely controlled, as the hidden sixteenths under the quarters and the hidden dotted thirty-seconds under the eighths demonstrates.

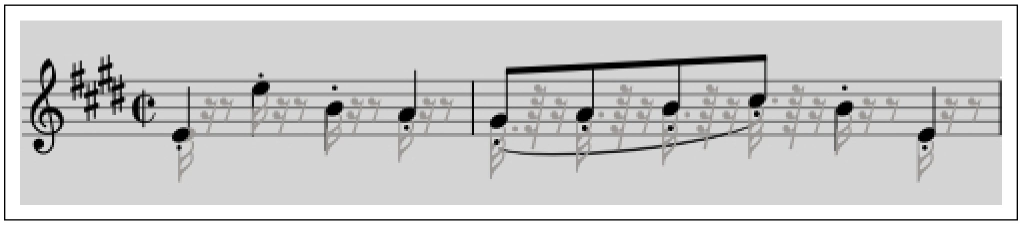


**Fig. 47.** *Hidden voice simulating staccato*

***Repeated notes, portato*** Since the samples in a soundfont begin immediately (i.e. there is no silence before the initial attack), repeated notes in legato phrases often end up sounding like single long notes punctuated by little hiccoughs. Worse, if the attack is not sufficiently distinctive, the repetition may not be audible at all.

The solution to this problem is the *portato* articulation ($\bar{\cdot}$) which, with the right gate time, can be used visibly (appears in the final score and is reflected in the playback) and invisibly (plays but is hidden in the final score) to render repeated notes clearly. Figure 48 shows two bars of an *allegro* passage for flutes and oboes with hidden portato over the repeated notes. Without them, the repetition wouldn't be heard.

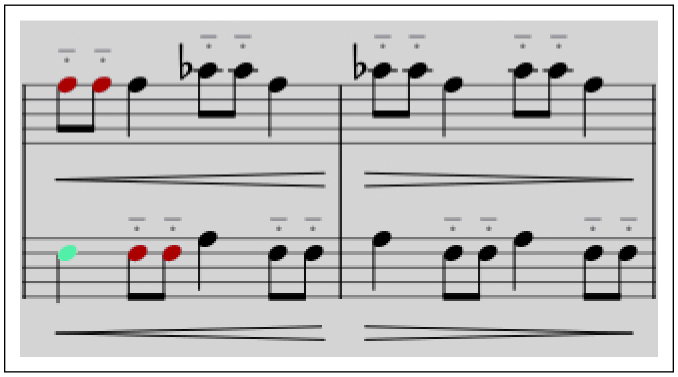


**Fig. 48.** *Hidden portato to improve playback of repeated notes*

***Phrase/slur ends*** When notes are slurred or legato phrases end, a tiny, articulating break distinguishes them from what comes next. Wind and brass players introduce it through tonguing, string players by changing bow. In MuseScore, it is accomplished by shortening the terminal note of the slur or phrase slightly.

With MuseScore 1.x, this used to be a simple matter: the gate off-time of notes could be adjusted from the score. MuseScore 2.x removed the functionality. Its restoration is, with apologies to Shakespeare, devoutly to be wished. Until then, there is a limited workaround.

The articulation gate times in the .mscx file can be exploited to introduce phrase/slur breaks, albeit in a rudimentary fashion. If an unused articulation is chosen from the six and its gate time edited to slightly shorter than full note value, the articulation, hidden, can be added to the final notes of phrases and slurs. Since every instrument in the .mscx file has its own articulations, the gate times can be tailored appropriately for each one.[11]

Figure 49 shows a passage for winds with hidden *marcati*, highlighted in pink, "punctuating" slurred note groupings and phrase ends. At *alla breve* with a tempo of *allegro*, a marcato gate time of "90" (10% shorter than the notated value) would work well. At slower tempi, "95" (5% shorter) or

[11]Gate times apply to the whole staff, not individual channels, so, for example, muted trumpets cannot have a different *staccato* gate time than non-muted.

even "98" (2% shorter) might do the trick. The break should be subtle. In **4/4** time at *andante*, "90" would make it too long.

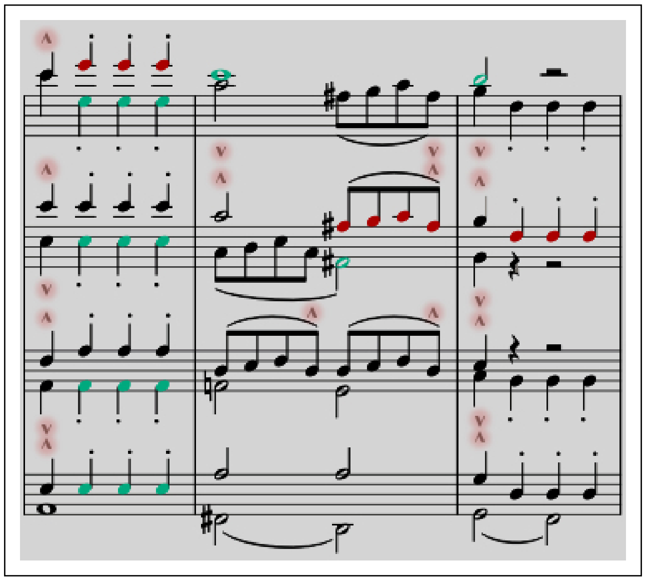


**Fig. 49.** *Hidden articulations delineating phrase/slur ends*

The limitations of the workaround are obvious: since gate time is a percentage of notated value, the break after longer notes (quarters, halfs, etc.) is longer than the break after shorter ones (eighths, sixteenths, etc.). Just the same, the workaround is effective when judiciously applied.

## Accelerando, ritardando

**Tempo markings** can be inserted anywhere in a score and put to use controlling the rate of accelerando, ritardando, and rubato. An alternative method can be used for ritardando, but tempo markings are the only way to create accelerando.

### *Accelerando*

Accelerando is usually indicated in the score by "**Accellerando - - - - -**," but instructions of this type have no associated midi controls and therefore no effect on playback.[12] Hidden tempo markings must be attached to the affected notes to achieve the desired increase in tempo. The procedure is fussy, but not difficult.

1. Select the first affected note.

[12]See the MuseScore Handbook, *Custom lines and properties* for instructions on creating accelerando and ritardando lines.

2. Press **Alt-t** to create a tempo marking.
3. In the Inspector, uncheck **Follow text** to activate the **Tempo** field.
4. Change the BPM in the **Tempo** field to the desired value.
5. Click the tempo marking and select all the text (**Ctrl-a**).
6. Type in the BPM from **Tempo**.
7. Hide the tempo marking.
8. Proceed similarly for the remainder of the accelerando.

Figure 50 demonstrates a short passage for piano solo with an accelerando from 80 bpm (♩= **80**) to 90 bpm.

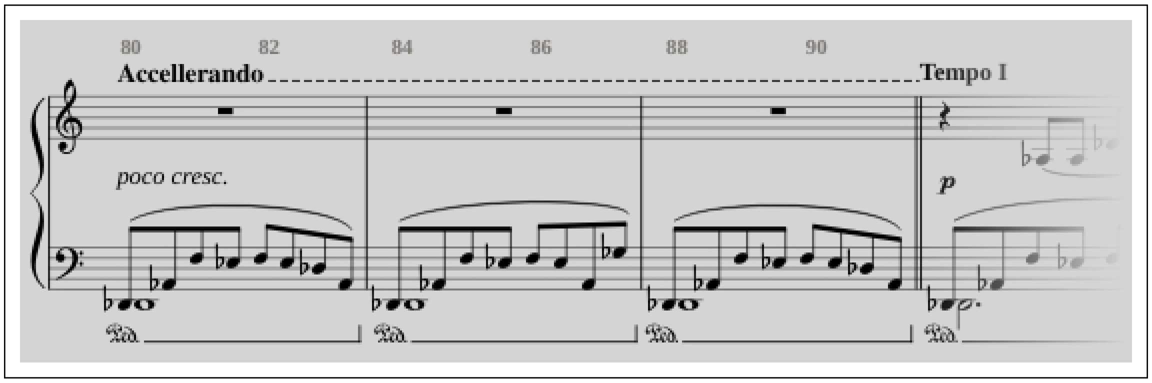


**Fig. 50.** *Hidden tempo markings controlling accelerando*

The regularity of the tempo increase in the example should not be construed as a template for managing *accelerandi*. Even more so than crescendi and diminuendi, gradual changes of tempo almost never proceed at a uniform pace. Getting them right is a hallmark of a good midi orchestration.

### *Ritardando*

Ritardando, like accelerando, can be accomplished with tempo markings, but unlike accelerando, an alternative means is available: the **Time stretch** property of articulations. Non-playing articulations (i.e. with **Play** unchecked in the Inspector) still respect time stretch, so attaching non-playing, hidden articulations with increasing time stretches provides a convenient, fine-grained way to progressively slow the tempo.[13]

***Phrase endings*** The natural tendency for players to relax the tempo slightly toward the ends of expressive phrases is more sensitively handled by hidden, time-stretched articulations than by hidden tempo markings. Even

[13] The articulation need be applied only in one staff; all others are affected.

when a phrase doesn't call for an expressive relaxing of the tempo, players often give the final note a little more time. The tendency is so reflexive that they are often unaware of it. A hidden, time-stretched articulation is the most effective way of introducing it. Similarly, agogic accents are best handled by time stretching, as, of course, are *fermate*.

## Ornaments

Ornaments in the score are respected during playback. The **Ornament Style** dropdown in the Inspector allows choosing between "Default" (starts on the principal note) and "Baroque" (starts on the auxiliary note). While correctly rendered with respect to notes, the ornaments are algorithmically derived and dependent upon tempo for speed of execution. It is sometimes necessary therefore to silence them (**Play** unchecked in the Inspector) and write them out in full. A hidden voice may be used, or staff that has been set up for the purpose and which will be hidden from the printable score.

Figure 51 shows the opening of Bach's *Goldberg Variations*, first as it would appear in the printable score, then with the ornaments written out in full in a hidden voice, and finally with the fully-notated ornaments on a staff that can be hidden from the printable score.[14] The non-playing notes and ornaments are shown in pink. Both of these methods result in correct playback, but, as can be seen, the hidden staff method is clearer and easier to work with.

At faster tempi, ornaments may need to be made louder. If the default execution during playback is satisfactory, a hidden note of equivalent length in another voice *with the same articulation added to it* can be used to increase the volume If not, the ornament must be written out in full and each note's velocity increased.

Depending on the instrument, soundfont, tempo, and dynamic, the initial note of an ornament may need an accent. It can be provided by a hidden note, or by increasing the velocity of the first note when the ornament is written out in full.

[14]The examples correct the mordent in bar one, which MuseScore snaps out too rapidly (sixty-fourths), and the appogiatura E5 in bar two, which MuseScore executes as a quarter note.

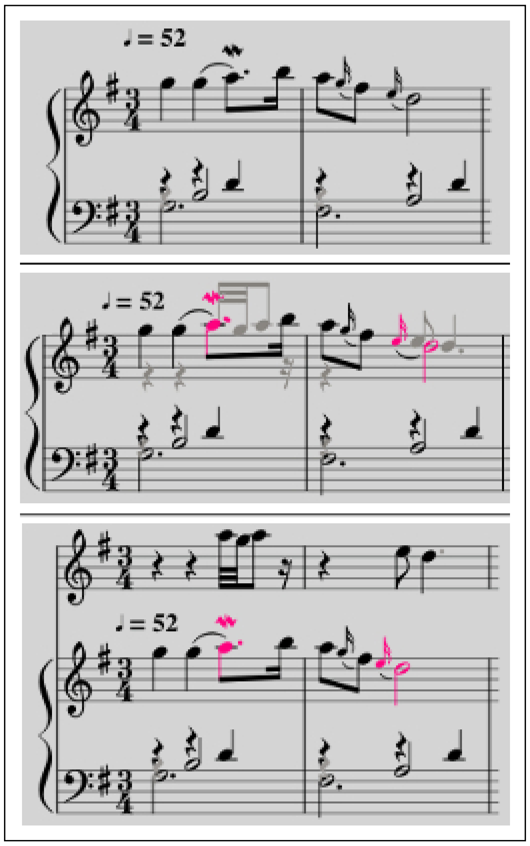


**Fig. 51.** *Ornaments written out in full (hidden voice, hideable staff)*

### *Raising/lowering the auxiliary note*

By default, the auxiliary notes of ornaments are chosen from neighbouring notes in the current key signature unless those notes have been raised or flattened in the same bar prior to the ornament.

Altering auxiliary note(s) when there is no preceding accidental requires putting a hidden note, raised or flattened, coincident with the ornament. Figure 52 shows a double-prall on G5 in C minor. Without a hidden accidental, the sounding upper auxiliary is A♭. With a hidden, non-playing accidental, the auxiliary becomes A♮. Note that had an A♮ appeared in the same bar beforehand, there would be no need to add the hidden one.[15]

### *Long trills*

Like other ornaments, the long trill (*tr*) is dependent upon tempo for speed of execution. This means that rapid long trills at slower tempi are not

[15] Be careful of enharmonics when supplying hidden accidentals to ornaments. MuseScore cannot, for example, play a G-G♯ trill; it must be G-A♭.

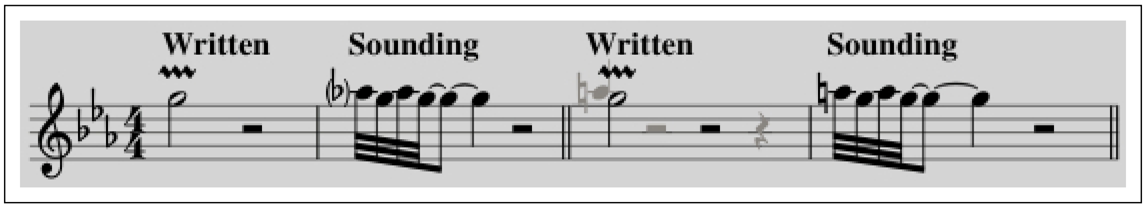


**Fig. 52.** *Double-prall with normal and raised auxiliary note*

possible with MuseScore's default *tr* playback and the trills must be written out in full. Putting them on a hideable staff is preferable to keeping them on the same staff in a hidden voice because the impact on the layout of the final score is less.

Long trills are rarely executed in precise tempo (i.e. as exact thirty-seconds or sixty-fourths), so finding the right note value for long trills involves a little experimentation. For example, a long trill rendered in thirty-seconds may seem too slow, but in sixty-fourths too fast. The solution is to increase the speed of the thirty-seconds by making them triplets. The tuplet groupings will not be audible owing to the speed of the trill and the absence of accents.

***Timpani rolls*** By convention, the *tr* symbol is used to indicate a timpani roll. The symbol needs to be made non-playing and the channel changed to that of the timpani roll soundfont, otherwise MuseScore will play an alternating note trill. The change is indicated by hidden **Staff Text** (**Ctrl-t**), as Figure 53 demonstrates, with the actual switch accomplished in the **Staff Text Properties** dialogue.

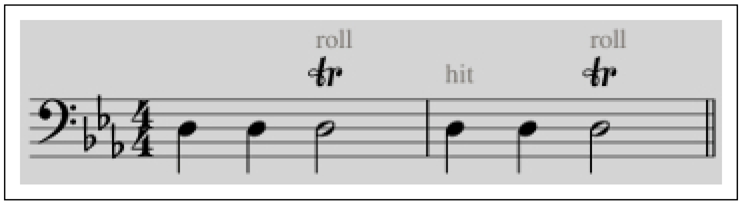


**Fig. 53.** *Timpani rolls with hidden channel changes*

## Arpeggiando

MuseScore treats *arpeggiandi* like ornaments: they start on the beat and are tempo-dependent for speed of execution. Since neither of these is likely to produce the desired result, arpeggiandi must be silenced and written out in full, tied to the principle notes, on a hideable staff or in hidden notes, as seen in Figure 54. In many instances, the vertical squiggle can be dispensed with entirely, since harpists always arpeggiate chords unless otherwise directed. A word of caution: if the square bracket from **Arpeggios & Glissandi** in the Palette is used to indicate *non arp.* in harp parts, the bracket must be silenced since it arpeggiates the enclosed notes.

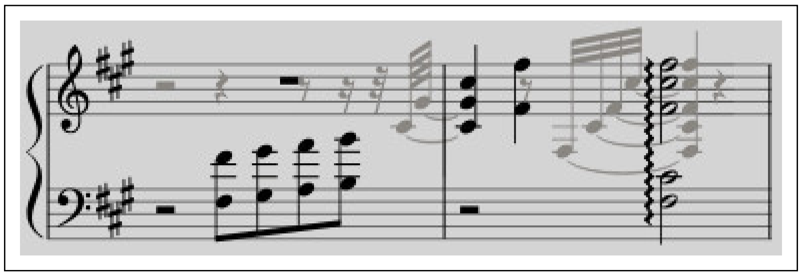


**Fig. 54.** *Harp arpeggiandi*

## Glissando

Continuous glissando (portamento) is not possible with MuseScore, however properly executed stepped glissando is, via **Articulations and ornaments** in the Palette.

## Tremolo

Playback of measured ("barred") tremolo from **Tremolo** in the Palette is executed correctly for both single notes and intervals ("fingered tremolo"). In the case of unmeasured tremolo in the strings, the indication *trem.* should be added as **Staff Text**, rather than using barred notation, and the channel switched to a tremolo strings soundfont.

## Channel switching

Channel switching allows for the easy, temporary substitution or addition of soundfonts on a given staff. Its uses are many:

- switching between *pizz.* and *arco*, or *con sord.* and *ord.*
- switching between instruments sharing a staff, e.g. Trombone III/Bass Trombone
- switching to "loud" (***f***-plus) soundfonts for instruments that display significant timbral differences at different dynamic levels (e.g. "harmony" horns to *blastissimo* horns)
- switching between solo and chorused winds
- layering soundfonts to improve timbre, fullness, or articulation

In all cases, the switch entails adding a text direction to the staff (**Ctrl-t**), since it is only from **Staff text properties** that the switch can be made. The procedure for switching channels is outlined in Applying chorusing to wind departments.

Channel switches are usually accompanied by a visible text direction, e.g. *a due* or *pizz.*, but there are some instances where the text indicating the switch must be hidden, notably the last three items. The first is straightforward—simply hide the text with the channel switch—but the other two require some explanation.

### *Solo vs. section winds, single-staff polyphony*

In the normal course of things, a text indication in the score tells wind instruments sharing a staff whether a single instrument is called for, or more than one (*a due*, *a tre*, etc.). **Staff Text Properties** allows switching to the appropriate channel.

There are times, however, when the indication and its attendant channel switch must be hidden. This occurs when two instruments sharing a staff are playing polyphony and happen to land on the same note. The channel for both voices is, of course, *normal* (both are "solo" lines), but when the parts coincide, the two notes occupying the same position on the staff result in a note played by a single instrument at twice the volume instead of two instruments at the same volume.[16]

If the coincident note is short, silencing one of the parts is a quick solution. Coincident notes of longer duration require momentarily switching to the *sect.* channel in one of the voices and silencing the other. Figure 55 illustrates. At the unison D5 in the first bar, the upper voice is silenced and lower switched to the *sect.* channel. Immediately afterwards, where the oboes part company again, the lower voice is switched back to the *normal* channel.

### *Layering*

It is sometimes desirable to alter the characteristics of an instrument through layering—adding a touch of an additional soundfont. This is frequently the case with strings, where the compromises that determine the

[16]Note that this behaviour is exhibited only when MuseScore is attached to an external sampler (the "virtual orchestra"). It does not occur when MuseScore's internal synthesizer is being used.

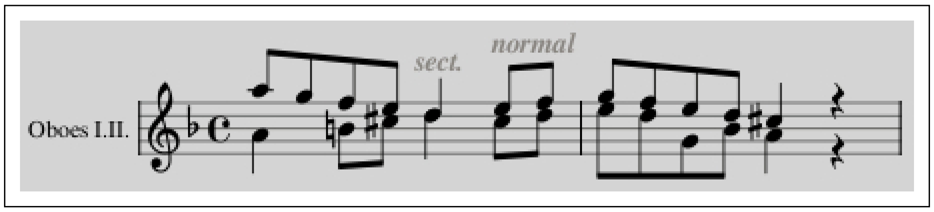


**Fig. 55.** *Hidden switch between* normal *and* sect. *channels*

selection of a general purpose soundfont may result in poor performance in some passages.

For example, if the normal soundfont was selected for smoothness of legato, it may need help in staccato passages, which is accomplished by duplicating the passage in a hidden voice whose channel is set to a brighter soundfont with a stronger attack. Balancing the two requires experimentation (with hidden dynamics or adjusted note velocities), but the effort is usually rewarded by a noticeable improvement in the quality of the staccato.

***Layering in the sampler*** In the case of always needing layered soundfonts for a particular instrument, the effect can be achieved by configuring two LinuxSampler instruments to listen in on the same port and channel and balancing the secondary soundfont in LinuxSampler (rather than Non-mixer). Bear in mind, though, that layering is not a cure-all for inadequate soundfonts and should be implemented with a light touch. A good discussion of the subject is to be found in Westlund's online article *Layering orchestral samples*.

# Conclusion: The weak link

*You can't make a silk purse out of a sow's ear.*

Perhaps not, but you can still make a decent purse.

Those attempting serious midi orchestration on the Linux operating system can be forgiven for hearing the proverb echo in their brains like an ear worm. And it's not just one sow's ear, but two: the generally shoddy quality of freely-available soundfonts on the Web, and the unwieldy complexity of prodding music, not just notes, out of MuseScore's midi playback.

While acknowledging MuseScore's standing as the premier, open source WYSIWYG engraving program, it must admitted that it is the weakest link in the Linux midi orchestration chain. The other components discussed herein are sometimes awkward to work with, but they provide the functionality expected of them, in the manner expected. Not so MuseScore.

In order for an engraving program to serve as the front end for what amounts to an orchestral DAW, it should provide control of all essential midi parameters *from the score*. What's more, it should do so in a consistent manner that does not require more than a modicum of trickery.

The MuseScore team warns that their program is not intended to be a midi front end, an assertion that frankly begs the question: Why does MuseScore ship with a sampler-capable synthesizer, GM soundbank, and partial—some would say incomplete—midi implementation? Halfway along the road to "beautiful scores that perform themselves," why dig your heels in and insist, "That's not what we intended?"

It would not take a great deal for MuseScore to serve as a reasonably complete front end for an orchestral DAW. Most of what's needed is already there. The list of what's missing is short, could be implemented without radical changes to the codebase, and made available via the Inspector.

- **Notes**
  - time stretch
  - gate ontime and offtime

- **Ornaments**
  - adjustable speed (as sixteenths, thirty-seconds, sixty-fourths... and tuplets thereof)
- **Arpeggiando**
  - on the beat or before it
  - adjustable speed (as for ornaments)
- **Glissando, portamento**
  - stepless glissando and portamento
- **Articulations**
  - changeable gate times
- **Hairpins**
  - swell through notes

Some of these may be implemented in the next release (3.0), but it is to be hoped that all of them will make their way quickly into MuseScore. There comes a time in the development of software when a project's very success sometimes requires its scope to be expanded, regardless of the original mandate. MuseScore 2.0 stands on such a cusp.

# Appendix A: Convolution reverb

Two types of reverb are appropriate for midi orchestration: algorithmic and convolution. Algorithmic, upon which the setup in this paper is based, uses a mathematical formula to calculate a virtual reverberant space and determine how sounds within it behave. Convolution reverb uses snapshots (Impulse Responses) of real acoustical spaces, which are added to the input signal. The result is that the signal sounds as it would in the hall where the IRs were recorded. Both have pros and cons, and both have passionate advocates—a clear indication that the choice is personal. Mattias Westlund's *Orchestral Positioning: Choosing a Reverb* is a good introduction to the subject.

Convolution reverb is somewhat simpler to implement in Non-mixer, and makes full use of the ambisonic Spatialization Console for single point Y- and Z-axis panning. The advantages are bought at the cost of installing and setting up **jconvolver**, a recipe for which follows.

1. Install jconvolver. Most Linux distros have the jconvolver package. Debian-based distros can use

   ```
   sudo apt-get install jconvolver
   ```

   If the configuration files are packaged separately, it is not necessary to install them.
2. Retrieve the file 'ambiverb.tar.bz2'.

   ```
   wget http://non.tuxfamily.org/ambiverb.tar.bz2
   ```

3. Change to the $HOME directory and untar ambiverb.tar.bz2.

   ```
   cd
   tar xvjf ambiverb.tar.bz2
   ```

   This creates a directory called ambiverb. Change into it.

   ```
   cd ambiverb
   ```

4. Retrieve the file 'jconvolver-reverbs.tar.bz2'.

   ```
   wget http://kokkinizita.linuxaudio.org/linuxaudio\
     /downloads/jconvolver-reverbs.tar.bz2
   ```

5. Untar jconvolver-reverbs.tar.bz2.

   ```
   tar xvjf jconvolver-reverbs.tar.bz2
   ```

This creates a directory called ‘reverbs’. Change into it.

```
cd reverbs
```

6. Create directories for the reverb (IR) files.[1]

```
mkdir hamilton maeshowe lyddington porihall \
  r1-nuclear-reactor-hall yorkminster
```

7. Retrieve the IR files for each configuration and install in the appropriate directory.

```
cd hamilton
wget http://space-net.org.uk/files/HM2_000_YZ_48k.wav
wget http://space-net.org.uk/files/HM2_000_WX_48k.wav

cd ../lyddington
wget http://space-net.org.uk/files/Lyd3_000_YZ_48k.wav
wget http://space-net.org.uk/files/Lyd3_000_WX_48k.wav

cd ../maeshowe
wget http://space-net.org.uk/files/MH3_000_YZ_48k.wav
wget http://space-net.org.uk/files/MH3_000_WX_48k.wav

cd ../r1-nuclear-reactor-hall
wget \
  http://www.openairlib.net/sites/default/files/\
  auralization/data/audiolab/r1-nuclear-reactor-hall/\
  b-format/r1_bformat-48k.wav

cd ../porihall
wget \
  http://legacy.spa.aalto.fi/projects/poririrs/\
  wavs/sndfld.zip
unzip sndfld.zip

cd ../yorkminster
wget http://space-net.org.uk/files/\
  Minster1_000_YZ_48k.wav
wget http://space-net.org.uk/files/\
  Minster1_000_WX_48k.wav
```

[1]Note that directories for the ‘santa-elisabetta’ and ‘sala-concerti-cdm’ configurations are already installed.

8. Return to the ambiverb directory.

   ```
   cd ~/ambiverb
   ```

9. Open each .conf file and search for the text "**`# Replace by whatever required...`**" Change the path underneath so that jconvolver can find the installed IR files. For example, in 'hamilton.conf', replace the line

   ```
   /cd /audio/reverbs/spacenet
   ```

   with

   ```
   /cd /home/<username>/ambiverb/reverbs/hamilton
   ```

Jconvolver is now ready for use.

## Adding jconvolver to Non-session-manager

The procedure for setting up convolution reverb with jconvolver assumes a working virtual orchestra as described in the body of the paper.

Begin by making a copy of the NSM session holding the virtual orchestra. Give it an appropriate name, e.g. 'orchestra-jconvolv'. Launch NSM and open the new orchestra, which, at this stage, is identical to the original.

Click **Add Client to Session**. Type 'new new' in the **Enter executable name** field. Click **Okay**, which calls up the **NSM Proxy** dialogue. Type 'jconvolver' in the **Executable** field. Ignore the **Config File** field and enter the full path and name of the desired reverb's configuration file in **Arguments** (e.g. /home/<*username*>/ambiverb/hamilton.conf).

Click **Start**, then, in the NSM window, stop jconvolver (click the square box) and restart it (the right-arrow). In future, new reverbs are loaded by clicking **GUI**, changing the name of the configuration file, clicking **Start**, then stopping and restarting jconvolver.

## Setting up Non-mixer to use convolution reverb

Raise the Non-mixer window and remove all the strips with group dsp "Sections"[2] and those with group dsp "Zita-<*section*>." Remove the master Reverb strip as well.

[2]Raising and lowering the gain of entire sections is not recommended when convolution reverb with ambisonics is being used.

Remove the Aux and panner plugins from the department strips and add a **Spatializer** by changing to **Signl** view, right-clicking on the Gain module, and selecting **Insert→Spatializer**.

Finally, add two strips before the final Master strip called "Early" and "Late," both with group dsp "Reverb." Change the number of channels on the Early strip to four, leave Late at one.

## Carla connections

In Carla, the connections from MuseScore to LinuxSampler, Linux-Sampler to the instrument modules, and the instrument modules to the department modules remain the same. New connections must be made from the department modules to "Reverb" and "Master." Each department module has nine outputs, five for the reverb signal and four for the dry.

### *Reverb signal connections*

The reverb outputs are split into early reflections (four outputs) and late reflections (one output). Each department's reverb outputs are connected to the corresponding Reverb module inputs. **late reverb/out-1** goes to **Late/in-1**. **Late/out-1** goes to jconvolver's **In.Tail**. Figure 56 demonstrates with the Flutes department module.

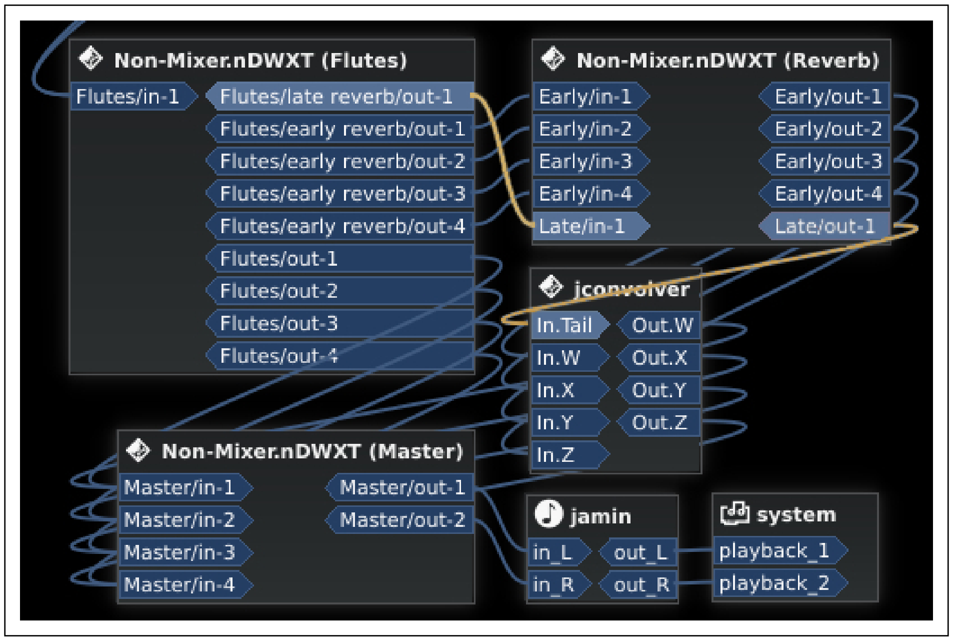


**Fig. 56.** *Late reverb connections*

The **early reverb/out-1...4** outputs are connected to the Reverb module's **Early/in-1...4** inputs. **Early/out-1...4** go to jconvolver's **In.W...Z**, and **Out.W...Z** go to **Master/In-1...4**. Figure 57 shows the complete early reverb signal path.

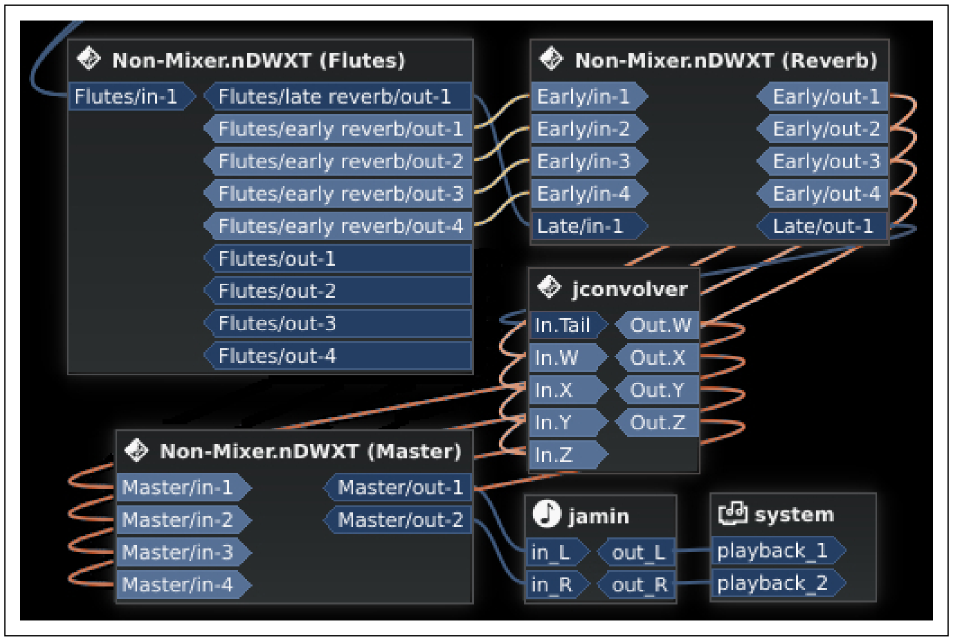


**Fig. 57.** *Early reverb connections*

### *Dry signal connections*

The last set of connections is from the department module's **out-1...4**, the dry signal, to **Master/In-1...4** as seen in Figure 58.

Following the examples shown for Flutes, a finished setup has every departments' outputs similarly connected to "Reverb," "jconvolver," and "Master." The outputs from "Master," through "jamin" and then to "system," are the same for both algorithmic and convolution reverb setups.

## Non-mixer —
## Spatialization console, convolution reverb

### *Spatialization console*

With spatializers on the department strips and jconvolver providing convolution reverb, positioning the departments in Non-mixer is done entirely from the Spatialization console (**F8**). Both left/right and front/back placement are accomplished by moving a single, labelled point. In other words, there is no need to pan and set reverb separately.

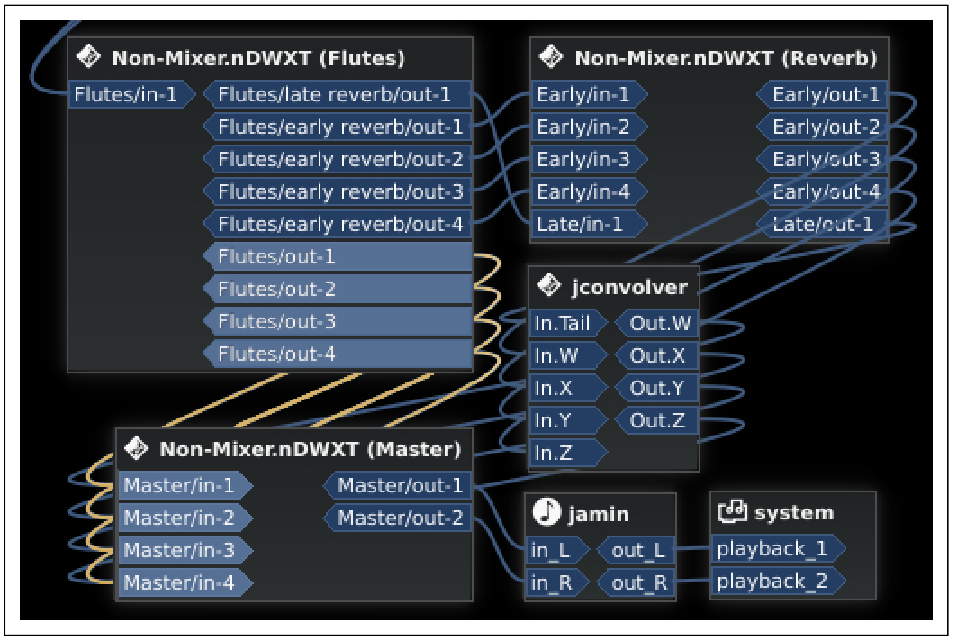


**Fig. 58.** *Dry signal to master*

Since the Z-axis is represented by the console's X-axis, the departments of the orchestra can be positioned visually, approximating what is seen in the seating diagram in Figure 24. Care must be taken, however, to trust the ears more than the eyes.

## *Convolution: early reflections, room size*

Early reflections are the initial echoes created by sounds bouncing off reflective surfaces—walls, ceiling, floors, etc. They typically reach the ears 5 to 100 milliseconds after the direct sound, sometimes before the onset of full reverberation. The loudness of these echoes and the length of time it takes them to arrive are what determine the perception of room size.

The "Early" strip in Non-mixer raises and lowers the gain of early reflections, effectively establishing room size. The higher the gain, the smaller the room; the lower the gain, the larger the room.

## *Convolution: late reflections, reverberance*

Late reflections are the diffuse, blended echoes that persist after the direct sound and early reflections have reached the ears. They determine the overall reverberance of the room—its character, so to speak. Raising the gain of the "Late" strip creates a more reverberant ("echoe-y") space; lowering it makes the space drier.

### *Balancing early and late reflections*

Putting early and late reflections into perspective, when one sings in the shower, one is in a small, highly reverberant space. Such a space has prominent early reflections—they reach the ear almost immediately—and quite a lot of late reflection because the surroundings absorb little sound. On the other hand, if one sings in the nave of a cathedral, the early reflections are much less prominent and the late reflection considerably more in evidence.

Balancing these two extremes within a real venue's acoustic properties can create a variety of reverbs of distinctly different character.

## Algo vs. convolv: pros and cons

Convolution reverb is a good, general solution for panning and reverb. The reverb is realistic—real, in fact—and its integration with Non-mixer's Spatialization console significantly reduces the work involved with orchestral positioning. Its chief liability is that the character of the reverb can only be modified by switching to a new set of impulse response files, i.e. loading a new configuration file in jconvolver.

Algorithmic reverb offers the possibility of fine-grained control over reverb characteristics (delay, decay, equalization, etc.), but requires patience and skill to achieve optimal Y- and Z-axis positioning of orchestral departments.

Neither can be said to be superior to the other in terms of realism. Some have argued that putting sampled instruments in a real hall exacerbates their fakeness. Others argue that it enhances the general realism of the listening experience. On the other side of the fence are those who feel that if the instruments are fake, it's better to create an artificial, tailor-made reverb that mitigates or disguises their digital origins.

In the end, it is the music itself that should dictate the choice. Nothing prevents having two NSM sessions for the virtual orchestra, one with algorithmic reverb and one with convolution.

# Appendix B: Online resources

Orchestration has never been easy to master. Instrumentology—knowing the pitch ranges, timbres, and playing techniques of all the instruments in the orchestra—is, in itself, a large body of learning to acquire, and it's only the beginning.

The following sites represent the best the Web has offer those seeking to learn the art of orchestration.

## Instrumentology

### *Vienna Symphonic Library Academy*

Vienna Symphonic Library is a company that sells proprietary samples with frightening price tags. Some of the profits have been re-invested in the musical community in the form of a free, online instrumentology site that is second to none. It is a one-stop location for learning the basics of any instrument (range, playing techniques), listening to it its different registers, and discovering how it combines with other instruments.

**Vienna Symphonic Library Academy**
*https://www.vsl.co.at/en/Academy/Instrumentology*

## Orchestration

### *The Principles of Orchestration Online*

Northern Sounds, vendors of the reasonably-priced Garritan Personal Orchestra, have also invested in the community by providing a free online course based on Rimsky-Korsakov's *The Principles of Orchestration*. The seminal text is presented in its entirety with commentary and emendations by Professor Alan Belkin. Each chapter is presented as a single lesson, with audio versions of RK's musical examples. Not enough praise can be heaped upon Northern Sounds for putting *Principles* online.

**The Principles of Orchestration Online**
*http://www.northernsounds.com/forum/forumdisplay.php/77-Principles-of-Orchestration-On-line*

Supplementing *Principles* is Belkin's own exemplary site.

**Alan Belkin Music, Orchestration**
*http://alanbelkinmusic.com/site/en/index.php/orchestration*

### *The Idiomatic Orchestra*

Karl Aage Rasmussen and Lasse Laursen's *The Idiomatic Orchestra* takes a thought-provoking look at the art of orchestration, building upon the standard texts—*Principles*, Berlioz's *Treatise on Instrumentation*, Gardner Read's *Thesaurus of Orchestral Devices*, and others.

**The Idiomatic Orchestra**
*http://theidiomaticorchestra.net/introduction*

## Midi orchestration

### *matthiaswestlund.net*

Several of Westlund's articles are referred to throughout this book. His complete series on midi orchestration is the best there is online.

**Mattias Westlund**
*http://mattiaswestlund.net/?page_id=29*

# Appendix C: External links (cited)

The links to external sites in this paper are clickable in a PDF viewer. For printed copies, the following gives the links' URLs along with the page where they appear.

**Pg. 3**: "Cadence documentation"
*http://kxstudio.linuxaudio.org/Documentation:Manual:cadence_introduction*

**Pg. 4**: "KXStudio"
*http://kxstudio.linuxaudio.org*

**Pg. 12**: "LMMS"
*https://lmms.io*

**Pg. 19**: "Non Session Manager User Manual"
*http://non.tuxfamily.org/nsm*

**Pg. 20**: "MuseScore with LinuxSampler"
*https://musescore.org/en/node/21159*

**Pg. 20**: "Configuring LinuxSampler"
*https://musescore.org/en/node/21159#configuring-linuxsampler*

**Pg. 26**: "Non-mixer manual"
*http://non.tuxfamily.org/mixer/doc/MANUAL.html*

**Pg. 31**: "KXStudio site"
*http://kxstudio.linuxaudio.org/Applications:Carla*

**Pg. 46** (footnote): "Swami"
*http://www.swamiproject.org*

**Pg. 46** (footnote): "Polyphone"
*http://polyphone-soundfonts.com/en*

**Pg. 50**: "Orchestral Positioning: Panning"
*http://mattiaswestlund.net/?page_id=254*

**Pg. 51**: "Orchestral Positioning: Reverb in theory"
*http://mattiaswestlund.net/?page_id=440*

**Pg. 55, 57**: "Mastering with JAMIN"
*http://www.penguinproducer.com/Blog/2011/09/mastering-with-jamin*

**Pg. 59**: "Lilypond"
*http://lilypond.org/text-input.html*

**Pg. 59**: "Rosegarden"
*http://www.rosegardenmusic.com*

**Pg. 60**: "MuseScore Handbook"
*https://musescore.org/en/handbook*

**Pg. 60**: "Mastering MuseScore"
*https://musescore.org/en/books/mastering-musescore*

**Pg. 60**: "MuseScore forum"
*https://musescore.org/en/forum/11*

**Pg. 73** (footnote): "Improving Playback from MuseScore 2.0 Piano Scores"
*http://missives-from-the-edge.blogspot.ca/2015/10/*
*improving-playback-from-musescore-2.html*

**Pg. 79**: "MuseScore Handbook, Custom lines and properties"
*https://musescore.org/en/handbook/line#custom-lines*

**Pg. 85**: "Layering orchestral samples"
*http://mattiaswestlund.net/?page_id=642*

**Pg. 87**: "*Orchestral Positioning: Choosing a Reverb*"
*http://mattiaswestlund.net/?page_id=549*

# Licence and colophon

## Colophon

This paper was typeset with groff and the mom macroset using Aldine for the body copy and Optima for titles, headings, and page headers.