Qtractor

Linux Audio & MIDI Multitrack Workstation

User Manual
Version 0.3.0

December, 2008

by

Rui Nuno Capela
James Laco Hines
Stephen Doonan

Table of Contents

1 Introduction

1.1 Abstract

Qtractor is a multi-track Audio and MIDI recorder and editor. The program is written in C++, and for the GUI (graphical user interface) elements, the Qt4 Toolkit and Qt Designer are used. Qtractor is free open-source software, licensed under the GPL, and the project welcomes all collaboration and review from the Linux audio developer and user community in particular, and the public in general.

Currently the Qtractor project has one developer, the originator of the project, Rui Nuno Capela. Development was started April of 2005, initially as a Qt3 application. Since October 2006, it is officially a Qt4 [2] application.

The initial target OS platform is Linux, in which ALSA (Advanced Linux Sound Architecture [4]) and JACK (the Jack Audio Connection Kit [3]) form the supporting infrastructure for recognizing sources of digital audio and MIDI (musical instrument digital interface) data, communicating with those sources and routing the data to and from various locations and programs (applications, including Qtractor) both inside and outside the computer and involving both software and hardware interfaces.

The goal is to develop Qtractor into a more and more full-featured and robust digital audio/MIDI workstation, especially appropriate for personal home recording studio use.

1. *Illustration 1.1: Main GUI window showing audio & MIDI tracks, Mixer & Connections windows*

1.2 Introduction

Although Qtractor will become more and more full-featured as it is developed, it can already be comfortably used by hobbyists as a personal home recording studio or “bedroom studio.” It can record, import, arrange and edit both digital audio and MIDI data. The functionality of Qtractor is contained within a graphical desktop environment that will be familiar to users of other popular multi-track recording/editing applications on any computer operating system, and follows the same design principles with many of the same or similar elements.

In addition to recording digital audio and MIDI, Qtractor provides an environment for multi-track clip-oriented composing techniques common in modern music-making and aims to be intuitive and easy to use, yet powerful enough for the serious recording enthusiast.

Note: Qtractor is not what is known as a “tracker” type of audio/MIDI application, although it has the potential to function in that way if needed.

When used merely as an audio and/or MIDI recorder (a MIDI recorder was historically called a “sequencer”) or arranger, Qtractor is non-destructive, which means that the underlying files that contain the audio or MIDI data are not altered when those files are apparently cut into pieces, duplicated, pulled or pasted into a different order in time, or manipulated in any number of ways within the main Window (GUI interface) of Qtractor. However, when used as an audio or MIDI recorder, for example, or when editing previously recorded MIDI data in the dedicated MIDI editor, Qtractor’s actions can be destructive in the sense that newly recorded data (or altered MIDI data) replaces previously recorded data on the same track.

2 Installing and Configuring Qtractor

2.1 About Compiling Qtractor from its Source Code

If Qtractor is not available as a package for your particular type of Linux (a .deb package for Debian, Ubuntu or other Debian-based system or an .rpm package for Red Hat, Fedora, SUSE, etc.), then it must be compiled into an executable application from its source code (from the C++ programming code in which Qtractor is written) before it can be installed. In that case, one’s computer must have an appropriate compiler program installed (such as G++, the GNU C++ compiler) in order to compile the C++ source code of Qtractor.

For those who have experience compiling programs, the following preparation and instructions based on *autoconf* will be familiar. The process is fairly easy and straightforward, but both the process and the “build environment” (a collection of programs necessary for compiling) can be confusing for those who are not experienced, so those persons may wish to learn a little about the process first. There are many books and online resources such as Linux forums, email lists, wikis that can be consulted and used to learn how to compile and install programs’ source code.

2.2 Preparation—Required and Optional Prerequisites

In order to compile the source code of Qtractor to create an executable program, as well as to run Qtractor, some software must be installed (the mandatory software listed below) and some may be installed if the user wishes to enhance the abilities of Qtractor (the optional support libraries).

2.2.1 Mandatory Software

- **Qt 4** (core, gui, xml) - C++ class library and tools for cross-platform development and internationalization
  http://www.trolltech.org/products/qt/
- **JACK** Audio Connection Kit
  `http://jackaudio.org/`
- **ALSA** - Advanced Linux Sound Architecture
  `http://www.alsa-project.org/`
- **libsndfile** - C library for reading and writing files containing sampled sound
  `http://www.mega-nerd.com/libsndfile/`
- **LADSPA**- Linux Audio Developer's Simple Plugin API
  `http://www.ladspa.org/`

2.2.2 Optional Support Libraries (at build time)

If the functionality that these additional software libraries provides is desired for Qtractor, they must be installed before Qtractor itself is compiled from its source code.

- **libvorbis** (enc, file) - Ogg Vorbis audio compression
  `http://xiph.org/vorbis/`
- **libmad** - High-quality MPEG audio decoder
  `http://www.underbit.com/products/mad/`
- **libsamplerate** - The secret rabbit code, C library for audio sample rate conversion
  `http://breakfastquay.com/rubberband/`

- **librubberband** - Rubber Band Audio Time Stretcher, an audio time-stretching and pitch-shifting library
  http://breakfastquay.com/rubberband/
- **liblo -** Lightweight OSC implementation (needed for DSSI GUI support)
  http://liblo.sourceforge.net/
- **DSSI** - An API for soft synth plugins with custom user interfaces
  http://dssi.sourceforge.net/
- **VST-SDK** - Steinberg's Virtual Studio Technology
  http://www.steinberg.net/

2.3 Downloading Qtractor

2.3.1 Qtractor for Everyone

Qtractor is still in its alpha stages of development, but is already fully functional. The latest versions are publicly available from the qtractor.sourceforge.net project web site [1]:

```
http://qtractor.sourceforge.net/
```

2.3.2 Qtractor for the Experienced and Adventurous

The “bleeding-edge” source code may be found in the CVS repository, through anonymous (pserver) access with the following instructions:

At the command line (in a text terminal or terminal window) login to the CVS repository:

```
cvs -d:pserver:anonymous@qtractor.cvs.sourceforge.net:/cvsroot/qtractor login
```

When prompted for a password, hit enter and proceed for check-out (all in the same line):

```
cvs -z3 -d:pserver:anonymous@qtractor.cvs.sourceforge.net:/cvsroot/qtractor co
qtractor
```

Prepare the configure script on the just created qtractor source tree directory:

```
cd qtractor
```

```
make -f Makefile.cvs
```

Hopefully, the source tree will be now ready for build and installation.

2.4 Compiling and Installing Qtractor

2.4.1 Standard Compiling and Installation

After downloading Qtractor and decompressing and extracting the archive if necessary (with the applications gzip and tar, for example), change directory in a command-line terminal to the resultant qtractor directory. Once inside the qtractor directory, type and enter the following command:

```
./configure && make
```

NOTE: To see all configuration options before entering the command sequence above, type

```
./configure –help
```

After typing the *configure* and *make* commands and waiting until the program has finished being compiled, become an administrator of your system (using either the sudo or su command to become, temporarily, the user “root”) and finish the installation by entering the following command:

```
make install
```

which will copy the qtractor binary executable (the Qtractor application or program) and associated desktop and icon files to common standard system locations.

2.4.2 Compiling for Native Linux VST Support (optional)

VST is a “Virtual Studio Technology” developed by Steinberg Media Technologies GmbH for their own proprietary audio and MIDI applications. VST support is not very easy to accomplish and is suggested only for experienced users. Because of licensing issues for this proprietary software, one must download the VST SDK (software developer’s kit) from the Steinberg Media Technologies GmbH website, specifically searching in the

third-party developers section. It doesn't matter whether you choose version 2.3 or 2.4 of VST, but choose one and only one. *Do not use VST 3.0. It will not work.*

In order to download the VST SDK zip-archive you will have to accept the license and supply some personal data, then download and unpack (de-compress and/or de-archive) the pertinent program header files, which are found in one of the directories listed below.

VST SDK 2.3:

Directory:

```
vstsdk2.3/source/common/
```

Files:

```
aeffectx.h
AEffect.h
```

VST SDK 2.4:

Directory:

```
vstsdk2.4/pluginterfaces/vst2.x/
```

Files:

```
aeffectx.h
aeffect.h
```

Just copy the two files to somewhere else in your computer's directory structure. It is recommended that you copy those files into a standard "include" directory (eg. /usr/local/include or /usr/include), in which case all will be handled "automagically" by the ./configure build step. Otherwise you'll need do supply the path yourself, as in:

```
./configure –with-vst=/path/to/vstsdk2.x/include
```

Once Qtractor is properly compiled, you will probably want to download some native VST plugins. But Qtractor must be told where it can find these VST plugins. To accomplish this, currently you'll need to set (create and assign a value to) a variable known as an "environment variable." The variable must be named VST_PATH and the value assigned to this environment variable must be the path (location on the computer) where the VST plugins have been placed. You can do this by entering the command (in a command-line terminal or terminal window):

```
export VST_PATH=/path/to/vst_plugins
```

Some ready made Linux VST plug-ins can be found on the following web sites:

```
http://www.linux-vst.com/
http://www.linux-vst.com/
http://cern.linux.vst.googlepages.com/home
```

2.4.3 Compiling Qtractor with Debugging Code Included

Although Qtractor is a mostly stable program, there could be problems that eventually show up. This short guide will explain to you how to build Qtractor with debugging code built in, making it easier to locate where the code exhibits problems.

Rebuild it all from scratch, with:

```
./configure --enable-debug && make
```

Enable core dumps in a shell session:

```
ulimit -c unlimited
```

From the same shell command line, run the program until it crashes. You'll see something like this in the output when it happens:

```
Segmentation fault (core dumped)
```

Locate the dumped core file. Depending on your environmental settings it might be just named core or something like core.1234 (1234 is the process-id number of the crashing program) located on the last directory the program was current.

Load the core dump file into gdb:

```
gdb ./qtractor /path/to/core
```

At the gdb prompt just enter:

```
gdb> bt
```

or:

```
gdb> thread apply all bt
```

2.5 Qtractor's Configuration Settings File

Qtractor keeps a separate set of its run-time settings and configuration state (a "memory" of certain settings) for each person who uses Qtractor, in a file located in the user's home directory as in the following example (on your computer the word "user" in the example would be replaced with your own user name):

```
/home/user/.config/rncbc.org/Qtractor.conf
```

Normally, there is no need to edit this file because it is recreated and rewritten every time Qtractor is run.

2.6 Audio and MIDI Input

Qtractor can record both digital audio and MIDI data, but it does not know what audio or MIDI data you wish to record nor does Qtractor automatically make any connections to sources for that data. Instead, you must route audio and MIDI data to Qtractor manually, like building a pipeline from the source of the audio or MIDI to Qtractor. Qtractor includes a utility for doing this: the Connections window, pictured below. But Qtractor and its Connections utility depend upon the supporting software infrastructure mentioned previously, composed of ALSA and Jack, in order to both recognize sources of data and to receive data from those sources. ALSA is responsible for knowing about and communicating with audio and MIDI hardware and some software, and Jack is used for routing audio and MIDI data to and from various hardware and software "ports" within the computer or attached to it. Both ALSA and Jack, working together, make it possible to route audio and MIDI data to Qtractor. You simply must remember to connect at least one source of that data to Qtractor first, before you can record some of that data in Qtractor.

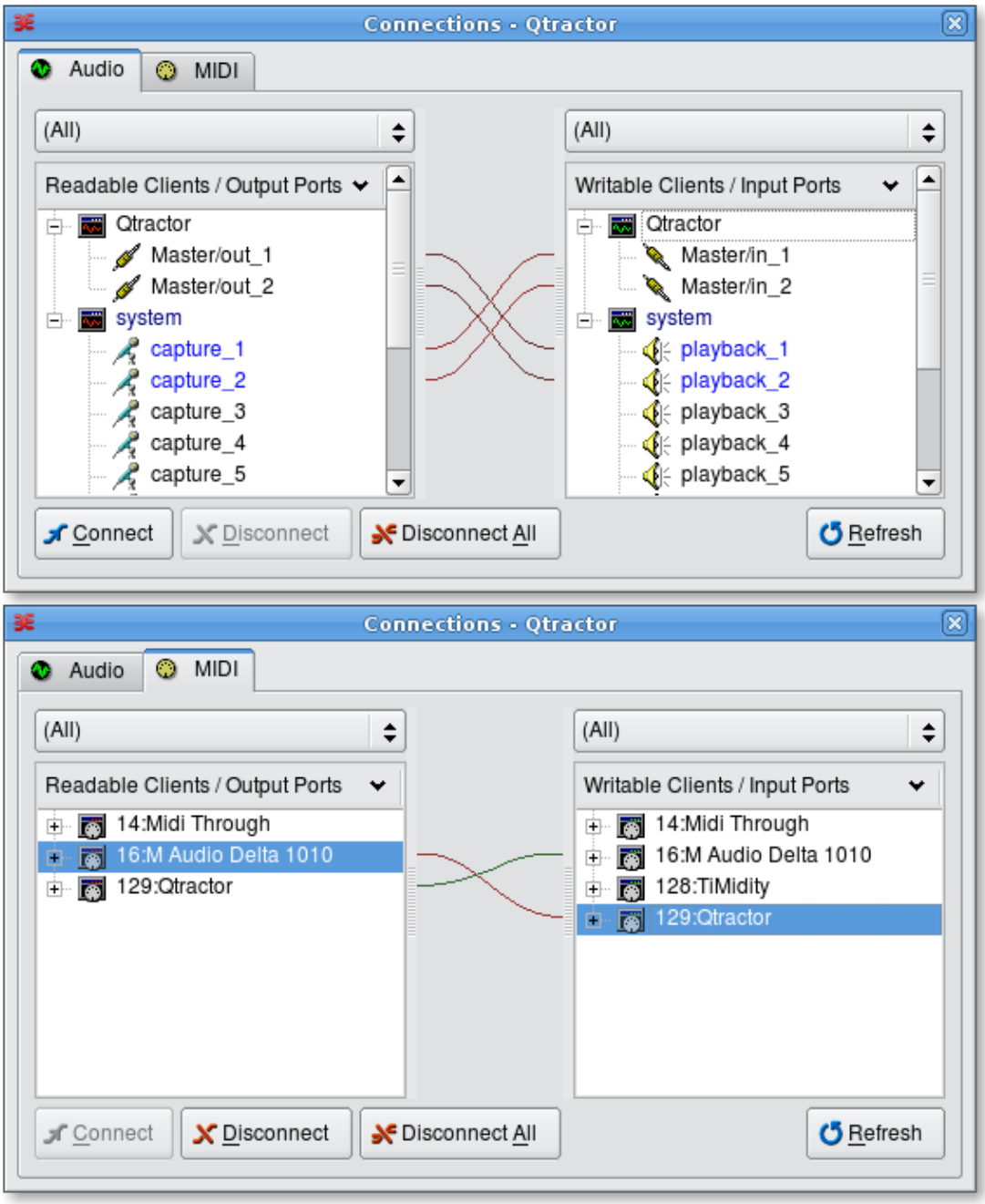


Illustration 2.1: Qtractor's Connections window, showing both the Audio and MIDI tabs; "readable" ports are sources of data (where audio or MIDI data can come from) while "writable" ports are places that data can be routed to (sent to).

3 Learning Qtractor—An Example Session

This chapter is written for those who may not be very familiar with digital audio recording or MIDI "sequencing" applications, or who wish to gain a quick overview of how Qtractor works and can be used before exploring the program in greater depth and learning its features in detail. It describes an example Qtractor session and serves as a walk-through of the program. The reader can follow the writer as he creates a Qtractor session and uses Qtractor to record, import and edit MIDI and audio data.

3.1 Preparation

You'll use Qtractor to record or import several tracks of MIDI and audio data. A MIDI-triggered tone generator (in this case, a rack-mounted tone generator outside the computer, although it could just as easily be a "soft synth" inside the computer) will produce the sound for the MIDI parts as Qtractor plays them back. The audio tracks will be either recorded from some external source, or pre-recorded audio files will be imported into, played back and edited in Qtractor. A final "mix down" audio track in Qtractor will contain the combined audio result of playing back and simultaneously recording the other audio and MIDI tracks.

3.2 Importing an Audio File

The first thing you would like to do is to import a pre-recorded audio file of drums and other percussion, to form the basis of the rest of the recording session.

3.3 Connecting the MIDI data source to Qtractor

Your MIDI piano-like keyboard normally routes its MIDI data (created when you strike the keys, for example) to its own internal tone generator, which then sounds like you're playing a real piano, or electric piano, harpsichord, bass guitar, etc. However, for this project in Qtractor you want to route the external keyboard's MIDI data to Qtractor. So, using a standard MIDI cable you connect the keyboard's MIDI output to the MIDI input of your sound card, and inside the computer you will route the MIDI data from the sound card to a Qtractor MIDI input bus.

After connecting the MIDI cables, launch Qtractor, which also launches the JACK daemon (jackd, part of the Jack Audio Connection Kit mentioned previously) if jackd was not already running. Then open Qtractor's Connections window, pictured in Illustration 2.1, page 9. In the Connections window MIDI tab, connect the MIDI output of the sound card (in the left pane, marked "Readable Clients / Output Ports") to Qtractor's listing in the right pane of the window (marked "Writable Clients / Input Ports"). One can connect an output port (the source of the data, MIDI or audio) to an input port (the port that will receive the data), in several ways: one is to highlight a port in the list in the left pane, highlight a port in the right pane, then right-click (third-button-click if a person is left handed and uses a mouse in a reverse configuration) and from the pop-up menu select "Connect;" another way is to click a port on the left side and with the mouse button held down, "drag" the mouse to a port in the list at the right until the port is highlighted, then release the mouse button. Using either method, a line representing a "virtual cable" will appear between the two ports in the middle section of the window.

Then close Qtractor's Connections window by clicking its button near the top of Qtractor's main window and workspace.

3.4 Creating a MIDI track

Now that Qtractor can receive MIDI data, it's time to create the first track in order to record that data. Right-click in the blank pane at the left in Qtractor's main window and from the popup menu choose "Add track...," (or you could choose the menu item **Track** → **Add Track**...).

Qtractor's Track Properties window opens.

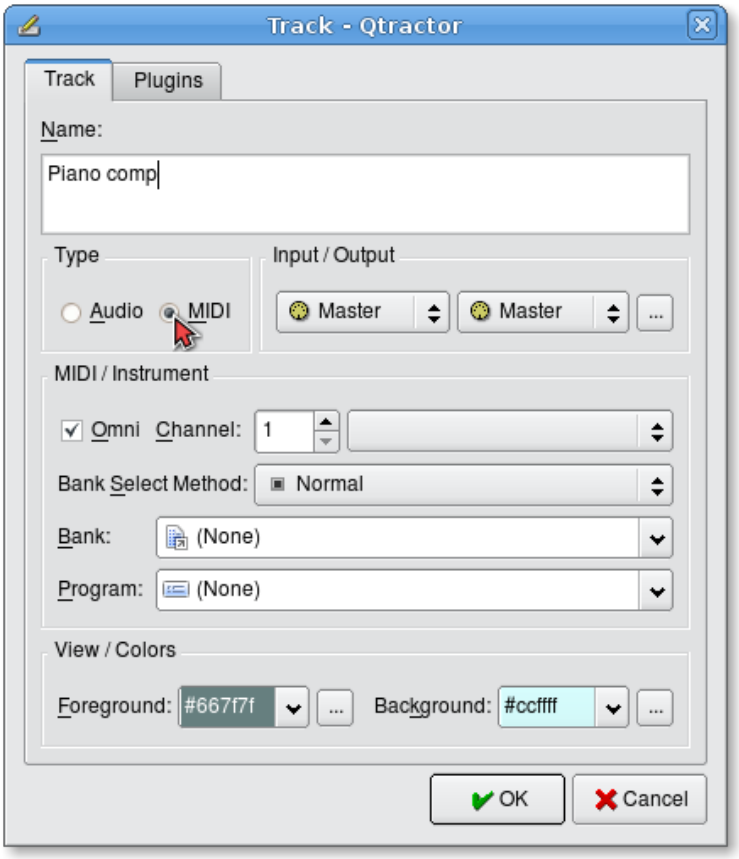
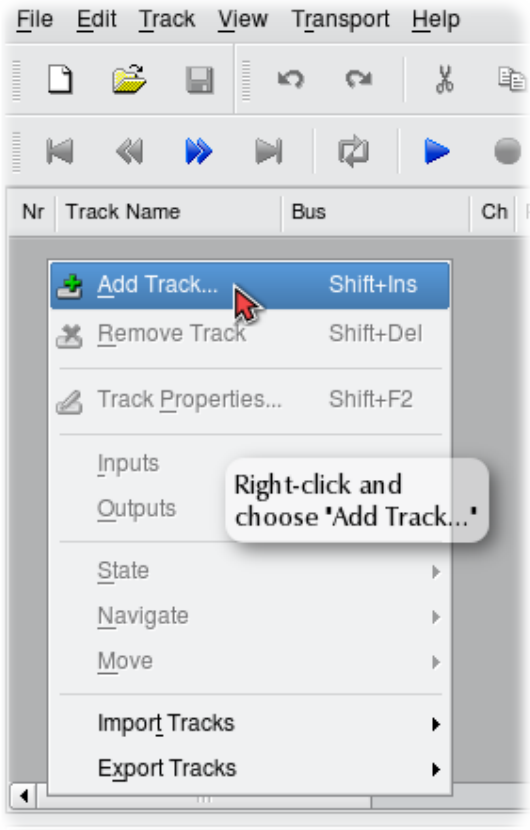


Illustration 3.1: Track Properties window



In the Track Properties window, click the MIDI radio button: you want this track to be used for MIDI rather than audio data. As soon as you click the MIDI button, Qtractor automatically connects the track to Qtractor's master MIDI input and output busses. These can be changed by using the drop-down lists and the ellipsis button in the Input/Output area of the Track Properties window, but there is no reason right now to do so. You can replace the default name "Track 1" with, for instance, "Piano comp."

Leave the MIDI Channel at the default "1," although this really makes no difference at the moment because this channel designation is mainly for output. That is to say, when the track is played back and its MIDI data is sent to somewhere outside Qtractor (like to a tone generator or synthesizer) that data will be sent on the MIDI channel specified here (there are 16 possible channels). But you're going to record now, so the eventual output MIDI channel doesn't matter and can easily be changed later, after you have recorded all the MIDI tracks you wish to record, and before you play them back.

What *does* matter however is the **Omni** checkbox. This setting determines whether Qtractor will record MIDI information received on *any* MIDI channel (omni means all), or whether (if *un*checked) will record only the MIDI data it receives on the same MIDI channel number as the output channel number displayed just to the right of the Omni checkbox. Your MIDI keyboard might be set to transmit MIDI data on only one channel (e.g., MIDI channel 1), so you have to make sure that each of the MIDI tracks you plan to record will receive data from any MIDI channel (which will of course include the MIDI channel your MIDI keyboard is transmitting on) by selecting the Omni checkbox, despite the fact that all MIDI tracks will later be set to *send* their MIDI data each on a different channel (channels 1, 2, 3, etc.). Click the "OK" button and a new empty track appears in Qtractor's main window.

Now is a good time to save this new session, so choose the menu item **File** → **Save...**. This opens the Session Properties window, Illustration 4.4, page 17. In the Session tab of the Session Properties window, give the project (session) a name (Qtractor will append the filename extension .qtr (or .qts) to this name to indicate a Qtractor session file), and you may choose to create a directory for Qtractor to store the MIDI and audio files that will become a part of this project or session. Give the directory you create a name similar to the session name.

Now that the session has been saved, open the Session Properties window again using the menu item **File → Properties.** This time click the Properties tab of the window and set both the time signature (roughly, how many beats per measure) and the tempo so that you can use the metronome to help you keep the MIDI parts synchronized as you record them.

You need to arm or set the track ready for recording, by clicking on the “record” button for that track. That button is the “R” button that is present on every track strip, besides the other “M” and “S”, respectively for “Mute” and “Solo” track state settings. You can also set the record ready state for the current selected track by checking the menu item **Track → State → Record**. Either way the “R” button will turn red and the track is then set armed and ready for recording.

Now it's time to turn on session recording mode. This is accomplished by clicking on the Record button in the main transport tool-bar (big red circle). You can do the same action through the menu item **Transport → Record**.

Now you're ready to record your first MIDI track. Take a little breath and... press “Play”, the space-bar or **Transport → Play** menu item: you're rolling. Hit the MIDI keyboard and see a brand new MIDI clip taking shape while you're performing.

When done, press “Play” again to stop. Save your session and enjoy.

4 Qtractor—An Overview

4.1 Routing—Connections, Ports, Tracks and Buses

4.1.1 Routing—General Concepts and Information

Qtractor can record and play digital data, specifically digital audio and MIDI data. To record, Qtractor must *get* the audio or MIDI data from somewhere, and to play that audio or MIDI data Qtractor must *send* it to somewhere that is capable of understanding an audio or MIDI data stream and producing (for example) some sound. The process of directing such digital data to and from—and through—certain software and hardware is called routing, which is simply choosing the route that the data will take. Not much routing is done automatically; most routing is left to the user, because it is the user's preferences, desires and goals in any specific project that in large part determine how the audio and MIDI data should be routed.

In order to route data, one must have a knowledge and understanding of the various connections and pathways that are available and through which data can flow or be deposited. A *bus* is like a pipeline through which data can travel. Often, more than one stream of data can flow through a single bus, side by side so to speak. A *port* is like a valve at one or both ends of a bus. The valve is normally closed, not allowing any data to flow through unless a connection, such as another bus, is made to the other side of the valve. In that case the valve (port) opens to allow data to pass through it from one bus or pipeline into another. However, simply because a connection is made and the port is open doesn't mean that data is flowing through the available port and buses. The data flow in either direction is initiated by the software or hardware at one or both of the ends of the buses. Often, the data in buses can travel in only one direction. Sometimes it can travel in both directions (duplex mode).

Qtractor has input buses (both audio and MIDI) to which "data streams" from other software or hardware can be connected, and output buses which can be connected to the ports and buses of other software or hardware both inside and outside the computer, using routing procedures explained in this manual.

A track can be thought of as a place where digital audio or MIDI data is deposited rather than a pathway through which such data moves. The data is placed into a track by a recording or paste or import procedure, for example. Despite this, tracks are a part of the signal or data flow, the pathway that audio or MIDI data can take through Qtractor and out of Qtractor, via a bus, to other software or to a hardware port on the sound card installed into one's computer, for example. The reason that a track is considered part of the signal or data flow is that a track can be examined (by Qtractor) in a sequential fashion to "pluck" a *copy* of the data from the track and send that data to one or more buses and through those buses to other destinations. During recording, a track becomes the *destination* for audio or MIDI data coming into Qtractor's input bus(es). During playback, a track becomes the *source* of audio or MIDI data that is then sent through a Qtractor output bus to a destination such as to the sound card and its attached speakers.

Routing is very flexible and therefore signal/data flow can become very complex. Data streams can merge and flow through the same bus for a while, or split and go through separate buses to different destinations, or even through different buses to the same destination, meanwhile passing through separate and different intermediate software or hardware environments in which the data is possibly manipulated or altered in some way. Because of this complexity it is beneficial to develop a solid understanding of the issues related to routing and the many possible routes or pathways that data can take, and the many possible connections that can be made, all at the user's discretion.

Below is a representation of the connections (ports) buses and routes available in Qtractor through which audio or MIDI data can flow.

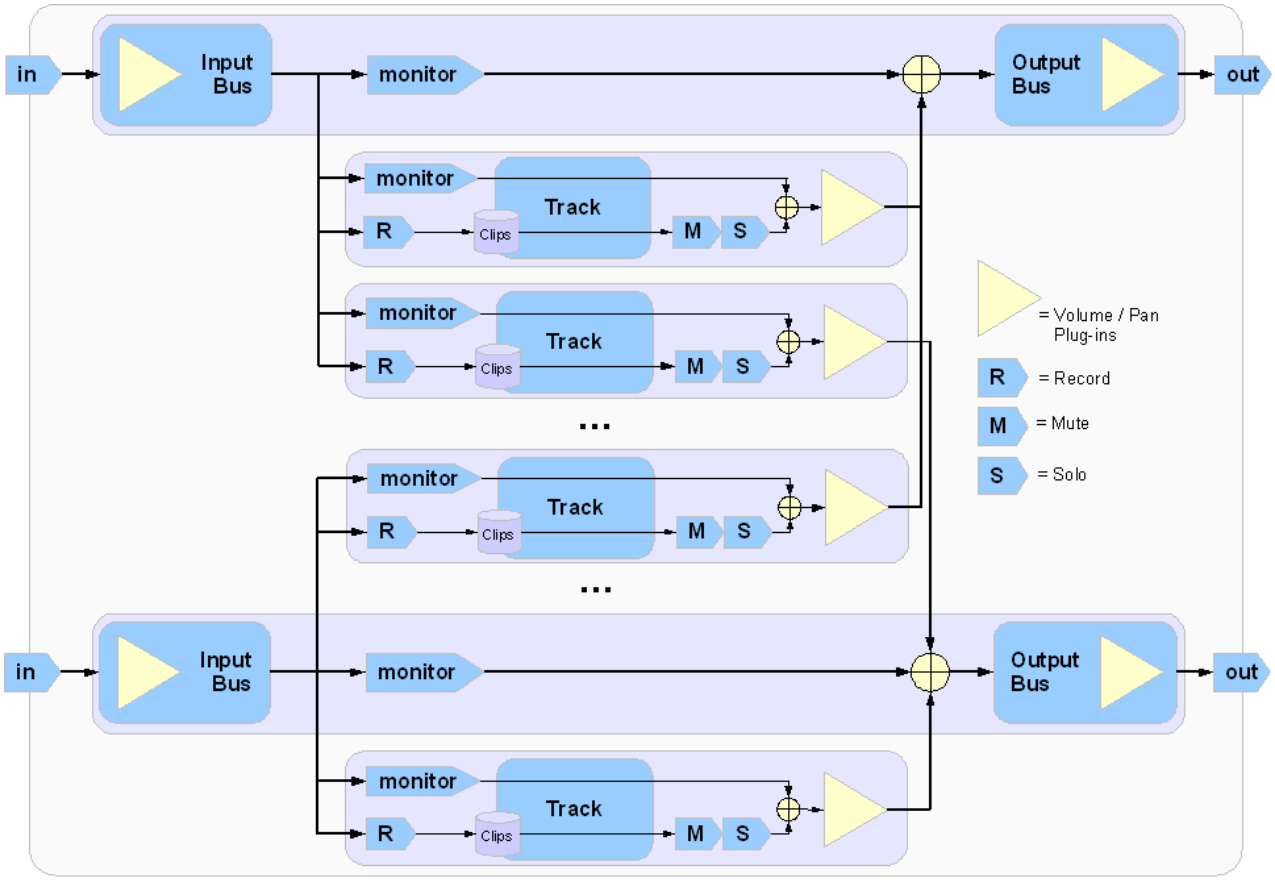


Illustration 4.1: Audio & MIDI data flow, in, through and out of Qtractor

4.1.2 Routing in Qtractor

Routing in Qtractor is accomplished in several ways including:

- using the Connections window (Illustration 2.1) to connect Qtractor's input and output buses to outside *sources* (for "reading" data in order to record it, for example) and *destinations* (for "writing" data to an outside destination in order to play back and listen to it, for example);
- in the Track Properties window (reference to illustration goes here...) where input and output buses are assigned to each track;
- changing the "state" of each track (their are four states, explained in detail later: record, mute, solo and monitor, some of which are mutually exclusive and others which are not);
- using the buttons "R" (record), "M" (mute) and "S" (solo) which are visible within each track's entry in Qtractor's main window;
- adding plugins to tracks or buses in the Mixer window or **View** → **Buses** window.

4.1.3 Routing—Technical Notes

Qtractor is a fairly massive multi-threaded application. For instance, each audio clip has a dedicated disk I/O executive thread, which synchronizes with the master engine and, for all purposes, to central JACK real-time audio processing cycle, through a lock-free ring-buffer. These audio file ring-buffers are recycled (filled/emptied) at one second threshold, and has a maximum streaming capacity of 4-5 seconds of audio sample data. Smaller clips are permanently cached in a RAM buffer.

Audio thread scheduling is mastered and mandated through the JACK callback API model. MIDI clip- events are queued in anticipation through one MIDI output thread, which feeds a ALSA sequencer queue, synchronized on one second periods to the JACK process cycle. A single thread is responsible for listening (polling) for MIDI input, and multiplexes all incoming events through record-armed MIDI tracks. Time stamping is done through the ALSA sequencer facility.

Looping is made possible through the audio file buffering layer, right at the disk I/O thread context. The same consideration is adopted for MIDI output queuing. JACK transport support is not an option, as playback positioning is constantly kept in soft-chase fashion. Audio frame relocation is accounted from successive JACK client process cycles (i.e. buffer-period resolution).

On this particular design, JACK and ALSA sequencer ports are logically aggregated as buses with respect to the audio and MIDI signal routing paths, functioning as fundamental device interfaces. Input buses, through exposing their respective input ports, are responsible inlets on capture and recording. Output buses are the main signal outlets and are responsible as playback and, more importantly, as mix-down devices.

Buses are independently assigned to tracks. Each track is assigned to one input bus for recording, and to one output bus for playback and mix-down. The assigned output bus determines the number of channels the track supports. Clips bounded to disparate multichannel audio files, for which their number of channels do not match with proper bus/track's one, are automatically resolved on mix-down. Illustration 4.1, page 14 shows one typical signal flow block diagram.

4.2 Qtractor's Main Window and Work Area

Qtractor's graphical user interface follows a standard design of most modern digital audio and MIDI workstations. The interface is easy and intuitive enough to easily interact with in order to discover the potential of the underlying inner core of the application where its functionality is implemented.

The following illustration shows an overall view of the GUI with an example session loaded into the

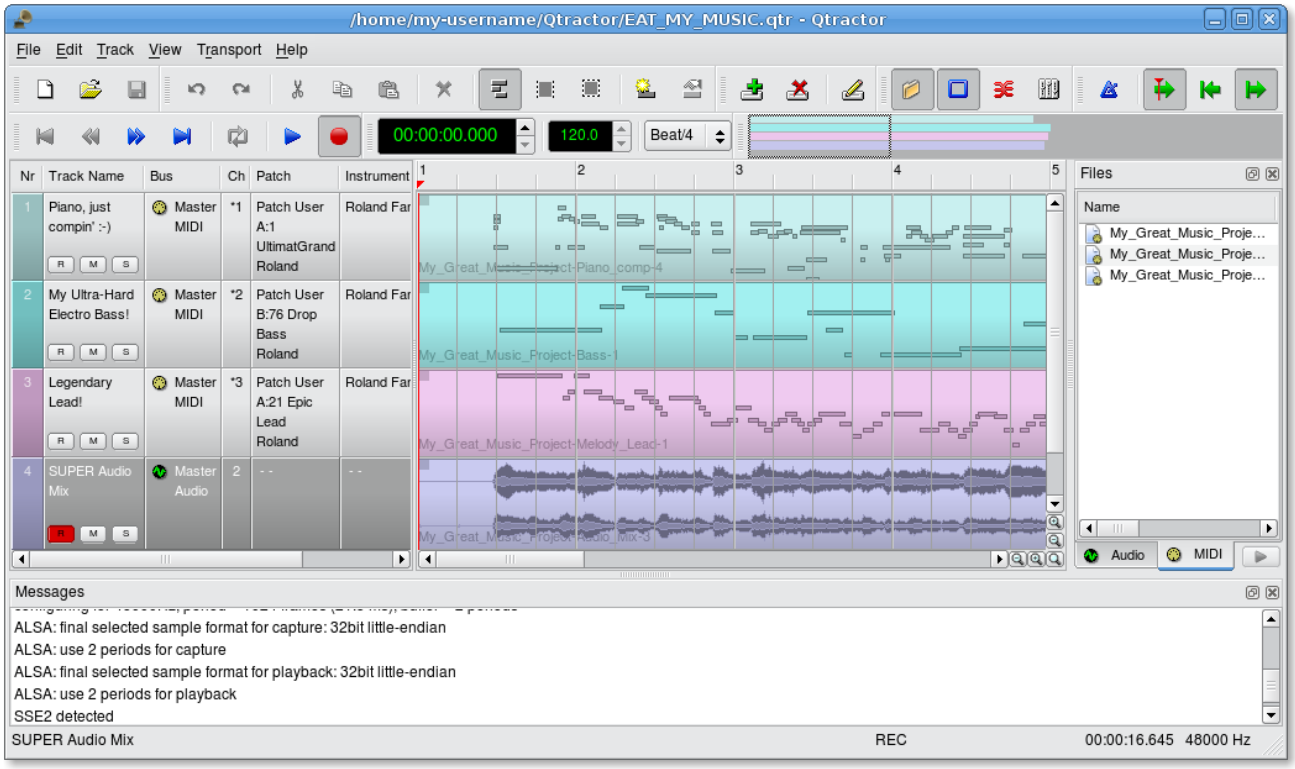


Illustration 4.2: Qtractor's main window and work area

workspace.

The main Qtactor window is initially laid out in this fashion:

- Menu and Tool-bars at the top
- Track list and information at the left
- Track (audio and MIDI) data view representation and file list on the right, with time scale above.

The track data view section in the middle is where most of the action takes place. It contains visual displays and representations of audio waveforms or MIDI data. This section is used for editing Clip objects (portions or all of any particular audio or MIDI file, recorded or imported) and for navigation within the project or "session."

Qtractor also has other useful windows, such as the Mixer window, and the Connections window. Both these windows can be opened using the F8 key for Connections window (the patch-bay), and the F9 key for the Mixer window. These items are also selectable from the View menu.

Two utility windows are additionally featured: the *Messages* window, specially suited for debugging, and the *Files* window, where audio and MIDI files are organized and selected on demand.

Dialog windows for editing *session*, *track* and *clip* properties are also accessible in their proper context, which will be discussed in their respective sections.

Finally, session and application configuration options are assisted through respective customizing dialogs: *Buses, Instruments and Options*, available from the View menu in the main menu bar.

4.3 Understanding a Qtractor Session (recording or editing)

A Qtractor session project contains all the information about all your Clip Objects, placement of Clip Objects, Mixer setup, plug-ins, tempo, time signature and Connections patch-bay. When creating or saving a project, all this, and any related settings are saved on your hard disk within this session project file.

4.3.1 Session Audio Sample Rate

It is important to note that Qtractor sessions are locked to a session project sample rate. This is dependent on the sample rate of the JACK [3] server running at the time the session is created.

Any attempt to convert non-matching sample-rate sessions will result in a recommendation warning message.

However, individual audio clip files are automatically converted on playback in real-time to the host sample-rate (via libsamplerate [8]). This method, while it works very well, is not the recommended method due to possible errors in the real-time sample rate conversion. Real-time sample rate conversion is also going to use quite a bit more valuable CPU resources.

Rui Nuno Capela is working toward eliminating this shortcoming by taking control of JACK from within Qtractor, and restarting it using the session's project parameters. This will ultimately reconnect any plug-ins, set the proper sample rate, etc. Until this feature is available, please follow the recommendations listed above.

4.3.2 Session Properties including Time Signature and Tempo

To access the session properties, choose the **File** / **Properties** menu item. Here, you can name your Qtractor session, set the tempo, time signature (how many beats per bar) and decide how many "ticks" (the smallest time unit in a session) will be within the time span of each beat.

Although you may select any time signature and tempo for your session, by present design Sessions can only contain a single constant tempo. Tempo must be regarded as a global setting of a session. Qtractor does not presently support tempo mapping, but may eventually do so.

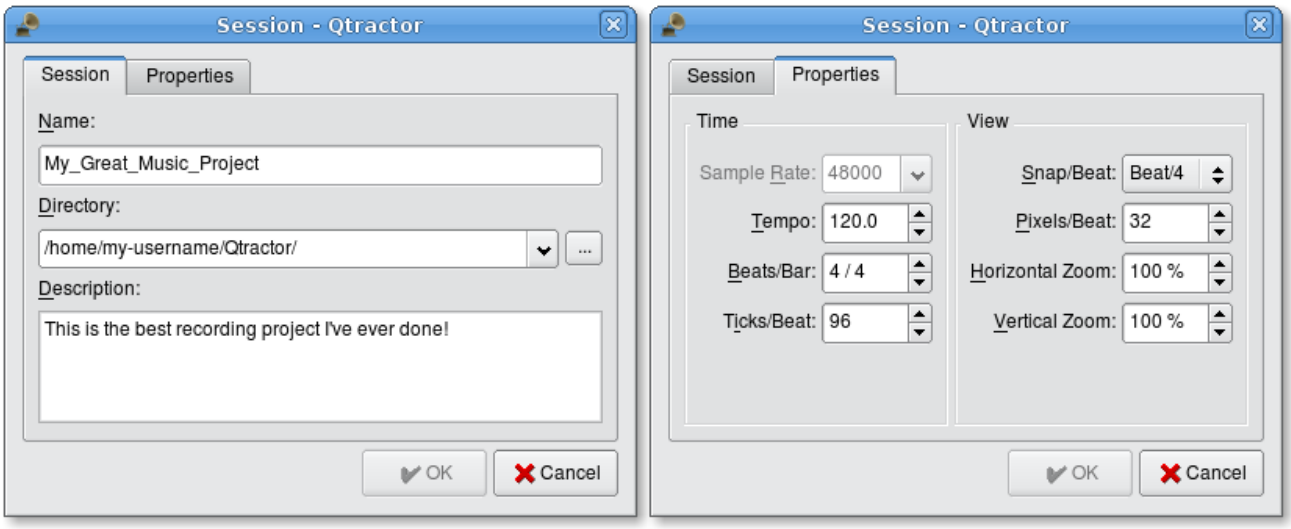


Illustration 4.3: Session properties window showing both Session and Properties tabs

A new feature, which applies to existing audio clips, will reflect all tempo changes with a corresponding time-stretching effect. Time-stretching is thus applied in real-time at the buffering level, as a custom WSOLA algorithm based and derived from the SoundTouch [12] library.

4.3.3 Session Options

Access the Options windows via the **View** / **Options** menu item. This window, and its tabs allows you to control the global parameters of Qtractor, and these are the default settings which are not saved within your session file.

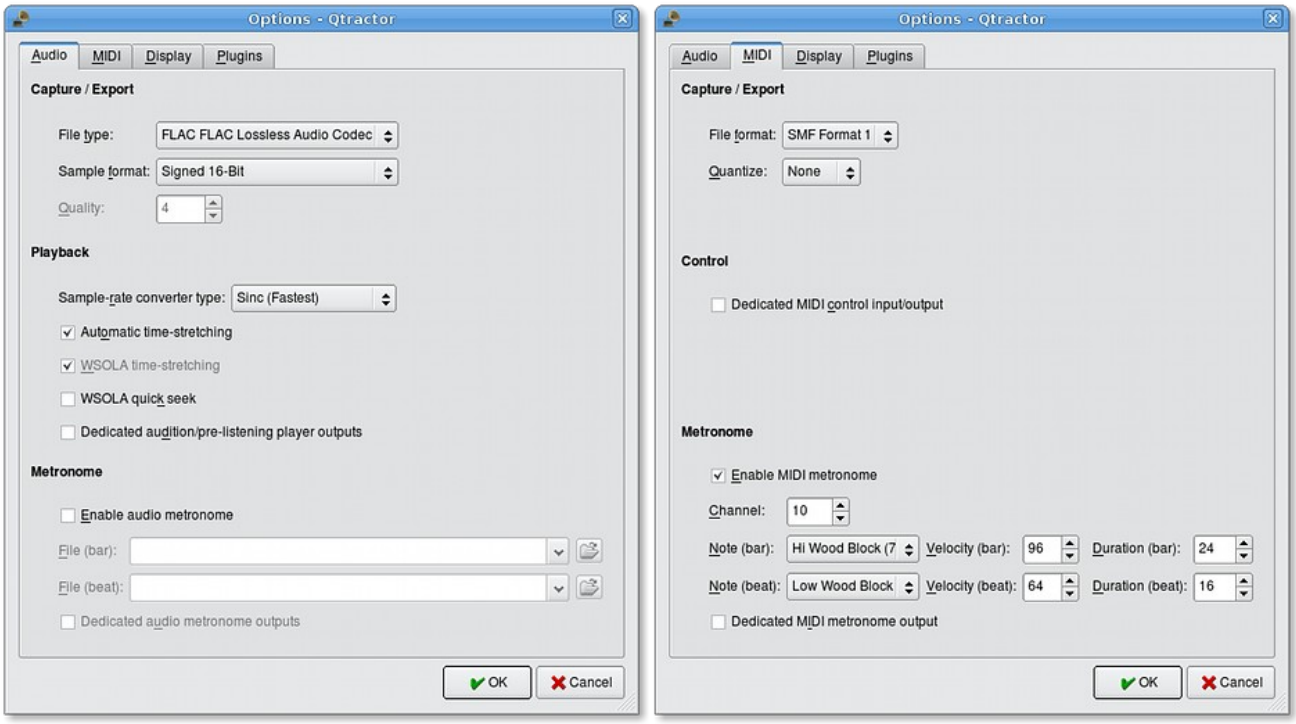


Illustration 4.4: Session Options window, showing both Audio and MIDI tabs (sections)

4.4 Files

Sound file selection is made available through a tabbed mini-organizer. Audio and MIDI file lists are kept separate on their respective tabs.

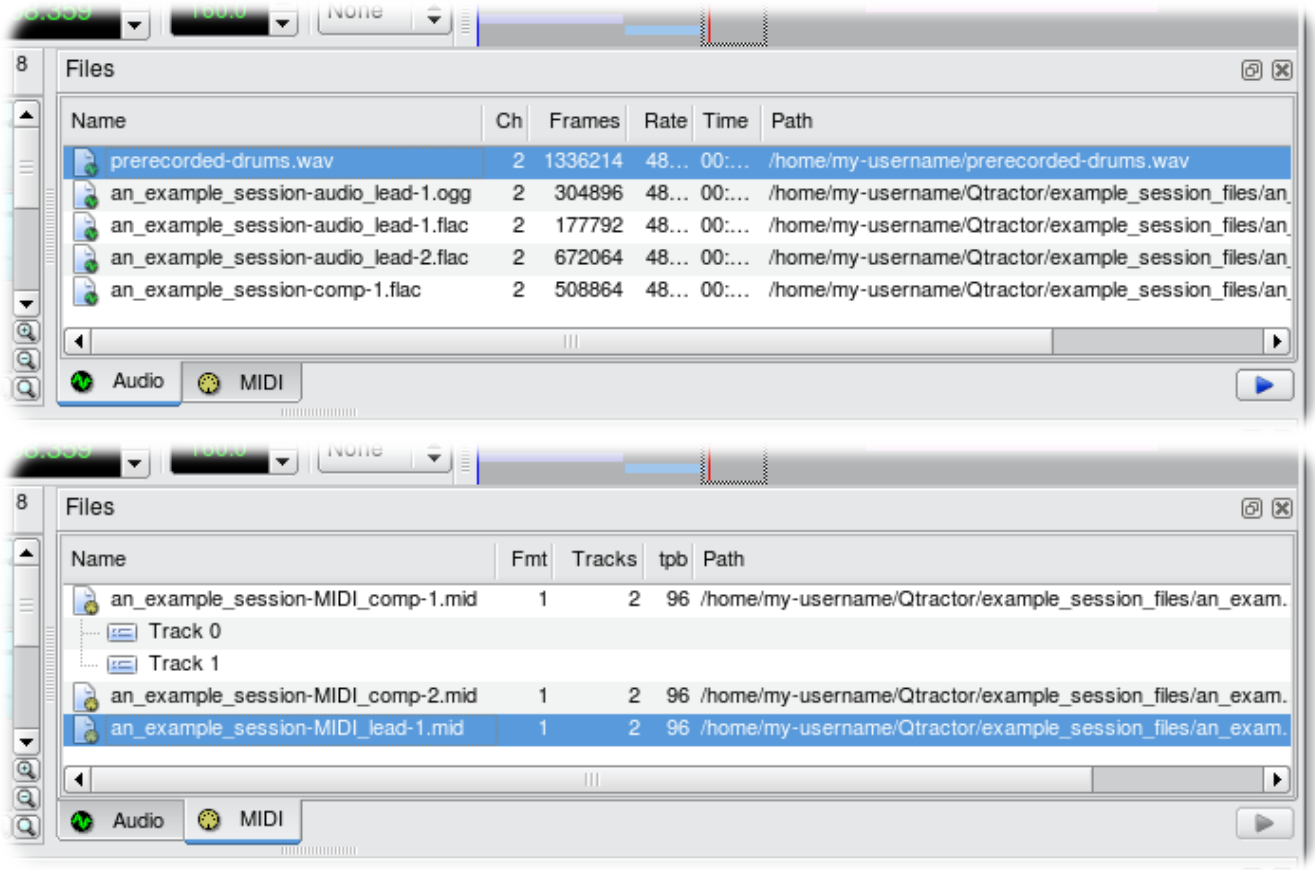


Illustration 4.5: The Files pane of Qtractor's main workspace; both Audio and MIDI tabs are shown

Files can be imported using the menu command **Track** / **Import Tracks** / **Audio...** or **MIDI...**, or by using the context menu ("right-click" menu) in either the Audio or MIDI tabs of the Files pane and choosing **Add Files...**. Individual and multiple files can be drag-and-dropped from the desktop environment and within the provided tree list. This lists all the files which are referred in the working arrangement session. File items can be drag-and-dropped directly into the track window, thus creating new clips in the working arrangement. This is mainly used as a your audio/MIDI file data pool.

The Files pool can also allow you to preview the files shown in the windows either by double-clicking the file name, or by click the play button on the lower right hand side of the Files Pool window.

Audio file format support is provided by libsndfile [5], and supports wav, aiff, flac, au, etc. Optional libraries provide support for both ogg and mp3 formats. libvorbis [6] (ogg) and libmad [7] (mp3). MIDI file support covers the usual SMF formats 0 and 1, through a native, home-brew implementation.

4.5 Clips

4.5.1 Clip Summary

Clip objects are the elemental items of a session arrangement, and can contain either Audio or MIDI data. A Clip Object is merely a region of an actual sample or MIDI file. A clip object is also non destructive, and can be copied, truncated and time stretched as if it were actual audio/MIDI data.

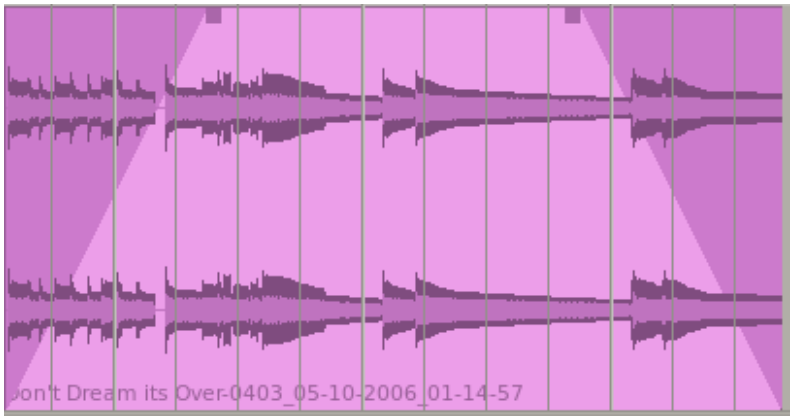


Illustration 4.6: An audio clip, with left and right upper corner handles drawn inward to create a fade-in and fade-out

4.5.2 Audio Clip Properties

Clip properties include its label (Name), File path, start time (location), offset and length (in frames), fade-in and fade-out length (in frames) and time stretch percentage, respectively, from the start and end of the clip. Although fade-in and fade-outs are always displayed as straight lines, the actual audio volume (gain) and MIDI velocity effect can be opted to be of either linear, square or cubic characteristic, in as for an approximation to the logarithmic model of human ear perception.

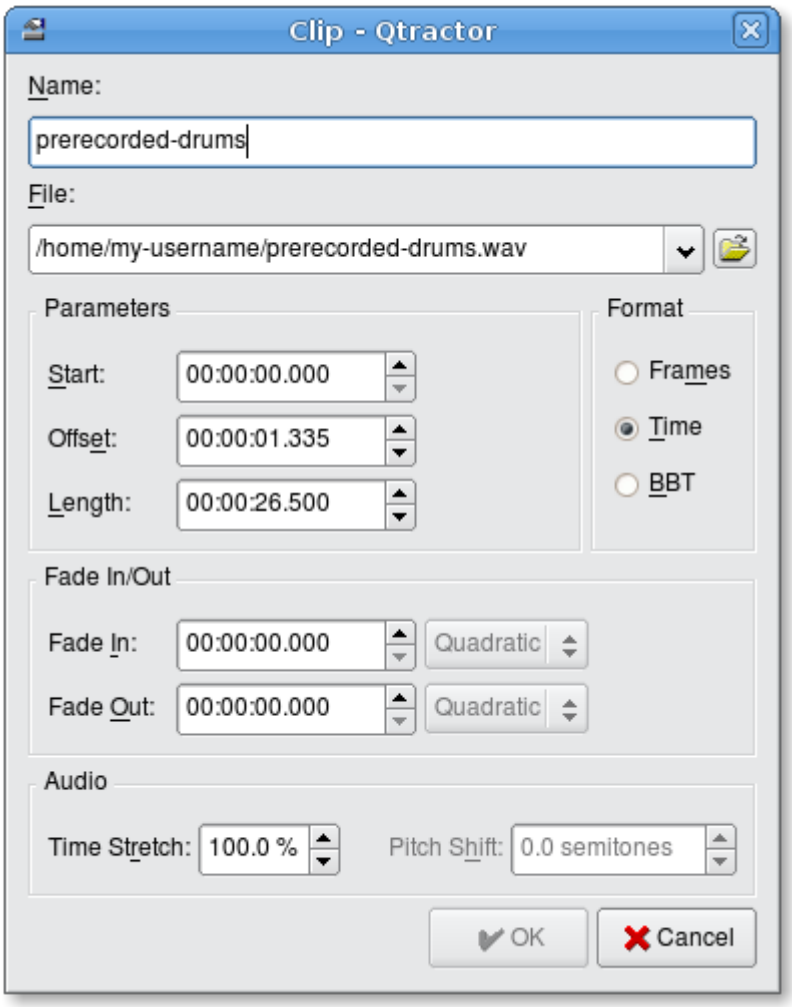


Illustration 4.7: Audio clip properties window, which appears with a double-click or right-click → Clip → Edit... on an audio clip's visual representation in Qtractor's main workspace

Clips are placed on tracks, either by importing audio and MIDI files as new tracks, or by dragging and dropping files into the track-view arranger window. Empty clips may also be created by right clicking on a track, and choosing **Edit** / **Clip** / **New...**. After being placed on their respective tracks, you may perform clip-region operations such as drag, copy, cut, paste, delete, truncate, fade in/out etc. Altering clip fade-in and fade-out is accomplished by dragging the handles (square boxes) on the top ends of any clip. A Clip may be split by positioning the playback head where you want to split it, right click the Clip, and choose **Edit** / **Clip** / **Split**. If you use this often, it is far more convenient to assign a keyboard shortcut to accomplish this task.

Most clip editing operations are accomplished through the usual mouse interaction, by first selecting one or multiple clips and/or regions, and applying the edit action upon the resulting selection.

There are three selection modes available: clip, range and rectangular modes (**Edit** / **Select Mode**).

- In **Clip** mode, clip objects are selected as a whole with no sub-clip regions possible.
- In **Range** mode, clip object regions are selected on all tracks between a given time interval or range.
- In **Rectangular** mode, only the regions that fall under a rectangular area are selected, this means for adjacent tracks and clips only

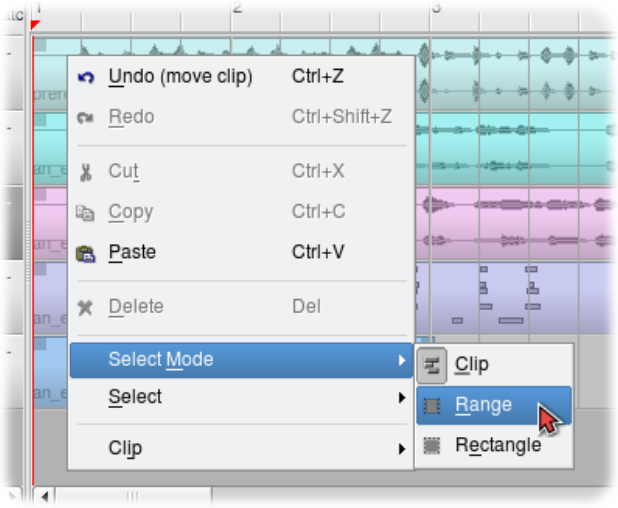


Illustration 4.8: Select Mode context menu (right-click menu)

4.5.3 Clips and Tracks

Tracks may be armed for recording, making way for creating new audio and MIDI clip files with captured material. Tracks can also be muted and soloed on mix-down, which also applies when exporting. Most editing operations should be possible while playback is rolling (but not completely safe though; there are many procedural helpers, but not completely assisted with lock-free primitives, yet)

Illustration 4.9: Mute, Solo & Record buttons for individual tracks

MIDI clip objects are representations of a sequence of events of one single MIDI channel, as extracted from a SMF format 0 file or of one single track, as from a SMF format 1, either in whole or in part.

4.6 Qtractor Main Workspace—Tracks Area

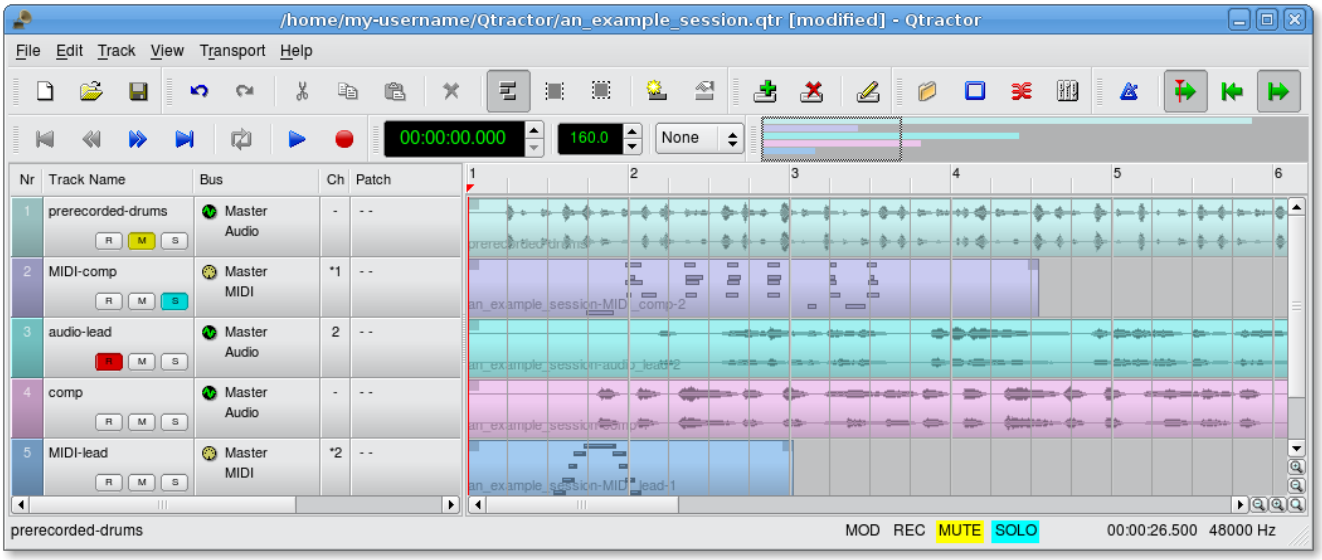


Illustration 4.10: Audio and MIDI tracks in the main workspace of Qtractor

Tracks are arranged as a sequence of one or more overlapping clips of the same file type, either audio or MIDI. The tracks window is the main application workspace, serving as a virtual canvas of a multi-track composition arranger. Most of the editing operations are made on this tracks area of the main workspace window.

The tracks area has two panes, the left one displays the list of tracks with their respective properties and the center-right pane is the main tracks view canvas window where multi-track composition and arranging activity is pictured and performed. As usual, tracks are stacked on horizontal strips and clips are layered on a bi-dimensional grid, in time sequence for each track strip. Time is modeled on the horizontal axis and pictured by a bar-beat scale ruler at the top of the track-view.

Clips may be conveniently aligned to discrete time positions, depending on the current *snap* mode setting. When not set to “None”, the snapping is always carried out to MIDI resolution, quantized to ticks per quarter note granularity.

Each track has its own user assignable colors for better visual identification. Audio clips are displayed with approximate waveform graphic, with peak and RMS signal envelopes as read from the respective audio file segment. MIDI clips are shown as a *piano-roll* like graphic, with note events shown as small rectangles, depicting pitch, time and duration.

All session, track and clip editing operations are undo/redo-able. Discrete view zooming and track vertical resizing operations are also available.

4.7 Mixer

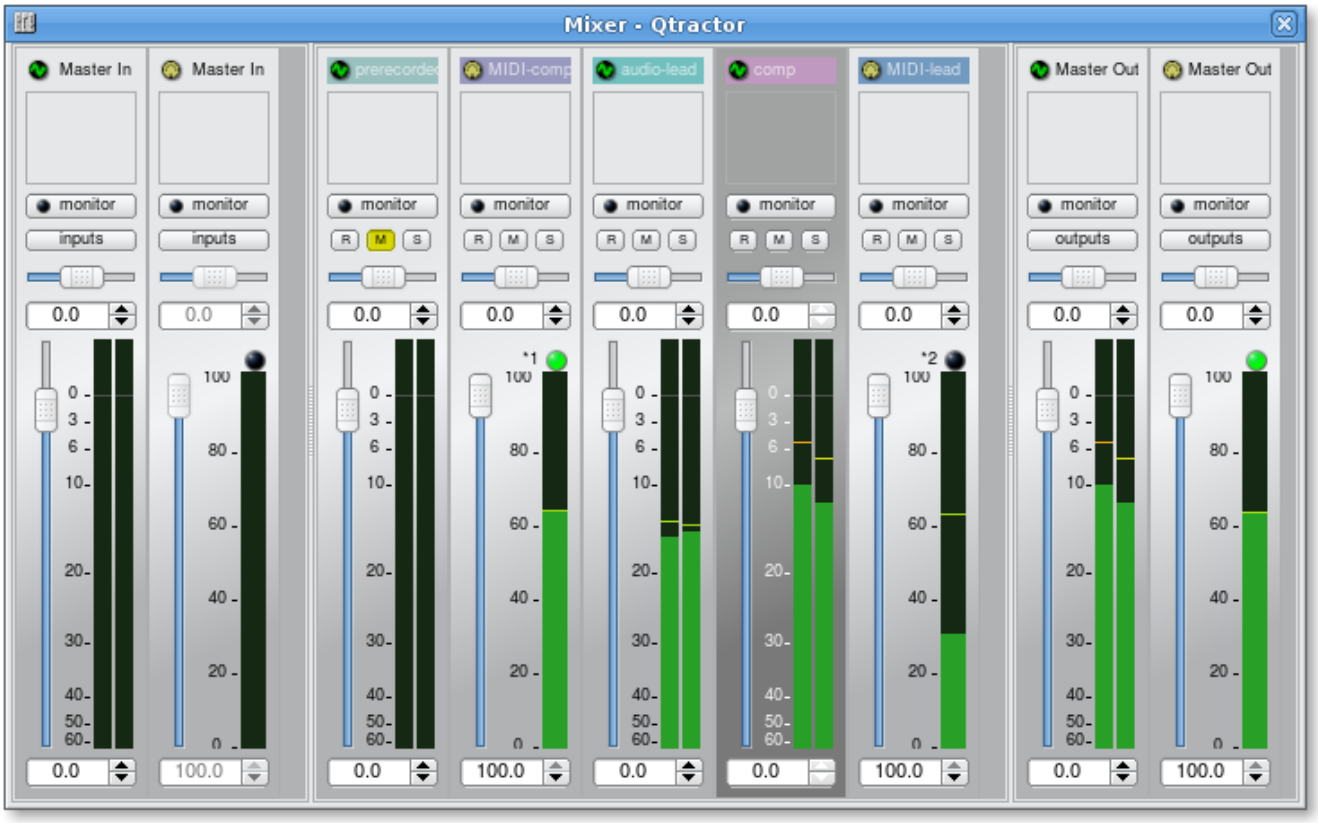


Illustration 4.11: Mixer window

The mixer window serves for session control, monitoring, recording and assistance in mix-down operations. The mixer is divided in three panes: the left accommodates all input buses, the center with individual track strips and the right for the output buses. Each mixer strip offers a volume and pan control and monitors each one of the respective buses and tracks. Audio strips also offers the possibility to chain plug-in effects (LADSPA [9]).

Monitoring is presented in the form of peak level meters for audio and note event velocity for MIDI, both with fall-off *eye-candy*. MIDI mixer strips also feature an output event activity LED.

Audio volume is presented on a dBfs scale (IEC 268-10) and pan is applied in approximated equal-power effect (trigonometric weighting). For MIDI tracks, volume control is implemented through respective channel controller-7 and system-exclusive master volume for output buses. MIDI pan control is only available for track strips and is implemented through channel controller-10. MIDI input buses have volume and pan controls disabled.

4.8 Connections Window

The Connections window (Illustration 2.1, page 9)serves to establish the audio and MIDI port connections between the internal core layer input and output buses (ports), and the external devices or client applications. Incidentally the Connections window can also be used to make connections between external client application ports, either JACK clients for audio, or ALSA sequencer clients for MIDI. In fact, it almost completely replicates the very same functionality of QjackCtl [10]. All connections on the existing input and output buses are properly saved and restored upon session recall.

4.9 Audio Effects Plug-ins

4.9.1 Summary

There are three types of plug-ins supported within Qtractor. LADSPA, DSSI and VST. Plug-in support is available for all audio input and output buses and for all audio tracks. All plug-in types are aggregated seamlessly as one single instance on a multi-channel context and can be individually selected, activated and moved within the plug-in chain order.

Also, you may drag-and-drop all plug-in instances over the mixer strip channels. You can move, drag and drop inside the same strip or over to, and from any other. You may also copy plug-ins from one channel strip to another as well. A mini menu will ask if you want to copy or move the plug-in. Individual plug-in control parameters can be modified in real-time through provided dialog windows and maintained as named presets for re-usability.

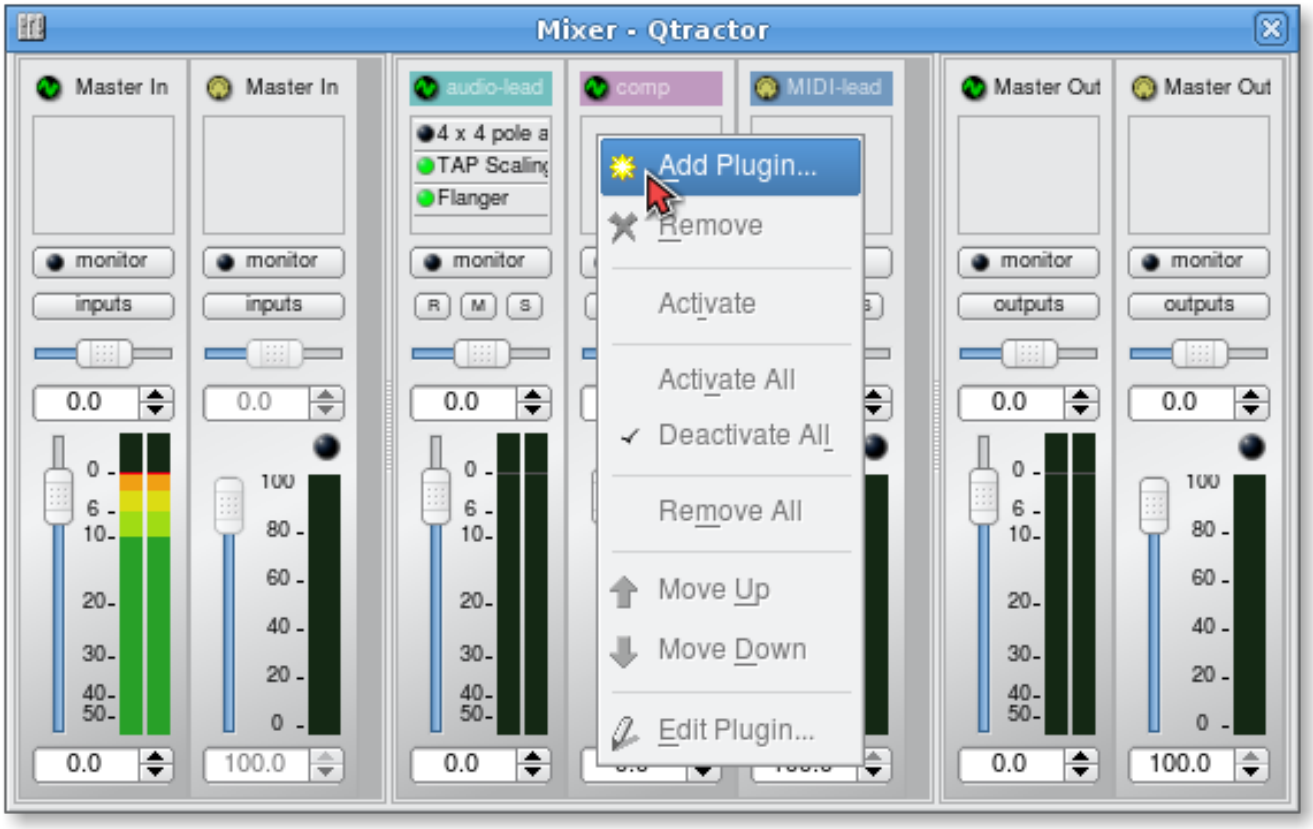


Illustration 4.12: Mixer window showing audio plug-in area at top; plug-ins are added, removed, activated/deactivated, moved in the processing chain and their settings edited using a right-click context menu.

4.9.2 LADSPA

LADSPA [9] has been the Linux audio plug-in standard for many years. There are literally hundreds of LADSPA plug-ins available for Linux. LADSPA plug-ins give the user many standard options such as Equalization, Filtering, Reverb, Chorus, Amp and speaker simulation, etc. Qtractor must be told the location (path in the file-system hierarchy) of the LADSPA plug-ins that have been installed by the user. This path can be specified in the **View** / **Options...** window, under the Plugins tab (see Illustration 4.13 below).

4.9.3 DSSI

DSSI plug-in support is available for DSSI effects plug-ins. DSSI Instrument plug-ins are not yet supported. You must have the core DSSI subsystem installed in order for this type of plug-in to function. When DSSI is present, the DSSI-VST wrapper may also be used. This wrapper uses WINE (http://www.winehq.org) to allow a user to run native Windows® VST applications. The DSSI paths may be set within the **View** / **Options...** window under the Plugins tab, or with an environment variable.

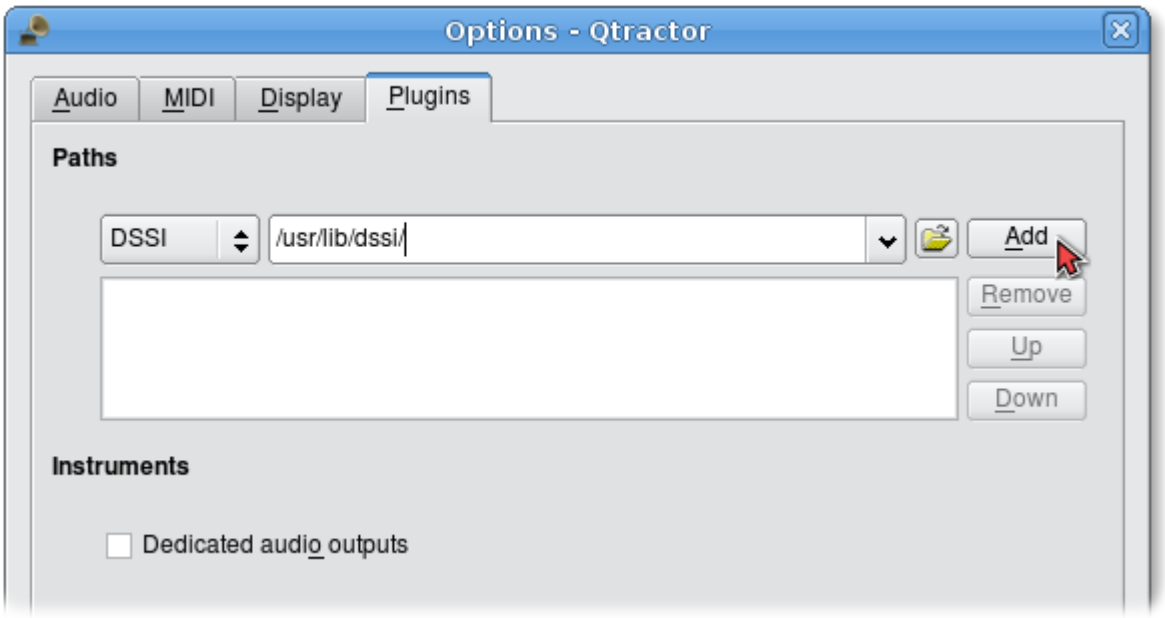


*Illustration 4.13: The locations (paths) of LADSPA and DSSI audio plugins can be specified in the **View** → **Options** window, under the Plugins tab.*

4.9.4 VST (Linux Native)

Native Linux VST plug-in support is also available. Presently, there are only a few native Linux VSTs available, but more should be on the way soon, thanks to some aggressive ongoing projects.

Please see section 2.4.2 for complete information on building Qtractor with native VST support.

Note: Native Linux VST support does NOT include running of Windows® VST plug-ins. Please use the DSSI-VST wrapper when attempting to use this type of plug-in, and make sure your Windows® VST plug-ins are located within your DSSI path environment variable (the variable name is **VST_PATH**).

4.10 MIDI Instruments

There are many MIDI hardware tone generators available from a variety of manufacturers such as Yamaha, Roland and Korg.. In Qtractor, information about the sound patches and sound banks of these tone generators is obtained from “instrument definition” files. Qtractor supports the instrument definition files used by the Cakewalk / Sonar [11] MIDI sequencer software, offering a convenient MIDI bank-select/program-change mapping for existing MIDI instrument patch names, and easier, intelligible selection of MIDI track channels. These Cakewalk / Sonar instrument definition files (.ins files) can be imported using the **View** / **Instruments...** window.

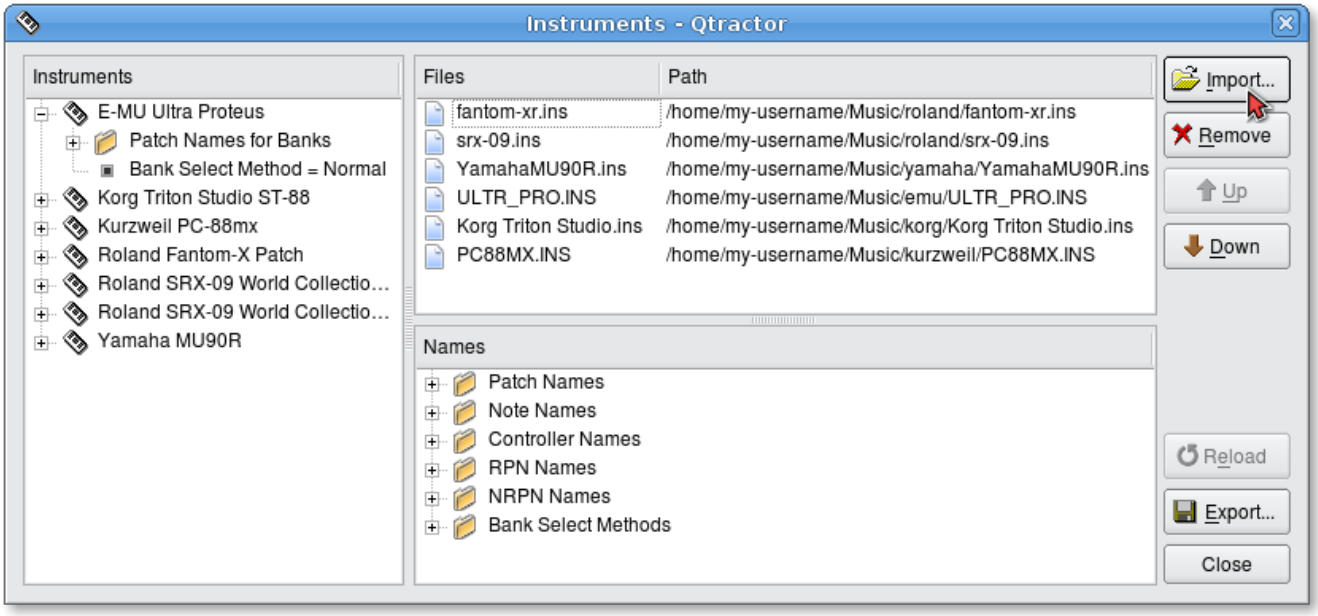


Illustration 4.14: Instruments window, where MIDI instrument definition files, for various popular MIDI tone generators, can be imported, exported and moved.

Cakewalk / Sonar instrument definition files for many popular MIDI tone generators and synthesizers can be downloaded from http://www.cakewalknet.com/ in the “Downloads” section of the website.

4.11 MIDI Editor

Each MIDI clip content may be readily edited under a dedicated and fairly complete “piano-roll” type editor,, with individual pitch, velocity and controller, editable trough the usual GUI operations such as: multi-extended selection, drag-and-drop, move, cut, copy, paste, deletion of every event in the MIDI sequence is rightly accessible on the fly.

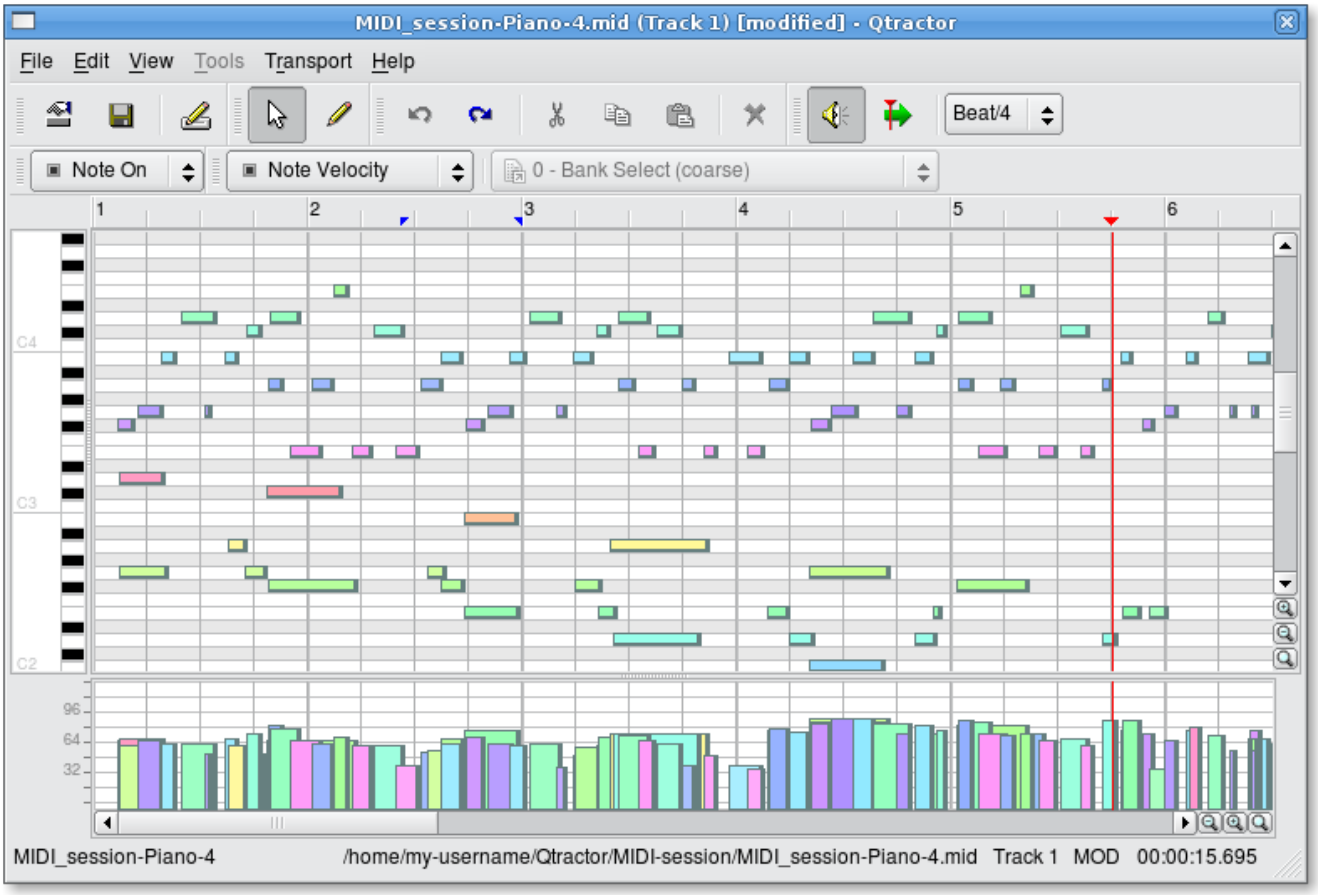


Illustration 4.15: MIDI Editor window, showing notes in a familiar “piano roll” type of graphical display, above, and a graph of their velocities below. The notes and their velocities can be selected and edited in many ways, using the mouse, menu commands and context menus.

Special tools for batch processing are also implemented and applicable to the any event selection: quantize, transpose, normalize, randomize and resize. All MIDI editing operations are available and processed in real-time, effective while playback. Several *MIDI Editor* instances may be active and open in any time, provided each one refers to its own clip.

All MIDI content may be saved as standard MIDI files (SMF) Format 0 (all tracks reduced to one track, preserving channel assignment data) or format 1 (multi-track). The format for the SMF files (format 0 or 1) may be set in **View** / **Options...** under the MIDI tab.

4.12 Audio / MIDI Export

All or part of the session may be exported to one audio or MIDI file. *Audio export* is implemented through the special JACK *freewheel* mode, thus faster than real-time, resulting in the complete and exact mix-down of selected audio material into a designated audio file of the opted format (wav, flac, au, aiff or ogg). *MIDI export* is just the same but for MIDI material only, resulting in the merging and concatenation of selected MIDI tracks and clips into a single MIDI file (SMF Format 0 or 1). The user-preferred format for exported audio files (OGG, FLAC, WAV, etc.) and MIDI files (SMF 0, all tracks reduced to single track or SMF 1, multi-track) can be set in the Options window (**View** / **Options...**).

*Illustration 4.16: Audio export window, where the audio tracks in a session can be named and exported as a single audio track in the particular format previously specified by the user in the **View** → **Options** window.*

4.13 Keyboard Shortcuts Editor

Keyboard shortcuts are useful for the power user, in such that it provides for a quick mechanism for performing often used commands quickly, without the use of your mouse.

Keyboard shortcuts may be customized to your preference by using the shortcuts editor. This editor may be found in the Help menu (**Help** / **Shortcuts...**).

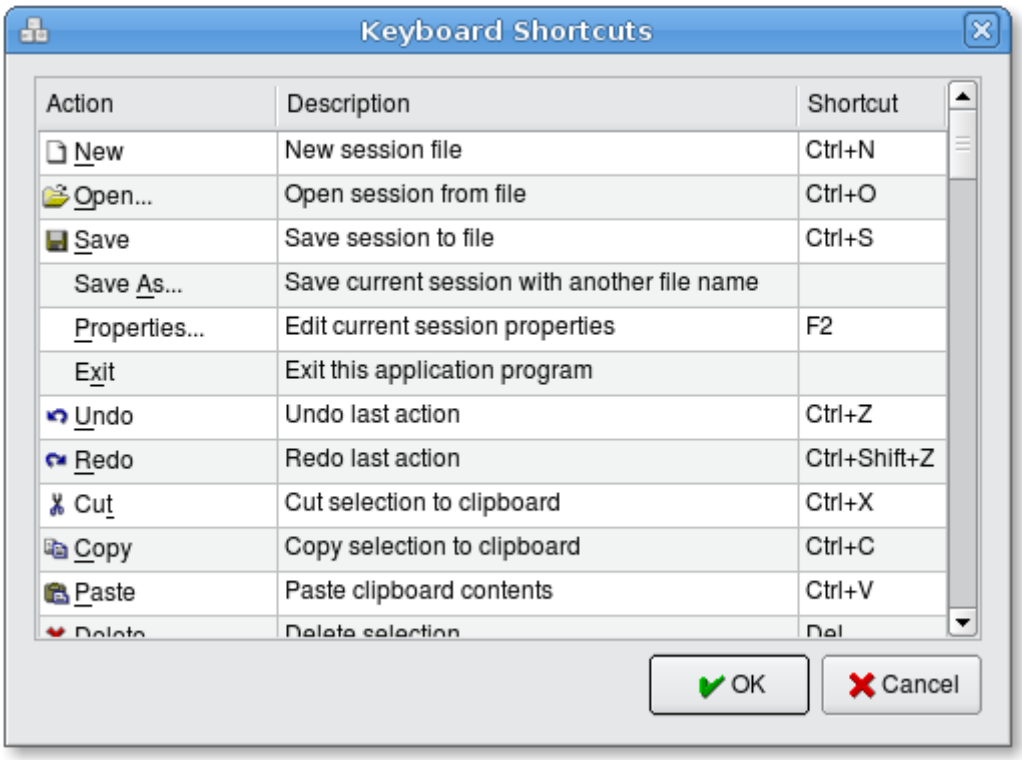


*Illustration 4.17: Keyboard Shortcuts Editor window, available from the **Help** / **Shortcuts...** menu item*

It is very straight forward in its use. Simply find the item you want to create a shortcut for, left click in the shortcut cell, and type on the key you wish to be assigned to that function. You will then see whatever key or key combination you chose appear in the context.

5 Qtractor Main Menu

5.1 File Menu

- **New** — creates a new session project.
- **Open...** — opens a previously created session project.
- **Open Recent** — contains a list of several of your last used session projects.
- **Save** — saves your current session project
- **Save As...** — saves your current session project with naming conventions
- **Properties...** — opens the session properties dialog
- **Exit** — exits the program.

5.2 Edit Menu

- **Undo** — undo the last action.
- **Redo** — redo the last action.
- **Cut** — deletes and copies the item to the clipboard.
- **Copy** — copies the item to the clipboard
- **Paste** — pastes the item from the clipboard.
- **Delete** — deletes the selected item.
- **Select Mode** — selects the edit mode, one of Clip, Range or Rectangle.
- **Select** — None, Range, Track or All
- **Clip** — New..., Edit..., Split, Normalize and Export...

5.3 Track Menu

- **Add Track...** — add either a new audio or MIDI track.
- **Remove Track** — removes a track.
- **Track Properties...** — calls the track properties dialog.
- **Inputs** — shows current track inputs.
- **Outputs** — shows current track outputs.
- **State** — current track state: Record, Mute, Solo, Monitor.
- **Navigate** — current track navigation: First, Previous, Next, Last.
- **Move** — move current track navigation: Top, Up, Down, Bottom.
- **Auto-monitor** — current track auto-monitoring.
- **Import Tracks** — import either audio or MIDI track.
- **Export Tracks** — export either audio or MIDI track.

5.4 View Menu

- **Menu Bar** — toggle whether the menu bar is shown.
- **Status Bar** — toggle whether the Status Bar is shown.
- **Tool Bars** — toggles various tool bars on and off.
- **Files** — toggles whether the Files directory is on or off.
- **Messages** — toggles whether the Messages dialog is on or off.
- **Connections** — displays the Connections dialog.
- **Mixer** — displays the Mixer window.

- **Zoom** — zoom the main view: In, Out, Reset.
- **Snap** — change the snap-per-beat setting.
- **Refresh** — refresh the view contents.
- **Instruments...** — displays the instruments dialog.
- **Buses...** — opens the buses editor dialog.
- **Options...** — opens the program options dialog.

5.5 Transport Menu

- **Backward** — playhead jumps backward to beginning of range markers or session
- **Rewind** — playhead moves backward at greater than usual speed
- **Fast Forward** — playhead moves forward at greater than usual speed
- **Forward** — playhead jumps forward to marker or end of session
- **Loop** — play the range marked as a loop, over and over again
- **Loop Set** — mark selected range as a loop
- **Play** — playhead moves forward at normal speed
- **Record** — prepare to record audio or MIDI on record-enabled tracks
- **Punch** — record the range marked to punch in/out
- **Punch Set** — mark selected range to record punch in/out
- **Metronome** — turn the metronome's audio or MIDI output on or off
- **Follow Playhead** — the tracks will scroll so that the playhead is always visible
- **Auto Backward** — playhead returns to beginning of range markers or session automatically when playhead is stopped
- **Continue Past End** — playhead will continue to move forward even after end of recorded tracks is reached

5.6 Help Menu

- **Shortcuts...** — lists the keyboard key combination equivalents of menu items and other commands
- **About...** — information about Qtractor
- **About Qt...** — information about Qt, the graphics toolkit used for Qtractor's user interface.

6 Appendixes

6.1 References

Certain words, terms or items in the manual have a bracketed number after them, indicating a reference to items in this appendix, where sources are given for more information about those items.

[1] **Qtractor** - An Audio/MIDI multi-track sequencer

http://qtractor.sourceforge.net/

http://sourceforge.net/projects/qtractor/

[2] **Qt 4** - C++ class library and tools for cross-platform development and internationalization.

http://www.trolltech.org/products/qt/

[3] **JACK** Audio Connection Kit

http://jackaudio.org/

[4] **ALSA** - Advanced Linux Sound Architecture

http://www.alsa-project.org/

[5] **libsndfile** - C library for reading and writing files containing sampled sound

http://www.mega-nerd.com/libsndfile/

[6] **libvorbis** - Ogg Vorbis audio compression

http://xiph.org/vorbis/

[7] **libmad** - High-quality MPEG audio decoder

http://www.underbit.com/products/mad/

[8] **libsamplerate** – The secret rabbit code, C library for audio sample rate conversion

http://www.mega-nerd.com/SRC/

[9] **LADSPA** - Linux Audio Developer's Simple Plugin API

http://www.ladspa.org/

[10] **QjackCtl** – JACK Audio Connection Kit - Qt GUI Interface

http://qjackctl.sourceforge.net/

http://sourceforge.net/projects/qjackctl/

[11] **Cakewalk** - Powerful and easy to use products for music creation and recording

http://www.cakewalk.com/

ftp://ftp.cakewalk.com/pub/InstrumentDefinitions/

[12] **SoundTouch** – Sound Processing Library

http://www.surina.net/soundtouch/

[13] **rncbc.org** - The personal web presence of Rui Nuno Capela, creator and developer of Qtractor and other applications or application GUIs such as Qjackctl (a GUI for jackd), Qsynth (a GUI for fluidsynth), Qsampler, etc.

http://www.rncbc.org/

6.2 Colophon

This manual was produced using the free, open-source office suite OpenOffice.org office, and typeset using the free Liberation family of typefaces from Red Hat Linux which consist of the typefaces (fonts) Liberation Serif, Liberation Sans and Liberation Mono, all of which have the same metrics (proportions) as the proprietary “Times New Roman,” “Arial” and “Courier New,” respectively.

OpenOffice.org

http://www.openoffice.org/

Red Hat’s Liberation Typefaces

https://www.redhat.com/promo/fonts/

6.3 Contact Us

For questions, comments and suggestions regarding Qtractor, please contact:

- Rui Nuno Capela, rncbc@rncbc.org

For questions, comments and suggestions regarding the Qtractor manual, please contact:

- James Laco Hines, jhines@cebridge.net
- Stephen Doonan, stephen.doonan@gmail.com

Index