Qtractor - An Audio/MIDI multi-track sequencer

Version 0.1.0

Frivolous Debutante

Rui Nuno Capela

rncbc.org

January 2008

Abstract

Qtractor is an Audio/ MIDI multi-track sequencer application written in C++ around the Qt4 toolkit using Qt Designer. The initial target platform will be Linux, where the Jack Audio Connection Kit (JACK) for audio, and the Advanced Linux Sound Architecture (ALSA) for MIDI, are the main infrastructures to evolve as a fairly-featured Desktop Audio/MIDI Workstation GUI, specially dedicated to the personal home-studio.

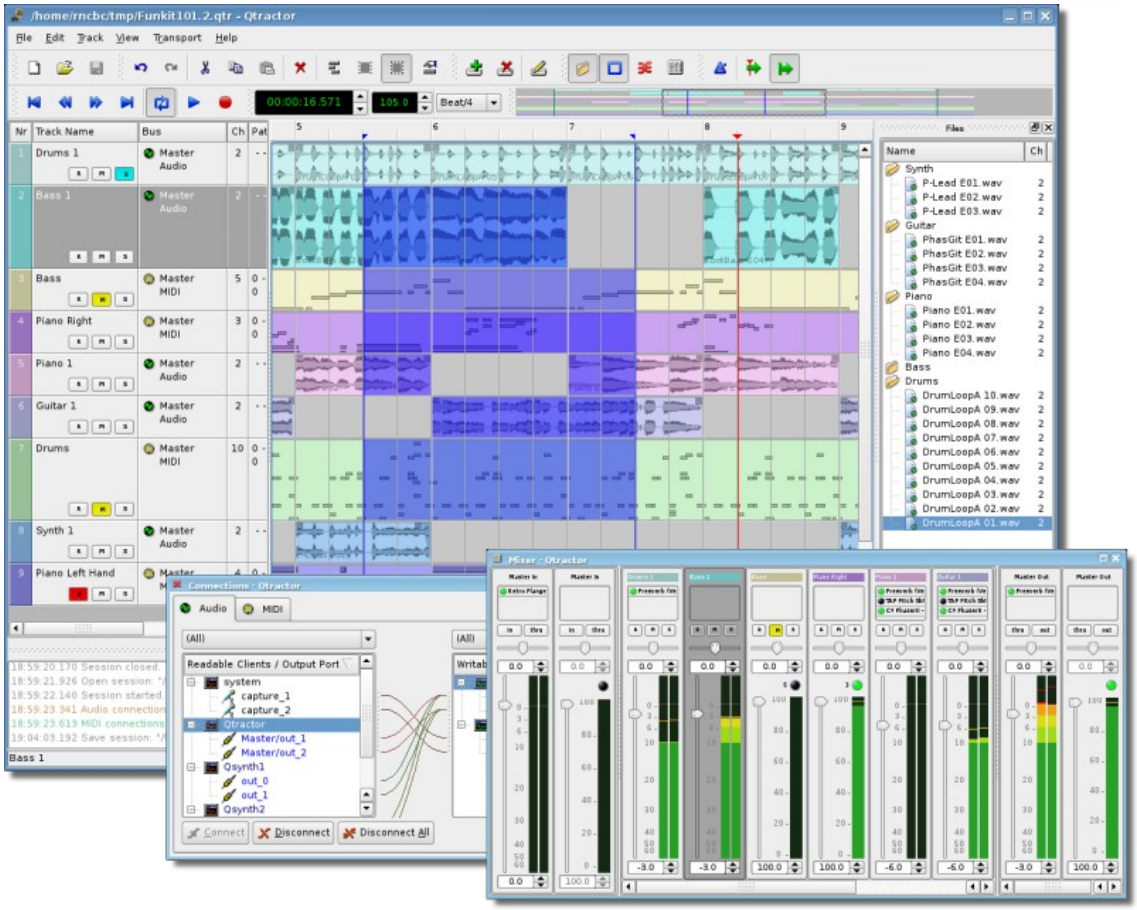

Figure 1: Main GUI aspect: Tracks, Mixer and Connections

Introduction

As a new Linux Audio offering, Qtractor [1] targets and positions itself comfortably tagged as for the *techno-boy bedroom home-studio*. However, in general, it is just yet another digital audio and MIDI multi-track composition and arranger software application. The design and functionality model takes as fundamental the now usual multi-track composing techniques for modern music-making. It aims to be intuitive and easy to use. Unfortunately, for the classic and erudite musician, there are and will be no plans to integrate any kind of score editor.

Qtractor is tentatively to be a part of the Linux home-studio. Even though built on a monolithic application architecture, it is not about to be the whole of it. Besides its name, it should be noted that it is not the same kind nor genre of that once called as *tracker* software. However, it has all the possibilities to be as much the same playground.

Currently the hobby work of one developer, development has started circa 2Q2005, initially as a Qt3 application. Since October 2006 it's officially a Qt4 [2] application. It is still a work in progress project.

It is natively hardwired and exclusive to the JACK [3] audio infrastructure and the ALSA [4] sequencer for MIDI, thus currently being a Linux-only application. There are no personal intentions on support to any other platforms.

Qtractor is free open-source software, licensed under the GPL and is welcoming all collaboration and review from the Linux Audio developer and user community in particular and the public in general.

Requirements

The software requirements for build and run-time are listed as follows:

Mandatory:

- **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/

Optional (opted-in at build time):

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

Download

Qtractor is still in some alpha stage of development, but already fully functional. The latest released source code is publicly available from the qtractor.sourceforge.net project web site [1].

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

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:

```
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.

Installation

The standard procedure for source distributions based on *autoconf* follows, through the now quite usual build command sequence:

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

and complete installation by having the appropriate (root) authority:

```
make install
```

which will end installing by copying the `qtractor` binary executable and desktop and icon files to common standard base system locations.

Configuration

Qtractor holds its run-time settings and configuration state per user, in a file located as `$HOME/.config/rncbc.org/Qtractor.conf` . Normally, there is no need to edit this file, as it is recreated and rewritten every time `qtractor` is run.

GUI

Qtractor user interface design is thought as mainstream and standard as it could be on modern computer desktop environments. The presentation layer is supposed to be easy for the average user to interact and to discover the full potential and functionality of the inner core application layer (i.e. Audio and MIDI engines, in a nutshell). Figure 1 shows an overall aspect of the GUI, with example session loaded into the workspace.

The main presentation window includes the usual *menu*, *tool-bars* with common action icon-buttons, and the main application workspace where one finds the *track list* on the left and the *track time* scale ruler at the top of the main arranger *track view*, which is where main action takes place and tracks and clips are graphically displayed.

Core application functionality is complemented with a couple of top-level floating windows: the *mixer* window, resembling a conventional mixing console assisting in the control and monitoring operations, and the *connections* window, featuring integrated inter-application device connectivity and routing (patch-bay).

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. Finally, session and application configuration options are assisted through respective customizing dialogs: *buses*, *instruments* and *options*.

Sessions

Qtractor sessions are defined as the complete and internally descriptive state of an arrangement made with the software. This state description is materialized in document form, as a XML encoded file. A session is therefore the computational description of all properties, variables, references and parameters of all audio and MIDI files and plug-ins that composes the working musical arrangement, put together while working with the software.

Effectively, sessions are formed by one or more tracks, which in turn includes their respective and fundamental elements, the clips. All possible and relevant information is stored in the session file, making it perfectly possible to restore the current working session at a later time. Sessions can thus be created as new, saved and loaded on demand, as found usual on document based GUI applications. Qtractor is a SDI application, only one session can be loaded for work at anyone time.

It is important to note that Qtractor sessions are locked hard to one, and only one, fixed audio signal sample-rate, exactly the one the JACK [3] server is running at the time the session is started. Any attempt to convert disparate sample-rate sessions is subject to 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]).

Another restriction worth mentioning is that sessions have constant tempo (BPM) by current design decision, but can be otherwise changed at any time. However, it must be still regarded as a global property of the whole musical arrangement, that meaning there is no support for a multi-tempo and time signature map, yet.

One recent 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.

Files

Sound file selection is made available through a tabbed mini-organizer and convenient widget. Audio and MIDI file lists are kept separated on their respective tabs. Files can be explicitly added and grouped into a hierarchical tree list. Individual and multiple files can be explicitly 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

Audio file format support is the same as the one provided by libsndfile [5] (wav, aiff, flac, au, etc.) and, optionally libvorbis [6] (ogg) and libmad [7] (mp3). MIDI file support covers the usual SMF formats 0 and 1, through native, home-brew implementation.

Tracks and Clips

Clips are fully described as the elemental items of a session arrangement, being either integral or partial parts of existing audio and MIDI material, stored on external formated files.

As is, Qtractor is technically described as a non-destructive sequencer and arranger. It does not affect, alter or modify in any way, none of the audio or MIDI files that are loaded and represented as clips. Files resulting from capture and recording operations are the notable exceptions as is the explicit changes made through specialized clip editing (e.g., *MIDI Editor*).

Once created, all recorded files are thereafter loaded as usual. All editing operations are accomplished in exclusive parametric form, having no resemblance whatsoever on the file system.

Audio clips are representations of the whole or part of a single audio file. MIDI clips 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 one, either in whole or in part.

Clip properties may include its label (name), start time (location), offset and length (in frames), fade-in and fade-out length (in frames), 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.

Clips are placed on tracks, either by importing integral audio and MIDI files as new tracks or by drag-and-dropping files into the track-view arranger window. After being initially placed on their respective tracks, clips are subject targets to featured clip-region operations: drag-move, copy, cut, paste, delete, etc. Altering clip fade-in and fade-out is also carried out by dragging corresponding visual handles.

Most clip editing operations are accomplished through usual GUI 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 modalities. In clip-mode,

clips are selected as a whole with no sub-clip regions possible. In range-mode, clip 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, that meaning for adjacent tracks and clips only.

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).

Engines and Buses

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 buffer (resident in RAM).

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 1-second periods to the JACK process cycle. A single thread is responsible to listening (polling) on MIDI input and multiplexes all incoming events through record-armed MIDI tracks. Time stamping is done through ALSA sequencer facility.

Looping is assisted 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, most 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. Figure 2 shows one typical signal flow block diagram.

By default, “Master” buses are automatically created at session startup, being stereo for audio (2 channels ports, auto-connected) and single port for MIDI (16 logically addressable channels). Bus ports are accessible for arbitrary connection to and from external client applications or devices, through the *connections* window interface.

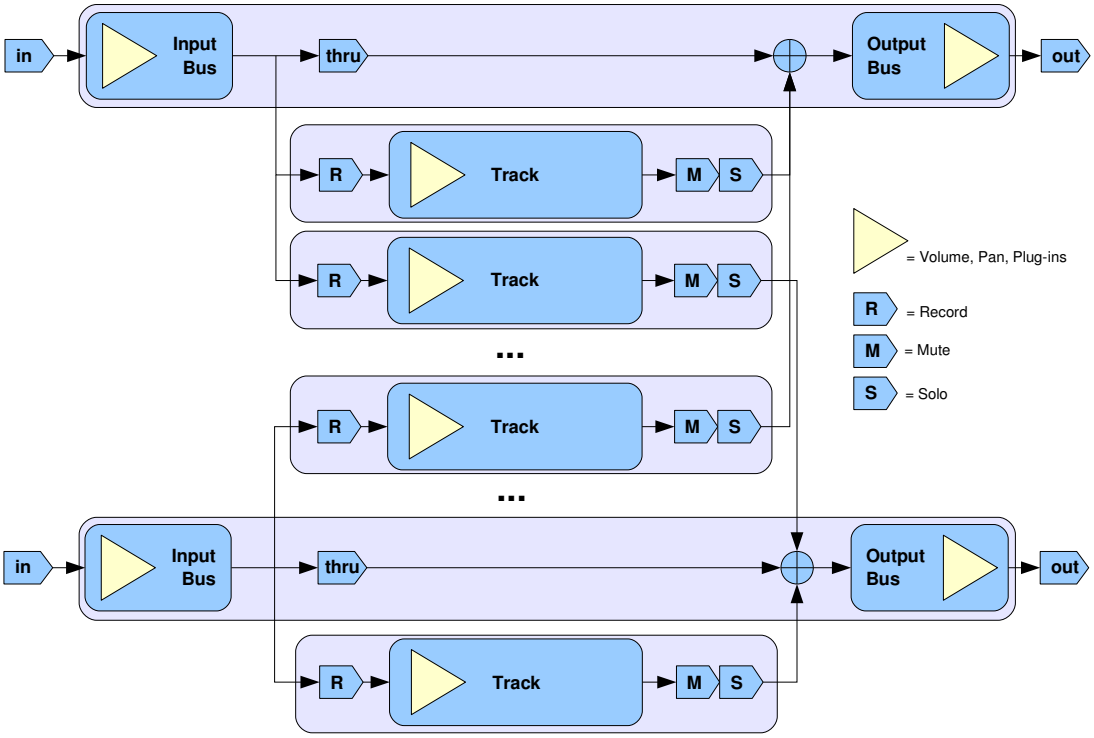

Figure 2: Buses / Tracks signal flow diagram

Track View

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 track list-view window.

The track list-view window has two panes, the left one displays the list of tracks with their respective properties and the center-right pane is the proper track-view canvas window where main 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.

Mixer

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.

Connections Patch-bay

The connections window serves the establishing of 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.

Audio Effects Plug-ins

LADSPA [9] plug-in support is available for all audio input and output buses and for all audio tracks. Plug-ins 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. Individual plug-in control parameters can be modified in real-time through provided dialog windows and maintained as named presets for re-usability.

MIDI Instruments

As a special feature, Cakewalk [11] instrument definition files (.ins) are natively supported, thus offering a convenient MIDI bank-select/program-change mapping for existing MIDI instrument patch names, and easier, intelligible selection of MIDI track channels.

MIDI Editor

Each MIDI clip content may be readily edited under a dedicated and fairly complete piano roll *MIDI Editor* widget, with individual pitch, velocity and controller editable trough the usual GUI operations. Multi-extended selection, drag-and-drop, move, cut, copy, paste, deletion of every event in the MIDI sequence is rightly accessible on the fly (Figure 3).

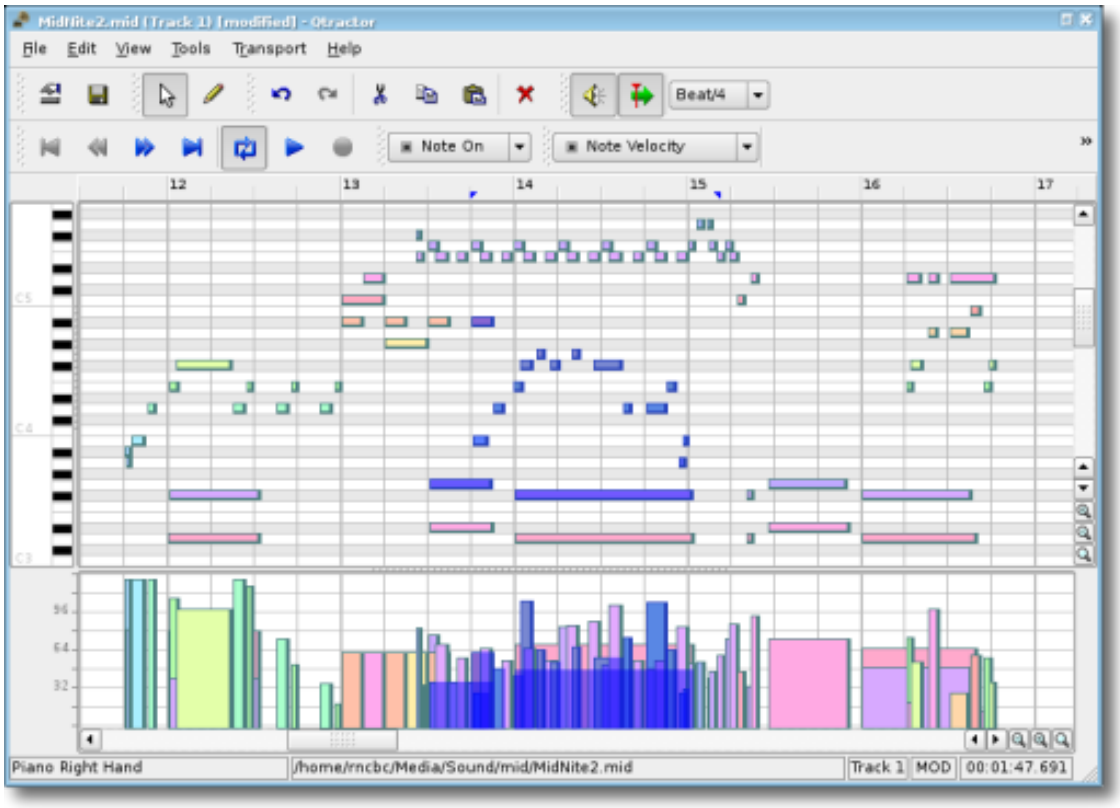

Figure 3: The MIDI Editor GUI aspect

Special home-brew 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 changed MIDI content is subject for being saved to regular standard MIDI files (SMF Format 0 or 1).

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).

Future Thoughts

As of its current status, there are many and rather fundamental functionality still missing that tear Qtractor apart from a finished product, let alone for the quest of its own goals. It's still a work in progress. In my own personal agenda priority, the following are the ones for taking care in times to come, in no special order:

- Metronome clicks (already implemented, in audio and MIDI options)
- Time-stretching (already implemented) and pitch-shifting of individual audio clips
- DSSI and native VSTi plug-in support
- Clip locking and ad-hoc clip editing
- Automation and dynamic curves (volume, pan, plug-in parameters. controllers)
- Auto-cross fading of overlapped clips
- Markers and named ranges
- Punch-in/out and looped recording (adding takes)
- Mixer presets and fader groups
- MIDI Editor draw mode and groove/swing quantize
- MIDI SysEx librarian
- MIDI event list editor and filter
- MIDI control feedback
- MIDI Time Code and SMPTE
- MIDI Clock sync
- JACK-MIDI support
- OSC or D-BUS interface
- Next generation LV2 plug-ins
- Integrated scripting (angelscript?)
- Tempo and time signature map

Conclusion

As fundamental as is, Qtractor might be just some clone of earlier and existing software, being blatantly one of the Cakewalk's [11] Pro Audio series. It is however more than that, when regarded from the free open-source software development point of view, much like some cauldron framework, user-oriented, programmable, pattern sequencer, eventually targeted as a potential toolbox and workbench for easy, direct, live music-making and experimentalism

Acknowledgements

I am grateful to the free software open-source community in general and to the Linux Audio developers and users in particular, who dedicated their valuable time to the development and support of free audio and MIDI software, being Qtractor just one humble manifestation of such class of human endeavor.

References

[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** - My personal web presence, blog, forum and other mundane material
http://www.rncbc.org/