Sequencer24 Developer's Reference Manual

0.9.4

Generated by Doxygen 1.8.9.1

Sat Sep 12 2015 15:32:05

Contents

1 Sequencer24

Author(s) Chris Ahlstrom 2015-09-05

1.1 Introduction

Sequencer24 is a minor cleanup, refactoring, and documentation of the Seq24 live-play MIDI sequencer.

The current document describes the functions, classes, modules, and other entities used in this project.

For now, please read the ROADMAP and README files to understand the genesis of this project.

Also, we have pretty deeply documented *Seq24* and *Sequencer24* with PDF files that can be generated by git-cloning the following projects, installing a number of tools related to PDF and LaTeX, and running "make":

- `https://github.com/ahlstromcj/seq24-doc.git`
- `https://github.com/ahlstromcj/sequencer24-doc.git`

In the present document, we've left out a fair amount of side-material to cut down on the size of the document. For example, the main module, redundant Windows support, utility headers like easy_macros.h, simple stuff like the mutex module, the fruity variants (at least the ones already refactored into their own modules), etc., are all left out.

2 Licenses

Library This application and its libraries, sub-applications, and documents.

Author(s) Chris Ahlstrom 2015-08-14

2.1 License Terms for the This Project.

Wherever the tag $XPC_SUITE_GPL_LICENSE$ appears, or wherever reference to the GPL licensing scheme (any version) is mentioned, substitute the appropriate license text, depending on whether the project is a library, application, documentation, or server software. We're not going to include paragraphs of licensing information in every module; you are responsible for coming here to read the licensing information.

These licenses apply to each sub-project and file artifact in the project with which this license description was packaged.

Wherever the term **XPC** is encountered in this project, it refers to my projects, which go beyond the package that contains this document.

2.2 XPC Application License

The **XPC** application license is either the **GNU GPLv2**. or the **GNU GPLv3**. Generally, projects that originate with me use the latter language, while projects I have extended may specify the former license.

Copyright (C) 2015-2015 by Chris Ahlstrom

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU GPL version 3 license can also be found here:

http://www.gnu.org/licenses/gpl-3.0.txt

2.3 XPC Library License

The **XPC** library license is the **GNU LGPLv3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Lesser Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU LGPL version 3 license can also be found here:

http://www.gnu.org/licenses/lgpl-3.0.txt

2.4 XPC Documentation License

The **XPC** documentation license is the **GNU FDLv1.3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This documentation is free documentation; you can redistribute it and/or modify it under the terms of the GNU Free Documentation License as published by the Free Software Foundation; either version 1.3 of the License, or (at your option) any later version.

This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Free Documentation License for more details.

You should have received a copy of the GNU Free Documentation License along with this documentation; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU FDL version 1.3 license can also be found here:

http://www.gnu.org/licenses/fdl.txt

2.5 XPC Affero License

The **XPC** "Affero" license is the **GNU AGPLv3**.

Copyright (C) 2015-2015 by Chris Ahlstrom

This server software is free server software; you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation; either version 1.3 of the License, or (at your option) any later version.

This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Free Documentation License for more details.

You should have received a copy of the GNU Affero General Public License along with this server software; if not, write to the

```
Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA.
```

The text of the GNU AGPL version 3 license can also be found here:

http://www.gnu.org/licenses/agpl-3.0.txt

At the present time, no **XPC** project uses the "Affero" license.

2.6 XPC License Summary

Include one of these licenses in your Doxygen documentation with one of the following Doxygen tags specified above:

```
\ref gpl_license_subproject
\ref gpl_license_application
\ref gpl_license_library
\ref gpl_license_documentation
\ref gpl_license_affero
```

For more information on navigating GNU licensing, see this page:

```
http://www.gnu.org/licenses/
```

Copies of these licenses (and some logos) are provided in the `licenses` directory of the main project (or you can search for them at *gnu.org*).

3 Todo List

Global mainwnd::mainwnd (perform *a_p)

Offload most of the work into an initialization function like options does; make the perform parameter a reference.

Better as a member function.

Global perfedit::perfedit (perform *a_perf)

Offload most of the work into an initialization function like options does; make the perform parameter a reference.

Global Seq24SeqEventInput::on_button_press_event (GdkEventButton *a_ev, seqevent &ths)

Needs update.

Global seqedit::seqedit (sequence *a_seq, perform *a_perf, int a_pos)

Offload most of the work into an initialization function like options does; make the sequence and perform parameters references.

Global sequence::remove_marked ()

Verify that this is the correct way to handle changing iterators.

4 Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

5 Data Structure Index

5.1 Data Structures

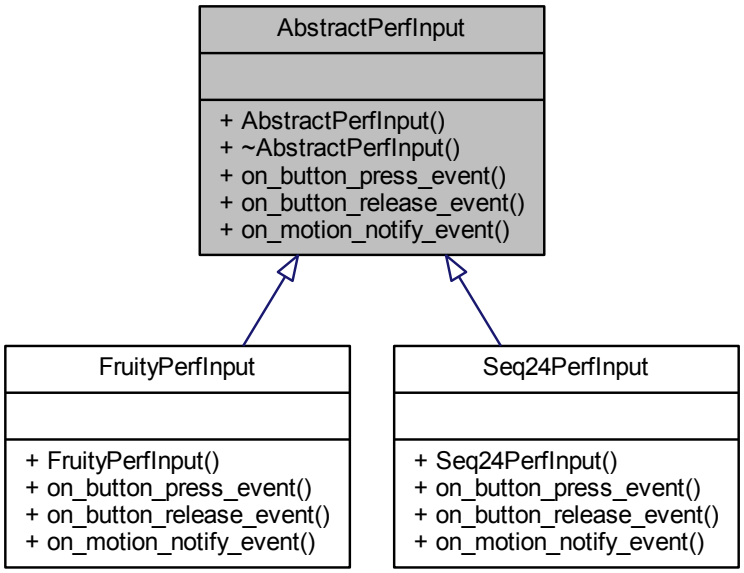
Here are the data structures with brief descriptions:

6 Data Structure Documentation

6.1 AbstractPerfInput Class Reference

Provides an abstract base class to provide the minimal interface for the various "perf input" classes.

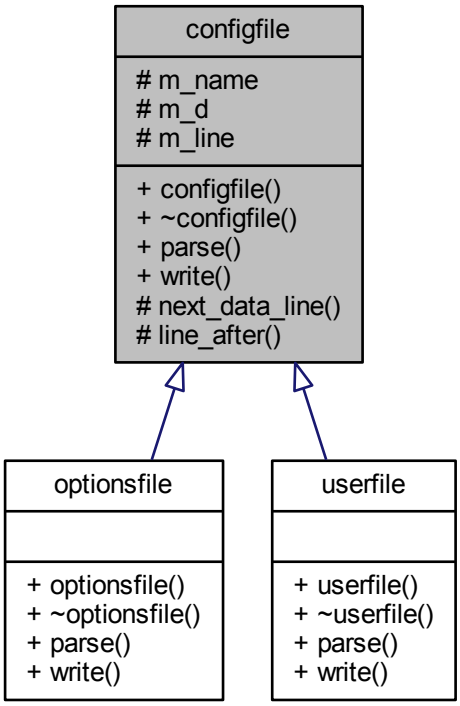
Inheritance diagram for AbstractPerfInput:



6.2 configfile Class Reference

This class is the abstract base class for optionsfile and userfile.

Inheritance diagram for configfile:



Public Member Functions

- configfile (const std::string &a_name)

  Provides the string constructor for a configuration file.
- virtual ~configfile ()

  A rote constructor needed for a base class.

Protected Member Functions

- void next_data_line (std::ifstream &a_file)

  Gets the next line of data from an input stream.
- void line_after (std::ifstream &a_file, const std::string &a_tag)

  This function gets a specific line of text, specified as a tag.

Protected Attributes

- std::string m_name

  Provides the name of the file.
- unsigned char * m_d

  Points to an allocated buffer that holds the data for the configuration file.
- char m_line [SEQ24_LINE_MAX]

  The current line of text being processed.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 configfile::configfile (const std::string & *a_name*)

Parameters

a_name	The name of the configuration file.

6.2.2 Member Function Documentation

6.2.2.1 void configfile::next_data_line (std::ifstream & *a_file*) [protected]

If the line starts with a number-sign, a space (!), or a null, it is skipped, to try the next line. This occurs until an EOF is encountered.

We may try to convert this item to a reference; pointers can be subject to problems. For example, what if someone passes a nullpointer? For speed, we don't check it.

Member m_line is a "global" return value.

Parameters

a_file	Points to an input stream.

6.2.2.2 void configfile::line_after (std::ifstream & *a_file,* const std::string & *a_tag*) [protected]

Parameters

a_file	Points to the input file stream.
a_tag	Provides a tag to be found. Lines are read until a match occurs with this tag.

6.2.3 Field Documentation

6.2.3.1 char configfile::m_line[SEQ24_LINE_MAX] [protected]

This member receives an input line, and so needs to be a character buffer.

6.3 event Class Reference

Provides events for management of MIDI events.

Public Member Functions

- event ()

  This constructor simply initializes all of the class members.
- ~event ()

  This destructor explicitly deletes m_sysex and sets it to null.
- bool operator< (const event &rhsevent) const

  If the current timestamp equal the event's timestamp, then this function returns true if the current rank is less than the event's rank.
- void set_timestamp (unsigned long a_time)

  'Setter' function for member m_timestamp
- long get_timestamp () const

  'Getter' function for member m_timestamp
- void mod_timestamp (unsigned long a_mod)

  Calculates the value of the current timestamp modulo the given parameter.

- void set_status (char status)

  Sets the m_status member to the value of a_status.
- unsigned char get_status () const

  'Getter' function for member m_status
- void set_data (char D1)

  Clears the most-significant-bit of the a_D1 parameter, and sets it into the first byte of m_data.
- void set_data (char D1, char D2)

  Clears the most-significant-bit of both parameters, and sets them into the first and second bytes of m_data.
- void get_data (unsigned char *D0, unsigned char *D1)

  Retrieves the two data bytes from m_data[] and copies each into its respective parameter.
- void increment_data1 ()

  Increments the first data byte (m_data[1]) and clears the most significant bit.
- void decrement_data1 ()

  Decrements the first data byte (m_data[1]) and clears the most significant bit.
- void increment_data2 ()

  Increments the second data byte (m_data[1]) and clears the most significant bit.
- void decrement_data2 ()

  Decrements the second data byte (m_data[1]) and clears the most significant bit.
- void start_sysex ()

  Deletes and clears out the SYSEX buffer.
- bool append_sysex (unsigned char *a_data, long size)

  Appends SYSEX data to a new buffer.
- unsigned char * get_sysex () const

  'Getter' function for member m_sysex
- void set_size (long a_size)

  'Setter' function for member m_size
- long get_size () const

  'Getter' function for member m_size
- void link (event *a_event)

  Sets m_has_link and sets m_link to the provided event pointer.
- event * get_linked () const

  'Getter' function for member m_linked
- bool is_linked () const

  'Getter' function for member m_has_link
- void clear_link ()

  'Setter' function for member m_has_link
- void paint ()

  'Setter' function for member m_painted
- void unpaint ()

  'Setter' function for member m_painted
- bool is_painted () const

  'Getter' function for member m_painted
- void mark ()

  'Setter' function for member m_marked
- void unmark ()

  'Setter' function for member m_marked
- bool is_marked () const

  'Getter' function for member m_marked
- void select ()

  'Setter' function for member m_selected
- void unselect ()

'Setter' function for member m_selected

- bool is_selected () const

  'Getter' function for member m_selected
- void make_clock ()

  Sets m_status to EVENT_MIDI_CLOCK;.
- unsigned char get_note () const

  Assuming m_data[] holds a note, get the note number, which is in the first data byte, m_data[0].
- void set_note (char a_note)

  Sets the note number, clearing off the most-significant-bit and assigning it to the first data byte, m_data[0].
- unsigned char get_note_velocity () const

  'Getter' function for member m_data[1], the note velocity.
- void set_note_velocity (int a_vel)

  Sets the note velocity, with is held in the second data byte, m_data[1].
- bool is_note_on () const

  Returns true if m_status is EVENT_NOTE_ON.
- bool is_note_off () const

  Returns true if m_status is EVENT_NOTE_OFF.
- void print ()

  Prints out the timestamp, data size, the current status byte, any SYSEX data if present, or the two data bytes for the status byte.

Friends

- class **sequence**

6.3.1 Detailed Description

A MIDI event consists of 3 bytes:

```
-#  Status byte, 1sssnnn, where the sss bits specify the type of
    message, and the nnnn bits denote the channel number.
    The status byte always starts with 0.
-#  The first data byte, 0xxxxxxx, where the data byte always
    start with 0, and the xxxxxxx values range from 0 to 127.
-#  The second data byte, 0xxxxxxx.
```

This class may have too many member functions.

6.3.2 Member Function Documentation

6.3.2.1 bool event::operator< (const event & *a_rhsevent*) const

Otherwise, it returns true if the current timestamp is less than the event's timestamp.

Warning

The less-than operator is supposed to support a "strict weak ordering", and is supposed to leave equivalent values in the same order they were before the sort. However, every time we load and save our sample MIDI file, events get reversed. Here are program-changes that get reversed:

```
    Save N:     0070: 6E 00 C4 48 00 C4 0C 00  C4 57 00 C4 19 00 C4 26
    Save N+1:   0070: 6E 00 C4 26 00 C4 19 00  C4 57 00 C4 0C 00 C4 48

The 0070 is the offset within the versions of the
b4uacuse-seq24.midi file.
```

6.3.2.2 void event::mod_timestamp (unsigned long *a_mod*) [inline]

Parameters

a_mod	The value to mod the timestamp against.

Returns

Returns a value ranging from 0 to a_mod-1.

6.3.2.3 void event::set_status (char *a_status*)

If a_status is a non-channel event, then the channel portion of the status is cleared.

6.3.2.4 bool event::append_sysex (unsigned char * *a_data,* long *a_size*)

First, a buffer of size m_size+a_size is created. The existing SYSEX data (stored in m_sysex) is copied to this buffer. Then the data represented by a_data and a_size is appended to that data buffer. Then the original SYSEX buffer, m_sysex, is deleted, and m_sysex is assigned to the new buffer..

Warning

This function does not check any pointers.

Parameters

a_data	Provides the additional SYSEX data.
a_size	Provides the size of the additional SYSEX data.

Returns

Returns false if there was an EVENT_SYSEX_END byte in the appended data.

6.4 font Class Reference

This class provides a wrapper for rendering fonts that are encoded as a 16 x 16 pixmap file in XPM format.

Public Types

- enum Color {
  BLACK,
  WHITE,
  BLACK_ON_YELLOW,
  YELLOW_ON_BLACK }

Public Member Functions

- font ()

  Rote default constructor.
- void init (Glib::RefPtr< Gdk::Window > a_window)

  Initialization function for a window on which fonts will be drawn.
- void render_string_on_drawable (Glib::RefPtr< Gdk::GC > m_gc, int x, int y, Glib::RefPtr< Gdk::Drawable > a_draw, const char *str, font::Color col)

  Draws a text string.

6.4.1 Member Enumeration Documentation

6.4.1.1 enum font::Color

Enumerator

BLACK A simple enumeration to describe the basic colors used in writing text. Basically, these two values cause the selection of one or another pixmap (font_b_xpm and font_w_xpm). We've added two more pixmaps to draw black text on a yellow background (font_y.xpm) and yellow text on a black background (font_yb.xpm).

```
The first supported color.  A black font on a white
background.
```

WHITE The second supported color. A white font on a black background.

BLACK_ON_YELLOW A new color, for drawing black text on a yellow background.

YELLOW_ON_BLACK A new color, for drawing yellow text on a black background.

6.4.2 Member Function Documentation

6.4.2.1 void font::init (Glib::RefPtr< Gdk::Window > *a_window*)

This function loads two pixmaps that contain the characters to be used to draw text strings. Both pixmaps provide a 16 x 16 grid of boxes, and each box contains one of the 256 characters in this font set.

One pixmap has white characters on a black background, and other other has black characters on a white background. See the descriptions of the c_text_x and c_text_y variables in the globals module.

6.4.2.2 void font::render_string_on_drawable (Glib::RefPtr< Gdk::GC > *a_gc,* int *x,* int *y,* Glib::RefPtr< Gdk::Drawable > *a_draw,* const char * *str,* font::Color *col*)

This function grabs the proper font bitmap, extracts the current character pixmap from it, and slaps it down where it needs to be to render the character in the string.

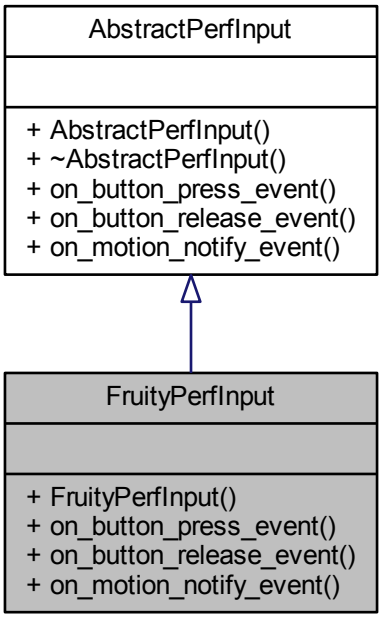
Parameters

a_gc	Provides the graphics context for drawing the text using GTK+.
x	The horizontal location of the text.
y	The vertical location of the text.
a_draw	The drawable object on which to draw the text.
str	The string to draw. Should use a constant string reference instead.
col	The font color to use to draw the string. The only support values are font::BLACK and font::WHITE, and the correct colors are provided by selecting one of two font pixmaps, as described in the init() function.

6.5 FruityPerfInput Class Reference

Implements the performance input of that certain fruity sequencer that people seem to like.

Inheritance diagram for FruityPerfInput:



Public Member Functions

- bool on_button_press_event (GdkEventButton *a_ev, perfroll &roll)

  Handles a button-press event in the Fruity manner.
- bool on_button_release_event (GdkEventButton *a_ev, perfroll &roll)

  Handles a button-release event.
- bool on_motion_notify_event (GdkEventMotion *a_ev, perfroll &roll)

  Handles a Fruity motion-notify event.

6.6 keybindentry Class Reference

Class for management of application key-bindings.

Inherits Entry.

Public Member Functions

- keybindentry (type t, unsigned int *location_to_write=nullptr, perform *p=nullptr, long s=0)

  This constructor initializes the member with values dependent on the value type provided in the first parameter.
- void set (unsigned int val)

  Gets the key name from the integer value; if there is one, then it is printed into a temporary buffer, otherwise the value is printed into that buffer as is.
- virtual bool on_key_press_event (GdkEventKey *event)

  Handles a key press by calling set() with the event's key value.

Friends

- class **options**

6.6.1 Member Function Documentation

6.6.1.1 void keybindentry::set (unsigned int *val*)

Then we call set_text(buf). The set_width_char() function is then called.

6.6.1.2 bool keybindentry::on_key_press_event (GdkEventKey * *event*) [virtual]

This value is used to set the event or key depending on the value of m_type.

6.7 lash Class Reference

This class supports LASH operations, if compiled with LASH support (i.e LASH_SUPPORT is defined).

Public Member Functions

- lash (int argc, char **argv)
  This constructor calls lash_extract(), using the command-line arguments, if LASH_SUPPORT is enabled.
- void init (perform *perform)
  Initializes LASH support, if enabled.
- void set_alsa_client_id (int id)
  Make ourselves a LASH ALSA client.
- void start ()
  Process any LASH events every 250 msec, which is an arbitrarily chosen interval.

6.7.1 Detailed Description

All of the #ifdef skeleton work is done in this class in such a way that any other part of the code can use this class whether or not lash support is actually built in; the functions will just do nothing.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 lash::lash (int *argc*, char ** *argv*)

We fixed the crazy usage of argc and argv here and in the client code in the seq24 module.

6.8 maintime Class Reference

This class provides the drawing of the progress bar at the top of the main window, along with the "pills" that move in time with the measures.

Inherits DrawingArea.

Public Member Functions

- maintime ()
  This constructor sets up the colors black, white, and grey, and then allocates them.
- int idle_progress (long a_ticks)
  This function clears the window, sets the foreground to black, draws the "time" window's rectangle, and more.

6.8.1 Constructor & Destructor Documentation

6.8.1.1 maintime::maintime ( )

In the constructor you can only allocate colors; get_window() would return 0 because the windows has not yet been realized.

6.8.2 Member Function Documentation

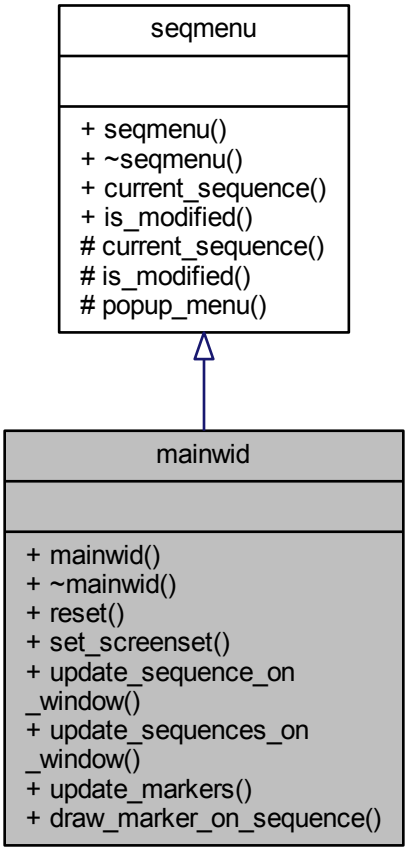
6.8.2.1 int maintime::idle_progress (long *a_ticks*)

Idle hands do the devil's work. We need to figure at a high level what this routine draws, what a maintime is, and where it is located.

6.9 mainwid Class Reference

This class implement the piano roll area of the application.

Inheritance diagram for mainwid:



Public Member Functions

- mainwid (perform *a_p)

  Static array of characters for use in toggling patterns.
- ~mainwid ()

  A rote destructor.
- void reset ()

  This function redraws everything and queues up a redraw operation.
- void set_screenset (int a_ss)

  Set the current screen set.
- void update_sequence_on_window (int a_seq)

  Updates the image of one sequencer.
- void update_sequences_on_window ()

  Updates the image of multiple sequencers.
- void update_markers (int a_ticks)

  Draw the cursors (long vertical bars) on each sequence, so that they follow the playing progress of each sequence in the mainwid (Patterns Panel.)
- void draw_marker_on_sequence (int a_seq, int a_tick)

  Does the actual drawing of one pattern/sequence position marker, a vertical progress bar.

Additional Inherited Members

6.9.1 Constructor & Destructor Documentation

6.9.1.1 mainwid::mainwid (perform * *a_p*)

These look like the "Sequence toggle keys" in the Options / Keyboard dialog, except that they are upper-case here, and lower-case in that configuration dialog.

Obsolete Its only use was in this module, and is commented out below, replaced by another lookup method.

```
const char mainwid::m_seq_to_char[c_seqs_in_set] =
{
    '1', 'Q', 'A', 'Z',
    '2', 'W', 'S', 'X',
    '3', 'E', 'D', 'C',
    '4', 'R', 'F', 'V',
    '5', 'T', 'G', 'B',
    '6', 'Y', 'H', 'N',
    '7', 'U', 'J', 'M',
    '8', 'I', 'K', ','
};
```

This constructor sets a lot of the members, but not all. And it asks for a size of c_mainwid_x by c_mainwid_y. It adds GDK masks for button presses, releases, and motion, and key presses and focus changes.

6.9.2 Member Function Documentation

6.9.2.1 void mainwid::draw_marker_on_sequence (int *a_seq,* int *a_tick*)

If the sequence has no events, this function doesn't bother even drawing a position marker.

6.10 mainwnd Class Reference

This class implements the functionality of the main window of the application, except for the Patterns Panel functionality, which is implemented in the mainwid class.

Inherits Window, and performcallback.

Public Member Functions

- mainwnd (perform *a_p)
  The constructor the main window of the application.
- ~mainwnd ()
  This destructor must explicitly delete some allocated resources.
- void open_file (const std::string &)
  Opens a MIDI file.

6.10.1 Constructor & Destructor Documentation

6.10.1.1 mainwnd::mainwnd (perform * *a_p*)

This constructor is way too large; it would be nicer to provide a number of well-named initialization functions.

Parameters

a_p	Refers to the main performance object.

Todo Offload most of the work into an initialization function like options does; make the perform parameter a reference.

Todo Better as a member function.

File menu items, their accelerator keys, and their hot keys.

View menu items and their hot keys.

Help menu items

Top panel items, including the logo (updated for the new version of this application) and the "timeline" progress bar.

6.11 mastermidibus Class Reference

The class that "supervises" all of the midibus objects?

Public Member Functions

- mastermidibus ()
  The mastermidibus constructor fills the array with our busses.
- ~mastermidibus ()
  The destructor deletes all of the output busses, clears out the ALSA events, stops and frees the queue, and closes ALSA for this application.
- void init ()
  Initialize the mastermidibus.
- snd_seq_t * get_alsa_seq () const
  'Getter' function for member m_alsa_seq
- int get_num_out_buses () const
  'Getter' function for member m_num_out_buses
- int get_num_in_buses () const
  'Getter' function for member m_num_in_buses
- void set_bpm (int a_bpm)
  Set the BPM value (beats per minute).
- void set_ppqn (int a_ppqn)

Set the PPQN value (parts per quarter note).

- int get_bpm () const

  'Getter' function for member m_bpm

- int get_ppqn () const

  'Getter' function for member m_ppqn

- std::string get_midi_out_bus_name (int a_bus)

  Get the MIDI output buss name for the given (legal) buss number.

- std::string get_midi_in_bus_name (int a_bus)

  Get the MIDI input buss name for the given (legal) buss number.

- void print ()

  Print some information about the available MIDI output busses.

- void flush ()

  Flushes our local queue events out into ALSA.

- void start ()

  Starts all of the configured output busses up to m_num_out_buses.

- void stop ()

  Stops each of the output busses.

- void clock (long a_tick)

  Generates the MIDI clock for each of the output busses.

- void continue_from (long a_tick)

  Gets the output busses running again.

- void init_clock (long a_tick)

  Initializes the clock of each of the output busses.

- int poll_for_midi ()

  Initiate a poll() on the existing poll descriptors.

- bool is_more_input ()

  Test the ALSA sequencer to see if any more input is pending.

- bool get_midi_event (event *a_in)

  Grab a MIDI event.

- void set_sequence_input (bool a_state, sequence *a_seq)

  Set the input sequence object, and set the m_dumping_input value to the given state.

- bool is_dumping () const

  'Getter' function for member m_dumping_input

- sequence * get_sequence () const

  'Getter' function for member m_seq

- void sysex (event *a_event)

  Handle the sending of SYSEX events.

- void port_start (int a_client, int a_port)

  Start the given ALSA MIDI port.

- void port_exit (int a_client, int a_port)

  Turn off the given port for the given client.

- void play (unsigned char a_bus, event *a_e24, unsigned char a_channel)

  Handle the playing of MIDI events on the MIDI buss given by the parameter, as long as it is a legal buss number.

- void set_clock (unsigned char a_bus, clock_e a_clock_type)

  Set the clock for the given (legal) buss number.

- clock_e get_clock (unsigned char a_bus)

  Get the clock for the given (legal) buss number.

- void set_input (unsigned char a_bus, bool a_inputing)

  Set the status of the given input buss, if a legal buss number.

- bool get_input (unsigned char a_bus)

  Get the input for the given (legal) buss number.

6.11.1 Member Function Documentation

6.11.1.1 void mastermidibus::init ( )

It initializes 16 MIDI output busses, a hardwired constant, 16. Only one MIDI input buss is initialized.

6.11.1.2 void mastermidibus::set_bpm (int *a_bpm*)

This is done by creating an ALSA tempo structure, adding tempo information to it, and then setting the ALSA sequencer object with this information.

Threadsafe

6.11.1.3 void mastermidibus::set_ppqn (int *a_ppqn*)

This is done by creating an ALSA tempo structure, adding tempo information to it, and then setting the ALSA sequencer object with this information.

Threadsafe

6.11.1.4 void mastermidibus::flush ( )

Threadsafe

6.11.1.5 void mastermidibus::start ( )

Threadsafe

6.11.1.6 void mastermidibus::stop ( )

Threadsafe

6.11.1.7 void mastermidibus::clock (long *a_tick*)

Threadsafe

6.11.1.8 void mastermidibus::continue_from (long *a_tick*)

Threadsafe

6.11.1.9 void mastermidibus::init_clock (long *a_tick*)

Threadsafe

6.11.1.10 bool mastermidibus::is_more_input ( )

Threadsafe

Does this function really need to be locked?

6.11.1.11 bool mastermidibus::get_midi_event (event * *a_in*)

Threadsafe

6.11.1.12 void mastermidibus::set_sequence_input (bool *a_state,* sequence * *a_seq*)

Threadsafe

6.11.1.13 void mastermidibus::sysex (event * *a_ev*)

Threadsafe

6.11.1.14 void mastermidibus::port_start (int *a_client,* int *a_port*)

Threadsafe Quite a lot is done during the lock!

6.11.1.15 void mastermidibus::port_exit (int *a_client,* int *a_port*)

Threadsafe

6.11.1.16 void mastermidibus::play (unsigned char *a_bus,* event * *a_e24,* unsigned char *a_channel*)

Threadsafe

6.11.1.17 void mastermidibus::set_clock (unsigned char *a_bus,* clock_e *a_clock_type*)

The legality checks are a little loose, however.

Threadsafe

6.11.1.18 void mastermidibus::set_input (unsigned char *a_bus,* bool *a_inputing*)

Why is another buss-count constant, and a global one at that, being used? And I thought there was only one input buss anyway!

Threadsafe

6.12 midibus Class Reference

Provides a class for handling the MIDI buss on Linux.

Public Member Functions

- midibus (int a_localclient, int a_destclient, int a_destport, snd_seq_t *a_seq, const char *a_client_name, const char *a_port_name, int a_id, int a_queue)

  Provides a constructor with client number, port number, ALSA sequencer support, name of client, name of port.
- midibus (int a_localclient, snd_seq_t *a_seq, int a_id, int a_queue)

  Secondary constructor.
- ~midibus ()

  A rote empty destructor.
- bool init_out ()

  Initialize the MIDI output port.
- bool init_in ()

  Initialize the MIDI input port.
- bool deinit_in ()

  Deinitialize the MIDI input?
- bool init_out_sub ()

  Initialize the output in a different way?
- bool init_in_sub ()

  Initialize the output in a different way?
- void print ()

  Prints m_name.
- const std::string & get_name () const

  'Getter' function for member n_name
- int get_id () const

  'Getter' function for member m_id
- void play (event *a_e24, unsigned char a_channel)

This play() function takes a native event, encodes it to ALSA event, and puts it in the queue.

- void sysex (event *a_e24)

  Takes a native SYSEX event, encodes it to an ALSA event, and then puts it in the queue.
- void start ()

  This function gets the MIDI clock a-runnin', if the clock type is not e_clock_off.
- void stop ()

  Stop the MIDI buss.
- void clock (long a_tick)

  Generates the MIDI clock, starting at the given tick value.
- void continue_from (long a_tick)

  Continue from the given tick.
- void init_clock (long a_tick)

  Initialize the clock, continuing from the given tick.
- void set_clock (clock_e a_clock_type)

  'Setter' function for member m_clock_type
- clock_e get_clock () const

  'Getter' function for member m_clock_type
- void set_input (bool a_inputing)

  Input functions.
- bool get_input () const

  'Getter' function for member m_inputing
- void flush ()

  Flushes our local queue events out into ALSA.
- int get_client () const

  'Getter' function for member m_dest_addr_client The address of client.
- int get_port () const

  'Getter' function for member m_dest_addr_port

Static Public Member Functions

- static void set_clock_mod (int a_clock_mod)

  Set the clock mod to the given value, if legal.
- static int get_clock_mod ()

  Get the clock mod.

Friends

- class mastermidibus

  The master MIDI bus sets up the buss.

6.12.1 Member Function Documentation

6.12.1.1 void midibus::set_input (bool *a_inputing*)

Set status to of "inputting" to the given value.

If the parameter is true, then init_in() is called; otherwise, deinit_in() is called.

6.13 midifile Class Reference

This class handles the parsing and writing of MIDI files.

Public Member Functions

- midifile (const std::string &name, bool propformat=true)

  Principal constructor.
- ~midifile ()

  A rote destructor.
- bool parse (perform *a_perf, int a_screen_set)

  This function opens a binary MIDI file and parses it into sequences and other application objects.
- bool write (perform *a_perf)

  Write the whole MIDI data and Seq24 information out to the file.

6.13.1 Detailed Description

In addition to the standard MIDI tracks, it also handles some "private" or "proprietary" tracks specific to Seq24. It does not, however, handle SYSEX events.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 midifile::midifile (const std::string & *a_name,* bool *propformat* = `true`)

Parameters

a_name	Provides the name of the MIDI file to be read or written.
propformat	If true, write out the MIDI file using the MIDI-compliant sequencer-specific prefix in from of the seq24-specific SeqSpec tags defined in the globals module. This option is true by default. Note that this option is only used in writing; reading can handle either format transparently.

6.13.3 Member Function Documentation

6.13.3.1 bool midifile::parse (perform * *a_perf,* int *a_screen_set*)

In addition to the standard MIDI track data in a normal track, Seq24 adds four sequencer-specific events just before the end of the track:

```
    c_triggers_new:     SeqSpec FF 7F 1C 24 24 00 08 00 00 ...
    c_midibus:          SeqSpec FF 7F 05 24 24 00 01 00
    c_timesig:          SeqSpec FF 7F 06 24 24 00 06 04 04
    c_midich:           SeqSpec FF 7F 05 24 24 00 02 06

Standard MIDI provides for the port and channel specifications, but
they are apparently considered obsolete:

Obsolete meta-event:               Replacement:

    MIDI port (buss):   FF 21 01 po     Device (port) name: FF 09 len text
    MIDI channel:       FF 20 01 ch

What do other applications use for specifying port/channel?
```

6.14 options Class Reference

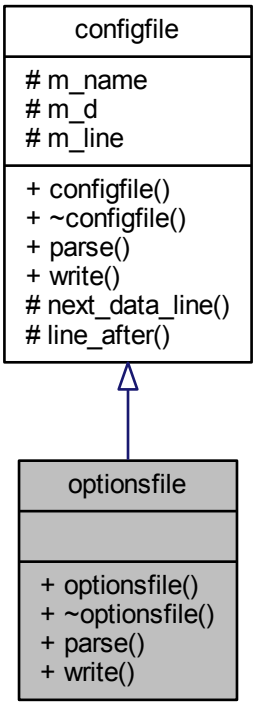
This class supports a full tabbed options dialog.

Inherits Dialog.

6.15 optionsfile Class Reference

Provides a file for reading and writing the application' main configuration file.

Inheritance diagram for optionsfile:



Public Member Functions

- optionsfile (const std::string &a_name)
  Principal constructor.
- ~optionsfile ()
  A rote destructor.
- bool parse (perform *a_perf)
  Parse the ~/.seq24rc file.
- bool write (perform *a_perf)
  This options-writing function is just about as complex as the options-reading function.

Additional Inherited Members

6.15.1 Member Function Documentation

6.15.1.1 bool optionsfile::parse (perform * *a_perf*) [virtual]

[midi-control]

Get the number of sequence definitions provided in the [midi-control] section. Ranges from 32 on up. Then read in all of the sequence lines. The first 32 apply to the first screen set. There can also be a comment line "# mute in

group" followed by 32 more lines. Then there are addditional comments and single lines for BPM up, BPM down, Screen Set Up, Screen Set Down, Mod Replace, Mod Snapshot, Mod Queue, Mod Gmute, Mod Glearn, and Screen Set Play. These are all forms of MIDI automation useful to control the playback while not sitting near the computer.

[mute-group]

The mute-group starts with a line that indicates up to 32 mute-groups are defined. A common value is 1024, which means there are 32 groups times 32 keys. But this value is currently thrown away. This value is followed by 32 lines of data, each contained 4 sets of 8 settings. See the seq24-doc project on GitHub for a much more detailed description of this section.

[midi-clock]

The MIDI-clock section defines the clocking value for up to 16 output busses. The first number, 16, indicates how many busses are specified. Generally, these busses are shown to the user with names such as "[1] seq24 1".

[keyboard-control]

The keyboard control defines the keys that will toggle the stage of each of up to 32 patterns in a pattern/sequence box. These keys are displayed in each box as a reminder. The first number specifies the Key number, and the second number specifies the Sequence number.

[keyboard-group]

The keyboard group specifies more automation for the application. The first number specifies the Key number, and the second number specifies the Group number. This section should be better described in the seq24-doc project on GitHub.

[jack-transport]

This section covers various JACK settings, one setting per line. In order, the following numbers are specfied:

```
-   jack_transport - Enable sync with JACK Transport.
-   jack_master - Seq24 will attempt to serve as JACK Master.
-   jack_master_cond - Seq24 will fail to be Master if there is
    already a Master set.
-   jack_start_mode:
    -   0 = Playback will be in Live mode.  Use this to allow
        muting and unmuting of loops.
    -   1 = Playback will use the Song Editor's data.
```

[midi-input]

This section covers the MIDI input busses, and has a format similar to "[midi-clock]". Generally, these busses are shown to the user with names such as "[1] seq24 1", and currently there is only one input buss. The first field is the port number, and the second number indicates whether it is disabled (0), or enabled (1).

[midi-clock-mod-ticks]

This section covers.... One common value is 64.

[manual-alsa-ports]

This section covers.... Set to 1 if you want seq24 to create its own ALSA ports and not connect to other clients.

[last-used-dir]

This section simply holds the last path-name that was used to read or write a MIDI file. We still need to add a check for a valid path, and currently the path must start with a "/", so it is not suitable for Windows.

[interaction-method]

This section specified the kind of mouse interaction.

- 0 = 'seq24' (original Seq24 method).
- 1 = 'fruity' (similar to a certain fruity sequencer we like).

The second data line is set to "1" if Mod4 can be used to keep seq24 in note-adding mode even after the right-click is released, and "0" otherwise.

Implements configfile.

6.16 perfedit Class Reference

This class supports a Performance Editor that is used to arrange the patterns/sequences defined in the patterns panel, I think.

Inherits Window.

Public Member Functions

- perfedit (perform *a_perf)

  Principal constructor, has a pointer to a perform object.
- ~perfedit ()

  This rote constructor does nothing.
- void init_before_show ()

  This function forwards its call to the perfroll function of the same name.
- void is_modified (bool flag)

  'Setter' function for member m_modified
- bool is_modified () const

  'Getter' function for member m_modified

6.16.1 Detailed Description

It has a seqroll and piano roll? No, it has a perform, a perfnames, a perfroll, and a perftime.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 perfedit::perfedit (perform * *a_perf*)

We've reordered the pointer members and put them in the initializer list to make the constructor a bit cleaner.

Parameters

a_perf	Refers to the main performance object.

Todo Offload most of the work into an initialization function like options does; make the perform parameter a reference.

6.16.2.2 perfedit::~perfedit ( )

We're going to have to run the application through valgrind to make sure that nothing is left behind.

6.16.3 Member Function Documentation

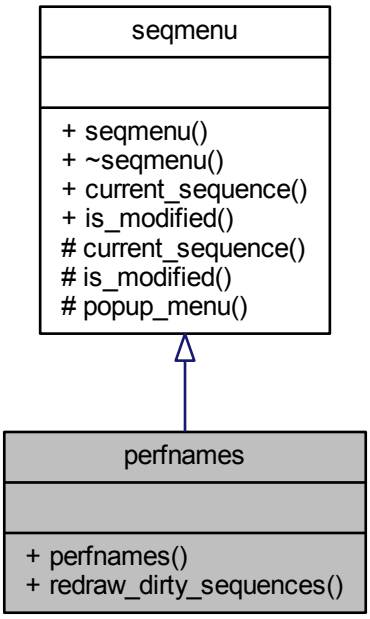
6.16.3.1 void perfedit::init_before_show ( )

It does not seem to need to also forward to the perftime function of the same name.

6.17 perfnames Class Reference

This class implements the left-side keyboard in the patterns window.

Inheritance diagram for perfnames:



Public Member Functions

- perfnames (perform *a_perf, Gtk::Adjustment *a_vadjust)

  Adjustments to the performance window.
- void redraw_dirty_sequences ()

  Redraws sequences that have been modified.

Additional Inherited Members

6.17.1 Constructor & Destructor Documentation

6.17.1.1 perfnames::perfnames (perform * *a_perf,* Gtk::Adjustment * *a_vadjust*)

Sequences that don't have events show up as black-on-yellow. This feature is enabled by default. To disable this feature, configure the build with the –disable-highlight option.

```
#define HIGHLIGHT_EMPTY_SEQS    // undefine for normal empty seqs
```

Principal constructor for this user-interface object.

6.18 perform Class Reference

This class supports the performance mode.

Public Types

- typedef std::map< unsigned int, long > SlotMap

  This typedef defines a map in which the key is the keycode, that is, the integer value of a keystroke, and the value is the pattern/sequence number or slot.
- typedef std::map< long, unsigned int > RevSlotMap

  This typedef is like SlotMap, but used for lookup in the other direction.

Public Member Functions

- perform ()

  This construction initializes a vast number of member variables, some of them public!
- ~perform ()

  The destructor sets some running flags to false, signals this condition, then joins the input and output threads if the were launched.
- mastermidibus & master_bus ()

  'Getter' function for member m_master_bus
- bool is_running () const

  'Getter' function for member m_running
- bool is_learn_mode () const

  'Getter' function for member m_mode_group_learn
- void init ()

  Initializes the master MIDI bus.
- void clear_all ()

  Clears all of the patterns/sequences.
- void launch_input_thread ()

  Creates the input thread using input_thread_func().
- void launch_output_thread ()

  Creates the output thread using output_thread_func().
- void init_jack ()

  Initializes JACK support, if JACK_SUPPORT is defined.
- void deinit_jack ()

  Tears down the JACK infrastructure.
- void add_sequence (sequence *a_seq, int a_perf)

  Adds a pattern/sequence pointer to the list of patterns.
- void delete_sequence (int a_num)

  Deletes a pattern/sequence by number.
- bool is_sequence_in_edit (int a_num)

  Check if the pattern/sequence, given by number, has an edit in progress.
- void clear_sequence_triggers (int a_seq)

  Clears the patterns/sequence for the given sequence, if it is active.
- bool is_sequence_valid (int a_sequence) const

  Provides common code to check for the bounds of a sequence number.
- bool is_sequence_invalid (int a_sequence) const

  Provides common code to check for the bounds of a sequence number.
- void set_left_tick (long a_tick)

  Set the left marker at the given tick.
- long get_left_tick () const

  'Getter' function for member m_left_tick
- void set_starting_tick (long a_tick)

  'Setter' function for member m_starting_tick

- long get_starting_tick () const

  'Getter' function for member m_starting_tick
- void set_right_tick (long a_tick)

  Set the right marker at the given tick.
- long get_right_tick () const

  'Getter' function for member m_right_tick
- void move_triggers (bool a_direction)

  If the left tick is less than the right tick, then, for each sequence that is active, its triggers are moved by the difference between the right and left in the specified direction.
- void copy_triggers ()

  If the left tick is less than the right tick, then, for each sequence that is active, its triggers are copied, offset by the difference between the right and left.
- void push_trigger_undo ()

  For every active sequence, call that sequence's push_trigger_undo() function.
- void pop_trigger_undo ()

  For every active sequence, call that sequence's pop_trigger_undo() function.
- void print ()

  An information printing function with its body commented out.
- midi_control * get_midi_control_toggle (unsigned int a_seq)

  Retrieves a value from m_midi_cc_toggle[].
- midi_control * get_midi_control_on (unsigned int a_seq)

  Retrieves a value from m_midi_cc_on[].
- midi_control * get_midi_control_off (unsigned int a_seq)

  Retrieves a value from m_midi_cc_off[].
- void handle_midi_control (int a_control, bool a_state)

  Handle the MIDI Control values that provide some automation for the application.
- void set_screen_set_notepad (int a_screen_set, std::string *a_note)

  Copies the given string into m_screen_set_notepad[].
- std::string * get_screen_set_notepad (int a_screen_set)

  Retrieves the given string from m_screen_set_notepad[].
- void set_screenset (int a_ss)

  Sets the m_screen_set value (the index or ID of the current screen set).
- int get_screenset () const

  'Getter' function for member m_screen_set
- void set_playing_screenset ()

  Sets the screen set that is active, based on the value of m_playing_screen.
- int get_playing_screenset () const

  'Getter' function for member m_playing_screen
- void mute_group_tracks ()

  Will need to study this one more closely.
- void select_and_mute_group (int a_g_group)

  Select a mute group and then mutes the track in the group.
- void set_mode_group_mute ()

  'Setter' function for member m_mode_group
- void unset_mode_group_mute ()

  'Setter' function for member m_mode_group Unsets this member.
- void select_group_mute (int a_g_mute)

  Makes some checks and sets the group mute flag.
- void set_mode_group_learn ()

  Sets the group-mute mode, then the group-learn mode, then notifies all of the notification subscribers.
- void unset_mode_group_learn ()

Notifies all of the notification subscribers that group-learn is being turned off.

- void select_mute_group (int a_group)

  Will need to study this one more closely.
- void start (bool a_state)

  If JACK is note running, call inner_start() with the given state.
- void stop ()

  If JACK is note running, call inner_stop().
- bool jack_session_event ()

  Writes the MIDI file named "<jack session dir>-file.mid" using a mididfile object, quits if told to by JACK, and can free the JACK session event.
- void start_jack ()

  If JACK is supported, starts the JACK transport.
- void stop_jack ()

  If JACK is supported, stops the JACK transport.
- void position_jack (bool a_state)

  If JACK is supported and running, sets the position of the transport.
- void off_sequences ()

  For all active patterns/sequences, set the playing state to false.
- void all_notes_off ()

  For all active patterns/sequences, turn off its playing notes.
- void set_active (int a_sequence, bool a_active)

  Sets or unsets the active state of the given pattern/sequence number.
- void set_was_active (int a_sequence)

  Sets was-active flags: main, edit, perf, and names.
- bool is_active (int a_sequence)

  Checks the pattern/sequence for activity.
- bool is_dirty_main (int a_sequence)

  Checks the pattern/sequence for main-dirtiness.
- bool is_dirty_edit (int a_sequence)

  Checks the pattern/sequence for edit-dirtiness.
- bool is_dirty_perf (int a_sequence)

  Checks the pattern/sequence for perf-dirtiness.
- bool is_dirty_names (int a_sequence)

  Checks the pattern/sequence for names-dirtiness.
- void new_sequence (int a_sequence)

  Creates a new pattern/sequence for the given slot, and sets the new pattern's master MIDI bus address.
- sequence * get_sequence (int a_sequence)

  Retrieves the actual sequence, based on the pattern/sequence number.
- void reset_sequences ()

  For all active patterns/sequences, get its playing state, turn off the playing notes, set playing to false, zero the markers, and, if not in playback mode, restore the playing state.
- void play (long a_tick)

  Plays all notes to the current tick.
- void set_orig_ticks (long a_tick)

  For every pattern/sequence that is active, sets the "original ticks" value for the pattern.
- void set_bpm (int a_bpm)

  Sets the value of the BPM into the master MIDI buss, after making sure it is squelched to be between 20 and 500.
- int get_bpm ()

  Retrieves the BPM setting of the master MIDI buss.
- void set_looping (bool a_looping)

  'Setter' function for member m_looping

- void set_sequence_control_status (int a_status)

  If the given status is present in the c_status_snapshot, the playing state is saved.
- void unset_sequence_control_status (int a_status)

  If the given status is present in the c_status_snapshot, the playing state is restored.
- void set_group_mute_state (int a_g_track, bool a_mute_state)

  'Setter' function for member m_mute_group
- bool get_group_mute_state (int a_g_track)

  'Getter' function for member m_mute_group
- void mute_all_tracks ()

  Mutes all tracks in the current set of active patterns/sequences.
- void output_func ()

  Performance output function.
- void input_func ()

  This function is called by input_thread_func().
- long get_max_trigger ()

  Locates the largest trigger value among the active sequences.
- void set_offset (int a_offset)

  Calculates the offset into the screen sets.
- void save_playing_state ()

  For all active patterns/sequences, this function gets the playing status and saves it in m_sequence_state[i].
- void restore_playing_state ()

  For all active patterns/sequences, this function gets the playing status from m_sequence_state[i] and sets it for the sequence.
- void set_key_event (unsigned int keycode, long sequence_slot)

  At construction time, this function sets up one keycode and one event slot.
- void set_key_group (unsigned int keycode, long group_slot)

  At construction time, this function sets up one keycode and one group slot.
- bool show_ui_sequence_key () const

  ***Accessor** m_show_ui_sequency_key*

Data Fields

- unsigned int m_key_bpm_up

  Provides key assignments for some key sequencer features.

Friends

- class **midifile**
- class **optionsfile**
- class **options**
- int jack_sync_callback (jack_transport_state_t state, jack_position_t *pos, void *arg)

  This JACK synchronization callback informs the specified perform object of the current state and parameters of JACK.
- void jack_shutdown (void *arg)

  Shutdown JACK by clearing the perform::m_jack_running flag.
- void jack_timebase_callback (jack_transport_state_t state, jack_nframes_t nframes, jack_position_t *pos, int new_pos, void *arg)

  This function...

6.18.1 Detailed Description

It has way too many data members, many of the public. Might be ripe for refactoring.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 perform::~perform ( )

Finally, any active patterns/sequences are deleted.

6.18.3 Member Function Documentation

6.18.3.1 void perform::launch_input_thread ( )

This might be a good candidate for a small thread class derived from a small base class.

6.18.3.2 void perform::launch_output_thread ( )

This might be a good candidate for a small thread class derived from a small base class.

6.18.3.3 void perform::add_sequence (sequence * *a_seq*, int *a_perf*)

No check is made for a null pointer.

Check for preferred. This occurs if a_perf is in the valid range (0 to c_max_sequence) and it is not active. If preferred, then add it and activate it.

Otherwise, iterate through all patterns from a_perf to c_max_sequence and add and activate the first one that is not active.

Is there a usefulness in setting the sequence's tag?

Warning

The logic of the if-statement in this function was such that *a_perf* could be out-of-bounds in the else-clause. We reworked the logic to be airtight. This bug was caught by gcc 4.8.3 on CentOS, but not on gcc 4.9.3 on Debian Sid! However, this decision-making seems goofy, and we ought to revisit it!

Parameters

a_seq	The number or index of the pattern/sequence to add. If this value is out-of-range, then it is ignored.
a_perf	The performance number of the pattern?

6.18.3.4 void perform::clear_sequence_triggers (int *a_seq*)

Parameters

a_seq	Provides the desired sequence. Hopefull, the is_active() function validates this value.

6.18.3.5 bool perform::is_sequence_valid (int *a_sequence*) const [inline]

Returns

Returns true if the sequence number is valid.

6.18.3.6 bool perform::is_sequence_invalid (int *a_sequence*) const [inline]

Returns

Returns true if the sequence number is invalid.

6.18.3.7 void perform::move_triggers (bool *a_direction*)

Parameters

a_direction	Specifies the desired direction; false = left, true = right.

6.18.3.8 void perform::copy_triggers ( )

This copies the triggers between the L marker and R marker to the R marker.

6.18.3.9 midi_control * perform::get_midi_control_toggle (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object. Note that this value is unsigned simply to make the legality check of the parameter easier.

6.18.3.10 midi_control * perform::get_midi_control_on (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object.

6.18.3.11 midi_control * perform::get_midi_control_off (unsigned int *a_seq*)

Parameters

a_seq	Provides a control value (such as c_midi_control_bpm_up) to use to retrieve the desired midi_control object.

6.18.3.12 void perform::set_screen_set_notepad (int *a_screen_set,* std::string * *a_notepad*)

Parameters

a_screen_set	The ID number of the string set, an index into the m_screen_set_xxx[] arrays.
a_notepad	Provides the string date to copy into the notepad. Not sure why a pointer is used, instead of nice "const std::string &" parameter. And this pointer isn't checked.

6.18.3.13 std::string * perform::get_screen_set_notepad (int *a_screen_set*)

Parameters

a_screen_set	The ID number of the string set, an index into the m_screen_set_xxx[] arrays.

6.18.3.14 void perform::set_screenset (int *a_ss*)

Parameters

a_ss	The index of the desired string set. It is forced to range from 0 to c_max_sets - 1.

6.18.3.15 void perform::set_playing_screenset ( )

For each value up to c_seqs_in_set (32), the index of the current sequence in the currently screen set (m_playing↩ _screen) is obtained. If it is active and the sequence actually exists

Modifies m_playing_screen, and mutes the group tracks.

6.18.3.16 void perform::unset_mode_group_learn ( )

Then unsets the group-learn mode flag..

6.18.3.17 void perform::start (bool *a_state*)

Parameters

a_state	What does this state mean?

6.18.3.18 void perform::stop ( )

The logic seems backward her, in that we call inner_stop() if JACK is not running. Or perhaps we misunderstand the meaning of m_jack_running?

6.18.3.19 bool perform::jack_session_event ( )

ca 2015-07-24 Just a note: The OMA (OpenMandrivaAssociation) patch was already applied to seq24 v.0.9.2. It put quotes around the –file argument.

Why are we using a Glib::ustring here? Convenience. But with C++11, we could use a lexical_cast<>. No more ustring, baby!

It doesn't really matter; this function can call Gtk::Main::quit().

6.18.3.20 void perform::position_jack (bool *a_state*)

Warning

A lot of this code is effectively disabled by an early return statement.

6.18.3.21 void perform::all_notes_off ( )

Then flush the MIDI buss.

6.18.3.22 void perform::set_was_active (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

6.18.3.23 bool perform::is_active (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the value of the active-flag, or false if the pattern was invalid.

6.18.3.24 bool perform::is_dirty_main (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-main flag value, before setting it to false. Returns false if the pattern was invalid.

6.18.3.25 bool perform::is_dirty_edit (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-edit flag value, before setting it to false. Returns false if the pattern was invalid.

6.18.3.26 bool perform::is_dirty_perf (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-perf flag value, before setting it to false. Returns false if the pattern/sequence number was invalid.

6.18.3.27 bool perform::is_dirty_names (int *a_sequence*)

Parameters

a_sequence	The pattern number. It is checked for invalidity.

Returns

Returns the was-active-names flag value, before setting it to false. Returns false if the pattern/sequence number was invalid.

6.18.3.28 void perform::new_sequence (int *a_sequence*)

Then it activates the pattern.

It doesn't deal with thrown exceptions.

6.18.3.29 void perform::reset_sequences ( )

Then flush the MIDI buss.

6.18.3.30 void perform::play (long *a_tick*)

Starts the playing of all the patterns/sequences.

This function just runs down the list of sequences and has them dump their events.

Parameters

a_tick	Provides the tick at which to start playing.

6.18.3.31 void perform::set_orig_ticks (long *a_tick*)

Parameters

a_tick	

6.18.3.32 void perform::set_bpm (int *a_bpm*)

The value is set only if neither JACK nor this performance object are running.

6.18.3.33 void perform::set_sequence_control_status (int *a_status*)

Then the given status is OR'd into the m_control_status.

6.18.3.34 void perform::unset_sequence_control_status (int *a_status*)

Then the given status is reversed in m_control_status.

6.18.3.35 void perform::output_func ( )

1. Get delta time (current - last).
2. Get delta ticks from time.
3. Add to current_ticks.
4. Compute prebuffer ticks.
5. Play from current tick to prebuffer.

Figure out how much time we need to sleep, and do it.

6.18.3.36 long perform::get_max_trigger ( )

Returns

Returns the highest trigger value, or zero. It is not clear why this function doesn't return a "no trigger found" value. Is there always at least one trigger, at 0?

6.18.3.37 void perform::set_offset (int *a_offset*) [inline]

Sets m_offset = a_offset * c_mainwnd_rows * c_mainwnd_cols;

Parameters

a_offset	The desired offset.

6.18.3.38 void perform::set_key_event (unsigned int *keycode,* long *sequence_slot*)

It is called 32 times, corresponding the pattern/sequence slots in the Patterns window.

6.18.3.39 void perform::set_key_group (unsigned int *keycode,* long *group_slot*)

It is called 32 times, corresponding the pattern/sequence slots in the Patterns window.

6.18.3.40 bool perform::show_ui_sequence_key ( ) const [inline]

Used in mainwid, options, optionsfile, userfile, and perform.

6.18.4 Friends And Related Function Documentation

6.18.4.1 int jack_sync_callback (jack_transport_state_t *state,* jack_position_t * *pos,* void * *arg*) [friend]

Parameters

state	The JACK Transport state.
pos	The JACK position value.

arg	The pointer to the perform object. Currently not checked for nullity.

6.18.5 Field Documentation

6.18.5.1 unsigned int perform::m_key_bpm_up

Used in mainwnd, options, optionsfile, perfedit, seqroll, userfile, and perform.

6.19 perfroll Class Reference

This class implements the performance roll user interface.

Inherits DrawingArea.

Public Member Functions

- perfroll (perform *a_perf, Gtk::Adjustment *a_hadjust, Gtk::Adjustment *a_vadjust)

  Principal constructor.
- ~perfroll ()

  This destructor deletes the interaction object.
- void set_guides (int a_snap, int a_measure, int a_beat)

  This function sets the snap, measure, and beats members, fills in the background, and queues up a draw operation.
- void update_sizes ()

  Updates the sizes of various items.
- void init_before_show ()

  Sets the roll-lengths ticks member.
- void fill_background_pixmap ()

  This function updates the background of the Performance roll.
- void increment_size ()

  *Increments the value of m_roll_length_ticks by the PPQN * 512, then calls update_sizes().*
- void draw_progress ()

  Draws the progess line that shows where we are in the performance.
- void redraw_dirty_sequences ()

  Redraws patterns/sequences that have been modified.

Friends

- class **FruityPerfInput**
- class **Seq24PerfInput**

6.20 perftime Class Reference

This class implements drawing the piano time at the top of the "performance window", also known as the "song editor".

Inherits DrawingArea.

Public Member Functions

- perftime (perform *a_perf, Gtk::Adjustment *a_hadjust)
  Principal constructor.
- void set_guides (int a_snap, int a_measure)
  Sets the snap value and the measure-length members.
- void increment_size ()
  This function does nothing.

6.20.1 Constructor & Destructor Documentation

6.20.1.1 perftime::perftime (perform * *a_perf,* Gtk::Adjustment * *a_hadjust*)

In the constructor you can only allocate colors; get_window() returns 0 because we have not been realized.

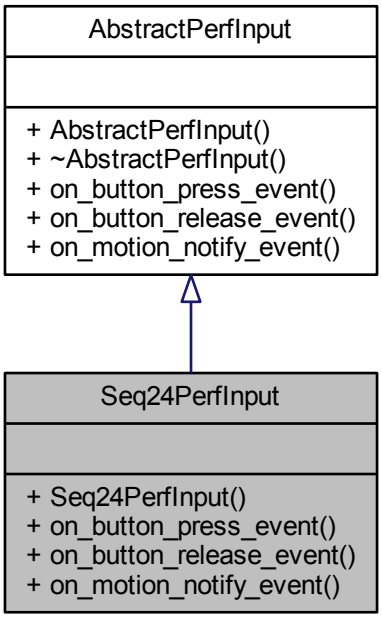
6.21 rect Class Reference

A small helper class representing a rectangle.

6.22 Seq24PerfInput Class Reference

Implements the default performance input characteristics of this application.

Inheritance diagram for Seq24PerfInput:



Public Member Functions

- bool on_button_press_event (GdkEventButton *a_ev, perfroll &roll)

Handles the normal variety of button-press event.

- bool on_button_release_event (GdkEventButton *a_ev, perfroll &roll)

  Handles various button-release events.

- bool on_motion_notify_event (GdkEventMotion *a_ev, perfroll &roll)

  Handles the normal motion-notify event.

6.22.1 Member Function Documentation

6.22.1.1 bool Seq24PerfInput::on_button_press_event (GdkEventButton * *a_ev,* perfroll & *roll*) [virtual]

Is there any easy way to use ctrl-left-click as the middle button here?

Implements AbstractPerfInput.

6.22.1.2 bool Seq24PerfInput::on_button_release_event (GdkEventButton * *a_ev,* perfroll & *roll*) [virtual]

Any use for the middle-button or ctrl-left-click we can add?

Implements AbstractPerfInput.

6.23 Seq24SeqEventInput Struct Reference

This structure implement the normal interaction methods for Seq24.

Public Member Functions

- Seq24SeqEventInput ()

  Default constructor.

- void set_adding (bool a_adding, seqevent &ths)

  Changes the mouse cursor to a pencil or a left pointer in the given seqevent aobject, depending on the first parameter.

- bool on_button_press_event (GdkEventButton *a_ev, seqevent &ths)

  Implements the on-button-press event callback.

- bool on_button_release_event (GdkEventButton *a_ev, seqevent &ths)

  Implements the on-button-release callback.

- bool on_motion_notify_event (GdkEventMotion *a_ev, seqevent &ths)

  Implements the on-motion-notify event.

6.23.1 Member Function Documentation

6.23.1.1 void Seq24SeqEventInput::set_adding (bool *a_adding,* seqevent & *seqev*)

Modifies m_adding as well.

6.23.1.2 bool Seq24SeqEventInput::on_button_press_event (GdkEventButton * *a_ev,* seqevent & *ths*)

Todo Needs update.

6.24 Seq24SeqRollInput Struct Reference

Implements the Seq24 mouse interaction paradigm for the seqroll.

Public Member Functions

- Seq24SeqRollInput ()

  Default constructor.
- void set_adding (bool a_adding, seqroll &ths)

  Changes the mouse cursor pixmap according to whether a note is being added or not.
- bool on_button_press_event (GdkEventButton *a_ev, seqroll &ths)

  Implements the on-button-press event handling for the Seq24 style of mouse interaction.
- bool on_button_release_event (GdkEventButton *a_ev, seqroll &ths)

  Implements the on-button-release event handling for the Seq24 style of mouse interaction.
- bool on_motion_notify_event (GdkEventMotion *a_ev, seqroll &ths)

  Implements the on-motion-notify event handling for the Seq24 style of mouse interaction.

6.24.1 Member Function Documentation

6.24.1.1 void Seq24SeqRollInput::set_adding (bool *a_adding,* seqroll & *sroll*)

(Which?) popup menu calls this. It is actually a right click, I think.

6.25 seqdata Class Reference

This class supports drawing piano-roll eventis on a window.

Inherits DrawingArea.

Public Member Functions

- seqdata (sequence *a_seq, int a_zoom, Gtk::Adjustment *a_hadjust)

  Principal constructor.
- void reset ()

  This function calls update_size().
- void redraw ()

  Updates the pixmap and queues up a redraw operation.
- void set_zoom (int a_zoom)

  Sets the zoom to the given value and resets the view via the reset function.
- void set_data_type (unsigned char a_status, unsigned char a_control)

  Sets the status to the given value, and the control to the optional given value, which defaults to 0, then calls redraw().
- int idle_redraw ()

  Draws events on this object's built-in window and pixmap.

Friends

- class **seqroll**
- class **seqevent**

6.25.1 Constructor & Destructor Documentation

6.25.1.1 seqdata::seqdata (sequence * *a_seq,* int *a_zoom,* Gtk::Adjustment * *a_hadjust*)

In the constructor you can only allocate colors, get_window() returns 0 because we have not been realized.

6.25.2 Member Function Documentation

6.25.2.1 void seqdata::reset ( )

Then, regardless of whether the view is realized, updates the pixmap and queues up a draw operation.

Note

If it weren't for the is_realized() condition, we could just call update_sizes(), which does all this anyway.

6.25.2.2 void seqdata::redraw ( ) [inline]

We need to make this an inline function and use it as common code.

6.25.2.3 void seqdata::set_zoom (int *a_zoom*)

This begs the question, do we have GUI access to the zoom setting?

6.25.2.4 int seqdata::idle_redraw ( )

This drawing is done only if there is no dragging in progress, to guarantee no flicker.

6.26 seqedit Class Reference

Implements the Pattern Editor, which has references to:

Inherits Window.

Public Member Functions

- seqedit (sequence *a_seq, perform *a_perf, int a_pos)

  Connects to a menu item, tells the performance to launch the timer thread.
- ~seqedit ()

  A rote destructor.

6.26.1 Detailed Description

- perform
- seqroll
- seqkeys
- seqdata
- seqtime
- seqevent
- sequence

This class has a metric ton of user-interface objects and other members.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 seqedit::seqedit (sequence * *a_seq*, perform * *a_perf*, int *a_pos*)

But this is an unused, empty function.

void seqedit::menu_action_quantise () { } Principal constructor.

Todo Offload most of the work into an initialization function like options does; make the sequence and perform parameters references.

6.27 seqevent Class Reference

Implements the piano event drawing area.

Inherits DrawingArea.

Public Member Functions

- seqevent (sequence *a_seq, int a_zoom, int a_snap, seqdata *a_seqdata_wid, Gtk::Adjustment *a_hadjust)

  Principal constructor.
- void reset ()

  This function basically resets the whole widget as if it was realized again.
- void redraw ()

  Adjusts the scrolling offset for ticks, updates the pixmap, and draws it on the window.
- void set_zoom (int a_zoom)

  Sets zoom to the given value, and resets if the value ended up being changed.
- void set_snap (int a_snap)

  'Setter' function for member m_snap
- void set_data_type (unsigned char a_status, unsigned char a_control)

  Sets the status to the given parameter, and the CC value to the given optional control parameter, which defaults to 0.
- void update_sizes ()

  If the window is realized, this function creates a pixmap with window dimensions, the updates the pixmap, and queues up a redraw.
- void draw_background ()

  This function updates the background.
- void draw_events_on_pixmap ()

  This function fills the main pixmap with events.
- void draw_pixmap_on_window ()

  This function currently just queues up a draw operation for the pixmap.
- void draw_selection_on_window ()

  Draw the selected events on the window.
- void update_pixmap ()

  Redraws the background pixmap on the main pixmap, then puts the events on.
- int idle_redraw ()

  Implements redraw while idling.

Friends

- struct **Seq24SeqEventInput**

6.27.1 Member Function Documentation

6.27.1.1 void seqevent::set_snap (int *a_snap*) [inline]

Simply sets the snap member.

6.27.1.2 void seqevent::set_data_type (unsigned char *a_status,* unsigned char *a_control* = 0)

Then redraws.

6.27.1.3 void seqevent::update_sizes ( )

This ends up filling the background with dotted lines, etc.

6.27.1.4 void seqevent::draw_background ( )

It sets the foreground to white, draws the rectangle.

6.27.1.5 void seqevent::draw_pixmap_on_window ( )

Old comments:

```
It then tells event to do the same.

We changed something on this window, and chances are we need to
update the event widget as well and update our velocity window.

    m_seqdata_wid->update_pixmap();
    m_seqdata_wid->draw_pixmap_on_window();
    RCB ??
```

6.27.1.6 int seqevent::idle_redraw ( )

Who calls this routine?

6.28 seqkeys Class Reference

This class implements the left side piano of the pattern/sequence editor.

Inherits DrawingArea.

Public Member Functions

- seqkeys (sequence *a_seq, Gtk::Adjustment *a_vadjust)
  Principal constructor.
- void set_scale (int a_scale)
  Sets the musical scale, then resets.
- void set_key (int a_key)
  Sets the musical key, then resets.
- void set_hint_key (int a_key)
  Sets a key to grey so that it can serve as a scale hint.
- void set_hint_state (bool a_state)
  Sets the hint state to the given value.

6.28.1 Member Function Documentation

6.28.1.1 void seqkeys::set_hint_state (bool *a_state*)

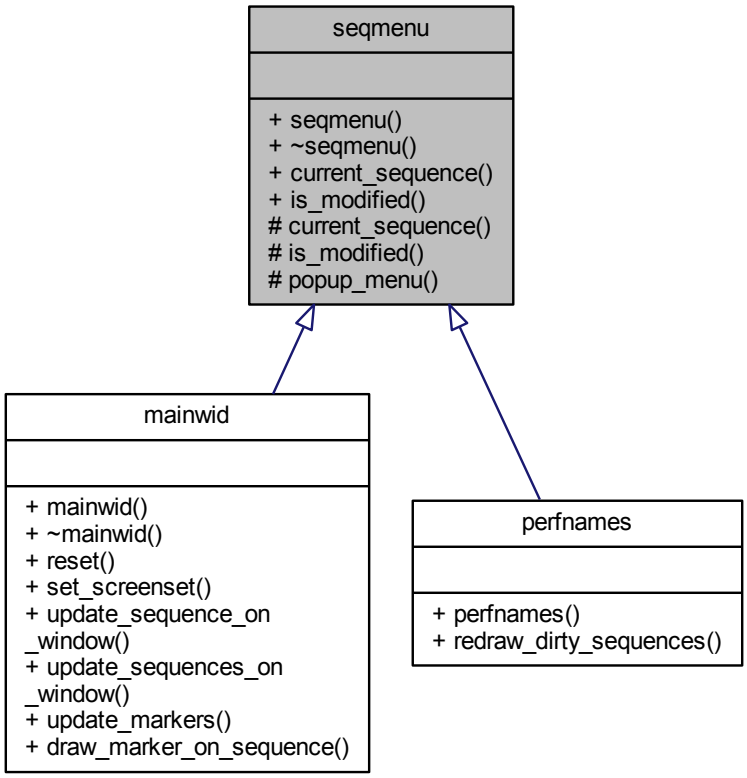
Parameters

a_state	Provides the value for hinting, where true == on, false == off.

6.29 seqmenu Class Reference

This class handles the right-click menu of the sequence slots in the pattern window.

Inheritance diagram for seqmenu:



Public Member Functions

- seqmenu (perform *a_p)

  Principal constructor.
- virtual ~seqmenu ()

  Provides a rote base-class destructor.
- int current_sequence () const

  'Getter' function for member m_current_seq
- bool is_modified () const

  'Getter' function for member m_modified

Protected Member Functions

- void current_sequence (int seq)

'Setter' function for member m_current_seq

- void is_modified (bool flag)

  'Setter' function for member m_modified

- void popup_menu ()

  This function sets up the File menu entries.

6.29.1 Detailed Description

It is an abstract base class.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 seqmenu::seqmenu (perform * a_p)

Apart from filling in some fo the members, this function initializes the clipboard, so that we don't get a crash on a paste with no previous copy.

6.29.2.2 seqmenu::~seqmenu ( ) [virtual]

A rote destructor.

This is necessary in an abstraction base class.

If we determine that we need to delete the m_seqedit pointer, we can do it here. But that is not likely, because we can have many new seqedit objects in play, because we can edit many at once.

6.30 seqroll Class Reference

Implements the piano roll section of the pattern editor.

Inherits DrawingArea.

Public Member Functions

- seqroll (perform *a_perf, sequence *a_seq, int a_zoom, int a_snap, seqdata *a_seqdata_wid, seqevent *a_seqevent_wid, seqkeys *a_seqkeys_wid, int a_pos, Gtk::Adjustment *a_hadjust, Gtk::Adjustment *a_↩ vadjust)

  Principal constructor.

- ~seqroll ()

  Provides a destructor to delete allocated objects.

- void reset ()

  This function basically resets the whole widget as if it was realized again.

- void redraw ()

  Redraws unless m_ignore_redraw is true.

- void redraw_events ()

  Redraws events unless m_ignore_redraw is true.

- void set_key (int a_key)

  Sets the music key to the given value, and then resets the view.

- void set_scale (int a_scale)

  Sets the music scale to the given value, and then resets the view.

- void set_snap (int a_snap)

  Sets the snap to the given value, and then resets the view.

- void set_zoom (int a_zoom)

  Sets the zoom to the given value, and then resets the view.

- void set_note_length (int a_note_length)

  'Setter' function for member m_note_length
- void set_ignore_redraw (bool a_ignore)

  'Setter' function for member m_ignore_redraw
- void set_data_type (unsigned char a_status, unsigned char a_control)

  Sets the status to the given parameter, and the CC value to the given optional control parameter, which defaults to 0.
- void set_background_sequence (bool a_state, int a_seq)

  This function sets the given sequence onto the piano roll of the pattern editor, so that the musician can have another pattern to play against.
- void update_pixmap ()

  This function draws the background pixmap on the main pixmap, and then draws the events on it.
- void update_sizes ()

  Update the sizes of items based on zoom, PPQN, BPM, BW (beat width) and more.
- void update_background ()

  Updates the background of this window.
- void draw_background_on_pixmap ()

  Draws the main pixmap.
- void draw_events_on_pixmap ()

  Fills the main pixmap with events.
- void draw_selection_on_window ()

  Draws the current selecton on the main window.
- void draw_progress_on_window ()

  Draw a progress line on the window.
- int idle_redraw ()

  Draw the events on the main window and on the pixmap.
- void start_paste ()

  Starts a paste operation.

Friends

- struct **Seq24SeqRollInput**

6.30.1 Member Function Documentation

6.30.1.1 void seqroll::reset ( )

It's almost identical to the change_horz() function!

6.30.1.2 void seqroll::set_data_type (unsigned char *a_status*, unsigned char *a_control* = `0`)

Unlike the same function in seqevent, this version does not redraw.

6.30.1.3 void seqroll::set_background_sequence (bool *a_state*, int *a_seq*)

The a_state parameter sets the boolean m_drawing_background_seq.

6.30.1.4 void seqroll::draw_events_on_pixmap ( )

Just calls draw_events_on().

6.31 seqtime Class Reference

This class implements the piano time, whatever that is.

Inherits DrawingArea.

Public Member Functions

- seqtime (sequence *a_seq, int a_zoom, Gtk::Adjustment *a_hadjust)

  Principal constructor.
- void reset ()

  Sets the scroll offset tick and x values, updates the sizes and the pixmap, and resets the window.
- void redraw ()

  Very similar to the reset() function, except it doesn't update the sizes.
- void set_zoom (int a_zoom)

  Sets the zoom to the given value and resets the window.

6.31.1 Constructor & Destructor Documentation

6.31.1.1 seqtime::seqtime (sequence * *a_seq,* int *a_zoom,* Gtk::Adjustment * *a_hadjust*)

In the constructor you can only allocate colors; get_window() returns 0 because the window is not yet realized>

6.32 sequence Class Reference

The sequence class is firstly a receptable for a single track of MIDI data read from a MIDI file or edited into a pattern.

Public Types

- enum select_action_e {
  e_select ,
  e_deselect,
  e_toggle_selection,
  e_remove_one }

Public Member Functions

- sequence ()

  Principal constructor.
- ~sequence ()

  A rote destructor.
- sequence & operator= (const sequence &a_rhs)

  Principal assignment operator.
- int event_count () const

  Returns the number of events stored in m_eventss.
- void push_undo ()

  Pushes the list-event into the undo-list.
- void pop_undo ()

  If there are items on the undo list, this function pushes the list-event into the redo-list, puts the top of the undo-list into the list-event, pops from the undo-list, calls verify_and_link(), and then calls unselect.
- void pop_redo ()

  If there are items on the redo list, this function pushes the list-event into the undo-list, puts the top of the redo-list into the list-event, pops from the redo-list, calls verify_and_link(), and then calls unselect.
- void push_trigger_undo ()

  Pushes the list-trigger into the trigger undo-list, then flags each item in the undo-list as unselected.
- void pop_trigger_undo ()

  If the trigger undo-list has any items, the list-trigger is pushed 9nto the redo list, the top of the undo-list is coped into the list-trigger, and then pops from the undo-list.

- void set_name (const std::string &a_name)

  Sets the sequence name member, m_name.
- void set_name (char *a_name)

  Sets the sequence name member, m_name.
- void set_bpm (long a_beats_per_measure)

  'Setter' function for member m_time_beats_per_measure
- long get_bpm () const

  'Getter' function for member m_time_beats_per_measure
- void set_bw (long a_beat_width)

  'Setter' function for member m_time_beat_width
- long get_bw () const

  'Getter' function for member m_time_beat_width
- void set_rec_vol (long a_rec_vol)

  'Setter' function for member m_rec_vol
- void set_song_mute (bool a_mute)

  'Setter' function for member m_song_mute
- bool get_song_mute () const

  'Getter' function for member m_song_mute
- const char * get_name () const

  'Getter' function for member m_name
- void set_editing (bool a_edit)

  'Setter' function for member m_editing
- bool get_editing () const

  'Getter' function for member m_editing
- void set_raise (bool a_edit)

  'Setter' function for member m_raise
- bool get_raise (void) const

  'Getter' function for member m_raise
- void set_length (long a_len, bool a_adjust_triggers=true)

  Sets the length (m_length) and adjusts triggers for it if desired.
- long get_length () const

  'Getter' function for member m_length
- long get_last_tick ()

  Returns the last tick played, and is used by the editor's idle function.
- void set_playing (bool)

  Sets the playing state of this sequence.
- bool get_playing () const

  'Getter' function for member m_playing
- void toggle_playing ()

  Toggles the playing status of this sequence.
- void toggle_queued ()

  'Setter' function for member m_queued and m_queued_tick
- void off_queued ()

  'Setter' function for member m_queued
- bool get_queued () const

  'Getter' function for member m_queued
- long get_queued_tick () const

  'Getter' function for member m_queued_tick
- void set_recording (bool)

  'Setter' function for member m_recording and m_notes_on
- bool get_recording () const

'Getter' function for member m_recording

- void set_snap_tick (int a_st)

  'Setter' function for member m_snap_tick
- void set_quantized_rec (bool a_qr)

  'Setter' function for member m_quantized_rec
- bool get_quantized_rec () const

  'Getter' function for member m_quantized_rec
- void set_thru (bool)

  'Setter' function for member m_thru
- bool get_thru () const

  'Getter' function for member m_thru
- bool is_dirty_main ()

  Returns the value of the dirty main flag, and sets that flag to false (i.e.
- bool is_dirty_edit ()

  Returns the value of the dirty edit flag, and sets that flag to false.
- bool is_dirty_perf ()

  Returns the value of the dirty performance flag, and sets that flag to false.
- bool is_dirty_names ()

  Returns the value of the dirty names (heh heh) flag, and sets that flag to false.
- void set_dirty_mp ()

  Sets the dirty flags for names, main, and performance.
- void set_dirty ()

  Call set_dirty_mp() and then sets the dirty flag for editing.
- unsigned char get_midi_channel () const

  'Getter' function for member m_midi_channel
- void set_midi_channel (unsigned char a_ch)

  Sets the m_midi_channel number.
- void print ()

  Prints a list of the currently-held events.
- void print_triggers ()

  Prints a list of the currently-held triggers.
- void play (long a_tick, bool a_playback_mode)

  The play() function dumps notes starting from the given tick, and it pre-buffers ahead.
- void set_orig_tick (long a_tick)

  'Setter' function for member m_last_tick
- void add_event (const event *a_e)

  Adds an event to the internal event list in a sorted manner.
- void add_trigger (long a_tick, long a_length, long a_offset=0, bool a_adjust_offset=true)

  Adds a trigger.
- void split_trigger (long a_tick)

  Splits a trigger.
- void grow_trigger (long a_tick_from, long a_tick_to, long a_length)

  Grows a trigger.
- void del_trigger (long a_tick)

  Deletes a trigger, that brackets the given tick, from the trigger-list.
- bool unselect_triggers ()

  Always returns false!
- bool intersectTriggers (long position, long &start, long &end)

  This function examines each trigger in the trigger list.
- bool intersectNotes (long position, long position_note, long &start, long &end, long &note)

  This function examines each note in the event list.

Adds a event of a given status value and data values, at a given tick location.

- void stream_event (event *a_ev)

  Streams the given event.
- void change_event_data_range (long a_tick_s, long a_tick_f, unsigned char a_status, unsigned char a_cc, int a_d_s, int a_d_f)

  Changes the event data range.
- void increment_selected (unsigned char a_status, unsigned char a_control)

  Increments events the match the given status and control values.
- void decrement_selected (unsigned char a_status, unsigned char a_control)

  Decrements events the match the given status and control values.
- void grow_selected (long a_delta_tick)

  Moves note off event.
- void stretch_selected (long a_delta_tick)

  Performs a stretch operation on the selected events.
- void remove_marked ()

  Removes marked events.
- void mark_selected ()

  Marks the selected events.
- void unpaint_all ()

  Unpaints all list-events.
- void unselect ()

  Deselects all events, unconditionally.
- void verify_and_link ()

  This function verifies state: all note-ons have an off, and it links note-offs with their note-ons.
- void link_new ()

  Links a new event.
- void zero_markers ()

  Resets everything to zero.
- void play_note_on (int a_note)

  Plays a note from the piano roll on the main bus on the master MIDI buss.
- void play_note_off (int a_note)

  Turns off a note from the piano roll on the main bus on the master MIDI buss.
- void off_playing_notes ()

  Sends a note-off event for all active notes.
- void reset_draw_marker ()

  This refreshes the play marker to the last tick.
- void reset_draw_trigger_marker ()

  Threadsafe
- draw_type get_next_note_event (long *a_tick_s, long *a_tick_f, int *a_note, bool *a_selected, int *a_velocity)

  Each call to seqdata() fills the passed references with a events elements, and returns true.
- int get_lowest_note_event ()

  Threadsafe
- int get_highest_note_event ()

  Threadsafe
- bool get_next_event (unsigned char a_status, unsigned char a_cc, long *a_tick, unsigned char *a_D0, unsigned char *a_D1, bool *a_selected)

  Get the next event in the event list that matches the given status and control character.
- bool get_next_event (unsigned char *a_status, unsigned char *a_cc)

  Get the next event in the event list.
- bool get_next_trigger (long *a_tick_on, long *a_tick_off, bool *a_selected, long *a_tick_offset)

Get the next trigger in the trigger list, and set the parameters based on that trigger.

- void fill_list (CharList *a_list, int a_pos)

  This function fills the given character list with MIDI data from the current sequence, preparatory to writing it to a file.

- void transpose_notes (int a_steps, int a_scale)

  Transposes notes by the given steps, in accordance with the given scale.

6.32.1 Detailed Description

More members than you can shake a stick at.

6.32.2 Member Enumeration Documentation

6.32.2.1 enum sequence::select_action_e

Enumerator

e_select This enumeration is used in selecting events and note. Se the select_note_events() and select_events() functions.

e_deselect To deselect the event under the cursor.

e_toggle_selection To toggle the selection of the event under the cursor.

e_remove_one To remove one note under the cursor.

6.32.3 Member Function Documentation

6.32.3.1 sequence & sequence::operator= (const sequence & *a_rhs*)

Follows the stock rules for such an operator, but does a little more then just assign member values.

Threadsafe

6.32.3.2 int sequence::event_count ( ) const

Threadsafe

6.32.3.3 void sequence::push_undo ( )

Threadsafe

6.32.3.4 void sequence::pop_undo ( )

Threadsafe

6.32.3.5 void sequence::pop_redo ( )

Threadsafe

6.32.3.6 void sequence::push_trigger_undo ( )

Threadsafe

6.32.3.7 void sequence::set_bpm (long *a_beats_per_measure*)

Threadsafe

6.32.3.8 void sequence::set_bw (long *a_beat_width*)

Threadsafe

6.32.3.9 long sequence::get_bw ( ) const [inline]

Threadsafe

6.32.3.10 void sequence::set_rec_vol (long *a_rec_vol*)

Threadsafe

6.32.3.11 void sequence::set_length (long *a_len,* bool *a_adjust_triggers* = true)

Threadsafe

6.32.3.12 void sequence::set_playing (bool *a_p*)

When playing, and the sequencer is running, notes get dumped to the ALSA buffers.

Parameters

a_p	Provides the playing status to set. True means to turn on the playing, false means to turn it off, and turn off any notes still playing.

6.32.3.13 void sequence::toggle_queued ( )

Toggles the queued flag and sets the dirty-mp flag. Also calculates the queued tick based on m_last_tick.

Threadsafe

6.32.3.14 void sequence::off_queued ( )

Toggles the queued flag and sets the dirty-mp flag.

Threadsafe

6.32.3.15 void sequence::set_recording (bool *a_r*)

Threadsafe

6.32.3.16 void sequence::set_snap_tick (int *a_st*)

Threadsafe

6.32.3.17 void sequence::set_quantized_rec (bool *a_qr*)

Threadsafe

6.32.3.18 void sequence::set_thru (bool *a_r*)

Threadsafe

6.32.3.19 bool sequence::is_dirty_main ( )

resets it). This flag signals that a redraw is needed from recording.

Threadsafe

6.32.3.20 bool sequence::is_dirty_edit ( )

Threadsafe

6.32.3.21 bool sequence::is_dirty_perf ( )

Threadsafe

6.32.3.22 bool sequence::is_dirty_names ( )

Threadsafe

6.32.3.23 void sequence::set_dirty_mp ( )

Not threadsafe

6.32.3.24 void sequence::set_dirty ( )

Threadsafe

6.32.3.25 void sequence::set_midi_channel (unsigned char *a_ch*)

Threadsafe

6.32.3.26 void sequence::print ( )

Not threadsafe

6.32.3.27 void sequence::print_triggers ( )

Not threadsafe

6.32.3.28 void sequence::play (long *a_tick,* bool *a_playback_mode*)

This function is called by the sequencer thread, performance. The tick comes in as global tick.

It turns the sequence off after we play in this frame.

Threadsafe

6.32.3.29 void sequence::set_orig_tick (long *a_tick*)

Threadsafe

6.32.3.30 void sequence::add_event (const **event** * *a_e*)

Then it reset the draw-marker and sets the dirty flag.

Currently, when reading a MIDI file (see the midifile module's parse function), only the main events (notes, aftertouch, pitch, program changes, etc.) are added with this function. So, we can rely on reading only playable events into a sequence.

This module (sequencer) adds all of those events as well, but it can surely add other events. We should assume that any events added by sequencer are playable.

Threadsafe

Warning

> This pushing (and, in writing the MIDI file, the popping), causes events with identical timestamps to be written in reverse order. Doesn't affect functionality, but it's puzzling until one understands what is happening.

6.32.3.31 void sequence::add_trigger (long *a_tick,* long *a_length,* long *a_offset* = `0`, bool *a_adjust_offset* = `true`)

If a_state = true, the range is on. If a_state = false, the range is off.

What is this?

```
is      ie
<      ><        ><        >
es             ee
<               >
XX
```

```
es ee
<   >
<>

es    ee
<      >
<    >

es     ee
<       >
<    >
```

6.32.3.32 void sequence::split_trigger (long *a_tick*)

This is the public overload of split_trigger.

Threadsafe

6.32.3.33 void sequence::grow_trigger (long *a_tick_from,* long *a_tick_to,* long *a_length*)

Threadsafe

6.32.3.34 void sequence::del_trigger (long *a_tick*)

Threadsafe

6.32.3.35 bool sequence::intersectTriggers (long *position,* long & *start,* long & *end*)

If the given position is between the current trigger's tick-start and tick-end values, the these values are copied to the start and end parameters, respectively, and then we exit.

Threadsafe

Parameters

position	The position to examine.
start	The destination for the starting tick (m_tick_start) of the matching trigger.
end	The destination for the ending tick (m_tick_end) of the matching trigger.

Returns

Returns true if a trigger was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

6.32.3.36 bool sequence::intersectNotes (long *position,* long *position_note,* long & *start,* long & *end,* long & *note*)

If the given position is between the current notes on and off time values, values, the these values are copied to the start and end parameters, respectively, the note value is copied to the note parameter, and then we exit.

Threadsafe

Parameters

position	The position to examine.
position_note	I think this is the note value we might be looking for ???
start	The destination for the starting tick (m_tick_start) of the matching trigger.
end	The destination for the ending tick (m_tick_end) of the matching trigger.
note	The destination for the note of the matching event.

Returns

Returns true if a event was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

6.32.3.37 bool sequence::intersectEvents (long *posstart,* long *posend,* long *status,* long & *start*)

If the given position is between the current trigger's tick-start and tick-end values, the these values are copied to the start and end parameters, respectively, and then we exit.

Threadsafe

Parameters

posstart	The starting position to examine.
posend	The ending position to examine.
status	The desired status value.
start	The destination for the starting tick (m_tick_start) of the matching trigger.

Returns

Returns true if a event was found whose start/end ticks contained the position. Otherwise, false is returned, and the start and end return parameters should not be used.

6.32.3.38 void sequence::move_selected_triggers_to (long *a_tick,* bool *a_adjust_offset,* int *a_which* = 2)

```
          min_tick][0                1][max_tick
                           2


-   If we are moving the 0, use first as offset.
-   If we are moving the 1, use the last as the offset.
-   If we are moving both (2), use first as offset.
```

Threadsafe

6.32.3.39 long sequence::get_selected_trigger_start_tick ( )

Threadsafe

6.32.3.40 long sequence::get_selected_trigger_end_tick ( )

Threadsafe

6.32.3.41 long sequence::get_max_trigger ( )

Threadsafe

6.32.3.42 void sequence::move_triggers (long *a_start_tick,* long *a_distance,* bool *a_direction*)

Threadsafe

6.32.3.43 void sequence::copy_triggers (long *a_start_tick,* long *a_distance*)

```
... a
[      ][      ]
...
... a
...

5   7    play
3        offset
8   10   play

X...X...X...X...X...X...X...X...X...X...
L       R
[        ] [     ]  []  orig
[                   ]

        <<
        [     ]    [  ][ ]  [] split on the R marker, shift first
```

```
[     ]         [     ]
delete middle
[     ][ ]  []          move ticks
[     ][     ]

L        R
[     ][ ] [     ]  [] split on L
[     ][             ]

[     ]         [ ] [     ]  [] increase all after L
[     ]         [             ]
```

Copies triggers to...

Threadsafe

6.32.3.44 void sequence::clear_triggers ( )

Threadsafe

6.32.3.45 void sequence::set_midi_bus (char *a_mb*)

Threadsafe

6.32.3.46 void sequence::set_master_midi_bus (mastermidibus * *a_mmb*)

Threadsafe

6.32.3.47 int sequence::select_note_events (long *a_tick_s,* int *a_note_h,* long *a_tick_f,* int *a_note_l,* select_action_e *a_action*)

Returns the number selected.

Threadsafe

6.32.3.48 int sequence::select_events (long *a_tick_s,* long *a_tick_f,* unsigned char *a_status,* unsigned char *a_cc,* select_action_e *a_action*)

Note that there is also an overloaded version of this function.

Threadsafe

6.32.3.49 int sequence::select_events (unsigned char *a_status,* unsigned char *a_cc,* bool *a_inverse* = `false`)

Note that there is also an overloaded version of this function.

Threadsafe

Warning

This used to be a void function, so it just returns 0 for now.

6.32.3.50 int sequence::get_num_selected_notes ( )

Threadsafe

6.32.3.51 int sequence::get_num_selected_events (unsigned char *a_status,* unsigned char *a_cc*)

If the event is a control change (CC), then it must also match the given CC value.

Threadsafe

6.32.3.52 void sequence::select_all ( )

Threadsafe

6.32.3.53 void sequence::copy_selected ( )

Threadsafe

6.32.3.54 void sequence::paste_selected (long *a_tick,* int *a_note*)

I wonder if we can get away with just getting a reference to m_events_clipboard, rather than copying the whole thing, for speed.

Threadsafe

6.32.3.55 void sequence::add_note (long *a_tick,* long *a_length,* int *a_note,* bool *a_paint* = `false`)

It adds a single note-on / note-off pair.

The a_paint parameter indicates if we care about the painted event, so then the function runs though the events and deletes the painted ones that overlap the ones we want to add.

Threadsafe

6.32.3.56 void sequence::add_event (long *a_tick,* unsigned char *a_status,* unsigned char *a_d0,* unsigned char *a_d1,* bool *a_paint* = `false`)

The a_paint parameter indicates if we care about the painted event, so then the function runs though the events and deletes the painted ones that overlap the ones we want to add.

Threadsafe

6.32.3.57 void sequence::stream_event (event * *a_ev*)

Threadsafe

6.32.3.58 void sequence::change_event_data_range (long *a_tick_s,* long *a_tick_f,* unsigned char *a_status,* unsigned char *a_cc,* int *a_data_s,* int *a_data_f*)

Changes only selected events, if any.

Threadsafe

Let t == the current tick value; ts == tick start value; tf == tick finish value; ds = data start value; df == data finish value; d = the new data value.

Then

```
             df (t - ts) + ds (tf - t)
        d = --------------------------
                    tf  -   ts

If this were an interpolation formula it would be:

                              t -  ts
        d = ds + (df - ds) ---------
                              tf - ts

Something is not quite right; to be investigated.

\param a_tick_s
    Provides the starting tick value.

\param a_tick_f
    Provides the ending tick value.

\param a_status
    Provides the event status that is to be changed.

\param a_cc
    Provides the event control value.
```

```
\param a_data_s
    Provides the starting data value.

\param a_data_f
    Provides the finishing data value.
```

6.32.3.59 void sequence::increment_selected (unsigned char *a_stat,* unsigned char *a_control*)

The supported statuses are:

- EVENT_NOTE_ON
- EVENT_NOTE_OFF
- EVENT_AFTERTOUCH
- EVENT_CONTROL_CHANGE
- EVENT_PITCH_WHEEL
- EVENT_PROGRAM_CHANGE
- EVENT_CHANNEL_PRESSURE

Threadsafe

6.32.3.60 void sequence::decrement_selected (unsigned char *a_stat,* unsigned char *a_control*)

The supported statuses are:

- EVENT_NOTE_ON
- EVENT_NOTE_OFF
- EVENT_AFTERTOUCH
- EVENT_CONTROL_CHANGE
- EVENT_PITCH_WHEEL
- EVENT_PROGRAM_CHANGE
- EVENT_CHANNEL_PRESSURE

Threadsafe

6.32.3.61 void sequence::grow_selected (long *a_delta_tick*)

Threadsafe

6.32.3.62 void sequence::stretch_selected (long *a_delta_tick*)

This should move a note off event, according to old comments, but it doesn't seem to do that. See the grow_selected() function.

Threadsafe

6.32.3.63 void sequence::remove_marked ( )

Threadsafe

Todo Verify that this is the correct way to handle changing iterators.

6.32.3.64 void sequence::mark_selected ( )

Threadsafe

6.32.3.65 void sequence::unpaint_all ( )

Threadsafe

6.32.3.66 void sequence::unselect ( )

Threadsafe

6.32.3.67 void sequence::verify_and_link ( )

Threadsafe

6.32.3.68 void sequence::link_new ( )

Threadsafe

6.32.3.69 void sequence::zero_markers ( )

This function is used when the sequencer stops.

Threadsafe

6.32.3.70 void sequence::play_note_on (int *a_note*)

It flushes a note to the midibus to preview its sound, used by the virtual piano.

Threadsafe

6.32.3.71 void sequence::play_note_off (int *a_note*)

Threadsafe

6.32.3.72 void sequence::off_playing_notes ( )

Threadsafe

6.32.3.73 void sequence::reset_draw_marker ( )

It resets the draw marker so that calls to get_next_note_event() will start from the first event.

Threadsafe

6.32.3.74 draw_type sequence::get_next_note_event (long * *a_tick_s,* long * *a_tick_f,* int * *a_note,* bool * *a_selected,* int * *a_velocity*)

When it has no more events, returns a false.

6.32.3.75 bool sequence::get_next_event (unsigned char *a_status,* unsigned char *a_cc,* long * *a_tick,* unsigned char * *a_D0,* unsigned char * *a_D1,* bool * *a_selected*)

Then set the rest of the parameters parameters using that event.

6.32.3.76 bool sequence::get_next_event (unsigned char * *a_status,* unsigned char * *a_cc*)

Then set the status and control character parameters using that event.

6.32.3.77 void sequence::fill_list (CharList * *a_list,* int *a_pos*)

Note that some of the events might not come out in the same order they were stored in (we see that with program-change events.

6.33 trigger Class Reference

This class is used in playback.

Public Member Functions

- trigger ()

  Initializes the trigger structure.

- bool operator< (const trigger &rhs)

  This operator compares only the m_tick_start members.

6.33.1 Detailed Description

Making its members public makes it really "just" a structure.

6.34 user_instrument_definition Struct Reference

This structure corresponds to `[user-instrument-0]` definitions in the `~/.seq24usr` file.

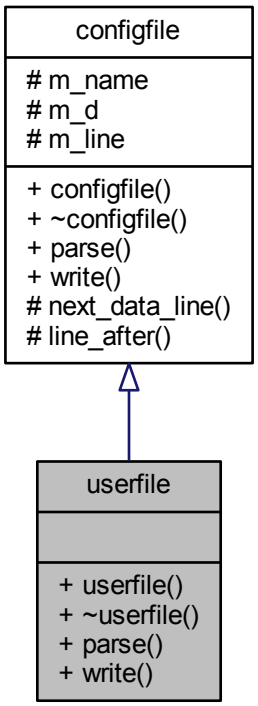
6.35 user_midi_bus_definition Struct Reference

This structure corresponds to `[user-midi-bus-0]` definitions in the `~/.seq24usr` file.

6.36 userfile Class Reference

Supports the user's ~/.seq24usr configuration file.

Inheritance diagram for userfile:



Public Member Functions

- userfile (const std::string &a_name)

Principal constructor.

- ~userfile ()

  A rote destructor needed for a derived class.

- bool parse (perform *a_perf)

  Parses a "usr" file, filling in the given perform object.

- bool write (perform *a_perf)

  This function just returns false, as there is no "perfrom" information in the user-file.

Additional Inherited Members

6.36.1 Member Function Documentation

6.36.1.1 bool userfile::parse (perform * *a_perf*) [virtual]

This function opens the file as a text file (line-oriented).

Implements configfile.

Index