QuteCsound, a Csound Frontend

Andrés CABRERA
Sonic Arts Research Centre
Queen's University Belfast
UK
acabrera01@qub.ac.uk

Abstract

QuteCsound is a front-end application for Csound written using the Qt toolkit. It has been developed since 2008, and is now part of the Csound distribution for Windows and OS X. It is a code editor for Csound, and provides many features for real-time control of the Csound engine, through graphical control interfaces and live score processing.

Keywords

Csound, Front-end, Qt, Widgets, Interface builder

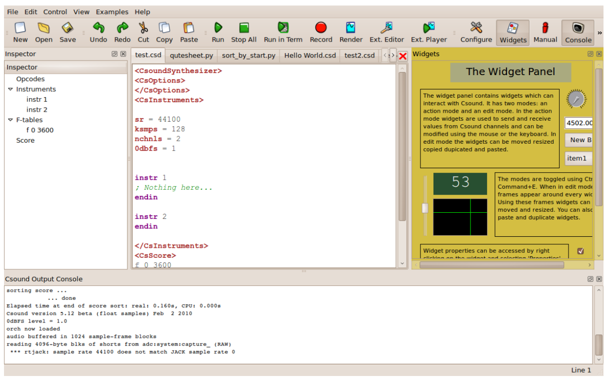
1 Introduction

QuteCsound was born out of the desire to have a cross-platform front-end for Csound [Boulanger, 2000] like MacCsound [Ingalls, 2005] which only runs on OS X. It is based on the idea of having real-time control of Csound through graphical widgets, and making Csound accessible for novice users. It is however designed to be also a powerful editor for advanced users and also offline (non-realtime) work. The most recent version is 0.5.0. It has been tested on Linux, Mac OS X, Windows and Solaris. QuteCsound has been translated to Spanish, French, Italian, German and Portuguese.

One of the main goals was also to make a new interface for Csound which would be comfortable for a musician whose background is not in programming. Existing cross-platform interfaces for Csound were either too basic or somewhat impractical. The Csound Manual [Cabrera, 2010] is very comprehensive, but few front-ends take full advantage of its resources like opcode listing in XML format.

QuteCsound uses the Csound API [ffitch, 2005], which enables tight integration and control of Csound.

It is licensed under the GPLv3 and LGPLv2 for compatibility with the Csound licence to ease distribution alongside Csound. More information on QuteCsound can be found in the QuteCsound Front page[1] and the QuteCsound sources and binaries can be found on its Sourceforge page[2].


File Edit Control View Examples Help
New Open Save Undo Redo Cut Copy Paste Run Stop All Run in Term Record Render Ext. Editor Ext. Player Configure Widgets Manual Console
Inspector
Opcodes
Instruments
instr 1
instr 2
F-tables
f 0 3600
Score
test.csd qutesheet.py sort_by_start.py Hello World.csd test2.csd
Widgets
The Widget Panel
4502.00
New B
item1
53
Csound Output Console
Line 1

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 128
nchnls = 2
0dbfs = 1


instr 1
; Nothing here...
endin

instr 2
endin

</CsInstruments>
<CsScore>
f 0 3600
```

The widget panel contains widgets which can interact with Csound. It has two modes: an action mode and an edit mode. In the action mode widgets are used to send and receive values from Csound channels and can be modified using the mouse or the keyboard. In edit mode the widgets can be moved resized copied dupicated and pasted.

The modes are toggled using Ctr Command+E. When in edit mode frames appear around every wic Using these frames widgets can moved and resized. You can alsc paste and duplicate widgets.

Widget properties can be accessed by right

```
sorting score ...
        ... done
Elapsed time at end of score sort: real: 0.160s, CPU: 0.000s
Csound version 5.12 beta (float samples) Feb  2 2010
0dBFS level = 1.0
orch now loaded
audio buffered in 1024 sample-frame blocks
reading 4096-byte blks of shorts from adc:system:capture_ (RAW)
 *** rtjack: sample rate 44100 does not match JACK sample rate 0
```

Figure 1: The main QuteCsound window

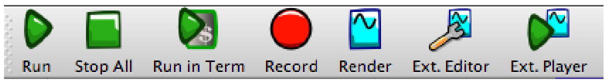


Figure 2: The transport bar

2 GUI

The main application window is shown in figure 1. The main component is a text editor with syntax highlighting. Documents can be opened as separate tabs in this area. There is a large icon bar with the main actions, which contains the usual open/save and cut/copy/paste action icons, a section with transport controls (see figure 2), and a section for handling visibility of the rest of the application windows and panels.

Around the editor many dockable widgets can be positioned freely. These dockable widgets are:

Widget Panel In the widget panel, control widgets like sliders and knobs can be created. The design and manipulation of the

[1]http://qutecsound.sourceforge.net
[2]http://www.sourceforge.net/projects/qutecsound

widgets is all graphical, and requires no textual programming. See 2.1 below.

Manual Panel The html version of the Csound Manual can be displayed in this panel. The reference for an opcode under the editor cursor can be called with the default short-cut Shift+F1.

Console Panel The output from Csound for the current document is displayed in this panel.

Inspector Panel Shows a dynamically generated hierarchical list of important sections from the current file. It shows instrument definitions and labels, definition of User defined opcodes, f-table definitions, and score sections.

When the current document tab is changed, all the panels which depend on the document like the widget panel and the inspector panel, change to show the current data.

There are three additional windows in the GUI:

Configuration dialog Allows setting options for Csound execution, Environment and interface options.

Utilities dialog Simplifies usage of the Csound utilities -applications which preprocess files for certain opcodes- can be called and controlled through this dialog window.

Live Event Panels These windows which vary in number according to the current document contain a spreadsheet-like interface for manipulating Csound score events which can be sent, manipulated and looped while Csound is running. They can also be processed using the simple python *qutesheet API* (see section 2.3).

2.1 Widgets

The widget panel allows creation of versatile graphical control interfaces. Data widgets provide bi-directional interchange of values with Csound through the API. The values can be updated synchronously or asynchronously depending on the QuteCsound configuration and the usage of *invalue/outvalue* or *chnget/chnset*[3] opcodes in the csd file.

[3]The invalue/outvalue opcodes call a registered callback when they are used, while chnget/chnset only change internal values which must be polled

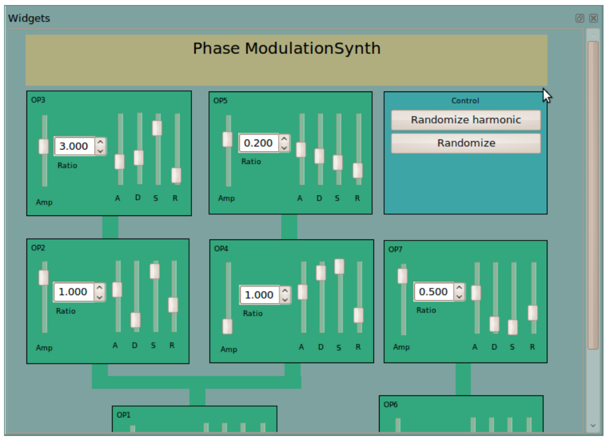


Figure 3: Widget Panel detail

An example of how the widget panel can be used is shown in figure 3.

These are the available data widgets:

Slider Ordinary sliders which become horizontal or vertical depending on the relation between width and height.

Knob Simple rotary knobs.

Labels and Displays Text widgets that display immutable text (labels), or text that can be changed only from Csound, not using the mouse[4].

Text Editor A text entry widget.

Number widgets The ScrollNumber and SpinBox widgets offer number value inputs and outputs with mouse and keyboard control.

Menus The menu widget allows creating a Drop-down or Combo Box for selection from a list.

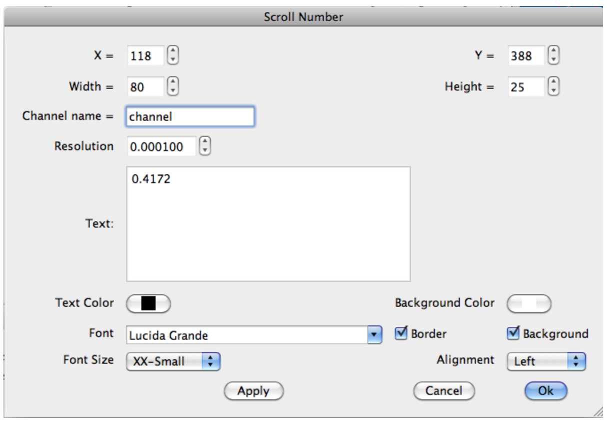
Controller The controller widget can be a slider, a meter or an XY controller. It can also be used as a “LED” display.

Button Buttons can be data widgets (sending different values depending on whether they are pressed or not) or score event generators. They can also hold images.

CheckBox A simple checkbox which can take a value of 1 or 0.

There are three other widgets which are used to display information from Csound:

[4]While this separation is not really necessary or useful, it follows the MacCsound format choices, and was kept for compatibility purposes only


Scroll Number

X = 118
Y = 388
Width = 80
Height = 25
Channel name = channel
Resolution 0.000100
Text: 0.4172
Text Color
Background Color
Font Lucida Grande
[✓ Border]
[✓ Background]
Font Size XX-Small
Alignment Left

Figure 4: The Widget Preferences dialog

Graphs The Graph Widget displays Csound F-tables, as they are created by Csound, and also displays data from the *dispfft* and *display* opcodes, which allow monitoring any signal or its spectrum. There is a ComboBox which allows selection of the current tables. The table shown can also be selected through a Csound API channel.

Scopes The Scope widget shows visual representations of Csound output buffer. It can act as a traditional Oscilloscope or as a Lissajou, or Poincare display.

Consoles The Console widget shows the Csound console output as a widget in the widget panel.

Widgets can be created by right-clicking on the widget panel, and selecting a type of widget from a list. They can be moved and resized by entering the 'Edit Mode' with the keyboard short-cut Ctrl+E.

All widgets have a configuration dialog which can be shown by right clicking on them and selecting 'Properties' or by double clicking when in edit mode. The Properties dialog for a text widget is shown on figure 4. Properties like size and position can be set in these dialogs. The channel name through which the widget transmits (if it can according to its type) can also be set here.

To receive data from the widgets, they must be assigned to a control variable (*k-rate*) using the *invalue* opcode like this:

```
kval invalue "channel"
```

Conversely, to send data to a widget, the *outvalue* opcode can be used.

```
outvalue "chan", kval
```

There are some handy organization functions for selected widgets like align and distribute to aid in creating better looking widget panels.

The widgets are saved as text in the csd file, but this text is always hidden from the user[5]. Each widget is currently represented by a single line of text, which looks something like this:

```
ioSlider {23, 244} {414, 32} -1.000000
1.000000 -1.000000 amount
```

This follows the original MacCsound format strictly, which enables QuteCsound to open and generate files which are interchangeable with MacCsound. It is somewhat human-readable and editable, but impossible to extend without breaking compatibility. A lot of internal work has already been done to move past the MacCsound widget format to an XML based format which will allow easier extensibility and parsing, and new widget types.

2.2 Live events panel

Each document can have any number of Live Event Panels. A live event panel is a place where Csound score events can be placed for interactive usage during a Csound run. It is shown in figure 5. The live event panel can display the information in the traditional Csound score format or as a spreadsheet with editable cells. It offers a group of common processing functions which can be applied quickly to a group of cells like addition, multiplication, and generation functions like fill linearly or exponentially or random number generation. There is also support for some of the basic Csound Score preprocessor features like tempo control and the carry operator '.'.

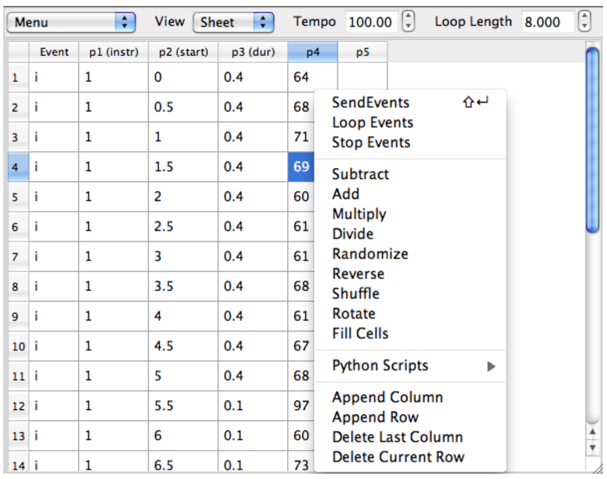
Score events can be copied/pasted to/from text view and sheet view seamlessly.

Live Event Panels are also stored as text in the main csd file, and are hidden from the user in the main text editor.

2.3 The QuteSheet Python API

A simple python API has been devised to enable transformation of data from the Live Events Panel using Python. The cells (both the selected and the complete set) are passed to the python script as arrays in any particular organization (by rows, by columns, by individual cells), and can be returned with a single function specifying the new data and where it should

[5]This behavior will probably change in the future to allow text modification of the widgets



Menu · View: Sheet · Tempo: 100.00 · Loop Length: 8.000

	Event	p1 (instr)	p2 (start)	p3 (dur)	p4	p5
1	i	1	0	0.4	64	
2	i	1	0.5	0.4	68	
3	i	1	1	0.4	71	
4	i	1	1.5	0.4	69	
5	i	1	2	0.4	60	
6	i	1	2.5	0.4	61	
7	i	1	3	0.4	61	
8	i	1	3.5	0.4	68	
9	i	1	4	0.4	61	
10	i	1	4.5	0.4	67	
11	i	1	5	0.4	68	
12	i	1	5.5	0.1	97	
13	i	1	6	0.1	60	
14	i	1	6.5	0.1	73	

Figure 5: The Live Events Panel

be placed in the sheet. The python scripts can be stored in a directory which is scanned every time the right-click menu in the event sheet is triggered, effectively allowing "live coding" of score transformations while the Csound audio engine is running.

2.4 Code Graph

QuteCsound can parse the currently active document to generate a dot language file, which can be rendered using graphviz[6] . Although complicated files produce diagrams that are too complex, this feature is useful for beginners to see how the variables and opcodes are connected in a visual way. The output of this action can be seen in figure 6.

3 Architecture

QuteCsound requires Qt 4[Nokia, 2010], Csound and libsndfile[de Castro Lopo, 2010] to build.

Qt is a mature cross-platform graphical toolkit, which is used in several well known free-software audio projects like QJackCtl. It also has cross-platform libraries for non GUI stuff like networking, XML parsing, printing and threading which has saved time and avoided the need for additional dependencies. It also has extensive facilities for internationalization and translation. It is distributed in separate dynamic libraries which can be distributed with the executable.

Libsndfile is a well established audiofile

[6]Graphviz is a package for generating flowchart style graphics using the dot language. More information can be found on www.graphviz.org

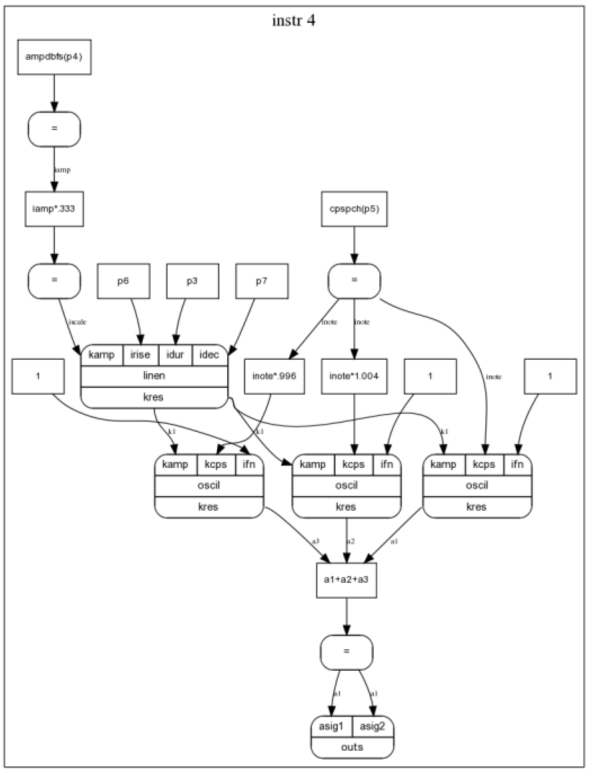


Figure 6: The Code Graph Output

read/write library, which is very well maintained, stable and efficient.

All audio and MIDI I/O is handled by Csound itself, except the Record function, which copies the Csound output buffer after every control block to a ring buffer and writes it to disk from another thread. This means that QuteCsound supports output to Jack, Coreaudio and WinMME as well as generic interfaces like Portaudio.

Csound can be run either as a completely independent process in a separate Terminal application, on the same application thread (usually undesirable) or on a separate thread using the API through the *CsoundPerformanceThread*[7] C++ interface from `interfaces/csPerfThread.hpp` in the Csound sources.

Data update is requested by Csound synchronously through its callback functions (set using the `csoundSetInputValueCallback` and `csoundSetOutputValueCallback` functions) when the *invalue/outvalue* opcodes are used. Data for Csound is polled in that callback, but data for the widgets is queued, to be processed

[7]this class handles some of the threading issues associated with passing events to Csound and with starting/pausing/stopping the engine

at a slower rate. The values can also be updated asynchronously with the *chnset/chnget* opcodes. The values are then updated from a timer triggered thread in QuteCsound, forcing locking of values.

An interesting feature of MacCsound which has been emulated in QuteCsound is that widgets with the same channel name update each other even when Csound is not running. This is useful to avoid Csound code for common things like showing the numeric value of a slider widget, and it also gives a (false but expected and rewarding) sense of an "always running" engine. In practical terms, this implies that when Csound is running, widgets must first take values generated from Csound, and then propagate their values to other widgets.

3.1 Documentation integration

The Csound documentation is highly integrated in QuteCsound. It can be easily called from the Help menu, or to find the reference for the opcode under the editor cursor. The documentation also contains an XML file of all the opcode definitions organized by category. This file is used for syntax highlight, but also for showing opcode syntax in the status bar, populating an opcode selector organized by categories and building the code graph code (to know the input/output variable names and types).

4 Examples Menu

The Examples menu in QuteCsound includes a large set of introductory examples and tutorials, reference examples for the widgets and a group of 'classic' Csound compositions like Boulanger's *Trapped in Convert*, and Csound realizations of important historical pieces like Chowning's *Stria* and Stockhausen's *Studie II*. The examples menu also contains a set of realtime synths, some useful files for things like I/O monitoring and file processing, and a 'Favorites' menu which shows the csd files from a user specified directory.

5 The future

Plans for the future include completion of the new widget format, and development of new widgets like a table widget. It would be nice to have a synchronization option to make loops sync to a master loop.

A lot of work has been done to enable the exporting of stand alone applications, so this will hopefully be finished soon.

And of course fix some of the bugs along the way.

6 Acknowledgements

Many thanks to the Csound developers, specially John ffitch and Victor Lazzarini for their assistance with usage of the API during the development of QuteCsound, and their quick response to issues and particular needs. Also, many thanks go to the users who have contributed many ideas, translations, bug reports and testing. Thanks to people like Joachim Heintz, Andy Fillebrown and François Pinot, for their contribution to development and packaging.

References

Richard Boulanger, editor. 2000. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming*. MIT Press.

Andres Cabrera, editor. 2010. *The Csound 5.12 Manual*.

Erik de Castro Lopo. 2010. Libsndfile. http://www.mega-nerd.com/libsndfile/api.html.

John ffitch. 2005. On the design of csound 5. In *Proceedings of the 2005 Linux Audio Conference*, ZKM, Karlsruhe, Germany.

Matt Ingalls. 2005. Maccsound. http://www.csounds.com/matt/MacCsound/.

Nokia. 2010. Qt – cross-platform application and ui framework. http://qt.nokia.com/.