The Faust Online Compiler: a Web-Based IDE for the Faust Programming Language

Romain Michon, Yann Orlarey

The Faust Online Compiler: a Web-Based IDE for the Faust Programming Language

Romain MICHON and **Yann ORLAREY**
GRAME, Centre national de création musicale
9 rue du Garet
69202 Lyon,
France,
rmnichon@gmail.com and orlarey@grame.fr

Abstract

The Faust Online Compiler is a PHP/JavaScript based web application that provides a cross-platform and cross-processor programming environment for the Faust language. It allows to use most of Faust features directly in a web browser and it integrates an editable catalog of examples making it a platform to easily share and use Faust objects.

Keywords

Faust, Web Application, Digital Signal Processing.

1 Introduction

Faust [Orlarey et al., 2002] is a functional programming language specifically designed for real-time signal processing and synthesis. Thanks to specific architecture files, a single FAUST program can be used to produce code for a variety of platforms and plug-in formats. These architecture files act as wrappers and describe the interactions with the host audio and GUI system.

The Faust Online Compiler is a PHP/-JavaScript based web application that provides a cross-platform and cross-processor programming environment for the Faust language. It makes it possible to use most of Faust features directly in a web browser. In many ways, we think it is solving one of Faust's weakness: the great number of dependencies that using its architecture files involved. Indeed, even though Faust doesn't need any specific library to be installed on a system to use it, some architecture files prove to be quite complicated to use because of their dependencies. It is also solving the system architecture issues that some Windows users, as an example, might have encountered in the past.

The Faust Online Compiler integrates an editable catalog of examples making it a platform to easily share and use Faust objects. We also think it is a pedagogical tool where it is possible to see in the most interactive way a digital signal processing statement, its C++ equivalent, its diagram representation, its documentation and finally its results.

It has been implemented in the frame of the development of the new Faust website[1] and it can be easily embedded in any HTML web page or installed on an Apache server.

2 Interface

The Faust Online Compiler is a web application made of one fixed size block. Its size has been defined to fit in the new Faust website but it can be easily embedded and scaled to be adapted to any `div` of an `HTML` page using the `iframe` mechanism (see 4). A *full page* mode is also available to provide a more comfortable workspace.

The navigation between the states of the Faust object is made through different tabs that can be selected in a navigation bar (see frame 1 in figure 1).

2.1 Editing the code

The Faust code in the Faust Online Compiler can be edited using two different ways. The most obvious one is by using the code editor that can be found in the *Faust Code* tab. The code editor `CodeMirror`[2] is used to carry out this task. It has been chosen because its compatibility with a great number of web browsers (Firefox, Safari, Opera, IE, Google Chrome, etc.). It uses a syntax highlighting specific to the Faust programming language.

Thanks to an `AJAX` script the Faust Online Compiler also allows to drop any `.dsp` file con-

[1] `http://faust.grame.fr/`
[2] `http://codemirror.net/`

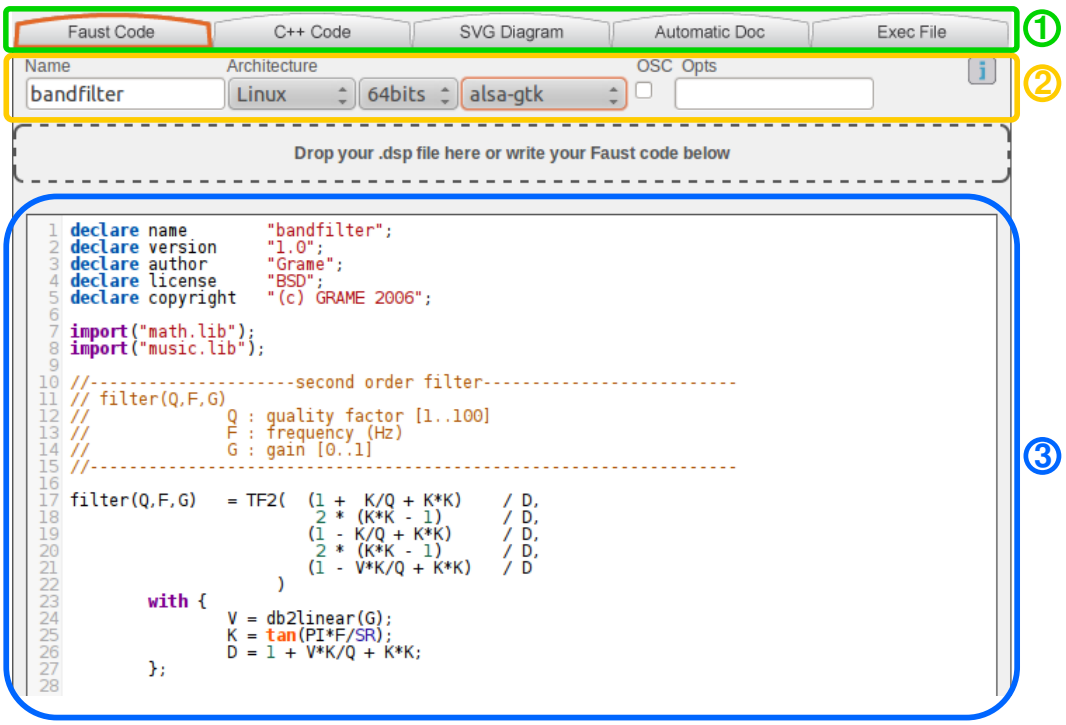

Figure 1: Overview of the compiler zone of the FAUST ONLINE COMPILER. Frame 1 contains the navigation bar, frame 2 the compilation options and frame 3 the FAUST code editor or the different kind of results of the compilation.

taining a FAUST program in the file drop area that is located just above the code editor (see figure 1). This drop area is present in every tabs. This allows to compile a FAUST code into many different elements in a very interactive way and very quickly.

2.2 Compilation

2.2.1 Compilation Options

The compilation options menu can be found just under the navigation bar (see frame 2 in figure 1). It allows to set the kind of executable file that will be generated during the `C++` compilation. Most of the FAUST architectures can be used and the `OSC` support option [Fober et al., 2011] can be selected if it is allowed by the architecture. Further options can be given to the Faust compiler by filling the `Opts` text area.

The 64 bits Linux server (Ubuntu) that hosts the FAUST ONLINE COMPILER at GRAME makes it possible to compile the `C++` code generated by the FAUST compiler in 32 or 64 bits for the following platforms (in function of the selected architecture):

- Linux
- Windows using the `MinGW`[3] cross compiler
- OSX 10.6 using the distant command execution system of `SSH`[4]

If any of these options is changed in the `C++ code` or the `Exec File` tabs, the result will be automatically updated.

2.2.2 Compilation

In order to carry out the compilation, a temporary folder containing the FAUST file and a Makefile generated in function of the options given by the user and the architecture he selected is created on the server. It will be used by the FAUST ONLINE COMPILER to generate the `C++` code, the `SVG` diagram, the documentation, the executable file and the different downloadable packages.

[3] `http://www.mingw.org/`
[4] `http://www.openssh.com/`

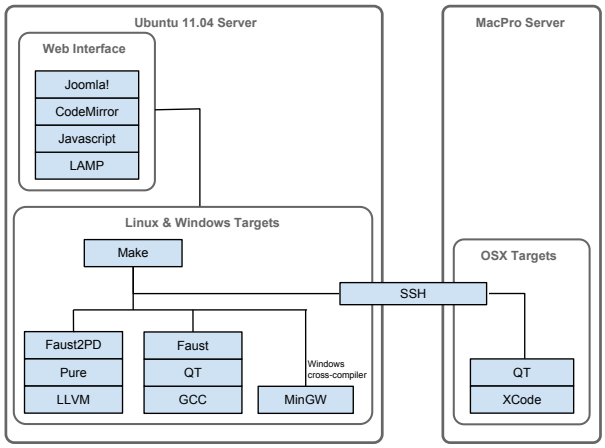

Figure 2: Diagram of the server components.

2.3 Getting the C++ Code

If the `C++` tab is selected, the FAUST compiler is called on the server by the Makefile that will also use `Highlight`[5] to color the generated `C++` code.

If the FAUST code given by the user contains some errors, the message returned by the FAUST compiler will be displayed instead of the `C++` code. The FAUST ONLINE COMPILER will react the same way in the case of the tabs `SVG diagram` and `Automatic Doc`.

2.4 Displaying the SVG diagram of the Faust Code

When the `SVG Diagram` button is clicked, in the same way than for the `C++` code, the Makefile created on the server call the FAUST compiler to generate a `SVG` block diagram of the FAUST code. The result is then displayed just under the file dropping area (see figure 3) and can be browsed by clicking on the different boxes.

2.5 Displaying the Mathematical Documentation of the Faust Object

Some recent developments made at GRAME in the frame of the ASTREE project (see *Acknowledgments*) on the FAUST compiler make it possible to generate automatically an all-comprehensive mathematical documentation of a FAUST program under the form of a complete set of LaTeX formulas and diagrams [Barkati et al., 2011]. The goals of such a self mathematical documentation are:

1. *Preservation*, i.e. to preserve signal processors, independently from any computer language but only under a mathematical form;
2. *Validation*, i.e. to bring some help for debugging tasks, by showing the formulas as they are really computed after the compilation stage;
3. *Teaching*, i.e. to give a new teaching support, as a bridge between code and formulas for signal processing;
4. *Publishing*, i.e. to output publishing material, by preparing LaTeX formulas and SVG block diagrams easy to include in a paper.

The FAUST ONLINE COMPILER is compatible with this function and can display the resulting pdf file using `Google Document Viewer`[6]. The Makefile described in 2.3 is also used here to carry out the compilation process.

A screen shot of the `Automatic Doc` tab can be seen in figure 4.

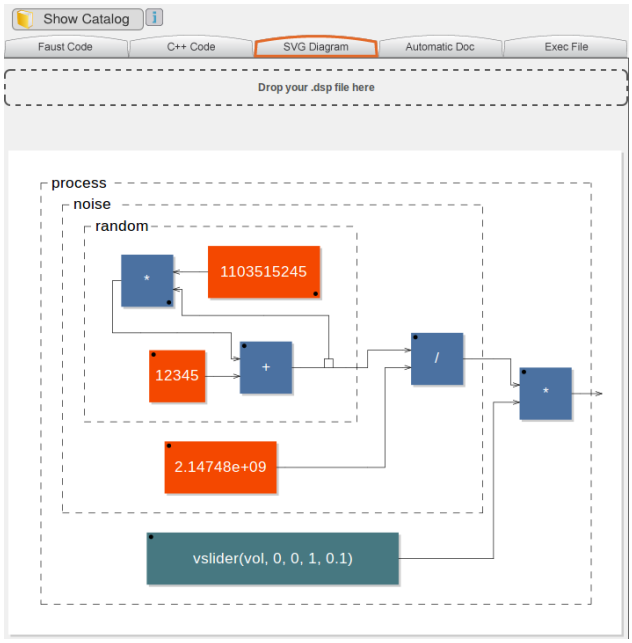

Figure 3: Screenshot of the `SVG` Diagram tab.

[5] `http://www.andre-simon.de/doku/highlight/en/highlight.html`

[6] `https://docs.google.com/viewer/`

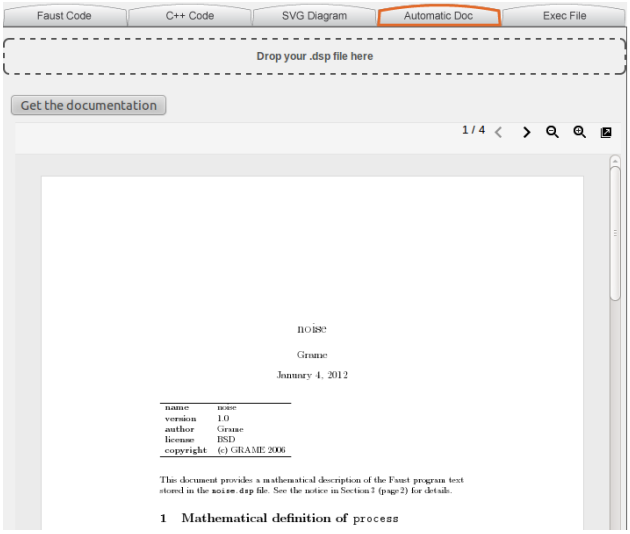

Figure 4: Screenshot of the `Automatic Doc` tab.

2.6 Executable File

When the `Exec File` button is clicked in the navigation bar, the `C++` compilation is carried out in function of the platform and the processor architecture selected. If this task is successful, a download button appear. Otherwise the error message returned by the `C++` compiler is displayed. The downloadable file will be either an executable file either a package containing several files, in function of the FAUST architecture.

3 Catalog of Examples

The FAUST ONLINE COMPILER has been upgraded with an interactive platform to easily share and use pieces of FAUST code: the catalog of examples. It is made of two elements: the catalog itself and the example saver.

3.1 The Catalog

The catalog has the form of a file browser where each FAUST code is sorted into different categories (see figure 6):

- Effects
- Faust-STK [Michon and Smith, 2011]
- Synthesizers
- Tools

A set of "user" categories editable by the users were also created.

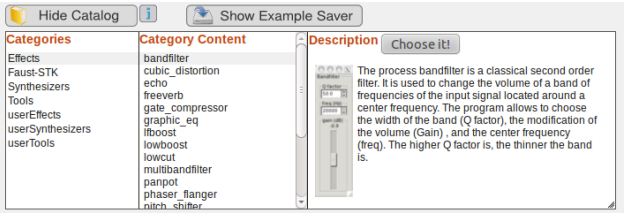

Figure 6: The catalog of examples of the FAUST ONLINE COMPILER.

When a FAUST object is selected, a short description and a screenshot of it are displayed in the `Description` column. The object can then be used in the online compiler either by double-clicking on it either by pushing the `Choose it!` button.

The catalog of examples can be displayed in every tabs of the FAUST ONLINE COMPILER. So, as an example, someone can choose just to see the block diagram of an object or its automatically generated documentation.

3.2 The Example Saver

As said before, the catalog of examples of the FAUST ONLINE COMPILER is intended to be a platform to share FAUST objects between users. This is made possible by the example saver (see figure 7) that allows someone to save the FAUST code he edited or he dropped within the catalog. No identification is required to carry out this task: anyone can freely upload his work in the catalog. When creating a new example, users can also add a quick description to their object as well as a screenshot.

If an existing example is modified by a user different than the one who created it, every users that contributed to this example will be notified by e-mail of the changes made in their FAUST code. Finally, any user can freely delete examples from the catalog. As in the case mentioned before, editors will be notified of this modification with an e-mail.

In order to keep a good order in the catalog and to avoid duplicates or SPAMs, a voting system has been implemented. This can also help to promote the best user examples from the catalog.

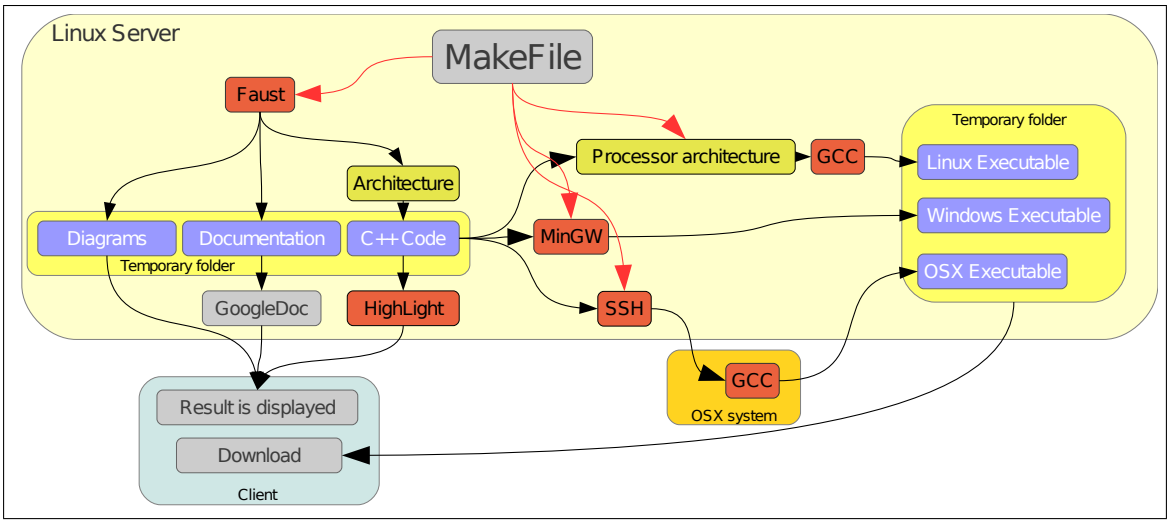

Figure 5: FAUST ONLINE COMPILER overview.

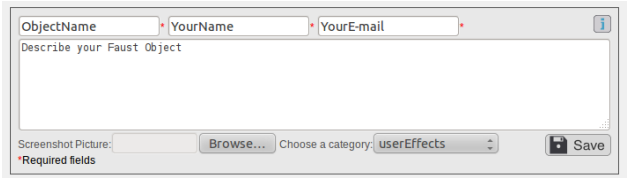

Figure 7: The example saver of the FAUST ONLINE COMPILER.

4 Exporting and embedding the Faust Online Compiler

The FAUST ONLINE COMPILER can be very easily embedded in any website using an *iframe*:

```
<iframe src="http://faust.grame.fr/
    compiler" style="border:none"
    height="1200px" width="722px"></
    iframe>
```

It can also be installed on an Apache server[7] just by copying the source code available on the FAUST repository[8] on it.

5 Current and Future Works

Recent improvements on the FAUST compiler allow it to generate `JAVA` and `LLVM` code instead of `C++` from a FAUST code. This obviously brings to the idea of implementing a FAUST architecture to generate `JAVA` applets directly usable from a web browser. Thus, the FAUST ONLINE COMPILER should soon be upgraded with a new tab running a web app generated in function of a FAUST code edited in the `Faust Code` section. Our hope is that the `JAVA` applet will be updated almost in real time without having to restart it every time a change would be made. Therefore, this feature could offer some sort of live codding possibilities.

As the FAUST ONLINE COMPILER prove to be a good solution to the platform and dependencies issues related to the use of architecture files in FAUST, it can be interesting to establish some links between it and the `FaustWorks`[9] program. In that special case, `FaustWorks` would be used to edit the FAUST code and see its resulting diagram, `C++` code, etc. but the compilation would be carried out on the server allowing it to generate FAUST applications for any platforms, processors and architectures.

6 Conclusion

The FAUST ONLINE COMPILER makes the FAUST experience easier than ever by offering a programming environment ready to use and free from any installation and compatibility issues.

It is certain that web technologies are having an increasingly important role in the domain of music technology. The arrival of new standards

[7] `http://www.apache.org/`
[8] `http://sourceforge.net/projects/faudiostream/`
[9] `FaustWorks` is an IDE for FAUST implemented with `Qt`.

like HTML5 or of new web-audio APIs is opening the way to new paths of exploration such as "Web Digital Signal Processing".

7 Acknowledgements

This work has been carried out in the frame of the ASTREE project[10] (ANR-08-CORD-003) on the preservation of real-time musical pieces.

References

Karim Barkati, Dominique Fober, Stéphane Letz, and Yann Orlarey. 2011. Two recent extensions to the faust compiler. In *Proceedings of the Linux Audio Conference (LAC-2011)*, pages 1–8, National University of Ireland, Maynooth, Ireland.

Dominique Fober, Yann Orlarey, and Stéphane Letz. 2011. Faust architecture design and osc support. In *Proceedings of the Conference on Digital Audio Effects (DAFx-11)*, pages 213–216, IRCAM, Paris, France.

Romain Michon and Julius Smith. 2011. Faust-stk: a set of linear and nonlinear physical models for the faust programming language. In *Proceedings of the Conference on Digital Audio Effects (DAFx-11)*, pages 199–204, IRCAM, Paris, France.

Yann Orlarey, Dominique Fober, and Stéphane Letz. 2002. An algebra for block diagram languages. In *Proceedings of the International Computer Music Conference (ICMA)*, pages 542–547, Gothenburg, Sweden.

[10] http://www.ircam.fr/