

Plugin Programming with Faust

Albert Gräf <aggraef@gmail.com>

IKM, Music-Informatics at JGU Mainz

<https://bitbucket.org/agraef/agraef.bitbucket.org>

miniLAC 2016 @ Berlin



JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Plugin Programming with Faust

This workshop is all about **Faust** (by Grame) and the Faust **LV2** and **VST** plugin architectures (by yours truly). Both architectures work on **Linux** and **Mac** (porting to Windows should be a piece of cake!) and produce functionally equivalent plugins (use LV2 with open-source and VST with commercial DAWs).

- ▶ Installing the required software
- ▶ How to program basic instrument and effect plugins with Faust
- ▶ Compiling LV2 and VST plugins
- ▶ Using Faust plugins in various Linux DAWs (Ardour, Bitwig, Qtractor, Tracktion) and modular hosts (Carla)
- ▶ Optional features: custom GUIs, MIDI control and MIDI tuning (MTS) capabilities



Let's Talk About Faust!



J. GUTENBERG
UNIVERSITÄT MAINZ

Faust: DSP for Everyone!?

- ▶ Let's face it: DSP is **hard**! Lots of math and special coding techniques.
- ▶ **But:** Others have learned it; you can, too!
- ▶ **Faust** makes coding dsps a lot easier – less code to write, the Faust compiler takes care of most of the boring (and error-prone) technicalities.
- ▶ Lots of good **open-source code** and **ready-made components** available – you don't have to start from scratch.
- ▶ Lots of **in-depth documentation** available, too.
<https://ccrma.stanford.edu/~jos/aspf/>
- ▶ You don't even have to install Faust – use the **Faust online compiler** to compile your Faust sources on the web!



Faust Pros

- ▶ **formal semantics** – Faust gives you a mathematical *specification* of your signal processor, not just executable code
- ▶ very **high-level** – you concentrate on the algorithm, the Faust compiler produces optimized code and takes care of all the nitty-gritty details
- ▶ **cross platform** – use of “architectures” (C++ glue code) to target different environments and plugin APIs
- ▶ output is **C++ code** – you can still tweak the results if you’re a seasoned dsp programmer
- ▶ experimental **faust2** branch gives you an embeddable **JIT compiler** for rapid development tools (FaustLive, pMix) and **web audio** support



Faust Cons

- ▶ **learning curve** – you have to learn a new programming language
- ▶ **functional programming** requires a different mindset than C/C++ (no side effects!)
- ▶ still lacks a few important features – most notably **short-circuit conditionals** and **multi-rate processing**
- ▶ if your **target architecture** isn't supported yet (increasingly unlikely), you'll have to roll your own architecture code



Installing the VST SDK Sources

(Only needed for compiling Faust VST plugins. Best installed before Faust, otherwise you'll have to edit the `faust2faustvst` script.)

The **VST SDK** is proprietary software, you need to get it from Steinberg:
<http://www.steinberg.net/en/company/developers.html>

The scripts and Makefiles look for the SDK in various locations under `/usr` and `/usr/local`. **NOTE:** Rename the folder so that it doesn't contain spaces. E.g.:

```
unzip vstsdk365_12_11_2015_build_67.zip
sudo mkdir -p /usr/local/src
sudo mv VST3\ SDK /usr/local/src/vstsdk
```

Arch users: Install the SDK from the AUR (`steinberg-vst36`).



JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Installing Faust

- ▶ **Faust** (includes compiler, architectures and helper scripts):

```
git clone git://git.code.sf.net/p/faudiostream/code faust
cd faust
make
sudo make install
```

- ▶ Use `make world` for additional bits and pieces (OSC and HTTP support, needs `liblo` and `libmicrohttpd`).
- ▶ You can also use the `faust2` branch for the real cutting-edge stuff (adds even more dependencies such as LLVM).
- ▶ **Arch users:** You can find up-to-date packages in the AUR (`faust-git` and `faust2-git`).



Additional Prerequisites

You should be able to find these in your distro's package repositories:

- ▶ **Boost** headers: <http://www.boost.org/>
- ▶ **LV2** headers: <http://lv2plug.in/>
- ▶ **Qt** (for the custom GUI support): <http://www.qt.io/>

You need Qt4 for Ardour, Qt5 for the other DAWs.

Arch users: Install packages `boost`, `lv2`, `qt4`, `qt5-base`, `qt5-x11extras`.



Faust-LV2 and -VST

```
git clone https://bitbucket.org/agraef/faust-lv2.git  
git clone https://bitbucket.org/agraef/faust-vst.git
```

You only need this for the Makefile and the examples, the architectures and helper scripts are also included in the latest Faust git sources.

Arch users: You can find faust-lv2-git and faust-vst-git packages with the sample plugins in the AUR. (These are configured for Qt5 GUIs and dynamic manifest support and will install into /usr/lib.)



JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

DAW and Other LV2/VST Hosts

- ▶ We'll mostly use **Qtractor** for the demo. You can probably find this in your distro's package repositories. Latest (March a.k.a. "Hazy Photon") sources can be found here: <http://downloads.sourceforge.net/qtractor/qtractor-0.7.5.tar.gz>
- ▶ **Other DAWs** and sequencers that you might want to try: Ardour, Bitwig Studio, Radium, Reaper (Mac), Renoise, Tracktion, ...
- ▶ FalkTX's **Carla** works both as a stand-alone modular host and as a bridge to run LV2 plugins in VST-only hosts and vice versa.
- ▶ The author's **lv2plugin~** external can be used to run LV2 plugins in Pd (without GUI).



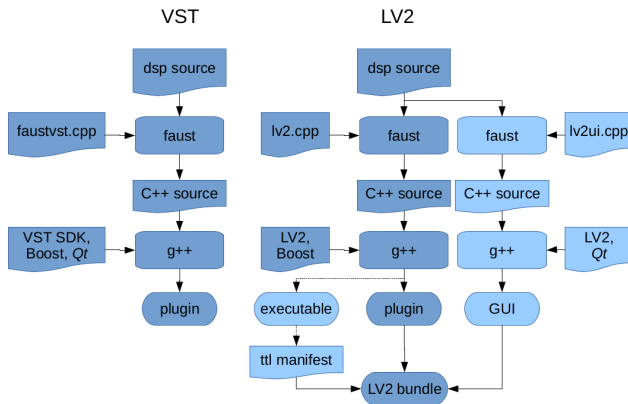
Compiling the Sample Plugins

- ▶ Basic compilation: just make
- ▶ Compile with Qt GUI support: `make gui=1`
`make gui=1 qmake=/usr/lib/qt4/bin/qmake`
- ▶ LV2s with dynamic manifests (needs host support):
`make dyn-manifests=1`
- ▶ Installing the sample plugins: just copy them into your `~/.lv2` or `~/.vst` folders (create if necessary), e.g.:

```
mkdir -p ~/.lv2
cp -r lv2/faust.lv2 ~/.lv2
mkdir -p ~/.vst/faust
cp examples/*.so ~/.vst/faust
```



Compilation Process



Using the Online Compiler

<http://faust.grame.fr/onlinecompiler>

Drop your .dsp file here

Faust Code

Compiled Code

Diagrams

Automatic Doc

Exec File

Name

Language

Architecture

OSC


amp

C++

Linux

vst-64bits-qt5

☐



```
1  /* -----
2  author: "Albert Graef"
3  name: "amp"
4  version: "1.0"
5  Code generated with Faust 2.0.a41 (http://faust.grame.fr)
6  ----- */
7
8  #ifndef __mydsp_H__
9  #define __mydsp_H__
10
11  /*-----
12  FAUST Architecture File
13  Copyright (C) 2014-2016 Albert Graef <aggraeef@gmail.com>
14  -----
15  This program is free software; you can redistribute it and/or modify
16  it under the terms of the GNU Lesser General Public License as
17  published by the Free Software Foundation; either version 3 of the
18  License, or (at your option) any later version.
19
20  This program is distributed in the hope that it will be useful,
21  but WITHOUT ANY WARRANTY; without even the implied warranty of
22  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
23  GNU Lesser General Public License for more details.
24
25  You should have received a copy of the GNU Lesser General Public
26  License along with the GNU C Library; if not, write to the Free
27  Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
28  02111-1307 USA.
29  -----
30  -----*/
31
32  /* VST architecture for Faust synths. */
33
34  /* NOTE: This requires one of the Boost headers (boost/circular_buffer.hpp),
35  so to compile Faust programs created with this architecture you need to
36  have at least the Boost headers installed somewhere on your include path
37  (the Boost libraries aren't needed). */
38
39  #include <stdlib.h>
40  #include <stdint.h>
```



GUTENBERG
UNIVERSITÄT MAINZ

Using the Helper Script

```
faust2lv2 myplugin.dsp; faust2faustvst myplugin.dsp
```

- ▶ -keep: keep C++ source code for tweaking
- ▶ -gui, -qt4, -qt5: custom GUI support (use QMAKE environment variable to set the path to the qmake executable if needed)
- ▶ -osc, -httpd, -qrcode: OSC / HTTP support (control plugins with OSC devices and web browsers)
- ▶ -style *style*: set the GUI style sheet (Default, Grey, etc., cf. /usr/local/include/faust/gui/Styles/)

LV2 only:

- ▶ -dyn-manifest: use dynamic manifests (recommended when using the tuning control; requires LV2 host support)
- ▶ -uri-prefix *uri*: set the prefix of the LV2 plugin URI



JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Rolling Your Own: A Complete Walk-Through

```
git clone https://github.com/agraef/lac16-faust-demo.git
```

- ▶ Ex01: basic sine generator
- ▶ Ex02: add requisite meta information (name, author, etc.)
- ▶ Ex03: add a few partials and gain/gate controls
- ▶ Ex04: add an envelop, master volume and pan controls
- ▶ Ex05: add nvoices meta data to create an instrument
- ▶ Ex06: add a custom GUI, automation, MIDI control
- ▶ Ex07: add a smoothing filter to avoid zipper noise

Also see included Makefile and README.



Instrument Support

- ▶ use `nvoices` declaration in Faust source or `-nvoices` option of helper script
- ▶ needs special `freq`, `gain` and `gate` controls
- ▶ the following are all supported automatically by instruments, no need to implement them in the Faust program:
 - ▶ pitch bends
 - ▶ MIDI CC 120+123 (all-sounds-off, all-notes-off)
 - ▶ RPNs 0 (pitch bend range), 1/2 (channel fine/coarse tuning)
 - ▶ MTS (MIDI Tuning Standard) sysex messages (see below)



Qt GUI Support

- ▶ was created by **Roman Svidler**, original source available at <https://github.com/rosvid/faust-vst-qt>
- ▶ leverages **existing Qt support** in Faust library (faustqt.h)
- ▶ creates a GUI following the **hierarchical layout** of controls in the Faust source
- ▶ Ardour needs **Qt4** GUIs (works in LV2 only), other hosts doing fine with **Qt5** (both LV2 and VST)
- ▶ **no working Mac support** yet (LV2+Qt4 compiles, but not working in Ardour OSX; VST still lacks Mac-specific widget reparenting code, only compiles without custom GUI right now)



MIDI CC Support

- ▶ increasingly less important, normally you can just use your DAWs MIDI learn capabilities instead
- ▶ but it's there if needed, just use `[midi:ctrl n]` attributes in control labels (active controls only)
- ▶ fully multi-timbral (keeps track of individual CC data for each MIDI channel)
- ▶ can be disabled with `-nomidicc` option if not needed/wanted



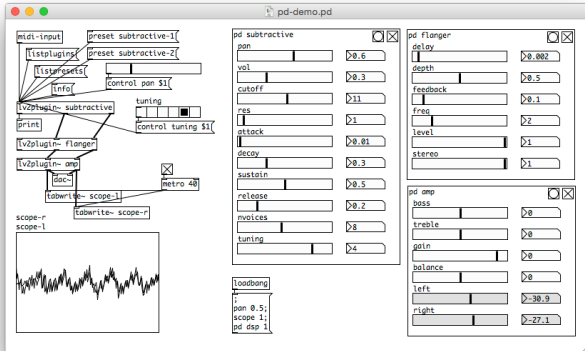
MIDI Tuning (MTS) Support

- ▶ supported MTS sysex message formats: universal realtime and non-realtime scale/octave 1 or 2 byte tunings (see faust-lv2 docs for details)
- ▶ MIDI tuning control: drop MTS sysex files into `~/ .faust/tuning` (or `~/Library/Faust/Tuning` on the Mac), tuning then becomes an automatable parameter
- ▶ handy tool for creating the sysex files:
<https://bitbucket.org/agraef/sclsysex>
- ▶ Caveat: MIDI tuning control in LV2 works best with *dynamic manifests*



Running LV2 Plugins in Pd

<https://bitbucket.org/agraef/pd-lv2plugin>



Conclusion

Creating great working LV2 and VST instrument and effect plugins with Faust has never been easier.

Future work:

- ▶ Port the Qt GUIs to Mac OS X. We should also look into lighter, more plugin-friendly alternatives for the GUI toolkit.
- ▶ Port the architecture to other targets (Audio Units, Pd, ...).
- ▶ Special support for the MOD (LV2 architecture).
- ▶ Support for MPE, the LV2 Time extension and MIDI outputs (passive controls).



Special Acknowledgements

- ▶ **Roman Svidler** for his Qt GUI code. Thanks Roman!
- ▶ **Dave Phillips** for testing and feedback. Thanks Dave!

Dave Phillips: “**Faustian**”

Created with Bitwig Studio running various faust-vst plugins, Fons’ zita-verb and samples from Loop Loft and Freesound.