WORKSHOP: LEARNING AND USING FAUST IN THE PD ENVIRONMENT

Albert Gräf

IKM, Music-Informatics
Johannes Gutenberg University (JGU) Mainz, Germany
aggraef@gmail.com

ABSTRACT

The workshop presents pd-faustgen2 as an interactive learning tool and live-coding programming environment for the Faust programming language. pd-faustgen2 runs inside the well-known Pd computer music software. The workshop introduces pd-faustgen2 in a fairly informal, hands-on manner. It is aimed at developers and educators trying to learn Faust, or teach their students about it.

1. INTRODUCTION

Learning Faust can be a daunting task. The language is actually not that complicated, and offers some striking advantages, making the endeavor well worth the effort. Nevertheless, Faust's underlying concepts often seem alien to musicians accustomed to graphical environments, and even to programmers well-versed in traditional (imperative and object-oriented) programming languages. This sometimes leaves newcomers at a loss, wondering how to accomplish even the most basic dsp tasks. These hurdles can be mitigated, however, with an interactive environment that fosters creative exploration, and helps building and testing Faust programs in quick iterations, using the technique of live-coding.

pd-faustgen2[1] aims to provide such an environment inside the well-known Pd software. The external is based on Pierre Guillot's pd-faustgen[2], which in turn was inspired by Grame's faustgen object for Max/MSP. pd-faustgen2 is a comprehensive update of pd-faustgen with many new features which bring it up to par with both Grame's faustgen and the author's pd-faust [1]. In particular, the external offers automatically generated Pd GUI elements as well as control through hardware interfaces (MIDI and OSC), so that all aspects of the Faust program under construction can be inspected and played with. The integrated Pd loader extension means that you can just type the name of a dsp file and have it recognized as an external object. An additional boon is that Faust programs are easily interfaced to all the other multimedia capabilities that Pd offers. pd-faustgen2 thus provides a complete environment for learning and utilizing Faust which will be useful for both development, self-learning, and higher education (i.e., computer music and dsp courses).

2. ABOUT THE WORKSHOP

The workshop introduces pd-faustgen2 in a hands-on fashion. The target audience are developers and educators trying to learn Faust (or teach their students about it). It is assumed that attendants have a passing familiarity with Pd and Faust, but we'll introduce them both, so no deep knowledge of either environment is required. We cover the following topics:

[1]https://github.com/agraef/pd-faustgen
[2]https://github.com/CICM/pd-faustgen

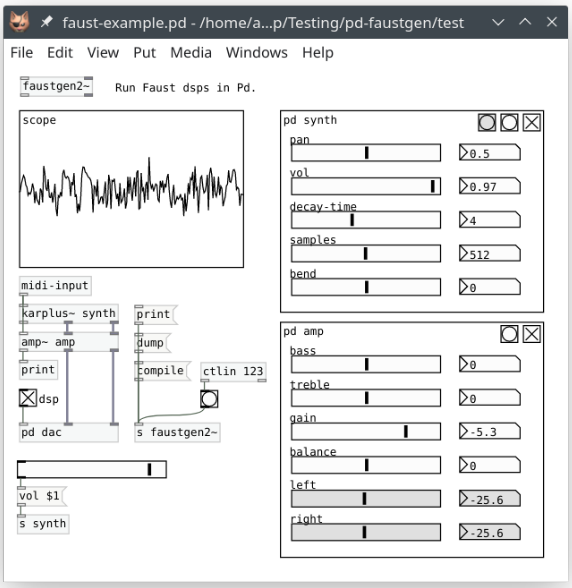

Figure 1: pd-faustgen2 example.

- A quick introduction to Pd and Faust, including some basic examples
- What you need: getting set up with pd-faustgen2 on Linux, Mac and Windows
- The pd-faustgen2 live-coding workflow: adding Faust programs to your Pd patches, generating Pd GUIs, recompiling after edits
- Adding MIDI control, recording and playing back OSC automation
- Some hands-on examples, e.g.: software synthesis, guitar effects, controlling modular hardware, etc.

Duration of the workshop: approximately 60 min

3. REFERENCES

[1] Albert Gräf, "Pd-Faust: An integrated environment for running Faust objects in Pd," in *Proceedings of the 10th International Linux Audio Conference*, Stanford University, California, US, 2012, pp. 101–109, CCRMA.