pd-faust: An integrated environment for running Faust objects in Pd

Albert Gräf
Department of Music Informatics
Johannes Gutenberg University Mainz
Contents

- Brief introduction to Faust
- Old Pd-Faust interface
- New interface
- Implementation
- Demo

Learn more:

Pure and Faust: Functional Programming for Media Applications, Tue, 04/17/2012 - 5:15pm – 6:30pm, CCRMA Classroom [Knoll 217]
import("music.lib");
vol = hslider("vol", 0.3, 0, 1, 0.01);
pan = hslider("pan", 0.5, 0, 1, 0.01);
freq = hslider("pitch", 440, 20, 2000, 0.01);
process = osci(freq)*vol : panner(pan);
Old Pd-Faust interface

- `puredata.cpp` architecture turns Faust dsp into Pd external
- `faust2pd` script generates Pd GUI for the control variables
- **static**: GUI must be regenerated and patch reloaded after changes
New Pd-Faust interface

- **dynamic**: Faust dsp can be reloaded while patch is running, GUI gets regenerated instantly
- **idea**: leverage Pure-Faust interface [LAC 2011] which is capable of hot-swapping Faust modules
- use Pd's **FUDI** protocol to create GUI inside subpatch
Additional goodies

- all control logic implemented in Pure
- both native Faust modules (.so) and LLVM bitcode modules (.bc) are supported
- more options to configure the GUI layout
- built-in MIDI playback, MMC sync
- MIDI and OSC controller mappings
- automation
- livecoding
Implementation

- pd-faust: written in Pure, gets compiled to native library of Pd externals (pd -lib)
- requires pd-pure (Pd Pure script loader) and pure-faust (Pure-Faust interface)
- fdsp~ and fsynth~ objects for effect and polyphonic synth units
fdsp~ and fsynth~ objects

inlets: 1 for control input, 1 per audio input channel

fdsp~ chorus fx1 0

dsp name

instance name

MIDI channel

#voices

fsynth~ organ synth1 0 8

outlets: 1 for control output, 1 per audio output channel
Demo
Future Work

- Better OSC sequencing (preferably via external DAW/sequencer software such as Ardour3 or Qtractor)
- Port pd-faust to other environments (Jack, LV2, SuperCollider...)