Hot Topics in Musical Acoustics Applied to Real-Time Sound Synthesis

Julius Smith
CCRMA, Stanford University

Acoustical Society of America 166th Meeting
Hot Topics Session

December 4, 2013
Outline
New Enabling Technologies

1. Smart-Phones and Tablets
   - High-quality audio in (mono) and out (stereo)
   - Fast multicore processors (exponentially growing speed)
   - Multitouch display screens (5 for iPhone, 11 for iPad)

2. Domain-Specific Languages
   - Functional AUdio STream (FAUST) — High-Level Signal-Processing Language (Yann Orlarey, GRAME)
   - Synth-A-Modeler — Physical Model “Diagram” to FAUST (Edgar Berdahl)

3. New Convex Optimization Techniques and Formulations (Esteban Maestre)
Smart-Phone/ Tablet Example: moForte Guitar
moForte Guitar

Real time on the iPhone 4S (all written in FAUST):

- Full electric-guitar + effects:
  - Six vibrating strings — general excitations
  - Distortion
  - Compression
  - Phaser
  - Five-band parametric equalizer
- Responds to
  - accelerometer, gyros, touches (plucks), swipes (strumming), …
- It is challenging to fully utilize five points of multitouch on the iPhone and eleven on the iPad!
Real time on the iPhone 4S (all written in FAUST):

- Full electric-guitar + effects:
  - Six vibrating strings — general excitations
  - Distortion
  - Compression
  - Phaser
  - Five-band parametric equalizer
  - Feedback
  - Wah pedal or Autowah
  - Flanger
  - Reverb

- Responds to
  - accelerometer, gyros, touches (plucks), swipes (strumming), …

- It is challenging to fully utilize five points of multitouch on the iPhone and eleven on the iPad!
moForte Guitar

Real time on the iPhone 4S (all written in FAUST):

- Full electric-guitar + effects:
  - Six vibrating strings — general excitations
  - Distortion
  - Compression
  - Phaser
  - Five-band parametric equalizer
  - Feedback
  - Wah pedal or Autowah
  - Flanger
  - Reverb

- Responds to accelerometer, gyros, touches (plucks), swipes (strumming), ...

- It is challenging to fully utilize five points of multitouch on the iPhone and eleven on the iPad!
Effects Running in Real Time on the iPhone 4S with All Six Strings Playing
iPhone CPU Performance

Enabling Technologies

- moForte Guitar
- CPU Performance
- Sound Examples

Faust Intro

Research
Distortion Guitar Sound Examples

- moForte Guitar Demo Video
- Distortion Guitar: (WAV) (MP3)
- Amplifier Feedback 1: (WAV) (MP3)
- Amplifier Feedback 2: (WAV) (MP3)
FAUST Introduction
**FAUST Language**

**Short FAUST Program Examples:**

```faust
process = _ ;
process = + ;
process = _ , _ : + : _ ;
process = pole(0.9) with { pole(p) = + ~ *(p); } ;
```

**Partial C++ Output for Last Example Above:**

```cpp
virtual void compute (int count, FAUSTFLOAT** input,
                     FAUSTFLOAT** output)
{
    FAUSTFLOAT* input0 = input[0];
    FAUSTFLOAT* output0 = output[0];
    for (int i=0; i<count; i++) {
        fRec0[0] = ((float)input0[i] + (0.9f * fRec0[1]));
        output0[i] = (FAUSTFLOAT)fRec0[0];
        // post processing
        fRec0[1] = fRec0[0];
    }
}
```
**FAUST Example:** zita_rev1.dsp

**Source Code (FAUSTeffect.lib):**

\[
zita_rev_fdn(f1,f2,t60dc,t60m,fsmax) =
((bus(2*N) \rightarrow allpass_combs(N) : feedbackmatrix(N)) \sim
(delayfilters(N,freqs,durs) : fbdelaylines(N))) ...
\]

**Block Diagram (drawn by the FAUST compiler):**

![Block Diagram](image)

**JACK-Aware Standalone App (generated from FAUST source):**

![JACK-Aware App](image)
## FAUST Main Programs Supported

<table>
<thead>
<tr>
<th>Shell Script</th>
<th>Arch. File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>faust2jack (f2j)</td>
<td>jack-gtk.cpp, jack-qt.cpp</td>
<td>JACK GTK standalone application</td>
</tr>
<tr>
<td>faust2jaqt</td>
<td></td>
<td>JACK QT4 standalone application</td>
</tr>
<tr>
<td>faust2caqt (f2ca)</td>
<td>ca-qt.cpp, pa-gtk.cpp, pa-qt.cpp</td>
<td>CoreAudio QT4 standalone application</td>
</tr>
<tr>
<td>faust2pa</td>
<td></td>
<td>PortAudio GTK standalone application</td>
</tr>
<tr>
<td>faust2paqt</td>
<td></td>
<td>PortAudio QT4 standalone application</td>
</tr>
<tr>
<td>faust2netjackqt</td>
<td>netjack-qt.cpp</td>
<td>server with libnetjack support</td>
</tr>
<tr>
<td>faust2oss</td>
<td>oss-gtk.cpp</td>
<td>OSS GTK standalone application</td>
</tr>
<tr>
<td>faust2alsa</td>
<td>alsa-gtk.cpp, alsa-qt.cpp</td>
<td>ALSA GTK standalone application</td>
</tr>
<tr>
<td>faust2alqt</td>
<td>alsa-console.cpp</td>
<td>ALSA QT4 standalone application</td>
</tr>
<tr>
<td>faust2aalsaconsole</td>
<td>android.cpp</td>
<td>ALSA terminal program</td>
</tr>
<tr>
<td>faust2android</td>
<td></td>
<td>Android phone/tablet application</td>
</tr>
<tr>
<td>faust2ios</td>
<td>ios-coreaudio[-jack].cpp</td>
<td>iOS phone/tablet application</td>
</tr>
<tr>
<td>faust2rpi*</td>
<td>[alsa—netjack]-console.cpp</td>
<td>Raspberry Pi application</td>
</tr>
<tr>
<td>faust2octave (f2o)</td>
<td>sndfile.cpp, bench.cpp, octave.cpp</td>
<td>sound file transformation command speed benchmark output signals to Octave (matlab)</td>
</tr>
</tbody>
</table>
# FAUST Plugin Architectures Supported

<table>
<thead>
<tr>
<th>Shell Script</th>
<th>Arch. File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>faust2ladspa</td>
<td>ladspa.cpp</td>
<td>Linux LADSPA effect plugin</td>
</tr>
<tr>
<td>faust2dssi</td>
<td>dssi.cpp</td>
<td>Linux synth plugin, extending LADSPAA</td>
</tr>
<tr>
<td>faust2lv2</td>
<td>lv2.cpp</td>
<td>Linux synth plugin, newer lv2 type</td>
</tr>
<tr>
<td>faust2lv2-synth</td>
<td>lv2synth.cpp</td>
<td>Linux synth plugin, newer lv2 type</td>
</tr>
<tr>
<td>faust2vst</td>
<td>vst2p4.cpp</td>
<td>VST 2.4 plugin</td>
</tr>
<tr>
<td>faust2vsti</td>
<td>vsti-poly.cpp</td>
<td>VSTi instrument</td>
</tr>
<tr>
<td><strong>faust2au (f2au)</strong></td>
<td>au-effect.cpp</td>
<td>Apple Audio Unit (AU) effect plugin</td>
</tr>
<tr>
<td>faust2au</td>
<td>au-instrument.cpp</td>
<td>Apple AU instrument plugin</td>
</tr>
<tr>
<td>faust2max</td>
<td>max-msp.cpp</td>
<td>Max/MSP external</td>
</tr>
<tr>
<td>faust2puredata</td>
<td>puredata.cpp</td>
<td>PD external</td>
</tr>
<tr>
<td>faust2supercollider</td>
<td>supercollider.cpp</td>
<td>SuperCollider Unit Generator</td>
</tr>
<tr>
<td>faust2q</td>
<td>q.cpp</td>
<td>Snd-RT music programming language</td>
</tr>
<tr>
<td>faust2csound</td>
<td>csound.cpp</td>
<td>Q language plugin</td>
</tr>
<tr>
<td>faust2alchemy</td>
<td>alchemy-as.cpp</td>
<td>CSOUND opcode</td>
</tr>
<tr>
<td>faust2q</td>
<td>q.cpp</td>
<td>Flash/ActionScript plugin (for Web browsers)</td>
</tr>
</tbody>
</table>
Getting Started with FAUST

- Faust Website: [http://faust.grame.fr/](http://faust.grame.fr/)  
  (Click on “Online Examples” and type in some FAUST code!)

- Faust Intro:  
  [https://ccrma.stanford.edu/~jos/aspf/](https://ccrma.stanford.edu/~jos/aspf/)  
  (Google Search: “Audio Signal Processing in Faust”)

- **FAUST** is Free Open Source Software (FOSS) for Mac OS, Linux, and Windows
Ongoing Research
**Synth-A-Modeler Block-Diagram to FAUST Translator**

- **FAUST** is excellent for specifying platform-independent signal-processing block diagrams
- For *physical models* a higher-level front-end is helpful
- Physical objects need *bidirectional* connections
- Linux Audio Conference 2012 (LAC-12) Paper:
  
  [http://lac.linuxaudio.org/2012/papers/34.pdf](http://lac.linuxaudio.org/2012/papers/34.pdf)


- GPL Free Software:
  
  [git clone https://github.com/eberdahl/SaM.git](https://github.com/eberdahl/SaM.git)
2D Bridge Modeling for Bowed Strings

Scattering Delay Networks (SDN)

“Scattering Delay Networks” by Enzo De Sena, Hüseyin Hacihabiboğlu, Zoran Cvetković, and Julius O. Smith III