Building Tools For Musicians: #6

Another Random Menagerie

Today

- Matthias Herder – SpektralFilter
- Plugins #1

Plugins

- Picking a plugin API (or not)
  - Plugin basics
  - Plugin details

Picking an API

- VST
- AudioUnit
- DirectX
- TDM
- RTAS
- MAS
- LADSPA
- LV2
- DSSI

Common Elements of Plugin APIs

- A “process” callback where the real work is done
- Instantiation
- De-instantiation
- Set parameter values
- Start/Suspend/Resume/Stop
- Set sample rate
- Set block size

Less essential elements

- Parameter name
- Parameter ranges, units, step sizes etc.
- Parameter default values
- Label for current parameter value
- Plugin latency
- Name, author, unique ID

Instantiation

- Plugins are dynamically loaded shared objects/libraries (“DLL”)
- Host loads the plugin code using OS facilities
- Host checks if a known function name exists
- If so, calls it get to a description of what is in the file/plugin
- Description contains:
- user-relevant data (plugin name/id),
- host relevant data (number and ID's of parameters
- functions to be called at certain time

Instantiation 2

- Host looks in the plugin descriptor to find the instantiation function
- Calls it ... gets back a live, working plugin
- When done with the plugin, passes it to an de-de-instantiation/cleanup function found in the descriptor
- Note: descriptor is independent of any particular plugin *instance*
- Host might (or might not) unload the plugin file using OS facilities
- On OS X, plugin discovery & instantiation are handled by system libraries

Meta-izing Plugin Development

- Write more than one plugin – discover most plugin code is common to them all
  - Why write this over and over?
- Steve Harris (swh) and his 190 LADSPA plugins

SWH Plugin Code

- Write plugin as XML file
- Stick necessary DSP code in sections of XML
- Process before compiling to generate actual C

```
<?xml version="1.0"?>
<!DOCTYPE ladspa SYSTEM "ladspa-swh.dtd">
<?xml-stylesheet href="ladspa.css" type="text/css"?>

<ladspa>
  <global>
    <meta name="maker" value="Steve Harris &lt;steve@plugin.org.uk&gt;"/>
    <meta name="copyright" value="GPL"/>
    <meta name="properties" value="HARD_RT_CAPABLE"/>
  </global>

  <plugin label="inv" id="1429" class="UtilityPlugin">
    <name>Inverter</name>
    <p>A utility plugin that inverts the signal, also (wrongly) known as a 180
degree phase shift.</p>

    <callback event="run"><![CDATA[
      unsigned long pos;

      for (pos = 0; pos < sample_count; pos++) {
        buffer_write(output[pos], -input[pos]);
      }
    ]]></callback>

    <port label="input" dir="input" type="audio">
      <name>Input</name>
    </port>

    <port label="output" dir="output" type="audio">
      <name>Output</name>
    </port>
  </plugin>
</ladspa>
```

SWH builds a plugin

```
# Rule to build .c files from XML source
%.c:%.xml
	./makestub.pl $*.xml > $*.c
```

A step or two further

- Why write the plugin in C at all?
- Not a particularly expressive language for DSP operations
- All that common code in every plugin
- What if you want to generate plugins for different APIs/platforms?

Enter: FAUST

- A programming language for real-time digital signal processing
- Compiled into C++
- Can be included as-is in other programs OR
- Wrapped in standard code to create a variety of plugins or even standalone apps
- Standalone: ALSA/JACK/libsndfile
- Plugin: VST/LADSPA/Max/PureData/SuperCollider/Q

Faust Concepts

A Faust program describes a signal processor by combining primitive operations on signals (like +, −, ∗, /, √, sin, cos, . . .) using an algebra of high level composition operators

f ~ g recursive composition
f , g parallel composition
f : g sequential composition
f <: g split composition
f :> g merge composition

Faust Example

```
random = +(12345) ~ *(1103515245);
noise   = random/2147483647.0;
process = noise * checkbox("generate");
```

What To Do With Faust

- Generate self-contained C++ code:

```
faust noise.dsp
```

- Generate a JACK client with a GTK app:

```
faust -a jack-gtk.app -o noise.cpp  noise.dsp
```

- Generate a VST plugin:

```
faust -a vst.app -o noise.cpp noise.dsp
```

- Generate SuperCollider ugen:

```
faust -a supercollider.cpp -o noise.cpp
noise.dsp
```

Karplus-Strong in Faust

```
random = +(12345) ~ *(1103515245);
noise  = random/2147483647.0;
impulse(x) = x - mem(x) : >(0.0);
decay(n,x) = x - (x>0.0)/n;
release(n) = + ~ decay(n);
trigger(n) = button("play") : impulse : release(n) : >(0.0);
index(n) = &(n-1) ~ +(1);
delay(n,d,x)= rwtable( n, 0.0, index(n),x, (index(n)-
int(d))&(n-1));
average(x) = (x+mem(x))/2;
resonator(d,a) = (+ : delay(4096, d-1)) ~ (average : *(1.0-
a));

dur = hslider("duration",128,2,512,1);
att = hslider("attenuation",0.1,0,1,0.01);

process = noise : *(trigger(dur)) : resonator(dur,att);
```

```
class mydsp : public dsp
{
 private:
  int     R0_0;
  float   fcheckbox0;
 public:
  virtual int getNumInputs() {
    return 0;
  }
  virtual int getNumOutputs() {
    return 1;
  }
  virtual void init(int samplingFreq) {
    fSamplingFreq = samplingFreq;
    R0_0 = 0;
    fcheckbox0 = 0.0;
  }
  virtual void buildUserInterface(UI* ui)
{
    ui->openVerticalBox("faust");
    ui->addCheckButton("generate",

&fcheckbox0);
    ui->closeBox();
  }
  virtual void compute (int count,
           float** input, float** output)
{
    float* output0; output0 = output[0];
    float ftemp0 =
4.656613e-10f*fcheckbox0;
    for (int i=0; i<count; i++) {
      R0_0 = (12345 + (1103515245 *
R0_0));
      output0[i] = (ftemp0 * R0_0);
    }
  }
```

Benefits of Faust

- Automatically generated GUIs
- Portable to many plugin platforms
- Often faster then writing in C/C++

How can it be faster?

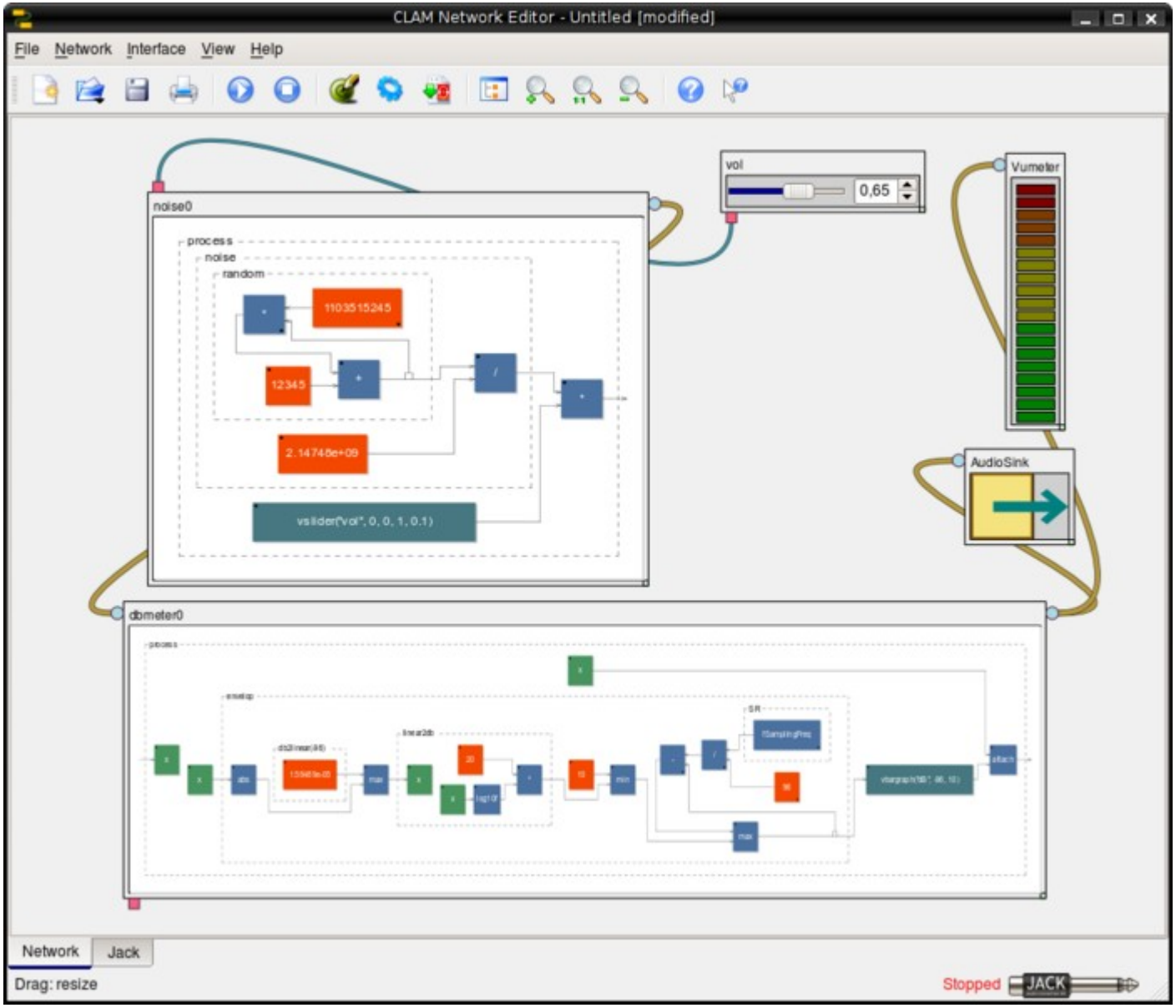
- Programmers like certains kinds of constructs
  - Easier to write them
  - Easier to read them
  - Easier to reason about them
  - Faust compiler doesn't care!
  - Dataflow-based optimization

Another Faust Feature

- Generate SVG diagrams of the data graph

```
faust -svg noise.dsp
```

CLAM & Faust


CLAM Network Editor - Untitled [modified]
File Network Interface View Help
vol
0,65
Vumeter
noise0
process
noise
random
1103515245
12345
2.14748e+09
vslider("vol", 0, 0, 1, 0.1)
AudioSink
dbmeter0
Network
Jack
Drag: resize
Stopped
JACK