Faust audio DSP language in the Web

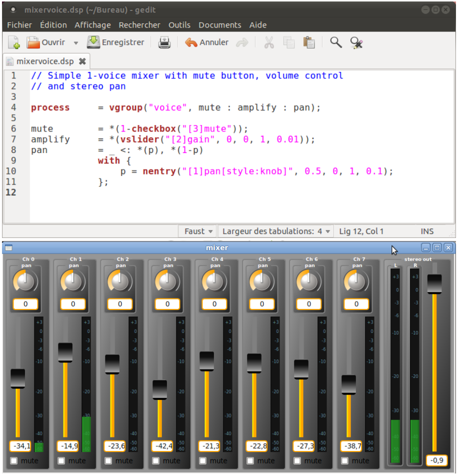
Stephane Letz, Sarah Denoux, Yann Orlarey, Dominique Fober
GRAME
Centre national de création musicale

Linux Audio Conference, Mainz, 2015/04/10

Faust

Speed up audio application and plug-in development

Faust offers an abstract high-level notation to describe DSP algorithms in a concise and effective manner.



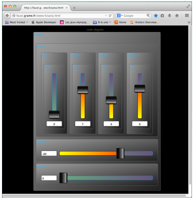
mixervoice.dsp (~/Bureau) - gedit
Fichier Édition Affichage Rechercher Outils Documents Aide
Ouvrir Enregistrer Annuler
mixervoice.dsp

```
// Simple 1-voice mixer with mute button, volume control
// and stereo pan

process     = vgroup("voice", mute : amplify : pan);

mute        = *(1-checkbox("[3]mute"));
amplify     = *(vslider("[2]gain", 0, 0, 1, 0.01));
pan         = _ <: *(p), *(1-p)
            with {
                p = nentry("[1]pan[style:knob]", 0.5, 0, 1, 0.1);
            };
```

Faust

Develop once, deploy everywhere

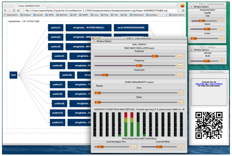
Audio applications designers have to deploy their work on a variety of platforms (Linux, OSX, Windows, Android, iOS, embedded devices, etc). One of Faust strong ideas is to write the DSP once and easily deploy it on a wide number of systems.



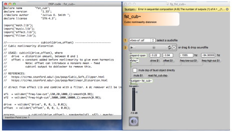
Faust

Make Faust compilation technology widely usable

Faust compilation technology is accessible using the **online compiler**, the **embedded compiler library version**, or the **FaustWeb remote compilation API** that produces various target binaries.

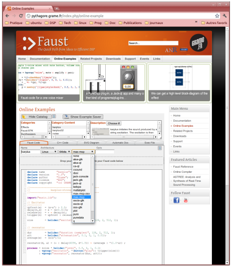


FaustLive with FaustWeb access



```
declare name        "fst_cub";
declare version        "1.33";
//declare author       "Julius O. Smith ";
declare license        "STK-4.3";

import("math.lib");
import("music.lib");
import("effect.lib");
import("filter.lib");

//-------------------- cubicnl(drive,offset) ----------------------
// Cubic nonlinearity distortion
//
// USAGE: cubicnl(drive,offset), where
//   drive  = distortion amount, between 0 and 1
//   offset = constant added before nonlinearity to give even harmonics
//            Note: offset can introduce a nonzero mean - feed
//            cubicnl output to dcblocker to remove this.
//
// REFERENCES:
//   https://ccrma.stanford.edu/~jos/pasp/Cubic_Soft_Clipper.html
//   https://ccrma.stanford.edu/~jos/pasp/Nonlinear_Distortion.html

// direct from effect lib and combine with a filter. A dc remover will be in

sf1  = vslider("freq-low-cut",130,20,1000,1):smooth(0.99);
sf2  = vslider("freq-high-cut",5000,1000,10000,1):smooth(0.99);

drive  = vslider("drive", 0, 0, 1, 0.01);
offset  = vslider("offset", 0, 0, 1, 0.01);

process        = cubicnl(drive,offset) : speakerbp(sf1, sf2) : mvecho;
```

Faust in Max/MSP (faustgen using libfaust.so)



Online compiler

Audio on the WEB

WEB technologies like **asm.js**, **Web Audio API** or **Web components** aim to change the way we design, publish and share musical applications. Using these technologies *procedural content* can now be shared and combined as easily as *multimedia content*! Grame offers several Web technologies:

- **libfaust.js + asm.js target (emscripten + Faust backend)**: embeddable JavaScript/asm.js Faust compiler;
- **FaustWeb**: remote multi-target compilation API;
- **Faust Playground**: simplifying Faust program design.

Audio on the WEB

Targeting the Web Audio API (1)

The **Web Audio API** is a high-level JavaScript API for processing and generating audio in Web applications:

- native optimized C++/assembly nodes;
- JavaScript/asm.js **ScriptProcessor** nodes;
- connected to create an audio generating/processing graph.

Audio on the WEB

Targeting the Web Audio API (2)

How to generate **ScriptProcessor** nodes ?

- they can be "manually written" in pure JavaScript;
- or in asm.js for better performance (but this is difficult...);
- or automatically generated from DSP code already written in C/C++... (emscripten);
- or **automatically generated from a Domain-Specific Language**.

Audio on the WEB

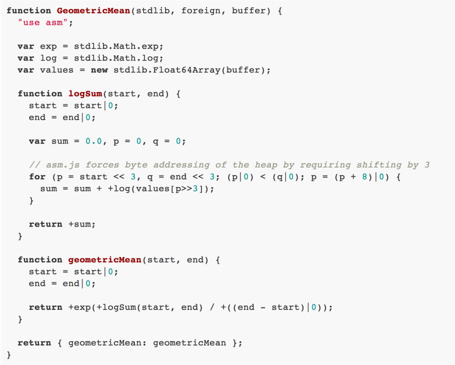
Asm.js code generation (1)

Asm.js is developed by Mozilla along with **Emscripten**:

- **asm.js**: an extremely restricted subset of JavaScript that provides only strictly-typed integers, floats, arithmetic, function calls, and heap accesses (using typed arrays);
- **asm.js** variables, computation, return values types are annotated;
- **asm.js** can easily be optimized;
- future extensions like **SIMD.js** (vectorized types in JavaScript).

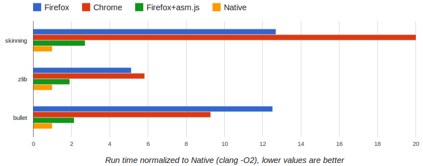
Audio on the WEB

Asm.js code generation (2)



```
function GeometricMean(stdlib, foreign, buffer) {
  "use asm";

  var exp = stdlib.Math.exp;
  var log = stdlib.Math.log;
  var values = new stdlib.Float64Array(buffer);

  function logSum(start, end) {
    start = start|0;
    end = end|0;

    var sum = 0.0, p = 0, q = 0;

    // asm.js forces byte addressing of the heap by requiring shifting by 3
    for (p = start << 3, q = end << 3; (p|0) < (q|0); p = (p + 8)|0) {
      sum = sum + +log(values[p>>3]);
    }

    return +sum;
  }

  function geometricMean(start, end) {
    start = start|0;
    end = end|0;

    return +exp(+logSum(start, end) / +((end - start)|0));
  }

  return { geometricMean: geometricMean };
}
```

Example of asm.js module



asm.js benchmark (2 to 3 times slower than native code...)

Audio on the WEB

Asm.js code generation (3)

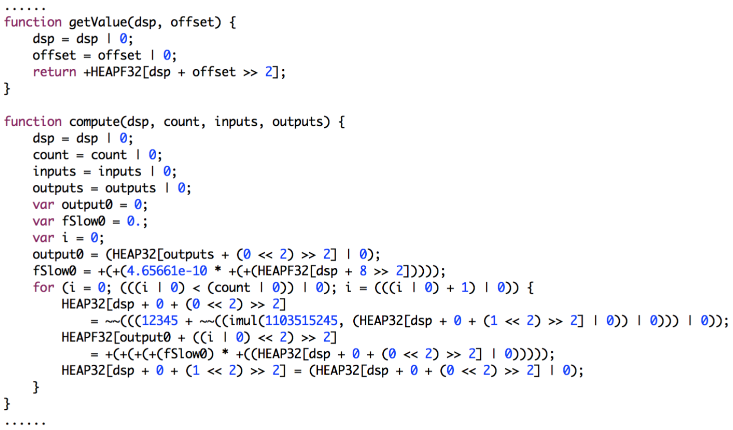
Generating asm.js with Emcripten:

- Emscripten C/C++ to JavaScript (asm.js) compiler developed by Mozilla starting in 2011;
- Eases the porting of huge C/C++ codebases on the Web.

Audio on the WEB

Asm.js code generation (4)

Asm.js backend in Faust compiler: produces the asm.js module + some pure JavaScript helper functions:

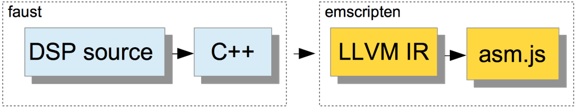


```
......
function getValue(dsp, offset) {
    dsp = dsp | 0;
    offset = offset | 0;
    return +HEAPF32[dsp + offset >> 2];
}

function compute(dsp, count, inputs, outputs) {
    dsp = dsp | 0;
    count = count | 0;
    inputs = inputs | 0;
    outputs = outputs | 0;
    var output0 = 0;
    var fSlow0 = 0.;
    var i = 0;
    output0 = (HEAP32[outputs + (0 << 2) >> 2] | 0);
    fSlow0 = +(+(4.65661e-10 * +(+(HEAPF32[dsp + 8 >> 2]))));
    for (i = 0; (((i | 0) < (count | 0)) | 0); i = (((i | 0) + 1) | 0)) {
        HEAP32[dsp + 0 + (0 << 2) >> 2]
            = ~~(((12345 + ~~((imul(1103515245, (HEAP32[dsp + 0 + (1 << 2) >> 2] | 0)) | 0))) | 0));
        HEAPF32[output0 + ((i | 0) << 2) >> 2]
            = +(+(+(+(fSlow0) * +((HEAP32[dsp + 0 + (0 << 2) >> 2] | 0)))));
        HEAP32[dsp + 0 + (1 << 2) >> 2] = (HEAP32[dsp + 0 + (0 << 2) >> 2] | 0);
    }
}
......
```

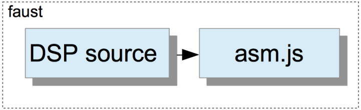
Audio on the WEB

Asm.js code generation (5)

Static compilation chain (Faust DSP to asm.js) allows to generate self-contained HTML pages.



- using emscripten as an intermediate step:



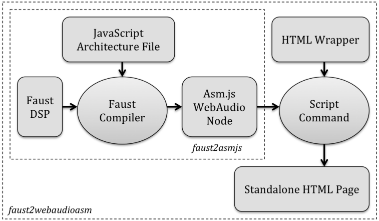
- or using direct asm.js code generation:

Audio on the WEB

JavaScript compilation: asm.js generation

Static compilation chain scripts:

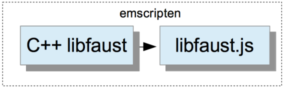
- takes Faust DSP, compiles it to asm.js, wraps it with additional JavaScript code to obtain a fully functional Web Audio node;
- wraps the Web Audio node in a HTML template to obtain a self-contained DSP node in the page.



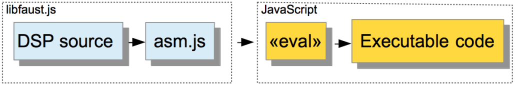
Audio on the WEB

JavaScript compilation: asm.js generation

Dynamic compilation chain (libfaust.js + asm.js backend) allows to embed the complete compilation chain in the browser:



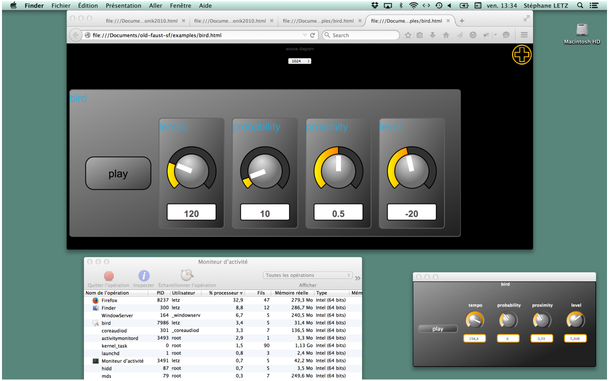
- first compile C++ **libfaust** for the Web (**libfaust.js**)



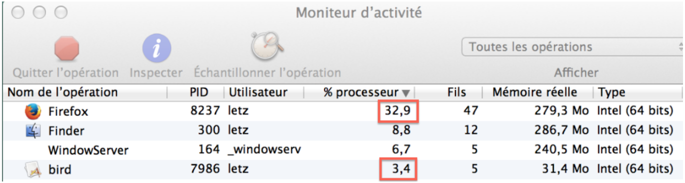
- compilation of an asm.js module happens at parse time of the source code. If parse time is triggered with 'eval' then **dynamic compilation occurs**.

Benchmark of a CPU light application

Bird ported on the Web


Finder Fichier Édition Présentation Aller Fenêtre Aide
ven. 13:34 Stéphane LETZ
file:///Docume...onik2010.html
file:///Docume...ples/bird.html
file:///Documents/old-faust-sf/examples/bird.html
Search
Macintosh HD
1024
bird
play
tempo
probability
proximity
level
120
10
0.5
-20
Moniteur d'activité
Quitter l'opération
Inspecter
Échantillonner l'opération
Toutes les opérations
Afficher
294,4
6
0,39
5,8dB

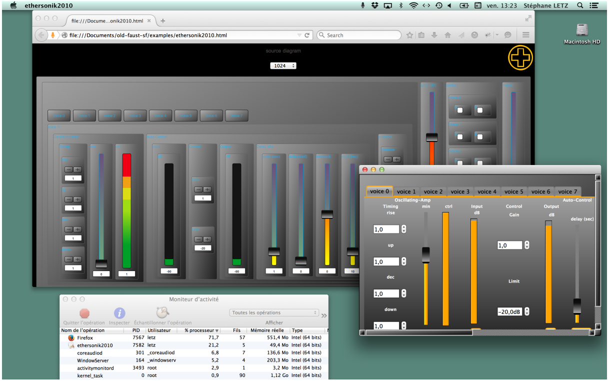
Nom de l'opération	PID	Utilisateur	% processeur ▼	Fils	Mémoire réelle	Type
Firefox	8237	letz	32,9	47	279,3 Mo	Intel (64 bits)
Finder	300	letz	8,8	12	286,7 Mo	Intel (64 bits)
WindowServer	164	_windowserv	6,7	5	240,5 Mo	Intel (64 bits)
bird	7986	letz	3,4	5	31,4 Mo	Intel (64 bits)
coreaudiod	301	_coreaudiod	3,3	7	136,5 Mo	Intel (64 bits)
activitymonitord	3493	root	2,9	1	3,3 Mo	Intel (64 bits)
kernel_task	0	root	1,5	90	1,13 Go	Intel (64 bits)
launchd	1	root	0,8	3	2,4 Mo	Intel (64 bits)
Moniteur d'activité	3491	letz	0,7	5	42,2 Mo	Intel (64 bits)
hidd	87	root	0,7	5	3,5 Mo	Intel (64 bits)
mds	79	root	0,3	7	249,6 Mo	Intel (64 bits)



Nom de l'opération	PID	Utilisateur	% processeur ▼	Fils	Mémoire réelle	Type
Firefox	8237	letz	32,9	47	279,3 Mo	Intel (64 bits)
Finder	300	letz	8,8	12	286,7 Mo	Intel (64 bits)
WindowServer	164	_windowserv	6,7	5	240,5 Mo	Intel (64 bits)
bird	7986	letz	3,4	5	31,4 Mo	Intel (64 bits)

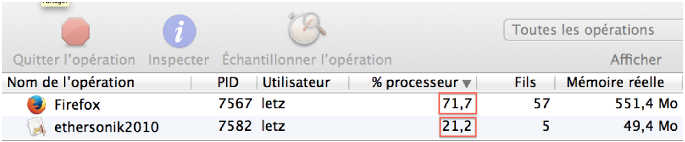
Benchmark of a CPU heavy application

Yann Orlarey's Ethersonik ported on the Web



file:///Documents/old-faust-sf/examples/ethersonik2010.html

Nom de l'opération	PID	Utilisateur	% processeur	Fils	Mémoire réelle	Type
Firefox	7567	letz	71,7	57	551,4 Mo	Intel (64 bits)
ethersonik2010	7582	letz	21,2	5	49,4 Mo	Intel (64 bits)
coreaudiod	301	_coreaudiod	6,8	7	136,6 Mo	Intel (64 bits)
WindowServer	164	_windowserv	5,2	4	203,3 Mo	Intel (64 bits)
activitymonitord	3493	root	2,9	1	3,2 Mo	Intel (64 bits)
kernel_task	0	root	0,9	90	1,12 Go	Intel (64 bits)



Nom de l'opération	PID	Utilisateur	% processeur	Fils	Mémoire réelle
Firefox	7567	letz	71,7	57	551,4 Mo
ethersonik2010	7582	letz	21,2	5	49,4 Mo

Demo

- faust2webaudioasm script
  - From harpsichord.dsp to harpsichord.html

- faust2asmjs
  - Harpsichord

    Thomas Cipierre & Laurent Pottier (Saint-Etienne, France)
  - foo-yc20

    Sampo Savolainen (Helsinki, Finland)

- libfaust.js
  - FaustPlayground: create Faust patches online

Conclusions and perpectives

- still some issues with the Web Audio API: implementation, performance CPU/latency (audio workers: moving the ScriptProcessor nodes in the audio thread);
- really usable for serious work? still to be proved...
- but at least already usable for deployment, distribution, teaching purposes...

GRAME – R&D

Softwares

Softwares developed in different research projects are freely available under GPL/LGPL licenses:

- Faust: `http://faust.grame.fr`:
  - Faust: `git.code.sf.net/p/faudiostream/code`
  - FaustLive: `git.code.sf.net/p/faudiostream/faustlive`
  - FaustWorks: `git.code.sf.net/p/faudiostream/faustworks`
  - FaustWeb: `git://git.code.sf.net/p/faudiostream/faustweb`

GRAME – R&D

Recent publications on Faust

- Denoux, Letz, Orlarey, Fober 2014: *FAUSTLIVE: Just-In-Time Faust Compiler... and much more*. LAC 2014.
- Denoux, Letz, Orlarey, Fober 2014: *FaustLive un compilateur à la volée pour Faust ... et bien plus encore*, Journées d'Informatique Musicale, Bourges.
- Brune de Chiffreville 2014: *Using Faust with Ros*. Rapport de Stage, GRAME.
- Denoux, Letz, Orlarey, Fober 2015: *Composing a web of audio applications*, WAC 2015, Paris.
- Letz, Denoux, Orlarey, Fober 2014: *Faust audio DSP language in the Web*, LAC 2015