Using Faust in Dplug

By: Ethan Reker

Why Faust?

- Generates highly optimized DSP code
- Can be exported to many backends
- Includes a large library of well-written dsp algorithms
- Writing DSP code is very fast and intuitive
- Functional programming rules

Converting Clipit Example to Faust code

https://github.com/abaga129/dplug/blob/clipit-faust/examples/clipit-faust/dsp/clipit.dsp

Parameters

We make use of Faust's UIMetadata to link Dplug Parameters.

We must make sure that the Faust's parameters are ordered the same as our Dplug client.

Faust

```
inputGain = ba.db2linear(hslider("[0]Input Gain[unit:%]", 0, -12, 12, 0.1));
clipAmount = (hslider("[1]Clip Amount[unit:%]", 0, 0, 100, 0.1) / 100) + 0.01;
outputGain = ba.db2linear(hslider("[2]Output Gain[unit:%]", 0, -12, 12, 0.1));
mixAmount = hslider("[3]Mix Amount[unit:%]", 100, 0, 100, 0.1) / 100;
mode = checkbox("[4]Mode");
bassBoost = hslider("[5]Bass Boost[unit:%]", 0, 0, 6, 0.01);
```

Dplug

```
enum : int
{
    paramInputGain,
    paramClip,
    paramOutputGain,
    paramMix,
    paramMode,
    paramBassBoost
}
```

Parameters Continued

Note that each UI control in Faust has [<integer>] at the beginning of it's name. This determines the order of controls. Our dplug.d architecture file uses this to sort the parameters.

The parameter label will be used to match Faust parameters to Dplug parameters when updating values.

Add Faust file to Dplug project

- Create a folder to hold your dsp file(s). For example: dsp

- Add the following to your dub.json (you may want to replace “clipit” with the name of your project)

```
	“preBuildCommands”: ["faust dsp/clipit.dsp -lang dlang -a dplug.d -cn ClipIt -vec -o src/clipit.d"],
	“versions”: ["faustoverride"],
```

Import the generated dsp code

In your main.d file. Add an import for the generated D file.

```
import clipit;
```

Then your Dplug Client will inherit from FaustClient rather than dplug.client.Client

```
final class ClipitClient : FaustClient
{
public:
nothrow:
@nogc:
```

Then lastly you may remove all of the overrides in your Client class other than “IGraphics createGraphics()”

Demo

Downsides

- Multirate not feasible with Faust (No upsampling/downsampling)
- Logarithmic Parameters have not been implemented in architecture file
- Adds more complexity (new language, added dependencies, etc)
- Can’t directly optimize code (at least not easily)

Questions?

Useful resources

Faust Documentation

https://faustdoc.grame.fr/