FAUST / DPLUG INTEGRATION

Ethan Reker

WHY USE FAUST?

- Large library of highly reusable DSP components
- Functional approach to DSP programming is more natural and enables rapid prototyping
- No need to deal with buffers
- Portable DSP code
- Compiler can vectorize the generated D code
- Well documented

DOWNSIDES

- Generated code is ugly and would be very difficult to debug (havent had a problem with this yet *fingers crossed*)
- Adds an extra dependency to the chain
- No multi-rate so upsampling/downsampling have to be done from Dplug then passed through the faust module at the upsampled rate.

MY WORKFLOW

- DSP design in faust web IDE
  - Determine parameters at this point and use FAUSTUI for rapid prototyping
- Create new GUI-less project or if adding to existing project disable GUI.
- Compile with faust and target dplug using architecture file

HOW DOES THE INTEGRATION WORK

- preBuildCommand in dub.json to invoke the faust compiler
- Faust generates D code based off of the C++ implementation
- This code is injected into an architecture file (optional)
- Dplug client inherits from generated client and implements GUI.

INITIALIZING THE FAUST MODULE

Dplug Client

```
this()
{
    buildFaustModule();
}

void buildFaustModule()
{
    _dsp = mallocNew!(FAUSTCLASS)();
    FaustParamAccess _faustUI = mallocNew!FaustParamAccess();
    _dsp.buildUserInterface(cast(UI*)(&_faustUI));
    _faustParams = _faustUI.readParams();
}
```

MAPPING PARAMETERS

```
override Parameter[] buildParameters()
{
    auto params = makeVec!Parameter();
    buildFaustModule();
    int faustParamIndexStart = 0;
    foreach(param; _faustParams)
    {
        if (param.isButton)
        {
            params ~= mallocNew!BoolParameter(faustParamIndexS
        // ... ommited other param types for brevity
    }
    return params.releaseData();
}
```

UPDATING PARAMETERS

```
void updateFaustParams()
{
    for(int paramIndex = 0; paramIndex < _faustParams.length;
    {
        auto dplugParam = params()[paramIndex];
        auto faustParam = _faustParams[paramIndex];
        if (cast(FloatParameter)dplugParam)
        {
            *(faustParam.val) = (cast(FloatParameter)dplugPara
        }
        ... //excluded others for brevity
    }
}
```

AUDIO PROCESSING

```
override void processAudio(const(float*)[] inputs, float*[]out
{
    int numInputs = cast(int)inputs.length;
    int numOutputs = cast(int)outputs.length;

    int minChan = numInputs > numOutputs ? numOutputs : numInp
    updateFaustParams();
    _dsp.compute(frames, cast(float*[])inputs, cast(float*[])o

    for (int chan = minChan; chan < numOutputs; ++chan)
        outputs[chan][0..frames] = 0;
}
```

AREAS THAT COULD BE IMPROVED

- The architecture file could really be improved to support multiple faust DSP modules per Dplug project
- Linking parameters is a bit messy and wastes a lot of CPU cycles each process callback.
- No midi support for the D backend currently

QUESTIONS?

USEFUL RESOURCES

Faust Syntax	https://faustdoc.grame.fr/manual/syntax/
Dplug Example	https://github.com/ctrecordings/dplug-faust-example
Faust Libraries	https://faustlibraries.grame.fr/
Faust Web IDE	https://faustide.grame.fr/