# COMP372/471 - Faust

***Release 1.0***

February 14, 2017

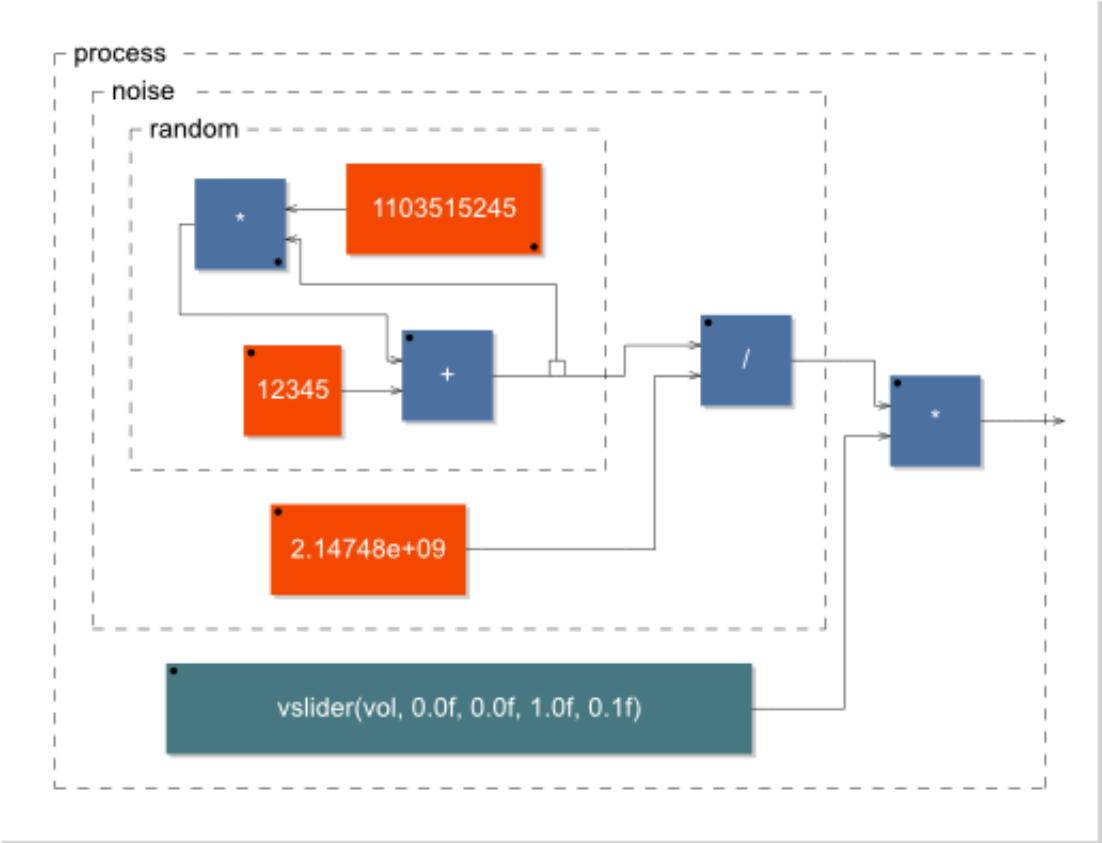
Contents

This is a presentation on the Faust programming language for Loyola University Chicago's COMP372/471 Programming Langauges course. It was created by Griffin Moe using the Sphinx documentation tool and is hosted on ReadTheDocs.

CHAPTER 1

# Introduction to Faust

- Faust - Functional Audio Stream
- Real-time signal processing and synthesis
- GPL licensed
- Developed by GRAME Centre National de Creation Musicale



CHAPTER 2

# Why Faust?

- Free and open source vs. MATLAB
- Specialized for audio streams vs. GNU Octave
- Can match or outperform native C++

## 2.1 Great operating system support

- Windows
- OSX
- Linux
- Android / iOS

## 2.2 Great platform support

- LADSPA
- Virtual Studio Technology / Audio Units
- SuperCollider
- Csound
- ...and many more!

CHAPTER 3

# The Language

- Purely functional language
- Faust compiler transpiles to C++ code
- Built-in UI widgets
- No run-time libraries required!
- Block-diagram syntax
- Intrinisically signal-based:
    1. Digital signals - discrete time functions
    2. Signal processor - Operate functions on signals (2nd order)
    3. Composition operators - Ties processors together (3rd order)

## 3.1 Signal Processor Examples

Generates silence:

```
process = 0;
```

Passes input signal to output, a cable:

```
process = _;
```

Downmixes stereo signal to mono:

```
process = +;
```

## 3.2 Processor Composition

We combine discrete processors using the following operators:

| Oper | Operator Function |
|---|---|
| f~g | Recursive composition |
| f,g | Parallel composition |
| f:g | Sequential composition |
| f<:g | Split composition |
| f:>g | Merge composition |

Similar to our second example in the section above, a stereo cable:

```
process = _,_;
```

Resursion used to create a one sample delay:

```
//Y(t) = X(t) + Y(t1)
process = + ~ _;
```

CHAPTER 4

# Examples

## 4.1 Noise Generator - Faust Code

```
declare name        "noise";
declare version     "1.0";
declare author      "Grame";
declare license     "BSD";
declare copyright   "(c) GRAME 2006";

// noise generator

random  = +(12345)~*(1103515245);
noise   = random/2147483647.0;

process = noise * vslider("vol", 0, 0, 1, 0.1);
```

## 4.2 Noise Generator - C++ Code

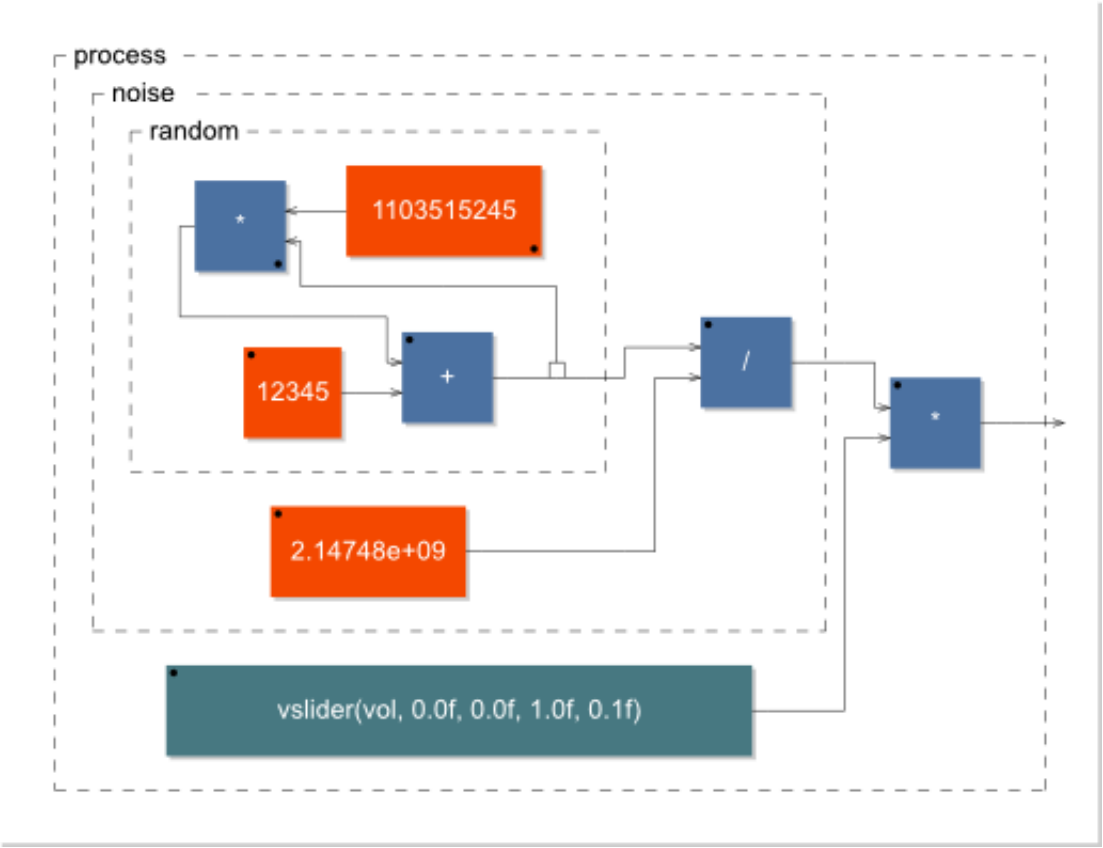
```
//-----------------------------------------------------
// author: "Grame"
// copyright: "(c) GRAME 2006"
// license: "BSD"
// name: "noise"
// version: "1.0"
//
// Code generated with Faust 2.0.a21 (http://faust.grame.fr)
//-----------------------------------------------------
#ifndef FAUSTFLOAT
#define FAUSTFLOAT float
#endif



#ifndef FAUSTCLASS
#define FAUSTCLASS mydsp
#endif

class mydsp : public dsp {

  private:
```

```
        int iRec0[2];
        FAUSTFLOAT fvslider0;
        int fSamplingFreq;

public:

        void static metadata(Meta* m) {
                m->declare("author", "Grame");
                m->declare("copyright", "(c) GRAME 2006");
                m->declare("license", "BSD");
                m->declare("name", "noise");
                m->declare("version", "1.0");
        }

        virtual int getNumInputs() {
                return 0;

        }
        virtual int getNumOutputs() {
                return 1;

        }
        virtual int getInputRate(int channel) {
                int rate;
                switch (channel) {
                        default: {
                                rate = -1;
                                break;
                        }

                }
                return rate;

        }
    virtual int getOutputRate(int channel) {
                int rate;
                switch (channel) {
                        case 0: {
                                rate = 1;
                                break;
                        }
                        default: {
                                rate = -1;
                                break;
                        }

                }
                return rate;

    }

    static void classInit(int samplingFreq) {

    }

    virtual void instanceInit(int samplingFreq) {
                fSamplingFreq = samplingFreq;
                fvslider0 = FAUSTFLOAT(0.);
```

```
                for (int i = 0; (i < 2); i = (i + 1)) {
                        iRec0[i] = 0;

                }

        }

        virtual void init(int samplingFreq) {
                classInit(samplingFreq);
                instanceInit(samplingFreq);
        }

        virtual void buildUserInterface(UI* interface) {
                interface->openVerticalBox("noise");
                interface->addVerticalSlider("vol", &fvslider0, 0.f, 0.f, 1.f, 0.1f);
                interface->closeBox();

        }

        virtual void compute(int count, FAUSTFLOAT** inputs, FAUSTFLOAT** outputs) {
                FAUSTFLOAT* output0 = outputs[0];
                float fSlow0 = (4.65661e-10f * float(fvslider0));
                for (int i = 0; (i < count); i = (i + 1)) {
                        iRec0[0] = (12345 + (1103515245 * iRec0[1]));
                        output0[i] = FAUSTFLOAT((fSlow0 * float(iRec0[0])));
                        iRec0[1] = iRec0[0];

                }

        }
};
```

## 4.3 Noise Generator - Signal Path Diagram

For more examples: http://faust.grame.fr/index.php/online-examples


process
noise
random
*
1103515245
12345
+
/
*
2.14748e+09
vslider(vol, 0.0f, 0.0f, 1.0f, 0.1f)

CHAPTER 5

# Evaluation

- **Readability (4/5)** - Better than other specification langauges
- **Productivity (4/5)** - No escaping C++...
- **Community (3/5)** - Around since 2002, lack of excitement
- **Major Projects:**
  - Guitarix
  - Faust -> FPGA
  - Recent Academic Papers
  - No commercial use (?)
- **Ecosystem (4/5)** - Neat tools available
- **Coolness (4/5)** - Excellent abstraction, C++ sucks...

CHAPTER 6

# Conclusion

- Great environment for test algorithms without C++
- Free and open source, Faust > Max or MATLAB
- Transpiles to C++
- Support for SuperCollider, Max, Pure Data, etc.
- Block diagram visualizations
- Generates VST and AU plug-ins!!!

CHAPTER 7

# Resources

## 7.1 Tutorials

- Julius O. Smith's (CCRMA) Faust Tutorial - The best tutorial available in my opinion, Smith has also published some excellent textbooks as well.
- GRAME Faust Tutorials - These are GRAME's tutorials on Faust, has more of a mathematical leaning than Smith's tutorial.
- Faust Reference - Who doesn't love a good manual?

## 7.2 DSP Resources

- Musimathics - Volume 2 - I absolutely recommend this book if you are interested in learning signal processing. It assumes very little of the reader, very easy to read.
- MusicDSP.org - Contains a list of code examples of various algorithms in a variety of langauges, including C/C++, MATLAB, and Delphi.
- The Scientist and Engineer's Guide to Digital Signal Processing - A free textbook covering DSP in fantastic detail. Fairly easy read as well, though more geared towards digital communications.

## 7.3 Faust Tools

- FauseWorks - A Faust IDE that displays Faust and transpiled C++ code, and automatically generates signal path block diagrams.
- Vim-Faust - An open source syntax-highlighting plugin for Vim, developed by yours truly!