Noise

Grame, Yghe

March 9, 2010

name	Noise
version	1.1
author	Grame, Yghe
license	BSD
copyright	(c)GRAME 2009

```
//-----------------------------------------------------------------
// Noise generator and demo file for the Faust math documentation
//-----------------------------------------------------------------

declare name      "Noise";
declare version   "1.1";
declare author    "Grame";
declare author    "Yghe";
declare license   "BSD";
declare copyright  "(c)GRAME 2009";
```

1 Presentation of the "noise.dsp" Faust program

This program describes a white noise generator with an interactive volume, using a random function.

1.1 The random function

```
random  = +(int(12345))~*(int(1103515245));
```

The `random` function describes a generator of random numbers, which equation follows. You should notice hereby the use of an integer arithmetic on 32 bits, relying on integer wrapping for big numbers.

1. Output signal y such that

$$y(t) = r_1(t)$$

2. Input signal (none)

3. Intermediate signal r_1 such that

$$r_1(t) = 12345 \oplus 1103515245 \odot r_1(t-1)$$

1.2 The noise function

```
noise  = (int(random))/(int(random+1));
```

The white noise then corresponds to:

1. Output signal y such that

$$y(t) = s_1(t)$$

2. Input signal (none)

3. Intermediate signal s_1 such that

$$s_1(t) = \mathrm{int}\,(r_1(t)) \oslash \mathrm{int}\,(1 \oplus r_1(t))$$

1.3 Just add a user interface element to play volume!

```
process = noise * vslider("Volume[style:knob]", 0, 0, 1, 0.1);
```

Endly, the sound level of this program is controlled by a user slider, which gives the following equation:

1. Output signal y such that

$$y(t) = u_{s1}(t) \cdot s_1(t)$$

2. Input signal (none)

3. User-interface input signal u_{s1} such that

$$\text{"Volume"} \quad u_{s1}(t) \in [\,0, 1\,] \quad (\text{default value} = 0)$$

2 Block-diagram schema of process

This process is illustrated on figure 1.

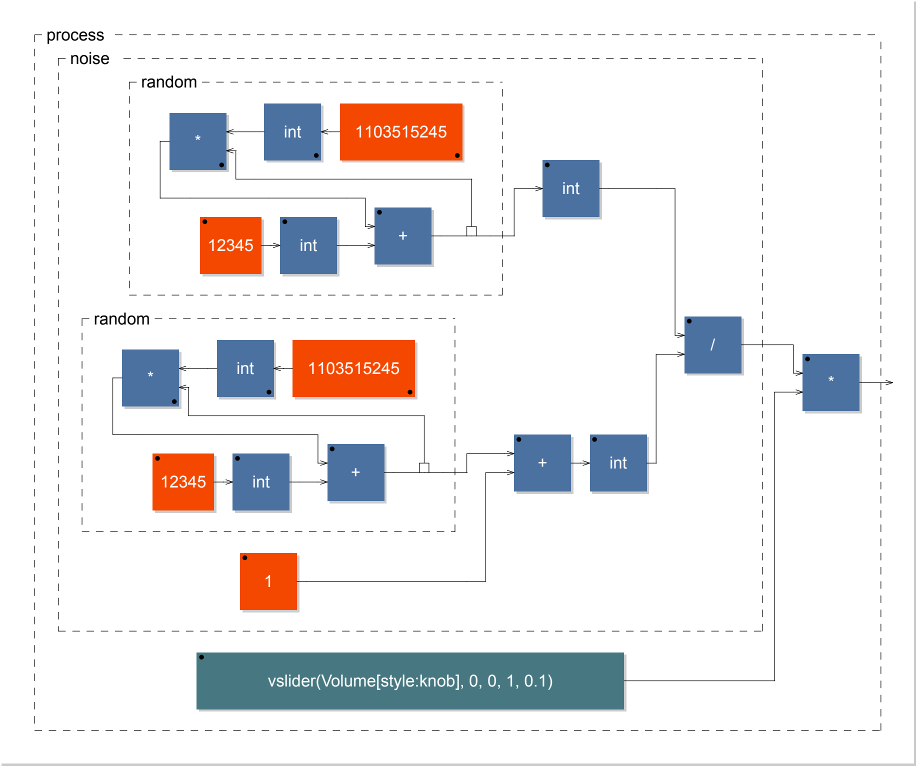

Figure 1: Block diagram of `process`

3 Notice of this documentation

You might be careful of certain information and naming conventions used in this documentation:

- This document was generated using Faust version 0.9.13 on March 09, 2010.
- The value of a Faust program is the result of applying the signal transformer denoted by the expression to which the `process` identifier is bound to input signals, running at the f_S sampling frequency.
- Faust (*Functional Audio Stream*) is a functional programming language designed for synchronous real-time signal processing and synthesis applications. A Faust program is a set of bindings of identifiers to expressions that denote signal transformers. A signal s in S is a function mapping[1] times $t \in \mathbb{Z}$ to values $s(t) \in \mathbb{R}$, while a signal transformer is a function

---

[1] Faust assumes that $\forall\, s \in S, \forall\, t \in \mathbb{Z}, s(t) = 0$ when $t < 0$.

from S^n to S^m, where $n, m \in \mathbb{N}$. See the Faust manual for additional information (http://faust.grame.fr).

- Every mathematical formula derived from a Faust expression is assumed, in this document, to having been normalized (in an implementation-dependent manner) by the Faust compiler.
- A block diagram is a graphical representation of the Faust binding of an identifier I to an expression E; each graph is put in a box labeled by I. Subexpressions of E are recursively displayed as long as the whole picture fits in one page.
- $\forall\, x \in \mathbb{R}$,

$$\mathrm{int}(x) = \begin{cases} \lfloor x \rfloor & \text{if } x > 0 \\ \lceil x \rceil & \text{if } x < 0 \\ 0 & \text{if } x = 0 \end{cases} .$$

- This document uses the following integer operations:

operation	*name*	*semantics*
$i \oplus j$	integer addition	$\mathrm{normalize}(i+j)$, in $\mathbb{Z}$
$i \odot j$	integer multiplication	$\mathrm{normalize}(i \cdot j)$, in $\mathbb{Z}$
$i \oslash j$	integer division	$\mathrm{normalize}(\mathrm{int}(i/j))$, in $\mathbb{Q}$

Integer operations in Faust are inspired by the semantics of operations on the n-bit two's complement representation of integer numbers; they are internal composition laws on the subset $[\,-2^{n-1}, 2^{n-1}-1\,]$ of $\mathbb{Z}$, with $n = 32$. For any integer binary operation $\times$ on $\mathbb{Z}$, the $\otimes$ operation is defined as: $i \otimes j = \mathrm{normalize}(i \times j)$, with

$$\mathrm{normalize}(i) = i - N \cdot \mathrm{sign}(i) \cdot \left\lfloor \frac{|i| + N/2 + (\mathrm{sign}(i)-1)/2}{N} \right\rfloor ,$$

where $N = 2^n$ and $\mathrm{sign}(i) = 0$ if $i = 0$ and $i/|i|$ otherwise. Unary integer operations are defined likewise.

- The `noisemetadata-mdoc/` directory may also include the following subdirectories:
  - `cpp/` for Faust compiled code;
  - `pdf/` which contains this document;
  - `src/` for all Faust sources used (even libraries);
  - `svg/` for block diagrams, encoded using the Scalable Vector Graphics format (http://www.w3.org/Graphics/SVG/);
  - `tex/` for the LaTeX source of this document.

4 Listing of the input code

The following listing shows the input Faust code, parsed to compile this mathematical documentation.

Listing 1: `noisemetadata.dsp`

```
//----------------------------------------------------------------
// Noise generator and demo file for the Faust math documentation
//----------------------------------------------------------------

declare name      "Noise";
declare version   "1.1";
declare author    "Grame";
declare author    "Yghe";
declare license   "BSD";
declare copyright  "(c)GRAME 2009";


random = +(int(12345))~*(int(1103515245));


noise  = (int(random))/(int(random+1));


process = noise * vslider("Volume[style:knob]", 0, 0, 1, 0.1);
```