Modal Spring Reverb in Faust - 20.04.21

Implementation of a *crappy* physical springreverb model in Faust using modal synthesis.

Projekt in den Kursen Audiotechnik und DigitaleAudiosignalverarbeitung im WS20/21 und SS21 an /der BHT

Block Diagram of possible Effect Unit

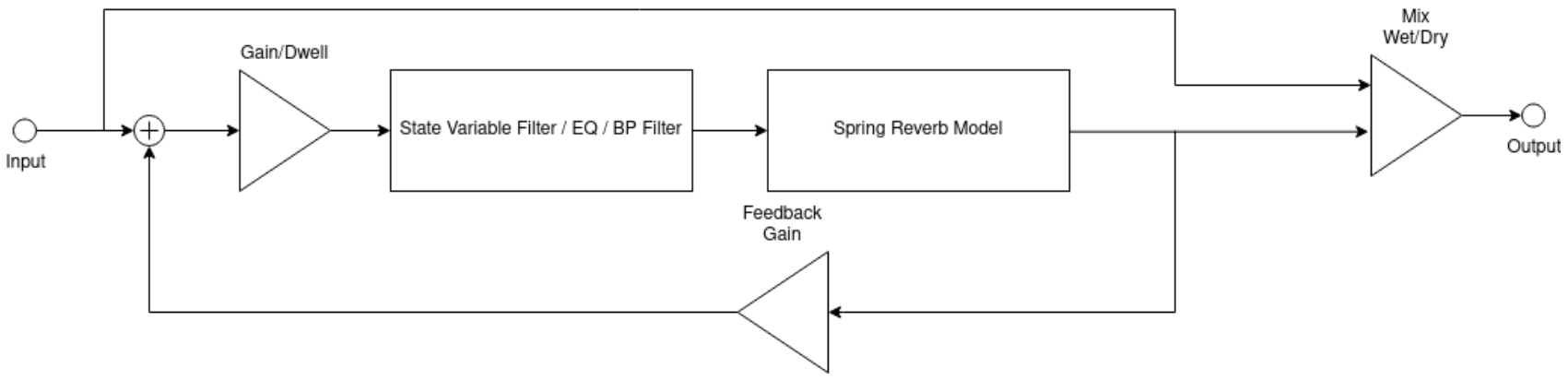

Figure 1: Blockdiagram

Notes taken during Faust Kadenze Course

Project Plan

KW	Date	Task
14	06.04	Learning Faust
15	13.04	Learning Faust
16	20.04	Learning Faust
17	27.04	Improve Model
18	04.05	Improve Model
19	11.05	Implementation of Model
20	18.05	Implementation of Model
21	25.05	Create Presentation
22	01.06	Project Presentation

Declarations

```
declare name "Spring Reverb";
declare version "0.1";
declare author "Jan Abel";

declare interface "SmartKeyboard{
    'Number of Keyboards' : '1',
    ...
}";
```

Enviroments Grouping

```
myconst = environment{
    v1 = 0.2;
    v2 = 0.5;
};

process = myconst.v1 +  myconst.v2;
```

User Defined Functions

Example: Dry-Wet Function

```
echo(d,f) = + ~ (@(d) : *(f));
pingpong(d,f) = echo(2*d,f) <: _,@(d);

// High Order Function
// Function of a Function
drywet(fx) = _ <: _, fx : *(1-w) , *(w) :> _
    with {
        w = vslider("dry-wet[style:knob]", 0.5, 0, 1, 0.01)
    };

process = button("play") : pm.djembe(60, 0.3, 0.4,1) :drywe
```

Recursive Functions

```
duplicate(1,x) = x;
duplicate(n,x) = x, duplicate(n-1,x)

count((x, xs)) = 1 + count(xs);
```

Signal Generators

```
process = 1;
```

$$y(t>=0) = 1$$

```
process = os.osc(10);
```

$$y(t) = \sin(2\ \pi\ 10)$$

Composition Operation

Split(priority 1)

`A <: B`

Merge(priority 1)

`A :> B`

Sequentital(priority 2)

`A : B`

Parallel(priority 3)

`A , B`
$y(t) = A(t), B(t)$

Recursion(priority 4)

`A ~ B`

Identiy Function (“Wire”)

Arithmetic Operation

+, -, *, /, %, ^

3 Diffrent Types of Notation

Core Syntax

```
process = _, 0.5 : *;
```

Infix Notation

```
process = _*0.5;
```

Prefix Notation

```
process = *(_,0.5);
```

Comparision Operation

“>”, “>=”, “==”, “!=”, “<”, “<=”
y(t) = 1, when x0(t) operator x1(t), 0 otherwise

Bitwise Operation

&, *|*, *xor*, *«*, *»*

Trigonomic Operation

acos, *asin*, *atan*, *atan2*, *cos*, *sin*, *tan*

Example: Sinewave Osciallator

```
import("stdfaust.lib");
// ma.SR - Current Sampling Rate
phasor(f) = f/ ma.SR : (+,1:fmod)~_;
process = phasor(440) * 2 * ma.PI : sin;
```

Log. and Exp. Functions

exp, *log*, *log10*, *pow*, *sqrt*

Other Function

Absolute, Minimal, Maximal > abs, min, max

floor, ceil of an Float > floor, ceil

Floating Point modulo, Float remainder > fmod, remainder

Closest Int of given float > rint

Example: Simple Distortion (Hard Clipping)

```
import("stdfaust.lib");
gain = hslider("gain",0,0,1,0.01);
process = *(100) : min(1) : max (-1) : *(gain);
```

Selectors and Casting

Switch between signals > select 2, select3

Convert samples to > int, float

Example: Selection of 2 with button

```
import("stdfaust.lib");
gain = hslider("gain", 0.1, 0, 1, 0.01);
process = button("440/880Hz"), os.osc(440), os.osc(880) : s
```

Example Selection of 3 with nentry

```
import("stdfaust.lib");
gain = hslider("gain", 0.1, 0, 1, 0.01);
process = nentry("Selector", 0,0,3,1), os.osc(440), os.osc
```

Delaylines

1-Sample Delay

mem

Variable number Sample Delay

@

Example: 1-Second Delay

process = 1, ma.SR : @;

Example: Dirac Impulse

```
import("stdfaust.lib");
dirac = 1-1';
process = dirac;
```

Tables

Read-only table

rdtable

Example: Dirac Impulse every 4096 Samples

```
import("stdfaust.lib");
dirac = 1-1';
phase = 1 : +~_ : %(4096);
process = 4096, dirac, phase : rdtable;
```

Read-Write Table

rwtable

User Interface

Vertical slider

```
vslider("name", std_value, min_value, max_value, stepsize);
```

Knob

```
vslider("name[style:knob]", std_value, min_value, max_value
```

Horizontal slider

```
hslider("name", std_value, min_value, max_value, stepsize);
```

Button

```
button("name");
```

nentry

```
nentry("name", std_value, min_value, max_value, stepsize)
```

checkbox

```
checkbox("name")
```