# Realtime music programming with Snd-Rt.

Kjetil Matheussen, NOTAM.

•

• Making a sound
• Making a GUI
• Sample by sample processing
• Using a CLM instrument
• Using Common Music
• Using Pure Data
• Lisp macros and meta programming
• Using Faust?

# What is Snd-Rt?

- Snd-Rt is an environment for doing realtime music programming.
- Snd-Rt lives inside the Snd sound editor. Snd provides CLM, Guile, and a lot more such as lots of example instruments and synthesis routines. Snd contains approx. 590.000 LOC!
- Snd-Rt consists of a sound engine and a a compiler.
- The engine is hard realtime safe, has frame-accurate scheduling and timing, etc.
- The compiler generates hard realtime safe code.

# More or less similar projects 1/3

- Faust:
-
- 1. Provides sample by sample programming, but its just a compiler, not an environment.
- 2. Faust is a pure functional programming language, while Snd-Rt is mostly an imperative programming language.
- 3. Faust generates code which is about 2-10 times faster than Snd-rt.

# More or less similar projects 2/3

- SuperCollider: Quite similar, but does not support sample by sample programming. This means that Supercollider is usually much faster, but that you can't do everything with it.

# More or less similar projects 3/3

- CLM running in Common Lisp, Scheme, Forth or Ruby. Does not have a scheduler and does not have support for hard realtime.
- 
- Snd-Rt provides a scheduler and hard realtime support to CLM.

# Example: Making a sound

- (**<rt-out>** (oscil))

# Example: Scheduling

```
(begin
  (<rt-out> :dur 1    3
                (oscil :frequency 400))
  (<rt-out> :dur 2.5 4
                (oscil :frequency 500)))
```

# Example: Sample by sample processing

```
(define phase 0)

(define freq (hz->radians 440))

(<rt-play>  (lambda ()
                    (out (sin phase))
                    (inc! phase freq)))
```

# Example: GUI

```
(<rt-out> (* (<slider> “Amp” 0 0.5 1)
             (<checkbutton> “On/Off” #t)
             (oscil)))

(<rt-out> (* (<slider> “Amp” 0 0.5 1)
             (<checkbutton> “On/Off” #t)
             (oscil :frequency 0
                    (hz->radians (<slider> “Freq”
                                           20 200 4000
                                           :log #t))))
```

# Example: Ladspa

```
(define am (make-ladspa “am_pitchshift_1433”
                        “amPitchshift”))

(<rt-out> (ladspa-run am (vct (in 0))))

(ladspa-set! am 0 1.5)

(make-ladspa-gui am)
```

# Example: Using a CLM instrument

- It's usually enough only changing a few lines to transform a CLM instrument into an Snd-Rt instrument.
- 
- (example: fm-bell from clm-ins.scm)

# Common Music and Snd-Rt

- Making Snd-Rt work interact directly with Common Music is possible.
-
- (load “/usr/share/cm/src/cm.scm”)
- (cm)

# Example: Snd/Pd

```
(define instrument
   (<rt-out> 0 (* (extern amp 0.6)
                  (oscil)))))

(pd-inlet 0 'float
   (lambda (new-amp)
      (set! (-> instrument amp) new-amp))))
```

# Sidenote: What did *extern* in the previous example do?

- 
- (**<rt-out>** (oscil (extern (make-oscil))))
- 
- ...is the same as:
- 
- (let ((oscil (make-oscil)))
- (**<rt-out>** (oscil oscil)))
- 

# Snd as a Pd external

- Snd/Pd can process both Pd data and Pd signals.
- Data processing happens in a different thread. Communication between Pd and Snd is being performed by using two lockless ringbuffers.
-

# San Dysth

- San Dysth is a standalone realtime software synth application written in Snd using Snd-Rt to generate sound.

# Lessons

- I've had lots of trouble because of static typing.
- Reliability is more important than speed.
- Convenience is usually more important than speed.

# Future work

- Remove static typing. Replace with dynamic typing.
- Make Guile have better support for soft realtime.
- Add support for Faust:
- (**<rt-faust>** (import “music.lib”)
- (= smooth “*(1-c) : +~*(c)”)
- (= vol (: (hslider “volume (db)” 0 -96 0 0.1)
- db2linear
- (smooth 0.99)))
- (= freq (hslider “freq” 1000 0 24000 0.1))
- (= process (vgroup “Osc” (* (osci freq) vol)))
- [I think this should be very simple to implement]