SND-RT: An uncompromising system for programming sound and music

Kjetil Matheussen, k.s.matheussen@notam02.no. NOTAM and Department of Informatics, University of Oslo

1. Overview

SND-RT is an environment for using high-level programming languages to make sound and music. Some of its most distinct features are:

- A new type of garbage collector which is suitable for realtime audio DSP.
- All programming happens on the sample level, so making "black boxes" in C or C++ are unnecessary.
- By using the special **block** operator, signal processing code is placed directly in the code controlling the flow of events. This implies:
  - No separation between *score language* and *audio language*
  - No hidden audio signal graph to control
  - Less need for signal buses

2. Programming

Language features

- Access to three languages: Faust, "RT" and Stalin.
- Strongly timed coroutines and dynamic control rate. Similar to ChucK.
- Closures
- Higher order functions
- Interactive development
- May use CLM or Faust for DSP operations
- Faust is integrated in both "RT" and Stalin

Using Stalin Scheme for realtime DSP programming

- Stalin has a very aggressive whole-program optimizer which often makes code run faster than C.
- It is a functional language with dynamic typing.
- Allocating closures, lists, CLM generators, Faust instances, etc. are all performed in realtime using the Rollendurchmesserzeit-sammler garbage colletor.

A sine-wave grain cloud

Example

Hard realtime

```
(<rt-stalin>
 (while #t
   (wait (random 30):-ms)
   (spawn
     (define osc  (make-oscil :freq (between 50 2000)))
     (define dur  (between     400  2000):-ms)
     (define e    (make-env  '((0 0)(.5 .05)(1 0)) :dur dur))
     (block :dur dur
       (out (*  (env e)
                (oscil osc))))))
```

A complete polyphonic midi softsynth

Example

Hard realtime

```
(<rt-stalin>
 (while #t
   (wait-midi :command note-on
     (spawn
       (define osc     (make-oscil :freq (midi-to-freq (midi-note))))
       (define player (spawn (block (out (*  (midi-vol)
                                              (oscil osc))))))
       (wait-midi :command note-off :note (midi-note)
         (stop player))))))
```

3. The block operator

For creating efficient sample iteration loops

- A native operator which can not be implemented efficiently only using dynamic control rate.
- **block** can be placed anywhere in code, even inside conditionals.
- This example implements an oscillator using **block** to iterate over all samples, manually incrementing the phase for each iteration:

```
(<rt-stalin>
  (define phase -0.062)
  (block (out (sin (inc! phase 0.062)))))
```

4. Rollendurchmesserzeitsammler

A conservative garbage collector for audio DSP

How it works

Simple basic technique: All roots and pointer-holding memory are copied to a separate buffer at regular intervals. The garbage collector runs in a parallel thread and finds garbage just by inspecting that buffer.

- Hard realtime safe.
- Provides an extremely efficient memory allocator. DSP Code may run faster if using Rollendurchmesserzeitsammler instead of custom memory pools.
- Available as a separate library: http://www.notam02.no/~kjetism/rollendurchmesserzeitsammler/

5. Acknowledgments

- CLM and Snd are made by Bill Schottstaedt, Stanford University.
- Faust is made by Yann Orlarey, Grame Centre National de Création Musicale.
- Stalin is made by Jeffrey Mark Siskind, Purdue University