SOUNDLAB: Lazy signal combinators

Max Rottenkolber <max@mr.gy>

May 29, 2013

Table of Contents

- Hi, my name is Max Rottenkolber.
- `SOUNDLAB` is a tool I designed to explore sound synthesis.
- It makes projecting ideas through speakers easy.
- Its much like writing a script for the speaker.
- Imagine I would hand out colorful glass shards and distorting mirrors so you can mess around with the beamer. That's the visual counterpart of `SOUNDLAB`.
- **I - III** Quickly explain the underlying concepts.
- **IV** Have fun with `SOUNDLAB`.

1 HOW DOES THIS WORK?

Figure 1. *The workflow.*

- We compile a signal.
- We pass a signal to our sample function.
- The sample function produces a *WAVE* file.
- We then playback the result with `aplay` or similar.

The sum of sines, e.g.
A function which maps time to amplitude (-1..1) e.g. `#'SIN`.

- The sample function collects the results of a linear sequence of calls to a signal function.

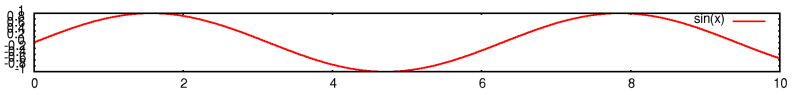

Figure 2. *Sine.*

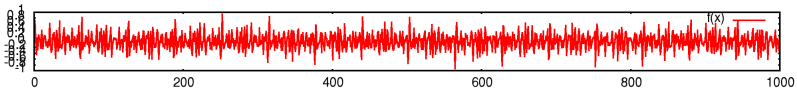

Figure 3. *Sine noise.*

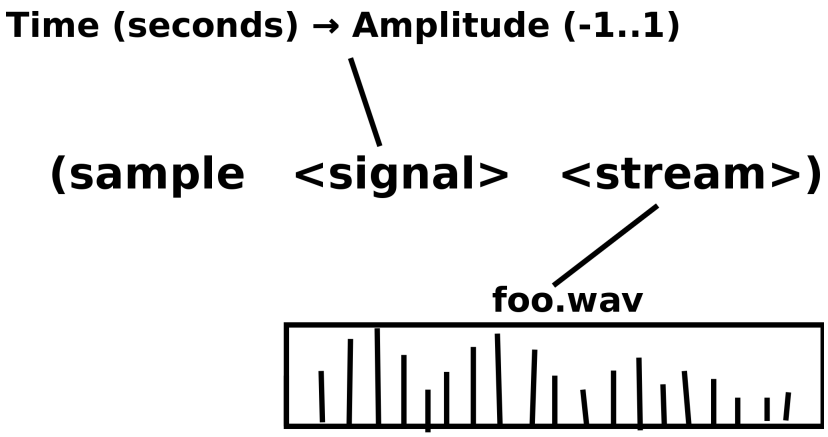

Figure 4. *Sampling simplified.*

- The results are written do disk in a *WAVE* file, which is basically an array of fixnums.
- Nothing happens. What's wrong?
- The recording is fine. 1Hz is way below the human hearing threshold.
- Increase frequency.

```
(defun chord (&rest signals)

  (lambda (x)
    (/ (loop for signal in signals
          sum (funcall signal x))
       (length signals)))

  (divide (apply #'add signals)
          (flatline (length signals))))
```

Figure 5. *Implementing* `CHORD`.

```
(FUNCTION (&REST (FUNCTION (REAL) REAL))
          (FUNCTION (REAL) REAL))
```

Figure 6. *A common interface for time and amplitude modulation.*

2 SIGNAL COMBINATORS

- Let's say we want to implement a function `CHORD`.
- The first implementation is fairly straight forward thanks to the `LOOP` macro.
- This gets complicated really fast.
- The second implementation uses the signal combinator approach
- It reuses two combinators `DIVIDE` and `ADD` and a primitive signal `FLATLINE`.
- It's based on a common interface which grants composability and reuseability.

3 OVERVIEW OF THE LIBRARY

- Primitive signals: Functions which generate basic waveforms.

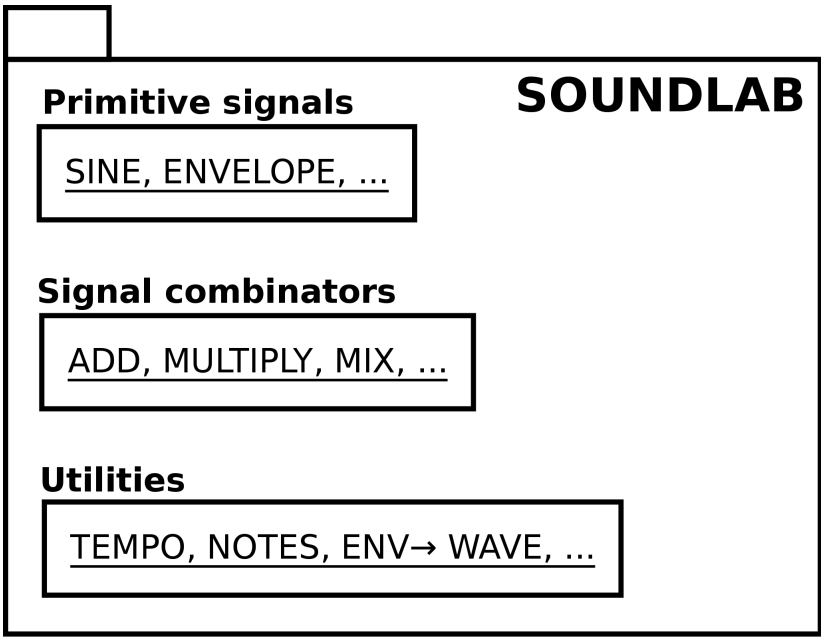

Figure 7. *Library overview.*

- Combinators: Combine primitive or compound signals to new compond signals.
- Utilities: Mostly functions dealing with time, musical scale and glue for different types of signals.

4 USING SOUNDLAB

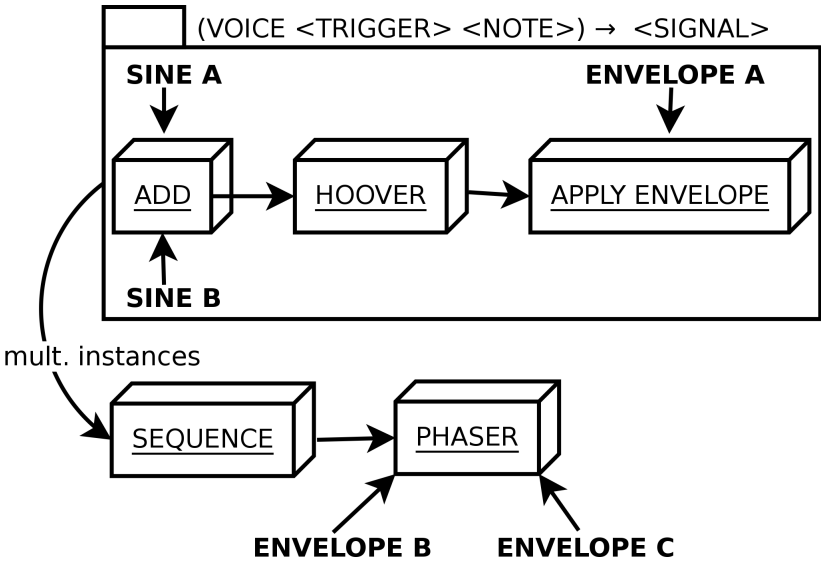

Figure 8. *What we are going to do.*

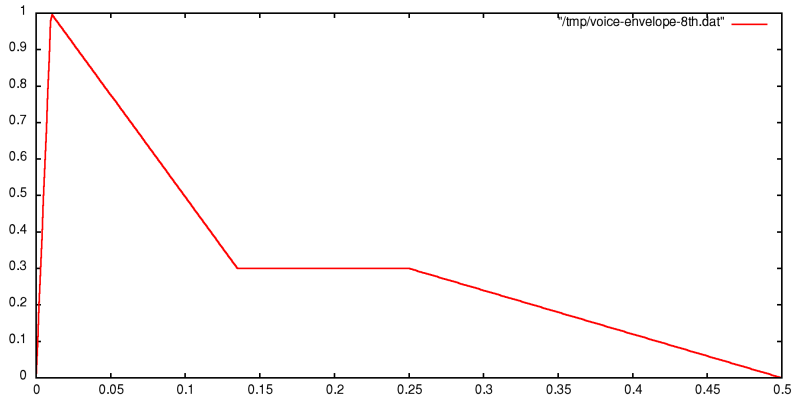

Figure 9. *Voice envelope.*

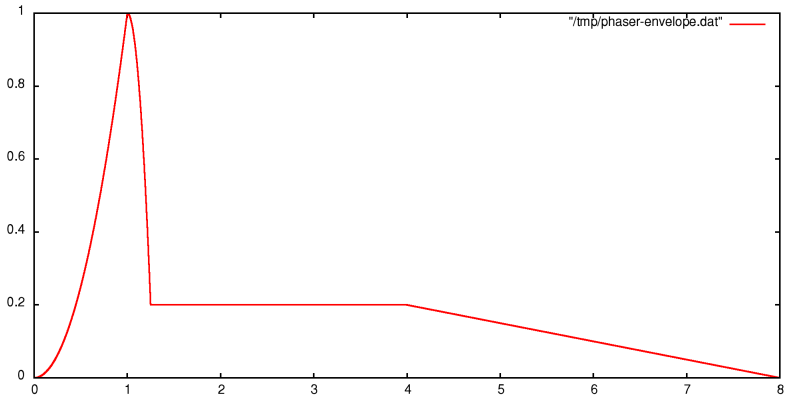

Figure 10. *Phaser envelope.*