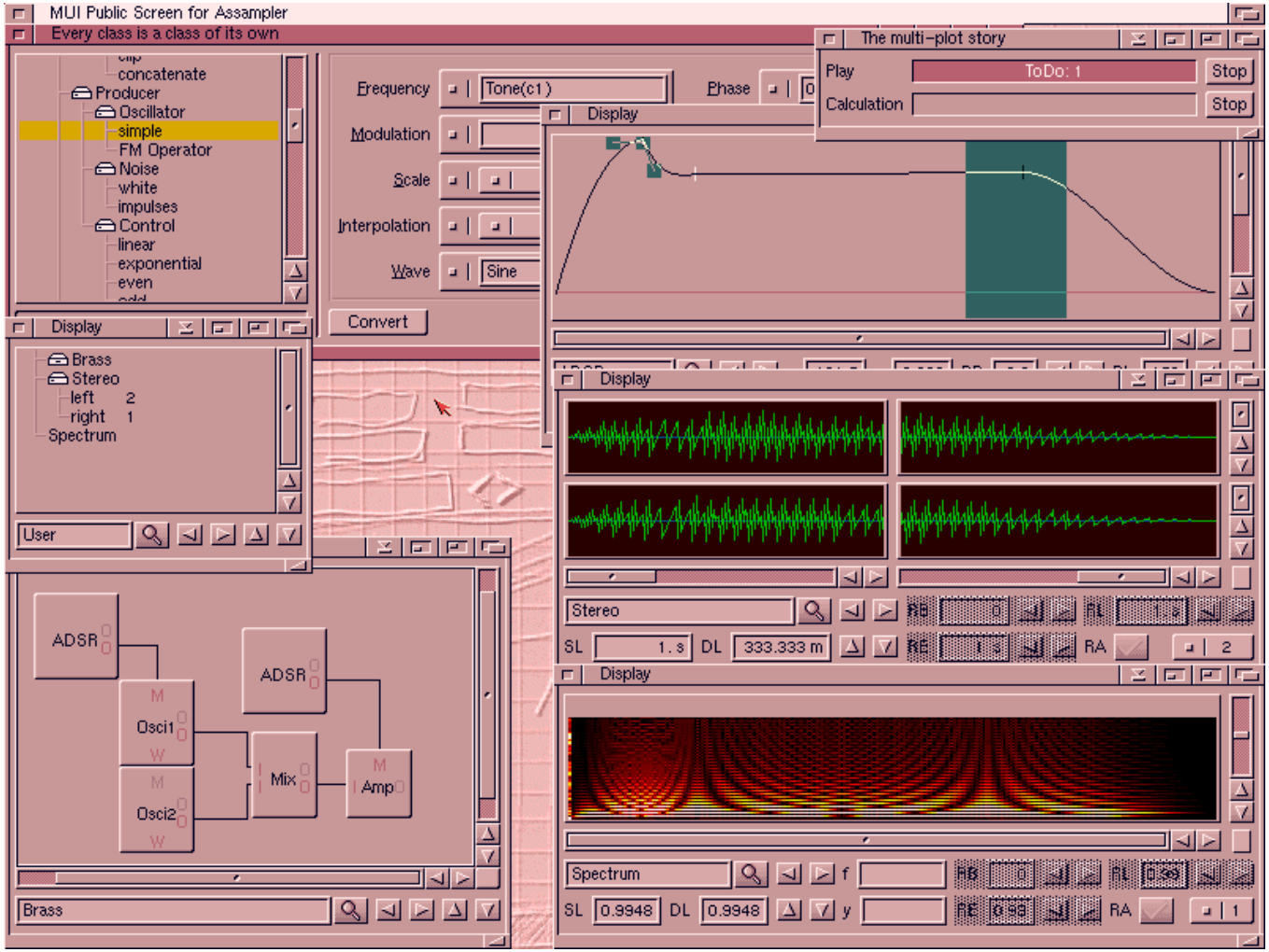
Haskell musiziert

Henning Thielemann

2007-07-10


MUI Public Screen for Assampler
Every class is a class of its own
concatenate
Producer
Oscillator
simple
FM Operator
Noise
white
impulses
Control
linear
exponential
even
Frequency
Tone(c1)
Phase
Modulation
Scale
Interpolation
Wave
Sine
Convert
The multi-plot story
Play
ToDo: 1
Stop
Calculation
Stop
Display
Brass
Stereo
left 2
right 1
Spectrum
User
ADSR
ADSR
Osci1
Osci2
Mix
Amp
Brass
Stereo
SL 1.s
DL 333.333 m
RA 2
Spectrum
SL 0.9948
DL 0.9948
RA 1

Assampler

- 1999 für Commodore Amiga
- Zusammenstellen von Klangverarbeitungsalgorithmen: graphisch oder per Formular
- blockweise Bedarfsauswertung
- physikalische Einheiten
- Beispiel: Ping
  Sinuston, Exponentialkurve, Verstärker

Assampler - Probleme

- Aufteilen von Signalstrom (Gabelung)
- Rückkopplung, Latenz
- Abgleich von Abtastraten und Amplitude
- Umgang mit Zufallszahlen, kohärentes Rauschen
- flexiblere Automatisierung (Beispiel: Mehrbandvocoder)
- Übersichtlichkeit: Modularisierung, Dokumentation
- Musikarrangement

Haskell

- Bedarfsauswertung (lazy evaluation) erlaubt
  - Programmierung in Reihenfolge von Signalabhängigkeit
  - Verarbeitung in zeitlicher Reihenfolge (streaming)
- Bedarfsauswertung erlaubt Rückkopplung
  - `let y = x + delay y in y`
  - `fix (\y -> x + delay y)`
- Gabelungen (sharing)
- Zufallszahlen: Monade
- Analogie: Typenabgleich (type inference)
  und Abstimmung von Abtastrate und Aussteuerung
- Compiler für Haskell

Signalverarbeitung mit funktionaler Programmierung

```
ping =
    envelope
        (exponential 0.1)
        (oscillator 440)
```

Signalparameter - maschinennah

digitale Hardware-Synthesizer, CSound, SuperCollider:

- feste Abtastraten für
  1. hörbare Signale
  2. Steuersignale
  3. Notenwiedergabe
- untypisiert
- keine physikalischen Einheiten

Signalparameter - abstrakt

Ziel: Annäherung an analoge Daten

Synthesizer-Bibliothek für Haskell:

- frei wählbare Abtastraten, automatischer Abgleich
- statisch typisiert
  Typen für Abtastwerte:
  - kontinuierliche Werte – `Double`
  - Wertetupel für Stereophonie – `(Double,Double)`
  - An/Aus-Steuerung – `Bool`
- physikalische Einheiten (dynamisch überprüft)
  - Modellierung natürlicher Vorgänge
  - Nachbau Analogsynthesizer
  - höhere Programmiersicherheit

Signalverarbeitung: Hörbeispiele

Haskell (Konserve)

- Fliege
- Glasbruch
- Rauschen mit Tschebyscheff-Filter

SuperCollider (Echtzeit)

- Blubberblasen
- Wind
- Bass mit Filter

Haskore

- Paul Hudak: „The Haskell School of Expression. Learning Functional Programming Through Multimedia“
- Algebra der Musik
  - sequentielle (+:+) und parallele (=:=) Komposition
  - Rechengesetze

$$
\begin{aligned}
\texttt{(a +:+ b) +:+ c} &= \texttt{a +:+ (b +:+ c)} \\
\texttt{(a =:= b) =:= c} &= \texttt{a =:= (b =:= c)} \\
\texttt{a =:= b} &= \texttt{b =:= a} \\
&\vdots
\end{aligned}
$$

- Fortschritt gegenüber SoundTracker&Co.: Formulierung von Ideen, weniger Kopieren von Abschnitten
- Haskell erlaubt gleichzeitig Musikarrangement und Signalerzeugung

Begleithilfen

- regelmäßige aber nicht-periodische Muster:
  Quersummen der natürlichen Zahlen modulo 4 im 4er-System
- Gitarrenakkorde

Wiedergabemöglichkeiten

- Haskell-Synthesizer
  - Unter Wasser: Bandpass auf Sägezahn
  - Häcksler: FM-Synthese
  - Schwanensee: Bandpass spielt Melodie auf Akkordbegleitung
- MIDI-Dateien: ChildSong
- CSound: Alle Vögel sind schon da
- SuperCollider (Haskell-Schnittstelle von Rohan Drape)
  - Glissando
  - Chill out

Ausblick

- Abgleich der Klangparameter für unendlich viele Klänge
- Echtzeit-Synthesizer mit `ByteString`&Co.
- Echtzeit-MIDI
- mehr Interaktivität:
  - Klangausgabe bei Tastendruck
  - Hervorhebung aktuell gespielter Note bei Wiedergabe

Assampler

Amiga-Assampler

- Präsenz: `http://www.assampler.de/`
- Saugen: `http://aminet.net/search?query=assampler`
- Amiga-Emulator:
  - Linux: `http://www.rcdrummond.net/uae/`
  - Windows: `http://www.winuae.net/`

Haskore

Leider sehr viele Abhängigkeiten,
die nicht unbedingt nötig sind

- Internetpräsenz von Paul Hudak für Original-Haskore:
  `http://www.haskell.org/haskore/`
- Auf Haskore basierende Projekte:
  `http://www.haskell.org/haskellwiki/Haskore`
- Grundlegend überarbeitetes Paket (Baustelle!):
  `http://darcs.haskell.org/haskore/`
- unterstützt MIDI-Player Timidity, CSound, SuperCollider

Klangsynthese

- Große Baustelle!
  `http://darcs.haskell.org/synthesizer/`
- benötigt
  - `http://darcs.haskell.org/unique-logic/`
  - `http://darcs.haskell.org/numericprelude/`
  - `http://darcs.haskell.org/shell-pipe/`
- unterstützt Sox für Klangwiedergabe
  und Konvertierung in übliche Audioformate
- Für Demonstrationen:
  `http://darcs.haskell.org/dafx/`

SuperCollider

- SuperCollider: Echtzeit-Synthesizer von James McCartney
  `http://www.audiosynth.com/`
- Ansteuerung mit Haskell
  - Open Sound Control: (benötigt für Haskore)
    `http://slavepianos.org/rd/sw/sw-78`
  - SuperCollider: (benötigt für Haskore)
    `http://www.slavepianos.org/rd/sw/sw-69`
  - SuperCollider-Zubehör:
    `http://slavepianos.org/rd/sw/sw-76`
  - außerdem benötigt für Haskore:
    `http://darcs.haskell.org/supercollider-ht/`

SuperCollider-Beispiele

osc-Dateien enthalten Kommandos für SuperCollider

Audiodaten berechnen:

```
$ scsynth -D 0 -o 2 -N song.osc _ song.aiff 44100 AIFF int16
$ bunzip2 --stdout song.osc.bz2 | \
      scsynth -D 0 -o 2 -N _ _ song.aiff 44100 AIFF int16
```

CSound

- „Sound-Compiler“ von Barry Vercoe
- `http://www.csounds.com/`
- Sprache für Beschreibung von Klängen (Orchestra)
- Liste von Klangereignissen (Score)
- Haskore erzeugt beide Dateiarten

Nicht zu vergessen ...

- Digitale Signalverarbeitung mit Haskell
  von Matthew Donadio:
  `http://haskelldsp.sourceforge.net/`,
  `http://darcs.haskell.org/dsp/`
- Physikalische Klangmodellierung mit Clean
  von Jerzy Karczmarczuk:
  "Functional Framework for Sound Generation"
  `http://users.info.unicaen.fr/~karczma/arpap/cleasyn.pdf`
- Diskussionen über **Haskell und Kunst**:
  `http://lists.lurk.org/mailman/listinfo/haskell-art`