Modular synthesizers?

Where does Yampa enter the picture?

- Music can be seen as a hybrid phenomenon. Thus interesting to explore a hybrid approach to programming music and musical applications.
- Yampa’s programming model is very reminiscent of programming modular synthesizers:

![Diagram of Yampa network](image)

- Fun application! Useful for teaching?

What have we done?

Framework for programming modular synthesizers in Yampa:

- Sound-generating and sound-shaping modules
- Additional supporting infrastructure:
  - Input: MIDI files (musical scores), keyboard
  - Output: audio files (.wav), sound card
  - Reading SoundFont files (instrument definitions)
- Status: proof-of-concept, but decent performance.

Example 1: Sine oscillator

```
oscSine f cv
oscSine :: Frequency -> SF CV Sample
oscSine f0 = proc cv -> do
  let f = f0 * (2 ** cv)
  phi ← integral −≺ 2 * pi * f
  returnA ← sin phi
constant 0 ≫ oscSine 440
```

Example 2: Vibrato

```
oscSine 5.0
oscSine f
*0.05
constant 0 ≫ oscSine 440
```

Example 3: 50’s Sci Fi

```
sciFi :: SF () Sample
sciFi = proc () → do
  und ← arr (+0.2) ≪ oscSine 3.0 ← 0
  sup ← arr (+1.0) ≪ integral ← −0.25
  audio ← oscSine 440 ← und + sup
  returnA ← audio
```

Envelope Generators (1)

```
envGen :: CV → [(Time, CV)] → (Maybe Int)
  → SF (Event () (CV, Event ()))
envEx = envGen 0 [(0.5, 1), (0.5, 0.5), (1.0, 0.5), (0.7, 0)]
(Just 3)
```
Read samples (instrument recordings) from SoundFont file into internal table.

→ Oszillator similar to sine oscillator, except table lookup and interpolation instead of computing the sine.

Envelope Generators (2)

How to implement?
Integration of a step function yields suitable shapes:

Envelope Generators (3)

Envelope generator with predetermined shape:

\[
\text{afterEach} : [(\text{Time}, b)] \rightarrow SF a (\text{Event} b)\\
\text{hold} \quad a \rightarrow SF (\text{Event} a) a\\
\text{steps} = \text{afterEach} [(0.7, 2), (0.5, -1), (0.5, 0), (1, -0.7), (0.7, 0)]
\]

Example 4: Bell

\[
\text{envGen} \, CV \rightarrow [ (\text{Time}, CV)] \rightarrow SF, a CV\\
\text{envGen} \, l0 \, tls = \text{envGenAux} \, l0 \, tls \, \rightarrow \text{afterEach} \, trs \, \rightarrow \text{hold} \, r0
\]

\[
\int \rightarrow \text{integral} \rightarrow \text{arr} (+ \, l0)
\]

\[
\text{afterEach} \, [(\text{Time}, b)] \rightarrow SF \, a \, (\text{Event} \, b)
\]

Envelope Generators (5)

Example 5: Tinkling Bell

Envelope generator responding to key off:

\[
\text{envGen} \, CV \rightarrow [ (\text{Time}, CV)] \rightarrow SF, a CV\\
\text{envGen} \, l0 \, tls \rightarrow \text{envGenAux} \, l0 \, tls1 \, \rightarrow \text{afterEach} \, trs1 \, \rightarrow \text{hold} \, r0
\]

where

\[
(r0, trs) = \text{toRates} \, l0 \, tls
\]

Example 6: Playing a C-major scale

\[
\text{scale} : SF () \rightarrow \text{Sample}\\
\text{scale} = (\text{afterEach} [(0.0, 60), (2.0, 62), (2.0, 64), (2.0, 66), (2.0, 67), (2.0, 69), (2.0, 71), (2.0, 72)])
\]

\[
\Rightarrow \text{constant} ()
\]

\[
\Rightarrow \text{arr} (\text{fmap} (\text{bell} \circ \text{midiNoteToFreq}))
\]

\[
\Rightarrow \text{rSwitch} (\text{constant} 0)
\]

Example 7: Playing simultaneous notes

\[
\text{mysterySong} : SF () \rightarrow \text{Sample, Event}\\
\text{mysterySong} = \text{proc} \, t \rightarrow \text{do}\\
\text{t} \rightarrow \text{tinkle} \, \rightarrow ()\\
\text{m} \rightarrow \text{mystery} \, \rightarrow ()\\
\Rightarrow \text{return} \, (0.4 \times t + 0.6 \times m)
\]

A polyphonic synthesizer (1)

Sample-playing monophonic synthesizer:

- Read samples (instrument recordings) from SoundFont file into internal table.
- Oscillator similar to sine oscillator, except table lookup and interpolation instead of computing the sine.

SoundFont synthesizer structure:
A polyphonic synthesizer (2)

Exploit Yampa’s switching capabilities to:
• create and switch in a mono synth instance in response to each note on event;
• switch out the instance in response to a corresponding note off event.

Switched-on Yampa?

Software and paper: www.cs.nott.ac.uk/~gqg