Q: A Functional Programming Language for Multimedia Applications

Albert Gräf

Dept. of Music-Informatics
Quick overview

- What?
- Why?
- The Library
- MIDI, Audio and OSC Interfaces
- Demo
- Conclusion
What?

- A **functional programming language** based on **term rewriting**.
- Programs are collections of **algebraic equations**.
- Executing a program means to **evaluate an expression**.

\[
\text{sqr} \ X = \ X \times \ X;
\]
\[
\text{sqr} \ 2 + \text{sqr} \ (2+3) \Rightarrow 2\times2 + \text{sqr} \ (2+3) \\
\Rightarrow 4 + \text{sqr} \ (2+3) \Rightarrow 4 + \text{sqr} \ 5 \\
\Rightarrow 4 + 5 \times 5 \Rightarrow 4 + 25 \Rightarrow 29
\]

\[
\text{gcdiv} \ X \ Y = \begin{cases} 
  \text{gcdiv} \ Y \ X & \text{if } Y > X; \\
  \text{gcdiv} \ Y \ (X \mod Y) & \text{if } Y > 0; \\
  X & \text{otherwise}; 
\end{cases}
\]

\[
\text{qsort} \ [ ] = [ ]; \\
\text{qsort} \ [X|Xs] = \text{qsort} \ (\text{filter} \ (<X) \ Xs) ++ [X] ++ \text{qsort} \ (\text{filter} \ (\geq X) \ Xs);
\]
Q: A Functional Programming Language

Why?

- Started as a (master) research project on pattern matching techniques for term rewriting.
- Idea was to turn this into a simple, practical programming language (ca. 1991).
- Turned out quite different from both ML and Haskell. Simpler. Interpreted. Dynamic typing. ⇒ “functional scripting language”
- Multimedia facilities in other modern-style FPLs were missing when I needed them, decided to do my own.
Q: A Functional Programming Language

The Library

Standard Library:
- lists, streams,
- containers,
- lambda calculus, ...

GUI+Graphics:
- Tcl/Tk, GGI,
- Freetype,
- ImageMagick

POSIX:
- I/O,
- processes,
- threads, sockets,
- regexps, ...

Multimedia:
- audio, MIDI, OSC,
- OpenGL, Xine

Q-Interpreter

C/C++

Q-SWIG

Scientific programming:
- Octave, OpenDX,
- Graph library

Web:
- Apache module,
- XML+XSLT, Curl,
- ODBC
MIDI Interface

- based on Grame's *MidiShare*
- dynamic routing and realtime processing of MIDI messages
- algebraic `MidiMsg` type; sequences are represented as lists
- standard MIDI file support
import midi;

/* register a MidiShare client and establish I/O connections */
def REF = midi_open "Transpose",
    IO = midi_client_ref "MidiShare/ALSA Bridge",
    _ = midi_connect IO REF || midi_connect REF IO;

/* transpose note on and off messages, leave other messages unchanged */
transp K (note_on CH N V)
    = note_on CH (N+K) V;
transp K (note_off CH N V)
    = note_off CH (N+K) V;
transp K MSG    = MSG otherwise;

/* the following loop repeatedly reads a message, transposes it and 
   immediately outputs the transformed message */
transp_loop K   = midi_send REF 0 (transp K MSG) || transp_loop K
    where (_,_,_,MSG) = midi_get REF;
Audio Interface

- **audio** module: *PortAudio* interface
- **sndfile** module: *Libsndfile* interface
- **wave** module: simple wave generation and manipulation operations, wave drawing, interface to *libsamplerate* and *FFTW*
Q: A Functional Programming Language

OSC Interface

- implements Berkeley's *Open Sound Control* protocol
- all standard OSC features supported, including nested bundles
- UDP support
- special support for *SuperCollider*
- current version is written in Q; might use *liblo* in the future
OSC Interface

/* note offs: set the gate of the synth to 0 and put it at the end of the queue */
loop P Q (_,note_on _ N 0)
    = n_set I ("gate",0) || loop P Q midiin
      where (I,_) = P!N, P = delete P N, Q = append Q I;
    = loop P Q midiin otherwise;
loop P Q (T,note_off CH N _)
    = loop P Q (T,note_on CH N 0);

/* note ons: turn note off if already sounding, then get a new voice from the queue and set its gate to 1 */
loop P Q (T,note_on CH N V)
    = n_set I ("gate",0) || loop P Q (T,note_on CH N V)
      where (I,_) = P!N, P = delete P N, Q = append Q I;
      loop P Q midiin
      where [I|Q] = Q, FREQ = freq N,
        P = insert P (N,(I,FREQ));
Q: A Functional Programming Language

Demo
Conclusion

- Q: a modern-style functional programming language based on term rewriting.
- Already good support for multimedia and computer music applications.
- Future work: library support (Jack, LADSPA, DSSI, ...), high-level interfaces.
- It's free! (GPL)
- More info: q-lang.sf.net