FASTLab & HeavenEverywhere Present

CSL6 & Siren9 Demo

Stephen Travis Pope - stephen@heaveneverywhere.com - 2020.05.21

1

OVERVIEW

- **CSL** = CREATE Signal Library
  - C++ class library and example apps
  - Audio synthesis, processing and spatialization
- **Siren** = High-level Tools for Music & Audio
  - Smalltalk class library, tools and example apps
  - Music composition and structure-oriented interaction

2

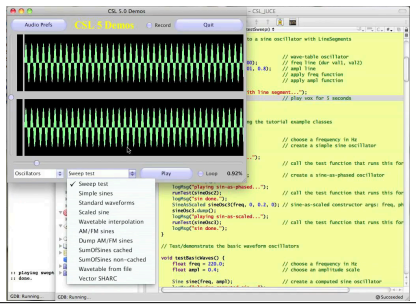
THE CREATE SIGNAL LIBRARY (CSL)

- Since the 1990s, teaching DASP in C++ at UCSB
  - Six-part course sequence on digital audio programming
  - See *TheBigMATBook*
- Now: DASP framework for use with JUCE (opt.)
  - Audio-centric app development
  - GUIs using JUCE (or native or javascript)
  - Control via MIDI, OSC or other means
  - Cross-platform: Mac, Linux, Windows, iOS, Android…
  - Used in numerous commercial apps, games, etc.

3

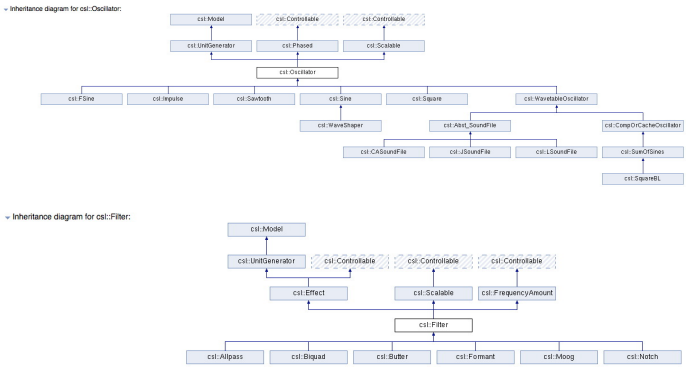
PREVIOUS CSL DEMO VIDEOS

- CSL 5.0 demo videos (2006)
  - https://vimeo.com/409028476
  - https://vimeo.com/409027546



4

CSL CLASS HIERARCHIES



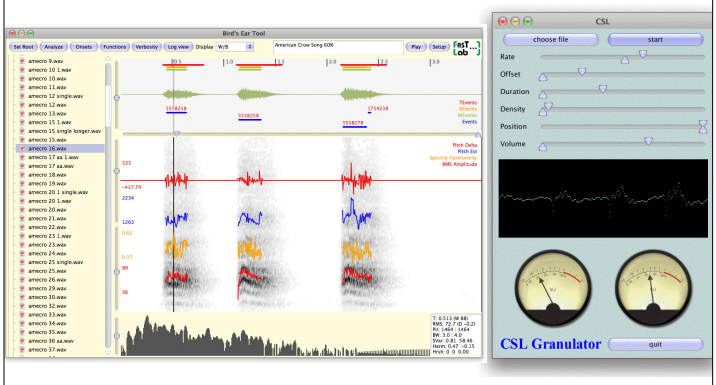
5

CSL CODE EXAMPLE

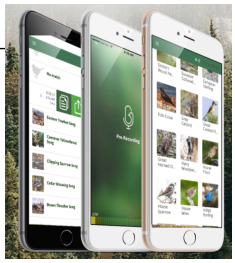
```
// CSL demo to play dynamic band-passed white noise

void testDynamicFilters() {
    float dur = 6.0f;                          // seconds to play
    WhiteNoise noiz(1.0);                      // noise generator
    RandEnvelope ctr(3, 1000, 2000, 600);      // filter center/bw random walk envelopes
    RandEnvelope bw(3, 100, 100, 40);          // c'tor args: (frq, amp, offset, step)
    Butter butter(noiz, kBandPass, ctr, bw);   // Butterworth filter (inp, type, frq, bw)
    center.trigger();                          // trigger the envelopes
    bw.trigger();
    gIO->setRoot(butter);                      // plug the filter into the global IO
    sleepSec(dur);                             // sleep for the desired duration
    gIO->clearRoot();                          // clear the IO's root to make silence
}
```

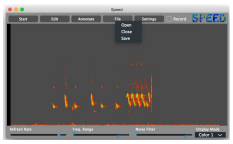
6

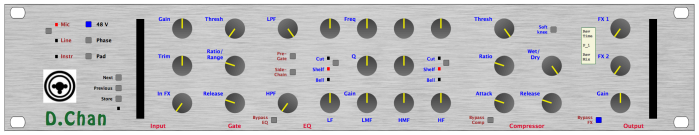
CSL APP EXAMPLES



CSL APP EXAMPLES 2







CSL INSTRUMENTS

- Bundle a DSP graph with accessors for its parameters and a method to trigger its envelopes
- Drive from MIDI, OSC or GUI interfaces
- Construct libraries with many instruments
  - All instruments sum to a common mixer & effects chain
  - Instruments in the library may respond to different modes of interaction (MIDI, OSC, GUI)

CSL INSTRUMENTS AND OSC

- Sample player instrument accessors
  - "fi", set_file_f
  - "am", set_amplitude_f
  - "ra", set_rate_f
  - "po", set_position_f
  - "st", set_start_f
  - "en", set_stop_f
  - "at", set_attack_f
  - "de", set_decay_f
- OSC note formats
  - ampl, pos
  - ampl, pos, rate
  - ampl, pos, start, stop
  - ampl, pos, rate, start, stop
  - ampl, pos, start, stop, attack, decay
  - ampl, pos, rate, start, stop, attack, decay
- OSC address space, commands

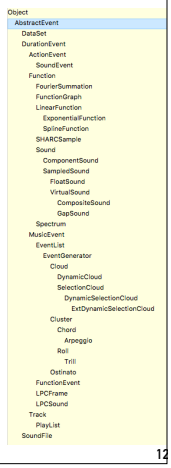
/i1/du f 2.0	set duration
/i1/am f 0.5	set amplitude
/i1/fi s "name.aiff"	set file
/i1/p	play-note

CREATING A MULTI-TIMBRAL SOFT-SYNTH

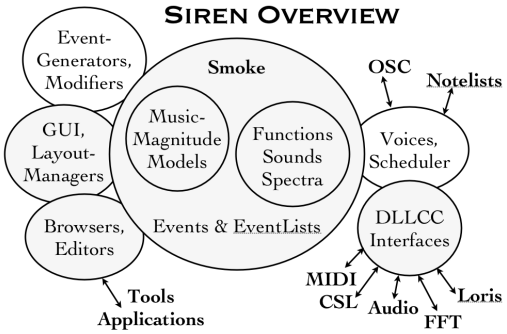
```
logMsg("Setting up [string, sampler, FM] library");
InstrumentVector lib;                          // instrument library
Mixer mix(2);                                  // Create the main stereo output mixer
for (unsigned i = 0; i < 16; i++) {            // 16 plucked strings (amp, frq, pos)
    StringInstrument * in = new StringInstrument(0.2f, 440.0f, 0.0f);
    lib.push_back(in);   mix.addInput(in);
}
char * names[] = { "moon.snd", "wet.snd", "round.snd", "shine.snd"};
for (unsigned i = 16; i < 32; i++) {           // 16 sound files
    SndFileInstrument0 * in = new SndFileInstrument0(dataFolder(), names[i % 4]);
    lib.push_back(in);   mix.addInput(in);
}
for (unsigned i = 32; i < 48; i++) {           // 16 FM voices
    FMInstrument * in = new FMInstrument();
    lib.push_back(in);   mix.addInput(in);
}
setupOSCInstrLibrary(lib);                     // add the instrument library to OSC namespace
```


SIREN SMALLTALK TOOLS

- Since 1984: DoubleTalk, HSTK, MODE and Siren
- Elements:
  - the Smoke music representation language (music magnitudes, events, generators, functions and sounds)
  - voices, schedulers and I/O drivers (real-time and file-based voices, sound, OSC and MIDI I/O)
  - user interface components for musical applications (UI layout tools, widgets and examples)
  - several built-in applications (editors and browsers for several kinds of Siren objects)
  - interfaces to external libraries for audio/MIDI/OSC I/O, and packages such as CSL, SuperCollider and Loris

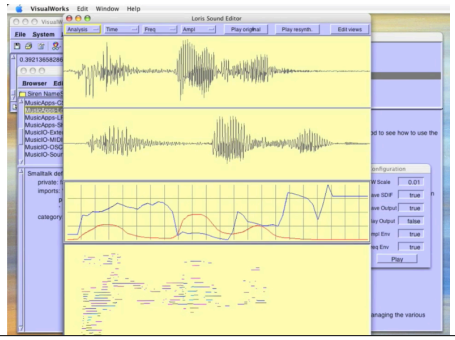


SIREN OVERVIEW



PREVIOUS SIREN DEMO VIDEOS

- Siren demo video (2007)
  - https://vimeo.com/120751122



SMOKE MUSIC REPRESENTATION

- The SMOKE representation supports the following:
  - abstract models of the musical magnitudes (pitch, loudness and duration)
  - flexible grain-size of "events" (from grain to whole composition)
  - event, control, and sampled sound description
  - nested/hierarchical event-trees; parallel or sequential organization
  - instrument/note (voice/event, performer/music) abstractions
    - separation of "data" from "interpretation" ("what" vs. "how"--voices)
  - "middle-level" musical structures (e.g., chords, clusters, or trills)
  - no time model, duration only

SMOKE EVENT EXAMPLES

- Musical "notes" (concatenate music magnitudes or arbitrary name/value tuples)

  440 Hz, 270 msec, 44 dB

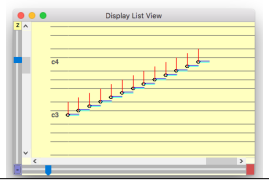
  60 key, (1/4 beat), #ff ampl, (#voice -> #flute),
  (#embrochure -> #tight)

- Verbose format

  (MusicEvent dur: 1/4 pitch: 'c3') color: #green

SMOKE EVENT-LIST EXAMPLES

- A chord = 3 events at the same time
  ((0 => (1/2 beat, 'd3' pitch, 'mf' ampl)),
  (0 => (1/2 beat, 'fs3' pitch, 'mf' ampl)),
  (0 => (1/2 beat, 'a4' pitch, 'mf' ampl)))
- C'tor methods
  (EventList scaleFrom: 48 to: 60 in: 1.5) open



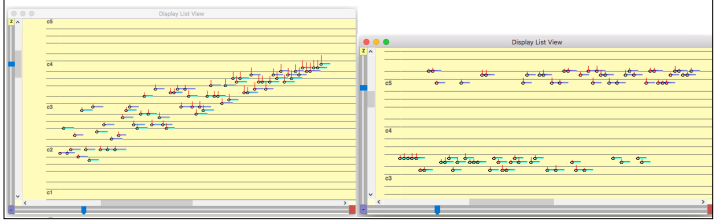
(Hauer-Steffens notation)

EVENT GENERATORS AND EVENT MODIFIERS

- Objects that model "middle-level" musical structures
- Have flexible c'tors and methods to return their event lists
- May be used in interactive player threads
- Examples
  - Chord, Cluster, Cloud, SelectionCloud, DynamicSelectionCloud
  - Roll, Ostinato, Trill, Arpeggio, Peal
- EventLists created by EventGenerators can be processed further with filters, rules, etc.
- EventModifiers apply functions to event properties

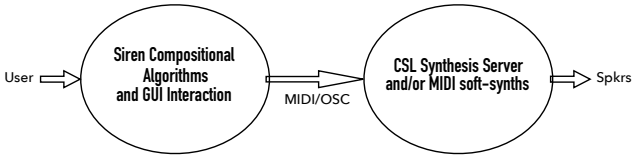
EVENT GENERATOR EXAMPLES

(DynamicCloud dur: 6 "give starting/ending selection ranges"
pitch: #((30 to: 49) (60 to: 60))
ampl: #((20 to: 40) (90 to: 120))
voice: #((1) (1))
density: 12) open



SIREN + CSL

- Siren runs compositional algorithms and GUI interaction
- CSL runs synthesis server controlled by OSC and/or MIDI
- Other (off-the-shelf) synthesis components possible



SUMMARY, LINKS

- Summary
  - Mature advanced cross-platform toolkits specialized for the very different tasks of music composition and performance
- **Siren**
  - http://fastlabinc.com/Siren
  - http://github.com/stpope/Siren9
- **CSL**
  - http://fastlabinc.com/CSL
  - http://github.com/stpope/CSL6

DEMO

- Canned Siren demos (included in built-in workbook)
- Sketches from *SleepingSword* (WIP)