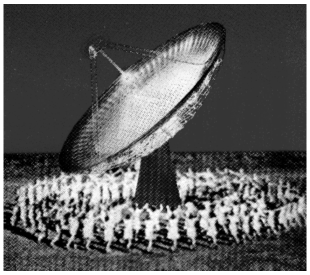
Siren: Sound and Music Tools for Squeak



Stephen T. Pope
CREATE, UCSB
`stp@create.ucsb.edu`

Outline

- **Squeak Smalltalk**
- **Squeak Cross-Compiler Technology**
- **Siren Background**
- **Siren Models**
- **Siren Sound Synthesis**
- **Demonstration**
- ***All source on the net, for more info, see: http://www.create.ucsb.edu/Siren***

Squeak Smalltalk

- **VI based on PARC Smalltalk-80 (v1)**
- **Devel at Apple, WDI, etc.**
- **New VM Written in Primarily Smalltalk(!) and translated to C**
- **Garbage Collector in Smalltalk**
- **Morphic GUI Framework (WIP)**
- **Ported to Mac, DOS/Win, UN*X, Wince, ...**

Squeak VM and Cross-Compiler (Ingalls et al.)

- **Squeak Smalltalk-to-C Translator**
- **Accepts a Subset of Smalltalk**
- **Compile C functions as "primitives"**
- **Used to generate the VM**
  (except OS interface, which is in C)
- **Used for DSP routines**
  (e.g., Sound Synthesis Classes)

Morphic UI Framework

- **Originally built as the UI framework for *Self* by John Maloney**
- **Display-list graphics merged with window/tool composition**
- **"Only objects on the screen!"**
- **Objects have slot inspectors, core samplers, viewers, etc.**
- **Tools are "composable"**

Siren/MODE Background

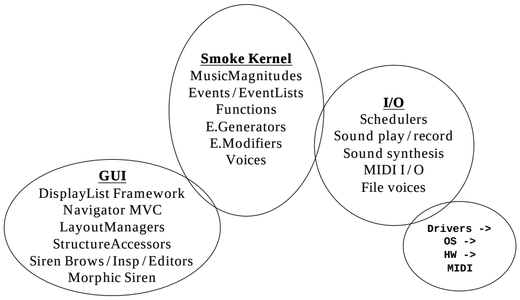
- **(I reimplement and rename it every 6 years or so.)**
- **SmallSong, DoubleTalk, HyperScore ToolKit, MODE, *Siren***
- **The Smoke Representation Language**
- **Siren I/O: Voices & Drivers**
- **Siren GUIs**
- **Siren Applications**

Why Siren?

- **Provide a Flexible and Extensible Environment for Musicians**
- **Address Tasks of Composition, Realization, and Production**
- **Support Working with *Sound***
- **Provide Extensible GUIs**

No Need to Support “Historical” Music

Siren Architecture



Smoke Kernel
MusicMagnitudes
Events / EventLists
Functions
E.Generators
E.Modifiers
Voices

I/O
Schedulers
Sound play / record
Sound synthesis
MIDI I / O
File voices

GUI
DisplayList Framework
Navigator MVC
LayoutManagers
StructureAccessors
Siren Brows / Insp / Editors
Morphic Siren

```
Drivers ->
OS ->
HW ->
MIDI
```

Smoke Music Representation

- **Smoke Consists of Classes for:**
  - **MusicMagnitudes**
  - **Events and EventLists**
  - **Functions**
  - **Voices, Ports, and Drivers**
  - **EventGenerators**
  - **EventModifiers**

Music Magnitude Models

- **Abstractions**

  Chronos, Chroma, Ergon, Positus
- **Representation Classes**

  Pitch, Duration, Loudness
- **Implementation Classes**

  HertzPitch, SymbolicP, RatioP, MIDIP

  (Pitch value: 'c3') == ('c3' pitch)

  (Amplitude value: #mf) == (#mf ampl)
- **Mixed-mode Arithmetic**

  (#f4 pitch + 80 Hz)
- **Extended MusicMagnitudes**
  - **ConditionalDuration, Sharpness**

Events

- **Events are just Property Lists**

  (with [optional] durations but *no start times*)
- **There are *Verbose* and *Terse* Formats**

```
(DurationEvent dur: 250 msec voice: #flute)

((880 Hz, 250 msec, (#voice -> #flute),
        0.7071 ampl) accent: #sfz)

anEvent color: #green; shape: #round
```

Event Lists

- **List of (Delay -> Event) Associations**

  The delay is the event’s start time relative to the start of the list (i.e., it’s a duration)
- **Methods to Add, Remove, Filter Events**
- **Methods to “perform” Events on their Voices**
- **Verbose and Terse Formats**

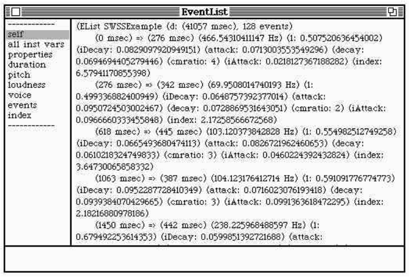
Event List Examples

```
[(EventList newNamed: #Chord1)
      add: ((1/2  beat), "d3"  pitch,  "mf" ampl)  at: 0;
      add: ((1/2  beat), "fs3"  pitch, "mf" ampl)  at: 0 … ]

 (0      => 440 Hz, 250 msec, -3 dB) ,
((1/4) => (471 Hz, 0.37 beat, #ff))

 EventList named: 'phrase1'
    fromSelectors: #(duration: loudness: phoneme:)
    values: (Array
      with: #(595 545 545 540 570 800 540)
      with: #(0.8 0.4 0.5 0.3 0.2 0.7 0.1)
      with: #(dun kel kam mer ge sprae che))
```

Event List Inspector



```
(EList SWSSExample (d: (41057 msec), 128 events)
    (0 msec) => (276 msec) (466.54310411147 Hz) (1: 0.507520636454002)
(iDecay: 0.0829097920949151) (attack: 0.0713003553549296) (decay:
0.0694694405279446) (cmratio: 4) (iAttack: 0.0218127367188282) (index:
6.57941170855398)
    (276 msec) => (342 msec) (69.9508014740193 Hz) (1:
0.499336882400949) (iDecay: 0.0648757392377014) (attack:
0.0950724503002467) (decay: 0.0728869531643051) (cmratio: 2) (iAttack:
0.0966660333455848) (index: 2.17258566672568)
    (618 msec) => (445 msec) (103.120373842828 Hz) (1: 0.554982512749258)
(iDecay: 0.0665493680474113) (attack: 0.0826721962460653) (decay:
0.0610218324749833) (cmratio: 3) (iAttack: 0.0460224392432824) (index:
3.64730065858332)
    (1063 msec) => (387 msec) (104.123176412714 Hz) (1: 0.591091776774773)
(iDecay: 0.0952287728410349) (attack: 0.0716023076193418) (decay:
0.0939384070429665) (cmratio: 3) (iAttack: 0.0991363618472295) (index:
2.18216880978186)
    (1450 msec) => (442 msec) (238.225968488597 Hz) (1:
0.679492253614353) (iDecay: 0.0599851392721688) (attack:
```

EventGenerators

- **Models of “Middle-Level” Structures**
- **Support “Composition by Refinement”**
- **Provide “Constant Performability”**
- **An Extensible Framework for Composition**

EventModifiers

- **Apply a function to an event list’s event properties**
- **Do so eagerly (at declaration time) or lazily (at performance time)**
- **EMods can be composed**
- **Rely on Smoke Function Models**

Siren Performance: Voices

- **Events or EventLists have “abstract” Properties and Voices**
- **A Voice is a Property-to-Parameter Mapper (e.g., HzPitch -> MIDIPitch to play a Hz-oriented score on MIDI)**
- **Voices can have Ports and Devices, or Files**

The Siren Scheduler

- **Class Scheduler and sole instance *Schedule* can have clients registered to receive the *scheduleAt*: message**
- **They may do something in response to it, and may answer a time when they wish to be scheduled again.**
- **Smalltalk-only scheduler is pretty fast!**

MIDI I/O

- Instance of MIDIPort calls primitives
- Their glue code is written in ST80 and translated into C; it calls driver fcns.
- The portable driver layer implements the module defined by the primitives
- Several Back-end Driver Interfaces

MIDI I/O Layers

MIDIPort instance methods

Interpreter glue code (ST -> C)

Portable C layer

Platform-specific C interface

Platform MIDI driver / libraries

Siren Performance

- Scores of real-time synthesis voices on lap-tops
- Full-bandwidth MIDI I/O
- Complex structure-editing GUIs (under development)
- Smalltalk-level Scheduler can flood MIDI (msec-level timing)
- Siren 2.2 is 200 Classes, 2400 Methods

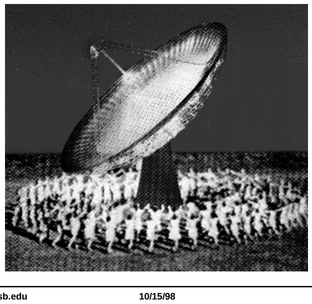
Future (1999)

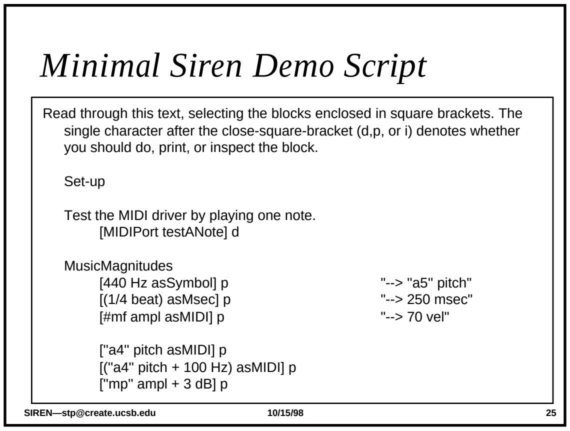
- Squeak clients on many platforms talk to DB “Stones” at CREATE
- Squeak translated synthesis, DSP, mixing (should rival C-based SWSS)
- Scalable tool/instrument—central resources and distributed access
- DRIVE, Creatophone, Paleo, Time-Machine, IDIOT, and IDP Projects

Future (2000+)

- ATM-based wide-area sound/music computing with end-user nodes at many bandwidths (GIOP/ATM to 10T)
- DB queries to and operations on very large score/sound databases
- Poly-channel I/O and pluriphonic projection from synth. SW or disks
- New tool paradigm

Siren





Minimal Siren Demo Script

Read through this text, selecting the blocks enclosed in square brackets. The single character after the close-square-bracket (d,p, or i) denotes whether you should do, print, or inspect the block.

Set-up

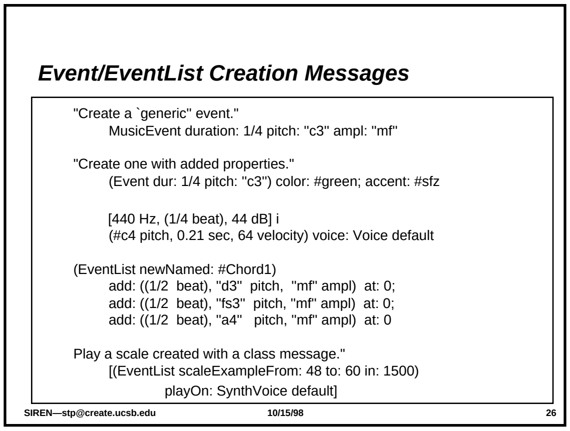
Test the MIDI driver by playing one note.

```
        [MIDIPort testANote] d
```

MusicMagnitudes

```
        [440 Hz asSymbol] p                        "--> "a5" pitch"
        [(1/4 beat) asMsec] p                      "--> 250 msec"
        [#mf ampl asMIDI] p                        "--> 70 vel"

        ["a4" pitch asMIDI] p
        [("a4" pitch + 100 Hz) asMIDI] p
        ["mp" ampl + 3 dB] p
```



Event/EventList Creation Messages

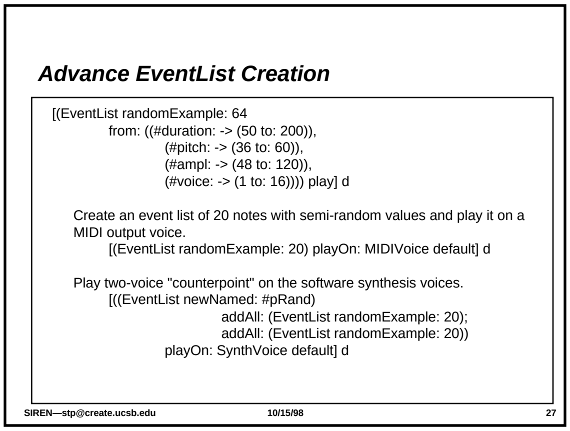
```
    "Create a `generic" event."
        MusicEvent duration: 1/4 pitch: "c3" ampl: "mf"

    "Create one with added properties."
        (Event dur: 1/4 pitch: "c3") color: #green; accent: #sfz


        [440 Hz, (1/4 beat), 44 dB] i
        (#c4 pitch, 0.21 sec, 64 velocity) voice: Voice default

    (EventList newNamed: #Chord1)
        add: ((1/2  beat), "d3"  pitch,  "mf" ampl)  at: 0;
        add: ((1/2  beat), "fs3"  pitch, "mf" ampl)  at: 0;
        add: ((1/2  beat), "a4"   pitch, "mf" ampl)  at: 0

    Play a scale created with a class message."
        [(EventList scaleExampleFrom: 48 to: 60 in: 1500)
                playOn: SynthVoice default]
```



Advance EventList Creation

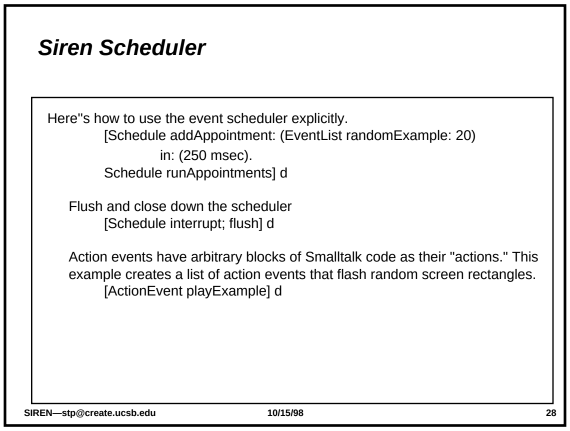
```
 [(EventList randomExample: 64
          from: ((#duration: -> (50 to: 200)),
                    (#pitch: -> (36 to: 60)),
                    (#ampl: -> (48 to: 120)),
                    (#voice: -> (1 to: 16)))) play] d
```

Create an event list of 20 notes with semi-random values and play it on a MIDI output voice.

```
        [(EventList randomExample: 20) playOn: MIDIVoice default] d
```

Play two-voice "counterpoint" on the software synthesis voices.

```
        [((EventList newNamed: #pRand)
                        addAll: (EventList randomExample: 20);
                        addAll: (EventList randomExample: 20))
                playOn: SynthVoice default] d
```



Siren Scheduler

Here"s how to use the event scheduler explicitly.

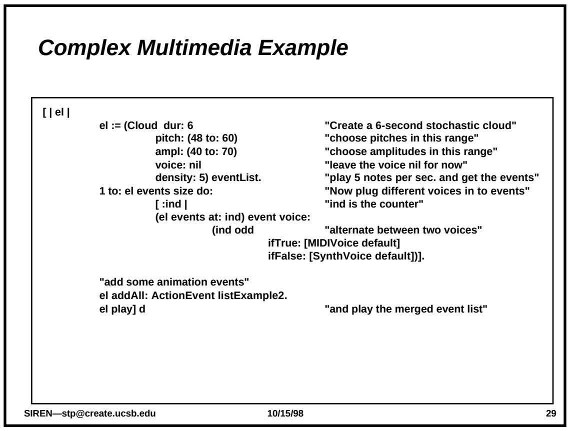
```
        [Schedule addAppointment: (EventList randomExample: 20)
                in: (250 msec).
        Schedule runAppointments] d
```

Flush and close down the scheduler

```
        [Schedule interrupt; flush] d
```

Action events have arbitrary blocks of Smalltalk code as their "actions." This example creates a list of action events that flash random screen rectangles.

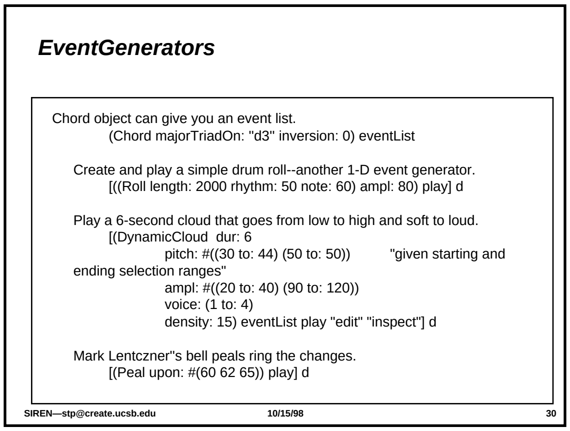
```
        [ActionEvent playExample] d
```



Complex Multimedia Example

```
[ | el |
        el := (Cloud  dur: 6                      "Create a 6-second stochastic cloud"
                pitch: (48 to: 60)                "choose pitches in this range"
                ampl: (40 to: 70)                 "choose amplitudes in this range"
                voice: nil                        "leave the voice nil for now"
                density: 5) eventList.            "play 5 notes per sec. and get the events"
        1 to: el events size do:                  "Now plug different voices in to events"
                [ :ind |                          "ind is the counter"
                (el events at: ind) event voice:
                        (ind odd                  "alternate between two voices"
                                ifTrue: [MIDIVoice default]
                                ifFalse: [SynthVoice default])].

        "add some animation events"
        el addAll: ActionEvent listExample2.
        el play] d                                "and play the merged event list"
```



EventGenerators

```
  Chord object can give you an event list.
        (Chord majorTriadOn: "d3" inversion: 0) eventList

    Create and play a simple drum roll--another 1-D event generator.
        [((Roll length: 2000 rhythm: 50 note: 60) ampl: 80) play] d

    Play a 6-second cloud that goes from low to high and soft to loud.
        [(DynamicCloud  dur: 6
                pitch: #((30 to: 44) (50 to: 50))          "given starting and
    ending selection ranges"
                ampl: #((20 to: 40) (90 to: 120))
                voice: (1 to: 4)
                density: 15) eventList play "edit" "inspect"] d

    Mark Lentczner"s bell peals ring the changes.
        [(Peal upon: #(60 62 65)) play] d
```

MIDI Control

```
"Demonstrate program change by setting up an organ instrument to play on."
          [MIDIPort testProgramChange] d
"Down-load a general MIDI patch for a 4-voice organ."
          [MIDIPort setupOrgan. MIDIPort default test] d

"Down-load a general MIDI patch for a 16-voice percussion ensemble."
          [MIDIPort setupTunedPercussion. MIDIPort testAScale] d

"Demonstrate control commands by playing a note and making a crescendo
    with the volume pedal."
          [MIDIPort testControlContinuous] d

"Demonstrate pitch-bend by playing two notes and bending them."
          [MIDIPort testBend] d
```

The Siren Graphics Framework

```
Display lines
          [DisplayListView example1] d

    Display rectangles
          [DisplayListView example2] d

    Open a sequence view on a random event list.
          [TimeSequenceView randomExample] d
```

Utilities

```
    "ANO"
          "MIDIPort allNotesOff"
    "Close down and clean up."
          "MIDIPort cleanUp"
```