The Siren Music/Sound Package for Squeak Smalltalk

Stephen Travis Pope
Center for Research in Electronic Art Technology, Dept. of Music
U. C. Santa Barbara, Santa Barbara, CA, 93106 USA
+1 805 967-2621
stp@create.ucsb.edu

ABSTRACT

The Siren system is a general-purpose music composition and production framework integrated with Squeak Smalltalk (1); it is a re-implementation of the Musical Object Development Environment (MODE) (5), the software component of the on-going "Interim DynaPiano" project. Siren is a Smalltalk class library (about 200 classes) for building musical applications; it runs on a variety of platforms with support for MIDI and audio I/O. Siren's source code is available for free on the Internet; see the Siren home page at the URL http://www.create.ucsb.edu/htmls/siren.html.

Keywords
Computer music, DSP, Smalltalk, multimedia, representation languages

THE SIREN SYSTEM

There are several elements to Siren:

- the Smoke music representation (music magnitudes, events, event lists, generators, functions, and sounds);
- voices, schedulers and I/O drivers (real-time and file-based voices for sound and MIDI I/O);
- user interface components for musical applications (UI framework, widgets, and tools); and
- several built-in applications (editors and browsers for Siren objects).

Siren's predecessors (The HyperScore ToolKit and the MODE) are documented in the book "The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology" (5), in papers in the Proceedings of the 1986, 1987, 1989, 1991, 1992, 1994, 1996, and 1997 International Computer Music Conferences (ICMCs), in an article on the Interim DynaPiano in *Computer Music Journal* 16:3, 1992, and in the book *Musical Signal Processing* (6) Most of these, and related documents are available from the URL http://www.create.ucsb.edu/~stp/publ.html.

The 1998 OOPSLA Siren poster concentrates on (a) the Smoke music description language, (b) the real-time MIDI and sound I/O facilities implemented using the Squeak Smalltalk-to-C translator and VM support, and (c) the Morphic-based GUIs for the 2.2 version of Siren. The demonstration will use an Apple laptop with MIDI and sound I/O.

The Smoke Music Representation Language

The "kernel" of Siren is the set of classes for music magnitudes, functions and sounds, events, event lists and event structures known as the Smallmusic Object Kernel (Smoke) music representation. Smoke is described in terms of two related description languages (music input languages), a compact binary interchange format, and concrete data structures. The high-level packages of Siren—voices, sound/DSP, compositional structures, and the user interface framework—interoperate using Smoke event lists. Smoke supports the following levels of description:

- abstract models of the basic musical quantities (scalar magnitudes such as pitch, loudness or duration);
- instrument/note (e.g., voice/event, performer/score) abstractions;
- sound functions, granular description, or other (non-note-oriented) description abstractions;
- flexible grain-size of "events" in terms of "notes," "grains," "elements," or "textures;"
- event, control, and sampled sound description levels;
- nested/hierarchical event-tree structures for flexible description of "parts," "tracks," or other parallel or sequential organizations;
- separation of "data" from "interpretation" (what vs. how in terms of providing for interpretation objects);
- abstractions for the description of "middle-level" musical structures (e.g., chords, clusters, or trills);
- annotation of event tree structures supporting the creation of heterarchies (lattices) and hypermedia networks;
- annotation including common-practice notation possible;
- description of sampled sound synthesis and processing models such as sound file mixing or DSP;
- possibility of building converters for many common formats, such as MIDI data, note lists, DSP code, or mixing scripts (this is an application issue); and
- possibility of parsing live performance into Smoke, and of interpreting it (in some rendition) in real-time (this is an application issue).

Smoke objects also have behaviors for managing several special types of links, which are seen simply as properties where the property name is a symbol such as *usedToBe*, *isTonalAnswerTo*, or *obeysRubato*, and the property value is another Smoke object, e.g., an event list. With this facility, one can built multimedia hypermedia navigators for arbitrary Smoke networks. The three example link names shown above could be used to implement event lists with version history, to embed analytical information in scores, or to attach real-time performance controllers to event lists, respectively.

The poster presentation will give extended examples of Smoke usage.

Siren I/O

The "performance" of events takes place via Voice objects. Event properties are assumed to be independent of the parameters of any synthesis instrument or algorithm. A voice object is a "property-to-parameter mapper" that knows about one or more output or input formats for Smoke data. There are voice "device drivers" for common file storage formats—such as cmusic note lists, the Adagio language, MIDI file format, or phase vocoder scripts—or for use with real-time schedulers connected to MIDI or sampled sound drivers. These classes can be refined to add new event and signal file formats or multilevel mapping (e.g., for MIDI system exclusive messages) in an abstract way. Voice objects can also read input streams (e.g., real-time controller data or output from a coprocess), and send messages to other voices, schedulers, event modifiers or event generators. This is how one uses the system for real-time control of complex structures.

Real-time music I/O in Siren is managed by Squeak primitive interfaces to sound and MIDI OS-level drivers. The glue code for these primitives is written in Smalltalk and translated to C for linking with the Squeak virtual machine (itself written in Smalltalk and translated). Several sets of primitives exist for Squeak on various platforms, including support for sound synthesis, digital audio signal processing, MIDI event-oriented and continuous controller I/O, and VM-level scheduling.

Navigator MVC in Siren

The Smalltalk-80 Model-View-Controller (MVC) user interface paradigm (2), is well-known and widely imitated. The traditional three-part MVC architecture involves a model object representing the state and behavior of the domain model—in our case, an event list or signal. The view object presents the state of the model on the display, and the controller object sends messages to the model and/or the view in response to user input.

"Navigator MVC" (4) (see the figure) is a factoring of the controller/editor and view for higher levels of reuse. The fundamental feature of this architecture is that all applications are built as display list editors (i.e., the generic tool is "smart draw"), with special layout manager objects for translating the model structure into a graphical display list representation and for translating structure interaction into model manipulation.

A StructureAccessor is an object that acts as a translator or protocol converter. An example might be an object that responds to the typical messages of a tree node or member of a hierarchy (e.g., What's your name? Do you have and children/sub-nodes? Who are they? Add this child to them.). One specific, concrete subclass of this might know how to apply that language to navigate through a hierarchical event list (by querying the event list's hierarchy). The role of the LayoutManager object is central to building Navigator MVC applications. Siren's layout manager objects can take data structures (like event lists) and create display lists for time-sequential (i.e., time running left-to-right or top-to-bottom), hierarchical (i.e., indented list or tree-like), network or graph (e.g., transition diagram), or other layout formats. The editor role of Navigator MVC is played by a smaller number of very generic (and therefore reusable) objects such as EventListEditor or SampledSoundEditor, which are shared by most of the applications in the system.

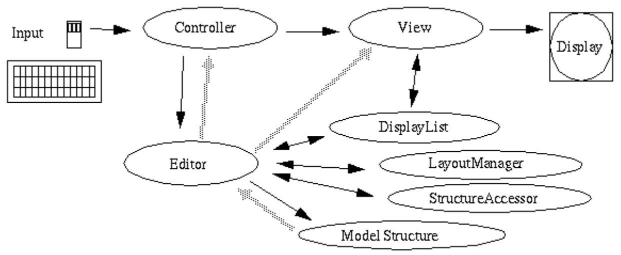

Figure: Navigator MVC Architecture

Much of the work of building a new tool within the MODE often goes into customizing the interaction and manipulation mechanisms, rather than just the layout of standard pluggable view components. Building a new notation by customizing a layout manager class and (optionally) a view and controller, is relatively easy. Adding new structure accessors to present new perspectives of structures based on properties or link types can be used to extend the range of applications and to construct new hypermedia link navigators. This architecture means that views and controllers are extremely generic (applications are modeled as structured graphics editors), and that the bulk of many applications' special functionality resides in a small number of changes to existing accessor and layout manager classes.

The Siren implementation of Navigator MVC is integrated with the Morphic (3) graphics framework.

REFERENCES

1. Ingalls, D., T. Kaehler, J. Maloney, S. Wallace, and A. Kay. "Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself " *Proc. OOPSLA 1997.*

2. Krasner, G. and S. T. Pope. "A Cookbook for the Model-View-Controller User Interface Paradigm in Smalltalk-80." *Journal of Object-Oriented Programming* 1(3), 1988.

3. Maloney, J. and Smith, R., "Directness and Liveness in the Morphic User Interface Construction Environment," *Proc. UIST '95*.

4. Pope, S. T., N. Harter, and K. Pier. "A Navigator for UNIX." *1989 ACM SIGCHI video collection.*

5. Pope, S. T. "An Introduction to MODE." in S. T. Pope, ed. *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology.* MIT Press, 1991.

6. Roads, C., S. T. Pope, G. DePoli, and A. Piccialli, eds. *Musical Signal Processing.* Swets & Zeitlinger, 1997.