Musical and Extra-Musical Applications of the NeXT Music Kit

David A. Jaffe
NeXT Computer Inc., 900 Chesapeake Drive, Redwood City, CA. 94063
phone: (415) 366-0900, Fax: (415) 780 3715, e-mail: david@next.com

Abstract: The NeXT Music Kit is a programmer's library of objects for DSP sound synthesis, MIDI processing, and music manipulations. Although oriented around the needs of music, its functionality is quite general and includes paradigms such as message sequencing and real-time signal processing. Previous ICMC papers described aspects of the Music Kit's structure and implementation. In this paper, the Music Kit is viewed from the standpoint of its use in a variety of domains, including algorithmic composition, real-time interactive DSP and MIDI performance, sound manipulation, DSP orchestration, psycho-acoustic research, teaching, multi-media data visualization, animation, and games. The live presentation of this paper focuses on demonstrations of Music Kit-based applications, while the written text looks in detail at the Music Kit mechanisms behind the scenes.

1. Pay No Attention to the Kit Behind the Screen

The Music Kit is a software package that generalizes the functionality needed by music and sound applications and makes those capabilities available in a set of reusable programming modules (Objective-C objects). The object-oriented approach makes it easy to develop applications without burdening the programmer with irrelevant details. At the same time, the Music Kit provides access to a fine level of detail and thus *enables* advanced applications to be created. The word "enable" is crucial here. The Music Kit is not itself an application and has no graphic user interface. (The sole exception, the ScoreFile Language, is a combination of data format and simple programming language oriented toward to the advanced user.) To the contrary, the Music Kit strives to avoid imposing its own bias as to what functionality an application implements and presents to the user. The goal is for the Music Kit to become invisible.

Since its introduction with the Release 1.0 NeXT Software, the Music Kit has been behind the scenes in a wide variety of applications in domains such as algorithmic composition, real-time interactive DSP and MIDI performance, sound manipulation, DSP orchestration, psycho-acoustic research, teaching, multi-media data visualization, animation, and games. We examine a few categories of applications. In each category, we discuss a specific application, and describe the Music Kit mechanism it uses.

2. Event Scheduling Applications

Many applications require future events to be scheduled and processed, while responding to incoming asynchronous events such as MIDI, keyboard and mouse input. The effect of the events may be to generate or modify sound or graphics or even to change the entire behavior of the application. In fact, there is no way to anticipate what an application may want to do in response to an event.

As an example, let us examine **Ringo**, a simple animated drum machine by Douglas Fulton. When **Ringo** is active, a cyclical pattern is synthesized on the DSP ad infinitum. The user can specify the pattern by selecting a set of buttons from a matrix, where each column represents a division of a beat and each row represents an instrument. Each selected button signifies a note that plays on the corresponding instrument at the corresponding position in the pattern. As each note plays, its button "lights up". Another row of animated "lights" above the buttons indicates the current beat. Let us look at how **Ringo** uses the Music Kit scheduling mechanism to synchronize animation and the synthesized sound.

The backbone of the Music Kit scheduler is the Conductor object. The Conductor abstracts the concept of sequencing, divorcing it from its strict association with MIDI. Each instance of Conductor is a scheduler with its own tempo, time offset, and scheduling queue, and each can be paused or resumed independently. The application schedules an event by making a request to a Conductor, asking it to send an Objective-C message to an object at a particular time in the future. For example, the Objective-C statement below is used by **Ringo** to ask a Conductor to send a message with the selector "noteOn:" to a DSP synthesis voice two beats from the present time, passing a particular Note object as an argument. **Ringo** also requests the Conductor to tell the button to blink at the same time:

```
[conductor sel:@selector(noteOn:) to:patch withDelay:2.0 argCount:1, note];
[conductor sel:@selector(blink) to:button withDelay:2.0 argCount:0];
```

A request may be rescinded any time before the scheduled event occurs. This has proven to be a crucial capability. For example, **Ringo** does this when a user suddenly disables a button whose note was already scheduled.

3. Event Processing Applications

We have seen how a general scheduling paradigm allows the Music Kit to conduct a performance without knowing what it is conducting. Similarly, the Music Kit provides a way for programmers to create modules that process or generate events without needing to know the origin or destination of the events, nor even the details of the events themselves.

An application that relies heavily on this mechanism is **Ensemble**, a programming example/demo shipped with the Release 2.0 NeXT software, by Michael McNabb. **Ensemble** provides a variety of basic functionality such as sequencing and DSP orchestration, but it excels particularly in the domain of real time interactive performance. Each **Ensemble** document describes how MIDI and score input are mapped, processed in real time, used to trigger algorithmic processes, and connected to DSP, MIDI and soundfile playback. The performer can switch documents under MIDI control, allowing him to completely reconfigure the applicatIon during a performance. In addition, **Ensemble** allows programming-literate users to write and dynamically load their own processing modules that respond to and schedule events in arbitrary ways. The author has used **Ensemble** in his collaborative performances with percussionist/composer Andy Schloss.

The Music Kit makes this flexibility possible by abstracting both the events themselves and the way events are passed between modules. The events themselves are represented as Note objects, each of which consists principally of an unordered collection of labeled variables (parameters). Because the variables have names, a module can pay attention to only those parameters that concern it, as well as add parameters that may be unfamiliar to other modules. Furthermore, a parameter's value may be a number, string or object, such as another Note object, adding an added dimension of flexibility.

Similarly, Notes are passed between modules in such a way that a given module needs know nothing about where the Note came from or where

it is going. This is accomplished by having a patchable network of Note-generating modules called "Performers", Note-processing modules called "NoteFilters", and Note-rendering modules called "Instruments". A Performer sends Notes to its outputs (NoteSender objects) without knowing anything about the NoteFilters or Instruments to which the output is connected. **Ensemble** does its work of processing real time MIDI by using chains of NoteFilters. In fact, each **Ensemble** document is primarily an archived chain of NoteFilters.

4. DSP Signal Processing Applications

Another important category of applications is those that produce or process a steady stream of data. While this data is typically sound, it may be any other form of sampled information such as is produced by scientific instruments. Such signal processing applications differ from event processing applications only in that the data is continuous and typically more dense. The Music Kit provides a modular unit generator-based synthesis model. We will take a close look at one application that illustrates particularly well the modular nature of the signal processing architecture.

SPASM by Perry Cook is a combination laboratory program and musical interface to a physical model of the human vocal tract, implemented on the NeXT DSP with the Music Kit. The model is built from UnitGenerator subclasses such as one-pole filters, connected together with "patchpoints" represented as SynthData objects. Where the Music Kit-supplied UnitGenerators did not provide the functionality he needed, Perry was able to build his own UnitGenerator as a subclass of the abstract UnitGenerator class and use the utility **dspwrap** to turn the DSP code into a form that the Music Kit can dynamically load into the DSP during a performance.

UnitGenerator objects have fairly high level interfaces. Oscillator frequencies are given in Hertz and numbers are in floating point format. For the most part, only when a programmer builds his own UnitGenerator does he have to "descend" to the level of DSP fixed-point arithmetic. However, due to the heavy computational demands of real time signal processing, there are some low level details, such as the notion of DSP memory spaces, that the Music Kit exports to the programmer to allow him to write optimal code. This allows him to design a DSP instrument that makes optimal use of fast on-chip DSP parallel memory move instructions. For example, **SPASM** can run multiple copies of the complex voice model in real time. In addition, because of its encapsulation in a Music Kit SynthPatch object, is MIDI controllable with no extra work on the part of the programmer.

5. Applications With Parametric Sound Effects

Games and data animation can be greatly enhanced by the use of *parametric sounds effects*, which we define as sounds that can be changed each time they are played. The ear quickly tires of hearing the same sound played at the same frequency with the same amplitude every time a graphic event occurs. In addition, matching the quality of the sound to the graphic event produces a more compelling simulation. The Music Kit is ideal for generating parametric sound effects. For example in **SortingInActionNoisily** (a Release 2.0 NeXT demo/example modified by the author to produce sound), sorting is animated and accompanied by a sound whose pitch and position in stereo space reflect the elements being swapped in each sort operation. Another example is **VOID**, a multi-player three-dimensional "space war" game from Lighthouse Design, Ltd. This application uses Music Kit parametric sounds to give each space ship and explosion a sound reflecting the distance and location of the ship and the severity of the explosion.

6. Smooth Animation Applications

Smooth animation is often built on a different model than scheduling-based animation. For example, in **LinesPlus** (a NeXT-supplied demo/example modified by the author to produce sound), the animation runs as fast as the CPU can go. The graphics consist of an animated set of connected lines whose end points move independently with randomly chosen direction and velocity and reflect when they hit the edges of the window. The accompanying sound consists of glissandi whose instantaneous frequency depends on the location of each corner of the polygon being animated. In this case, the Music Kit scheduler is not used to do the glissandi, since there's no way to know when the corners will reach their targets. Instead, the code that does the drawing also sends a frequency update to the Music Kit. This method of control, then, is similar to the MIDI approach to "continuous" controllers.

7. A Brief Survey of Other Application Areas

The Music Kit is well-suited to pedagogic applications. For example, **QuizzEar** by Douglas Fulton is an ear training program that combines the Sound and Music Kits. The application synthesizes a pitch and asks the student to sing a particular musical interval with respect to that pitch. It then records the student's voice and displays how close he was to the correct answer. **QuizzEar** also has a "Verdi mode" where the application plays a fragment from a Verdi opera and sonically highlights one interval somewhere in the middle. The student is asked to identify that interval. Other pedagogic applications include **Bessie** by Tony Holland, oriented toward teaching of FM computer music theory.

Various editors for Music Kit data types are beginning to appear, a welcome trend. Jean Larouche has done a graphic waveform editor in which the user can draw a waveform, spectrum, or filter and simultaneously hear the results synthesized with the Music Kit. Fernando Pablo López Lezcano has done a graphic envelope editor that accurately displays the effect of the asymptotic envelopes used by the Music Kit.

The Music Kit is often used as a way of producing traditional computer music tape pieces, but with lightning-fast turn around time. The standard computer music curriculum at CCRMA, Stanford, now uses **Common Music** by Rick Taube as a language for generating Music Kit scorefiles, which are then played instantly on the DSP and MIDI with the command line utility **playscore** (Release 2.0 NeXT Software). In addition, novel interactive pieces in which the listener can participate in the creation of the music are being created with the Music Kit. Applications in this area include Brad Garton's **Looching** and **Chaosakitty**. Chris Chafe is working on a Music Kit algorithmic composition programs based on iteration of non-linear functions.

Psychoacoustic experiments using the Music Kit have been designed by Dave Mellinger and others. One such program, designed by John Pirece and implemented by the author, measured the ear's sensitivity to repetition rate of tone bursts.

Applications for exploring tuning systems include **Just** by Bill Parod and **Temperament** by Mary Simoni.

Music notation programs include **Nutation** by Glen Diener and **MusicProse** by Coda Software, both of which use the Music Kit to play scores.

Many of these applications are available on archive servers. For example, the CCRMA archive server contains **Common Music** (cm.tarfile.Z), **SPASM** (SPASM.tar) and **ResoLab** (ResoLab.Z).

8. Summary

The Music Kit offers a high level programmatic interface to functionalities commonly used in music and other applications. It is based on a forward-looking design that uses a high-level representation of music in comparison with the more compact but weaker MIDI representation. With the recent generation of 040-based NeXT machines and the multi-DSP boards such as the Ariel QuintProcessor, the hardware has caught up with the software and the Music Kit is being extensively used in many areas.

9. References

Cook, P.R. "Identification and Control of Parameters in an Articulatory Vocal Tract Model..." Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford Univ. 1991.

Cook, P.R. "SPASM: a Real-Time Vocal Tract Physical Model Editor/Controller and Singer: the Companion Software Synthesis System," Colloque les Mod'eles Physiques Dans L'Analyse, la Production et la Cre'ation Sonore, ACROE, Grenoble, 1990.

Cook, P.R. "Synthesis of the Singing Voice Using a Physically Parameterized Model of the Human Vocal Tract," Proc. of the 1989 ICMC, pp. 69-72.

Cook, P.R. "TBone: An Interactive WaveGuide Brass Instrument Synthesis Workbench for the NeXT Machine," Proc. of the 1991 ICMC.

Diener, Glendon. "Nutation: Structural Organization Versus Graphical Generality In a Common Music Notation Program", Proc. of the 1989 ICMC, pp. 86-89.

Diener, Glendon. "Conceptual Integrity in a Music Notation Program", Proc. of the 1990 ICMC, pp. 86-89.

Diener, Glendon. "TTrees: a Tool for the Compositional Environment," "The Well-Tempered Object," M.I.T. Press., ed. Stephen Travis Pope.

Jaffe, David "Efficient Dynamic Resource Management on Multiple DSPs, as Implemented in the NeXT Music Kit." Proc. of the 1990 International Computer Music Conf., pp. 188-190.

Jaffe, David "An Overview of the NeXT Music Kit." Proc. of the 1989 ICMC, pp. 135-138.

Jaffe, David and Lee Boynton. "An Overview of the Sound and Music Kits for the NeXT Computer. " 1989. Computer Music Journal, 14(2):48-55. Also in "The Well-Tempered Object", M.I.T. Press.

McNabb, Michael. "Ensemble: An Extensible Real-Time Performance Environment." Proc. 89th Audio Engineering Society Convention. Los Angeles, CA, 1990.

Pierce, John R. "Periodity and Pitch Perception". Journal of the Acoustical Society of America, Oct. 1991. Vol. 89.

Smith, Julius O., David Jaffe, and Lee Boynton. "Music System Architecture on the NeXT Computer. Proc. 1989 Audio Engineering Society Conference. Los Angeles, CA, 1989.

Smith, Julius O. "Unit-Generator Implementation on the NeXT DSP Chip". Proc. 1989 ICMC, pp. 303-306.

Taube, H. "Common Music–A Compositional Language in Common Lisp and CLOS". Proc. 1989 ICMC, pp. 316-319.