Next Steps from NeXTSTEP: MusicKit and SoundKit in a New World

Stephen Brandon
Department of Music
University of Glasgow
e-mail: S.Brandon@music.gla.ac.uk

Leigh M. Smith
tomandandy Music Inc.
New York, NY
e-mail: leigh@tomandandy.com

This paper describes the new implementation and port of the NeXT MusicKit, and a clone of the NeXT SoundKit—the SndKit, on a number of different platforms, old and new. It will then outline some of the strengths and uses of the kits, and demonstrate several applications which have made the transition from NeXTSTEP to MacOS-X and WebObjects/NT.

1. Introduction

Apple's purchase of NeXT Inc. has not spelled the end of the Objective-C language, the NeXTSTEP programming environment, nor the OO kits that paved the way for the MusicKit and SoundKit. Instead, the OpenStep API which grew out of NeXTSTEP has been incorporated into Apple's latest operating systems. Furthermore, the open source (GPL) implementation of OPENSTEP's two key object frameworks (FoundationKit and AppKit) is close to completion (http://www.gnustep.org), and runs on multiple platforms (see figure 1).

The pedigree of the MusicKit should be well-known to ICMC attendees during the 1990s. In brief, it provides an abstraction of the scoring, timing, and performance of musical data. On NeXTSTEP, it could utilise a DSP chip for real-time synthesis, as well as handling real-time MIDI streams. Its strengths however were not limited to real-time performance: its comprehensive internal structure was utilised in many other applications requiring musical data representation.

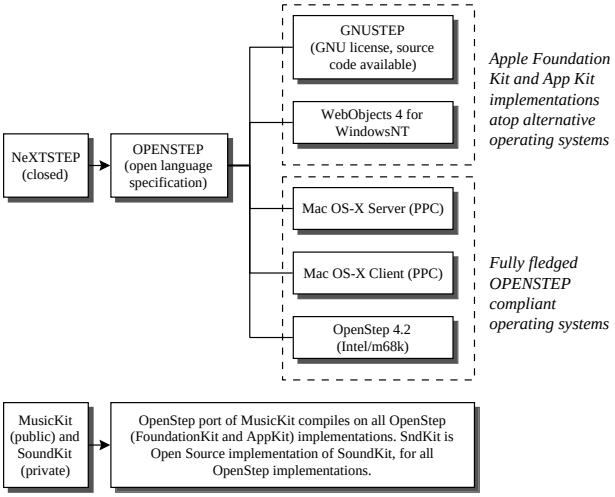

Figure 1: Lineage of OPENSTEP and MusicKit

2. The MusicKit at Glasgow University

The need to upgrade the computer network at Glasgow University in 1998-99 from aging NeXT computers prompted an examination of which applications and technologies were most important to our research and teaching efforts. It was seen that we had a heavy investment in teaching software based on the MusicKit, and in the music notation software Calliope.app, which also relies in part on the MusicKit.

Looking ahead to the future possibilities of OpenStep, we could see justification for porting the MusicKit and other key applications such as Calliope.app to OpenStep. An OpenStep port would also insulate us against any particular platform decision, given its cross-platform nature.

3. Porting the MusicKit to OpenStep

The NeXTSTEP/Intel version of the MusicKit contains drivers for various sound cards which support the MusicKit DSP functions. The NeXTSTEP versions of the drivers ran without modification on OpenStep/Intel, simplifying the porting and testing of the kit. OpenStep/Intel was likely to be the only OpenStep architecture which supports the hardware DSP functionality, so it made sense to use that platform for initial porting and testing.

3.1 Language changes

OpenStep makes some major changes to NeXTSTEP at a foundational level. The major change is the adoption of object reference counting, a step in the direction of automatic garbage collection (cf Java). OpenStep also defines more efficient classes for dealing with filenames and other strings (replacing char*), for arrays and hashtables (NSArray, NDictionary), and for arbitrary blocks of data (NSData).

All the public MusicKit classes are now prefixed with MK to match the Foundation/AppKit model, e.g. MKNote, MKOrchestra. In a similar manner to the changes in other Frameworks when OpenStep-ified, as well as name changes, there are object allocation changes, returning auto-released objects e.g. [Midi new] has been renamed [MKMidi midi] in keeping with OpenStep conventions. `allocFromZone:onDevice:` and `allocFromZone: onDevice:hostName:` have been replaced with corresponding `-initOnDevice:` and -initOnDevice: hostName: instance methods and +midiOnDevice: +midiOnDevice:hostName: class methods to support OpenStep allocation conventions.

Most new Foundation classes have been adopted for use, with the exception of the retention of the char* type for dealing with MusicKit Scorefiles. The scorefile parsing engine had been heavily optimised, and the ability of the new string handling objects (NSString) to handle Unicode strings was not considered a good enough reason to sacrifice parsing speed. The parser does however now recognise alternative line ending characters.

Some additional classes have been added:
MKSamplerInstrument enables arbitary soundfiles to be played back in emulation of a hardware sample playback synthesiser. *MKMixerInstrument* allows scorefile scripted mixing of sound files.

3.2 Static and Dynamic Libraries

In line with OpenStep programming practice, the MusicKit now builds as run-time dynamic libraries, called frameworks, rather than as a static library which is linked against at compile time. This reduces binary sizes of applications using the MusicKit. In fact, there are now several frameworks in the MusicKit: MusicKit.framework, MKUnitGenerators.framework, MKDSP.framework and MKSynthPatches.framework. Platforms not supporting the dsp, synthpatch and unitgenerator frameworks naturally do not require them. The platform-specific MKPerformSndMIDI.framework is also required for programs and applications requiring performance.

Dynamic libraries ("dll"s on Windows) are traditionally a source of incompatibility and grief due to changes in the API over time. Apple's frameworks however contain a versioning system which can allow previous versions of the framework to live in the same package, and applications that must link against a particular version are free to do so. It is the aim of the developers to utilise versioning in an effective manner.

3.3 Audio and MIDI I/O on Various Platforms

MacOS-X Server: The original NeXT 68K MusicKit MIDI driver has been ported to MacOSX-Server (PowerPC) by the second author. The MusicKit now uses the SndKit for audio I/O. As the audio implementation of MacOS-X is in a state of flux at the time of writing, the use of SndKit limits the impact of change on the rest of the MusicKit.

Windows98/NT (with WebObjects): A Windows framework has been developed (MKPerformSndMIDI) by Leigh Smith utilising Microsoft DirectMusic/DirectSound which closely emulates the Mach (kernel in OpenStep and MacOs-X) MIDI functionality of the original drivers.

Linux/GNUSTEP: A series of stubs are in place for a MKPerformSndMIDI framework to be written to interface with native Linux audio and MIDI APIs. We eagerly await developments in the Linux audio arena before proceeding with this project.

4. The SndKit

In late 1998 it looked increasingly likely that NeXT's SoundKit would be deprecated, thereby orphaning a number of classic NeXTSTEP applications relying on the Sound class, and the sound-editor-in-a-box SoundView class. The author therefore wrote a clean-room implementation called "SndKit", focusing on the non-audio-I/O related classes and functions of the SoundKit. It therefore implements soundfile reading and writing; the Snd class; format, channel and sample rate conversion; and the SndView class. Through `#ifdef` statements, the SndKit implements native audio I/O methods on whichever platform it is compiled on (OpenStep4.2, Windows, MacOS-X).

Not long afterwards, Apple released the source code of the original SoundKit under its APSL license; the MusicKit has adopted the SndKit regardless.

4.1 The SndView Class

The new SndView class implements the entire API of SoundView, and adds some significant new functionality. The level of zooming has been increased to an arbitrary figure of one sample to more than 30 horizontal pixels. At this level of zoom, vertical cross-hairs are placed to show exact sample locations. Rudimentary multi-channel support (greater than 2-channel) is built in internally, and in future more than two channels will be able to be edited and displayed. Caching of display data has been heavily optimised, is now configurable, and is memorised internally with cache objects even while a portion of the SndView has been scrolled out of sight.

4.2 SndKit Omissions

The deprecation of the SoundKit by Apple appears to have been in order to pave the way towards QuickTime becoming the underlying media and I/O framework on MacOS-X. There is therefore room for object designers to package QuickTime I/O calls into higher level frameworks. Because the QuickTime API is also available on Windows, this may also provide an easy way of creating cross-platform object oriented audio API, although on Windows this may then imply an extra level of abstraction and latency.

A full implementation of audio I/O and monitoring in the SndKit therefore needs to address multiple channel full duplex I/O, as well as "SndTracker" capability, where UI objects can respond to audio levels moving through the system. In order to attract serious audio developers, latency must be kept to a bare minimum.

5. Framework Dependencies

The MusicKit does not depend on the AppKit. Therefore, it can be utilised by command-line and other non-GUI applications. This also makes certain MusicKit applications very portable onto GNUSTEP platforms where libFoundation (the GNUSTEP version of the FoundationKit) is well developed. This includes Microsoft Windows *without* WebObjects. This also brings up the possibility of utilising the MusicKit for interesting interactive web applications on WebObjects, either on MacOS-X, Windows NT, HP-UX or indeed Solaris.

The SndKit, because it contains GUI code to support the SndView object, has a dependency on the AppKit. This inhibits its utility in WebObjects and on GNUSTEP platforms where the graphical portions of the AppKit are not yet so well developed. In some situations this may also cause a problem in creating command-line or other non-graphical applications using the SndKit on Mach-based platforms.

6. Strengths and Utility

The MusicKit provides a set of music representation classes such as MKNote, MKPart and MKScore, as well as MKEnvelope, MKWaveTable and MKTimbre that provide a flexible high-level structure for musical data. The MusicKit also provides a number of file formats for storing these structures. ScoreFile is a simple text-based scripting language that allows musical data to be represented in a convenient, human-readable form and supports simple programming structures. The MusicKit also supports reading and writing musical data as Standard MIDI files, binary scorefiles and OPENSTEP archived objects.

The MusicKit makes scheduling and sequencing extremely simple. A MusicKit "performance" consists of sending scheduled Objective-C messages and handling asynchronous events such as incoming MIDI and OPENSTEP events. The MKConductor class is in charge of dispatching all messages and managing the notion of time. A MKConductor may have a tempo, a time map, or may be set to synchronise to MIDI time code. Since all time control is managed in the MKConductor itself, the difference between these time representations is transparent.

Another aspect of the MusicKit performance is a dynamically patchable MKNote handling network consisting of three classes, MKPerformer, MKNoteFilter and MKInstrument. MKPerformer subclasses are MKNote generators, that sequence over NSArrays of MKNotes or create MKNotes on the fly. They contain outputs that may be connected to MKNoteFilters or MKInstruments. MKInstrument subclasses realise MKNotes in some manner, for example by playing them on the DSP or via MIDI, and contain inputs that may be connected to MKNoteFilters or MKPerformers. MKNoteFilters are intermediate processors that contain both inputs and outputs. This scheme makes it easy to create a performance scheme for nearly any application. The inputs and outputs are represented as MKNoteSenders and MKNoteReceivers, respectively.

The MusicKit's performance apparatus is based on the notion that messages execute quickly. Thus, time stays constant during the execution of a scheduled or event-triggered message. This has the advantage of allowing a large number of messages to happen at exactly the same time. The MIDI and DSP drivers support time-ordered queues of events, thus allowing the application some latitude in computing these updates, while still providing an instantaneous execution of the updates themselves.

7. License and Availability

7.1 License

The MusicKit is an open source code release, with the exception of the NeXT hardware implementation of the low-level sound and DSP drivers. Researchers and developers may study the source or even customize the MusicKit and DSP Tools to suit their needs. Commercial software developers may freely incorporate and adapt the software to accelerate development of software products. To encourage widest use, the license is similar in philosophy to FreeBSD, in contrast to Gnu Public License (GPL) or LGPL.

7.2 Obtaining the MusicKit

Since July 1999, the MusicKit has been actively maintained by Dr. Leigh Smith of tomandandy music Inc. <leigh@tomandandy.com> via a closed CVS system. The distribution is obtainable on the net from http://www.tomandandy.com/MusicKit. The distribution is also listed on the OpenStep resource portal SoftTrak http://www.stepwise.com/Apps/WebObjects/Softrak.
Enhancements can be sent to <leigh@tomandandy.com> to have them incorporated for future releases.

8. Some SndKit and MusicKit Applications

8.1 Spectro.app

Gary Scavone's spectral analysis tool (spectrum and waterfall graphs) relied heavily on Sound and SoundView (now Snd and SndView) objects. The port from NeXTSTEP to OpenStep took two days—now runs on OpenStep4.2 and WebObjects / WinNT. Spectro.app should compile out of the box on MacOS-X.

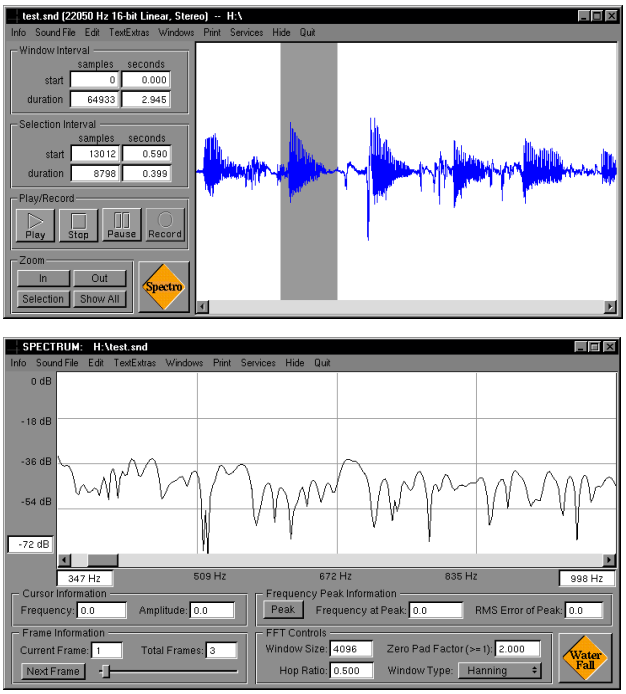

Figure 2: Spectro4.app running on Windows, showing SndView in action

8.2 TwoWaves.app

This is a rewrite, rather than a port, of a NeXTSTEP classic. Originally it used the MusicKit DSP routines to generate two waves, with parameters adjustable by the user in real time. With the demise of MusicKit DSP, and the speed of current processors, sound is now generated on the fly and pre-mixed into a third Snd object for playback. Compiles on OpenStep4.2, WebObjects/NT and MacOS-X.

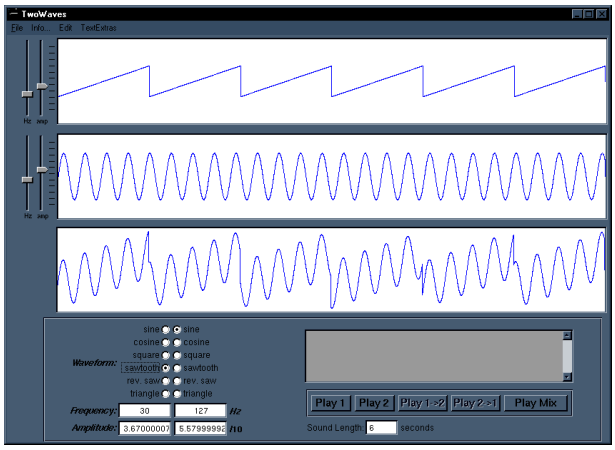

Figure 3: TwoWaves.app on Windows, showing three SndViews

8.3 rt.app

Paul Lansky's playlist-based real-time sound mixer. The "driver" portion of the application utilises the NeXT SoundKit audio streaming functions, so the application currently only runs on

OpenStep4.2. The port to MacOS-X will rely on low level audio API becoming available.

8.4 Calliope.app

This major port of Dr William Clocksin's notation application has been vital at Glasgow University. Although Calliope does not rely heavily upon the MusicKit, the MIDI I/O options require it. Calliope runs on NeXTSTEP, OpenStep4.2 and partially on WebObjects / NT. A MacOS-X port is planned.

See http://www.CL.cam.ac.uk/users/wfc/calliope.html

8.5 NoteAbility.app

Keith Hamel's notation application relies heavily on the MusicKit, and is being ported to MacOS-X. [Hamel, 1994]

See http://debussy.music.ubc.ca/~opus1

8.6 Example apps in MusicKit Distribution

The MusicKit distribution includes several command line and graphical (AppKit based) utilities that demonstrate and exercise the majority of classes in the MusicKit.

Other commercial grade music applications are also currently in development utilising the MusicKit.

9. Contributors and History

David A. Jaffe and Julius O. Smith III at NeXT did the original design, with David coding the Objective C and Julius most of the 56K DSP. Their original design appeared in [Jaffe, Boynton 1989]. The Ensemble application and much of the SynthPatch library were written by Michael McNabb. Douglas Fulton had primary responsibility for the documentation. Dana Massie, James A. Moorer, Lee Boynton, Greg Kellogg, Douglas Keislar, Michael Minnick, Perry Cook, John Strawn, Rob Poor and Roger Dannenberg made code and design contributions also. Following NeXT's release of the source to Stanford in 1994, David did the port to Intel NeXTSTEP and the MPU-401 MIDI and DSP drivers. There were some other bug fix contributors (acknowledged in code comments).

Stephen Brandon <sbrandon@music.gla.ac.uk> did the initial OpenStep port in early 1998 and the majority of the conversion work. Leigh Smith <leigh@tomandandy.com> fixed some bugs and ported the MusicKit and MIDI drivers to Intel and PowerPC Rhapsody in late 1998 then reorganised the packages and documentation for MacOS-X Server. The port from Rhapsody to MacOS-X Server was trivial. The frameworks were then ported to Windows 98/NT using DirectMusic. The MusicKit now uses the SndKit, rather than the SoundKit for its sound processing. Keith Hamel tested and bug fixed the MacOS-X Server version. Raphael Sebbe contributed changes to SndKit to port to MacOS-X.

Bibliography

Hamel, K. 1994. "NoteAbility: A Music Notation System That Combines Musical Intelligence With Graphic Flexibility" *Proceedings of the 1994 International Computer Music Conference,* pp.303-306. Aarhus, Denmark: Computer Music Assoc.

Jaffe, D. 1989. "An Overview of the NeXT Music Kit" *Proceedings of the 1989 International Computer Music Conference*, pp.135-138. Columbus, Ohio: Computer Music Assoc.

Jaffe, D. 1991. "Musical and Extra-Musical Applications of the NeXT Music Kit" *Proceedings of the 1991 International Computer Music Conference*, pp.521-524. Montreal, Canada: Computer Music Assoc.

Jaffe, D. and Boynton, Lee R. 1989. "An Overview of the Sound and Music Kits for the NeXT Computer" *Computer Music Journal* 13(2):44-55, 1989

Smith III, Julius and Jaffe, D. and Boynton, Lee R. 1991. "Music System Architecture on the NeXT Computer" *Proceedings of the Audio Engineering Society Conference*, Los Angeles, CA.