Editor: Christof Ebert ■ Vector Consulting ■ christof.ebert@vector-consulting.de

CLAM: A Framework for Audio and Music Application Development

Xavier Amatriain, *University of California, Santa Barbara*

Music and open source? Are there any connections between these two? After several years of describing tools, middleware, services, and applications up the open software systems stack, let's look now at frameworks. CLAM is an OSS framework for processing audio and music. It lets developers create and manage complex audio effects and applications on top of them, whether for experiments, research, or business. And it's built on OSS libraries. Xavier Amatriain highlights the major features of this award-winning framework and points to a variety of links for further information. Let there be music!
—Christof Ebert

The importance of visual and graphical application frameworks such as MVC (model-view-controller) and Interviews to the development of software frameworks is well known. But the audio and music computing field also has a long, rich history of frameworks for application builders. And it's here we find CLAM, winner of the 2006 ACM Open Source Multimedia Software Competition.

CLAM stands for C++ Library for Audio and Music, a name that doesn't reflect its current status as a full-fledged framework. CLAM originated in an effort to organize a repository of audio-processing algorithms, and the early development goal was simply to offer a class library. But it soon became obvious that the project was of interest to a much broader audience. CLAM adopted the GNU General Public License (GPL) and became public in the context of the AGNULA (A GNU/Linux Audio) distribution.

What is CLAM?

CLAM is a C++ framework that offers a complete R&D platform for the audio and music domain. It includes an abstract model for audio systems, a repository of processing algorithms and data types, and a number of tools such as audio and MIDI (musical instrument digital interface) I/O. Developers can exploit all these features to build cross-platform applications or rapid prototypes for testing signal- and media-processing algorithms and systems. Everything in CLAM—from the low-level, operating-system access primitives to the GUI—is cross-platform and regularly compiled under several Linux distributions as well as Mac OS X and Windows.

The framework also offers several standalone applications, including

- SMSTools, a graphical tool for audio analysis, synthesis, and transformation;
- Annotator, an application for semiautomatic music analysis that includes goodies such as automatic chord recognition;
- Salto, a real-time brass-instrument synthesizer; and
- Network Editor, a visual-building tool.

The Network Editor has become CLAM's flagship application. It lets developers dynamically patch process graphs by inserting ready-to-use blackbox components in an application through

 0740-7459/07/$20.00 © 2007 IEEE

Table 1

Open source audio and music environments

Features	CLAM	Jsyn	Marsyas	Open Sound World	Pure Data	STK (Synthesis ToolKit in C++)
Language	C++	Java	C++	C++	C	C++
Can be used for stand-alone application development	Yes	Yes	Yes	Yes	No	Yes
Visual-building application	Yes	Yes	No	Yes	Yes	No
Description	Complete software framework for developing audio and music applications. Offers a conceptual metamodel as well as tools and a visual builder.	Audio synthesis API for Java application developers dealing with computer music. You can use Jsyn to create applets for Web pages.	Conceptually similar to CLAM but with fewer tools and a focus on music-information retrieval and audio analysis.	Extensible programming environment for interactive audio processing and synthesis. It allows for visual patching and high-level C++ expressions.	Graphical programming environment used mainly for artistic purposes. It has a large user base and is similar to the commercial Max/MSP package.	Software toolkit for audio synthesis, especially physical modeling. Doesn't depend on external libraries.
Web page	www.clam.iua.upf.edu	www.softsynth.com/jsyn/	http://opihi.cs.uvic.ca/marsyas	http://osw.sourceforge.net	http://puredata.info	http://ccrma.stanford.edu/software/stk

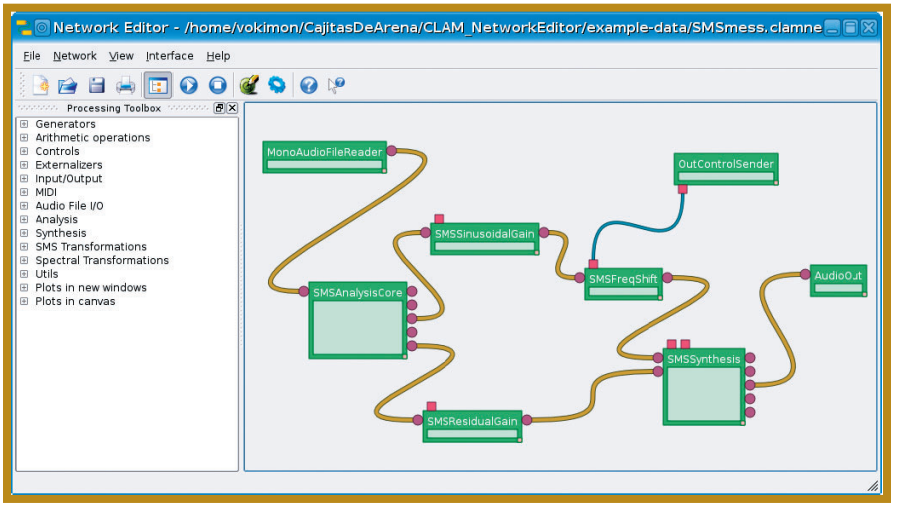

Figure 1. The CLAM Network Editor. Developers select processing objects, which are CLAM's atomic units, from the left menu categories. Then they place and connect the objects in the canvas to build a processing graph or network. Brown connections represent data flows; blue connections represent event-driven control flows.

a GUI (see figure 1). Furthermore, you can use the resulting XML file to create stand-alone applications without writing a single line of code.

CLAM isn't the only open source environment for audio and music development. Table 1 compares it to several other environments. Each environment focuses on a particular aspect of the audio domain and how to present it to the user. In many senses, CLAM is a superset of the other environments because it offers a metamodel and tools for all possible applications, ranging from audio analysis to synthesis and real-time processing. You can also use it as a complete application development framework, a class library, or a rapid-prototyping environment with a GUI.

Audio and music open source libraries

Figure 2 shows the components of a typical audio application. To offer such a wide range of services, CLAM relies on third-party open source audio tools. CLAM and its metamodel act as a gluing point and offer a common service interface.

Streaming audio to and from the sound card is one of the most demanding audio-application services. If you need to offer it both efficiently and across different platforms, the task can be daunting. Fortunately, several open source solutions exist. The most well-known and widely used is PortAudio, a complete toolkit for cross-platform audio input and output. RtAudio is a similar library but more modest in its scope.

Handling sound files is a similarly complex service. Supporting the many available formats (each with its own bit

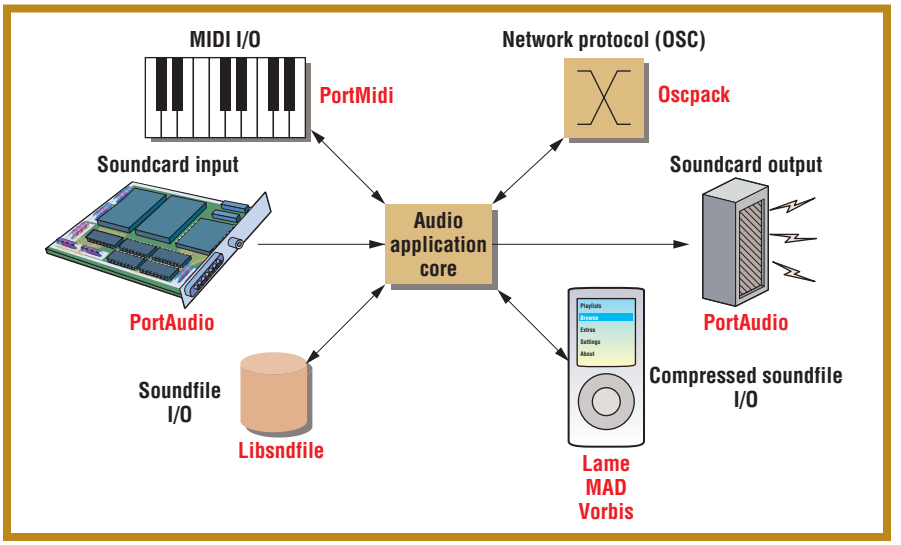

Figure 2. A typical audio application. Generic services are in red font; open source libraries that can provide the service are in black font.

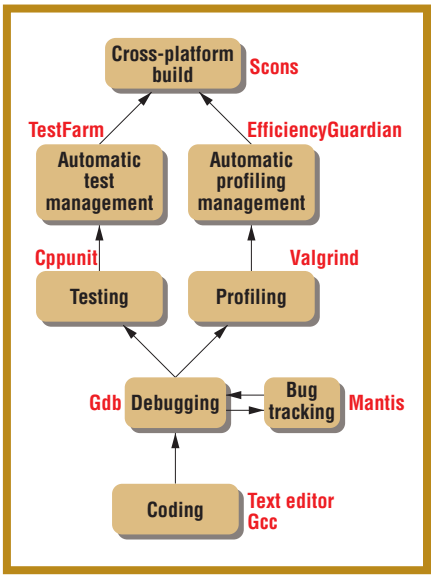

Figure 3. CLAM process activities (in black) and tools to support them (in red).

rates, encodings, and so on) would be almost unthinkable without relying on an existing library. Fortunately, libsndfile supports virtually any sound file format. But to support compressed formats, you'll need to look elsewhere. For MPEG1 (aka MP3) and MPEG2 decoding, you can use either Underbit's MPEG Audio Decoder (MAD) or the mpg123 library; for encoding, refer to Lame. If you'd rather support a patent-free format such as ogg, the libvorbis library will give you all the needed support for that compression technology, which many listeners consider superior to MP3.

Other typical audio-related services that you might need are support for ID3 metadata tags (offered by ID3lib or MAD), for MIDI I/O (see PortMidi), and for the Open Sound Control networking protocol, which is enabled by several implementations such as Oscpack in C++.

Open source tools and libraries to build CLAM

We've developed CLAM entirely using open source tools and third-party libraries that relate not only to our particular domain but also to general-purpose domains. Although the framework is completely multiplatform, we prefer to develop on Linux using open source tools. It's beyond this column's scope to describe all the tools and decisions involved, but figure 3 should give an overall picture of the process and how it's supported.

We had to overcome three nonaudio-related open source challenges. The first one was storage. Apart from the audio-specific storage formats, applications need to store different data such as configurations, metadata, or process outputs. CLAM supports this requirement by basing all data storage on the popular XML format. In CLAM, every object in memory has automatically built-in XML persistency. If your framework or application needs cross-platform open source XML support, you should look at libxml and XercesC. We started using XercesC because it was more developed and well supported in all platforms at the time. However, we've since switched to libxml for its greater efficiency and stability. If your application requires only event-based XML processing, take a look at Arabica, a toolkit that provides uniform access to a number of libraries including libxml and XercesC.

Second, we needed to build efficient cross-platform GUIs, and we needed to rely on an existing toolkit. We first chose Fltk, a light toolkit that can yield good results. But after the latest release of Qt from Trolltech, we've made it our preferred toolkit. Once Trolltech solved the problem of a GPL-compatible license for MS Windows, Qt stood out as a superior choice.

Finally, we also needed an open source solution to fast-Fourier-transform computing. Performing FFTs efficiently is a major issue in many signal-processing-intensive software packages. Arguably, the most reliable open source package for the task is the Massachusetts Institute of Technology's FFTW. It's offered on a dual-licensing scheme. You can use FFTW under the GPL conditions or link it into proprietary software by paying a licensing fee. If the license fee is an issue, you might want to check out the Ooura FFT implementation, which can be almost as efficient in most cases. CLAM supports both.

As you can see, the audio and music field is very active in the open source arena, where application frameworks also flourish. In both these contexts, CLAM is a long-term, relatively successful OSS project that's used open source tools exclusively in its development. Even before its first stable 1.0 release, CLAM has already proved its potential and quality. A framework's success tends to depend more on the users than on the quality of the development itself. Nevertheless, if offering good quality software is the first step in attracting open source users, the almost 200 registered users of the framework should grow exponentially in the near future.

Related Web Sites

CLAM: www.clam.iua.upf.edu

Distributions and open source audio repositories
- **AGNULA**: www.agnula.org
- **Linux Sound**: http://linuxsound.org
- **Planet CCRMA**: http://ccrma.stanford.edu/planetccrma/software

CLAM-related environments
- **CREATE Signal Library**: www.create.ucsb.edu/CSL
- **JSyn**: www.softsynth.com/jsyn/
- **Marsyas**: http://opihi.cs.uvic.ca/marsyas
- **Open Sound World**: http://osw.sourceforge.net
- **Pure Data**: http://puredata.info
- **Sound Object Library**: http://music.nuim.ie/musictec/SndObj
- **Synthesis Toolkit in C++**: http://ccrma.stanford.edu/software/stk

Audio libraries
- **ID3lib**: http://id3lib.sourceforge.net
- **Lame**: http://lame.sourceforge.net
- **Libsndfile**: www.mega-nerd.com/libsndfile
- **Libvorbis**: http://xiph.org/vorbis
- **MPEG Audio Decoder**: www.underbit.com/products/mad
- **Mpg123**: www.mpg123.de
- **Oscpack**: www.audiomulch.com/~rossb/code/oscpack
- **PortAudio**: www.portaudio.com
- **PortMIDI**: www.cs.cmu.edu/~music/portmusic
- **RtAudio**: www.music.mcgill.ca/~gary/rtaudio

General interest libraries and tools
- **Arabica**: www.jezuk.co.uk/cgibin/view/arabica
- **Bugzilla**: www.bugzilla.org
- **Doxygen**: www.stack.nl/~dimitri/doxygen
- **EfficiencyGuardian**: http://sourceforge.net/projects/efficiencyguard
- **FFTW**: www.fftw.org
- **Fltk**: www.fltk.org
- **Gforge**: http://gforge.org
- **Libxml**: http://xmlsoft.org
- **Mantis**: www.mantisbt.org
- **Ooura FFT**: http://momonga.t.utokyo.ac.jp/~ooura/fft.html
- **Scons**: www.scons.org
- **Testfarm**: www.iua.upf.es/~parumi/testfarm
- **Trolltech's Qt**: www.trolltech.com/products/qt
- **XercesC**: http://xml.apache.org/xercesc

Acknowledgments

CLAM is a team effort that throughout the years has included around 20 different contributors with a stable core development team of three or four developers. At this point, Pau Arumí and David García anchor the team in Barcelona with me in Santa Barbara. The project has received several grants, and the Catalan government is supporting its current development.

I thank Christof Ebert for his support in writing this column.

Xavier Amatriain is research director of the Center for Research in Electronic Art Technology at the University of California, Santa Barbara. Contact him at xavier@create.ucsb.edu; www.create.ucsb.edu/xavier.