CLAM: An OO Framework for Developing Audio and Music Applications

Xavier Amatriain
Music Technology Group
Pompeu Fabra University
Barcelona, Spain
xamat@iua.upf.es

Maarten de Boer
Music Technology Group
Pompeu Fabra University
Barcelona, Spain
mdeboer@iua.upf.es

Enrique Robledo
Music Technology Group
Pompeu Fabra University
Barcelona, Spain
erobledo@iua.upf.es

David Garcia
Music Technology Group
Pompeu Fabra University
Barcelona, Spain
dgarcia@iua.upf.es

ABSTRACT
CLAM (C++ Library for Audio and Music) is a framework for audio and music programming. It may be used for developing any type of audio or music application as well as for doing more complex research related with the field. In this paper we introduce the practicalities of CLAM's first release as well as some of the sample application that have been developed within the framework. See [1] for a more conceptual approach to the description of the CLAM framework.

Keywords
Development framework, DSP, audio, music, GPL, free software

1. INTRODUCTION
The Music Technology Group of the Pompeu Fabra University is a research group where more than forty engineers and programmers are involved in different projects in the area of sound and music[2]. Although different tools are used most of the research (and of course all the applications) is done programming directly on C++.

Two years ago, it was clear that the amount of lines of code and projects related with those were becoming hardly manageable. Although the code had been written using an OO language, it was highly unstructured and the result of applying kludge after kludge. The task of getting a newcomer to understand what was going on was almost unbearable.

The initial objective of the CLAM project was "To offer a complete, flexible and platform independent Sound Analysis/Synthesis C++ platform to meet current and future needs of all MTG projects." (quoted from CLAM's first Working Draft). Those initial objectives have slightly changed since then, mainly to accommodate to the fact that the library is no longer seen as an internal tool for the MTG but as a library that will be licensed under the GPL (GNU Public License) terms and conditions[3] in the course of the Agnula IST European Project[4]. Agnula (A GNU Linux Audio distribution) plans on offering a complete Linux distribution, both in Debian and RedHat versions, focused on promoting the use of free software for audio and music.

2. HOW IT WAS MADE
CLAM is programmed in C++. Code is regularly compiled with gcc in Linux, Microsoft and Intel compilers in Windows and Code Warrior in Windows and MacOS. We have had the chance to test how differently compilers behave and how bad most of them adjust to the ANSI ISO standard.

All this is specially true when it comes to the use of the most recent C++ features, such as templates, and related techniques, such as template metaprogramming. This techniques where initially considered as potentially useful in the CLAM framework, but this lack of language support in most compilers, together with the need of optimizing the compiling speed of the library, has led to a rather scarce use of them.

On the other hand, a technique considered obsolete as it is the use of C macros, has proven very useful to minimize programmer's effort and enable the implementation of rather complex behaviors (one of the good things of developing with a multi-paradigm language like C++ is that you can always find a more or less immediate workaround[5]). Also, C macros are a simple compiler feature which is available in all C++ development platforms.

3. WHAT CLAM HAS TO OFFER
At the time of this writing, around 300 C++ classes (70.000 loc) exist in the CVS repository. Although the website [6] is up since July 2002, the first formal public release is due in November 2002 coinciding with the first milestone of Agnula.

CLAM brings the world of software design and engineering to DSP developers who could care less about it. For doing so, it offers some general infrastructure but, most importantly, it forces users to follow some "good coding principles" and it provides a general model for easy (re)usability. Thus, the user of the framework only has to concentrate on writing signal processing algorithms and, eventually, modeling new data structures or implementing particular flow control schemes.

The core of the library is a repository of digital signal processing algorithms related to audio and music. These algorithms can be used for a wide range of applications but, at the time of this writing, they are mostly related with the MTG's research field, which is mainly spectral analysis synthesis and transformations. Nevertheless, the framework has been designed so that further additions can be done without much hassle. This is mainly due to the fact that a CLAM processing network can acknowledge any kind of processing data as long as it complies to the required interface (i.e. derives from *ProcessingData* base abstract class).

CLAM compliant data classes are the only possible input of *Processing* objects and they offer some special services: attributes can be instantiated and removed dynamically, they can be visited as components of a composite [7], a standard

getter/setter interface is automatically derived for all of them at compile time and, furthermore, a serialization service to XML is also automatically provided [8].

Processing classes encapsulate all processing algorithms in a CLAM application. They offer scalability so that any final CLAM network can be looked at as a Processing Composite that includes any number (and levels) of *Processing* components inside. *Processing* objects respond to synchronous processing data and asynchronous control events.

CLAM also offers a GUI module that allows the decoupling of any particular third party toolkit from the system model. This service can be used with any toolkit but it is currently being used mostly with FLTK [9], for being this the only cross-platform graphical toolkit that has an acceptable free software status.

4. WHAT CLAM CAN BE USED FOR

CLAM responded to an urgent internal need for having a structured repository of signal processing tools focused on audio and music. For that reason, it has been used as an internal development framework since its very beginning. Of course, our patient users have had to cope with multiple refactoring [10] periods that have led to the implementation of a rather complex branching scheme on our CVS repository. But on the other hand, we have been able to implement an spiral iteration process, redefining requirements and redesigning our model at each turn.

Thus, CLAM applications have been developed and have been used as benchmarks to test the feasibility of the library. In the following lines, the main characteristics of these applications are outlined. All of them share the same underlying structure and have at least a Windows and Linux version that highlights the portability spirit of the framework.

SALTO[11] is a software based synthesizer. It is based on a spectral modeling technique named SMS[12]. It implements a general architecture for these synthesizers but it is currently only prepared to produce high quality sax and trumpet synthesis. Pre-analyzed data are loaded upon initialization. The synthesizer responds to incoming MIDI data or to musical data stored in an XML file. Output sound can be either stored to disk or streamed to the sound card on real-time. Its GUI allows to modify synthesis parameters on real-time.

Time Machine[13] is a high quality time stretching algorithm that is already being used in some commercial products. It is a clear example of how the core of CLAM processing can be used in isolation as it lacks of any GUI, audio input/output...

SMS Analysis/Synthesis illustrates the core of the research being carried out at the MTG. It is a complete rework of the already public SMSTools application that has become flagship of the group. Configurations are loaded from XML files. Using these parameters the input sound can be analyzed, looked at using the GUI, transformed and synthesized back. The result of the analysis can be stored in MPEG-7 like XML format [14].

RAPIDD [15] has been designed for performing real-time audio processing/transformation. It has proven to be a robust, reliable and efficient sound processor in a live performance. A prototype was used for accomplishing real-time morphing of a harp and a viola in a composition by Gabriel Brnic that was performed in the Multiphonies 2002 concert cycle at the GRM in Paris. The GUI for that particular application was developed using QT and it used a CLAM-based signal processing library as its sound engine

OOPSLA'02, November 4-8, 2002, Seattle, Washington, USA.

2002 ACM 02/0011.

5. ACKNOWLEDGMENTS

The work reported in this paper has been partially funded by the IST European programs AGNULA and CUIDADO.

6. REFERENCES

[1] Amatriain, X.; Arumí, P.; Ramírez, M. CLAM, Yet Another Library for Audio and Music Processing?. In OOPSLA 2002 Proceedings (Companion Material). Seattle, 2002.

[2] UPF's MTG homepage: http://www.iua.upf.es/mtg

[3] Free Software Foundation. Gnu general public license (gpl) terms and conditions. http://www.gnu.org/copyleft/gpl.html.

[4] AGNULA website: http://www.agnula.org

[5] Stroustrup, B. Why C++ is not only an object-oriented programming language. In OOPSLA'95 Proceedings (1995)

[6] CLAM website: http://www.iua.upf.es/mtg/clam

[7] Gamma, E., Helm R., Johnson, R., and Vlissides, J. Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[8] Garcia, D; Amatriain, X. XML as a means of control for audio processing, synthesis and analysis. In Proceedings of the MOSART Workshop on Current Research Directions in Computer Music. Barcelona, Spain, 2001.

[9] Fast Light Toolkit Homepage: http://www.fltk.org

[10] Martin Fowler's Refactoring homepage: http:/www.refactoring.com

[11] Haas, J. 2001. 'SALTO - A Spectral Domain Saxophone Synthesizer'. Proceedings of MOSART Workshop on Current Research Directions in Computer Music. Barcelona

[12] Serra, X. 1996. Musical Sound Modeling with Sinusoids plus Noise, in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, Musical Signal Processing. Swets & Zeitlinger Publishers.

[13] Bonada, J. 2000. Automatic technique in frequency domain for near-lossless time-scale modification of audio. Proceedings of the 2000 International Computer Music Conference. San Francisco: Computer Music Association.

[14] Martínez, J. M., Overview of the MPEG-7 Standard http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm

[15] Robledo, E. 2002. RAPPID: Robust Real Time audio processing with CLAM. Proceedings of 5th International Conference on Digital Audio Effects. Hamburg, German