RTcmix and the Open Source / Free Software Model
David Topper
University of Virginia
McIntire Department of Music
Charlottesville, VA 22903
topper@virginia.edu

Abstract

While not entirely by design, Cmix has been an open system since its inception. Sharing code between a core group of developers was more or less the rule of the day for Unix software developed in the 1980s. The rapid expanse of computer music applications under the Nextstep operating system furthered this trend. Similarly, in the early 1990s, the still open nature of the Cmix source code, combined with newer and faster CPU technology, led to the development of a real-time version called RTcmix under the SGI platform. Soon thereafter, a Linux version appeared.

Work now continues on RTcmix in many different areas. The internal scheduling algorithm has been refined. The ability to load instruments dynamically at runtime has been added. Dozens of interfaces have been developed to send control messages to the RTcmix synthesis engine. An API for creating such interfaces has even been designed. Data can be read from MIDI and special control devices. There are also now several mailing lists and online bulletin boards where people exchange ideas and bug fixes.

This paper will briefly document the history of (RT)c mix and evaluate its development from within the framework of the Open Source software paradigm. A primary contention being that this software paradigm offers unique benefits, particularly to the field of computer music, one that capitalizes on the sense of community and freedom of expression that the Open Source model promotes.

Recent applications developed under SGI and Linux will also be presented from that perspective. That is, each application will be briefly demonstrated and explained from the perspective of the program's author (paraphrased by the paper's author), documenting how RTcmix and its Open Source model allowed for a powerful type of artistic freedom both conceptually and technologically. Problems with the Open Source model will also be addressed such as quality and version control. Future direction will be postulated. Question and answer feedback will be solicited.

I. Open Source History

The idea of infinite recursion seems to best explain the virtues of Open Software: an idea taken into many hands, each with the ability to refine and evolve the original notion. The paradigm promotes progress and evolution by design in that it allows anyone to join in and contribute, building upon previous work. An outgrowth of this mass evolutionary characteristic is interaction. Open software projects have a tendency to interact with each other. The idea is in essence very similar to the scientific method, where work is published, reviewed, and often built upon by peers. Imagine if Einstein or Newton placed a copyright on their work. Dr. Mann of Wearcomp\(^1\) provides several eloquent discourses.

The GNU project\(^1\) was officially conceived in 1983, founded by Richard Stallman. A related organization, the Free Software Foundation, works in part to assist the continued development of GNU. The GNU Public License is a commonly used document to make a given piece of software, free software. The GPL basically states three things: A user may or may not pay for GNU software, but once in possession, the user can:

1. freely copy the program and give it to friends and coworkers,
2. freely change the program, having access to the source code,
3. freely redistribute and improved version, and even charge for it.

A familiar phrase within the community is, "The word 'free' above pertains to freedom, not price."

The Open Source software movement\(^2\) started in 1998 and was sparked into creation largely by Netscape's announcement to release its source code. The definition of Open Source is taken from
II. Cmix History

Cmix was derived from the MIX program, a 20 track mixer written by Paul Lansky in 1978, which ran on VMS in FORTRAN on IBM mainframes. Synthesis abilities were added, and Paul ported Cmix to run on a PDP11/34 in 1983-84 under BSD2.9 Unix. In 1985, he moved it over to Ultrys on a DEC MicroVAX. He and Lars Graf added the MINC parser in 1987, with help from Brad Garton and Dave Madole on various parts.

Doug Scott and Paul ported an initial real-time version to SGI in 1995. While sometime in 1993, I took a first stab at a Linux port which had to use Sox to unswap big-endian files. Then in late 1995, Brad Garton and myself created a real-time version with a scheduler. Those enhancements were incorporated into the Linux version. Brad also added the ability to read TCP socket data. That version was presented at IC7C '97. Shortly thereafter, Luke DuBois added an API for writing to TCP sockets, which facilitated interface development. Doug Scott set up a mechanism to dynamically load instruments at runtime in 1997. The Linux version was brought up to date shortly thereafter. Enhancements have continued, most recently with John Gibson’s interface to Bill Schottstaedt’s snlib for audio file IO, continued development of real-time pfield control and the setup of a CVS server for the Linux version. Work is under way to put RTcmix under the GPL. None of this development would have been possible if the original Cmix and subsequent RTcmix versions had not been open.

IV. RTcmix Future and Projects

Future RTcmix versions can hope to employ: shared audio buffers, more robust real-time pfield control, interfaces to other parsing languages such as Perl, the integration of audio with graphics and video output, support for SMP, and of course more interfaces. With an Open Source paradigm, anything is possible. Many RTcmix projects are often designed to suit specific performance and/or compositional needs.

There have been many Cmix/RTcmix projects over the years. Some work was lost when development of the Nextstep system ended, but much is currently being resurrected. Recent projects employ TCP/IP communication protocols to send control messages to the RTcmix engine. Following is a short description of some projects:
http://www.music.columbia.edu/cmc.html

Luke is considered by many to be the resident expert on Buchla operation at Columbia’s CMC. An outgrowth of his analog experience has resulted in the creation of the Virtchla, where a sequencer sends control data to RTcmix.

Treembre: Doug Geers, Columbia University 
http://www.music.columbia.edu/cmc.html

Doug is both a programmer and composer at the Columbia CMC. The Treembre program allows pitch objects to be organized in a tree-hierarchy, which can be modified at any level, maintaining subtree relationships.

Patchmix: Mara Helmuth, CCM, Cincinnati 
http://meowing.ccm.uc.edu/

Mara is the Director of CCM2, at the University of Cincinnati. Patchmix was originally designed under Nextstep for use with Cmix. One can create a flow chart of unit generator icons which are turned into Cmix code, compiled and run within Patchmix.

Psldier: David Topper, VCCM, Virginia 
http://www.virginia.edu/~music/VCCM.html

I am the Technical Director at the Virginia Center for Computer Music. The psldier program allows a user to control any RTcmix instrument’s pfield in real time. It hopes to be a general purpose tool to control RTcmix interments. It was designed under Linux, using GTK, a free GUI toolkit.

V. Conclusion

There are very real economic factors to consider when looking at Open Source and computer music. Most notably, many composers and students don’t have a great deal of money. This unfortunately puts many creative tools only in the hands of the select few who can afford it. Open software, and free software in particular, eliminate this problem and open up the use of high end tools to everyone, from the general (operating system) up to the specific (application).

It seems to make particular sense for tools to be open to avid users. Imagine a mechanic able to easily change the design of his/her scalpel. The point at which Open Source development meets developer is often the very home of creativity. What would inspire someone to contribute a new feature? Often times its a feature they would like to use and is not yet available. Such a development model promotes creativity and innovation by design.

Consider the diversity of individual studios. Few have precisely the same combination of hardware. What an amazing possibility, if everyone were able to have access to each other’s studios and offer improvements which were almost immediately available to everyone. Such a synthesis is already beginning, but hasn’t yet made a clear impact on popular music.

Anyone with a home studio has at one point or another confronted the frustrating prospect of getting everything to work together. Whether it be connecting an older synth to a MIDI timepiece, or trying to find a patch that works with such-and-such program. The Open Source model addresses this directly. If something doesn’t work properly, anyone can fix it. Beyond simple functionality, the potential interoperability between various Open Source music components is limitless.

The Open Source community would by most standards not be considered large in 1998. Even today, it isn’t exactly mainstream, much like the ICMA. The sense of community was and remains prevalent and important to both. With the help of tools and people interacting with each other, the phenomenon has grown. My contention is that the same can be true for the ICMA and software designed as a result of affiliation with it.

Resources

1. GNU foundation: www.gnu.org
2. Open Source project: www.opensource.org
3. Linux: www.linux.org
4. Perl: www.perl.org
5. FreeBSD: www.freebsd.org
6. Apache: www.apache.org
7. CVS: www.cyclic.com
8. Wearcomp Computer: www.wearcomp.org