Making Interactive Electroacoustic Music with Computer through the Use of RTMix -- a Real-Time Interactive Electroacoustic Music Performance, Composition, and Coaching Interface

IVICA ICO BUKVIC
Department of Theory, Composition, and Musicology
College Conservatory of Music, University of Cincinnati
3346 Sherlock Ave. #21
Cincinnati, OH 45220
U.S.A.
http://meowing.ccm.uc.edu/~ico/

Abstract: - With the technological advancements in computer technology, the making of the real-time computer-aided interactive music has become a reality. However, due to lack of comprehensive software, such form of artistic expression still proves to be a daunting task. RTMix is an open-source software application that has been designed for the highly stable and scalable Linux platform and whose function is to provide a transparent real-time and user-friendly performance interface, as well as the universally portable and easily editable/reproducible "scorefile" format for the live and interactive multimedia works. RTMix's implementation addresses the problem of lack of comprehensive and accurate interactive electroacoustic music computer interface by providing an all-in-one solution: it can be used for composition, coaching, and most importantly performance of complex interactive works that would otherwise be very hard to execute with currently available software tools. Its abstractness and extreme flexibility enables user to simultaneously control a variety of independent applications, such as RTCmix, Csound, aplay, sox, mpg123, as well as any other process that can be triggered via UNIX shell.

Key-Words: - Computer, Music, Interactive, RTMix, Linux, Electroacoustic, Open-source, Art, Computer-aided.

1 Introduction

Although the interactive electroacoustic medium has been in existence for some time, the lack of affordable computing power has made many composers shy away from this type of artistic expression. Throughout the past decade things have rapidly changed with the advances in technology that provided us with microcomputers of unprecedented computing power, ultimately making this exciting medium more accessible than ever before. Yet, to this day the interactive electroacoustic music has been hindered by the serious limitations of the available software performance interfaces, forcing many composers and performers who have expressed interest in this medium, to resort to a simple stopwatch in order to synchronize the "live" (i.e. performer) and recorded/computer-generated counterparts. Subsequently, through the utilization of such rudimentary equipment for coordination of the performance, most pieces had to be simplified in order to ensure adequately accurate interaction and execution, thus ultimately further limiting composer's expression. Such state of flux has resulted in a recursive problem where composers and performers alike are simply discouraged by the current dichotomy between the exponentially growing computational power and the stagnating lack of development of user-friendly interface(s). RTMix is an application that has been initially designed in order to address this issue and has since grown to provide an extended array of features, making it an "all-in-one" solution.

2 Historical Background

Although we can track the interactive electroacoustic music endeavours well into the late 19th century with the instruments like Singing Arc and Musical Telegraph [1] and early 20th century with the use of Theremin [2], the real revolution in interactive electroacoustic music came during the early 80's with the advent of MIDI standard, when the idea of interactive music had finally become a viable option, rather than a daring stunt. Many interactive interfaces nowadays utilize MIDI protocol to trigger a variety of both musical (i.e. triggering synthesizers and samplers) and non-musical events (i.e. stage lighting). More importantly, MIDI is nowadays being used in very popular object-oriented visual programming environments whose primary focus is on multimedia, such as Pure-Data [3], Max [4], jMax [5], and EyesWeb [6]. Despite the popularity of MIDI standard, the real-time DSP software and hardware did not materialize until the early 90's simply due to lack of affordable computing power. Since, a large variety of both software and hardware solutions have been readily available, such as MSP (extended objects for the Max software), RTCmix [7], Csound (real-time aspect) [8], Kyma-Capybara software/hardware combination [9], various proprietary hardware modules, DJ equipment, and many more. With such offering, two inherently different solutions have emerged, both of which are currently able to furnish composer with the tools required to make and, more importantly, perform interactive electroacoustic music. One of them is hardware, and the other is software. Furthermore, software can be subdivided into proprietary (i.e. commercial) and the open-source, a distinction whose importance will be addressed shortly.

3 Selecting the Best Approach

What is inherently different between the hardware and software solutions is obviously the price which proportionally corresponds to their accessibility. More importantly, most of the hardware solutions have a rather limited scope of effectiveness, usually being designed for a particular setting and/or software, lacking modifiability and expandability (although arguably the same can be said for a portable notebook, and certainly there are exceptions to this rule, i.e. Kyma-Capybara). While such an approach is certainly a valid one, especially in situations where the hardware offers all-encompassing and relatively flexible environment, such as Kyma-Capybara, the problem with it is that such environment usually does not utilize computer as the central processing unit, nor offers user the flexibility to simultaneously employ different tools and applications. Also, such hardware is simply out of reach for many individuals due to costs involved.

Commercial (a.k.a. proprietary) software is usually a more affordable solution that utilizes now vastly more powerful computers, making it certainly a feasible alternative to the aforementioned option. Yet, I find this solution somewhat limiting due to its closed-source nature which for many composers (including myself) means that their freedom of expression will be limited by the software's design. More importantly, most of the proprietary software currently available focuses on the lucrative aspect of the market that consists of sequencers, sound editors, and post-production tools. Thus, they do not provide performance-oriented interactive interface since that is simply not their intention. As an exception, there is Max/MSP and other similar applications which offer a variety of interactive tools. But, in terms of their abstractness, it is often a time-consuming process to come up with a user-friendly performance-oriented front-end that is typically work-specific, and usually requires a performer who is knowledgeable with the application's environment and feels comfortable using it. In addition, the composer's presence is typically needed in order to ensure the proper execution of the work.

The open-source software has several strong points when compared with the above-stated solutions. One of them is certainly affordability, while other is availability of the source code. This means that when an application has been provided to the community, the code can be easily modified and tailored to individual needs, as well as potentially more easily transferred to another platform. This model has attracted a lot of developers to contribute to the open-source community, and as such, there are numerous software applications whose focus is on interactive music. Some of the examples include PD, jMax, RTCmix, and others. The limitations of the open-source approach are that among the numerous software packages, there are only few fully functional applications that often lack documentation, and most importantly, most of them have little or no ability to interact with each other. Simply, none of them offer a unified transparent performance interface that is capable of triggering different types of events and applications, while at the same time relaying all the pertinent information to the performer in a visually appealing and concise manner.

After considering all of the above-mentioned options, as well as facing the typical limitations of the available interactive electroacoustic music software, I came to the conclusion that not only is there a need for a unified interface that would drive interactive work's compositional process and performance regardless of the types and combinations of the utilized software, but also that the best setting for the RTMix application would be an open-source approach using the open-source operating system – Linux. The open-source environment is not only the most affordable solution, but more importantly, most of the Linux OS open-source applications have one crucial thing in common, one thread that could be used as a building block for a unified interface – they all can be triggered via UNIX shell system calls.

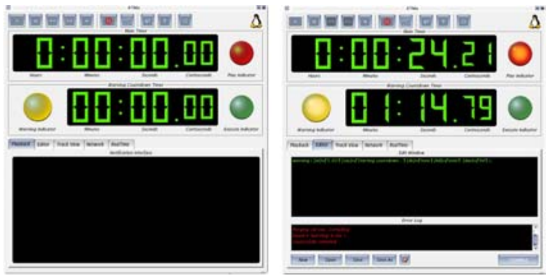

Fig.1 RTMix with “play” and “edit” tabs open.

4 RTMix Interface

RTMix (titled after the RTCmix real-time audio manipulating and synthesis scripting language that was used for my *Slipstreamscapes III: The Sea* and *Slipstreamscapes V: Lullaby* interactive electroacoustic works) was initially created as a simple C program due to fact that my work required a coordinating and event-triggering interface. I’ve initially designed this tool in order to ensure an accurate execution of complex interactive patterns shared between the guitarist and the computer. Since, the application has been ported utilizing Qt toolkit and C++ language, and currently offers a user-friendly GUI interface that is populated with visual stimuli used to grab the performer’s attention as needed, while keeping distraction to a minimum. The performance event-list is now abstracted into a scripting language that is compiled prior to playback and has elaborate error-checking routines. The interface also has advanced timing features, such as a warning countdown counter, as well as “timer bookmarks” (a kind of a *fast forward* and *rewind* functionality), most of which will be discussed shortly. Furthermore, the application’s functionality has been vastly expanded, and it can now be used not only for performance purposes, but also for composition, as well as coaching and rehearsals.

RTMix interface (see figure 1 for the interface’s screenshots) currently consists of one window and two tabs. The main window can be split-up vertically into the top half with the “transport” and timer widget and the bottom half that consists of series of tabs. Currently, the application utilizes two tabs: the “play” tab, and the “edit” tab.

The “transport” widget is populated by basic transport buttons (i.e. play, stop, pause, etc.) and additional not-so-obvious, yet powerful features, such as the “panic button,” whose function is to cease all active events and stop the performance in the case of an audio feedback or another similar critical problem, and a “toggle size” (a.k.a. “modes”) button that toggles between the timer-only and the full window views. This feature is especially useful in cases where performance asks for an interface with minimal distractions (see Figure 2). A relatively big timer is located below the transport controls whose main purpose is to provide legibility in performance settings. The timer can be vertically divided into the “main” timer and a countdown (a.k.a. “warning”) timer. The main timer’s function is to simply linearly count time from the beginning to the end of a work, while the warning timer can be enabled at any moment by utilizing the scripting language for the purpose of warning performer of incoming important “checkpoint” in the piece. Such a feature ensures accurate and timely execution of an event. Both timers have their respective visual stimuli that correspond to their events. For instance, the red “play” indicator comes on when the performance has been started, or blinks when the performance is paused. The yellow “warning” indicator comes on when the warning counter begins counting, and blinks when timer is about to run out. When the warning timer reaches zero the red “execute” indicator lights up in order to visually reinforce new event’s temporal downbeat (if such event is required). The visual appearance of these visualizations as well as their behaviour is fully customizable, and one can easily use just about any combination of images to portray their activity by simply placing new images with particular names in the specified location on the hard drive.

The “play” tab consists of just one window that displays all of the coaching and performance text messages, while on the “edit” tab, there is a textual script editor and an error log window where syntax errors and other warnings are displayed during the compiling of the script. Below the editor and error log windows are the file I/O [input/output] controls for storage and retrieval of the script files, “purge error log” button for clearing the error log window, as well as the “compile” button that parses the event “scorefile” and stores the parsed information in computer’s RAM in a form of a “cue list.”

4.1 RTMix Scripting Language

The scripting language provided in RTMix is rather powerful due to its inherent flexibility. It has been built in Qt/C++ and utilizes complex string operations in order to provide parsing and error-checking routines. The language relies upon the system calls in order to invoke various events and since most of the Linux applications (including ones that bear no relation to audio) are executable from the shell, one could easily coordinate a whole array of actions via this simple interface, such as PD, jMax, Csound, RTCmix, mpg123, aplay, ecasound etc., as well as non-audio events, for instance opening images in an internet browser or starting an OpenGL animation in order to accompany the piece with visual stimuli. The scripting language currently comprehends only a half-dozen commands (event, warning, text, checkpoint, pause, stop, clear) and in its syntax strongly resembles Paul Lansky's RT script. Each command serves a particular, and in most cases a self-explanatory function: an "event" call is used for triggering an event at a given time, while providing an optional textual comment to accompany it; a "warning" is similar to an event, except that it also utilizes a warning timer for the purposes of anticipation of the actual event's temporal downbeat; a "text" is simply a textual event that is to be displayed on the notification interface; a "checkpoint" is used to annotate sections of the piece, so that both in rehearsal and performance settings, the user can easily fast-forward, rewind, or simply jump to a particular spot in the piece; the "pause" and the "stop" objects are self-explanatory, while the "clear" option purges the notification window, rendering it clear. As an example, an "event" (i.e. playing a sound) 5.32 seconds into the piece with a textual notification, would have the following syntax:

```
event( [at] = {"5.32"} [do] = {"cmixplay sound.wav"} [say]
= {"Playing sound.wav"} [kill] = {"cmixplay"} );;
```

In addition, these scripts can be saved, edited, and, since they utilize a simple text-file format, they are extremely portable. With such flexibility, one could easily create a whole collage of events of tremendous complexity. As such, the scripting language does not only serve as a performance tool, but also poses as a rather formidable compositional environment, as well as a virtual coach. The potential of its coaching abilities is immeasurable because many interactive works currently require elaborate setups and commonly the presence of the composer, causing significant technical obstacles for performance. With RTMix, it is now possible to conveniently e-mail annotated "scorefiles" to the performer(s) [and any other involved personnel], who would after uploading the newly acquired script, rehearse the work while being tutored and warned through the application of possible pitfalls, difficult sections in the piece, as well as anticipate important downbeats. RTMix can be utilized in any performance setting that requires great amounts of coordination between different parts regardless of the composition's medium.

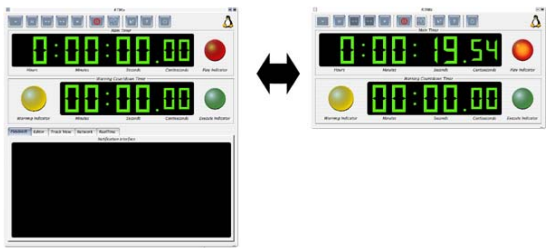

Fig.2 Switching between "performance" and "practice" modes.

5 Becoming an All-in-one Solution

RTMix in its current state is extremely stable and provides all of the above-mentioned features. What is also very important is that RTMix has a very small computer resource utilization footprint, meaning that in its peaks utilizes less than 3% of the total CPU's power [info gathered using "kpm" tool running Mandrake 8.1 Linux OS on the Pentium III 500Mhz machine]. Such design makes it relatively transparent to the system, leaving the vast majority of the computing power for the audio processes. RTMix is also very accurate, being able to trigger events with a .01 second precision. Its features make it a multi-purpose tool: it poses as a sophisticated timer and event triggering mechanism useful for the interactive and multimedia performances, it is a flexible compositional environment, a virtual coaching tool, provides a convenient way of storing composition and performance information in a concise, portable, and reproducible fashion, and can be used in non-electronic environments, such as extremely complex acoustic works that require strict timing.

RTMix has so far been utilized in my Thesis work *Slipstreamscapes III: The Sea* that was performed as a part of my Masters Thesis recital, as well as on the *Music01* summer music festival. Its latest version (0.16) was also used for the *Slipstreamscapes V: Lullaby* premiere at the *Music02* festival that took place in June 2002. Computers utilized were custom-built desktop Pentium-III 500Mhz computer and a 1GHz Pentium-III Dell Inspiron 8000 notebook. Both performances were favorably received. RTMix has been developed in close collaboration with several performers whose thoughts and comments were taken into account while designing the interface. Feedback provided from several performers revealed strong interest in usage of this application as well as performance of the interactive music employing it.

5.1 Future Developments

While RTMix is already rather flexible and functional software, future versions will provide features that will make it stand out as a unique application that encompasses a full range of both the compositional and performance tools for the interactive electroacoustic medium. Most of these features will be implemented by the end of the summer 2002. They include networkability of multiple RTMix instances running simultaneously on multiple computers making it easy to utilize it as a synchronization tool in the chamber, or even orchestral settings. The scripting language will be vastly expanded enabling user to trigger more interface-oriented events. There will be an ability to filter text events by assigning them different levels of importance (i.e. performer's vs. mixing technician's text) providing performers and other performance participants with an option to filter only messages pertaining to them, as well as an ability to disable potentially distracting visualizations. In addition, script will have a random number generating object that will enable utilization of RTMix in musical settings that rely on aleatory and indeterminacy. Furthermore, RTMix will have a MIDI interface so that the performer(s) will have a direct control over the performance interface through use of external foot-pedals and similar MIDI equipment. RTMix will also have a convenient metronome feature as well as a track window that will pose as an alternative to the text-based scripting, offering a more user-friendly approach to composing/editing music.

6 Acknowledgments

First off, I would like to thank my wife Anamaria for her kind, loving support and understanding, as well as my parents who have through their hard work and sacrifice provided me with the opportunity to obtain a high-quality education. I would also like to thank my mentor Mara Helmuth for her ongoing support and insightful guidance.

References:

[1] Chadabe, J. "Electric Sound." New Jersey: Prentice Hall, inc. 1997, pp.3.

[2] Chadabe, J. "Electric Sound." New Jersey: Prentice Hall, inc. 1997, pp.8.

[3] Puckette, M. 2002. "Pure Data Page." http://www-crca.ucsd.edu/~msp/software.html.

[4] Cycling '74. 2002. "Max/MSP Page." http://www.cycling74.com/products/maxmsp.html.

[5] Ircam. 2002. "jMax Page." http://www.ircam.fr/jmax/.

[6] InfoMus. 2002. "EyesWeb Page." http://www.infomus.dist.unige.it/.

[7] Garton, B., and D. Topper. 1997. "RTcmix – Using CMIX in Real-Time." *Proceedingsof the International Computer Music Confernce.* International Computer Music Association.

[7] Topper, D. 2000. "(RT)cmix for Linux." http://www.people.virginia.edu/~djt7p/Cmix/Cmix.html.

[8] Thompson, R. S. "The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming." Massachusetts: The MIT Press, 2000.

[9] Chadabe, J. "Electric Sound." New Jersey: Prentice Hall, inc. 1997, pp.265.