SLINC 1.0: A GUI for controlling synthesis and linking instruments in RTcmix

David Topper
Virginia Center for Computer Music
University of Virginia, Charlottesville, VA 22903 USA
topper@virginia.edu http://www.people.virginia.edu/~topper

Abstract

SLINC is graphical interface for controlling parameter fields of RTcmix instruments in real time, written using the powerful open source Gimp Tool Kit (GTK). It runs under Linux systems, and is capable of controlling multiprocessor, and multichannel RTcmix instruments. SLINC also supports a lightweight parser which can be used to send musical sequences, with the synthesis parameters of each element being controlled in real time.

Input control can be the GUI itself (e.g., sliders and dials) as well as MIDI, Joystick, and mouse / touchpad. SLINC works on any Linux system, but has been a result of parallel work on the Semi and Portable Audio Workstation (SPAWN and PAWN). By allowing flexible real time control of musical events, a laptop running SLINC and RTcmix can be used extensively in performance. Research on the current state of using less standard, controllers will also be presented.

I. Overview

SLINC is a graphical user interface for controlling RTcmix in real time. Over the past several years many special purpose interfaces have been created, often with the intention of realizing a specific idea or paradigm. SLINC is a more generic attempt, with a primary focus being on flexibility and ease of use. The goal is to facilitate controlling various parameters from seemingly limitless sources hopefully well beyond the scope of any premeditated design.

Currently there are a limited number of external control inputs, MIDI being the most popular. Not surprisingly, a main function of SLINC is to map MIDI events to various types of control mechanisms. External control, however, is not limited to MIDI alone. Virtually any input to a computer can be a control source. SLINC is designed to facilitate new connections. A recent example is controlling various synthesis and effects parameters from a normal game joystick and mouse.

II. Internal Structure

SLINC relies heavily upon the powerful features of GTK (Gimp Tool Kit) and support GDK libraries (see Resources). An obvious reliance is the ability to draw windows, sliders and other GUI widgets. But perhaps even more important is the internal callback structure employed by GTK which allows different mechanisms to interact, which is the primary function of SLINC. In the end, SLINC sends data via TCP socket connection to the RTcmix engine, which in turn listens to the socket and parses commands accordingly in real time. This classic client server architecture promotes modular design that SLINC exploits.

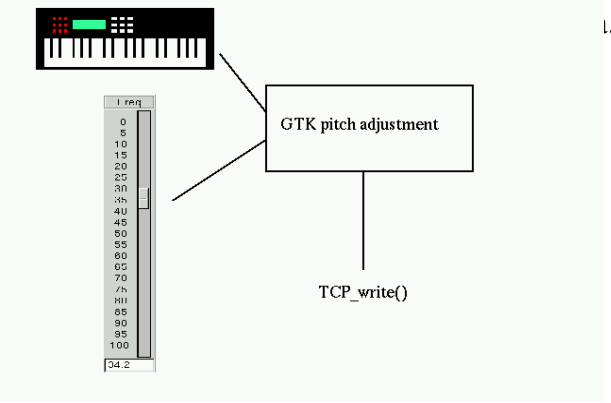
A cornerstone of the GTK callback structure used by SLINC is the *adjustment*. It might be thought of as the nuclear unit of application design. It is not, however, the only unit. Many musical GUI applications make heavy use of the infamous slider as a primary building block. In

GTK, a slider (know as a category of *range* widgets) is a widget which can control various aspects of an *adjustment*, namely its value.

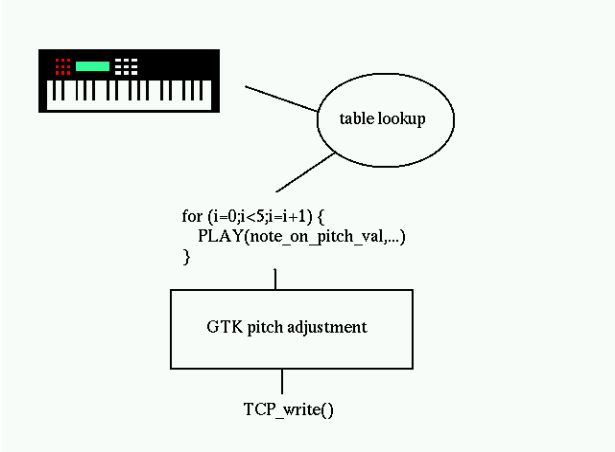
The basic process is as follows. A widget (e.g., a slider), and specific signal are connected to an adjustment. When a slider is changed, it emits a signal. Through the adjustment/signal/callback mechanism, when a slider is changed, so is the adjustment's value. Other widgets and functions can be connected to an adjustment so that when they change, so does everything else. The alternative approach would be to create a complicated network of signals and connections, which would be far less elegant and more difficult to program. Adjustments can also be set "manually" by special gtk_set_adjustment() function calls.

GDK (GTK's supporting library) provides similar functionality with broader scope. The function gdk_input_add() allows monitoring of virtually any device or object (e.g., the serial, midi, or joystick port). When some change is detected on the device, the same signal and callback mechanism described previously can be triggered.

The following simple example should illustrate using these ideas in tandem. Let's assume control of pitch is done by both slider and MIDI note_on events. So in this case pitch is a (GTK style) adjustment. The midi port is monitored by gtk_input_add(). Whenever MIDI byte come in via the port, the appropriate callback functions are triggered and adjustments changed. Specifically, a midi event parser and a function to set the adjustment value. Similarly, a slider (range widget) is also connected to the pitch adjustment. So whenever a note_on event is received, the adjustment and slider are changed to reflect the new value. Finally, another callback is connected to the pitch adjustment. B h a . T



The elegance of the paradigm allows for great flexibility. Not limited to simple widgets and adjustments, callback functions can be used to implement higher level functions. Namely, a lightweight MINC style parser written with yacc and lex. MINC is the scripting language used by RTcmix. By means similar to the above, a MIDI note_on event can trigger a callback function which reads in a buffer containing a script. The buffer is parsed and TCP data (along with any other adjustment setting) sent out accordingly.



In order to allow multiple mappings of controls and events, a lookup table is consulted to determine which events should control what structures. In the case of the last example, a MIDI note_on event lookup would return a pointer to a string buffer containing the mini parser instructions. In the previous example, the table might be used to set scale ranges. The user may want key values from C2 to C5 to span only one octave of pitch values.

III. Development Issues

The Linux operating system offers many virtues for program development. Its open source paradigm has led to a robust, stable, high performance platform that continues to evolve. It has grown immensely in popularity from humble beginnings as a hacker's hobby. There are, however, issues that directly affect software projects like SLINC.

On a certain level, a software application is limited by the devices it can work with. This is not to say that the more devices that work with a given system, the better the system is. Rather, a degree of operability is necessary for an application to be viable. SLINC running under Linux has just reached this stage.

MIDI and audio support under Linux is slightly limited, but rapidly improving. There are two primary APIs for both: OSS (Open Sound System) and ALSA (Advanced Linux Sound Architecture). The former, despite its name, is a commercial product and the latter a free, open source attempt to provide professional grade audio support. Both have virtues and support several commercial MIDI and digital audio cards.

Both SLINC and RTcmix currently only support the OSS audio api. ALSA support is under development, but is needed by both as the two drivers cannot coexist (i.e., it is not possible to use OSS for audio and ALSA for MIDI). SLINC does, however, offer internal support for the Midiator systems MS-124w serial midi box. This allows MIDI input on any PC with a serial port, most notably laptops. Such support would not have been possible with help and specifications of the MS-124w made publicly available by Midiator Systems.

The freely available GTK libraries were a primary inspiration for SLINC. The internal signal/adjustment/callback structure seemed perfectly matched for a control message routing application. Online examples, tutorials, and an active mailing list community were essential in getting rapidly fluent with the API and its higher functions like gdk_input_add(). Future development is planned in the areas of new control systems and devices. Due to the flexible nature of GTK, work can concentrate on specifics, rather than on how to fit the end result into the existing framework. That is, only the respective new callback routines need be designed.

The PAWN and SPAWN platforms are also a direct response to these issues. SLINC is partly an attempt to provide usable interface for those platforms. The hope is that by providing consistent and reliable platform with compatible components, audio development can continue to grow in terms of both body of work and support by audio and other hardware vendors.

IV. Resources

Source code and executables for SLINC can downloaded from:

http://presto.music.virginia.edu/SLINC

The latest source code for RTcmix can be obtained via ftp:

ftp://presto.music.virginia.edu/pub/rtcmix

GTK source and documentation can be found at:

http://www.gtk.org

Information on PAWN and SPAWN can be found at:

http://www.audio-workstations.com

A good starting place for information on Linux is:

http://www.linux.org

The website for OSS is located at:

http://www.opensound.com

The ALSA website is:

http://www.alsa-project.org.