Patchmix: a Cmix Instrument-Builder

Mara Helmuth

College-Conservatory of Music, University of Cincinnati, Cincinnati, OH 45221-0095

Abstract:

Patchmix is an application for instrument design employing the Cmix music programming language. This new version runs on Silicon Graphics IRIX 6.x. It is written on the X windowing system in C++ with Motif and OpenGL, for portability to other platforms. The original versions were written in C++ on X, and in Objective C on NeXTstep. As in the older versions, signal processing software instruments can be created by connecting unit generators in a flow chart. Patchmix writes non-realtime Cmix code, compiles and runs it with test score data. The approach gives maximum flexibility in instrument design, as the code, written in Cmix, can be modified later with the control of C language programming. New features involve making use of RTcminx's real time capabilities, for more spontaneous sound shaping. RTcminx C++ modules can be created, with certain parameters modifiable in real time through the graphical user interface. Other enhancements of this version include better graphical display of icons and connections, more refined connections between unit generators, additional unit generators and graphical function design. The objective was to create an application that allows access to the inner workings of sound when necessary, but to make the manipulation of sound graphically simple and clear when possible. The new version realizes these goals better through a fuller graphical access to the possibilities of music programming with Cmix. Patchmix also functions as an educational tool for teaching music programming, signal processing, Cmix and C.

1 Introduction

1.1 Goals

The purpose of creating Patchmix was to have the fluidity in composing with sound that you have at the piano or drumset, without losing all of the control of sound design possible with digital signal processing capabilities. The program is oriented toward composition, as software instruments are created for future use or modification. The instruments are written in C using the Cmix music programming language, compiled, and tested from the Patchmix interface. In creating the current version real time features of Rtcminx are incorporated, providing some performance and improvisational capabilities from the interface with data/score generation control, including some algorithmic control. Finally, a concern for portability motivated the platform choices.

1.2 History

Paul Lansky's Cmix highly extensible music programming language allows flexibility in instrument design [Lansky, 1990]. Patchmix was originally created on X in C++ from 1989-1990 at the Columbia Computer Music Center [Helmuth 1990]. It then was expanded and ported to NeXTstep / Objective C [Helmuth, 1991]. Brad Garton's Rtcminx, extensions of Cmix for real time control, was demonstrated in 1997 [Garton and Topper, 1997]. The new version of Patchmix was created on Silicon Graphics IRIX 6.x and works with current versions of Cmix and RTcminx.

2 Platform

The Silicon Graphics platform was chosen for its audio and graphics capabilities and development environment. X Version 11 is an extensible, network-transparent window system. The X Toolkit Intrinsic C Library for creating interface objects, with Motif's library of pre-built objects provided widgets for the interface. OpenGL, a streamlined, hardware-independent software interface to graphics hardware provided drawing capabilities. Patchmix is written in C++ for the benefits of object-oriented programming. As C++, X, Motif and OpenGL are available for a number of systems, Patchmix will be easily portable to other systems on which Cmix runs.

3 Structure of Program

3.1 Interface

An event-driven program relies on its interface for effective use by the composer. Therefore, the interface was carefully designed to be as streamlined and clear as possible. Instrument construction is done from one main window (fig. 1). Under the main menu with the usual file saving, editing and control commands, a unit generator selection panel appears with icons representing types of code...
modules available within Cmix. Each icon represents several unit generators; for example the `oscil` can refer to a (menu-selected) oscillator, interpolating oscillator, or buzz waveform generator. The composer can also create his/her own code modules and see them represented on the icon panel. Below the icon selection panel is the patch drawing area. The composer places selected icons on the patch area and connects them into a flow chart representing the signal processing path. Unit generator options and parameters can be edited from dialog panels. Below the patch window are control buttons for the UNIX processes to compile the generated code, run the Cmix process, and play the resulting sounds.

3.2 Code Generation

From the events occurring at the interface level, the program stores information on the unit generators in the current instrument, and how they are connected. When the composer compiles the instrument, the `C++ Instrum` class creates the lines of code needed for each unit generator, including variable declarations, assignment statements, and sample loop statements. It traverses an n-ary tree of the unit generators to construct the code lists. When done, it writes the code lists into the Cmix instrument, with a Makefile. The code is then ready for testing, or can be modified “by hand” (in C) for complex algorithms that would be difficult to represent graphically.

3.3 The Score

Cmix instruments are generally run by redirecting a Minc programming language “score”, or data file into it. Within Patchmix, a default score file is created with several functions available, and basic parameter fields such as start time, and duration of sound. As the composer selects unit generator parameter values, a “p-field” designation means that the parameter comes from the score file, giving more control to the person running the instrument. A score window is created newly for each instrument with the default and any additional parameter fields. The composer can put values into the score, or select an algorithm to produce a number of values with a Minc “for” loop.

4 Real time capabilities

When the mode selection of Cmix/RTcmix is made, the Instrum class stores the appropriate code. Cmix instruments are in C, while RTcmix instruments are C++ classes. In both cases the instruments are compiled and then run with scores.

Real time instruments can have a continuous series of notes sent to them interactively, with parameters controlled by sliders on the score window.

5 Education

Patchmix has instructional value, for teaching music programming and Cmix. The signal processing flow chart presents a clear view of what happens in a typical instrument. The generated code can be investigated to learn how to represent the instrument within the Cmix programming language. Also, for students learning the C or C++ languages, working code segments are provided with sound examples. Patchmix and its source code are freely available at http://meowing.ccm.uc.edu under music software.

6 Future work

Future extensions will be improvements in the patch window, graphical control, and score window control, particularly for RTcmix. Looping, windowing and delays will be handled in a more flexible way. The ability to save and reopen patches will be implemented. Also, cross-network control should allow Patchmix to send sound from host to host. Another future enhancement would be to make use of the dynamic loading of instrument objects, which could be used as needed on a higher level patch.

7 Acknowledgments

In addition to those mentioned in the references and others who helped create Cmix, two other people helped with Patchmix. Ico Bukvic created new icons for the unit generators. Ronjan Sidkar ported some of the Objective C code to C++. Students in my music programming classes will be implementing future features.

References

fig. 1 Patchmix main window with unit generator selection panel (top), instrument patch construction area (middle), and control/mode buttons (bottom).