PATCHMIX: A C++ X Graphical Interface to Cmix

Marx Helnacht
Columbia University Computer Music
709 Dodge
Columbia University
New York, NY 10027
marx@woof.columbia.edu

ABSTRACT
An object-oriented X graphical interface to the Cmix music-programming language has been written which allows the composer to create a flow chart of connected unit generator icons. The graph is compiled into an instrument from Cmix library or user-written functions. C++ unit generator, tree and instrument objects transparently set up signal processing code within the constructed Cmix instrument function. This instrument is then compiled, run with test data, and played from within the interface. It provides an opportunity for those learning Cmix to produce instruments easily, and allows the composer to create and test instruments quickly, visually and intuitively.

The goal for Patchmix was to implement a user-friendly, graphical interface with which one could design Cmix instruments. A flow chart of unit generators is often a useful visual representation of an instrument in software synthesis. If unit generator icons can be defined, manipulated and connected graphically, that information can be interpreted by an intelligent program to generate C code. The ability to create sounds visually and intuitively and hear them immediately aids the compositional process. Even composers without C skills can make use of the accuracy and power of Cmix, and use Patchmix to learn programming.

Background
Cmix is a flexible music programming language written by Paul Lansky. It provides a library of unit generator functions and systems functions for handling notes and samples. A YACC parser, Minc, interprets the data or score files. User-written functions can be compiled with the Cmix object to produce an independent Cmix instrument. While Cmix does not have most of the limitations of other sound synthesis systems, it does require the user to know enough C programming to handle structures and pointers.

A Cmix instrument consists of a C function in which a loop generates samples which are modified by the functions specified by the user. Patchmix selects from the user-created diagram, which variables should be declared, what functions to call (indicated by the unit generators) and when to call them. If further processing is requested, such as changing parameters over the course of a note, Patchmix sets up additional loops or variables. Parameters, such as start time or waveform used, may be either an within the instrument or made accessible to the instrument user through the data or score files. For a usable instrument to be created, the structure of the data must also be clear and under the control of the user.

Overview
Patchmix constructs a Cmix instrument from user input, compiles it, queries the user for test data (or supplies defaults), runs the job, and plays the resulting sound file. The Cmix function code, along with

ICMC GLASGOW 99 PROCEEDINGS
other needed files then reside within a directory created by Patchmix for the user's inspection and edification.

A Patchmix user can construct the C functions for a complex instrument in a few minutes with mouse and keyboard input. Minor changes in algorithm or data can be quickly made and heard without ever leaving the interface. This facility allows for spontaneity and continuity in the creative process which is usually found only with MIDI or analog studio equipment. Since no writing or compiling is done until the compile button is hit, there is no need to store complex changes in algorithm. Each version of an instrument may be saved under a new name.

The Graphical Environment
To build this symbolic compiler a flexible graphical platform was needed. The X windowing system allowed extensible event processing and image representation as well as portability to many systems. The graphical layout of Patchmix is in three sections: a menu bar for selecting unit generators, a canvas on which the flow chart is constructed, and a panel of buttons which compile the Cmix instrument and create or handle the sound file. Unit generator icons can be manipulated and resized quickly with default or new input values.

The invention was to simulate, in a more versatile domain, the working environment in an analog studio with a synthesizer, where modules can be physically connected and the patch represents the flow of a signal which is transformed by a system. This modular approach makes it easy to visualize (or auralize) each step in the construction of a sound.

Educational Tool
Patchmix could be used as an instructional aid for both the musical (timbral) and programming aspects of synthesis. Those without much programming experience can see more easily the connection between the modules of C code within the Cmix function and the connected unit generator on the screen.

The Object-Oriented Approach
C++ objects operate transparently within Patchmix to create the desired instrument. The instrument object calls upon an n-ary tree class to structure the relationships between the instances of unit generators. Oscillator, buzz, filter and other objects are derived from a base unit-generators class. They create and declare the variables needed and incorporate user-specified data from parameter fields in the data. The instrument builder translates the relationships of modules from the tree into Cmix assignments and function calls. The sound file class creates (by running the instrument), rescales and plays the sound file.

Option:
One possible limitation of this program was that the specified set of unit generators may not always be adequate. There may always be a need for a new filter equation or more radically different addition to the sound synthesis process. Therefore, it is still possible within Patchmix, as with Cmix, to take advantage of its functionality and yet extend its scope by incorporating a function that one has written. The user creates and names a new icon, supplies information about how it relates to the construction of the instrument, and places it appropriately in the flow chart. Patchmix will call the user-written function within its Cmix function.

Another feature which seemed helpful was to provide an easy way to construct an instrument which

ICMC GLASGOW 990 PROCEEDINGS 274
produces many events combined into one "note". For granular synthesis or other techniques which require a number of discrete events, the "framing" option was devised. The user can specify the relationship of the sample loop to the framing loop, and how parameters will change between frames.

Conclusion
Patchmix is an intelligent interface which can construct Cmix instruments from a user-specified diagram of unit generator icons. It is versatile enough to create complex and flexible instruments, including those with user-written functions. Because it is graphical, the user can work visually and construct a flow chart of modules to produce the desired sounds. The X windowing system provides a highly configurable graphical platform. Although Patchmix bypasses the need for knowledge of C programming to create instruments, it can help the user to learn Cmix and programming since it produces usable Cmix functions. Because default values are supplied by the interface, the instrument can be generated with a minimum of information from the user.

References