The Composer's Toolbox
Jonathan Nathan and Dale Striez
Colby College, Waterville Maine USA

Abstract
The Composer's Toolbox project grew out of a need to fill the gap between 'closed-system' commercial packages and high-end composition languages. Its primary aim is to facilitate composers' efforts during the precomposition process. This has been accomplished by creating a series of configurable "tool" objects (both MIDI and DSP) which support such diverse processes as graphic and algorithmic control of MIDI event streams, algorithmically controlled evolution of harmonic structures, and DSP operations on AIFF sound files. All tools have been designed as small, self-contained modules (many with graphic representations) which can be implemented individually or in groups based on a composer's particular needs. Practically all modules are usable "out of the box" via default characteristics, but also possess options for customization at a variety of levels. Since it is the authors' intent to create an environment that can be shared among, and extended by, its users, complete, copiously documented source code is provided for all modules. The system is implemented in Macintosh Common Lisp running CLOS and runs comfortably on a Macintosh II with 4 MB of RAM.

Introduction
The Composer's Toolbox is a software package that was designed to help composers who want to work with ideas quickly and easily during the precomposition process. While visual artists have always had the option of beginning a work by "roughing out the canvas", putting in general outlines of images to come, organizing the visual space, etc., composers have largely been relegated to working at the more micro-level of individual events (pitches, rhythms, etc.). Although the advent of computer sequencers, "smart" notation programs, etc. has had a major role in the simplification of many compositional tasks, most of these applications still require that a composer go through numerous manual and time-consuming single-event data entry processes before ideas can be dealt with on the gestural level. And while languages like C-Musuc, Pla, and CommonMusic have given a long way to provide algorithmic control over musical data flow, their steep learning curves and non-real-time nature make them less than satisfactory for the free-wheeling nature of the early stages of the sketching process.

The Composer's Toolbox attempts to provide an environment in which many menial tasks of data manipulation have been automated, allowing a composer to spend more time considering the larger issues of musical structure, flow, etc. The package consists of two modules, one devoted to MIDI input and output (CT-MIDI), and one to processes involving the manipulation of sampled sound data (CT-DSP).

An Overview of General Features:
• Automatic and/or simplification of numerous data manipulation tasks
• Intuitive, user-extendable graphic interface
• Complete, fully documented source code for the entire environment
• Primitives written in device-independent CLOS
• Modules run under Macintosh Common Lisp, allowing the user full access to both the Macintosh toolbox, and the riches of Lisp for further extensions and modifications
• The package is available free of charge to anyone interested in using it; the only stipulation being that the author be given access to any extensions, bug fixes, etc. that are undertaken

The Toolbox Interface Hierarchy
The Composer's Toolbox was designed to allow the user a variety of interface options. At the lowest level, the module's primitive functions can be called in Common Lisp on a command-line basis. At this level users have complete flexibility to use or modify the functions at will. On a slightly higher level, dialog boxes and menus have been created to allow users to manipulate features and MIDI events with minimal knowledge of Common Lisp. The dialog box displayed in Figure 2 is an example. On an even higher level, graphical displays provide a true "point-and-click" environment (see Figure 3). All interface elements have been defined as objects, and can take on any form the user sees fit for new or extended components.

ICMC 242
I. The CT-MIDI module

Module Highlights
- Data objects that allow the representation of MIDI data collections as macro-events which can be manipulated on a variety of levels
- Extremely powerful, flexible, and user-extensible search and transformation shells
- A series of predefined functions which provide a wide variety of options for searching and editing data (globally or individually) within macro-objects
- Apple MIDI Manager compatibility
- The ability to read and write standard MIDI format files

The CT-MIDI module was designed to provide an easy and intuitive means of testing out "what-if's" during the precomposition process by providing a collection of data structures and functions which allow the building up and manipulation of complex hierarchical structures. It makes no pretense at being a full-blown sequencer or MIDI-composition package (although its primitives could certainly be used to form foundations for either). It doesn’t provide for traditional notation, nor does it attempt to replicate the extensive recording and playback options that can be found in commercial applications like Performer™ or Vision™.

The CT-MIDI module's central repository of data is an object called a "Gesture". Gestures contain three "slots". The first, the name slot, holds a name used to refer to the object. The second slot, the event-block slot, holds a pointer to an Event-Block object, which in turn points to a series of "Midi-Event" objects. The third slot, the Sub-Generates slot, holds pointers to zero or more other gesture objects along with offset times of each relative to the parent. A gesture's sub-gestures (which are just other gestures) may contain, in addition to their own event-blocks, further sub-gestures, allowing for the nesting of materials to virtually any level, while maintaining complete autonomy among all the elements. A gesture, together with its sub-gestures, the sub-gesture's sub-gestures, etc., may refer to many thousands of MIDI-events. However, it is still regarded by the Toolbox as a single entity. As such it can be manipulated (copied, transformed, etc.) without dealing directly with any of its component parts.

Figure 1 The Gesture and its related objects

An Overview of CT-MIDI Functions

The CT-MIDI module provides the means to manipulate gestures and their sub-gestures as a single unit, in groups consisting of subsets of the MIDI events of the gesture, or as individual MIDI events. For manipulating gestures as a whole, the package provides means to create, delete, and copy gestures; to add or delete sub-gestures; and to change sub-gesture offset times. Gestures can also be "agglomerated", by collecting all events from the gesture's sub-gestures "tree" and putting them together on one level in the event block of the parent gesture. Once a gesture is created, the user can easily add new events to it, appending to the events already in the gesture, merging them with existing events, or shifting the current events to create a time slot for the new events.

One of the most useful features of the CT-MIDI module is its ability to search for events based on a wide variety of criteria and then to transform those events in a variety of ways. The searching tools provided with the module allow the user to find almost any collection of events of a given type in a gesture tree (including non-contiguous events). Search criteria can take the form of built-in search functions, custom user-defined search functions, or a combination of the two. Built-in functions provide options for searching for data within MIDI-events that are above, below, between, among, or equal to user-specified values. A random choice option is also available. Search functions can also be combined using
conjunction and disjunction (e.g., the user could search for all notes of pitch greater than MIDI note-number 60 or less than 20, or for all notes for which the MIDI note-number is greater than 90 and the velocity is between 20 and 40).

Once the desired events have been found, they can be manipulated in a variety of ways. Standard editing operations (delete, cut, copy, paste) are supported and undo-able. Specific data values (e.g., channel, or data) within selected events can also be modified by choosing from the variety of built-in transformation functions or by creating new custom-defined ones.

Interacting with the Search/Transform Shell: an Example:

To see more specifically how the searching and transforming of events is accomplished, consider the following hypothetical problem and the way CT-MIDI allows the user to solve it. Suppose the user wishes to transpose certain notes in a gesture "G" up an octave. Suppose that the notes that are to be transposed are all those notes that have MIDI note-numbers between 0 and 40 and velocities over 80, or that begin at least 15 seconds after the beginning of the gesture. The user initiates the process by calling up the dialog box shown in Figure 2. This dialog box allows the user to transform, cut, copy, or delete selected events from a gesture and its subgestures. Using a series of pop-up menus, the user selects which operation to perform, the gesture to perform it on, and the type of events in that gesture to be acted upon. They then indicate the appropriate selection criteria for finding the desired events within the gesture. Once the selection criteria has been established, the user decides which transformation functions to apply to the selected events. Clicking the "Transform" button with the indicated selection criteria specified in Fig. 2 will cause the selected events to have their pitches incremented by 12, thus raising the notes an octave.

Connections with the outside world

CT-MIDI is compatible with Apple's MIDI Manager. When the module is loaded, input, output, and time ports are automatically created which can then be connected to other MIDI Manager-compatible devices using calls from within Lisp or via the Patchbay™ application. Issuing the command to "play" any gesture will cause CT-MIDI to parse the gesture and its family of subgestures (or, if specified, only the top-level event-block) and create the appropriate MIDI data to be sent out the output port. Parsed data can also be streamed to a buffer for playback at a later time or on cue from real-time input. Gesture data can be stored in standard MIDI file format, or as Lisp text files. When loaded, the Lisp text files will recreate all gestures and their various subgesture trees.

II. The CT-DSP Module

Module Highlights

• A collection of algorithms which implement DSP functions not commonly available in

ICMC 244
commercial packages
• Configurable graphic tool objects which provide a friendly interface to the algorithms
• The ability to read and write standard AIFP sound files

Since the bulk of programming to date has been directed toward implementing and testing the MIDI component of the Composer's Toolbox, major portions of this CT-DSP module remain under development. The overall goal of the module is to provide a user-friendly, graphically-oriented gateway to useful DSP functions which are not currently available in commercial soundfile processing packages. These include such processes as dynamic panning, Dopplerizing, and Phase Vocodering. As in the case of CT-MIDI, there has been no attempt made to replicate the extensive recording, graphic editing, and playback facilities present in commercial packages such as Digidesign's SoundTools.

In addition to a set of functions which support AIFP soundfile input, buffering, and output, the CT-DSP module currently contains one fully functional process which allows the dynamic stereo panning of mono soundfiles. Graphic interface with the panning algorithm is provided by means of a "rubber-line" tool, in which the user can specify position within the stereo field by simply dragging segments of the line toward the top or bottom of its bounding rectangle. The panning tool is an excellent example of the Toolbox's implementation of high-level graphic tool objects. It has been created in a completely modular fashion, so that it can be implemented along with other tools to create customized data manipulation interfaces. Since its output is both configurable and largely generic, it is available for use in a wide variety of situations which require control over data which is most effectively represented within an X-Y coordinate system (e.g., time-varying processes like Dopplering and dynamic pitch shifting).

**Figure 3. The Curve Pan Interface**

Future Directions

Plans for the immediate future of the package are threefold:
1. Work has already begun on extending the CT-MIDI module to include more graphically-oriented interaction with gestures and the processes which work upon them.
2. The collection of DSP tools and graphic interfaces to them will gradually be extended.
3. Interfaces with third-party hardware, most immediately Digidesign Corp.'s Sound Accelerator card, are planned to bring DSP tools to near real-time functionality.

It is our assumption that as the Composer's Toolbox is used in actual composition, needs for additional tools (both interface and algorithmic) will arise. By making the system as open-ended and flexible as possible, we have attempted to create an environment within which improvements and extensions can be easily implemented and integrated. By providing fully documented versions of the package free-of-charge to interested users, we hope to involve a larger community both in defining future directions for package development, and in contributing programming efforts to that development.

TCMC 245