Krayola
A Flexible, Robust, and User-friendly Precompositional Sketching Environment
Dale Skrien and Jonathan Hallstrom - Colby College, Waterville, Maine
(djskrien@colby.edu and jhalstr@colby.edu)

Abstract
The Krayola sketching environment provides composers with an easy-to-use, functionally robust means of experimenting with musical raw materials (themes, harmonic patterns, timbral flows, etc.) during the precompositional process. Its simple and intuitive top-level interface functions like a sophisticated "paint" program for MIDI events, with the added capability of allowing the introduction of algorithmic controls over MIDI data pattern creation and flow (rhythmic patterns, harmonic progressions, velocity patterns, etc.), in essence providing an environment within which code and mouse position can be freely combined to allow users to paint the output of algorithms.

1. Overview of the Composers Toolbox System
The Composers Toolbox is a collection of software tools aimed principally at aiding composers during the precomposition process. The environment was developed in, and runs under, Apple's Macintosh Common Lisp. At present, the package contains two sub-environments, one devoted to DSP operations on sampled sound files, and one devoted to the manipulation of MIDI event streams. The system both reads and writes standard AIFF (Audio Interchange File Format) and MIDI files, and is compatible with Apple's MIDI Manager. The Krayola algorithmic painting module was developed to provide users with an easy to use graphic link to the system's extensive options for algorithmic control over MIDI data streams. As with all components of the system, the complete Krayola module, including a user's manual and copiously documented source code, is available free of charge (for non-commercial use) to the general computer music community.

2. The ToolBox's principal low-level MIDI data structure: The gesture object
The Krayola module is intimately linked with Composers Toolbox's fundamental MIDI data structure, the Gesture (see fig. 1). Gestures are data objects which serve as repositories for four types of information: 1) a name for the gesture, 2) a list of individual midi event types, 3) zero or more subgestures (which are merely other gestures), and 4) an offset time that is used to determine when the system should output the gesture to the MIDI Driver in relation to its parent gesture (if any).

In addition to performing the obvious function of referring to things like motives, phrases, etc., gestures can do a number of other things. Since a gesture can refer to anything from a single MIDI event (a pitch change message, for instance) to an entire composition, and gestures can be nested to any level, it is possible to create extremely complex hierarchical structures that can still be manipulated as single entities.

fig. 2: The gesture object

A gesture's subgestures may contain, in addition to their own event lists, further subgestures, etc., allowing for the nesting of materials to virtually any level, while maintaining complete autonomy among all the elements. A gesture, together with its subgestures, the subgesture's subgestures, 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. The Krayola module takes full advantage of these various manipulation options through an intuitive graphic interface.

3A.2 108 ICMC Proceedings 1993
3. The Krayola event entry interface

At its simplest level, the Krayola event entry interface is quite similar to that of a typical sequencer program in “piano roll” mode. Using a combination of the note placement and arrow tools (see fig. 2), individual notes can be entered in the time/pitch grid and selected either individually or in groups (using the standard Macintosh shift-click or dragged marquee selection methods). Once selected, events can be dragged to change their duration or position in the pitch/time grid, or option-dragged to replicate them.

Fig 2. The Krayola main display window

4. Controls over the placement of horizontal collections of events

The true power of Krayola emerges in the additional controls it provides for the placement of pitches, rhythms, etc., most of which are available through the Draw options dialog (see figure 3). In its default state, this dialog contains the options available at the simple note entry level: MIDI channel is set to 1, velocities will all take the value of 60, and durations and offsets will be determined by the place where the mouse is clicked in the pitch/time grid (indicated by the symbol x in various fields of the configuration window). To provide for more complex collections of pitches, durations, etc., the user needs only to alter these default values. By entering multiple integer values into any of the configuration slots, these values will be iterated as the mouse is clicked repeatedly in the main display window. In the case of offsets and durations, the interpreter “watches” the position of the mouse after it has been clicked and invokes a new iteration of the pitch/velocity/channel data when the requested duration/time offset has elapsed. X’s (mouse position dependent values) can be mixed freely with specifically designated pitches, durations, etc. to provide varying degrees of control over how movement of the mouse affects the placement of events in the window.
5. Controls over the placement of vertical collections of events
The Draw parameters interface also provides straightforward options for control over the creation and placement of chords. Placing note values in parentheses causes those values to be drawn simultaneously. If an X (mouse position dependent value) is entered as the first value in a parenthesis-delimited group, all subsequent values will be interpreted in terms of their relative position to the point at which the mouse was clicked (e.g., X 2 -5 will result in a 3-note chord with pitches two semitones above and 5 semitones below the value corresponding to the point where the mouse was clicked). Placing an X after a series of values will cause the same values to be interpreted as literal MIDI numbers (e.g., (50 62 X) would result in a 3-note chord containing the pitches 50, 60, and the value corresponding to the point where the mouse was clicked. Sequences of chords (both literal and mouse position-controlled) and individual pitches can be freely combined.

6. Controlling rhythm and duration
By entering multiple offsets and durations in their respective slots, it is possible to create mouse-position-controlled rhythmic patterns. Used in conjunction with the above options for creating chord and individual pitch sequences, it is possible to use these patterns to control both the horizontal and vertical flow of the composition in very sophisticated ways.

7. Algorithmic controls
Any (and all) control parameters can take LISP functions as their arguments. The environment provides options either for evaluating the function as a one-time basis (with the results of the evaluation placed in the parameter slot for further editing), or for invoking the function repeatedly as the mouse is clicked in the main display window (including allowing the value that results from the click to be used as input to the function). Both hand-entered patterns and function calls can be caused to revert the passing of data at each mouse click, or to continue to cycle through the generated values in subsequent clicks. Functions and hand-entered values can be mixed within individual slots to provide virtually limitless controls over the painting of pitch/time collections.

8. Display controls in the main event window
Krayola provides a wide variety of options for controlling the display, selection, and editing of gesture data. By grouping a series of gestures under a single parent gesture and then displaying the parent in the main Krayola window, it is possible to view and edit multi-“track”, multi-channel event streams simultaneously (including the ability to cut, copy, etc., across multiple gestures). On systems utilizing a color or grayscale monitor, events can be “colored” either by channel or by gesture, providing a clear and concise view of the relationships existing in the data being edited. When working with large amounts of data, or when fine-tuning individual events, the interface allows some slices to be selected and zoomed-in or out, including a “full zoom out” option which will fit all data associated with a gesture (and if requested its subgestures) in a single window frame, thereby allowing the user to “step back” at any time and view an entire work in a single window “pane”.

3A.2 183 ICMC Proceedings 1993
9. Displaying subsets of data: the display/select interface

Using the display/select options interface (see fig. 4), subsets of data can be isolated for further editing. This interface allows the user to build complex collections of "and/or" clauses (which can include both system and user-defined algorithmic controls) which will be applied to any data which is currently being displayed in the main window (including the collecting of non-contiguous data). Once the user has isolated the data he/she wishes to display, the interface can be switched to "select" mode to gather notes for editing (see fig. 4).

Fig. 4. The display and select control dialogs

10. Editing controls

One of Composers Toolbox most powerful features is its transformation shell. In contrast to a typical MIDI sequencer program which is only able to do simple transformations of data (transpose, reverse, scale time, etc.), Composers Toolbox provides the means for introducing algorithmic controls over data transformation in virtually limitless ways. Once the user has isolated and selected the data he/she wishes to transform, the edit interface can be invoked (see fig.5). In addition to standard, system-defined transformation procedures, the interface provides the means to introduce user-defined functions to control the transformation process.

Fig. 5. The transform options dialog

11. Conclusions

The addition of the Krayola MIDI paint module to the already robust collection of tools contained in the Composers Toolbox extends the usefulness of the system by providing a link between the virtues of a typical graphics-based sequencer program and those of a full-fledged algorithmic composition language. Currently under development is an extension of the Krayola paradigm to include the display and manipulation of whole gestures as objects in the same way that it is currently able to work with the events they contain.