The Max Timeline Object

David Zicarelli
Opcode Systems, Inc.
3950 Fabian Way Suite 100
Palo Alto, California 94303
zicarell@well.sf.ca.us

Michael Lee
CNMAT, UC Berkeley
1750 Arch Street
Berkeley, CA 94709
lee@cnmat.berkeley.edu

Abstract

The timeline object adds a graphical scoring capability to the Max programming environment on the Macintosh. Composing with the timeline involves first selecting from a repertoire of *actions* which provide hardware control or implement specific compositional algorithms, then building a score of messages in a graphical editing window. This paper discusses the motivations for the project, provides an overview of its organization, and suggests how users will work with the object.

1 The Need for the Timeline Object

The development of the Max timeline object was motivated by the perceived lack of a good way of representing and organizing sequential events when writing a "piece" with Max. But rather than worry about the clarity of event representation, we decided to focus on how the ability to organize events sequentially would foster a way of working with Max that might be more amenable to some people. Currently, the typical Max user who wishes to incorporate some kind of pre-stored material within his/her piece typically draws a graph in a *table* or types numbers into a *coll* object. These numbers are initially meaningless (Zicarelli 1991) and later become notes, durations, or other quantities once they have been transformed into messages to control some piece of equipment.

While the storage of meaningless numbers is a powerful technique with a long history in computer music, one cannot escape the fact that the process of storing and editing numbers *qua* numbers is not always facilitated by a completely generic procedure. To a word processor or spreadsheet, one's numbers could just as well be today's values of New York Stock Exchange securities beginning with the letter 'N' as a carefully wrought melody. We speculated that tools which could be customized to reflect the application of one's numbers might facilitate the process of creating both the numbers and their surrounding meaning-giving context.

There are two other approaches to the scoring problem within Max that we have seen. Miller Puckette's *explode* object (Puckette 1990) uses displays bars representing note events against a grid. These bars can be tied to additional numerical values. Although one is not forced to use explode's events for anything in particular, its interface tends toward the use of the vertical location of the event in the grid for pitch and the horizontal values for event start and duration. One novel feature of explode is that it does not contain its own source of time advancement, relying on the Max user to write a patch that steps through the events. This means that rather sophisticated interactions between the score and what might be happening elsewhere are possible. Piché and DesMeures' more recent object *MaxGen* (1991) emphasizes time functions. It has a more complete interface than explode and also allows the user to define up to eight functions that output values simultaneously and independently.

The timeline differs from these objects in that it is concerned not just with the problem of making the score, but making the entire piece that the score will play. One embeds explode and MaxGen (and table and coll for that matter) within a larger patch. If you want to distinguish multiple "tracks" of output or different elements within the same event, you need to work with different outlets coming from the object.

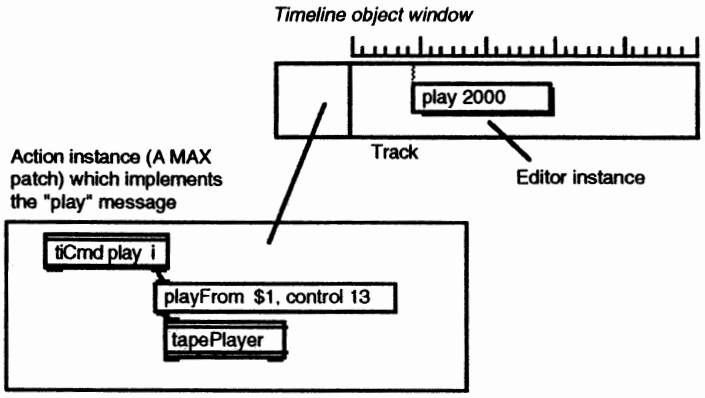
The timeline object can also be used this way, but we intended that its typical use will turn the approach of explode, table, coll, and MaxGen inside-out. Rather than embed the timeline within a patch, you embed patches within a timeline. We believe this provides a "structure" to a piece created with Max and seems more "object-oriented" because the patches you embed (what we call *actions*) can be re-used in any number of timelines. The timeline also emphasizes that text messages as well as numbers can be meaningless. Within an action, you can define a "syntax" that the events in a timeline will utilize.

In general, the timeline object addresses the problems of scoring within Max not just at the level of representing or rendering the score, but in connecting the score to what it is controlling. Indeed, while the representational aspect of the timeline object has probably consumed the most effort in the project, it is probably its least novel component. However, given our extensible framework, we envision that some of the best aspects of scoring approaches in explode and MaxGen could be incorporated into what we call *editors* for the timeline object. Since these editors are just Max external objects that respond to a specified group of messages, we don't need to write them ourselves. The details of how one goes about writing a timeline editor are too boring to recount in any detail here.

Finally, the timeline follows explode's principle and not require the use of its own timing. The timeline can operate in an internal millisecond clock mode which is useful when sync-ing MIDI events to a Quicktime movie. But in general, the timeline maintains its own scheduler which watches a number we call *fictional time*. Anyone, including patches embedded with a timeline, can mess with this number; moving it forward, backward, or keeping it still until a condition is met.

2 Organization of a Timeline

The figure below outlines the relationship between the various parts of the timeline. The timeline object itself consists of a window which contains a set of *tracks*. Time runs from left to right in a track, and tracks are arranged vertically. Tracks contain *events* which are edited by the user with instances of *editors*. The events are ultimately messages sent to *actions*, which are either C programs or Max patches that contain special timeline-related objects. An action can also control the timeline itself, changing fictional time or muting a track (just to name two examples). Since actions are not "procedures" which are called but are always waiting to do something, they can be sources of input as well as output. For example, an action might receive an Apple event which advanced the timeline to a specified marker.

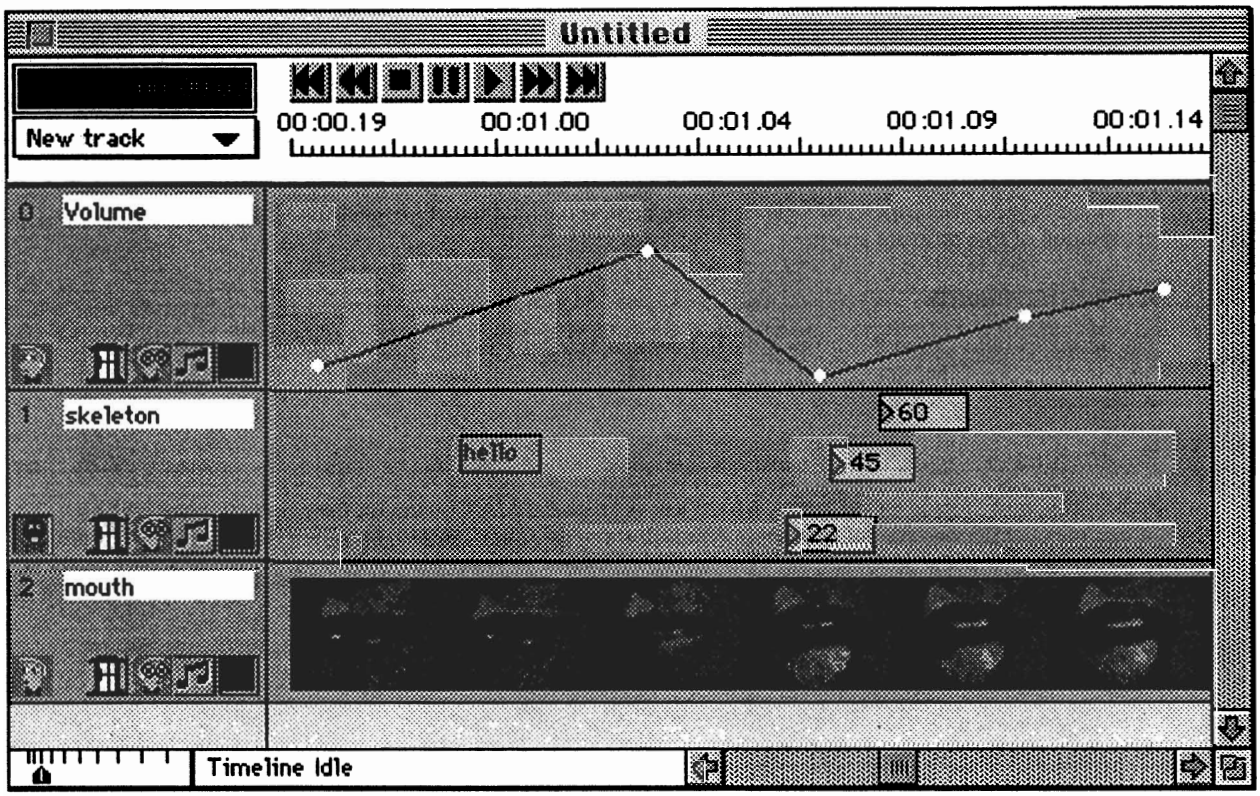


A small part of an Action patch is shown above. It receives the command "play" from a Track. This command is then translated into a message (contained in a traditional Max message box) which is output to a fictitious object called "tapePlayer" that presumably would start the transport of a computer-controlled tape recorder. The idea is that the complexity of the control protocol is hidden within the action (and/or inside the object that implements the protocol), so the timeline user need only type the word *play* into a box to make the tape start moving. Alternatively, the action could be written in a piece-specific way so that the action of playing from frame 2000 was named "Overture." This word would then be displayed in the track; a somewhat more meaningful indicator than simply a number associated with a section start.

A common way to construct an action patcher is by using the special Max external object *tiCmd* ("timeline command"). The typed-in arguments to tiCmd define the command verb and the data types for any arguments to the command. In our example, when time reaches the left edge of the editor box in the track, the message "play" is sent to the action patcher associated with the track. When this happens, the message's arguments (if any) are sent out the outlets of the tiCmd object, according to the tiCmd object's type specification. The leftmost outlet of tiCmd always outputs a bang when it receives a message from a track, and an optional argument allows an additional outlet to send out a bang when time reaches the right edge of the editor box in a track.

3 The Timeline User Interface

Below we have reproduced a typical screen from a timeline window showing tracks containing some of the editors we've developed thus far. There is a track with a function editor, a track containing editors for standard text messages and number boxes (that send the *int* message), and a track containing a thumbnail editor for the movie object. Editors are either linked to the Note that the middle track is associated with a Max object that has been written to work with the timeline—it places its own icon at the left side of the window. Only two or three additional messages are required to turn any existing Max object into an action or allow it to be accessed directly by messages within a timeline track when it is used within an action patcher.



To create a new track, the user chooses one the currently loaded actions in the New track pop-up menu above the names of the tracks. Or, she can open a new patcher or external object to use as an action by choosing Other... from this menu. When you want to create a new event in a timeline track, you click to get a pop-up menu of all the command verbs defined in the action associated with the track; for example those that have been defined by tiCmd objects in the patcher. You can edit an action patcher while it is being used in the context of a timeline. When you add or delete tiCmds or other timeline-related objects, the track of the timeline is appropriately adjusted. This allows you to develop a score and the processes the score will control in parallel.

4 Applications and Future Directions

We will conclude by suggesting how we expect the timeline object to be used and expanded. First, we envision that a library of actions for various hardware devices could be created. For example, the essential commands to control a laser disk player could be placed together in a patcher. When a user wishes to employ a laser disk player in her piece, she creates a track associated with the laser disk action patcher, then adds control messages that will happen at various times.

For the individual composer, the action represents a way to make re-usable "instruments" that can be controlled from several different scores. If each form of control over an instrument is given a (meaningful) name in a tiCmd object, the timeline can be used to create scores. Such instruments could range from a simple MIDI note player to an autonomous algorithm which received high-level "guidance" from the timeline score.

When deciding how events should be arranged in time, remember that the horizontal dimension does not have to be real time. It can just represent a "tag" for an event. If a timeline's source of time is external, fictitious time can jump instantly between these tags and never move continuously. Since it is often the start of an event which is tagged, moving time to a tag location can be used to trigger ongoing processes which are not directly synchronized to the "passage" of fictitious time in the timeline score.

Finally, the timeline provides a framework for the integration of Max-style interactive performance in the context of quasi-linear multimedia material. For example, Max objects within an action patcher could allow a performer to navigate within the presentation space defined in the timeline through artful changes to the value of fictitious time. Actions can also serve to buffer the performer from the idiosyncrasies of equipment which was never designed to be "played" as if it were a musical instrument. The numerous challenges posed by the performance of multimedia (not to mention the questionable aesthetic value of "multimedia" itself) are beyond the scope of this paper, but we believe the timeline object will facilitate a fruitful exploration of this problem space.

Acknowledgments

The authors would like to thank Jim Montgomery and Marsha Vdovin, formerly of Opcode Systems, Inc. for their support of the project, and the Center for New Music and Audio Technologies for supporting our collaboration.

References

Piché, Jean, and Stéphane DesMeures, MAXGen: A graphical MAX object for the generation and editing of complex continuous control functions, *Proceedings of the ICMC*, p. 375, Montreal, 1991.

Puckette, Miller, Explode: A user interface for sequencing and score following, *Proceedings of the ICMC*, pp. 259-261, Glasgow, 1990.

Zicarelli, David, Communicating with meaningless numbers, *Computer Music Journal*, 15(4), pp. 74-77, 1991.