DEVELOPING A WINDOWS AND MACINTOSH GRAPHICAL USER INTERFACE FOR THE MATHEWS RADIO-BATON SYSTEM

Richard Boulanger
rboulanger@berklee.edu
Berklee College of Music
Boston, MA 02115 USA

Young Jun Choi
choi@cSounds.com
Berklee College of Music
Boston, MA 02115 USA

Luigi Castelli
superbigio@yahoo.com
Berklee College of Music
Boston, MA 02115 USA

Abstract

The Mathews Radio Baton System is a rugged, portable, and wonderfully responsive 3D hardware controller supported by an ever expanding set of C functions and C++ classes. These continue to facilitate user programmability and customization of the system. In addition, the package also includes Mathews' text-based "expressive" sequencer – *The Conductor Program*. As it stands, the current Mathews Radio Baton System represents an affordable and powerfully expressive virtual controller. But there has always been a serious missing piece to this text-based system. The lack of an intuitive graphical user interface has turned away many a potential user. This demonstration will showcase two fully working graphical interfaces for The Mathews Radio Baton. The first uses *Max/MSP* and the *MegaMax* collection to visually model and literally map all the hardware and software functions of the Radio Baton System on the Macitosh. The second uses the Windows API and Visual Basic to do the same for the Windows PC. Both will be demonstrated running on laptop computers and both controlling real-time software synthesizers – *MSP* on the Mac and *Csound* on the PC.

INTRODUCTION

The Mathews' Radio-Baton System (Figure 1), is a 3D MIDI controller that tracks and maps the X, Y and Z motions of the ends of two batons held in the hands of a performer (Figure 2).

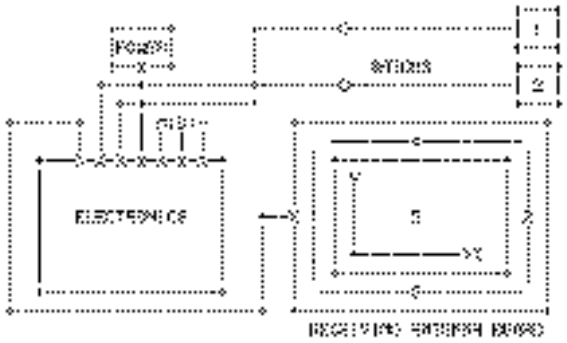


Figure 1. Schematic of the Mathews Radio Baton System. *Batons 1 & 2* are connected to the *Electronics Box*. MIDI In/Out/Thru connectors allow communication between the system and a standard PC. An additional cable connects the *Receiving Antenna Surface* (comprised of 5 receiving antennas numbered 1-5).

Figure 2. Max Mathews performing on his 1999 Radio Baton.

How the Radio Baton Works

At the end of each baton is a small radio-transmitting antenna. As show in figure 1, on the receiving antenna surface are 5 receiving antennas – four long thin antennas arranged along the four edges of the board (#1,2,3 & 4) and one large antenna covering the entire center area of the board (#5). The closer a baton is to a given receiver, the stronger the signal at that receiver. By comparing the signal strengths at the #1 and #2 antennas, the computer in the electronics box can determine the X position of the baton. Comparing the #3 and #4 strengths gives the Y position. The #5 strength gives the height above the board or Z position. The two batons operate at different frequencies and thus can be independently tracked.

A low frequency of about 50kHz is used for the radio signals. It is appropriate to describe the Radio-Baton as a capacitance sensor. The radio-frequency signal is really used as a method of measuring the capacitance between a transmitting antenna electrode and a receiving antenna electrode. The relationship between antenna signal strengths and XYZ coordinates is complex but the Radio-Baton computer is good at making the needed conversions. A table lookup procedure is used to speed the calculation. The Radio-Baton can compute the positions of the batons every 4 milliseconds. The accuracy of measurements is about 1 part in 100. The X and Y information is linearized and mapped onto the standard MIDI range of 0-127. The Z data decreases from about 100 when a Baton is on the antenna surface to about 30 when the Baton is about 3 feet above the antenna surface.

Triggers

In addition to providing XYZ data, one of the most important functions of the Radio-Baton is to send triggers over MIDI. A trigger can be generated when either baton touches an invisible plane that can be positioned at various heights above the antenna board. This plane is called the *Hit-Level*. In order to avoid *double triggering*, a second plane is also positioned slightly above the *hit-plane*. This plane is called the *set-level*. A baton must be raised above the *set-level* before a second trigger can be generated. This method for generating triggers and avoiding double-triggers using *hit & set levels* works well for most musical purposes.

WINCONDUCTOR

Initially written in the spring of 1999, and revised and expanded in the spring of 2000, Young Jun Choi's *WinConductor* program (Figures 3), was one of the projects that resulted from an Interval Research, Inc. grant in support of "Alternate Controller Research" at Berklee. This grant provided for the purchase of several PC's and Radio Baton systems (loaned directly to students) and brought Max Mathews to Berklee for two weeks of daily C++ programming seminars and Radio-Baton performance workshops.In the main window, as show in figure 3, menus allow the user to configure and store the MIDI setups; the baton communication channels, and to determine which windows to simultaneously display on the screen.

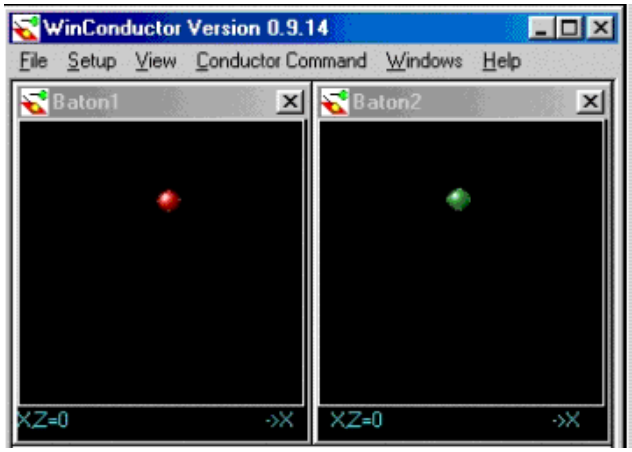


Figure 3. The Main Window for Young Choi's *WinConductor* Program. In this view, the X and Y position of batons 1 & 2 are continuously displayed as color coordinated balls.

A second window, shown in figure 4, allows the user to monitor the MIDI date and system status directly. Further, it is possible here to thin the amount of data selecting whether the system should poll the batons in a 20ms range from 40ms to 100ms.

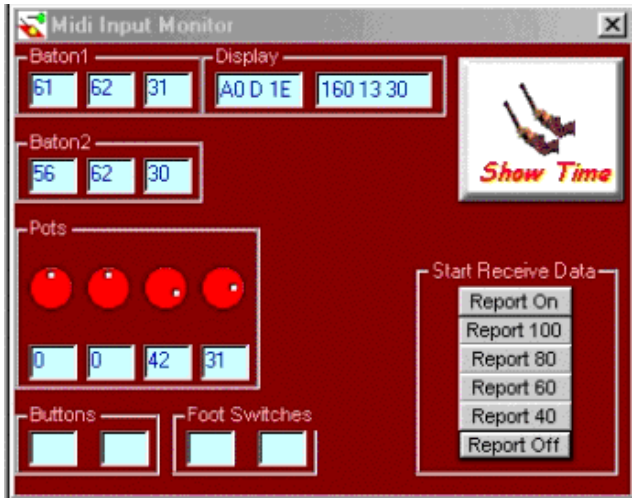


Figure 4. The MIDI Input Monitor window. Baton locations, plus the pot settings and button/foot-switch status (found on the electronic box), are continuously displayed. The "Show Time" button opens a single "full-screen" continuous XYZ display of both batons. "Controller thinning" by user selectable reporting is also supported.

An exciting new feature of the *WinConductor* program is the "Baton Mapping" dialog that allows the user to map, scale and channelize all the systems controls: XYZ and "Whack" from either baton plus Pots, Buttons and Foot Switches found on the electronics box. Most importantly, these mappings can be saved and recalled.

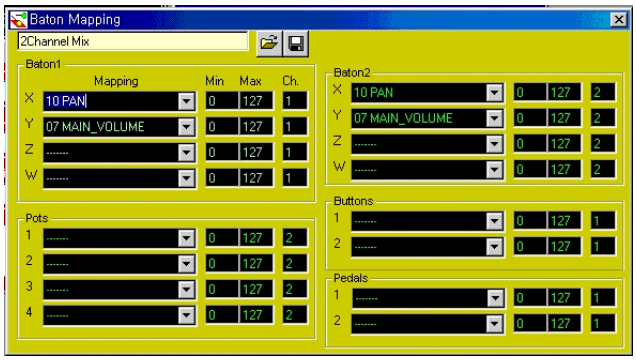


Figure 5. The *WinConductor* Baton Mapping dialog allowing one to map, scale and channelize XYZ and "Whack" from either baton as well as the Pots, Buttons and Foot Switches found on the electronics box to any MIDI controller number and save/recall these mappings.

A final feature of the *WinConductor* program is the "Easy Play Preset" dialog. Selections here assign each baton to a separate MIDI channel, allow the user to use pots 1 and 2 on the electronics box to chose a patch, and maps X to MIDI note, Y to velocity, and Z to controller 1 (Figure 6).

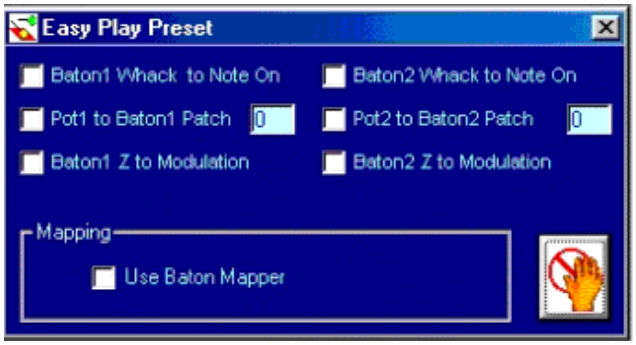
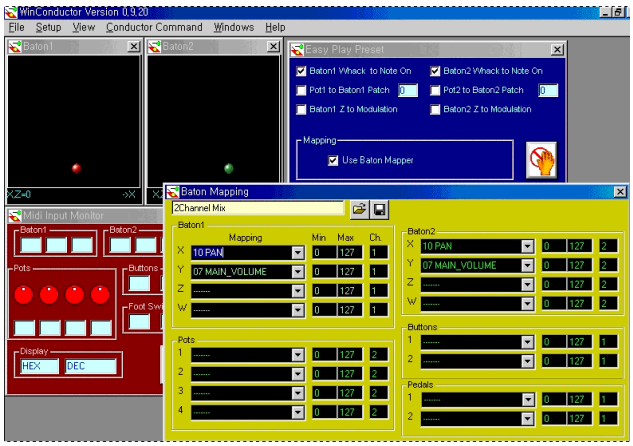


Figure 6. The *WinConductor* Easy Play Preset dialog with "all notes off" button and a check-box to launch and use the "baton mapper" settings.



BATONMAX

Initially written in the fall of 1999, and revised and expanded in the spring of 2000, Luigi Castelli's undergraduate thesis, the *BatonMax* program (Figure 7), grew out of work by David Zicarelli and Erik Oña at the 1999 June in Buffalo workshop. Like Young Jun Choi, Luigi Castelli was also fascinated by the Mathew's Radio-Baton System, but disappointed that it was mainly taught and hosted on a PC. Given that Berklee is primarily a Mac School and Castelli is a serious Max-MSP programmer, he set about to make the Radio-Baton more accessible to graphically oriented musicians in our department. Using Stephen Kay's MegaMax Collection of 3D objects and some custom graphical backgrounds and knobs, Castelli's interface, while being fully functional, has a custom and elegant look and feel (Figure 8).

Figure 7. The *BatonMax* Interface. The lower half of this panel replicates the pots, buttons and LCD display on the Radio-Baton electronics box. The upper section facilitates MIDI and communication setups and toggles the graphical displays and other program windows and panels.

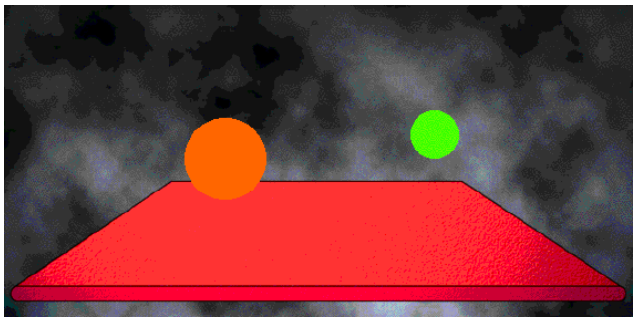


Figure 8. The *BatonMax* 3D display. Color-coded balls resize proportional to Y to create the illusion of distance and further the sense of perspective.

One of the most important features of Castelli's program is his implementation of the Radio-Baton's calibration routines (Figure 9). Convenient access to these controls makes the system much more reliable in concert situations where lighting and radio interference can "remap" and "rescale" all the controllers!

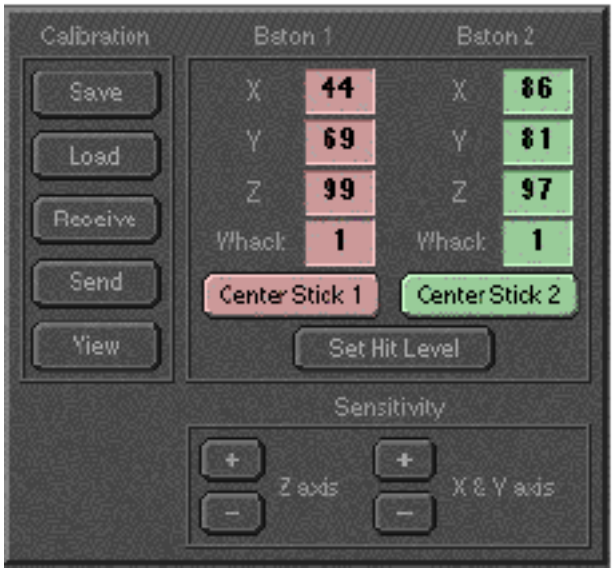


Figure 9. The *BatonMax* setup panel. Here the system can be calibrated and the calibrations saved and later downloaded back into the Radio-Baton computer.

Finally, Castelli truly enhanced the Radio-Baton system by adding a full-featured controller sequencer that allows the user to the user to record, play, solo, mute and overdub one or all of the Radio-Baton's controllers and play them back at any tempo (Figure 10).

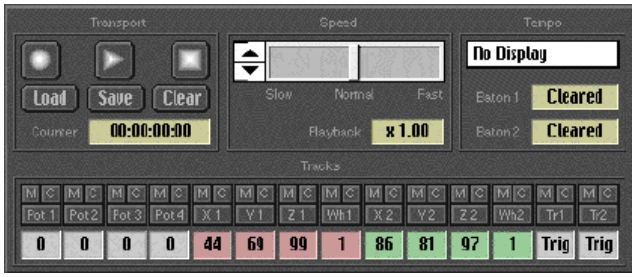


Figure 10. The *BatonMax* sequencer. This panel allows the user to record, play, solo, mute and overdub one or all of the Radio-Baton's controllers and play them back at any tempo.

CONCLUSION

The Mathews' Radio-Baton System is a wonderfully visual interface for computer music performance. It is fun to play and fun to watch. It makes possible the expressive performing of MIDI files and enables some wonderful real-time sound design explorations as well. But, for today's graphically oriented electronic musicians, the text-based nature of the system and its lack of support for the Macintosh have scared away many potential users. Inspired by Max Mathews himself and supported by Interval Research, two Berklee Music Synthesis Majors were able to put a face on the Radio-Baton System that has made it more immediately accessible to the MIDI musician of today.

Acknowledgements

The authors wish to acknowledge David Zicarelli, Erik Oña, Tobias Enhus, Bill Verplank, Max Mathews, and Interval Research, Inc. for their support of this research. Their input, feedback, advice, code, objects, abstractions, models and suggestions, as well as the financial support of Interval Research, Inc. have been instrumental in the development of the programs discussed in this paper.

REFERENCES

Boulanger, R.C. and Mathews, M.V. 1997 "The 1997 Mathews Radio-Baton & Improvisation Modes," *proceedings of the International Computer Music Conference*, Thessaloniki, September 1997.