PadMaster: banging on algorithms with alternative controllers

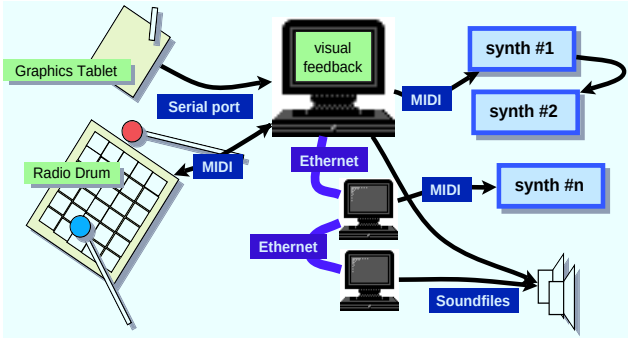
Fernando Lopez-Lezcano (*nando@ccrma.stanford.edu*)
CCRMA (Center for Computer Research in Music and Acoustics), Stanford University

ABSTRACT

This paper will describe the current implementation of PadMaster, a real-time improvisation environment running under the NextStep operating system on both NeXT hardware and Intel PCs. The system was designed with the Mathews/Boie Radio Drum in mind, but can now use alternative controllers, including widely available graphics tablets. The current version adds soundfile playback and algorithms to the preexisting palette of performance options.

1.0 The PadMaster program

PadMaster is a real-time improvisation environment written in Objective C that currently runs on NeXT workstations or Intel PCs running the NEXTSTEP operating system. PadMaster uses the MusicKit [4, 5, 6] as the basic foundation for controlling and performing musical events and the NeXT Soundkit as the basic resource for soundfile playback (an extension is in the works which will allow PadMaster to control other networked workstations for soundfile and or MIDI information playback).



2.0 Basic concepts: a virtual surface, controllers, pads and scenes

The performer interacts through the selected controller with a virtual surface that is modelled on the screen of the computer. This virtual surface contains one of more “Pads” which are the basic performance units of PadMaster. Pads are non-overlapping rectangular sections of the surface (of arbitrary size) and can be triggered through actions of the performer. Pads can control the playback of MIDI information or soundfiles either through sequences or algorithms. A Pad can react in different ways to a trigger event and can also link some of its control parameters to continuous position information provided by the selected controller. A set of Pads that are defined within the same surface are called collectively a “Scene”. PadMaster allows an indefinite number of Scenes to be defined. The performer interacts with the current scene and can switch between them during the performance, in effect redefining the behavior of the virtual surface. A Pad can be performing even though its Scene is not the current one, that is, the performer can trigger Pads in one of the Scenes and then move on to another one while the Pads keep performing in the background (they are not visible on the computer screen which always shows the state of the current Scene).

2.1 Controllers

PadMaster was originally designed to be controlled by the Mathews/Boie Radio Drum, which provides for six independent axes of control. The current software architecture has been opened to allow the use of alternative controllers.

2.1.1 The Radio Drum

Triggering of a Pad is effected by striking the surface of the drum with one of the batons and is further modified by the state of the two foot switches. When using the Radio Drum as a controller the number of available pads is constrained by its resolution, as the hit and position MIDI messages create a basic non-linear grid of 128x128 points. For the purpose of calibration and hit detection the surface is split into a matrix of 10x12 tiles, which are the building blocks of Pads. All pads are made up of tiles and can potentially be as small as one tile or as big as the whole drum surface. A frequently used configuration splits the surface into 30 pads arranged in a 5x6 matrix, each pad being composed of four tiles. Although smaller pads can be useful at times, they are not recommended because it can be difficult to get reliable hits (it is possible to inadvertently hit the contiguous pad).

2.1.2 Other MIDI controllers

Software is currently being written to allow the use of alternative MIDI controllers. As an example, a normal MIDI keyboard can be used, where keys are mapped to Pads and pitch bend, modulation wheels or pedals are mapped to axes of position control. Percussion controllers and the Lightning are also being considered as options.

2.1.3 Graphics tablets as performance controllers

As in the previous case the software is not finished at the time of this writing. Tablets provide a performance environment that is close to that of the Radio Baton. The three dimensional control is missing and with it goes a very important part of the gestural element that makes the Radio Drum so appealing as a controller. Tablets offer two alternative dimensions of control in addition to x-y position information. The first is pressure, which provides (sort of) a third dimension. The second is sensitivity to tilt of the pen, both in the x and y axes. Additional advantages of the tablet include the fact that it does not use MIDI bandwidth (in the case of the PC only, it uses one of the serial ports in NeXT hardware) and is an easily obtainable controller.

In addition to these options the mouse and keyboard can also be used to control most of the performance functions.

2.1.4 Mixing and matching controllers

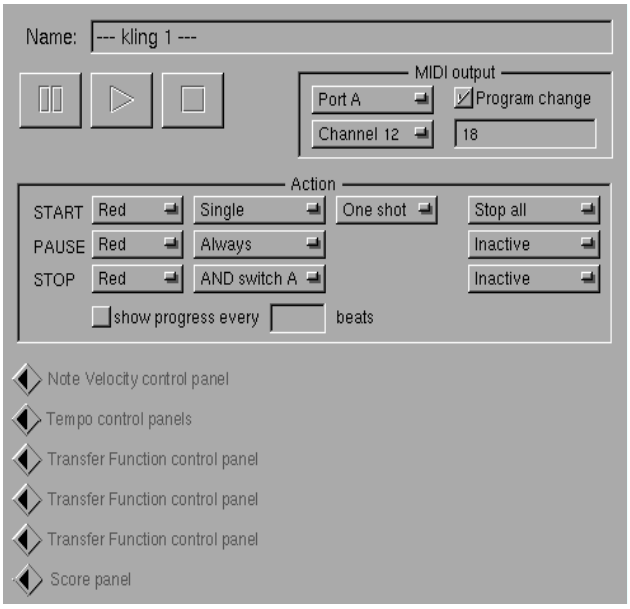
The fact that now several different controllers might be available, even at the same time, raises the issue on how to map things so that a piece can be written in a “controller independent” way. The current approach being worked on depends on establishing a mappings of axes of control to “virtual axes” for each controller, and then using those when creating the piece. Several maps would enable the performer to play the same piece with different controllers by activating the proper map depending on the detected controller configuration. Controller maps are user configurable.

3.0 The browser

PadMaster is a document oriented program. All the programming that the composer does to define the behavior of the virtual surface can be stored to or loaded from binary documents. Documents are edited through a browser where the composer or performer can selected the scene and pad to be edited. Once the selection is done one of the scrollable panes of the browser shows all editing parameters for the selected scene or pad. The program also offers the possibility of saving or loading the document to text files.

4.0 Performance Pads

A Performance Pad is the basic performance element of the program and can be programmed to control the playback of MIDI information or soundfiles. The graphical representation of the Pads on the screen gives visual feedback to the performer on their performing state.



Each type of Performance Pad has a number of fixed control parameters and a number of optional “Elements” that can be added or removed at will.

4.1 Sequence Pads

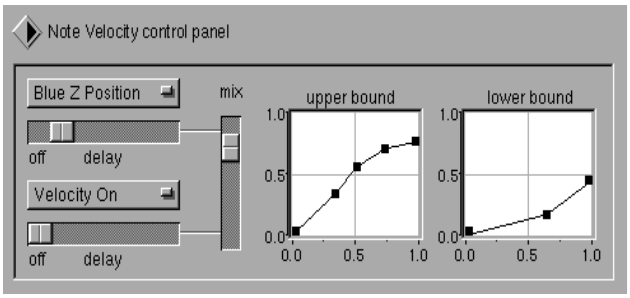
Sequence Pads control the performance of MIDI information. The following figure shows the editing pane of a Sequence Pad with all elements closed. Elements and subelements are shown as lines in an outline and can be opened for editing or closed so they don’t clutter the editing pane of the browser.

The composer can select the MIDI port, channel and an optional program change number. If the program change number is enabled, PadMaster keeps track of which pads are using a particular combination of port, channel and program change number so that Pads across the document can share the same channel but use it with different patches. Any active Pad using a particular combination disables during its performance any

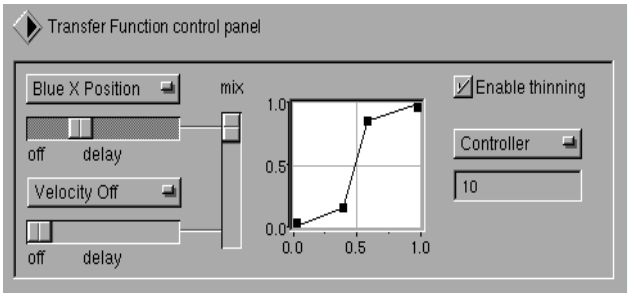
other Pad that uses the same port and channel but a different program change number.

Three types of trigger action can be defined in the current version of the program: start, pause/resume and stop the performance of a Pad. Actions can also have global side effects, for example stopping all performing Pads in the current Scene when the Pad starts performing. The “start” action can be further specified as starting only one sequence, several overlapping sequences with multiple triggers or a note by note performance of the score or algorithm (any of them can perform in single shot or looped modes).

Let’s open one of the elements of this particular Pad (in this case the Note Velocity control pane):



The two transfer functions represent the upper and lower limits over which the existing velocity of notes will be mapped depending on the input parameter. The input itself is a mix of the delayed or smoothed versions of the last trigger velocity and one axis of continuous position information. The position information can come from several sources, depending on the selected controller. In the case of the Radio Drum there are six axes of control plus two additional axes that are computed at trigger time (relative x and y coordinates inside the triggered Pad). Here’s another element, this time an open Transfer Function control pane:



As before, a mix of velocity and position information is fed to the transfer function which is used to generate a continuous MIDI information stream (pitch bend, pressure or continuous controllers).

Score elements contain a MusicKit MIDI score that stores the sequence of notes to be played. Several keywords have been added to the scorefile syntax so that performance of the score can be controlled from within the score itself. For example, a note containing the “*pause: 0*” keyword will immediately pause the performance of the Pad.

Algorithm elements contain a small Objective-C program that generates note objects during the performance. PadMaster supplies a well defined API (Application Programmer Interface) to the composer that provides most of the functionality that is necessary to write small algorithms. An algorithm can have some of its parameters controlled by the virtual axes of control.

These are just some examples of the elements that are available. The introduction in the current software release of elements that are in themselves separate objects opens the door to sharing elements between Pads, something very important in simplifying the task of programming the environment (the composer could, for example, program several interesting transfer functions, name them and then use them to create similar behavior in different Pads).

4.2 Soundfile Playback Pads

This type of Pad controls the performance of soundfiles. The composer can specify the file to be played and the action to be performed by the trigger (start, pause/resume and stop as before). An internal manager keeps track of the number of currently performing sounds and enables or disables Pads from being triggered to keep playback reliable.

5.0 Control Pads

Control Pads are used to trigger actions that globally affect the performance of a Scene. A pad can be programmed to change the current Scene when hit, thus redefining the behavior of the virtual surface of the drum. Control Pads can also be used to pause, resume or stop all playing Pads in the currently Scene.

6.0 PadMaster in performance

Although in many ways the current version of PadMaster has the same look as the previous, the program has been completely rewritten, almost from scratch in many cases. The result is a much more efficient way of doing things internally. A couple of significant examples are: the delay from triggering to activation of a performer has been reduced; switching between scenes no longer causes an audible pause in the performance and so on...

PadMaster has been used to compose and perform two pieces so far: “Espresso Machine”, for PadMaster and Radio Drum, two TG77’s and processed electronic cello (Chris Chafe, playing his celletto) and “With Room To Grow” for solo performer using the Radio Drum as controller.

7.0 Future developments

Most of the architectural changes in the current version have opened doors to new functionality, in particular the creation of several controller drivers. One of the most promising future developments is the use of remote workstations through network protocols to enhance the number of controllable devices.

References:

[1] Max Mathews, *The Stanford Radio Drum, 1990*

[2] Carlos Cerana (composer) / Adrian Rodriguez (programmer), *MiniMax, a piece for Radio Drum*

[3] Fernando Lopez-Lezcano, *PadMaster: an improvisation environment for real-time performance*. Proceedings of the 1995 International Computer Music Conf., Banff, Computer Music Association.

[4] J. Smith, D. Jaffe and L. Boynton. *Music System Architecture on the NeXT Computer.* Proceedings of the 1989 Audio Engineering Society Conference, Los Angeles, CA.

[5] D. Jaffe. *Musical and Extra-Musical Applications of the NeXT Music Kit*. Proceedings of the 1991 International Computer Music Conf., Montreal, Computer Music Association, pgs. 521-524.

[6] D. Jaffe, J. O. Smith, N. Porcaro. *The Music Kit on a PC*. Proceedings of the First Brazillian Symposium of Computation and Music, XIV Congress of the Brazillian Society of Computation, Caxambu, MG, 1994. pgs. 63-69.

[7] D. Jaffe and L. Boynton. 1989. *An Overview of the NeXT Music and Sound Kits.* Computer Music Journal, MIT Press, 14(2):48-55.