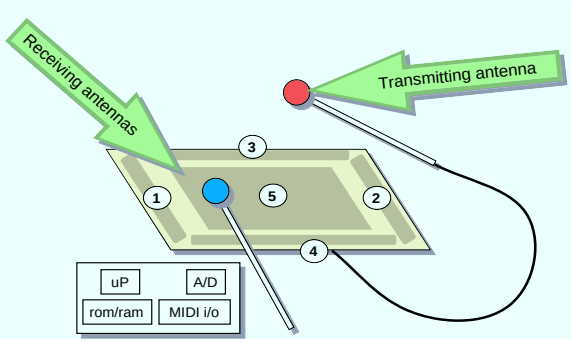
# PadMaster: an improvisation environment for real time performance

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

*ABSTRACT: This paper will describe the design and implementation of PadMaster, a real-time improvisation environment running under the NextStep operating system. The system currently uses the Mathews/Boie Radio Drum as a three dimensional controller for interaction with the performer.*

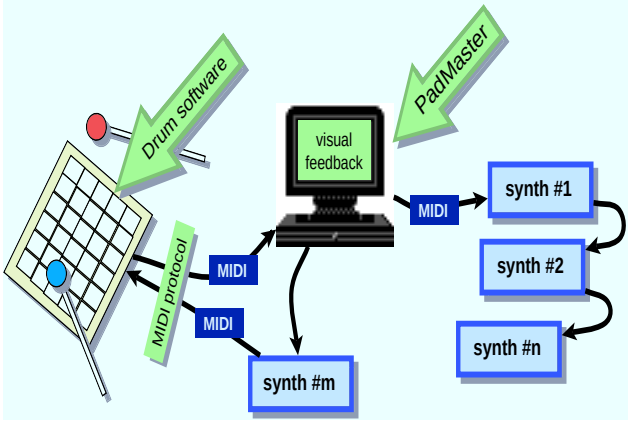
## 1.0 The Radio Drum and the MIDI communication protocol

The current implementation of the Stanford Radio Drum was developed by Max Mathews as a simpler alternative to Boie's design. The two batons act as radio transmitting antennas. There are five receiving antennas underneath the surface of the drum and the multiplexed A/D converter translates signal strength coming from the five receivers to numbers, which the microprocessor uses to calculate the absolute position of each baton in space. In addition to the batons, the Radio Drum hardware includes two switches and four potentiometers. It has a MIDI interface that it can use to communicate with computers or synthesizers.




The existing general purpose controller program (written by David Jaffe / Andrew Schloss) was completely redesigned. A more efficient and faster protocol was created to enable the computer to use the Radio Drum as a three dimensional controller with six degrees of freedom. This program is one of many that are stored in the Drum's firmware, and can be activated remotely with a system exclusive message. This is a short description of part of the protocol:

- **System exclusive configuration messages**: can be used to turn ON or OFF the communication program, set the MIDI channel used by it, dump and load the internal calibration tables, etc.
- **Trigger / Release messages:** sent by the drum when a baton hits / leaves the surface using continuous controllers 26 through 31. The message includes the x-y position and velocity of the hit or release.
- **Switches:** sent by the drum when one of the switches changes state using controllers 5E to 5F.
- **Poll request:** sent by the computer to request the position in space of the batons (channel pressure).
- **Poll answer:** sent by the drum in response to a poll request message using a series of channel pressure messages. It includes the position of both batons in space and optionally the position of the pots.




## 2.0 The PadMaster program

PadMaster is written in Objective C and runs on the NeXT workstation, which is connected through MIDI to the Radio Drum.It uses the Radio Drum as a controller and splits the surface of the drum into up to 30 virtual pads, each one independently programmable to react in a specific way to a hit and to the position information stream of one or more axes of control. Pads can be grouped into Scenes, so that the behavior of the surface of the drum can be subtly or radically altered during the course of a performance by dynamically jumping to a different Scene. The screen of the computer displays the virtual

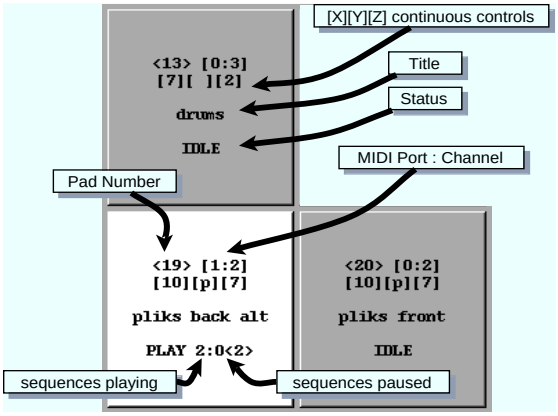
surface and gives visual feedback to the performer on the state of all the pads in the current Scene. The workstation is also connected through MIDI to one or more synthesizers. The virtual pads can be split in two types depending on their function: **Performance** and **Control**.

### 2.1 Performance Pads

Performance Pads can be individually programmed to control the playback of MIDI sequences, note generating algorithms or soundfiles. The graphical representation of the pads on the screen gives instant visual feedback to the performer. Pads change color and status messages dynamically according to their state. A performance pad that is playing remains active even if its corresponding Scene is not currently selected.

### 2.2 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 whole surface of the drum. Control pads can also be used to pause, resume or stop all playing pads in the currently selected Scene.




## 3. Inside a pad

Editable parameters inside each pad can be changed through a standard NextStep inspector window with several editing panes. The first pane can be used to select the action that is executed when the pad is hit. The possible actions include starting / pausing / resuming a sequence, starting a new overlapping sequence or playing the next note of a sequence. It also selects the MIDI port and channel for the pad and allows editing a graphical mapping of hit velocity to note velocity of the played sequence. The second pane edits the tempo options. Tempo can be global, per pad or per sequence inside a pad (when there is more than one instance of a sequence playing). There is a tempo envelope and it is also possible to control tempo with the hit velocity or with any of the six available axes of continuous control. The third pane lets you associate up to three continuous MIDI message streams (pitch bend, pressure or any controller) to the position of up to three of the six axes of control. All these mappings are created through graphical function editors. The fourth pane edits the sequence of notes that are played when the pad is hit. The sequence is expressed as a normal MusicKit format scorefile.

## 4. PadMaster in performance

PadMaster has been used to compose and perform “Espresso Machine”, a piece for PadMaster and Radio Drum, two TG77’s and processed electronic cello (Chris Chafe, playing his celletto). The piece is an environment for improvisation in which the PadMaster and celletto performers exchange ideas and play with predetermined materials. The piece is composed in three PadMaster Scenes, each with several groups of related materials that are triggered during the performance. One baton is reserved for triggering pads and the other for continuous three dimensional control.

## 5. Future developments

PadMaster is currently undergoing a complete rewrite to implement new and improved functionality. This includes resizable pads, triggering of algorithms and soundfiles, multiple computers running remote objects for increased capability (in an ethernet environment), pads with inheritance (useful to be able to group related pads) and a completely rewritten editing paradigm that resembles a database with multiple views.

## References:

[1] Max Mathews, *The Stanford Radio Drum, 1990*
[2] Carlos Cerana (composer) / Adrian Rodriguez (programmer), *MiniMax, a piece for Radio Drum*