# The Slidepipe: A Timeline-Based Controller for Real-Time Sample Manipulation

Mark Argo
Interactive Telecommunications
Program, New York University
721 Broadway, 4th Floor
New York, NY 10003
+1 9174153170
mark@argobot.com

## ABSTRACT
When working with sample-based media, a performer is managing timelines, loop points, sample parameters and effects parameters. The Slidepipe is a performance controller that gives the artist a visually simple way to work with their material. Its design is modular and lightweight, so it can be easily transported and quickly assembled. Also, its large stature magnifies the gestures associated with its play, providing a more convincing performance. In this paper, I will describe what the controller is, how this new controller interface has affected my live performance, and how it can be used in different performance scenarios.

## Keywords
Controller, Sample Manipulation, Live Performance, Open Sound Control, Human Computer Interaction

## 1. INTRODUCTION
The Slidepipe came out of several improvisational music sessions, where the computer keyboard and trackpad were the control sources. After experimenting with the customized performance software written in Max/MSP[1], important gestures were noted and sketched. This information, collected over the several sessions and compared between each, gave insight into what gestures were used most often, and which ones were similar. The goal of these experiments was to reduce the total number of gestures to three or four main movements.

From these sessions I determined that the most important gesture to implement was timeline control. With a trackpad it was difficult to make a selection on a timeline without performing a click-drag. My hands wanted to physically grab the in-out points and drag them around. Another key gesture was the adding/dampening of effects. Using a physical mixer, one can twist a knob left and right to determine the wetness of the effect on a particular channel of sound. This still feels like an abstraction from how the effect wants to be gestured. Again, I listened to my hands and decided that the proper gesture would be something linear. In my mind I imagined an old fashioned shower where a handle on a rope is used to adjust the volume of water.

Inspiration for the design of the device also came from the type of content to be worked with. A motivating idea was that this would be a controller for sampling and sequencing bluegrass and country music. Traditionally, folk musicians tended to fashion their own instruments out of whatever they had around them. This provided a unique design and sound for an instrument that was native to the artist who created it[2]. Imagine sitting on the front-porch of your country house, rover laying silent on the wooden planks, while you rock 'to and fro' in front of your large sampler contraption creating noise-scapes from the pastoral folk music. These images and design parameters motivated the development of the original Slidepipe, which was comprised almost entirely out of materials found on the streets of New York.

This initial version of the controller survived for a couple performances, but over time, being disassembled, moved, and reassembled, it lost functionality and wore quickly. Plus, the materials used in its construction were unnecessarily heavy. The redesign would have to hang on to the good functionality of the original, but make improvements on the 'gig-ability', the durability of the materials and the ease of breakdown-setup. These were the design criteria for the freshest iteration of the Slidepipe.

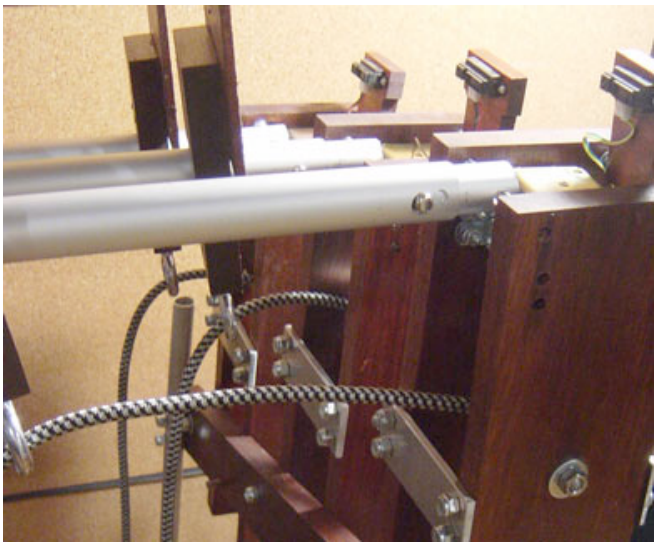

**Figure 1: The Slidepipe (a close-up)**

## 2. SYSTEM
The Slidepipe uses a horizontal pipe as its metaphor for a timeline. Each pipe is comprised of two 'pipeheads', two paddles and two ropes. Each 'pipehead' contains circuitry for reading the parameter state of the ropes and the paddles, as well as a mechanism for managing each ropes position. Each

side of the Slidepipe is connected to a master device that takes all the sensor data and formats it for the software. Three pedals also connect to the master device to provide extra functionality to the performance of the software. The master device then transmits the information to the host computer that is controlling the sound.

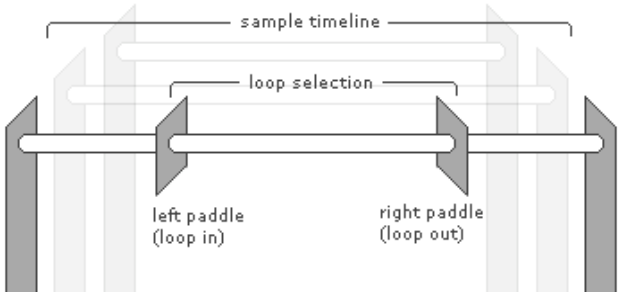


**Figure 2: Pipe as a timeline**

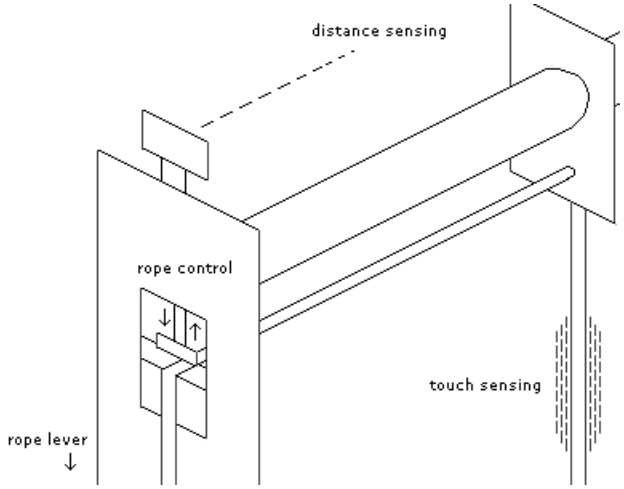


**Figure 3: Components of the System**

## 3. EXPERIENCE

Before playing The Slidepipe, the performer would have to be well acquainted with the software that it will be controlling. The Slidepipe is developed as a general performance controller, so it could control almost any flavor of media (video, sound, robotics, etc). However, what this implies is that there is going to be a specific mapping for each type of content you'd be working with. This mapping could even change between performances. So it is important that the performer knows exactly how the controller is mapped, so that they know what to expect when they move a paddle, or pull a rope. The general metaphor that applies across those different applications (video, sound, robotics, etc) is the physical representation of time and the selection of slices of time to manipulate.

Each pipe is a timeline, so it represents the time length of a sample. The two paddles represent the loop-in and loop-out points of the sample, so by moving the paddles along the horizontal pipe the performer affects what sound from the sample is being heard. If the paddles are close together, then you would hear a small amount of the sample looping rapidly. If the paddles are as far away from each other as possible, then you would hear the entire sample being looped.

Once the performer has the loop that they want to work with, they can use the ropes to manipulate the parameters of the sound. Parameters are assigned in the software, and can be things like volume and pan for sound, or opacity or playback rate for video. By utilizing the pedals, the ropes can have another set of mappings to various effects. If the pedal is pressed then the ropes become mapped to a second bank of parameters for effects such as delay, granular synthesis, arithmetic operations, feedback, etc.

The pedals add dual functionality to the entire system. If the pedal is pressed and held, then it changes the mapping of the ropes and paddles. If it is tapped then it changes the sample source that you're working with. This allows the performer to step through a score or a list of pre-selected clips that they want to work with.

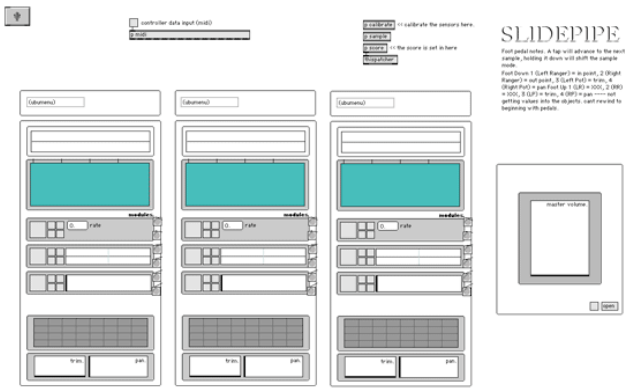


**Figure 4: Software Interface**

## 4. HARDWARE

The old Slidepipe design contained many technical flaws that added unwanted circuit noise and latency to the performance of the instrument. The sensor lines ran six feet to the analog-digital conversion (ADC) circuitry, along with some of the power lines that fed the circuitry on the pipeheads. This distance and poor wiring method added noise to the circuit. Also, these lines weren't filtered or decoupled, so the sensor values had lots of bumps and general noise, all of which needed to be compensated for in software which in turn slowed down the responsiveness of the controller. All of this was changed in the new version.

## 4.1 Slave Device

The new Slidepipe does the ADC near the pipeheads, giving the sensor lines less than two feet to travel. Each slide has its own circuitry to handle the ADC conversion and manage the pipeheads, and acts as a slave device to the master. When the master requests the current state of its sensors, the slave replies.

The components of the pipeheads are:

- a Sharp IR Ranger to determine the distance of the paddle.
- a linear A10k potentiometer as the axel to the lever mechanism controlled by the rope.
- a rope which acts as an electrode for touch sensing
- a solenoid, controlled by the touch sensitive ropes, locking them in place when they aren't being touched
- a locking mechanism for connecting to the pipe

Items used in the pipehead slave device are:

- PICMicro 18F2320 Microcontroller [3]
- QProx QT150 Capacitive Touch Sensor
- TIP120 transistors (for switching the solenoids)
- various status LEDs

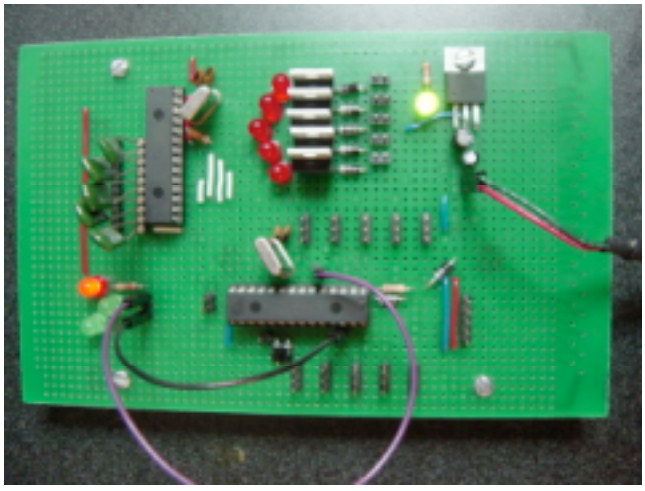

**Figure 5: Pipehead slave device**

## 4.2 Master Device

The boards use the i2c protocol for communicating with the master device, and feature an In-Circuit Serial Programming circuit for programming the microcontrollers.

Items used in the Slidepipe master device are:

- PICMicro 18F452 Microcontroller
- RTL8019 Ethernet Controller
- Maxim RS233 for serial debugging

The master device has a fairly simple operation. It reads the state of the pedals, and requests the sensor data from the individual slave devices storing the results in memory. It then formats the data in OpenSoundControl (OSC)[4] packets, and transmits those packets to the host computer over Ethernet. The master device also takes care of calibration and initialization of the controller. When the pedals are pressed on startup, it will enter a calibration mode allowing the performer to tune the sensors to the environment they are performing in.

The host computer uses Max/MSP to generate the sound, but since OSC has a fairly widespread implementation, it can be used with almost any performance software.

# 5. SOFTWARE

The Slidepipe software was written in Max/MSP for manipulating audio, so the objects being used and the treatment of samples are specific to audio content. Like the hardware, the software side of the Slidepipe has also gone through several revisions. The latest one has been a total reworking of the topology of the program, changing the way samples are selected and sequenced.

## 5.1 Sequencing Samples

The original intent when planning out the software was that the composer would select a group of individual sample files, in .aiff format, keeping them in a subfolder of the main patch. Each would be individually loaded into one of three buffers (one for each of the three pipes) and then manipulated and subsequently unloaded from the buffer to make room for a new sample. This required the MSP buffers to be constantly resized, and introduced a slight latency when the buffer would be filled. Of course, with loading buffers on the fly in MSP there was always the chance of total system halt. This system needed to be reworked.

Taking a cue from STEIM's LiSa software[5], the entire patch was rebuilt to a system based on regions. Individual samples are concatenated in an audio editor producing one master .aiff file. This file is then loaded into a custom written scoring program where the performer can select regions of the master file and create tables of sample start, and sample length points. The effects of this new system are tremendous. The latency involved in loading the buffers is gone, as well as the complex patchwork to rename and resize buffers on the fly, therefore increasing the software performance. In addition, from a compositional standpoint, the region-based system has made it easier to conceptualize a performance. The master file becomes the source of the performance, a unified palette for which to manipulate and spin into soundscapes.

## 5.2 Score Map

Developing a strategy for scoring a performance was considered since the inception of the Slidepipe. The structure had to be open to improvisation, but also rigid in the ordering of sample regions. Each pipe had to be considered separately, as if the performer was composing for three separate instruments that would compliment each other. What resulted

was a three-column table representing the mapping of regions over time. Each block in the column contained the following data: sample in point, sample length, sample rate, initial volume, initial pan, effect A, effect B.

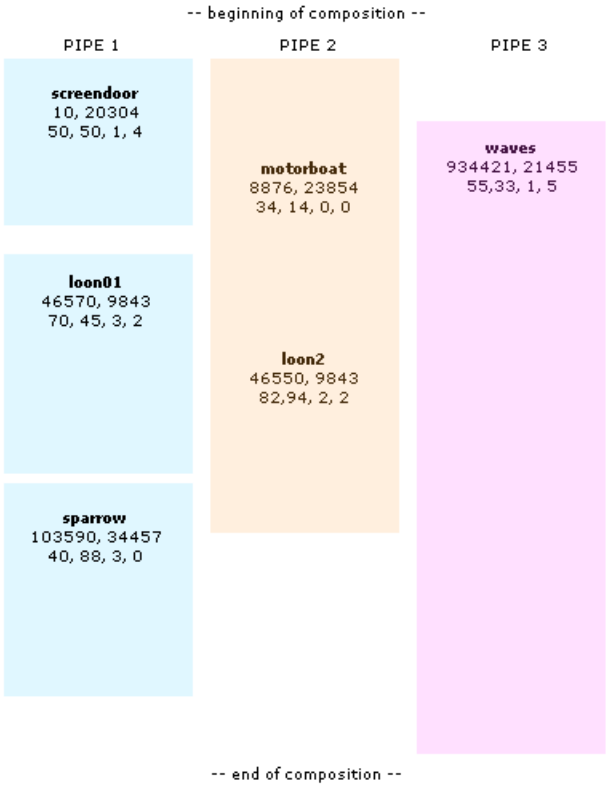


**Figure 6: Example Score Map**

The effect parameters map to another table that is hardcoded into the software, and will load the selected effect into one of the two effects channels (A & B) for each pipe. Unfortunately, this implementation of the score doesn't not support for initial parameters to be loaded into the effects, so when the score block is selected the effects default to the current state of the input to which they are mapped.

What results is a score map with a forward motion. A composition would be written, samples selected and sequenced, and then performed within a preset length of time. The piece is moved though block by block, with each block being loaded into the pipe using the foot pedals. This approach takes practice and skill, which is a necessary of any successfully performed composition and further justifies the Slidepipe as a legitimate instrument controller.

## 5.3 Hardware-Software Communication

With a desire to make the Slidepipe an instrument with future potential, I decided to implement Open Sound Control as the communication protocol between the Slidepipe hardware and the Max/MSP software. This choice was made for several reasons, but mostly for the technical challenge and the ability to expand the range of control to several host computers, perhaps even in geographically discreet locations. I was compelled by Matt Wright's presentation at NIME'03 that OSC was the future language of new gestural controllers. The namespacing language also related well to my particular controller. Since I had built expandability into the hardware circuitry, I envisioned an iterative namespace that could be easily expanded. Reusable software objects could then be written to handle the iterations of the 'pipe' identifier. The expansion of this system is almost without limit, unlike a MIDI implementation. By tweaking the packet format, I've been able to reduce the latency due to the extra byteload necessary for the master processor to communicate using OSC.

# 6. Future Considerations

The future of the Slidepipe will mostly be spent exploring the performance aspects of the controller. This will require several things. Firstly, I'd like to write a patch for controlling other types of media such as video or robotics. This will be to test out my theories on how the Slidepipe can be applied to all types of timeline-based media. Next, I'd like to loan the Slidepipe out to other performers within these various media types. I'd explain how it works and either provide a patch for them to play with, or help interface their patches with this controller. These steps should inform any other future work that will be done on the Slidepipe. I hope to create some recordings in the near future that will be added to the Slidepipe website to give examples of what can be done with this controller.

# 7. ACKNOWLEDGMENTS

Many thanks to my fellow students at NYU's Interactive Telecommunications Program, particularly those who bring the energy for not only building interesting new interfaces but creating new interesting sounds with them. Of course, thanks to Gideon D'Archangelo for his guidance and support during development. Finally, thanks to Tom Igoe, Eric Singer, Ahmi Wolf for technical insight, and Ian and Meagan for helping with construction.

# 8. REFERENCES

[1] Max/MSP. http://www.cycling74.com 2003.

[2] D'Archangelo, Gideon. *Recycling Music, Answering Back: Toward an Oral Tradition of Electronic Music.* Proceedings of NIME 2004.

[3] Microchip PICmicro 18-microprocessor family. http://www.microchip.com

[4] OpenSoundControl. http://cnmat.cnmat.berkeley.edu/OSC/,January 2003.

[5] LiSa. http://www.steim.org/steim/lisa.html 2001.

[6] Bentham, Jeremy. TCP/IP Lean: Web Servers for Embedded Systems. CMP Books, 2002.

[7] Roads, C. Microsound. MITPress, Cambridge, Ma, 2001.