

BounceSlider: Actuated Sliders for Music Performance and Composition

Romain Gabriel*, Johan Sandsjö**

* TableTop Interaction Lab (www.t2i.se), CSE
Chalmers University of Technology, Sweden
morten@fjeld.ch

Ali Shahrokni*, Morten Fjeld*

** Hidden Interaction (www.h-interaction.com)
Göteborg, Sweden
info@h-interaction.com

ABSTRACT

The ForceFeedback Slider (FFS) is a one-dimensional actuated slider using a motor to produce tangible interaction with position and force as input and output parameters. To create a new concept, we have built a mixing desk, placed six FFSs (two implemented here) into a partially realized SliderBox, and added a LED and two toggle buttons to each slider for additional interactivity. We have developed a tool called BounceSlider for improvising music. This application for real time music performance and composition uses a slider handle that can act as a ball. Users can lift and release the handle to set the ball in motion and produce a particular sound each time it bounces against the baseline. Based on physical characteristics, the user can create different sounds and loops by changing two settings: gravity (speed) and bounce type (ball physical characteristics). BounceSlider allows the user to create and save loops of Musical Instrument Digital Interface (Midi) with up to five sounds at a time.

ACM Classification Keywords

H.5.2 User Interfaces, Physical User Interfaces, Tangible User Interfaces (TUI), Haptic Interface

Author Keywords

sound, performance, composition, bimanual, manipulation, tabletop, multimodal, interaction, force feedback

INTRODUCTION

The ForceFeedback Slider (FFS) [1][2] consists of a sliding button (the handle) which can be shifted into 256 readable positions and attain an equal number of force levels. The employed FFS is based on a digital platform and provides an Application Programming Interface (API) developed in Java [3]. We have also added a LED and two additional toggle buttons on its base. This device can provide different

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

TEI 2008, February 18–20, 2008, Bonn, Germany.

Copyright 2008 ACM 978-1-60558-004-3/08/02...\$5.00.

kinds of feedback elicited by position changes, elasticity effects, and texture simulation. The next step in the FFS development is to put together the SliderBox device (Fig. 1, right) using an assemblage of six FFSs, one of which is generally used as the master slider for controlling global features (Fig. 5). With the objective of connecting many sliders, an additional main board was made with the ability to connect the maximum number of 16 devices to one another and a computer via a USB connection (Fig. 2).

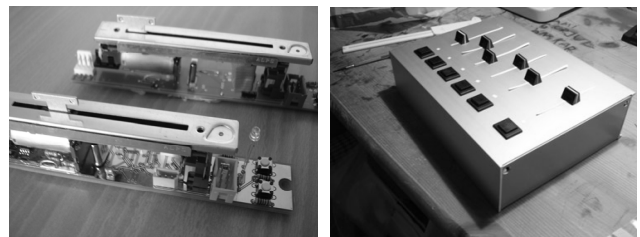


Figure 1: A pair of motorized sliders with LED and two toggle buttons (left) and a mock-up of the multi-slider interactive device with slider handles and buttons, SliderBox (right).

RELATED WORK

The use of simple haptic controllers for the manipulation of sound waves and live music performance was realized in THE PLANK [4]. Multimodal interaction for DJs [5] and the use of haptic metaphors for digital media [6] has inspired the work presented here. BeatCatch [7] uses force feedback to control beat structures. In BeatCatch, however, a continuous rhythm is manipulated.

FeelTheBeat [1] is an application allowing direct manipulation of sound during playback. The slider handle moves according to the sound amplitude envelope. By changing its position, the amplitude of the sound following the playback will be modified, creating a new sample. FeelTheBeat was written in C++ using the analogue slider, which is the old version of the FFS that was directly connected to a sound card.

Collaborative controllers with interpersonal haptic feedback for music performance have been explored in OROBORO [8]. While the work presented here shows a single user case

only, the case of two FFS users has been demonstrated¹ to offer interpersonal haptic feedback [3].

BOUNCE SLIDER CONCEPT

BounceSlider² is a new application built on the digital FFS input-output hardware³. The purpose of the application and research is to provide a tool for exploring perceived physical characteristics of *sound as an object*. We set out to create a playful, experimental device while exploring new possibilities in using haptics for computer music and electronic performance and composition.

The software produces audible sound objects created by inputting object mass (timbre) and gravity (1 = earth gravity) in the program environment. Each object is then assigned to a physical slider to represent the object in the physical world. The simulated size and weight of the object set the force and timbre (from low bass to high pitched treble) of the sound objects. After assigning a sound object to a slider, users can either release or bounce the slider handle downwards. The sound objects become audible as the handle hits the baseline of its cut. Experimenting with the sliders can create sound layers within the loop. The master slider sets the overall tempo of the composition, hence modulating the loop length. The gravity parameter also controls the pace of each object's bounce, as well as its descending height and intensity.

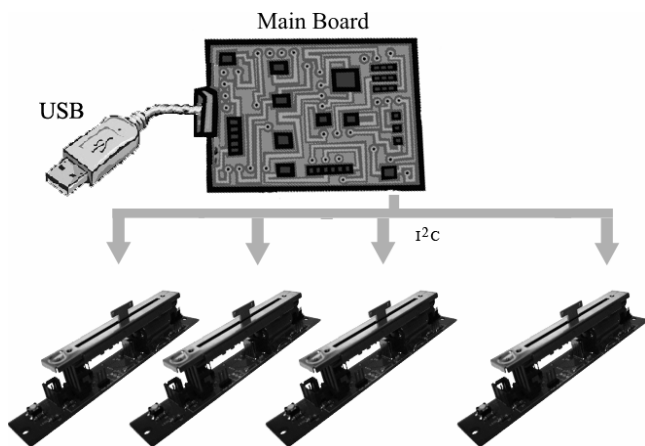


Figure 2: Integrated system with I2C serial bus-connecting main board and slider boards.

First prototype

The first prototype was constructed as a single graphical slider, giving an overview of the concept. When the user lifts and releases the handle, it moves downward and triggers an instrument note as it bounces. The handle is like a ball which falls with a constant speed and the instrument note is the sound of the impact. When the ball touches the ground, it bounces back up to a new position depending on

the input value. We can compare the bounce value to physical stiffness emulating, for instance, a tennis or squash ball. The bounce value is the percentage of loss of height between the previous and current positions. The current position is a percentage of the previous position, for instance, 80% (Fig. 3). Hence, users can hear a note playing faster and faster like the sound of a bouncing ball. Based on Java Sound, this application uses Musical Instrument Digital Interface (Midi) to provide note selection, octaves, 128 instruments, and 58 drums. A few alternative combinations of these were pre-selected and tested. Judging the effects using pitch variations, we finally chose an approach that was a simple play of the melodic interval dependent on the next bounce. The sound terminates just before the next bounce. While this kind of variation is interesting, in the future it may be better to use a sound maker optionally including pitch variations only.

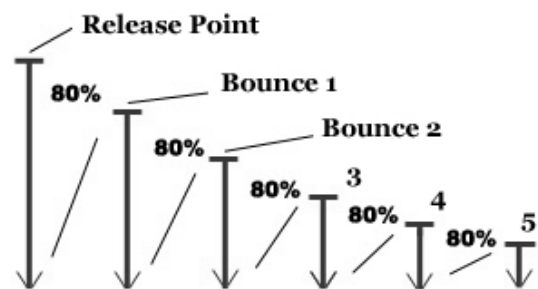


Figure 3: Bounce percentage illustration with a value of 80%.

GUI

This first version of the graphical user interface (GUI) (Fig. 4) is made up of five slider configurations and plays the sounds simultaneously. We utilized a GUI that would separate each slider from its operating elements. The user can select the desired instrument, note, and octave for each slider. The master slider sets the gravity and bounce levels for each slider separately. Gravity affects the bounce speed and Bounce stands for the percentage of loss in height from one bounce to another. The master slider is a global slider which changes features for all the sound slider, and is marked by a thicker line and white color coding. We chose to respect the DJ mixing desk format and positioned the master slider on the right-hand side. The two buttons situated on its bottom are used to switch between the gravity and bounce selections. Some options are also available for each individual slider including basic composition options such as Mute (on/off) and gain selection (percentage). In addition, a Loop (on/off) mode is provided that restarts the release ball when it comes to a rest. Furthermore, one has the choice of putting the master slider options on hold to allow each slider to create diverse rhythms.

For application setups, there is a menu which enables the user to save and load a session, allowing for composition replay during later use. Another setup is a save and load configuration used to select default instruments for when the user wishes to realize a new real-time mix.

¹ Video: http://www.t2i.se/pub/media/FFS_remote_02.wmv

² Video: <http://www.t2i.se/pub/media/BounceSlider.wmv>

³ FFS project web site: <http://www.t2i.se/projects/ff.php>

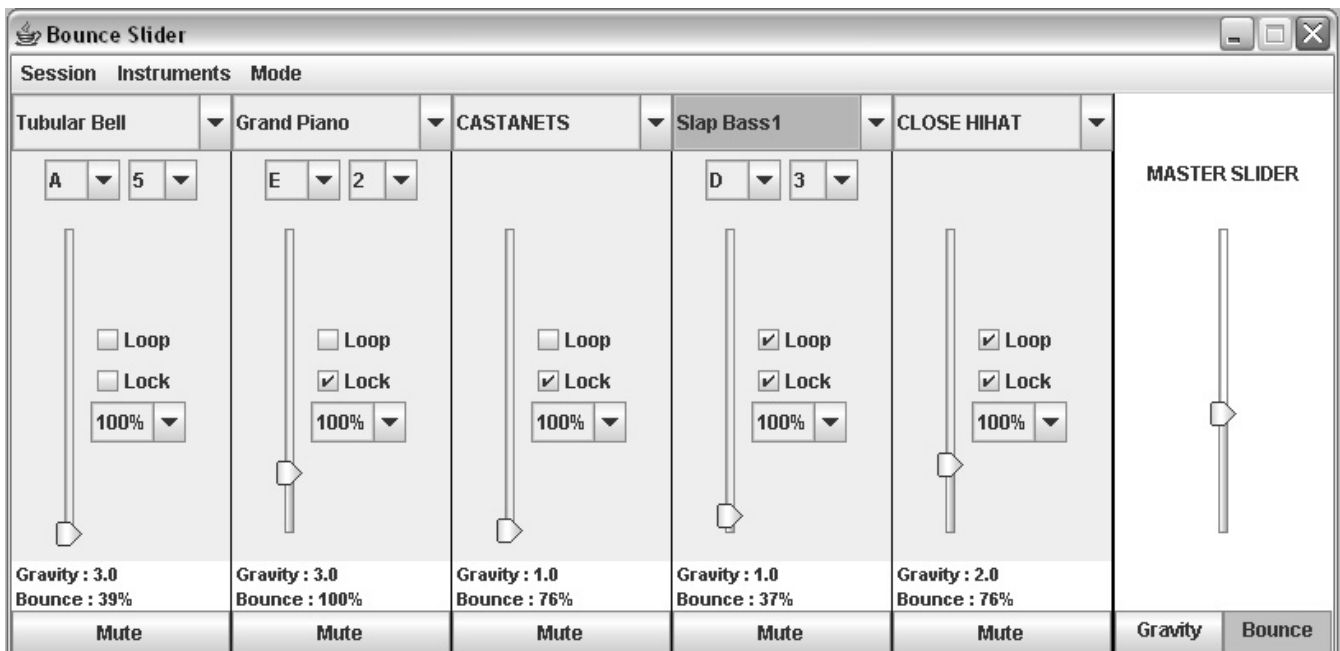


Figure 4: GUI and an overview of features during a use scenario.

Use scenario

Starting the application, users select options in the first column corresponding to the first slider. Users may select an instrument (e.g. Tubular Bell, Grand Piano) in the GUI multibox situated on the top of each column and then select a note (e.g. A, D, E) and an octave (e.g. 1, 2, 3) for the slider, except when using drums. Once this step is complete, users can start the bounce and check the loop box to modify the gravity and bounce while listening to the result. Afterwards, they can check Lock (on/off) tick to fix the master slider values on the first slider. Lastly, users can select the gain (percentage) on the first slider and put the sound on Mute (on/off), allowing them to start creating a sound with the second slider. This use of special features requires rich visual cues, which is why we added two labels at the base of each column to show which gravity and bounce values are actually used by each slider.

FFS IMPLEMENTATION

A particular feature of the BounceSlider version presented here is its ability to be launched with or without an adjoined FFS. This propagates a more flexible interface using a parallel development for FFS and JSlider⁴. Because the SliderBox was not finished during development, we have only executed the application on two sliders. There is one glitch that occurs when users send a lot of data to the FFS, making it fall and bounce like a ball. As a result of the constant data flux, it is difficult for the application to detect

user input when the user’s hand moves a handle. A user can initiate a new bounce by pushing the handle upwards and reduce the loop size by holding the handle.

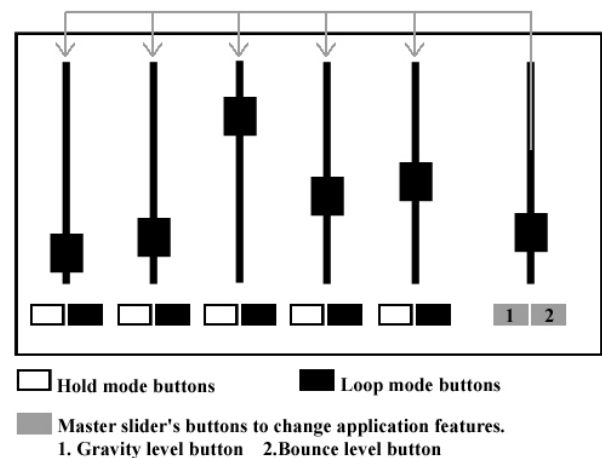


Figure 5: SliderBox interface and button configuration. Buttons are: hold mode buttons (white), loop mode buttons (black), and master slider’s buttons to change between setting (1) gravity and (2) bounce level (grey).

When users release the handle, the sound plays and the slider moves downward until it touches the ground and reascends to a given height depending on the bounce value. After a while, the slider stops and the user can restart it by lifting the handle again and releasing it at some arbitrary position. In loop mode, when the slider drops repeatedly, the user may optionally try to push it upward, an action that the application is programmed to monitor and detect. When

⁴ <https://java.sun.com/j2se/1.4.2/docs/api/javaw/swing/JSlider.html>

the user releases the FFS for the first time, the start position is stored and the slider starts to move downward. The user has to lift the slider up again in order to change the loop and create a new start position. A slider listener detects whether the latest current position is higher than the release position, and only if it is higher will it stop the current loop and linger. During this time, the slider is waiting for the current position to be fixed somewhere. In fact, it is storing the maximum value of the current position until the slider starts to descend again. When the current value is lower than the maximum found, the maximum position becomes the release position and the loop restarts.

When users select a position, they have to wait until the bounce's maximum becomes lower than the position they wish to select. This represents a problem especially if the bounce value is at hundred percent since it means that the loop will be constant. If the start position is the top, the loop length can no longer be modified. To correct this indiscretion, a new behavior was developed. If the handle is held in position for more than one second during the play, as tracked by a timer, the start position changes and the loop restarts there. For the JQuerySlider, the listener is easier to configure. Users just click on the slider, the loop discontinues, and then they select a new start position.

DISCUSSION AND FUTURE WORK

During the concluding stages of the presented prototype's development, some problems concerning sound synchronization were still unresolved. Sometimes a slider lags behind and some sounds are not played when the selected gravity is too strong. This problem originates in the Midi event's sound event feature. The sound events are too often sent and not received. To control this problem, we had to change some features and so now, sounds can be played a bit faster. However, the alterations consume valuable resources. We will change the sound format to waveform audio format (WAV) or MP3 in order to solve this problem. Music sampling could offer enhanced flexibility and a superior range of possible sounds. Consequently, users could choose their own samples.

This version of the BounceSlider works properly and produces haptic interactive music. There are plenty of options that allow users to perform and compose music in new and uncommon ways, enabling personal diversity in output. However, improvements can still be made. One potential modification could be in the form of a tool that enables the user to record a session as a sample WAV or MP3. Another improvement could be a larger palette of playing styles. Pitch modification and new effects or filters could be used to create new sounds thus making the application more versatile.

The most important future work will be to modify and adapt this application for the now partly realized SliderBox, which does not use the same chipset and so, does not work like the previous one. Consequently, the FFS API will have to be changed for the SliderBox and this application rewritten. At any rate, some elements must be improved such as sound play and the slider interaction listener. Moreover, we have observed that one FFS already consumes considerable computational power. To allow an application with multiple ForceFeedback Sliders (FFS) to run with little delay, code optimization will be necessary.

A final point to consider for future work is the choice of functionalities we will apply to the SliderBox (Fig. 5). In the interest of making this tangible interface efficient, we must select among options which would allow the user to make real compositions using only this device. User tests are being planned.

ACKNOWLEDGMENTS

Julio Jenaro Rodriguez designed and built the hardware for the ForceFeedback Slider.

REFERENCES

1. Andersen, T.H., Huber, R., Kretz, A., and Fjeld, M. 2006. Feel the Beat: Direct Manipulation of Sound during Playback. In *Proc. TableTop 2006*, IEEE, 123-124.
2. Shahrokni, A., Jenaro, J., Gustafsson, T., Vinnberg, A., Sandsjö, J., and Fjeld, M. 2006. One-dimensional force feedback slider: going from an analogue to a digital platform. In *Proc. NordiCHI '06*, vol. 189, 453-456.
3. Jenaro, J., Shahrokni, A., Schrittenloher, and M., Fjeld, M. 2007. One-Dimensional Force Feedback Slider: Digital platform. In *Proc. Workshop at the IEEE Virtual Reality 2007 Conference: Mixed Reality User Interfaces: Specification, Authoring, Adaptation (MRUI07)*, 47-51.
4. Verplank, B., Gurevich, M. and Mathews, M. 2002: The Plank: Designing a simple haptic controller. In *Proc. NIME 2002*, 1-4.
5. Beamish, T., Maclean, K., and Fels, S. 2004. Manipulating music: multimodal interaction for DJs. In *Proc. CHI 2004*, 327-334.
6. Snibbe, S. S., MacLean, K. E., Shaw, R., Roderick, J. B., Verplank, W., and Scheeff, M. 2001. Haptic Metaphors for Digital Media. In *Proc. UIST 2001*.
7. Rydberg, L. and Sandsjö, J. 2002. BeatCatch: visual and tactile rhythm box. In *Proceedings of the Second Nordic Conference on Human-Computer interaction (Aarhus, Denmark, October 19 - 23, 2002)*. In *Proc. NordiCHI '02*, vol. 31, 299-302.
8. Carlile, J. and Hartmann, B. 2004. OROBORO: a collaborative controller with interpersonal haptic feedback. In *Proc. NIME 2005*, 250-251.