2F1213 Musical Communication and Music Technology

Design of a HCI for Skipproof

Félix Mandoux
mandoux (at) gmx (dot) fr

Sören Wohlthat
ufb0 (at) rz (dot) uni-karlsruhe (dot) de

June 14, 2004

Abstract

1 Introduction

The turntable has changed into a musical instrument under the last thirty years. A new way of creating music was established: Turntablism with its most important feature: scratching.
For scratching, the turntablist controls mainly two parameters:

- the movement of the record.
- the position of the crossfader for volume and mixing.

Therefore, it seemed to be simple to modelize scratching. However, many attempts were not very successful and the need of knowledge about scratching raised. Kjetil Falkenberg Hansen and Roberto Bresin (see for the moment [2, 1]) studied it and developed *Skipproof* [3], a *Pure Data* [7] patch to use their scratching model.
Skipproof allows to create and control scratches by switching the sound on and off and emulating a moving record including short back and forth movements. Further on, it has several parameters to simulate conditions of different turntables

and vinyls, to select songs or samples, and ready-made scratching patterns. This leads to new ways of creating scratching sounds, for example changing the vinyl condition while scratching. To be able to use all the possibilities, it needs a good human-computer interaction. Several interfaces for *Skipproof* have been designed until now, mouse and screen based ones and more instrument like ones like the Radiobaton and a keyboad MIDI control.
In this project we tried to realize a new interface using *La Kitchen*'s *Toaster* with several sensors.
This article presents first the hardware and some ideas about its usage. Then the software design and the very complex mapping of inputs to outputs is explained. Finally some personal comments and ambitions are given.

2 La Kitchens "Toaster"

The *Toaster* is a 16 channel A/D converter designed for music applications. Its design and distribution is done by *La Kitchen* (see [5]). For compatibility purposes the *Toaster* disposes of a MIDI interface, but its faster TCP/IP network interface is of much more interest. On the backside of the *Toaster* 16 inputs allow the connection of specially designed sensors or home made sensors.

2.1 Available Sensors

All sensors are equipped with a 6.3mm male jack-connector fitting into the *Toaster*. The voltage range of them reaches generally from 0 to 5 Volts. The characteristi-ocs of all commercial sensors distributed by *La Kitchen* can be found at [6]. For this work the following sensors were available:

- Acceleration sensor 1D
- Acceleration sensor 2D
- Flexion sensor
- Pressure/Force sensor
- Light sensor
- 1D magnet field sensor
- Rotary sensor (potentiometer)
- Linear sensor (fader)

- Temperature sensor

Most of the sensors react quite fast, except of the temperature sensor which takes 30 seconds to send the new correct value. Therefore it has not been tested in this project.

2.2 The Toasters sister: *Kroonde*

The *Kroonde* provides the same interface as the *Toaster* but the sensors are wireless sensors. The advantage of the box is that the performer can move in the room, but its latency time due to the wireless protocol 802.11b is higher. By simply changing the listening UDP port of the *pd*-script the *Kroonde* can be used instead of the *Toaster*.

3 Ideas for the Design of a HCI

Designing a human control interface for a music instrument is difficult. On the one hand it has to be easy to use, but on the other hand it would be booring after a short time if it is too easy. The performer should be able to create music with the HCI in a short time, but progressing by training and learning should be also possible.

3.1 Hardware Design

The sensors of *La Kitchen*s "Toaster" (see [5]) are equiped with a long cable and can therfore be installed anywhere on the body. The idea to install the sensors on the whole body is seducing, because each movement of the performer could modify the behaviour of the "instrument" and its sounds. Finding a suitable setup for the sensors is difficult and very time consuming. In our case we sticked the sensors with adhesive tape on different parts of the hand and on a table (Figure 1).

3.2 Software Design

The hardware design can be changed by the performer: positioning and the sensors working range is different for each performer. *Skipproof* was realized by Kjetil Falkenberg Hansen (see [3]) in the *Pure Data (PD)* environment (see [7]), a graphical programming interface for musicians, similar to *MAX*. Our software is also PD based and connects to the *Toaster* via network using the TCP/IP protocol. The structure of the HCI software for the *Toaster* can be represented several blocs (Figure 2).

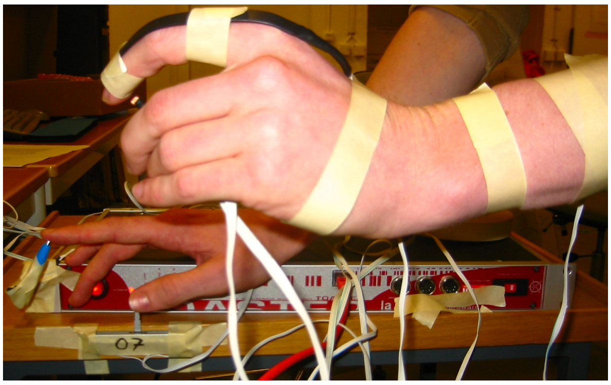


Figure 1: An example for positionning of sensors on arm, hand and table.

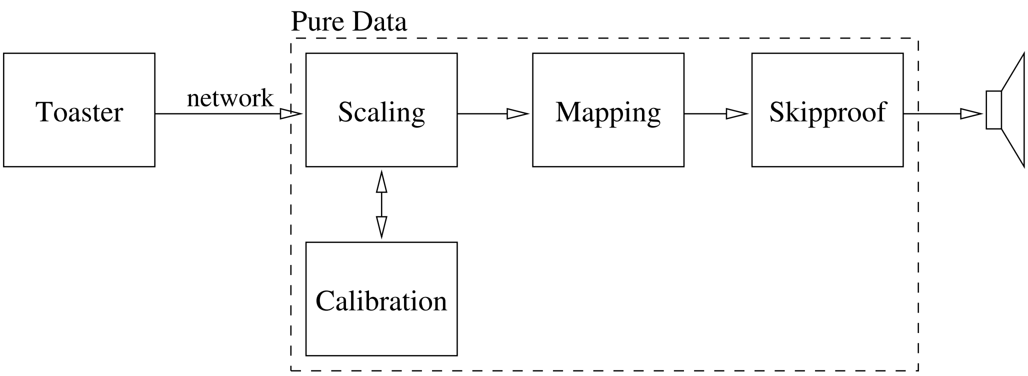


Figure 2: The software connecting the *Toaster* to *Skipproof* is composed by several blocks.

The “scaling” bloc receives the data of the *Toaster* via network and applies some scaling factores issued from the calibration bloc. To profit from the full working range of some sensors we integrated a calibration tool in the software. The calibration can be done for each sensor and at every moment. A button for each sensor resets the calibration bloc and starts the calibration. In this way the output values from the *Toaster* can be scaled on the full internal 16 bit range of the interface. The technique of the highly complex mapping bloc is described in the next section. From the mapping bloc the parameter values are sent to *Skipproof*. The graphical interface of *Skipproof* using the *Gripd* library (see [8]) is updated in realtime to get a visual feedback of the parameters controlled by the *Toaster*. Calibration and mapping can be done by the performer via the *pd*-script shown in Figure 3.

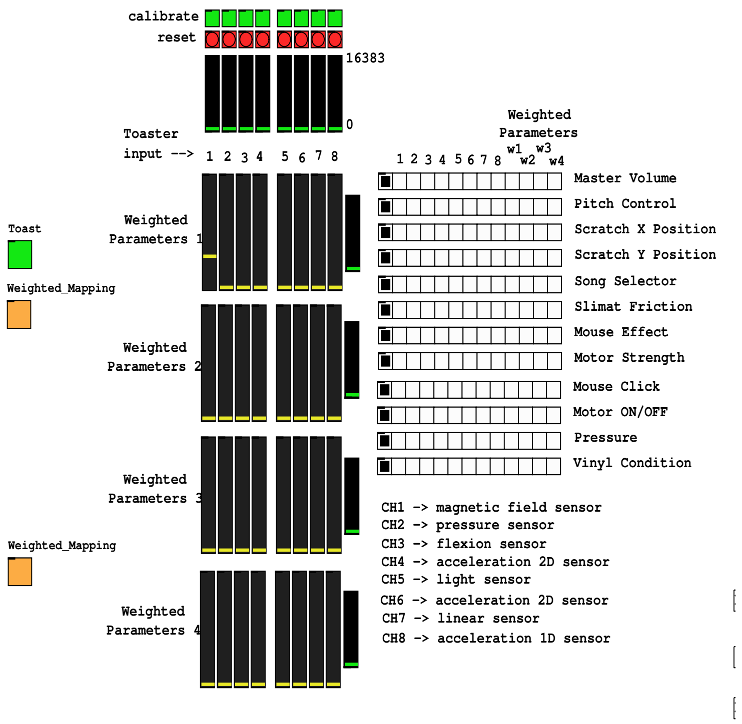


Figure 3: The buttons on the top left switch the script to calibration mode and reset the scaling factors. The 8 vertical sliders under them show the output values of the toaster. 1-to-1 mapping can be done with the radio buttons on the right side, where the buttons 1 to 8 correspond to the first 8 channels of the *Toaster* and the buttons 9 to 12 correspond to the 4 weighted parameters. The weighting can be fixed with the 32 vertical sliders on the right side. Each channel can contribute to one of the 4 parameters in its own way.

4 Mapping

The mapping is one of the most important and most difficult features in musical human-computer interaction. It can be done in several ways. Andy Hunt and Ross Kirk describe two different ways in [4]: the analytical and the holistic way. The analytical one consists in studying sequentially a parameter at once, in logical order. The holistic way consists in looking at all parameters as a unit, or better to look at the whole object or subject. Individual details are less important than the overall effect.

Because of this we decided to create a highly variable mapping tool which allows the analytical and the holistic mode and combinations of them.

We thought of using a matrix to map the sensors on the parameters. However, we could not find an adapted matrix object in *PD* and had not enough time to write one on our own. Therefore, we built a script[1], which maps 16 controllers to one parameter whereas it is possible to weight the controllers. To these parameters we revere as weighted parameters. This script implements the main part of the holistic mode which is a n-to-one mapping, including a weighting between different sensors affecting one parameter. Further on, a second script[2] allows selecting for each *Skipproof* parameter a controller. This controller can be the unmodified one sensor or the weighted combination of several sensors. Therefore it is possible to use a sensor for multiple parameters and one parameter can be influenced by multiple sensors, this is the so-called n-to-n mapping.

5 Tests and Appreciation

In our first test setup we connected five sensors to the ready-made scatch templates of Skipproof. These templates are classical well known scratching techniques and can be played with various tempi and extended or limited disk movement. We used the flexion sensor to vary the speed of the disk movement and the pressure sensor to change the movement extension. The light sensor starts and stops the scratch, the slider selects different techniques and the potentiometer the sound sample. We fixed the Potentiometer and the slider on the table to be usable with the right hand. The other sensors were fixed on the left hand. The flexion sensor was taped to the forfinger, while the pressure was taped on the thumb. In this way it was possible to press with the thumb against the middle finger. Between both was also the light sensor. The light sensor could be controlled through the forfinger by tipping on its top (see Figure 1).
This configuration gave us really fast several nice sounding results. However, it was almost impossible to produce the same sound twice. It was not possible to time the start and control the speed and disk movement exactly. Especially the pressure sensor was hard to control. Further on, if a scratch was played with low speed the volume decreased or seemed to. Therefore, we mapped a weighted parameter combining the magnetic field sensor and the flexion sensor to control the volume. We placed the magnetic field sensor on the left arm. Maximal vulume could be obtained with a horizontal arm position. Because of the properties of the magnetic sensor the signal changes by turning the arm. This weighted parameter compensated the volume problem quite well. Finally, a nice secondary effect appeared: the volume could be controlled also with ease. It was really nice to play around with different movements of the arm and soon the whole upper body was

[1] Mapping16weightedsignalsfromthetoastertooneoutlet.pd
[2] interface_toaster_skipproof.pd

moving.

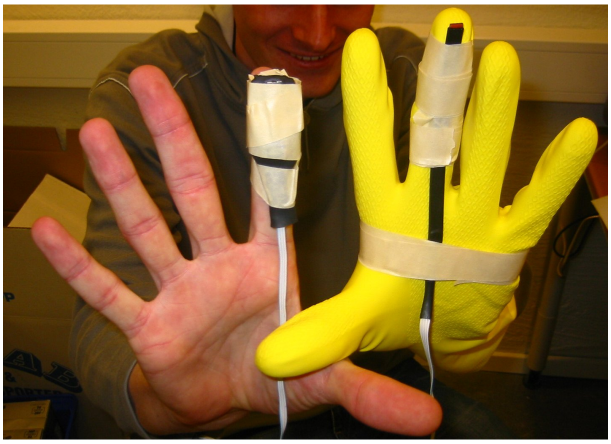
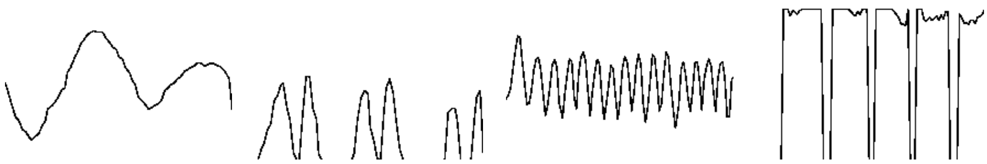


Figure 4: The second configuration showing the flexion sensor on the left hand and the magnetic field sensor on the right one.

Our next setup reduced automatic scratching to the minimum. As scratching is mainly controlled by two parameters we decided to use one sensor to define the position of the record and one to simulate a cross fader by connecting it to the master volume. Our first idea was to use the flexion sensor for the record position and the magnetic field sensor for the volume. But some attempts showed that it was easier to control the magnetic sensor over a wide range than the flexion sensor and that a very fast controller was needed to switch the volume on and off. Therefore, we mapped the magnetic field sensor to the record position and used the pressure sensor for the volume (see Figure 4). We calibrated the pressure sensor only for a very small range, so that low pressure was enough to switch off the sound completely. This setting was quite successfull to do various scratching techniques (see Figure 5).

The last configuration was so easy to use that simple scratch techniques could be learned fast. The fun factor increased rapidly by trying new movements and by repeating or combining them. This might be a really good configuration to learn the usual techniques. Finally it seems also to be a good basis to extend with new sensors to allow the simultanous control of more parameters.



(a) Crab - record position.

(b) Crab - crossfader position.

(c) Rolltear - record position.

(d) Rolltear - crossfader position.

(e) Flare Twiddle - record position.

(f) Flare Twiddle - crossfader position.

(g) Uzi - record position.

(h) Uzi - crossfader position.

Figure 5: Some examples of the resulting record and crossfader position using the *Toaster* as interface for *Skipproof*.

6 Conclusion and Future Work

The connection of the *Toaster* to *Skipproof* was done with success. The *Toaster*s very simple interface permits to connect it with ease to *Pure Data*. More problematic were finally the calibration and the mapping of the *Toaster*s outputs to *Skipproof*s inputs. After many different software realisations a very polyvalent calibration and mapping tool could be designed. Now many different mapping situations can be tested with ease by simply modifying some parameters. The last developed controller is a glove and can be adapted to each performer by recalibrating the sensors. Finally with this setup it is possible to play simple scratches without training, but an intensive training session permits also to play complicated and interesting ones.

6.1 Future Work

To find a suitable positioning of the sensors and mapping to the *Skipproof* parameters the time devoted to this project was too short. A more extensive search of possible configurations could be an interesting work. The interface of the sensors for the *Toaster* are so simple that the design of new ones could complete the collection. To avoid reconfiguring the interface all the time, a memory function for the mapping and calibration as *Pure Data* script should be helpful. Some of the sensors provided with the *Toaster* have a linear behaviour, other ones non-linear behaviour. In future, the calibration or mapping tool could be extended to apply simple mathematical functions like exponentials or inversions to the sensor output.

References

[1] Kjetil Falkenberg Hansen, *Playing the turnable: An introduction to scratching*, Speech, Music and Hearing, KTH, Stockholm, Sweden, TMH-QPSR, Vol 42: 69-79, 2001.

[2] Kjetil Falkenberg Hansen and Roberto Bresin, *Analysis of a genuine scratch performance*, Speech, Music and Hearing, KTH, Stockholm, Sweden.

[3] Homepage of Kjetil Falkenberg Hansen, *Source and documentation for Skipproof*, `http://www.speech.kth.se/~hansen/scratchstart.html`.

[4] Andy Hunt and Ross Kirk, *Mapping Strategies for Musical Performance*, in M. Wanderley and M. Battier, eds. Trends in Gestural Control of Music; Ircam - Centre Pompidou, 2000.

[5] La Kitchen, *The Toaster*, `http://www.la-kitchen.fr/KITCHENLAB/toaster-en.html`.

[6] List of the sensors of *The Toaster*, `http://www.la-kitchen.fr/KITCHENLAB/sensors.pdf`.

[7] Pure Data, *Programing language and interface for musicians*, `http://www-crca.ucsd.edu/~msp/software.html`.

[8] Gripd, *Graphical user interfaces for Pure Data*, `http://crca.ucsd.edu/~jsarlo/gripd/`.