6to6Mappr

An educational tool for fast and easy mapping of input devices to musical parameters

Gelineck, Steven; Böttcher, Niels

Published in:
Proceedings of the 7th Audio Mostly Conference

DOI (link to publication from Publisher):
10.1145/2371456.2371475

Publication date:
2012

Document Version
Accepted author manuscript, peer reviewed version

6to6Mappr

An educational tool for fast and easy mapping of input devices to musical parameters

Steven Gelineck
Aalborg University Copenhagen
Department of Architecture, Design and Media Technology
Copenhagen, Denmark
stg@create.aau.dk

Niels Böttcher
Aalborg University Copenhagen
Department of Architecture, Design and Media Technology
Copenhagen, Denmark
nib@create.aau.dk

ABSTRACT

This paper presents a tool for mapping several commonly available novel controllers to various musical parameters in order to provide technological novices with a way to exploit new forms of musical interaction provided by technology. The tool automatically connects to a control interface specified by the user and maps input parameters of that interface to musical output. Besides being able to select a specific input device, the user can choose between the input parameters, the output device (for instance a physical modeling synthesizer), parameters of that output device and the mapping between the two. The paper presents the motivation, design and implementation of the tool and presents initial experience with using the tool in an educational setting where non-technical conservatory students were introduced to music technology using the tool.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation (I.7)**]: Sound and Music Computing—*systems*; K.3 [**Computer Uses in Education**]: Computer-assisted instruction (CAI); K.3 [**Personal Computing**]: Application Packages

General Terms

Design, Human Factors

Keywords

Music Technology, Education, Mapping, New interfaces for musical expression, Musical gestures, Novice users, Musical mapping

1. INTRODUCTION

In order to understand how technology can enrich musical expression, one must get a sense of how musical output can change based on new forms of interaction made available by new technology. The task of opening this understanding for complete novices has motivated the work presented here. The idea has been to create a software platform that lets complete technological novices create alternative musical interfaces by mapping alternative controllers to musical output. The software presented here builds upon several open source platforms for detecting gestures and routing gesture-data to other software platforms using for instance MIDI or Open Sound Control[1] (OSC). In order to sufficiently use these platforms one needs to be somewhat familiar with interpretation of MIDI and/or OSC data. The idea here has been to collect approaches in one easy-to-use environment that lets a novice user create sound right away without having to concern themselves with the management of data.

The rest of the paper is organized as follows. Section 2 describes some related work within the field accounting for the need for an easier open source approach. Section 3 describes the design considerations on which the development of the software is based. Section 4 accounts for implementation of the software, and finally, section 5 presents some initial impressions of the system based on a large workshop (100+ students) that was conducted at the Rhythmic Music Conservatory Copenhagen.

2. RELATED WORKS

Other platforms do exist that let users map input parameters of novel controllers to musical parameters of their choice. However, most of the tools expect users to familiarize themselves with communication protocols (MIDI/OSC) and/or development tools such as Max/MSP, Pd, Chuck or SuperCollider. Furthermore, most of the the available software platforms that ease this task are not open source.

OSCulator[2] is an example of a software tool for routing OSC data from input devices to software platforms. It lets the user choose between different input devices and output the incoming data from the chosen device using a variety of different message types (for instance MIDI CC or OSC routing). The power of OSCulator lies in its flexibility. However, while it is fairly easy to use for people with technical experience, for non-technical users it takes some effort to get a system up and running. Also, the OSCulator is not open source. Other open source tools exist that route data from

[1] http://opensoundcontrol.org/
[2] http://www.osculator.net/

AM '12, September 26 - 28 2012, Corfu, Greece

a number of controllers. GlovePie[3] works for instance with Wii-mote, virtual reality gloves, joysticks and trackers, and the OpenNI framework[4] works for skeleton tracking using the Microsoft Kinect (along with many other "natural interfaces"). However, they are targeted more towards developers and not at all suited for technological novices. Other powerful applications such as STEIM's Junxion[5], ControllerMate[6] or the Musical Gestures Toolbox [7] also provide many possibilities of using novel technological control environments and mapping those for musical purposes, but those are also made with developers or at least tech-savvy users in mind.

Amoung the more advanced approaches for making exploration of mapping paradigms available for non-experts is the Wekinator [2] that lets users create machine learning models for mapping between input and output data. [5] presents a very detailed overview of tools for exploring the mapping from gesture to sound and introduces the SARC EyesWeb Catalog, which is an Eyesweb[7] library that implements a large selection of gesture recognition algorithms. Lastly, [3] presents approaches to using laptop sensors in computer music performances also presenting a toolkit for managing inbuilt laptop sensors. Again the tools presented above target expert technical users.

3. DESIGN

A set of criteria was formed in order to guide the design of the mapping tool. The criteria were based on the effort towards developing a tool that created the balance between constraints and flexibility necessary for non technical users to have the tool working instantly while still providing enough control for deeper exploration of parameter mapping, adjusting musical inputs and outputs to fit specific needs. The design criteria were:

- Open source.
- Easy to install.
- Be able to output sound within 3 mouse clicks after installation.
- Demand a minimum of guidance.
- Implement a variety of commonly used control interfaces (webcam, Wii-mote, Kinect, etc.).
- Implement a variety of musical outputs (synths, effects, MIDI out, etc.).
- Be interesting for technological novices as well as tech-savvy users.
- Enable hands-on exploration and experimentation with mapping strategies.

Additionally, the platform was developed with a certain context in mind that would help guide the design process. The context was an educational workshop that enabled many students (100+) from many different musical backgrounds to experiment with new musical control paradigms. The idea was to expose the students to a multitude of different possibilities of input and output within a constrained and somewhat uniform environment.

[3]http://glovepie.org/glovepie.php
[4]http://openni.org/
[5]http://www.steim.org/junxion
[6]http://www.orderedbytes.com/controllermate
[7]http://www.eyesweb.org/

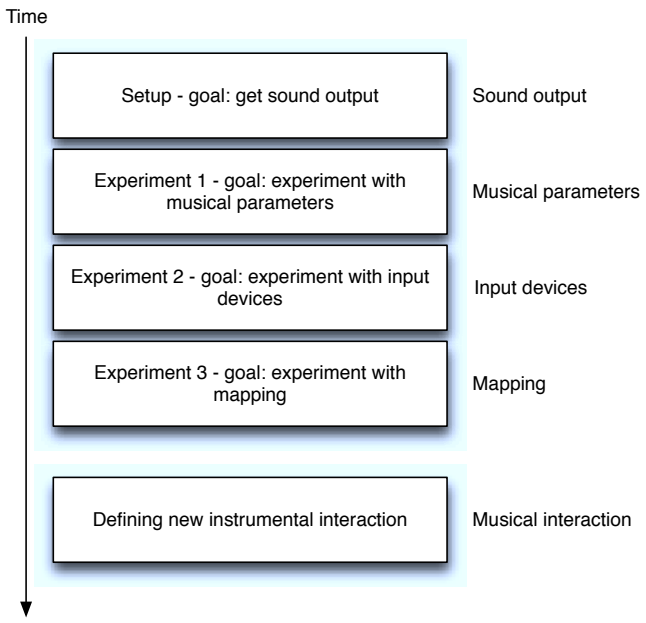

Figure 1: Depicts the intended learning process of the 6to6Mappr.

As mentioned earlier, many platforms let users somehow setup control structures for controlling musical parameters. However, most demand that several choices are made before one is able to get the system up and running, and most are targeted towards developers or at least technologically experienced users. While these approaches do provide flexibility they increase the amount of effort needed from initially encountering the program to actually achieving musical output. A central design approach has been to provide "instant music, subtlety later" as Perry Cook puts it [1].

The solution to this was to limit the user to a maximum of 6 input/output parameters. This way the mapping space can be fixed from the beginning and default settings can help the user to be up and running within seconds. Besides, continuous gestural control of 6 musical parameters simultaneously provides quite a high amount of expressive potential - especially when provided additional control of the mapping layer. While each input device (wii-mote, Kinect, iPhone, etc..) had several input parameters to choose from, only 6 of them were to be mapped to musical parameters. Similarly, musical outputs had several adjustable parameters, but only 6 of them could be assigned at the same time to input parameters.

The idea was to let the user choose an input device and output destination from a list and then worry about adjusting mapping parameters once up and running. As mentioned earlier, too many tools demand that the user makes many choices before the system is ready to go. In this approach the goal was to provide instant control, and then subtlety could be adjusted later.

After experimenting with different relations between controller and musical output, one would have the opportunity to dive into the intermediate mapping layer in order to really

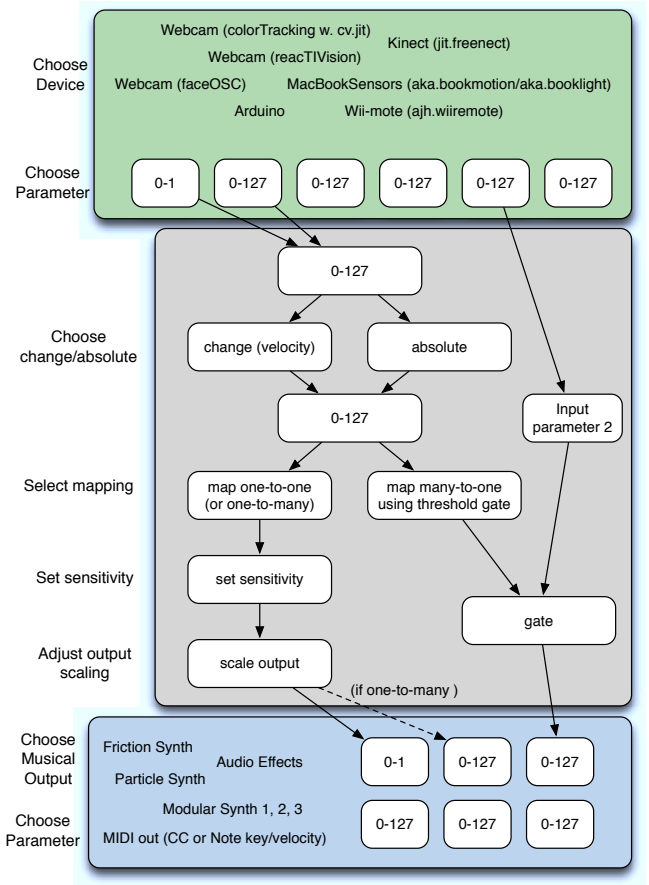

Figure 2: Overview of how the mapping between input device parameters and musical output parameters is taken care of.

define the relationship between input and output, thereby defining the instrument being played. See Figure 1 for an overview of the intended learning process.

The most important mapping settings available to the user was defining minimum and maximum values that each input device parameter would generate, along with the sensitivity (filtering/averaging of the input data) and randomness of the output. Instead of directly mapping the absolute value of an input parameter to an output parameter the user also had the possibility of using the velocity of value change as an input parameter—this for instance made it possible to use velocity of a gesture for triggering a musical parameter (for instance input energy to a physical model), thereby achieving a more instrument like feel (as proposed by Hunt et al. [6]).

In order to support a one-to-many mapping strategy, mapping the same input parameter to multiple outputs was important. In order to enable many-to-one mapping a triggering system was designed to let the user trigger the output from one input parameter using a different input parameter. For instance one could send accelerometer data from the Wii-mote only if the A-button was pressed/de-pressed. This was done by letting the user define a certain value that when crossed would instantly gate a different input parameter of choice.

Every input parameter is normalized to MIDI, meaning that all sensor input has a min/max value of 0-127. Correspondingly every musical parameter also has a min/max value of 0-127. Some input parameters are only binary (as for instance the A button on the wii-mote). This produces an output of either 0 or 127. Some musical parameters are also binary (as for instance *delay on/off*). These are triggered as an input value crosses a center threshold of 64. If the user for instance wants to control the *delay on/off* by rolling the wii-mote from left to right, the delay will turn on at the moment where the role output crosses from 63 to 64, which occurs as the device is level.

Figure 2 provides an overview of the mapping possibilities provided by the system. The top layer involves first a choice of input device after which specific parameters of that device can be selected. The middle layer provides several choices for how each parameter can be processed, and finally, the bottom layer involves selecting the musical output and the specific parameters the user wishes to control in realtime.

4. IMPLEMENTATION

The 6to6Mappr was built as a standalone application for Mac only using Max/MSP and was heavily based on open source platforms and Max/MSP externals developed by others (see list of compatible input devices in Section 4.1). As described earlier many implementations exist that handle communication between Max/MSP and external devices. However, most implementations demand experience with Max/MSP before being able to use the devices for musical purposes. The idea has been to integrate the work by fellow researchers into one single tool that is setup to route data by carrying out few simple steps. Not only is the tool easier to learn, it also offers a lot of different possibilities for exploring quite different input controllers.

As mentioned in the design section the software follows a scaffolding principle of presenting a clear constrained structure that works right away letting the user dig deeper adjusting more parameters the more experienced they get.

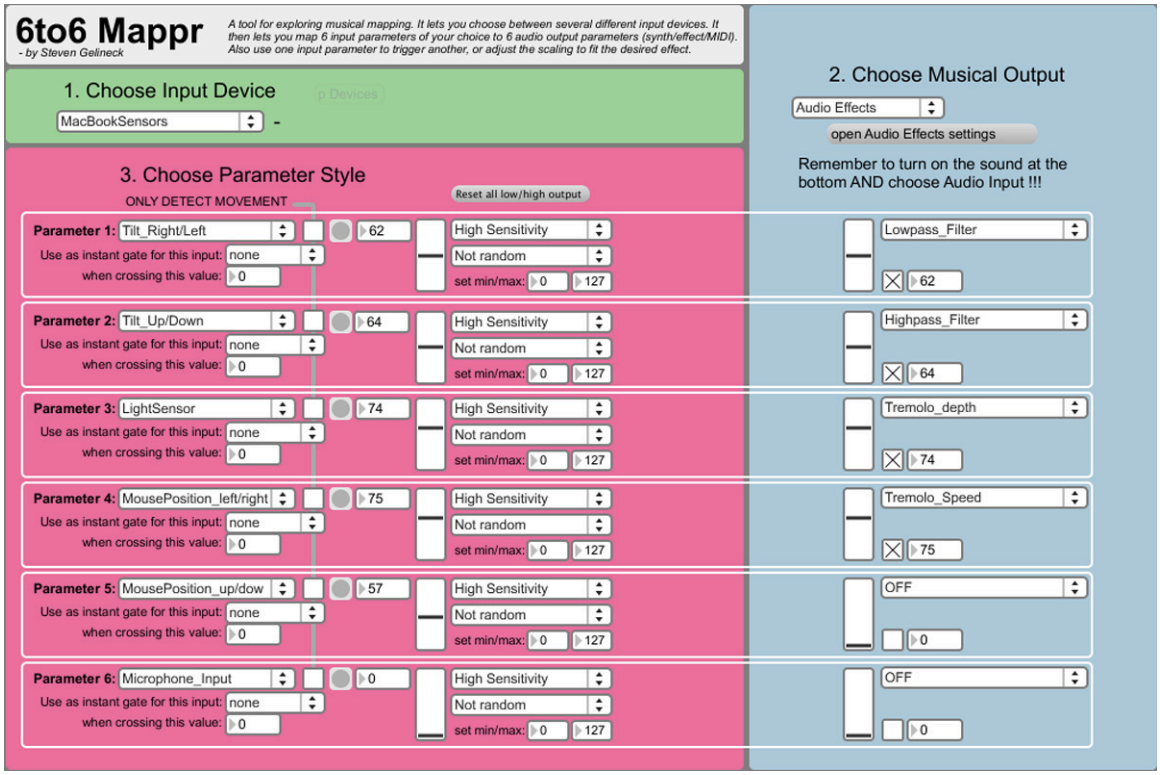

Figure 3: The central part of the graphical user interface for the 6to6Mappr is divided into three sections - Input Device, Musical Output and Parameter Style.

4.1 Input Devices

Initially a set of input controllers were chosen based on their availability and the interestingness of gesture input possibilities. The input devices were:

1. Webcam color/movement tracking (position tracking of two different colors and horizontal/vertical movement tracking) - developed by the authors using the cv.jit library[8] by Jean-Marc Pelletier.

2. Webcam fiducial tracking (position, rotation and on/off of up to 6 printable fiducials) - based on the reacTIVision toolkit [8].

3. Webcam face tracking (horizontal/vertical movement, face size, eyebrow height, mouth height/width) - based on faceOSC by Kyle McDonald[9].

4. MacBook sensors (3-axis tilt, light sensor, mouse position, microphone and keyboard) - based on the aka. bookmotion and aka.booklight externals by Masayuki Akamatsu[10].

5. Microsoft Kinect (centroid tracking (X/Y) in 4 different regions) - based on the jit.freenect.grab[11] external by Jean-Marc Pelletier.

6. iPhone, iPad, iPod Touch (X,Y position of three pads) - based on the Mrmr OSC Controller[12].

7. Wii-mote (right/left role, vertical tilt, nunchuck joystick X,Y, A and B buttons on/off) - based on the ajh.wiiremote external[13] by Alexander Harker.

8. Arduino (implementing force sensors, distance sensors, dials, buttons - this particular implementation thus takes 6 analogue inputs and 2 digital inputs and is specifically implemented for the workshop described later).

As seen in the list of compatible input devices the 6to6Mappr uses open source platforms and externals in a semi-pre-determined way for the sake of ease of use. Most of the input devices work using externals by others (Wii-mote, Kinect, face tracking, reacTIVision, iPhone, MacBook Sensors), but the data is processed internally in the software in order to produce parameters that make instant mapping to musical output possible.

As the user chooses an input device the 6 first input parameters are automatically setup to be mapped. Additional input device controls are highlighted to give the user the ability to adjust settings for the selected input device (for instance for choosing colors for color tracking, to reconnect the wii-mote, or to set the depth of view for the Kinect).

4.2 Musical Output

[8]http://jmpelletier.com/cvjit/
[9]https://github.com/downloads/kylemcdonald/ofxFaceTracker/FaceOSC.zip
[10]http://www.iamas.ac.jp/ aka/max/
[11]http://jmpelletier.com/freenect/
[12]http://mrmr.noisepages.com/
[13]http://www.alexanderjharker.co.uk/Software.html

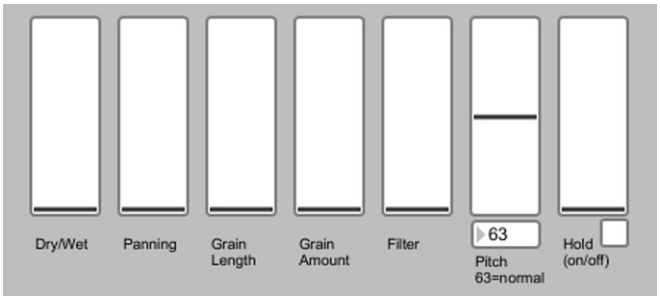

Figure 4: All parameters of a musical output device are adjustable using a simple graphical user interface.

Most of the musical output of the 6to6Mappr is created by the authors by implementing commonly known audio processing techniques using Max/MSP (for instance tremolo, delay, filtering, simple FM/AM synth). Two physical models borrowed from previous projects [4] and a granular effect based on the Granular Toolkit by Nathan Wolek [9] is also implemented. Finally, the user is able to route MIDI messages in order to control external musical software. The following is a list of the available musical output.

1. Basic audio effects (filters, tremolo, delay, flanger, reverb, volume) applied to input audio or a loaded audio sample.

2. Particle model (amount of particles, two resonance frequencies, randomness, tone).

3. Friction model (velocity, downward force, two resonance frequencies, filter, delay).

4. Granular effects (amount of grains, grain length, filter, pitch, panning, hold on/off, dry/wet) - based on the Granular Toolkit by Wolek.

5. Modular synths (frequency, modulation rate, depth, loop-size, filter, delay).

6. MIDI out (note, velocity, control change).

When selecting a musical output only the first 4 parameters of that output are automatically setup and the mapping is established - see Figure 3. The user is free to select 2 more outputs, thereby mapping all 6 input device parameters to output parameters. The reason for only setting up 4 parameters automatically is to keep the interaction more clear for the user as a default. Controlling 6 parameters at a time is cognitively demanding and difficult to initially overview. 4 parameters as a default provided an adequate balance—later informal evaluation also showed that only few users utilized more than 4 parameters as they designed their novel instrument. When a musical output is selected, additional controls are highlighted for adjusting additional parameters. Additionally, the user is able to adjust each musical parameter of the musical output using a simple graphical user interface—see Figure 4. In this way it is possible to fully adjust the settings of a synth controlling only few of the parameters realtime with the input controller.

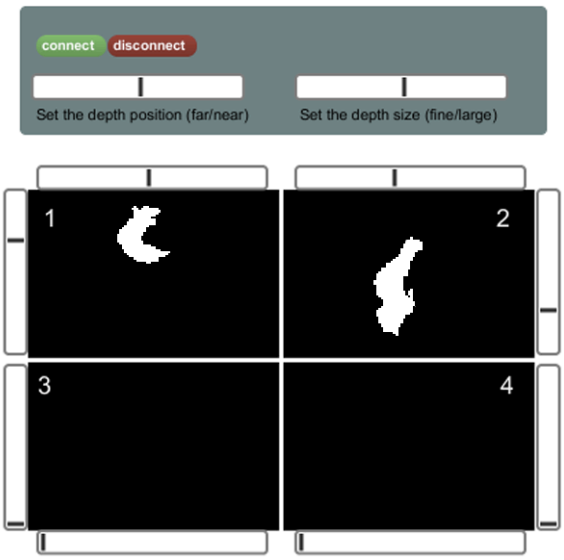

Figure 5: The interface for additional settings and control of the Kinect implementation. The user is able to detect X,Y positions in 4 different areas. Here only areas one and two are active.

4.3 Mapping and Additional Settings

The mapping between the two layers is adjusted using drop-down menus and sliders—see Figure 3. Besides being able to select input device, musical output and adjusting mapping between the two, the software also lets the user save presets, choose sound input from microphone or sound sample, set audio drivers, set musical scales, MIDI channels, etc.

5. THE SOFTWARE IN CONTEXT

The 6to6Mappr was used at a workshop for giving non-technical music conservatory students hands-on experience with using technology to enhance, augment or support musical expression. A great challenge—besides the large amount of participants (more than 100 students)—was that the participants had many different musical backgrounds ranging from solo musicians and vocalists to audio technicians and even music management students. The workshop (which also consisted of other sessions such as concerts, talks, debates) was carried out at the The Rhythmic Music Conservatory Copenhagen over a duration of three days—with exercises based on the 6to6Mappr lasting for approximately 2-3 hours each day. Experiences gained from the workshop will be used here as an informal evaluation of how well the system performs in context.

5.1 Day 1

The first day the students were to gain experience with installing, running and getting their first feel for the software. The idea was to get all students to try all of the different input devices at least once in order to get a feel for how they could be utilized. External hardware was provided (Wii-motes, Kinects and Arduinos) although not enough for everyone to use at once.

All devices turned out to be fairly easy to get running—especially the Wii-mote and webcam based devices. While

the Kinect was also easy to connect, students found it difficult to understand the specific implementation of the Kinect depth map (they could set a depth threshold for when centroid tracking was active and output X,Y coordinates for centroids in four different quadrants—see Figure 5—the idea being to activate 4 different areas with their hands). Setting up the Arduino was also difficult although pre soldered sensors where made available. Students seemed to be afraid of breaking sensors by handling or connecting them wrongly.

On the first day students used most time controlling the modular synths or audio effects applied to sound samples they loaded into the software. The two physical models, the granular effects and the MIDI routing seemed to be explored more during day two.

5.2 Day 2

On the second day of the workshop students were put into performance groups (of around 8 persons per group). They were given 2 hours to put together a performance that would somehow utilize novel input devices and the functionalities of the software. Here more time was used on routing MIDI data for controlling software synthesizers in their preferred DAW, but also for manipulating sound being played by others using the basic effects implemented in the 6to6Mappr. Among the most successful implementations was a voice and orchestra manipulated by a granular effect controlled by MacBook shake and tilt, and also a simple pentatonic bell-like MIDI synth controlled by tracking colored balloons being thrown into the air. Most of the devices were implemented as is, meaning for instance that the wii-controller was simply manipulated by hand (as opposed to attaching it to some larger device altering the physicality of the gesture involved). In that respect students did not seem to have time enough to work on actually *building* a new interface.

5.3 Day 3

On the final day students were asked to (individually or in small groups) prepare an instrument for a large improvisation jam session lasting 30 minutes in the large concert hall. The jam session involved all participating students playing at the same time demanding more control and greater ability to adapt to changes in the overall musical environment. Most students used laptop based devices (MacBook sensors and webcam) and most used MIDI routing and audio effects for musical output. No one used Arduino and Kinect for this final individual event. It seemed like more focus was put on gaining sufficient musical control in comparison to day 2 where more focus was put on musical performance, establishing a good communication with the audience. Therefore setups were smaller and confined to finer gestures.

5.4 Initial learnings

Performance-wise the software worked without any issues reported throughout the workshop. Everyone was able to work with the software and achieve gestural control of musical properties. Complete novices had a few issues with routing sound in and out of the software, getting it to work with sound cards, etc. This has not been the main focus while developing the 6to6Mappr but will be dealt with in future versions. As mentioned earlier it is important that the software is very easy to set up.

For this kind of short workshop we found that the possibilities should have been more restricted in order to get students more in depth with working with the actual mapping layer. Almost all students applied some sort of one-to-one mapping from input parameter to musical output parameter. Only a few worked with scaling between input and output and even fewer used parameter changes (as opposed to absolute values) to control output. For future similar workshops either more time is required to settle on a device of choice or more constraints should be introduced. While presenting the user with the many built-in musical outputs was powerful in learning the tool fast, it seemed that most were interested in using the 6to6Mappr for routing MIDI giving them more musical flexibility than using the built in musical outputs. A future implementation might put more emphasis on easy MIDI routing with a minimum of built-in sound used only to get the user up and running fast.

The current version of the 6to6Mappr is available at `http://media.aau.dk/~stg/6to6`.

6. CONCLUSIONS

We have presented a piece of open source software called 6to6Mappr that lets novice users explore musical mapping of novel input device data to musical output. It lets the user select between several different input devices and choose a maximum of 6 input parameters of said device, mapping them to 6 audio output parameters (synths/effects/MIDI). Additional functionality provides the opportunity to adjust the mapping layer, for instance using one input parameter for gating another or adjusting the scaling of an input parameter to fit the desired musical output. The idea behind the software has been to make a tool that anyone could learn to use in a very short amount of time. The software was used as the basis for a hands-on workshop that had the purpose of introducing non-technical conservatory musicians to novel musical interaction provided by new technology.

Overall the software was easy to use for almost everyone involved in the workshop, the strengths being the ease of setup and intuitive understanding of the mapping from input to output.

In its current form, the system is perhaps mostly an educational tool, however many will be able to use it for other purposes. There seems to be a great segment of musicians that are interested in new ways of controlling music but who are not motivated to use time and effort on familiarizing themselves with Max/MSP, PureData or similar technological platforms that until now have been necessary for exploration of this area. It is our hope that open source software like the one presented here will enable more interesting music to be created and performed as non-technical musicians gain access to these capabilities.

7. ACKNOWLEDGMENTS

We would like to thank Ben Cahill and David Joakim Stubbe Teglbjærg for bug testing and for helping out during the workshop.

8. REFERENCES

[1] P. Cook. Principles for designing computer music controllers. In *Proceedings of the New Interfaces for Musical Expression (NIME)*, 2001.

[2] R. Fiebrink, D. Trueman, and P. Cook. A metainstrument for interactive, on-the-fly machine learning. In *Proc. NIME*, 2009.

[3] R. Fiebrink, G. Wang, and P. Cook. DonÕt forget the laptop: Using native input capabilities for expressive musical control. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 164–167, 2007.

[4] S. Gelineck and S. Serafin. A practical approach towards an exploratory framework for physical modeling. *Computer Music Journal*, 34(2):51–65, 2010.

[5] N. Gillian, R. Knapp, and S. O′Modhrain. A machine learning toolbox for musician computer interaction. In *Proceedings of the 2011 International Conference on New Interfaces for Musical Expression (NIME11)*, 2011.

[6] A. Hunt, M. Wanderley, and M. Paradis. The importance of parameter mapping in electronic instrument design. *Journal of New Music Research*, 32(4):429–440, 2003.

[7] A. Jensenius, R. Godøy, and M. Wanderley. Developing tools for studying musical gestures within the max/msp/jitter environment. In *Proceedings of the 2005 International Computer Music Conference*, pages 282–285, 2005.

[8] M. Kaltenbrunner. reactivision and tuio: a tangible tabletop toolkit. In *Proceedings of the ACM international Conference on interactive Tabletops and Surfaces*, pages 9–16. ACM, 2009.

[9] N. Wolek. A granular toolkit for cycling74's max/msp. In *SEAMUS 2002 National Conference*, 2002.