MAPPING STRATEGIES IN DIGITAL MUSICAL INSTRUMENTS: CONCEPTS AND TOOLS

Marcelo M. Wanderley,
with contributions from Joseph Malloch
and Stephen Sinclair
Input Devices and Music Interaction Laboratory
Centre for Interdisciplinary Research in Music Media and Technology, McGill University, Quebec
marcelo.wanderley@mcgill.ca

Abstract

This article discusses the importance of the choice of mapping strategies for the performance with digital musical instruments (DMI) and presents tools for the intuitive design of mapping strategies in a collaborative setting.

1. What is Mapping?

Digital musical instruments (cf. Figure 1) are composed of a *control surface* (also called g*estural controller*, input device or *hardware interface*) typically controlling sound synthesis parameters and/or other types of outputs such as video, vibrotactile or force-feedback. The definition on how this control is defined, i.e. how the outputs of the control surface will affect the inputs of the sound synthesis algorithm (and possibly vice-versa) is called *mapping*.

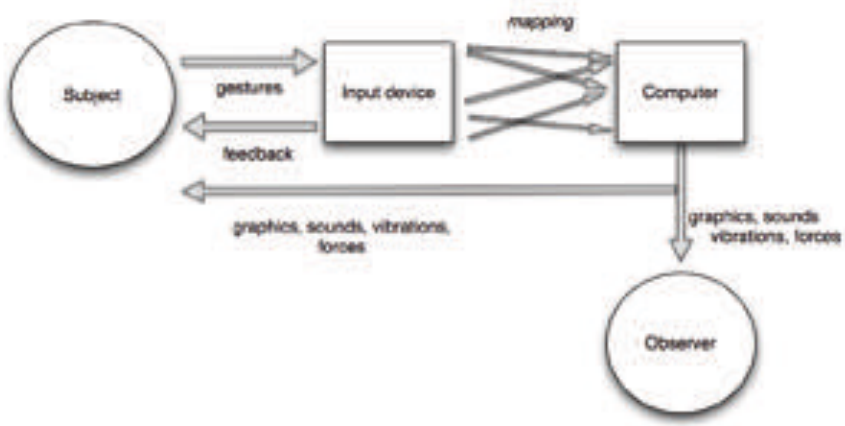

Figure 1: A representation of a Digital Musical Instrument (DMI)

Due to the inherent separation between the control surface—usually a hardware device using a variety of electronic sensors and actuators to respectively capture performer actions and render vibrations and force feedback him or her—and the sound synthesis algorithm—typically a computer program simulating the way a structure vibrates or how sound propagates in a solid body (physical models), the spectrum of a given sound or some abstract mathematical equation (signal models)—the choice of mapping is an integral part of DMI design.
When using physical models there may be "natural" ways to create mappings, for instance, connecting a force captured by a sensor to the force input of a sound synthesis algorithm. There is still a mapping to be made, since the two force variables are present in different systems (the control surface and the computer) and some signal processing will likely be required. However, nothing prevents the designer from using mappings that are not physically-inspired—perhaps connecting a position output from a control surface to the force input of a synthesis algorithm—even if the results will likely be surprising[1]. There are many combinations of control surface and synthesizer design that will not allow these "natural" mappings, but which may yield interesting and rewarding artistic results.

In other words, changing the mapping between controller and synthesizer may dramatically affect the performance technique, sound output, and the audience's perception of a DMI.

Nevertheless, although this has been understood since the beginning of electrical (and later electronic) musical instruments, to our knowledge there is still no clear set of guidelines on how to design mapping strategies, especially for the case of sound synthesis with signal models. Also, there are almost no published reports on how to test the efficiency of chosen mapping strategies, with the notable exception of [Hunt, 1999]. To our knowledge, the choice of mapping strategies is still an open problem, and in many cases a trial-and-error process.

(1) It may also just not be possible to easily measure the physical variable needed to be mapped to the input of the sound synthesizer, for instance due to obtrusiveness of sensors (at least, of affordable sensors). In such cases compromises will likely need to be made and taken into account in the design of mappings.

Finally, because recently a large DIY (do-it-yourself) electronics community has blossomed thanks to the widespread availability of electronic sensors and intuitive microcontroller platforms—e.g. the Arduino [Kushner, 2011]—as well as low cost, powerful computing platforms able to generate complex sounds (e.g. cell phones and tablets), the ability to quickly and efficiently prototype mapping strategies is of utmost importance for the design of novel DMIs.

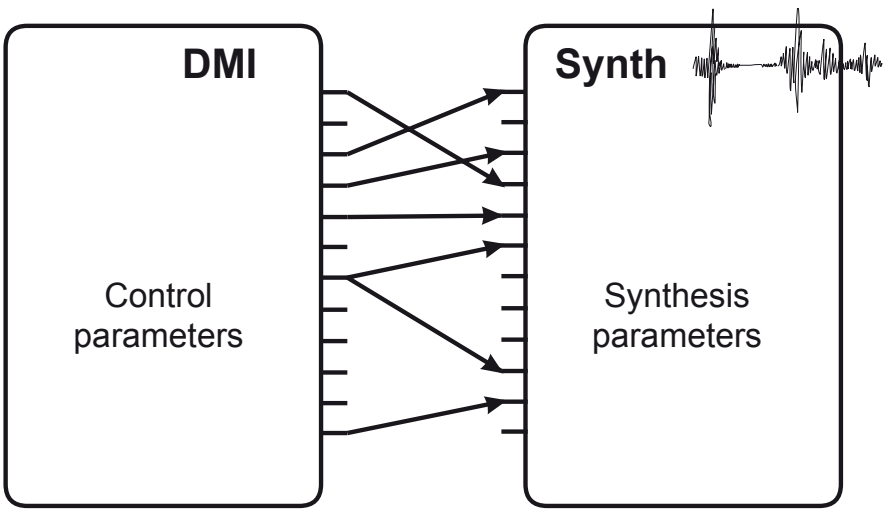

Figure 2: A simplified representation of mapping between sensor data and sound synthesis parameters, from [Malloch, Sinclair and Wanderley 2007b]

2. Tools for Mapping

It is then apparent that tools are needed to help designing and prototyping mappings in novel digital musical instruments. But how to approach the design of mapping strategies? Some of the possible requirements for a mapping prototyping tool include:

The tool should allow designers without extensive technical knowledge to create and modify their own mappings. This requirement eliminates the common need from composers and performers for continuous help from technical assistants who are able to program software tools such as Max/MSP or PureData, or even lower level programming languages such as C.

The tool should provide the ability to easily create and destroy mappings on the fly,

so that experimentation is encouraged.

Gesture devices and synthesis algorithms should automatically present and describe themselves in the mapping prototyping environment. This requirement eliminates the need to understand the variables sent (received) in the system, including their types and ranges.

Availability of a range of signal conditioning functions useful to a variety of situations (sensor data linearization, leaky integration, simple low-pass filtering, etc.), for instance, similarly to the work of H.-C. [Steiner, 2005].

3. LibMapper

Our solution to these requirements is a software tools called LibMapper [Malloch, Sinclair and Wanderley 2007b] (www.libmapper.org), designed as part of the Digital Orchestra Project (http://www.music.mcgill.ca/musictech/DigitalOrchestra/) at McGill University.

LibMapper offers the following features:

- An open-source, platform-independent network architecture which lends itself to a distributed mapping designs.
- Designed as a decentralized network for the management of peer-to-peer data connections using the *Open Sound Control* communication protocol.
- Controllers and synthesizers are able to announce their presence and make their input and output parameters available for arbitrary connections on-the-fly.
- Controllers and synthesizers can interface with each other over UDP/IP by means of an OSC-controlled arbitrator.
- A (memory-less) graphical user interface (GUI) allowing gestural data streams to be dynamically connected and modified. It forms a separate program from the other devices, transmitting and receiving OSC messages on the same multicast admin bus. An arbitrary number of GUIs can be used simultaneously on the network, allowing collaborative design of mapping strategies.

Presently, two GUIs have been designed to represent data at different levels, for