A Flexible Tool for the Visualization and Manipulation of Musical Mapping Networks

Aaron Henry Krajeski

Department of Music Technology
Schulich School of Music, McGill University
Montreal, Canada

July 2013

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts.

2013/07/06

Abstract

This report describes the use of LaTeX to format a thesis. A number of topics are covered: content and organization of the thesis, LaTeX macros for controlling the thesis layout, formatting mathematical expressions, generating bibliographic references, importing figures and graphs, generating graphs in MATLAB, and formatting tables. The LaTeX macros used to format a thesis (and this document) are described.

Acknowledgments

Acknowledge this, asshole.

Preface

There are some things I should probably pre-face, certainly not reface.

Contents

List of Figures

various graphs of response time (discussion) screenshot of drawing screenshot of saving/loading screenshot of main view screenshot of grid view

List of Tables

List of Acronyms

IDMIL	Input Devices for Musical Interaction Laboratory
MVC	Model View Controller
DMI	Digital Musical Instrument
OSC	Open Sound Control
GUI	Graphical User Interface
API	Application Programming Interface

Chapter 1

Introduction & Motivation

"In order that our tools, and their uses, develop effectively: (A) we shall have to give still more attention to doing the approximately right, rather than the exactly wrong..." (Tuckey 1965)

Throughout the vast majority of human history the term "musical instrument" has signified both the physical object with which the musician interacted *and* the direct source of the sound created: a violin with vibrating strings, a reeded saxophone, a timpani with its membrane, etc. With the advent of electronic sound in the late 19th century, it became possible for interactive objects to be separated from the sound producing devices they control (Chadabe 2000). As technological development progressed, so did the capacity to divide musical instruments into independent parts. With digitization it is now not only possible to arbitrarily connect a control element to any sound synthesis dimension, but also to modify this association according to the whims of the user. Since mechanical linkages are no longer necessary in the design of musical instruments, control surfaces can, and often do, take on a variety of wild and arbitrary shapes and modes of interaction.[1] All that is necessary for this process is for control devices to output some kind of electronic signal that other, sound-producing instruments can accept. With no obvious means of implementation, the success or failure of these new digital musical instruments (DMIs) often depends on how artfully their output signals are "mapped" to synthesis parameters.

More and more frequently, the mapping itself becomes a part of the expressive element

[1]International Conference on New Interfaces for Musical Expression. [Online]. Available: `http://www.nime.org/`. Accessed June 23, 2013.

of a musical work (Hunt and Kirk 2000), as it associates itself with both composition and performance with certain DMIs. Thus is becomes necessary for mapping to be dynamic and interactive: sometimes poured over in composition studios, or sometimes edited mid-piece. Musicians are not necessarily computer programmers, thus ideally the act of mapping should not require computer expertise. This means that on top of the low-level layer of interactive mapping (simply instructing a machine to connect signals to others in specific ways), there needs to exist an interface to make such an activity easy, logical, intuitive and in line with the artistic process.

As the actual act of mapping is as expansive and nebulous as the instruments it hopes to assist, thus the design of such a mapping interface presents many interesting challenges. Due to the tremendously wide variety of possible use cases, several seemingly contradictory goals emerge: What is the best way to visually represent complex musical networks while simultaneously allowing for the user to easily manipulate them? How can systems with many devices and signals be well represented while still allowing in-depth control of small networks? How can an interface be transparent to non-technical users while still accommodating all possible functionality that advanced users may wish to use?

1.1 Context and Motivation

The world of digital musical instruments is still dominated by keyboard type input devices. Though many novel DMIs currently exist (and many more are being created) these devices are usually unique and often difficult to use without their creator being present (Cook 2009). Since mapping is such an important feature of DMIs, a means of transparently editing them could inspire more people to use novel musical controllers. In response to this challenge, the libmapper protocol was created at the Input Devices and Music Interaction Laboratory (IDMIL) (Malloch et al. 2008). The tool is summarized by its website as follows:

> “libmapper is an open-source, cross-platform software library for declaring data signals on a shared network and enabling arbitrary connections to be made between them. libmapper creates a distributed mapping system/network, with no central points of failure, the potential for tight collaboration and easy parallelization of media synthesis. The main focus of libmapper development is

to provide tools for creating and using systems for interactive control of media synthesis."[2]

In its most basic state, libmapper takes the form of an application programming interface (API). APIs are primarily a means for different pieces of computer software to communicate with one another. The only possible way to communicate directly with the libmapper API is through coded text. For example, the following portion of code causes a synthesizer to announce itself and begin communicating with other devices on a libmapper-enabled network (Malloch et al. 2008):

```
#include <mapper.h>
mapper_admin_init();
my_admin = mapper_admin_new("tester", MAPPER_DEVICE_SYNTH, 8000);
mapper_admin_input_add(my_admin, "/test/input","i"))
mapper_admin_input_add(my_admin, "/test/another_input","f"))

// Loop until port and identifier ordinal are allocated.
while ( !my_admin->port.locked || !my_admin->ordinal.locked )
{
   usleep(10000); // wait 10 ms
   mapper_admin_poll(my_admin);
}

for (;;)
{
   usleep(10000);
   mapper_admin_poll(my_admin);
}
```

Fig. 1.1 A sample of libmapper code

This is obviously inaccessible to users who do not have the time or desire to read through documentation files, or those who have no knowledge of programming semantics. A steep learning curve is especially a problem for a network tool like libmapper: because it is primarily a means of communication between instruments, it can only be successful if it is widely adopted. A libmapper-enabled controller will only be useful if many high quality libmapper synthesizers exist. In turn, synthesizer makers will only have incentive

[2]libmapper: a library for connecting things. [Online]. Available: libmapper.org. Accessed June, 2013

to incorporate libmapper into their designs if there are already controllers that use the system.

An API can be contrasted with a graphical user interface (GUI), an interface that contains abstractions on top of the raw code. These abstractions can be features like buttons, menus, visual representations of data, etc. In general, GUIs are designed to be familiar to those who have used digital devices in the past, and thus easy to learn and use. Two GUIs have been created for libmapper (see section 2.4.4): a basic interface built in Max/MSP[3] and vizmapper (Rudraraju 2011), a more abstract representation of a libmapper network. Both of these GUIs have their strengths, yet neither adequately meets the full range of possible use cases for libmapper. A more flexible approach is required if the GUI is to be usable in situations with hundreds of signals, transparent for systems with multi-leveled hierarchical devices, intuitive during performances where devices output light and haptic feedback as well as sound, and responsive for tasks where speed of manipulation is an absolutely necessity.

With such an interface in place, libmapper can greatly expand its user base. As a result, more controller and synthesizer designers may choose to incorporate libmapper into their devices, and in turn these devices will be easier to learn and use. Hopefully the end result will be greater adoption of non keyboard-based DMIs in the electronic music community.

1.2 Project Overview

The focus of this project is to create a graphical user interface for libmapper. This interface aims to be flexible and intuitive, simultaneously allowing for useful control of the full range of possible libmapper networks while also not intimidating non-technical users with complexity. The presupposed solution to this problem is to provide users with multiple independent modes of viewing and interacting with the network. Certain view modes can excel in providing precise control, while others can help users understand the structure of complex networks. The idea is to provide multiple imperfect solutions to an unsolvable problem, so that each can be "...approximately right, rather than exactly wrong" (Tuckey).

This project was structured in four major, non-sequential parts: a review of prior visualized mapping interfaces, the integration of presently available GUIs for libmapper, the

[3]MAX: You make the machine that makes your music. [Online]. Available: `http://cycling74.com/`. Accessed June 17, 2013

extension of interface features and the collection of user feedback. Results of the research phase informed implementation and are presented here. Development began by updating a cross-platform implementation of the current Max/MSP-based GUI, while integrating functionality from vizmapper. New view modes were integrated into design while refining functionality of the previous ones. Throughout the design process, the GUI was provided to potential users who gave feedback on the strengths, weaknesses and potential avenues for improvement.

1.3 Thesis Overview

The remainder of this document is organized as follows. Chapter 2 outlines concepts necessary for providing context for this thesis project. A wide variety of domains inform the creation of a musical mapping interface. Special attention is paid to mapping theory, data visualization, relevant existing user interfaces, user centric design techniques and specifics of libmapper itself. Chapter 3 describes the design and implementation of the libmapperGUI. This chapter presents design decisions made, technical details of implementation and how the user-centric approach informed the process. Chapter 4 evaluates results, both on the empirical level of software performance as well as qualitative user feedback. Finally, Chapter 5 presents conclusions of the work and suggests further developments for the software.

1.4 Contributions

The contributions of this thesis are: the exploration of issues related to user interface design for musical mapping networks, the design and implementation of an interface for libmapper that aims to improve on usability and flexibility of the system, and this thesis document, which describes the research and development therein.

Chapter 2

Background

Dynamic mapping is becoming an increasingly important requirement for digital musical instruments. This chapter surveys currently available tools that allow for manipulation of musical and non-musical networks in real time. The first section presents a review of mapping itself, both from a theoretical and a musical standpoint. This portion also introduces the libmapper application programming interface. The second section reviews relevant work in the visual representation of information. The following portion describes applicable techniques in user interface design. Finally, a review of user interfaces for mapping is presented.

2.1 Mapping

At the most fundamental level, *mapping* is the act of associating two or more sets of information. Mappings can be mathematical, computational, linguistic (like translation), geographic, or even poetic[1]. Within the context of DMI design mapping is the relationship between sensor outputs and synthesis inputs. The entire character of a new instrument can be drastically altered though mapping, even while control surface and sound source are held constant (Hunt et al. 2003). As a result, the theoretical formalism of mapping becomes yet another necessary tool in the modern instrument designer's arsenal.

[1]What is metaphor if not the association of unlike things?

2.1.1 Mapping Theory

Mapping as function and the four mapping classes

From the perspective of mathematics, the term *mapping* is very nearly synonymous with *function* (Halmos 1970), as both describe how one set of numbers corresponds with another. The first group is commonly refered to as the *domain* and the second as the *codomain* or *range*. An in-depth review of functions in mathematics is beyond the scope of this thesis, however a few fundamental examples will be useful for reference in section 2.1.2. The following are instances of two basic types of mathematical functions:

$$y = 2x - 1 \tag{2.1}$$

$$y = x^2 \tag{2.2}$$

Each function takes a single input value (x) and *maps* that number onto its range (y). The fact that each of these equations take in only a single number as input, and output a single number in turn, means they can be graphed in a two dimensional space. This is not necessarily the case, as functions can input and output lists of numbers (vectors). Mathematically they are not very interesting, but they represent two fundamentally different *kinds* of functions.

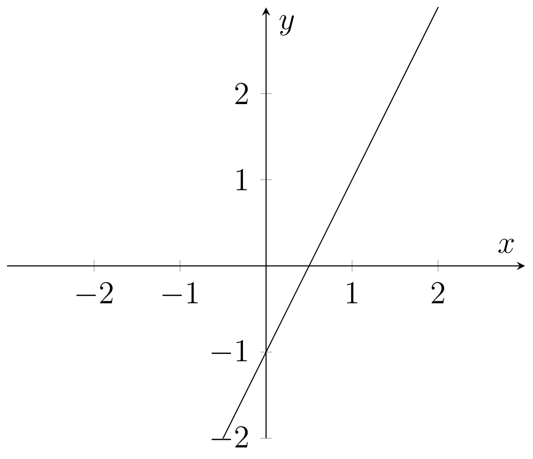

Fig. 2.1 The function described in equation 2.1, graphed in two dimensions.

For equation 2.1 each input value has *one and only one* corresponding output value. The same is true if the function is to be inverted, as each output value corresponds to only one input value. The range is simply a scaled and shifted version of the domain. The mapping's *one-to-one* nature can clearly be seen in figure 2.1.

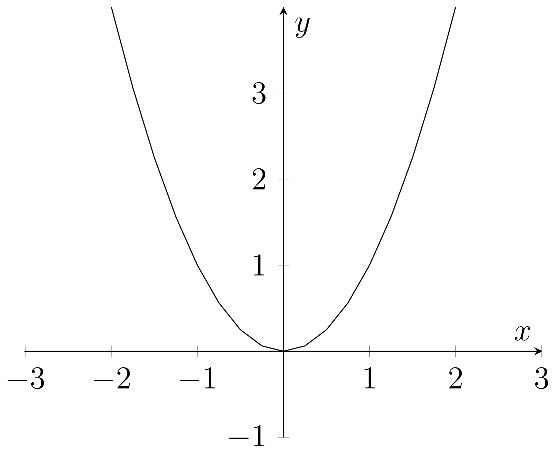

Fig. 2.2 Equation 2.2 projected on the Cartesian plane.

This is not the case for equation 2.2, for although each input has only one output, single positions in the codomain can have multiple corresponding inputs (e.g. both 3^2 *and* -3^2 are equal to 9). Thus we can consider equation 2.2 to be a classical example of a *many-to-one* mapping. In figure 2.2 the range of the function is wrapped back onto itself such that a horizontal line could intersect the curve twice.

Two more categories of mapping are relevant to instrument design, an example of each:

$$y = \pm\sqrt{x} \tag{2.3}$$

$$y = \pm\sqrt{1 - x^2} \tag{2.4}$$

They are not considered to be functions by mathematicians[2], but are nonetheless important for our purposes. In equation 2.3 a single input can result in multiple outputs (an input of 4 results in the output of *both* 2 and -2), yet each output has only a single input. This is simply the inverse function of equation 2.2, and is an example of a *one-to-many* mapping. On a graph of such a mapping, a *vertical* line may cross at multiple points. The final equation is that of a circle centered at the origin with a radius of one. This is a *many-to-many* mapping, as both it an its inverse result in multiple outputs from a single input.

[2]In mathematics, a true function can have no more than one output value for every input value.

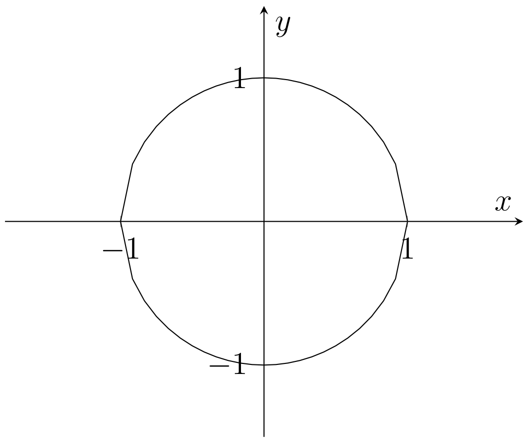

Fig. 2.3 Equation 2.4, a many-to-many mapping.

Though a graphical plane is the most common way for mathematicians to visualize two-dimensional functions, drawing the direct association between input and output will be more useful going forward. Figure 2.4 provides an illustration of such an approach. The astute reader will notice a striking similarity to the GUI view mode described in section 3.1.1, and a many to many mapping

Mapping as association

In computer science, a mapping is less commonly referred to as a function and more usually called an *associative array* or a *dictionary*, though the word *map* is also used (Mehlhorn and Sanders 2008). This type of data structure[3] is generally the most flexible way for computers store information. An associative array consists of key/value pairs, where the *value* is the data to be stored and the *key* is the reference to that data.

Table 2.1 An example of key/value pairs (contries and currencies)

key	value
Canada	Dollar
France	Euro
Bahrain	Dinar
Germany	Euro
Angola	Kwanza
USA	Dollar

[3]What computer scientist call particular ways of storing and organizing data.

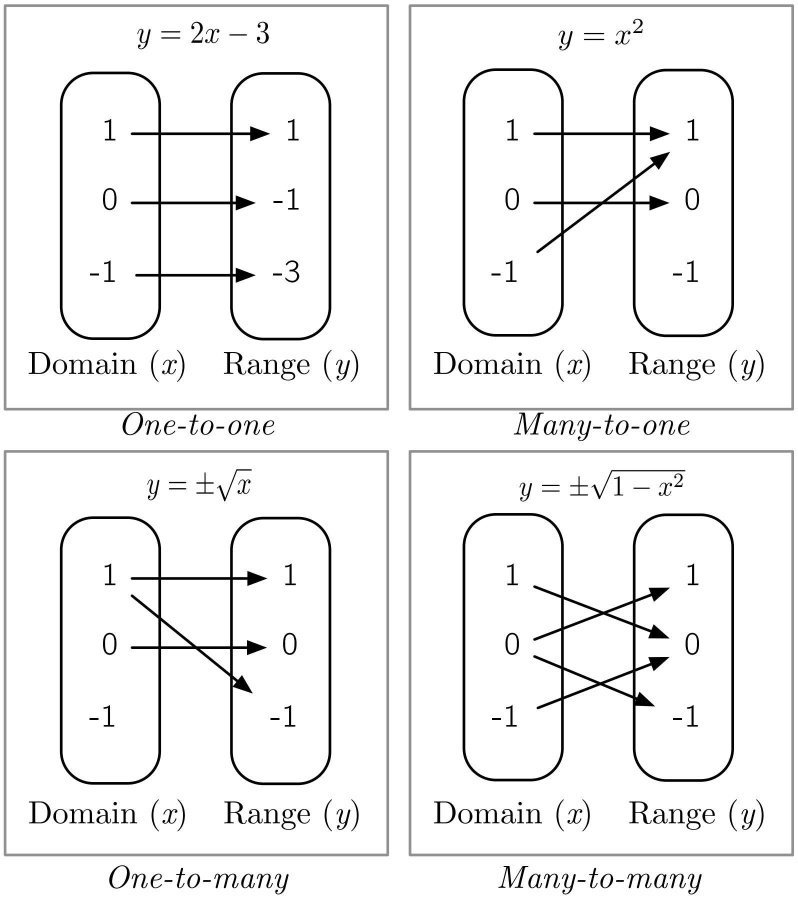

Fig. 2.4 The four mapping classes

In table 2.1 the data is non-numeric and associations between keys and values are arbitrary (from a mathematical point of view). There obviously exists no distinct function that can transform a countries name into the name of its currency, thus the computer must explicitly remember the associations between the words in the form of a *hash table.* At the lowest level, computers store information on a vast array of zeros and ones, and the value "Kwanza" only arises through a non-trivial process of encoding and decoding. In order to retrieve it the computer *must* know where it can be found. The hash table takes the input of a key, finds the address for the value and returns it. In this way the hash table is literally the association between two sets of data and thus the mapping between them.

The four mapping classes outlined in the above section are not limited to the functional domain. The associative array in table 2.1 is another example of a many-to-one mapping, as many countries have the same currency name. In this vain a one-to-many mapping could be the same keys with values switched to “Former Monarchs” (“France” would map to both “Louis XVI” and “Napoleon III”, etc), while a value of “Official Languages” would be a many-to-many mapping (“Canada” maps to both “English” and “French” while both “Canada” and “France” map to “French”).

Though most applicably represented in computer science, data structures like associative arrays appear in many other fields. Library card catalogs (one-to-one), multilingual dictionaries (many-to-many) and address books (many-to-one) are all very straightforward instances of key/value pairs. In a library card catalog the call number even acts as a sort of hash table. In a large library, a book that is placed in the incorrect position on the shelves will likely be lost for a very long time. Thus the system must remember the keys (titles) and associated values (the books themselves) but also their positions in memory, their call numbers.

The above concepts from mathematics and computer science supply a base of knowledge for understanding mappings theoretically and provide a starting point for analyzing them in a musical context.

2.1.2 Mapping for Digital Musical Instruments

- Cite vijay about the inputs-outputs/outputs-¿inputs/sources-¿destinations thing
- many2one/one2one/one2many/many2many, certainly can cite a gestures class readgin (Hunt et al. 2000)
- These two contexts have also been referred to as a systems point of view and a functional point of view respectively [11]. Systems = network, functional = connection properties Two types of mapping (Nort 2010)

2.1.3 libmapper

joe’s libmapper paper: (Malloch et al. 2008) joe’s other paper?

Open Sound Control

OSC: (Wright and Freed 1997)

Structure of libmapper Networks

Control of libmapper Devices and Signals

1. GDIF: (Jensenius et al. 2006) Describes the namespace that libmapper uses, "Ideally, it should be possible to store all sorts of data from various commercial and custom made controllers, motion capture and computer vision systems, as well as results from differ- ent types of gesture analysis, in a coherent and consistent way. This would make it possible to use the information with different software, platforms and devices, and also allow for sharing data between research institutions."
2. disembodied performance
3. Wanderley's mapping paper (Hunt et al. 2000)
4. Jamoma (Place and Lossius 2006)
5. surely some other stuff from class
6. METADATA, and data

2.2 Data Visualization

The graphical user interface described in this thesis presents users with solely visual information. As it is a tool to be used with primarily sound producing objects auditory feedback is problematic and common digital devices (laptops, tablets, etc.) provide us with no means of producing haptic response. Mapping systems can contain tremendous amounts of information: device names, digital addresses, numbers of signals, signal names, units, ranges, data types, expressions, parent devices and any kind of meta-data a libmapper user may choose to add to his or her devices. As a result, it is necessary a review how best to visually represent vast amounts of structured data.

2.2.1 Graphical Perception

Heirarchical Structures

Dense Information

2.2.2 Visualization Techniques

Filtering

Spark Lines

Dash Plots

2.2.3 Visualization Systems

Allosphere?, Braun Braun: view OSC data flows (Bullock 2008), HEB?

1. Allosphere? :(Höllerer et al. 2007)
2. Heirarchical edge bundling: (Holten 2006)
3. Tukey: (Tuckey 1965)
4. Envisioning information: (Tufte 2006)
5. Beautiful Evidence: (Tufte 1990)
6. The other Tufte book I have at home.
7. OSC data flows with Braun (Bullock 2008)

2.3 User Interface Design

2.3.1 A Brief History of Electronic User Interfaces

2.3.2 Task Analysis

2.3.3 Recall and Recognition?

2.3.4 Collaborative Network Interfaces

MPG Care Package (Wolek 2010)

2.3.5 The Model-View-Controller Architecture

MVC Krasner Pope (Krasner and Pope 1988)

2.3.6 User Centric Design

Organizational context (Kling 1977) Usability testing (Corry et al. 1997) Information professionals (Schulze 2001)

1. Inclusive interconnections (Booth 2010)
2. Sense Stage (Baalman et al. 2010)

2.4 Relevant User Interfaces

2.4.1 Junxion

Junxion (STEIM 2004)

2.4.2 Osculator

Osculator: mapping OSC stuff (Wildora 2012)

2.4.3 Other Similar Interfaces

Integra (Bullock et al. 2011) Eaganmatrix: GRID VIEW! (Audio 2103) Patchage: a linking, dragging, connecting interface (Robillard 2011)

2.4.4 Prior Interfaces for libmapper

Vizmapper (Rudraraju 2011)

2.5 Summary

Chapter 3

Design & Implementation

Development of a graphical user interface for libmapper creates a unique challenge. Obviously such an interface is a practical tool, and should function as such, yet it also must work in concert with DMIs which are inherently designed for abstract and creative use. For the purposes of this project, the assumed solution to this innate paradox is to provide the user with multiple independent modes of control. This assumption was made based on experiences with prior user interfaces for libmapper (vizmapper, max mapperGUI): for each interface users reported excellent functionality for certain use cases, and poor functionality for others. Libmapper itself is an extremely flexible API that makes few assumptions as to the network of devices and signals, nor how they are being mapped. It is fitting that a GUI for libmapper would be equally as flexible. In lieu of a single perfect solution for network visualization an interactivity, providing users with various independent solutions provided a good compromise.

3.1 Development of a “Modular” Interface

3.1.1 List view

3.1.2 Grid view

3.1.3 Hive plot

3.1.4 Cluster view (vizmapper)

3.2 The Model-View-Controller

Because a modular design is desired, the Model-View-Controller (MVC) metaphor for structuring software applications as described in [KrasnerPope88] was used as a general framework for structuring the application. In fact, the whole scale swapping in and out of independent visual modes can be thought of as a quintessential implementation of MVC.

3.2.1 The Model

The model consists of an abstract copy of the network, residing on the local machine. Independent views can consult this data, but cannot directly modify it.

3.2.2 Controller-View Pairs

3.3 Graphical Design

wiggly arrows

3.3.1 Typography

3.4 User Centric Design

use cases

3.5 Robustness and Responsiveness

speed tests

Chapter 4

Results & Discussion

4.1 Undoing and Redoing in a Collaborative Distributed Environment

4.2 Edge Use Cases

4.3 User Feedback

4.4 Modular vs Hard-Coded

4.4.1 Was the approach successful?

Are sections graphically unified? (Is this even necessary?)

4.5 Visualization vs Interaction

4.6 Unimplemented Features

1. Prefix filtering
2. Network selection

4.7 Different namespaces

Chapter 5

Conclusions & Future Work

5.1 Summary and Conclusions

5.2 Future Work

References

Audio, H. 2103. Eagen matrix. `http://www.hakenaudio.com/Continuum/eaganmatrixoverv.html`.

Baalman, M., V. de Belleval, C. L. Salter, J. Malloch, J. Thibodeau, and M. M. Wanderley. 2010. Sense/stage - low cost, open source wireless sensor infrastructure for live performance and interactive, real-time environments. In *Proc. of Linux Audio Conference*, 242–249.

Booth, G. 2010. Inclusive interconnections: Towards open-ended parameter-sharing for laptop ensemble. Master's thesis, University of Huddersfield, Huddersfield, England.

Bullock, J. 2008, March. Braun. Last accessed June, 19 2013, `http://sourceforge.net/projects/braun/`.

Bullock, J., D. Beattie, and J. Turner. 2011. Integra live: a new graphical user interface for live electronic music. In *Proc. of International Conference on New Interfaces for Musical Expression*, 387 – 392.

Chadabe, J. 2000, February. The electronic century part i: Beginnings, electronic musician. *Electronic Musician*: 74–90.

Cook, P. R. 2009. Re-designing principles for computer music controllers: a case study of squeezevox maggie. In *Proc. of the International Conference on New Interfaces for Musical Expression*, 262–263.

Corry, M. D., T. W. Frick, and L. Hansen. 1997. User-centered design and usability testing of a web site: An illustrative case study. *Educational Technology Research and Development* 45 (4): 65–76.

Halmos, P. R. 1970. *Native Set Theory.* Springer-Verlag.

Höllerer, T., J. Kuchera-Morin, and X. Amatriain. 2007. The allosphere: A large-scale immersive surround-view instrument. In *Proc. of Workshop on Emerging Displays Technologies*, 21 – 28. ACM Press.

Holten, D. 2006, September/October. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphiscs* 12 (5): 741–748.

Hunt, A., and R. Kirk. 2000. Mapping strategies for musical performance. *Trends in Gestural Control of Music*.

Hunt, A., M. M. Wanderley, and R. Kirk. 2000. Towards a model for instrumental mapping in expert musical interaction. In *Proc. of International Computer Music Conference*, 2–5.

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

Jensenius, A. R., T. Kvifte, and R. I. Godøy. 2006. Towards a gesture description interchange format. In *Proc. of the International Conference on New Interfaces for Musical Expression*, 176–179.

Kling, R. 1977, December. The organizational context of user-centered software designs. *MIS Quarterly* 1 (4): 41–52.

Krasner, G., and S. Pope. 1988. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming* 1 (3): 26–49.

Malloch, J., S. Sinclair, and M. M. Wanderley. 2008. A network-based framework for collaborative development and performance of digital musical instruments. *R. Kronland-Martinet, S. Ystad, and K Jensen. (Eds.): CMMR 2007, - Proc. of Computer Music Modeling and Retrieval 2007, Conference, LNCS 4969. Berlin Heidelberg: Springer-Verlag*: 401–425.

Mehlhorn, K., and P. Sanders. 2008. *Algorithms and Data Structures: The Basic Toolbox*. Springer.

Nort, D. V. 2010, January. *Modular and Adaptive Control of Sound Processing.* Ph. D. thesis, McGill University, Montreal, Canada.

Place, T., and T. Lossius. 2006. Jamoma: A modular standard for structuring patches in max. In *Proc. of International Computer Music Conference (ICMC 2006).*

Robillard, D. 2011, January. Patchage. `http://drobilla.net/software/patchage/`.

Rudraraju, V. 2011, December. A tool for configuring mappings for musical systems using wireless sensor networks. Master's thesis, McGill University, Montreal, Canada.

Schulze, A. N. 2001. User-centered design for information professionals. *Journal of Education for Library and Information Science,* 42 (2): 116–122.

STEIM. 2004, Summer. Junxion - products of interest. *Computer Music Journal* 28 (2): 105–107.

Tuckey, J. W. 1965, April. The technical tools of statistics. *The American Statistician* 19 (2): 23–28.

Tufte, E. R. 1990. *Envisioning Information.* Graphics Press.

Tufte, E. R. 2006. *Beautiful Evidence.* Graphics Press.

Wildora. 2012, May. Osculator. `http://www.osculator.net/`.

Wolek, N. 2010. The mpg carepackage: coordinating collective improvisation in max/msp. In *Proc. of the Society for Electro-Acoustic Music in the United States.*

Wright, M., and A. Freed. 1997. Open soundcontrol: A new protocol for communicating with sound synthesizers. In *Proc. of International Computer Music Conference*, 101–104.