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

June 2013

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

2013/06/26

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" 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 19$^{\text{th}}$ 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 modular 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 would 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 particular 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, 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 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 and synthesizers. 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 start 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. This is especially a problem for a network interface 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

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

exist. Synthesizer makers will also only have incentive 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, systems with multi-leveled hierarchical devices, performances where devices output light and haptic feedback as well as sound, or 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 the libmapper API. 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 updating and integration of presently available GUIs for libmapper, the extension of interface features and the collection of user feedback. Results

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

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 of improvement for each mode.

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 designs, specifically those built to handle a great deal of information and be adaptable to a variety of uses, 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

2.1 Mapping

2.1.1 Mapping Theory

2.1.2 Mapping for Digital Musical Instruments

2.1.3 libmapper

joe's libmapper paper: (Malloch et al. 2008) joe's other paper? (earlier), his master's thesis

Open Sound Control

OSC: (Wright and Freed 1997)

Structure of libmapper Networks

Control of libmapper Devices and Signals

1. GDIF: (Jensenius et al. 2006)

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

2.2 Data Visualization

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 User Centric Design

use cases

3.2 Development of a "Modular" Interface

3.3 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.

3.3.1 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.

3.3.2 Controller-View Pairs

3.4 Graphical Design

wiggly arrows

3.4.1 Typography

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.

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.

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.

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.