Introducing CrossMapper: Another Tool for Mapping Musical Control Parameters

Liam O'Sullivan, Dermot Furlong, Frank Boland
Trinity College Dublin
Electronic & Electrical Engineering,
Trinity College, Dublin 2,
Ireland.
lmosulli@tcd.ie, dfurlong@tcd.ie, fboland@tcd.ie

ABSTRACT
Development of new musical interfaces often requires experimentation with the mapping of available controller inputs to output parameters. Useful mappings for a particular application may be complex in nature, with one or more inputs being linked to one or more outputs. Existing development environments are commonly used to program such mappings, while code libraries provide powerful data-stream manipulation. However, room exists for a standalone application with a simpler graphical user interface for dynamically patching between inputs and outputs. This paper presents an early prototype version of a software tool that allows the user to route control signals in real time, using various messaging formats. It is cross-platform and runs as a standalone application in desktop and Android OS versions. The latter allows the users of mobile devices to experiment with mapping signals to and from physical computing components using the inbuilt multi-touch screen. Potential uses therefore include real-time mapping during performance in a more expressive manner than facilitated by existing tools.

Keywords
Mapping, Software Tools, Android.

1. INTRODUCTION
Interactive music systems (IMS) have been described in terms of three functional stages; sensing, processing and response [3]. The processing stage must route the sensed input to the response stage in order to provide feedback. Existing software tools allow the experimentation with this *mapping* of control inputs onto output parameters. However, in the course of earlier research it became clear that space existed for a new approach to existing mapping interfaces. The initial development of such an interface is now presented.

1.1 Origins and Motivation
Research by the lead author involves tailoring surface-based computer interfaces such as tabletops and multi-touch screens for musical control [8]. This has required the investigation of mappings between the properties of onscreen graphical shapes and output sound parameters. During the course of a subjective study exploring aspects of perceptual analogies across sense modalities, it was necessary to vary and test the preferences for such mappings [9]. This was carried out using a graphical representation programmed in the Processing Development Environment [25] communicating over Open Sound Control (OSC) [21] with a sound synthesiser patch created in the *Max/MSP* environment [17]. However, it was found to be unwieldy to re-route control to parameters, as the functionality of the program in either environment had to be modified. A dynamically configurable node-based interface layer was desirable. It was soon realised that such an implementation would be useful in other aspects of the research into surface-based interfaces used as musical controllers. Tangible controllers and physical sensors can augment the native touch- and object-sensing capabilities of such devices [11]. An ability to map input from physical computing components and hardware devices without hard-coding this into prototypes would save time and effort.

It was also found that practitioners from related fields of research (such as computer game audio) engaged with the idea of dynamic mappings but were unfamiliar with the common tools used within the computer music and NIME communities. Even within the music-oriented fields, comments suggested a need for "a MIDI-yoke type" mapping solution and that a "standalone application with a light footprint could be very handy", particularly one that was "neater, more useable".

It was therefore decided that space existed for an easy-to-use interface with flexible, dynamic mapping and the ability to accommodate various communications protocols. This paper introduces a prototype software implementation that attempts to address these needs.

The paper is organized as follows: A brief review of literature discusses parameter mapping in musical interfaces to identify the key design goals of the tool. The software prototype is then presented, followed by an informal evaluation and discussion of the implemented features. Future enhancements to the tool are then suggested.

2. PARAMETER MAPPING
It has been said that the character of an interactive music system is largely defined by the mapping of inputs to outputs [4]. Much has been written on the subject that will not be repeated here and good overview of key aspects is available in a previous publication [5]. A pertinent observation made in that article is the importance of complex mapping schemes in developing expressive interfaces. A classic example of such a mapping is given, referring to the control of pitch on a violin; this may be approximated as a weighted sum of bow pressure and finger position. The benefits of using an intermediate mapping layer (possibly perceptually described) have also been observed [1]. This project addresses an aggregate of these

issues while also considering the following aspect of control mapping that has not attracted as much attention in the literature.

2.1 Mapping as Music

As has been noted in the context of musical performance, the legacy hardware arrangement of mouse, keyboard and video display coupled with the Windows, Icon, Menu and Pointer graphical user interface (GUI) is not always conducive to creative play or engaging interactions analogous to the feeling of 'flow' evoked by traditional instruments [5]. When considering the new interface, the question arose as to why the modification of mappings was not itself considered a creative task. What new forms of expression could be uncovered from exploring this added layer of abstraction, with mappings being treated as musical elements? An analogy could be made to the process of instrumentation, but with more flexibility as to the mixing and scaling of parts to particular voices.

Such ideas have previously been touched upon, for example, in a discussion of the mapping of control to digital audio effects (DAFx) used as part of musical performance [10] and in a review of composers' views on mapping in algorithmic composition [3]. While mappings are seemingly used as musical elements, there has not been widespread examination of the idea in practice. A real-time interface with expressive capability would allow experimentation with the 'playing' of mappings. A well-designed interface would be useful as visual feedback to the performer, providing a clear representation of the system state. Should it be desired (and with suitable aesthetic modification), it could also impart heightened audience appreciation of the evolving connections and relationships between gesture and output that may be used throughout a performance.

3. SIMILAR WORK

Several software systems exist that facilitate exploration of input-output parameter mapping. While some of these are sophisticated systems offering extensive functionality, it will be shown that a gap exists for the approach being outlined here due to the limitations described in each case.

A number of tools have origins in work on spatial interfaces for the control of musical processes in real-time [7]. The IRCAM *MnM* mapping toolbox, for example, is part of the *FTM* external object library [14] for *Max/MSP*. The user builds patches with existing *Max/MSP* GUI elements or programs their own compatible objects. A strength of the toolbox is its suitability for real-time play and use in performance. For instance, an example patch allows the specification of two-to-many mappings (useful for the control of a varied and nuanced timbre-space with a typical physical input device) using a two-dimensional controller and a set of linear sliders. The system can associate points on the controller with particular slider arrangements through a learning algorithm. The user is then free to navigate about the planar space, with the system interpolating between mappings. Similarly, the *MetaSurface* is an interface for interpolating between mapping 'snapshots' for two-to-many mappings [2]. An example implementation is included with the *AudioMulch* software [12]. In this version, communication is over OSC or MIDI only and is designed to control sound generation and modification modules included in the commercial software product. A more recent implementation is the *nodes* object first included in version 5 of *Max/MSP*. This allows many inputs to be weighted and combined to a single output based on the positions of graphical nodes, which may overlap. The distance to the mouse cursor and the size of the node are used to determine the effect of an input stream on the summed output.

The Input Devices and Music Interaction Laboratory at McGill University have also done considerable work on interfacing novel instruments with sound production systems. Their mapping tools for digital musical instruments (DMIs) are encapsulated in *libmapper,* a library of code in the C language with a focus on distributed devices communicating as peers over a network [16]. The Application Programming Interface (API) is feature-rich, providing access to signal processing functionality such as gesture analysis and the use of an intermediate mapping layer to move from gestural semantics to sound semantics. A web-based GUI allows quick selection of parameters via namespaces and mapping can be easily patched from inputs to outputs. The *libmapper* creators state that many-to-one mappings are not yet implemented with dedicated combining functions- input streams are simply interleaved when connected to the same output. Reasons for this include the difficulties in encoding the necessary relationships between gestures in the intermediate mapping layer, suggesting this should be done at the gesture analysis stage. There is also a perceived lack of a requirement for such mappings in practice, although the inclusion of combining functions was designated as future work. In addition, the *libmapper* program is designed to work with OSC-enabled devices and so does not support input from other communications protocols.

Junxion is a fully-featured routing application that can handle many signal types, including video [15]. It is a commercial release, is available for *OSX* on Apple computers only and uses traditional GUI elements such as sliders and pull-down menus.

While not strictly a mapping tool in the vein of the above, *OSCulator* [23] provides connectivity to human interface devices (HIDs) and software-emulated controllers such as *TouchOSC* [27]. The application is a commercial release and is solely available for OSX. Other software programs exist that perform tasks related to parameter mapping, such as MIDI-to-OSC conversion (e.g. *OSC VST Bridge* [24] and *Moco* [20]) and mouse event capture and conversion to MIDI (e.g. *MouseTrap* [18]). In these simpler programs mappings are generally simple, one-to-one routings and are modified using conventional widgets.

The *reacTable* is a tabletop system that uses a modular synthesis paradigm to generate and control sound output [6]. As such, it does represent the routing of signals to and from functional blocks using a patching metaphor and therefore includes mapping functionality. However, it is a commercial system and more closely resembles an instrument in its own right than a mapping tool.

Some combinations of the above tools could conceivably achieve most tasks desired in any mapping exercise. However, it was felt that a single standalone application that addressed key functionality would be advantageous for novices experimenting with digital musical instruments and/or for simpler interactive projects for audio-visual applications. In addition, it was noted that almost all existing implementations used mouse-based control for the specification and administration of mappings and so did not readily support the 'playing' of mappings suggested in section 2.1.

4. SOFTWARE PROTOTYPE

4.1 Design Goals

The design goals which emerged from the initial motivations for the project and subsequent consideration of similar work were as follows:

- A minimal interface design, adhering to the design adage 'Keep It Simple, Stupid'.
- Ability to dynamically change complex mappings and adjust weights of combined mappings.

- Connection of external devices/ multiple communications protocols and messaging formats.
- Use as a performance instrument as well as an experimental setup, through the provision of a multi-touch-capable interface.
- Freely available, open source, cross-platform, standalone application with a low computational footprint.

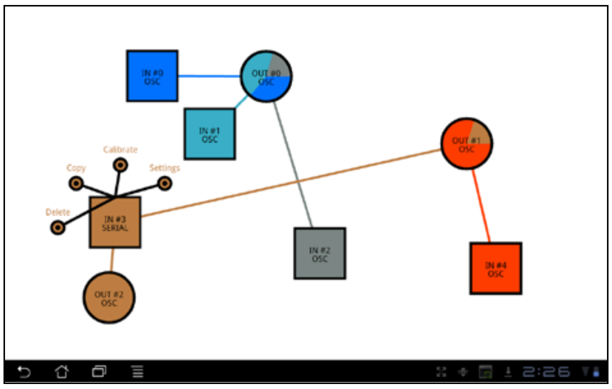

Figure 1. The prototype CrossMapper interface running on an Android device.

4.2 Implementation

A prototype software program named *CrossMapper* implementing the mapping functionality has been developed [21]. Versions of the application run on OSX, Windows and Android. A screen shot of the Android edition of the interface running on an Asus Eee Pad tablet is provided in figure 1. The functions currently implemented are outlined below:

- Creation and removal of input- and output-nodes, making and breaking of mapping connections.
- Dynamic weighted summing of inputs at outputs, controllable via the relative positions of nodes.
- Intuitive visualisation of weighting coefficients.
- Input from OSC or Serial Bus, output to OSC.
- Calibration of inputs with auto-ranging to floating point representation in range [0.0, 1.0].
- Mouse-based interface control for desktop use with multi-touch mode for Android devices.
- Settings dialog for specification of OSC address configuration (IP address and port).

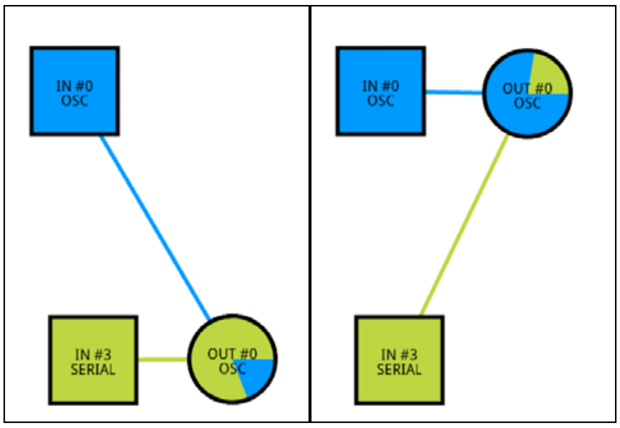

Figure 2. Visualizations of input weightings.

The main functionality of the interface is currently provided through the use of square input nodes and circular output nodes. These may be patched together by tap-dragging (on the Android version) from an input to an output. Existing connections may be broken by tap-dragging from an output to an input.

Multiple nodes may simultaneously be freely and independently moved about the interface space. The weighting of an input node may be modified through its position relative to its connected outputs. As shown in figure 2, the relative weighting is inversely proportional to distance between nodes and is represented using the color of the connected inputs (disconnected nodes appear white). In order to preserve the ability to independently manipulate the effect of a single parameter, an input may exist in several copies.

Node-specific settings appear from a pop-up menu when a node is double-tapped, as shown in figure 3. A video demonstration of the software is available online at the address specified in Appendix A [21].

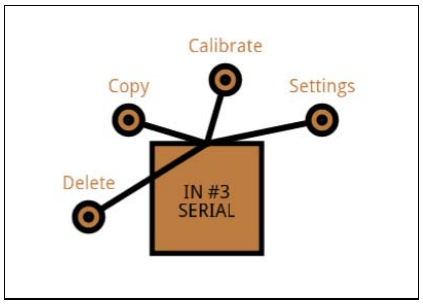

Figure 3. Pop-out of input node options.

5. EVALUATION

The prototype version has just recently been completed and so evaluation is minimal at this stage. Subjective comments on the system have been positive, particularly on the ability to manipulate mappings spatially. The visualization of input weightings has also drawn praise as it is easy to see at-a-glance how inputs affect outputs.

5.1 Computational Performance

The application runs 16 nodes with < 10% processor utilization on an average laptop computer running Windows 7 (Dell Precision 2400, Dual CPU @ 2.67 GHz). This is a satisfactorily 'lightweight' draw on computer system resources for the prototype version, but it is expected that the efficiency of the system will increase with code revisions and optimisation. On Android (Asus Eee Pad Transformer), the GUI runs 12 nodes without drop-off in rendering performance from the designed frame-rate of 30 Hz.

5.2 Discussion

This software is not intended to replace the sophisticated systems discussed in section 3. The flexibility and depth of functionality encapsulated in those examples offers the computer music researcher powerful tools for the investigation of advanced mapping schemes and large parameter spaces. It is felt that the simplified layout offered by CrossMapper will be useful for artists seeking to experiment with mappings in their projects, as a pedagogical tool and as a performance interface. In particular, the Android version seems well-suited to the live control of DAFx, where output parameters are not as numerous as in the case of some synthesis techniques [10]. The clear presentation on a mobile tablet computer has benefits for some performance environments, including low light and when positioned on-stage. Perhaps more profoundly, it is hoped that the interface design will lead to a broader exploration of mapping-as-music. At present, the interface offers more expressive opportunity than mouse-based input through the direct manipulation capability of the multi-touch version.

6. CONCLUSION AND FUTURE WORK

This paper presented the motivations for the development of a new mapping interface and its current prototype implementation. As this is a first iteration of the software, the immediate concerns are with beta-testing and debugging. Features to be implemented as soon as possible include:

- Save and recall functions for system settings and configuration presets.
- All node-specific settings including individual namespace specification.
- Ability to zoom and scale the workspace to make effective use of screen 'real estate' and allow use of an increased number of nodes.
- Ability to use other communications protocols e.g. input from Tangible User Input-Output, and devices using MIDI, Android Accessory Protocol, Human Interface Devices (HID) etc.
- Firmware for standardized Arduino connectivity, similar to Firmata [13] and for Android Accessory Mode devices (e.g. IOIO).

At the suggestion of several users, a Max 5 object encapsulating the CrossMapper interface is being developed, to be followed by an object for Pure Data [26].

A number of workshops are taking place to trial the software and provide an evaluation of its use. It is envisaged that the stable version will be incorporated into teaching practice on the Music & Media Technologies post-graduate course at Trinity College Dublin [19]. This will help to inform further development of the system as a pedagogical tool and as a performance interface.

It is not an immediate goal of the project to provide advanced signal processing capability, such as gesture recognition, as these needs are already being addressed by some of the more sophisticated systems discussed in section 3. However, it will be necessary to include some additional techniques for signal conditioning (e.g. smoothing filters) and combination of inputs (e.g. biasing, multiplication). These will be considered only if the simplicity of the interface presentation can be maintained and are currently being developed as options in the node pop-out menu.

The goal of the project is the provision of an intuitive interface with an easily understood modus operandi, yet with the potential for expressivity. At present, the simplicity of the interface and the ability to manipulate using multi-touch interaction achieves this to some degree.

7. ACKNOWLEDGMENTS

Many thanks go to the beta testers involved in the early versions of the software and to the computer music researchers who have provided useful critique and encouragement. The lead author's PhD work is supported by the Irish Research Council for Science, Engineering and Technology.

8. REFERENCES

[1] Arfib, D., Couturier, J. M., Kessous, L., and Verfaille, V. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound 7, 2 (*2002), 127–144.

[2] Bencina, R. The Metasurface – Applying Natural Neighbour Interpolation to Two-to-Many Mapping. *Proceedings of the 2005 Conference on New Interfaces for Musical Expression (NIME'05)* (Vancouver, BC, Canada, May 26-28, 2005), 101-104.

[3] Doornbusch, P., Composers' Views on Mapping in Algorithmic Composition. *Organised Sound 7, 2 (*2002) 145–56.

[4] Drummond, J., Understanding Interactive Systems. *Organised Sound 14, 2* (2009), 124–133.

[5] Hunt, A. and Kirk, R., Mapping Strategies for Musical Performance. *Trends in Gestural Control of Music*, M. Wanderley and M. Battier, Editors, 2000.

[6] Jorda, S., Geiger, G. Alonso, M., and Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. *Proceedings of the first international conference on Tangible and Embedded Interaction (TEI07)* Baton Rouge, Louisiana. 15-17 February 2007.

[7] Momeni, A., Wessel, D. Characterizing and controlling musical material intuitively with geometric models. *Proceedings of the 2003 conference on New Interfaces for Musical Expression (NIME '03)* (National University of Singapore, Singapore, Singapore, 2003), 54-62.

[8] O'Sullivan, L. and Boland, F. Tailoring Tabletop Interfaces for Musical Control. *ACM International Conference on Interactive Tabletops and Surfaces*, (Saarbrücken, Germany, November 7–10, 2010). Available at: http://www.mee.tcd.ie/~lmosulli/pubs.html.

[9] O'Sullivan, L. and Boland, F. Visualizing and Controlling Sound with Graphical Interfaces. *Audio Engineering Society 41st Conference: Audio for Games* (London, UK, 2-4 February 2011). Available at: http://www.mee.tcd.ie/~lmosulli/pubs.html.

[10] Verfaille, V., Wanderley, M., and Depalle, P. Mapping Strategies for Gestural and Adaptive Control of Digital Audio Effects. *Journal of New Music Research*, 35(1):71-93, 2006.

[11] Wang, J., D'Alessandro, N., and Pritchard, B. SQUEEZY: Extending a Multi-touch Screen with Force Sensing Objects for Controlling Articulatory Synthesis. *Proceedings of the 2011 Conference on New Interfaces for Musical Expression (NIME'11)* (Oslo, Norway, May 30-June 1, 2011), 531-532.

9. APPENDIX A: LIST OF URLS

[12] Bencina, R., AudioMulch interactive music studio. http://www.audiomulch.com/.

[13] Firmata. http://firmata.org/wiki/Main_Page

[14] FTM & Co., IRCAM. http://ftm.ircam.fr/

[15] Junxion, Steim.org. http://www.steim.org/steim/junxion_v4.html

[16] libmapper, IDMIL. http://www.idmil.org/software/libmapper

[17] Max/MSP, Cycling 74. http://cycling74.com/products/max/

[18] MouseTrap. www.humatic.de/webapps/htm/app/doc/MouseTrap.htm

[19] Music and Media Technologies, Trinity College Dublin. http://www.mee.tcd.ie/mmt/

[20] Moco. http://addi.tv/moco/

[21] O'Sullivan, L., CrossMapper project information. http://www.mee.tcd.ie/~lmosulli/

[22] Open Sound Control. http://www.opensoundcontrol.org

[23] Osculator, Wildora. http://www.osculator.net/

[24] OSC VST Bridge. http://oscvstbridge.sourceforge.net/

[25] Processing Development Environment. http://www.processing.org

[26] Pure Data: http://puredata.info/

[27] TouchOSC. http://hexler.net/software/touchosc