Ergonomics of Touch-screen Interfaces

Vincent Goudard

ICLI
PORTO
2018

liveinterfaces.org

Ergonomics of Touch-screen Interfaces

The MP.TUI Library for Max

Vincent Goudard
goudard@lam.jussieu.fr

Sorbonne Université, Collegium Musicæ,
Paris, France

Abstract

The design of digital musical instruments, freed from the physical constraints of acoustics, is essentially driven by issues in ergonomics and representation related to the musical context. Moreover, the programmability of virtual instruments allows dynamic reconfigurations of mapping relationships between gestural interfaces and synthesis. In this respect, graphical interfaces stand on the edge between representation and control. Recently enhanced by the advent of multitouch, they allow all kind of tangible interactions. Their customization (behaviour, shape, colour, etc.) plays a crucial role, whether for the virtuosity of professional musicians, for the accessibility of people with disabilities or for particular contexts such as collective interaction on the same touch-screen. I will first raise a few aspects of visual ergonomics that inspired this research then present recent developments of dynamic, polyphonic and customizable touch-screen interfaces, based on the concept of “dynamic intermediate model” and an ad-hoc protocol for expressive control.

Keywords
HCI
Visual interface
Multi-touch
Polyphony
Max

1. Visual aspects of instrument design

Instrument design encompass several aspects which are subject to ergonomics and that affect its visual appearance. I will run through a few of these aspects, taking examples from the acoustic instruments, as a retrospective on what led us to the developments we are carrying out with digital musical instruments (DMI).

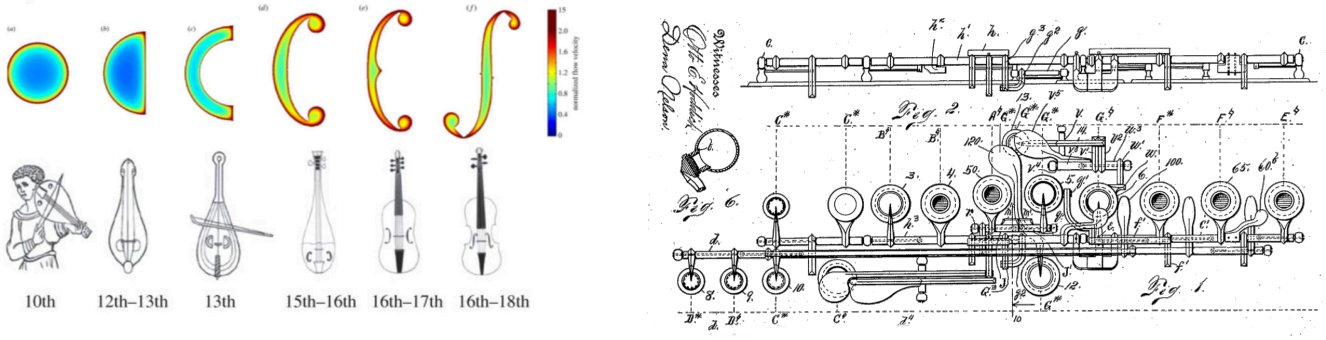

Figure 1. ***Left:*** *The evolution of air resonance power efficiency in the violin and its ancestors (Nia, Jain, Liu, et al, 2015).* ***Right:*** *Extract from patent by J. Djalma on "Improvements to key system Boehm flute", 1908.*

Adapting to sound

Instrument design is concerned with the quality of sound. While this is most obvious in acoustic instruments whose shape has direct consequences on the sound output (as exemplified in figure 1, left) the peculiar shapes that came out of traditional luthery also gave rise to a number of iconic elements (e.g. f-holes) and form factors (e.g. bigger size yields lower pitch) associated with the idea of an instrument. Moreover, DMIs may embed acoustic transducers, such as piezo microphones or tactile speakers, that influence the acoustic design of their hardware parts.

Adapting to the body

The instrument also adapts to the body. An interesting example is the evolution of the traverso to the western concert flute, with the help of the Boehm system in the years 1840 (cf. Figure 1, right). This system of keywork decouples the gesture topology from the airflow and resonance topology. By using shafts and finger plates, it enabled to enhance sound by making larger holes and placing them at adequate locations for the resonance, while the keys could be placed at convenient locations for the flutist's hands.

The Boehm system can be called an "intermediate model" between the gesture and the sound production, made of a mechanical system in this case. Most instruments combine various such "intermediate models" to amplify, enrich, displace, focus, multiply performers gestures and generate movements outside the scope of the human body's possibilities : bass drum pedals, piano hammers and dampers, bows and plectra, etc.

Adapting to music theory

Music instruments also embed elements of music theory. For instance, the upper part of a keyboard (black and white keys) represents the chromatic scale, while the lower part (white keys only) represents the diatonic C-major scale. The sizing and positioning of these keys is an interesting tradeoff between the mechanical constraints of the hammer system and a

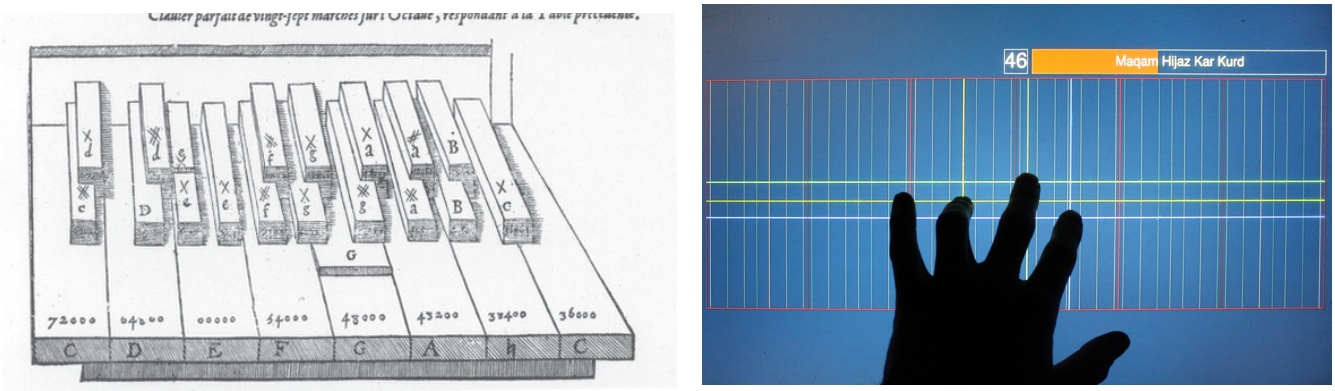

Figure 2. ***Left:*** *Twenty-seven-steps keyboard invented by Mersenne (1636)* **Right:** *A pitch-space with a micro-tonal scale representation made with mp.TUI. Brightness of the bars representing pitch quanta will fade in/out depending on the amount of quantization.*

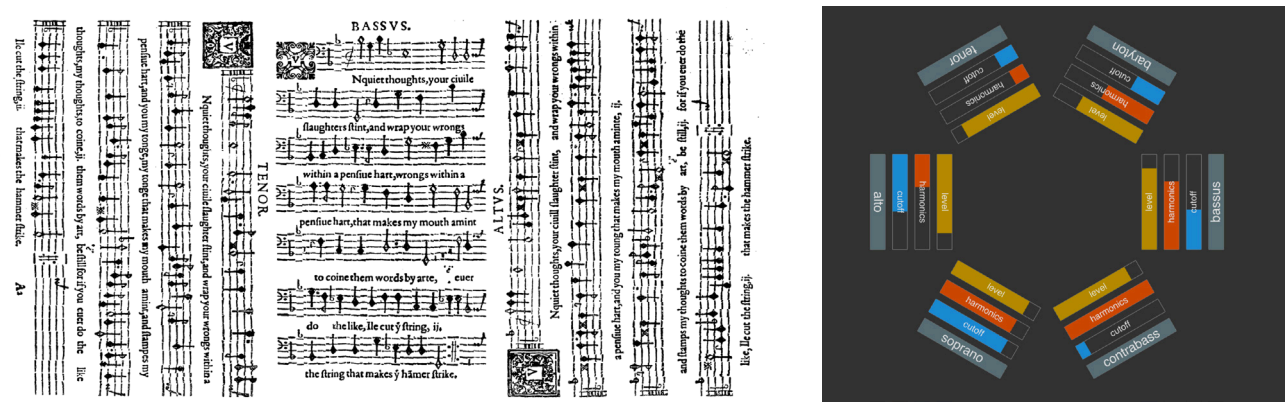

*Figure 3.****Left:*** *The First Book of Songes (Dowland, John), Edition : London: Peter Short, 1597. Source : IMLSP* ***Right:*** *Simple sliders GUI for 6 players located around a common interface.*

uniform representation of both the diatonic and chromatic scales. Moreover the octave width is such that it fits under a stretched hand, allowing to play any interval within an octave with a single hand, somehow reflecting octaves equivalence. Keyboards have been subject to many experimentations with micro-tonal pitch systems, intonations and note layouts, using hexagonal grids or several layers of keys (figure 2). As a symbolic system, such music theory can be easily encoded in computers. Music production softwares contain so many functions and rules based on music theory that these hardly fit the interface. Thor Magnusson talks of "epistemic tools" to describe the DMI, stating that it is designed with "such a high degree of symbolic pertinence that it becomes a system of knowledge and thinking in its own terms" (Magnusson, 2009). As such, this "system of knowledge" is an imaginary landscape to be explored, a sonic territory for which the instrument's interface and mapping can metaphorically stand as a map.

Adapting to the context

If we stretch a little bit the notion of musical instrument to simply consider them as tools to make music, then scores, concert halls, audience and more generally, the performance context also take part and influence instrument design. Oriented scores (figure 3, left) is an example of adapting the score to the context of

"table music", in this case enabling the musicians to read the score while they are sitting around a table. Similarly, screen-based DMIs can adapt their layout to the number of performers by presenting each of them a group of UI elements oriented toward them (figure 3, right).

Adapting to the experiment

Eventually, the process of designing music instruments contains a great deal of empirical work. The process of adjusting the settings of an instrument will often require direct feedback for hand-made fine tunings, until it sounds and feels good.

2. Graphical User Interfaces

Until the last decade, GUI were mostly operated with the mouse on square monitor screens. Software interfaces design was thus oriented toward offline processing, with sequential actions controlled by a small set of GUI components types which have become a de facto standard : buttons, knobs, sliders, and menus. To attain the fast parallel control that live music requires, performers would often use external devices such as MIDI interfaces.

On-screen parallel control came with multitouch interfaces. Although first developed in the 1980s, notably by Lee, Buxton and Smith (1985), they came to a broader audience only after the turn of the century with works by Ishii and Ullmer (1997) or Paradiso (2002). Cheaper multitouch technologies (Han, 2005) and augmented reality techniques (Dietz and Leigh, 2001; Patten, Recht and Ishii, 2002; Costanza, Shelley and Robinson, 2003) contributed to spread interest for such systems, which eventually led to commercial products in the music market such as Jazzmutant's Lemur, or the reacTable* (Jordà, Kaltenbrunner, Geiger and Bencina, 2005). Following this, tablets apps like TouchOSC, Control or later Mira[1] enabled end-user to compose their own GUI layouts on multitouch devices. However, the components remained mostly tied to a vertical/horizontal scheme not particularly suited to forearm- or wrist-centered movements or any freer layout.[2]

Apart from the now-usual gestures such as swipe or pinch-zoom, multi-touch interfaces gave rise to a number of strategies to interpret touch data: the same gesture will yield a different response if performed with a single or multiple touch, or depending on the order in which fingers touch the screen. And, as for the temporal interactions specific to music performance, multi-touch screens allows for timed gestural combinations in a way the mouse could not offer.

3. The mp.TUI library

The mp.TUI library[3] was born out of two previous works which will be presented briefly: the concept of "Dynamic Intermediate Model" and the "Modular Polyphony (MP)" framework.

Dynamic intermediate models

The idea of DIM (Goudard, Genevois, Ghomi and Doval, 2011) was an attempt at transposing the concept of intermediate models such as those found in the traditional instruments into the digital world. It was also an attempt to find a better term to qualify such systems which translate and transform gestures than the widespread name of "mapping". This term lets the reader think of simple connections between a controller and a synth and does not reflect the real interaction design at work. With the help of computers, intermediate models become dynamic, both in the sense that the system can provide energy, but also in the sense that it can change on the fly and evolve during the performance itself.[4]

This study was concerned with connecting, arranging and composing such models into compounds and interactive scenarios. One of the issues was to find an efficient protocol to communicate between the models, that takes into account their asynchronous, polyphonic, heterogeneous and ephemeral nature. That is what led to the definition of the MP-framework.

1 TouchOSC by Hexler : https://hexler.net/software/touchosc, Control by Charlie Roberts : http://charlie- roberts.com/Control/, Mira by Cycling'74: https://cycling74.com/products/mira

2 A notable exception is the reacTable* (Jordà, 2005) whose design is mostly driven by circular components allow- ing collective use around the device.

3 Sources available at https://github.com/LAM-IJLRA/ModularPolyphony-TUI/.

4 A video presenting Dynamic Intermediate Models is available here : https://vimeo.com/25740547/

The MP framework

The MP framework (Goudard and Genevois, 2017) was born out of the need for polyphonic expressive control in a modular digital luthe-rie environment such as Max. It allows to easily process incoming multi-touch data (TUIO and mouse) with usual Max objects, wrapped in poly~ object. The MP framework is made of modules (*"mp-blocks"*) that process asynchro-nous events (*"mp-events"*). An mp-event is an abstract temporal object, somewhat similar to a MIDI note, that can travel several processing paths in parallel and be merged or associated with other *mp-events.* Each *mp-block* can pro-cess several *mp-events* in parallel.

An *mp-event* is defined by a set of *mp-messa-ges.* These messages are made of control para-meters tagged with a unique value identifying the *mp-event*. The message format is minimali-stic : a unique identifier, a parameter name fol-lowed by a list of values. Two parameter names are reserved. The "state" parameter, which can be set either to *on*, *off* or *update*, defines the way incoming mp-messages are to be interpreted. The "guest" parameter can be used to create relations between events (e.g. parent/child) and combine them in a single process. This protocol allows to design full mapping paths from poly-phonic controllers such as MPE or multi-touch interfaces down to "expressive"[5] polyphonic synthesis, passing through several stages of control-transformation.

Overview of the mp.TUI library

The mp.TUI library is built on top of the MP pro-tocol. Its provides a framework based on Max's patching logics to create new multitouch UI components in an OpenGL context and over-come some limitations found in GUI available in the patching environment. For instance, GUI are usually oriented on a horizontal/vertical layout with a top-down reading orientation while one may like to have several orientations, like in the situation presented on figure 3. The layering of various components may require custom col-ours and transparencies, and one may want to include more complex visual interfaces than sliders and knobs, e.g. particles, video, 3d models, shaders (figure 5), etc.

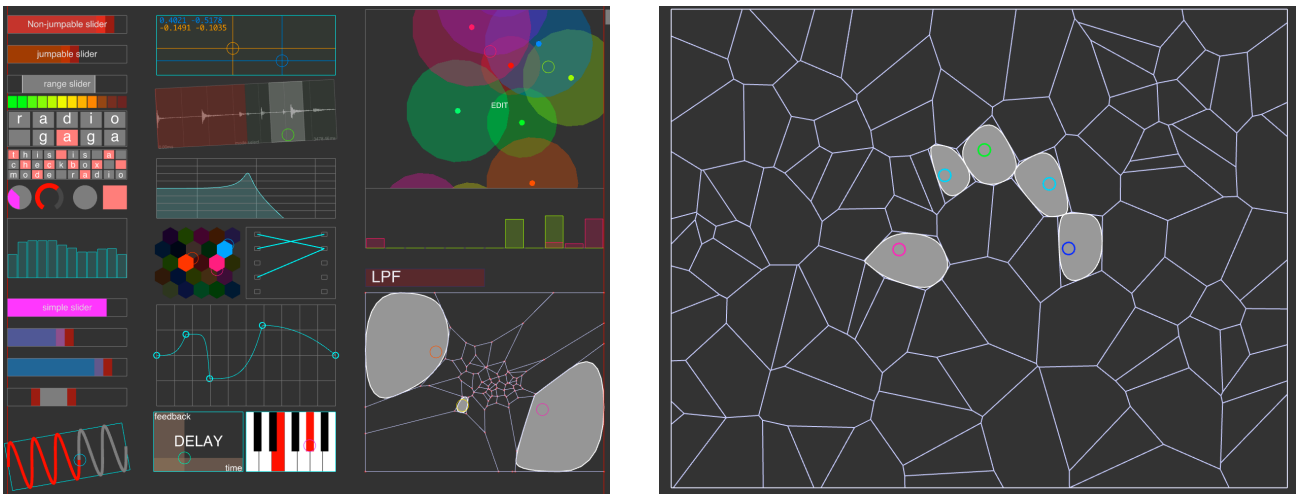

Figure 4. ***Left:*** *overview of some mp.TUI components.* ***Right:*** *5-fingers touching a dynamic Voronoï model.*

5 "Expressive" meaning here that it allows the polyphonic modulation of previously triggered sound-events.

The possibility to design audiovisual objects in Max with a tight relation between gesture, audio and visuals allows to integrate them into custom dynamic scenarios: narrative stories for educational workshops with kids, reactive screen-scores, custom visualizations for visually impaired people, museum exhibitions with specific graphic charters, reactive adaptation to screen formats, experimental graphics for the aesthetic of live artistic performances, etc.

The underlying MP-protocol allows to activate GUI components on the fly in a way similar to the triggering of polyphonic notes. An example is shown on figure 4 (left) where the top-right "node" object dynamically instantiate multi-sliders objects below matching each active cursor on the node-object (two of them in this case).

The components of the library are of a set of abstractions of three types:

1. **System components**, which implement the essential functions to wrap graphics into a pickable element. This includes the "mp.TUI.hub" which retrieves the data from the mouse interaction on the OpenGL window as well as TUIO messages received by UDP and send them to the picked GUI components.

2. **GUI components**, which are ready-made instances of common and not-so-common widgets such as sliders, keyboards, breakpoint functions, etc.

3. **Utilities**, a set of abstractions which make it possible to easily create new components by proposing useful functions for interaction design (viewing transforms, management of the polyphony on an element, pinch-zoom, computing deltas etc.)

The components make use of hierarchical geometry transform,[6] which allows to get world-related or object-related coordinates regardless of the UI component's position, scale and orientation. This also allows to create groups of components, like one would do in any CAD software. Following the empirical nature of digital lutherie claimed above, an "edition mode" is also available to quickly manipulate UI components' position, scale and orientation by hand (figure 6).

Performances

The mp.TUI library is fully developed with "vanilla" objects of Max' distribution. This approach, although more CPU-expensive than compiled objects has the advantage of letting any Max user hack the components and adapt them to their needs. Besides, mp.TUI components are essentially relying on OpenGL so that most of the computation load is left to the GPU. UI picking is made with the help of the Bullet-Physic[7] engine embedded in Max. While this may be more costly for some simple shapes, it allows us to design UI component of any shape and orientation, like bent sliders or hollow shapes, and to potentially animate them, like in the "bouncing balls" example where several 2D cursors can be moved around and launched in the bounding box.

As a library built on top of Max, mp.TUI is not the most optimized GUI system one can think of, especially in term of memory usage. Instantiating 50 sliders in Max take no time and has hardly any memory imprint while doing the same with mp.TUI will use some 250Mb and require several seconds on a recent laptop.

As far as its usage is concerned, since most parts of the mp.TUI components are running in an OpenGL context, the components responsiveness is tied to the OpenGL context scheduler, typically ranging between 20 and 60 FPS, yielding a latency from 15 to 40ms (in addition to the interface's own latency). As experienced in several use-cases, the touch-to-display latency (Ng, 2012) was reasonable enough for polyphonic modulations in a live musical context, offering a responsiveness almost similar to existing apps like Mira or TouchOSC.

6 With the jit.anim.node object in Max.

7 http://bulletphysics.org/

Figure 5. "The phonetogram", an app whose UI was designed with mp.TUI, showing a layered UI using shaders to blur the background.

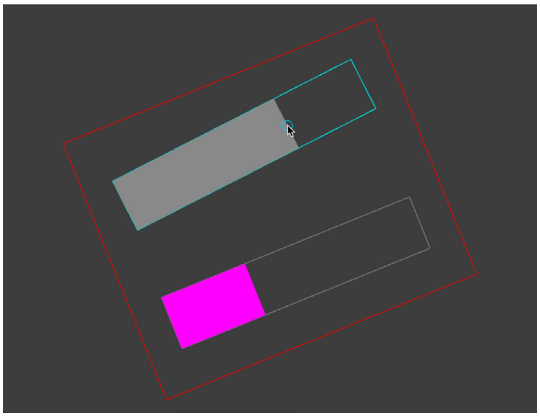

Figure 6. grouping objects and manipulation "by hand".

4. Perspectives

The mp.TUI library aims at easing the design and development of original interactive GUI with strong interaction with sound. Components can be tailored for specific needs such as those encountered in the domain of digital lutheries with high-level Max patches, allowing non-expert programmers to build their own by reusing and modifying existing components. Components of the mp.TUI library can be mixed with and/or use advanced graphics, allowing for lively and aesthetic representation and control. Although the library is more memory- and CPU-costly than native Max GUI, it allows for experimenting with visual interaction design in a high-level visual programming environment. Much can be developed in this area open to creativity and it is hoped that this open library will help interface designer to come up with new exciting ways of representing and interacting with live music and sound.

Acknowledgements. This work is part of an ongoing doctoral research funded by the Collegium Musicæ, Sorbonne-Université.

Costanza, E, Shelley, S. B., and Robinson, J.. 2003. Introducing Audio d-touch: A Tangible User Interface for Music Composition and Performance. In Proceedings of the 2003 International Conference on Digital Audio Effects, London, UK.

Davidson, P. l. and Han, J. 2006. Synthesis and control on large scale multi-touch sensing displays. In : *Proceedings of the 2006 conference on New interfaces for musical expression.* IRCAM—Centre Pompidou, 2006. p. 216-219.

Dietz, P. and Leigh, D. 2001. DiamondTouch: a Multi- User Touch Technology. In Proceedings of the 14th Annual ACM Symposium on User interface Software and Technology (Orlando, Florida, November 11 - 14, 2001). UIST '01. ACM Press, New York, NY, 219-226.

Goudard, V. and Genevois, H. 2017. "Mapping modulaire de processus polyphoniques", Proceedings of the Journées d'Informatique Musicale. Paris, France.

Goudard, V., Genevois, H., Ghomi, E. and Doval, B. 2011. "Dynamic Intermediate Models for audiographic synthesis". In : *8th Sound and Music Computing Conference*. Padova University Press, 2011. p. 486.

Han, J. Y. 2005. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology (Seattle, WA, USA, October 23 - 26, 2005). UIST '05. ACM Press, New York, NY, 115-118.

Ishii, H. and Ullmer, B. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Atlanta, Georgia, United States, March 22 - 27, 1997). S. Pemberton, Ed. CHI '97. ACM Press, New York, NY, 234-241.

Jordà, S. & Kaltenbrunner, M. & Geiger, G. & Bencina, R. 2005. The reacTable*. In Proceedings of the International Computer Music Conference (ICMC2005), Barcelona (Spain)

Kaltenbrunner, M. & Geiger, G. & Jordà, S. 2004. Dynamic Patches for Live Musical Performance. In Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04), Hamamatsu, Japan

Lee, S. K., Buxton, W. and Smith, K. C. 1985. A Multi- Touch Three Dimensional Touch-Sensitive Tablet. In Proceedings of CHI '85 (April 1985), ACM/SIGCHI, NY, 1985, pp. 21–25.

Magnusson, T. 2009. Of epistemic tools: Musical instruments as cognitive extensions. *Organised Sound,* 2009, vol. 14, no 2, p. 168-176.

Nia, Hadi T., Jain, Ankita D., Liu, Yuming, et al. 2015. Air resonance power efficiency evolved in the violin and its ancestors. *The Journal of the Acoustical Society of America*, vol. 138, no 3, p. 1911-1911.

Ng, A., Lepinski, J., Wigdor, D., et al. 2012. Designing for low-latency direct-touch input. In : *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 2012. p. 453-464.

Paradiso, J. A. 2002. "Several Sensor Approaches that Retrofit Large Surfaces for Interactivity". Presented at the UbiComp 2002 Workshop on Collaboration with Interactive Walls and Tables, Gothenburg, Sweden.

Patten, J., Recht, B and Ishii, H. 2002. "Audiopad: a tag-based interface for musical performance". In : *Proceedings of the 2002 conference on New interfaces for musical expression.* National University of Singapore, p. 1-6.

Salazar, S. and Wang, G. 2014. "Auraglyph: handwritten computer music composition and design," in Proceedings of the international conference on new interfaces for musical expression, London, United Kingdom, pp. 106-109.