Sputnik

Project Report HCC Project Seminar 2011

Simon Wallner*

13th December 2011

This paper evaluates *Sputnik* a 3D environment with which the user can freely interact through an elastic *arc of light/fishing rod* metaphor, to explore, create and interact with virtual *sound objects*. These sound objects are placed in the scene and react to the user's input by sending MIDI commands to an external audio program thus creating or manipulating the sound.

1 Reading Guide

Appendices Code Prerequisitions Video CD etc...

2 Introduction

Computer music is around us for some time now and through the use of the computer musicians have sheer endless possibilities of musical expression. With this plethora of possibilities comes the need for constraints and control to harness this expressive potential. Over the recent years many standard and non-standard interface have been developed, ranging from the ordinary button-fader-nob MIDI interface to more elaborate interfaces and systems like the *reactable*[Jordà et al., 2007], *mixiTUI*[Pedersen and Hornbæ k, 2009] or commercial solutions like the *Novation Launchpad*[1] or *Native Instruments Maschine*[2] to name but a few.

With the advent of motion based controllers in consumer entertainment systems, marked by the release of the *Wii*[3] console in late 2006, motion controllers became

*me@simonwallner.at

[1] http://www.novationmusic.com/products/midi_controllers/launchpad

[2] http://www.native-instruments.com/#/en/products/producer/maschine/

[3] http://de.wikipedia.org/wiki/Wii

widely and cheaply available. This and their interface capabilities make them the ideal tools to explore the realm of *new interfaces for musical expression.*

A common problem of computer music interfaces is that often the process of sound creation is not readily comprehensible. Seeing a performer on stage behind their laptop twisting knobs and adjusting faders might be ambiguous to an uninformed observer. It can be hard to relate the artist's action to the resulting sounds. This can hinder the experience and might go as far as to the point where the audience suspects that an artist just pressed play, as interviews conducted by [Pedersen and Hornbæ k, 2009] show.

This paper introduces *Sputnik*, a system that uses a *Wiimote* controller to interact with a dynamic 3D scene. In the scene, a variety of sound creating objects are placed that send MIDI signals to an external audio program upon the user's interaction.

Users can freely navigate the 3D scene and interact with it through an elastic *arc of light/fishing rod* metaphor. It seems as if the *arc of light* was coming out of the Wiimote and reaches into the scene, acting as an extension of the user's body into the virtual space. With this bodily extension users can *grab* and *drag* objects around the 3D scene.

In this paper I evaluate the qualities of the *arc or light* metaphor and how the design decisions/constraints of the system influence its expressive potential both visually and musically. This evaluation is grounded in a user study of XXX users.

Based on these findings and the theoretical framework of [Ullmer and Ishii, 2000] the similarities and differences between *Sputnik* and tangible user interfaces are discussed.

3 Paper Outline

The following section gives an overview over related work in the field of *New Interfaces for Musical Expression* and tangible user interfaces. Section **??** goes into detail about Sputnik, both on a conceptual and a technical level. Section 8 describes the performed user study and the paper is finally concluded in section 9 where the findings are discussed.

4 Related Work

Practical Work Only a few projects exist that go into a similar direction as Sputnik. The *Virtual Xylophone*[Mäki-Patola et al., 2005] is a virtual reality system in which the user can place xylophone bars of different pitch in the scene and then struck them with a virtual mallet. By translating the configuration and mapping of the real instrument into the VR environment, new modes of play emerge. [Zappi et al., 2010] created a virtual controller for *Ableton Live* that allows users to create simple proxy objects in a VR environment, bind them to certain controls and use them effectively as

virtual sliders. [Rodet et al., 2005] created a virtual environment for an exhibition setting. Users interact with the system via a 6-DOF motion tracker with tactile feedback. However, the user's actions in the system are highly constrained.

More projects can be found in the realm of Tangible User Interfaces. With *mixiTUI* [Pedersen and Hornbæ k, 2009] created a table top tangible interface for a sequencer that aimed not only to be functional but also to visually enrich the artists performance. Interviews with musicians and an extensive user study have been performed. [Jordà et al., 2007] created the famous *reacTable*, also a table top tangible interface that allows the creation and manipulation of music by composing various objects on its surface.

After the release of the Wii in late 2006, the Wiimote motion controller received some attention in and outside the field of musical interfaces: [Kiefer et al., 2008] assess the general qualities of the Wiimote as a musical controller and [Miller, 2010] uses the Wiimote and sensor bar to create the *Wiiolin*, a virtual violin that mimics the real instrument and can be played either in an upright position like a cello or horizontally like a violin. It senses the button presses and tracks the movement of the *bow*, i.e. the sensor bar to create the sounds.

Not a Wiimote but still impressive, [Miyama, 2010] uses an low resolution distance sensor array to control the many parameters of a synthesizer. A small gui application is merely used for monitoring the system's state, and sound creation is done in pd.

Theoretical Work The field of *Tangible User Interfaces (TUI)* provides part of the theoretical background for this work. Work of [Fitzmaurice et al., 1995] and then later [Ishii and Ullmer, 1997] introduced this term and the wider concept. [Shaer, 2009] Gives a very good overview over this field as well as the history of TUI studies. [Ullmer and Ishii, 2000] introduced *MCRpd*, a formal model for describing and analysing TUIs that will be used in section **??**.

[Sharlin et al., 2004] introduced *spacial TUIs* that focuses on *I/O unification* by tightly coupling the action and perception spaces and embodying a clear state representation across all sensory modalities.

Entering the musical realm [Fels and Lyons, 2011] give a good overview and general introduction into the field of *NIMEs (New Interfaces for Musical Expression)*. [Cook, 2001] shares 13 general principles for designing computer music controllers that resulted from his long lasting experience in this field. [Dobrian and Koppelman, 2006] asks the question of virtuosity and expression by pointing out the elephant in the room, e.g. the lack thereof and also the lack of a comparable standard repertoire.

In contrary to that [Gurevich and Treviño, 2007] question the hegemonic *composer–interpret–listener* relation in favour of a more holistic *ecological* view of musical expression. Later work by [Gurevich et al., 2010] evaluated a highly constrained, prototypical one-button instrument that spurred a wide variety of play styles in test users.

Closing the loop to design and HCI, [Magnusson, 2010] gives a good overview over

the field of *affordance* and elaborates on *contraints* from different viewing angles and how they impact and support creativity. Finally, [Wanderley and Orio, 2002] goes into depth over evaluating input devices for musical expression in the context of HCI.

5 Sputnik Overview

Sputnik is a *New Interface for Musical Expression* that combines 3D graphics with the capabilities of the Wiimote controller. The user is presented with colourful 3D scene that contains various simple objects. The user can freely navigate this scene and interact with these object to create sounds.

The user can interact with the system via a Wiimote and Nunchuck controller. The Nunchuck controls the users movement in the 3D space and the Wiimote is used to control the camera and to interact with the objects in the scene.

5.1 Setup

The system is set up in a room with an overhead mounted video beamer. The sensor bar can either be placed on the upper or lower edge of the projected image. It consists of two IR emitters that allows the IR camera in the Wiimote to track its relative orientation in space. A Wiimote and Nunchuck controller are used and only a single person at a time can use the system. Figure 1 illustrates the setup.

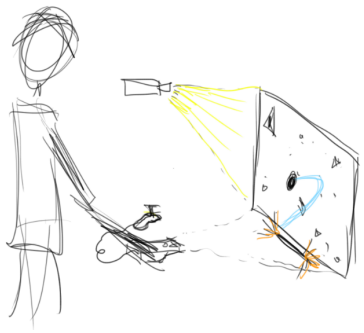

Figure 1: Standard set up of Sputnik

5.2 Navigation and Camera Controls

Sputnik's virtual scene uses a fixed *up direction* and the user can move through the scene by pushing the Nunchuck's analogue stick in the respective direction. Pushing

the stick forward moves the camera into the scene, pushing it left moves the camera to the left and vice versa.

Tilting and panning is controlled by pointing the Wiimote to the top/bottom/left/right of the screen. The farther it is pointed away from the neutral center position the faster the camera movement is. Figure 2 illustrates the navigation.

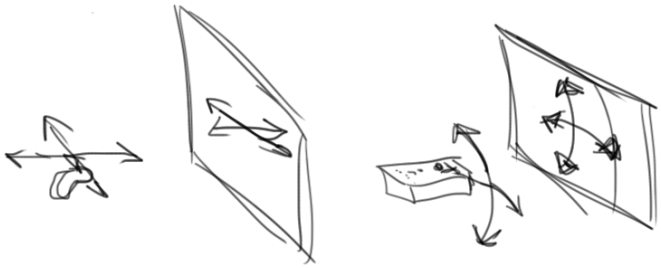

Figure 2: The Nunchuk's analogue stick controls the camera's dolly and track movements, the Wiimote's IR pointer controls the camera's tilt and pan movements.

5.3 Interaction

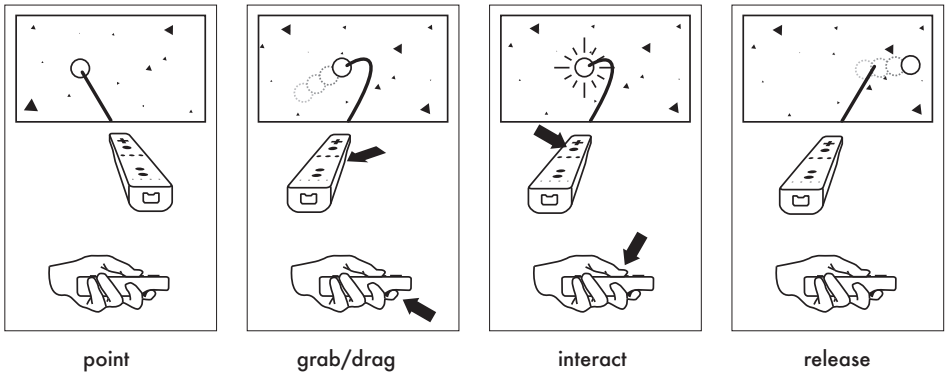

Figure 3: The basic interaction vocabulary of Sputnik

Figure 3 illustrates the basic interaction vocabulary of Sputnik. The user can interact with the scene via an *arc of light* metaphor. It seems as if the arc of light was coming out of the Wiimote and reaches into the scene, acting as a bodily extension of the user into the virtual space. Through this *arc* objects in the scene can be *pointed* at, *grabbed* and *dragged* around.

Objects behave in a simplified yet physically plausible way, each featuring distinct weight and friction. Dragging objects causes the arc to bend like a fishing rod, reflecting the physical properties of the object.

Each object in the scene can react individually to user interaction. The following classes of objects exist in Sputnik:

Sampler (red sphere) The *sampler* object reacts to the press of the *A Button* while it is grabbed. While the button is held the sample is played in a loop and stops immediately when the button or the object is released. Playback always starts at the beginning of the sample.

Player (yellow sphere) The *player* object reacts to the press of the *A Button* while it is grabbed. Pressing the *A button* starts and stops the player. It is not automatically stopped when it is released. Playback is looped and always starts at the beginning.

Tape Machine (grey sphere) Modelled after an old tape machine and inspired by *musique concrète* the *tape machine* object controls the speed of the play back by the object's movement speed in the virtual space. The faster it moves the faster the playback. Playback is looped.

Harmonic Harp (orange spheres) The orange spheres form a kind kind *harmonic harp*. Each sphere controls a single sine wave oscillator and the volume is determined by the the spheres distance to is origin. Additionally a force exists that drags the spheres towards their respective origins. The oscillators are tuned to the natural harmonic series.

By interacting with the different objects and arranging the in the virtual scene users can create musical performances that are both musically and visually expressive.

5.4 The Arc Of Light Metaphor

The *arc of light* is the core of Sputnik. It should be easy to pick up and understand even for novice users. It has the following properties:

If no object is grabbed, the *arc* follows the input of the user directly. The input is not filtered by the application and is used *as is*. This creates a slight jittering but on the other hand makes it very responsive. Low pass filtering of the Wiimote input was explicitly not used in order to achieve high responsiveness.

This filtering however is indirectly introduced through the physical properties of the interactive objects. Giving this filter process a physical representation should it ideally make it transparent to the user and achieve the desired effect (even very strong filtering) without making the system feel laggy or dull.

5.5 Simplified Physics

Sputnik uses simplified physics simulations to give the interactive objects their distinct physical properties. An objects properties are described by *mass*, a *movement vector*, as well as a movement *damping constant*. SI units are used unless otherwise noted.

The dragging force of the arc on an object is linear in the distance from the object to the intersection of the unbended arc with the plane that goes through the object with a plane normal pointing at the camera.

In each frame, the speed of an object is then computed as

$$v_i = v_{i-1} + \frac{F * \Delta T}{m} \tag{1}$$

where v_i is the object's movement vector in frame i, F is the current force, m is the mass and ΔT is the frame time. Force is only applied at the center of the objects, and thus never causes a rotation of the object.

Damping of objects movement is implemented by a simple exponential decay, given as

$$v_i = v_{i-1} * e^{-\lambda * \Delta T} \tag{2}$$

where λ is the damping constant.

Even with theses simple formulas a convincing and plausible physical system can be created.

5.6 Creating sound

Sputnik's interactive objects use the MIDI protocol to communicate with external applications. Thanks to the simplicity and the pervasiveness of this protocol, virtually every music software can be used together Sputnik.

In the current set up Sputnik *pure data (pd)*[4] is used to create sound. A simple patch is used that builds on *boctok-1*, a small collection of pd patches written by the author prior to this project.

6 Implementation

Sputnik is implemented in C++ with Mac OS X as its development platform. The project is split into two sub projects: The special purpose Sputnik and the more general purpose *kocmoc-core*. Kocmoc-core provides a host of core services that are then used by sputnik to build the final system. Some code of kocmoc-core existed prior to this project, but most of it was created during this project.

[4] `http://puredata.info/`

The source code of of both Sputnik and kocmoc-core is licensed under the MIT license and is available to the public on github[5] [6].

Sputnik uses OpenGL 2.1 with a few extensions to display the virtual scene. The renderer is fairly simple and displays unlit, textured objects with baked ambient occlusion maps. A simplified physically correct computation of the homogeneous fog is realised in the vertex and fragment shader and a post processing effect is applied. This effect adds a barrel distortion and vignetting as well as a simple form of full screen anti aliasing.

The *Assimp*[7] and *devIL*[8] libraries are used to load assets and font rendering is implemented using the *freetype*[9] library.

RtMidi[10] is used to send messages to external applications and *WiiC*[11] was used to interface with the Wiimote and Nunchuck controller.

Figure 4 illustrates the data flow in the main run-loop. First the devices are polled, which causes callbacks to be fired. The components are subsequently updated and finally the scene is sent to the renderer.

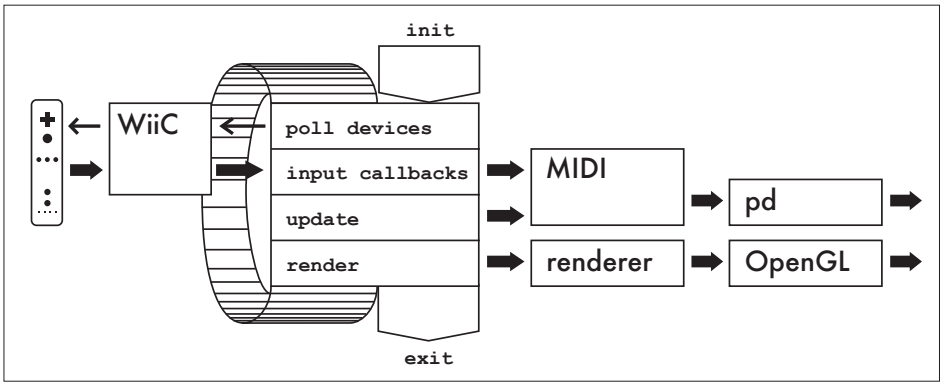

Figure 4: The data flow in the run-loop.

6.1 The Virtual Scene

Figure 5(a) shows an overview shot of sputnik. The scene consists of the following parts:

[5] https://github.com/SimonWallner/sputnik
[6] https://github.com/SimonWallner/kocmoc-core
[7] http://assimp.sourceforge.net/
[8] http://openil.sourceforge.net/
[9] http://www.freetype.org/
[10] http://www.music.mcgill.ca/ gary/rtmidi/
[11] http://wiic.sourceforge.net/

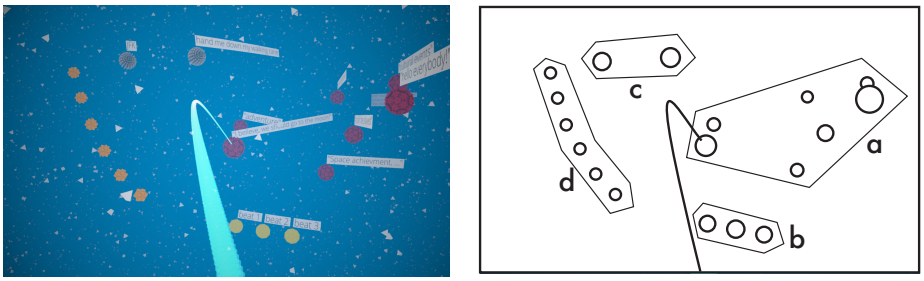

Figure 5: The objects in the scene: (a) Samplers, (b) Players, (c) Tape Machines, (d) Harmonic Harp

1. The light blue and bent *arc of light* starting in the middle of the lower edge and going *into* the picture
2. Musical objects the user can interact with: sampler (red), player (yellow), tape machine (grey), harmonic harp (orange)
3. Textual Labels on the musical objects. These help the performer as well as they convey meaning to the audience.
4. The star field. It is randomly generated and the user cannot interact with it. It serves an aesthetic purpose as well as it is an important orientation help. (see section **??**)
5. The coloured fog. The fog provides an important depth cue to the human perception.

The visual representation of Sputnik serves a dual purpose. It it the sole graphical interface to the performer and thus has to display all relevant information that is needed for the performance but at the same time has to be visually pleasing to the audience. Common music software only focuses on the first, leaving the visual performance in most cases to a dedicated VJ. Sputnik tries to bridge those two areas by using the same interface for both the performer as well as the audience.

The textual labels play an important role in Sputnik. On the one hand they provide important information to the performer and on the other hand they act as an additional communication channel to the audience. The audience can read the labels and better understand what is happening and how the musical performance is built up from smaller building blocks. It can also introduce an element of anticipation when an object with a certain label is visible but the performer does not yet interact with it.

(a) (b)

Figure 6: (a) a group of *samplers*, (b) a close-up of a sampler with its textual label

7 Wiimote Input

The Wiimote controller was chosen for its pointing functionality as well as the motion sensing capabilities, even though they haven't been used in the final project. The accompanying Nunchuck controller was also used to allow the standard analogue stick driven navigation.

The Wiimote has an built in low resolution IR camera and its feed is directly processed on chip. The feed itself is not accessible but data of up to 4 tracked IR points can be read. the Wiimote's update rate is reported to be $100Hz$ which gives an lower bound for the worst case lag of $10ms$. The used Wiimote library takes care of most of the processing and and conveniently returns the computed pointer location in relative screen coordinates.

A drastic constraint of the Wiimote's IR pointer is the relatively narrow field of view. It is very easy to leave the small sensing area resulting in no data. During the development it turned out that using the Wiimote for the tilt/pan movements of the camera can alleviate this problem a little. Users seem to intuitively try to compensate the camera rotation thus better maintaining a focus on the neutral area in the center of the screen. Without it users easily lost the focus an had troubles finding *back onto the screen*.

The used library supports more than one Wiimote and the input system in Sputnik would also support multiple input devices at the same time. The option to *dual wield* two Wiimotes and interact with two arcs of light simultaneously was given up in favour of the more standard analogue stick track/dolly controls to allow the users to easily navigate the scene.

7.1 Mapping the System

What further distinguishes Sputnik from other projects or interfaces is the spacial component. The user cannot only drag objects to create sounds but can also change the spacial configuration of the system by moving objects. All objects that are not bound to a fixed origin (i.e. the harmonic harp) can be freely moved around and be thus allowing the system to be reconfigured at runtime.

The tape machine and harmonic harp provide the most interesting examples for possible mappings of the system. Those two interactive objects use the physical properties of the individual objects and map them to sound parameters. This is something that is not possible with conventional hardware or software interfaces.

8 Evaluation

3-5 pages

Describe the evaluation according to the research questions. Describe the process and the observed results.

8.1 What to evaluate

8.2 Assumptions, expected outcome

8.3 describe the evaluation

8.4 give the results

9 Discussion

3-5 pages

Discuss the results form the evaluation and answer the research questions.

1. How can the arc of light/fishing rod metaphor be used for intuitive interaction. How does lag impact the system?
2. What meaningful mappings can be derived from the interaction with and the visualisation of the virtual scene.

10 Future Work

11 Conclusion

0.5 pages

12 Acknowledgements

thank Esben, thank the participants,

References

[Cook, 2001] Cook, P. (2001). Principles for designing computer music controllers. In *Proceedings of the 2001 conference on New interfaces for musical expression*, NIME '01, pages 1–4, Singapore, Singapore. National University of Singapore.

[Dobrian and Koppelman, 2006] Dobrian, C. and Koppelman, D. (2006). The E in NIME: musical expression with new computer interfaces. In *NIME*, pages 277–282.

[Fels and Lyons, 2011] Fels, S. and Lyons, M. (2011). Siggraph 2011 Course Notes Advances in New Interfaces for Musical Expression. *Notes*.

[Fitzmaurice et al., 1995] Fitzmaurice, G. W., Ishii, H., and Buxton, W. A. S. (1995). Bricks. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, pages 442–449, New York, New York, USA. ACM Press.

[Gurevich et al., 2010] Gurevich, M., Stapleton, P., and Marquez-Borbon, A. (2010). Style and Constraint in Electronic Musical Instruments. In *NIME*, number Nime, pages 106–111.

[Gurevich and Treviño, 2007] Gurevich, M. and Treviño, J. (2007). Expression and its discontents. In *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*, page 106, New York, New York, USA. ACM Press.

[Ishii and Ullmer, 1997] Ishii, H. and Ullmer, B. (1997). *Tangible bits*. ACM Press, New York, New York, USA.

[Jordà et al., 2007] Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. (2007). The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*, page 139, New York, New York, USA. ACM Press.

[Kiefer et al., 2008] Kiefer, C., Collins, N., and Fitzpatrick, G. (2008). Evaluating the

wiimote as a musical controller. *Proceedings of the International Computer Music Conference*, pages 17–17.

[Magnusson, 2010] Magnusson, T. (2010). Designing Constraints: Composing and Performing with Digital Musical Systems. *Computer Music Journal*, 34(4):62–73.

[Mäki-Patola et al., 2005] Mäki-Patola, T., Laitinen, J., Kanerva, A., and Takala, T. (2005). Experiments with virtual reality instruments. *Virtual Reality*, pages 11–16.

[Miller, 2010] Miller, J. (2010). Wiiolin: a virtual instrument using the Wii remote. In *NIME*, number June, page 497ff.

[Miyama, 2010] Miyama, C. (2010). Peacock : A Non-haptic 3D Performance Interface. In *NIME*, number Nime, pages 380–382.

[Pedersen and Hornbæ k, 2009] Pedersen, E. W. and Hornbæ k, K. (2009). mixiTUI. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction - TEI '09*, page 223, New York, New York, USA. ACM Press.

[Rodet et al., 2005] Rodet, X., Lambert, J.-P., Cahen, R., Gaudy, T., Guedy, F., Gosselin, F., and Mobuchon, P. (2005). Study of haptic and visual interaction for sound and music control in the phase project. *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05)*, pages 109–114.

[Shaer, 2009] Shaer, O. (2009). Tangible User Interfaces: Past, Present, and Future Directions. *Foundations and Trends® in Human–Computer Interaction*, 3(1-2):1–137.

[Sharlin et al., 2004] Sharlin, E., Watson, B., Kitamura, Y., Kishino, F., and Itoh, Y. (2004). On tangible user interfaces, humans and spatiality. *Personal and Ubiquitous Computing*, 8(5):338–346.

[Ullmer and Ishii, 2000] Ullmer, B. and Ishii, H. (2000). Emerging frameworks for tangible user interfaces. *IBM Systems Journal*, 39(3):915–931.

[Wanderley and Orio, 2002] Wanderley, M. M. and Orio, N. (2002). Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal*, 26(3):62–76.

[Zappi et al., 2010] Zappi, V., Italiano, I., Brogni, A., and Caldwell, D. (2010). OSC Virtual Controller. In *NIME*, number Nime, pages 297–302.