Using Leap Motion to create a physically interactive musical interface

Kevin Subrt Pizarro

Abstract— Nowadays, we can easily find among us more and more electronical devices that a few years ago, were only feasible in science fiction: Augmented reality glasses, smart watches, autonomous cars… It is hard to deny that we're living in an age of astounding technological breakthroughs, where each day brings new advancements, and progress is faster than ever. This project aims to contribute to the aforementioned progress, focusing in a recent technology: the motion tracking device called Leap Motion, not only by exploring its virtual object interaction capabilities, but pushing its limits to allow physical interaction with real-world objects. The final goal of this project, and its contribution to this field, is the creation of a musical device that uses the Leap Motion technology to provide an environment where users can interact with physical elements to play with music.

Index Terms— Experimentation, Human computer interaction, Leap Motion, motion tracking, musical interaction, physical interaction, user interface.

1 Introduction

One of the most fascinating apsects of computer technology is how it allows us to interact with things that go beyond reality. To make this possible, along the years, there have been many ways of interacting with technology: keyboards, pointers, mouse devices, touchscreens, etc. Nowadays, thanks to the ever-growing capabilities of computers, it's possible to take a step forward and interact with our favourite content, this time by only using our hands. Technologies like the Leap Motion, which will be the central element of this project, allow their users to get closer than ever to technology, by letting them touch, grab, pinch, pull, swipe any virtual elements they desire without the need of anything else than the device, and their hands.

In most cases, these virtual objects have a considerable flaw: they don't exist in the physical world. This project aims to change this by making use of the Leap technology out of its intended purpose, allowing its users to interact with physical objects, instead of virtual ones, to perform various actions in a software application.

The theme chosen for the application is a sound mixing app, that provides different ways of playing and experimenting with music. MagSound, the name of the system, allows its users to interact with various physical elements in an interactive surface, creating and modifying different sounds and audio effects without having to look at the screen.

1.2 Main contributions of this project

The main contributions of this project to the field of human computer interaction with Leap Motion are:

- The experimentation and analysis of the capabilities of the Leap Motion device when interacting with virtual and physical objects.
- The creation of various test apps that serve as examples of using the Leap Motion to interact with musical elements.
- The creation of an API using Javascript that extends the basic functionalities of the original Leap Motion API, providing new methods that allow to detect and process interaction with physical objects like on buttons and sliders with robustness.
- An audio mixing app that allows to interact with different music files, combining them and applying different audio effects.
- The MagSound system, a physically interactive musical device that makes use of the Leap Motion technology and combines all the previously mentioned elements to produce a final product that can be enjoyed by its users.

1.2 Structure of this paper

This document has the following structure: Section 2 is an overview of the state of the art of the project, where its background and context is analyzed. In Section 3, the development methodologies followed in the creation of the project are explained. Section 4 belongs to the first phase of the project, the experimentation phase, and it's where all the information related to the different experiments can be found. Section 5 is about the second phase of the project,

the development of the MagSound system. All the development process is explained in that section and its different sub-sections, with information about the different components of the system. In ection 6, the results of the project are discussed and analyzed. Nearing the end of the document, in Section 7 we can find the project's conclusions, followed by the acknowledgments and the bibliography.

2 State of the art

2.1 Leap Motion Technology

Fig 1. Leap Motion device connected to a laptop. The three IR LED's can be appreciated.

The Leap Motion device, shown in *Figure 1*, is a USB peripheral composed by 2 monochromatic IR cameras and 3 IR LED's, designed to be placed in a flat surface. These sensors allow the device to "observe" a hemispherical area of approximately 0.5m of diameter, tracking the user's hands and fingers position with a relatively large amount of precision.
The smaller observation area and higher resolution of the device differentiates the product from Microsoft's Kinect, which is more suitable for whole-body tracking in a space the size of a living room. The controller is capable of performing tasks such as navigating a website, using pinch-to-zoom gestures on maps, high-precision drawing, and manipulating complex 3D data visualizations [1].

2.2 Context

Despite giving the impression that it's a recent invention, Motion tracking technology has been around for nearly two decades, mainly being used by filmmakers and videogame creators to capture the movements of the actors and translate them to computer generated characters. It's in the recent years that this type of technology has flourished, thanks to the quick growth of the processing capabilities of both professional and domestic computers.

Domestic motion tracking and gesture recognition devices, like the Leap Motion or Microsoft's Kinect, are now affordable technologies that are starting to be popular among the general public. This opens a large window of possibilities to develop new and creative software that allows its users to interact with technology in unprecedented ways.

In the case of the Leap Motion, it's main use has been oriented to creating applications that allow users to interact with virtual elements, most recently combining it with virtual reality headsets. It's much less common to find applications that make use of the device to interact with physical elements, mostly because it was not designed for this purpose. This is where this project comes into play inside its context, since it focuses on pushing the limitations of this technology in order to give it a new use that differs from the already established.

2.3 Project background

One of the most relevant influences of this project is another project, conducted by fellow student Daniel Garcia during the 2015-2016 course as his Final Project. His work, called "*Interacción con contenidos multimedia mediante LeapMotion*" consisted of an introduction to the LeapMotion technology, focusing on the device's behavior and capabilities. In his work, he analyzed the different interactions that can be performed with the device, both with physical and virtual elements, providing many examples and demonstrations that allowed to understand the system.

His project served as a useful guide in relation to which interactions are more effective with the Leap device and which are not. This helped our project to easily discard those physical elements that are less effective, and focus on the ones that produce better results.

3 Methodology

This project has been carried out by a multidisciplinary team composed by experts in different fields, that have clearly benefitted the obtainment of diversified requirements and feedback. The 2 experts are:

- The project tutor, a software expert who provided feedback related to the usage of the Leap Motion API, object detection techniques and in general, information about some of the best practices of software development.
- Daniel Norton, a musical artist who provided insight into the field of computer generated audio, sharing his knowledge about audio libraries and usability of musical interfaces.

From a software development point of view, the project was split into two different and well-defined phases. The first half of the project was strongly driven by experimentation, whereas the second half followed a more conventional development process based on prototyping.

3.1 Experimentation Driven Development

For the first phase of the project, the virtual and physical experimentation phases, given the unfamiliarity with the capabilities of the Leap device and the lack of experience when developing with it, it was necessary to follow an experimentation driven methodology [2].

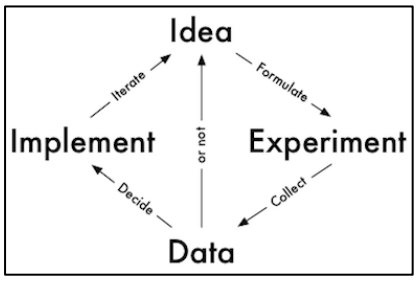

Fig 2. Lifecycle of an Experimentation driven development.

As seen in *Figure 2*, the main concept behind this methodology is that each iteration begins with a new idea which generates an experiment. Once the experiment is done, the data about it must be collected and analyzed in order to decide if its results are worth being implemented to the system or not.

3.2 Prototype Based Development

For the second phase of this project, the development of the MagSound project, given the emphasis on usability and on making an intuitive interface that adjusts to the user's expectations, a prototype based development methodology [3] was used.

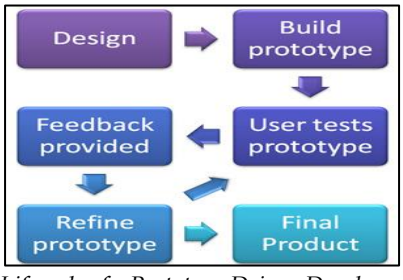

Fig 3. Lifecycle of a Prototype Driven Development.

A prototype based development, as shown in *Figure 3*, consists in designing and creating an initial prototype of the system, and then constantly refining it in response to the user's feedback until the final version of the product is achieved. Following this methodology was very useful for this part of the project because this way, the constant feedback obtained in the weekly meetings with the TFG tutors could be taken into account, allowing to add new interaction elements to the prototype and test the existing ones.

4 Phase 1: Testing the Capabilities of the Leap Motion Controller

4.1 Experiments without physical interaction

During this phase of the project, most of the effort went into creating small and simple web applications (called *virtual tests)* to experiment with the Leap Motion device's capabilities when interacting with virtual on-screen elements. These applications, although sharing many common traits, present an increasing complexity. Each of them, tests out different ways to interact with audio elements, using Javascript and HTML5 along with the Leap Motion Javascript SDK [4]. The graphical elements are created using the KineticJS HTML5 Canvas library, and the audio elements are implemented using the Web Audio API [5].

Links to various video demonstrations of the different apps can be found in Section 8 of this document.

4.1.1 Motion data and hand visualization

The main objective of the first virtual test was to successfully implement the Leap Motion device in a web environment using Javascript. In order to test that the device is appropriately set up and working, the app provides information related to the tracking data of hands and fingers as well as a live representation of the fingertip positions using the *canvas* HTML element. This test helps to understand how the Leap device reads and processes the motion data, and allows to analyze the different functionalities provided by the API [6].

4.1.2 Touch interaction

In the second virtual test, a rudimentary soundboard was created, in order to understand how the Leap device detects touch interaction. The soundboard consisted of four animal pictures that emit different sounds when touched. A new set of functions were created to calculate the bounding area of each image, and guarantee that the sounds are only played when the pointer's coordinates (index's fingertip) match with them. The depth (Z-axis) of the finger is also considered, in order to differentiate between *hovering* a picture and *touching it*.

All these functionalities were not implemented with enough robustness in the standard Leap API, so they had to be created and added to it. This test allowed to better understand the Leap Motion coordinate system, and it also was a good first encounter with the Web Audio API, learning how to create a simple script that allows to play local sound files when an event is triggered (in this case, when an image is touched).

4.1.3 Music interaction with Web Audio API

4.1.3.1 Virtual Theremin

Simplified version of the instrument, requiring only one hand to play it. When the user's hand moves vertically (Y-Axis), the amplitude (volume) is increased or decreased, and when it moves horizontally (X-Axis), the frequency is modified. This works by using the Web Audio API oscillator element. Thanks to this simple test, some advanced functionalities of the Web Audio API were learnt, allowing their use in future experiments [7].

4.1.3.2 Simple music player with audio effects

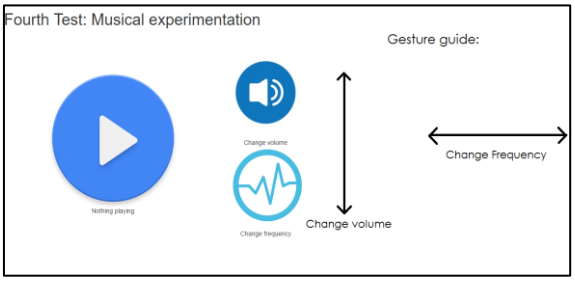

Fig 4. Third test overview. The interface consists of a Play/Pause button, and Volume and Frequency toggles.

The main objective of this test was to give a different approach to the virtual Theremin experiment, combining it with the experience obtained in the first and second tests, in order to focus on something more complex and realistic like a music player.

The primitive music player shown in *Figure 4* allows the user to interact with a sound file using only his hands, playing, pausing and changing different parameters of the song being played. All these audio control functionalities, and their mapping to the different hand gestures, had to be created from the ground up, adding to the number of new functions created.

4.2 Experiments with physical interaction

In this part of the project, the focus shifted towards adapting the previously created music player to a physical interface, allowing the user to interact with the different on-screen elements without having to look at it.

The Leap Motion device was not designed to allow interaction with physical elements, so in order to overcome this flaw, it would be necessary to replicate each physical element, making a virtual version of it. This would allow the user to indirectly interact with the virtual element when interacting with its physical counterpart, creating the ilusion that only the physical one exists.

In order to allow the sensors to detect physical interaction, it was necessary to consider different ways of setting up the Leap device and the physical elements to maximize the precision with which it detected the user's hand when interacting with them.

4.2.1 Choosing an optimal layout

In order to analyze what was the best way of setting up the Leap device to clearly focus on physical objects, 2 initial ideas came to life:

- Setting up the device to point downwards, focusing on the table; this was quickly discarded since the IR beams reflected on the table surface and didn't allow the device to work properly.
- Mounting the Leap device on a vertical surface, so it would focus partially on the table, where the physical objects would be placed. In order to do this, the coordinate system of the device and the arrangement of the buttons and sliders had to be changed. Unfortunately, although working with the music player created in the fourth test, a vast amount of precision was lost with the device set up in this position, since the fingers further away from the device weren't recognized properly.

These 2 rather ineffective results led up to the following conclusion: In the case of our project, the best way of setting the Leap device is pointing upwards. Once having this concept clear, the definitive layout, which is explained in the following section, was created.

4.2.2 Vertical layout and first physical prototype

After struggling to find an optimal layout for the physical interface, the definitive idea was to adhere the physical elements (sliders and buttons) to a vertical surface using magnets, right above the Leap device which would be pointing upwards, as seen in *Figure 5*.

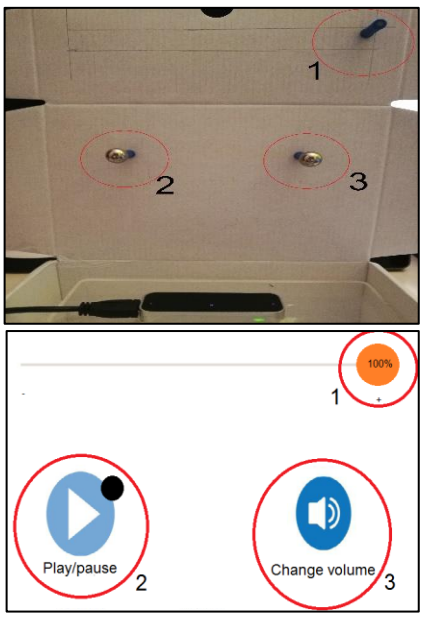

Fig 5. Virtual interface (bottom) that matches the layout of the physical interface (top).

There are many advantages of using this setup: firstly, it allows the Leap device to be in a fixed position in relation to the box, allowing to easily set fixed locations for the buttons and slider. Secondly, the magnets allow to slide the elements adhered to the cardboard with a certain amount of freedom. Thirdly and most importantly, this setup allows to use the device in its best orientation, granting a minimal loss of precision.

This new layout allowed to create a virtual interface that adjusted to the dimensions of the physical surface, mapping the position of the physical buttons and sliders to their virtual counterparts.

This simple test app allows the user to play and pause an audio file and set its volume properties by interacting with the physical elements.

4.3 Physical interaction detection

In order to precisely detect when the user is interacting with a physical element, it's necessary to match the dimensions of the physical interaction area with the virtual one. The interaction area of the Leap Motion device, called *Interaction Box*, translates to a cubic area of maximum 25cm wide, 25cm tall and 15cm deep, in which the device is placed below it, centered in the X and Z axis [8]. This can be better understood by observing *Figure 6*.

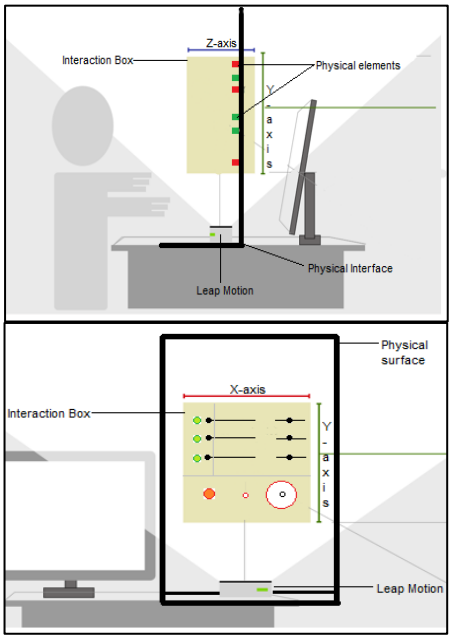

Fig 6. Graphical representation of the Interaction Box. Lateral view is on the top and frontal view on the bottom.

The size of the *Interaction Box* has been adjusted so it fits the size of the interface. By setting the Leap device in a fixed position in respect with the physical interface, it's possible to do a 1 to 1 correlation between the board's coordinates and the Leap's coordinates. Then, by using the *normalizePoint()* function, it's possible to calculate the exact position of the different elements.

For each physical element, an "activation area" was created, determined by its coordinates inside the larger area of the *Interaction Box*. User interaction is detected when the user's index finger enters said area, in which case, the system has been programmed to interpret that the element is being touched. Any interactions done inside an element's activation area will reflect on the virtual version of the element, generating events, or modifying values in the audio mixing software.

The dimensions of an object dictate its activation area, and determines at which point of the Z axis the object is in contact with the user's finger. This calculation has been implemented in a new function that discriminates between hovering and touching an object. The dimensions of the activation areas are slightly larger than the real dimensions of their objects (approximately 20%) in order to provide a necessary margin of error to compensate the inaccuracy of the Leap device [9].

5 Phase 2: The *MagSound* project

The MagSound project is the development of a system that allows users to play with music by interacting with diferent physical elements, making use of the motion detection capabilities of the Leap Motion Device and the new functionalities added to its API during the different experiments. This system gathers, recycles and puts to good use all the knowledge and experience obtained in the first phase of the project, the experimentation phase, in order to create a solid, useful and well defined final prototype that can be enjoyed by the users. It is a materialization of all the concepts and ideas previously obtained, expanding, and giving them a meaningful purpose.

5.1 System requirements and objectives

Given that the MagSound project follows a prototype driven development methodology, the input of new requirements and objectives was constant in every new version of the system. This differs from the more classical system requirement capture process that takes place at the initial phases of conventional development methodologies. However, some general goals and objectives had to be set in order to guide the general direction of the project and delimit its scope:

- 2 interfaces must compose the system: a physical one, and a virtual one that maps all the elements of the first one.
- The physical interface must provide different physical elements for the user to interact with them in different ways.
- The system must support different forms of user interaction with audio elements and music.
- The user must have a certain degree of freedom when interacting with the system, allowing creativity.
- The user must be able to completely use all of the system's functionalities by only interacting with the physical interface, ideally being unaware of the existence of the virtual interface.

These general requirements allowed to shape and define the first prototype of the system. After creating it, each new sub-sequent version generated new feedback, and brought newer and more accurate requirements that slowly refined the system's functionalities and design, taking it to its final version.

5.2 Technology used

For the development of the MagSound project, many different software libraries and hardware components have been used in order to make possible its creation.

5.2.1 *Software Technology used*

- Javascript
- HTML 5
- KineticJS (HTML5 canvas library)
- Web Audio API
- WAMP Server
- LeapJS
- Notepad++ (Development environament)
- Microsoft Office 2016 (Documentation)
- Skype (Online communication)
- Dropbox (File sharing)
- Google Chrome browser

5.2.2 *Hardware Technology used*

- Leap Motion Device
- Personal Computer
- LUNS magnetic board
- MAME style pushbuttons
- ALPS sliding potentiometers

5.3 System software architechture

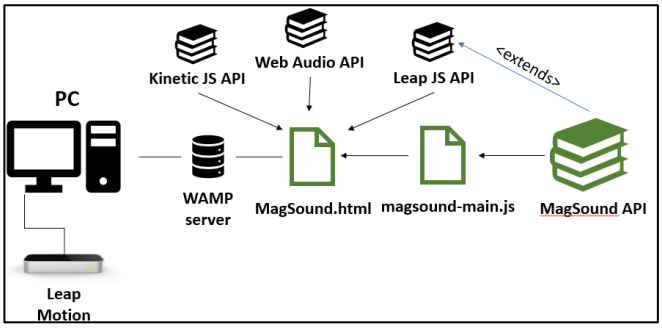

Fig 7. Overview of the architechture of the system.

The structure of the system can be found in *Figure* 7. From a software perspective, the mixing app of the MagSound system is a web application made with Javascript. It's composed by an HTML interface, called *MagSound.html* where all the visual elements are created and all the scripts, libraries and style sheets are loaded. It all runs in a local WAMP server. Most of the app's functionalities are located in the main Javascript file called *magsound-main.js*, which uses the functions and classes created in the *MagSound API*, an extension of the original *Leap Motion Javascript API* that contains all our new functions. There is where all the code related to the different functionalities of the app, including the audio management, event management and motion detection functionalities, has been created.

5.4 The virtual interface

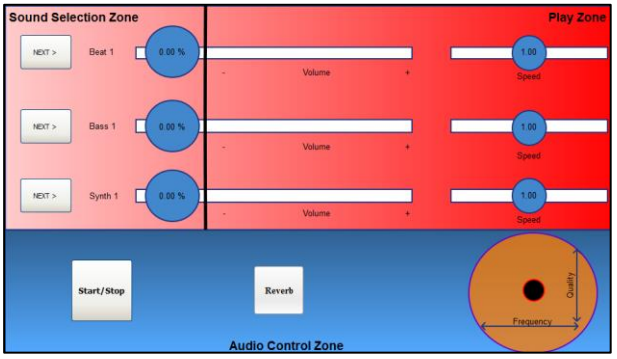

Fig 8. Overview of the virtual interface, dubbed The Zones interface.

The virtual interface, also called "Zones interface", is the main component of the MagSound system. The interaction area, as shown in *Figure 8*, is divided in 3 zones, each of them with its own role inside the system:

- The **Sound Selection zone** is where the different audio files can be selected and changed. Overall, there are 3 audio channels where different types of sounds can be selected. These audio channels are represented by each of the 3 long sliders shown in *Figure 8*. In the basic configuration of the system, the first audio channel represented by the top slider is loaded with various beat tracks, the central slider/audio channel is loaded with bass tracks and the bottom channel is loaded with different melodies. The button next to each slider allows the user to toggle the tracks being played on their channel. The blue circle (slider knob) represents the volume of the track being played on that channel. As long as a slider knob is inside the Sound Selection zone, it will be muted.
- The **Play zone** is where active sounds are being played. In order to play a sound, the user must drag the slider knob from the Sound Selection zone into the Play zone. The position of the knob inside the Play zone determines its volume. Next to each volume slider, a set of shorter speed sliders can be found. These sliders allow the users to adjust the playback speed of the track being played in that channel.
- The **Audio Control zone** is where different parameters of the output mix can be modified. Here the user can find general controls like the play/pause button, and different audio effect inputs like the reverb button, which adds an echo effect to the mix, or the filter touchpad, which activates a frequency filter inside the active audio channels, allowing to modify their frequency and quality values.

This interface is an attempt to provide a simplified version of a professional mixing software, while maintaining a simple and intuitive layout that can be used and understood by both experienced and unexperienced users. Its interactable elements boil down to buttons and sliders, given that they are intuitive and easy to interact with. These virtual elements are also easily translatable to the physical interface, simplifying the task of finding physical counterparts that share the same characteristics.

5.5 The physical interface

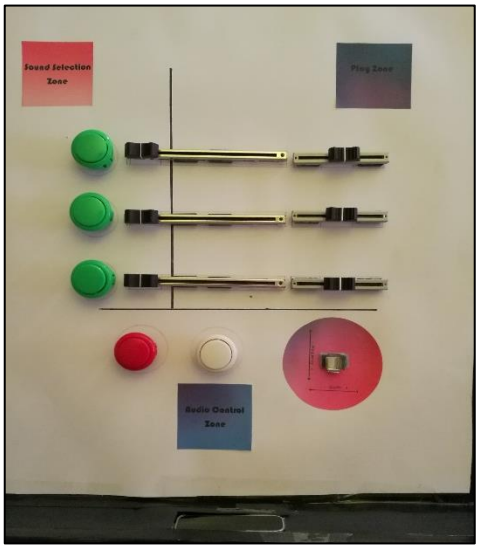

Fig 9. Overview of the physical interface. Note the position of the Leap device below the different input elements.

The physical interface, shown in *Figure 9*, consists in a wooden magnetic board that can be placed in vertical position, and has a little base that allows to place the Leap device [10]. When comparing *Figure 8* with *Figure 9*, we can appreciate that each element of the virtual interface is represented in the physical interface. This is the interface which the user will always interact with. It allows to make full use of the virtual interface's functionalities even when being unaware of its existence.

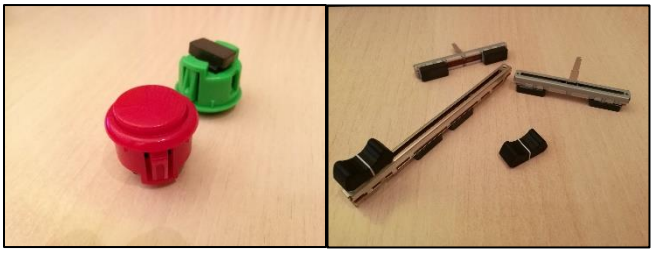

Fig 10. Closer look at the physical buttons (left) and the physical sliders (right). Note the magnets adhered in the rear part of each element.

The main elements present in the interface are long and short sliders and different coloured buttons as shown in *Figure 10*. The buttons are real, clickable, arcade buttons. This design is easily understandable by the user, and provides a good feedback when interacting with them. The different colours provide some guidance to where each button should be placed. In order to help the user realize if a button has been properly pressed or not, the system plays different audio cues whenever an interaction has been correctly detected.

On the other hand, the sliders are real potentiometers that can be interacted with by sliding the knob along their central railing. This design, together with the concave shape of the knob, guarantees that the user input is detected with precision. The long sliders are mapped to the volume virtual sliders, and the shorter ones are mapped to the speed sliders.

5.5.1 Adding and removing elements to the physical interface

Although the highest precision is achieved when all the physical elements are present on the board in their predefined positions, the system is designed to give the user a certain amount of freedom, allowing him to add and remove the different sliders and buttons. This is possible thanks to the *Edit* mode. Whenever the system is stopped, whether because it's in its initial state, or because the user has pressed the *Pause* button, the system enters a state in which it is possible to change the layout of the physical interface.

In order to add new elements to the interface, the user must attach them to their predefined positions in the interaction area. When empty, the physical surface presents different markings that indicate where the different elements can be placed. This design resembles a traditional puzzle game since it allows a certain degree of freedom in the order in which the elements are placed, but limits the possibilities when choosing their position. This restriction had to be made to make sure that the physical elements are always in a postion that maps correctly to their virtual counterparts.

The *Edit* mode also allows to remove elements from the interface. In order to remove an element from the interface, the user must "kill" it by touching it, and then proceed to detach it from the board. When an element is killed, an audio cue is played indicating that it can be safely removed. When the user exits the *Edit* mode and resumes playback, any sound or effect being generated by a removed element, will have disappeared.

Some video demonstrations explaining this feature can be found in Section 8 of this document.

5.6 Integration of the Web Audio API

The audio environment chosen to create and manage all the audio functionalities of the MagSound system has been the Web Audio API, given its integration with Javascript, its compatibility across browsers and its powerful but intuitive audio management capabilities [11].

The Web Audio API represents an audio channel as a set of interconnected nodes, that go from the source node which contains the audio, to the output node which emits the sound through the speakers [12]. Between these two nodes, many different extra nodes can be added to modify the parameters of the output sound, such as filter nodes, gain nodes or convolver nodes.

In the MagSound mixing software, 4 audio channels (or *audio contexts*) have been created: one for each of the three rows of sliders, and a fourth one for the different audio cues and other independent sounds. In *Figure 11*, the node hierarchy and structure of one of the audio channels created for the mixing app is represented.

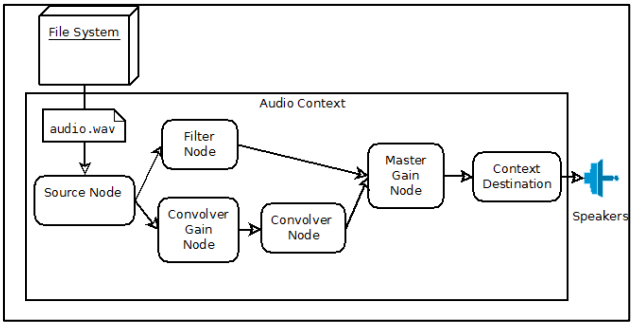

Fig 11. Graph that represents the different audio nodes used in the MagSound app.

All the different audio files are loaded into source nodes using an XMLHttp request. Then, these source nodes are connected to two different sets of audio effect nodes: a lowpass frequency filter node, and a convolver node. The first effect allows the user to filter the sound, modifying its frequency and quality, producing some interesting effects. On the other hand, the convolver node adds a reverb effect to the mix, making the sounds play with echo. Both effect nodes are connected to a master gain node, which controls the volume of the channel. Finally, it all connects with the context's destination, in our case the speakers, which allow the sounds to be played.

6 Results and User Experience

Given the clear division of the project into two separate phases, the analysis of the results obtained from each part is different. The results of the first phase are more technically oriented, while the results of the second one are focused on the usability and the user experience of the system.

6.1 Results of the experimentation phase

The first part of the project focused on testing the capabilities of the Leap Motion device, creating various incremental experiments that generated a large amount of knowledge and feedback.

When testing the device in virtual-only experiments, the results were very positive. It's proven that the Leap device performs very well when interacting with elements on a 2D graphical interface composed of various elements such as buttons, sliders, movable elements, etc. It also detects with good precision different gestures such as pinching and swiping, allowing the developer to create more complex interaction methods.

When trying to translate all these interactions to a physical interface, and apply them to real-world objects rather than virtual ones, is when the Leap's performance starts to falter. Then, the main flaw of the device becomes obvious: It was not designed to support interaction with physical objects. Despite the pessimism of the previous statement, there are ways of working around it, as it has been successfully proven through the different physical experiments and the development of the final system.

Different types of interactions produced different results. Generally, interactions that required various fingers to be performed, like when pinching to grab an object, in most cases aren't correctly detected by the sensors, lacking the necessary precision to correctly determine if the gesture is being applied to a physical object or an empty space. On the other hand, interactions that required only 1 finger to be performed, provided much better results, detecting with higher precision the interaction with physical objects.

The results obtained in this phase of the project determined the design of the MagSound physical interface. The two main interaction elements, buttons and sliders, had proven to be the most effective input mehods.

6.2 Results of the MagSound project

The MagSound system was created with the goal of allowing its users to interact with the system in an intuitive and simple way. Having this in mind, many design choices were influenced by the way the user interprets and understands the interface. In order to obtain this feedback, there was a constant communication with both tuthors of the project, along with some usability tests performed by untrained and unfamiliarized users.

One of the most relevant conclusions extracted from the tests was that users were prone to using 2 to 3 fingers when interacting with the physical elements. This, as explained in *section 6.1* of the document, causes the Leap device to lose a great amount of precision, misinterpreting the position of the fingers. In order to overcome this situation, the design of the physical elements has been carefully chosen, selecting buttons and sliders that force the user to

unconsciously use only 1 finger. That is the reason why the buttons are rather small, and the slider knobs have a pronounced concave shape, having space to only put the index finger.

In general, the results of the user experience interacting with the system were satisfactory. Despite requiring some initial explanations about the general design of the interface and the operation of some of the physical elements, the system has proven to be intuitive enough to allow them to use it with freedom. The different audio cues played by the system when the user does certain interaction, helps them to easily understand what effects their actions are having in the mix that's being played. It generally doesn't take long until they understand how the system is tracking their interactions, but until that moment, there's a short period of "magic" where they believe that only the physical interface exists.

7 Conclusions

Having gone through the different phases of the project, its features, technologies, and results, this document is reaching an end. We have porgressively seen how the Leap Motion's capabilities can be used to achieve physical interaction detection, going from the creation of simple experimental apps, to the creation of a final, well defined system that makes use of all the previously obtained knowledge.

We can conclude that the Leap device, although not being designed to support physical interaction, can be used to create the illusion for the users that it does. By making a 1 to 1 correlation between the physical and the virtual elements, we can allow users to interact with a user interface only by using real-World objects. This solution, although being effective for static and simple interfaces, would present some flaws when used for more dynamic interfaces, since it limits the user's freedom and possible interactions.

Nontheless, thanks to this technique, it has been possible to create the MagSound system, a fully functioning physical interface that uses Leap Motion out of its intended use, allowing to interact with physical objects. This can be considered a success in relation to the initial planning and requirements of the project, proving that the Leap Motion technology can be used beyond from what it was designed for, and give it an interesting use, in our case, for playing with music.

7.1 Future development

Some of the current limitations of the Leap Device when detecting interaction with physical elements, could be overcome by implementing some additional system that detects the objects' positions and features. The best way of doing this would be to implement a background subtraction algorithm that analyzes the physical surface, detecting wether an object is inside the Leap's area of interaction or not. This would allow to add and remove elements with a higher fidelity, and even associate different functionalities to different objects as their shape is recognized.

Given the low quality of the cameras inside the Leap device, it would be necessary to add an external, higher quality camera that provides a clearer image of the surface, facilitating the implementation of the background subtraction system.

This improvement over the original design would increase the possibilities when interacting with the system, giving more freedom to the users and allowing more creativity.

8 Video demonstrations

In this section, we can find links to videos that demonstrate the functionality of the different test apps and the MagSound system:

- Video 1 – TFG First Test:
https://youtu.be/ZGmm9Zwfu7Y?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

- Video 2 – TFG Second Test:
https://youtu.be/hPCt_OYU0sU?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

- Video 3 – TFG Third Test:
https://youtu.be/NE__m9bD4pI?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

- Video 4 – TFG Fourth Test:
https://youtu.be/kDAMU1nho-g?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

- Video 5 – TFG MagSound V1:
https://youtu.be/sGnR8TKEezo?list=PL3FGc_TXBDUffSR-PaGLnG_I-aii-G0c4j

Video 6 – TFG MagSound V2:
https://youtu.be/caxqQc0Pp-U?list=PL3FGc_TXBDUffSR-PaGLnG_I-aii-G0c4j

Video 7 – TFG MagSound V3:
https://youtu.be/WpQ7yz9gKXw?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

Video 8 – TFG MagSound Final:
https://youtu.be/d7JkiQWfEHY?list=PL3FGc_TXBDU-ffSRPaGLnG_I-aii-G0c4j

ACKNOWLEDGMENTS

Special thanks to both tutors Fernando Vilariño and Dan Norton for their guidance and support throughout the project, and to Daniel Garcia for sharing his experience on the subject with me.

BIBLIOGRAPHY

[1] Terdiman, Daniel, "*Leap Motion: 3D hands-free motion control, unbound*". Cutting Edge. cnet. [Online] Available at: https://www.cnet.com/news/leap-motion-3d-hands-free-motion-control-unbound/ Visited: 27/09/2016.

[2] S. Auvray. "Experiment Driven Development – The Post-Agile way". InfoQueue. [Online]. Available at: https://www.infoq.com/news/2010/02/edd-post-agile. Visited: 20/10/2016

[3] A.M Davis. "Operational prototyping: a new development approach". Department of computer science, Colorado University, Colorado Springs, CO, USA. [Online] Available at: http://www.ing.unp.edu.ar/asignaturas/is/papers/t.pdf . Visited: 12/11/2016

[4] Leap Motion Development Team, *Getting started with Javascript*. Leap Motion Developer. [Online]. Available at: https://developer.leapmotion.com/getting-started/javascript Visited: 28/10/2016

[5] S. Michaud *et al*, *Using the Web Audio API*. Mozilla Developer Network. 2016. [Online]. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Using_Web_Audio_API. Visited: 01/11/2016

[6] G. Marin, F. Dominio, P. Zanuttigh, "Features extraction from Leap Motion data", inside *Hand gesture recognition with Leap Motion and Kinect devices*. Department of Information Engineering, University of Padova. IEEE, 2014, pp. 02-04

[7] C. Wilson, *Web Audio changes in m36*. Google Developers. 2016. [Online]. Available at: https://developers.google.com/web/updates/2014/07/Web-Audio-Changes-in-m36. Visited: 31/10/2016

[8] A. Colgan, *Coordinate system – The interaction box*. Leap Motion Development Team. [Online]. Available at: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html Visited: 15/10/2016

[9] J.Guna, G.Jakus, M.Pogačnik, S.Tomažič, J.Sodnik. "An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking". Faculty of Electrical Engineering, University of Ljubljana, Ljubljana. Slovenia. 2014. Published by MDPI AG, Basel, Switzerland. [Online]. Available at: http://www.mdpi.com/1424-8220/14/2/3702/html. Visited: 17/11/2016

[10] A. Davis, *How to build your own Leap Motion Art installation*. Leap Motion Blog. [Online]. Available at: http://blog.leapmotion.com/how-to-build-your-own-leap-motion-art-installation. Visited: 25/10/2016

[11] B. Smus, *Web Audio API: Advanced sound for games and interactive apps*, free virtual edition. Oreilly Media inc. Newton, USA. 2013. [Online]. Available at: http://chimera.labs.oreilly.com/books/1234000001552/index.html. Visited: 02/11/2016

[12] Tizen Developers, *Advanced Web Audio API usage*. Tizen developers community. [Online]. Available at: https://developer.tizen.org/community/tip-tech/advanced-web-audio-api-usage. Visited: 10/12/2016

- *E-mail de contacte: kevin.subrt@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Fernando Vilariño Freire (CVC) i Daniel Alexander Norton*
- *Curs 2016/17*