DAVID HAMILTON

MU803

MUSIC TECHNOLOGY PROJECT

AKI PASOULAS

LEAP SAMPLE

WORD COUNT 3,249

Music Technology Project – Leap Sample

The desired end product for the Music Technology Project was to create a standalone OSX application, a software instrument with interactivity and creative input control as the core component. Designed to record, load and manipulate audio samples live, the software will exhibit two primary playback modes, first using similar techniques to granular synthesis and second as a form of sample looping. The infra-red, hardware tracking system, Leap Motion, was to be used as the main control input, with the application designed to be used as an instrument with which you can perform live, varying multiple parameters using nothing but the motion of your hands.

Of all the many creative influences behind the inception of this project, the mi.mu gloves, championed by composers and musicians such as Imogen Heap and Tim Exile, represents the most clear and concise. The gloves, a form of wearable technology, offer the user a means with which they can control and map messages for use within audio digital workstations and electronic instruments. They are built using a multitude of physical tracking sensors, such as gyroscopes, accelerometers and flex sensors, combined with low latency I/O broadcasting over wi-fi. The use of these sensors allow a performer to map an extensive list of hand and arm motions to musical parameters, freeing the performer from the confines of the keyboard and mouse and allowing them to explore the opportunities offered by mobility, rather than being routed to a single spot in front of their laptop or instrument.

Whilst the gloves offer a ground-breaking means with which musicians can manipulate and create music, they unfortunately come at an incredibly steep price, out of reach for many musicians, let alone hobbyists. With their recent Kickstarter unsuccessfully reaching their goal, receiving £132,776 of the desired £200,000, it is clear that there is a desire for this form of technology to be made commercially available, however, with a single mi.mu glove offered at a donation price of £1,200 or above, many would be backers were incapable of fully committing to the funding.

My goal in this project was to offer a fully featured software instrument that could be controlled with similar hand motions and gestures, released for free and designed to be used with the Leap Motion controller. The Leap Motion is a widely available motion sensor, similar to technologies such as the Microsoft Kinect. The Leap Motion can track a wide array of hand motions and gestures, mapping the positioning of the individual bones of the hand within a three-dimensional space. The product is also designed to be developer friendly and to be used within the realms of gaming and application interactivity. With an RRP of around £70 and often found at most retailers for much less (usually around £30-£40), this low price and wide availability is crucial in offering a unique musical experience to a wide audience, attempting to fill the gap in the market left by the mi.mu project. The desired end result is to produce an application, at a fully featured beta development stage, ready to be released to the public, entirely for free.

Within my broader body of research, there are a number of other artists, authors and products that have played their own independent role in the development of my concept and eventual code. The writing of Todd Winkler in *Composing Interactive Music* was a keystone in the formation of my ideas, discussing the relationship between performer and computer and the means in which 'the computers capabilities are used to create new musical relationships that may exist only between humans and computers in a digital world' (2001: 4-5). These definitions and discussions influenced my approach in considering how the software would react in kind to the performer and how it would relay that information, reinforcing the bond between performer and software.

In *Bodily Expressions in Electronic Music*, Deniz Peters discusses the "inseparable bond between body and music"(2012: 1) and ponders the disembodiment between computer music and physical actions. This serves as an interesting point of contemplation within in the context of my own project, which itself offers no interaction between the user and a physical object of any kind, allowing the performer to create musical expressions from nothing but the abstract concept of space in front of them and their own movements within it. Whilst investigating the 2005 performance *Immanence*, Susan Kozel states ‘literality between physical gesture and emission of sound is a building block for coherence in any interactive relationship between dancer and sound’ (2012: 65). It is this viewpoint that represents the exact interactive relationships that I wish to form between sound and performer; with logical and relatable coherence between movements and the sounds they are capable of producing. The ability to open and close a filter simply by opening and closing your hand, this should feel as natural to the performer as pressing the key of a piano. The untraditional output of artists such as Leafcutter John have also played their own role in opening my mind to new manners in which any form of data can be controlled, whether it be using candles to adjust Max MSP parameters or growing piezoelectric microphones using crystals.

These untraditional control techniques are also an important aspect of my idea as it offers musical control to people who have zero musical experience or restricted physical ability. Whilst the application would take a great deal of skill and practice to master, the plug and play nature of the software offers the capacity for users to create interesting and pleasing musical material in a manner much easier than offered by traditional instruments such as guitar or piano. Creating an interactive system that can produce results from such primitive and natural body movements is the key to my conceptual process and using hand motion as my primary input control endows users with the ability to create sound using the simplest and most obvious motions that we often take for granted, such as the movement within a three-dimensional space, or the axis upon which we rotate and position our hands. Another one of the more meaningful prospects of developing software in this manner is gifting those with particular physical disabilities the opportunity to create music, something which they may otherwise be incapable of. This includes motor skill disabilities such as dyspraxia or muscular dystrophy, or even conditions such as arthritis or Parkinson’s disease. The ability to assign both broad and intricate musical techniques to these rudimentary body motions can offer a unique and radical creative experience.

Along with these external influences, a sound instillation I created in the final year of my undergraduate degree played an important role in the impetus behind the creation of this application. The piece was designed in Max MSP and involved a hardware element, with the conceptual focus upon creating an object that could react to independent environmental aspects, such as temperature, light or colour, and produce unique generative musical material based on this data. In the development and display of this piece, dubbed Monolithic Synthesis, I came to find that many people who viewed the object were most interested in the interaction between their own presence and the sound output, forcing themselves into a position in which they became the primary focus of the piece and manipulating the sensors to demand musical results. Most importantly was witnessing the interaction between the object and those who had little to no musical knowledge, discovering the satisfaction and thrill offered by the ability to create or at least affect musical results with ease, something which they may had not experienced up until that point. This reasoning is also pivotal in why I chose to create my own software instrument, rather than mapping values for use with digital audio workstations or instruments, entities that usually require a level of musical knowledge to operate.

Building the software was a step-by-step process that involved perpetually switching between audio and Leap Motion development. However, the development initially began with the planning and building of the audio processors. The majority of these processes, manipulations and synthesis techniques were built upon knowledge I had already gained during my undergraduate degree, with my new patch designs heavily influenced by earlier attempts and assignments. The focus then, was upon making my pervious designs more efficient, sophisticated and audibly diverse as well as integrating new features and functions.

As mentioned, the software was to feature two modes, the primary mode was to be a granular synthesiser, a technique used to break down a sound into small snippets, or grains, of audio, often between 1ms-50ms (the microsound scale) and playing them back in fast succession. However, I also included the ability to use grains that are much longer than you would find within a traditional granular synthesiser, something which is useful in creating denser and uninterrupted sound textures. This was built using the poly~ object in Max and was created with feature rich control in mind, enabling the modulation of multiple parameters to form a diverse range of sound material. The second feature was influenced by the prototype synthesiser, *Collidoscope*, designed by Ben Bengler and Fiore Martin, which played a key role in the inception of this application. This mode consists of a uncomplicated sample looper that allows the user to adjust the speed and pitch of the playback, as well as offering a larger window size of which one can select and process sound. Pitch manipulation is available in both modes, an important facet of the application when considering the performance aspect, especially collaborative performances. Pitch shifting works in three modes, allowing the motions to shift freely a full octave in either direction, snap to semi-tones or even map to a number of common musical scales.

These two sound producing modules were then fed into a number of audio effects that, for the most part, take their origin from a variety of popular audio effects used in live performance. These included frequency filters, modulation effects, delays and reverbs. The writing of these effects varied, from simple processes such as amplitude modulation for tremolo and ring modulation effects, to complex computations, such as algorithmic reverbs with pitch shifting written into the feedback line or even modeling famous effects units, such as attempting to capture the characteristics of a *Binson Echorec* delay. The goal here was to offer a small range of effects that can be manipulated using the Leap Motion, adding an abundance of options to the live performance aspect of audio manipulation, whether it be adding the illusion of space, manipulating the timbre or distorting the temporal quality.

Due to that fact that I wanted to be to focus my efforts primarily upon how I would control the manipulation of audio, I decided that the best option to deal with the tracking capture was to use an external already built for the task. After a great deal of investigation into various max objects and externals, I finally settled upon the leapmotion.mxo, built by Jules Francoise. This external can capture and output all sensory data captured by the Leap Motion, including many that were not featured in the final build, such as the location of the metacarpal bones or the yaw axis of the hand. Going forward I would like to build my own object or external for performing similar functions, made more efficient by limiting the data to the key figures the software requires. However, for the development of this software, it offered an incredibly easy means of accurately capturing and routing the data I needed.

Whilst the external object dealt with the capture of sensory information, a great deal of decision making and calculation was needed to parse this information into appropriate variables that I could actually use. I had decided quite early on in the testing stages which hand motions would be the most appropriate for use as input control. This testing was performed to analyze and discover which motions were tracked the most accurately by the Leap Motion, as well as which ones felt the most comfortable to modulate. The most obvious to control, and easy to capture, was the position of the hands with the space above the Leap Motion. This simply scaled the X, Y and Z-axis positioning of the palm into values of 0-1000. This control represents the most natural form of control and is most useful when controlling data that requires maximum accuracy.

The next controls required varying degrees of data calculation and specific equations. It was clear that the Leap Motion could track the principal axis (aircraft principal axis) of the hand with a great degree of accuracy. The pitch and roll were the most comfortable controls to modulate, with yaw acting as a confusing parameter when combined with other motions and therefore disregarded. Both roll and pitch required working out the differential in the distance of certain points of the hand within the range of a particular axis. The pitch was determined by calculating the differential between the wrist and the distal (the tip) of the middle finger, within the confines of the Z-axis. Meaning that, as the pitch is increased (or decreased), the tip of the middle finger will move closer to the wrist within that axis and will result in a smaller differential. In a similar fashion, the roll of the hand was worked out by calculating the differential between the proximal bones of the thumb and little finger within the confines of the Y-axis.

The final motion I wanted to include in the software was the ability to use gripping motions, or conversely the extending of the finger span. The Leap Motion itself actually tracks these movements, with it outputting both a pinch and a grip variable. However, I found this data to be wildly temperamental and unreliable for use as a control input, offering jittery data and often entirely missing gestures. I discovered that calculating the differential between the distal of the thumb and little finger was much more reliable. This did pose a number of challenges however, as I had to calculate the distance between the two within the confines of multiple axis at one time, with the various differentials constantly compensating for one another.

It was important that audio recording could also be controlled by using the Leap Motion alone, enabling live capture to be an integral part of a live performance, negating the need to transition between mouse and Leap Motion. The controller has a small number of gestures that can be detected by the hardware itself, these include motions such as screen tapping and hand rotation. The most reliable of these motions however, was a swiping motion, similar to briskly turning the page of a newspaper. Whilst this control is both to used enable and disable recording, as well as clearing the buffer, the gesture detection is inconsistent and often unreliable, requiring incredibly precise and controlled gestures to register.

Unfortunately it is these small tracking issues that end up representing the Leap Motions most evident disadvantages, especially when compared to physical tracking devices such as the mi.mu gloves. There are a multitude of variables which can affect the efficiency of the tracking, this can range from the obvious, such as poor light levels or smudges on the body of the hardware, to issues such as obscuring one hand with another, rapid movements or simply the act of mistaking the right hand for the left hand or vice versa. There are certain issues that can be accounted for within the software such as jittery data as previously mentioned, rectified by smoothing and limiting values sent from the Leap Motion. Issues such as light levels and camera smudges are obviously mentioned in the Leap Motion manual itself and the ideal environment in which it should be used is discussed in the software and driver instillation, however, this environment is not necessarily ideal for live music performance.

Accounting for these tracking issues was one of the most important aspects when designing this application and whilst the eventual techniques used may be simple, there were a multitude of other smoothing and data clipping techniques that were tested in depth, attempting to discover which method represented the most reliable and accurate tracking data. Whilst these disadvantages in tracking may have a minor negative effect on software development, it is difficult to condemn the Leap Motion due to the price and availability, and as previously mentioned, the performance to price ratio is overwhelming when compared to many of the physical tracking devices currently on the market.

Other aspects such as preset storage, and control mapping is discussed in detail within the patches, but my primary focus was to offer an application that allowed the user a great deal of freedom to map and recall controls, making decisions in whichever way they please. During my initial stage of development I had decided to lock certain controls to particular mappings, however, after product testing, I found this limited certain techniques and motions from being utilized as well as stifling a certain element of creativity.

As discussed earlier, the final product was to be designed as a beta release, representing a stage in the development cycle in which the application contains the majority of features that will be included in the final release. In its current state, the software perfectly represents this, with only a small number of features that may be added in the future depending on user feedback (for example, the addition of logarithmic or exponential scaling for the leap motion data, dry audio path or the ability to preview a sound file). During the alpha development cycle, which represents all development up until this point, the software (and a Leap Motion controller) was distributed to a small number of friends and colleagues, with the desire to gain user feedback, helping me to add or adjust features. This resulted in the inclusion of reverse data scaling for example or the constant tweaking of data smoothing and control scaling. This stage was also essential in making decisions regarding the aesthetics and layout of the software. Opinions from various people were collated regarding how best to separate the control sections and the most appropriate manner in which to operate features such as the visualiser, information or audio settings.

Now that the software has entered the first stage of the beta development cycle it is ready for its initial public release. This will involve a free version of the software in its current state, released with promotional material, onto the official Leap Motion store. I plan to advertise the product upon forums and across social media, with an attempt to gather a large enough user group to perform in depth product testing. Going forward, this stage of bug reporting and user feedback will be crucial in producing a finished product ready for commercial release. It will mostly serve to iron out any errant bugs manifested within the application still, as well as to further CPU efficiency across various systems. The length of this beta testing stage and the date of the final release will be dependent upon the success of this testing phase. It is also worth noting that during this stage I may look into adding the ability to utilize other methods of hand tracking, such as the Microsoft Handpose or the Myo. I am also planning to release a version for use with Ableton and whilst it represents a great level of difficulty, I will be investigating how I might port the software to the VST format.

REFERENCES

[WRITTEN]

BISOP, C. (2012). *Artificial Hells: Participatory Art and the Politics of Spectatorship.*
Verso Books.

CIPRIANI, A. (2013). *Electronic Music and Sound Design: Theory and Practice with Max and MSP, Vol. 1.*
Contemponet.

CIPRIANI, A. (2014). *Electronic Music and Sound Design: Theory and Practice with Max and MSP, Vol. 2.*
Contemponet.

COLLINS, N. (2006). *Handmade Electronic Music; The Art of Hardware Hacking.*
Routledge.

DI SCIPIO, A. (2003). *'Sound is the interface': from interactive to ecosystemic signal processing.*
https://www.ak.tu-berlin.de/fileadmin/a0135/Unterrichtsmaterial/Di_Scipio/Sound_is_the_interface.PDF

FARNELL, A. (2010). *Designing Sound.*
MIT Press.

HOLMES, T. (2012). *Electronic and Experimental Music.*
Routledge.

PETERS, D. (2012*). Bodily Expression in Electronic Music.*
Routledge.

WINKLER, T. (2001). *Composing Interactive Music; Techniques and Ideas Using Max.*
The MIT Press.

[VIDEO]

DOCTOR MIX. (2015). *Crazy Synthesiser Demo.*
https://www.youtube.com/watch?v=9XMfKYVu_fg
[Accessed 23rd May 2016]

LEAFCUTTERJOHN. (2013). *Light sensing algorithmic music prototype V3 Max/MSP and light sensors.*
https://www.youtube.com/watch?v=qTvE0NqpAiE
[Accessed 23rd May 2016]

STUDIO XX. (2006). *Immanence – Technologies of Inner Space.*
https://secure.studioxx.org/en/matricule/2005fes50140c-1
[Accessed 23rd May 2016]

WIRED UK. (2013). *Imogen Heap Performance with Musical Gloves Demo: Full Wired Talk 2012.*
https://www.youtube.com/watch?v=6btFObRRD9k
[Accessed 23rd May 2016]

[ONLINE]

AKAMATSU, M. (2016). *aka.objects.*
http://akamatsu.org/aka/max/objects/
[Accessed 23rd May 2016]

COLLIDOSCOPE. (2016). *collidoscope.*
http://collidoscope.io/
[Accessed 23rd May 2016]

COLLINS, K. (2014). *Imogen Heap's music-manipulating gloves on Kickstarter.*
http://www.wired.co.uk/news/archive/2014-03/24/imogen-heap-glove-kickstarter
[Accessed 23rd May 2016]

DISABLED WORLD. (ND). *Physical & Mobility Impairments.*
http://www.disabled-world.com/disability/types/mobility/
[Accessed 23rd May 2016]

FRANCOISE, J. (2016). *Leap Motion Skeletal Tracking in Max.* (leapmotion.mxo external)
http://julesfrancoise.com/leapmotion/
[Accessed 23rd May 2016]

THE GLOVES PROJECT.
http://theglovesproject.com/
[Accessed 23rd May 2016]

KICKSTARTER. (2014). *Mi.Mu Glove for Music.*
https://www.kickstarter.com/projects/mimu/mimu-glove-for-music
[Accessed 23rd May 2016]

LEAFCUTTERJOHN. (2011). *Bigger, Harder, Stronger, Homemade Piezoelectric Crystals.*
http://leafcutterjohn.com/?p=1548
[Accessed 23rd May 2016]

LEAPMOTION. (2016).
http://store-eur.leapmotion.com/
[Accessed 23rd May 2016]

MI.MUGLOVES. (2016). *The mi.mu gloves.*
http://mimugloves.com/#TheGloves
[Accessed 23rd May 2016]

MICROSOFT RESEARCH. (2015). *Handpose.*
http://research.microsoft.com/en-us/projects/handpose/
[Accessed 23rd May 2016]

MYO. (2016). *Myo.*
https://www.myo.com/
[Accessed 23rd May 2016]

NCBI. (2013). *How does the hand work?*
http://www.ncbi.nlm.nih.gov/books/NBK279362/
[Accessed 23rd May 2016]

TLDP. (ND). *Maintaining a Project: Interacting with Users.*
http://www.tldp.org/HOWTO/Software-Proj-Mgmt-HOWTO/users.html#ALPHABETA
[Accessed 23rd May 2016]

VR TIMES. (2015). *List of Hand Tracking Tech for VR.*
http://www.virtualrealitytimes.com/2015/03/10/list-of-hand-tracking-tech-vr/
[Accessed 23rd May 2016]

Appendix I - Max MSP patches

Just a few notes regarding the opening and functioning of the Max patches. Firstly, and most importantly, to run the patch correctly it must be opened in Max MSP 7 or above, with the latest version being the preferred option (this is mostly due to new features such as the groove~ time stretching functions).

The parent patch hosts a number of bpatchers, poly objects and small utility objects, all built by myself. These all contain notes and descriptions and are colour coded and organised within the Max Patches folder as such..

Red – parent patch

This only regards a single object, the main Leap Sample patch itself. This hosts all other patchers and is the file used to create the standalone application.

Orange – bpatchers

These are the front-end objects for each main function of the application, such as recording or audio settings.

Yellow – audio_effects/granular_synth

These contain the poly~ objects used in audio synthesis and manipulation, such as the main granular synthesis computations or modulation effects.

Green – utility_objects

This folder is host to the minor objects that I used across the project, such as cross faders or control scaling.

Items marked in blue are the image files I created, for use as dials, backgrounds, overlays and instructions. Purple represents audio files, such as samples or the window file for the granular synthesiser. Lastly, grey represents miscellaneous files such as the preset file or the text files for the coll objects.

Appendix II – Using the Application

The main application file is located within the Application folder, this does not require Max MSP and will run as a standalone on any recent OSX system. The application itself contains instructions on to how to use it, loading by default (they can also be brought back up at any time by clicking the information icon in the top left). However, just a few notes regarding the correct operation of the software. Firstly, ensure you have the presets.json file located within the same folder as the application, this will allow you to read and write presets for later recall. I have included a preset file with a small number of presets, some of which were used in the creation of my videos, these serve as an example of the different forms of control and manipulation offered by the software as well a starting point for a user to create their own presets.

Along with recording audio, you can also drag your own audio files into the audio_samples folder located within the same directory as the application. This will allow you to select them from the drop down menu in the audio loader. Again, a small number of samples are included, representing a variety of sound material, some of which was used in my video material.