***­Preliminary Project Report***

***Introduction:***

Despite my love for them, I’ve never not found myself frustrated at the ridiculous number of controls on most synthesisers. Robert Moog’s original Modular synthesizers have hundreds of nobs and buttons on them in their most basic form, and depending on configuration, the number could rival the number of degrees in a massive circle; Goldsmiths University’s very own Roland 100m Modular has a figure worryingly close to that round number. Even in the age of soft synths, where the possibilities of intelligent interfaces are essentially endless, the most popular current products, Rob Papen’s Albino, GForce’s Oddity, have if anything increased the number of controls. I personally feel the dizzying array of controls on most synthesisers is frightening to those who are not familiar, and therefore means a lot of people will never try to learn even the basics of synthesis, and in the case of more experienced users, may be driven to give up before they reach the intricate details of sound design using synthesisers.

The idea for this project is to create a synthesiser that takes away the dazzling on-screen controls and replaces them with gestures, familiar and simple gestures, gestures that will hopefully, through research and smart programming, be intuitive and therefore breakdown the very real barrier between confusing user interfaces and fantastically designed sound engines. My long term vision is a piece of software that means players can go from a simple starting sound to whatever sound they can imagine, quicker than is currently possible, and in a way that is much more enjoyable and natural than is currently possible. Hopefully this project will become to synthesisers what gesticulating is to speaking, a way of showing what you really mean or want to achieve, without have to spell it out in mouse movements.

***Aims and Objectives:***

Overall, the aim of this project is to create a synthesiser that will be used primarily for production purposes, but can also be used for performance and teaching purposes. As the user will control the synth using their hands, playing the synth at the same time will not be possible, however I want to user to be able to quickly switch between gesture and note control (ie the controllers remain attached to the hand when playing the keyboard). Encapsulated within this main aim are the following aims and objectives:

* Aesthetically pleasing and ergonomic controllers
* A synth engine that can produce a solid array of sounds, not leaving the user wanting.
* Easy to use and informative GUI, I want the finished product to be a single program that the user can simply open and easily pair the controllers.
* The onscreen GUI teaches users how to use it the synth and what gestures do what.
* When the user opens the program, I want them to be greeted with a list of gestures and a small graphic of the gesture, that they can perform along with what these gestures do. As they move through the different sections of the synth the list of gestures changes according to section.
* Quick gesture response. The idea is that the synth becomes a way for people to create sounds on a synth quickly and easily, so taking a long time recognizing and categorising gestures isn’t really an option.

***Methods:***

* Adafruit Feather and Accelerometers, Bluetooth LE for communication between controllers and program. ADXL345 Triple-Axis Accelerometers. I might add gyroscopes if when further into the development it becomes apparent their added functionality would be greatly beneficial to the synthesizers capabilities.
* Machine Learning for gesture recognition and distinction (C++).
* Max MSP or openFrameworks & Maxim for the synthesizer itself, depending on the ease of integration. The main program will be made in C++, but I generally prefer using Max MSP as a way of creating a synthesizer, and so if this is easily possible I’ll do that.
* MIDI in, the notes will be controller by a MIDI keyboard.
* C++ and addons for all other aspects, ofxMaxim, ofxGui/DatGui etc.

***Project Plan and Description of Tasks:***

I’ve split the tasks for this project into four main areas:

1. Hardware design and communication
2. Synthesis design and parameter selection
3. Gesture design, selection and interaction
4. User Interface
5. Hardware design and communication:

Hardware need to fit all size hands. The communication and connection setup between the program and controllers must be simple.

As previously mentioned, as the user will be holding a controller in each hand, playing the synth at the same time is not possible (or desired!). Therefore, the controllers need to be designed so that they are attached to the hand, allowing the user to remove their grip on the controller and use their fingers to play the key board.

With my current level of programming, I’m finding it really hard to get the Adafruit data straight into Xcode. I know it is possible and I want to negate the need for any other software to keep things as simple as possible.

1. Synthesis design and parameter selection:

I’ve looked at the controls found on the most popular software and hardware synthesisers and deduced what controls are found on the majority. From this I’ve divided the synthesis design and parameters into 5 sections: Oscillator Section, Filter Section, Envelope Section, Arpeggiator Section, FX Section. I need to look further and do more research into the specific parameters in each section.

I’ve already chosen to have only FM synthesis and Additive synthesis, which Additive the default and primary method of synthesis.

1. Gesture design, selection and interaction:

One of the big problems with using Machine Learning and synthesis control is that giving every single parameter its own gesture would mean having to distinguish between a large number of gestures, and the more gestures you have, naturally the difference between gestures will decrease, making it much harder to have a high accuracy algorithm.

Given this, I’ve decided to break the synthesiser into sections. While the user is in each section, a particular gesture will be mapped to a parameter or control, but in a different section that same gesture will mean a different thing. Using this I’ve drawn a ‘Gesture Tree’ (see below), that I think means I only need to distinguish between 5-6 gestures maximum to control all areas of the synth.

Further, I’ve come up with the idea of ‘Select’ and ‘Control’ gestures. Select gestures will use the Machine Learning algorithm to differentiate between gestures and be used to define discrete parameters, ie create sinewave oscillator, or saw wave oscillator. Control gestures will simply use the raw data from the accelerometers to control continuous parameters, ie envelope attack or release time, or filter cutoff for example.

In terms of tasks for this area, I need to get stuck into the Machine Learning side to figure out which gestures are most distinguishable from each other. I also need to work out an order of sections, so that movement through the synthesiser is intuitive and natural for the user, as well as finalise a list of the parameters I want to include.

The start of each gesture will be controlled by either clicking one or both of the controllers. The controllers should be released at the end of the gesture. Doing these means I can use a time based approach to logging XZY values. I’ll also need to find the optimum gap between each reading so that the program is both fast and there are enough readings to distinguish between gestures.

I will also need to work out a way of calibrating the orientation of the controller at the start so the gestures match the machine learning database.

1. User Interface:

A quality of implementation of most of the above is pointless if the user interface is hard to understand and difficult to use. I need to do a lot of research and selection into gestural controller interfaces and see which ones I like and think are easy and natural to use.

As the function of gestures are not initially as obvious as a button with ‘Arpeggiator On/Off’, the GUI needs to be clear in explaining what gestures control what at each point. The idea with this synth and its HCI is that eventually users will know off by heart what each gesture does.

***Project Schedule:***

***February 13 – 19:*** Complete a working prototype of one of the controllers and establish a way of getting the accelerometer data into openFrameworks and the other programs I’ll be using. This is one of the most important steps, without the controllers working I can’t much meaningful work elsewhere. This week I’ll also need to decide what gestures I’m going to use.

Also: mucho research.

***February 20 – 26:***Once the data is coming from the controller/s to the IDE, I need to begin calibration, normalization, setting up scaling so the values I’m getting are workable. Once that’s done I need to generate a dataset for my chosen gestures, and start creating a Machine Learning algorithm to classify each of the gestures.

Also: mucho research.

***February 27 – March 5:***Continue working on gesture recognition and the Machine Learning algorithm. Hopefully by this point I’ll have some ability to distinguish between gestures. Start draft final project report.

***March 6 – 12:*** Start working on the synth engine. Begin final project draft. I need to have made a pretty decent plan about how I’m going to build the synth as the gesture tree (structure of synth movement) will decide how I program the synth in terms of order of programming.

***March 13 – 19:*** Continue synth engine and gesture machine learning stuff.

***March 20 – 26:***Synth engine needs to be finished by this point; shouldn’t be too hard given I’ve studied and practiced this kind of stuff for the last 2 and a half years. Once it’s finished I can begin mapping the gestures to the synthesiser parameters.

***March 27 – April 2:*** Take the synth out of the serial and start developing a GUI.

***April 3 – 9:***  Keep perfecting the gestures and machine learning and its integration with the synthesiser controls. Carry on working on the GUI and the general flow of controls.

***April 10 – 16:*** Work on the Bluetooth from this point. Hopefully I can have both controllers working well, with solid gesture recognition and control via serial. Also, have sorted out the mechanism of moving forwards and backwards through the synth depending on what section you are in. Having the Bluetooth on the way from this point is essential. If it isn’t going well I need to organize a sit down with Phoenix or Pete, someone who really knows their Bluetooth LE/Objective-C++ stuff.

***April 17 – 23:***

***April 24 – 30:*** Leaving these two weeks free as its fairly unrealistic to assume I’ll keep on track with the above schedule, these weeks to the finish line will hopefully be uneventful but inevitably won’t be.

***May 1 – 7:*** Testing and evaluation, get at least 10 people to have a go with the synth and ask the questions that I’ve

***May 8 – 14:*** Design a 15 minute presentation of expresSynth. Talk about the reasons why, influences, what research told me, how my initial design changed, how I planned to make it, how I made it, the techniques used, what problems I came across and how I overcame them, show how I ended up with it, give a demonstration, what would I do differently in the future, say thanks.

***May 15: SUBMISSION!!!***

***Bibliography:***

Some initial papers I’ve read through at so far:

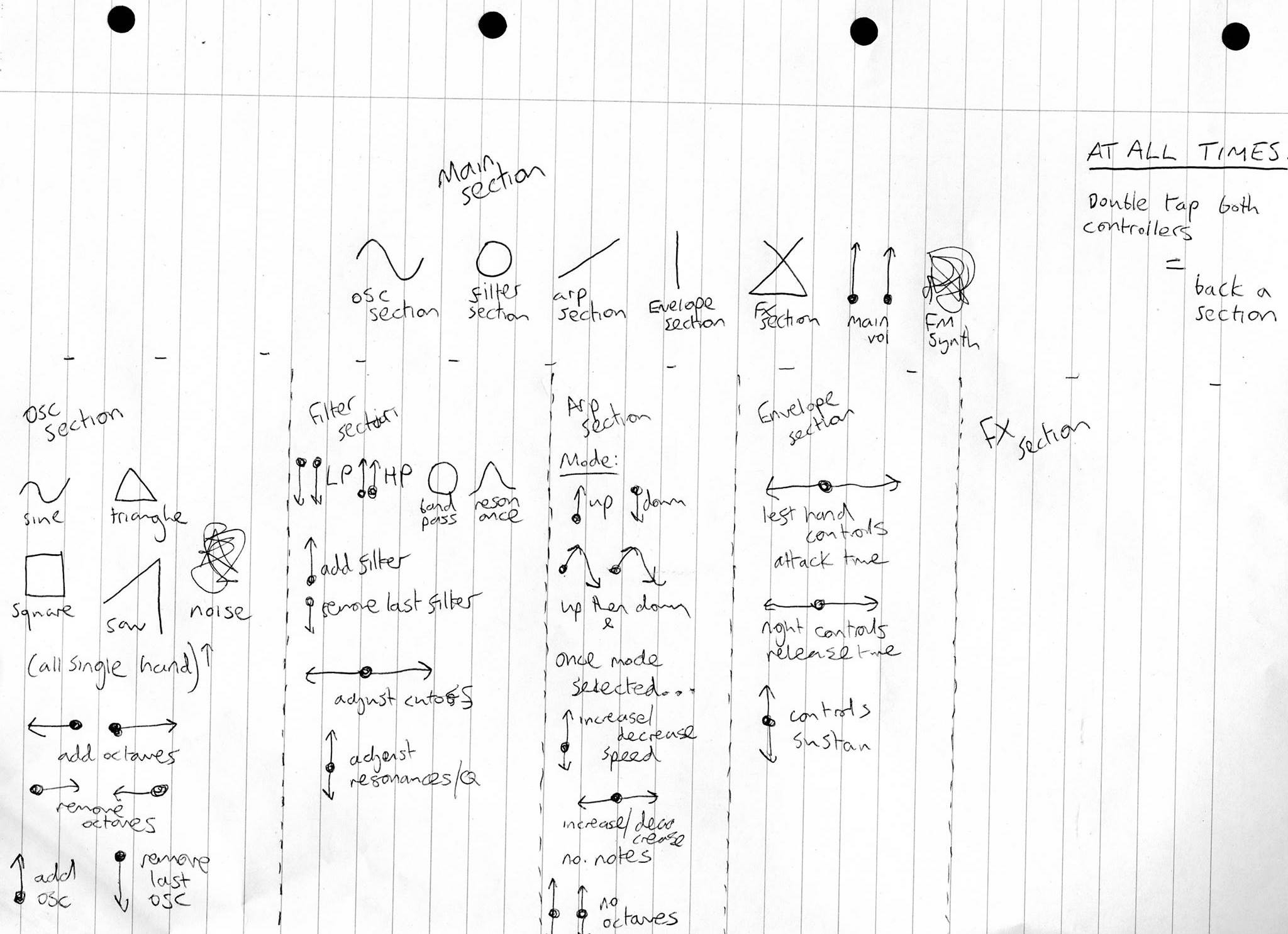
<http://www.music.mcgill.ca/~mwanderley/MUMT-615/Papers/Class02/P.WanDep.pdf>

<http://www.nime.org/proceedings/2016/nime2016_paper0059.pdf>

<http://www.nyu.edu/classes/bello/DCMS_files/Silber_Cho_proposal.pdf>

***Gesture Tree (below):***

Below is a basic sketch of my ‘gesture tree’. In keeping with what I mentioned earlier that having a different gesture mapped to each and every function would mean too many gestures, I’ve divided the functions into sections and drawn simple lines and shapes depicting what I think each gesture might be. I chose the gestures as they should be easily distinguishable, particularly given you can use either the left, right or both controllers, allowing three times as many possible controls than just one controller.



Introduction, as the one above I like it. Maybe add some about why this project suits me (focus on the interface and design side) (c. 500 words).

1. **Introduction**

Despite my love for them, I’ve never not found myself frustrated at the number of controls on most synthesisers. Robert Moog’s original Modular synthesizers have a huge amount of nobs and buttons on them in their most basic form, and depending on configuration, the number could go into the hundreds; Goldsmiths University’s very own Roland 100m Modular has a figure worryingly close to that number. Even now, in the age of software synthesizers, where the possibilities of intelligent interfaces are essentially endless due to very limited programming and design constraints, today’s most popular current products, Rob Papen’s Albino and GForce’s Oddity to name a few, have if anything increased the number of controls. I can’t help but feel that the dizzying array of controls on most synthesisers is frightening to those who are not familiar, and therefore means a lot of people will never try to learn the basics of synthesis, and in the case of more experienced users, may be driven to give up before they discover the intricate details of sound synthesis using software applications.

The idea for this project is to create a synthesiser that takes away the dazzling on-screen controls and replaces them with gestures, familiar and simple gestures, gestures that will hopefully, through research and smart programming, be intuitive and therefore breakdown the very real barrier between confusing user interfaces and fantastically designed sound engines. It is also to create a synthesizer with a modern interface, less cluttered than most of the top synthesizers that are out there today, and also to create a tool that not only acts a tool in production and sound design, but can teach people what the controls are and what they do to sound, in an enjoyable and straight forward manner. Further, there’s no reason that if the project works, the foundation of the gestural control system could be used with other applications.

I believe that what I am trying to achieve and the techniques required suits the skills that I have learnt and integrates the aspects of software programming, digital signal processing and design that I have found myself both enjoying the most and succeeding at the most throughout my three years at Goldsmiths. I have consistently performed well and enjoyed DSP, particularly in creating and modelling analogue synthesis and sound design, while I fell that as a fairly hands-on person adding a physical aspect to the project gives an opportunity to uses skills and thought processes, physical design and HCI for example, that wouldn’t be required in a purely code based project.

My long term vision is a piece of software that means players can go from a simple starting sound to whatever sound they can imagine, quicker than is currently possible, and in a way that is more enjoyable, more intuitive and more memorable than is currently possible. The technological world is evolving and developing at a rate that is only likely to increase in the future. The world’s biggest technology companies are now more than ever focusing on improving the cohesion and relationship between humans and computers, and having taken a programming degree with heavy focus on music, creating a tool that offers a different way of interacting with and controlling musical software is a logical, interesting and fitting project to undertake.

In short, I believe that the current synthesiser user interfaces are both outdated and an obstacle, their cluttered displays are a hindrance to both creativity and expression and I feel sooner rather than later an alternative method of control needs to be developed.

Further description of why what has been summarized in the introduction is a problem, give examples, outline similar products that already exist and explain how my idea is a different and helps to solve a different problem. What approach I am going to take and what the output will be (c. 1000 words).

1. **The Problem Posed**

Compared to more commercial applications, like mobile telephones, laptops and tablets, music production software has fallen behind in the race towards the perfect interface. In 2004 the most advanced telephone interfaces were slow, unintuitive, awkward and confusing. We are now in 2017 and the latest iPhone operating system is so familiar and intelligent to use that most people could probably operate them without looking. Unfortunately, the 2004 Logic Pro 7 interface is worrying similar to the current edition. The same can be said for the almost identical Sculpture synthesizer interfaces, seemingly unchanged despite 13 years of development. Rob Papen’s Albino software synthesizer has hardly changed interface despite 3 renditions. While many of the world’s essential digital devices have seen constant developmental changes and improvements to their human-computer interaction and interface, the same cannot be said for most musical software.

Scrolling down on a phone or laptop used to be the dull process of many button presses, the same could be said for keypresses on a laptop. Nowadays you scroll with a flick of a finger. Increasing the pitch, or changing the waveform on a software synthesiser used to be the press of a button, well, it still is. Would a similar flick of a finger or wave of a hand not be a more futuristic and natural approach?

My intended approach to this problem is to look into the gestural systems that already exist, both in the musical sphere and outside, in areas such as gaming and television control, and analyse how they work, their successes and failures, and what kind of implementation might realistically work. I intend on building a hardware controller for a piece of software that includes the necessary gesture recognition algorithms along with a comprehensive synthesizer and the accompanying user interface.

Many of the most well-known gestural systems today, such as Microsoft Kinect, utilise the techniques of computer vision in combination with machine learning to achieve the goals, however in my opinion, despite the advantages of using computer vision system, a hardware controller offers a more achievable alternative given I will be able to choose the components and the respective data they’ll provide, rather than dealing with the unpredictability of a real life scene.

In order for it to be a product that can be evaluated to a degree where it can be concluded whether or not it is a genuinely viable alternative to the current available software synths, I intend it to be a prototype, not a finished product, with the focus not being on obtaining an extremely high level of accuracy in gesture recognition, or on a fully accomplished hardware controller and synth engine, but with all of the aspects and completed and combined to a level where it can be trialled and tested, and its intentions obvious enough to allow a view towards future developments and hopefully a final product.

Influences, origin of ideas and research into gestural controllers, gestural interaction with music, synth interface design, synth engine design and music teaching tools. Emphasis on justification, look at examples, their advantages and disadvantages and make a decision as to what I want to take from those examples. Influences in terms of synth sound, appearance, interaction, everything (c. 1500 words).

Synth and Sound design influences:

<http://theconversation.com/sublime-design-the-moog-synthesiser-26460>

<https://www.attackmagazine.com/features/interview/dave-smith-synth-evolution-midi/>

<https://pdfs.semanticscholar.org/0727/218882b17241abc2380694577862859763a2.pdf>

Interface design influences:

<https://www.researchgate.net/publication/228720346_Synthesizer_user_interface_design-lessons_learned_from_a_heuristic_review>

<http://www.synthtopia.com/content/2011/11/17/the-strange-agency-isnt-afraid-to-design-a-synth-with-a-mind-blowing-user-interface/>

<https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/>

http://www.cs.uml.edu/~grinstei/91.510/Papers/p45-chang.pdf

Gesture and controller design influences:

<http://www.nyu.edu/classes/bello/DCMS_files/Silber_Cho_proposal.pdf>

<http://www.nime.org/proceedings/2016/nime2016_paper0059.pdf>

<http://www.trustedreviews.com/samsung-smart-tv-voice-and-gesture-control-systems-review-gesture-control-page-2>

<http://www.imogenheap.co.uk/thegloves/>

<https://eprints.soton.ac.uk/261149/1/GestureTaxonomyJuly21.pdf>

<http://www.nime.org/proceedings/2016/nime2016_paper0039.pdf>

<https://core.ac.uk/download/pdf/16413069.pdf>

Music Teaching tools influences:

Synth game type things maybe:

PRIMARY AND SECONDARY RESEARCH

*Maybe look for inspiration quotes from designers about forwarding design (apple, Microsoft, publishing and graphic designs, sound designers about forwarding sound designs, synth designers about forwarding synth designs.*

*With a focus on my belief that design and creative, as a completely free entity is constantly innovative, and improving in all ways, only differing by preferred styles, explain this shit much better with quotes n pictures n shit.*

*Things are becoming less hands on, telephones, Apple Watch, Kinect, VR, having to manually do things is past ting.*

*More things are gestural and natural, I believe eventually you’ll never have to type or manually press things etc. etc. etc:*

*Cluttered synthesizers:*

*Why a teaching tool? Would be good for more people to learn so more people could get involved in the industry, from music production to actually building and creating the synthesizers.*

*Genuinely find a better way of interacting with synthesisers, and a way that you don’t have to be taught or read a manual to work out.*

*I personally feel the task in hand fits my skill set.*

*Even though the number of controls on even the most basic synth will still be fairly high, and that there will have to be at least some representation of each control on the screen somewhere, the idea is that anyone who uses a piece of software or hardware will eventually be able to use it, the on screen gesture prompts will eventually be learnt, and therefore the user will not need an interface whatsoever, whereas even if someone becomes a complete expert at Albino synth for example, the cluttered UI will still be there, and could lead to confusion.*

**3. Research**

There are quite a number of fields that I need to consider to ensure I understand why and how products have been successful, or failed in the past and how I might shape my design. The key areas I’ve looked into are previously successful examples of and my own favourite implementations of software DSP synthesis and sound design, intuitive and ingenious user interface designs, the actual gestures that are found in existing products, to give me an insight into what is actually possible, the physical controller design, and the general interaction of gestural systems as a whole.

**3.1 Research into Gestural Systems and Controllers**

Beginning with gestural systems as whole, the one that really stood out to me, perhaps unsurprisingly, was the Nintendo Wii. The Wii was, and still is an incredibly successful product, selling 101.63 million units since its 2006 release, with over 30 million units alone shipped in 2008 1. Its design hasn’t changed in almost a decade and its continued popularity and market dominance has meant no similar product has been able to take a significant hold in the gestural game sector. Microsoft’s Kinect add-on for their Xbox range of consoles has sold 24 million units since its release 4 years ago; 4 years after the Wii’s release it had shipped 84 million units.

The key difference between the Wii and the Kinect, and why it stands out to me for this project is that where the Kinect uses Computer Vision, a technique that extracts and interprets data from a camera, essentially automating the ability of the human eye in the digital world, the Wii uses a less sophisticated method, utilising a physical accelerometer in combination with standard buttons and infrared communication to send raw acceleration data to its hub, where that data is computed and transformed into on screen reactions.

There are advantages and disadvantages to both, however for my intents and purposes, the methods that suits best is the Wii method. In my opinion, computer vision is too complicated for an application such as this. Using a camera that observes the real world means your data is subject to the randomness and variability of the real world, for instance someone walking past the screen, different people having different hand sizes or way of gesticulating, or different backgrounds behind the user.

There are of course problems with the Wii method, the key one being that different users use the controller in different ways, giving different data that needs to be mapped to the same function. This problem can be broken down into smaller ones, device orientation, gesture speed, gesture size, specific gesture shape, all that can cause the pattern recognition to completely fail if not accounted for, but if approached correctly can all be solved relatively easily.

The Wii isn’t however the only handheld controller to consider. Another piece of hardware that has received attention in recent years is mi.mu’s gesture controlled gloves, developed and most notably used by electronic artist Imogen Heap. The glove, like the Wiimote (Nintendo’s official name for the actual controller part of its Wii system) uses accelerometers and other data sensors that are then manipulated and mapped to different functions. The idea of a glove for music control doesn’t originate from mi.mu’s and Heap’s 2014 glove, one of the original commercially released examples was from Nintendo who brought out their “Power Glove” in 1989. The Power Glove however was considered a commercial failure, it was too big and uncomfortable for its possibilities4.

Imogen Heap’s glove is still in development stage, 28 years after the release of the Power Glove, showing that this approach to handheld controls is still a long from replicating the commercial viability and success of the Nintendo Wii. My personal opinion on gesture controlled gloves is that they prove too complicated to build and have too much potential to be ugly and uncomfortable. Further, the act of putting on, taking off and just wearing a glove is quite prohibitive due to its appearance alone.

**3.2 Research into User Interfaces**

The main aim of my project is to find an alternative to the cluttered and confusing user interfaces that are found on many of today’s most popular software synths. Graphical User Interface design and Human Computer Interaction are two huge topics of research and the quality of implementation on products, be it mobile phones, ATM’s, desktop computer applications or anything else is key to their success.

Apple’s design principles cite clarity, deference and depth as the three primary themes that differentiate their iOS design from other platforms 4. For clarity they further explain that text is legible at every size, negative space, color, fonts, graphics and interface elements subtly highlight important content, while for depth they say that distinct visual layers convey hierarchy, impart vitality and facilitate understanding 4.

The reason I cite Apple’s design principles is that as an Apple product user I feel their UI design is fantastically clear and understandable, and the above principles, if followed correctly and cleverly should lead to a successful interface. Of the example soft synths I pointed out earlier, Rob Papen’s Albino, G-Force’s Oddity and Apple’s Sculpture, along with hardware alternatives, in particular the modular synths, the Roland 100m for example, it can be said that Apple’s design principles are not adopted in their design. The one that stands out to be the most is layers. Almost all software synthesizers use only one layer and cram all of the controls into that one space. There are very few examples where a menu is available and you can delve in and out of the oscillator or filter sections for example, with that section filling the screen, allowing clearing presentation of the available controls. Manipulating virtual knobs and sliders on a screen with a mouse or even with a multi-touch tablet is awkward. While the ‘virtual hardware’ look may be appealing, it’s too fiddly to perform with5, let alone if all of the synths controls are squashed into an 800x600 pixel space.

Regardless of what gestures I end up using, they need to be explained in a way that is unambiguous and clear. The depiction of each gesture and the way that depiction is interpreted is of paramount important to the success of the program. Nintendo and Microsoft both utilize simple, cartoon-like animations to show what action the user must perform. Other programs use real-life video clips while some others use worded descriptions. Given the resounding success of the Wii and Kinect platforms the obvious choice for me to use is the cartoon animations. User interfaces are often based on static presentations, a model ill-suited for conveying change. Consequently, events on the screen frequently startle and confine users. Cartoon animation, in constant is exceedingly successful at engaging its audience; even the most bizarre events are easily comprehended6. Cartoon animations, although they can be as intricate and elaborate as you wish, are most commonly fairly simple, including only the required details to depict what they require. This there, is the route that I feel would best suit my UI design. Additionally, stylistically I feel a UI with cartoon animations is more fun than the serious dark and metallic designs found on most of today’s soft synths.

**3.3 Research into Synth Design**

As much as I want this to project to find a new way of controlling a synthesizer, I also want the synthesizer to be a comprehensive and satisfying tool for music production. Through my own use of synthesizers along with some examples that have stood the test of time, the Minimoog for example, I think you can deduce a certain number of features you need to have a synth that can make an array of sounds complete enough for most users.

The two synths that I want to design my synth around are the before mentioned Minimoog and also the Roland Jupiter-6. Both of these synthesizers have two Voltage Controller Oscillators, one Voltage Controlled Filter, a single LFO and two Envelopes and develop a wide array of sounds from fairly few controls.

Reasoning – Is this sensible and viable given my research. What evidence can I give to suggest it might be success and not obsolete. Essentially sum up the research and influence section (c. 500 words).

*This could be a kind of prior research section.*

*Maybe I should ask some producers if this is something they might use.*

*There is obviously a possibility that at the end of this I find myself with something that works, is quite fun to use, but is practically useless. In the situation, at least I will have included and developed in areas that I think separately may have a place in further work or developments, ie the synth engine, I’ve hopefully created a synthesizer that can be used standalone with manual controls because it has a strong array of sounds. In terms of the gestural control, I might find that it’s place is not with software synthesis but with other applications, like cooking, or walking the dog. I’ve enjoyed the UI design, and I think it could provide inspiration for design in other products of areas of design, for example when we have smart newspapers that utilise my sexy animations.*

Design, build, software, hardware, technologies, technicalities, methods. Explain my choice of gestures and how many gestures I could reasonable differentiate between. Explain the interface design and link back to points made earlier in the research section, explain the controller design linking back to research, why wireless or not etc. etc. etc. Lots of diagrams, circuit drawings, (c. 1500 words).

*Synth technique and design, link back to research in what signal path I design, what synths I wanted it to be like, explain all the details, give signal path diagrams and compare to Moog/Jupiter. What range of sounds does this potentially offer? Is it capable of offering interesting and different sounds, yes, filter modulation is important to create moving, interesting sounds, otherwise they can end up sounding rather similar and dull? Allowing unlimited tremolo depth allows FM synthesis to an extent.*

*Controller design. Why did I choose Bluetooth? Explain the signal path, what accelerometer did I use and why, why didn’t I use a gyroscope/magnetometer as well, because I had enough detail, want to remove variance and improve PCA for Machine Learning algorithms. Originally had 2 controllers but realised variance would be too high and as such just had one motion controller and one button controller. This doubles the amount of possible controls. Having two buttons on one controller essentially achieves the same thing, makes the product potentially cheaper, easier to connect, less likely to lose etc. Wii has one controller (plus nanchuck if necessary), PS4, xBox One, TV’s, phones, all one controller. Peterson Slide Viewer. Best option I had between a AirWicks controller, a bottle of effervescent Vitamin C tables, the Slide Viewer and a Wireless Mouse – Put the picture up here.*

*Workflow of the synth, why is it split (less shit on each screen, more manageable) into sections and what sections are they. This is to minimise the number of gestures to differentiate from, because of this the maximum gestures I need to distinguish between at any one point is 5, considerably easier than 10 or more. Why have I gone for multiple screens, moving backwards and forwards between them, mainly because soft synths commonly don’t utilise this ability and cram it all on to one screen.*

*Include code snippets or the most important stuff oooooh yes.*

*Machine learning and data from the controller. What algorithms did I use and why, how did I optimise and/or calibrate the algorithm for better accuracy. Did I use all three values from accelerometers? Research into Machine Learning applications on accelerometer data for past experience.*

*Turns out I didn’t use ML. I designed an algorithm that calculated the total distance between the shapes, based on the x and y acceleration values at identically spaced increments.*

*Talk about the vectorNormalization function and incorporate code from it.*

*Do the same with the serialIn function and all that stuff.*

*User Interface design. Took inspiration from my influences and research. Talk about the GIFs and why the animation side of things works. Talk about the patterns and waveform section and why they are there and what they do for the interface and interaction i.e. keep the interface from being static and still.*

*WHY HAS MY RESEARCH INFLUENCED THESE DECISIONS?*

**4. Implementation**

* 1. **Controller Design and Build**
  2. **Connection and Data Parsing**
  3. **Gesture Selection**
  4. **Data Manipulation, Normalizing and Comparison for Gesture Recognition**
  5. **User Interface Implementation and Program Structure**
  6. **Synth Implementation**

Maybe further research undertaken after the initial design and prototyping (c. 500 words).

Controller shape and design – initial prototyping has made me realise that without having some type of mathematical or gyroscopic calibration, having a controller that everyone can hold in the same way is absolutely vital to the success of the gesture recognition algorithm.

Shapes, a lot of clashing, the sinewave clashed with the circle wave as in terms of acceleration, there are much more similar than they appear. This in ML is called a latent factor, I think. This was also the case for the bandpass shape, this shape clashed with more than one of the other shapes. The most common misevaluation was between the bandpass and the sine, but the bandpass and the arp was also similarly problematic. As such I removed the bandpass movement entirely as it was a little bitch.

After calibrating the shapes every time I opened the program, I decided I need to evaluate what the best option would be for the final program, a pre-programmed and hard coded dataset contained in a txt (or other) file in the program that cannot be altered, or calibration per user at the start of the program. Each has different advantages and disadvantages, but these depend on the expected accuracy of the gesture recognition on a dataset that I have drawn, against another user’s gestures. If the current operator calibrates the system, it will be much better suited to the little differences in the way they interpret the gestures prompts/instructions, as well as other small, but important differences like the way the user holds the controller.

I guess this would promote testing between each of the shapes individually and draw up a table of the difference values to show which ones work best together.

In terms of testing for the calibration and I guess initial value testing. I needed to formulate the best, most efficient and most rewarding/appropriate method of testing. I decided that the most important factor to work on, in terms of testing and what I might be able to then do with the data I get, testing for different type of calibration was the most important, as that would hopefully show if the product is actually viable for multi-person usage, and not that I’ve overfit the algorithm to my way of using it, while I was programming it.

As such I decided to test all 11 shapes, performing each shape twice against a calibration (therefore getting 22 values – one each for the different between the shape performed and each of the calibrated shapes). For 11 shape that then makes that 22 number up to 242, I will then do 5 different calibrations, meaning 110 tests, producing 1210 values, from I should then be able to deduce the sets of shapes that will work best together, given the distance between them is highest, therefore increasing the likely success of the algorithm and program.

Display the table of results, the conclusions and ultimate which gestures I will have on each of the red and blue control buttons. Maybe put this section up in the design and build section.

Problems, issues, things that weren’t possible in the end and how they were overcome (c. 1000 words).

Hell of an issue with the Bluetooth and serial in, this made me think about connection and device choice in the future, perhaps wifi might be a better option, of maybe to try a different technique altogether use a Computer Vision approach and eliminate controllers altogether, like my original project proposal. In terms of the controllers, my soldering/PCB-ing and other electronic skills aren’t fantastic and maybe in future, for a prototype project where the functionality of the software rather than the aesthetic of the controller is the most important thing, perhaps using a Wiimote and an associated library will have cut out a lot of time, leaving more to spend on the manipulation of the data, perhaps leading to a more efficient, accurate and enjoyable application.

Machine Learning, if you can call it machine learning as there isn’t really any learning going on. I have a data set and I’m doing comparisons between that data set and the incoming gesture. The application doesn’t actually improve with an individual user’s ongoing usage. Perhaps incorporating this in the future might again provide a more accurate response from the program.

Variance – talk about, document, and present the data in terms of changes to the g-force rating and the hz data stream rate

Talk about little changes I’ve had to make to the shapes i.e. reversing the circle so that it isn’t to similar to the sine wave.

Having one controller instead of two, primarily because of my inability to work with BLE, and the fact that that means having a USB Midi Keyboard plugged in as well as the ExpressController plugged in at the same time means most laptop’s USB port supply will be exhausted, some may even be exhausted at one, which would mean in the future, adding BLE connectivity would be absolutely vital.

Calibration, I’m going to need everyone to do each of the control gestures as a calibration stage at the start, this is highly necessary, as otherwise variance is gonna be nuts. HOWEVER, if I test with other people on my gesture calibration and on theirs, and then compute the accuracy, see wagwan, make a table etc etc etc.

Testing, evaluation and questions to users. Get at least 10 people to answer questions about the synth, sum these up and then action any changes that have been apparent given the replies to the questions. List of questions (c. 2000 words):

*If you have to draw a sine wave and a saw wave, what you do?*

*Take a look at these cute little gifs, what do you think these instructions intend you to do?*

*Do you like the interface or do you find it confusing, annoying, clever? Voice your opinion baby.*

*Were you able to produce a solid amount of sounds that you liked with relative ease?*

*If you’ve used software synths before (which you definitely have), would you say this synth is more enjoyable to use or not?*

*Did the whole system run smoothly?*

*Get people to think of a synth sound and try to make it using the synth and then ask how it felt for them.*

*Synth sound, UI, controller appearance, controller feel, overall feel, time factor, loading time, clarity of instructions and controls.*

Things I would do differently in the future and why. What have I learnt from the project and would I undertake a similar task in the future? Have I achieved what I set out to achieve (c.1000 words).

Conclusion (c 500 words).

ALSO – NEED A VIDEO, PLAN A GOOD PRESENTATION FOR PRES DAY, MAYBE A LEAFLET WITH SIMPLE INSTRUCTIONS N SHIT.

Bibliography

1 "Wii Sales". *En.wikipedia.org*. N.p., 2017. Web. 7 Apr. 2017.

2 "Wii Remote". *En.wikipedia.org*. N.p., 2017. Web. 7 Apr. 2017.

3 “Gesture Recognition and Interaction with a Glove Controller”, Gerard Llorach Tó, University of Adelaide, Web, July 2013 (http://gerardllorach.weebly.com/uploads/2/6/7/1/26710587/gesturerecognitionandinteractionwithaglovecontroller\_gerardllorach.pdf);

4 "Design Principles - Overview - Ios Human Interface Guidelines". *Developer.apple.com*. N.p., 2017. Web. 7 Apr. 2017.

5 Power, Combining et al. "The Strange Agency Isn’T Afraid To Design A Synth With A Mind-Blowing User Interface". Synthtopia. N.p., 2017. Web. 7 Apr. 2017.

6 “Animation: From Cartoons to the User Interface”, Bay-Wei Chang, David Ungar, Web, 7 Arp. 2017.

7

8

9

10

Full code.