***­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:***

***May 8 – 14:***

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

***Bibliography:***

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

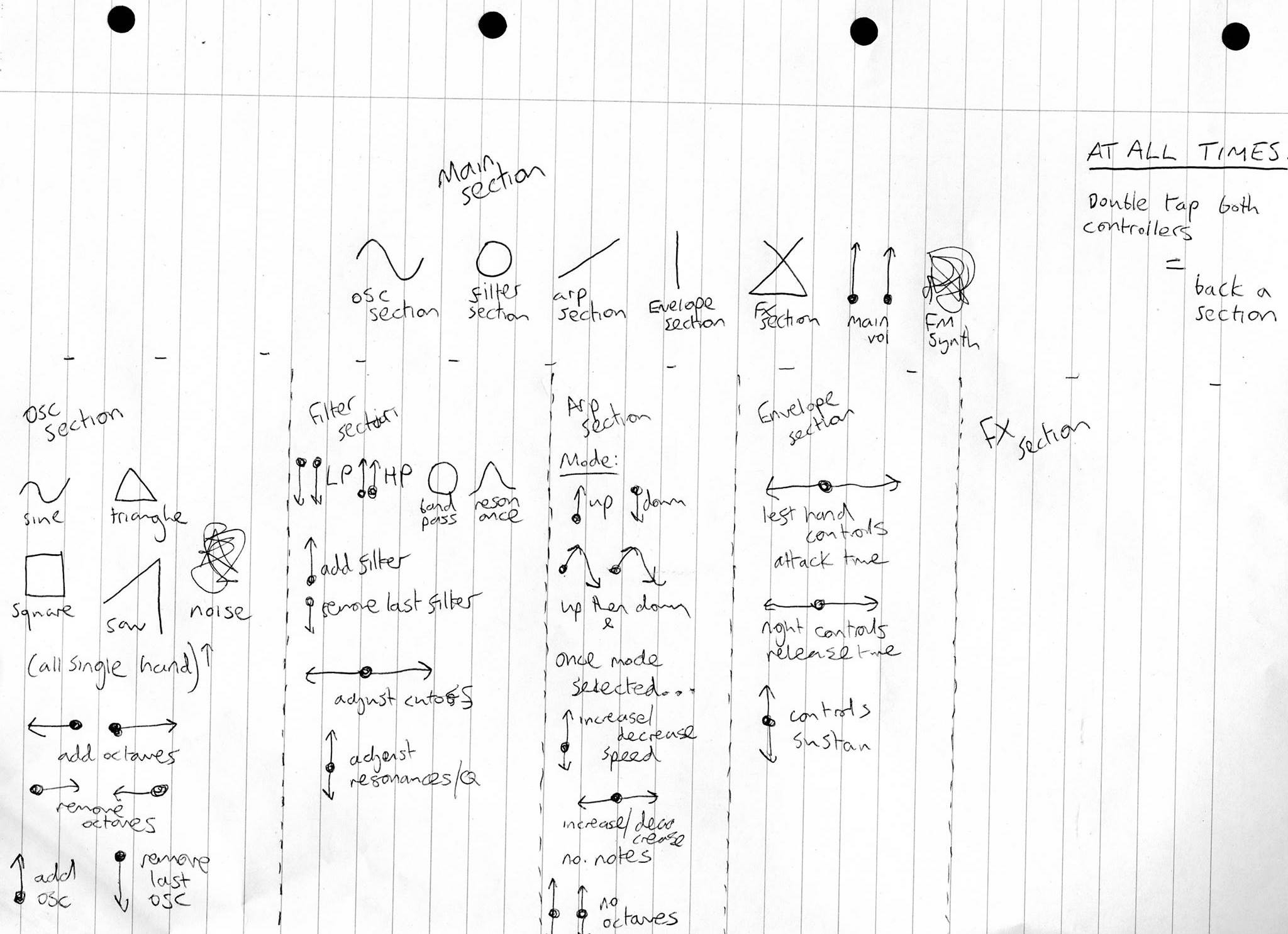
<http://www.music.mcgill.ca/~mwanderley/MUMT-615/Papers/Class02/P.WanDep.pdf>

<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).

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 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. 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.

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 having to spell it out.

Objectives and why I have chosen this project, what am I trying to achieve, who is the target audience and why, is it a production or performance tool and why. How do I think this synth might improve synthesisers? As it is a production tool, have I asked producers if they might use it? (c. 1000 words).

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 moving towards better things, 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.

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

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.

Workflow of the synth, why is it split 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.

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.

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?

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

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.

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).