TempoVR - Virtual Reality Synthesis

Aaron Lynch

N00182364

Supervisor: Timm Jeschawitz

Second Reader: Joachim Pietsch

Year 4 Thesis 2021-22

DL836 BSc (Hons) in Creative Computing

Abstract

The aim of this project was to construct a VR based system which allowed Users to experience and create digital music in Virtual Reality, with a major focus on synthesis and digital sequencers. There were a few different reasons behind the rationale for this application, one was that currently there are not many music creation/interaction applications out there for Virtual reality headsets, and those that do exist are either very technical/specific or do not actually offer true creative freedom to the User to create something unique. The purpose of my application is to give Users of any musical skill level a chance to learn about musical sequencers and use them in order to create original sounds/songs. The application will also allow Users to interact with a real digital synthesiser in VR, this will allow the User to have complete freedom in altering the sounds they create on a synthesis level which in itself is very powerful.

In order to develop a VR application of this kind I will use the Unity Engine as it is very VR friendly and also works extremely well with the oculus platform which is the main target device for the application. Alongside this a native synthesiser plugin for Unity called HELM will be used, this will allow access through C# script to use/alter the native synth from inside the engine. To gain knowledge of the process of development I would need to follow, I first began by collecting requirements for the app and investigating similar applications out there. Following this a Design document was put together outlining the visual and backend design, this included creating wireframes, colour palettes and environment designs. From there the implementation of the application began, this was an ever evolving stage as new things were designed and implemented during this time to address new issues/solutions in the application design. Testing phases were carried out through the development of the application in order to assure that the features being developed were heading in the right direction, then following implementation some further final testing was performed to bug test and gauge User reactions to the application. Testing was generally performed in person with friends,family and housemates which provided me with excellent insight into how the application was progressing.

Finally looking back over the project, there are definitely more things I would like to add and also different approaches that I would take to developing certain aspects of the Application. Firstly implementing a more robust recording system would be high on my priority as the current system works really well but lacks freedom to browse though your collection of recordings. Something I would approach differently is using a framework to manage my VR fundamentals such as hand interactions/animations, levers,switches and dials as this was all created from scratch by myself and in turn detracted development time from the core Idea for the application which was the creation of music.

Acknowledgements

Firstly I'd like to express my sincere gratitude to my supervisor Timm Jeschawitz, for helping to guide and encourage me through this busy and difficult year. In particular I am very appreciative given that the situation with COVID-19 was still very much unknown near the beginning of the year causing students and lecturers alike to once again adapt. Timm always ensured that I had the help or advice I needed by having weekly meetings where we would review my work and what still needed to be completed. It helped greatly that Timm also shared personal interest and experience in the field I had chosen to base my project off and this was evident through the guidance I received. I would also like to offer special thanks to my second reader Joachim Pietsch, for providing me with a great look into what I needed to improve on and helping me keep a clear view of what needed to be completed going into the next semester. His outside perspective offered great insight into possible new features and his in depth knowledge of VR technologies also aided with idea's to fully utilise all aspects of the Virtual Reality toolkit. Finally I would also like to thank Mohammed Cherbatji for organising the project sprint layout and guiding our year through the new approach towards the project that the college was taking.

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your own files on a hard disk where they can be accessed by others. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student : N00182364

Signed:

Aaron Lynch

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction

The aim of this application is to provide a Virtual Reality space for Users to create digital music, that is approachable and intuitive allowing Users of all skill levels to create and learn digital music creation. Currently there are not very many music creation applications based in VR, this would mean that my application targets a mostly vacant area of Virtual Reality where there is a need for innovation. My application aims to take a slightly different approach too by mainly focusing on sequencer based music, harking back to the digital music sequencers of the 1980's like rolands 303 or 909 machines. These devices are reimagined in VR for my application alongside real synthesis to give Users access to an era of music/technology they may have missed. This is particularly relevant as many of these machines have seen a resurgence in use in digital music over the past twenty years but alongside this so too have the cost price of instruments like this. This creates another advantage to my application as it removes the barrier to entry created by the cost/accessibility of these devices.

The aim of this report is to explain clearly each phase of the development process. This started with gathering the requirements needed to create the application. Here I studied users to determine what their needs from the application would be and also researched relevant similar applications out there to determine what was already currently working and what could possibly be improved on in current music creation applications both in VR and out. After this was the Design phase where the applications design and environment was created based on what we had learned from our requirements phase. Many different planning techniques were used during the project in order to ensure that development kept on track.

My application Tempo VR has been mainly built using a combination of the Unity game engine and Audio HELM's synth plugin. These technologies were heavily used during the Implementation phase of this project to create the application. This report shows what steps were taken in order to create both the backend and frontend design. Also, this report shows the functional testing that was carried out in order to ensure that the features and functionalities I had implemented were working as

intended. This report also detailed a lot of the difficulties faced during development and how I as the developer resolved these issues. Finally in this report I give a reflection on my experience in the project and what maybe could have been changed or done differently in order to improve the application.

2 Requirements

2.1 Introduction

The product I will be developing for my final year project is a Virtual Reality Application that aims to allow Users to interact with and learn about digital music creation/synthesis in a fun environment. This application will involve Users creating and arranging sequenced loops driven from a Digital Synthesiser engine, allowing users to create music in a much more interactive environment. While there are not too many currently existing musical applications for VR, there are free very powerful applications that allow users to synthesise or arrange songs in VR. But in my opinion they are not approachable to those not already versed in digital music creation. This is where I hope that my application can differ itself as it will aim to provide a music creation experience that doesn't require prior knowledge and functions more similar to a game; where the User learns as they interact with the environment and objects presented in front of them. I also feel that while it is a more approachable style of application for the general user and could serve great to introduce new Users to music creation, the application could also offer a great opportunity for those who already currently create digital music. This would create a space where they could use audio samples they have created and perform and manipulate them in VR, this offers a whole new side to the enjoyment of creating music.

To help me figure out what different features users would require I first looked into other similar applications to see what services they provided and how they aimed to stand out to Users. I have also created user personas and will be conducting interviews to help me further connect with general user requirements of the project. With the aim to collect data that would help me further understand people's use of current music creation apps and VR apps I created a survey.

2.2 Requirements gathering

2.2.1 Similar applications

42Tones SynthVR:

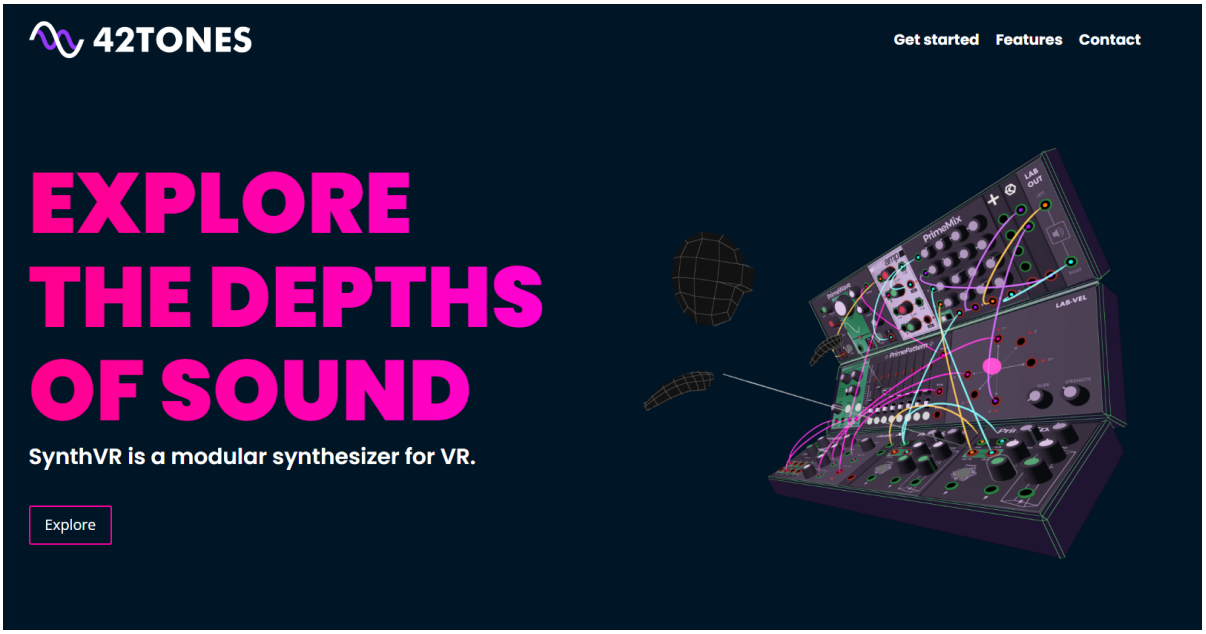

Fig 2.1 Image of 42Tones application.

SynthVR by 42Tones is currently one of the only true examples of synthesis in VR. In SynthVR, you can enjoy a modular synthesiser experience enhanced by VR, enabling you to touch, feel and see sounds you create in new ways (42tones, 2022). The application gives the User access to a range of components that operate identical to real life modular synth components that would normally cost a lot of real life money. This offers Users to create music in VR with real musical synthesis with the User having complete control over the sound.

Fig 2.2 Example of a modular interface inside Synth VR by 42Tones

To understand what makes SynthVR so unique, it is necessary to understand what modular synthesis is. Modular synthesisers are synthesisers composed of separate modules that perform different functions. The modules can be connected together by the User to create a patch. What makes modular synthesis so exciting is that you can pick and choose exactly which oscillator, effect, or other module will suit your needs and build your own unique synth from scratch, where the building blocks are nearly limitless. This Application offers synthesiser enthusiasts the ability to explore this unique side of electronic music in a much more practical environment as normally modular synth racks can take up large amounts of space and require different physical setups for different sounds.

In relation to my own project, there are many similarities between the experience SynthVR is offering and the experience I intend my own Application to offer. SynthVR offers Users the feeling that they are in control of every aspect of the sound they are creating, bringing the User as close as possible to the process of creating digital sound/music. I want Users in my application to feel like they too have many different parameters that they can use to alter the sounds they create, giving them true control over their music. Where I feel my Application will differ is in its approach to providing this experience, SynthVR while extremely powerful and true to real life can be extremely overwhelming and complex even to those who are knowledgeable in the field. It provides a raw experience that suits many enthusiasts but would most definitely scare Users who are new to creating electronic music away. My Applications aims to provide a more intuitive and interactive experience with similar qualities of music creation to SynthVR but with tools and a world that anyone will find accessible, fun and easy to use.

Advantages:

- Allows the User a huge amount of control over the sounds they create from beginning to end.

- Functions exactly like real world examples of modular synths, so prior knowledge will apply here, also what you learn in the application can be applied in the real world if you were to use a modular synth.

- Allows collaboration with other Users/creators, where you can share and work on musical ideas together.

Disadvantages:

- Requires a large pool of prior knowledge to understand how modular synths work, if new to modular synthesis then the application will be very overwhelming.

- In order to learn how to use the different modules a User will most likely need to read up on the various functions of them as just using them in the application is not enough to understand how they work.

- Due to the complex nature and pure freedom of modular synthesis, many users may feel trapped, and unable to create the exact sound they are looking for given how random sounds produced by modular synths can be.

- No free trial, so Users must purchase the application not knowing whether modular synthesis is something they would be interested in.

Tribe XR:

Fig 2.3 Cover image for Tribe VR DJ School.

Tribe XR is another VR Application that I found during my research that shared quite a few similarities with the Application I intend to create for my project. Tribe XR is a VR DJ application which also helps you learn a little about music mixing and live performing as a DJ. It comes with preloaded samples which you can use to edit and remix tracks, you can also import songs from your desktop to edit and play with also if preferred. While the Application mentioned before SynthVR focuses more on creating the sounds; Tribe Xr focuses more on the mixing and performance elements of music creation.

There is a tutorial to show you how to use each dial, button and slider. The load in screen has an environment with professional Dj decks you can use to fade between tracks, change volumes, pitches and even scratch the decks. There is a multiplayer mode to play with friends or to join one of Tribes virtual DJ lessons where an actual instructor will come in and teach a class on how to DJ. This multiplayer element is

definitely something I would be interested in trying to implement in my own Application.

Tribe XR is an Application that is quite similar to what I aim to make in the way that It is more for a fun interactive experience, rather than a technical piece of software focusing solely on musical synthesis. Tribe XR is a much more approachable experience for those new to music creation and mixing, and incorporates a fun interactive approach to learning. But this is also where the limitations of Tribe XR are apparent as it does not provide an environment where you can create new music from scratch. It only provides the mixing, performance and playback aspect of music creation. In my project I believe that a mix of real musical synthesis similar to SynthVR and the interactive performance elements of Tribe XR will create a perfect Application for new or experienced Users to create, share and enjoy electronic music.

Advantages:

- Huge library of built in tutorials and guides making the Application very easy to use.

- Functions exactly like real world DJ decks would, so Users can learn how to perform and mix songs accurately.

- Very large and active online community, with plenty of different multiplayer game modes to enjoy.

Disadvantages:

- Very limited if your goal is to actually create your own music.

- Users may struggle to use populars songs they like and want to mix with due to copyright laws.

- Quite an expensive application given its limited functionality, when looking at digital music creation in general.

- No free trial, so Users must purchase the application not knowing whether modular synthesis is something they would be interested in.

2.2.2 Interviews

I have conducted interviews with 2 users to find out what the important features for them for the app are. There can be various issues that arise in interviews so these will be noted and grouped together into a document.

Interview Questions:

1. Have you ever played a VR game/Application?
2. Have you played any musical or rhythm based games in the past? Name them if so.
3. Have you ever used any digital music creation software to create or play your own music?
4. Have you ever used digital or physical synthesisers before?
5. Does the idea of music creation in VR interest you?
6. When creating music in VR would you prefer to
    a. Create and learn from a technical perspective.
    b. Create and learn in a fun interactive environment more similar to a game.
7. Would it interest you to be able to create/experience music with others in VR, rather than a solo experience?
8. Please select 3 of the following features that you feel would be most important to VR music creation applications.
    a. The ability to export songs/sessions you have created.
    b. The ability to Import your own samples for use in sessions.

2.2.3 Survey

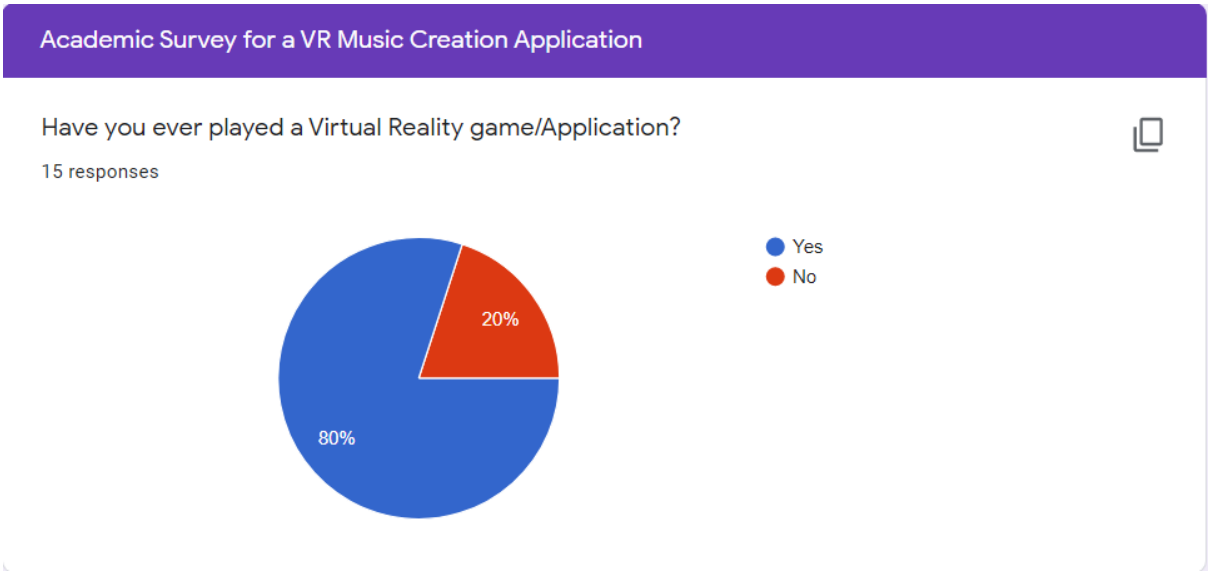

Fig 2.4 Survey Results

The developers created a Survey to find out information for the Application idea. I started off with a simple rangefinder of have you ever played a VR game or Application, eighty percent of my survey had so I knew I could use a good portion of the data.

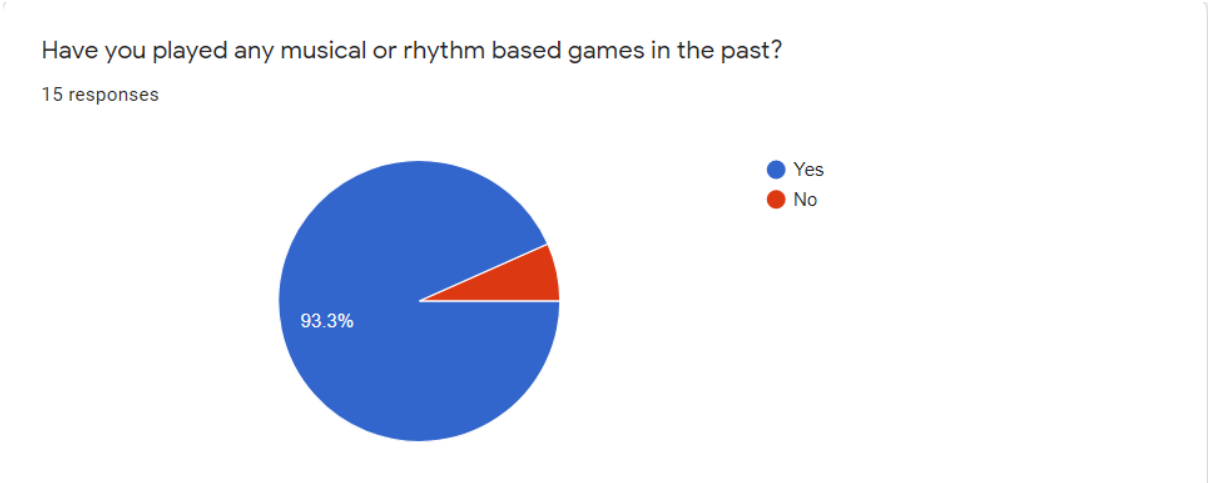

Fig 2.5 Survey Results

Whilst the Application I am developing is not a rhythm game, I felt it a good question to gauge peoples interactions so far with musical games and applications. Following this the participants of the survey were asked to name any if they had played any musical or rhythm games, allowing me to see what is already popular with people.

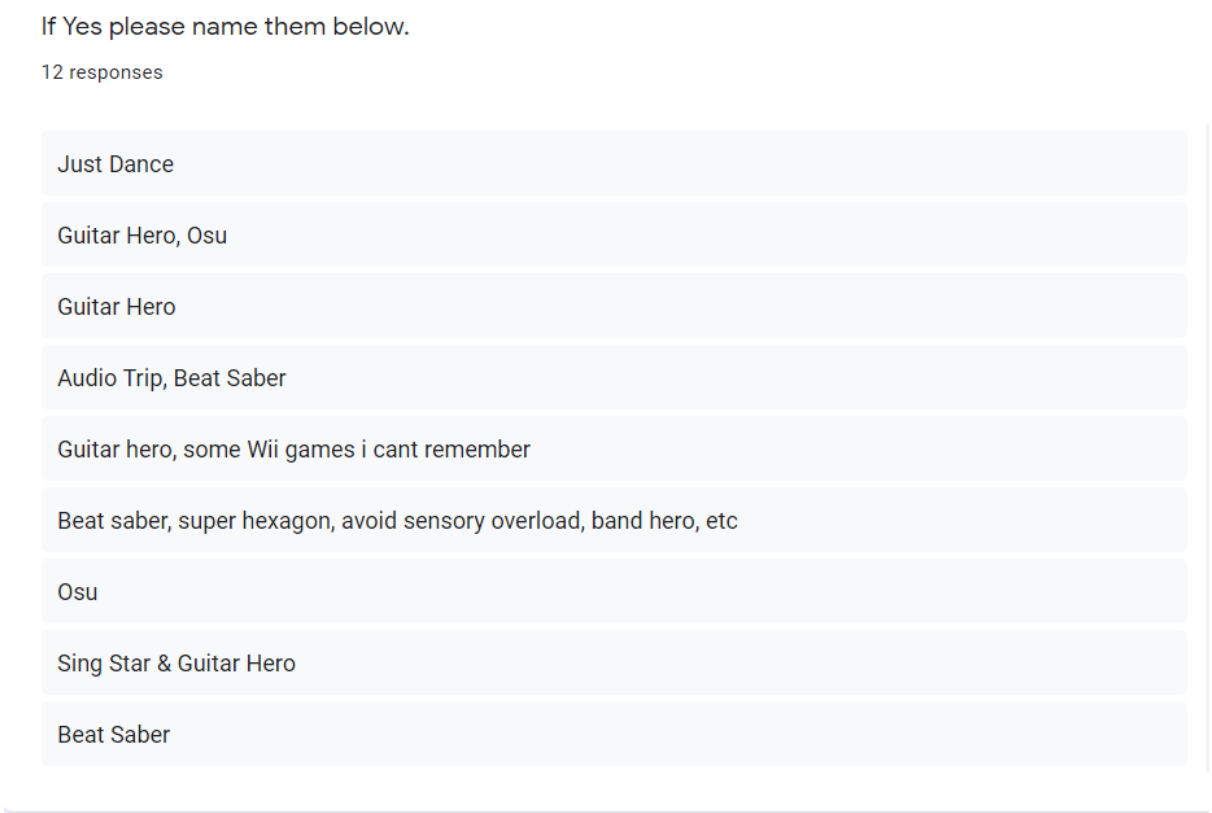

Fig 2.6 Survey Results

Following this It was next important to gauge the general public's experience with using any digital music creation software to create their own music and name the software used if true. Surprisingly over half our participants have actually used digital music creation software, this provides great confidence that applications for music creation definitely have a demand with the professional but also the non-professional environment.

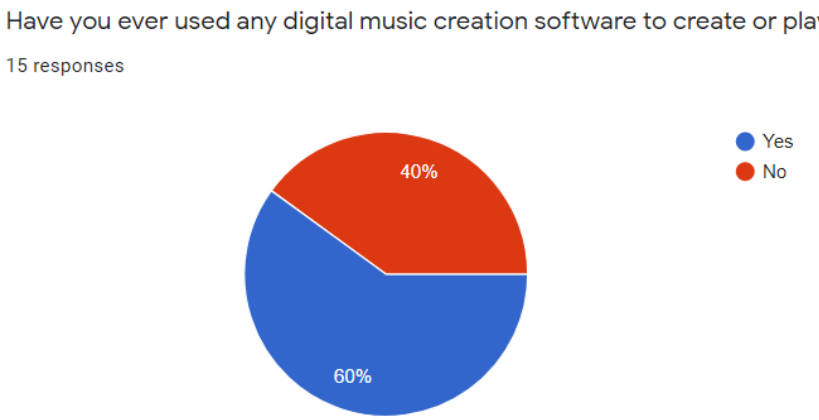

Fig 2.7 Survey Results

Fig 2.8 Survey Results

Next participants were asked if they had ever used any digital or physical synthesizers before, only just over a quarter had, which informed us that not many people have had that experience. This fit's the goal of my Application perfectly as one of its intentions is to introduce new people to synthesisers.

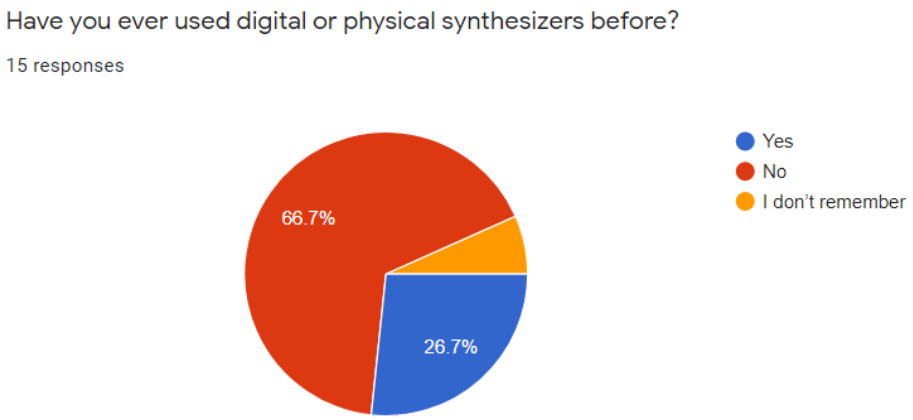

Fig 2.9 Survey Results

Next a scale based question ranging from one to five was asked to gauge the possible popularity and interest in an Application like this, thankfully it scored very high in most people's survey. With only 1 applicant showing no interest.

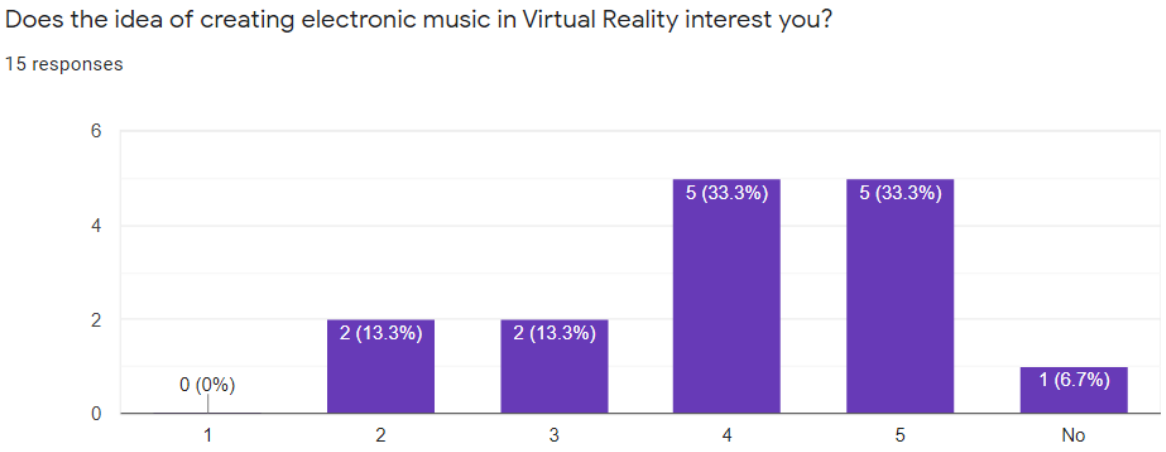

Fig 2.10 Survey Results

With this next question it was my goal to determine what direction the application should take with regards to how Users would create, learn and interact in the Virtual World. The results showed that people by and far preferred the idea of learning in a game-like environment with fun being the main focus rather than a more technical environment.

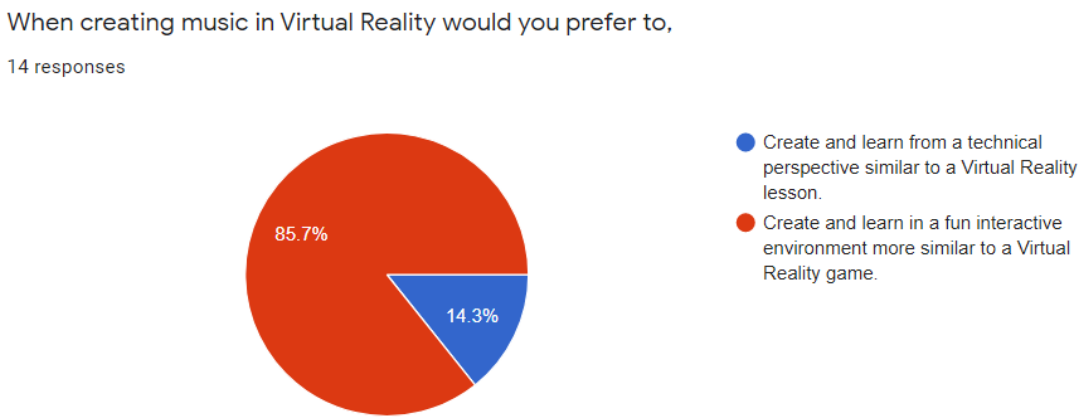

Fig 2.11 Survey Results

Next I felt it was important to determine if a Multiplayer route would suit a project like this, a whooping 73% said yes. Hopefully this is something that can be implemented during development as most Users would clearly gain more enjoyment from the application if they could experience it with friends.

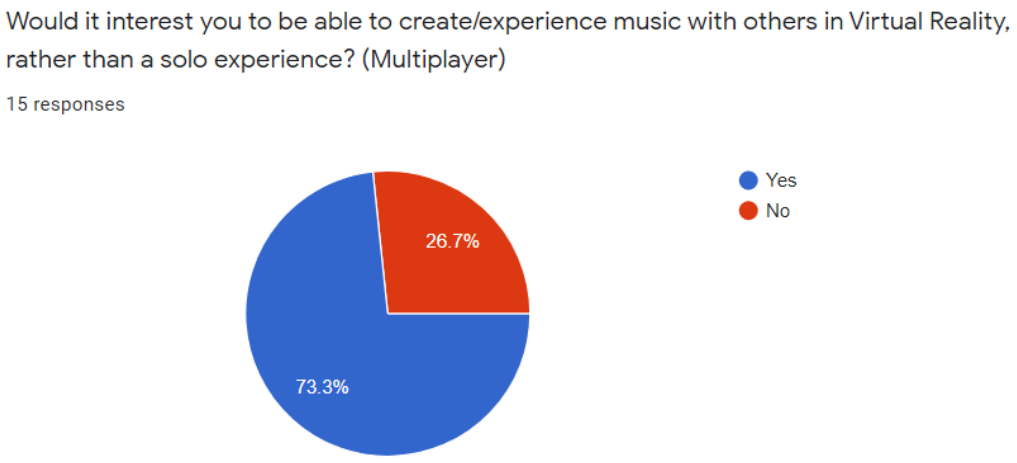

Fig 2.12 Survey Results

In the next question participants were asked to select three out of the following five options based on importance to the application. This will help to decide what aspects of the development process should be handled first to ensure their inclusion in the project.

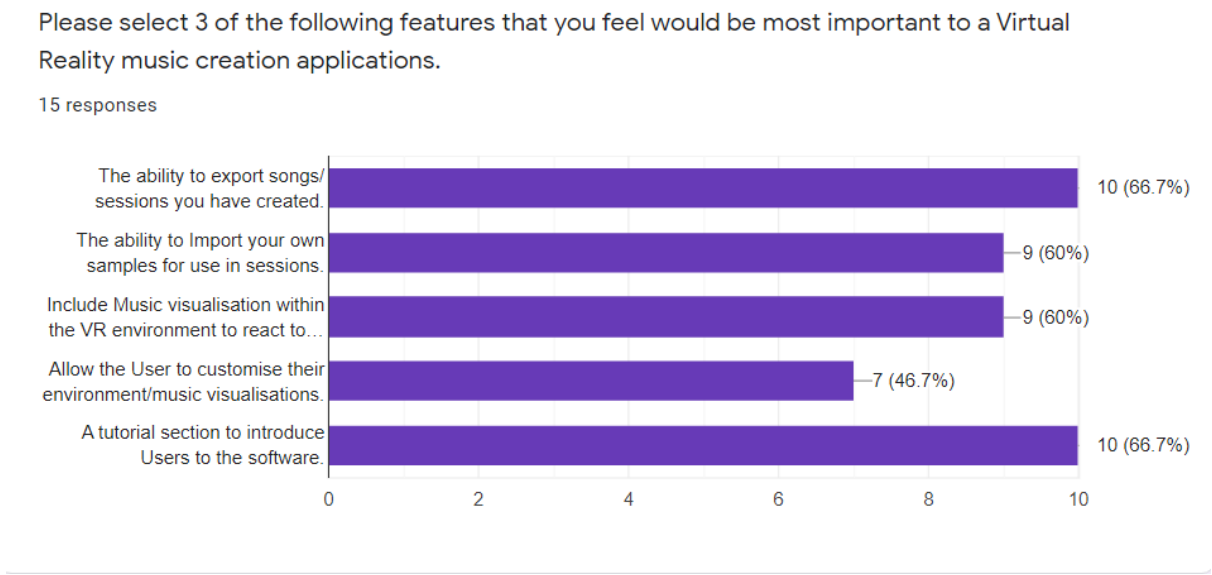

Fig 2.13 Survey Results

Finally participants were asked if they had any other suggestions for or comments about the Application, to leave them in the comment box at the end of the survey. This can help the project by possibly hearing new suggestions for the app or comments suggesting possibly a better approach to a feature in the app.

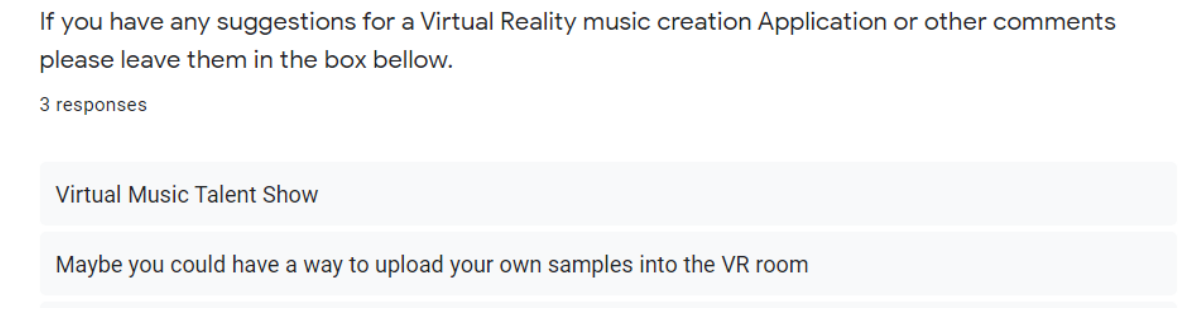

Fig 2.14 Survey Results

2.3 Requirements modelling

2.3.1 Personas

Persona 1:

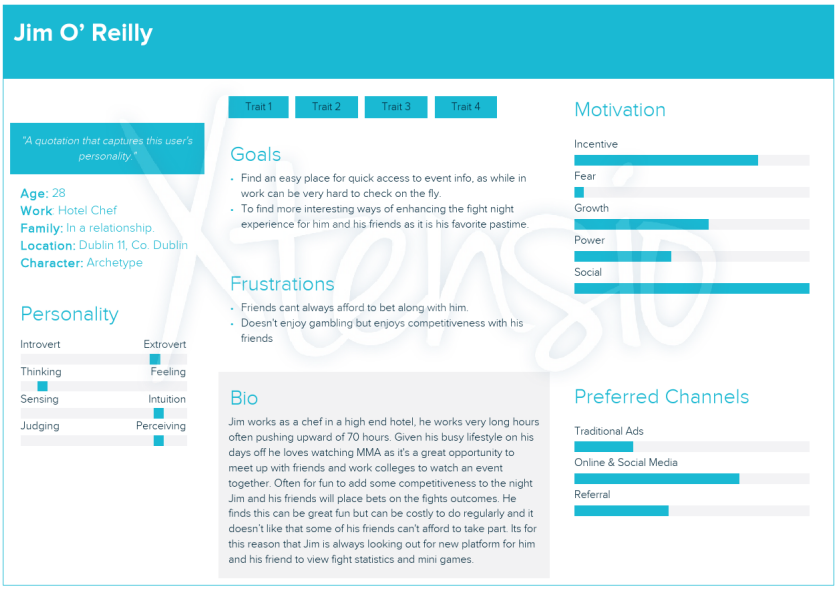

Fig 2.15 Persona creation

Persona 2:

Fig 2.16 Persona creation

2.3.2 Functional requirements

#	Functional Requirement	Priority
1	Allow User access to 4 Channels for creating their tracks; Chord, Melody, Drum and Bass.	High
2	As a user I want to be able to view information about how the various tracks/loops operate.	High
3	Allow the user the ability to use premade loops to create basic tracks for each channel.	High
4	Introduce Helm synthesiser to project allowing admin full controls.	High
5	Using Helms sequencing code allows users the ability to place samples into sequences.	High
6	Also allow the possibility for note selection for each note on the sequencer for the user.	Medium
7	Allow users to record a session so they can export their creations.	Medium
8	The ability to play multiplayer sessions over the internet.	Medium
9	Allow users the ability to add their own samples into the application.	Low
10	Also allow Users to export or Save loops they have created.	Low

2.3.3 Non-functional requirements

#	Non-Functional Requirements	Classification
1	Having the VR application compatible with all VR Headset models.	Usability
2	If allowing User samples, only allow audio files to be passed to the application.	Security
4	Application will need descriptions of how various aspects of the application work.	Usability
5	Applications looping/sequencing systems should be lag and delay free.	Reliability
6	Colour coding system to allow the user to navigate channels in an easy fashion.	Usability

2.3.4 Use Case Diagrams

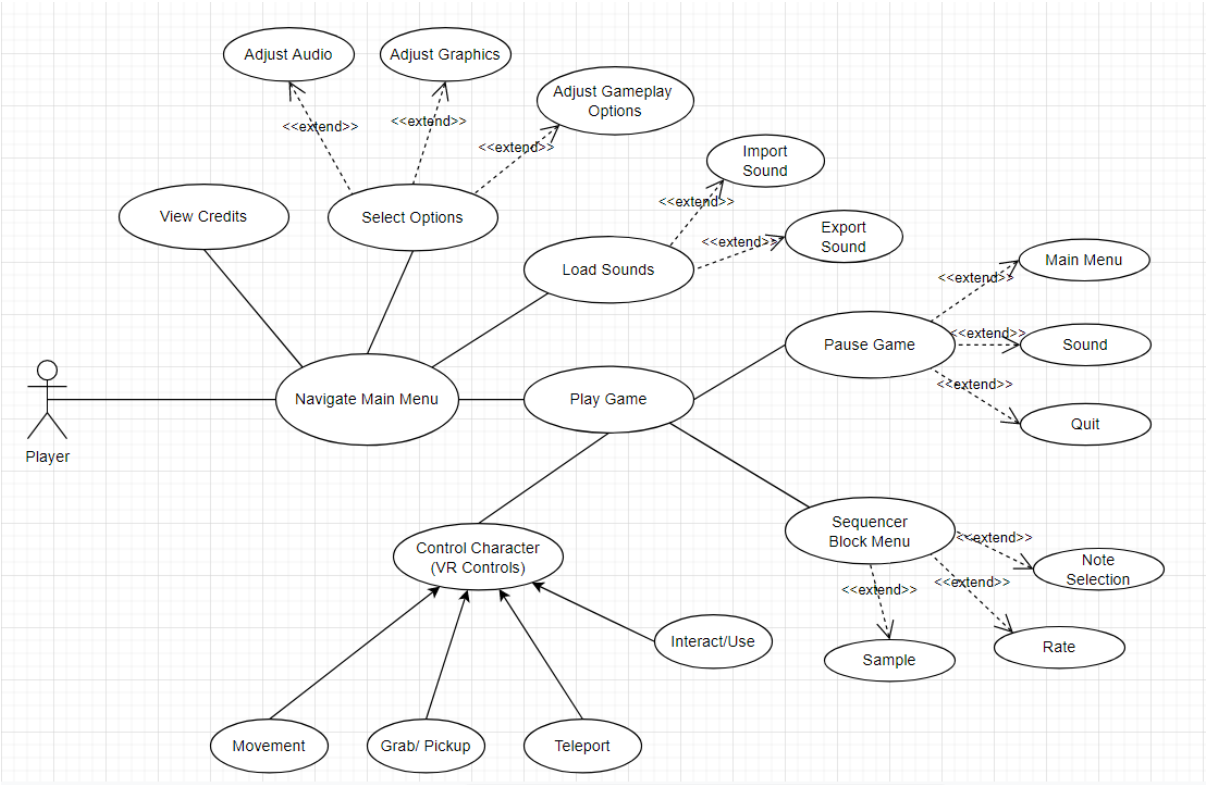

Fig 2.17 Use Case Diagram that was created to show a users possible paths within the application.

2.4 Feasibility

The following technologies are what will be used in the development of the Virtual Reality Application, Unity Game Engine, XR development Kit and Audio Helm's Unity synthesiser plugin. The Unity Game Engine is a cross-platform game engine developed by Unity Technologies, while it covers many platforms it has become particularly popular in VR development due to its easy to use toolset and VR plugins. Unity gives allows creators the ability to create games and experiences in 3D, with the engine offering a primary scripting API in C#. Audio Helms Unity synthesiser is a full-featured native synthesizer, MIDI sequencer and sampler. This plugin will allow the project to give Users access to interacting with real musical synthesis in VR. This plugin can be used in order to keep a sequencer clock from within a Unity scene and also allow objects within the VR world to be used to interact with the synthesiser.

Challenge	Description	Solution
Lack of experience developing VR applications through Unity.	Unity, while a very approachable engine/framework, is completely new to me and I possess zero prior knowledge in Unity. So overcoming early learning curves will be important.	One of the most supported platforms for educational tutorials. So many free learning courses to choose from making unity much more approachable.
Time management	Trying to keep up with various deadlines through development could be hard. Certain parts of development cannot start before others are completed. Making the timeline of the project quite busy.	Good planning and use of activity boards can help keep the project running smoothly and on time. Weekly meetings with my supervisor can also serve as a method of keeping on track.
Use of real synthesiser sequencers/ sequencer code is completely new to me also and not as well documented.	Due to the intention to introduce a real sequencer clock to keep all tracks on time, this will require a lot of research and time to implement as once again I have no experience in audio software creation and there is not as much information out there on the internet relating to it.	Fortunately a synthesiser plugin known as HELM exists and the documentation for it is very strong. This will help a lot in the implementation of the HELM plugin.

2.5 Conclusion

The Requirements part of the project gave me a more in-depth idea of my targeted users and increased my knowledge of the non-functional and functional requirements for creating a VR music creation Application. This section has helped me make a detailed list of my goals for the Application and conclude certain aspects of it, especially design-wise.

After producing interviews and getting an insight from the Users and publishing online surveys, the users' needs were appreciated and noted. I was given valuable insight and ideas of features that could be helpful and used to improve the Application. For example, one User in my online survey suggested the ability to upload and use your own audio samples to use inside the world, this prompted me to alter my questionnaire to gauge interest in a feature such as this. The support for this feature to be included turned out to be very strong which informed me that this would be a vital part of the app to improve the Users experience.

3 Design

3.1 Introduction

The application for this project is mainly focused on two things, Musical creation in VR and Electronic synthesis. The idea being developed for this project is a Virtual Reality application focused on digital music creation, this will allow users to play with and perform on various electronic synths/drum machines and alter the parameters of these electronic instruments to create a more physically interactive experience than traditional digital music creation. The target users for an application like this many would expect to be just those musically inclined but that is contrary to the aim of the application which is to provide an approachable digital music creation experience where users of all skill levels can enjoy and create from within the application. In this document, opportunities will improve the current necessary information gathered to build an application that will meet the requirements that were determined from the previous document. This chapter is divided into two sections: Program Design and User Interface Design.

The Program Design section is where the VR app's key structure and features are made. This will entail an overview of the technologies that will be used to make the application. This section will silhouette the app's creation and use of Unity and the design pattern that the app has. It shows the entities of the app and the relationship belonging to them. The Class and Sequence diagrams are also included in this section, which will help the creators making the app. The Entity-Relationship Diagram of the application is another important part of the Program Design.

The User Interface Design section will pilot the developer in creating a useful user interface of the app. This section will give a better description of how the user will be able to navigate between the options, menus and sections of the application. An overview of the design for each menu will be displayed and the use of wireframes to help better show their layout. In the Design section aspects such as the UI's colour

palette and typography are decided to help create an easy to interpret user interface while still keeping the design interesting and engaging.

3.2 Program Design

3.2.1 Technologies

The technologies being used to create this application are:

- Unity
- Helm

The technology Unity was chosen because it is considered by many to be one of the most powerful and approachable game engines out there to develop with. Unity provides developers with the tools to develop and create games applications quicker, more stable and make it much easier for multiple developers to work on the project together. Unity comes equipped with its own physics engine, a testing editor for fast testing, great UI creation tools and the Unity asset store allowing developers to find any asset they may need or to share their own created assets with other developers. Unity also provides animation controls, audio mixing, particle systems and optimisation of a systems CPU and memory usage.

For the implementation of the synth and sequencer aspects of the project HELM will be used; Helm is an open source software based synthesizer that can be used to create electronic music from a computer. What makes Helm stand out from many other paid/unpaid VST synths, is its ability to work and run natively in many different operating systems/devices. This has allowed a Helm plugin to be developed for Unity which allows the ability to run the Helm native synth from within a Unity project. This is very powerful as running the synth locally from within the game engine will cause there to be nearly no performance issues or lag due to the synth.

3.2.2 Structure of Unity

Unity is a 2D/3D engine and framework used by game developers to design and create indie and AAA games, however it is also known to be used in other creative areas such as applications, animation and art. This software is not only powerful, it is user friendly and free to use, with cross platform support.

Unitys highly customisable rendering technology, as well as its various tools aid in the creation of high quality visual effects. Unity is great for beginners as it is user friendly and easy to learn, you don't need to know how to code using C# (or JavaScript); However having an understanding of it does make the process easier. Unity includes video tutorials on the home page - as well as online tutorials and learning resources on their website which makes it an easy engine to learn how to use effectively even for beginners. Unity is a very popular engine; especially in the realm of mobile gaming - with titles such as Pokemon Go, Call of duty mobile and Beat saber created in this engine.

All Unity games work on a scene based system, everything that runs in your game exists in a scene. This means that a finished game is just a collection of scenes that interlink. In order to see anything within a scene there must always be a camera for the Users view. Nearly everything within a Unity scene is a GameObject, this is the base class for all objects in a Unity scene (see figure 1).

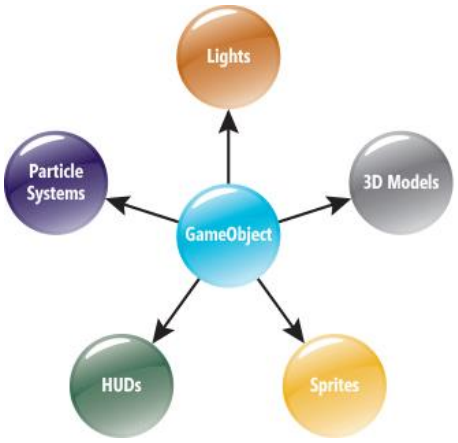

Fig 3.1 Diagram of relationships between components within Unity, nearly all relate back to a gameobject.

Unity supports all major 3D applications (including Photoshop documents) and most audio formats. Unity allows you to import, assemble assets, write code to interact with your objects, create or import animations etc. Having said all this - one of the best features of Unity is the Unity asset store: where the user can purchase artwork, 3D models, 3D assets, animations, audio effects, songs, plug ins, visual scripting systems, shaders, textures, partial effects; of which many of these offer free versions.

3.2.3 Design Patterns

In this project the MVC design pattern will be used. This is used to organise the application by separating it into three key parts, the model,view and controller. Firstly The model contains only the application data; it contains no logic describing how to present the data. The view presents the model's data to the user (Interface/Detection). The view knows how to access the model's data, but it does not know what this data means. This is where the controller comes in, the controller acts as a bridge between the view and the model handling all the logic of the application (Decision/Action). It takes care of all the user requests, inputs and listens to events triggered by the view. By splitting the thinking process into data, interface, and decisions, developers can reduce the number of source files that must be searched in order to add or fix functionality.

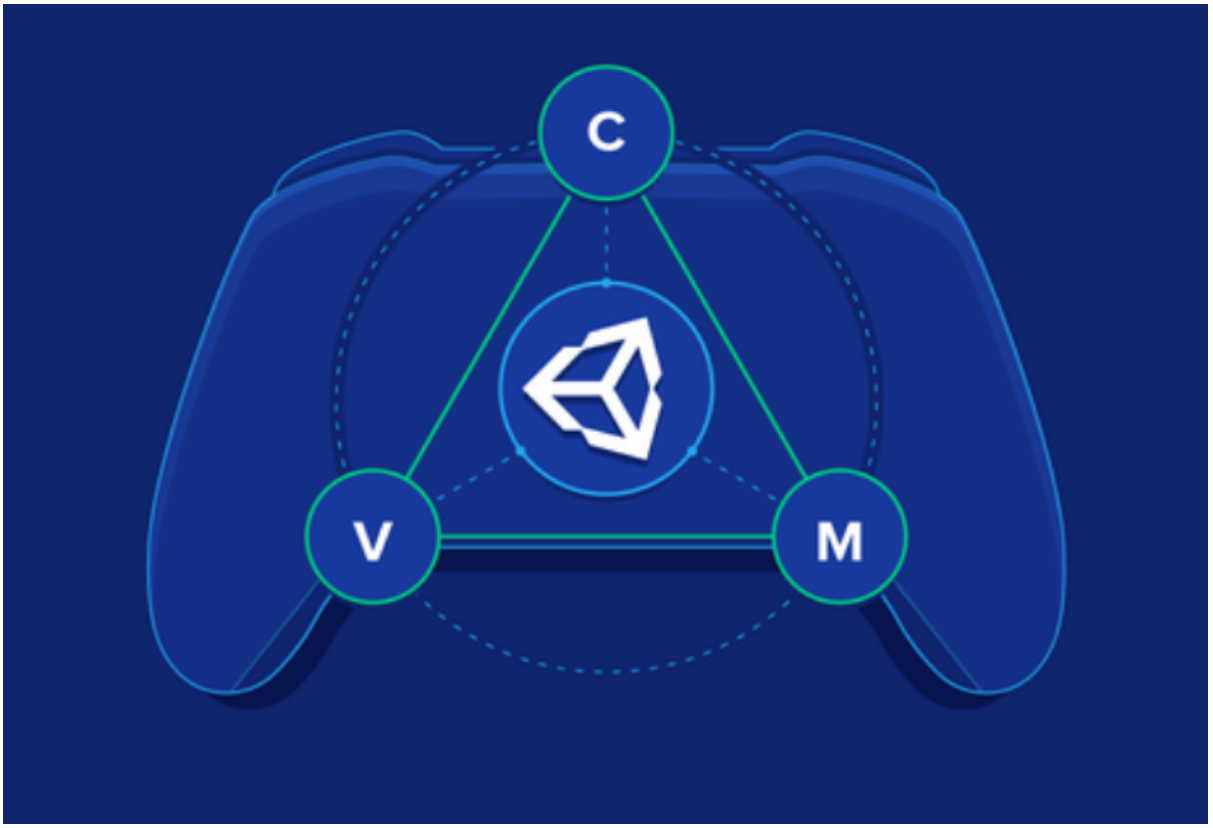

Fig 3.2 Diagram of MVC design pattern

3.2.4 Application architecture

Below you can see a diagram laying out the architecture of the application and how the different interplay between the different application parts will operate.

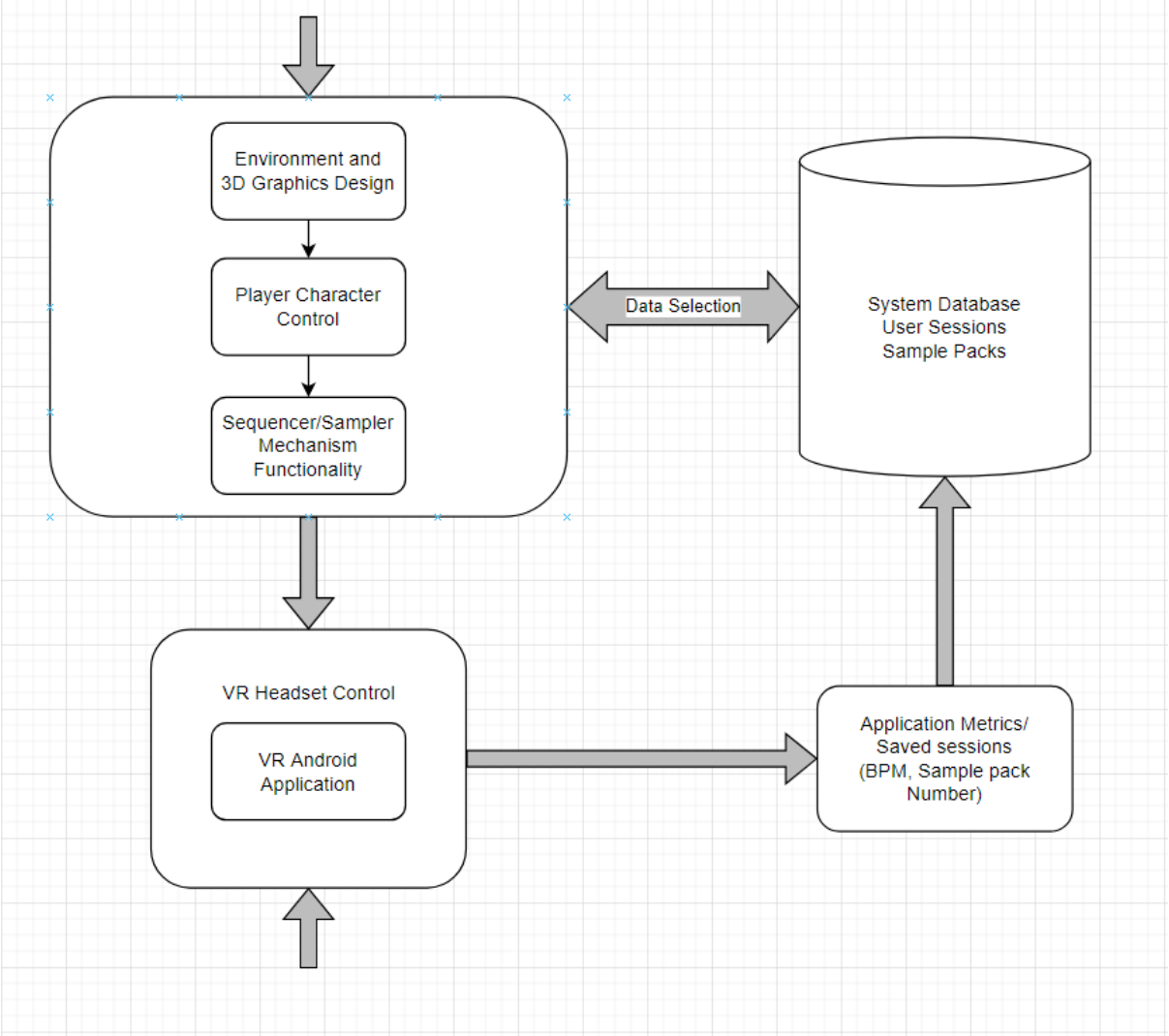

Fig 3.3 This diagram shows the architecture of the application and the various communications carried out during operation of the application.

3.2.5 Database design

In relation to database design, the project being created does not require a stereotypical database as most of the assets will be stored within the application itself. User samples and exported audio captures would be saved locally on the device within the already existing file structure of the project. There is a possibility that an online database will be created to interact with the application to allow users to upload their audio samples to a website online. This would require a database for the website that would be designed later into development if an online application is required for this functionality.

3.3 User interface design

3.3.1 Wireframe

Below you can see some of the wireframes that have been mocked up for the various interface and menu designs required for the application. The first one below is a wireframe of the main menu the User is greeted by upon entering the application. While this wire frame does not show the final design in any way it gives an idea of what will feature on the menu and where things should be positioned. For all the wireframes a grid is used in order to keep everything in line and accurate to where it will be positioned.

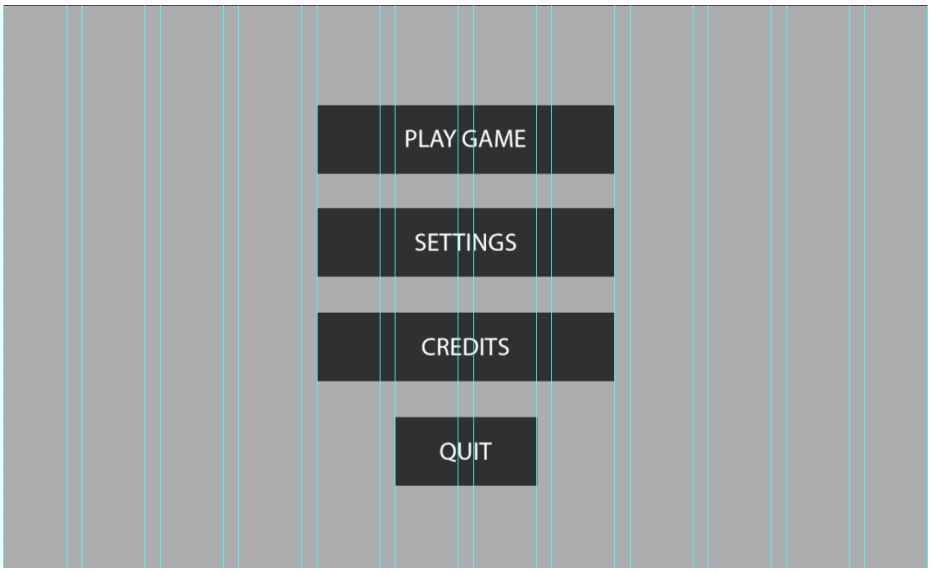

Fig 3.4 Wireframe prototype created for beginning menu system.

Following this, next a wireframe was created for the interface of what the in game sequencers may look like, here the idea was to build a 8 x 8 grid of buttons to represent the different 1/8th note blocks in the sequencer code. The idea being that when a button is pressed that note is added to the sequencer at that position. It was very helpful to mock these designs up in wireframes as it helped create an understanding of the different requirements of the sequencers. It was realised that the below design is more suited to the drum sequencer as it does not have the ability to easily display sharp and flat notes which would be required to create melodies. This has prompted the creation of another different design for a note based sequencer.

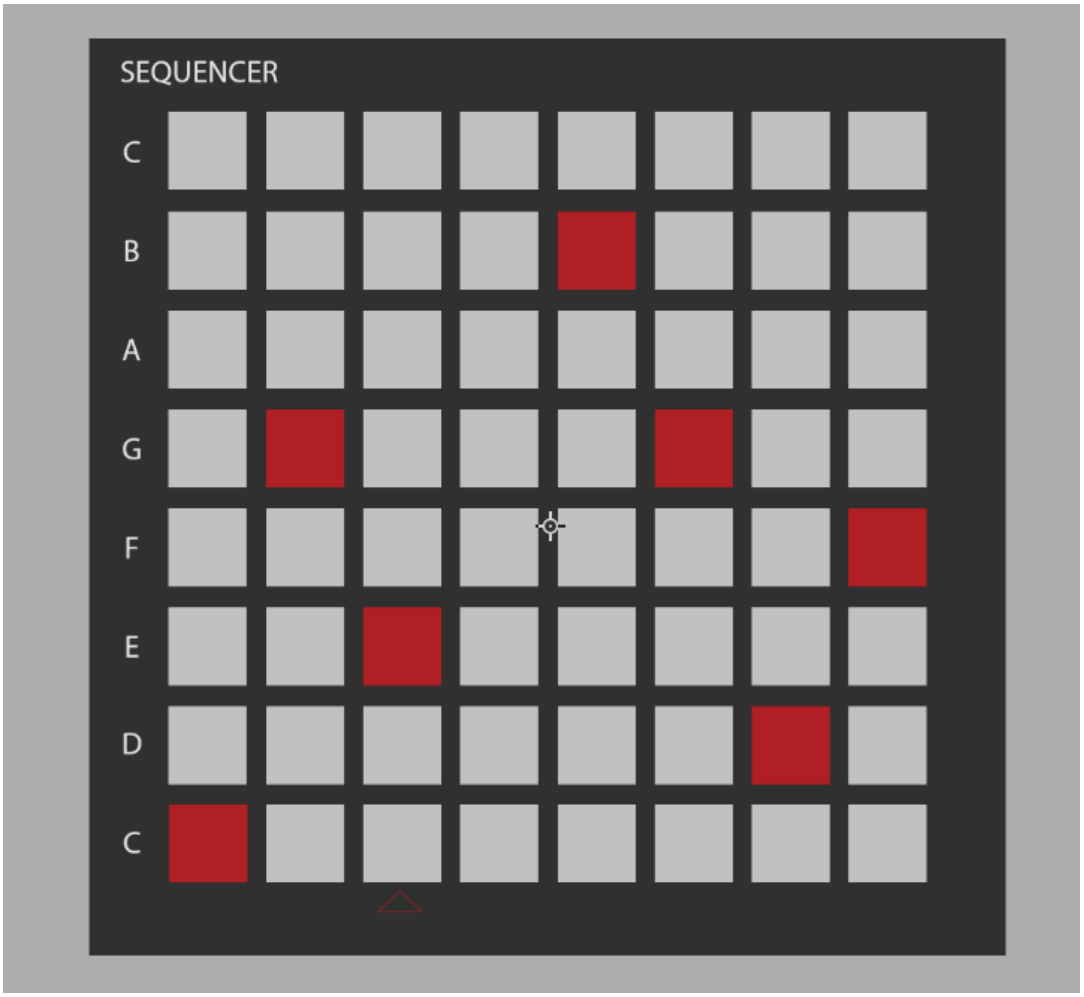

Fig 3.5 High fidelity prototype design of the sequencer design, created in photoshop.

Following these designs, designs for a continuous arpeggiator were worked on and imagined how this would look in VR, it was here that the task of creating a robust user interface that reflected real musical logic but also easy for newcomers to interpret was met. It was decided that to achieve this for an arpeggiator such as this, the visuals of a piano would be the most common to both parties. So a test interface was prototyped for this where the user would socket blocks into the keys they wanted to activate. After implementing this, it became obvious that the keys should instead be touch toggle activated and the socket method of activating notes would suit a pinboard style arpeggiator which was the next interface created.

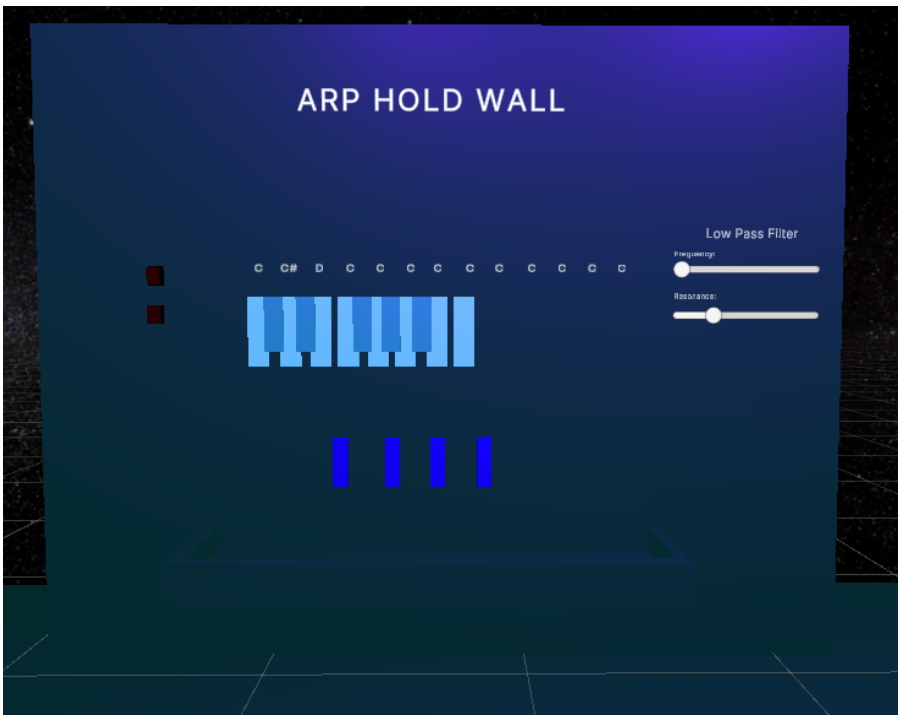

Fig 3.6 Prototype sequencer 3d designs

After realising that the socket system would probably suit a pinboard style interface new designs were created based off of my previous designs of what my sequencers might look like. This pinboard style sequencer was designed with the notes ranging a full octave going up on the left and sixteen slots to symbolise each spot in the sixteen bit sequencer

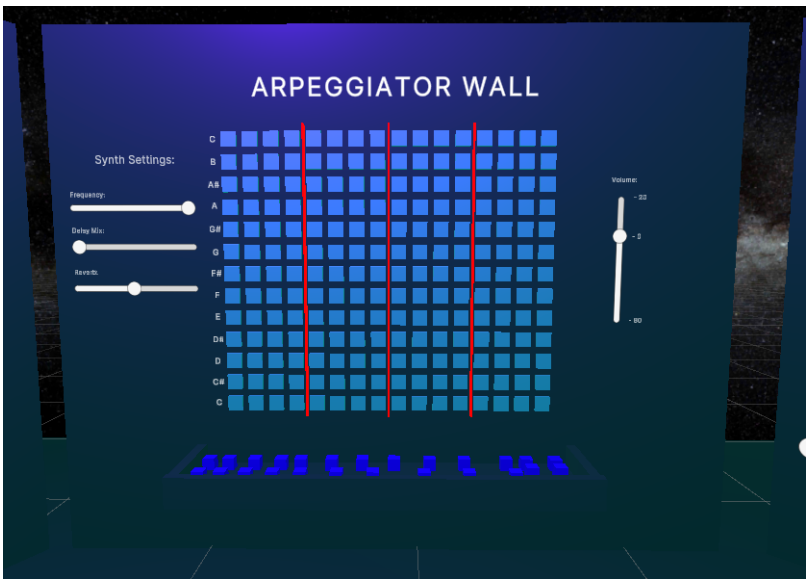

Fig 3.7 Prototype sequencer 3d design

3.3.2 User Flow Diagram

Following creating wireframes, User flow diagrams were created to explain and show the path of a User within the application and their choices. This helped to give an accurate insight into how a User would navigate the application, which is Important to understand before you begin to implement your designs.

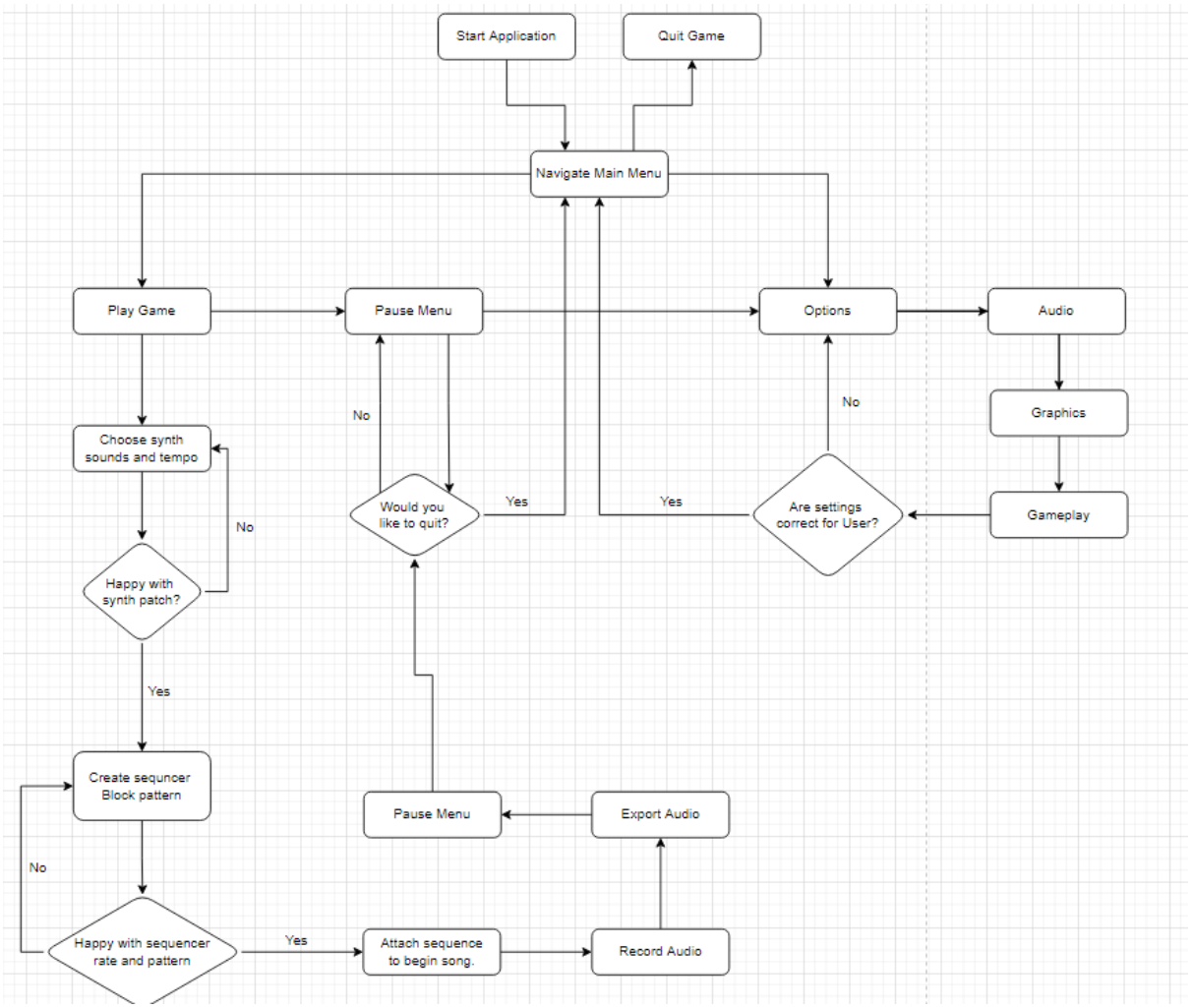

Fig 3.8 TempoVR User Flow Diagram

3.3.3 Style guide

While the main focus of this project is functionality and creating a musical tool for Users to use and play with, the visual appeal of the application is important too as this will help create the environment and vibe of the application which is very important in a VR game. The general aim with the style of the game is to create a neutral space that does not upon entering overly stimulate the user with too much information. While the aim for the look of the application was to be sleek, the visuals of the application should not be so much as to distract from the functional goal of the application, instead they should help improve the Users understanding of the tools within the game and enhance the music creation experience through use of music visualisation.

3.3.4 Colour Palette

The main colours I have decided for the UI elements and Audio Interface elements are a mixture black, white and greys to deliver a neutral palette helping the important information to be presented clearly, as this is very important in relation to the audio component settings the User will interact with. This will be accompanied by varied greys to bring the design together. Guardsman's Red as a Primary colour to act as an accent/highlight to various parts of the UI and audio Interfaces.

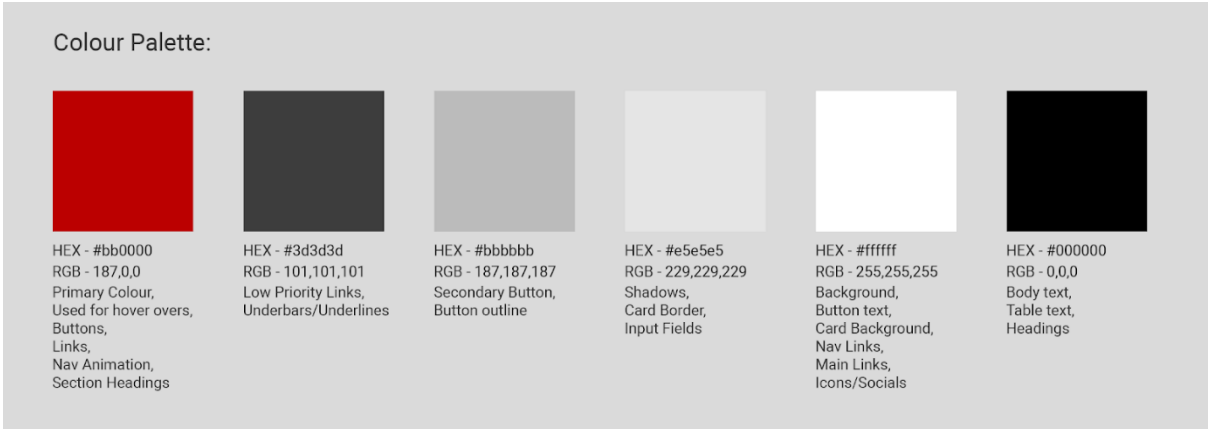

Fig 3.9 Early colour palette for the Synth wall designs

3.3.5 Level Design

Level design is where my application will differ slightly to other Virtual reality games, unlike most games my virtual reality application will take place all from one level the entire time. There is no need for multiple level designs in my application as it is not the goal of the application for the User to progress through a set of levels. It is more important that Users feel a level of consistency in the level design so they can learn where things are and how to navigate my environment. As the User will spend most of their time in this one level, the importance of creating a strong and intuitive level design is even greater. It is my goal to have a level that requires very little movement and struggle for the user to achieve things. I want all objects and tools the User may need to feel like they are always within grabbing distance without creating a cluttered environment.

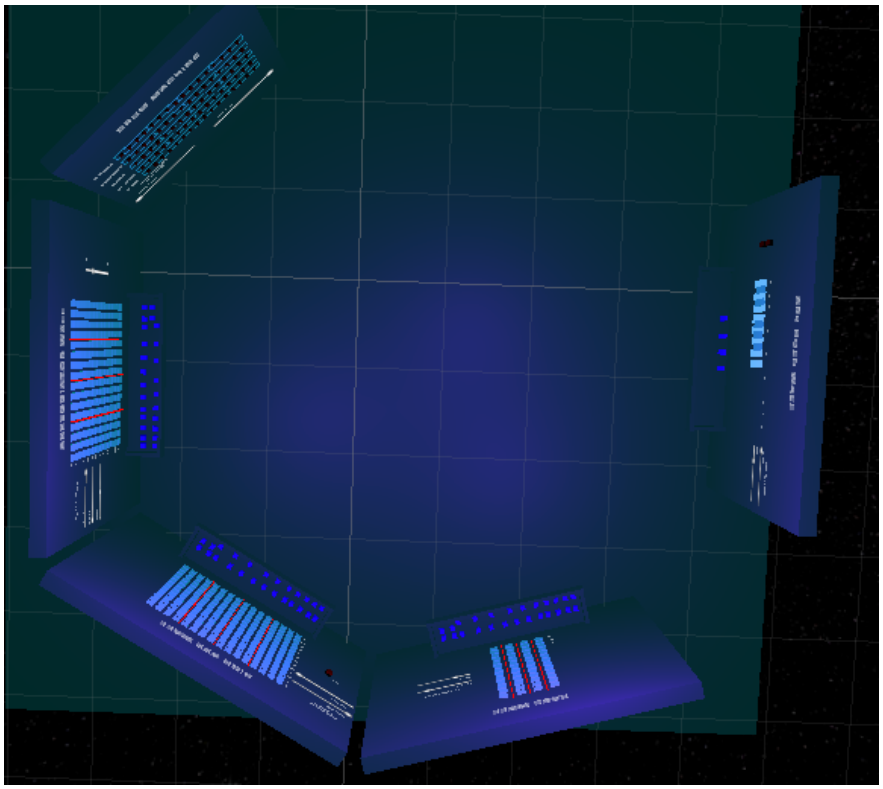

Fig 3.10 Example of current Test Layout.

3.3.6 Environment

The concept for the environment I want to create is laid out in a paper prototype below. In the VR world the user should feel at ease and relaxed, to achieve this a very neutral colour palette for the environment will be used so as not to take center stage away from the interface of the VR world. The User will be surrounded by four interface tables which relate to different aspects of digital musical creation almost like loo station. The area surrounding this would be an open space with no objects to distract from anything. It is an aim of mine to create a form of musical visualisation to surround the user within the environment. This will allow the User to feel like they are surrounded by the sounds which they are creating.

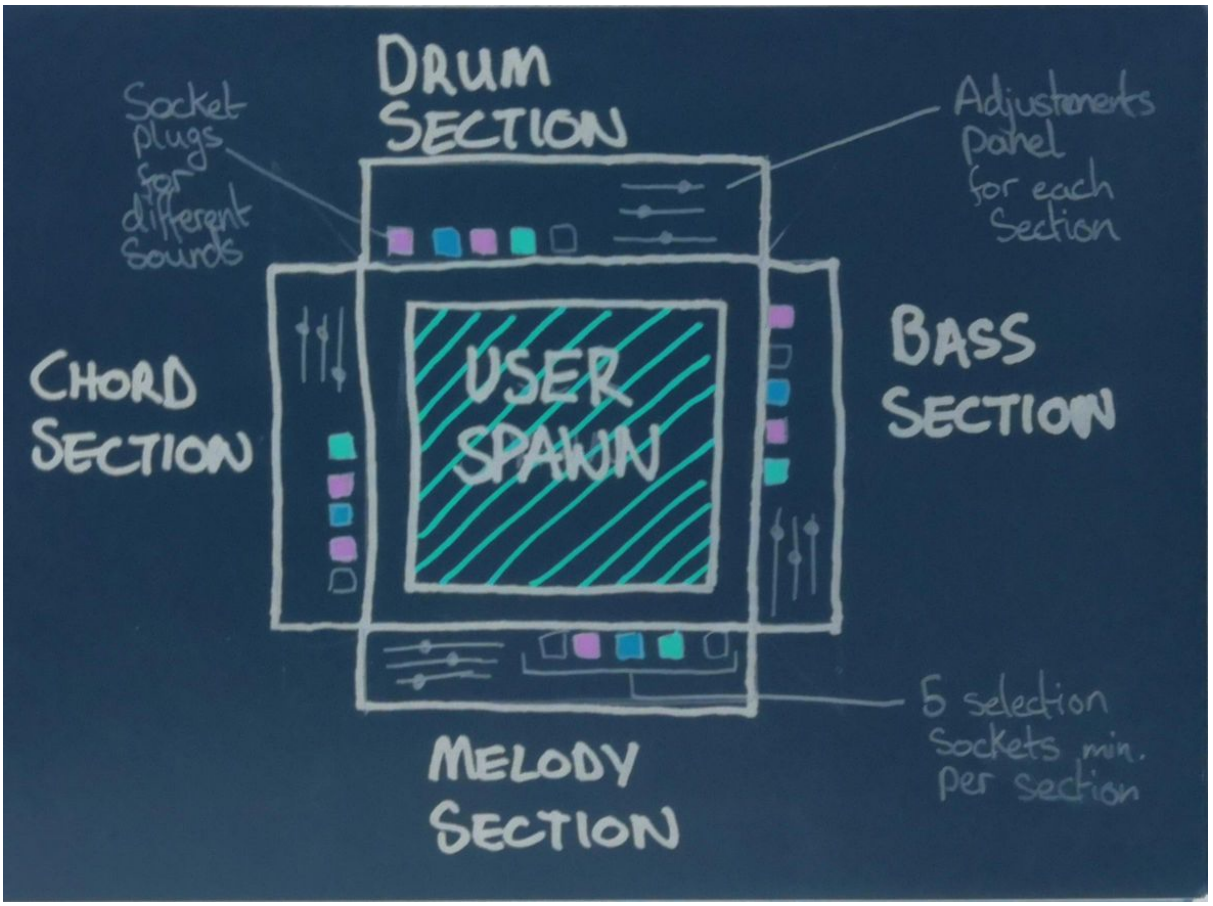

Fig 3.11 Paper Prototype for environment design/layout.

3.4 Conclusion

In conclusion this chapter has been useful in helping to provide a guide for myself on how the VR application will be developed and how its design will look. The primary aim in this section of the write up was to ensure the design of the program and its User interface was clear and easy to interact with for the User. When working with software for digital music creation it is very important that the visuals of your interface are clear and provide useful information and feedback on the music you are creating. It is also important, in particular to VR applications, that the design of the User's environment is intuitive and easy to navigate and all User inputs and controls are responsive to the User. These are aspects of the development that will be kept as a high priority throughout the application's creation.

In this section of the document I discuss the technologies that will be used to create my VR application and how the user will interact and navigate the game. Through use of wireframes, an idea about the desired look and layout of the various interfaces the user will interact with was noted so I can clearly see the vision of the application. The style guide is used to establish a consistent look and feel to the UI design through the VR application.

4 Implementation

4.1 Introduction

As previously mentioned the goal of this project is to develop an approachable and fun VR music creation app, providing a user access to real synthesising and sequencing tools from inside VR. In order to achieve this the application has been developed in Unity, which stands as a great option for VR development especially with the wireless oculus quest which is the main platform I have in mind for the application as it is considered by most that all VR headsets will move towards being untethered in the near future. Unity will provide me the ability to build and create the VR world, its physics and all of the scripting required to build the logic for a project such as this (see fig 4.1).

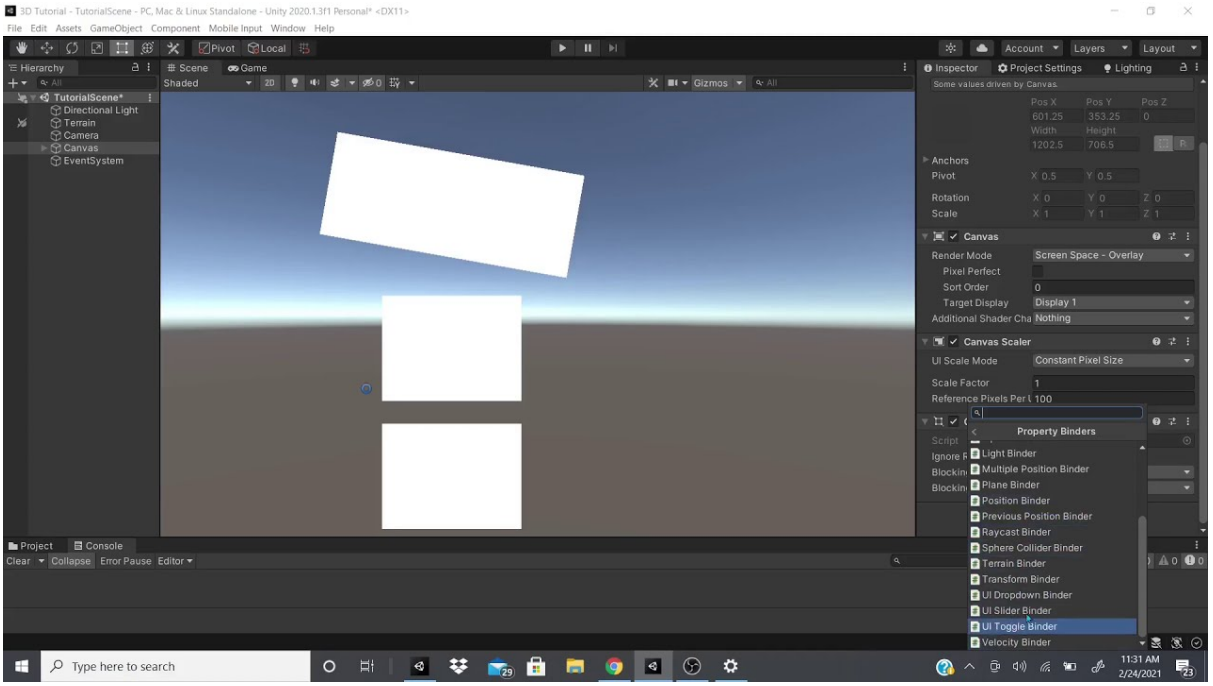

Fig 4.1 Example of a project inside Unity

While I am using Unity to construct the world and logic of the project, for the synthesiser that the user will have access to I am using a Unity asset known as Helm. Helm is a standalone open source digital synthesiser that has been adapted and ported to the Unity asset store. I originally used Helm for personal music

projects inside Ableton, which is a more common use of the synthesiser. The main intention behind the Unity asset was to give developers the ability to create dynamic music for their games. In relation to my own project idea, my aim was to instead create VR tools to allow the User to use Helm as a native Synth and create original and Unique music from within the VR world. To achieve this I will have to implement an interesting and interactive UI for the User to interact with as without this making sense of any part of the Helm Synthesiser would be impossible. This would require rethinking how traditional musical sequencers and synths work in the real world and how to make these more interactive and intuitive for the VR world. The first goal will be to integrate the sequencer code within Helm to the synthesiser and then building tools for the user to use the synth.

4.2 Scrum Methodology

Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. This framework is structured in phases for project management and is mainly used in active software development. In Scrum methodology, the team generally consists of the Product Owner, a Scrum Master and the development team. The Scrum Methodology is made up of three artefacts: Product Backlog, Sprint Backlog, and Burn Down Chart.While this is intended more for a larger development team and I am a solo developer in this project, the principles of this methodology can still apply to the development process.

The Scrum master:

The Scrum master is the one who keeps the team working together, communicating and staying focused. They are a supportive team member ensuring aspects of collaboration, value-based prioritisation, planning and improvement are happening within the team.

They help the product owner to better understand and communicate the value. They aid with managing backlog and helping plan the work within the team. They help the Development team to deliver the value, helping them to self-organise, focus on the outcomes and manage blockers.

The Product owner:

The product owner is the one who works with the Development team and has a vision of what the team should deliver. As the Product owner is the teammate who represents the business and ensures they are delivering the most value; It is important for the Product owner to understand not only the customer, but what it is that they want to deliver to the customer. The communication between the Product owner and users/customers is key; as feedback is one the essential components for team improvement. The product owner should know everything that is in the backlog

and it is important that other team members communicate with them when adding items to the backlog.

The Development team:

The Development team are the developers, they're the team member who puts the vision into action. The development team can be varied and includes programmers, designers, writers etc.

PRODUCT BACKLOG EXAMPLE						
ID	As a...	I want to be able to...	So that...	Priority	Sprint	Status
1	Administrator	see a list of all members and visitors	I can monitor site visits	Must	1	Done
2	Administrator	add new categories	I can allow members to create engaging content	Must	1	Done
3	Administrator	add new security groups	security levels are appropriate	Must	1	Done
4	Administrator	add new keywords	content is easy to group and search for	Must	1	Done
5	Administrator	delete comments	offensive content is removed	Must	1	Done
6	Administrator	block entries	competitors and offenders cannot submit content	Must	1	Done
7	Administrator	change site branding	the site is future-proofed in case brand changes	Could	1	Done
8	Member	change my password	I can keep secure	Must	1	Done
9	Member	update my contact details	I can be contacted by Administrators	Must	2	Work in Progress
10	Member	update my email preferences	I'm not bombarded with junk email	Should	2	Work in Progress
11	Member	share content to social networks	I can promote what I find interesting	Could	2	Work in Progress
12	Visitor	create an account	I can benefit from member discounts	Must		To be started
13	Visitor	login	I can post new entries	Must		To be started
14	Visitor	add comments	I can have a say	Must		To be started
15	Visitor	suggest improvements	I can contribute to the site usability	Should		To be started
16	Visitor	contact the Administrators	I can directly submit a query	Could		To be started
17	Visitor	follow a member's updates	I'm informed of updates from members I find interesting	Should		To be started
18	Visitor	view a member's profile	I can know more about a member	Must		To be started
19	Administrator	generate incoming traffic report	I can understand where traffic is coming from	Must		To be started

Fig 4.2 A Product Backlog template . Retrieved https://www.altexsoft.com/media/2018/06/Product-Backlog-WIth-User-Stories_m.jpg

Fig 4.3 A Sprint Backlog template. Retrieved
https://www.visual-paradigm.com/servlet/editor-content/scrum/what-is-sprint-backlog-in-scrum/sites/7/2018/12/sprint-backlog.png

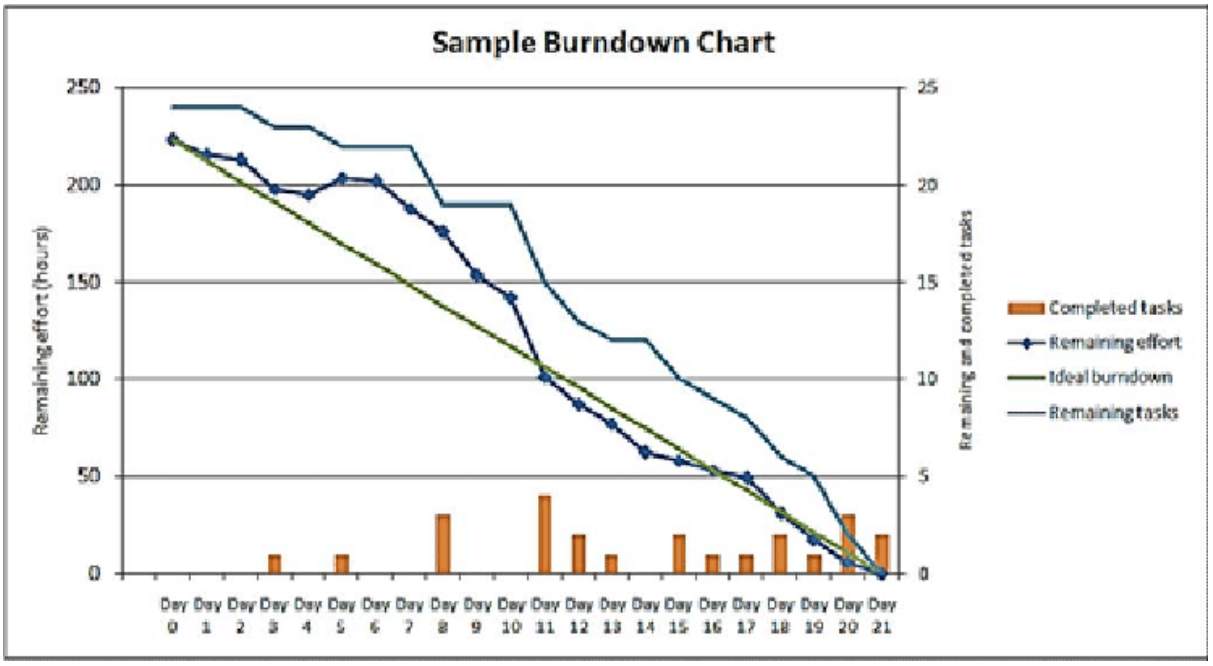

Fig 4.4 A sample Burn Down Chart. Retrieved https://www.researchgate.net/profile/Wael-Mohsen/publication/321741375/figure/fig1/AS:614348489768965@1523483502062/Sample-burndown-chart.png

4.3 Development environment

For my Integrated development environment (IDE) to use for all the scripting requirements, I used visual studio as it is the standard IDE to use side by side with Unity which is where most of the main development is taking place. Visual studio proved a great tool for making the scripting process easy and clear to understand as you are able to install extensions so that it understands the language you are scripting in. This helps greatly improve accuracy and removes wasted time with small simple to fix errors. Alongside Unity, because I am making a VR application I require a VR headset to test and develop on. For this I used my own personal Oculus Quest 1 which allowed me as much access to the headset as needed in order to develop the app.

Currently the application is backed-up and posted on my college GitHub in order to make the development process smooth and organised, this can also prove useful for accessing older builds of the application.

4.4 Sprint 1

4.4.1 Goal

- Start creating a draft version of my Requirements document.
- Further explore my plans for the design of the app and think about what ideas or features would suit the app.

4.4.2 Item 1

Research and evaluate other similar applications to see how they approach their own sites.

4.4.3 Item 2

Decide from looking over similar applications what made the app good and bad, and how that can help us improve our own design. Also figure out what standard requirements are already expected from applications like this.

4.5 Sprint 2

4.5.1 Goal

- Refine and finish the Requirements document.
- Set up my development environment/install all assets and packages required.
- Start building/create the basics of the Unity project.

4.5.2 Item 1

Create an online survey to gauge interest in our product field. Try to have as much of a diverse survey group as possible to get accurate representation of general public interest. Then analyse all the collected data from the survey, creating visual charts and aids to display the data.

4.5.3 Item 2

Created the foundation for the application in Unity implementing a file structure in the project window to keep the various aspects of the project organised (see fig 4.5), Import all assets and begin to implement simple instances of Helm in order to start gauging requirements for the scripting of the project. Begin inspecting and using the sequencers scripts in order to figure out their functionality and how they can be manipulated/used.

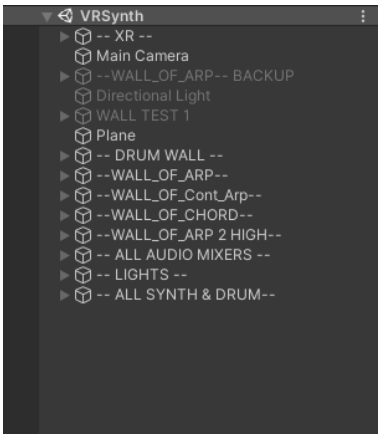

Fig 4.5

In order to access Helm it must be added to one of the mixers channels before the Attenuation component, this instance of helm then belongs to this channel. Anything affecting this channel will affect the sound output of the Helm Synth.

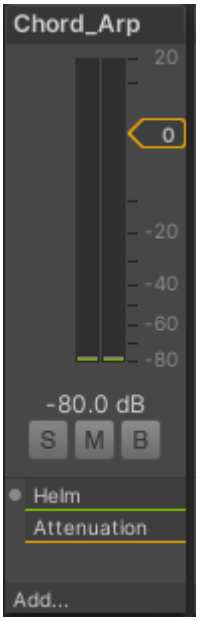

Fig 4.6 Example of attaching Helm to an audio channel within a mixer.

4.6 Sprint 3

4.6.1 Goal

For sprint three the main goal was to start properly into implementing and testing many of the core concepts behind the different interfaces/instruments. After figuring out how each component of HELM worked in the previous sprint it was time to implement a method for Users to interact with HELM. In this sprint these systems were created alongside powerful base UI's to provide an interface for the User to interact with the scripts created.

4.6.2 Item 1

The first step in the process of creating these sequencers in Unity that can interact with the HELM synth instantiations placed on the mixer channels is to create an empty game object in Unity (see fig 4.7). This empty game object will be named according to what type of sequencer it will be, this is because there will be multiple separate sequencers for each of the different synth walls. A Helm sequencer script component is then attached to this game object.

Fig 4.7 Project layout for Sequencers

The Helm sequencer script is a C# script file included with the Helm asset package. This script comes with a basic sequencer that has adjustable parameters and some basic methods to apply changes to the sequence blocks (see fig 4.8). The basic parameters provided include the length of a sequence, the division of the sequence and most importantly the channel that this sequencer belongs to. This channel parameter must be set to the same value as its correlating Helm synth in order to affect it. The Helm sequencer script provides its own interface, but this is only accessible from within the Unity editor which means it is not usable from within the application. This means that separate custom scripts must be created in order to perform changes to the Helm sequencer script live in the game engine.

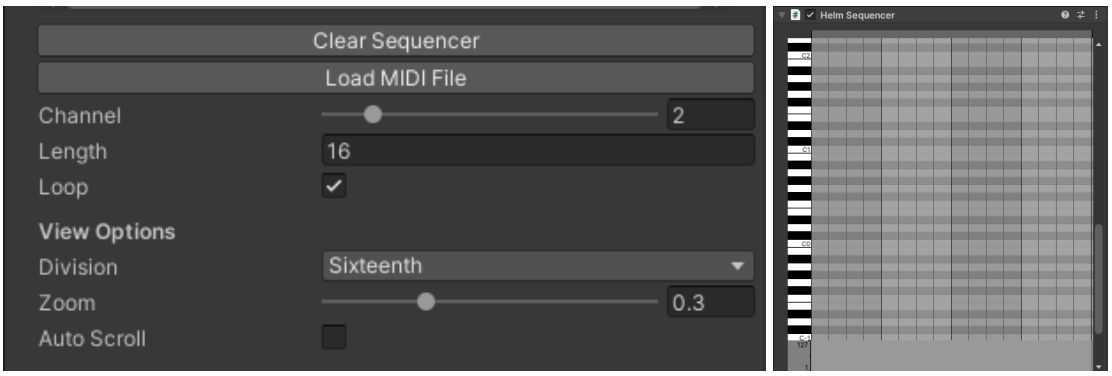

Fig 4.8 Helm sequencer script component interface

In order to achieve this it was decided in previous design phases that a socket based system would be used. The idea behind this was to have flat squares on a wall acting as plug sockets to represent the note at point in a sequence. The User would then pick up cube game objects from a tray and plug these into the square sockets on the wall, when the User does this the intention is for the socket to activate a script that inputs the correct note and note position onto the given sequencer based off which socket the User has input the cube into. Below in fig 4.9 you can see an example of this, here a flat square object is created and named Note_Plug_0/1 with the numbers here correlating to what note and point in the sequence it belongs to. The script to input onto the sequencer is attached to this object and activated by the socket object that is a child of it. The socket activates the script when a cube is plugged in or out.

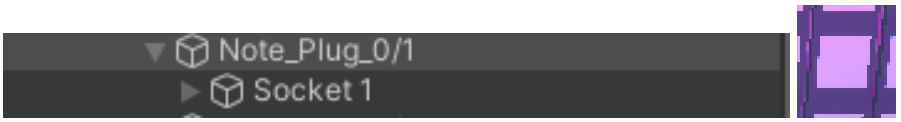

Fig 4.9 Example of the file architecture used in creating the note sockets.

Below you can see the custom Block sequencer Script that was created from scratch to perform the action of inputting or removing blocks from the sequencer (Fig 4.10). The Script requires some parameters to be passed into it to provide it with the information about which section of the sequencer it correlates to. The starting note parameter refers to what note the given block is supposed to play, this is measured

as a midi value meaning all notes are stored from 0 to 127. The Step in Sequence parameter refers to what point time the given note belongs to on the sequencer, if the sequencer is set to sixteenth notes then a value of 14 would mean the 14th sixteenth note in the sequence. The script also requires that a Helm sequencer such as the one shown in fig 4.8 is passed in to it so that the script can perform actions to that sequencer.

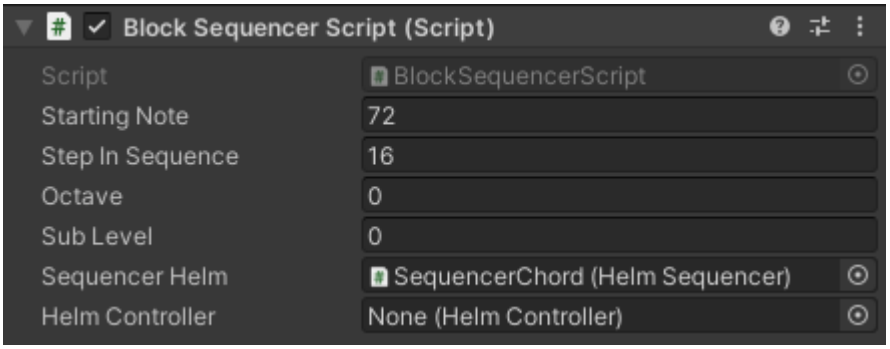

Fig 4.10 Component view of Block Sequencer Script, here you can see what variables it expects to be passed into it.

In fig 4.11 you can see all of the methods that were written to perform different actions to the sequencer passed into it. These methods were broken down into two main categories, methods to add/remove a single sixteenth note to/from the sequencer and methods to add/remove a long one bar block note (one bar equating to 16 sixteenth notes) to/from the sequencer. The two methods that act on single sixteenth notes are called addNoteToSequence() and removeNoteFromSequence(), these methods are intended to be used to create melodic sequences that move in sixteenth notes. The other two methods addChordToSequence() and removeChordFromSequence() function almost the exact same but instead input a long block note too the sequencer with a length of sixteen sixteenths which equates to one full bar, this is to allow users to create actual long playing chords. In order to explain how these methods function the first method addNoteToSequencer() can be used as an example. When this method is called a Helm function to add a note is called, this function requires the sequencer that was passed in to perform the action on and a few parameters passed through with it to specify the details of the note you want to add. The three parameters passed through are first the actual midi value from 0 to 127 of the note that this block represents, the next parameter is the point in

the sequence this block belongs to so that placement in the sequence is correct. The final parameter passed into the script is the end point of the note within the sequence, in the case of adding a single sixteenth note this will be the second parameter plus one as it is a length of one sixteenth. It is this final parameter that differs the methods as if you wanted to add a long phrase the final parameter would be the step in the sequence plus sixteen in order to make a note play for a full bar.

```
public void addNoteToSequencer()
{
    sequencerHelm.AddNote( note: startingNote, start: stepInSequence, end: stepInSequence+1);
}

0 references
public void removeNoteFromSequencer()
{
    sequencerHelm.RemoveNotesInRange( note: startingNote, start: stepInSequence, end: stepInSequence+1);
}

0 references
public void addChordToSequencer()
{
    sequencerHelm.AddNote( note: startingNote, start: stepInSequence, end: stepInSequence+16);
}

0 references
public void removeChordFromSequencer()
{
    sequencerHelm.RemoveNotesInRange( note: startingNote, start: stepInSequence, end: stepInSequence+16);
}

0 references
public void addFullNoteToSequencer()
{
    sequencerHelm.AddNote( note: startingNote+(octave*12), start: stepInSequence, end: stepInSequence+16);
}

0 references
public void removeFullNoteFromSequencer()
{
    sequencerHelm.RemoveNotesInRange( note: startingNote+(octave*12), start: stepInSequence, end: stepInSequence+16);
}
```

Fig 4.11 Code snippet from inside the Block Sequencer Script showing the different functions it has for note placement and removal.

Once everything is connected and input correctly into the script in respect to its correlating note block, the script can then be connected to the note blocks XR socket interactor component. When this is done the interactor events on the socket for on select and on exited can be used to execute different methods from the Block Sequencer Script depending on whether a block is plugged in or out. You can see in fig 4.12 that for the on-select trigger of the socket the addNoteToSequencer() method is called, which adds a note to the sequencer. Below this you can then see that the on exited trigger the removeNoteFromSequencer() method is called,

removing the note that was previously added. This method for altering the sequencers was replicated and adapted throughout the application in order to create the different synth walls, sometimes altered versions where needed to suit the different needs of a sequencer.

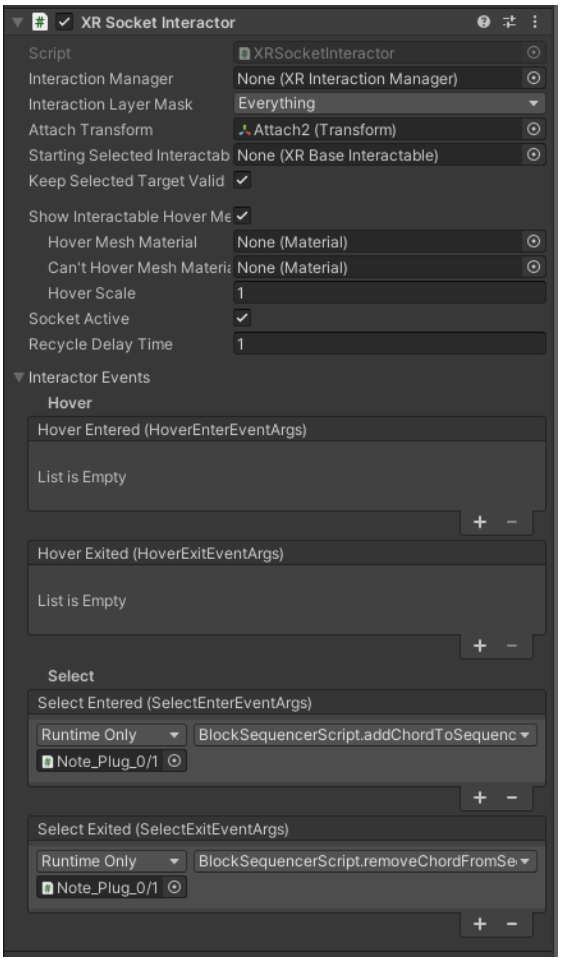

Fig 4.12 An XR socket interactor used as a trigger for the script.

4.6.3 Item 2

With the application now being capable of allowing Users to add/remove notes on a sequencer the groundwork to build a full interactive synth interface for the User is there, but in order for Users to have complete control over the sounds they create they must have access to adjust the parameters of the Helm synthesiser. Without this the User is creating sequences but not actually getting to be involved in the sound synthesis process. In order to achieve this a script must be created to interact with the parameters of the Helm synth and translate in engine actions by the User to affect the synths parameters. Firstly before any parameters of Helm can be affected by a script, the parameters you intend to use must be changed to an exposed parameter and given a variable name. To access Helms parameters, you firstly navigate to the mixer channel with which the instance of Helm you want to affect is running on. When the channel is selected, click show all options and a dropdown of all Helms parameters will appear (see fig 4.13).

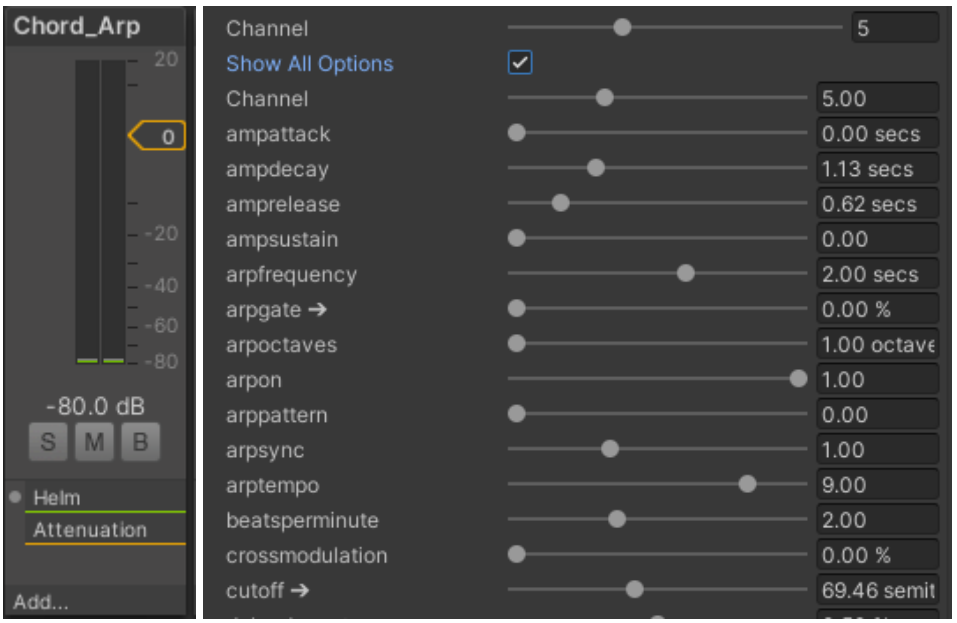

Fig 4.13 Showing Helm parameters accessible from the channel to which it is attached.

From here the required parameters can be exposed to use in scripts, when a parameter is exposed a small arrow appears beside it in the inspector. When a parameter is exposed a name must be given to the exposed parameter, this name will act as its reference when being used in scripts. These exposed parameters will then be displayed in a separate list on the mixer editor as shown in fig 4.14.

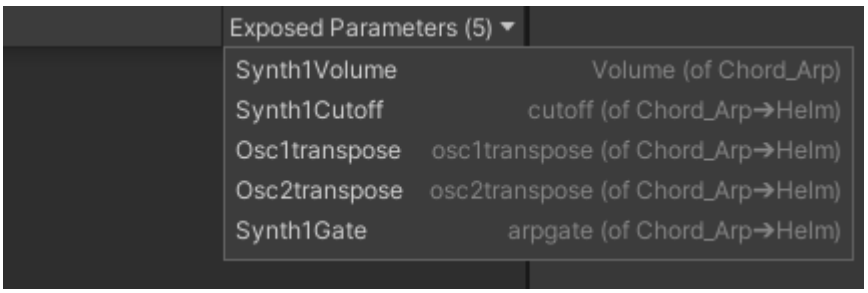

Fig 4.14 Here Helm parameters have been exposed and renamed so they can be accessed by script.

Now that the Helm parameters are exposed it is possible for a script to access them by their given name and perform changes to the values. To implement this an empty game object is created named after the synth it will control/mix and placed in the audio mixer section in the project. Then a script was created called Mix Levels and added to the game object as a component (see fig 4.15). This script first requires that an audio mixer be passed into it, this is so that the script can get access to that mixer's exposed parameters.

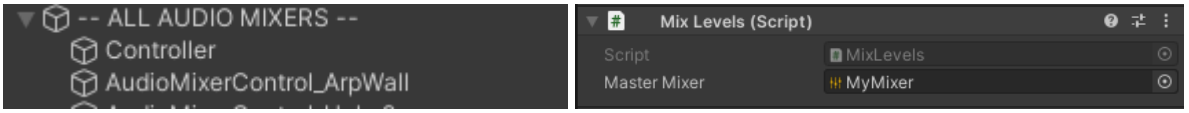

Fig 4.15 The Mix Levels script is attached to a empty game object named after the synth it belongs to.

Inside the Mix Levels script there is a function for each of the parameters that are exposed (see fig 4.16). These functions all work the same way, first the function expects a float value which in this case will be provided by the slider or dial that is created later. Inside the function a SetFloat() function is used on the mixer that was passed into the script, this SetFloat() function expects two values. First it needs a String value for the name of the exposed parameter on that mixer you want to set and second it requires a float value to set as the value which in this case will be the value passed in by the slider.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

0 references
public class MixLevels : MonoBehaviour
{
    8 references
    public AudioMixer masterMixer;
    //public float cutOffLvl = 0.0f; Synth1Delay Synth1Reverb Synth1Volume Synth1DelayFreq D

    0 references
    public void SetCutOffLvl(float cutOffLvl)
    {
        masterMixer.SetFloat( name: "Synth1Cutoff", value: cutOffLvl);
    }

    0 references
    public void SetDelayLvl(float delayLvl)
    {
        masterMixer.SetFloat( name: "Synth1Delay", value: delayLvl);
    }

    0 references
    public void SetDelayFreqLvl(float delayFreqLvl)
    {
        masterMixer.SetFloat( name: "Synth1DelayFreq", value: delayFreqLvl);
    }

```

Fig 4.16 Mix Levels script, these functions are used to pass values to the exposed parameters.

Once the script is in place it is possible to simply create a Unity slider object and pas the on-value changed trigger the object containing the Mix Levels script. From here any of Mix Levels functions can be called and the slider value will be passed in as the float that the script is expecting, this then changes the exposed parameter on the mixer each frame (see fig 4.17). This means that Users can change and interact with the parameters of the Helm synthesiser live from within the game engine.

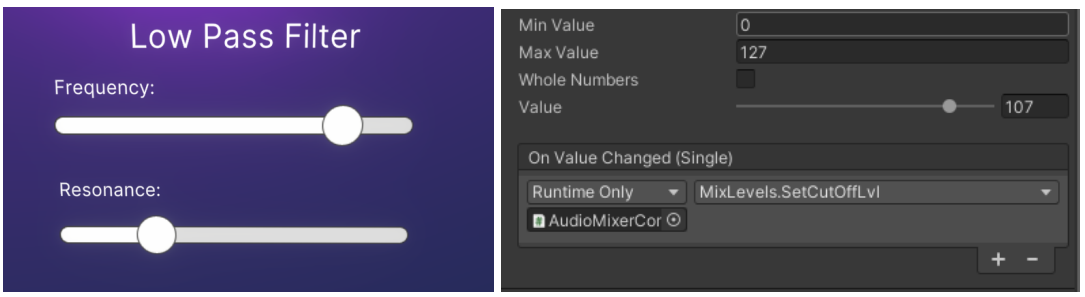

Fig 4.17 A slider used to pass its value into the Mix Levels script.

4.6.4 Item 3

Now that the User has parameter controls it is possible to create a proper synthesiser interface for the User to interact with based on designs laid out during the design section of this project. To achieve this it is just a case of scaling up the socket based blocks created in Sprint 3 item 1. The layout of the sequencer was designed to act almost like a pinboard, changing the placement of a block from left to right would change where it was in the timing of the sequence whereas moving the block up or down would change the note. If the sequencer was a 16 bit sequencer then it would be sixteen blocks long and thirteen blocks high to represent the thirteen notes in an octave. To organise the build of the sequencer an empty game object labeled with the note that row represented was created, then within this game object a row of Note_Plug Objects created in Sprint 3 item 1 correlating to the steps in the sequence are placed (see fig 4.18).

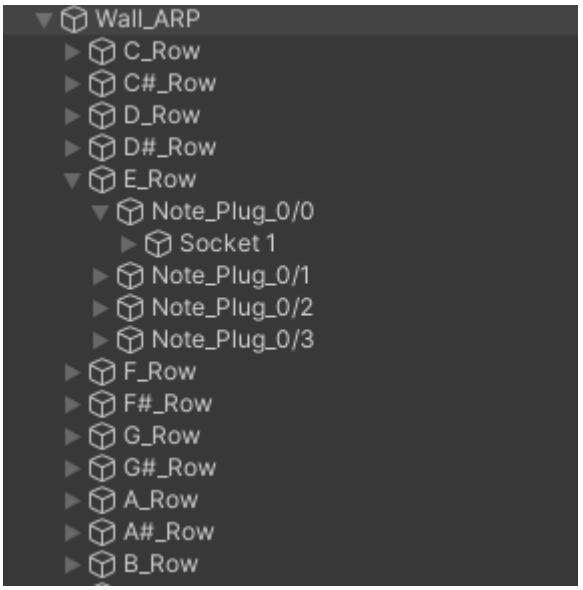

Fig 4.18 The hierarchy for notes within pinboard style sequencer.

Now it is just a case of arranging the Note_Plugs into position within the game engine so that they are in the rows correlating to their note. When each notes row is then stacked in order of ascending notes the pinboard style sequencer takes shape. A tray filled with simple cube objects has been placed below the Note_Plugs so that

the User can pick up a cube an socket it into a spot on the grid, then that location on the Helm sequencers will be activated and the synth will play that note at that time on loop (see 4.19). This allows Users to create original musical sequences that loop and will stay in time with the global Helm clock, this means that this sequencer will be in sync with any other sequencer in the application.

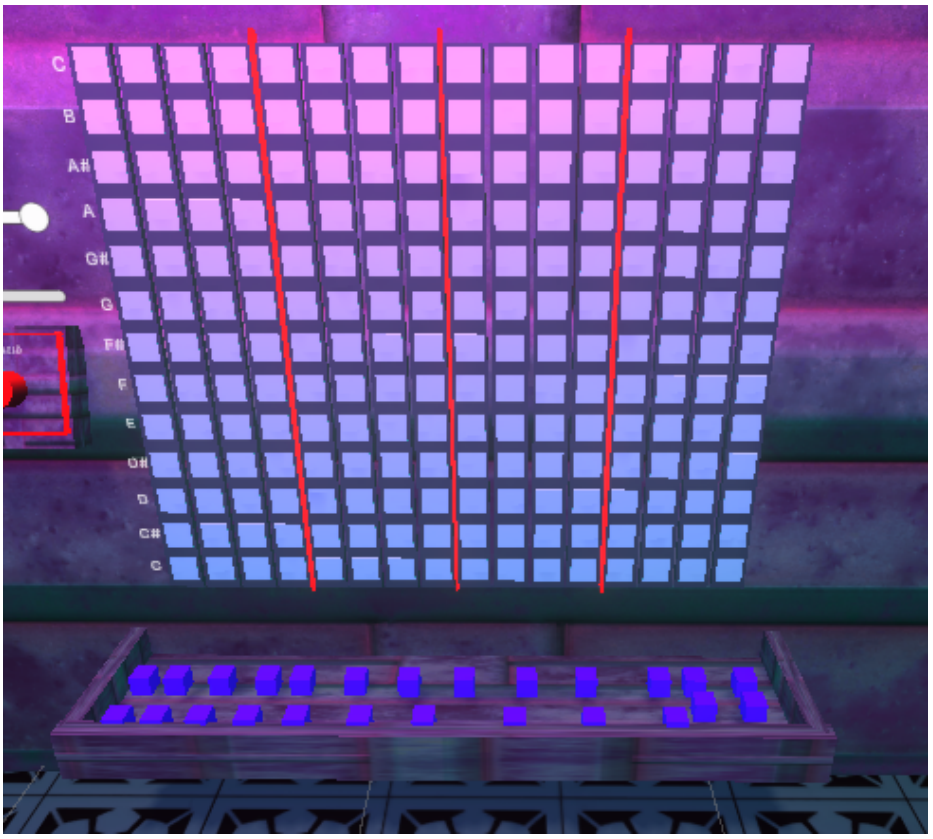

Fig 4.19 Finished prototype of block sequencer.

4.6.5 Item 4

Now with a sequencer already created, it is possible to use parts of it as a template that can be altered to create different styles of sequencers. In the case of a drum sequencer, from the research section it was discovered that a different design approach may be necessary for this style of sequencer. In order to visualise rhythms often traditional drum machines chose a button based system so this was selected as the approach for creating a VR adaptation. The main difference is that instead of the notes being added to the sequencer through a socket system, instead the User can press a button to activate that row's sound at that point in the sequence. In fig 4.20 you can see an example of the drum sequencer interface that was created. The buttons are placed in rows of 16 divided into groups of 4 to indicate each bar with 4 bars making up the sequence loop.

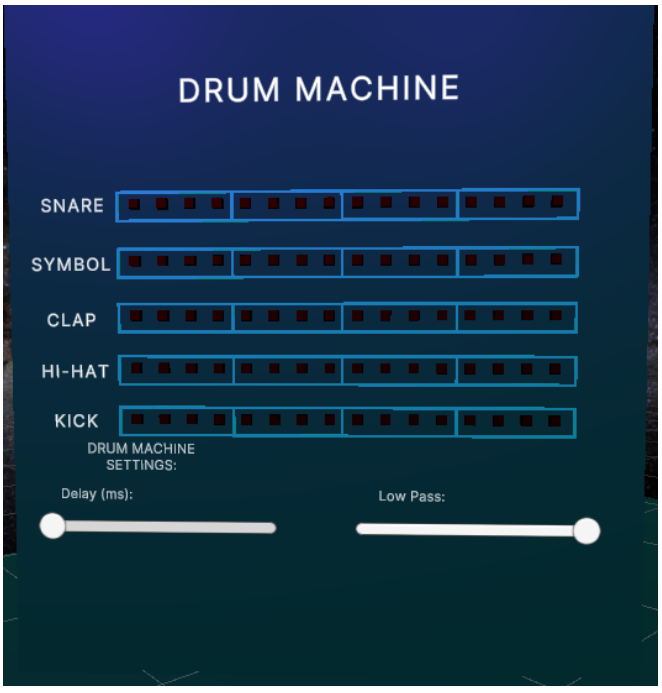

Fig 4.20 Drum sequencers built off same concept as others only using buttons instead of sockets.

4.7 Sprint 4

4.7.1 Goal

Review back over previously written sections to add more content and review/rewrite certain sections of my requirements, Design and implementation chapters. Continue to implement more functionality to the application. Begin working on the sound mixing behind the scenes to improve audio quality and make it so that all synths are running from the one mixer.

4.7.2 Item 1

Removing currently unused mixers that were previously all separate for each instance of the HELM synth, now all synth engines run as members of the same master channel. This is important as when they were all separate there were small issues with audio feedback, possibly due to collisions. Now global volume control is possible and each individual track will have its own volume on the same mixer.

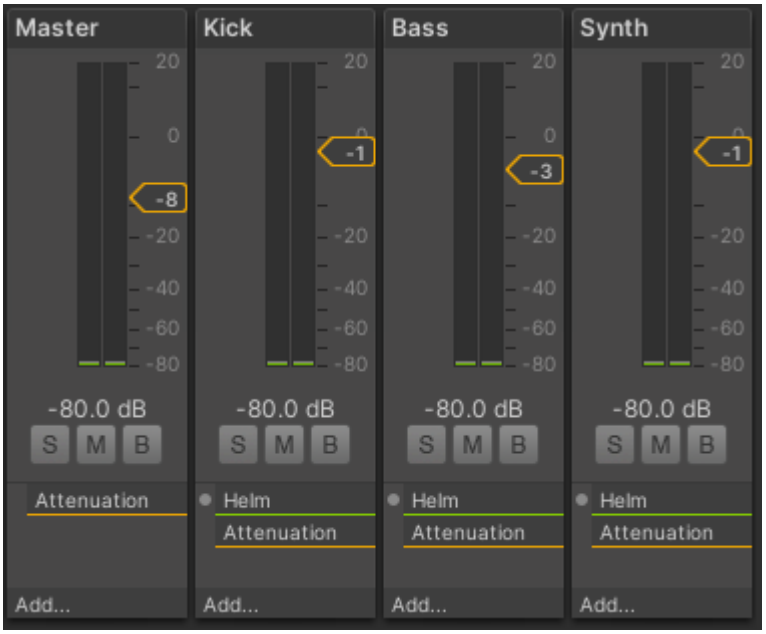

Fig 4.21 Creating sub channels within main mixer for each of the synth walls

4.8 Sprint 5

4.8.1 Goal

Begin starting on a testing phase of the project to gain data from Users as to what can be improved. Continue implementation of musical/sequencer features, mainly aim to implement levers and dials to give the user better methods to interact with the synths.

4.8.2 Item 1

The application was Tested in person with two Users to gain information about improvements. The Users played a test prototype that was created from the previous sprints. See section 5 Testing for more detail on User test outcomes and discoveries.

4.8.3 Item 2

As laid out in the goal section of this sprint the next element to be developed was better methods for the User to interact with the Helm parameters as currently the User must use flat 2d UI's grabbing these with a laser extending from their hand. This method was in no way VR friendly as it provided Users with no accuracy and felt extremely clunky to use. A more natural approach would be to have physical 3d objects such as sliders, levers and dials that alter the parameters. For this sprint creating dials that could rotate on a fixed point 360 degrees and read out a value from 0-100% was the goal as this method of interaction would suit the synthesiser parameter controls the best. Below in fig 4.22 you can see the final version of the dials that were created for the application.

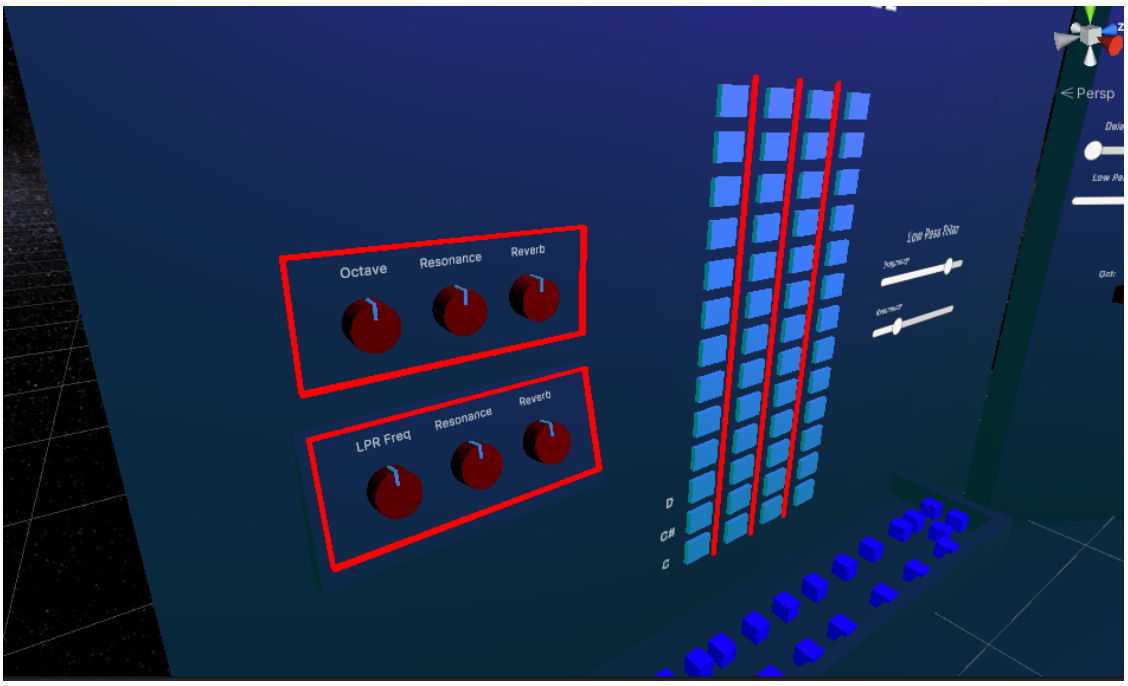

Fig 4.22 Finished example of Dials that were created for the project.

The dial models themselves are quite simple and were created from scratch using simple cylinder and cube objects from Unity, these were then placed as children in an empty game object named dial, this final dial object must also be set with some physics limitations so that it can only rotate on one axis and cannot move. Now that the dial model was created, the housing for the dial which consisted of a flat rectangle that the dial was attached to was next. This would act as the base plate for the dial and it is here that the rotation script will be placed to track the movement of the dial. In Fig 4.23 you can see the hierarchy that was created to create one dial, alongside the previously mentioned components there is also a collider for recognising when the Users hand is interacting with the dial and also a blank game object in this case called FilterID. This last component acts as an ID for what parameter this dial is intended to interact with.

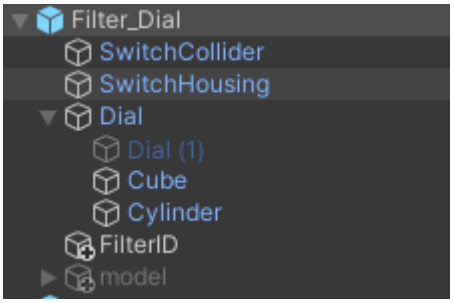

Fig 4.23 Project hierarchy for the dials that where created.

On the object for the dials housing is placed an open source script intended for calculating object rotation. This script requires a few things to be passed into it, first it requires that the fixed object (in this the dial) that you want to track the rotation of be passed. Next it requires two values, one is the snap rotation amount which is the amount of degrees the dial will turn when the tolerance is hit, The second is the angle tolerance which is the amount of rotation from the user's hand required to cause the dial to turn.

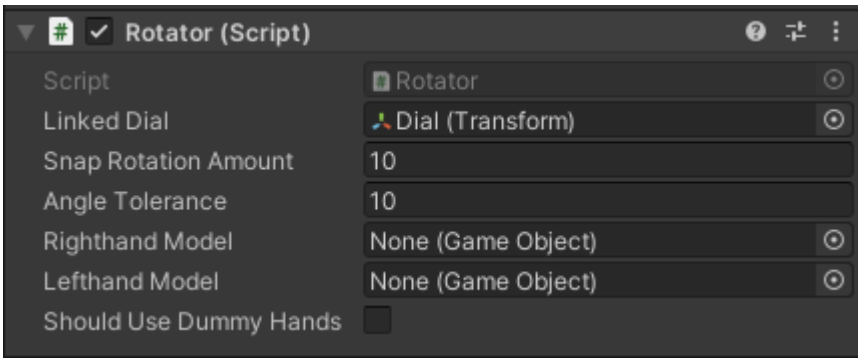

Fig 4.24 Rotator script used for tracking the current rotation of the dial respective to its start position.

Inside the script there are different functions to perform calculations on the passed in object. Below in Fig 4.25 you can see one of the main functions utilised in capturing the rotation of the dial, here a function called GetRotationDistance() is passed in the current rotation of the dial and then checks it against the dials original rotation. From this the current degrees of rotation from the start point can be found.

```
private void GetRotationDistance(float currentAngle)
{
    if (!requiresStartAngle)
    {
        var float angleDifference = Mathf.Abs(f: startAngle - currentAngle);

        if (angleDifference > angleTolerance)
        {
            if (angleDifference > 270f) //checking to see if the user has gone from 0-360 - a very tiny movement but will trigger the angletolerance
            {
                float angleCheck;

                if (startAngle < currentAngle)
                {
                    angleCheck = CheckAngle(currentAngle, startAngle);

                    if (angleCheck < angleTolerance)
                        return;
                    else
                    {
                        RotateDialClockwise();
                        startAngle = currentAngle;
                    }
                }
                else if (startAngle > currentAngle)
                {
                    angleCheck = CheckAngle(currentAngle, startAngle);

                    if (angleCheck < angleTolerance)
                        return;
                    else
                    {
                        RotateDialAntiClockwise();
                        startAngle = currentAngle;
                    }
                }
            }
            else
            {
                if (startAngle < currentAngle)
                {
                    RotateDialAntiClockwise();
                    startAngle = currentAngle;
                }
                else if (startAngle > currentAngle)
                {
                    RotateDialClockwise();
                    startAngle = currentAngle;
                }
            }
```

Fig 4.25 Function for checking angle difference

Finally one more separate script was required to take the rotation results from the Rotator script and translate this into a percentage value from 0-100% and then in the same way the Mix Levels script worked in the previous sprint this percentage can be used to adjust the exposed parameters of a given mixer. The script required to be passed the Mixer that the exposed parameters belonged to and the ID object for that dial so that the script can tell what parameter this dial is intended to alter.

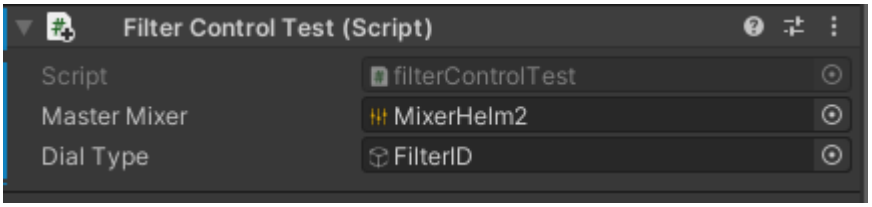

Fig 4.26 Filter Control test Script, this functions similar to the Mix Levels script from before but instead applies some conversion to the dial values before being passed into the synth

```
public void DialChanged(float dialvalue)
{
    //Debug.Log(dialvalue);


    var float requiredFilterPercentage = (dialvalue / 360f) *100f;
    var float ResPercentage = dialvalue / 360f;

    var float filterFlip = 100f;
    var float osc1Value = -36.0f;
    var float osc2Value = 0.0f;

    var float testVar = 0.0f;

    if(dialType.name == "FilterID")
    {
        //messured in 0-100%
        masterMixer.SetFloat( name: "Synth1Cutoff", value: filterFlip - requiredFilterPercentage);
    }
    else if(dialType.name == "ResID")
    {
        masterMixer.SetFloat( name: "Synth1Res", value: requiredFilterPercentage);
    }
    else if(dialType.name == "ReverbID")
    {
        masterMixer.SetFloat( name: "Synth1Reverb", value: ResPercentage);
    }
    else if(dialType.name == "GateID")
    {
        masterMixer.SetFloat( name: "Synth1Gate", value: ResPercentage);
       // Debug.Log("GateTest:"+ ResPercentage);
       Debug.Log( message: "GateTest:"+ masterMixer.GetFloat( name: "Synth1Gate", value: out testVar));
    }
```

Fig 4.27 Filter Control test Script, this functions similar to the Mix Levels script from before but instead applies some conversion to the dial values before being passed into the synth

4.9 Sprint 6

4.9.1 Goal

To begin working on a main mixer board for all the different sequencers/audio outputs so that a User has a main hub with which they can mix the synths from. This requires creating physical sliders instead of the current 2d sliders so that the User can interact with them naturally. After Implementing the mixer I plan to expand further on it by adding in controls for each channel.

4.9.2 Item 1

Created sliders to then implement in a Mixer board where their values correlate to the volume of the mixers channel.

To begin with a free 3d model of an audio slider was acquired online to act as the model for the slider the user would interact with, however this was just a 3d model and processed no script to give it any function or track any slider values. Therefore a script had to be created for it functioning similar to the method implemented for the dials in the previous sprint. The model itself is split into two parts the channel for the slider and the slider knob (see fig 4.29), these are both placed inside an empty game object that will act as the housing to which the script will be attached to track the slider's movements (see fig 4.28). The slider object must also have a configurable joint component attached to it so that its movement is restricted to only up and down.

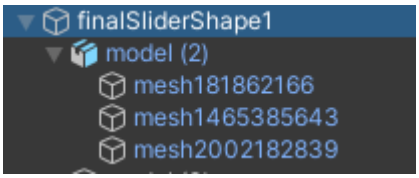

Fig 4.28 Project layout for Slider model.

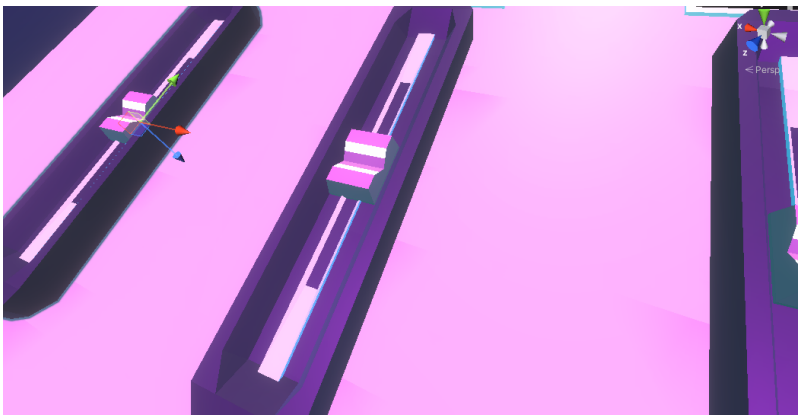

Fig 4.29 Example of visuals and positioning of slider.

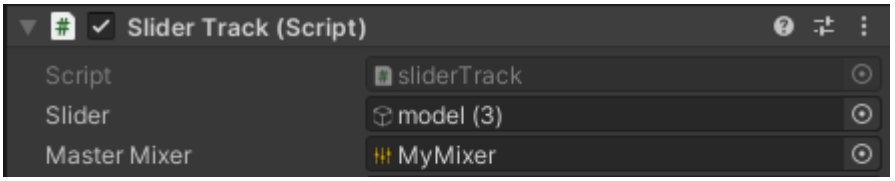

Fig 4.30 The Slider Track script that was created requires that the sliders model be passed into it so that it can track its position from within the game engine.

Inside the Slider Track script that was created to track the position of the slider in the game world, you can see that the start position of the passed in slider is saved in the start function of the script which runs as soon as the application is launched. The position of the slider is then constantly checked every frame and then compared against the start position of the slider, from this a percentage value can be retrieved relative to the position of the slider. This can then be applied to the volume value of the mixer that has been passed into the script (see fig 4.31). Once the script had been developed this slider system could be duplicated for each channel on the mixer for each synth so that a user would have access to a full volume mixing desk (see fig 4.32).

```
0 references
void Start()
{
    startPosition = slider.transform.position.y;
    zeroPos = startPosition - 0.10f;
    topPos = startPosition + 0.10f;
    onePer = (topPos - zeroPos)/100;
    //Debug.Log("Start slider value " + startPosition);
}

// Update is called once per frame
0 references
void Update()
{
    //currentPosDiff = startPosition - currentPosition;
    //zeroPos - currentPosition;

    currentPosition = slider.transform.position.y;



    mixerValue = ((((topPos - currentPosition) / 0.002f) - 100.0f)*-1) - 80.0f;
    masterMixer.SetFloat( name: "Synth1Volume", value: mixerValue);
    text.text = mixerValue + "%";
}
```

Fig 4.31 Slider Track script tracking the location of the slider object.

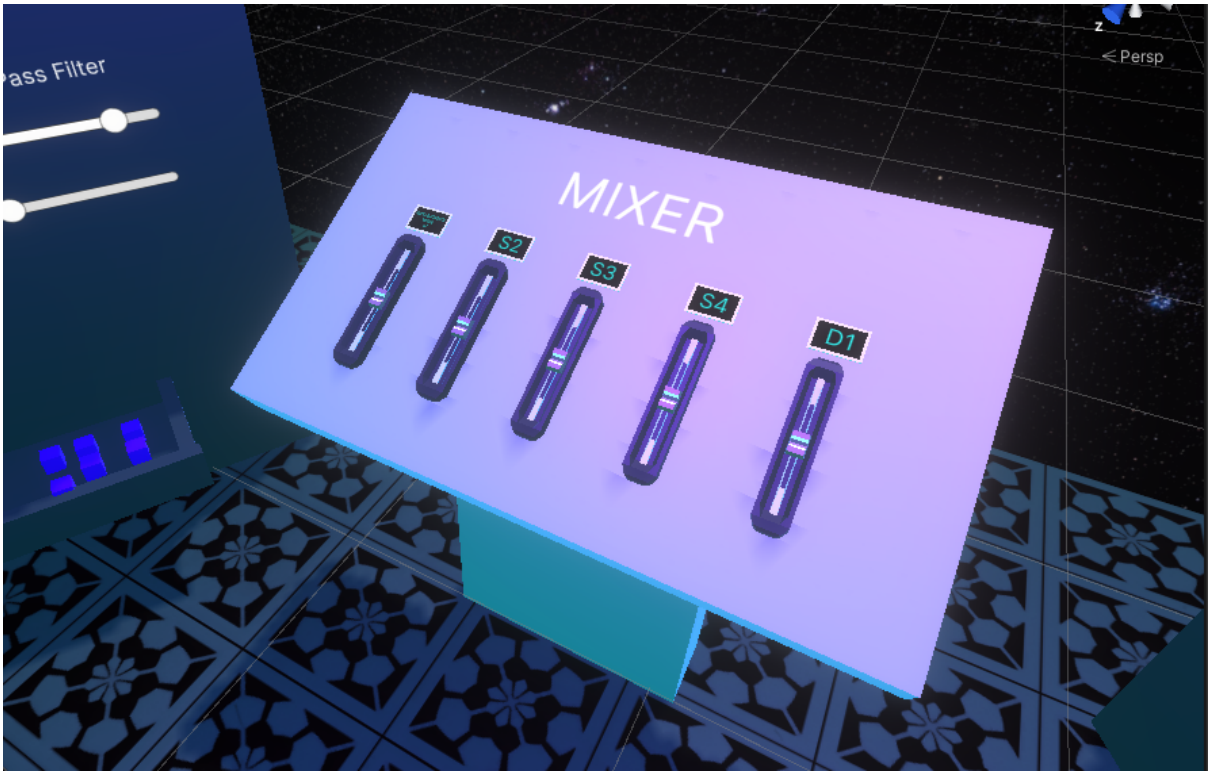

Fig 4.32 Final mixer design with sliders being duplicated out for each synth wall.

4.10 Sprint 7

4.10.1 Goal

Finalise up some of the parameter controls for the synths and the dials used to control them. Once one synth's parameters are dialed in, it can be replicated across all of them. Also create new models for the Users hands and animations for these new hands so as to provide users with a more realistic and immersive experience while also providing more useful and functional hand gestures.

4.10.2 Item 1

Linking more exposed parameters for C# scripts to then be able to interact with these parameters. In Helm parameters are often measured in completely different scales, for example some use a percentage value from 0-100 while others like that in fig 4.33 are a range of semitones from -20 to 256. Because of this it means that certain parameters require extra functions or scripts to convert the dial or slider values into the correct data range for that parameter to understand.

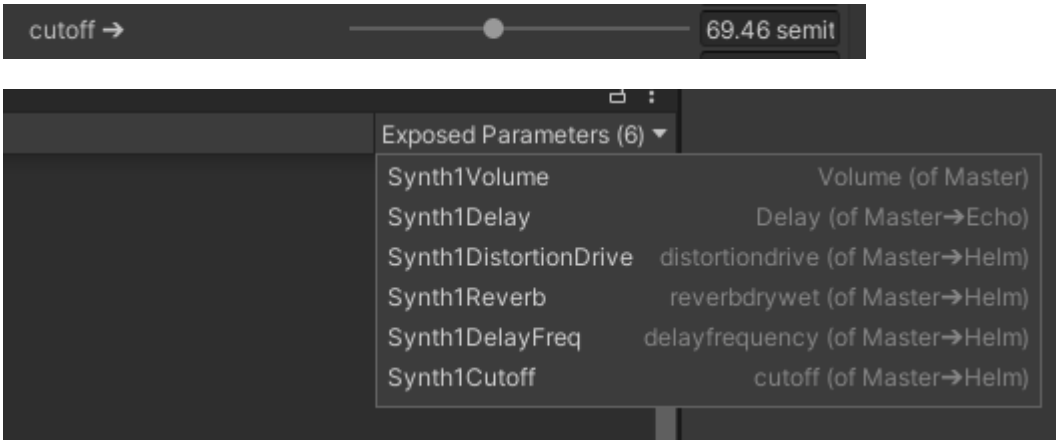

Fig 4.33 Linking more exposed parameters for my C# scripts.

4.10.3 Item 2

Finalising hand models and creating animations for these hands, the main goal was to make it intuitive when the user opens/closes hands to grab things and give the user the ability to point so as to make using buttons a lot easier. In order for Users to be able to interact better with the VR world that had been created, improved hand models would be needed with animations reacting to the users controller inputs. This is important as it will help Users interact with the synth controls and it also improves the immersion of the application. Firstly high res hand models were retrieved from the free unity developer package to use as hand for our User. A left and right model was placed as a child of each hand controller in the XR Rig, this would ensure that the hand model followed the players controller (see fig 4.34 & 4.35.

Fig 4.34 placing of hand models as child objects of XR Rig's controller objects.

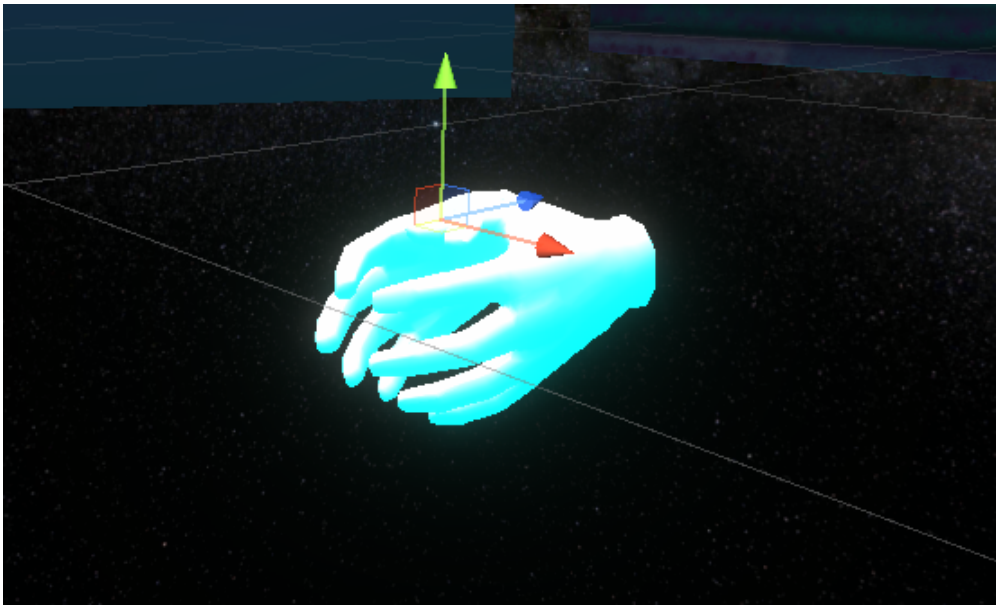

Fig 4.35 Higher res hand models used to represent the users hands.

With the models now following the Users hands they now just needed to react to the users inputs, for this an Animator component will be used with hand animations made from scratch with the Unity record system (see fig 4.36). Once these animations are created they can be manipulated by script to cycle through them based on a value, in this case it will be the grip and trigger button. The trigger representing the index and thumb and the grip of the bottom three fingers of the hand(see fig 4.37), this would allow the user to make an open hand, point, close a fist and any combination between them adding a lot of control to the user's ability to manipulate the environment around them.

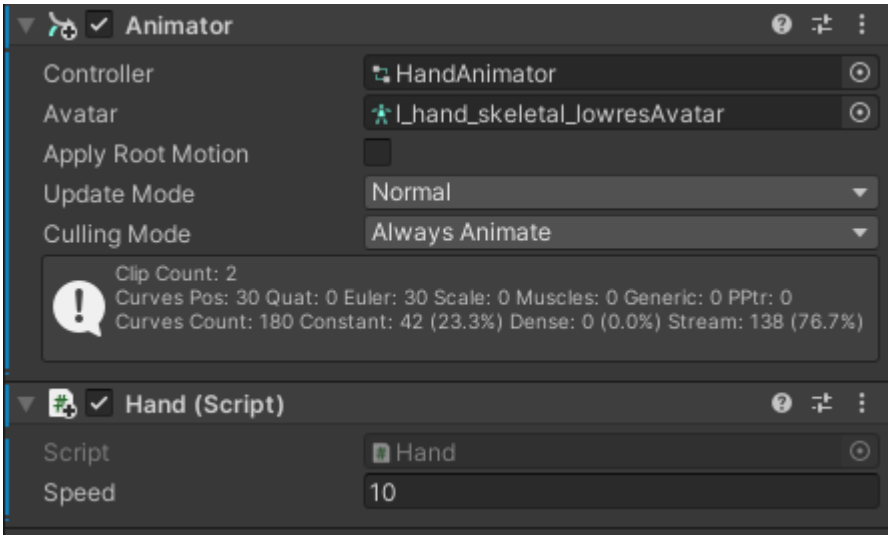

Fig 4.36 Animator component with a custom script to manage the hand actions.

```
void Start()
{
    animator = GetComponent<Animator>();
}

// Update is called once per frame
0 references
void Update()
{
    AnimateHand();
}

1 reference
internal void SetGrip(float v)
{
    gripTarget = v;
}

1 reference
internal void SetTrigger(float v)
{
    triggerTarget = v;
}

1 reference
void AnimateHand()
{
    if(gripCurrent != gripTarget)
    {
        gripCurrent = Mathf.MoveTowards( current: gripCurrent, target: gripTarget, maxDelta: Time.deltaTime * speed);
        animator.SetFloat( name: animatorGripParam, value: gripCurrent);
    }

    if(triggerCurrent != triggerTarget)
    {
        triggerCurrent = Mathf.MoveTowards( current: triggerCurrent, target: triggerTarget, maxDelta: Time.deltaTime * speed);
        animator.SetFloat( name: animatorTriggerParam, value: triggerCurrent);
    }
}
```

Fig 4.37 Script for checking grip and trigger values to then apply them to hand animator.

Fig 4.38 Example of point pose

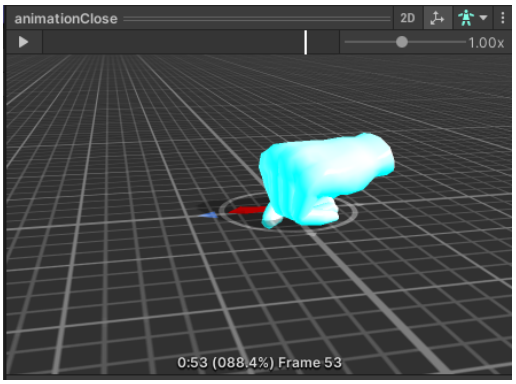

Fig 4.39 closed fist animation

4.11 Sprint 8

4.11.1 Goal

Re-Investigate possibilities of allowing the User to record loops/ or a session in total so that users can listen back to pieces they have created and mixed.

4.11.2 Item 1

With Unity not providing any support or plugin to record in engine audio, instead following research an old C++ script designed for Unity 2015 that allowed capture of all audio channels into a .wav file was discovered. This code was then adapted to c# which the current Unity engine uses, after some tinkering and changes to make the script function properly it is now possible to connect this script to an gameobject with an Audio Listener component attached to it. This Audio Listener will feed all of its audio channels into the Output Audio Recorder script, where functions can then be called to begin recording that audio to a .wav saved locally on the device.

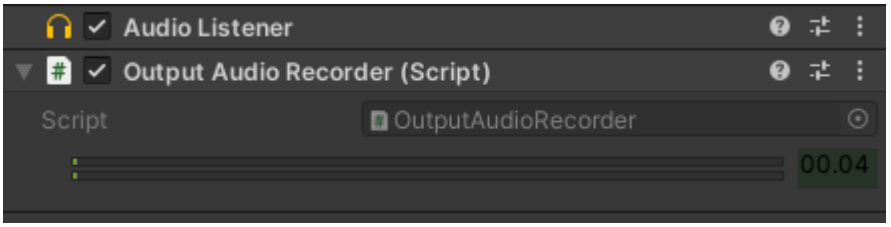

Fig 4.40 Output Audio Recorder script attached to an object with a Audio Listener attached to it.

The control of the recording function is assigned to a button on the Mixer that when pressed calls the StartRecording() function from the script, this will then start to record all Mixer channels in the scene up to a set maximum file size or until the User selects the button again at which point the script will encode the audio into a .wav file (see 4.41)

```
    public void StartRecording(string recordFileName = null) {
        if (string.IsNullOrEmpty(value: recordFileName)) {
            fileName = DEFAULT_FILENAME + FILE_EXTENSION;
        }
        else
        {
            //currently it works only for .wav files so be sure the name is correct
            fileName = Path.GetFileNameWithoutExtension(path: recordFileName) + FILE_EXTENSION;
        }
#if UNITY_EDITOR
        Debug.Log(message: "StartRecording " + fileName);
#endif
        if (!recOutput) {
            StartWriting(name: fileName);
            recOutput = true;
        }
        else {
            Debug.LogError(message: "Recording is in progress already");
        }
    }

    0 references
    public void StopRecording() {
#if UNITY_EDITOR
        Debug.Log(message: "StopRecording");
#endif
        recOutput = false;
        WriteHeader();
    }
```

Fig 4.41 The two functions required to start and stop the audio capture,

Fig 4.42 Basic button UI for recording.

4.12 Sprint 9

In Sprint 9 the main goal was to finalise each of the synth/sequencer walls so that they all functioned as intended and processed UI's that where easy to understand and interact with. Changes to the environment where also implemented with this including the introduction of coloured lighting and a new skybox. Below are images of each of the synth walls towards there final versions.

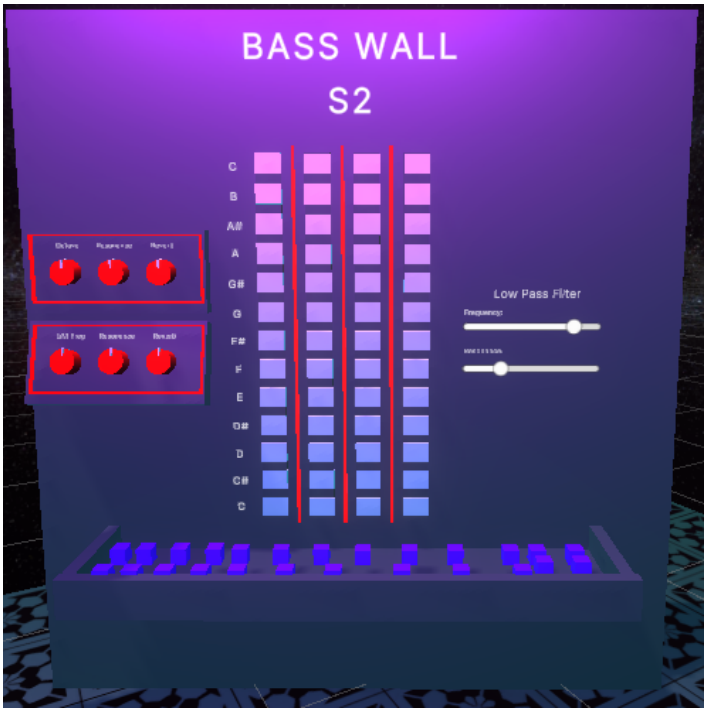

Fig 4.43 A near finished Bass wall, each block represents one bar.

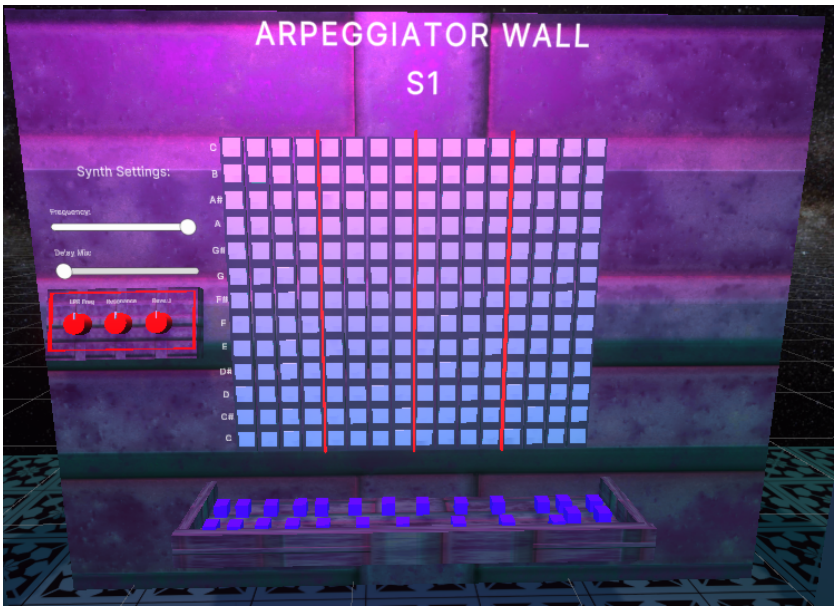

Fig 4.44 Arpeggiator Wall, features a 16 bit pinboard sequencer.

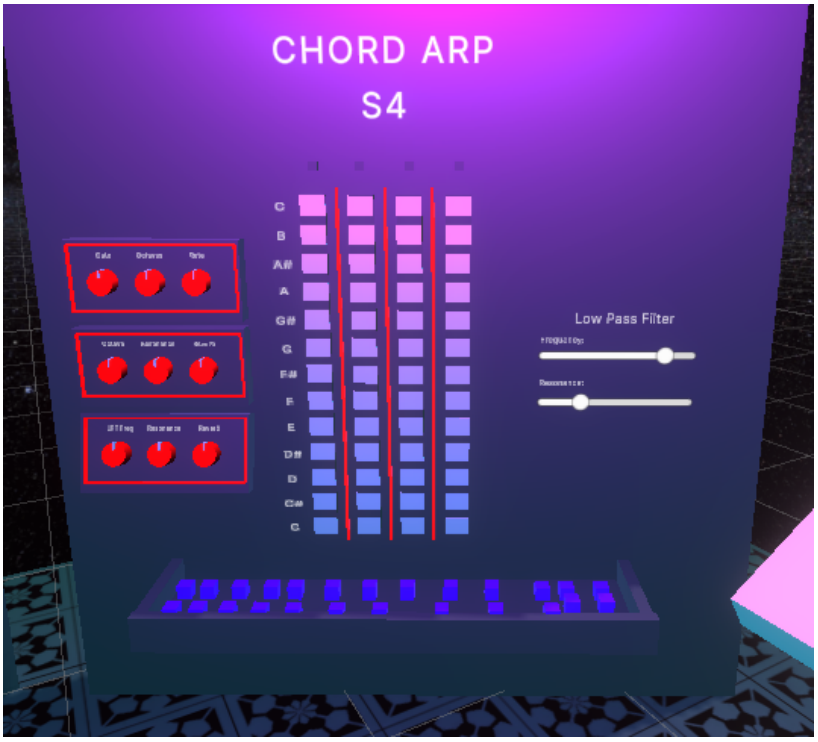

Fig 4.45 Chord arp wall which is capable of arpeggiating full chords, each block is one bar.

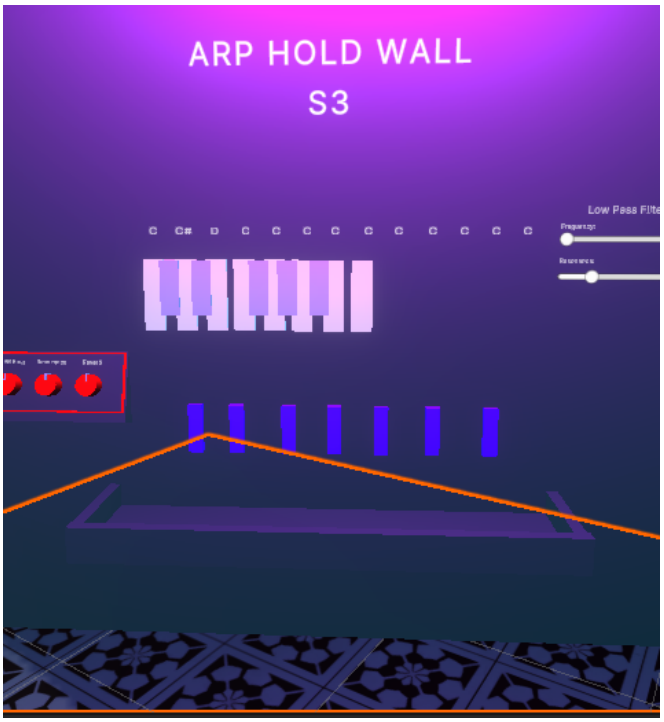

Fig 4.46 Continuos arp wall with a keyboard interface.

4.13 Conclusion

This Implementation chapter shows the technologies that were used for developing the application and what the implementation order of tasks would be throughout the development process. This chapter also explains how the Scrum Methodology is generally used within project management and how it can benefit the development process. Finally this chapter also lays out in detail the various features that were implemented throughout development and during what sprint they were worked on. This chapter helped greatly in keeping the project on schedule and ensured that no aspects of the application were overlooked. It also served as a useful log of development progress.

5 Testing

5.1 Introduction

This chapter describes the testing that has been undertaken for the application. This chapter is presented in two sections:

1. Functional Testing
2. User Testing

Functional testing is a type of software testing whereby the system is tested against the functional requirements. The app is tested by looking to see if the actual output for a given input corresponds with the expected output. The tests should be based on the requirements for the app. The results of functional testing can indicate if a piece of software is functional and working, but not if the software is easy to use.

5.2 Functional Testing

This section describes the functional tests which were carried out on the application. These functional tests can be categorised as: Navigation/Controls and Music Functionality. Functional testing generally uses a Black Box Testing technique which means that the internal logic of the system being tested is not of interest to the tester. The tester is only interested in whether the actual output agrees with the expected output.

5.2.1 Navigation/Controls

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Testing movement	left/right analog	Movement and turning control	Continuous move with snap turn	Working as intended
2	Test Laser pointer for controls/teleport	Click Action button	Laser appears out of hand	Laser appears out of hand	Working as intended
3	Laser based teleportation	Action button, point, then trigger to teleport	teleport to the selected location.	No teleport.	Issue was with floor not recognised as teleport location.
4	Changing hand models for high detail. Testing animations that were created for pointing and making a fist.	Grip and trigger. Grip being the bottom 3 fingers and trigger being the index and thumb.	point and closed fist	Worked after tweaking the animation.	Originally the animation continuously looped which was not the desired effect.

5.2.2 Music Functionality

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Socketed blocks input block to sequence	Socket a block	Sequencer will feature block at placed location	When live plays that note at that location	Using sockets and HELM sequencer script worked as intended
2	Test HELM parameter control with 2D interface	Change 2D slider value to alter exposed HELM parameter	In this test altering the filter is the goal.	Worked but range of slider and range of parameters do not align so inaccurate adjustment.	Creating a script for each param may be necessary to have adjustments aligned.
3	Audio Delay Test	Change the parameter of delay using a slider.	Audio delay based on ms	Does not function as intended, delay was inconsistent.	Workaround found using echo component with a decay. Achieving the same outcome.
4	New chord wall test	Input block which will fill full bar of sequencer	Long bass notes of 1 bar length.	Functioned as intended.	Discovered can also be used for multi note

		instead of 1/16 block as before.			chords if synth is set to poly.
5	Mixer slider test.	Moving the physical 3d slider should alter the mixer audio level.	Changing of channel volumes.	Changes audio but the slider glitches out detaching from its base.	Fix was created using physics based joints with boundaries.

5.3 User Testing

5.3.1 Introduction

The application being developed is a VR application that aims to allow Users the ability to create and record original digital music creations and experience music creation from a different perspective.
The User will learn how to interact with musical sequencers through various 3D interfaces; alongside this Users will be able to control and tweak the parameters of the synth's that are connected to these sequencers. Furthermore, combining these with a mixing channel, Users will be able to create interacting sequences that formulate into a song. When ready they can press the record button and proceed to perform their track live; with the recording automatically being saved as a .wav file to the VR device. The main goal of the application was to make digital musical sequencers more accessible - no matter whether you are an expert or a beginner and reimagine the current approach to digital music's daw's.

.

5.3.2 Tasks

Here are the tasks the User is required to complete for the test:

- Move/Navigate around the space to familiarise themselves with the controls.
- Navigate to any synth wall off your choice.
- Using the blocks provided, create a sequence; play with the parameter controls to fine tune the sound to your liking.
- Repeat this process for another synth.
- Input a sequence into the drum sequence wall by pushing the buttons.
- Navigate to the Mixer wall, adjust and mix your volume levels here.
- When ready press the record button to start recording your performance.
- Stop the recording so it can automatically save.

5.3.3 Describing Participants

The target demographic for this application is quite broad as people of all ages could find enjoyment out of an application like this. However, given that the application is VR based; this means realistically the main demographic are those within the 16-34 age group - as it makes up the largest portion of VR users. Other than that anyone with an interest in music will find enjoyment out of this application and would furthermore make great test participants.

5.3.4 Ease of use / Ease of learning testing

The application is being tested by ease of use to evaluate how easy the application is to use. Testing the User interactions of the application is important and if it is easy for the users to complete the tasks given. This can be done by calculating the number of errors, how long it takes for the users to complete the tasks, the overall relative efficiency, the post task satisfaction and finally the task level satisfaction.

For the testing, a focus is put on performing ease of use testing to see how easily users can navigate the application and complete the tasks. This will give an accurate insight into whether the application is correctly set up and laid out to allow the user to navigate it without much thought, completing the tasks with ease. This can be achieved by calculating the number of errors, how long it takes for the users to complete the tasks, the overall relative efficiency, the post task satisfaction and finally the task level satisfaction. Any extra user interactions that Users may have had that were unforeseen were also recorded.

5.3.5 Test Environment

For the testing environment a personal Oculus Quest VR headset was used. While some early testing was performed using the home computer as the application source being pushed to the headset, eventually my later final test where performed with the application running locally on the oculus quest. This greatly helped the testing environment as it allowed users to test the application untethered, leading to a greater sense of immersion and less external distractions affecting the testing results. It also made testing users externally easy as the personal VR headset was portable, which expanded the range of users the application could be tested on. While Users were testing the device, guidance/aid was always readily available and provided to users during the testing at all times. Apart from this Users were encouraged to test and explore the application. Users were given a list of tasks to complete, but if they instead felt like exploring something else unrelated to the list they were encouraged to do so, as the application is a creative product.

5.3.6 Running the Test / Forms

Introduction:

Welcome to the TempoVR Usability and User experience test run by AaronLynch. Below is a consent form, if you wish to take part in testing the application, I will ask you to fill out this form beforehand. Once completed I will ask you to fill out a pre-test survey to help give me an idea about yourself. Following this you will complete a set of tasks within the application. After you complete the tasks to the best of your ability, I will ask you to complete a post-test survey so I can get accurate documentation of what your experience was in the test and as to whether you have any suggestions to further help improve the application and its ease of use.

Consent form:

TempoVR Consent Form

Owners: Aaron Lynch

Student Numbers: N00182364

Contact Information:

You can me at N00182364@student.iadt.ie

CONSENT

Participation in this usability study is optional. All your information will remain strictly confidential. The description and findings may be used to help improve the TempoVR application. However, at no time will your name or any other identifications be used. You can withdraw your consent to the experiment and stop participation at any time.

First Name: _____Sophie_____

Last Name: _____Egan_____

E-Signature of Participant: _____Sophie Egan_____

Thank You for your participation!

5.4 Conclusion

In conclusion the testing phase of the project was of vital importance and one that continued throughout the entire project. The functionality test performed provided great test information to access aspects of the application that were not performing correctly, it also allowed guidelines to be laid out of different functionalities the application should be able to perform. Another positive from the functionality tests was that it acted as a great test log to look back on if information was needed on how a certain problem was encountered or fixed. The User testing that was performed provided vital information after each prototype, from these tests what aspects to focus on leading into development of the next prototype were evaluated . It also acted as great support to ensure a direction for creating an application that Users would genuinely enjoy using.

6 Project Management

6.1 Introduction

This chapter describes how the project was managed and how well the group worked together as a team. It shows the phases of the project, going from the project idea through the requirements gathering, the specification for the project, the design, implementation and testing phases for the project.

6.2 Project Phases

6.2.1 Proposal

During the proposal phase of this project, brainstorming sessions were held to develop ideas for the project. Over the summer - different areas for project ideas were explored and a decision was made to create something Audio/music related. In the final year, starting Virtual Reality as a module was inspiring, combining this inspiration with a love for music encouraged the idea to create an audio based application. From here; different possible audio applications that could be created in VR were explored, the possibilities were huge. The creation of digital music through the use of a DAW (digital audio workstation) such as ableton or fruit loops has always been of interest; in particular - the use of digital recreations of classic synthesisers and sequencers in order to create digital music. Unfortunately, owning physical versions of many of these audio devices is very expensive and not easily attainable for many people; therefore a decision was made that this would be the perfect candidate for the creation of this proposal.

The creation of a Virtual Reality space wherein Users can interact with real synthesisers and sequencers reimagined and experienced from the perspective of a virtual world; allowing users to mix, create and record original tracks all from a Virtual Reality headset.

6.2.2 Requirements

The project began by evaluating and researching similar VR applications that emphasised the use of audio creation technology. The research was expanded furthermore to include all music based VR/web applications as there are currently very few active examples of VR applications for creating digital music. Through this research and evaluation all advantages and disadvantages of the applications were listed and contemplated - which helped define possible issues in the future and aided in identifying any features or functionalities that could be implemented in the VR application. During this time, a list of all the functional and nonfunctional requirements that the application would have was created, this helped with beginning to construct a draft version of interview and survey questions.

Following this interviews were conducted and an online survey was published - with the intention to determine how Users felt about the concept for the VR application, as well as what features Users would find interesting in an application such as this. Two interviews were conducted in total; this provided great insight into what fundamentals the User would require within the application. The online survey provided general feedback about how the application may be received by Users, as well as how certain UI features might be used and once what features Users may be personally interested in.

From all of the data collected in both the interview and survey phases; user personas were created. These personas were given accurate attributes, goals, feelings, jobs and concerns. These personas were then thinned down to two; - picking those that accurately represented the type of user that would most likely use the application. Finally, it was important to explore what sort of technologies would be needed to create an application such as this; after the extensive User research a realisation occurred that the feeling of freedom was most important to Users, therefore a decision was made to develop in Unity - as this works best with the Oculus platform and which would allow the User an untethered experience.

Next a decision was made to use a synthesiser framework called HELM (a framework which was originally intended for creating procedurally generated game music), but instead could be manipulated through C# code to use as a full blown synth engine.

6.2.3 Design

The design phase of the application was completing all the environment, sound and UI design of the application - this aided to give a strong idea of how the different parts of the application would be created. This began by firstly; designing and evaluating various paper prototypes of the application, and then iterating on each of these prototypes, tweaking different factors until the design was completed. To help make these decisions User scenarios from the chosen persona's were developed. This gave an accurate insight into tasks Users would want to perform while using the application. After a general design and user experience was decided upon, digital wireframes of the applications design and paper prototypes of possible environment designs were created. Finally, accurate photoshop prototypes were created of the various possible sequencer interfaces, these were then shown to test users to gauge a reaction to a polished representation of what the final application UI should look like. This helped give vital feedback on mostly the UI/content layout.

6.2.4 Implementation

Leading into this phase of the project, emphasis was put on working towards implementing all of the completed design work from the previous phase of the project. With a good idea about how to approach the general backend and frontend components of the application, building a basic prototype was the goal. This prototype was created to test the implementation of the required components and assets for any VR application and to implement a test of the HELM synthesiser asset in use. Upon completion of the prototype, a working game environment was created - with an XR rig for the User to control, basic hand models with object interaction and a working test interface for the HELM synth which the User could produce sounds

from. This was an excellent start to the implementation as it provided feedback on what would work in the design plan and what parts may need to be changed.

Following this first prototype, working on a second prototype was the next step, this one with the intention of creating systems and scripts to use and control HELM's sequencer scripts. In order to achieve and test this, a rudimentary drum machine was created, where clicking any of 16 buttons in a given row would input a block at that point onto a sequencer, then would play the associated sound at that time. This worked great and when replicated out for each sound on the drum machine, the User now had the ability to create full drum sequences. Now with the use of a method for scripting physical interfaces to interact with the sequencers, this method could be extended on to go an extra step and have the sequencers interact with the HELM synthesiser also. Once this was achieved it was easier to replicate and test different interface designs and create multiple individual sequencer walls all running their own unique synth. This completed the second prototype, which allowed more testing to be performed so the design could be finalised going into the last few sprints of implementation.

The final prototype created was near the end of the implementation sprints and bears the most resemblance to the final application. This prototype was created with the intention of finalising the design and function of all the synth/sequencer interfaces. Firstly more complex hand models were created; with custom animations for hand poses, allowing users to interact with buttons/controls easier as this was a suggestion made from earlier tests. Next dials and sliders were also created from scratch, with custom scripts to calculate and pull values from them based on their position. These new controls were connected to exposed parameters on the various synths allowing Users to control the sound the synths were creating. Finally a mixer was created so that Users could mix their various together into songs, later with the ability to record sessions to a .wav file from the mixer.

6.2.5 Testing

The testing phase of the project was split into two sections: functional testing and User based testing. For the functional testing, the main goal was to create some functional requirements the application would have to meet during its implementation phase in order to work correctly. These requirements were ever changing and growing as the project was being developed. The app is tested by checking if the actual output for a given input corresponds with the expected output. In the case of the application, these functional tests can be categorised as: Navigation/Controls and Music Functionality. The benefit of this style of testing is that the results of functional testing can indicate if a piece of software is functional and working as intended. It also acts as a log of the various problems encountered/solved while creating the application.

The second method of testing performed was User testing, this step in the testing phase was significant when creating a VR application. The User experience is very important as bugs/issues with an application can not only be off putting to the User, but in VR it can make them feel physically ill. Due to this, a decision was made to hold multiple User tests at different stages of the development, using my prototypes as the test application. Three prototypes of the application were created in total, each performing a User test. The aims of each prototype User test were different, each prototype tested different aspects and each test more complete than the last. However, the general goal of the tests was the same - to ensure that the application that the User was presented with was: easy to understand, easy to navigate, comfortable and provided not just audio creation but a musical experience. While Users were given a short list of tasks to complete for the test, Vr applications are very different to the tests, as once the User is in the application they do not have access to the list so eventually tend to just explore. This enhances the testing experience as you get very honest feedback on your application, as the User is not as restricted or guided as with most User tests.

6.3 SCRUM Methodology

Once certain technical issues were resolved with setting up Unity development from home, the higher priority requirements which had been decided during the Requirements phase of the project could then be focused on. In the early sprints, a focus was put onto creating a strong vision of what the final design and feel of the application would look like. This helped greatly in the later sprints, as once the main functionality was created, the focus could be put towards the previously polished environment/interface design, using it as a template to both work from and implement into the project. The sprints helped keep in mind what should already have been completed and what to move onto at different phases of the project. This is explained in more detail in the Implementation section of this report.

6.4 Project Management Tools

6.4.1 GitHub

Github was used so that all of my Unity project files could be stored online in a repository, this was essential for multiple reasons. One very important reason was to provide project backups in case the main file was corrupted, while also providing an easy way to copy the project from one device to another. Unity projects are extremely large and complex file structures and therefore can often be quite tricky to copy and move from one computer to the next, GitHub completely avoids this issue. In order to perform all of the commits and pulls the GitHub desktop app was used. A second reason to use GitHub was that as new commits were created and pushed to the repository GitHub acted as a great project log, keeping track of the progress on the project and providing the ability to return to previous builds of the application if necessary.

Below is a link to the project GitHub Repo:

https://github.com/AaronIADT/TempoVR

6.5 Reflection

6.5.1 Your views on the project

I thoroughly enjoyed working on my project as it was based on a topic I already had quite an interest in. I had been interested in and making digital music as a hobby within different DAW's (digital audio workstation) for years and it had always been a goal of mine to create an application related to digital music. While the sprints for the project did not begin until second semester, I was already in semester one continuing to use different digital DAW's noting features of the applications or their digital instruments. It was because of my Virtual Reality module that I got introduced to VR as a development medium, from here the idea to combine digital music creation with Virtual Reality was born. Because I already had quite a clear idea by the sprint beginnings of what I wanted to create, this helped speed up the early phases of the project involving research and design. Weekly meetings with my supervisor ensured that all the pre-planning phases were completed on schedule, and this provided a very positive start.

Moving into the Implementation phases of the project nearer the later sprints the workload began to increase as I still had a lot of documentation to write while beginning to build the application. Having only started using Unity that year, at the beginning there was a large learning curve to the Unity UI but my strong coding background aided me heavily with the scripting aspects of the project. Through the Implementation process I encountered various issues with my intended approach to creating various aspects, but fortunately through use of online learning tools and advice from my supervisor nearly all problems were solved. Fortunately development of the application as a whole went very smoothly which allowed me plenty of time to work on the documentation for the project, reviewing User testing and further innovation on the application design. I also found that positioning the project in the second half of the year without other modules running at the same time to be very helpful. This allowed me to dedicate the time required to create a large software project.

6.5.2 Completing a large software development project

Working on large software development projects can be quite intimidating sometimes, as the scale of all the different elements that must be completed and co-exist together can seem like too much to manage. At first it can be hard to balance task's but as the year went on I found it easier to prioritise the correct things. The biggest lesson I learned from completing a large project like this is that planning and following that schedule strictly is essential, otherwise certain things always get left until too late. Keeping track of your progress allows you to view your project workload in a more positive way because as you complete tasks you actively see your workload get smaller and all you've accomplished so far. In order to manage large software projects it is essential to maintain some form of version control such as GitHub, without this large projects become hard to maintain order in and it can become very hard to reverse inevitable errors and mistakes in the project. These methods also help greatly when collaborating with others or simply just require help with an aspect of the project.

6.5.3 Working with a supervisor

Having a supervisor on a project on this scale was very helpful. My supervisor Timm ensured I stayed on schedule and was there to help me with approaching the various problems I faced during my project. Timm met with me weekly to keep me focused on what was to be completed during each sprint and helped me greatly with completing and formatting my write ups. I personally found myself to be in quite a lucky situation as my supervisor Timm had a shared interest in the project I was attempting as it was based in the same field as a project he created in the past. This was of great help as it always felt like Timm had great insight to provide on all aspects of the project and when I had questions for him, I was not having to explain very much as he was familiar with many of the audio concepts I was speaking about. Finally Timm continuously reassured me about the work I had completed and offered honest advice, like where improvements could be made. This helped with my confidence as it reaffirmed that the work I had done so far was moving in the right direction, and this in turn motivated me further.

6.5.4 Technical skills

Personally I feel I have gained a lot of technical skills from completing this project, having never created a 3d Video game or Unity project this was completely new to me. With the project being completely based within the Unity game engine, I've had to familiarise myself with the software on an in-depth level. I now possess the ability to produce full VR ready application in the Unity engine having gained technical skills in Unity, Animation, Modeling and Scripting in C#. In particular the scripting aspect of this project has increased my coding abilities a lot, as previously I had not worked with C# as a language and had never had to create code based scripts that then interacted with framework components.

Another technical skill I gained during this project would be working in VR as a development medium, this in itself comes with many individual small technical skills to be mastered. Following this project I feel I am capable at creating applications that can cater to a VR experience. As well as that I found that I learned to put my design skills to use and become more time efficient with them. I have always enjoyed UI design and mocking up prototypes/interfaces, but this year has taught me to be much more efficient with my time and come out with more polished work because of it.

6.5.5 Further competencies and skills

Further competencies and skills that could help me in the workplace becoming more proficient are possibly improving my abilities at using common games/VR frameworks, as these frameworks provide a lot of great features/tools that can save developers time to focus on the important aspects of the project. While I gained invaluable skills creating many of the core functionality aspects of my project, a lot of development time could have been saved if I had used a framework for certain aspects. This would have translated into a new skill as many professional development companies use these same frameworks. As well as this, increasing my abilities at the other aspects of Video Games development such as animation and modeling would definitely be beneficial to me in the future as my project required me to learn the basics, becoming quite important to create the application. Developing these skills further would lead to designing more unique and interesting applications in the future.

6.6 Conclusion

In this chapter the developmental process of the final year project is discussed, where the aim was to create a VR application of which Users could create original electronic music in VR. The phases of each task completion, the issues faced on the way and the steps taken to solve these issues are explained. Reflecting at the various aspects of working on large projects like this with a supervisor or on a solo project. Managing the organisation of the project through the use of GitHub,as well as Project Log and Supervisor meetings is shown. Through this section the views looking back on the project are described, providing insight into things that have been learned and skills gained because of the project. The project management for this project has been extremely helpful and was a net positive on the project as a whole.

7 Conclusion

The main aim of this project was to create a VR application that could introduce people to a new medium to interact with and create digital music in. Currently digital music creation has two main mediums, creating through software in flat applications from a screen or through recording in analog sessions using an old synthesiser. With this application the aim was to mould these two together into a format that was approachable from a VR perspective. As music can be quite a powerful and immersive experience, it makes sense that VR music applications are quite popular. The goals of the project were to also acquire as many new skills as possible so as to improve development projects in the future or to continue development of this application in the future.

The technologies used in this project were the Unity Game Engine and Helm Audio plugin for Unity. Unity was chosen as it is considered to be one of the most powerful and approachable game engines out there to develop with especially with the oculus quest which was the main target platform for my application. Unity comes equipped with its own physics engine, great UI creation tools, animation controls, audio mixing, particle systems and the Unity asset store allowing developers to find any asset they may need. Helm is an open source software based synthesiser that can be used to create electronic music from a computer. What makes Helm stand out from many other paid/unpaid VST synths, is its ability to work and run natively in Unity and other platforms. Helm will allow me to provide Users with access to a real synth in the engine rather than the common approach of sampling.

Conducting user research during the requirements phase of the project provided a more in-depth idea of the target user and increased knowledge on the non-functional and functional requirements of the application. After conducting interviews, gaining insight from test users and publishing online surveys, the users' needs were appreciated and noted. Ideas of features were given that could be helpful and used to improve the application. Studying other competitors in the VR music field also helped a lot, especially to see what standards were already required in the industry.

Next the design phase of the project began, during this phase design decisions about the application were laid out while taking all of the research conducted previously into consideration. Through use of wireframes basic synthesiser interfaces were mapped out and designed with associating colour palettes too. These were then used as guides to create higher fidelity photoshop prototypes of these interfaces so that the synth walls could be planned out in advance so that when the implementation began a clear direction for the application was in mind. Finally a paper prototype was created to map out the Environment design, this would determine the layout of the application and lighting decisions.

The Implementation portion of the project was where all of the concepts planned and discussed in the previous section became relevant as it was time to create them. This was definitely one of the longest and on-going phases of the project as there were constantly new functions and designs being implemented to improve the User experience. In this section the synth wall prototyped in the design phase where created utilising a socketting system to create musical sequences, from here other types of synthesisers were developed to add a variety for the User to play and mix. Functional and non-functional requirements were also implemented here to ensure that all aspects of the application were developed and worked on.

Functional testing was carried out to ensure that the features that had been added were all functioning correctly and performing without bugs or crashes. The user testing allowed insight into how testers would interact and perform tasks in the application. These tests were performed in person so that assistance and feedback could be provided if needed as VR applications are quite new to many Users. This user testing provided vital insight into how users interpreted the application and some changes that should be made.

Overall the final application created by the end of the project is most definitely a success. The intended goal of creating a VR application that Users could create original music in with a key focus on sequencers was definitely achieved. The application can run multiple complex sequences and drum rhythms, all keeping in

sync with one another through a global clock. The difficulties met through the development of this project have helped develop new important skills that will be very useful in future development situations. This project also heavily used GitHub for version control, this provided great support throughout development as Unity projects are huge complex files. GitHub ensured that changes to the application could be tracked and backups or rollbacks were always available.

The project could definitely be developed much further, the application could benefit greatly from some standard VR frameworks that are mostly inexpensive. These would free up development on standar VR functions so that further development could be spent on the sequencers/synths. More synth parameters and effects could also be added to further the Users control over the sounds their synths produce. Finally, provide Users the tools to create more complex melodies with blocks that can adjust their length so as to play long notes and short in quick succession on the same sequencer. This would greatly add to the abilities of the application.

References

Audio Helm: Main Page. (2022). Retrieved 24 April 2022, from https://tytel.org/audiohelm/scripting/

Unity with MVC: How to Level Up Your Game Development. (2022). Retrieved 8 May 2022, from https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development

Reference: Gerrard, B. (2016). Analytics, technology and high-performance sport. In Critical issues in global sport management (pp. 227-240). Routledge.

Cyberdream VR | Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound. (2022). Retrieved 8 May 2022, from https://dl.acm.org/doi/abs/10.1145/3356590.3356637

Unity - Developing Your First Game with Unity and C#. (2015). Retrieved 8 May 2022, from https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp

Audio Helm for Unity. (2022). Retrieved 1 May 2022, from https://tytel.org/audiohelm/

Comfort and Accessibility - Unity Learn. (2022). Retrieved 5 May 2022, from https://learn.unity.com/tutorial/3-1-comfort-and-accessibility?uv=2020.3&courseId=60183276edbc2a2e6c4c7dae&projectId=6018353dedbc2a0f634b7918

Cuadrado, Francisco. (2015). The use of sequencer tools during the composition process: A field study. Journal of Music Technology and Education. 8. 55. 10.1386/jmte.8.1.55_1. from

https://www.researchgate.net/publication/281288726_The_use_of_sequencer_tools_during_the_composition_process_A_field_study

Checa, David & Bustillo, Andrés. (2020). A Review of Immersive Virtual Reality Serious Games to enhance Learning and Training. Multimedia Tools and Applications. 79. 10.1007/s11042-019-08348-9.

https://www.researchgate.net/publication/337951952_A_Review_of_Immersive_Virtual_Reality_Serious_Games_to_enhance_Learning_and_Training

Technologies, U. (2022). Unity - Manual: Audio Delay Effect. Retrieved 8 May 2022, from https://docs.unity3d.com/560/Documentation/Manual/class-AudioDelayEffect.html

VR Software Setup - Unity Learn. (2022). Retrieved 8 May 2022, from https://learn.unity.com/tutorial/0-1-set-up-unity-and-your-vr-device?uv=2020.3&courseId=60183276edbc2a2e6c4c7dae

Setting up Virtual Reality and Augmented Reality Learning Environment in Unity. (2022). Retrieved 8 May 2022, from https://ieeexplore.ieee.org/document/8088512

Weinel, J. (2020). Visualising Rave Music in Virtual Reality: Symbolic and interactive approaches, from https://doi.org/10.14236/ewic/eva2020.13

Wikipedia Contributors. (2021, July 10). Music visualization. Wikipedia; Wikimedia Foundation, from https://en.wikipedia.org/wiki/Music_visualization

The Basics of Music Visualizers - IPR. (2019, December 6). IPR.17 from https://www.ipr.edu/blogs/audio-production/the-basics-of-music-visualizers/

Mirzaei, M., Kan, P., & Kaufmann, H. (2021). Head Up Visualization of Spatial Sound Sources in Virtual Reality for Deaf and Hard-of-Hearing People. 2021 IEEE Virtual Reality and 3D User Interfaces (VR). https://doi.org/10.1109/vr50410.2021.00083

Choy, C., & Reich, M. (n.d.). Music Visualizer and Synesthesia Simulation in Virtual Reality from http://jmgphd.com/wp-content/uploads/2019/06/group3ssmviz_Final-Report.pdf

Mäki-Patola, T., Laitinen, J., Kanerva, A., & Takala, T. (n.d.). Experiments with Virtual Reality Instruments. Retrieved November 3, 2021, from https://www.nime.org/proceedings/2005/nime2005_011.pdf

Ng, K., Armitage, J., & Mclean, A. G. R. (2014). The Colour of Music: Real-time Music Visualisation with Synaesthetic Sound-colour Mapping from https://doi.org/10.14236/ewic/eva2014.3

Bates, E., & Boland, F. (2016). Spatial Music, Virtual Reality, and 360 Media. Audio Engineering Society. Retrieved from https://www.aes.org/e-lib/browse.cfm?elib=18496