Recorder Academy: A Gamified Approach to Beginner Music Practice

Alex Killian

N00212207

Supervisor: Timm Jeschawitz

Second Reader: Cyril Connolly

Year 4 2024/25

DL836 BSc (Hons) in Creative Computing

Abstract

'Recorder Academy' is a game developed to make recorder practice more engaging for children and young adults. Drawing on Zichermann and Cunningham's (2011) definition of gamification as "the process of game-thinking and game mechanics to engage users in solving problems," the project aimed to use gameplay mechanics and gamifications elements to motivate consistent music practice. This project is developed using the Unity game engine and used real time pitch estimation technology, the application was built using an iterative design process, with individual features tested throughout and user testing conducted once a usable version was completed. Findings suggest that a gamified approach can increase motivation and enjoyment in music practice. With additional time 'Recorder Academy' would allow users to practice their favourite songs to further improve engagement and motivation and will be optimised to allow users to practice a range of instruments.

Acknowledgements

I would like to sincerely thank my supervisor Timm Jeschawitz for the support and assistance during the design and development process. I would also like to thank my family and close friends for their love and support.

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 to leave copies of your own files on a hard disk where they can be accessed by other. 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: Alex Killian

Signed

_____

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

Table of Contents

Table of Figures 8

1 Introduction 1

2 Research 2

3 Requirements 4

3.1 Introduction 4

3.2 Requirements gathering 4

3.2.1 Similar applications 4

3.3 Requirements modelling 6

3.3.1 Personas 6

3.3.2 Functional requirements 8

3.3.3 Non-functional requirements 8

3.3.4 Use Case Diagrams 8

3.4 Feasibility 9

3.5 Conclusion 9

4 Design 10

4.1 Introduction 10

4.2 Program Design 10

4.2.1 Technologies 10

4.2.2 Structure of Unity Program 11

4.2.3 Application architecture (1 page) 23

4.2.4 Process design 24

4.3 User interface design 24

4.3.1 Wireframe 25

4.3.2 User Flow Diagram 26

4.3.3 Style guide 27

4.3.4 Storyboard 28
4.3.5 Environment 30
4.4 Conclusion 30
5 Implementation 32
5.1 Introduction 32
5.2 Scrum Methodology 32
5.3 Development environment 33
5.4 Sprint 1 – Ideation and proposal 33
5.4.1 Goal 33
5.4.2 Proposal 33
5.4.3 Research 34
5.5 Sprint 2 – Functional Requirements and prototype V1 34
5.5.1 Goal 34
5.5.2 Functional Requirements 35
5.5.3 Prototype V1 35
5.6 Sprint 3 – Interim Presentation and Amplitude Prototype 36
5.6.1 Goal 36
5.6.2 Amplitude prototype 37
5.7 Sprint 4 – Prototype V2 Button Input 38
5.7.1 Goal 38
5.7.2 Development on audio input 39
5.7.3 Button Input Prototype 40
5.8 Sprint 5 – Prototype V3 40
5.8.1 Goal 41
5.8.2 Win Condition Rework 41
5.9 Sprint 6 – Additional Functionality 42
5.9.1 Goal 43
5.9.2 Note Instantiation 43

5.9.3 Point System adjustments 44
5.10 Sprint 7 - Design 45
5.10.1 Goal 45
5.10.2 Note Element 45
5.10.3 Screens 45
5.10.4 Visual Feedback 46
5.11 Conclusion 46
6 Testing 47
6.1 Introduction 47
6.2 Functional Testing 47
6.2.1 Pitch Estimation 47
6.2.2 Point System 48
6.2.3 Navigation 48
6.2.4 Discussion of Functional Testing Results 49
6.3 User Testing 49
6.4 Conclusion 53
7 Project Management 54
7.1 Introduction 54
7.2 Project Phases 54
7.2.1 Proposal 55
7.2.2 Requirements 55
7.2.3 Design 56
7.2.4 Implementation 56
7.2.5 Testing 57
7.3 Reflection 57
7.3.1 Your views on the project 57
7.3.2 Completing a large software development project 57
7.3.3 Working with a supervisor 58

7.3.4 Technical and project skills gained 58
7.4 Conclusion 58
8 Conclusion 60
References 62

Table of Figures

Figure 1 Yousician gameplay screen 5
Figure 2 Guitar Hero gameplay screen 5
Figure 3 Guitar shaped controller used for Guitar Hero 6
Figure 4 Persona 1 potential user of 'Recorder Academy' 7
Figure 5 Persona 2 a character who would benefit from 'Recorder Academy' without necessarily using the application 7
Figure 6 Use Case Diagram for 'Recorder Academy' 8
Figure 7 'Recorder Academy' Home Screen 11
Figure 8 Home Screen scene element breakdown 12
Figure 9 Instruction page for ‘Recorder Academy’ ..... **Error! Bookmark not defined.**
Figure 10 Button functionality on home screen script 13
Figure 11 Build Settings 13
Figure 12 Main Game screen on first view 14
Figure 13 Timer Script 14
Figure 14 Where remaining time is set in the unity inspector 15
Figure 15 Note game object 15
Figure 16 Script handling note movement 15
Figure 17 New note trigger wall. Outlined in green 16
Figure 18 Script handling block instantiation 16
Figure 19 Cube status script 17
Figure 20 Names array 17
Figure 21 name position map 18
Figure 22 positions array 18
Figure 23 Target area with trigger attached outlined in green and orange 18

Figure 24 on trigger enter invoking start recording 19
Figure 25 audio recorder script getting frequency and converting to name..... 19
Figure 26 debug log output when D is played 19
Figure 27 Reading text on TextMesh Pro component..... 20
Figure 28 Debug log showing the required note..... 20
Figure 29 comparing required note to dectecting note to give a point 20
Figure 30 Total points counter 21
Figure 31 Debug log feedback when correct note is played..... 21
Figure 32 Point awarded text that appears as point is given..... 21
Figure 33 Target area turns to green when point is awarded 21
Figure 34 Target are turn red when incorrect note is played..... 22
Figure 35 Debug log when incorrect note is played..... 22
Figure 36 incorrect note count 22
Figure 37 Game over screen 22
Figure 38 Block Diagram for 'Recorder Academy'..... 23
Figure 39 Pseudo code hand written..... 24
Figure 40 First preliminary wireframe of 'Recorder Academy' 25
Figure 41 Second wireframe for 'Recorder Academy'..... **Error! Bookmark not defined.**
Figure 42 Wireframe 2..... 26
Figure 43 Figma prototype..... 26
Figure 44 Colour Palette 27
Figure 45 first scene of storyboard 28
Figure 46 Second scene of storyboard 29
Figure 47 Scene 3 of the storyboard 29
Figure 48 Last scene in storyboard..... 30
Figure 49 Scrum methodology diagram 32
Figure 50 Recorder Academy first prototype 36
Figure 51 Amplitude prototype code 37
Figure 52 Clap threshold variable – minimum value necessary to detect a peak..... 38
Figure 53 Intended output when peak was detected 38
Figure 54 First iteration pitch estimation script..... 39
Figure 55 Button input getting the letter from the text mesh pro component..... 40
Figure 56 Visual aid for above paragraph..... 41
Figure 57 Code enabling this functionality 42

Figure 58 Block instantiation code at random y value 43
Figure 59 instantiating new cube and only editing that cubes text component 43
Figure 60 code checking is the cube status script is on cube if not adding and if points awarded is true not adding another point 44
Figure 61 Code giving point and changing point awarded to true 44
Figure 62 Cube status component 45
Figure 63 Preliminary screening questions 49
Figure 64 Infographic of age breakdown of testers of Recorder Academy 50
Figure 65 Demographic of testers that have music ability 50
Figure 66 Different music instruments that testers had ability in 51
Figure 67 User testing results on game mechanics 51
Figure 68 User testing results on user experience 51
Figure 69 Tester number 3 response to the opinion questions 52
Figure 70 Tester number 5 response to the opinion questions 52
Figure 71 Tester 2 and 6 response requesting adding a song feature 53

1 Introduction

Everything has the potential to be fun (Zichermann, Cunningham 2011). The fun quotient believes that anything can be made fun. Think of video games like FarmVille and PowerWash Simulator. These hugely popular games are based on such mundane tasks as planting crops and washing houses. What makes this game fun isn't the activity but game mechanics. The aim of 'Recorder Academy' is to use this philosophy of the fun quotient and apply it to the act of practicing a musical instrument. More specifically the recorder.

The goal of 'Recorder Academy' is to create a video game for PC platforms that will turn practicing the recorder and other instruments into something fun and engaging. This will help keep children and young people committed and motivated to keep playing music.

This application can be applied in classrooms to aid teachers to motivate students in lessons and modernize music class and cater to youths of the digital age. While 'Recorder Academy' is not designed to replace the traditional classroom experience it can be an aid to teachers and student to enhance engagement and motivation.

'Recorder Academy' is developed in the Unity game development engine. The engine based in C# allows creation of 3D and 2D games. Scripts are written in C# and Unity's in-built TextMesh Pro component will allow for clear and aesthetic UI design. Additional design assets will be provided by the Unity Asset Store with credit to the original artist

Project management was handled in Miro. With notes written and images collected based on each sprint on the development had taken place during the sprint.

'Recorder Academy' Requires players to play a given recorder note at the correct time to get a point. A moving note game object moves down the screen towards the player once this note hits a target area it triggers a pitch estimation script that uses FFT and SRH algorithm to convert the note being played to the note value using its distinct frequency. The moving note objects have a note letter printed on them (i.e. A, B etc.) after the note that is being played gets converted to a letter value it is compared to the letter value on the note object. If these two letters are the same the player receives a point.

As stated, the main design elements of 'Recorder Academy' are moving notes that move along a musical stave. Inspiration for this design was taken from the 2005 Harmonix game Guitar Hero. Additionally, as 'Recorder Academy' is targeted towards children and young people visual design will be colourful with changing colours aiding visual feedback. For instance, when a correct note is played and the player earns a point the target area turns green and when the inverse occurs the target area turns red.

With each new functionality added to ‘Recorder Academy’ during development it was tested, and adjustments were made as needed. User testing was done to measure the games effectiveness, playability and for any errors that may have been missed during the development.

2 Research

Research done for ‘Recorder Academy’ fell into two categories:

- Background Research
- Technical Research

Background research regarded gamifications effects on music education and how gamification of day-to-day tasks can increase motivation and engagement.

Researched highlighted key features such as point systems, leaderboards, and badges that can increase student engagement. Point systems help increase motivation as they create a scene of accomplishment in the player. A player seeing their high score raise encourages the player to put more time into the game. Leaderboards play into the human instinct of competitive nature and social comparison. Like points a player seeing their name rise up a leaderboard encourages them to play more often and increases motivation.

Research was also done one a case study that indicated gamified approaches can lead to better learning outcomes compared to traditional methods, particularly in music education, where incorporating game elements can foster a state of immersion and enhance motivation. The study was conducted at Universidade Politécnica de Valência in which three groups of students in the second cycle of basic music education were tested. These students focused on guitar, recorder, or singing lessons. The groups included a control group using traditional teaching methods, a group utilizing multimedia materials in the classroom, and a group employing a game-based approach on the Moodle platform. The control group experienced conventional music instruction with a teacher and sheet music, while the multimedia group accessed videos and interactive content under educator supervision. The findings revealed that the game-based approach achieved a 100% success rate in summative evaluations, compared to 84% for the multimedia group and 64% for the control group. These results suggest that gamified learning experiences can enhance student motivation and overall understanding of music education content. Research concluded that integrating gamification into music education can make learning more enjoyable and effective, encouraging students to engage with instruments and music theory.

Technical Research regarded research into similar applications that already exist. How unity and other games handled pitch analysis and the Fast Fourier Transform (FFT) and SRH algorithms.

Preliminary research was done on similar applications and project using pitch analysis in unity. When the idea for 'Recorder Academy' was being refined it needed to be established how pitch analysis worked in the Unity game engine. Namely a project done by Alexandre Bruffa (2022) that used real-time audio analysis in Unity using the FFT algorithm to effect video shaders. While this project was not wholly useful regarding the project it gave a fundamental understanding of pitch analysis through unity and show that it is possible.

The FFT algorithm is used in unity to take the sound data from the devices in built or external microphone and breaking it down to its specific frequencies. This is done by firstly storing the data collected over a time. This stored data is defined as a block under two parameters a sampling rate (fs) which is the number of samples recorded per second and the block length (BL) is the number of frequency bins in a block that gets sent through the algorithm. With the two initial parameters we can get the bandwidth (fn) which is the maximum frequency that the FFT can detect. For instance, if fs is set to 48kHz (48000Hz) then fn = fs/2, fn=24kHz. Measurement duration (D) is the time collected to collect one block. In the case of fs=48kHz (48000Hz) and BL=1024 D=bl/fs D=21.33ms. Finally for preliminary parameters is frequency resolution (df) which is the spacing between each FFT block. Df is calculated with fs/bl, using the same values as previous 48kHz (48000Hz) / 1024 which mean df = 46.88Hz. So, in this example the sounds are played and ever 21.33ms we are getting 1024 frequency bins and the spacing between each bin is 46.88Hz. FFT is applied to each bin and the frequencies are extracted from each bin.

These first initial frequencies are then passed through a summation of residual harmonies algorithm to extract the fundamental pitch from the gathered frequencies. SRH takes the initial frequency as f0. In the spectrum data it extracts multiples of f0. For instance, in the case of initial frequency of 100hz the multiples f0(100),1f0(200),2f0(300) … and so on until fk which is the final multiple that it will check. This fk value is determined by the FFTs block length. So, if a FFT block length of 1024 SRH will check 1024f0 will be analysed. For each f0 candidate that is a direct multiple it is given more weight and is considered correct. For each candidate that is not a multiple or has some variance is downweighed. The summation of the respective weights determines the final note value.

3 Requirements

3.1 Introduction

The functional requirements of 'Recorder Academy' are the requirements and features needed for the game to function as intended. This includes user interface to allow the player to start the game and the main gameplay features as well.

Requirements are broken down into two sections

- Functional Requirements: These are the requirements that are needed for the game to work and run as expected
- Non-Functional Requirements: These are desirable requirements that make the game enjoyable and include usability and design

3.2 Requirements gathering

Requirements gathering is the process of identifying and documenting the needs and expectations of players for a system or project. It involves various techniques such as, surveys, prototyping, and competitor analysis. Requirement gathering is important as it gives the developers an understanding of the product needs and ensures the product meets user expectations. Requirement gathering also confirms if the product is feasible within technical constraints and aligns with project goals. Effective requirements gathering helps minimize misunderstandings and create a clear roadmap for development.

3.2.1 Similar applications

Research into the following applications was done. This research investigated each application as they use similar features whether that be gameplay or for use in music and pitch analysis.

- Yousician
- Guitar Hero

‘Yousician’ is a music education mobile application. It uses the same principles as ‘Recorder Academy’ using real time pitch analysis to listen the player playing their instrument of choice.

Figure 1 Yousician gameplay screen

Yousician allows players to practice either guitar, bass, ukulele or singing in real time to popular songs. Using pitch analysis and audio recognition to provide instant feedback to the player on rhythm and notes allows for a clear image on where the player needs to improve.

Yousician was used as heavy inspiration for ‘Recorder Academy’ with the main gameplay mechanics being very similar to the goals of the project. The pitch analysis works very similarly to ‘Recorder Academy’ as well as the gamified learning aspect.

However, Yousician is very limited when it comes to free play. To get the full experience for Yousician it requires a paid subscription to access the lessons and popular songs. This approach makes it inaccessible to some players with a more advanced base knowledge of their chose instrument or music theory.

The functional requirements gathered from Yousician were the pitch analysis aspect as well as player feedback and a point system

Guitar Hero a console developed by Harmonix was the second game ‘Recorder Academy’ has requirements modelled after.

Guitar Hero has players playing a guitar controller to popular songs. Disk representing notes move towards the player on a scrolling highway, once these notes reach the end of the screen the player presses the correctly coloured button to gain points.

Figure 2 Guitar Hero gameplay screen

Figure 3 Guitar shaped controller used for Guitar Hero

Guitar Hero has a fun mode of play with fast-paced gameplay making the game addictive and engaging. The scrolling highway towards the player feels more immersive and encourages players to play again and again. This technique of scrolling notes was more effective than Yousician's use of side scrolling as that mimics the musical stave, Guitar Hero's method is more gamified.

In contrast to Yousician however Guitar Hero offers no music education aspect, using a fake guitar controller rather than a real guitar.

From Guitar Hero the functional requirement gathered is the generating notes that move down the screen towards the player.

3.3 Requirements modelling

3.3.1 Personas

Personas are used in requirements modelling to put the users' needs first. They are fictional characters that would be potential users of the software. This helps developers put themselves in the shoes of the user and think about the software from their point of view

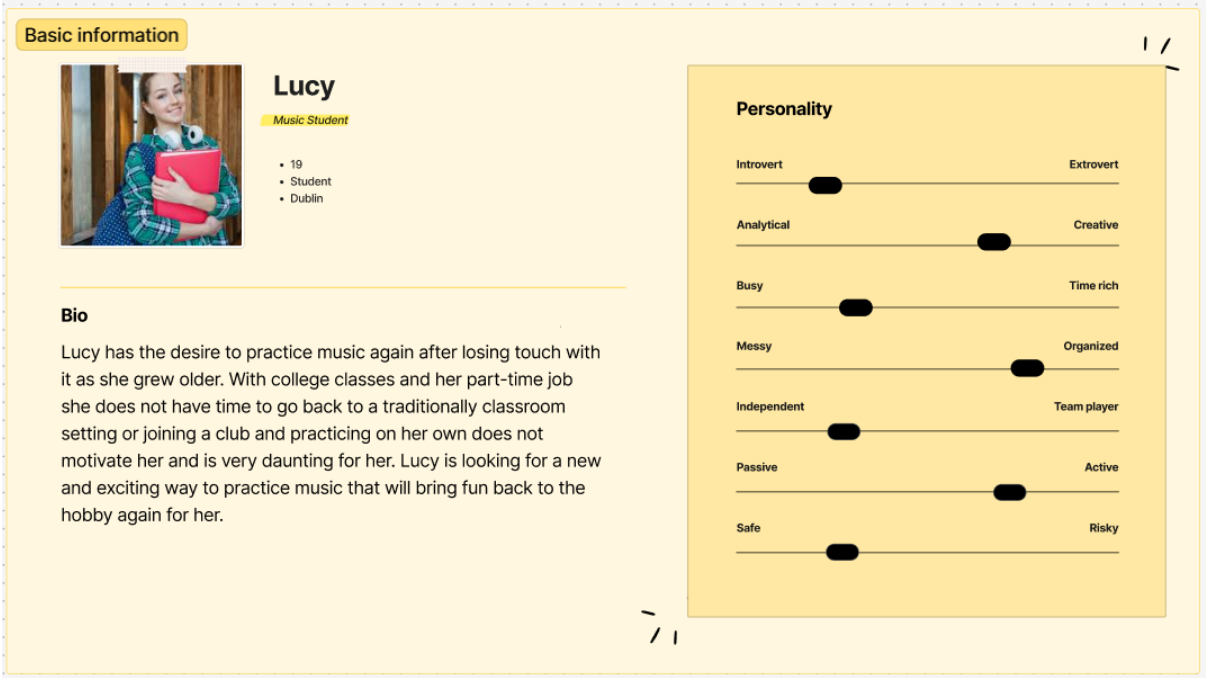


Figure 4 Persona 1 potential user of 'Recorder Academy'

Pictured above is Lucy a potential user of 'Recorder Academy'. Lucy is modelled with user needs in mind and is fictional character searching for an app similar to 'Recorder Academy'.

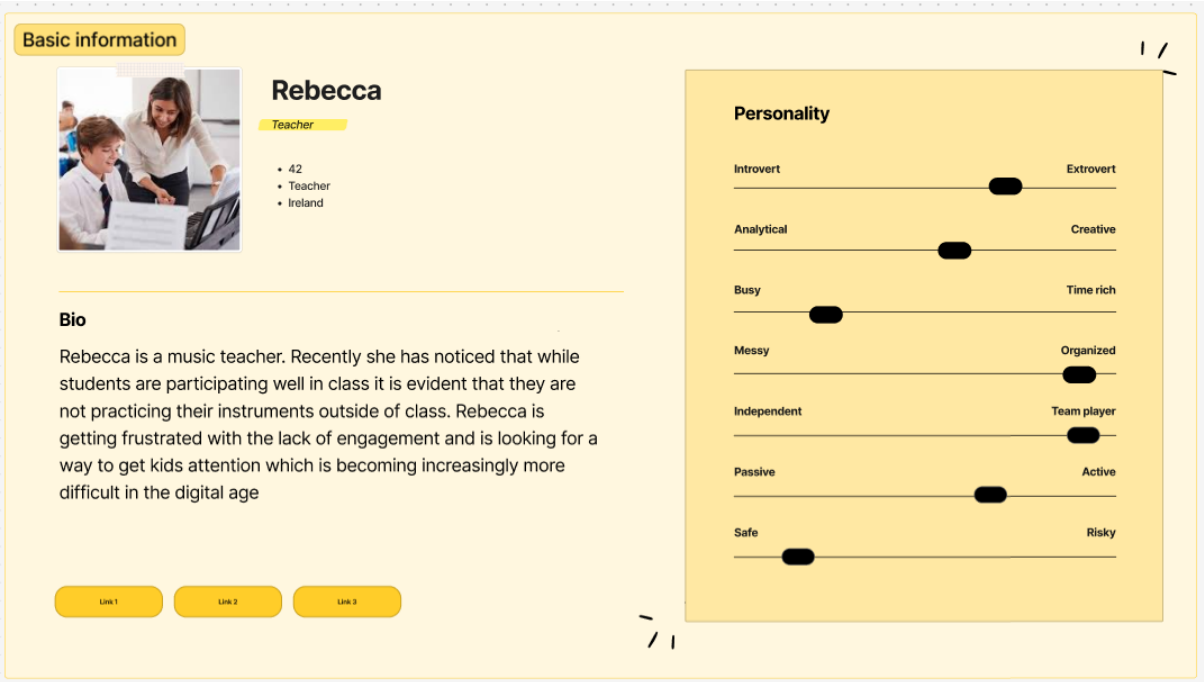


Figure 5 Persona 2 a character who would benefit from 'Recorder Academy' without necessarily using the application

Rebecca is modelled after a music teacher. Music teachers could potentially benefit from an application like 'Recorder Academy' as it would benefit student engagement to music outside of a classroom setting.

3.3.2 Functional requirements

Listed below is a list of the functional requirements for ‘Recorder Academy’

1. **Mic Pickup -** Unity's microphone class access the microphone on the device and records the sound input
2. **Note Recognition –** The sound recorded from the on-board device microphone is analysed and the note frequency is extracted and converted to a letter value. (i.e. if a 261.63 Hz frequency is detected it is converted a C letter value)
3. **Target Zone –** the target zone will trigger the script that handles the note recognition once an object enters the area
4. **Moving target objects -** Blocks with the note value that the user must play (ex. A) printed on it moves down the screen towards to the player.
5. **Point System -** A point system will be implemented to give players a point whenever the play the right note at the correct time
6. **Game Start + GUI -** The game needs basic GUI screens including a home screen which will allow the user to start the game.

3.3.3 Non-functional requirements

Listed below is a list of the Non-functional requirements for ‘Recorder Academy’

1. **Appealing and consistent Design –** The game should have appealing visuals to attract a player. Fonts and colours should be consistent to make the game look polished.
2. **Game should run smoothly –** The game should run smoothly with little to no errors effecting play
3. **Accessibility –** The game should be accessible to as many players as possible.
4. **Win/Lose condition –** For the game to feel complete there should be win and lose conditions. In the case of ‘Recorder Academy’ after the player misses three notes they lose.

3.3.4 Use Case Diagrams

Figure 6 shows a use case diagram for ‘Recorder Academy’ use case diagrams illustrate the interaction between the user and the application in a step by step method.

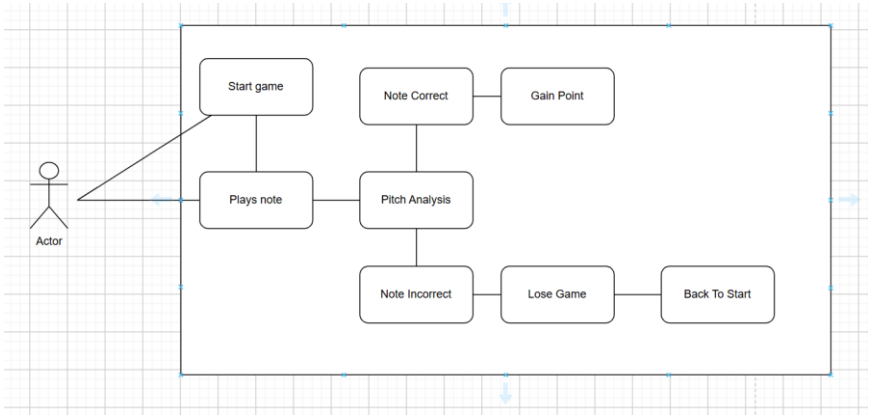


Figure 6 Use Case Diagram for 'Recorder Academy'

3.4 Feasibility

Table 1 displays the feasibility and problems that may occur during the development of this project

Challenge	Description	Solution
Limited unity experience	The limited knowledge with the unity game engine may be a challenge when it comes to aspects of the development	Help from online resources and my supervisor will be utilized along with tutorials.
Understanding pitch analysis	No prior understanding of pitch analysis through unity. Completely new section of learning	Using online resources and tutorials and example of similar projects can help with the basic understanding of how pitch analysis works through unity
Time Constraints	Keeping on top of such strict timelines does not leave much room for code breaking errors and design work	Paying close attention to the project schedule and working everyday

3.5 Conclusion

In the above chapter has demonstrated how the requirement for 'Recorder Academy' were gathered. The goal with this project to create an engaging and entertaining way to practice the recorder. This is done using real-time pitch analysis and gamification techniques. Looking to Yousician and Guitar Hero 'Recorder Academy' integrates pitch detection, scrolling notes and a point system. These technologies are designed to create a fun and engaging way to practice music.

Requirement gathering was conducted by looking to apps that perform a similar task and analysing the functional requirements of those games. Personas were also created to aid gathering by making sure the players needs come first during the development process. Functional requirements listed above are microphone input, note recognition, a target zone, moving note objects, and a point system whilst the non-functional requirements are smooth performance, accessibility, appealing design, and clear win/loss conditions.

Finally in this chapter is discussed the feasibility of the project and the potential problems that could become an issue as development progresses. The problems discussed were limited unity experience, no knowledge of pitch analysis and time constraints with the solutions to these being using online resources, looking to tutorials and existing project and a strict schedule.

4 Design

4.1 Introduction

This chapter discusses the program and user interface (UI) design of the application ‘Recorder Academy'. The program design will discuss the program architecture and the relationship between individual scripts to make the project function as desired and meet the functional requirements discussed in the chapter previous. This includes the pitch analysis technology, how it is being stored and used for scoring. All development was done within the Unity game engine using the C# language.

Covered in the UI design will be discussed why certain design choices were made, iteration of the design overtime and how certain design choices help the user experience. Game assets were provided by the unity asset store. UI implementation within Unity was done by the UI canvas game element as well as TextMesh Pro a set of unity tools that give greater control over using text in unity.

4.2 Program Design

Program design discusses what makes a game functional. This includes the structure of files and C# Scripts and how they relate to each other. In the context of ‘Recorder Academy’ this includes discussing how the real-time audio processing, accurate input detection, and responsive feedback systems function.

For ‘Recorder Academy’ an entity-component- system (ECS) was used for development. An ECS allows components and systems to be developed separately and then connected. Unity helps this approach by allowing individual C# scripts to be applied to individual game objects in the unity inspector. Components were able to be designed and positioned after the main functionality was functional.

4.2.1 Technologies

As discussed, prior the Unity game engine was the main technology used in this project. Unity is a robust 3D or 2D game engine. Unity was first introduced in 2005 and has developed a name for itself as a popular game engine for beginners and experts alike as it is useful in development for most game types including mobile, Personal computer and augmented reality and virtual reality games. Being around since 2005 it has grown a large online community of forums and online tutorials. This vast presence of online resources was one of the primary reasons Unity was chosen for ‘Recorder

Academy's' development. These online resources were helpful in the case of bugs or code logic that needed to be rewritten.

Unreal Engine a similar game engine could have been used for development but was decided against as it is primarily focused on high-end gaming and programming with C++ rather than C#. Unity was overall more light weight than Unreal and proved easier to manage during development.

4.2.2 Structure of Unity Program

In order of appearance to the user the following is a description of the file and scripting structure of 'Recorder Academy'

4.2.2.1 Home Screen and Game Start

When 'Recorder Academy' is first started the user is greeted with the Home Screen as seen in Figure 7. This home screen is created using the unity Canvas Object and TextMesh Pro button Objects.

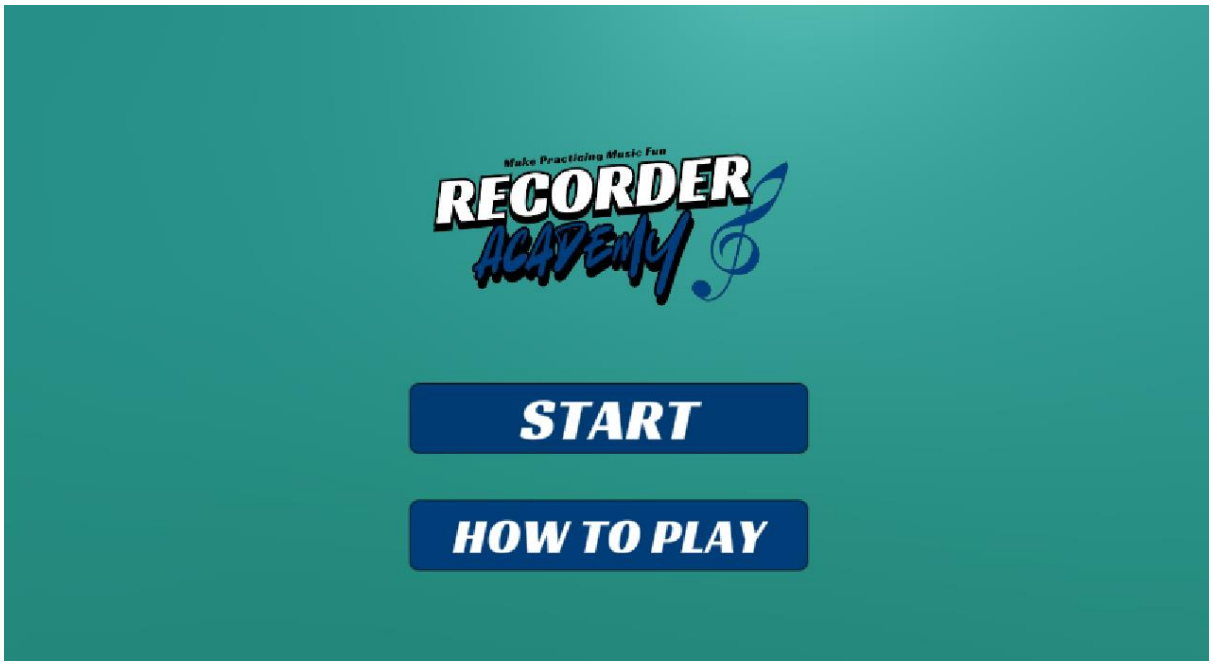


Figure 7 'Recorder Academy' Home Screen

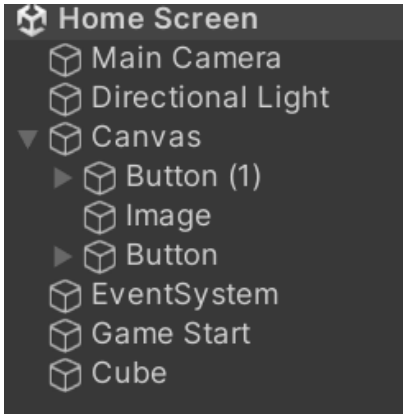


Figure 8 Home Screen scene element breakdown

From this screen Players Select to view the How to Play screen, A scene with similar element breakdown that displays the rules and instructions of play.

Figure 9 Instruction page of 'Recorder Academy'

Navigating between the two pages is done using the two buttons on the home page. Thes buttons are TextMesh Pro button game objects and are in built into Unity. Figure 10 shows the C# script that

handles this interaction. Using the LoadSceneAsync method in the unity scene manager loads the desired scene index as seen in the unity build panel (Figure 11).

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

0 references
public class mainMenu : MonoBehaviour
{
    0 references
    public void StartGame()
    {
        SceneManager.LoadSceneAsync(1);
    }

    0 references
    public void InstructionScreen()
    {
        SceneManager.LoadSceneAsync(2);
    }
}
```

Figure 10 Button functionality on home screen script

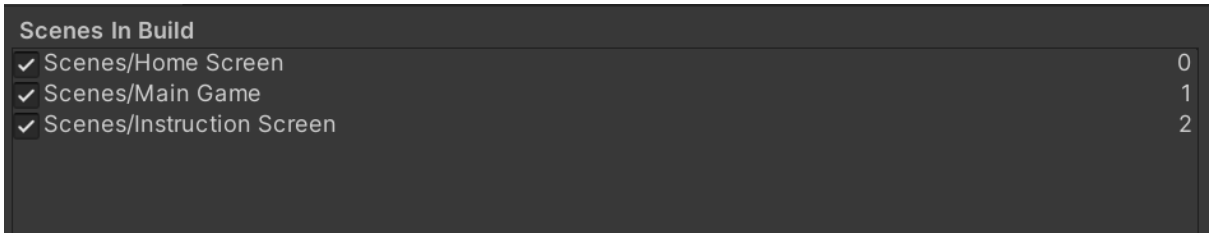


Figure 11 Build Settings

The script pictured in Figure 10 is attached to the button component and runs the Start Game function.

After the start game function is ran, users are redirected to the main game scene, with a 3 second countdown showing before the main game functionality starts. (Figure 12)

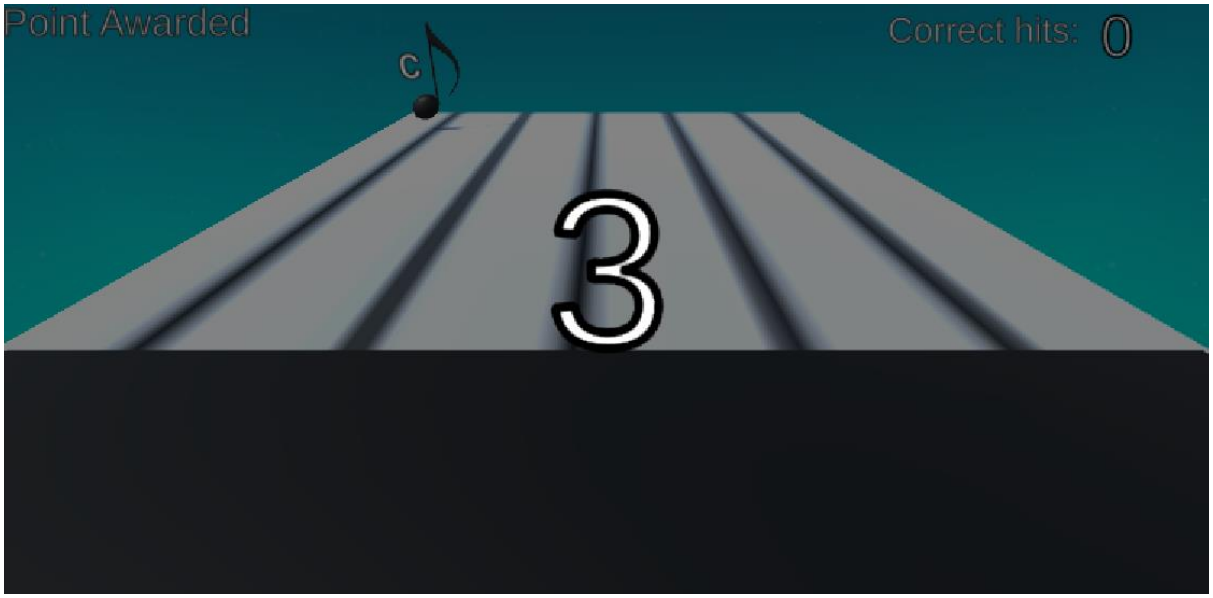


Figure 12 Main Game screen on first view

This three second countdown gives players a chance to prepare before gameplay begins. The timer is created using an updating TextMesh Pro component. Remaining time is set in the unity inspector as it is a public variable (Figure 14). For each second, remaining time is decrementing using the Time.deltaTime method. Once Remaining time hits zero it is set to 0 (otherwise it would keep running into negative time) and the startScreen overlay gets disabled. Once the start screen object gets disabled gameplay begins.

```
0 references
public class Timer : MonoBehaviour
{
    1 reference
    public TMP_Text timerText;
    5 references
    public float remainingTime;
    1 reference
    public GameObject startScreen;
    0 references
    void Update()
    {
        if (remainingTime > 0)
        {
            remainingTime -= Time.deltaTime;
        }
        else if (remainingTime < 0)
        {
            remainingTime = 0;
            startScreen.SetActive(false);

        }
        int seconds = Mathf.FloorToInt(remainingTime % 60);
        timerText.text = seconds.ToString();

    }
}
```

Figure 13 Timer Script

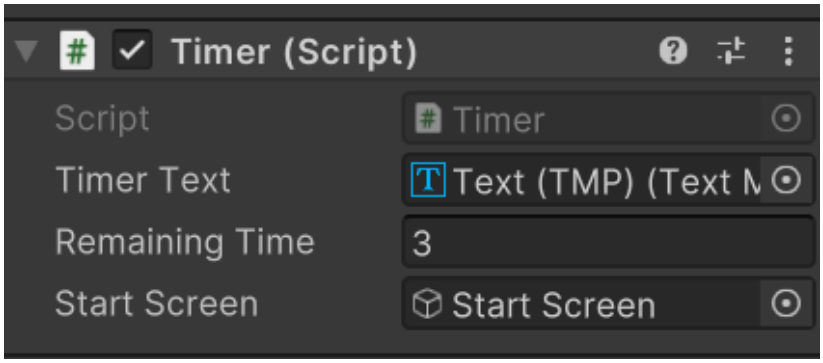


Figure 14 Where remaining time is set in the unity inspector

4.2.2.2 Note Component

The note component used in 'Recorder Academy' is the eight note prefab from the Notes package created by SaladMix Studio from the Unity Asset store. (Figure 15) These note objects move down the screen towards the player using the script shown in Figure 16.

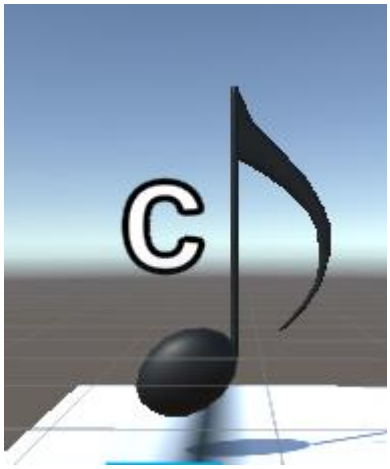


Figure 15 Note game object

```
0 references
void Start()
{
    randomMultiplier = Random.Range(1f, 5f);

    if (!delayCompleted) // Only wait if the delay hasn't been completed
    {
        StartCoroutine(WaitBeforeStart());
    }
    else
    {
        canMove = true; // Allow movement immediately for subsequent instances
    }
}

1 reference
IEnumerator WaitBeforeStart()
{
    yield return new WaitForSeconds(4); // Wait for 3 seconds
    delayCompleted = true; // Mark the delay as completed
    canMove = true; // Allow movement
}

0 references
void Update()
{
    if (canMove) // Only move if the flag is true
    {
        transform.Translate(Vector3.forward * baseSpeed * randomMultiplier * Time.deltaTime);
    }
}
```

Figure 16 Script handling note movement

The above script handles the notes movement down the screen. As mentioned previously the game does not start until the 3 second count down is complete. This means the first note on the screen pictured in Figure 15 doesn't start moving until the countdown has ended. To achieve this desired result the canMove Boolean value is added so that the transform.translate function does not happen until the countdown is over.

That is how the first note is handled. Notes after this are instantiated once the current note in play passes a trigger wall. The note passes through the wall, which is pictured in colour in Figure 17 for the sake of this document but during gameplay is completely invisible and triggers the script pictured in Figure 18.

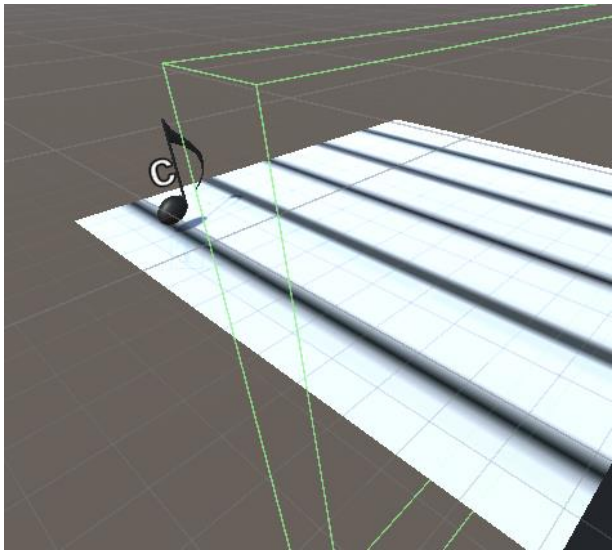


Figure 17 New note trigger wall. Outlined in green

```
0 references
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.layer == LayerMask.NameToLayer("BlockSpawnLayer")) // Only react to this layer
    {
        // Find the target cube by tag
        GameObject targetCube = GameObject.FindGameObjectWithTag(TargetCubeTag);
        if (targetCube != null)
        {
            // Instantiate a new block and store reference
            GameObject newCube = Instantiate(targetCube);

            if (!newCube.GetComponent<CubeStatus>())
            {
                newCube.AddComponent<CubeStatus>();
            }

            // Modify only the new block's TextMeshPro component
            TextMeshPro textComponent = newCube.GetComponentInChildren<TextMeshPro>();
            if (textComponent != null)
            {
                string randomName = _names[Random.Range(0, _names.Length)];
                textComponent.text = randomName;

               float position;
                if (_namePositionMap.TryGetValue(randomName, out position))
                {
                    Vector3 spawnPosition = new Vector3(-13.22f, -0.98f, position);
                    newCube.transform.position = spawnPosition;
                    newCube.transform.rotation = Quaternion.Euler(0f, 90f, 0f);
                }
            }
        }
```

Figure 18 Script handling block instantiation

Breaking down this section by section, once the note passes through the trigger wall it checks to see if the object that has entered the trigger has the target cube tag. Then setting that game object to a new cube and instantiating it. On this new cube adding the cube status script (Figure 19). The reason for this will be discussed as part of the point system section.

```
0 references
public class CubeStatus : MonoBehaviour
{
    0 references
    public bool pointsAwarded = false;
}
```

Figure 19 Cube status script

The note component has a text mesh pro component attached to it. This is unity's in-built tools for handling text. The code block is getting access to the text mesh pro component and changing its value from a names array (Figure 20). This letter value is the note the player is meant to play.

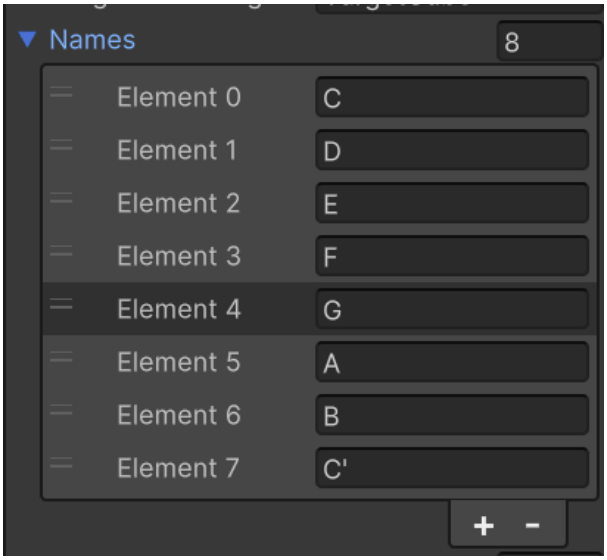


Figure 20 Names array

This names array is also added to a dictionary (Figure 21) that maps each note to a specific position this ensures that all notes with the same text value instantiates in the same position. This is done by mapping each letter value to its respective position. Using a for loop and adding that to the dictionary. In this instance C being index 0 in the names array is mapped to position 0 in the positions array (Figure 22). To get the final position in which the note is to instantiate the script reads the dictionary for the name and returns the equalling the position. This is then set as the position variable in which the note is instantiated.

```
3 references
private Dictionary<string, float> _namePositionMap;

0 references
private void Start()
{
    _namePositionMap = new Dictionary<string, float>();
    for (int i = 0; i < _names.Length; i++)
    {
        _namePositionMap.Add(_names[i], _positions[i]);
    }
}
```

Figure 21 name position map

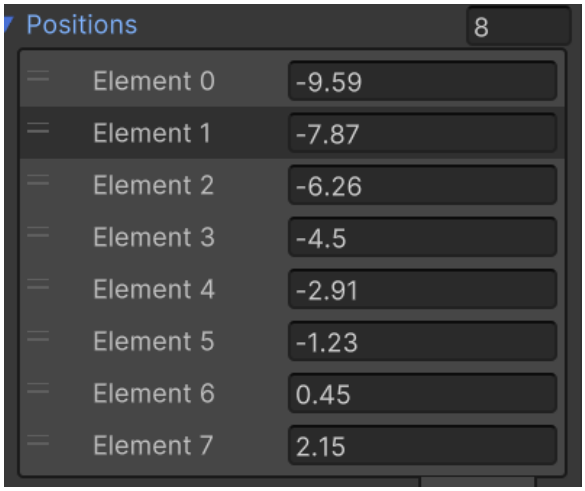


Figure 22 positions array

4.2.2.3 Pitch Estimation

We discussed in the previous section the concept of triggers and the on trigger enter method in unity. For the pitch estimation to function we use another trigger. Pictured in Figure 23 is the area in which the button input script runs. Similar to the instantiation script this checks to see if the note has the target cube script first and then runs logic after making this check.

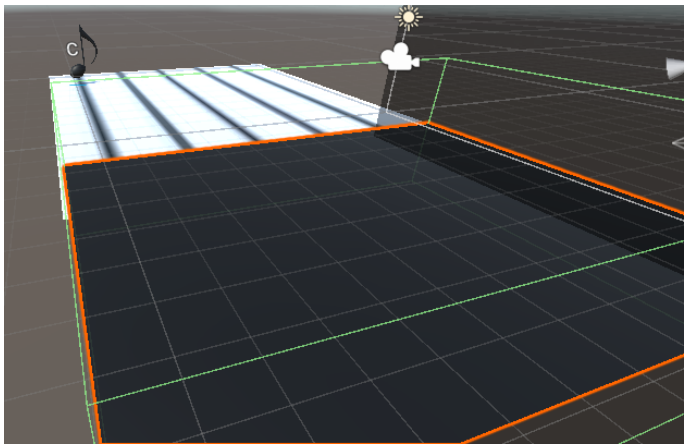


Figure 23 Target area with trigger attached outlined in green and orange

The core pitch estimation script is handled by audio pitch estimator (nakakq, 2020). This script takes the audio input and converts it into its frequency value. Then this frequency value is converted to a letter using the get name from frequency function. How this runs in game is once the note triggers the collider the button input script invokes the start recording repeatedly as the gameobject is within the trigger area.

```
0 references
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("TargetCube"))
    {
        Debug.Log("TargetCube entered trigger zone, starting recording.");
        currentTargetCube = other.gameObject;
        InvokeRepeating("StartRecording", 0f, 0.5f);
```

Figure 24 on trigger enter invoking start recording

Start recording is what handles the collection of sound data and conversion. It collects the sound data and processes it through the estimate function in the audio pitch estimator script. The frequency estimated is passed through the get name from frequency function that converts the frequency to its appropriate letter value. For instance, if the estimate function detects the frequency as 261.63 Hz (with a bit of variance to account for play accuracy and instrument variance) the get name from frequency converts this into C.

The estimate function works as follows takes the frequency input and analyses the frequencies. Then smooth the frequencies to extract the most dominant frequency present. This most dominant frequency is applied to a Summation of Residual Harmonic method to find the fundamental frequency played. This fundamental frequency is what is determined to be the frequency that is being played. Additionally, if there is no frequency detected it is returned to be null. This is why the script runs repeatedly the entire time the note component is in the target area to allow for the player to try play the correct note more than once.

```
// Get the detected frequency and note
float frequency = pitchEstimator.Estimate(audioSource);
if (float.IsNaN(frequency))
{
    Debug.LogError("Frequency is not a number Skipping.");
    return;
}

string detectedNote = GetNameFromFrequency(frequency).Trim().ToLower();
Debug.Log("Detected Note: " + detectedNote);
```

Figure 25 audio recorder script getting frequency and converting to name

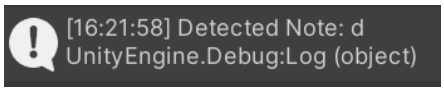


Figure 26 debug log output when D is played

4.2.2.4 TextMesh Pro

As mentioned previously the note game object has a TextMesh pro component attached to it that generates a new letter from the array as the new note is instantiated. Within the audio recorder script, the text attached to the letter is read and is saved as the required note variable. This is the note that the player needs to play to get a point

```
// Get the TMP_Text component from the targetCube
TMP_Text textMeshPro = targetCube.GetComponentInChildren<TMP_Text>();
if (textMeshPro == null)
{
    Debug.LogError("No TMP_Text found on TargetCube!");
    return;
}

string requiredNote = textMeshPro.text.Trim().ToLower();
Debug.Log("Required Note from TargetCube: " + requiredNote);
```

Figure 27 Reading text on TextMesh Pro component

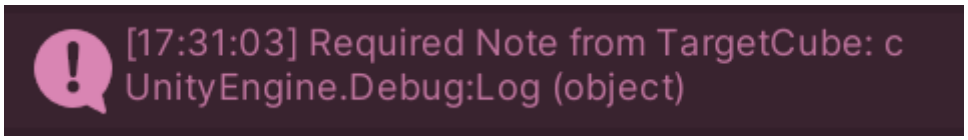


Figure 28 Debug log showing the required note

4.2.2.5 Point System

The two components listed above is what gives the players points. Once the sound inputs frequency is gathered and the frequency is converted to its letter value. It is compared to the letter value of TextMesh Pro component. If these two values are the same the player is awarded a point.

```
if (detectedNote == requiredNote)
{
    Debug.Log("Correct note played!");
    pointSystem.AddPoint();
    if (!targetCube.GetComponent<CubeStatus>())
    {
        targetCube.AddComponent<CubeStatus>().pointsAwarded = true;
    }
    else
    {
        targetCube.GetComponent<CubeStatus>().pointsAwarded = true;
    }
```

Figure 29 comparing required note to detecting note to give a point

As seen in figure 29 if the two letter values are correct a point is added and the points awarded boolean in the cube status script (Figure 19) is changed to true. This is added to prevent a player being able to receive multiple points from the one cube.

This also updates the user UI to give immediate feedback for when they have gained a point. A counter in the top right corner shows the number of total points the player has and text pops up saying point awarded also when a point is awarded

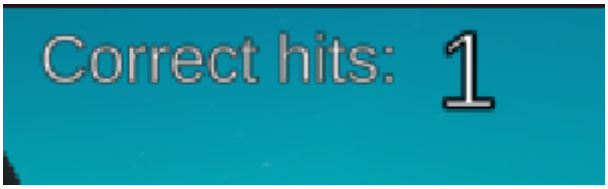


Figure 30 Total points counter

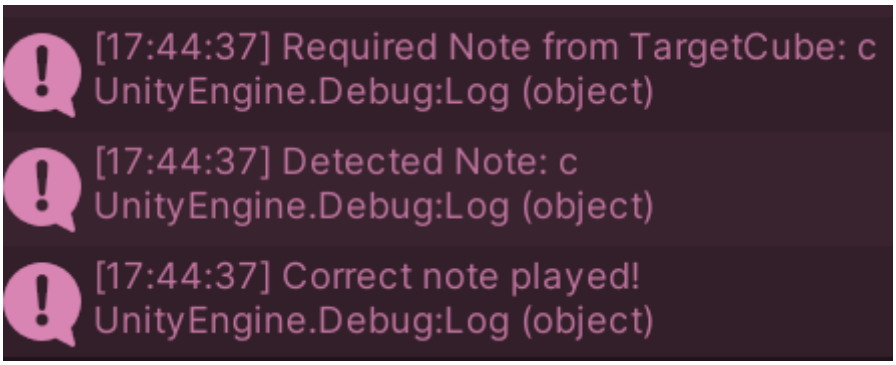


Figure 31 Debug log feedback when correct note is played

Figure 32 Point awarded text that appears as point is given

Additionally, when a point is given the target area in which the note is turns green. This provides additional feedback to the player so they can see at a glance what they played right.

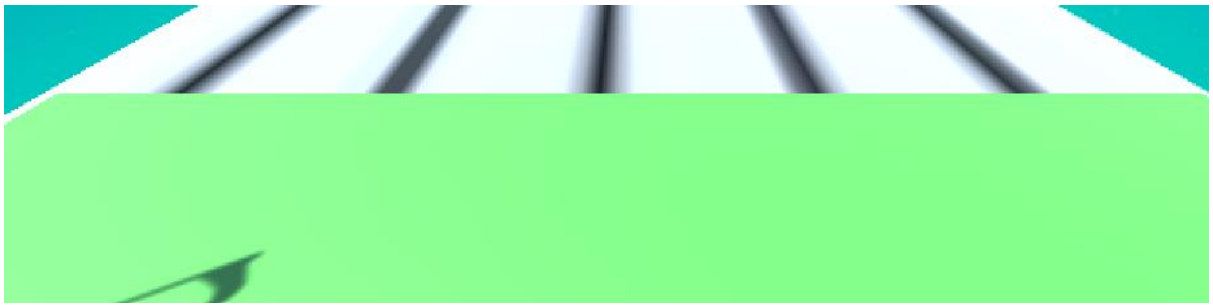


Figure 33 Target area turns to green when point is awarded

In contrast the target area turns to red when the incorrect note is played

Figure 34 Target are turn red when incorrect note is played

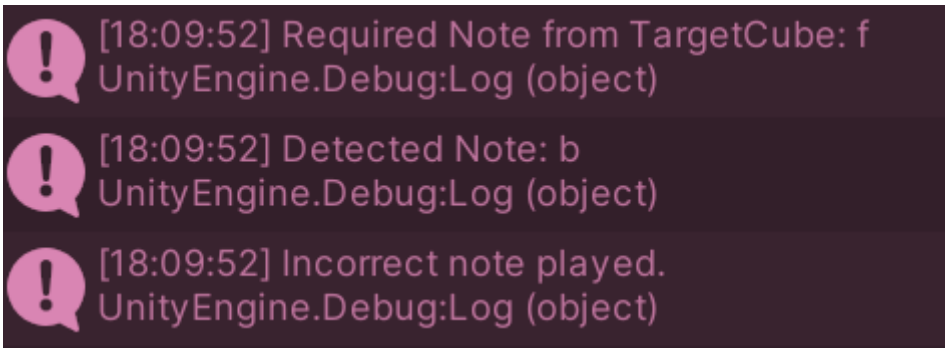


Figure 35 Debug log when incorrect note is played

4.2.2.6 Game End

When the player incorrectly plays 10 notes the game will end. The number of incorrect notes is stored in a variable and when this reaches above 10 the game over screen appears displaying the players total points and a button that will navigate them back to the homepage.

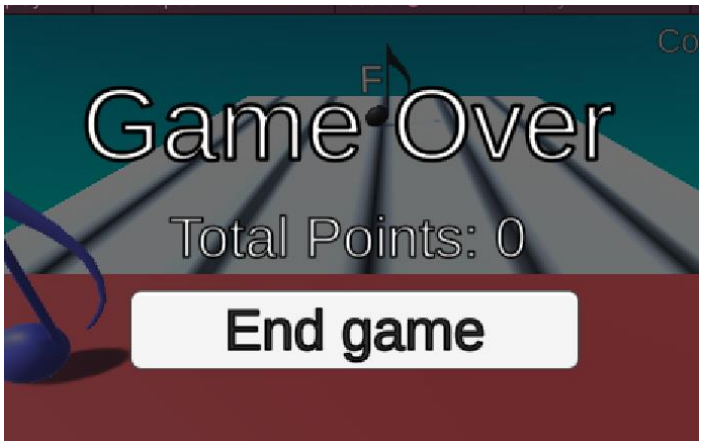
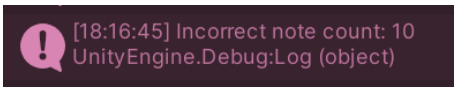


Figure 37 Game over screen

Figure 36 incorrect note count



4.2.3 Application architecture (1 page)

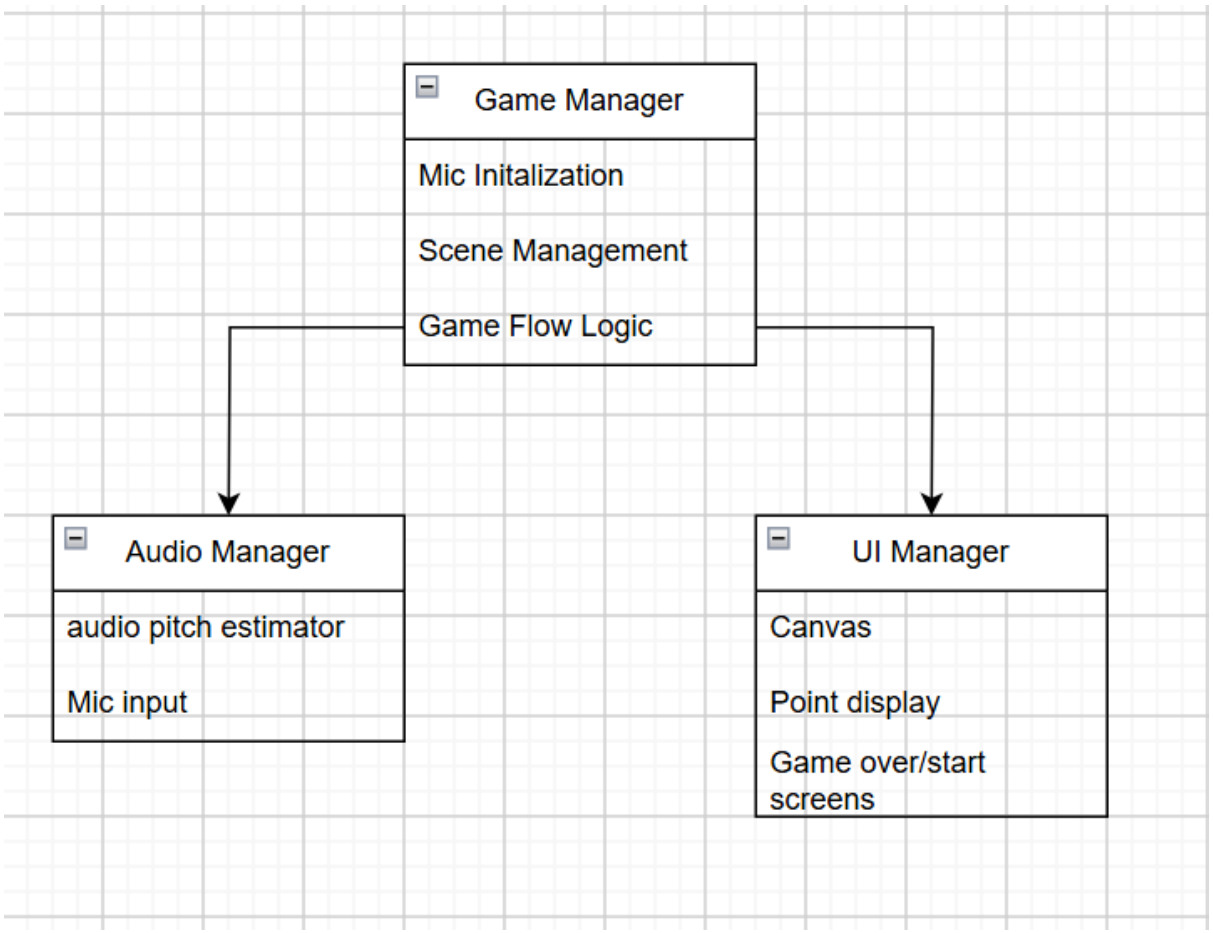


Figure 38 Block Diagram for 'Recorder Academy'

Figure 38 pictures the block diagram for Recorder Academy. It is a graphical representation of the workings behind the project. It starts with the main game manager that starts the game and handles the build make up and navigating between scenes. This also initializes the on-board microphone making sure it is ready for use. Game flow logic refers to the game movement and game mechanics namely the handling of components like the target area and backgrounds as well as the note components movement.

The audio manager is what takes in the mic input and passes it to the audio pitch estimator. This would include the block input and audio recorder scripts that handles the frequency converting from sound to the respective frequency and then its note value.

Finally, the UI manager would include the TextMesh Pro component on the note component as well as the UI screens to start the game and instruction screen also would include the point system and any feedback given to the player.

4.2.4 Process design

A games process design is the structure of how the game is developed and documented this will help meet project deadlines and make sure that all game requirements are met. In the development of 'Recorder Academy' flow charts and pseudocode were used to help visualize process design.

Figure 39 shows the pseudocode written during development. This was the first iteration of the core gameplay mechanic that compares the detected note frequency value to the required note letter value from the note component. This was the first stage of development of that section of code but helped the design process nonetheless as it gave a visual aid to what the code block needs to look like.

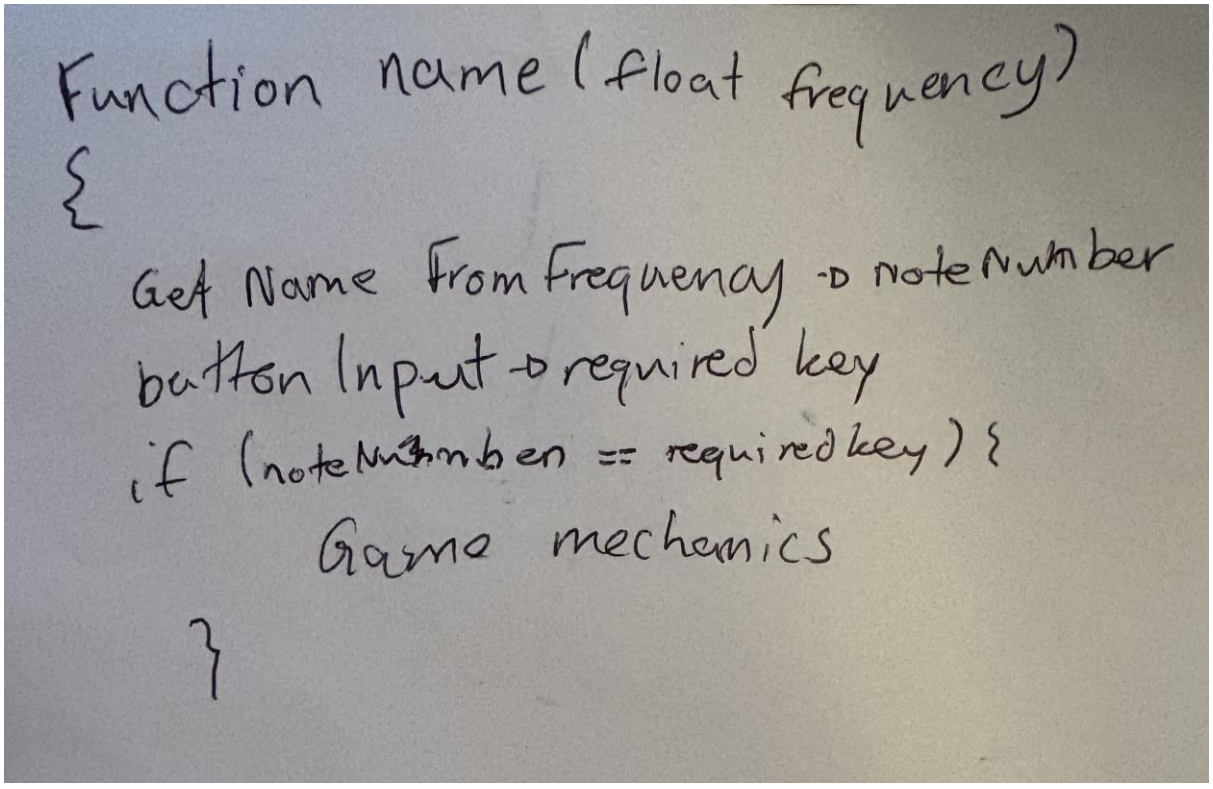


Figure 39 Pseudo code handwritten

4.3 User interface design

Good user interface design ensures that the target audience of 'Recorder Academy' in engaged with the game and is appealing to the viewer. In the case of this project good user interface design is needed to properly give immediate feedback to the player. A cluttered UI design leads to high mental load for the player so having a clean and concise user interface can improve user experience and lets users mental focus be on the gameplay.

‘Recorder Academy’s UI was handled using the unity’s in-built canvas object that allows for graphical user interfaces and interactable elements like buttons, sliders and text. The on-screen text was managed by the unity TextMesh Pro component which we have discussed previously.

4.3.1 Wireframe

A game wireframe shows the fundamental design of the game and fundamental movements without any game functionality. Wireframes for ‘Recorder Academy’ were made in Unity and Figma a user interface design application commonly used for web application prototyping and creating static web page mock ups.

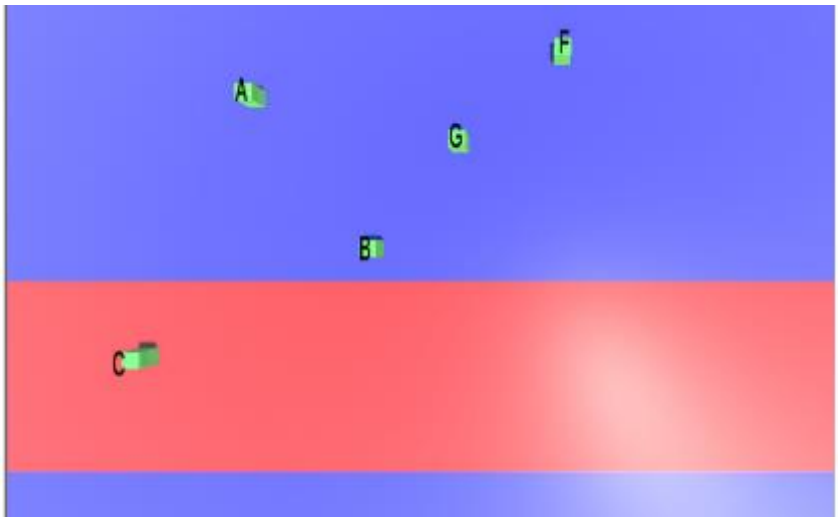


Figure 40 First preliminary wireframe of 'Recorder Academy'

Figure 40 shows the first wireframe for ‘Recorder Academy’ This was made to display game objects in movement towards a target area. The green cubes are in place of the note objects and move down the screen at random times and the red area is in the place of the target area.

The second iteration of the wireframe (Figure 41) was built on the same fundamental design principles as the first. The difference is a bit of added colour and angling gameplay so that the blocks appear to be moving towards the player to create a more immersive experience.

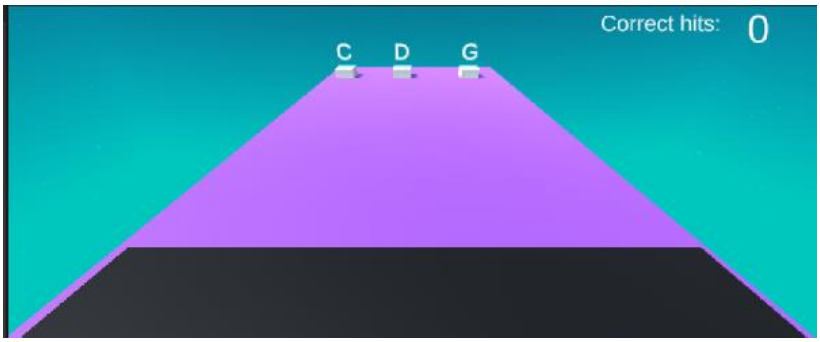


Figure 41 Second iteration wireframe

The previous two wireframes were created during the preliminary stages of development with no real sense of what the true design will look like in the end. Figure 43 shows a Figma wireframe displaying the first true design inspiration for the project. Figure 42 shows a similarly designed wireframe designed by the supervisor of this project Timm Jeschawitz.

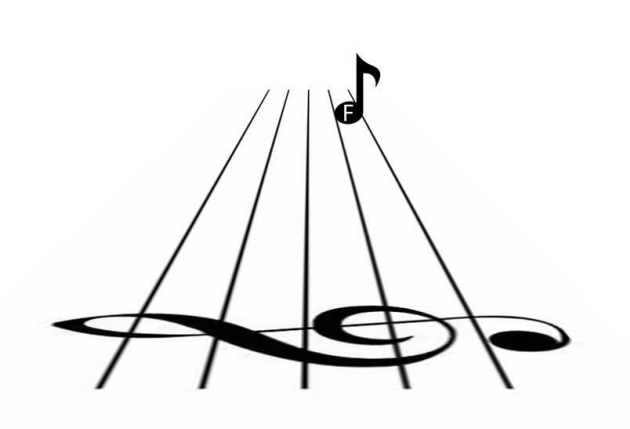


Figure 42 Wireframe 2

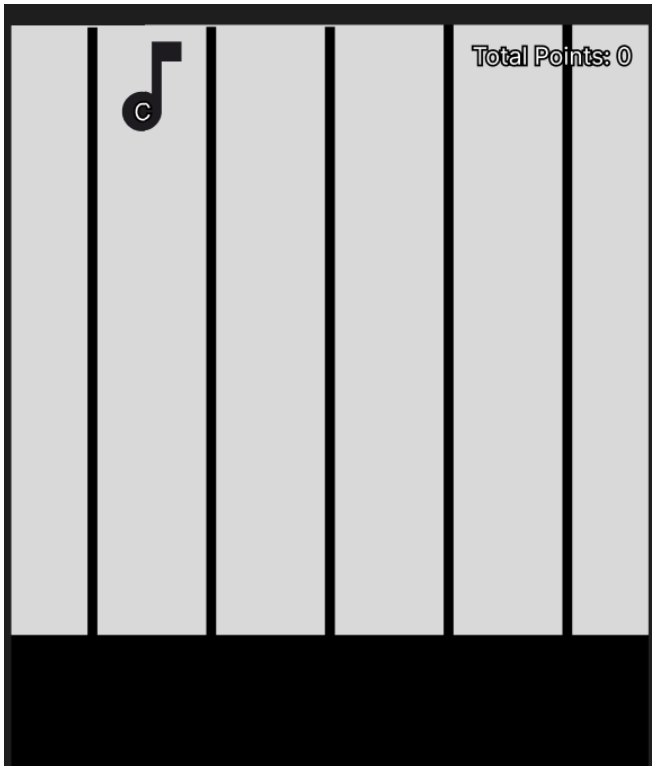
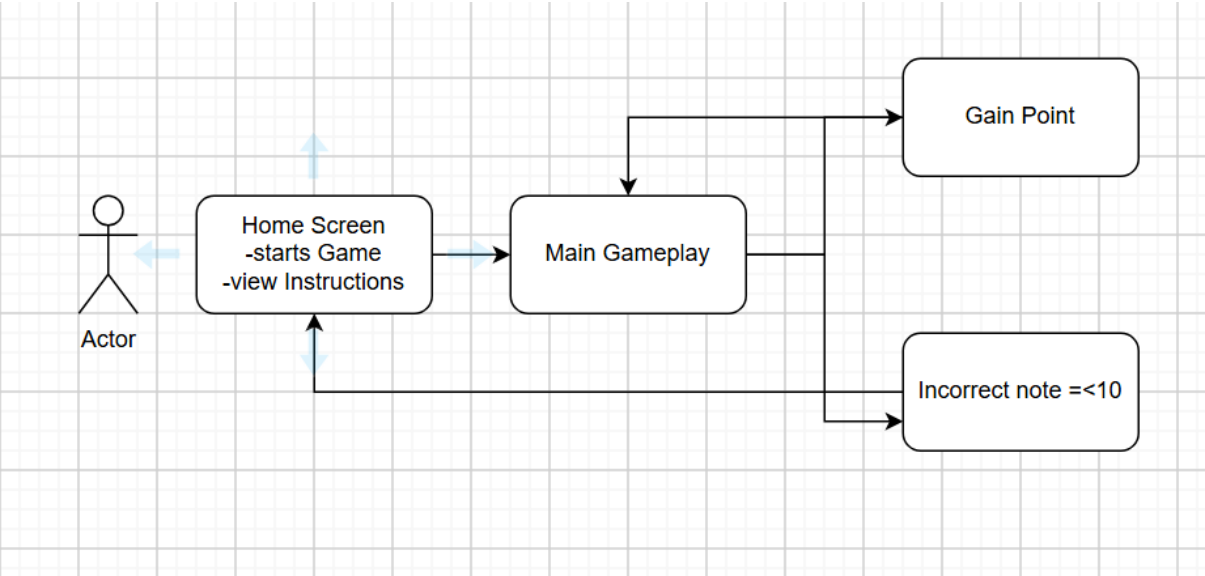


Figure 43 Figma prototype

The black lines on the white background are to mimic music stave commonly used to write sheet music.

4.3.2 User Flow Diagram

A user flow diagram is designed to illustrate the movement between scenes or pages a user can take in a game or web page. Figure 44 shows the user flow diagram of 'Recorder Academy'. The user starts on the home screen where they can choose to start the game or view the instruction page. If they choose to start the game, they are redirected to the main gameplay screen. They will stay in the main gameplay screen playing the game until they miss 10 notes in which case they will be redirected to the home screen. From there they can restart game play or exit the application.



4.3.3 Style guide

For this project design a colourful and inviting environment was chosen as this application is tailored to children and young people and that age range respond well to bright and bold colours. Blue being the primary used colour was to give a sense of calm and to improve cognitive performance (Hilliard, 2013).

A deep blue was chosen for use in the logo and the buttons seen on the home screen first and this colour was put into the adobe colour software to generate a colour palette.

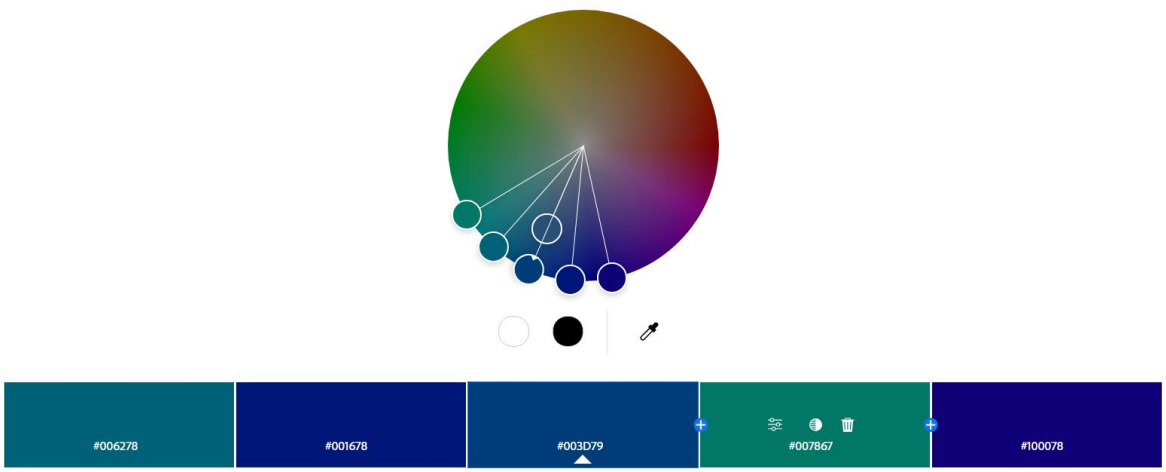


Figure 44 Colour Palette

An analogous colour scheme was chosen to generate a colour palette that will use the primary blue colour as an anchor and create shades of blue that will work well with it. For the background of the

home and instruction screen the teal shade was chosen to give good contrast and make the primary blue in the logo and button pop out.

For gameplay text was set to be white with a black outline to make them visible and easy to read at a glance. Similarly leaving the text white on the buttons keep them easy to read preventing potential confusion for the player.

4.3.4 Storyboard

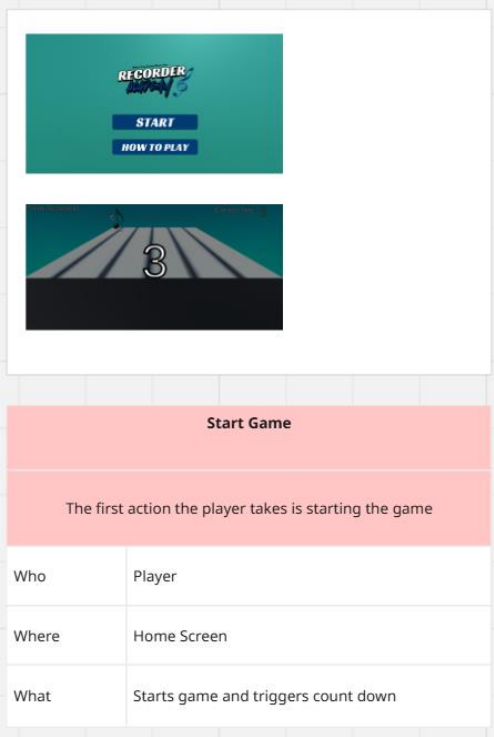
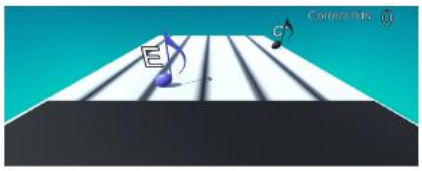


Figure 45 first scene of storyboard

Scene one of the story boards is player begins game. The game is booted up on the players hardware and they are greeted with the home screen. From the home screen they can start the game or view the instructions. Navigating to the instruction page also leads them back to the home page after they are done viewing that page. Once the player presses start, they are redirected to the main gameplay screen with a countdown overlay over top that counts down for three seconds.



Gameplay starts	
Main gameplay begins - Player gains point	
Who	Player
Where	Main gameplay screen
What	Gameplay begins player plays correct note to score points
What	Player gets visual feedback when point is earned

Scene 2 of the storyboard shows the gameplay screen. After the three second countdown the notes begin to move down the screen towards the player and once in the target area the player is promted to play the note visible on the note when the note turns blue. If the play the correct note they will earn a point and this repeats for each note they get correct.

Figure 46 Second scene of storyboard

Misses Point	
Main gameplay - Player misses point	
Who	Player
Where	Main gameplay screen
What	Player misses note and target area turns red

During the main gamplay if the player plays the wrong note or misses the timing, they do not gain a point. The target area turns red to notify the player that they have missed a note, and they can try again for the next one in the sequence.

Figure 47 Scene 3 of the storyboard

Game ends	
Main gameplay ends	
Who	Player
Where	Game over screen
What	Player misses 10 notes - Gameplay ends
What	Game over screen - Player redirects back to homepage

The last scene in the storyboard shows what happens when the player loses the game. The player loses the game once they have missed 10 notes in total. When this happens the game over screen overlays the game play and the moving objects stop moving. Players can then end the game and redirected back to the home screen and can start gameplay over again.

Figure 48 Last scene in storyboard

4.3.5 Environment

The environment in which 'Recorder Academy' is played resembles a musical stave to keep with the music theme. The environment also consists of a brightly coloured and vibrant background to engage players and text on the screen to show immediate feedback. There is no direct setting for the game as it is just the objects moving on the colourful background

4.4 Conclusion

The above chapter discussed the system and UI design for 'Recorder Academy'. The program designed covered the system architecture, file structures and relationships, real-time pitch analysis and how the system was programmed to meet all necessary functional requirements. Unity and C# were chosen for development due to their flexibility, lightweight build and extensive online resources.

The UI section discussed why certain design choices were made and how they led to and improved the overall design of 'Recorder Academy'. Leading to a pleasant user experience and helping improve player engagement and accessibility. These design choices were made using iterative design wireframes using Unity and Figma as well as user-flow diagrams and style guides. The design choices

were finally implemented using Unity's in-built canvas object as well the TextMesh Pro library to allow for wider text customization.

Lastly, the environment and immediate user feedback system was discussed and how the implementation of these made for a more immersive experience for the player. The target area turning either red or green whether the play gains a point or not provides a clear visual indicator and can help show the player in what areas they need to improve upon.

These structures working together lead to a fun and effective user experience that meet the minimum requirements that are needed to make 'Recorder Academy' function as intended and provide an immersive and engaging interactive experience for the player, leading them to play the game over and over again.

5 Implementation

5.1 Introduction

In this chapter will be discussed the implementation of 'Recorder Academy'. The project was implemented using the Unity game engine as discussed prior. The Unity game engine is a 3D or 2D development platform that was first introduced in 2005 at the apple worldwide development conference. Unity uses the C# programming language for scripting and has online asset store in which creators can upload game assets for others to use.

Within 'Recorder Academy' is using a pitch estimation script written by nakakq another unity user that has uploaded their script to git hub for other to use. This pitch estimation script uses the in-built microphone on the device and listen to the audio input given. It extracts the independent sound frequencies using FFT. The overall frequencies are smoothed, and the most dominant frequency has SRH applied to it to establish the residual harmonics and identify the note being played. This final frequency is then converted to its appropriate letter value and used for development.

Recorder Academy was developed using development sprints breaking down implementation into two weeks block to allow time for bug fixes and helped meet specific targets by breaking down the project into smaller deadlines this helped efficiency and ensured developers stayed on top of things.

5.2 Scrum Methodology

This project was developed using scrum methodology. Scrum is a method of developing that breaks down the project into smaller goals that are tackled independently within the development team. This proves more effective for team motivation as getting feedback comes in small chunks and the overall goal doesn't seem as daunting when it is broken down. Figure 49 shows a brief diagram of how the scrum methodology functions.

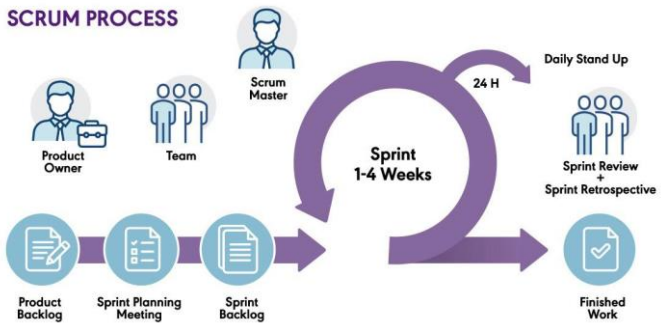


Figure 49 Scrum methodology diagram

In the context of this project sprints were conducted every two weeks with meeting with the project supervisor every 1 or 2 weeks. In the meeting would be discussed progress made towards the final product and progress and functionality that needs to be added before the next project meeting. This format made sure the project stayed on the specified development schedule and that requirements were not being looked over.

5.3 Development environment

The integrated development environment used (IDE) for the development of this project is Visual Studio Code. Visual Studio was chosen due to its seamless integration with Unity. Different version control of the project was managed through scenes in the Unity editor.

5.4 Sprint 1 – Ideation and proposal

Sprint 1 of the project spanned from the 6th to the 13th of January 2025. During this period was the ideation and proposal stage of the project. This included getting a first outline on what the game and what research needed to be done for the project.

This sprint was also necessary for finding if the project would be feasible within the timeline and technically and was the period in which the theory research was conducted (as discussed in section 2)

5.4.1 Goal

The goal for this sprint was to create a solid framework in which the project will be based backed up with research and a development timeline and what technologies were to be used. The first sprint focused heavily on ideation and whether the project idea is solid and could be done within the time and falls within areas of knowledge the developer holds.

5.4.2 Proposal

The first item was to create a proposal for the project a brief description of what the project would do and how it would be done. It was also to put gather the areas of research that goes along with the project and what technically needed to be done.

A summary of the project proposal was to explore the use of gamification and videos games in use for music teaching and education, with the overarching goal of being to create a game that would

aid music practice outside of the classroom and to create a fun and engaging experience for children and young people to practice a chosen instrument.

The project will use FFT and Unity 3D and use real-time pitch analysis to analyse sound input and provide real-time feedback to players. The focus on being to help players learning and practicing the instrument of their choosing and encouraging practice.

The proposal also references a timeline that needs to be met and potential challenges that may occur during development. The general timeline for the project was to be running from the 6th of January to 30th of April 2025 and to have a first iteration prototype be made late January. Challenges that may occur were a limited knowledge surrounding unity and FFT.

5.4.3 Research

During this sprint a research area that goes along with the project was developed. The decided on research area was gamification and how it effects music education. The full research description is available to view in section 2 but in summary research covered what gamification is and an overview on game mechanics. The MDA framework is also covered in this section and what makes a game enjoyable to play as well what drives motivation in players.

On the music education side, a case study was researched that showed a higher success rate in music education students when using a game like learning experience to learn an instrument and music theory. This helped give weight to the project and justified the theory that gamification aids music education outside of the classroom learning experience.

5.5 Sprint 2 – Functional Requirements and prototype V1

Sprint 2 was focused on gathering the functional requirements and how this was done. During this sprint the first prototype for ‘Recorder Academy’ was made and discussion on the pitch analysis aspect took place

5.5.1 Goal

The goal for this sprint was to create a rough first prototype for ‘Recorder Academy’ and to gather all the needed functional and non-functional requirements using competitor analysis and the use of personas.

5.5.2 Functional Requirements

An in-depth analysis on functional and non-functional requirements is available to view in section 3.3. In summary the modelling for requirements in 'Recorder Academy' was done by analysing applications that have a similar goal or design layout and breaking down what makes these applications functional

The apps studied were Guitar Hero and Yousician. These apps follow a similar design approach to 'Recorder Academy' with Guitar Hero using the falling blocks and timing the correct input to gain points. Yousician uses real-time pitch analysis to listen to the player musical input and provides immediate feedback much like 'Recorder Academy' is deigned to do

Personas are fictional characters created to help ensure the application meets player needs and so the developers can see the appeal of the app to the user by putting themselves in their shoes. The personas created were Lucy a student wishing to get back into music after losing touch with it as a child and wants a fun and engaging way to practice her instrument to improve her motivation. Rebecca is a music teacher that struggles to keep her students engaged during class time and find it's hard to persuade them to practice their instrument on their own time. Both of these personas would benefit from 'Recorder Academy' in separate ways. Lucy will benefit from using the application as it will help her stay motivated and Rebecca will benefit as the moral of students will improve in class if her students if they have a new fun way to practice.

5.5.3 Prototype V1

Item 2 during this sprint was to create a non-functional prototype that will display the fundamental movement of the game. Figure 50 shows the first prototype of 'Recorder Academy' with blocks with the desired letter value on it falling down the screen to the target area

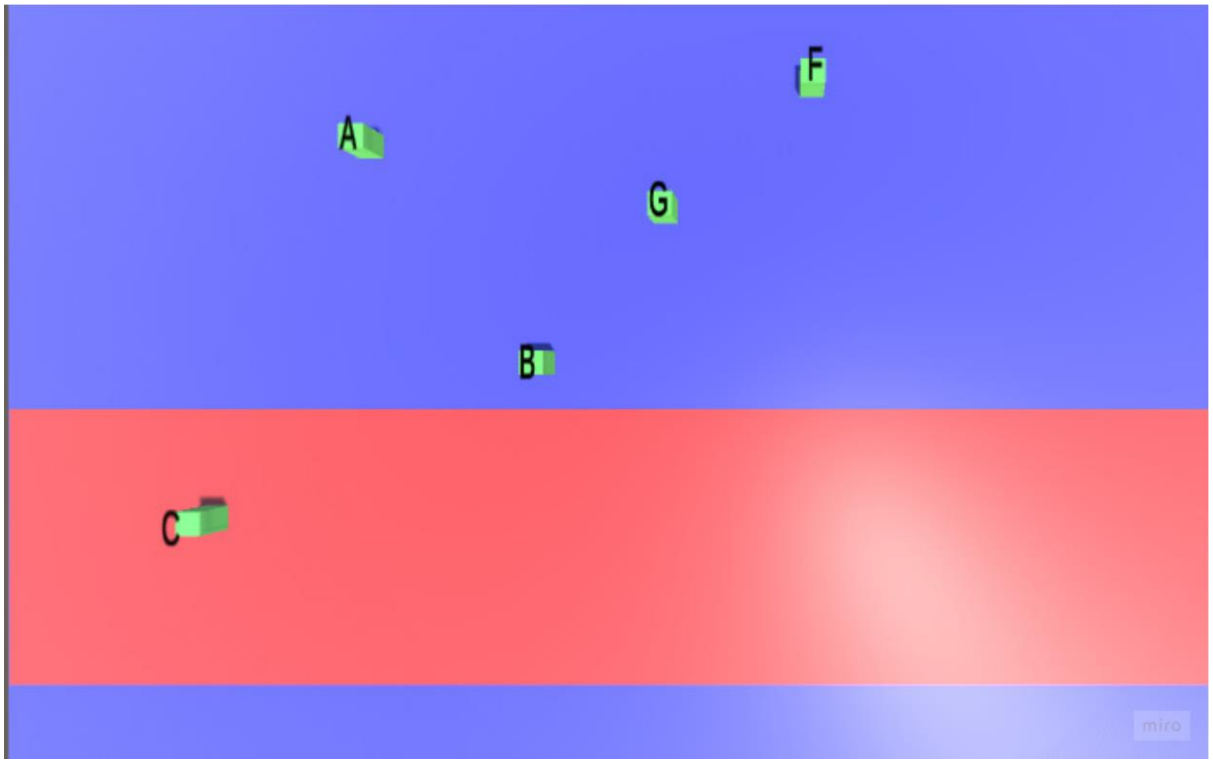


Figure 50 Recorder Academy first prototype

While this first iteration has no functionality other than falling cubes. It was to present to the supervisor of the project how the application would function once the pitch analysis element was added. It mostly served as a visual aid for explaining how the game will work.

Ideally this would have also been accompanied by some pitch analysis component but there was an error within the primary code that hindered that at the time, and it was not completed within this sprint.

5.6 Sprint 3 – Interim Presentation and Amplitude Prototype

During this sprint was the interim presentation. This was the first presentation of the project to the supervisor and the second reader. It was necessary to have a functioning prototype of some sort for this presentation as well as an overview of what had been done up to that date.

5.6.1 Goal

The goal with this sprint was to create a semi-functional prototype as proof of concept of the game. That prototype was to use amplitude as a stand in for what would become the pitch-analysis aspect of the game in later iterations.

Additionally, the goal was to create the interim presentation for the supervisor and second reader of the project that showed work done to that point, a project timeline and work that will be completed.

5.6.2 Amplitude prototype

It is mentioned in the previous sprint that the pitch analysis aspect was still non-functional and there was no clear way to fix it at the time. The alternative for this method was to create a prototype that utilizes peaks and amplitude rather than pitch. Figure 51 shows the code that did this.

```
1 reference
public void AnalyzeAudio()
{
    // Get audio data from the microphone
    microphoneInput.GetData(audioSamples, 0);
    Debug.Log("Get Data");

    // Perform frequency analysis using Unity's FFT
    AudioListener.GetSpectrumData(frequencySpectrum, 0, FFTWindow.Rectangular);
    Debug.Log("FFT");
}

1 reference
public bool DetectPeak(float[] spectrum)
{
    Debug.Log("DetectPeak");
    // Check for values above the threshold in the frequency spectrum
    foreach (float amplitude in spectrum)
    {
        Debug.Log($"my amp: {clapThreshold} {amplitude} {spectrum}");
        if (amplitude > clapThreshold)
        {
            Debug.Log("amp test");
            return true; // Peak detected
        }
    }

    return false; // No peak detected
}
```

Figure 51 Amplitude prototype code

This prototype was to mimic the principal game mechanics of the final game but removing the pitch analysis so that when the player just claps their hands at the right time that it would detect this change in amplitude and return a point. The moving block pictured in Figure 50 would trigger the analyse audio method and the microphone on the device would detect if the amplitude increase was above a certain threshold.

```
2 references
public float clapThreshold = 5.5E-11f; // Threshold for detecting a peak
```

Figure 52 Clap threshold variable – minimum value necessary to detect a peak

There were many problems with this approach though. Although it seemed to be a logical stand in at the time it did not quite help the development process as expected. With advice from the project supervisor, it was deemed not as useful as hoped.

The other problem with this approach was that on a technical level there were significant bugs and did not return the intended result most of the time. This came down to the fact that the microphone on the development device was extremely sensitive to noise and would detect a peak even if no real change in volume had occurred. This is the reason the clap threshold pictured in Figure 52 is an extremely low number.

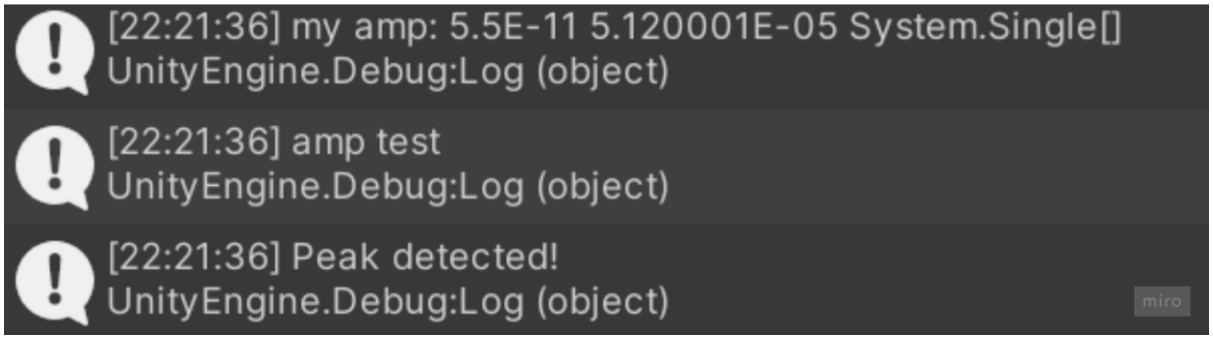


Figure 53 Intended output when peak was detected

5.7 Sprint 4 – Prototype V2 Button Input

At this point in the project it was expected to have the beginnings of the pitch-analysis aspect functional. However, this was presenting with persistent errors and under the advice of the project supervisor it was recommended to make a stand in application that uses keyboard inputs as a replacement for the audio input.

5.7.1 Goal

Create a second version prototype that will use keyboard inputs in place of audio inputs.

5.7.2 Development on audio input

The key partner script of the project audio pitch estimator was found during this time. The code in Figure 53 shows the first iteration of code used for detecting the correct pitch played. This however did not function as intended due to issues with microphone initialisation that could not be resolved within the sprint time frame.

```
0 references
void Update()
{
    if (audioSource.clip == null)
    {
        Debug.LogWarning("Microphone or Pitch Estimator is not initialized.");
        return;
    }

    // Get estimated pitch
    audioPitchEstimator.TestMethod();
    float estimatedPitch = audioPitchEstimator.Estimate(audioSource);

    if (estimatedPitch > 0) // Ensure a valid pitch is detected
    {
        Debug.Log(estimatedPitch);
        if (Mathf.Abs(estimatedPitch - C_NOTE_FREQ) <= TOLERANCE)
        {
            Debug.Log("Yes");
        }
        else
        {
            Debug.Log("No");
        }


    }
```

Figure 54 First iteration pitch estimation script

What this script is doing fundamentally was once the game starts the microphone gathers the sound data this is then put through the pitch estimation script and if the pitch is estimated to be a C it should return a Yes to the console and if it was to be a different note it would a no. However, this script never worked it was believed to be an issue with mic initialisation as at no point during testing did anything picked up by the microphone. To prevent development from coming to a complete stand still while this issue was being resolved the button input prototype started development.

5.7.3 Button Input Prototype

The button input script was a stand-in prototype whilst the pitch-analysis aspect was still in development. It took the moving blocks already established and the added TextMesh Pro component and read the text on the component and compare that to the keyboard key pressed to give a point

```
0 references
private void OnTriggerEnter(Collider other)
{

    // Check if the object entering is the cube by tag or name
    if (other.CompareTag("TargetCube"))
    {
        // Get the TMP_Text component from the cube
        TMP_Text textMeshPro = other.GetComponentInChildren<TMP_Text>();
        if (textMeshPro != null)
        {
            requiredKey = textMeshPro.text.Trim().ToLower();
            isInTargetArea = true;
            Debug.Log("Press " + requiredKey.ToUpper() + " now!");
        }

        Debug.Log("Something entered the target area: " + other.name);
    }
}
```

Figure 55 Button input getting the letter from the text mesh pro component

textMeshPro.text reads the text on the text mesh pro component once the object enters the trigger. This text is saved as required text. If the correct key is pressed whilst the object is in the target area the player gets a point. Like the first prototype this does not have any pitch analysis aspect, but it did help with overall development of the project. Reading the text off the text mesh pro component would become an important part of the point system of the final game and the point system used in this prototype is the same as the one in the final game with some minor changes.

5.8 Sprint 5 – Prototype V3

During sprint 5 the issue regarding the pitch analysis was resolved and final development could begin.

5.8.1 Goal

The goal was to create a first prototype of a functioning game with audio input and pitch analysis working. The goal with this sprint was to also rework the fundamental code structure to make the game more dynamic

5.8.2 Win Condition Rework

The point condition needed to be reworked entirely at the time of this sprint it was still only give a point of a C was played and this value for a C frequency was hard coded into the application. The audio pitch estimator was now returning the frequency played using the get name from frequency function could now be implemented to compare the letter values of each block. The TextMesh Pro value would be read once the note enters the target area. This would become the required note variable. After that the audio pitch estimator would get the fundamental frequency that would get passed through the get name from frequency function that would convert it to its letter value. So in the case of Figure 56 once the note enters the target area the required note will be C. if the player correctly plays a C note the audio pitch estimator will detect a fundamental frequency of approximately 261.37Hz this frequency will then get passed through the get name from frequency function that will convert 261.37Hz to a C this will become the detected note. The application then compares the two letter values. If they are the same the player gets a point.

Figure 56 Visual aid for above paragraph

```
// Get the TMP_Text component from the TargetCube
TMP_Text textMeshPro = targetCube.GetComponentInChildren<TMP_Text>();
if (textMeshPro == null)
{
    Debug.LogError("No TMP_Text found on TargetCube!");
    return;
}

string requiredNote = textMeshPro.text.Trim().ToLower();
Debug.Log("Required Note from TargetCube: " + requiredNote);

// Get the detected frequency and note
float frequency = pitchEstimator.Estimate(audioSource);
if (float.IsNaN(frequency))
{
    Debug.LogError("Frequency is not a number Skipping.");
    return;
}

string detectedNote = GetNameFromFrequency(frequency).Trim().ToLower();
Debug.Log("Detected Note: " + detectedNote);

// Compare notes
if (detectedNote == requiredNote)
{
    Debug.Log("Correct note played!");
    pointSystem.AddPoint();
```

Figure 57 Code enabling this functionality

5.9 Sprint 6 – Additional Functionality

Now that base functionality was correctly implemented. Game mechanics and features needed to be added to allow the game to be able work as intended. Up to this point the blocks were hard coded with the letters and block not instantiating in anyway. Once the three block went off the screen the blocks the game was effectively over. A way to dynamically instantiate blocks and text needed to be implemented.

With the current standing point system if the player was to play the correct note once the block would give points repeatedly whilst it was still in the target. For proper gameplay it needs to be implemented that only one point could be given per correct note played.

5.9.1 Goal

The goal for this sprint was to add more functionality to the game like instantiating blocks and a fully implemented point system

5.9.2 Note Instantiation

For the block to instantiate they need a trigger to activate. The trigger wall as seen in Figure 17 triggers the new block trigger script this handles the instantiation of new blocks to the game. During this sprint the blocks instantiated at a random y value along the line where the first note appears.

```
if (other.gameObject.layer == LayerMask.NameToLayer("BlockSpawnLayer")) // Only react to this layer
{
    Vector3 spawnPosition = new Vector3(-13.22f, -0.98f, Random.Range(-8f, 0f));
    Quaternion spawnRotation = Quaternion.Euler(0f, 90f, 0f);

    // Instantiate a new block and store reference
    GameObject newCube = Instantiate(TargetCube, spawnPosition, spawnRotation);
```


Figure 58 Block instantiation code at random y value

The text was already assigned to the block so that was instantiating randomly assigned to each block too as they were being instantiated from the array of letter values seen in Figure 20. Although at first this caused a problem as when the new block would be instantiated it would change the letter value that was attached to the block that triggered the new block instantiation in the first place. To ensure that this did happen the new block being instantiated to have been named new cube rather than just searching for the target cube tag which it was doing prior.

```
// Find the target cube by tag
GameObject targetCube = GameObject.FindGameObjectWithTag(TargetCubeTag);
if (targetCube != null)
{
    // Instantiate a new block and store reference
    GameObject newCube = Instantiate(targetCube);

    if (!newCube.GetComponent<CubeStatus>())
    {
        CubeStatus cubeStatus = newCube.AddComponent<CubeStatus>();
        cubeStatus.pointsAwarded = false; // Mark points as not awarded for this cube
        Debug.Log("CubeStatus component added to the new cube. Points awarded: " + cubeStatus.pointsAwarded);
    }

    // Modify only the new block's TextMeshPro component
    TextMeshPro textComponent = newCube.GetComponentInChildren<TextMeshPro>();
    if (textComponent != null)
    {
        string randomName = _names[Random.Range(0, _names.Length)];
        textComponent.text = randomName;
```

Figure 59 instantiating new cube and only editing that cubes text component

5.9.3 Point System adjustments

The point system needed some adjustments as once the player got a point the point counter would continue to give points whilst the block was still in the target area. To combat this a cube status component was added. The cube status ensured that only one point could be given to each correct note. Cube status uses a point awarded Boolean which when initially added to the block is set to false once a point has been awarded the Boolean changes to true. When the Boolean value is true it stops giving points to the player for that note and this repeats for each block instantiated.

```
// Check if points have already been awarded
CubeStatus cubeStatus = targetCube.GetComponent<CubeStatus>();
if (cubeStatus == null)
{
    cubeStatus = targetCube.AddComponent<CubeStatus>();
}

if (cubeStatus.pointsAwarded)
{
    Debug.Log("Points already awarded for this cube. Skipping.");
    return;
}
```

Figure 60 code checking is the cube status script is on cube if not adding and if points awarded is true not adding another point

```
// Compare notes
if (detectedNote == requiredNote)
{
    Debug.Log("Correct note played!");
    pointSystem.AddPoint();
    cubeStatus.pointsAwarded = true; // Mark points as awarded for this cube
    ChangeTargetAreaToGreen();

}
```

Figure 61 Code giving point and changing point awarded to true

```
0 references
public class CubeStatus : MonoBehaviour
{
    0 references
    public bool pointsAwarded = false;
}
```

Figure 62 Cube status component

5.10 Sprint 7 - Design

With all game mechanics and requirements met the final sprint for development added UI integration and playability changes. A home and instruction screen and minor aesthetic changes that would increase playability as well as changes to the UI to give immediate feedback to the player

5.10.1 Goal

To increase playability and user experience by adding a home and instruction screen as well immediate feedback.

5.10.2 Note Element

Up until this point the object that has been in play has been a cube that moves down the screen, so the blocks get changed to the note object discussed in section 4.2.2.2 and a stave design has been added to the gameplay screen to improve the overall look and feel of the gameplay.

Additionally, the notes were changed from spawning in a random position to always spawning in the same place based on the note that is attached. This is done by using a dictionary meaning each letter has a position definition. These definitions are assigned using an array map. I.e. array index 0 on the letter array gets mapped to array index 0 on the position array. This is also discussed further in section 4.2.2.2

5.10.3 Screens

Additional screens were added to the gameplay to allow players to start the game and a game over screen to indicate when the game is over. The start screen (Figure 7) had two buttons and redirects the players to the gameplay or the instruction screen.

The game over screen appears when 10 incorrect notes are detected the game over screen overlays over the game screen stops the game movement and shows a button that the allows the player to redirect back to the home screen.

5.10.4 Visual Feedback

Immediate feedback needed to be added to improve the overall gameplay and aid the players understanding of where they need improvement. This was done by once the player gains a point the target area turns green and when the player plays an incorrect note the target are turns red. This is discussed further in section 4.2.2.5

5.11 Conclusion

The development of *Recorder Academy* followed an iterative design and development approach and used Scrum methodology, allowing for the breakdown of progress into smaller sections and receive feedback on each section of development. Development started with early ideation and competitor analysis and progressed to prototype iterations and final gameplay implementation, the project steadily introduced real-time pitch detection, dynamic gameplay elements, and an engaging user interface. Despite issues with some sections of code especially regarding the pitch analysis aspect, a vital section of the program the project progressed to where it needed to be when broken down into parts to hit the development timeline. The final project is effective in aiding the practice of music using gameplay mechanics and creates a fun and engaging experience for players.

6 Testing

6.1 Introduction

This chapter describes the functional and user testing of 'Recorder Academy'. Both of which are necessary for ensuring the game is fully functional and enjoyable to the player. This will also help figure out where the applications downfalls are and how the game can be improved in the future.

6.2 Functional Testing

Functional testing is the process of testing the game against the needed functional requirements to make sure it meets the requirements, and the game works as intended. Functional testing was conducted on three different aspects of the game, the point system, pitch estimation and navigation. These three aspects were the most fundamental aspects of the game and ensured the game worked as intended

6.2.1 Pitch Estimation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Pitch Accuracy test – over a scale of notes how many are detected correctly	The C major scale played	8 notes detected correctly	8 notes detected correctly	Estimation has a slight delay, and notes need to be played multiple times

6.2.2 Point System

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
2	Correct pitch detected – Player receives point	Correct note played	One point	One point gained	
3	Incorrect pitch detected – Player doesn't receive point	Incorrect note played	No point	No point gained	
4	No pitch detected – Pitch Estimation doesn't run	Nothing played	Pitch detection skipped	Frequency is not a number skipping	
5	5 correct notes played	5 notes played correctly	5 points	Stops gaining points after a few notes	
6	3 correct notes played – 3 incorrect notes played	6 notes half correct/ half incorrect	3 points	Stops gaining points after a few notes	

6.2.3 Navigation

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
1	Navigate from home screen to game	Click on start on home screen	Navigated to game play screen	Navigation functions correctly	
2	Navigate from instructions screen to home	Click on back to home on instruction page	Navigates to home screen	Navigation functions correctly	
3	Navigate from game over screen to home	Click on home on the game over screen	Navigates to home screen	Navigation functions correctly	

6.2.4 Discussion of Functional Testing Results

For the most part the functional requirements of 'Recorder Academy' work as intended. When played separately the notes played are detected correctly and the navigation between screens work seamlessly as expected. However, the main issue with gameplay as it stands is that as play continues the player stops getting points after a certain amount is given. This issue is intermittent and there has been instances of gameplay where this has not been an issue at all, but it is still an issue that with further development will get fixed. The current thought for why this is occurring is the point awarded Boolean value attached to each note and there is an error with instantiation of this Boolean as new notes are being instantiated in. With further time and developed a more robust way of ensuring one note is equal to one point will be developed.

6.3 User Testing

User testing was conducted on 5 willing participants. With participants consent they were firstly screened on their age music ability. These preliminary questions were to gage if the user chosen to test the game falls within the target audience. Ideally the player would be between the ages of 18 to 25 and would have some musical experience. As the game is designed to be a practice tool rather than a learning tool some musical experience would be beneficial to the stability of the test results.

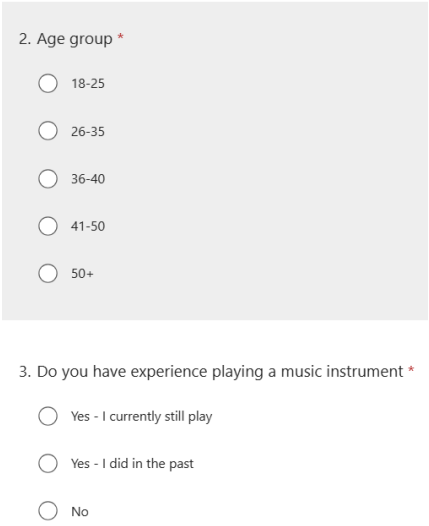


Figure 63 Preliminary screening questions

The players were then asked to play the game as they normally would. They were instructed to view the instructions and play the game when they were ready. A test supervisor was present for the test and was there to answer any questions as needed. However, they were instructed to provide as little

intervention as possible. This would help test how intuitive and clear the instructions and game play are. After playing the game for 5 minutes or reaching the game over screen. (whichever came first) the players were given a set of questions to answer these questions regarded the gameplay mechanics, user experience and an area to give general feedback and suggestions.

The test subjects ranged in age and music ability level. While the game is targeted towards young people with some music experience it was beneficial to test a range of ages and abilities to see if the design and instructions were inclusive to all ages. Out of the 5 participants 2 fell between 18-25, 1 person was aged between 26-35, 1 person was aged between 36-40 and finally one 1 person was aged between 41-50.

Figure 64 Infographic of age breakdown of testers of Recorder Academy

Within these 5 testers a range of music abilities were recorded as well. The options given in this question were, no, yes – I currently still play and yes – I did in the past. These options were chosen to account for testers that have underlying music skills but aren't currently utilising them, testers that are currently practicing music and testers that have no skill at all. These demographics are generally the assortment of player that will play 'Recorder Academy'.

Figure 65 Demographic of testers that have music ability

The majority of tester have music ability. In a range of instruments including piano, violin and recorder.

ID ↑	Name	Responses
1	Alexandra Killian	Piano
2	Alexandra Killian	Violin, Piano
3	Alexandra Killian	Recorder, Piano

Figure 66 Different music instruments that testers had ability in

To note here, all participants appear to be coming from the one name this is due to the fact that testers were asked to fill out the user survey on the device that 'Recorder Academy' was hosted on. For data collection purposes no name and details of testers were collected, and testers were tracked with the ID number instead.

After playing the game the game the testers were asked to select on what level they agreed or disagreed with statements pertaining to their experience of the game. For the most part the feedback for 'Recorder Academy' was positive. With the responses swaying mostly towards agree and strongly agree.

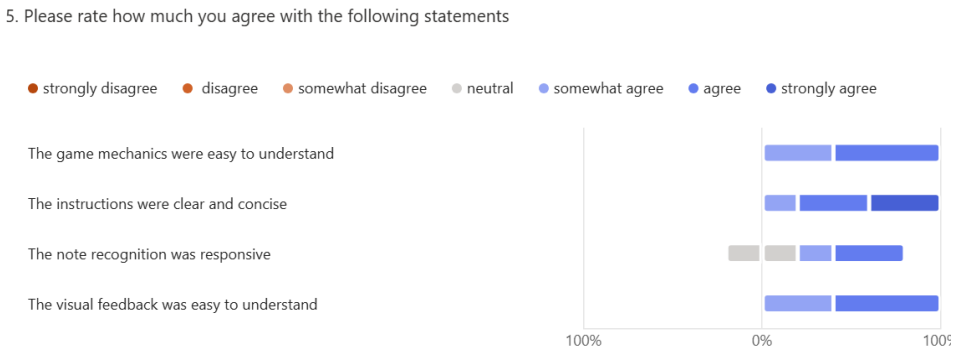


Figure 67 User testing results on game mechanics

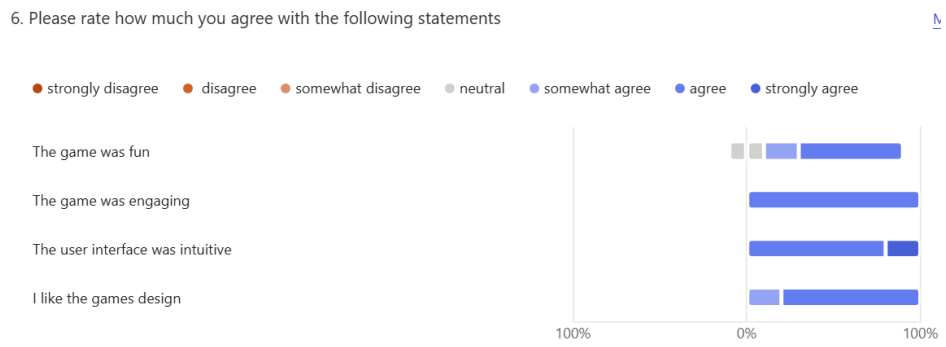


Figure 68 User testing results on user experience

Noticeable discrepancies in these results are the two testers that only somewhat agreed that the note recognition was responsive and that the game was fun.

Lastly there was a selection of optional question for the testers to answer based on their experience playing the game. These were kept general and allowed for elaboration if the tester wished to offer their in-depth feedback. For this section of discussion each tester will be referred to by their index number as these answers are subjective to the individual tester.

All testers found the game at least partially enjoyable, testers 3 and 5 were the outliers in this case as tester 3 found the game somewhat enjoyable but struggled with gameplay. This player was one of the players that had declared they had no previous experience playing a musical instrument so this could have influenced their experience of play. Tester 5 stated the game was difficult to follow and that the instructions were slightly hard to follow.

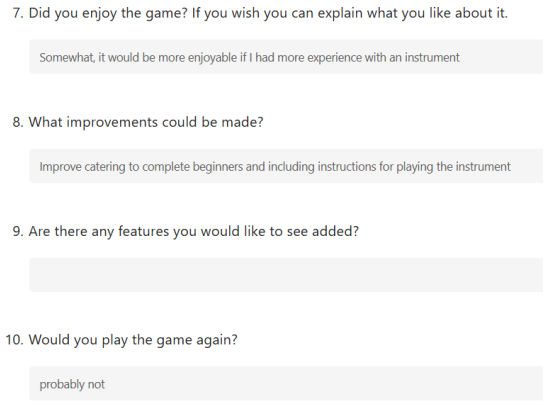


Figure 69 Tester number 3 response to the opinion questions

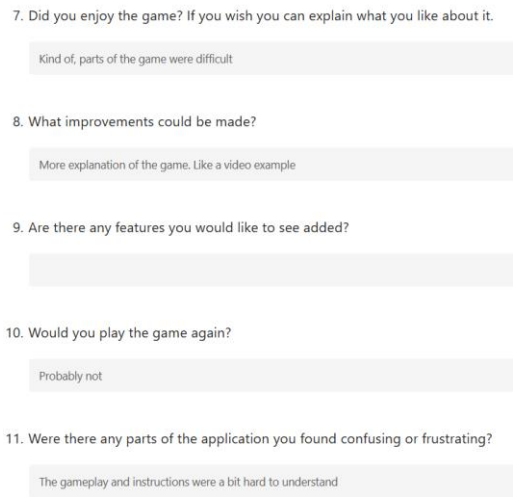


Figure 70 Tester number 5 response to the opinion questions

From these responses what can be gathered is that there should be changes made to the instruction pages to make the core gameplay mechanics easier to understand for the user for instance a video example of gameplay would act as a visual aid to the player and basic music instructions for complete beginners to recorder playing. Perhaps an infographic that would display the correct fingering of the note shape could be displayed during gameplay. The supervisor of this project suggested these features during development but were not able to be implemented at present due to time constraints.

Additionally, tester 2 and 6 of the game recommended a song feature be added to 'Recorder Academy' this was also feature that was discussed during development but had to be cut due to time constraints. In futures additions to the game this will be added to increase playability for users that would already have a robust music knowledge and want a fun and easy way to learn and practice new songs.

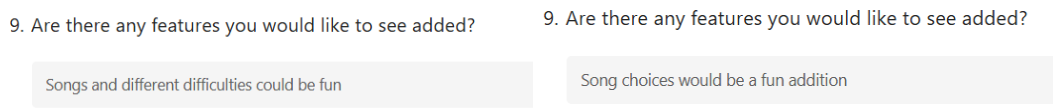


Figure 71 Tester 2 and 6 responses requesting adding a song feature

6.4 Conclusion

In conclusion, the functional and user testing of *Recorder Academy* highlighted that the functional gameplay mechanics of pitch estimation, point system, and navigation generally work as intended for the most part. However, testing revealed a minor but significant issue with the point system that will require further development to resolve. User testing was conducted across a range of ages and musical abilities and confirmed that the game is mostly intuitive and enjoyable to play though some players struggled with understanding instructions and gameplay without prior musical experience. These players can gain a more enjoyable gameplay experience with the addition of clearer instructional materials and visual aids such as gameplay videos and note fingering guides. Testers also suggested the implementation of a song feature would enhance long-term engagement. Overall, the testing process was necessary in validating the game's strengths and identifying key areas for improvement in future iterations.

7 Project Management

7.1 Introduction

Effective project management was vital to the development of 'Recorder Academy'. The project was exceptionally challenging technically and was put under a strict timeline, project management needed to be implemented to make sure the project reached its individual goals on time. The project was developed under the scrum methodology breaking down the project development process into two-week sprints with feedback being given at the end of each sprint. The following chapter breaks down the effectiveness of each sprint and the ways in which the project could have been development.

7.2 Project Phases

The project phases were broken down as follows

- Proposal and Initial research
- Requirements
- Design
- Implementation
- testing

All the project phases were important to make sure the game worked as intended within the given time frame. The proposal and initial research phase involved the idea for the game becoming a reality and research was done accompanying it to back up the effectiveness of the game.

Requirements included gathering the functional and non-functional requirements that would make 'Recorder Academy' functional. Breaking down the individual aspects of the game and how they would be created and making sure nothing gets forgotten or slips through the cracks

Design was the creation of the UI and system design including colour palette and fonts to create an inviting and aesthetically pleasing user environment that will courage users to play more and not weigh on the player's mental load.

Implementation was the programming of the game. Included the different file structure and how they react and work together to create the game and making sure it met all functional requirements. This was done in an iterative design plan slowly incorporating more game elements in each sprint.

Finally, testing was done to ensure that the game was functional on a technical level and easy and fun to use for the player.

7.2.1 Proposal

Project development started with the proposal and initial research phase in which the project was ideated on and the final idea for the game came to fruition. This idea then needed research to back it up to ensure its validity and to aid the games user experience and design.

The idea came about after being inspired by games that use an audio input as the controller, mobile games like scream go chicken and yousician. This idea was intriguing, and curiosity was built on whether it would be feasible within the constraint of the chosen game engine and whether it would be feasible within the developer's knowledge. With further research into other similar project and applications it was decided that the project would base around pitch analysis and for the research element could be paired with gamification and music education

Doing research into gamification was important for establishing the base goal of the project which was to create a fun and engaging new way for children and young adults to practice a chosen instrument. Research surrounding the foundations of gamification and what makes it effective ensured that 'Recorder Academy' would achieve this goal. It was also important to gain a baseline understanding of how and why gamification works and what techniques are implemented to create a more effective learning experience.

These techniques did help the research phase of the project however during this time there was still great apprehension on whether the game would be feasible within the developer's knowledge and whether it would be able to be completed within the given deadline.

7.2.2 Requirements

Gathering requirements was done to establish what was needed to make the game fundamentally functional. Breaking down the game to just its core functionality and how they would be created. This was done by looking at similar applications and finding similar projects that were available.

Competitor analysis was done on Guitar Hero and yousician. These games were either functionally similar or similar in design to the game that 'Recorder Academy' intended to be. Yousician used the same pitch analysis technology that would be used for the project and Guitar hero used falling note objects which would also be implemented.

After gathering the requirements, they were listed and broken down into each part that need to be created to make sure the game met these requirements. The requirements being presented as a list made sure nothing would be forgotten and became an easy way to check off progress as things were created.

7.2.3 Design

The design phase made sure the game would be appealing to look at and designing of the code necessary to make the game function properly. This was done through wireframing, iterative design and pseudocode to get a base understanding of the code structure.

Design was mostly done within unity with the initial design being falling blocks on a screen to give a visual aid to what the game would look like and some use of Figma to make a high design level wireframe. Making sure the game looked appealing. While design is important it came as an afterthought due to the implementation taking precedence. If there were to be more time for the project more energy would be put in to make sure the games UI design is user friendly and aesthetically appealing.

7.2.4 Implementation

Implementation covered the program and code design structure of the game. This code design went through multiple iterations before finally deciding on a straightforward way to handle the pitch analysis and adding that to the game's features.

As there were many issues with the pitch analysis technology at the beginning implementation was done by working on separate game features alongside the pitch analysis software. Then when each part of the program was functioning it was added to the game. It started with amplitude detection, then turned to button inputs and development of the text component and once the pitch analysis was working that was added and the final prototype of the game could be developed.

This phase of development was the most challenging with many hurdles along the way. The iterative design approach and support from the project supervisor did help this. Even though there were errors during the course of development it did not halt or stop production in any way and instead redirected attention to other necessary aspect of the project.

7.2.5 Testing

Testing covered the functional and user testing of the application. Functional testing was to ensure the individual elements worked together to create a functional game. The areas tested were the pitch estimation, point system and game navigation. These areas worked mostly as intended despite a significant bug in the point system that due to time constraints was unable to be fixed.

User testing was conducted on 5 participants over a range of ages and musical abilities. The response from user testing was overall positive with some testers providing valuable feedback of the instruction screen and how the game could be improved in the future.

Overall, the testing section was a success for the most part and with additional time the persistent bugs would be fixed and the improvements suggested by the testers would be implemented. Additionally with more time allowing a larger tester base would have been added to improve the overall test data and gain as many opinions as possible.

7.3 Reflection

7.3.1 Your views on the project

Initially this project produced a lot of apprehension about whether my knowledge would be enough to achieve the goals I had set. Despite those initial doubts, the project came together well, and the result is something I'm genuinely proud of. For the most part, work was done effectively under the Scrum methodology, and development was able to adapt when challenges arose. I acknowledge there was mistakes and downfalls within my knowledge that made this more difficult and there was many I could improve on. However, the game's progress feels like a strong accomplishment with where it got to be within the project course and with more time and development it can become something great. With additional time, adding a feature where notes spawn in rhythm with actual songs would greatly enhance replay ability and appeal to more musically experienced users. More time would also have allowed for greater attention to the game's visual design, helping it reach its full potential both in function and appearance.

7.3.2 Completing a large software development project

Working on a large development project was a steep learning curve, it was a very interesting experience and very fruitful for my education and I believe it will greatly help me in a professional setting. It was challenging keeping on top of the time constraints and making sure everything was

getting the appropriate amounts of time and attention for the development to progress but slowly seeing the game come together step by step was rewarding and fulfilling and I got a great sense of accomplishment as each piece of functionality was added.

7.3.3 Working with a supervisor

Working alongside a supervisor proved to be incredibly helpful throughout the project. The regular meetings helped structure the workflow and ensured steady progress. Each meeting was treated as a deadline and ensured certain functionality was working in time for the meeting. The feedback and suggestions provided during these meetings were motivating and brought ideas to the table that would help improve 'Recorder Academy' as a whole. Having a supervisor with strong knowledge in music was especially beneficial, as that was an area where I lacked experience, and their insights greatly supported the development of the game.

7.3.4 Technical and project skills gained

This project significantly strengthened my technical skills in Unity. Over the course of development, I gained a deeper understanding of Unity's core functionality and how to debug properly within unity using if statements and debug.log statements to confidently identify and fix issues efficiently. I also explored new technical areas, including microphone input handling, Fast Fourier FFT and SRH, which allowed for an understanding of how pitch detection and music analysis can be implemented within Unity.

alongside technical skills, this project helped develop my project management skills. Working on a large-scale project over an extended time period required more structure than I was used to and effective use the Scrum methodology was essential for staying organised. Managing tasks in smaller iterations allowed for more consistent progress and easier ensuring development did not come to a halt when problems arose. The project also helped improve my time management skills, and I learned to set achievable goals within specific timeframes and balance multiple areas of development effectively and simultaneously.

7.4 Conclusion

Effective project management played an important role in the development of 'Recorder Academy'. Using the Scrum methodology allowed the project to be broken into manageable, iterative sprints, allowing for steady progress even with technical challenges and time constraints. Each project

phase—from proposal and requirements gathering to design, implementation, and testing—contributed to the game's functionality, usability, and overall quality. While initial apprehensions and technical hurdles were present, regular reflection, guidance from the project supervisor, and the ability to adapt were very important for getting the project to where it is. Ultimately, this experience led to the creation of a functional and engaging user experience but also significantly strengthened the developer's technical abilities, time management, and confidence in handling large-scale software projects.

8 Conclusion

This project came to be from the belief that anything can be made fun. The goal of 'Recorder Academy' was to create a fun and engaging way to practice music through gamification and the use of game mechanics.

This project was developed in the unity game engine using the C# programming language for scripting. Unity was chosen due to its performance for rendering 3D games and its extensive online community for access to assets and online tutorials. Within Unity 'Recorder Academy' Fast Fourier Transform (FFT) and Summation of Residual Harmonies (SRH) algorithms to allow for real time pitch estimation. This allowed musical note input from the user to be recognised and evaluated. Unity's in-built canvas UI system and TextMesh Pro packages were utilised to create the user interface.

Background research done for 'Recorder Academy' was conducted on game mechanics and gamification techniques, gamification in the music classroom and FFT and SRH algorithms. Case studies and research studies showed that gamification within a music classroom setting improved engagement and results in the material. Technical research was conducted on the workings of FFT and SRH algorithms and how they would operate within Unity to convert the audio input into its fundamental frequency. Functional requirements were gathered by studying similar applications like Guitar Hero and Yousician to establish what needs to be included in the application for it to function as intended.

Guitar Hero was the inspiration for the scrolling note design of 'Recorder Academy'. Design of the application also centred around immediate feedback back to the player with the target area turning green or red when a point is awarded or not. Bold colour choices were made to appeal to a younger target audience. Design was developed iteratively using Unity and Figma with design elements being added throughout the development process.

Implementation was broken down into 7 2-week sprints. Different prototypes of the application were developed adding more features slowly and iterating on the project during each sprint. Firstly, was amplitude followed by button inputs and finally implementing the pitch estimation script created by GitHub user Nakakq to estimate the pitch. What this script does is convert the pitch to its respective letter value and compare that letter value to the TextMesh Pro object on the note. With each new feature and function added it was tested and finalised before adding to the project.

User and functional testing took place to conclude if the project met the functional requirements and to ensure that the game was intuitive and usable to the player. Functional testing was done on the pitch estimation aspect, the point system and navigation. It was during this time that a bug within the point system came to light and will need to be fixed in the future. User testing was done on 5 participants with a range of ages and music abilities being tested. The consensus of testers was overall positive with some suggestions of more clear instructions being implemented and the addition of songs would help increase replay ability.

Whilst there was technical challenges and mishaps throughout the development process ‘Recorder Academy’ meets the core functional requirements of estimating pitches correctly, awarding points and providing immediate feedback to the player. The game is not perfect with much room for improvement but given the tight development timeline the application works well as expected.

The project provided teaching in the Unity game engine and audio pitch estimation within Unity. Skills that relate to large scale project management, namely time management skills and an iterative design approach were improved greatly during the course of the project’s development.

In the future iterations of ‘Recorder Academy’ more attention will be put into the UI/UX of the project to further improve the look and feel of the application. Songs and difficulty settings will be added to appeal to wider range of players. Additionally further improvements to the fundamental code of the game to make the game run smoother and more consistently

Overall, this project has been a pleasure to work on and a fantastic learning experience. It taught me to overcome many challenges and helped me develop new skills.

References

Alexandre Bruffa. (2022, April 7). *Combining audio analysis and shaders customization with Unity3D*. Ab. https://alexandrebruffa.com/combining-real-time-audio-analysis-and-shaders-customization-with-unity3d/

Dickey, M. D. (2006). "Ninja Looting" for instructional design. *ACM SIGGRAPH 2006 Educators Program on - SIGGRAPH '06*. https://doi.org/10.1145/1179295.1179313

Dondlinger, M. (2007). Educational Video Game Design: A Review of the Literature. *Journal of Applied Educational Technology*, *4*(1). http://hypermedia468.pbworks.com/w/file/fetch/82077116/Dondlinger2007EducationalVideoGameDesign.pdf

Gabe Zichermann. (2011). *Gamification by Design.* O'Reilly Media.

Games, R. (2023a, May 8). *Make Your MAIN MENU Quickly! | Unity UI Tutorial for Beginners*. Www.youtube.com. https://www.youtube.com/watch?v=DX7HyN7oJjE

Games, R. (2023b, June 17). *Make a TIMER & COUNTDOWN in 5 Mins | Unity Tutorial for Beginners*. Www.youtube.com. https://www.youtube.com/watch?v=POq1i8FyRyQ

Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment*, *1*(1), 20. https://doi.org/10.1145/950566.950595

Gomes, C., Figueiredo, M., & Bidarra, J. (2014). Gamification in teaching music: case study. *EduRe'14*. https://repositorioaberto.uab.pt/handle/10400.2/3478

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does Gamification Work? -- a Literature Review of Empirical Studies on Gamification. *2014 47th Hawaii International Conference on System Sciences*, 3025–3034. https://doi.org/10.1109/hicss.2014.377

Hilliard, B. (2013). *Optimising Comprehension and Shaping Impressions COLOUR PSYCHOLOGY Section Title Page*. https://www.seahorses-consulting.com/DownloadableFiles/ColourPsychology.pdf

Hunicke, R., Leblanc, M., & Zubek, R. (2004). *MDA: A Formal Approach to Game Design and Game Research*. https://cdn.aaai.org/Workshops/2004/WS-04-04/WS04-04-001.pdf

Jenkins, H., Klopfer, E., Squire, K., & Tan, P. (2003). Entering the education arcade. *Computers in Entertainment*, *1*(1), 17. https://doi.org/10.1145/950566.950591

Kanazawa, M. (2022, January 23). *108 Gamification Elements and Mechanics*. Mambo Enterprise Gamification Software. https://mambo.io/blog/gamification-elements-and-mechanics

Liberty, S. (2023, December 4). *Gamification: What are the 8 Core Drives? - Sam Liberty - Medium*. Medium. https://sa-liberty.medium.com/gamification-what-are-the-8-core-drives-9c87ef85516d

nakakq. (2020). *GitHub - nakakq/AudioPitchEstimatorForUnity: A simple real-time pitch estimator for Unity*. GitHub. https://github.com/nakakq/AudioPitchEstimatorForUnity

Salzman, M. C., Loftin, R. B., Dede, C., & McGlynn, D. (1996). ScienceSpace. *Conference Companion on Human Factors in Computing Systems Common Ground - CHI '96*. https://doi.org/10.1145/257089.257167

Takahashi, D. (2011, January 3). *Zynga's CityVille becomes the biggest-ever app on Facebook*. VentureBeat. https://venturebeat.com/games/zyngas-cityville-becomes-the-biggest-ever-app-on-facebook/

Tortoise, R. M. (2023, October 20). *Procedural Generation: Endless Runner Unity Tutorial (Updated 2023)*. Www.youtube.com. https://www.youtube.com/watch?v=Ldyw5IFkEUQ

Wagner, C. (2016). *Digital Gamification in Private Music Education*. *7*(1), 115.