Music-Reactive Procedural Content Generation in Video Games

University of
St Andrews

Marcus Yiu Yeung Yip*

10 January 2024

*Supervised by Prof Ian Miguel

Abstract

Procedural Content Generation (PCG) has emerged as a powerful technique to create dynamic and engaging gameplay experiences. It has helped game developers create unique, immersive and expansive game worlds at a lower cost than with hand-crafted game content. The aim of this project is to further explore the possibilities of PCG in video games, marrying it with the idea of audio analysis to produce an enhanced gaming experience where players can enjoy both listening to music and playing video games in an entirely new, interesting way.

To achieve this, a game integrating the idea of music-reactive procedural content generation will be developed, tested and evaluated. The game will flip the concept of music being created to accompany different game scenarios and instead, will be able to dynamically construct game scenarios (by generating in-game content such as visuals, environments, gameplay mechanics) in response to the music fed into the system.

This paper explores the concept of Music-Reactive Procedural Content Generation, and an implementation of it into a prototype game. - A Rhythmic rail shooter titled Flo.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 12,755 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Acknowledgements

I would firstly like to extend my sincere gratitude to Professor Ian Miguel for supervising me throughout the completion of my project. This project brings together some of the things I am most passionate about in my life. Being able to discuss, share and build on these passions each week with Professor Miguel has been an incredibly enjoyable and insightful experience, I truly feel privileged to be able to work on this with him.

I am also thankful to Student Services and the School of Computer Science as a whole for pushing me to be the best version of myself whilst keeping my physical, emotional and mental well-being in check. My years studying here have definitely been both the most challenging and rewarding experiences I've ever had.

Finally, I would like to express a heartfelt thanks to my family and friends for always believing in me and encouraging me to achieve my best. Especially my parents, without their love, encouragement and support I would not have been able to get to where I am today and for that I am forever grateful.

Contents

1 Introduction

Procedural Content Generation (PCG) has emerged as a powerful technique to create dynamic and engaging gameplay experiences. It has helped game developers create unique, immersive and expansive game worlds at a lower cost than with hand-crafted game content. The aim of this project is to further explore the possibilities of PCG in video games, marrying it with the idea of audio analysis to produce an enhanced gaming experience where players can enjoy both listening to music and playing video games in an entirely new, interesting way.

To achieve this, a game integrating the idea of music-reactive procedural content generation will be developed, tested and evaluated. The game will flip the concept of music being created to accompany different game scenarios and instead, will be able to dynamically construct game scenarios (by generating in-game content such as visuals, environments, gameplay mechanics) in response to the music fed into the system.

As the gameplay should feel intuitive, the game should be able to represent the movement of music. Rail shooters are a sub-genre of the classic "Shoot 'em up" game genre where the player travels along a fixed path, with limited movement to evade damage whilst shooting enemies. They intrinsically incorporate the notion of movement as the player is constantly moving, yet, like listening to music, there is a lack of control over the direction in which the music moves towards. Additionally, rail shooters often include 3D graphics which can aid in making the game feel more natural. As such, the developed game will be a (musical) rail shooter.

2 Context Survey

Music-Reactive Procedural Content Generation describes the process of analysing audio input (music) in real-time and dynamically generating (video game) content, algorithmically, based on the findings. The purpose of this proposed method is to provide a novel way for people to enjoy both playing video games and listening to music at the same time within an immersive environment. Key ideas which contribute towards this concept include audio analysis and procedural content generation in video games, both of which will be further

detailed below in this context survey. Additionally, any currently existing related work will be outlined as to provide a baseline to the proof of concept of Music-Reactive Procedural Content Generation.

2.1 Procedural content generation

The concept of procedural content generation (PCG) has been around for several decades, although it has gained more attention and advancement in recent years. Fundamentally, PCG can be thought of as a method to turn unstructured randomness or noise into structured content by use of algorithms. In the modern age, many of the most popular video games enjoyed by people all over the world include some element of PCG. Examples of this can be observed in the creation of 3D voxel worlds in Minecraft[19] and the generation of tetrominos in Tetris[12]. Although the exact genesis of PCG is ambiguous, one of the earliest video games which frequently employed the method was Rogue[10]. From the rooms, corridors, enemies, loot- Rogue and later "Rogue-like" games used PCG to generate entirely new, unique dungeons which players could explore, keeping them entertained each time they played the game.

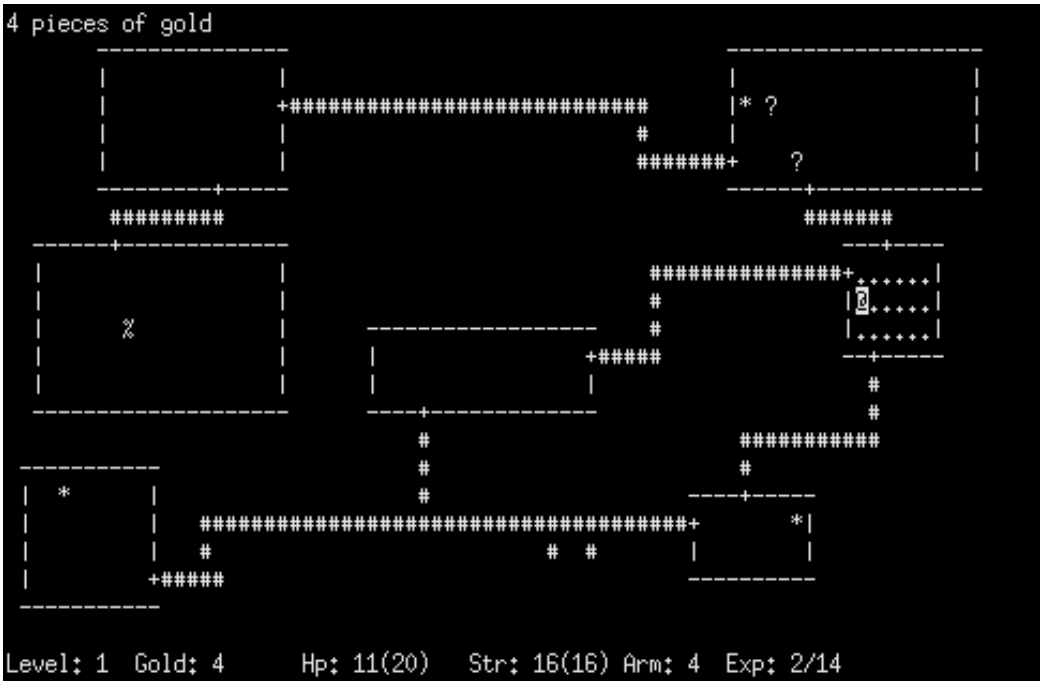

Figure 1: A screenshot of the BSD version of Rogue.[27]

Additionally, PCG allowed game developers to create these expansive, dynamic environments often at a fraction of the production and spatial cost of using pre-made handcrafted content. Not only was this essential for older, more space constrained gaming systems, but is an increasingly attractive feature of PCG as the number of people playing video games and therefore demand for new, unique, quality game content continues to rise.[25]

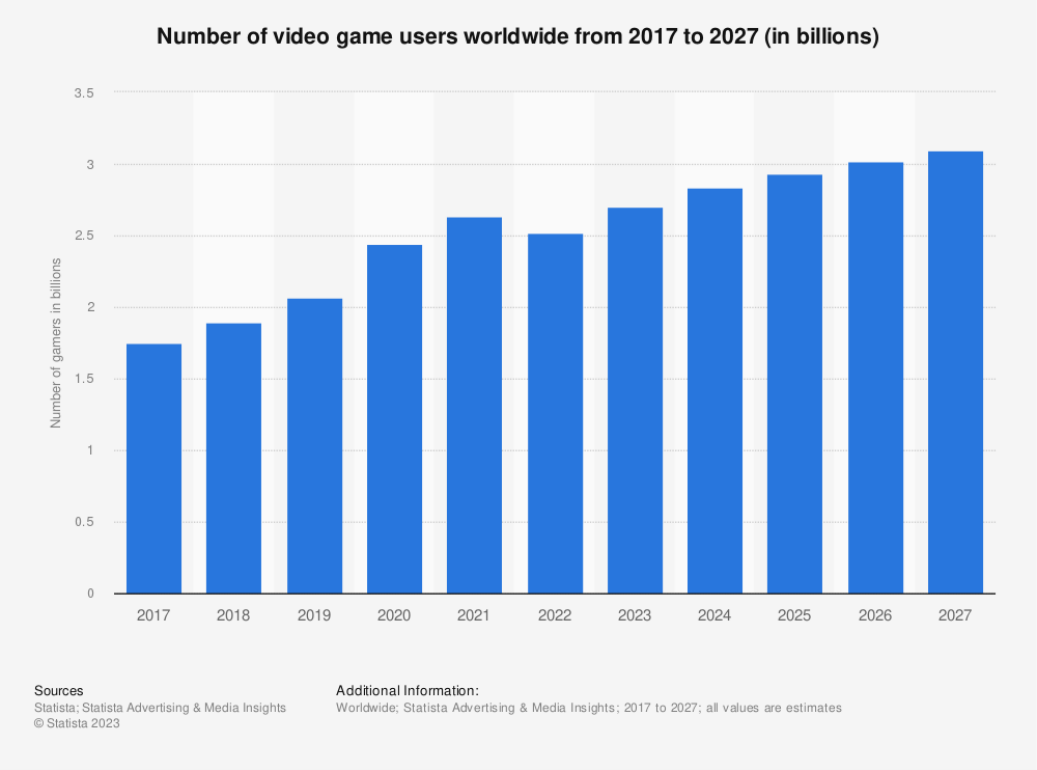

Figure 2: Number of video game users worldwide

Despite the benefits outlined above however, PCG is becoming increasingly more difficult to integrate well within newer titles. One of the main issues is that the content PCG algorithms generate are often expected to meet the ever increasing standards of more detailed and complex games. This serves as one of the primary driving factors for the increasing research being done on PCG both in academia and in industry. Hendrikx et al (2013)[7] finds that many PCG methods can be used to generate game content in layers. This suggests that there are levels of PCG which may not necessarily depend on the variety of PCG methods in use but rather how they can be applied in layers to generate increasingly complex aspects of a game, usually to add immersion. Minecraft[19] for example (along with many other notable games which make use of PCG such as Rogue[10] or Elite[11]) is able to utilise PCG to generate several layers of game content (to make a 3d voxel world) from a single seed and some given, predefined content (game bits: textures, behaviours, etc). Music Reactive Procedural Content Generation attempts to

take this a step further and derive a layer below game bits which is essentially the music fed into the system. The music will serve as the noise in which the PCG algorithms use to generate content.

Figure 3: An in-game screenshot of Minecraft Java Edition[26]

2.2 Audio Analysis

Audio analysis is a form of signal processing and refers to the process of extracting meaningful information from audio data. We, humans, perform audio analysis on a daily basis through listening to sounds surrounding us in the real world and interpreting meaning from it. Similarly, a computer system can be used to "listen" to audio signals and acquire meaning from it.

2.2.1 Amplitude Analysis

One of the properties of audio data that can be extracted is the amplitude, or how "loud" the audio signal is. There are several methods used to achieve this due to the term "loudness" being somewhat ambiguous, however, the RMS or Root Mean Square is often used as it correlates well with how we perceive loudness as humans.

RMS, as its name suggests, approximates perceived loudness by taking the root of the mean of the values squared.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

In the context of amplitude analysis this would mean firstly squaring the samples of the audio signal (discrete values representing the amplitude of the signal at different times), then adding them together and dividing that by the total amount of samples (to get the arithmetic mean) before finally taking the square root of the calculated mean. RMS works well for finding an average "power" value for oscillating signals as the peaks in the oscillations in opposite directions (positive, negative) will not cancel each other out (squared values can only be positive).

Calculating the RMS of an audio signal is also possible in real time as the incoming audio samples may be processed in the RMS calculation continuously as they arrive.

2.2.2 Stereo Analysis

Audio data often includes stereophonic sound information, that is, the audio signals make use of the left and right channels. This is often interleaved in a single stereo data buffer which can usually be split up into the mono channel components where further audio analysis may take place.

2.2.3 Fast Fourier Transform (FFT)

The revolutionary Fast Fourier Transform (FFT) algorithm originally devised by Cooley and Tukey (1965)[1] provides an efficient way of decomposing a signal into its constituent parts based on the principle of wave interference (complex waveforms can be represented as a sum of simpler sine waves). It is most often used in audio analysis to produce a frequency spectrum (denoting how much of (amplitude) and which sounds (frequencies) are present) of a discrete audio signal. This calculation can be done in real-time with modern hardware as the method exploits the predictable, and periodic nature of sine waves which can be observed from taking a DFT (Discrete Fourier Transform) of an audio signal. Using a FFT to analyse an audio signal instead of a

standard DFT reduces the number of computational steps required from $O(n^2)$ to $O(nlogn)$ where n is the size of the audio data. Since the FFT was originally proposed, it has found numerous applications in many fields of science and is heavily integrated within our daily lives.

2.2.4 Beat Detection

A beat represents a "pulse" in music. With many beats, they can represent the pace of the music (i.e) Tempo - BPM, Beats Per Minute) and the rhythmic patterns of the music (i.e) beats in a measure or segment of music). One of the numerous applications of FFTs is for beat detection. Given a frequency spectrum of an audio signal, one might be able to detect beats (and of a certain "colour" - within a range of frequencies) within it. Patin (2003)[2] realises this concept by virtually keeping track of the "average sound energy" and comparing that with the "instant sound energy" to detect "sound energy peaks" (and of specific frequency bands if specified), representing beats in the audio signal. A threshold value is also used in order to define how sensitive the algorithm is to peaks in the audio signal and which peaks correspond to a beat. One of the main issues with this approach would be that "beats" experienced by us as human listeners are not necessarily peaks in the sound energy though calibrating the threshold value and the range of frequencies in which beats are detected within may help with this.

2.2.4.1 Tempo Estimation Given that the beats of a piece of music can be detected, the tempo can be estimated in real time by dividing the number of beats by the amount of time passed. This estimate relies largely on the underlying beat detection algorithm.

2.2.5 Pitch Detection

Pitch represents the perceived frequency of a sound, in other words, how high or low that sound is as perceived by us. Pitch detection is the process of identifying or estimating pitch from a given audio signal. For real-time applications, this is a difficult problem that continues to give rise to active research as fundamentally, sounds and audio signals are often very complex. In the real-world, audio signals can contain a vast amount of frequencies, harmonics, noise, timbres, etc which makes extracting a single "note" or even fundamental frequency quite difficult. To add to this, extracting frequencies

from an audio signal (by FFT or other means) often includes some trade-off between resolution and latency which makes accurate pitch detection difficult to achieve in real-time. That being said, there are many proposed Pitch Detection Algorithms (PDAs) such as time-domain, frequency-domain, and hybrid approaches that function relatively well under specific circumstances.[3][4]

2.2.5.1 Harmonic Product Spectrum (Noll, 1969)[5] or HPS can be used to estimate the pitch, or more precisely, the fundamental frequency of an audio signal based on the principle of harmonics. Harmonics represent frequencies vibrating at positive integer multiples above a fundamental frequency. These harmonic frequencies naturally occur in the physical world (notably with acoustic instruments) due to resonance and wave interference. Based on this observation, the HPS describes the spectrum of fundamental frequencies multiplied by their harmonics. The HPS $\pi(\omega)$ is defined as:

$$\pi(\omega) = \prod_{k=1}^{K} |F(k\omega)|^2$$

where:

- ω is the fundamental frequency being considered
- K is the number of harmonics being considered
- F is the function returning the amplitude of a frequency signal

The intuition behind the extraction of the fundamental frequency with HPS is that the pitch peaks add coherently (with harmonics further emphasised by squared amplitudes) while uncorrelated parts of the spectrum do not. And so, the fundamental frequency can be estimated as the frequency corresponding to the maximum value calculated in the Harmonic Product Spectrum.

2.3 Related work

This section will provide an overview as to some of the additional existing research and products related to the concept of Music-Reactive Procedural Content Generation.

2.3.1 Background Music Reactive Games

Background Music Reactive Games (Khalid Aallouche et al., 2007)[22] are games that react to their background music. Khalid Aallouche et al. (2007) explored this concept by implementing and conducting a user study on a simple game prototype named, "Briquolo", which is essentially a spin-off of the classic arcade game "Breakout"[8] but with the background music reactive element, making use of their own proposed audio analysis methods. The proposed audio analysis methods found the Sound energy, Signal magnitude, Spectral centroid, Rolloff, Spectral flux, Drum track and Silence of music tracks fed into the system during a "Sound Processing" stage before gameplay ensued. Genre extraction was also performed with the use of MP3 ID3 tags. The results of the study showed an overwhelming preference (90%) for the prototype game in terms of entertainment amongst the participants.

2.3.2 Music visualisation

Music visualisation describes the generation of visuals based on musical audio input usually in real-time. The concept has been realised as a feature of electronic music visualisers (such as the Atari Video Music[24]) as well as media playing software. It should be noted that music visualisation methods may not necessarily perform any complex audio analysis in software and can simply be implemented as the result of the physical phenomena of playing music (i.e) in hardware). Despite this, music visualisations provide additional, visual stimuli which often adds to the entertainment of listening to music. They have evolved from displaying predefined patterns with simple amplitude analysis to displaying uniquely generated art in real time with complex audio analysis.

2.3.3 Video games

In this sub-section, the features of video games which relate to the concept of music reactive procedural content generation will be highlighted.

2.3.3.1 PaRappa the Rapper [13] is often regarded as the first true rhythm game (although technically, games had existed before which had made use of rhythm based actions such as the handheld game Simon[9]).

Figure 4: Electronic Simon game, circa 1978[28]

One of the primary reasons for this was because of how tightly connected the game play was to the game music. The game required players to engage in a call-and-response mechanic whereby players would be required to repeat a sequence of button presses in response to the game's visuals and in-time with the fixed music playing in the background. The game's user interface is intuitive with a bar displayed at the top of the screen during gameplay to show the movement of the music. The bar shows a moving icon based on the call-and-response mechanic and also includes the rhythm based patterns which the player must repeat. The player's performance can be seen based on a meter off to the side of the screen.

Figure 5: Screenshot of PaRappa the Rapper gameplay[29]

2.3.3.2 Monster Rancher [14] is an early example of a game which made use of audio related data to procedurally generate game content. The PlayStation game allowed players to insert their own CDs into the console, from which in-game monsters would be procedurally generated with certain characteristics according to the data read from the inserted CD. An interesting but clever implementation detail was that since these CDs were actually used to generate seeds, the seeds may be pre-calculated and stored by the developers beforehand so that specific CDs may produce specific seeds and generate specific, special types of enemies - making the link between the CD inserted and enemy generated much more pronounced.

2.3.3.3 iS: internal section [15] is a tube shooter which, in a certain mode, allowed players to insert their own CDs to generate game content. The gameplay comprised of the player ship, represented by some simple polygonal shapes, moving through a (visually music reactive with spinning and resizing shapes,) tube in which enemies would appear which the player would be able to shoot with certain weapons (each representing a different animal of the Chinese zodiac). Although the game was never released outside of Japan, the game was able to demonstrate the idea of procedural content generation,

presumably in real-time, based on CDs fed into the console. - Though its not clear that the game's generated content goes beyond that of the visuals of the tube.

Figure 6: Screenshot of iS: internal section gameplay

2.3.3.4 Vib-Ribbon [16] is a 2d rhythm game which further explored the PlayStation's capability to read CDs and generate game content in real-time. In Vib-Ribbon, players guide a rabbit character named "Vibri" along a line filled with a series of obstacles. Navigating obstacles required precise timing of button press events which were in sync with the background music. Vib-Ribbon allowed players to insert their own tracks (CDs) to produce unique levels with which the difficulty corresponded to the intensity of the music. As obstacles were required to be spawned in advance of Vibri's navigating them, the game made use of a lookahead system which looked

ahead in the music to find interesting features (e.g) beats) which may correspond to obstacles. The game's simplistic graphics makes very clear the connection between the game play and game music, even with player inserted music.

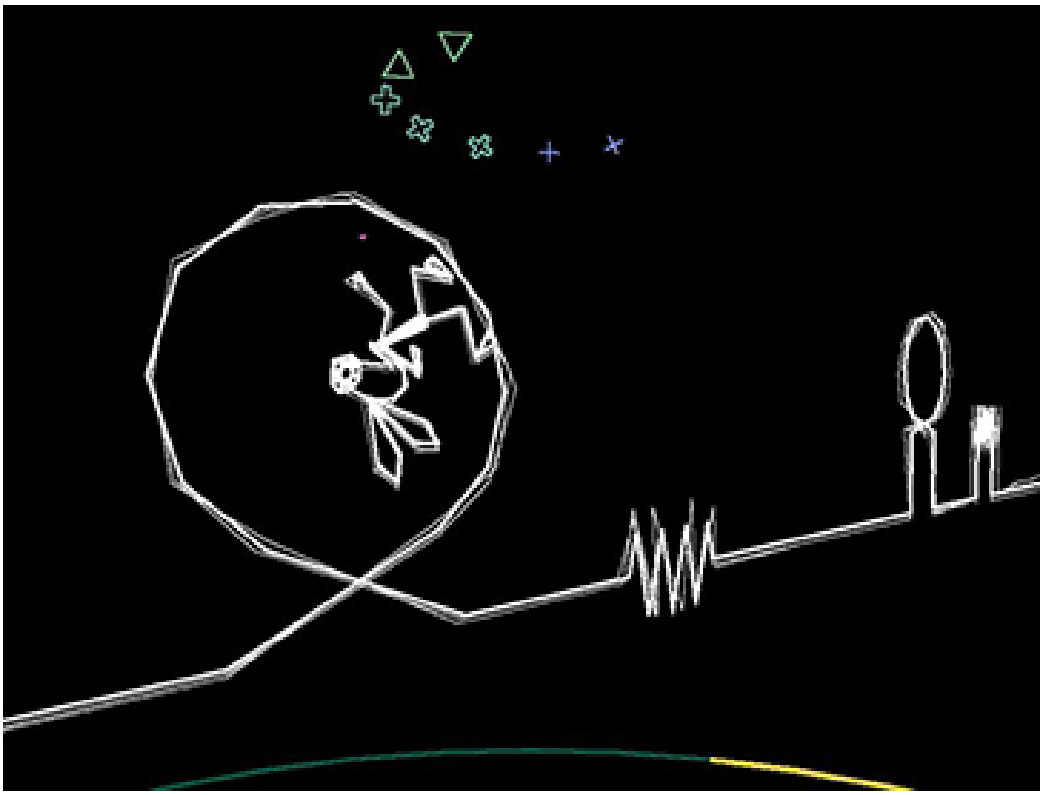

Figure 7: Screenshot of Vib-Ribbon gameplay

2.3.3.5 Rez [17] is an iconic musical rail shooter initially created with the purpose of creating a sense of synesthesia, or a full body experience for the player whilst playing the game. In Rez, players find themselves navigating a cyberspace, fighting off viruses to save an AI named Eden from sleeping forever. The game blends in elements of rhythm games and rail shooters in a 3d environment to create an immersive experience. One of the core game mechanics which exemplifies this blend is the gunplay. Whilst navigating the cyberspace, players are equipped with an auto-locking missile launcher which fires in sync with music playing in the background. As the player continues clearing layers within levels, not only does the music becomes more

intense with added elements of the synced gun shooting but the layout and enemies of each layer also change. Additionally, the game controller (and an optional "Trance Vibrator" which came with the game) provides the player with haptic feedback during gunplay to add to this idea of a "full body experience" (Auditory, Visual, Tactile).

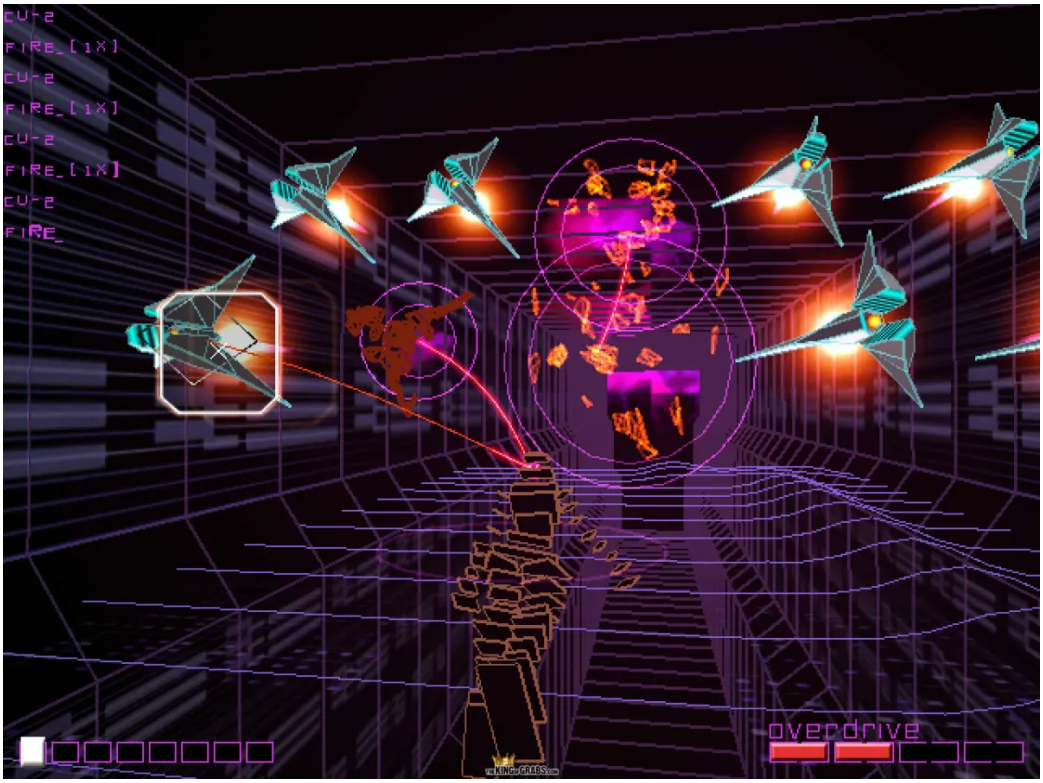

Figure 8: Screenshot of Rez gameplay[32]

Another example of how Rez creates this sense of synesthesia can be observed in how different assets within the game visually react to the music. From the background terrain to the player's health and player model, many of the game's assets are changing size, orientation, colour, etc based on the music playing in the background.

2.3.3.6 Audiosurf [18] (/Audiosurf 2 [20]) is a rhythm, puzzle, racing game. In Audiosurf, players use their music tracks to generate racing tracks within the game. As the player races along the track, they collect coloured

blocks and dodge obstacles to the beat and rhythm of the music. The layout of each track (height, colours, obstacles, blocks, route) is dynamically generated in a similar fashion as Vib-Ribbon generates their levels, based on a slight lookahead.

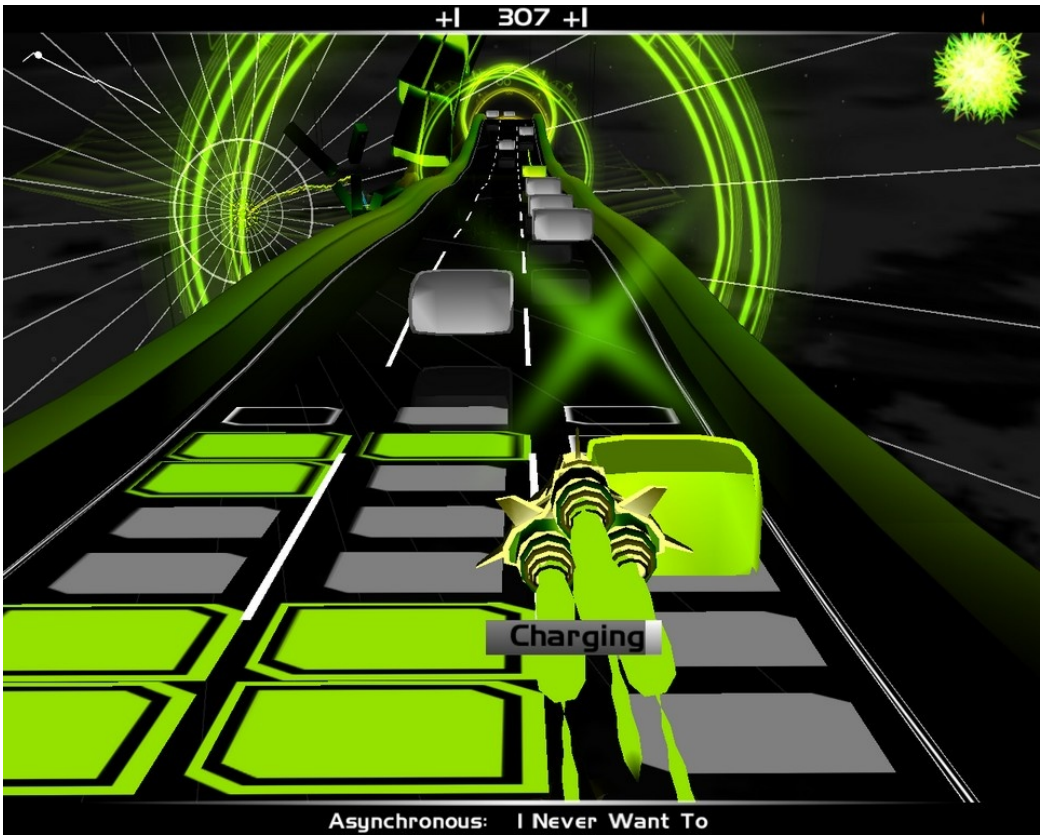

Figure 9: Screenshot of Audiosurf gameplay[33]

3 Requirements Specification

The requirements specification captures the properties the software solution must have in varying degrees of priority.

3.1 Primary

Meeting the primary requirements is essential to having a functional game that can be evaluated. The primary requirements of this project include:

- Developing a system capable of analysing music input and extracting relevant features (e.g. dynamics, pitch, tempo) dynamically.
- Designing algorithms for procedurally generating game content and the game environment (e.g. enemies, terrain) based on the extracted music features.
- Implementing a prototype game (musical rail shooter in Processing) demonstrating the integration of music-reactive procedural content generation.
- Evaluating the prototype game (and music-reactive PCG) in terms of player experience and gameplay quality.
- Contribute to the academic understanding of procedural content generation and its applications in game development.

3.2 Secondary

The secondary requirements build on the the primary requirements to provide a more well-rounded, robust solution with Music-Reactive Procedural Content Generation at its core. The secondary requirements of this project include:

- Extracting complex features of music (e.g. rhythm, melody or harmony) to use in PCG.
- Designing more complex music-reactive PCG algorithms (e.g. Shooting mechanics, Enemy models/behaviours) to dynamically generate every aspect of the game world.
- Conducting multiple rounds of evaluation for the prototype game.
- Improving the system to be robust to non-music or unexpected audio input.

3.3 Tertiary

The tertiary requirements describe additional desirable functionality the solution can include which make use of Music-Reactive Procedural Content Generation. The tertiary requirements of this project include:

- Support cross-fading playlists of tracks.
- Extracting very complex features of music (e.g. genre or timbre) to use in PCG.
- Including the microphone or system audio as an audio input to "play the environment".
- Predicting features (e.g. beats, melody, harmony) of the audio/music fed into the system dynamically to be able to generate precise timing based actions in advance for the player to act on during gameplay.

4 Software Engineering Process

4.1 Development Approach

A traditional iterative software engineering process, the iterative waterfall model, was taken to develop the software solution. The iterative waterfall model quite broadly follows these main steps: Analysis → Design → Implementation → Testing → Maintenance. Where iteration can occur at the Design, Implementation and Testing stages.

4.2 Justification

The iterative waterfall development methodology was used primarily over other development methodologies to give a better scope over the project. Especially when exploring new or niche concepts such as Music-Reactive Procedural Content Generation, a clear vision as to what needs to be implemented may be difficult to acquire. Specifying an initial set of requirements from which design, implementation and testing follows upon allows the development process to have more and better structure. In addition to this, because of the iterative component of this approach (in contrast to the rigid waterfall model), many of the benefits of other iterative development methodologies

such as Agile or Rapid Application Development approaches (often used in game development in particular) can be attained without sacrificing the structure traditional waterfall based approaches provide.

4.3 Execution

This software engineering process was realised through firstly specifying a set of objectives which define the requirements (this gives a direction - see Chapter 3. Requirements Specification) then conducting weekly meetings during the design, implementation and testing stages (where development of the software solution takes place - See Chapters 6 and 7) before documentation, evaluation and maintenance takes place (see Chapters 8 and 9).

5 Ethics

As this project will require human subjects for the evaluation of the prototype game, there are ethical concerns which must be considered. These ethical concerns are addressed by the artifact evaluation form provided in the appendix (A.1) which references the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

6 Design

This chapter gives an outline as to how the requirements specification detailed in Chapter 3 will be tackled throughout the development and implementation process (Chapter 7). The software solution, a rhythmic rail shooter named "$FL\bar{O}$", will be implemented conforming to a set of design philosophies which emphasise the overall desired functionality of the system.

6.1 Design Philosophies

The following design philosophies highlights the key functionality of the developed game, $FL\bar{O}$, to generate content procedurally through music in a reactive manner:

- Maximise **Reactivity**. The game should be able to react/respond directly to the music and audio signals fed into the system. This means minimising foresight such as reading data from the music which occurs at a later stage or using the length of a track (since ideally the system should be able to function without this in order to meet desirable requirements such as playing the environment, see Chapter 3).

- Maximise **Performance**. For real-time applications, performance is important as otherwise the system may seem unresponsive to the input. Without consideration for the performance of the application, there might not be a clear mapping between the inputs (being the music fed into the system and the player controller) and the outputs (gameplay mechanics and presentation).

- Maximise **Separation** of game data and logic. Separating the gameplay logic and data allows more fine grained control over the mechanics and presentation of the game as entirely separate entities. This is important for a system which deals with many different inputs as the discrepancies between the inputs can be minimised through careful calibration of parameters and weight values which can be accessed more easily with the separation of game data and logic.

Maximising both the reactivity and performance of the system can make the experience of playing the game and listening to the music less separate and more intertwined which is one of the main objectives of Music-Reactive Procedural Content Generation. Maximising the separation of game data and logic can help to highlight this feature.

6.2 Game Design

$FL\bar{O}$ is a three-dimensional, rhythmic, first person, rail shooter that reacts to music in real-time.

- **3D**: $FL\bar{O}$ makes use of 3D graphics providing the player with a more realistic and therefore immersive gaming experience.

- **Rhythmic**: The music-reactive aspect of the game will help the player feel more engaged with the game as multiple sensory inputs are triggered to the music being played in the background (auditory, visual), creating

an artificial sense of synesthesia and a mapping between the music and the gameplay. Throughout this chapter, these music-reactive elements will be highlighted.

- **First Person**: The player plays through the eyes and perspective of the character within the game. This pushes the narrative that the player is the character further enhancing the game's immersion.
- **Rail Shooter**: $FL\bar{O}$ is a Rail Shooter meaning the game's design focuses on a continuous forward movement along a predefined path, challenging players to react swiftly to oncoming obstacles and enemies. The movement of the music is represented by the path and the player's movement along it. This feature accommodates the music-reactive elements of the game as it establishes a clear mapping between the music being played and the game play experience, experienced by the player.

There are two main considerations for the design of the game. This includes how the game works (Game Mechanics) and how the game is presented (Game Presentation). Together, these broad categories define the how the game plays.

6.2.1 Game Mechanics

This section will extensively explore the mechanics featured in the game $FL\bar{O}$.

6.2.1.1 Audio Analysis Many of the mechanics in $FL\bar{O}$ are music-reactive meaning they utilise some component of audio analysis being performed on the song/music being played. This includes many of the analysis techniques detailed within the Context Survey conducted (in Chapter 2) such as amplitude analysis, beat detection, pitch detection, etc. The specific implementation of these audio analysis methods is described within the implementation chapter (Chapter 7).

6.2.1.2 Physics : $FL\bar{O}$ makes use of a simple physics engine which provides opportunities for additional game mechanics to be implemented such as **music-reactive movement** (for player and enemies) and **music-reactive shooting** (of weapons). The engine should function based on particles

abiding by Newton's laws of motion (Newtonian mechanics). Particles are an abstraction of physical bodies found in the real world though are typically point masses with little to no mass and are not subject to the mechanics of angular motion as physical bodies are. That being said, angular motion can be approximated inexpensively through rotational kinematics, i.e) Giving particles an orientation which may be updated, and basic shapes such as spheres of certain sizes may be able to represent them as more than point masses for interactions. The physics engine implemented within $FL\bar{O}$ will be able to simulate both linear and angular motion for particles in these ways within the confines of a three-dimensional space.

6.2.1.3 Player : Players play through an immersive first person perspective with keyboard inputs controlling player movement ("W" - Forward, "A" - Left, "S" - Backward, "D" - Right, Space - Jump) and the mouse being used for aiming and shooting. Player movements are emphasised based on the beats detected in the music being played. This can make the player "feel" the music to a greater extent as they move around since each detected beat will produce larger forces acting on them. The player may also time their movements strategically to evade incoming enemies and obstacles. - **Music-Reactive movement**.

Players are given (a maximum of) three hit/health points which can be replenished upon eliminating certain environmental hazards. Players lose when they lose all of their health, and win if they complete the stage with at least one health. In order to account for the unpredictable nature of the input (music), players will be invincible for a short period of time after receiving damage so that players do not lose all their health instantaneously. With music that produces levels of different difficulties (see 6.2.1.4 Environment) the rate at which the player is likely to lose/replenish health changes. This helps to solidify the idea that the difficulty of the game depends on the music being played - **Music-Reactive difficulty**.

One noteworthy design decision here is that although there are many music-reactive mechanics built within the game and the player controller, players should still feel as if they are in control of the character. As an example, beats were used to emphasise movement rather than the amplitude of the music. Additionally, the maximum amount of health the player has is fixed. This way, players

have a sense of control over their character and a better understanding of the mechanics but the gameplay is still affected by the music.

6.2.1.4 Weapons : Weapons, specifically guns, and shooting mechanics are essential to $FL\bar{O}$ as a shooter game.

In $FL\bar{O}$, the player shooting mechanics are affected by both the beats and amplitude of the music being played. The amount of damage weapons do and recoil they have are based upon the weapon's rate of fire and the amplitude of the song. In addition, if the player fires "on beat" to the music being played the amount of damage dealt by the weapon triples. - **Music-reactive shooting**

New weapons can also be unlocked and equipped in $FL\bar{O}$. These weapon rewards are generated as the player plays through different levels (songs) in the game and can only be unlocked upon successful completion of these levels. Weapon rewards are generated via "growing" each time a beat in the music is detected during gameplay. This growth step updates certain properties of weapon rewards such as it's shape, size, handedness (where on the screen the gun appears during gameplay), wear rating and rate of fire. How these weapons grow is based upon the player's performance and the real-time audio analyses conducted on the music playing in the background during gameplay such as its amplitude at different frequency ranges, amplitude of each audio channel (left/right), and tempo. - **Music-reactive weapon generation**

6.2.1.5 Environment : The environment in which the player navigates through in $FL\bar{O}$ is one filled with obstacles and enemies.

The terrain itself acts as an obstacle to the player with spikes and walls of various sizes corresponding to the music's waveforms and frequency spectrum (produced from conducting FFT analysis). These spikes and walls give the terrain a somewhat natural textured appearance as waveforms and frequency spectrums can include alot of noise. With less noise, the surface of the terrain may appear more rounded (shape of sine waves) and the walls may appear thinner (with specific frequencies having large amplitudes) which is still interesting (more so than a flat surface). As the player travels along a fixed path with continuous forward movement, the terrain that generates to

the music in real time will also reflect the music generated terrain (waveforms and frequency spectrum) from moments ago since the terrain is continuously moving towards the player in a sense. (The path itself represents the music and the time that has passed.) - **Music-Reactive Terrain Generation**. Additionally the terrain is split into two sides, left and right, which correspond to the waveforms found in the left and right sound channel's respectively. This gives a clear representation for the left and right sound channels within the game.

Other obstacles which lie in the path of the player include Rocks, Trees and Worms.

- **Rocks** are simple floating obstacles which the player will need to evade or destroy as they moves towards them (being generated on the path). Each rock generated will correspond to a specific frequency range. The health and size at which rocks are generated to be depends on the amplitude of the frequencies within its frequency range at the time of generation. Outwith the time of generation, the amplitude of the frequencies within the rock's frequency range will determine whether it "glows" or not. In this "glowing" state, rocks can be destroyed by the player to gain (one) health. - **Music-Reactive Rock Generation**. Note that shooting rocks will push them away from the player, this rewards players to time their shots to get destroy rocks quickly.

- **Trees** in $FL\bar{O}$ are obstacles which grow based on the beats of the music. Similar to rocks they are generated on the path the player is moving on and as such the player must evade or destroy them to avoid taking damage. The amount that a tree's branches spread upon growth depends on the intensity of the beat which prompted its growth. No health can be gained from destroying trees. - **Music-Reactive Tree Generation**

- **Worms**. Each worm is a set of five moving obstacles which move together to form a worm-like creature with worm-like movement. Worms "wriggle" with greater speed and size (over longer distances) based upon the amplitude of the music - **Music-Reactive Movement**. This gives them the appearance that they are "dancing" or more generally "wriggling" to the music. They try to target the player by wriggling in front of and blocking the them. Like rocks and trees, players must

avoid colliding with worms either by evading or destroying them in order to avoid taking damage. The size of worms however depends on the amplitude (volume) of the music being played in real-time making some worms harmless (too small) at times, this further illustrates the idea of **Music-Reactive Difficulty** as different songs will produce differently sized worms. Worms spawn on beats with health dependent on the amplitude of such beats. They come in three varieties depending on the amplitude of the song at the time of spawning:

1. Earth. Earth worms spawn if there are more "lows" in the music being played at the time. These worms have legs and the ability to traverse the generated terrain.
2. Water. Water worms spawn if there are more "mids" in the music being played at the time. These are basic worms that have the ability to wriggle fast. They do not wriggle across the terrain and simply spawn at the height of the player.
3. Air. Air worms spawn if there are more "highs" in the music being played at the time. These worms have wings which give them the ability to jump, glide and fly on beats of the music being played.

Note: Lows, Mids and Highs correspond to different frequency ranges in the music. Intuitively, low frequencies, middle frequencies and high frequencies. The scoring for each of these depends on the amplitude and the decay of these frequencies (See Chapter 7. Implementation).

Since worms are a set of moving obstacles, players can shoot and evade parts of the worm individually. However, unless the player destroys the head of these worms (the frontmost part/obstacle), these worms will regenerate their lost limbs/parts on beat with the music. - **Music-Reactive Worms**

6.2.1.6 Scoring : The scoring system reflects the player's performance in terms of: Distance travelled, Enemies killed, Player Health and Shooting accuracy. As the environment is generated based on the music being played, keeping the scoring system consistent for all levels makes this distinction in level difficulty clear.

6.2.2 Game Presentation

This section will focus on the design behind how the game $FL\bar{O}$ is presented visually and auditory.

6.2.2.1 Colours : One of the most vibrant ways music is represented in $FL\bar{O}$ is through the use of colour. Generally, the game will make use of the colour red for low frequencies, green for medium frequencies and blue for high frequencies. This gives each level a unique appearance depending on the type of music being played. Additionally, the overall amplitude of the music being played correlates to how bright these colours are in being displayed. - **Music-Reactive Colours**

6.2.2.2 Menu : On starting the game $FL\bar{O}$ players are greeted with a Title screen where they can preview, browse and select their weapon of choice and stage(/level/song) to play before entering the game.

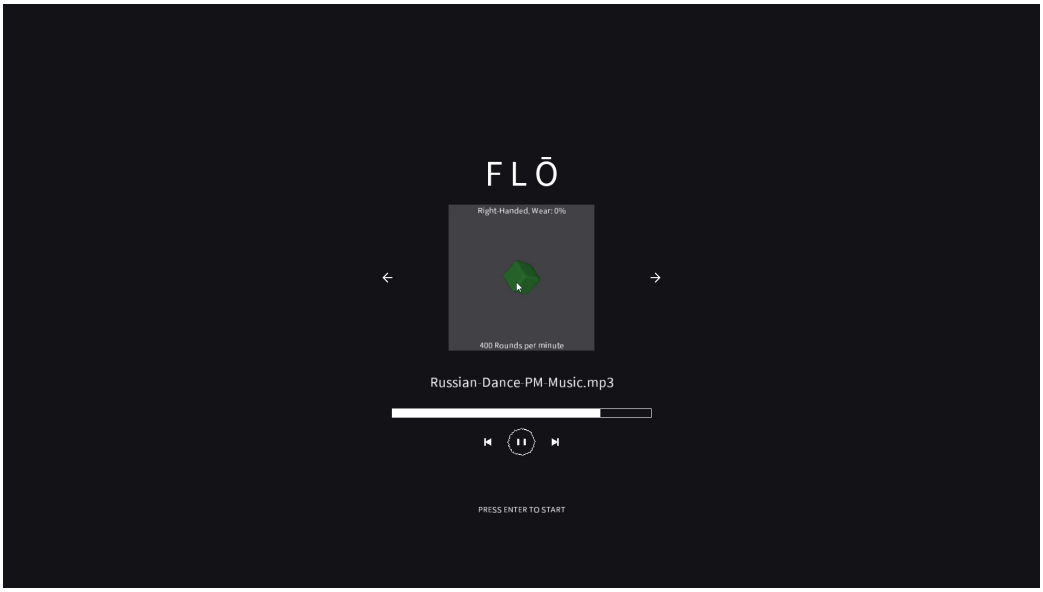

Figure 10: $FL\bar{O}$ Title Screen

Once the player dies or survives a stage they are brought to either a victory or defeat screen based on the outcome of their in-game performance (Whether they survived or not). Both the victory and defeat screens will show the

player their final score, out of the maximum score possible and allow the player to preview the gun they may or may not have unlocked (which depends on whether they won or lost). The defeat screen will additionally specify what the player died to, to lose the stage.

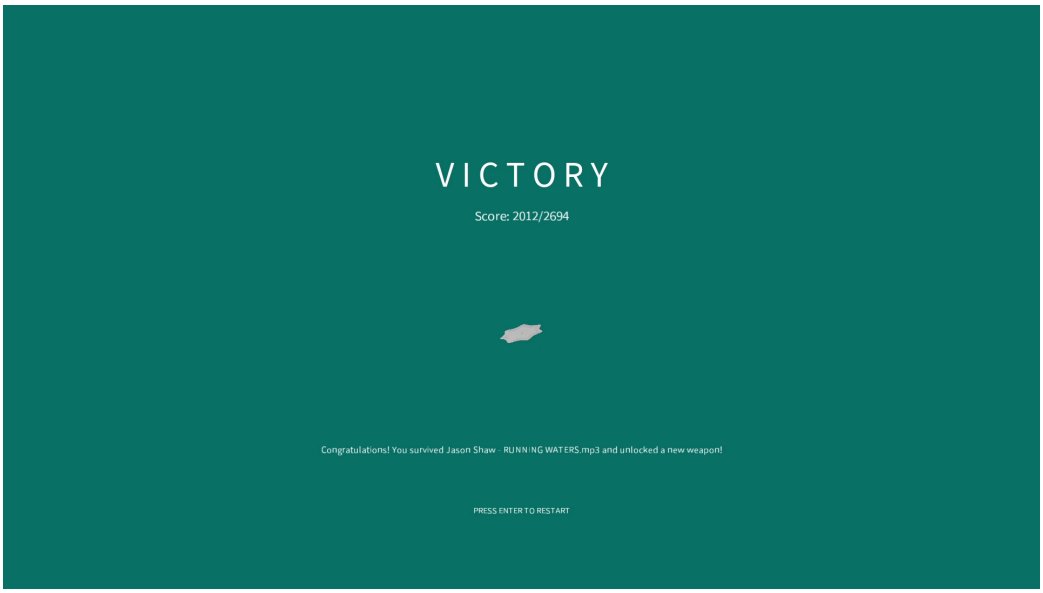

Figure 11: *FLŌ* Victory Screen

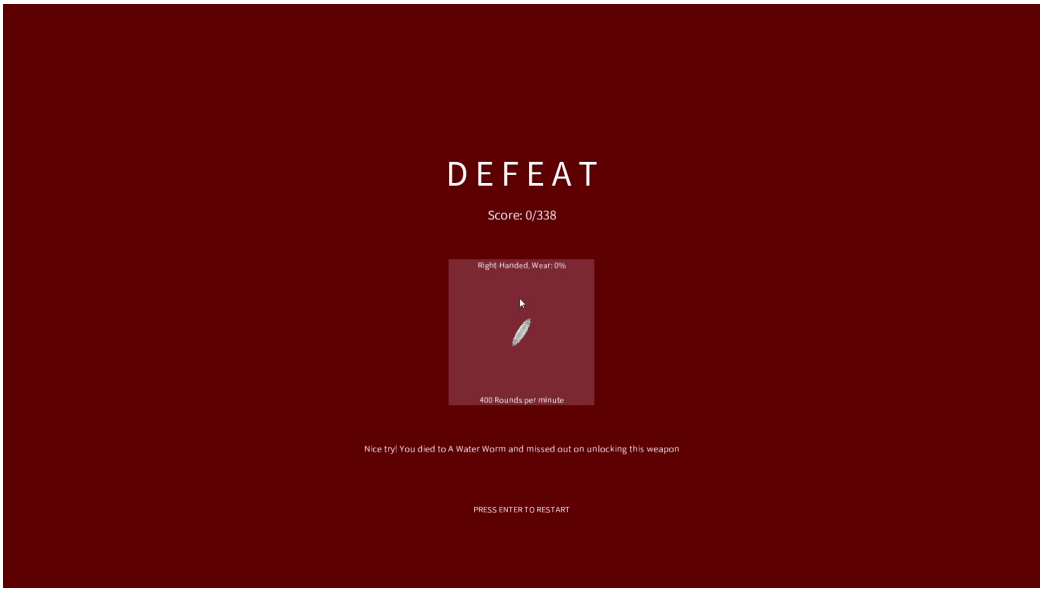

Figure 12: $FLŌ$ Defeat Screen

6.2.2.3 HUD : The in-game HUD (Heads-up display) displays the player's score at the top of the screen, along with a circular **music-reactive crosshair** (which takes the waveform of the music being played). This crosshair also provides information through hitmarkers that give the player feedback such as whether they hit something, how much damage they have dealt, and whether their shots were timed (on beat, critical hits).
Just below the crosshair, the player's health is represented as spinning heart shaped objects. These hearts will "beat" (increase in size) with the beats of the music. - **Music-Reactive health indicator**.
At the bottom of the screen, the control scheme for the game is presented. (Note that this was added in the second version of the game upon player feedback.) This makes it easier for players to understand the controls and also provides an opportunity to visualise beats more clearly as the controls light up for a brief moment upon every beat. This control interface additionally allows the player to see when their actions are timed with the beat of the music as a "golden colour" highlights when and which actions are successfully timed. - **Music-Reactive controls indicator**.

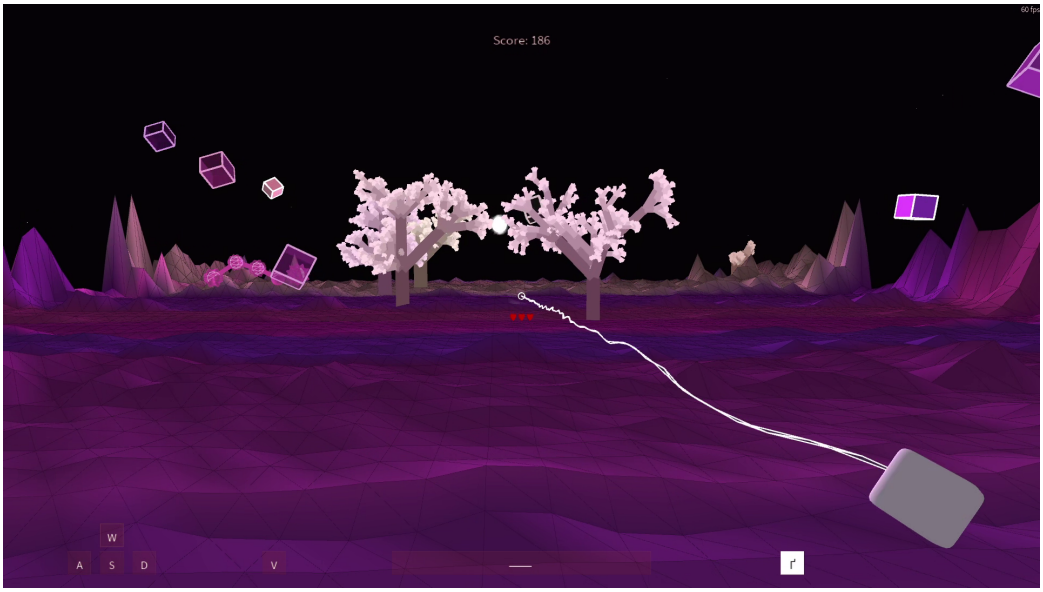

Figure 13: *FLŌ* Gameplay - Shooting Land

Figure 14: *FLŌ* Gameplay - Timed Jump

Figure 15: $FL\bar{O}$ Gameplay - Critical Hit

6.2.2.4 Weapons : Weapons in $FL\bar{O}$ come in a variety of forms with various shapes, sizes, handedness (where on the screen the gun appears during gameplay), wear ratings and rates of fire (See 6.2.1.4 Weapons). Most of these properties of an equipped weapon are fixed during gameplay and are not permanently modified as weapon rewards (that "grow" during gameplay) are. However, the size, rotation and colour of equipped weapons do change during gameplay (and when being previewed) in response to the music's amplitude and amplitude at different frequencies respectively.- **Music-Reactive Gun visuals**
When shooting weapons, not only do they glow white, a bullet tracer is also displayed on the player's screen. This is represented by the visual representation of the music's current waveform. Based upon whether the player's bullet has hit something (terrain, enemies, sun/moon, rocks, trees) or not, this tracer is either highlighted in white or given more of a muted transparent black colour. - **Music-Reactive bullet tracers**. Note that if the player does hit something, they are given feedback based on that as the object hit will glow white for a brief period.

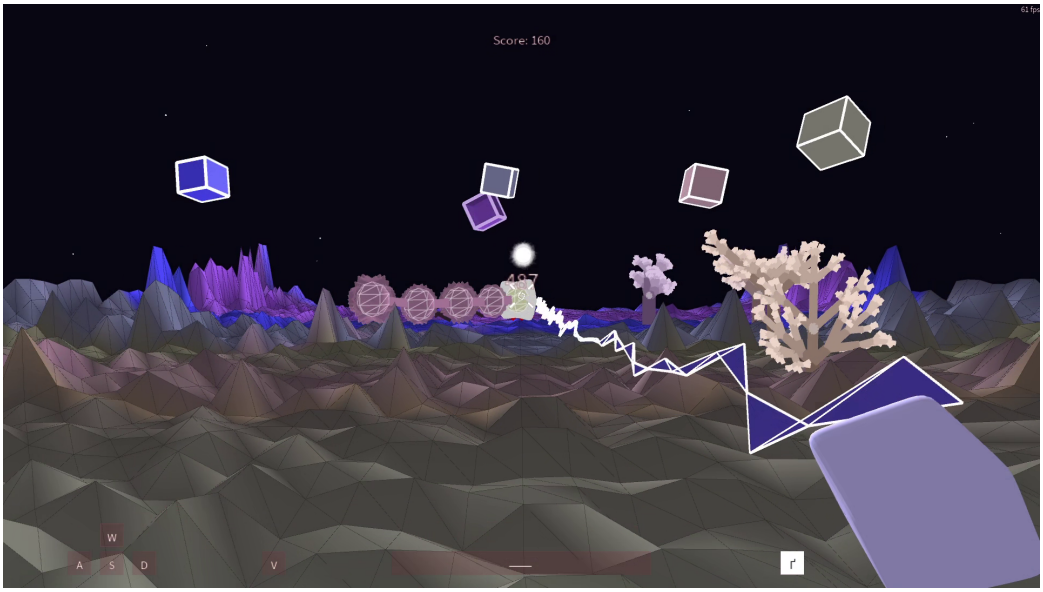

Figure 16: *FLŌ* Gameplay - Hit

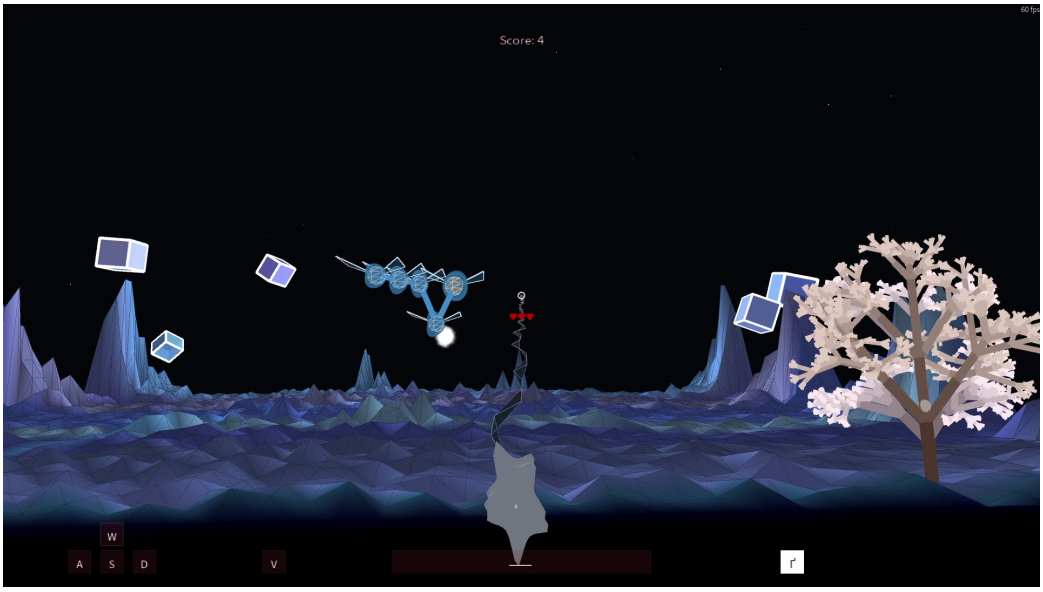

Figure 17: *FLŌ* Gameplay - Miss

In addition to the colours that are produced when shooting weapons, the

sound effects produced by a firing weapon changes based upon the music being played and whether the player has hit something. All shots will produce a tone (fuzzy triangle wave sound) which attempts to replicate the current pitch of the song. Shots that hit something will additionally play a neutral high white noise sound effect. - **Music-Reactive Weapon sounds**

6.2.2.5 Player : Since the game takes place in a first person perspective, there is no character model which represents the player. However several aspects of the player can be represented visually in other ways.
The player's health is not only represented by the spinning hearts under the crosshair but also as a colour filter over the screen and audio filter placed on the playing music. The lower health a player is, the more the screen is tinted with a red colour representing blood or danger. A low pass filter that becomes increasingly potent is also in effect as the player loses health, this both makes the level easier (as the overall amplitude of the song decreases) and has the additional effect of making the colours of the map more red as red corresponds to the lower frequency amplitudes of the song being played. - **Music-Reactive difficulty**

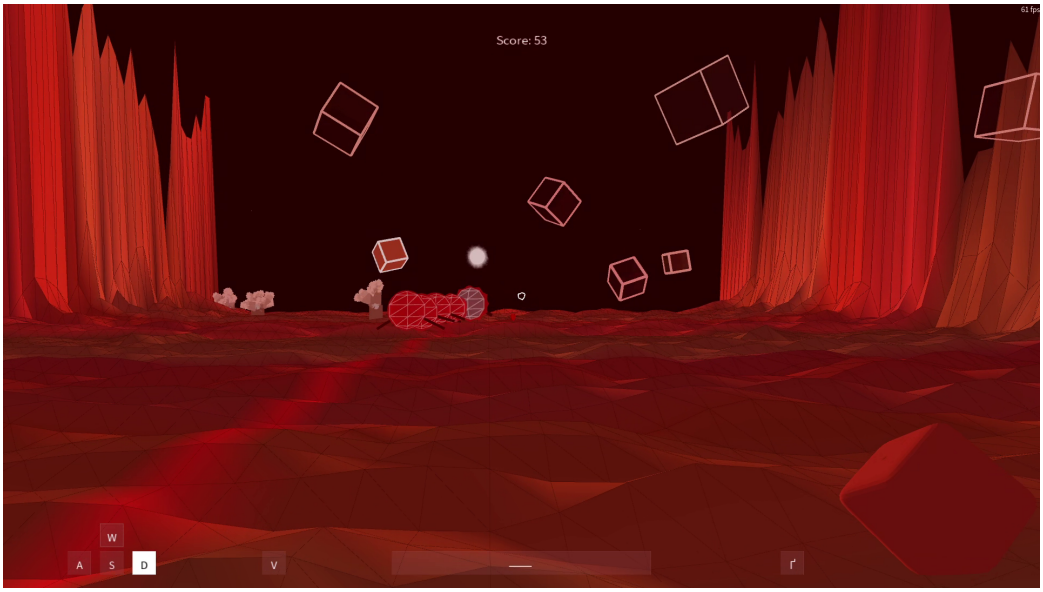

Figure 18: *FLŌ* Gameplay - Low HP

The player can also apply a low pass filter and high pass filter of their own by slowing down (going backwards) or speeding up (going forwards) respectively making the game easier or harder.

6.2.2.6 Environment : The environment is filled with visual cues which represent the music being played in the background.
The environment takes on colours as described earlier in this section, red for low frequencies, green for medium frequencies and blue for high frequencies with the overall amplitude of the music being how bright these colours are in general. **Music-Reactive Environment visuals**
One especially important visual indicator is whether rocks are "glowing" or not as destroying glowing rocks can heal the player. These special glowing rocks spin faster than regular rocks and have a brighter colour with a white outline. **Music-Reactive Rock visuals**

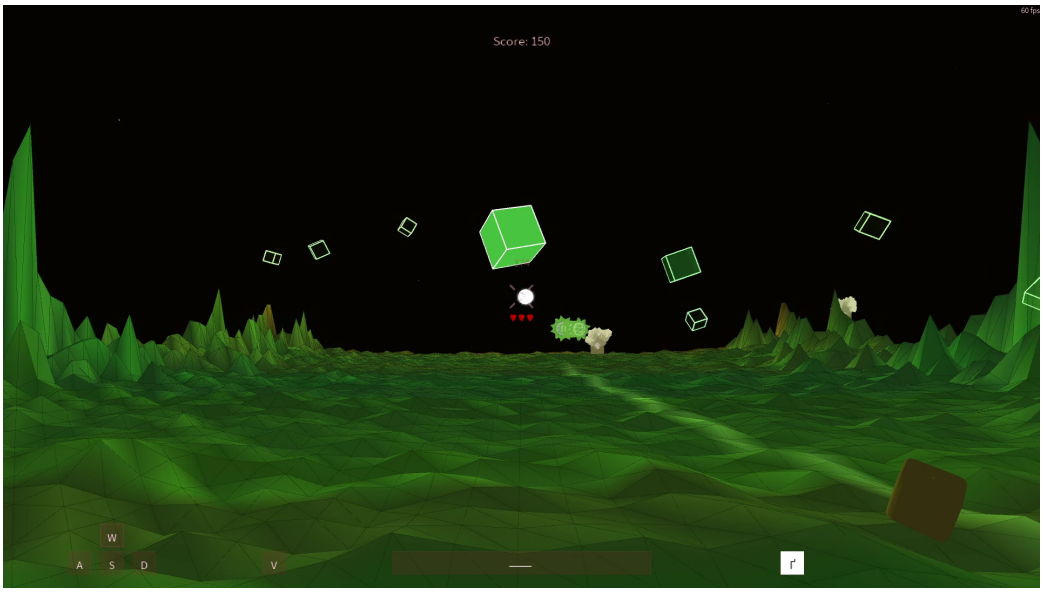

Figure 19: $FL\bar{O}$ Gameplay - Glowing rock

Trees are represented as a series of branching lines that get brighter as they get closer to the ends of the tree (from the base). Additionally, each tree's hitbox is represented as a small bright sphere. **Music-Reactive Tree visuals**
Enemies are displayed as a series of spherical shapes (generated based upon

the the amplitude and tempo of the music upon spawning) linked together by a line. They take on the colour of the environment they spawn within (which is determined by the amplitude of different frequency ranges) and may have legs or wings depending on their type. These legs and wings are procedurally animated which makes the worms seem more realistic. In order to give the player a better sense of the hitboxes of these worms, each sphere encapsulates a sphere representing the worm's hitbox with the head of the worm's hitbox being the complimentary colour of the music to stand out (Added in version 2.0). - **Music-Reactive Enemy visuals**

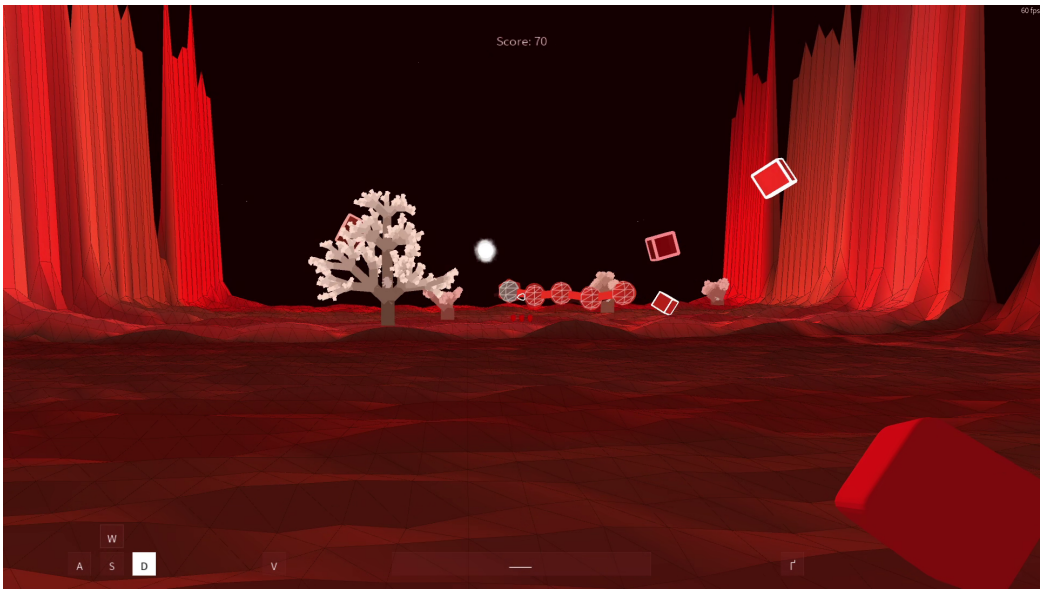

Figure 20: $FL\bar{O}$ Gameplay - Earth worm

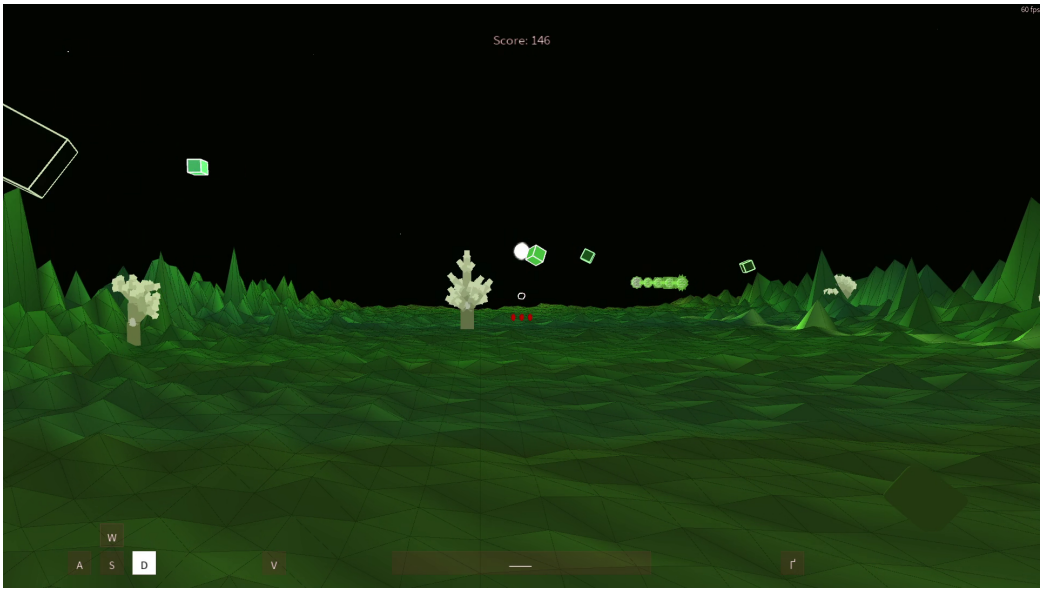

Figure 21: *FLŌ* Gameplay - Water worm

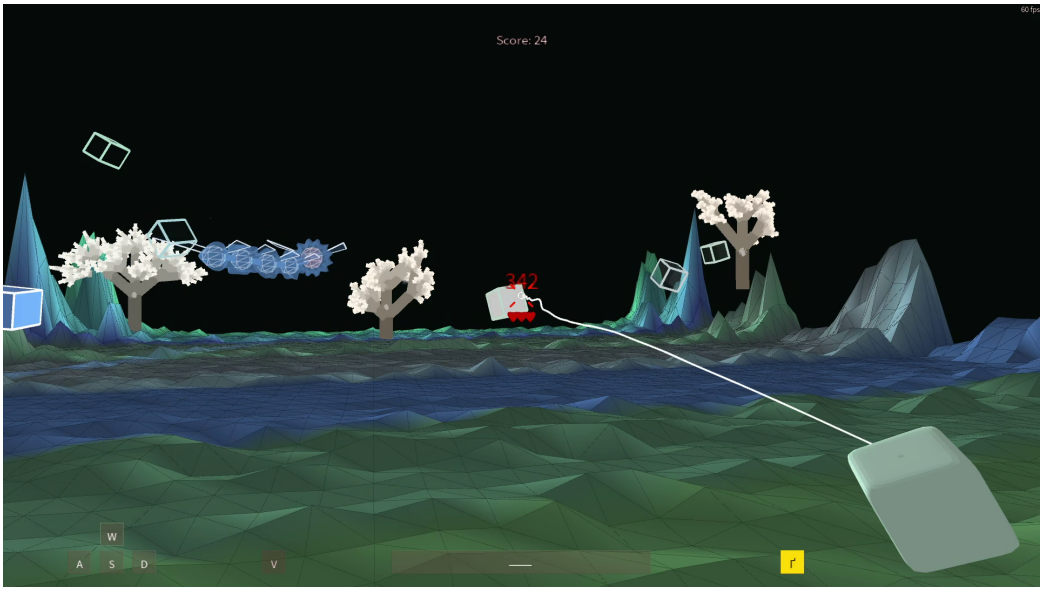

Figure 22: *FLŌ* Gameplay - Flying worm

As previously mentioned, the size of these worms are determined based upon

the amplitude of the song. There are also visual details in the environment that are music-reactive but serve little to no purpose in the gameplay such as the stars and the sun/moon. The stars are similar to rocks in that each star in the sky represents the amplitude of different frequency ranges meaning some stars glow more than others based on the music content. - **Music-Reactive stars**. The sun/moon is displayed as a sphere which gains "rays" and brightness depending on the amplitude of the music being played. The idea is that for quieter songs this sphere will appear more like the moon and for louder songs, more like the sun. - **Music-Reactive sun/moon**

7 Implementation

In this chapter, the particularly interesting parts of the game's implementation (based on the design detailed in the previous chapter) will be described. The game was implemented using the open-source programming language, Processing (lightweight graphics library built on top of Java) and the Minim audio library.

7.1 Physics

At the core of $FL\bar{O}$'s physics engine are particles with vectors describing an object's position, velocity and acceleration which each get recalculated every frame based on Newtonian Mechanics (Acceleration accumulates (before integration) $\rightarrow$ integrate for velocity $\rightarrow$ integrate for position). These particles are an abstract representation of physical entities in the game world. They act as point masses with additional properties giving them a size and orientation (in three different axes). This provides the opportunity for collisions to occur between them (as they have size) and for the mechanics of angular motion to be simulated (similar to how linear motion is described above as Newtonian mechanics). Applying linear/angular forces to these particles can simply be achieved by adding to the particles acceleration in the respective axes ($F = MA$). Note that angular motion and torque is only simulated with this approach as important variables such as the point of contact of forces is not considered.

The physics engine (along with many other mechanics) built into $FL\bar{O}$ is robust to variable frame rate. This is achieved by taking time, or rather the

change in time into consideration ($1second/framerate$) during the integration steps (each frame). To add more realism, a damping factor is also applied to simulate the loss of acceleration over time as often experienced in real life (due to friction, air resistance, etc).

Most of the entities in the game world are represented as particles with a certain size and mass. Subjecting an entity to the force of gravity is as simple as applying a downwards force to the entity's acceleration. Similarly, making an entity "jump" is as simple as applying an upwards force to the entity's acceleration. Some of the more interesting uses of this physics engine include:

- Allowing players to run smoothly across the terrain by simulating an upwards buoyancy force (opposite to gravity) using Archimedes' principle ($F = \rho \cdot V \cdot g$). Where ρ is the density of the fluid (approximately 1 for water), V is the amount of fluid displaced by the player (proportional to the distance of the player from the height of the terrain they are "floating" on) and g being the acceleration due to gravity.
- Allowing flying worms to glide by reducing the downwards force of gravity applied to it.

7.2 Optimisations

One noteworthy optimisation that was made relating to the physics engine implemented within $FL\bar{O}$ would be that all of these particles are essentially represented as "spheres". This makes it much simpler to perform many collisions checks between objects (simply the distance between their positions taking into account their sizes) and rays (casted when the player fire their weapon.)

Ray-casting was used to detect the player bullet collisions (Hit-scan weapons). Not only does this increase the performance of the application (since there is no need for bullet objects to be updated every frame). But it also makes the shooting and gunplay feel more robust and reliable as bullets are not subject to frame-rate dependent errors (think bullets travelling through objects between frames). In order to detect bullet collisions with worms, rocks, trees, and the sun/moon, a ray-sphere intersection calculation takes place. For bullet collision with the terrain, a ray-plane intersection calculation takes place

(Convincing enough approximation which boosts performance).

Another example of a performance related optimisation made without sacrificing the quality of the application would be the use of bilinear interpolation to calculate the height of the terrain at different points where the vertices of the terrain do not lie. Since the terrain is a height map with discrete values indicating heights at specific points of the land, movement without interpolation across this terrain can look "choppy". The simple solution would be to add more vertices such however this comes at a great performance cost, bilinear interpolation with less vertices seemed to improve the performance of the application as an alternative method.

7.3 Audio Analysis

Central to the feature of music reactive procedural content generation is the real-time audio analysis conducted within $FL\bar{O}$.

7.3.1 Amplitude analysis

Two different methods of amplitude analysis are used in $FL\bar{O}$. The most straightforward being the RMS value (as detailed in the context survey, Chapter 2) of the music at any given time. This was used in the calculations for the amplitude of the music (of left, right and mixed channels) when growing the weapon reward. The main downside of this method would be that it does not produce fine grained results for amplitudes at different frequency ranges which may be more useful when there is more than one game variable that can be modified according to the music e.g) Colours are made up of four values, Red, Green, Blue and Alpha. The other method of calculating amplitude of the music being played is with the Fast Fourier Transform as described below.

7.3.2 Fast Fourier Transform

The Fast Fourier Transform was primarily used to keep running "scores" of the amplitude of the song at different frequency ranges over time.

7.3.2.1 Amplitude of frequency ranges : In $FL\bar{O}$, there are three main scores that are used throughout including lows, mids and highs corresponding

to the amplitudes of low, medium and high frequencies. The boundaries for these frequency ranges were found empirically, based on observation from play testing and are defined roughly as 0hz $\rightarrow$ 200hz for lows, 200hz $\rightarrow$ 2000hz for mids, and 2000hz $\rightarrow$ 20000hz for highs.

Since many of the mechanics of $FL\bar{O}$ rely on the average amplitude level of the song being played at these different frequency ranges not being too high or low, rather than directly using these current amplitude levels for the scores, they are instead *added* to the scores which decay as time passes. The rate at which these scores decay can then be made independent from one another and can change based upon the currently calculated scores. This means that, for example, quieter songs can still produce acceptable "score" values despite the amplitude of the song being low and louder songs can still produce acceptable "score" values despite the amplitude of the song being high. Additionally, these scores can be calibrated with more precision relative to each other. Since FFT resolution is worse for lower frequencies, these lower frequency amplitude scores can be scaled to be more in-line with the higher frequency scores. With more acceptable, normalised "score" values, the game is more robust to the range of possible different inputs (different songs) and gameplay will still function as expected.

The FFT was also used in other areas to calculate the amplitude of other specific frequency ranges such as for rocks and stars.

7.3.2.2 Overall amplitude level : Rather than calculating amplitude of the music being played as the RMS value, the overall amplitude level can now also be calculated as the sum of the amplitude scores for low, medium and high frequencies. This provides a more stable amplitude level than the RMS value as it decays over time. Additionally, the amount of each component (lows, mids, highs) that make up the overall amplitude level can be specified which may be useful in emphasising particular features of the music playing (e.g) Emphasising the highs).

7.3.3 Stereo Analysis

The analyses of different audio channels is simply achieved by extracting information from the appropriate buffers (left, right, mixed). From this

information, further audio analysis was performed such as FFT for terrain generation or RMS amplitude analysis for weapon generation.

7.3.4 Beat Detection

Beat detection was achieved by comparing the "instant sound energy" with the "average sound energy". If this instant sound energy surpasses a certain threshold defined in part by the average sound energy, a beat is detected. As this beat calculation occurs on a frame by frame basis, it can be difficult for the player to execute beat specific mechanics. For this reason, when a beat is detected, a timer is initiates which gives players around 250 milliseconds to execute any beat specific mechanics.
Beat detection can also be used at certain frequency ranges to detect different "kinds" of beats such as a kick, snare or hat. This is used in $FL\bar{O}$ also for tree generation and growth.

7.3.4.1 Tempo Estimation : From beat detection, the tempo can simply be estimated as the number of beats divided by the time that has passed. (60000*beats)/millis().

7.3.5 Pitch Detection

To estimate the pitch of the song being played in real-time, the harmonic product spectrum was calculated, using the frequency spectrum produced by the Fast Fourier Transform performed each frame. This is achieved by multiplying the frequency spectrum with down-sampled versions of it (in factors of two). Once the HPS has been calculated, the pitch is simply the (fundamental) frequency corresponding to the largest value within the HPS. A few modifications were made to this method of pitch detection to make it more accurate:

1. Since the frequency resolution returned by the FFT calculation can be quite low for lower frequencies (below 2000hz), the estimated pitch is interpolated against nearby pitches with close scores.

2. Once the final estimated pitch has been calculated, this pitch is then tuned to the nearest musical note found in musical notation (A tuned to 440hz).

7.3.6 Waveform extraction

The waveforms of the music were extracted in real-time through querying the audio buffers (mix, left, right) for the samples of the audio.

7.3.7 Playing the environment

One of the most interesting features included with $FL\bar{O}$ is that the player is able to use their microphone audio as additional input to the "music" of the game. This is achieved by taking the audio samples of the microphone input and applying the same audio analysis techniques. Since all of the audio analyses happen in real-time, the fact that there is no "song or music length" does not pose as an issue.

7.4 Procedural Content Generation

7.4.1 Visuals

7.4.1.1 Colours : The vibrant colours of $FL\bar{O}$ are generated mainly by associating the amplitude scores (as detailed above) with certain hues. Firstly the running scores for low, medium and high frequencies are normalised to eachother. Then, they are each multiplied by the same value to determine the brightness of the colour. This multiplicative factor is often in correlation with the overall score (the sum of all scores, lows, mids, highs) however in some cases a higher value is used in order to make a colour more vibrant.

7.4.2 Shape Generation

$FL\bar{O}$ makes use of simple shapes such as cubes and spheres but can additionally specify shapes such as hearts and completely randomly generated shapes for weapons without the use of pre-existing models or 3d assets. This is achieved through the use of Supershapes and the Superformula[34] (extended into the third dimension[35]). The Superformula extended to the third dimension are defined through the following equations:

$$r(\phi) = \left[\left| \frac{1}{a} \cos(m\,\phi/4) \right|^{n2} + \left| \frac{1}{b} \sin(m\,\phi/4) \right|^{n3} \right]^{-1/n1}$$

Figure 23: Definition of Supershape in two dimensions[35]

$$x = r1(\theta)\cos(\theta)\, r2(\varphi)\cos(\varphi)$$
$$y = r1(\theta)\sin(\theta)\, r2(\varphi)\cos(\varphi)$$
$$z = r2(\varphi)\sin(\varphi)$$

$-\pi/2 \le \varphi \le \pi/2$ eg: latitude

$-\pi \le \theta \le \pi$ eg: longitude

Figure 24: Superformula extended to the third dimension using spherical product[35]

The first equation describes points of the supershape, similar to how a circle may be defined with points of a certain distance (radius, r) from the centre (going around with theta). The following equations describe the extension of this into the third dimension, using spherical coordinates to define points around the supershape.
With this, the parameters that define the supershapes $r1$ and $r2$ (m, n1, n2, n3, a, b) can be modified to generate 3d shapes of many different forms. In code, the step value which iterates around and plots the points of the sphere can also be modified to give supershapes higher or lower resolutions.
The most interesting application of shape generation with supershapes in $FL\bar{O}$ is in worm and weapon generation. To summarise:

- larger amplitude values increase the potential value of n1, n2 and n3 making the shape "spikier"
- faster tempo values increase the resolution of the shape making the shape have potentially more spikes or appear more rounded.

Each time an enemy is spawned or gun reward is grown, these values are updated to change its shape.

Additionally, Processing includes methods to apply transformations to visuals drawn on the screen. Weapon rewards make use of even more parameters to define their overall appearance:

- The gun's handedness depends on the amplitude of different audio channels throughout the song. E.g) A song that uses the right channel more than the left will produce a gun reward that's right handed.
- How much the gun is stretched in the x, y and z directions depends on the amplitude of the lows, mids and highs respectively.
- How much the gun is "sheared" in the x and y directions depends on the maximum amplitude values of the left and right audio buffers.

Weapons also have a wear value, but this is calculated based upon the player's performance (player score/max score). It is simply a factor which describes how vibrant the colours of the weapon are. This pushes players to achieve high scores in order to get weapons that display bright and vibrant colours.

7.4.3 Tree Generation/Growth

Trees take on a fractal like structure, their structures are defined by strings which abide by a set of formal grammar rules. Growing this tree means expanding this string by replacing parts of the string with valid syntactical rules according to the grammar - in other words applying production rules. This method of growth was inspired by the work of L-Systems [36]

In order to make the process music reactive, not only do trees grow on beats but the way they grow (which production rules are applied) is based upon what kind of beat was detected (kick, hat, snare).

8 Testing

Testing during the development phase of the project was carried out by trying out different songs of various different genres and verifying if the game plays as expected. On the extreme end of things, both silence and white noise were tested to ensure that gameplay is relatively stable. Additionally, the game was demonstrated a few times during the weekly supervisor meetings where critical feedback and new ideas were formed.

9 Evaluation and Critical Appraisal

To determine the overall success of the project, the prototype game developed, $FL\bar{O}$, was evaluated through conducting a player survey of five participants, comparing the game against the requirements outlined in Chapter 3, and comparing the game against related work in the field.

9.1 Player survey

A player survey was carried out to evaluate the initial version of the prototype game $FL\bar{O}$. The aim of this survey was to find out whether people enjoyed the game and the feature of Music-Reactive Procedural Generation. As such, the survey included questions about the gameplay experience (generally, visually, mechanically, and auditory) primarily between different songs (stages).

9.1.1 Method

- Five participants (Students/Staff) were recruited from within the university. Each of them had access to a computer with audio input and output.
- Participants were briefed and given instructions (via email) on how to conduct the evaluation based on the script provided within Appendix A.3. This included details on the install, play and evaluate the game.
- The results of the survey were collected via a questionnaire hosted on Google Forms.

The choice of the songs which were play-tested were specifically chosen to cover as wide a variety as possible. Unfortunately, this was quite difficult as music is subjective, and classifying two or three main genres which cover the entirety of music in existence may not be possible. That being said, I theorised that the main music genres included: Art/Classical music, Death Metal, And Folk. The reasoning for this is due to how each of these forms of music are transmitted via different mediums (i.e) Art/Classical $\rightarrow$ Written, Death Metal $\rightarrow$ Electronically, Folk $\rightarrow$ Orally) and that may have an effect on the music itself though this is purely speculation. The songs were copyright free and credit is provided below.

- False Awakenings - Anders (Death Metal.mp3)

- Les Petits Chanteurs de Passy - Salve Regina of Hermann Contract (Gregorian Chant.mp3)
- Acoustic/Folk Instrumental - Hyde - Free Instrumentals (Folk Instrumental.mp3)

The results of the survey were as follows:

9.1.2 General Questions

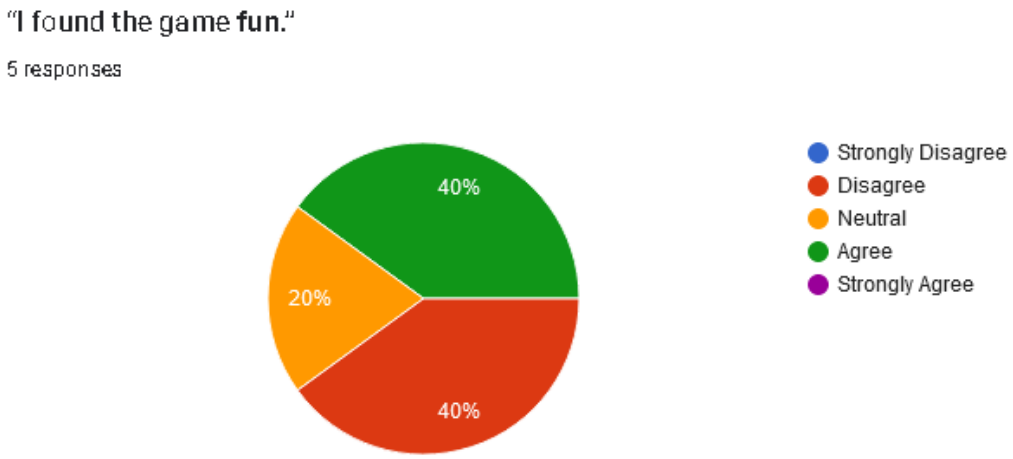

Figure 25: Question 1 Responses

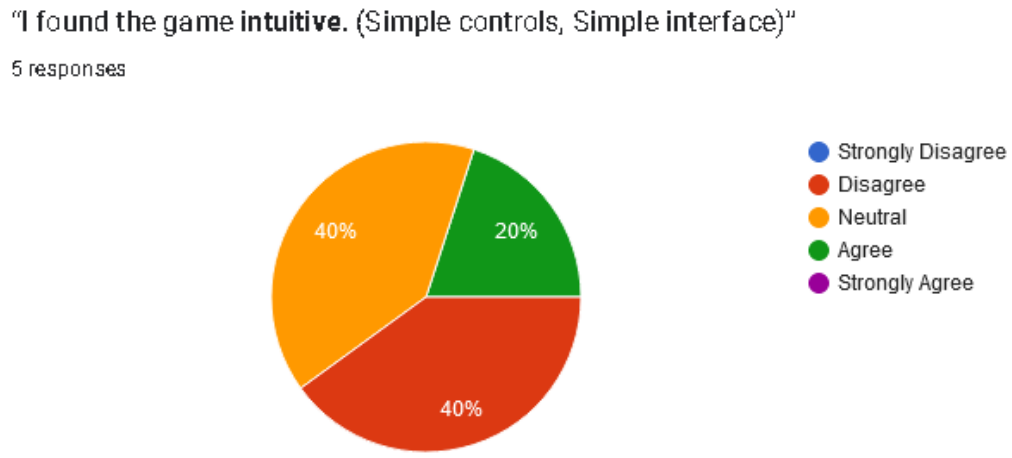

Figure 26: Question 2 Responses

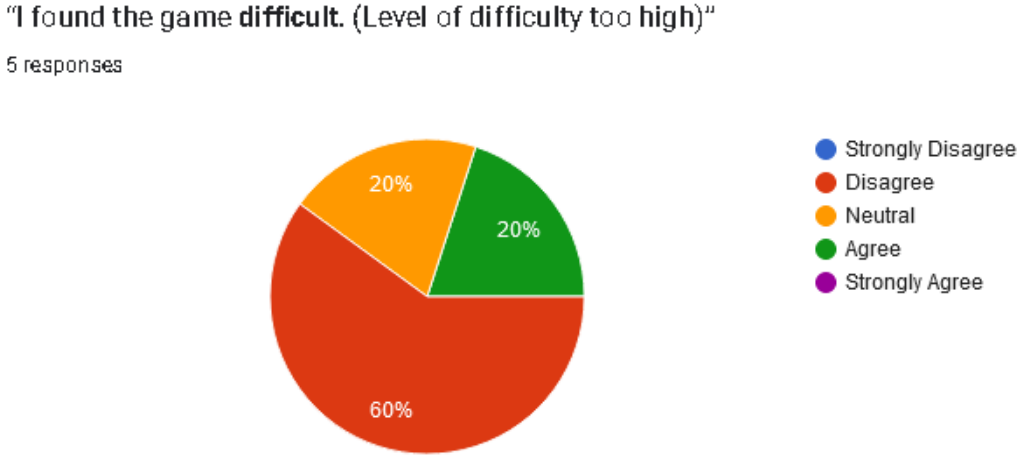

Figure 27: Question 3 Responses

9.1.3 Visual Experience Questions

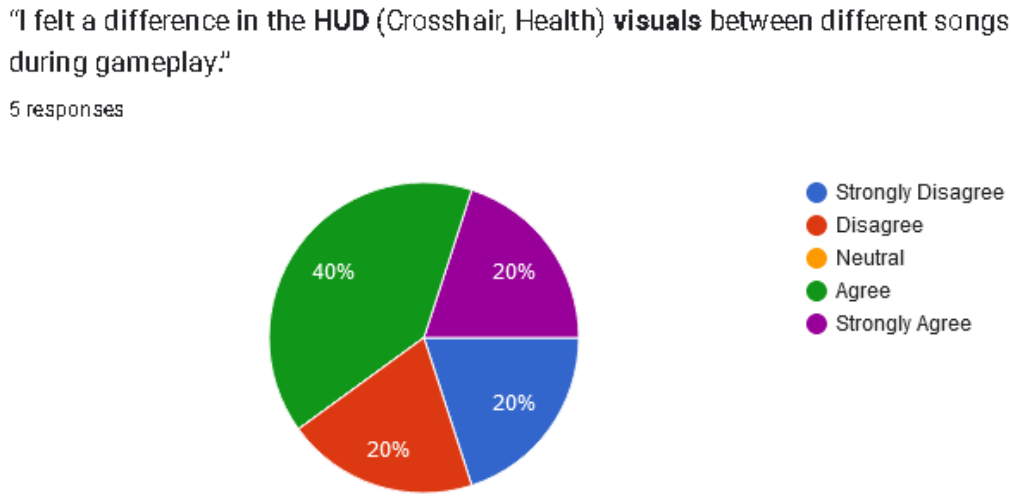

Figure 28: Question 4 Responses

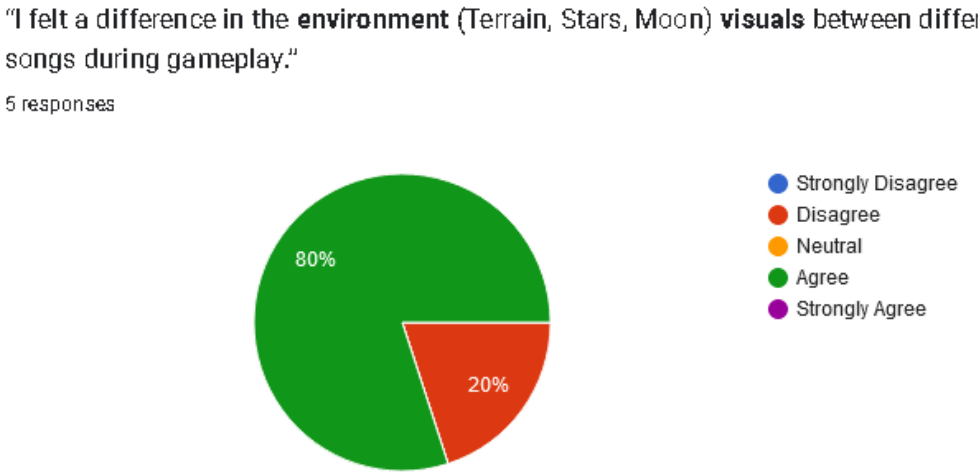

Figure 29: Question 5 Responses

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay."

5 responses

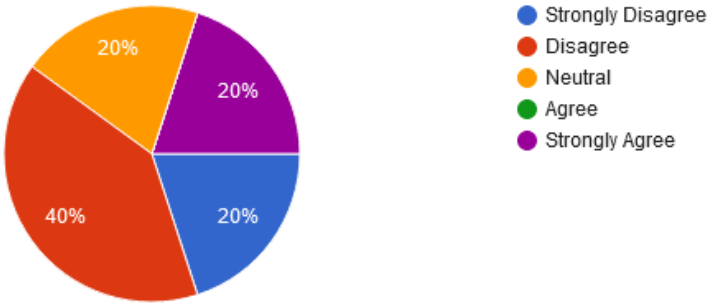

Figure 30: Question 6 Responses

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay."

5 responses

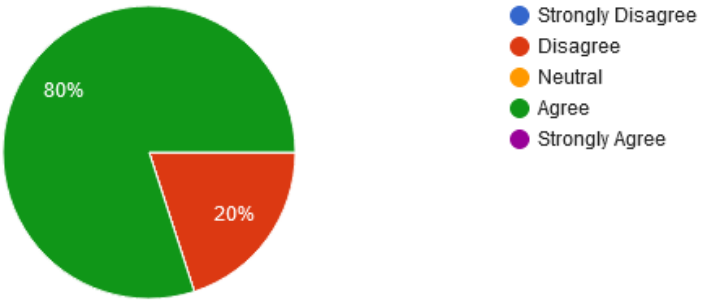

Figure 31: Question 7 Responses

9.1.4 Gameplay Questions

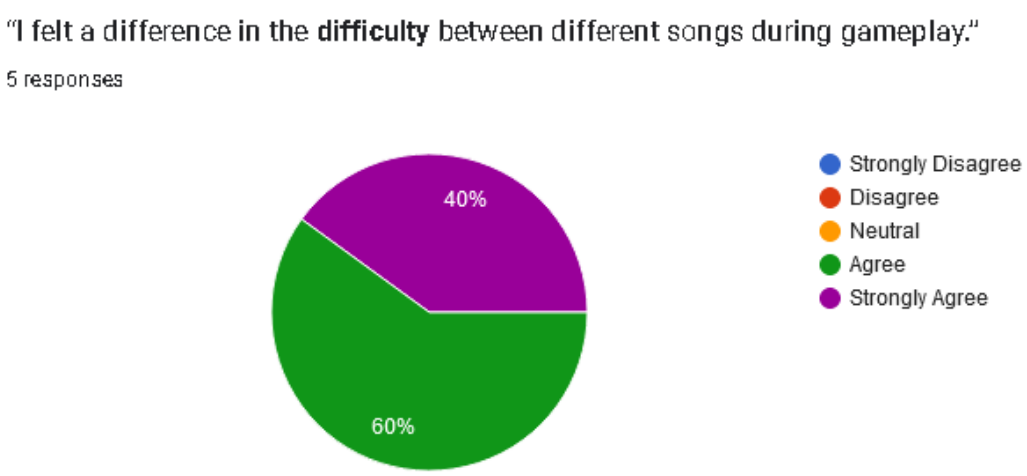

Figure 32: Question 8 Responses

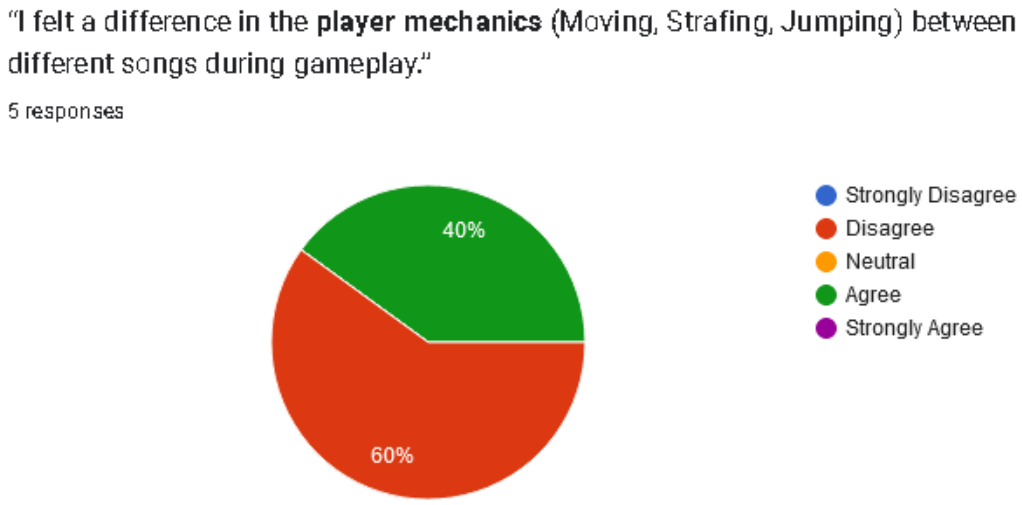

Figure 33: Question 9 Responses

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay."

5 responses

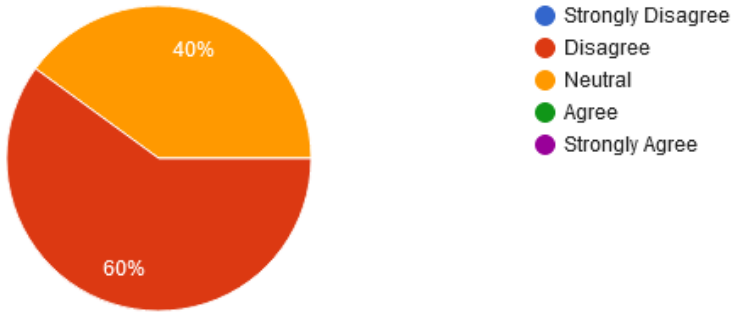

Figure 34: Question 10 Responses

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay."

5 responses

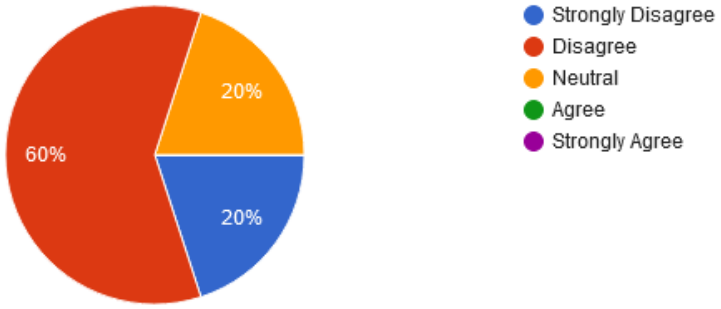

Figure 35: Question 11 Responses

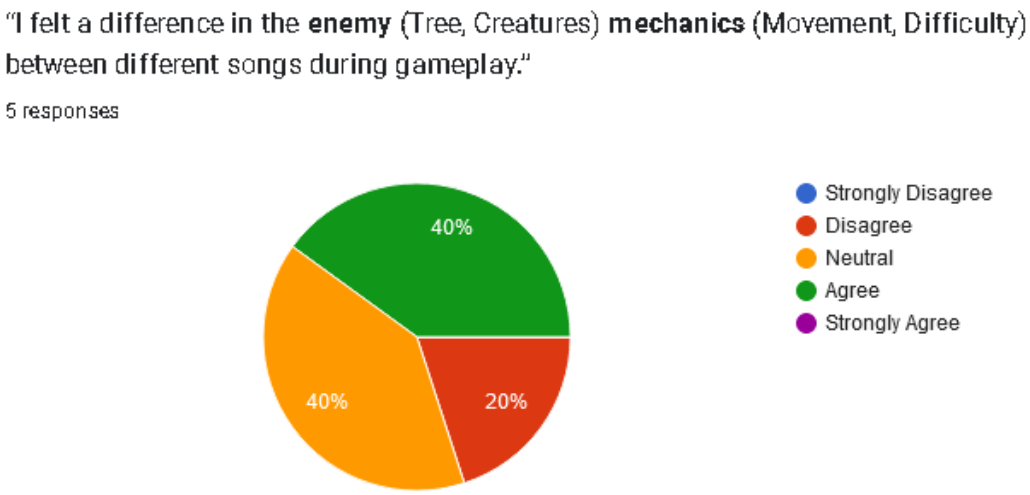

Figure 36: Question 12 Responses

9.1.5 Auditory Experience Questions

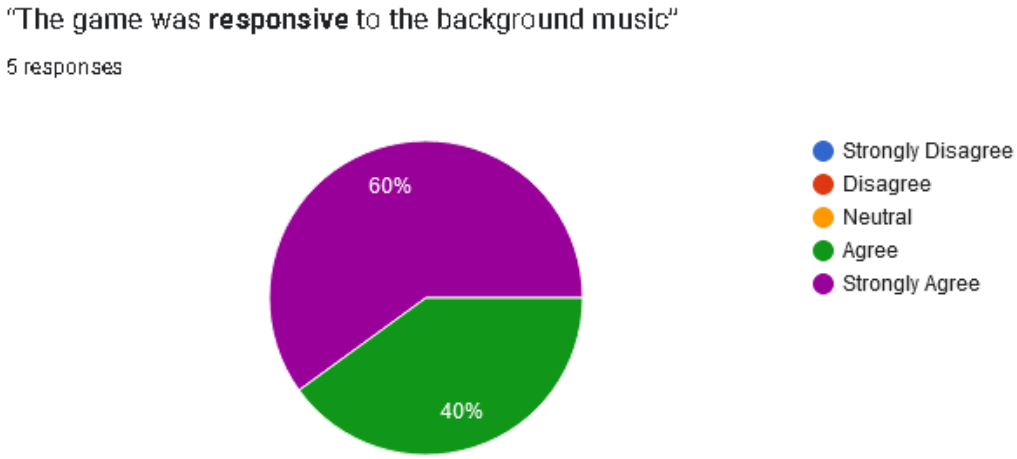

Figure 37: Question 13 Responses

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay."

5 responses

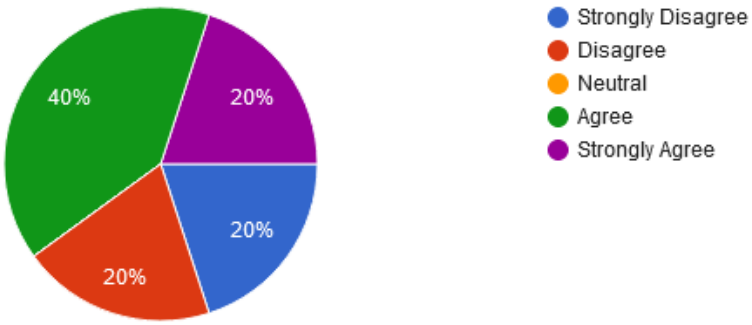

Figure 38: Question 14 Responses

"I felt a difference in the (health lost) **sound effects** during gameplay."

5 responses

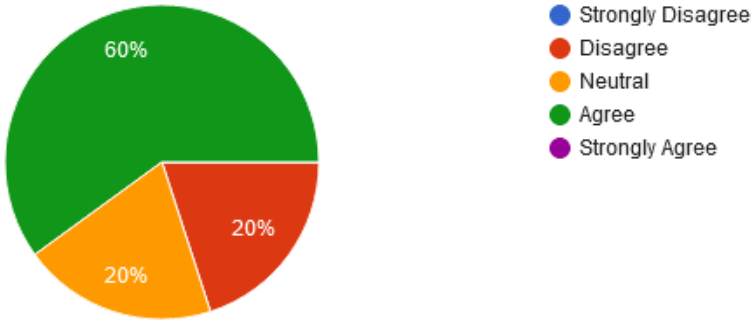

Figure 39: Question 15 Responses

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay."

5 responses

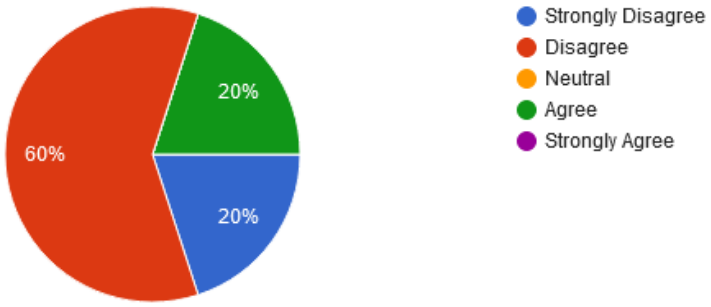

Figure 40: Question 16 Responses

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay."

5 responses

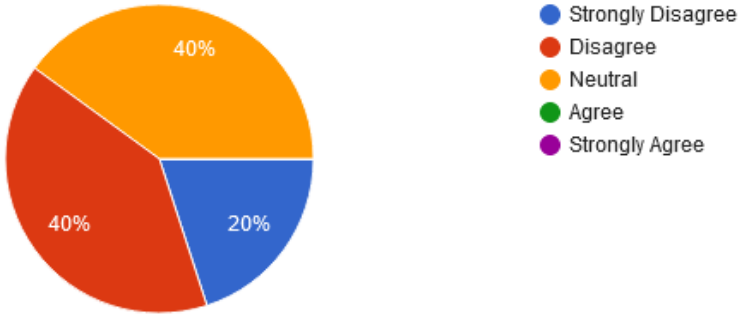

Figure 41: Question 17 Responses

"The **rhythm** of a song had an impact on the gameplay."

5 responses

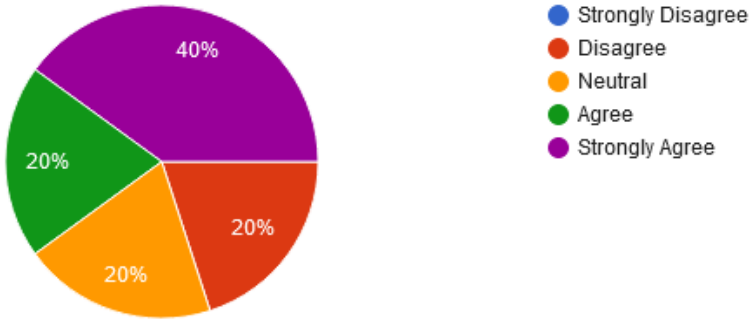

Figure 42: Question 18 Responses

"The **melody** of a song had an impact on the gameplay."

5 responses

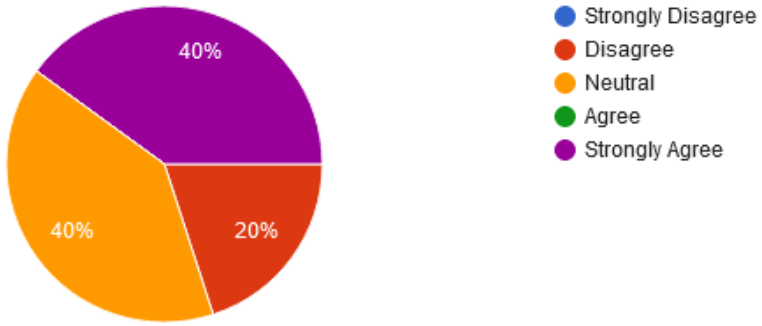

Figure 43: Question 19 Responses

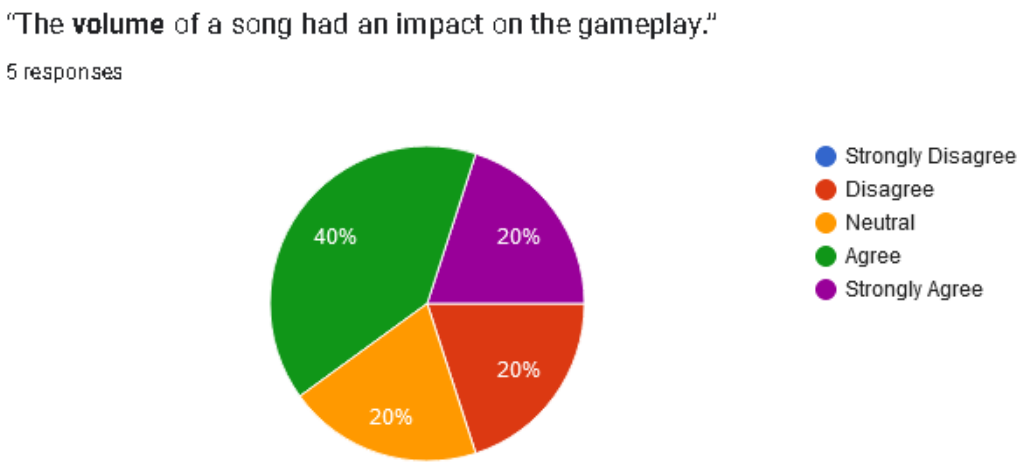

Figure 44: Question 20 Responses

9.1.6 Open-Ended Questions

"Did you encounter any technical issues while playing the game? If yes, please describe."

1. "No"
2. "Could not exit to menu after completing a level, had to exit to desktop and restart."
3. "No"
4. "Couldn't figure out how to escape to the menu aside from closing the game altogether..."
5. "During the Gregorian Chant song, I experienced a brief period where I was unable to stop firing. Also, while I'm unsure if this was an intended feature, during the Death Metal song the fire recoil was significantly larger compared to the other songs, which made it quite difficult to hit enemies consistently."

"What features or improvements would you like to see in future versions of the game?"

1. “I am playing with bluetooth headphones and there is a decent amount of lag, making it almost impossible to sync my movement and shooting with the music. A latency slider to delay or play the music sooner would be very useful and make the game more accessible.

   Custom songs would be nice!”

2. “Alright, let’s get started. This is going to come off as harsh, but I believe you have a solid core gameplay idea here. However, needs some polishing. Here are some suggestions. 1. Rhythm bar. A solid indicator of the beat of the track you are following would be a huge help with timing shots and jumps. 2. Weapon models. A distinction between weapons, especially one that gives you a good idea of how a weapon performs, would be excellent. 3. An armory where you can see unlocks. Seeing the weapons you can unlock and at what point thresholds would give an excellent goal to work towards. 4. Ingame tutorial. Introduce the mechanics in the game itself. 5. Make more use of vertical space. You have flying units and a lot of vertical space, use it.”

3. “I would say that 1) it is not obvious when and why I took damage, 2) not quite obvious when I make a critical hit (I think there is a tick in the size of the worm, but it could be made more visible), 3) The worm’s color is sometimes too close to that of the terrain, so it’s hard to spot them out, and 4) I can’t quite tell which part is the worm’s head; I think it’s the frontmost part, but it’s a bit hard to reach, and sometimes the worm is still alive when I (think that I) destroy the head.”

4. “Enemy-free mode would be appreciated for more casual players; it would be great if we could adjust the speed setting of our movement too. Different tree and rock designs would be cool, as well as other additions to the terrain to vary things more. Maybe changing the sky colour too.”

5. “Some of the songs did not spawn enemies like the worms for quite a while. I died to the rocks before I even saw a worm in Folk Instrumental, though of course I’m not sure what elements of a song would cause worms to spawn, so maybe this was intended?”

"Any additional comments or feedback you'd like to share about your experience with the game?"

1. "I am unsure of the design decision of making the enemies move with you, it doesnt really give you a sense of danger and makes it fiddly and unintuitive to shoot at them. The cubes being hp doesnt make much sense either in my opinion. More enemy variety would be nice as well as increased feedback such as damage numbers and healthbars. trees are also not a very clear ""obstacle"", something more obviously dangerous could be used. The filter effects when slowing down or speeding up are also quite distracting and not fun.

   The gun is not very obvious, maybe make it a shape other than a cube?

   I did play on a laptop with a trackpad so it made the game much harder. gamepad support would be great!

   The reason behind score is unclear, maybe add indicators such as killing an enemy gives +10 score to the player visually?

   Also feedback for syncing with the music would be really good in incentivising the user to do so

   The game starts off with the player looking at the ground. I didnt realise you were meant to look around so spend about 30 seconds just staring at the ground unsure if it was broken or not.

   The mic mechanic is cool but not sure how it would be actually used.

   Aside from visual changes, I am unsure how the music actually changed the gameplay? Maybe in enemy spawns and obstacle growth but it isnt very noticable

   I didnt feel much of an incentive to do anything. Just staying in the center sometimes moving left to right got me though it fine.

   The visual effects and syncing mechanics present are all good! It just needs a lot of refining to make the game more fun, good luck on the project!"

2. "1. Rhythm games work best with songs with a solid main beat. As such, I would recommend removing Gregorian Chants. 2. Because enemies move away from you, it can make landing a headshot very inconsistent, as the body can move in front of the head. This might

be remedied by enemies moving towards you. 3. Enemies had a tendency to blend in to the background, especially when they start small. Maybe make them a complementary color to the background? An outline would also help. 4. Enemies can linger in your hurtbox for extended periods, causing you to take multiple points of damage and obscuring your view. I would recommend enemies die instantly upon dealing damage to you (without awarding points). 5. It felt too easy to strafe around trees and spam shots at enemies instead of timing them to the beat. I don't have a fix for this one. 6. Healing feels a bit odd. Having to distinguish between normal floating cubes and spinning floating cubes can be tricky, and felt disconnected from the main mechanics. Maybe you could heal on a crit headshot instead to tie the two mechanics together? 7. Remove the muted music. Reducing the volume of the music while injured makes it harder to recover, which can lead to cascading failures. I would recommend reducing music volume for a brief window upon taking damage, then returning it too full volume. 8. Chunkier sound effects. The hits and especially taking damage need more impact. Make the crit sound effect distinct from the hit sound effect, and give the damage and jump sound effects. 9. Start facing forward. A minor problem, but I always started levels pointed at the ground and turned 90 degrees away from the environment. Maybe lock player head movement to only the play area? I would also recommend taking a look at Pistol Whip VR. It functions on a vaguely similar idea. With that all being said, thank you for inviting me to demo this. It was quite fun to explore this and think about how it could be improved. I wish you luck, and hope things go well if you decide to continue developing this."

3. "I think this idea of combining music and shooting is very creative! Keep it up!"

4. "It's a very cool concept. I loved being able to add my own songs and I think how much someone enjoys the game would definitely depend on the music that they choose (and how much they like it)"

5. "I really enjoyed the game! I like the combination of rhythm and FPS gameplay!"

9.1.7 Discussion

Overall, the play-testers seemed to have very mixed opinions about the game. A particularly interesting observation of the responses is that although all of the participants felt that the game responded to the music (Question 13 Responses), most of the players only felt this in some subset of the music-reactive game mechanics and presentation, hence the mixed reviews for every question. This suggests that the game may not give the player enough feedback, for each music-reactive game feature, for anyone to notice at least for different songs. To add to this, most participants felt that the game was not very intuitive, further suggesting this lack of feedback within certain areas of the game.
In the open ended responses, all of the players expressed positive reactions to the concept of Music-Reactive Procedural Content Generation however many felt that the game mechanics and visuals were not clear/intuitive. Additionally, a number of participants indicated the desire for more general game features and support for more latency prone systems.

9.2 Against Requirements

In terms of the completion of the project against the requirements set out in Chapter 3.

- All of the primary requirements were met
- Most of the secondary requirements were met (Did not complete multiple rounds of evaluation)
- One tertiary requirement was met (Playing the environment)

9.3 Against Related Work

$FL\bar{O}$ successfully implemented the feature of Music Reactive Procedural Content generation. This feature provides a mapping between music and gameplay in real time. Most similar work in this area although may have similar features, do not provide this mapping between music and gameplay as reactively as $FL\bar{O}$ (Vib-Ribbon[16] requires a further look-ahead, Background Music Reactive Games [21] requires an initial "sound processing" stage to

define a mapping and genre extraction is not achieved through audio analysis but through ID3 tags). $FL\bar{O}$ is able to demonstrate this property with the addition of "playing the environment".

10 Maintenance

With the constructive feedback that came from the evaluation survey for the game, $FL\bar{O}$, an updated version of the game was released. This version focused more on giving the player feedback for their actions and in turn (hopefully) making the game more intuitive. A number of bugs were also fixed. These additions are outlined below:

- Feature: Previewing of weapons.
- Bug: Bandpass filter that was used when player loses health was in constant effect, muting the colours of the game. Fix: Replaced with a low pass filter that is only in effect when the player is missing health.
- Feature: Music-Reactive Control Scheme
- Feature: Music-Reactive Crosshair updated to show damage and critical hits
- Feature: Extended the time for players to execute actions "on beat"
- Bug: Pitch Estimation with HPS did not used squared terms. Fix: Squared the terms in the HPS calculation
- Bug: Players started off facing the ground. Fix: Fix player's vision to start off looking forward
- Feature: Critical hits no longer one shot enemies and instead do 3x damage to balance fast firing weapons
- Feature: Worms stand out more with outlines and a highlighted head, along with connecting each part with a line.
- Feature: Player collision with enemies destroys enemies
- Feature: New songs which demonstrate the game's abilities to a better extent are added. Old ones removed. The new songs include

- RUNNING WATERS - Jason Shaw (Jason Shaw - RUNNING WATERS.mp3)
- big boost city - leon chang (leon chang - leon kart - 08 big boost city.mp3)
- Russian Dance - PM Music (Russian-Dance-PM-Music.mp3)
- We Shop Song - PM Music(We-Shop-Song-PM-Music.mp3)
- 2 Minutes and 30 Seconds of Silence - Anar Software LLC (2-minutes-and-30-seconds-of-silence.mp3)

11 Conclusions

11.1 Project Summary

In conclusion, this project explored the concept of Music-Reactive Procedural Content Generation through its implementation into the prototype game *FLŌ*.

11.2 Successes and Drawbacks

The project saw much success in the theory and the implementation of Music-Reactive Procedural Content Generation as a concept in the prototype game *FLŌ*. However, the enjoyment of the gameplay left much to be desired with overall mixed reviews based on the player survey conducted. Overall, the concept of Music-Reactive Procedural Content Generation in video games seems to be appealing however a well designed implementation of such a feature proves difficult as the mapping between the music and gameplay must be made clear and robust.

11.3 Future Work

11.3.1 Music-Reactive Procedural Content Generation

Music-Reactive Procedural Content Generation has a lot of potential to be a great feature of any game where music is the focus. Some immediate next steps for this feature would be to include more audio analysis tools in which video game content can be generated from. This can include techniques which extract the genre, or specific instruments from the music in real-time.

11.3.2 FLŌ

For the prototype game *FLŌ*, some possible next steps include: Adding a tutorial to make the game more easy to use/access, Adding more game-related features (not necessarily music-reactive) to make the game more fun, Implementing a more accurate pitch detection algorithm, Implementing a more accurate beat detection algorithm.

References

[1] Cooley, J. W., & Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation, 19(90), 297. https://doi.org/10.2307/2003354

[2] Patin, F. (2003). Beat Detection Algorithms. https://www.flipcode.com/misc/BeatDetectionAlgorithms.pdf

[3] L. Rabiner, M. Cheng, A. Rosenberg and C. McGonegal, "A comparative performance study of several pitch detection algorithms," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 5, pp. 399-418, October 1976, doi: 10.1109/TASSP.1976.1162846.

[4] Sukhostat, L., & İmamverdiyev, Y. (2015). A comparative analysis of pitch detection methods under the influence of different noise conditions. Journal of Voice, 29(4), 410–417. https://doi.org/10.1016/j.jvoice.2014.09.016

[5] Noll A.M. (1970): Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate. In: Symposium on Computer Processing in Communication; edited by Microwave Institute. New Yorks The University of Brooklyn Press; vol. 19, 779–797

[6] Shaker, N., Togelius, J., & Nelson, M. J. (2016). Procedural Content Generation in Games. In Computational Synthesis and Creative Systems. Springer International Publishing. https://doi.org/10.1007/978-3-319-42716-4

[7] Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games. ACM Transactions on Multimedia Computing, Communications, and Applications, 9(1), 1–22. https://doi.org/10.1145/2422956.2422957

[8] Steve Wozniak. (1976). Breakout[Arcade]. Atari, Inc.

[9] Ralph H. Baer & Howard J. Morrison. (1978). Simon [Handheld Electronic Game]. Hasbro.

[10] A.I. Design. (1980). Rogue: Exploring the Dungeons of Doom [Unix]. Epyx

[11] David Braben, Ian Bell. (1984). Elite [Acorn/BBC]. Acornsoft.

[12] Alexey Pajitnov. (1985). Tetris.

[13] NanaOn-Sha. (1996). PaRappa the Rapper [PlayStation]. Sony Computer Entertainment.

[14] Monster Rancher [PlayStation]. (1997). Tecmo.

[15] Positron. (1999). iS: internal section [PlayStation]. Square.

[16] NanaOn-Sha. (1999). Vib-Ribbon [PlayStation]. Sony Computer Entertainment.

[17] United Game Artists. (2001). Rez [Dreamcast]. Sega.

[18] Audiosurf [PC]. (2008). Fitterer, D. https://www.audio-surf.com/.

[19] Minecraft [PC]. (2011). Mojang Studios. https://www.minecraft.net/en-us

[20] Audiosurf 2 [PC]. (2015). Fitterer, D. https://www.audio-surf.com/.

[21] Juha Arrasvuori, & Holm, J. (2010). Background music reactive games. https://doi.org/10.1145/1930488.1930517

[22] Khalid Aallouche, Homam Albeiriss, Redouane Zarghoune, Juha Arrasvuori, Antti Eronen, & Holm, J. (2007). Implementation and evaluation of a background music reactive game. https://doi.org/10.5555/1367956.1367957

[23] Cachia, W., Aquilina, L., Héctor Martínez, & Yannakakis, G. N. (2014). Procedural generation of music-guided weapons. https://doi.org/10.1109/cig.2014.6932925

[24] Robert J. Brown. (1977). Atari Video Music. Atari, Inc.

Figures

[25] Statista. (August 27, 2023). Number of video game users worldwide from 2017 to 2027 (in billions) [Graph]. In Statista. Retrieved October 13, 2023, from https://www.statista.com/statistics/748044/number-video-gamers-world/

[26] Glitchunpatched. (2019). Beta.png [Image, Screenshot]. Minecraft Wiki. https://minecraft.fandom.com/wiki/Screenshot

[27] Thedarkb. (2021). Rogue_Screenshot.png [Image, Screenshot]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Rogue_Screenshot.png

[28] Shritwod. (2018). Simon Electronic Game.jpg [Photograph]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Simon_Electronic_Game.jpg

[29] Gameoverse. (2015). PaRappa The Rapper - Full Playthrough Thumbnail [Image, Screenshot]. YouTube. https://www.youtube.com/watch?v=y3SI4Grases

[30] Zeusdaz. (2016). INTERNAL SECTION (PS1- FULL GAME) at 44:00 [Image, Screenshot]. YouTube. https://www.youtube.com/watch?v=3lHUqpsZI20

[31] Zxcvbnm. (2022). Vib-Ribbon_gameplay.png [Image, Screenshot]. Wikipedia. https://en.wikipedia.org/wiki/File:Vib-Ribbon_gameplay.png

[32] David M. (2022). Rez gameplay [Image, Screenshot]. Den of Geek. https://www.denofgeek.com/games/sega-dreamcast-games-that-were-way-ahead-of-their-time/

[33] Thomas A. (2008). Audiosurf gameplay [Image, Screenshot]. GameSpot. https://www.gamespot.com/reviews/audiosurf-review/1900-6189185/

[34] Gielis, J. (2003), A generic geometric transformation that unifies a wide range of natural and abstract shapes. Am. J. Bot., 90: 333-338. https://doi.org/10.3732/ajb.90.3.333

[35] Bourke P. (2002 March). Supershapes (Superformula). Retrieved from https://paulbourke.net/geometry/supershape/

[36] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. Journal of Theoretical Biology, 18(3), 300–315. https://doi.org/10.1016/0022-5193(68)90080-5

A Appendix

A.1 Artifact Evaluation form

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
ARTIFACT EVALUATION FORM

Title of project

Music-Reactive Procedural Content Generation in Video Games

Name of researcher(s)

Marcus Yiu Yeung Yip

Name of supervisor

Ian James Miguel

Self audit has been conducted **YES** ☒ **NO** ☐

This project is covered by the ethical application CS15727.

Signature Student or Researcher

MYiP

Print Name

MARCUS YIU YEUNG YIP

Date

14/09/2023

Signature Lead Researcher or Supervisor

Print Name

Ian Miguel

Date

14/9/23

A.2 FLŌ

Versions 1 and 2 of the game *FLŌ*.
https://universityofstandrews907-my.sharepoint.com/:u:/g/personal/myyy1_st-andrews_ac_uk/EdB40xrLNVdFo7xOb7akM_IBVT33XtXqUC-XJeeDCtPkFQ

A.3 Script

Thank you again for showing interest in my project! Below I have outlined the steps required in order to carry out the evaluation process. If you could complete these steps before the end of this month that would be great!

1. Read the README.txt file attached to this email. It includes details surrounding the installation, running and playing of the game! ⟨∼5 mins⟩
2. Play each of the songs at least once. I recommend starting with "Folk Instrumental.mp3" to learn the controls (it may take 2 or 3 tries) before proceeding to "Gregorian Chant.mp3" and lastly "Death Metal.mp3". ⟨∼15 mins⟩
3. Fill out the anonymous evaluation form/questionnaire over at: https://forms.gle/h6oUZKTLxjYsGwMy6 ⟨∼10 mins⟩

Please do let me know if you have any questions surrounding the evaluation process or with playing the game!
Sincerely, Marcus

A.4 Player Guides/README.txt

FLO PLAYER GUIDE

Welcome to Flo, the Rhythmic Rail Shooter!

INSTALLATION

---

1. Extract Flo.zip

RUNNING THE GAME

1. Navigate to the Flo directory
2. Run Flo.exe
3. Have fun!

RULES AND MECHANICS

. The aim of the game is to shoot and dodge your way through waves of enemies and obstacles generated from the background music without dying.

. There are several types of enemies which you may encounter, each with different characteristics.
- Water Worm : A basic worm. It has the ability to wriggle fast.
- Earth Worm : A worm with legs. It has the ability to wriggle across the terrain.
- Air Worm : A worm with wings. It has the ability to fly.

. There are several obstacles which you may encounter, these include:
- The terrain: Spiky ground.
- (Cubic) rocks: Floating rocks.
- Plants/Trees: Growing trees.

. Completing songs/levels unlock weapons which will have features based on your performance and the song/level played.
- (Shape, Location, Firerate, Wear)

. TIPS:
- Shooting in time (on beat) with the music can give critical hits (one shot kills).
- Jumping in time (on beat) with the music can give an extra boost in height.
- Moving backwards and forwards makes the game easier and harder respectively.
- Shooting "glowing"/"outlined" rocks can replenish the player's HP.
- Trees hitboxes are outlined with a sphere, only hitting the main branch damages the player.
- Enemies can regenerate parts of their body as long as their head is still

alive.
. Their heads are naturally tougher than their body parts howerver are not invulnerable to critical hits.
- The scoring system reflects the player's performance in terms of: Distance travelled, Enemies killed, Health, Shooting accuracy.

. Additional Feature: The player is able to "play the sounds of their environment" via their microphone.

CONTROLS

W - Forward
A - Left
S - Backward
D - Right
Space - Jump

LClick - Shoot
Scroll - Change music volume
V - Toggle mic
Esc - Exit

Flo (Version 1.0), Marcus Yiu Yeung Yip (190019931)

FLO PLAYER GUIDE

Welcome to Flo, the Rhythmic Rail Shooter!

INSTALLATION

1. Extract Flo.zip

RUNNING THE GAME

1. Navigate to the Flo directory
2. Run Flo.exe
3. Have fun!

RULES AND MECHANICS

. The aim of the game is to shoot and dodge your way through waves of enemies and obstacles generated from the background music without dying.

. There are several types of enemies which you may encounter, each with different characteristics.
- Water Worm : A basic worm. It has the ability to wriggle fast.
- Earth Worm : A worm with legs. It has the ability to wriggle across the terrain.
- Air Worm : A worm with wings. It has the ability to fly.

. There are several obstacles which you may encounter, these include:
- The terrain: Spiky ground.
- (Cubic) rocks: Floating rocks.
- Plants/Trees: Growing trees.

. Completing songs/levels unlock weapons which will have features based on your performance and the song/level played.
- (Shape, Location, Firerate, Wear)

. TIPS:
- Shooting in time (on beat) with the music can give critical hits (3x damage!).
- Jumping in time (on beat) with the music can give an extra boost in height.
- Moving backwards and forwards makes the game easier and harder respectively.
- Shooting "glowing"/"outlined" rocks can replenish the player's HP.
- Trees hitboxes are outlined with a sphere, only hitting the main branch damages the player.
- Enemies can regenerate parts of their body as long as their head is still alive.
. Their heads are naturally tougher than their body parts howerver are not

invulnerable to critical hits.
- The scoring system reflects the player's performance in terms of: Distance travelled, Enemies killed, Health, Shooting accuracy.

. Additional Feature: The player is able to "play the sounds of their environment" via their microphone.

CONTROLS
———
W - Forward
A - Left
S - Backward
D - Right
Space - Jump

LClick - Shoot
Scroll - Change music volume
V - Toggle mic
Esc - Exit

———————————————————————

Flo (Version 2.0), Marcus Yiu Yeung Yip (190019931)

A.5 Questionnaire responses

Flō Evaluation Form (Questionnaire)

Evaluation form (Questionnaire) for the game "*Flō*" written by Marcus Yip (Version 1.0).

Ethics

This project is covered by the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

The participation in this project is completely voluntary and you can withdraw from the study at any time without giving an explanation and with no disbenefit. You can ask questions about the project and have had them answered satisfactorily.

In this project, no personal data will be collected. Data collected in this project will be anonymised and the raw data will be deleted within 3 months after the completion of the project. The data will be only accessible to the named researchers on the project and the data analysis results will be published in a dissertation or an academic publication.

I have read and understood the information provided above. By proceeding, I consent to participate in this research project. *

- (●) Agree

General

For the following statements, please choose the option which best describes your experience.

"I found the game **fun**." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **intuitive**. (Simple controls, Simple interface)" *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **difficult**. (Level of difficulty too high)" *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Visuals

For the following statements, please choose the option which best describes your **visual** experience. (Graphics of the game, how the game looks)

"I felt a difference in the **HUD** (Crosshair, Health) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain, Stars, Moon) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay." *

- ◉ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Mechanics

For the following statements, please choose the option which best describes your **gameplay** experience. (Rules and mechanics of the game, how the game works)

"I felt a difference in the **difficulty** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **player mechanics** (Moving, Strafing, Jumping) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **mechanics** (Movement, Difficulty) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

Sound

For the following statements, please choose the option which best describes your **auditory** experience. (Audio effects of the game, how the game sounds)

"The game was **responsive** to the background music" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the (health lost) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay." *

- (●) Strongly Disagree
- ( ) Disagree
- ( ) Neutral
- ( ) Agree
- ( ) Strongly Agree

"The **rhythm** of a song had an impact on the gameplay." *

- ( ) Strongly Disagree
- (●) Disagree
- ( ) Neutral
- ( ) Agree
- ( ) Strongly Agree

"The **melody** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The **volume** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

Open-ended

For the following questions, please provide a brief summary of your experience where applicable.

"Did you encounter any **technical issues** while playing the game? If yes, please describe." *

No

"What **features** or **improvements** would you like to see in **future versions** of the game?" *

I am playing with bluetooth headphones and there is a decent amount of lag, making it almost impossible to sync my movement and shooting with the music. A latency slider to delay or play the music sooner would be very useful and make the game more accessible.

Custom songs would be nice!

"Any additional **comments** or **feedback** you'd like to share about your experience with the game?" *

I am unsure of the design decision of making the enemies move with you, it doesnt really give you a sense of danger and makes it fiddly and unintuitive to shoot at them. The cubes being hp doesnt make much sense either in my opinion. More enemy variety would be nice as well as increased feedback such as damage numbers and healthbars. trees are also not a very clear "obstacle", something more obviously dangerous could be used. The filter effects when slowing down or speeding up are also quite distracting and not fun.

The gun is not very obvious, maybe make it a shape other than a cube?

I did play on a laptop with a trackpad so it made the game much harder. gamepad support would be great!

The reason behind score is unclear, maybe add indicators such as killing an enemy gives +10 score to the player visually?

Also feedback for syncing with the music would be really good in incentivising the user to do so

The game starts off with the player looking at the ground. I didnt realise you were meant to look around so spend about 30 seconds just staring at the ground unsure if it was broken or not.

The mic mechanic is cool but not sure how it would be actually used.

Aside from visual changes, I am unsure how the music actually changed the gameplay? Maybe in enemy spawns and obstacle growth but it isnt very noticable

I didnt feel much of an incentive to do anything. Just staying in the center sometimes moving left to right got me though it fine.

The visual effects and syncing mechanics present are all good! It just needs a lot of refining to make the game more fun, good luck on the project!

End

This marks the end of the questionnaire, I hope you had fun!

Once again, I'd like to thank you for participating in my research project!

Marcus.

Flō Evaluation Form (Questionnaire)

Evaluation form (Questionnaire) for the game "*Flō*" written by Marcus Yip (Version 1.0).

Ethics

This project is covered by the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

The participation in this project is completely voluntary and you can withdraw from the study at any time without giving an explanation and with no disbenefit. You can ask questions about the project and have had them answered satisfactorily.

In this project, no personal data will be collected. Data collected in this project will be anonymised and the raw data will be deleted within 3 months after the completion of the project. The data will be only accessible to the named researchers on the project and the data analysis results will be published in a dissertation or an academic publication.

I have read and understood the information provided above. By proceeding, I consent to participate in this research project. *

- (●) Agree

General

For the following statements, please choose the option which best describes your experience.

"I found the game **fun**." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **intuitive**. (Simple controls, Simple interface)" *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **difficult**. (Level of difficulty too high)" *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

Visuals

For the following statements, please choose the option which best describes your **visual** experience. (Graphics of the game, how the game looks)

"I felt a difference in the **HUD** (Crosshair, Health) **visuals** between different songs during gameplay." *

- ◉ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain, Stars, Moon) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Mechanics

For the following statements, please choose the option which best describes your **gameplay** experience. (Rules and mechanics of the game, how the game works)

"I felt a difference in the **difficulty** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the **player mechanics** (Moving, Strafing, Jumping) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay." *

- (●) Strongly Disagree
- ( ) Disagree
- ( ) Neutral
- ( ) Agree
- ( ) Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **mechanics** (Movement, Difficulty) between different songs during gameplay." *

- ( ) Strongly Disagree
- ( ) Disagree
- ( ) Neutral
- (●) Agree
- ( ) Strongly Agree

Sound

For the following statements, please choose the option which best describes your **auditory** experience. (Audio effects of the game, how the game sounds)

"The game was **responsive** to the background music" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (health lost) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay." *

- ◉ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The **rhythm** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"The **melody** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"The **volume** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

Open-ended

For the following questions, please provide a brief summary of your experience where applicable.

"Did you encounter any **technical issues** while playing the game? If yes, please describe." *

Could not exit to menu after completing a level, had to exit to desktop and restart.

"What **features** or **improvements** would you like to see in **future versions** of the game?" *

Alright, let's get started. This is going to come off as harsh, but I believe you have a solid core gameplay idea here. However, needs some polishing. Here are some suggestions.
1. Rhythm bar. A solid indicator of the beat of the track you are following would be a huge help with timing shots and jumps.
2. Weapon models. A distinction between weapons, especially one that gives you a good idea of how a weapon performs, would be excellent.
3. An armory where you can see unlocks. Seeing the weapons you can unlock and at what point thresholds would give an excellent goal to work towards.
4. Ingame tutorial. Introduce the mechanics in the game itself.
5. Make more use of vertical space. You have flying units and a lot of vertical space, use it.

"Any additional **comments** or **feedback** you'd like to share about your experience with the game?" *

1. Rhythm games work best with songs with a solid main beat. As such, I would recommend removing Gregorian Chants.
2. Because enemies move away from you, it can make landing a headshot very inconsistent, as the body can move in front of the head. This might be remedied by enemies moving towards you.
3. Enemies had a tendency to blend in to the background, especially when they start small. Maybe make them a complementary color to the background? An outline would also help.
4. Enemies can linger in your hurtbox for extended periods, causing you to take multiple points of damage and obscuring your view. I would recommend enemies die instantly upon dealing damage to you (without awarding points).
5. It felt too easy to strafe around trees and spam shots at enemies instead of timing them to the beat. I don't have a fix for this one.
6. Healing feels a bit odd. Having to distinguish between normal floating cubes and spinning floating cubes can be tricky, and felt disconnected from the main mechanics. Maybe you could heal on a crit headshot instead to tie the two mechanics together?
7. Remove the muted music. Reducing the volume of the music while injured makes it harder to recover, which can lead to cascading failures. I would recommend reducing music volume for a brief window upon taking damage, then returning it too full volume.
8. Chunkier sound effects. The hits and especially taking damage need more impact. Make the crit sound effect distinct from the hit sound effect, and give the damage and jump sound effects.
9. Start facing forward. A minor problem, but I always started levels pointed at the ground and turned 90 degrees away from the environment. Maybe lock player head movement to only the play area?
I would also recommend taking a look at Pistol Whip VR. It functions on a vaguely similar idea.
With that all being said, thank you for inviting me to demo this. It was quite fun to explore this and think about how it could be improved. I wish you luck, and hope things go well if you decide to continue developing this.

End

This marks the end of the questionnaire, I hope you had fun!

Once again, I'd like to thank you for participating in my research project!

Marcus.

Flō Evaluation Form (Questionnaire)

Evaluation form (Questionnaire) for the game "*Flō*" written by Marcus Yip (Version 1.0).

Ethics

This project is covered by the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

The participation in this project is completely voluntary and you can withdraw from the study at any time without giving an explanation and with no disbenefit. You can ask questions about the project and have had them answered satisfactorily.

In this project, no personal data will be collected. Data collected in this project will be anonymised and the raw data will be deleted within 3 months after the completion of the project. The data will be only accessible to the named researchers on the project and the data analysis results will be published in a dissertation or an academic publication.

I have read and understood the information provided above. By proceeding, I consent to participate in this research project. *

- (●) Agree

General

For the following statements, please choose the option which best describes your experience.

"I found the game **fun**." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **intuitive**. (Simple controls, Simple interface)" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I found the game **difficult**. (Level of difficulty too high)" *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Visuals

For the following statements, please choose the option which best describes your **visual** experience. (Graphics of the game, how the game looks)

"I felt a difference in the **HUD** (Crosshair, Health) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the **environment** (Terrain, Stars, Moon) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Mechanics

For the following statements, please choose the option which best describes your **gameplay** experience. (Rules and mechanics of the game, how the game works)

"I felt a difference in the **difficulty** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the **player mechanics** (Moving, Strafing, Jumping) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **mechanics** (Movement, Difficulty) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

Sound

For the following statements, please choose the option which best describes your **auditory** experience. (Audio effects of the game, how the game sounds)

"The game was **responsive** to the background music" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay." *

- ◉ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the (health lost) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"The **rhythm** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"The **melody** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"The **volume** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Open-ended

For the following questions, please provide a brief summary of your experience where applicable.

"Did you encounter any **technical issues** while playing the game? If yes, please describe." *

No

"What **features** or **improvements** would you like to see in **future versions** of the game?" *

I would say that 1) it is not obvious when and why I took damage, 2) not quite obvious when I make a critical hit (I think there is a tick in the size of the worm, but it could be made more visible), 3) The worm's color is sometimes too close to that of the terrain, so it's hard to spot them out, and 4) I can't quite tell which part is the worm's head; I think it's the frontmost part, but it's a bit hard to reach, and sometimes the worm is still alive when I (think that I) destroy the head.

"Any additional **comments** or **feedback** you'd like to share about your experience with the game?" *

I think this idea of combining music and shooting is very creative! Keep it up!

End

This marks the end of the questionnaire, I hope you had fun!

Once again, I'd like to thank you for participating in my research project!

Marcus.

Flō Evaluation Form (Questionnaire)

Evaluation form (Questionnaire) for the game "*Flō*" written by Marcus Yip (Version 1.0).

Ethics

This project is covered by the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

The participation in this project is completely voluntary and you can withdraw from the study at any time without giving an explanation and with no disbenefit. You can ask questions about the project and have had them answered satisfactorily.

In this project, no personal data will be collected. Data collected in this project will be anonymised and the raw data will be deleted within 3 months after the completion of the project. The data will be only accessible to the named researchers on the project and the data analysis results will be published in a dissertation or an academic publication.

I have read and understood the information provided above. By proceeding, I consent to participate in this research project. *

- (●) Agree

General

For the following statements, please choose the option which best describes your experience.

"I found the game **fun**." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I found the game **intuitive**. (Simple controls, Simple interface)" *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **difficult**. (Level of difficulty too high)" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Visuals

For the following statements, please choose the option which best describes your **visual** experience. (Graphics of the game, how the game looks)

"I felt a difference in the **HUD** (Crosshair, Health) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain, Stars, Moon) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Mechanics

For the following statements, please choose the option which best describes your **gameplay** experience. (Rules and mechanics of the game, how the game works)

"I felt a difference in the **difficulty** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **player mechanics** (Moving, Strafing, Jumping) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **mechanics** (Movement, Difficulty) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Sound

For the following statements, please choose the option which best describes your **auditory** experience. (Audio effects of the game, how the game sounds)

"The game was **responsive** to the background music" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (health lost) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"The **rhythm** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"The **melody** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"The **volume** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

Open-ended

For the following questions, please provide a brief summary of your experience where applicable.

"Did you encounter any **technical issues** while playing the game? If yes, please describe." *

Couldn't figure out how to escape to the menu aside from closing the game altogether...

"What **features** or **improvements** would you like to see in **future versions** of the game?" *

Enemy-free mode would be appreciated for more casual players; it would be great if we could adjust the speed setting of our movement too. Different tree and rock designs would be cool, as well as other additions to the terrain to vary things more. Maybe changing the sky colour too.

"Any additional **comments** or **feedback** you'd like to share about your experience with the game?" *

It's a very cool concept. I loved being able to add my own songs and I think how much someone enjoys the game would definitely depend on the music that they choose (and how much they like it)

End

This marks the end of the questionnaire, I hope you had fun!

Once again, I'd like to thank you for participating in my research project!

Marcus.

Flō Evaluation Form (Questionnaire)

Evaluation form (Questionnaire) for the game "*Flō*" written by Marcus Yip (Version 1.0).

Ethics

This project is covered by the ethics application for "Evaluation of artefacts produced for CS projects" (with the ethics approval code CS15727) at the University of St Andrews.

The participation in this project is completely voluntary and you can withdraw from the study at any time without giving an explanation and with no disbenefit. You can ask questions about the project and have had them answered satisfactorily.

In this project, no personal data will be collected. Data collected in this project will be anonymised and the raw data will be deleted within 3 months after the completion of the project. The data will be only accessible to the named researchers on the project and the data analysis results will be published in a dissertation or an academic publication.

I have read and understood the information provided above. By proceeding, I consent to participate in this research project. *

- (●) Agree

General

For the following statements, please choose the option which best describes your experience.

"I found the game **fun**." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I found the game **intuitive**. (Simple controls, Simple interface)" *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I found the game **difficult**. (Level of difficulty too high)" *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Visuals

For the following statements, please choose the option which best describes your **visual** experience. (Graphics of the game, how the game looks)

"I felt a difference in the **HUD** (Crosshair, Health) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain, Stars, Moon) **visuals** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun visuals** (Shape, Size, Location, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **visuals** (Shape, Size, Colour) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Mechanics

For the following statements, please choose the option which best describes your **gameplay** experience. (Rules and mechanics of the game, how the game works)

"I felt a difference in the **difficulty** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"I felt a difference in the **player mechanics** (Moving, Strafing, Jumping) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **environment** (Terrain) **mechanics** between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **gun mechanics** (Fire Rate, Damage) between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the **enemy** (Tree, Creatures) **mechanics** (Movement, Difficulty) between different songs during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Sound

For the following statements, please choose the option which best describes your **auditory** experience. (Audio effects of the game, how the game sounds)

"The game was **responsive** to the background music" *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the (sprinting, walking, movement) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"I felt a difference in the (health lost) **sound effects** during gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

"I felt a difference in the (gun shooting) **sound effects** between different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ◉ Agree
- ○ Strongly Agree

"The (gun shooting) **sound effects** were harmonious to the different songs during gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ◉ Neutral
- ○ Agree
- ○ Strongly Agree

"The **rhythm** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"The **melody** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ○ Disagree
- ○ Neutral
- ○ Agree
- ◉ Strongly Agree

"The **volume** of a song had an impact on the gameplay." *

- ○ Strongly Disagree
- ◉ Disagree
- ○ Neutral
- ○ Agree
- ○ Strongly Agree

Open-ended

For the following questions, please provide a brief summary of your experience where applicable.

"Did you encounter any **technical issues** while playing the game? If yes, please describe." *

During the Gregorian Chant song, I experienced a brief period where I was unable to stop firing. Also, while I'm unsure if this was an intended feature, during the Death Metal song the fire recoil was significantly larger compared to the other songs, which made it quite difficult to hit enemies consistently.

"What **features** or **improvements** would you like to see in **future versions** of the game?" *

Some of the songs did not spawn enemies like the worms for quite a while. I died to the rocks before I even saw a worm in Folk Instrumental, though of course I'm not sure what elements of a song would cause worms to spawn, so maybe this was intended?

"Any additional **comments** or **feedback** you'd like to share about your experience with the game?" *

I really enjoyed the game! I like the combination of rhythm and FPS gameplay!

End

This marks the end of the questionnaire, I hope you had fun!

Once again, I'd like to thank you for participating in my research project!

Marcus.