|  |
| --- |
| An Investigation into Markov Chains and Experience Driven Procedural Content Generation for Algorithmic Music  Andrew Milne  BSc Computer Games Technology, 2019 |

|  |
| --- |
| School of Design and Informatics  Abertay University |

# a - Table of Contents

[a - Table of Contents i](#_Toc7540860)

[b - Table of Figures iv](#_Toc7540861)

[b - Table of Tables v](#_Toc7540862)

[d - Acknowledgements vi](#_Toc7540863)

[e - Abstract vii](#_Toc7540864)

[f - Abbreviations, Symbols and Notation viii](#_Toc7540865)

[1 - Introduction 1](#_Toc7540866)

[1.1 - The problem with games 1](#_Toc7540867)

[1.2 - Conventional Methods 1](#_Toc7540868)

[1.3 - Procedural Music 3](#_Toc7540869)

[1.4 - Aims 4](#_Toc7540870)

[1.5 - Overview of Remaining Chapters 4](#_Toc7540871)

[2 - Literature Review 5](#_Toc7540872)

[2.1 - Generative Algorithms 5](#_Toc7540873)

[2.1.1 - Neural Networks 6](#_Toc7540874)

[2.1.2 - Genetic Algorithms 6](#_Toc7540875)

[2.1.3 - Markov Chains 7](#_Toc7540876)

[2.2 - Creating Emotion 10](#_Toc7540877)

[2.2.1 - Valence 11](#_Toc7540878)

[2.2.2 - Intensity 13](#_Toc7540879)

[2.2.3 - Game Play as the Composer 14](#_Toc7540880)

[2.3 - Musical Structure 15](#_Toc7540881)

[2.3.1 - Theme 15](#_Toc7540882)

[2.3.2 - Leitmotif 17](#_Toc7540883)

[2.3.2 - Period 19](#_Toc7540884)

[2.3.3 - Sentence 21](#_Toc7540885)

[2.4 - Summary 22](#_Toc7540886)

[3 – Methodology 22](#_Toc7540887)

[3.1 - Reading in MIDI files 22](#_Toc7540888)

[3.2 - Markov Chains 25](#_Toc7540889)

[3.2.1 - Frequency Distributions 25](#_Toc7540890)

[3.2.2 - Choosing Notes 26](#_Toc7540891)

[3.2.3 - Creating Structured Melodies 28](#_Toc7540892)

[3.2.4 - Playing the Music 28](#_Toc7540893)

[3.X - The Game 29](#_Toc7540894)

[3.X.1 - Basic Idea 29](#_Toc7540895)

[3.X.2 - Game Intensity 30](#_Toc7540896)

[3.X.3 - Game Valence 32](#_Toc7540897)

[3.X.4 - Intensity/Valence Calculations 32](#_Toc7540898)

[3.X - Questionnaire 34](#_Toc7540899)

[4 - Results 37](#_Toc7540900)

[4.1 - Previous Experience 37](#_Toc7540901)

[4.1.1 - How Often Do You Play Video Games? 37](#_Toc7540902)

[4.1.2 - Do you Play a Musical Instrument? 38](#_Toc7540903)

[4.1.3 - Gamer/Musician Relationship 39](#_Toc7540904)

[4.1.4 - How Important is Reactive Music? 39](#_Toc7540905)

[4.2 - Descriptions of the Music 40](#_Toc7540906)

[4.2.1 - At the Start of the Game 40](#_Toc7540907)

[4.2.2 - When the Light Changed to Orange 41](#_Toc7540908)

[4.2.3 - When the Light Changed to Red 41](#_Toc7540909)

[4.2.4 - When Being Chased by the Guards 41](#_Toc7540910)

[4.3 - Musical Related Events 42](#_Toc7540911)

[4.3.1 - Distribution of Answers 42](#_Toc7540912)

[4.3.2 - Player Seen by Guards 42](#_Toc7540913)

[4.3.3 - Player Close to a Guard 43](#_Toc7540914)

[4.3.4 - Player Close to a Laser Grid 44](#_Toc7540915)

[4.3.5 - Player’s Score 45](#_Toc7540916)

[4.3.6 - Player Seen by Security Camera 46](#_Toc7540917)

[4.3.7 - Player in a Secret Path 47](#_Toc7540918)

[4.3.8 - Player’s Speed 48](#_Toc7540919)

[4.3.9 - Players Number of Lives 49](#_Toc7540920)

[4.3.10 - Player Used a Teleporter 50](#_Toc7540921)

[5 - Discussion 51](#_Toc7540922)

[5.1 - Project Findings 51](#_Toc7540923)

[5.2 - Previous Experience 51](#_Toc7540924)

[5.3 - Descriptions of the Music 52](#_Toc7540925)

[5.4 - Musical Related Events 53](#_Toc7540926)

[5.3 - Problems and Solutions 54](#_Toc7540927)

[5.3.1 - 1st Order Markov 54](#_Toc7540928)

[5.3.2 - Lack of Structure 55](#_Toc7540929)

[5.3.3 - Note Divisions 55](#_Toc7540930)

[5.3.5 - Participant Count 56](#_Toc7540931)

[6 - Conclusion 56](#_Toc7540932)

[6.1 - Overview 56](#_Toc7540933)

[6.2 - Expanding the Markov 56](#_Toc7540934)

[6.3 - Other Algorithms 57](#_Toc7540935)

[7 - Appendices 58](#_Toc7540936)

[8 - References 59](#_Toc7540937)

[9 - Bibliography 60](#_Toc7540938)

# b - Table of Figures

[Figure 1: An excerpt of the score for "in C" 3](#_Toc7541017)

[Figure 2: 1st order Markov chain for the musical sequence 8](#_Toc7541018)

[Figure 3: 2nd order Markov chain for the musical sequence 9](#_Toc7541019)

[Figure 4: Graphical representation of the circumplex model of affect, showing how valence on the x-axis (pleasant - unpleasant) and intensity on the y-axis (activation - deactivation), and how they combine to produce the various emotional states (Posner, Russel, ad Peterson, 2005). 11](#_Toc7541020)

[Figure 5: The semitone variation in the seven heptatonic modes, with their spectral centroid marked. Note the decreasing spectral centroid. A-Lydian, B-Ionian, C-Mixolydian, D-Dorian, E-Aeolian, F-Phrygian, G-Locrian 13](#_Toc7541021)

[Figure 6: The Kratos Theme from God of War (McCreary, 2018) 16](#_Toc7541022)

[Figure 7: The beginning of The Imperial March, Darth Vader's Theme (Williams, 1980) 17](#_Toc7541023)

[Figure 8: Kratos' theme, his leitmotif is repeated three times, highlighted in red 18](#_Toc7541024)

[Figure 9: The first line of 'The Imperial March' with the leitmotif highlighted in red, it is then repeated (shown in blue), with some variation in pitch, however the rhythm is consistent 18](#_Toc7541025)

[Figure 10: The period structure of Kratos' theme, the basic idea outlined by red, with the two contrasting ideas in blue and green 19](#_Toc7541026)

[Figure 11: The Imperial March with its sentence structure highlighted. The first basic idea in red, the second basic idea in blue (note the tonal variation), and the continuation in green 21](#_Toc7541027)

[Figure 12: The MIDI sequence before adding rests (A), and the sequence after rests have been added into the spaces between notes (B). Blue bars represents note, orange bars represent rests. 24](#_Toc7541028)

[Figure 13: Adding rests with no minimum length (A), and where the minimum length is set to 0.25 (B) note the vastly reduced amount of rests added (from 26 to 3) 25](#_Toc7541029)

[Figure 14: The calculation for the random number weighting, using the intensity, the intensity\_weighting (how much the output of this calculation effect the random number generation, and the max\_frequency. 27](#_Toc7541030)

[Figure 15: How the MusicController calculates the time\_step 28](#_Toc7541031)

[Figure 16: The calculation to find a note's length in seconds at the current bpm 29](#_Toc7541032)

[Figure 17: The orbs the player must collect (A) and the robotic guards they try to avoid (B) 30](#_Toc7541033)

[Figure 18: State diagram for the robotic guards 31](#_Toc7541034)

[Figure 19: The default scaling values for the various event values. 34](#_Toc7541035)

[Figure 20: Intensity and Valence equations, a is each value’s scaling factor, as shown above in Figure 5 34](#_Toc7541036)

[Figure 21: The options given to the questionnaire participants when asked to describe the music 36](#_Toc7541037)

[Figure 22: Gaming experience of the participants 37](#_Toc7541038)

[Figure 23: Musical experience of the participants 38](#_Toc7541039)

[Figure 24: Combined gaming/musical experience. 39](#_Toc7541040)

[Figure 25: Participants response to 'how important is reactive music 39](#_Toc7541041)

[Figure 26: Responses when describing the music at the start of the game 40](#_Toc7541042)

[Figure 27: As Figure 5, but split by gaming experience 40](#_Toc7541043)

[Figure 28: As Figure 5, but split by musical experience 40](#_Toc7541044)

[Figure 29: Responses when describing music when the lights changed to orange 41](#_Toc7541045)

[Figure 30: Responses when describing music when the lights changed to red 41](#_Toc7541046)

[Figure 31: Responses when describing music when being chased by a guard 41](#_Toc7541047)

[Figure 32: Percentage of answer which were either correct, incorrect, or unsure 42](#_Toc7541048)

[Figure 33: Does being seen by a guard affect the music? 42](#_Toc7541049)

[Figure 34: As Figure 12, but split by gaming experience (Left), and musical experience (Right) 42](#_Toc7541050)

[Figure 35: Does being close to a guard affect the music? 43](#_Toc7541051)

[Figure 36: Shows the difference between what the application would use to calculate how close a guard was to the player (A), and how close the guard is in walking distance (B). 53](#_Toc7541052)

# b - Table of Tables

[Table 1: Frequency distribution of the musical sequence 8](#_Toc7540975)

[Table 1: A Representation of a frequency distribution. Each row represents a DependHolder with its MidiHolder in the left column and the rest of the cells being the NextNote list. The number in brackets before each NextNote is the frequency of the note pair ending in that note. The number after each DependHolder is the totalled frequencies of its NextNotes. (Note: this does not show values for note length, that is a NextNote list can contain multiple C#, but each has a different length) 26](#_Toc7540976)

[Table 2: The note generation process, in this case A is chosen. 26](#_Toc7540977)

[Table 3: The note generation process, the upper and lower limits are shown in red. Note that the rest cannot be generated in this scenario. 27](#_Toc7540978)

# d - Acknowledgements

Thanks to google and caffeine

# e - Abstract

# 1 - Introduction

## 1.1 - The problem with games

Music has always been an integral part of most art forms, from plays to blockbuster movies. When used effectively it can greatly influence how the composer wants the audience to feel at any specific moment. For example in theatrical production the composer may make the music swell when a character finally overcomes an obstacle, or become sombre at the death of a beloved character. Video games are no exception to this, but with one very important difference, the composer has no idea when a specific action will happen in the game, in essence, the composer can’t make the player press the start button.

However, the vast majority of today's games still rely on precomposed musical assets. Something clearly has to be done differently to allow a game’s music to be in sync with the events in a game. It would be fairly jarring if the main character died and the music did not change to reflect this, or if a player took longer than expected in a specific part of the game and the audio file ran its course. The game would then have to either play the same piece of music again, play a different song, or just stop playing music. Obviously, if a player plays a game for long enough they will encounter all the available music that it has to offer, which would eventually get boring.

## 1.2 - Conventional Methods

One way around this problem is to allow the game to directly influence what music is being played. This can be achieved by using a method called Horizontal Resequencing. This is when one piece of music is switched for another when a specific event occurs in the game (Phillips 2016). For example, if the player is exploring a cave, a cave theme may play, but when they leave the cave the music changes to the outside theme. However to achieve this the music would either have to abruptly change track, which could be harsh for the player if the change happens in a random point in the song. This problem can be mitigated by cross-fading the two music tracks (reducing the volume of one track, while increasing the volume of the other). Although to have this transition as unobtrusive as possible the two tracks would have to be in the same tempo (the speed of the music). This problem is amplified if the two tracks are in different keys, as this would create a musical dissonance. If the game is changing states very quickly, the music would be in danger of becoming increasingly jumbled, as the music would be comprised of random snippets of various tracks and so lose its musical integrity. Video games that include this technique are ‘World of Warcraft’ (Blizzard Entertainment, 2004) and ‘Final Fantasy XIII’ (Square Enix 1st Production Department, 2009).

Another method commonly used is Vertical Reorchestration. This is when a track is made up of several sections, which can be combined in different ways to represent the various states a game can be in. For example, a basic rhythm section when the player isn’t doing much, a more intense rhythm section when the excitement of the game increases, and an instrumental section for more emotional sections. Each of these sections can be introduced or removed depending on what is happening in the game. As the musical representation of the various states of the game is all essentially one track, rapidly changing between the states will no longer reduce the music’s cohesiveness. However, as each section has to work by itself and in any combination, this increases the workload for the composer. Some examples of video games which use this technique are ‘Red Dead Redemption’ (Rockstar San Diego, 2010), ‘Portal 2’ (Valve Corporation, 2011), ‘and Fallout New Vegas’ (Obsidian Entertainment, 2010). It would also still suffer from the problems of Horizontal Resequencing when changing to a completely different song.

For both of these methods, the more states a game has the more variation in the music there must be. Thus leading to an increase in the memory which needs to be used for storing the various music files. This is also not to say that they are mutually exclusive, as many video games will employ both techniques to some degree, an extreme example of this is ‘No Man’s Sky’ (Hello Games, 2016), which utilises more than 30 hours of raw audio files to piece together the soundtrack (Savage, 2019), although an important distinction to make in this case is that the game is not generating new music it, it is only ‘stitching’ together the audio files it is supplied (Weir, 2016).

## 1.3 - Procedural Music

The previously discussed methods for flexible game music are actually a less extreme version of one type of procedural music, that is some part of the performance of the musical piece is left up to chance, in terms of music generation this can be described as Transformational (Wooler, et al. 2005) as the underlying musical structure of a piece is unaltered, it is just the way the music combined that is changing, a more extreme version of this is “in C” by Terry Riley (1964).



Figure 1: An excerpt of the score for "in C"

This piece was designed for an undetermined number of performers, whom each has to perform each section in the order written, but each performer can independently repeat a section at any given time, while each performance is different, no new musical ideas are generated.

The second type is then Generative (Wooler, et al. 2005) is where some aspect of the composition of the piece is left up to chance and thus the amount of musical data is increased, an example of this is “Music of Changes” by John Cage (1951) which used a classic Chinese text named the “I Ching” to determine how the music would be composed. In theory, the “I Ching” could be used to repeatedly produce completely new music. As mentioned many games use a simple form of Transformational algorithms, in contrast to this there is a significant lack of games which utilise Generative algorithms (Collins, 2009), even though this process is not a new concept. One of the first games to utilise a Generative algorithm was “Otocky” (1987), a side-scrolling shooter which played a note when the player fired, the pitch of this note changed depending on the direction that the player fired, thus creating the main melody the game’s soundtrack.

## 1.4 - Aims

The main aims of this project are to develop a system which can utilise both transformational and generative algorithms to procedurally generate music, this will be accomplished by completing the following objectives;

* Research the literature about generative algorithms so the most appropriate one for the project can be chosen and then implemented.
* Research the literature about musical theory and what effect various forms of music have on a listener.
* Research how music theory and the chosen algorithm can be combined, so that the algorithm is steered in the correct narrative direction.
* Implement the chosen techniques into an application that will produce music for use in a video game scenario.
* The music produced will be able to adapt to external factors, such as what the player is doing in a game.
* To integrate the music application into a simple game, so that it can be ascertained if these problems have been solved.

## 1.5 - Overview of Remaining Chapters

Section 2 will be a summary of the various generative algorithms that have been used to produce music. It will also cover various musical techniques that can be used to lead a players emotions in specific narratively driven ways. It will also serve as a justification for the choices made during the planning process for this project.

Section 3 will be an explanation of the implementation of the chosen generative algorithm and how it utilises music theory to alter the music to fit the state of a game. It will also discuss how the generated music was assessed via a questionnaire and the relevance of the questions chosen.

Section 4 will evaluate the results obtained from the said questionnaire and the music that the program produces.

Section 5 will discuss the effectiveness of the music produced using the results of the questionnaire as evidence for this. It will also look at the project as a whole.

Section 6 will conclude this report and discuss future work.

# 2 - Literature Review

## 2.1 - Generative Algorithms

Generative algorithms appear in many forms, but at their core they work by taking the current step and then based on this pseudo-randomly generating the next step. How the algorithm generates this next step can be based on pre-composed music, which the program analyses and uses to work out the musical structure. It can also use a rule-based system, created from music theory, to generate this structure or a combination of these in a hybrid system. In the past decades, there have been a number of variants that have been utilised to produce in-game music, some of which are more appropriate than others. This section will outline some examples of these and their uses.

### 2.1.1 - Neural Networks

Neural networks were originally inspired by biological networks (Bishop, 1996), such as the human brain and the way in which it processes information with a series of interconnected processing units. It can also modify the connection between these units, thus allowing it to learn (van Gerven, 2017). Neural networks have long been used for pattern recognition, for example, if a human analysed a picture of letters it would be able to learn to identify or produce images it had not seen before.

This suggests that neural networks would be an appropriate algorithm for algorithmic music production. For example, Colombo et al. (2017) developed a system which produced music in the style of Irish and Klezmer folk, as this is the style they chose to train it on. Johnson (2017) intentionally used pieces from multiple genres so the trained network produced more ‘rounded’ music. However, in both of these experiments their chosen analysis method was to look at the distribution of notes compared to the training music. While they found that the produced music was mathematically similar they did not get human experts to evaluate how ‘good’ the music sounded, a common trend in many papers ADD MORE HERE. This is not always the case, for example, Prisco et al. (2017) utilised an algorithmic evaluation and a group of musical experts, who all had more than ten years in the music field. They found the music produced was of high quality and stylistically coherent, the experts were also able to point out flaws in the rhythmic elements of the music, giving Prisco et al. avenues for future work. However, as many of these papers’ goals are not focused on producing the music in real-time they make no mention of the time it took for their systems to be trained, produce the music, or their computational cost. This would make their use in a video-game scenario questionable.

### 2.1.2 - Genetic Algorithms

Like neural networks, genetic algorithms are also inspired by nature, however these are based on evolution. This works by generating several options, the one which best fits a specific set of criteria is chosen and the more options are generated based upon this. This can then be repeated until a satisfactory outcome is reached.

An example of this was a system created by Ostermann, Vatolkin, and Rudolph (2017) which generates a drum beat, in real-time, based on how a live band are performing. This generated a MIDI file of the players’ output and used this to choose the drum beat which fit best with the current musical solo. The musicians reported that this system was mostly successful when improvising along with a band. However this would then require some sort of pre-generated music to be present before the drum tracks could be altered, whether this was pre-composed or procedurally generated. The purpose of this system was as a practice aid for the musicians, rather than to create finished pieces.

Another way genetic algorithms have been used is to generate fitness functions, which can then numerically evaluate musical pieces, this technique was utilised by Loughran and O’Neill (2017). This would then allow for the music and the fitness functions to ‘evolve’ together. However the authors state that the music produces were not overly impressive, coupled with the fact that these generation techniques required many melodies to be generated at a time. This procedure would drastically increase the amount of processing power given over to the music production, which could have an impact on the performance of a game as a whole.

### 2.1.3 - Markov Chains

Markov chains are used for modelling a finite number of states, in terms of music this would be the notes in a given melody, and the probability of moving between them (Snodgrass and Ontañón, 2014). For example, if a small musical melody is ‘D, D, F#, D, D, D, E’, the frequency distribution matrix would be as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Next Note | | | Total Frequency |
| D | F# | E |
| Current Note | D | 3 | 1 | 1 | 5 |
| F# | 1 | 0 | 0 | 1 |
| E | 0 | 0 | 0 | 0 |

Table 1: Frequency distribution of the musical sequence

The frequency distribution is then used to calculate the chance of moving from one state to another by dividing the frequency of the new state, by the total states. For example when generating a new sequence and the current note is ‘D’, then the next note will have a 0.6 (3/5) chance to be another ‘D’, a 0.2 (1/5) to be ‘F#’ and the same chance for ‘E’. How the all of these different states interact with each other is shown below;

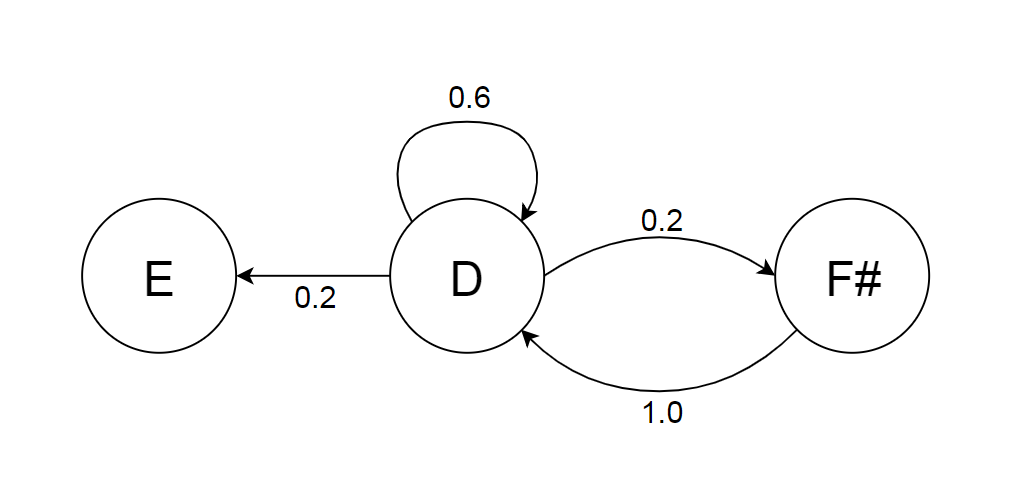


Figure 2: 1st order Markov chain for the musical sequence

This type of Markov chain is defined as a 1st order chain, as each new generated state is based solely on one step beforehand. A 2nd order chain would then be based on two steps before hand, and so on.

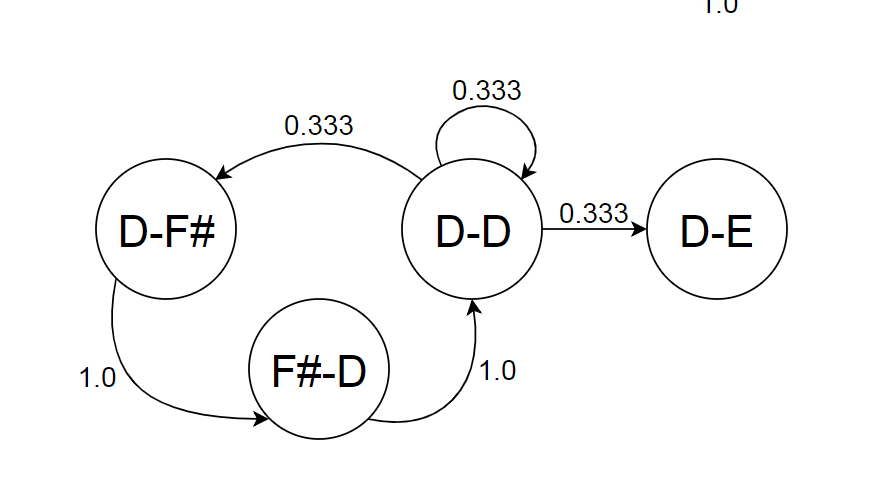


Figure 3: 2nd order Markov chain for the musical sequence

As the order increases the generated sequences have an increased level of similarity to the training data making the generated music sound less ‘random’, although this can also mean that the only possible sequence that can be generated is the trained data, which is not ideal when the purpose of the application is to generate new music.

Conversely to be too similar to the training data, a higher order chain can also produce states that were not even present in the training data, if this happens the application would require a method of overcoming this problem, such as stepping back through the data and then generating the failed sequence again (Snodgrass and Ontañón, 2013). Higher order chains may not even be worth the extra computing power that they require. Schulze and van der Merwe (2011) found that when presented with a human composed piece, music generated using a low order Markov, and one generated with a high order, the high order music was preferred the least. In regards to the randomness of the 1st order chains, Collins et al. (2011) found that generated music, based on Chopin’s Mazurkas, was difficult to discern from the real thing.

A recent example of Markov generated music was present in the game ‘Rise of the Tomb Raider’ (Crystal Dynamics, 2015) utilising the ‘dynamic percussion system’ (Intelligent Music System), this was originally developed by Brown (2012) for his PhD and utilises a combination of 1st order Markov chains, Genetic algorithms, and music theory to produce music at run-time. Following the other research on the topic, Brown found that the genetic algorithms took too long to generate music at run time, so this was done as the application loaded. While the original application was created to produce scored music for piano, it was utilised in ‘Rise of the Tomb Raider’ exclusively for the dynamic percussion, which would react to the various states that the player could find themselves in, from being completely hidden and the enemy not being aware of the player, to being engaged in combat. This dynamic drums were then set against pre-composed music (Lamperski and Tahouri, 2016).

Drums are a very common subject of procedurally generated music as they have a much lower level of variability in the number of note types that can be played (a standard drum kit has about 10 sounds, while a standard piano has around 80), this immediately reduces the complexity of any of the procedural techniques, coupled with the fact that drums notes have no musical restrictions on which notes can be played together, unlike melodic instruments. The length of a drum note also has less importance to the musical structure, as a drummer has much less control over a note’s attack and decay than a pianist. This allows rhythmic tracks to be much more ‘random’ sounding as they require a less cohesive structure than melodic sections.

## 2.2 - Creating Emotion

This sub-chapter will discuss the various musical rules which can be utilised in narrative elements of video games, and the ways in which music can be used to directly affect the player’s experience of a game.

When composing a piece of music with a particular emotional theme, the musical framework that it is built around is important for the narrative integrity of the piece. A clearer metric of the emotional state of a song is the valence and intensity (Schmidt and Trainor, 2001), these two terms are explained in the next subsections. These two attributes can be combined in varying proportions to create a wide range of emotions, for example high levels of both valence and intensity creates an exciting song, while high valence and low intensity creates a relaxing song (Cohrdes, et al., 2017). Mapping valence against intensity on a circumplex graph, first created by Russel (1978), shows the full range of emotions that can be expressed using this model, shown in Figure X.



Figure 4: Graphical representation of the circumplex model of affect, showing how valence on the x-axis (pleasant - unpleasant) and intensity on the y-axis (activation - deactivation), and how they combine to produce the various emotional states (Posner, Russel, ad Peterson, 2005).

### 2.2.1 - Valence

Valence is generally how ‘happy’ a song is perceived to be: high valence equating to high happiness. It is closely correlated with the musical concept of Brightness. This is metaphorically comparing music to luminosity: higher pitched sounds are associated with higher luminance. This was shown by Ludwig, Adachi, and Matsuzawa (2011), they required participants (both human and chimpanzee) to pick between a white or a black square while they played a note, both groups constantly associated higher pitches with brighter squares. However, this does not just apply to singular musical notes and can be applied to complete musical ideas. Bhattacharya and Lindsen (2016), showed that listening to music with a high valence made participants perceive a grey square as brighter than those who listened to music with a low valence, and Barbiere, Vidal, and Zellner (2007) found that participants associated happy songs with bright colours, and sad songs with grey.

One method of determining the brightness of a musical phrase is by calculating the spectral centroid (Schubert and Wolfe, 2005), this is the mean of the various frequencies that make up a musical idea. The various frequencies could be the resonant frequencies of a single note, when applied to the instrumentation of a piece (Eerola, Ferrer, and Alluri, 2012, and Jensen ), or all the notes in a whole musical piece. This explains why songs in a major key are commonly thought of as ‘happier’ than songs in a minor key as the spectral centroid of the minor scale is lower than the major scale’s key. This can be expanded as the major and minor scales are a subset of seven modes, these are; Lydian, Ionian (major), Mixolydian, Dorian, Aeolian (minor), Phrygian, and Locrian. These modes are made up of the looping pattern: three whole tones, a semitone, two whole tones, and a semitone, (it should be noted that a whole tone is equal to two semitones) although each mode has a different starting point, each mode also has a differing level of brightness, as shown in Fig. X

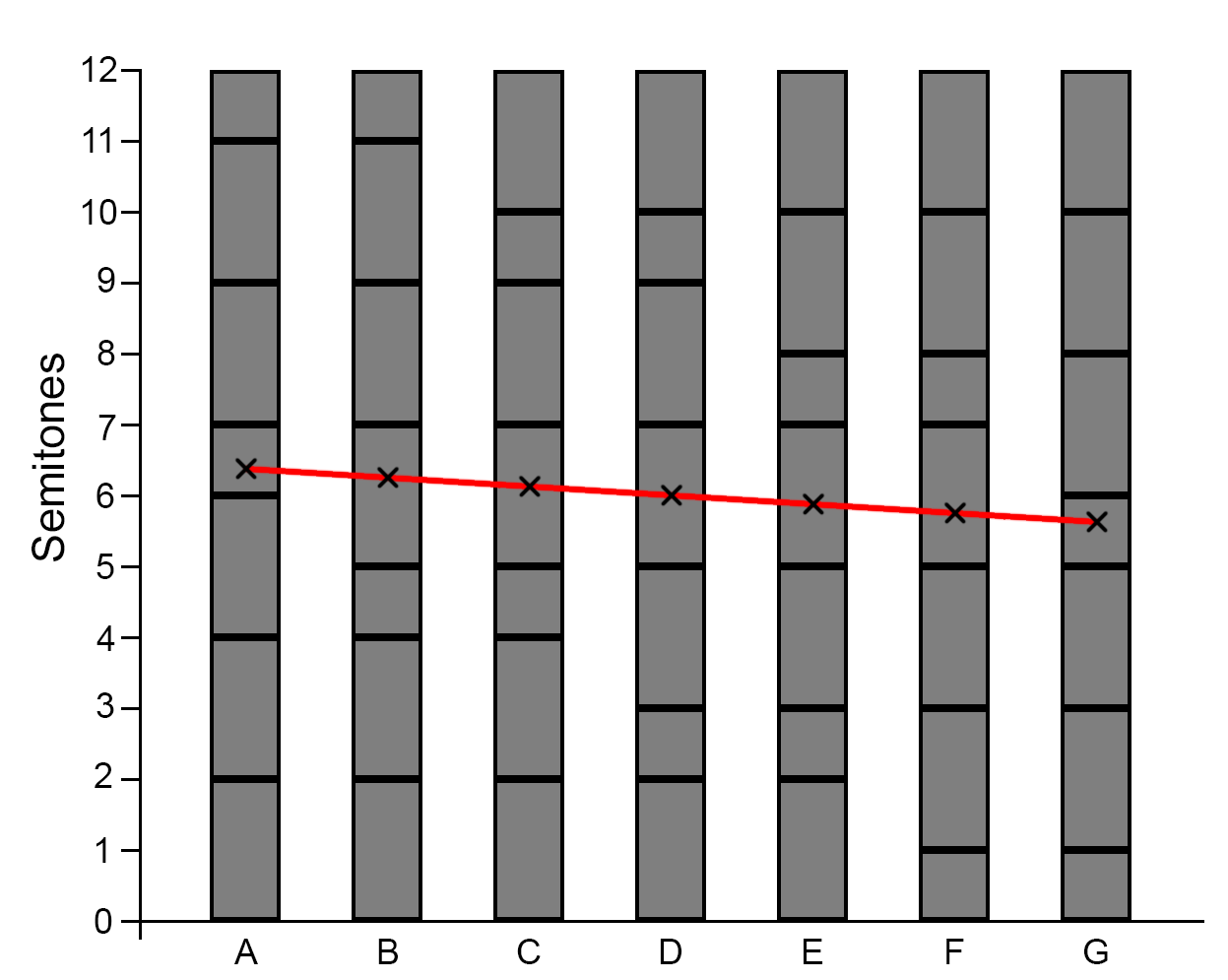


Figure 5: The semitone variation in the seven heptatonic modes, with their spectral centroid marked. Note the decreasing spectral centroid. A-Lydian, B-Ionian, C-Mixolydian, D-Dorian, E-Aeolian, F-Phrygian, G-Locrian

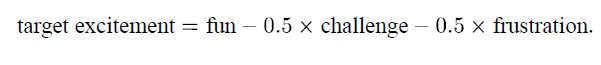
This tendency for higher pitched notes to be brighter than lower is consistent with the direction of a melody, that being a an ascending melody is perceived as brighter than a descending one (Collier and Hubbard, 2004)

### 2.2.2 - Intensity

The intensity of a piece of music mainly describes the various temporal variations that can be present (Droit-Volet, et al., 2013), such as the tempo or speed, the time of attack (the time it takes a note to reach its peak amplitude), the time of decay (the time it takes for a note to fade out), the rhythmic density (the amount of notes in a musical phrase), novelty (how much a the piece changes), and velocity (how loud a note is). The intensity of the music has been shown to directly correlate with the perceived arousal of the listener, or how alert they are (Dean, Bailes, and Schubert, 2011), although the energy (time of attack/decay, rhythmic density and velocity) has a greater influence than the tempo (North, Krause, and Sheriden, 2018). This was also shown in a study by Bramley, Dibben, and Rowe (2016), as they found that that tempo had little effect on the speed at which participants gambled. However, Mikutta, et al. (2013) found that tempo variations over the course of the piece would indeed increase arousal.

### 2.2.3 - Game Play as the Composer

Regardless of the method chosen to generate music, there needs to be some way that a game actively control the valence and/or the intensity of the music, an example of this is experience driven procedural content generation (EDPCG). Plans and Morelli (2012) used this for an infinite Mario Bros level generator, where they associated various actions in the game with frustration (time standing still and dying), challenge (alive time, time ducked), and fun (running time, coins collected, and monsters killed). These three variables were then combined using the following formula:



This excitement value was then used to control aspects of the music generation, such as the tempo, sparseness, and novelty (i.e. the intensity) and the musical scale used (i.e. the valence). While the authors found that the use of this system increased players’ enjoyment, they admit that they did not get enough testers to come to significant conclusions.

A similar experiment by Chan et al (2017) looked into how EDPCG can be used to inform the player of how well they are doing in a game. The game created to test this consisted of the player trying to find specific objects while avoiding hidden enemies. For this Chan et al used significantly less in game events to control the music, however this was to make the cause of the changes in the music more apparent to the player. They used; Tension (enemy proximity, low number of items collected, and low player health) and Progress (collectable proximity and high number of items collected). While this experiment also suffered from a low number of testers, the results found suggest that this method of procedural music control does indeed inform the player of how they are playing a game.

Another way that gameplay can affect the music generation is shown in an experiment by Mauceri and Majercik (2017). In this they created a music system based on a swarm algorithm reacting with a live performer playing a traditional musical instrument. The swarm would react to aspects of the performance such as; the amplitude would regulate the number of swarm members, the pitch would affect the general location of the swarm, and the musical cohesion would affect the swarm’s cohesion. The swarm would in turn interact with a granular synthesiser to produce the procedural music. For example, the difference in swarm member locations would determine the length of notes produced, and the mean Z location would affect the amplitude. While this method may not be appropriate in most game scenarios it would be interesting to have a similar system for a game where the antagonist was some sort of swarm or for it to be affecting the in-game wildlife.

## 2.3 - Musical Structure

### 2.3.1 - Theme

Another way to elicit an emotional response from music is to use themes (McCreary, 2018), these are ‘a prominent or frequently recurring melody’ (Oxford Dictionary, n.d.). These are used to musically represent characters and/or ideas, for example in ‘*God of War*’ (SIE Santa Monica Studio, 2018) the main character, Kratos, gets his own musical theme;

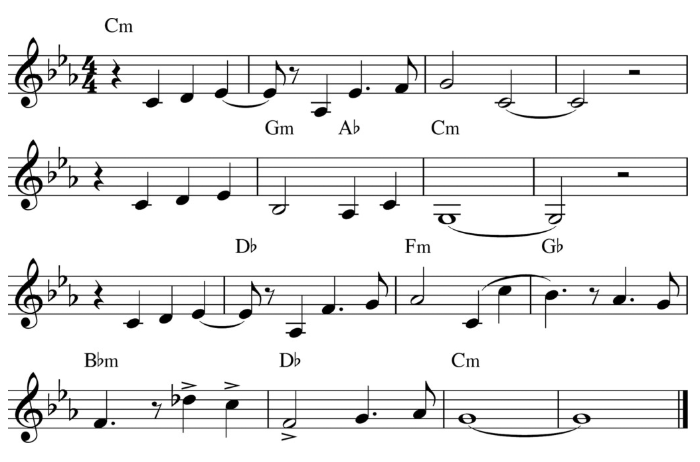


Figure 6: The Kratos Theme from God of War (McCreary, 2018)

In the game this theme is played, in some form, whenever Kratos appears on screen or his deeds are mentioned by other characters. As other characters in the game also get their own themes it allows for the music of a game to reflect what is happening in game, such as two themes being combined if two characters are interacting. A more famous example is featured in Star Wars Episode V ‘*The Empire Strikes Back*’(Kershner, 1980) where  ‘*The Imperial March*’ is introduced as the theme for the film’s antagonist, Darth Vader.



Figure 7: The beginning of The Imperial March, Darth Vader's Theme (Williams, 1980)

This theme mainly uses the Phrygian scale (low valence) to convey that Vader is the villain, coupled with the song’s short, loud notes (high intensity) it can be seen from Figure 4 that the piece conveys a stressful/tense feeling. Themes can also be used to show a character’s story arc. To continue the previous example of ‘*The Imperial March*’, in Star Wars Episode I ‘*The Phantom Menace*’ (Lucas, 1999), Vader is shown as a young innocent child, Anakin Skywalker. Here the theme uses the same chords as ‘*The Imperial March*’, but in their major variation, showing that while it is the same character he is not the villain yet.

### 2.3.2 - Leitmotif

A theme can be broken down into a short phrase know as a leitmotif, this can be used when playing a character’s entire theme may not be appropriate, but they work in the same way a theme does in terms of narrative meaning. For example Kratos’ leitmotif is the first three notes of his theme;



Figure 8: Kratos' theme, his leitmotif is repeated three times, highlighted in red

As is shown in Figure X the leitmotif is repeated three times in his theme to make it obvious to the listener that this is the case, in the full score for this piece the bass line is also playing the leitmotif as well. For ‘*The Imperial March*’ the leitmotif is also the first line;

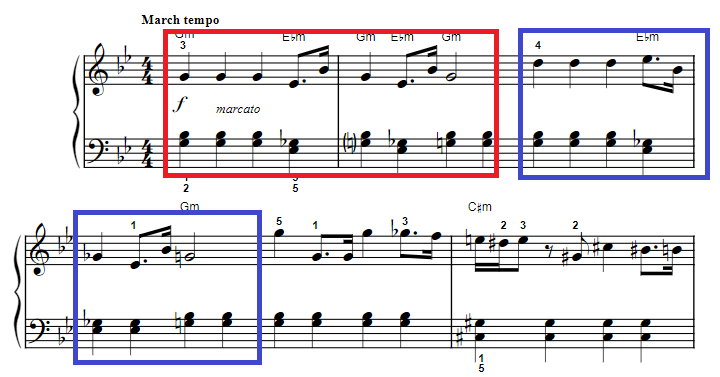


Figure 9: The first line of 'The Imperial March' with the leitmotif highlighted in red, it is then repeated (shown in blue), with some variation in pitch, however the rhythm is consistent

A procedural application would then have to have some sort of procedure for generating a theme, this could be taken directly from the output of the procedural technique chosen, however as shown above leitmotifs are often repeated throughout a theme. When composing a theme there are two main structures that are used, the Period and the Sentence.

### 2.3.2 - Period

<http://openmusictheory.com/period.html>

The Period is split up into four sections, the first two are collectively named the Antecedent, while the second two are the Consequent. Each of these are then made up of two parts, a basic idea and a contrasting idea. For both the Antecedent and the Consequent the basic idea is the same, although some tonal variation can happen, and the contrasting ideas are both different. An example of this is Kratos’ theme as can be seen from Figure X below.

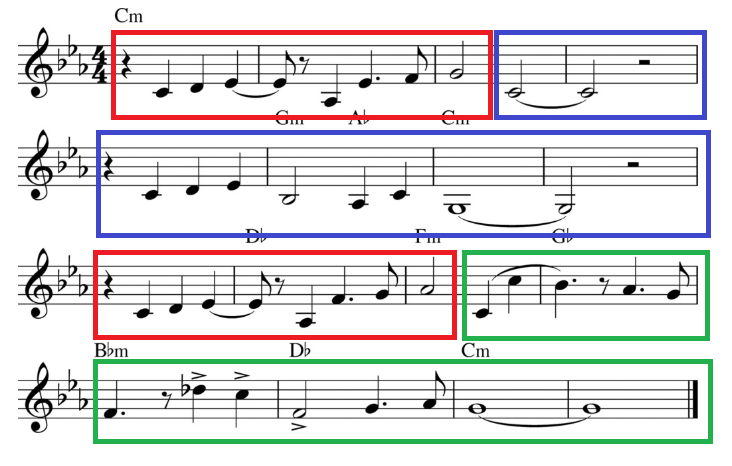


Figure 10: The period structure of Kratos' theme, the basic idea outlined by red, with the two contrasting ideas in blue and green

Bit about how the chord patterns are structured, end of antecedent is weak cadence, end of consequent is strong. Although im not doing this, cause time contraints.

### 2.3.3 - Sentence

<http://openmusictheory.com/sentence.html>

The Sentence is of similar idea, however in the first half, named the Presentation phase, the basic idea is repeated twice, and again the repetition need not be homogenous with the first. The second half, named the Continuation, expands upon the ideas given in the basic idea, and leads the piece to its conclusion. The Imperial March is an example of this type of structure, shown below in Figure X

Add bit about fragmentation/liquidation/those things?



Figure 11: The Imperial March with its sentence structure highlighted. The first basic idea in red, the second basic idea in blue (note the tonal variation), and the continuation in green

## 2.4 - Summary

The chosen procedural technique for this project was 1st Markov chains as they are well suited to pattern replication. This was used for melodic and rhythmic sections. The main melodic sections can be generated either in Period or Sentence form. EDPCG was used to control the intensity and valence of the produced music. For the melodic sections a set of rules were created to make sure the composed melodies were in the correct scales (decided upon by the valence). A simple game was produced to allow enough scenarios to showcase that intensity/valence variation.

NEED TO SAY THIS WILL BE EXPLAINED IN METHODS?

ALSO REWRITE THIS TO MAKE IT SOUND BETTER

# 3 – Methodology

The application was developed in the game engine Unity and written in C# using Visual Studio 2017.

it is a proven technology used to create a wide range of commercial titles

These were chosen for the project as they were the most accessible for the developer and as the development of a simple game was required this was the most time efficient platform to use.

The main class for the project is the MusicController, this contains the rules and information for the music theory. It also keeps track of the current scale that the music is to be generated in.

## 3.1 - Reading in MIDI files

The MIDI file format was chosen to store the musical data as it has a low file size and the appropriate data can be easily stored and accessed, with minimal impact on the memory allocation of a game. MIDI stores musical data as a series of events, such as: when a note is played (note on), when a note is stopped (note off), and if the music’s tempo changes, etc. While not always the case, it is convention to have each channel only contain the information for one melody. As there is no simple way to differentiate between multiple melodies sharing a channel and one melody that consists multiple notes playing at the same time, one melody per channel was assumed to be the case for this project.

To read in the MIDI files the NAudio library was used to access the event data. To do this the application retrieves all the events from a specified file and channel, and then loops through these looking for the ‘note on’ events. The ‘note on’ event was the only event that was considered as all the relevant musical data can be inferred from them. The other events consist of musical structure which will be controlled by the game events (see below) and so would not be used in the analysis. The relevant musical data extracted from these events was the time the note starts, its length, and its pitch. The note’s pitch is represented as an integer, i.e. C4 (261.63 Hz) is given the value of 60 and C#4 (277.18 Hz) is 61. When ‘note on’ is found its information is extracted and added to a MidiHolder class. Unfortunately, the NAudio library does not retain the length of each note in an easily accessed form, however it does retain it at the end of ‘note on’ events name, this string is then parsed and the length is extracted and then converted to a float. To make changing the musical mode easier (see section X below) each note’s pitch gets reduced down to the key of C (if original pitch is A, each note’s pitch is subtracted by 9, the semitone difference between A and C). This would also allow multiple songs to be combined at the read in stage.

As the MIDI files can be created from live performances, notes that would be equal in standard music notation can have discrepancies in their length and so would be counted as different notes when the frequency distribution is calculated (see below). To remedy this, each note’s length is rounded to specified minimum number, this is defaulted to 0.25 and then the note is added to a list. As MIDI files have no need to record the musical information for rests, as they just use a lack of notes playing for this, these need to be added so they can be included in the analysis process (see below). This was done by looping through the list of extracted MidiHolders and checking if the time a note stopped was the same as when the next note started, if this was not the case a rest note was created at this point (shown below in Figure X), to specify that this note was a rest its pitch was set to -1.

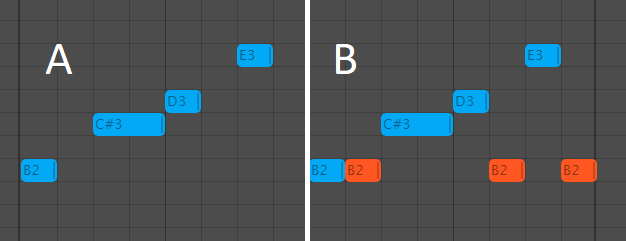


Figure 12: The MIDI sequence before adding rests (A), and the sequence after rests have been added into the spaces between notes (B). Blue bars represents note, orange bars represent rests.

The application then checks the length of the created rest, if this is greater than a specified amount (default value is 0.25) the rest is added into the list of MidiHolders at the correct position, if it is not greater than this, the rests length gets added onto the previous notes length. This is to combat MIDI files that have not been created properly, as shown below in Figure X the majority of the notes are 0.25 in length, but spaced at 0.5 and so a 0.25 length rest is created between most notes. When this file is used for a Markov chain, each note (state) would then always go to a rest, and that rest could then go to essentially any note creating a much more disjointed piece of music.

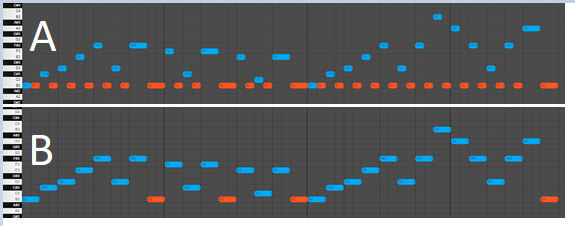


Figure 13: Adding rests with no minimum length (A), and where the minimum length is set to 0.25 (B) note the vastly reduced amount of rests added (from 26 to 3)

Once all the required rests have been added to the list of MidiHolder, the sequence is then looped through again to check for chords, that is when multiple notes start at the same time, if this is the case they are moved from the sequence to the first note’s list of pitches.

## 3.2 - Markov Chains

### 3.2.1 - Frequency Distributions

The next step in the process is to calculate the frequency distribution of the note pairs from the MidiHolder list. Two classes were created to retain the data for this, NextNote and DependHolder. NextNote contains a MidiHolder which saves the data from the second half of the note pair and the frequency that the pairing has occurred. Each DependHolder also contains an instance of a MidiHolder and a list of NextNotes. The frequency distribution is created with a list of DependHolders. The application loops through the sequence of notes and checks if each note has occurred in the MidiHolder of each DependHolder. If this is not the case, there is a new note pair and so a new DependHolder is created, and its MidiHolder is then set to the current note. The next note in the sequence is then added to the new DependHolder’s NextNote list, with a frequency of 1. Conversely if the current note was found in the DependHolders it then checks if the next note in the sequence has occurred in the NextNote list of that DependHolder, if it has it increments the frequency of that NextNote by 1, if not it adds the note to the NextNote list. This creates a data structure, an example of which is shown in Table X

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **MidiHolder** | **NextNotes** | | | |
| **DependHolder (16)** | D | (1) REST | (4) B | (2) C# | (9) A |
| **DependHolder (8)** | C# | (3) A | (1) F | (4) REST |  |
| **DependHolder (7)** | REST | (4) B | (3) G |  |  |

Table 2: A Representation of a frequency distribution. Each row represents a DependHolder with its MidiHolder in the left column and the rest of the cells being the NextNote list. The number in brackets before each NextNote is the frequency of the note pair ending in that note. The number after each DependHolder is the totalled frequencies of its NextNotes. (Note: this does not show values for note length, that is a NextNote list can contain multiple C#, but each has a different length)

For each DependHolder it then sums the frequencies of each of its NextNotes and then sorts them by length into numerical order, this is so the note selection process can be waited towards shorter notes (see section X).

### 3.2.2 - Choosing Notes

The function to generate a new note takes the frequency distribution and a MidiHolder as inputs. Although, it first gets the current intensity of the game from the MusicController and the intensity weighting, this is the amount the intensity will affect the note choosing process (discussed below). It then loops through the frequency distribution and checks to see if the inputted MidiHolder has occurred, if it has it then goes onto to choosing the new note to generate. It first generates a random number between 0 and the summed frequency count (from the DependHolder) and checks to see if the frequency of the first note in the NextNote is greater/equal than this, if not it adds the next NextNote’s frequency to this and checks again. This continues until the running frequency count is indeed greater than the random number. This is shown below in Figure X, where ‘A’ is chosen as the new note to be generated.

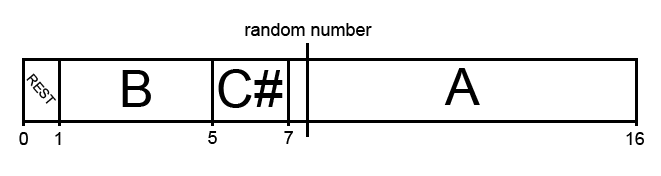


Table 3: The note generation process, in this case A is chosen.

To allow the intensity of the music to influence this process, a weighting is calculated using the following formula:

random\_weighting = -1 \* intensity \* intensity\_weighting \* max\_frequency

Figure 14: The calculation for the random number weighting, using the intensity, the intensity\_weighting (how much the output of this calculation effect the random number generation, and the max\_frequency.

The lower limit of the random number generation is then set to this random weighting, while the upper limit is set to maximum frequency plus this. Both limits are then clamped between 0 and the max frequency. This causes the random number generation to be weighted towards low numbers when the intensity is low (and so negative) and high numbers when its positive, this is shown in Figure X. Coupled with the fact that the NextNote are ordered by their length. This causes the note generated to more likely be longer when the intensity is low and shorter when the intensity is high.

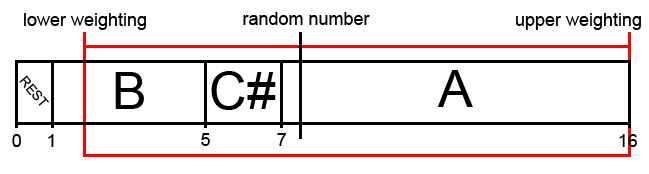


Table 4: The note generation process, the upper and lower limits are shown in red. Note that the rest cannot be generated in this scenario.

If this process goes through the entire frequency distribution and does not find a match, it tries this previously discussed process again, but ignoring the length of notes when comparing them, this causes the process to have a higher likelihood of finding a match, at the cost of musical integrity. If this also fails to find a match, a random note is chosen from the frequency distribution.

### 3.2.3 - Creating Structured Melodies

As both Period and Sentence structures (discussed above, see section XX) are made up of four short phrases, two of which are the same, three phrases are generated. This process uses the Markov chains process to generate notes (discussed above, see section [3.2.2 - Choosing Notes](https://docs.google.com/document/d/1-eXPu0vgbwyuGxYmOtao0lSkEjurGrp4qaqC3nrLUo4/edit#heading=h.2et92p0)), while it is doing this it keeps track of the current temporal length of the phrase, once this length exceeds a specified number, the generation stops and the last note’s length is reduced so the total length of the phrase complies with the specified maximum length. The three phrases are then added to a list of MidiHolders, the order this happens in depends on the structure used. For example, if the three phrases are named A, B, and C, Period structured would be ABAC and Sentence would be AABC.

### 3.2.4 - Playing the Music

The main concern of the application in terms of playing notes is keeping them in the correct timing. To keep track of the time for this, the application updates a variable using the MusicControllers ‘time\_step’, this is the length in seconds that the shortest note length (defaulted to 0.25 beats) takes at the current bpm, the calculation is as follows:

time\_step = 60 / (bpm / shortest\_note\_length)

Figure 15: How the MusicController calculates the time\_step

By comparing this against time from Unity’s inbuilt AudioSettings, the application determines if it needs to play a note, this is done by checking a counter against the current note’s length, this counter is also increased by the ‘shortest\_note\_length’ and reset to zero when a note is played. To stop the music going out of time if the bpm changes while a note is playing, each note is passed to a coroutine to be played.

This starts by checking if the note is a rest (if it is the pitch will be -1), if this is not the case it will continue. It then retrieves the necessary information from the note to be played (pitch and length). The pitch is then corrected, based on the current mode (discussed below, see XX). This is done by first getting the ‘base version’ of the note, which consists of subtracting 12 from the note’s pitch until it is below 12, for example if this process is done to C4, its note number of 60, the resulting ‘base’ note is 0, for D7 with a note number of 98 would be reduced to 2. The number of times 12 has to be subtracted is also saved as the amount the note’s pitch has to be reduced by. As the musical scales are in the range of 0 to 11, this base note is used to find the closest note in the scale, numbers that are in the scale are used immediately while numbers that are not are increased by 1. For example, if D is used (base note of 2) and the current scale is Locrian, D (i.e. 2) is not present in this mode so it is increased to D# (i.e. 3).

Once the base has been converted into the correct mode, it is then raised back into the correct pitch by adding 12 multiplied by the pitch shift. This new note is then passed to the synthesiser to be played. The note’s length will then be converted into a time in seconds using the following formula;

time = note\_length \* (60 / bpm)

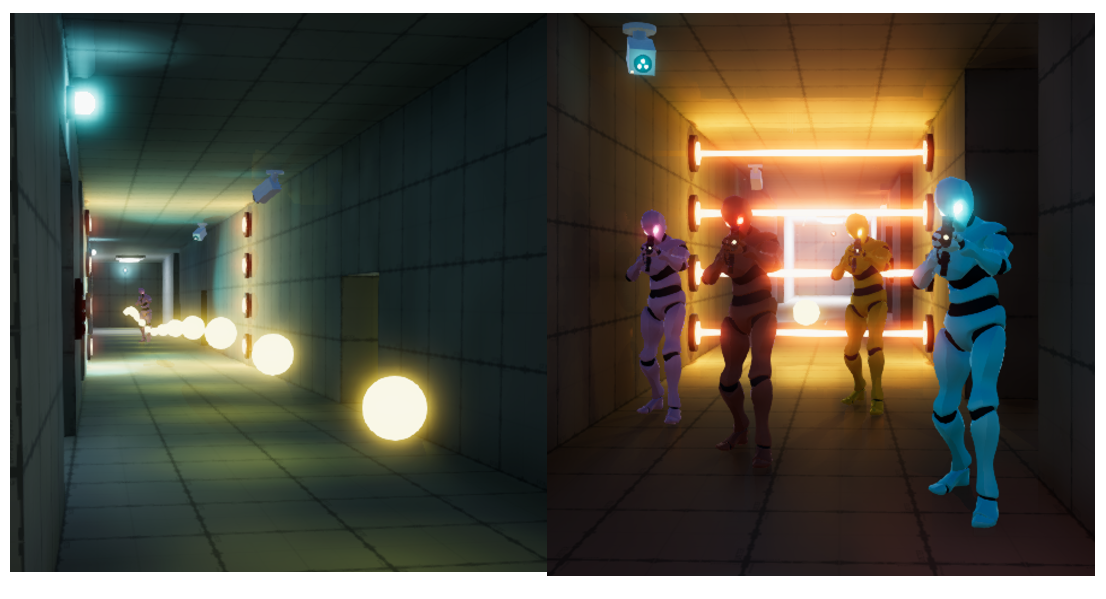
Figure 16: The calculation to find a note's length in seconds at the current bpm

The coroutine is then paused for the calculated length of time. Once this time has passed the synthesiser is then told to stop playing the note. The next note is then chosen, if the structured music is currently being played the next note is taken from the list of progenerated notes, if not then the current note is passed to the ‘choosing notes’ process (see above in section [3.2.2 - Choosing Notes](#_3.2.2_-_Choosing))

## 3.X - The Game

### 3.X.1 - Basic Idea

The first objective for designing the game was to determine the most appropriate genre which would provide enough variability in the gameplay to give the music generation application sufficient space to showcase how it can modify the produced music. The game that was chosen was a first-person stealth game, inspired by Pacman (XXXXXXXXXXXXX, 19XX), the player is tasked with collecting 200 orbs in a sci-fi style facility, while trying to avoid robotic guards, shown below in Figure X



A B

Figure 17: The orbs the player must collect (A) and the robotic guards they try to avoid (B)

### 3.X.2 - Game Intensity

The first way the game varies the intensity is the player’s movement speed, the player has three options for this; standing still, walking, and running, each having a larger effect on the intensity than the one before.

The player’s interactions with the guards also influences the intensity, the closer they are to the guards the higher the intensity is. The guards also have four states they can be in (ordered in decreasing intensity);

* Hunting - The player’s position is always known to the guards and they actively track them down.
* Chasing - When they see the player the guards will chase after them, until they can no longer see them.
* Searching - When they lose sight of the player they will go to the player’s last known location, when they reach this position they will turn in a circle to see if they can locate the player again.
* Patrolling - This is the default state where they follow pre-set paths around the facility.

These states interact with each other according to the following state diagram (Figure X).

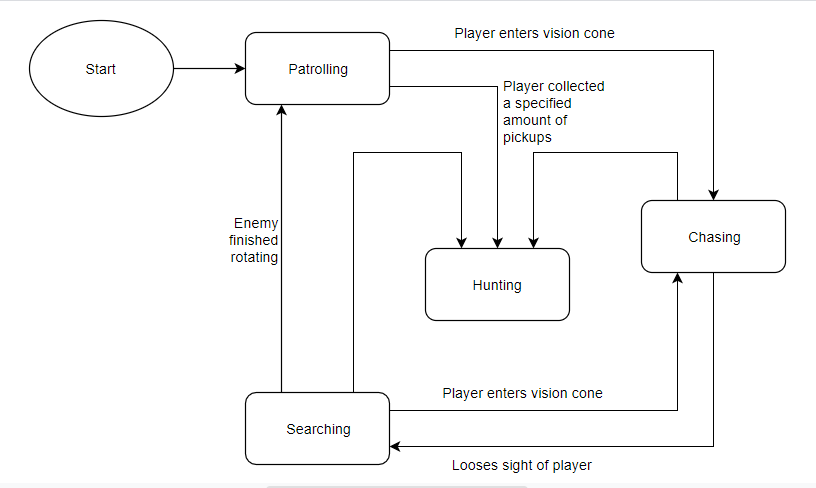


Figure 18: State diagram for the robotic guards

The amount of pickups the player has collected also has a positive effect on the intensity, the more they collect the higher the intensity. Other factors affect the intensity of the game, although they do not directly affect the music they make the game harder to complete and so the player will get more stressed. The first of which is that the player must also look out for security cameras (top left in Figure X - B), when they see the player they activate a laser grid which will block the player’s path (again shown in Figure X - B). The second is when half of the orbs are collected the facility goes into ‘lockdown’ mode, which causes the laser grids to be activated for the rest of the game, this change is indicated by a dialogue line and warning lights (Top right if Figure X - A) which change from their default blue to orange. When 80% of the orbs have been collected the facility goes into ‘hunting’ mode, indicated by another in-game dialogue announcement, the warning lights turning red. The robot guards at this point actively hunt down the player by permanently being in their ‘Chasing’ state.

### 3.X.3 - Game Valence

The first variable that affects the music’s valence is the number of orbs collected as this is the main way the player will track their progression, the higher this is the higher the valence is. The second is the number of lives the player has, starting with three and each time they collide with a guard they lose a life, and the valance is decreased. The state the guards are in affects the valence in a similar way that it affects the intensity, although for valence it has a negative effect.

### 3.X.4 - Intensity/Valence Calculations

To allow the various events in the game (described above) to affect the music, their value must first be calculated. This is done by scaling each value to be between 0 and 1. The exception to this, is the speed factor which is ranged from -1 to 1. These values are calculated as follows:

* Progression - The ratio of collected orbs to the total number that available.
* Guard Proximity - The distance between the player and the closest guard, this value is then scaled down by a minimum and maximum distance, clamped between 0 and 1, and then subtracted from 1.
* Guard State - The guard’s state is stored as an enum and so the integer value is used (patrolling = 0, hunting = 3), the highest of which is used, by dividing it by 3.
* Lives Left - The player’s current lives are divided by the starting number of lives.
* Speed - If the player is still the speed variable gets set to -1, walking sets it to 0, and running to 1. This value is then interpolated between these values so the change in the speeds affect on the intensity is smooth.

The values are then each multiplied by a specific scaling factor, so they have the desired effect on the intensity and valence. The default scaling values are shown below in Figure X. However, these values can be change from the Unity inspector to better suit what a developer wants to form the system.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Intensity** | **Valence** |
| Progression (P) | 1 | 1 |
| Guard Proximity (Gp) | 1 | 0 |
| Guard State (Gs) | 1 | -1 |
| Lives Left (L) | 0 | -1 |
| Speed (S) | 1 | 0 |

Figure 19: The default scaling values for the various event values.

The intensity and valence targets are then calculated using the following equation:

I = (a1S + a2Gp + a3Gs + a4P) / 4

V = (a5P + a6L+ a3Gs) / 3

Figure 20: Intensity and Valence equations, a is each value’s scaling factor, as shown above in Figure 5

The final intensity and valence are then modified with delta time so the transition between states is not too jarring (the speed at which this happens can also be set in the inspector). These values are then clamped between 1 and -1.

## 3.X - Questionnaire

The main aim of the questionnaire was to determine the effectiveness of the application. For ease of access the questionnaire was created using a Google form, so it could be sent to participants (the complete form can be found below in Appendix XX). The first section is on the previous experiences of the participants, this asks how much time they spend in a week playing video games, as video games are a common source of procedural/reactive music. It was hypothesised that people with this previous experience would be able to hear the effects with greater ease. For this same reason participants were also asked if they played a musical instrument, as they would be more familiar with changes in music. They were also asked if they thought reactive music was important.

They are then asked to download and play the game (a Windows and Mac build is supplied for ease). After they have done this, they are asked to describe the music at four points in the game: at the start. when the warning lights turn orange, when the lights then turn red, and when the guards are chasing them.

They are given seven options taken from the circumplex model (see section XX, Figure X), if they choose other they are given the option to write how they would describe the music in their own words, shown below in Figure X.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exciting | Stressful | Happy | Sad | Relaxed | Boring | Other |

(Optional) If other, how would you describe it?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Figure 21: The options given to the questionnaire participants when asked to describe the music

To ascertain if the participants could tell which events were causing the music to change they were shown nine distinct events and asked if they affected the music. These included the five events which did affect the music (discussed above, see section XX) and the remaining four had no direct effect on the music, these four were as follows;

* The player is close to a laser grid
* If a security camera has seen the player
* The player is inside a secret passage
* If the player has used a teleporter

To make sure the participants understood which event they were being asked about, a picture of the object in question was supplied. This was not done for the questions on the number of lives, speed of the player, and the score as these events contained several temporal elements and so a picture of it could not be easily obtained.

The participants were then asked to give more general feedback on the music and the game itself, to show the main way the application could be improved if worked on further.

# ­­4 - Results

## 4.1 - Previous Experience

### 4.1.1 - How Often Do You Play Video Games?

Figure 22: Gaming experience of the participants

### 4.1.2 - Do you Play a Musical Instrument?

Figure 23: Musical experience of the participants

### 4.1.3 - Gamer/Musician Relationship

Figure 24: Combined gaming/musical experience.

G/M = gamer and musician, NG/M = non-gamer and musician, G/NM = gamer and non-musician

### 4.1.4 - How Important is Reactive Music?

Figure 25: Participants response to 'how important is reactive music

## 4.2 - Descriptions of the Music

### 4.2.1 - At the Start of the Game

Figure 26: Responses when describing the music at the start of the game

Figure 27: As Figure 5, but split by gaming experience

Figure 28: As Figure 5, but split by musical experience

### 4.2.2 - When the Light Changed to Orange

Figure 29: Responses when describing music when the lights changed to orange

### 4.2.3 - When the Light Changed to Red

Figure 30: Responses when describing music when the lights changed to red

### 4.2.4 - When Being Chased by the Guards

Figure 31: Responses when describing music when being chased by a guard

## 4.3 - Musical Related Events

### 4.3.1 - Distribution of Answers

Figure 32: Percentage of answer which were either correct, incorrect, or unsure

### 4.3.2 - Player Seen by Guards

Figure 33: Does being seen by a guard affect the music?

Figure 34: As Figure 12, but split by gaming experience (Left), and musical experience (Right)

### 4.3.3 - Player Close to a Guard

Figure 35: Does being close to a guard affect the music?

### 4.3.4 - Player Close to a Laser Grid

### 4.3.5 - Player’s Score

### 4.3.6 - Player Seen by Security Camera

### 4.3.7 - Player in a Secret Path

### 4.3.8 - Player’s Speed

### 4.3.9 - Players Number of Lives

### 4.3.10 - Player Used a Teleporter

# 5 - Discussion

## 5.1 - Project Findings

The findings (shown above in section 4) show that the project was successful in fulfilling the aims set out in section 1.4 - Aims. The chosen techniques do indeed create music that is appropriate for a video game and, with its integration into a simple game, it was able to adapt to external factors. The fact that a clear majority of participants (~94%) agreed that it is important for music to adapt to the events of a game puts this area of research in a favourable light.

The following subsections will discuss what went well in the project, and the areas that did not work as well and how they could be improved.

## 5.2 - Previous Experience

18 participants were involved in the study, this was subdivided by two factors, gaming experience and musicianship. With the majority being gamers (Figure X) and non-musicians (Figure X) respectively. However, when these groups were combined the largest subgroup were the participants who were both gamers and musicians, at ~39%, closely followed by people who were neither, at ~33% (Figure X). It is noted however that further correlation between the answers given by gamers and musicians will be influenced by this.

An immediate difference shown amongst those with musical experience was that 50% of musicians mentioned tonal differences in their answers (i.e. the valence controlled changes). For example, when a guard saw them one musician said that the music “…*became darker…*”, or another participant described the musical change caused by collecting more orbs as “*…happier sounding music*”. This contrasted with the non-musicians who mainly mentioned the tempo changes, only 25% of them mentioned tonal differences. This may be due to musicians having a larger knowledge base of music theory, as it is an integral part of learning to play an instrument.

Another difference between those with musical experience and those without came from the fact that some of the in-game events have associated diegetic sound effects attached to them. This caused some of the participants to incorrectly say that an event was affecting the music, when in fact it was just playing a re-recovered sound clip. For example, when asked if being close to a laser grid affected the music, one participant answered that they “*Could hear the sizzle of the laser*”. Or, when using the teleporters, another said “*There was a sound effect attached to moving through it*”, and when seen by the cameras, “*sometimes added a "motion detected"* ”, incorrectly ascribing the guards’ sound effects (or the dialogue prompt when the lights turn orange) to the camera seeing them.

## 5.3 - Descriptions of the Music

At the start of the game, the intensity and valence are both set to neutral levels (both are around zero), the resultant music is then neutral in terms of its emotional state. This was supported by the even spread of answers when participants were asked to describe this, with the sad answer being the only option that was not chosen (shown in Figure X). Interestingly, there was a clear divide between those with and without gaming experience, as gamers chose emotions with lower intensity (happy, relaxing, and boring), while non-gamers chose from a wider range of emotions (also choosing the higher intensity choices; exciting and stressful), this was attributed to the fact that gamers are likely to have more experience with stressful games and so would not find the start of the game as stressful, as it was arguably the easiest part of the game.

The two groups were in more agreement when describing the music for the next three questions. When the lights changed to orange, which marks the halfway point of the game, almost two-thirds of participants described the music as exciting which lines up with the in-game circumplex graph. This is also in correlation with the in-game circumplex when the lights turn red and when the player was being chased by the guards as almost all the participants described these events as either stressful or exciting, with the majority choosing stressful (~70% for both).

## 5.4 - Musical Related Events

As shown in Figure X ~41% of participants correctly identified the events that had an effect, however, ~33% of answers were unsure. The events which caused the most doubt were as follows; being seen by a security camera, being close to a guard, and being inside a secret path. When talking to the participants these three events were also brought up as the most confusing for them.

It made sense to the participant that as the guards seeing them had an obvious effect (~78% of participants answered yes) then it would follow that the cameras seeing them would also have an effect (~38% of participants thought this to be the case).

One problem that was identified was that the proximity of a guard was calculated using the Pythagorean distance between the guards and the player, it then did not take into account the shape of the map. A participant could be very close to a guard (according to the application), but the distance the guard would have to travel to reach the player could be much higher, as shown below in Figure X.

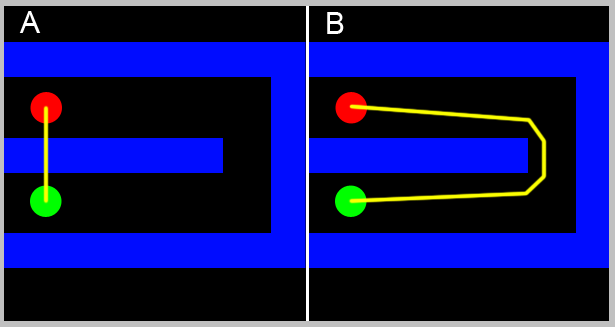


Figure 36: Shows the difference between what the application would use to calculate how close a guard was to the player (A), and how close the guard is in walking distance (B).

And for the secret passageways, the responses are most likely a result of the player’s speed affecting the music. As the majority of players that were observed did not change their speed for most of the game this caused the lack of a clear answer for the impact speed had on the music. However, the one place participants did consistently change speed was in the secret passageways, this was either to allow them to easily navigate through the smaller passageways or to let a guard pass by.

One interesting difference between subgroups occurred when the participants were asked if the players’ score had an effect on the music. While the majority (~55%) correctly said that it did have an effect on the music, this group was almost entirely made up of the gamers, ~82% of gamers answered yes, while the majority of non-gamers answered not sure (~71%). This is most likely due to games having a heavy focus on getting a high score, and so gamers would be more attuned to this having a positive effect on the music.

Another point of note is that initially the teleport event was defined as an event which had no effect on the music. While this is technically true, using the teleporter can directly affect either the intensity or the valence: using the teleporter will change the player’s position relative to the guards and so the proximity to guards will change, thus changing the music

## 5.3 - Problems and Solutions

### 5.3.1 - 1st Order Markov

One of the main issues with the project was that only a 1st order Markov chain was implemented. This would occasionally cause the generated music to sound fairly random, if the data set it was given was too long. This was a similar issue to the problem of adding too many rests (discussed above, see section XX). Too much variability in the inputted song causes each state to have too many possible next states. For this reason, it was chosen to limit the analysed music to one file, as having more would increase the variability between notes, as discussed above, having a higher order chain would then allow for increased amounts of data to be used when creating the frequency distribution.

Following on from this the song that was used had too much pitch variation, as each section of the song contained at least one 0.5 beat rest, the generated melody could jump between high and low sections. This was noticed by some of the participants, one noted there was a ‘*high pitched off key series of notes*”, when a guard saw them, even though the change in the game state would not specifically cause this effect, the application is then benefiting from humans’ ability to see patterns in places there are not [REFERENCE], although as stated this effect was not a deliberate one and so did not have the consistency to have a specific cause and effect, as shown by another participant who wrote “*…other parts (like the really high notes) seemed to come out of nowhere*”.

### 5.3.2 - Lack of Structure

The above-mentioned problem of the music sometimes sounding too random, also caused there to be a lack of overall structure in the music. This made it difficult to detect when the valence of the generated music changed for example, one participant noted that “*…it was hard to detect if there were big shifts in the music…*”. The fact that there was only one melodic part present also did not help this, as there was no coherent chord structure. This was implemented, but then removed as a sufficient way for the melody and the chords to work together was not found in the time allowed.

### 5.3.3 - Note Divisions

The application is not currently able to handle notes that are not devisable by the shortest length. Commonly referred to as tuplets, these are a series of notes that are equally split up into an irregular rhythm. The most common example of this is the triplet, this is when a note is subdivided into three equal parts currently, their start time and length will be rounded, by the set shortest note (defaulted to 0.25 beats), which would remove their musical structure.

### 5.3.5 - Participant Count

The number of participants was also relatively low in this project, causing the results to be weighted towards the participants who were both gamers and musicians. It cannot be stated if this is an actual causal relationship or a statistical anomaly, although this would be an interesting subject for further research. Simply getting more participants would be able to clear this up, preferably from all four subgroups.

# 6 - Conclusion

## 6.1 - Overview

This project set out to examine whether a procedural music system could be implemented which would work in tandem with a simple game to adapt to the in-game events. As the participants, on the whole, found that the emotional state of the music changed correctly with the in-game events, the project can be deemed a success. While the valence changes were sometimes too subtle (discussed above, see section XX), the intensity changes were not and so were successful in increasing the tension of the music as players progressed. One point of note is that care must be taken when correctly determining what in-game events can affect music if participants think that one should and there is no change this could lead to a breakdown of musical cohesion in the game.

## 6.2 - Expanding the Markov

As stated above (see section XX), the application would benefit from increasing the order of the Markov chain used to analyse the music. To further this, utilising a higher dimensional chain to analyse how the different channels interact would help the musical cohesion of the generated music, similar to Snodgrass and Ontañón’s (2014) experiments using Markov chains to analyse 2D images. This would also allow for multiple instruments to be played at the same time.

## 6.3 - Other Algorithms

Developing other algorithms would also be a viable course for future study for example, the attention-based neural network ‘Music Transformer’ (Chengzi, 2018) seems an interesting example which can generate music with a higher degree of coherence than other proposed methods. It self-references and so can replicate phrases that have appeared before in a piece. While the paper does not mention the length of time that is required to generate the music, as it would require Real-time music generation to work in a video game, an algorithm based on this concept would be an excellent addition to the field of research for procedural music.

# 7 - Appendices

## 7.1 - Questionnaire

# An Investigation into Experience Driven Procedural Music Generation

Please take your time to read the following information carefully and feel free to ask me any questions. The survey and the game should take no longer than 20 minutes to complete altogether.

Aim of Research:

This questionnaire is part of the evaluation section of the researcher’s CMP404 Honours Project Execution Module at Abertay University.

The aim of this survey is to evaluate how well experience driven procedural content generation works for music in games.

Content of the Questionnaire:

This survey will evaluate your previous experience with music and video game, and you well you think the music generation works. You will then be asked to play the game before continuing with the questionnaire.

Data Protection:

All the data recorded from this survey will be made anonymous. The researcher will do their best to not collect any identifiable data. The data will not be accessible by anyone other than the researcher and the researcher’s project supervisor and will be stored on the researcher’s password protected hard drive. A summary of results will be presented in the researcher's dissertation but no data that can be directly linked to you. The anonymised data will be kept until the study has been completed and the data will be destroyed on or before the 10th May 2019.

## Consent From

|  |  |
| --- | --- |
|  | Tick box to agree |
| I confirm I am 18 years old or older |  |
| I understand that I can withdraw from the project at any time before submitting the survey without having to provide a reason |  |
| I understand that the data I submit as part of this survey is stored anonymously |  |
| I understand and consent to how my data is processed and stored |  |
| I consent to participating in this survey as part of the study |  |

Contact information:

If you would like further information on this study or have questions regarding the survey please contact the researcher, Andrew Milne, by email at [1101624@abertay.ac.uk](mailto:1101624@abertay.ac.uk).  
You can contact the researcher’s supervisor, Grant Clarke, by email at   
[g.clark@abertay.ac.uk](mailto:g.clark@abertay.ac.uk).

The Research Ethics Committee, University of Abertay, has reviewed and approved this research study. If you have any ethical concerns or complaints about the conduct of this research, please contact the committee at [ResearchEthics@abertay.ac.uk](mailto:ResearchEthics@abertay.ac.uk).

## Previous Experience

### In a typical week, how often do you play games?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Never | < 30 mins | < 2 hrs | < 5hrs | > 5 hrs |

### Do you play a musical instrument?

Yes/no

### Reactive music is when the actions of in-game characters and/or events affect the game's music. How important do you think this is?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 (Not important) | 2 | 3 | 4 | 5 (Very important) |

## Before playing the game.

During the game look out for this type of lights, they will change colour as you progress.



Please now play the game before moving onto the next questions, pay attention to the music as you play

You can download the game here:

https://drive.google.com/open?id=11EqjHliXnR3\_lOL9w\_OrduV7ucdO7gcH

Note, you do not need to complete the game, getting to around 150 points will be sufficient

### How to play the game

Double click the file called HonoursGame.exe

This should take you to the main menu

Clicking quit will quit the game

Clicking play will launch the game

WASD to move your character

Mouse to look around

Left shift to run

Try to collect all 200 pellets without getting caught by the guards.

## How would you describe the music at these points in the game?

### At the start

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exciting | Stressful | Happy | Sad | Relaxed | Boring | Other |

(Optional) If other, how would you describe it?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### When the lights change to orange

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exciting | Stressful | Happy | Sad | Relaxed | Boring | Other |

(Optional) If other, how would you describe it?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### When the lights change to red

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exciting | Stressful | Happy | Sad | Relaxed | Boring | Other |

(Optional) If other, how would you describe it?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### When being chased by the guards

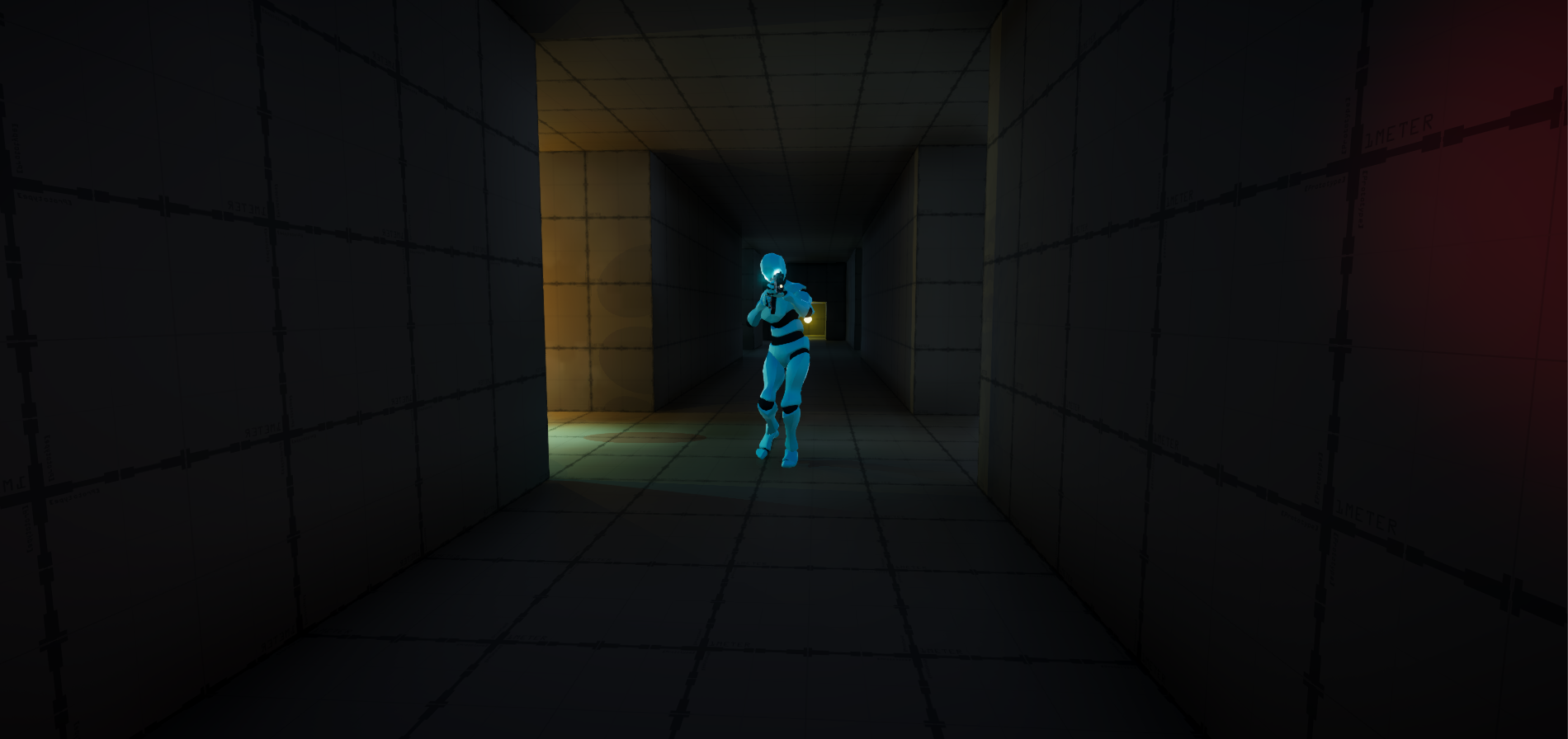
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exciting | Stressful | Happy | Sad | Relaxed | Boring | Other |

(Optional) If other, how would you describe it?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## You will now have to answer 9 questions, you will be told about an event that can occur in the game and asked if this event had any effect on the music. You can also say what effect you think this had.

### Did this event effect the music?



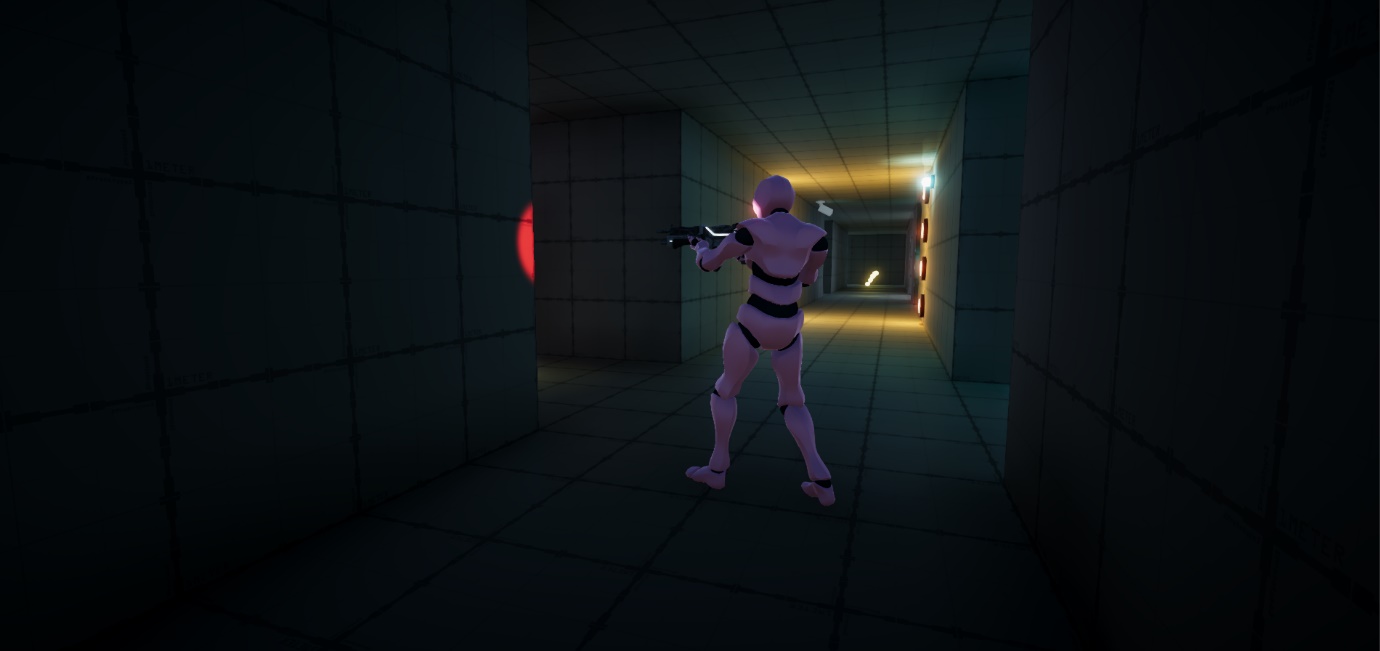
If a guard has seen you

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?



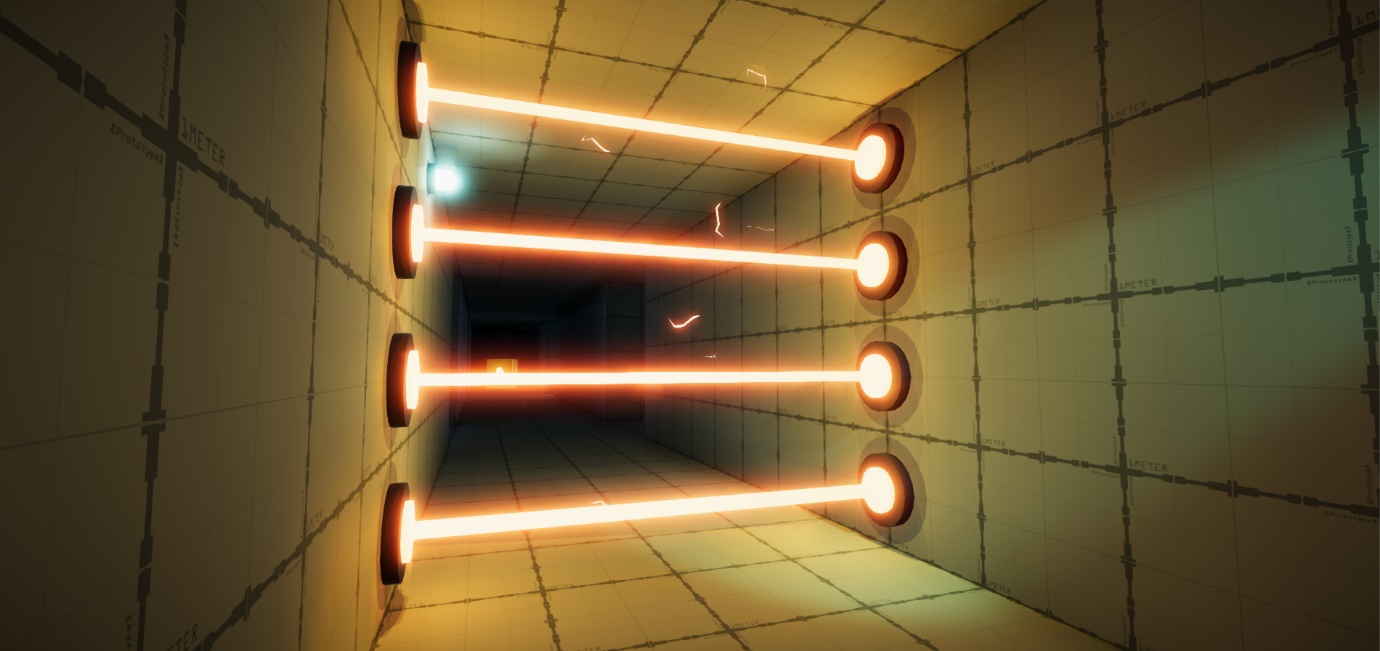
Being close to a guard

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?



Being close to the laser grids

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?

How high your score is

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?



If a security camera has seen you

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?



Being inside a secret passage

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?

How fast you are moving

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?

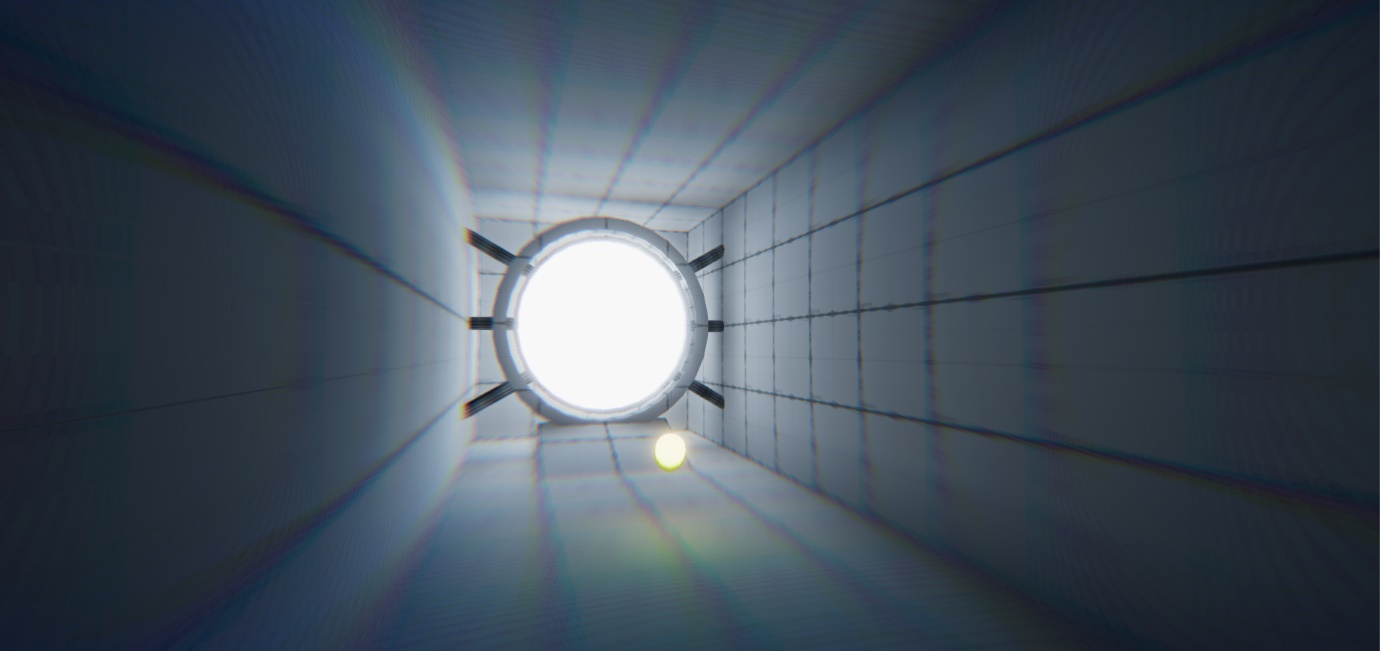
How many lives you have left

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Did this event effect the music?



Using the teleporters

|  |  |  |
| --- | --- | --- |
| Yes | No | Not Sure |

(Optional) If yes, how did this affect the music?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Thoughts on the project

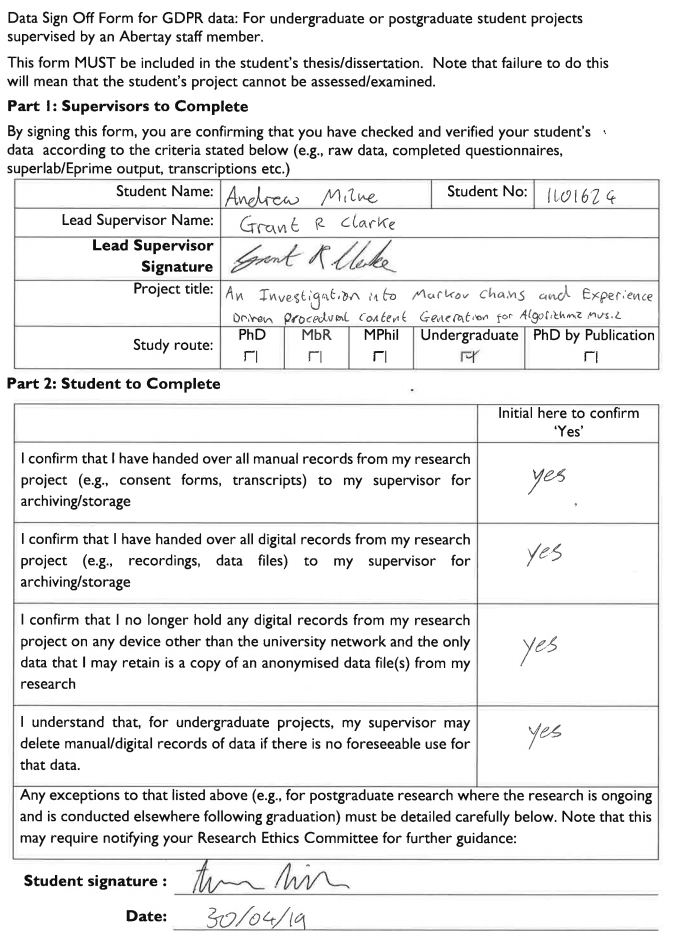
### (Optional) Do you have any feedback on the music present in the game?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### (Optional) Do you have any feedback on the game itself?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 7.2 - GDPR form



# 8 - References

Jayden Chan, Daza, J.J., Kwan, W. and Basu, A. (Oct 2017) *Facilitating player progression by implementing procedural music in videogames.* IEEE, pp. 2328.