# 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.