Implementing Procedural Music To Reflect Gameplay

Matthew West
BSc(Hons) Computer Games Technology

Institute of Art, Media and Computer Games
University of Abertay Dundee
May 2012

University of Abertay Dundee

Permission to copy

Author: Matthew West

Title: Implementing Procedural Music To Reflect Gameplay

Qualification: Bsc(Hons) Computer Games Technology

Date of Submission: 08/05/12

I agree that a copy may be made of the whole or any part of the above-mentioned project report without further reference to the undersigned

Signature:

Address: Achnahinich Househill Drive Nairn, IV12 5RX

Date:

Abstract

This project explores the possible techniques which could be used to create a procedural music system capable of creating dynamic music, reflective of gameplay. The technique created in this project combines the musical advantages of pre-composed music with the dynamic nature of a procedural system. The “Fuzzy Riff System” uses a bank of pre-composed riffs which are one bar in length. Each riff is stored with its start and end note values along with a value representing the mood of the riff (calm to frantic). A fuzzy logic system also creates a mood value based upon in-game variables, such as health, to represent the current mood of the game. The music system selects and plays appropriate riffs from several instruments based on the game's current mood level and the riffs which were previously played. A user test showed that the procedural music created was as musically pleasing as a pre-composed track. The results also showed that the procedural music was indeed reflective of in-game events and led to increased player enjoyment. Performance tests showed, that in comparison to using a pre-composed music track, memory usage did not increase and the application's start up time was significantly reduced. Tests also found an inefficient algorithm which was then revised.

Table of Contents

List of Figures

Appendix B – Stages of fuzzy Logic

Introduction

Procedural music is a concept that has been around since the middle ages (Roads. 1996). As times have moved on with the advent of computers, does this technique have a place in computer games? Compared to the standard method of pre-composed music in games which is linear and sounds the same every time, procedurally generated music can be dynamic and change in real-time according to live input. Computer games are a medium which could greatly benefit from the use of this. Compared to film, where the composer knows exactly when something interesting is going to happen and is able to construct the music around that point, computer games are controlled by the player who is unpredictable (McCuskey 2003). Therefore pre-composed music will not be able to highlight the player's actions accurately.

Imagine playing your favourite game with your own personalised soundtrack, matching and adapting to the way you play. Also, for large open world games with hundreds of hours of gameplay, non repeating music would become a viable option compared to creating hundreds of hours of composed music.

The aim of procedural music in games is to produce a dynamic musical track which adapts itself to reflect gameplay. The system can do this by manipulating/creating musical notes and structures. This is accomplished through using forms of artificial intelligence working with stored musical knowledge (weights in a neural network or Markov machine (Farnell 2007)). Finally, to reflect gameplay, the decisions made by the artificial intelligence must be in some way mapped to in-game variables. It would also be useful to describe what procedural music is not. It is not linear pre-composed music. The term is also usually reserved for techniques which work in real-time (Farnell 2007). A possible definition for procedural music which accounts for these points might be,

> "*A system which uses several inputs to control and manipulate musical aspects based on internal knowledge to create dynamic music in real-time*."

Although procedural music is not a new topic, it has rarely been adopted by computer games and there is very little in the way of games based research to base the creation of an application on. Research will instead look to techniques used by composers and how they can be implemented in a real-time game environment, techniques such as: Markov chains, neural networks, fractals and knowledge based systems. Another area which will be researched is the current methods used to create dynamic music in games. Two techniques are commonly adopted. Firstly, vertical re-orchestration which involves the dynamic mixing of the instrumentation of pre-composed segments of music to reflect gameplay (Winter 2005). The other technique used is horizontal re-sequencing where pre-composed sections of music are repositioned in time according to the state of gameplay (Winter 2005). These techniques can create dynamic music; however, their output is confined to the various sections and layers of the pre-composed tracks.

To achieve a synchronous relationship between the music and gameplay the procedural music must be affected by in-game events. This could be the mapping of certain game variables to musical characteristics that share some sort of resemblance. For example the number of enemies on screen relating to the note frequency of the music, conveying a sense of calm or franticness. This poses a problem as video game players usually have a large level of freedom as to when they perform actions. The rules of music on the other hand have very strict conditions relating to time.

The possible positive effects which dynamic music created in real-time could present are great. Player immersion could be brought to a new level though greater feedback. In most current games the audio feedback is limited to sound effects which play after a certain event occurs. For example picking up a coin or crossing an arbitrary threshold into low health. With procedural music, the changing mood of the game could be constantly reflected. To represent the player’s state of health the music could slowly become darker as health is lost, creating a more immersive and dramatic environment. The music itself could become a gameplay mechanic where the player bases decisions on the current state of the music. Decisions which would in turn re-shape the music.

Possibilities like this would give game designers greater freedom to make use of music as a part of gameplay and not treat it as an afterthought to merely put the icing on the cake.

The technique created in this project sees the combination of pre-composed music with procedurally generated structure. This has the advantage of retaining the many musical advantages of pre-composition, which does not require complicated AI, whilst gaining the dynamic re-activeness of procedural techniques.

The project looks to provide a basis of comparison between the highly used approach of pre-composed music against procedural music which is created in real-time. The merits and downfalls of each technique will be compared with a focus on player enjoyment. Other aspects such as development times and performance will be discussed to give a greater idea of how each technique could affect game development. The general aims of the project are stated below.

Aims

The aims of this project are to,

- **Explore the different methods of creating dynamic music.**
  Procedural music is a concept which many composers have worked with before, creating varying results. The techniques they used will be discussed and their compatibility in terms of their implementation within a games environment will be explored.

- **Devise a technique suitable for creating procedural music in games.**
  Based on research and trials, produce a technique which is successful in creating pleasing dynamic music in games. The technique must allow for in-game variables to shape the output of the music in real-time.

- **Create a playable prototype of a game which has procedural music.**
  Implement the devised technique into a small playable game to test and showcase the idea. A pre-composed linear music track will also be created to allow for comparison through user tests.

- **Evaluate the procedural methods developed**
  What are the advantages and disadvantages of the techniques which were implemented in the project? How do they compare to other techniques?

- **Judge the merit of procedural music over linear**
  To find whether procedural music is a viable technique to be used in games its effects must be compared to the industry's standard methods. This will be in terms of player enjoyment and immersion along with effects on gameplay. Also the impact on performance and development times must be considered when evaluating each method.

- **Make recommendations and discuss the future possibilities of procedural music within games.**
  In light of the project, the possible improvements which could be made to the devised technique will be discussed. The future work which could take place will also be explored, showing how procedural music could become a part of computer games.

Hopefully these aims will help show that procedural music can indeed be a valuable technique for use within games. Including how it can be effectively implemented to have a dynamic use within games. Encompassing all of the above aims and focusing the project, the question which will be guiding the work is,

Research Question

"*How can procedural music best be implemented into a video game to make use of game statistics and events to create dynamic music reflective of gameplay in real-time?*"

Background/Literature Review

Dynamic and adaptive music is something composers have strived to achieve for hundreds of years. In the classical period, musical parlour games such as 'Musikalisches Würfelspiel' were created, one of the most famous being attributed to Mozart (Edwards 2010). The composer would write small sections of music which would be randomly stitched together according to dice roles. These sections of music had to be carefully crafted to produce coherence, as a game with an average sixteen measures could produce around 46 quadrillion works. With the advent of computers, composers realised the potential for algorithmic composition as a computer could be programed to make use of complex and repetitive mathematics to study and create music. Lejaren Hiller was a pioneer of this approach to music: writing programs for the Illiac computer with the aid of Leonard Isaacson. In 1956 they composed the *Illiac Suite for String Quartet* (Cope 2001).

Music in games has been getting more attention as of late. There are games like "Red Dead Redemption" (RockStar Games 2010) which was nominated for 10 G.A.N.G awards (Game Audio Network Guild) and big names such as Hans Zimmer coming into the industry. Music and audio play a larger part in video games than they do for films where the unheard melodies (Gorbman 1987) act mostly as an accompaniment to the on screen narrative. Whereas in games, the sound plays a more active role which must be given the player's attention in order for them to make the correct response (Collins 2007). For this reason perhaps games are not suited to algorithmic music because as well as the algorithm having to overcome the challenge of producing pleasing and coherent music they must also now provide the player with important narrative information. However this is contradicted by Collins who says,

> "*Video games are perhaps an ideal media form for procedural music; after all, many elements of gameplay - especially the timing of events and actions - are unpredictable and occur in a non-linear fashion.*" (Collins 2009, p. 5)

Perhaps both arguments are right: more traditional methods of composition will still be needed where a game requires a specific emphasis or player reaction. However, for ambient or immersing music which does not play a large part in the narrative, the many benefits of music created procedurally in real-time could be utilised.

Although most current games are using linear music it is not to say they have no dynamic elements. Powerful audio engines such as “fmod” (Firelight Technologies 2012) are becoming widely used in the games industry. These allow composers to create dynamic compositions with branching structures which are affected by in game events. This can allow a game to smoothly switch between and layer several pre-recorded tracks (horizontal re-sequencing and vertical re-orchestration). However the variations are limited to the amount created by the composer. Procedural music on the other hand, can have countless variations with a much higher level of flexibility. An advantage with techniques such as horizontal and vertical re-orchestration is that as the music is pre-composed it will have a high musical standard. A composer will carefully craft the different layers so that they complement each other and create the atmosphere which the games designer is trying to achieve. As procedural music is created in real-time it runs the risk of not having the same musical structure and coherence as pre-composed music. All of the rules which lead to a piece of music being coherent will have to be encoded into the procedural system in some way for it to become a viable technique which developers could use. This brings the problem of implementing emotion and creativity which cannot be easily defined as a set of rules. There seem to be fundamental obstacles which may not be solved, including complex AI problems. The matter is complicated by the fact that there are no real metrics by which to judge the output of procedural music (Farnell 2007). This project aims to overcome these problems and present a suitable method for implementing procedural music in games.

Advantages

The possible advantages of procedural music are great. Firstly to immersion, procedural music would be able to change its motifs, structure and balance to reflect emotional dimensions (Farnell 2007). Dynamic moods in the music could reflect gameplay to a much higher level than pre-composed tracks. For example in a game which has moral elements, there might be pre-composed tracks to reflect good and evil play. Procedural music on the other hand could account for multiple levels of good and evil and the transitions in-between as the player interacts with the game world. Immersing the player in the game is a goal all developers try to achieve. Crowle discusses the use of dynamic music to achieve immersion saying,

> "*This invisible layer creates an environment for the player that has sounds and music that is always appropriately matched to the circumstances and events that are occurring in the game world.*
> *In this manner, it can be guaranteed that the player is never removed from the game due to irritating music that sounds out of place.*" (Crowle 2011, p. 28)

Aside from the beneficial effects of procedural music on the player, there can be great advantages to the developers as well. Take for instance the game 'Resistance 2' [Insomniac 2008], it boasts 420 hours of gameplay (McWertor 2008). Without the aid of generative music it becomes highly impractical to create non repetitive music for the game. If solid models for creating procedural music were designed developers could take advantage of them for some areas of their game, leaving the composers to work on key areas, saving time and money.

So if the benefits are clear, why are developers not taking full advantage of dynamic music within games? Firstly, it should be noted that there are rare cases of games using procedural music. For example, 'Spore' developed in 2008 by Maxis, a subsidiary of Electronic Arts. The audio team made use of a software tool called Pure Data. The version they used was customised by EA themselves. The procedurally generated music is most notable when in the customisation areas of the game where the player designs game elements such as creatures.

> "*The music shifts in tone based on what kind of creature you are building. Add combat accoutrements like sharp pincer claws or vicious buzzsaws to your critter and the music will subtly shift to a minor key and take on sinister overtones.*" (Kosak 2008, p. 1)

The game became renowned for its use of dynamic music, becoming a finalist for the 'Best Original Score for a Video Game' in both the 'Hollywood Music in Media Awards' and the 'International Film Music Critics Association' in 2009. Although the game was successful it had not spurred other developers to implement procedural music in their games. Maybe this can be answered by looking at the disadvantages of procedural music.

Disadvantages

One of the main challenges to the adoption of procedural music is due to there being an enormous investment of skills and knowledge in using existing tools (Farnell 2007). New tool chains will have to be developed which would also mean composers and audio programmers will need to be trained in new methods. The task of creating procedural music would be best suited to individuals who are both adept at using code and have compositional skills. This new job role may be hard to fill, especially as there is already a shortage of audio programmers (Broomhall 2011). Even when new methods of incorporating music into games are devised they are rarely included in published academic work, being reserved as a trade-secret (Rossoff 2007). Due to this, developers are less likely to take the risk in spending extra time and resources to devise procedural methods when they can still expect great results from composers using standard methods. This is well described in the following diagram (fig. 1) showing the rut which procedural audio is stuck in.

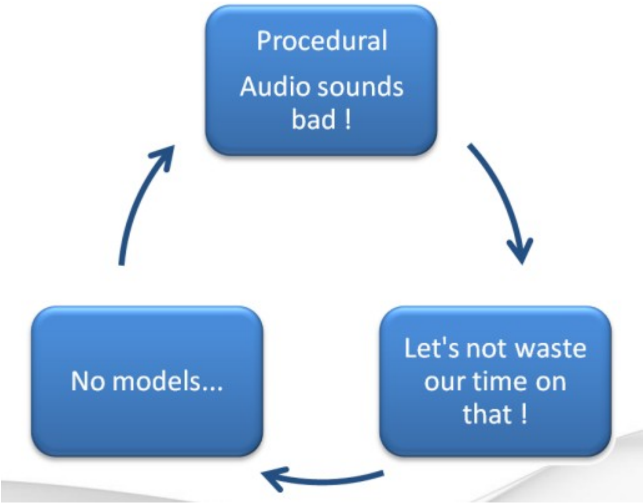

Procedural Audio sounds bad !
Let's not waste our time on that !
No models...

Figure 1: The vicious cycle procedural audio is stuck in. (Fournel 2011, p8)

Another difficulty in creating procedural music is the complexity involved. Crowle states that,

> "*Instead of having a musical track for an area of a map, now there must be various smaller tracks that can be combined to create many different tracks according to the circumstances in the game in that area. This creates a problem that cannot be overcome with capital or more manpower, it requires the elegant hand of a talented composer and nothing else.*" (Crowle 2011, p. 30)

This suggests that creating procedural music would require greater skill and development time, the latter being a precious commodity. Although if it became common practice composers would become skilled in the area and tools to aid the production could be established. Until that happens, it is a brave step for any company to make.

Moving from data to procedural models will never be a quick or complete process. That is not to say that procedural music cannot exist alongside the standard methods (Farnell 2007). Even when procedural music reaches a high level of quality, there will still be call for pre-composed music tracks to skilfully highlight very specific areas in games or for use in non-interactive cut-scenes.

Techniques

The techniques used by computer composers to create algorithmic music are known and discussed, however they are discussed from a musical point of view. This project is more concerned with their adaptability and suitability for games, for example efficiency and flexibility. Although we cannot directly look to research, we can compare the various methods to similar techniques used in other areas of games like graphics.

- Markov chains are used throughout games for areas of chance, for example in artificial intelligence. Due to their controlled random output which is based on past outputs it can be used to model systems that change through time in a non-deterministic way such as simulating weather (Dreiseitl 2010). Musically they have also been found to be very useful in producing chord progressions and melodies which should always be based on what has been played prior in the music. With their flexibility of being able to have several layers of dependency and customisable weightings, which could include in game variables, this algorithm should work well in a game environment.

- Neural networks are used by composers due to their ability to learn. The system which is initially blank is given knowledge through training with example inputs that expect certain outputs (Farnell 2007). By training these systems, composers can use the neural networks for very specific purposes with great results. However, in a game environment, we cannot have the player train the algorithm as they play the game. Pre-training in the games development stage would leave the neural network very specific and unusable for creating other styles of music in different areas of game. For this reason it does not present itself as a viable technique for games.

- Fractals have found many uses in computer graphics especially in procedural generation, possible uses such as generating realistic plants. However it has been found that although fractal methods lead to the generation of meaningful geometry, they might not produce meaningful music (Farnell 2007). As the project is looking to generate music to the best ability, methods which may be proven to work well in games are not necessarily useful unless they can produce music to a high standard.

- Composers use what they call musical grammars when generating algorithmic music. These grammars are a set of mathematical rules relating to music which will define the generation of the harmonies and rhythms: basically a way of coding the rules and restrictions composers have to adhere to when creating music. As these will be a fundamental set of rules they should not have any difficulties being implemented in a game environment. Also, these rules are of great importance to the music and therefore cannot be omitted.

- Although the aim is for the procedural algorithms to create new music based on game states it does not mean they cannot take some basis on existing musical structures and patterns. Knowledge based systems do just that. A bank of chord progressions and musical riffs can be stored so other techniques such as Markov chains can look to them for guidance when calculating the percentage change of playing certain notes or chords. By doing this, the music is more likely to be coherent and pleasing whilst the endless possibilities are not lost, they are simply guided: much in the same way as a jazz musician uses past experience when improvising.

Mapping system

To create procedural music which is affected by in-game statistics and actions the technique must include some sort of mapping system. This system should relate game variables to the output of the music in a meaningful way. To reflect the current mood of the game the music must also have dynamic moods. For example if you are doing well the music might be relaxed and easy going but when you start to get into trouble the music should reflect this and become more tense and foreboding. These states of moods shall be known as musical mood levels. As part of the procedural music technique the mapping system should increase or decrease these levels in correlation to what is happening in the game. It can be thought of as an adjustable slider. Many of these sliders could be used to represent different moods which would combine to reflect the overall feel of the game.

These musical mood levels might take many variables into account. For example the tension of the music could be based on: the number of enemies present, the player's health and the time left in the level. A system must be used to combine these values with controlled results. These results must also be smooth to prevent noticeable jumps in the music's mood. A control system technique which could effectively create this smooth output is a Fuzzy Inference System. This basic form of artificial intelligence could take in the current game variables and output meaningful results based on ambiguous rules which could be altered to give different results. To further explain the strengths and operation of this technique here is a brief overview.

Overview of Fuzzy Logic

Fuzzy logic was a concept created by Professor Lotfi Zadeh as a way of processing data which allowed for more ambiguous group allocation compared with the standard crisp approach (Kaehler 1998). This is best described by the below diagram (fig. 2). The first representation is of a crisp allocation where a person is considered tall only when they are above a certain height. If a person is below this height they therefore must be small. However this grouping is not like the real world where someone just below the line could be considered as fairly tall. The second representation shows a fuzzy logic allocation where a small person will be considered more of a tall person and less of a small person as they grow in life compared with them one day becoming known as tall.

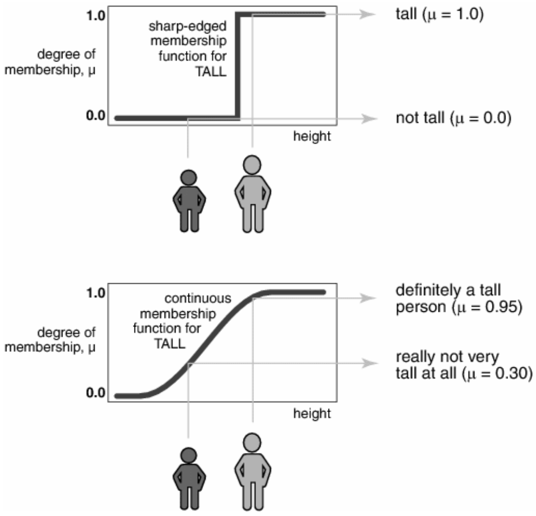

1.0
degree of membership, μ
0.0
sharp-edged membership function for TALL
height
tall (μ = 1.0)
not tall (μ = 0.0)
1.0
degree of membership, μ
0.0
continuous membership function for TALL
height
definitely a tall person (μ = 0.95)
really not very tall at all (μ = 0.30)

Figure 2: Visual representation of a fuzzy membership allocation compared with a crisp method. (MathWorks [no date])

This smoother interpretation of data is created with the use of a simple rule based system (if A and B then C) compared to attempting to model the system mathematically. The system is also empirically-based which makes for an easier creation, which is not reliant on technical skill. This makes it a flexible system as the user defines the rules which can easily be tweaked for improvement or drastic alteration.

One of the downfalls of fuzzy logic is that the integral rule base could become very large if many inputs and outputs are used. However Morgan Kaufman noted that fuzzy logic systems generally require 50 - 80 percent less rules than traditional AI rule systems to accomplish the same task (Bourg and Seeman 2004).

Although Fuzzy logic was conceived as a method for better handling data it has become a great tool for control system applications (Kaehler 1998). Hence it will prove to be a good tool for handling the musical mood levels in the proposed prototype game.

Conclusion of techniques

By evaluating the techniques employed by composers and other areas of games, certain methods seem to be more appropriate from a computer games point of view. The following techniques were selected and their intended use will be discussed in the next section.

- **Markov Chains:**
  Their ability to make decisions on past events makes them a good choice for handling melodies.

- **Musical Grammars:**
  Mathematical rules used to mimic the principles a musician would use when composing.

- **Knowledge Based System:**
  Can be used to store many riffs or musical patterns which can be played by themselves or edited by a Markov chain.

- **Fuzzy Logic System:**
  Its use as a control system should prove effective for handling the musical mood levels based on changing in-game variables.

Methodology

In order to evaluate the use of procedural music within games and its implementation a small prototype was created. The prototype had to fulfil certain criteria in order to answer the research question behind this project.

The prototype must contain:

- A playable level which can have varying outcomes.
- In game statistics which vary over time.
- Real-time procedural music system
- A separate pre-composed music track

To fulfil these criteria the game created was based on a pre-existing 2D platformer game. This was chosen for several reasons. Firstly as graphics were not a consideration of this project it was deemed an unnecessary use of time to create a complex 3D game. 2D platform games are also a very common game genre being one of the first popular genres (Klappenbach ND), so the core mechanics of the game are easily understood. By creating a simple level this allowed for more time and concentration to be given to the study and implementation of procedural music.

The prototype would have to include elements of game-play to which the generation of procedural music could be based on. A simple solution would be to have enemy characters in the game. The number of enemies could increase over time thus creating an element of game-play which can be tracked by the application and easily followed by the player. A second element which could be tracked is the players health. Health is a common attribute in games which can in itself create tension. By having the procedural music take this health value into account, hopefully the music will highlight and heighten this tension.

It was decided early on in the development process that the procedural music would be made to reflect the current tension within the game. This decision came naturally as the game was to be based on increasing numbers of enemies and the player's health. The idea being that the more enemies which are on screen the more frantic the music should be. Counter balanced by the player's health, the more health the player has the safer he is so the music should be less frantic. To encompass these core mechanics the prototype took the shape of a simple survival game where the player has to avoid increasing numbers of enemies whilst collecting health pickups. The addition of portals were implemented to add some depth to the game-play to hopefully entertain the players. An example of the finished game can be seen below (fig. 3). The executable version of the game along with gameplay videos can be found in (Appendix D).

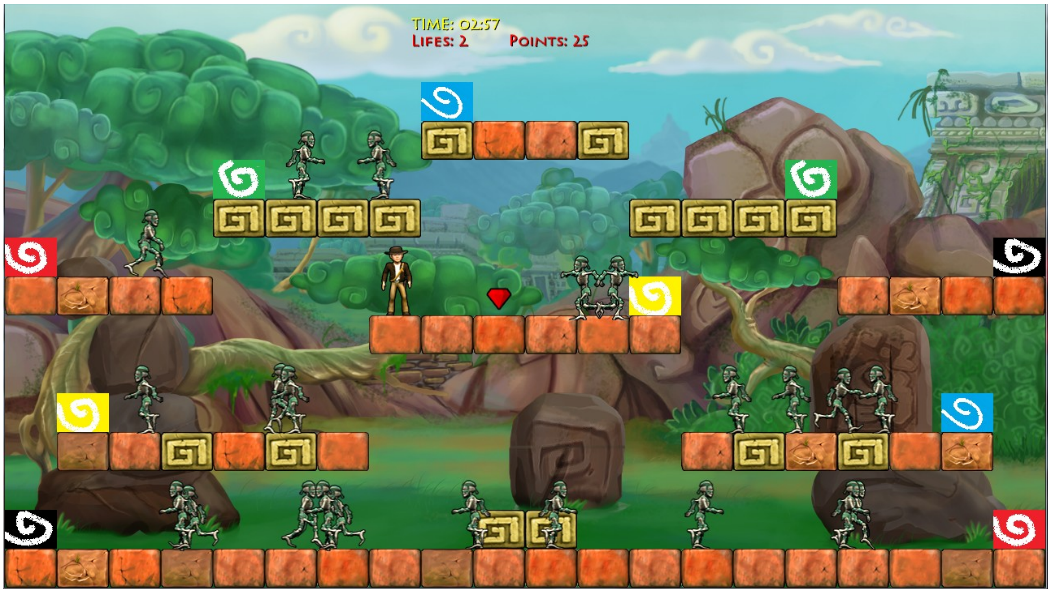


Figure 3: A screenshot of the finished prototype. Players had to avoid the enemies whilst collecting lives and utilising the coloured portals to stay alive.

As previously stated in the literature review, certain techniques used by algorithmic composers stood out as being viable techniques for use within a computer games. The main technique which was going to be used for note selection and melody creation was Markov chains. Let us look at how a Markov chain can be used to create a simple bass line. The table in (fig. 4) is a representation of a first order Markov chain where C, D, E… represent musical notes. The chance of what note will be played next is based on the previously played note; for example if a C was played last there is a higher change (30%)

the next note will be a D compared to another C (15%). The next table (fig. 5) shows a second order Markov chain where the last 2 notes played are taken into account. This allows finer control of the output. The same is said for third and fourth order Markov chains. The weightings assigned to each outcome are defined in the implementation stage so musical knowledge is needed by the algorithm's creator.

	Next note	**C**	**D**	**E**	**F**	**G**
Last note						
C		15%	30%	30%	15%	10%
D		15%	10%	30%	30%	15%
E		20%	15%	15%	20%	30%
F		10%	20%	30%	10%	30%
G		15%	10%	20%	30%	20%

Figure 4: 1st order Markov chain

	Next note	**C**	**D**	**E**	**F**	**G**
Last 2 Notes						
CC		0%	40%	35%	15%	5%
CD		15%	20%	20%	30%	15%
CE		20%	10%	15%	30%	25%
CF		5%	15%	30%	10%	40%
CG		15%	10%	25%	30%	20%

Figure 5: 2nd order Markov chain

After starting development on this technique many flaws became apparent. Foremost was the size of the rule base. The first table of weightings (fig. 4) shows the rules for a choice of only 5 notes. There are 25 weightings here. If the note selection was taken to include one full octave (12 notes) there would be 12^2 (144) weightings. These would each have to be assigned a percentage chance of firing which had been given careful thought based on musical knowledge. This soon becomes a very inefficient method. Using a 2^{nd} order Markov chain would bring the rules up to 12^3 (1728) weightings. The number of rules does not stop there however. This does not take into account the choice of the notes value; should it play a crotchet, quaver, minim or even a rest. Even if the technique was implemented the laborious work would only account for a single octave of playable notes with a very poor account of what notes had previously been played This makes the Markov chain method unsuitable for use within games development. A huge amount of time and effort would be needed to create a pleasing output and even then the Markov chain would only be able to produce melodies in the specific genre it was created for.

This called for a re-think on the approach of the procedural music creation. Another technique which was discussed earlier was knowledge based systems. These would hold banks of riffs or chord progressions which were pre-composed with musical intelligence. Techniques like Markov chains could look to these riffs and chord progression for guidance. For example a small musical melody could be described within the knowledge based system, note values and timings etc. A Markov chain could take this riff and augment it slightly using its own logic to create a new riff which could then be played. The advantage of this is that as the riffs and chord progressions are pre-composed they hold the musical intelligence of a composer compared to the simple logic of the Markov chain itself. Therefore the musical output is far more likely to sound pleasing. However, as the implementation of a Markov chain was deemed inefficient the knowledge based system could not be used in this way.

Bearing in mind the advantages and core concept behind the knowledge based system a new technique was devised. This new system would be a large bank of pre-composed riffs which would be intelligently selected and played. There were two prominent methods in which the riffs could be stored and played back. The first was as data. The values of the notes and their timings would be stored. Playback would either make use of a MIDI system or use sampled instrument notes which would play the sound of a single note at the right time. The MIDI system was disregarded due to research showing its issues with quality as mentioned in the literature review. Sampled instruments work in a very similar fashion to MIDI. However, instead of generating the sound of the instrument using Frequency Modulation (FM) synthesis or Wavetable synthesis, a live recording of a single note would be played back (Heckroth 2001).

The second version of storing the riffs would be as small pre-recorded sections of music. The benefit of this is that the riffs do not have to be translated into a numeric format suitable for storage as text, instead they are simply recorded. Likewise the playback does not involve interfacing with a MIDI system or a similar method. Instead a simple function can be used to play back the .WAV files when they are needed. The downside is that the storage of these .WAV files takes up a significant amount of space in comparison to the data methods (Heckroth 2001). However, for the planned prototype, storage was not a real issue and as the recording method would allow for easier implementation with similar results it was the chosen method for demonstrating procedural music.

The chosen musical style for the project was the 'twelve bar blues' due to its easy to follow chord progression structure which could be easily implemented in code. The blues would also complement the chosen riff playing technique as it quite closely emulates the way some musicians improvise, pulling from a bank of known riffs or licks. This would allow the project to have a less complex riff selection system, allowing for an effective prototype which demonstrates the fundamental concepts of the procedural music technique. The musical track would comprise of three sections: piano, bass guitar and drums. Many single bar riffs of these instruments were carefully recorded with the aim of allowing riffs to flow into one another. As an example, the piano part of the finished procedural music system comprises of 60 of these pre-recorded riffs. This number was found to have a musically pleasing result without much noticeable repetition for a game which lasted an average of two and a half minutes. Compare these 60 recordings with the thousands of rules which would be needed to create a relatively very simple melody using the Markov chain method and you can see the value of the created technique.

To play these riffs in game a class was created which handled all of the timings based on an underlying metronome which continuously runs when the game is playing. As the riffs are one bar in length, the system will select and play a new riff every bar. The selection process is based on the chord progression of the 'twelve bar blues'. As the blues moves through the chord progression, this is reflected in the selection process which selects a bass riff for example in the key of the current chord. The next addition to this system was devised to closely mimic the playing of a musician and give the musical track a greater sense of flow. Each riff has its own attribute object associated with it. Firstly this object was used to store the start and end notes of each riff. This allowed the selection process to play riffs with a start note close to the end note of the last riff played. This meant that the melody would not be sporadic and jump up or down in pitch too quickly. With all three sections of the musical track playing in time and following the chord progressions of the 'twelve bar blues', the output was successfully in sounding musical. During some early tests, listeners were unable to distinguish the music as being played by a computer.

Later research found a very similar approach to the above mentioned, developed in 1986 by Composer Peter Langston who termed it as "Riffology". Here is an explanation of his technique from the paper he wrote on the research in 1988,

> "*The riffology algorithm makes dynamically weighted random choices for many parameters such as which riff from a repertoire of 32 eight-note melody fragments to play next, how fast to play it, how loud to play it, when to omit or elide notes, when to insert a rhythmic break, and other such choices. To choose the next riff to play, the program selects a few possibilities randomly. From these it selects the riff that is 'easiest' to play, i.e. the riff whose starting note is closest to one scale step away from the previous riff's ending note.*" (Langston, 1988, p.6)

As the system can chose to omit or elide notes it is similar to the approach of using a Markov to edit the riffs from a Knowledge based system. It would seem that this is a superior technique due to the higher customisation of the musical output. The system could use in game variables to determine how the riffs would be edited creating greater reflection in the music. However this technique would require a lot more development time and involve a complex AI system. The lengthy process could see developers favour more traditional techniques which are a 'safe bet'.

The purpose of the procedural music in the prototype was to reflect the game-play, so the next stage in the development process was to create a mood system. The fuzzy logic system mentioned earlier was chosen due to its effectiveness as a control system. The fuzzy logic system can have a series of inputs and outputs, see (Appendix B) for the workings of fuzzy logic. In the case of the project the output will be the mood level assigned to the game at a specific time. The system will have two inputs, the number of enemies on screen and the player's current health. As mentioned earlier, the music system would reflect the tension in the game by becoming more or less frantic. As far as the fuzzy logic system is concerned it produces this franticness value quite easily with the use of the dynamic inputs (this value was made to be between 0 and 20). This franticness value is then passed through to the riff selection system. As previously mentioned every riff has an attributes object. One of the values assigned to each riff is a franticness rating between 1 and 20. Now when the procedural system is selecting the next riff it chooses a riff with a suitable franticness value to reflect the current tension in the game. This process is used for the piano, bass and drum tracks so they all complement each other and reflect the game-play. This however, only allows for the music to reflect one particular mood based on two specific game variables. If the system was to reflect multiple moods, more riffs would need to be composed and given multiple mood values. Also the fuzzy logic system could easily be changed to output multiple mood levels based on a varying number of game variables.

There is a technical disadvantage with using pre-recorded riffs or any other method which does not generate the instrument's sound using the processor (techniques such as FM synthesis). This disadvantage is, extra strain on the data pipeline. Each recorded sample must be brought from secondary storage into main RAM or direct to the sound core using a data pipeline (Farnell 2007). This means the technique is 'data bus' intensive and will have to compete with graphics data which is also data intensive (Farnell 2007). Although, as mentioned earlier, methods such as FM synthesis would be more intensive for the computer's processor (Veneri et al 2008). As audio is rarely given priority against graphics in games (Collins 2009), both methods of approach may find difficulties in being adopted.

That being said, the relatively simple approach of the technique created in this project (Fuzzy Riff System), will hopefully give an insight into the many advantages of procedural music in video games. As computers and consoles continue to become more advanced and powerful maybe procedural audio will find its place to grow and improve with more interest and research.

Testing and analysis

To evaluate the methods used in this project several approaches were used. Firstly a series of simple tests were devised to determine whether the system behaved as expected.

- Could the application keep a steady musical tempo, playing on time?
- Did the music become more frantic when the number of enemies and player health were manipulated?

The first question was answered quite simply by running the application along with a real metronome, making sure the game's metronome did not leave sync and the riffs always played at the start of a new bar. The first implementation of the in game metronome was unsuccessful at keeping accurate time. This was due to the time system which was built into the XNA game rounding the time values slightly and having erroneous values at high resolution. This is a known problem - as stated by Microsoft the resolution is approximately 10 milliseconds (MSDN 2010). This problem was solved by utilising the Stopwatch class provided in the .Net framework. The Stopwatch will use a high resolution mode for tracking time as long as the hardware and operating system support it. A test was performed to calculate the accuracy of the stopwatch. The result was 69 nanoseconds. The revised implementation of the games metronome successfully managed to stay in sync with a real metronome in a 10 minute test.

To test whether the procedural music reacted accordingly to changing game variables a test version of the game was created. The user could adjust both the player's health and the number on enemies with key presses. The test found that the procedural system reacted successfully, producing music with an according level of franticness. The test also showed that the rate at which this musical mood changed was not constant or immediate. As the system selects a new set of riffs at the start of every musical bar, the mood level could change just after this process meaning the update would not take effect until the start of the next bar. At a tempo of 120 this could be a wait of up to 2 seconds. This problem could be solved by using a previously discussed method. Storing the riffs as note data and affecting them with the use of an AI system. The remaining notes in a riff could be changed to create a more frantic feel whilst it is being played.

Once the prototype game was completed, an important test was to determine whether the procedural music system created pleasing music which could improve gameplay experiences. As the perception of music is subjective (Peng et al 2007) the produced music cannot be evaluated through computer tests. Instead a user test was used to determine whether listeners found the music enjoyable. The test saw 27 participants play the prototype game with instruction to pay close attention to the music. After their play-through they answered a short questionnaire (Appendix C) rating the music in several categories. To compare the procedural music with standard pre-composed music the test subjects also played the same game with a custom made musical track. This pre-composed track used the same instruments and play style as the procedural system to create a fair comparison test. Also the music was created by the same composer who produced all of the riffs.

The method used to analyse the results from the questionnaire groups will be a 'Student's t-test'. This analysis technique created in 1908 by Gosset (raju 2005) assesses whether the means of two groups are statistically different from each other in relation to the variation in the data (Trochim 2006). This is necessary as if only the means of results are compared, important information can be lost. Take the example below (fig. 6) where two sets of results have the same mean whilst having very different properties.

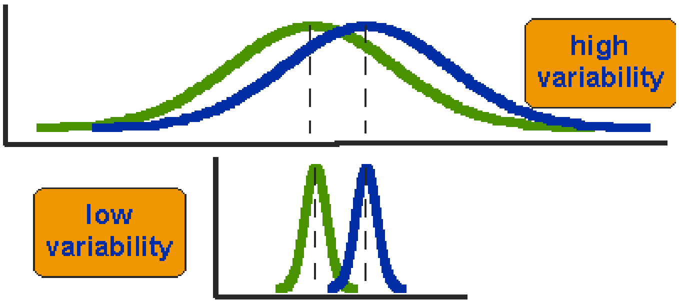


Figure 6: Two scenarios for differences between means (Trochim 2006).

The final method used for testing the application was a performance test. This would determine how much extra strain a procedural method could put on an application and whether this strain is a sufficient downfall of the produced techniques. The test compared the procedural music system with the pre-composed version in terms of the memory usage and processing time. The memory usage was easily observed as the value remained constant and could be viewed in Microsoft Window's resource monitor. To get a detailed picture of the processing time several different values were recorded: The applications start up time, average calculation time after start-up and finally the highest calculation time observed during gameplay. As these times were very small they were measured by the elapsed ticks in the underlying 'Stopwatch' mechanism which was set to high resolution. In this application one tick represents 69 nanoseconds. In the next section we shall discuss the results of the above tests and user questionnaires.

Results and discussion

Questionnaire Results

The test subjects were asked four questions after they played each version of the game.

1. How pleasing was the music in the game?
2. Was the music reflective of the state in the game?
3. How did the music affect your playing experience?
4. Did you find the music repetitive?

The results of all 27 participants were collected and analysed using a two tailed (paired) students t-test. This would show whether there is a significant change in opinion between the two versions of the game (with procedural music and with pre-composed music). The bar graph below (fig. 7) shows the results of this t-test. For the results to show a significant change which rejects the null hypothesis (Bower and Colton 2003) the 't values' must be over a certain threshold. In the case of this test, the 't values' must be higher than 2.06 or lower than -2.06. These values are represented by the red lines shown below (fig. 7).

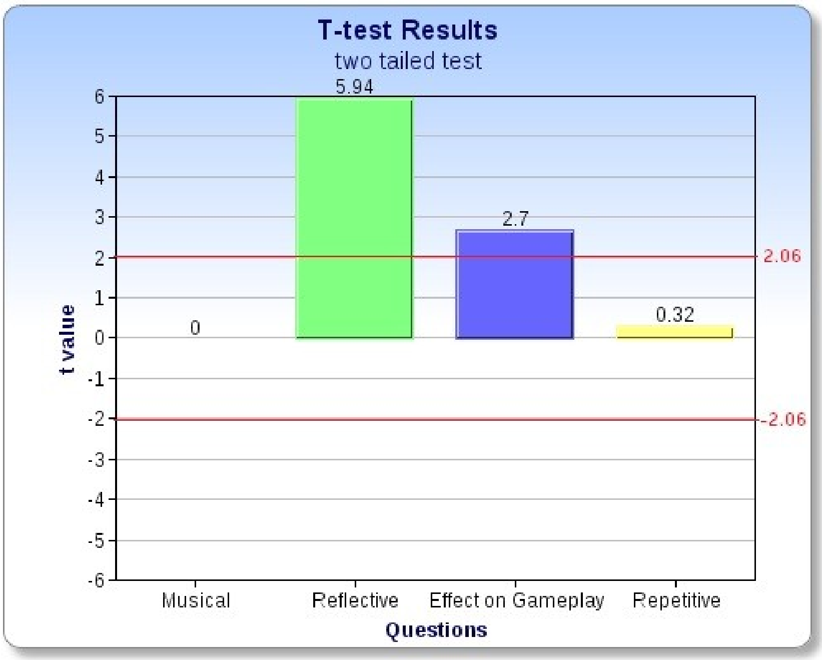


Figure 7: The results of the t-test. Values crossing the red line show a significant change.

Results for question 1:

As can be clearly seen, no statistical difference was observed between the perception of musical quality between the procedural and pre-composed music. Both versions of music had a mean score of 6.89 out of a possible 10 (where 0 was very "unpleasant" and 10 was "very pleasing"). This shows that the procedural music was successful in having musical qualities of a similar standard to pre-composed methods. This means the procedural music system created in this project successfully overcomes one of the main challenges facing procedural music (creating pleasing output) as stated in the literature review. Also this was one of the aims of the project.

Results for question 2:

Question 2 was perhaps one of the most important, "Was the music reflective of the state in the game?". The result of the t-test shows that the results far exceed the threshold of rejecting the null hypothesis, therefore this is a significant change in opinion. On a rating of 1 to 10(where 1 meant "no similarities" and 10 represented "extremely reflective") the pre-composed music had a mean score of 3.67 whereas the procedural music had a mean score of 6.37. That relates to the music being perceived as 74% more reflective. Creating dynamic and reflective music was one of the project aims and it would seem that the techniques used indeed create reflective music.

One fact which may have skewed these results was the events which occurred in each subjects' play-through. Depending on how the player performed in-game, they may have come across many or few moments where the music dynamically reacted. This would happen when they collected extra lives or lost lives. This discrepancy is thought to act in favour of the procedural music system as had all players performed well they would have heard more reflective changes in the music. Therefore this discrepancy does not detract from the positive result.

Results for question 3:

The third question was used to determine whether the procedural music increased the enjoyment of gameplay. The 't value' of this result again crossed the threshold, showing a significant change in opinion in favour of the procedural music. The participants rated the music's effect from 1 to 10 (1 being a “highly negative” effect and 10 being “highly positive”). The pre-composed music had a mean score of 5.67 and the procedural scored 6.30. This shows an 11% increase in positive effect.

As mentioned in the results of question two, there were discrepancies in the participants' play-throughs. In terms of enjoyment, we cannot assume that by hearing more reflective changes at different times the participants would have had increased/decreased enjoyment. This would require further testing where the changes in music were counted to determine whether this had a positive or negative effect. However, on the whole the procedural music was still perceived as creating a more enjoyable gameplay experience.

Results for question 4:

One of the worries associated with having a procedural music system based on pre-recorded riffs was that repetition would occur and distract players. The fourth question was used to find whether the players found the procedural music to be more repetitive than the pre-recorded. As can be seen in (fig. 7) the low 't value' of 0.32 does not cross the 2.06 threshold. This relates to a 75% chance that the change in results was merely a coincidence. This shows that the procedural music was not significantly deemed more repetitive and further proves the success of the chosen techniques.

The music may have been deemed as more repetitive had the gameplay resulted in the mood level remaining constant for a long period of time. Due to the nature of the riff selection system, if the mood level remained constant there would indeed be more repetition. However careful level design could overcome this problem along with using a larger bank of riffs. In the case of the prototype game however, the procedural music was not deemed as repetitive, having a mean score of 3.52 (where a score of 1 represents "no repetition" and 10 means "highly repetitive"). The pre-composed had a score of 3.41.

Comparative result

After each participant had played both versions of the game they were asked which version of the game they preferred in general. The pie-chart below (fig. 8) shows the result of that question. It can be clearly seen that a higher percentage of players preferred the version of the game which had the music created procedurally. This result by itself could show that the techniques developed in this project are successful in creating a procedural music system in a games environment.

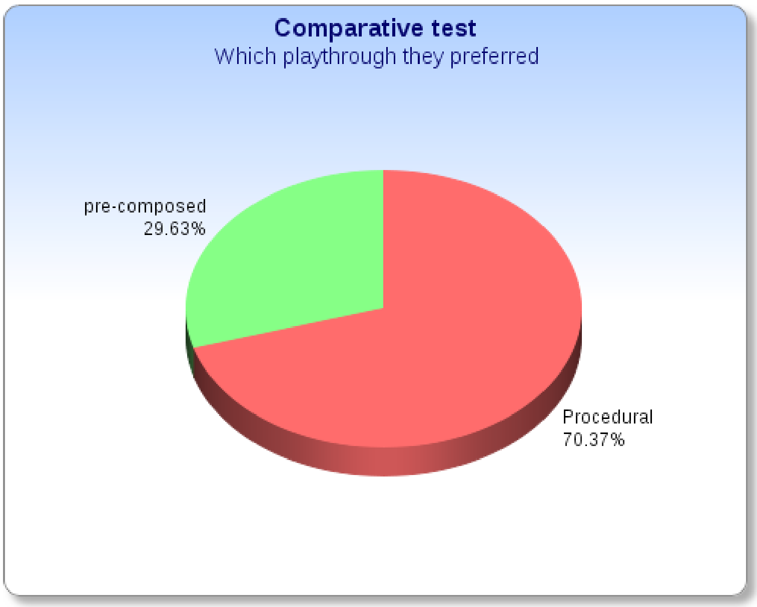


Figure 8: This pie-chart shows which version of the game most players preferred.

The participants also had the opportunity to leave comments about their experience with the games. Several common themes will be discussed. Participants who said they preferred the game with the pre-composed music often commented that they noticed no change in the music between the two versions. It was also observed that the more musically talented a participant was, the more obvious the reflectiveness of the procedural music appeared. This could mean that to some players the changes in music are too subtle for them to notice during gameplay. As noted by one of the participants, perhaps if the players knew which variables were controlling the music, the dynamic changes would become more apparent and enjoyable for the player.

A theme which was apparent in the players who preferred the game with the procedural music is that they noted the game felt more immersive and exciting. Some players noted that as the music increased in intensity the gameplay seemed to feel dangerous and more challenging. By making the players feel more immersed and excited in the game the procedural music shows itself as being a valuable tool for developers to use.

To help keep the testing process fair, half of the students played the pre-composed version of the game first whilst the other half played the procedural version first. It was thought that players would find the procedural music to be more reflective if they already had a linear music track to compare it to. However no noticeable effect was noted. As previously mentioned, the player's performance in-game may have shaped their opinion. To help combat this, if the participants did not survive their play-through for a long enough time they would restart the game. This ensured that the player's experienced the game long enough to form an opinion. However this did not address the fact that some players' actions may not have caused many dynamic changes in the music.

Performance Results

The observation of the memory usage showed the following results (fig. 9).

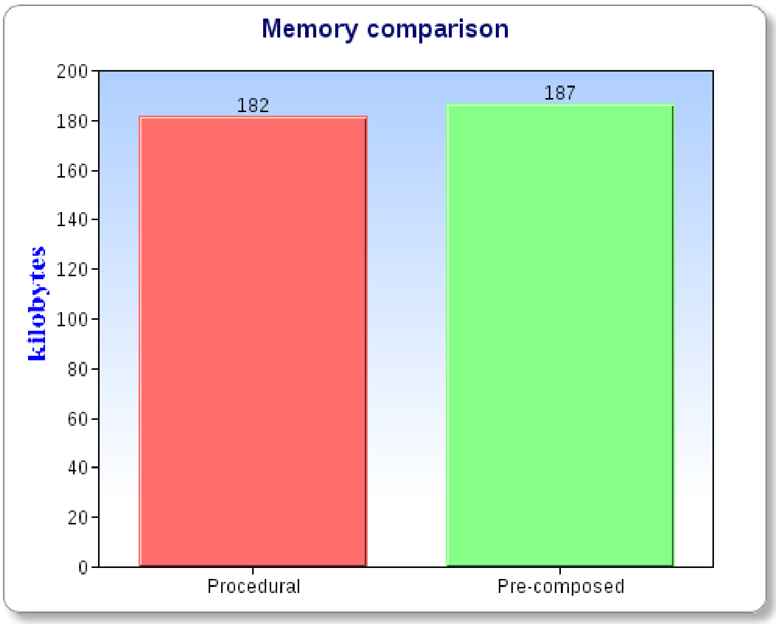


Figure 9: The amount of memory used by both versions of the game.

As can be seen. both implementations of the music system have very similar memory usage. The procedural method actually used 5k less memory than the pre-composed. This was a surprising result. It was thought that as the procedural system handled lots of small sections at a time it would add up to more memory usage compared with handling a single large file. All of the music files used were recorded with the same sample rate and bit depth to help keep this test as a fair comparison. The result shows that by implementing an effective procedural music system the memory requirements of a game do not have to suffer.

The second performance test was concerned with processing times. As there are a lot of stages involved in the procedural music system, this test would determine the efficiency of the techniques used and whether they could be considered by developers. The first result we shall discuss is the start-up time of the application (fig. 10). Note that 1 tick equals 69 nanoseconds.

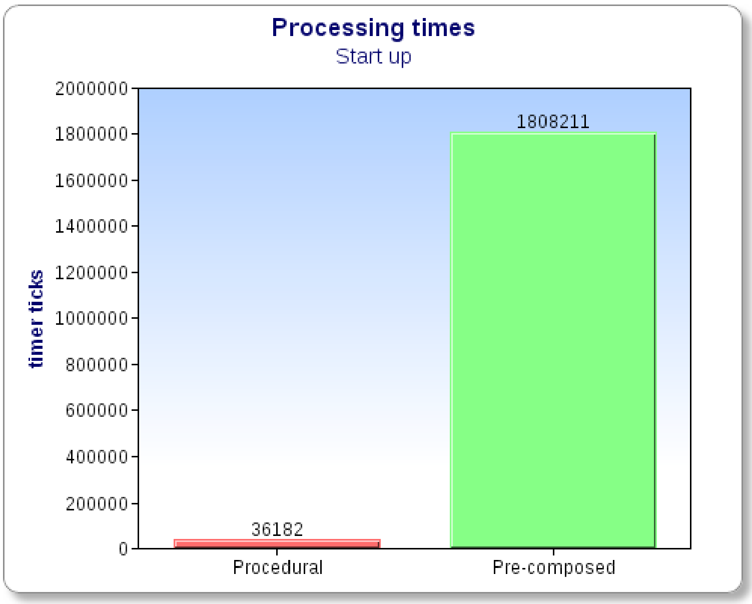


Figure 10: The start-up time of both versions of the game.

The difference in start-up time was very significant. The procedural system started almost 50 times faster than the pre-composed version. The pre-composed version has to process a 4 minute sound file on start-up which is causing this long processing time. In comparison, the procedural system only has to process 3 sound files which are 3 seconds in length. Although, before this can happen the procedural system must intelligently select these 3 riffs. If the techniques created in this project were used in a game, loading times could potentially drop which is another advantage for the adoption of procedural music.

The second processing time result (fig. 11) shows the average processing time of each music system after the initial start-up.

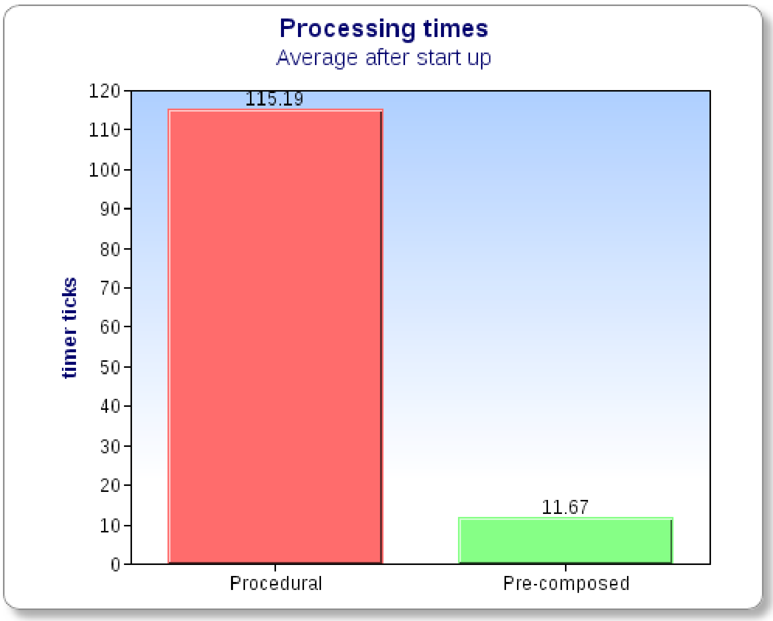


Figure 11: The average processing time after start-up.

The results of this test were as expected. Due to the extra calculations involved in the procedural music system the average processing time is greater, taking almost 10 times as long. To put it into perspective, this extra time equates to 0.007 milliseconds. This result along with the next result we shall discuss, pointed out an inefficiency in the riff selection technique. A revised method is believed to be far more efficient and could significantly drop the processing times for the procedural technique. This revised method shall be discussed after the results. However, at the moment the result stands and acts as a downfall for the procedural techniques developed. For high end games which are trying to utilise limited resources, procedural music could be seen as an extra strain on those resources.

The final result (fig. 12) shows the highest processing time observed after start-up during 2 minutes of gameplay. The results observed were very sporadic. The procedural system showed a range of values from 34312 to 216675. The bar chart (fig. 12) shows the average of 15 tests.

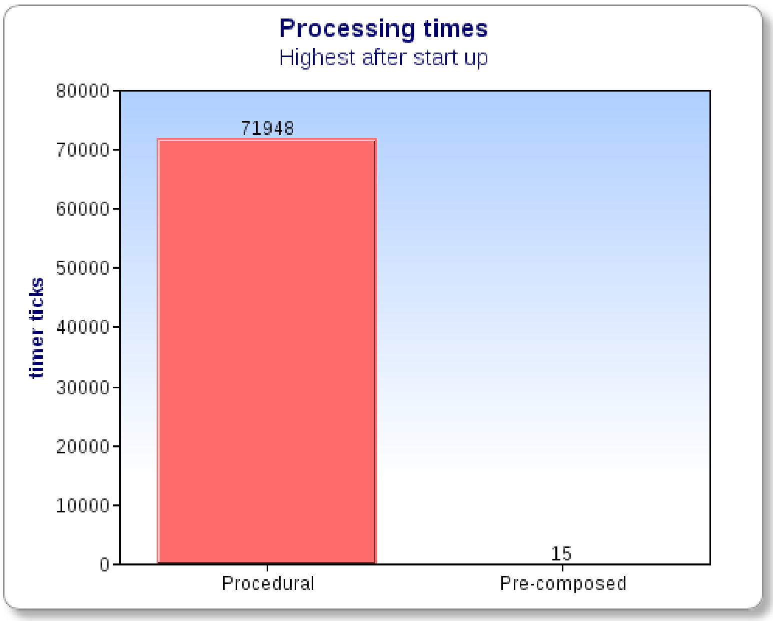


Figure 12: The highest processing time after start up.

The pre-composed result is very close to the average seen in the previous test. This shows that after start up there are no complications involved with playing pre-composed music. However the procedural result shows a significant change. This prompted further testing which found the problem to be caused by the riff selection system.

The task of the riff selection system is to search through a bank for an appropriate riff which has the right franticness value and a start note close to the last note played. A linear search could not be used as it would always select the same riffs. Instead the system checks the attributes of a random riff in the bank. If the selection is not a match, the system tries another random number, making sure not to pick the same one. This process continues until either a match is found or the system has tried 100 times, after which the system looks for a rougher match. After a further 50 attempts the system selects a riff which has a generic franticness value and start note position. However this has never occurred. This random search method is causing the large variance in the processing times of the procedural system. Sometimes a match will be found straight away, other times over 100 checks could be performed.

A far more effective way of searching was the method implemented for the selection of the bass riffs. Instead of storing all of the riffs in the same bank, the bass riffs were stored in banks according to their franticness rating. This meant that the system could search through a specific bank looking only for a good match in note continuity. The number of items it would search through is thereby significantly reduced. Another improvement could see the search method track its checks. This would allow the system to never select the same number twice and if a very good match has not been located after a number of tries, the best match it came across could be used. By significantly reducing the number of searches, it is thought that the processing times in all stages would significantly drop. This would present the procedural music system as a more viable and effective method to be used in games.

By analysing the results of the user and technical tests we gained insight into the performance of the procedural music system and its successfulness. The techniques used indeed create an output having musical properties which adhere to the fundamental rules of music composition. The effect of the procedural music on players was a positive one. In comparison to the traditionally used pre-composed music, players had greater immersion and enjoyment in the game. Testing also showed a flaw in the implementation and led to the discovery of a more efficient method.

Conclusion

There were many important findings that came from the completion of this project. By first exploring the techniques used by composers and possible new approaches, certain techniques were found to possess attributes which could prove useful. However, by testing these techniques in a game environment, many were deemed unfit for development. The original plan to use Markov chains to create melodies was abandoned due to its inefficiency in both development time and results. Instead, by drawing on the strengths of both pre-composed and procedural music, the more effective 'Fuzzy Riff System' was developed. This achieved the project aim of developing a dynamic music system suitable for use in games.

Another aim of the project was to implement such a system in a playable prototype game. This allowed the procedural music system to demonstrate its use in computer games. The game also serves as a testing environment. By using a simple 2D platform game template, development and research time were focused on the successful completion and refinement of the music system. The efficiency of the techniques which make up the 'Fuzzy Riff System' were tested. This revealed that the system indeed creates a musical output which can conform to the basic rules of composition. Testing also revealed that the system can effectively react to changes in game variables to produce dynamic music. This was achieved with the use of mood levels which are controlled by a 'Fuzzy Logic' system.

A very important finding of the project was the results from the testing stage which saw the procedural music system compared with a linear pre-composed track. The results showed that the procedural music is reflective of gameplay and enhances enjoyment of the game. It had been feared that the procedural techniques could not produce an output which was as musically pleasing as pre-composed music. The test results showed that this was not the case. Testing also proved that the riff system used did not create repetitive music. On a technical side, the results showed a flaw in the riff selection system and allowed for a more efficient method to be devised. With this revision the procedural system could be efficient in terms of both memory usage and processing time, presenting itself as an effective technique which could be adopted by games.

The question which was driving the research and development of this project was,

> "*How can procedural music best be implemented into a video game to make use of game statistics and events to create dynamic music reflective of gameplay in real-time?*"

On completion of the project, it is believed that by using a large bank of small riffs a procedural system can create music which has the same musical qualities as pre-composed methods. By assigning attributes to these riffs in terms of note values and their mood, a procedural system can intelligently select riffs for playback according to in-game events. The result is dynamic music which is reflective of gameplay. In achieving the aims of the project and creating an effective prototype for a procedural music system, it is believed that this project was a success.

Recommendations and future work

In light of the project certain recommendations can be made as to the best uses for the 'Fuzzy Riff System'. The technique requires that many riffs are carefully created and given attributes. This is a time consuming task which could discourage developers from adopting the technique. However, the system could be used to develop a powerful middleware. Currently developers are already utilising audio middleware programs such as 'fmod' to help create dynamic music. The proposed idea would be to create a program which has a large diverse library of riffs which already have many attributes associated with them. The program could be used in similar fashion to 'fmod': the game would simply feed the program the required mood levels in real-time and the middleware would output dynamic music based on these criteria. This would greatly cut down on development time and allow composers to concentrate on the important areas of the game. To allow for greater freedom, the middleware could also let composers create their own riffs and assign custom mood values using a friendly user interface. This would certainly be a powerful tool for creating dynamic music.

A technique which could improve the re-activeness of the procedural music and allow for finer control, would be the addition of an artificial intelligence system which could alter the riffs slightly during playback. For example, the system could alter key notes in a riff to change it from a major to minor key. This would involve storing all of the note data for the riffs which could then be interpreted by an AI system. This system would decide whether to omit, add or alter notes to create the required musical mood. This is similar to the "Riffology" technique created by composer Peter Langston. This technique would require a very complex AI system but if it was well implemented the advantages would be great.

The research undertaken in this project was important as music tends to be a neglected area of computer game research and improvement, with priority being given to advances in graphics. This may be due to developers not being fully aware of the effects music can have.

> "*The impact music can have may appear to be small to those unfamiliar with the field, but a simple glance at previous work demonstrates just how significant an impact music can have on the thoughts and actions of its listeners.*" (Rossoff 2007, p. 5)

If more research, time and effort were put into developing ideas such as procedural music, games could greatly benefit. Once solid procedural models are established they could save developers time and money whilst creating a more immersive and enjoyable games.

APPENDICES

Appendix A - Honours project proposal

Abstract

It has been concluded through the research of many composers' works that procedural algorithms can indeed produce pleasing music. This project will research how one can implement procedural music in a game environment to be reflective of in game events in real time. A 2D platform game will be created with algorithmic music as the sole source of music. 'Markov chains' along with 'knowledge based systems' and 'musical grammars' have been chosen as the main methods for the implementation of the procedural music. The expected outcome for the application is for it to produce satisfactory music affected by in-game states which, when compared to the currently adopted method of pre-recorded tracks, won't be as musically coherent or pleasing. Difficulties come from affecting the algorithms, which are based on time, with events not restricted to time. Also how to mathematically map in game variables to meaningful musical qualities.

Proposal Contents

Introduction

Procedural music is a concept that has been around since the middle ages (Roads. 1996). As times have moved on with the advent of computers, does this technique have a place in computer games? Compared to the standard method of sampled music in games which are linear and sound the same every time, procedurally generated music can be dynamic and change in real time according to live input. Computer games are a medium which could greatly benefit from the use of this. Compared to film where the composer knows exactly when something interesting is going to happen and are able to construct the music around that point, computer games are controlled by the player who is unpredictable (McCuskey 2003).
Imagine playing your favourite game with your own personalised sound track, matching and adapting to the way you play. Also, for large open world games with hundreds of hours of gameplay, non-repeating music would become a viable option compared to creating hundreds of hours of composed music.

Although computer generated music is not a new topic, it has not been adopted by the computer game industry and there is very little in the way of games based research and tested techniques to base the generation of the proposed application on. Therefore the research will look to techniques used by composers and how they can be implemented in a real time game environment, techniques such as: Markov chains, neural networks, fractals and knowledge based systems which will be discussed later in the paper. To achieve a synchronous relationship between the music and how the player is acting, these algorithms must be affected by in game events. This could be the mapping of certain game statistics and values to musical characteristics that share some sort of resemblance. For example the number of enemies on screen relating to the frequency of notes produced by the algorithms, conveying a sense of calm or franticness. This poses a problem as video game players usually have a large level of freedom as to when they perform actions. The musical grammars which will be implemented in the game on the other hand will have very strict conditions relating to time. Another potential problem, which integrating with computer games poses, is that the CPU intensive algorithms will have to compete for resources with other areas such as graphics and physics. Therefore, efficiency will have to be taken into account.

Research Question

With the added complexities of the stated problems and the many others which may later present themselves, the question guiding this research will be:

"*How can procedural music algorithms best be implemented into a video game to use player statistics and game events to create dynamic music reflective of gameplay in real-time?*"

The projects' objectives are to:

- Research algorithmic music techniques, selecting appropriate methods for implementation in a game environment.

- Create a playable video game with procedurally generated music which reacts to in game events.

- Evaluate the benefits and drawbacks of procedural music within games by comparison to standard pre-recorded techniques.

- Recommend the best techniques and approaches to creating dynamic procedural music in games and future improvements.

Background/Literature Review

Dynamic and adaptive music is something composers have strived to achieve for hundreds of years. In the classical period musical parlour games such as 'Musikalisches Würfelspiel' were created, one of the most famous being attributed to Mozart. The composer would write small sections of music that would be randomly stitched together according to dice roles. The sections of music had to be carefully crafted to produce coherence, as a game with an average sixteen measures could produce around 46 quadrillion works. With the advent of computers, composers realised the potential for algorithmic composition as a computer could be programed to make use of complex and repetitive mathematics to study and create music. Lejaren Hiller was a pioneer of this approach to music, writing programs for the Illiac computer with the aid of Leonard Isaacson. In 1956 they composed the *Illiac Suite for String Quartet* (Cope 2001).

Many would argue that it is impossible for a computer which has not gone through the trials of life to ever create music with meaning. Hofstadter said,

> "*Music is a language of emotions, and until programs have emotions as complex as ours, there is no way a program will write anything beautiful.*" (Hofstadter 2001, p.36)

However, after hearing a mazurka in the style of Chopin, composed by the computer program 'Emmy' created by David Cope he went on to say,

> "*It was new, it was unmistakably Chopin-like in spirit, and it was not emotionally empty. I was truly shaken.*" (Hofstadter 2001, p.38)

Hofstadter, now a believer, went on to give many talks on computer generated music and in doing so showed others that computer generated music could be pleasing and appear to have semantics. A quote from Langston sums this up as,

> "*This testifies to the ability of human consciousness to find meaning in the meaningless*." (Langston 1988, p.17)

Music in games has been getting more attention as of late. There are games like "Red Dead Redemption" (RoackStar Games 2010) which was nominated for 10 G.A.N.G awards (Game audio network guild) and big names such as Hans Zimmer coming into the industry. Music and audio play a larger part in video games than they do for films where the unheard melodies (Gorbman 1987) act mostly as an accompaniment to the on screen narrative. Whereas, in games the sound plays a more active role which must be given the player's attention in order to make the correct response (Collins 2007). For this reason perhaps games are not suited to algorithmic music because, as well as the algorithms having to overcome the challenge of producing pleasing and coherent music, they must also now provide the player with important narrative information. However, this is contradicted by Collins who says,

> "*Video games are perhaps an ideal media form for procedural music; after all, many elements of gameplay - especially the timing of events and actions - are unpredictable and occur in a non-linear fashion.*" (Collins 2009, p.5)

Perhaps both arguments are right. More traditional methods of composition will still be needed where a game requires specific emphasis or player reaction, but for ambient or immersing music, which doesn't play a large part in the narrative, the many benefits of music created procedurally in real-time could be great.

A large factor holding back the adoption of procedurally generated music is its realism and quality. Games in the past, that used algorithmic music, either used the computer to synthesise the instruments or adopted MIDI instruments. Although this gave a high flexibility for controlling the performance, the sounds were of poor quality compared to pre-recorded audio. This is backed up by Collins who says,

> "*One of the largest fears since the advent of Redbook audio in games in the late 1980s is that a return to MIDI will disappoint audiences who have now grown accustomed to high-quality samples and even live orchestras in games.*"
> (Collins 2009, p.12)

One very interesting conjecture to this argument is that visual graphics did not receive the same response.

> "*Nobody ever said to the developers of 3D games in 1995, "look guys, these 3D graphics you're producing aren't very realistic, why don't you do it like Myst?". Having seen photo-realistic scenery they were not spoiled by it and understood the benefits of real-time interactive content even if it wasn't yet realistic.*"
> (Farnell 2007, p.24)

Since then, in-game visuals have advanced dramatically to a very realistic level. If generated sounds had been treated the same way, would they have advanced at a similar pace? The same could be said for the musical qualities of algorithmic music. With more composers and programmers behind it and with budget and support, algorithmic music could see itself on the right side of game composer Pedro Camacho's belief which is stated on his blog as "*There are only two types of music, good and bad.*" (Camacho [No date]).

Although most current games are using linear music, it is not to say they have no dynamic elements. Powerful audio engines, such as Firelight Technologies' 'fmod', are becoming widely used in the games industry. These allow composers to create dynamic compositions, with branching structures which are effected by in game events. This can allow a game to smoothly switch between and layer several pre-recorded tracks (horizontal and vertical re-orchestration), however the variations are limited to the amount created by the composer. Procedural music on the other hand, can have countless variations with a much higher level of flexibility. If games did start to adopt procedural music, we would see the completion of the infamous full cycle trend. Games originally generated all the music on the processor and then in the mid-1990s the music moved to recorded tracks on disk (*McAlpine et al 2009)*. Procedural music and any form of more dynamic music for that matter will see itself slowly move back to the processor, which makes you wonder why it left in the first place.

The techniques used by computer composers to create algorithmic music are known and discussed, however they are discussed from a musical point of view. This project is more concerned with their adaptability and suitability for games, for example efficiency and flexibility. Although we cannot directly look to research, we can compare the various methods to similar techniques used in other areas of games, like graphics. Markov chains are used throughout games in areas of chance for example, in artificial intelligence. Due to their controlled random output, which is based on past outputs, it can be used to model systems that change through time in a non-deterministic way such as simulating weather (Dreiseitl 2010). Musically they have also been found to be very useful in producing chord progressions and melodies, which should always be based on what has been played prior in the music. With the flexibility of being able to have several layers of dependency and customisable weightings, which could include in game variables, this algorithm should work well in a game environment.

Neural networks are used by composers due to their ability to learn. The system, which is initially blank, is given knowledge through training with example inputs that expect certain outputs (Farnell 2007). By training these systems, composers can use the neural networks for very specific purposes with great results. However, in a game environment we cannot have the player train the algorithm as they play the game. Pre-training in the games' development stage would leave the neural network very specific and tailored to the programmer’s play-style instead of the players, for this reason it does not present itself as a viable technique for games.

Fractals have found many uses in computer graphics, especially in procedural generation. For example, generating realistic plants. However, it has been found that although fractal methods lead to the generation of meaningful geometry, they might not produce meaningful music (Farnell 2007). As the project is looking to generate music to the best of its ability, methods which may be proven to work well in games are not necessarily useful, unless they can produce music to a high standard.

Composers use what they call musical grammars when generating algorithmic music. The grammars are a set of mathematical rules relating to music which will define the generation of the harmonies and rhythms. This is basically a way of coding the rules and restrictions composers have to adhere to when creating music. As these will be a fundamental set of rules, they should not have any difficulties being implemented in a game environment, also these rules are of great importance to the music and therefore cannot be omitted.

Although the aim for the procedural algorithms is to create new music based on game states, it does not mean they cannot use existing musical structures and patterns as a basis. Knowledge based systems do just that. A bank of chord progressions and musical riffs can be stored so other techniques such as Markov chains can look to them for guidance when calculating the percentage change of playing certain notes or chords. By doing this, the music is more likely to be coherent and pleasing, whilst the endless amounts of possibilities are not lost, they are simply guided, much in the same way as a jazz musician uses past experience when improvising.

By evaluating the techniques employed by composers certain methods seem to be more appropriate form a computer games point of view. These techniques can be selected and their intended use discussed in the next section.

Methodology

Research design

The final product of the project will be a small 2D platform game where procedurally generated music is the only music source. The same game will be implemented with linear pre-composed music as well. This will serve as a basis to judge the two techniques.

The procedural music will be implemented to react to changes within the game in real time to help further discover its use for games. This will be done by making in-game variables such as health play a role in the equations of the music algorithms. The difficulty comes from using the game variables in such a way as to produce meaningful musical results.

The main techniques that have been chosen for implementation are; Markov chains, grammars and knowledge based systems. This is due to their high desirability from a musical point of view and their ability to merge well into a game environment. The first of these to be implemented will be a Markov chain which will control a simple bassline to play in sync with the player's footsteps in game. The use of grammars and a knowledge based system will then help to shape the bassline to be more musically structured. A similar approach will be taken when implementing the chord progressions, harmonies and melodies. The process will be an iterative one which will be assessed and improved on until the result is musically successful.

A twelve bar blues theme will be used to shape the structure and rules of the musical algorithms to aid in the creation of a pleasing and coherent piece of music. The twelve bar blues was chosen due to its easily programmable rules at the base level with a large scope for more complex structural additions. As it would seem that for music to be interesting it must have structure on many levels (Langston 1986, p.5). The next section will give a more detailed description of how a music algorithm can be created.

Strategy

The project will be created by adapting an existing 2D platform game in XNA to create a playable level which lends itself to linking in game events to music. The chosen algorithms will control the playback of single sampled instrument notes as opposed to using MIDI or generative instruments for a more realistic sound.

Let us look at how a Markov chain will be used to create a simple bassline. The table in (Figure 1) is a representation of a first order Markov chain where C, D, E… represent musical notes. The chance of what note will be played next is based off the previously played note. For example if a C was played last there is a higher change the next note will be a D compared to another C. The weightings will be defined in the implementation stage so musical knowledge is needed by the algorithms' creator.

(Figure 2) shows a second order Markov chain where the last 2 notes played are taken into account. This allows finer control of the output; the same is said for third and fourth order Markov chains. The Proposed application will use second order or greater for the control of the bassline.

	Next note	**C**	**D**	**E**	**F**	**G**
Last note						
C		15%	30%	30%	15%	10%
D		15%	10%	30%	30%	15%
E		20%	15%	15%	20%	30%
F		10%	20%	30%	10%	30%
G		15%	10%	20%	30%	20%

(Figure 1, 1st order)

	Next note	**C**	**D**	**E**	**F**	**G**
Last 2 Notes						
CC		0%	40%	35%	15%	5%
CD		15%	20%	20%	30%	15%
CE		20%	10%	15%	30%	25%
CF		5%	15%	30%	10%	40%
CG		15%	10%	25%	30%	20%

(Figure 2, 2nd order)

This simple bassline will be the first aspect of the music to be implemented and will be mapped to the players in game footsteps. The music will become more structured when the logic of a twelve bar blues theme is implemented to guide the chord progressions and rhythm. By using Grammars and knowledge based systems the bassline will again become more structured and musically pleasing. Similar approaches will be used when creating the drum beat, chords and melody.

A separate sound track will be composed using the traditional methods adopted by today's games. This will be used in the analysis stage to compare the two techniques along with the testing of the algorithms themselves.

Data Collection and analysis

To evaluate the procedural music two main approaches will be used;
Firstly creating in game conditions which would expect a certain output from the algorithms. For example increasing the amount of enemies to see if the output successfully reflects this with higher note output. This will test whether the mappings between in game events and the musical algorithms are successful.

To evaluate the effectiveness of the procedural music it will be compared to pre-recorded music by means of a user test. This is due to the fact that whether music is good or not is subjective and down to personal opinion. This user test will take the form of a questionnaire, which will be answered by several participants after playing the game with both the procedural music and a pre-composed musical. They will then be asked questions such as, which play though they found more enjoyable/immersive which will be followed by them rating the music in several categories.

With these findings and the prior research, a conclusion will be drawn as to which techniques are suitable for creating procedural music within computer games and how they can be best implemented to reflect game play.

Summary

As this proposal has shown, there is a distinct lack of adoption within the games industry for procedurally generated music. One can look to many composes through the ages to see that computers can indeed algorithmically create music which is pleasing. There are potentially many benefits to using such an adaptive technique in a medium which strives for interaction and a personalised player experience.

The project hopes to find how procedural music can be implemented into a computer game to make best use of the adaptive environment and how this will impact on the playing experience. The technical difficulties the medium poses will be explored and recommendations will be made as to how they can be best approached stating possible improvements that further work could bring.

References

Camacho, P.M. [No date] [online] composersforum Available from:
http://composersforum.ning.com/profile/PedroMacedoCamacho
[Accessed 30 Nov 2011]

Collins, K., Hawkins,S and Richardson, J, eds. 2007. *An Introduction to the Participatory and Non-Linear Aspects of Video.* Helsinki: Helsinki University Press. pp. 263-298.

Collins, K. 2009. An Introduction to Procedural Music in Video Games. *Contemporary Music Review*. 28(1), pp. 5 – 15.

Cope, D., Hofstadter, D. 2001. *Virtual Music*. London: The MIT Press

Dreiseitl, S. 2010. *Artificial Intelligence Markov Chain.* [Online]
http://www.umit.at/dataarchive/data91/ai_lecture12_markovchains_2up.pdf
[Accessed 4 Dec 2011]

Farnell, A. 2007. *An introduction to procedural audio and its application in computer games*. [online] Avaliable from:
http://obiwannabe.co.uk/html/papers/proc-audio/proc-audio.pdf
[Accessed 30 Nov 2011]

Firelight Technologies. 2011. *fmod* [software] available from: http://fmod.org/
[Accessed 30 Nov 2011]

Gorbman, C. 1987. *Unheard melodies, narrative film music*. London: BFI Publishing

Langston P. 1986. *(201) 644-2332 or Eedie & Eddie on the wire: An experiment in music generation.* New Jersey: Bellcore

Langston, P. S. 1988. *Six Techniques for Algorithmic Music Composition.* New Jersey: Bellcore

Mcalpine, K., Bett, M. and Scanlan. J. 2009. *Approaches to creating real-time adaptive music in Interactive entertainment: a musical perspective*. Paper presented at: AES 35th International Conference, London, UK, 11–13 February.

McCuskey, M., Lamothe, A. ed. 2003. *Beginning game audio programming.* Boston: Premier Press

Roads, C. 1996. *The Computer Music Tutorial*. Cambridge, Massachusetts: MIT Press

RockStar North., RockStar San Diego. 2010. *Red Dead Redemption*. [Disk]. Microsoft Xbox 360. New York: RoackStar Games

Bibliography

Booth, J. 2004. *A DirectMusic case study for Asheron's Call 2: The Fallen Kings.* In T. M. Fay, S. Selfon & T. J. Fay. eds., *DirectX 9 audio exposed: Interactive audio development.* pp. 473–498. Plano, TX: Wordware Publishing.

Brown, A. R. & Kerr, T. 2009. *Adaptive Music Techniques. Improvise*: The Australasian Computer Music Conference, Brisbane, Australia: pp. 26-31.

Humphrey, A. 2008. Algorithmic content in casual games. *Casual Games Quarterly*. 3(1), pp. 5 – 7. [online] Available from: http://archives.igda.org/casual/quarterly/3_1/igda_casual_game_quarterly_3_1.pdf. [Accessed 30 Nov 2011]

Wooller, R., A. R. Brown, et al. (2005). *A framework for comparison of processes in algorithmic music systems*. Generative Arts Practice, Sydney, Creativity and Cognition Studios Press, pp. 109-124.

End of Appendix A

Appendix B - The steps of Fuzzy Logic

A fuzzy logic controller consists of four main stages: fuzzification, rule base, inference mechanism and defuzzification which can be seen in figure 1 below (Karakose and Akin 2010)

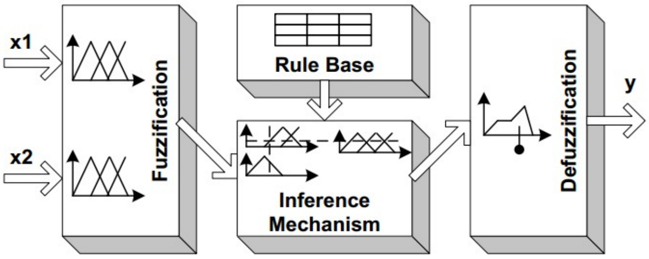


Figure. A1: Main stages of fuzzy Logic (MathWorks [No date])

Stage One: Fuzzification

The first step is to convert our various input data which will be in a crisp form into fuzzy values. These values will have a degree of membership to the various general states the input could belong to, which are called membership functions (Mfs). As an example (fig. 2) the distance a car is from a racing line could be naturally described as being 'Far Left', 'Close Left' etc. There are overlaps for all of the Mfs which lend to the smooth interpretation of the data. As can be seen, this example uses a triangular shape for the Mfs but a variety of other options are available, for example trapezoids and Gaussian curves. However the shape is generally less important compared to the number of Mfs and their placement within the 'universe of discourse' (Sivakotiah and Rekha 2011). The 'universe of discourse' is the possible range of an input value, in this example from -10 to 10. Three to seven Mfs are generally appropriate to cover this 'universe of discourse' (Sivakotiah and Rekha 2011). Here we use seven to help create a smoother output. The more Mfs which are used the more rules have to be created. This can become an overwhelming and hard to follow task if a large number is used.

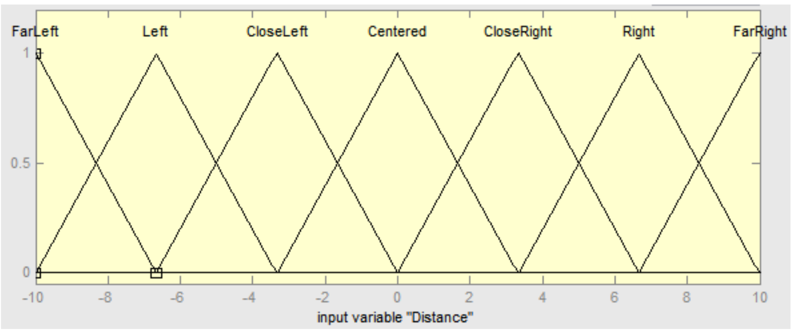


Figure. A2: Representation of membership functions

Stage two: Rule Base

A FIS is built upon set of rules. These rules are based

on the possible inputs, or combination of inputs, and the output which should be associated with them. These rules are created in a manner which is close to the human way of thinking about the situations (if A and B then C). For example, if the car is to the right of the racing line and the track is now making a left turn the logical course of action for the car to take is to make a sharp left turn to get back on track. When inputs are fed into the system they are compared with the (if A and B) blocks. One set of inputs can satisfy more than one rule set due to the overlap in Mfs (fig. 2). The consequent (then C) blocks of some rules are then satisfied (Kaehler 1998). If there is more than one part contributing to a given output state, the conclusions are then combined to form logical sums. This process can be done using different methods of combination. In Matlab we have 4 built in options for this. An AND or OR operation, each with their own methods which can be chosen. AND can use 'minimum' or 'product' and OR has 'maximum' or 'probabilistic or'. The input to these functions are two or more membership values from the fuzzified input variables and the output is a single truth value (MathWorks [No date]) this can be better understood with the below diagram (fig. 3) which uses the 'max' method. These resultant values are then fed into the inference process.

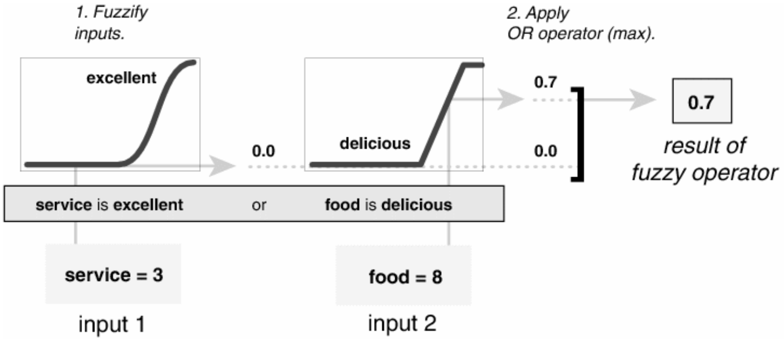


Figure. A3: Max method of combining fuzzy inputs (MathWorks [No date])

Stage Three: Inference Mechanism

Once we have determined the weight or 'firing strength' for every rule, which will be a value from zero to one, where one is fully engaged and zero is weightless, we execute the implication method. The implication method takes the value of truth found for a rule state (antecedent) and outputs a fuzzy set, reshaping the Mfs of the output (consequent). This can be visualised in figure 4 below. This process is done for every output state. There are several techniques which can be used for this process. In Matlab there are two built in methods: 'minimum', which will truncate the output of the fuzzy set and a 'product', which will scale the output of the fuzzy set (MathWorks [No date]).

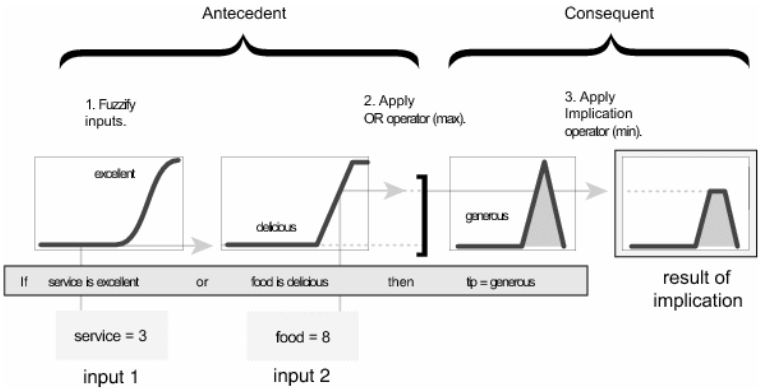


Figure. A4: Implication stage, finding the firing strengths of rules (MathWorks [No date])

Next, all of the fuzzy sets produced in the implication method have to be combined in some manner to produce a single fuzzy set which represents the whole system. This is known as the aggregation method and is depicted in figure 5. Again there are different approaches which can be taken to combine the fuzzy sets. Matlab offers 3 by default techniques:

- maximum: uses the highest value produced by all of the fuzzy sets for every point across the 'universe of discourse'.
- sum: simply sums all of the fuzzy sets together to create one final fuzzy set.
- probor: best described with an example, for two values the 'probor' method will look like this probor(a,b) = (a + b – ab).

The following diagram (fig. 5) shows three fuzzy sets created by the implication method being combined/aggregated into a single fuzzy set. The maximum technique is used.

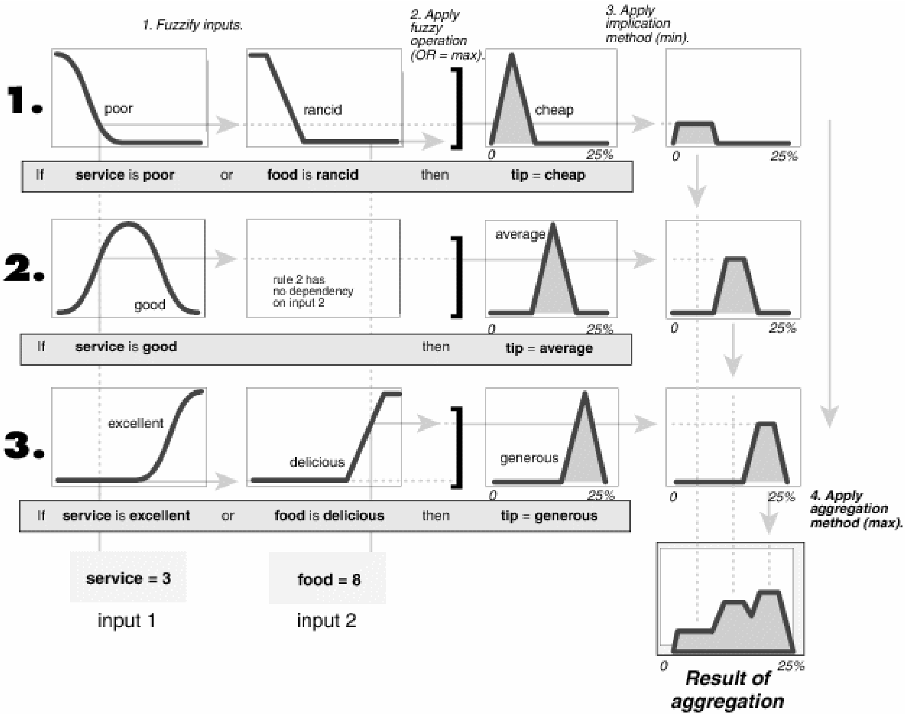

1. Fuzzify inputs.
2. Apply fuzzy operation (OR = max).
3. Apply implication method (min).
1.
poor
rancid
cheap
0
25%
If service is poor or food is rancid then tip = cheap
2.
good
rule 2 has no dependency on input 2
average
If service is good then tip = average
3.
excellent
delicious
generous
If service is excellent or food is delicious then tip = generous
4. Apply aggregation method (max).
service = 3
input 1
food = 8
input 2
Result of aggregation

Figure. A5: Aggregation stage which combines all the fuzzy sets (MathWorks [No date])

Stage Four: Defuzzification

In this final stage we need to convert our fuzzy data back into usable crisp data. The entire aggregated set needs to be represented by a single value. There are several defuzzification methods but perhaps the most common is the centroid method (MathWorks [No date]), which calculates the centre point of the final fuzzy set's area and this value will be our output. Figure 6 shows a representation of this in continuation with the other diagrams.

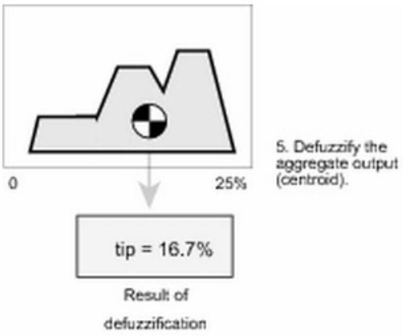
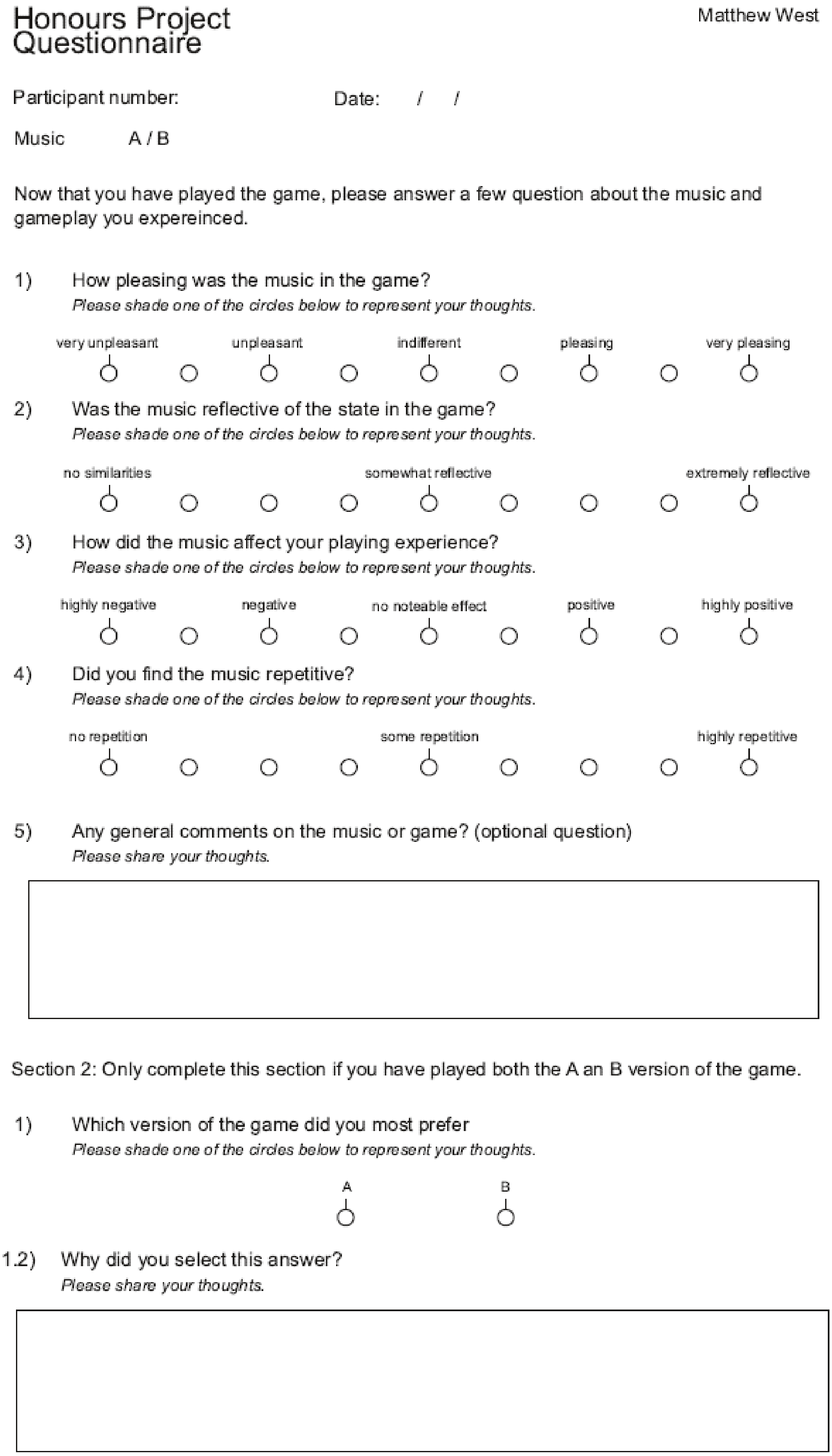

0
25%
5. Defuzzify the aggregate output (centroid).
tip = 16.7%
Result of defuzzification

Figure. A6: Defuzzification, using a centroid approach. Final output of system (MathWorks [No date])

Appendix C - Questionnaire



Honours Project Questionnaire

Matthew West

Participant number: Date: / /

Music A / B

Now that you have played the game, please answer a few question about the music and gameplay you expereinced.

1) How pleasing was the music in the game?
Please shade one of the circles below to represent your thoughts.

very unpleasant		unpleasant		indifferent		pleasing		very pleasing
○	○	○	○	○	○	○	○	○

2) Was the music reflective of the state in the game?
Please shade one of the circles below to represent your thoughts.

no similarities				somewhat reflective				extremely reflective
○	○	○	○	○	○	○	○	○

3) How did the music affect your playing experience?
Please shade one of the circles below to represent your thoughts.

highly negative		negative		no noteable effect		positive		highly positive
○	○	○	○	○	○	○	○	○

4) Did you find the music repetitive?
Please shade one of the circles below to represent your thoughts.

no repetition				some repetition				highly repetitive
○	○	○	○	○	○	○	○	○

5) Any general comments on the music or game? (optional question)
Please share your thoughts.

Section 2: Only complete this section if you have played both the A an B version of the game.

1) Which version of the game did you most prefer
Please shade one of the circles below to represent your thoughts.

A	B
○	○

1.2) Why did you select this answer?
Please share your thoughts.

Appendix D - Supplied media on disc

The disk located on the inside of the back cover contains the following media.

1. The finished playable prototype with procedural music.

2. The finished playable prototype with pre-composed music.

3. A video of gameplay - taken from the completed game with procedural music.

4. A video of gameplay - taken from the completed game with pre-composed music.

5. The project folder containing the code for the created application.

6. A digital .pdf version of this document.

REFERENCES

References

Bourg, M.D. and Seeman, G. 2004. *AI for Game Developers*. California: O'Reilly Media.

Bower, K.M. and Colton, J.A. 2003. *Why We Don't "Accept" the Null Hypothesis.* Technometrics, p.2-4. [online] Available from: http://www.minitab.com/uploadedFiles/Shared_Resources/Documents/Articles/not_accepting_null_hypothesis.pdf [Accessed 6th May 2012].

Broomhall, J. 2011. AES Audio for Games Conference. *Perfect Partners and Unsung Heroes: 20 Years of Game Audio's Inextricable Link between Creativity and Technology.* February 3rd. London: Bafta Building.

Collins, K. 2009. An Introduction to Procedural Music in Video Games. *Contemporary Music Review*. 28(1), pp. 5 – 15.

Collins, K., Hawkins,S and Richardson, J, eds. 2007. *An Introduction to the Participatory and Non-Linear Aspects of Video*. Helsinki: Helsinki University Press. pp. 263-298.

Cope, D., Hofstadter, D. 2001. *Virtual Music*. London: The MIT Press.

Crowle, R. 2011. *Dynamic Music In Video Games.* [online] Available from: http://www.robcrowle.com/projects/thesis.pdf [Accessed 6th May 2012].

Dreiseitl, S. 2010. *Artificial Intelligence Markov Chain.* [Online] http://www.umit.at/dataarchive/data91/ai_lecture12_markovchains_2up.pdf [Accessed 6th May 2012].

Edwards, M. 2010. *Algorithmic Composition: Computational Thinking in Music.* [online] Available from: http://people.ace.ed.ac.uk/staff/medward2/algorithmic-composition.pdf [Accessed 6th May 2012].

Farnell, A. 2007. *An introduction to procedural audio and its application in computer games.* [online] Available from: http://obiwannabe.co.uk/html/papers/proc-audio/proc-audio.pdf [Accessed 6th May 2012].

Firelight Technologies. 2011. fmod [software] Available from: http://fmod.org/ [Accessed 6th May 2012].

Fournel, N. 2011. GDC 2011. *Procedural Audio: Challenges & Opportunities.* March 8th. San Francisco: Moscone Center.

Gorbman, C. 1987. *Unheard melodies, narrative film music.* London: BFI Publishing.

Heckroth, J. 2001. *Tutorial: MIDI and Music Synthesis.* [online] Available from: http://www.midi.org/aboutmidi/tut_midimusicsynth.php [Accessed 6th May 2012].

Insomniac Games. 2008. Resistance 2. [Disk]. Sony Playstation 3. Sony Computer Entertainment.

Kaehler, S.D. 1998. *Fuzzy Logic Tutorial.* [online] Available from: http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html [Accessed 6th May 2012].

Karakose, M. and Akin, E. 2010. Block based fuzzy controllers. *International Journal Of Research And Reviews In Applied Sciences.* 3(1), pp. 100 – 110.

Klappenbach, M. [no date]. *What is a Platformer?*[online] Available from: http://compactiongames.about.com/od/gameindex/a/platformer_def.htm [Accessed 6th May 2012].

Kosak, D. 2008. The Beat Goes on: Dynamic Music in *Spor.* [online] Available from: http://uk.pc.gamespy.com/pc/spore/853810p2.html [Accessed 6th May 2012].

Langston, P. S. 1988. *Six Techniques for Algorithmic Music Composition.* New Jersey: Bellcore.

Mathworks. [No date]. *Fuzzy Inference System* [online]. Available from: http://www.mathworks.co.uk/help/toolbox/fuzzy/fp351dup8.html [Accessed 6th May 2012].

McCuskey, M., Lamothe, A. ed. 2003. *Beginning game audio programming.* Boston: Premier Press.

McWertor, M. 2008. *Resistance 2 takes 420 Hours to Complete.* [online] Available from: http://kotaku.com/5072781/resistance-2-takes-420-hours-to-complete [Accessed 6th May 2012].

MSDN. 2010. *DateTime.Now Property.* [online] Available from: http://msdn.microsoft.com/en-us/library/system.datetime.now.aspx [Accessed 6th May 2012].

Peng, W., Li, T. and Ogihara, M. 2007. *Music Clustering With Constraints.* [online] Available from: http://ismir2007.ismir.net/proceedings/ismir2007_p027_peng.pdf [Accessed 6th May 2012].

Raju, T.N.K. 2005. William Sealt Gosset And William A. Silverman: Two “Students” of Science. *Pediatrics.* 116(3): pp 732-735. [online] Available from: http://pediatrics.aappublications.org/content/116/3/732.full.pdf
[Accessed 6th May 2012].

Roads, C. 1996. *The Computer Music Tutorial.* Cambridge, Massachusetts: MIT Press.

RockStar North., RockStar San Diego. 2010. Red Dead Redemption. [Disk]. Microsoft Xbox 360. New York: RoackStar Games.

Rossoff, S.M. 2007. *Adapting Personal Music Based on Game Play.* [online] Available from:
http://dspace.library.uvic.ca:8080/bitstream/handle/1828/2328/SamRossoff.pdf
[Accessed 6th May 2012].

Sivakotiah, S. and Rekha, J. 2011. Speed control of brushless DC motor on resonant pole inverter using fuzzy logic controller. *International Journal Of Engineering Science And Technology.* 3(10), pp 7360 – 7367.

Trochim, W.M. *The Research Methods Knowledge Base*, 2nd Edition. [online] Available from: http://www.socialresearchmethods.net/kb/stat_t.php
[Accessed 6th May 2012].

Veneri, O., Gros, S. and Natkin, S. 2008. *Procedural Audio for Game using GAF.* [online] Available from: http://cedric.cnam.fr/PUBLIS/RC1568.pdf
[Accessed 6th May 2012].

Winter, R. 2005. *Interactive Music: Compositional Techniques for Communicating Different Emotional Qualities.* [online] Available from: http://www.speech.kth.se/prod/publications/files/1701.pdf
[Accessed 6th May 2012].

BIBLIOGRAPHY

Bibliography

Booth, J. 2004. *A DirectMusic case study for Asheron's Call 2: The Fallen Kings.* In T. M. Fay, S. Selfon & T. J. Fay. eds., *DirectX 9 audio exposed: Interactive audio development.* pp. 473–498. Plano, TX: Wordware Publishing.

Collins, K. 2008. *Game Sound: An introduction to the History, Theory and Practice of Video Game Music and Sound Design.* London: The MIT press.

Langston P. 1986. *(201) 644-2332 or Eedie & Eddie on the wire: An experiment in music generation.* New Jersey: Bellcore.

Mcalpine, K., Bett, M. and Scanlan. J. 2009. Approaches to creating real-time adaptive music in Interactive entertainment: a musical perspective. Paper presented at: *AES 35th International Conference, London, UK, 11–13 February.*

Pannerden, T.N., et al. 2011. International Computer Music Conference. *The NLN-Player: A System for Nonlinear Music in Games.* July 31. England: University of Huddersfield.

Wharton, A. and Collins, K. 2011. Subjective Measures of the Influence of Music Customization on the Video Game Play Experience: A Pilot Study. *The international journal of computer game research*. 11(2).