Södertörns högskola | Institutionen för naturvetenskap, miljö och teknik

Kandidatuppsats 15 hp | Medieteknik | Höstterminen 2013

A Study about Adaptive Music through Analysis and Design

Av: Santiago Ferrero

Handledare: Petri Lankoski

södertörns högskola
SÖDERTÖRN UNIVERSITY

Abstract

The purpose of this essay has been to research about adaptive music by analyzing and designing a music system which could handle adaptive music for our game *A Story About My Uncle*. This essay also goes through the thought process and techniques used when composing adaptive music, for example *vertical re-orchestration* and *horizontal re-sequencing.* For designing the system, I used an object oriented analysis and design approach, utilizing two of its models, the use case model and the class model. With the help of these models and a few system requirements based on the music composition, I was able to design a system which handles the composed music as planned. Although the system works in-game, there are still some issues that should be resolved to perfect the adaptive music experience.

Keywords: Adaptive music, Object oriented analysis & design, Music composition

Sammanfattning

Syftet med uppsatsen har varit att forska om adaptiv musik genom att analysera och designa ett system som kan hantera adaptiv musik till vårt spel *A Story About My Uncle*. Denna uppsats går också igenom tankeprocessen och tekniker som används när man komponerar adaptive music, som t.ex. *vertikal re-orkestrering* och *horisontel re-sekvensering*. För att designa systemet använde jag mig av två modeller inom objekt orienterad analys och design, fallmodellen och klassmodellen. Med hjälp av dessa två modeller och ett fåtal systemkrav baserad på den komponerade musiken kunde jag designa ett system som hanterar musiken som det hade varit tänkt. Även om systemet fungerar i spelet, så finns det fortfarande några problem som borde lösas för att fullända det adaptiva musiksystemet.

Nyckelord: Adaptiv musik, Objekt-orienterad analys & design, Musik komposition

Table of Contents

1. Introduction

In this essay I will be focusing on adaptive music in games and how it is composed and implemented. I will be researching this subject by designing a system which can handles the music, specifically for our game *A Story About My Uncle* (Sick Sheep, 2012). This essay is also part of a research project reported in two bachelor theses, this being the first thesis describing how the system was designed. The second essay, written by Oliver Eriksson and Philip Lindau (Eriksson & Lindau, 2013), describes how the system was implemented and evaluated. *A Story About My Uncle* is a very linear game with a story running through the whole game. However, even if the game is linear the player still has full control of when the story should unfold. Therefore a linear, or static, soundtrack would not suffice in this case due to some event which needs to be synced with the music. An adaptive soundtrack would help us tremendously and also create a rich musical experience for the player. To try our adaptive system we will use a level of our game and as the player gets comfortable with the game and starts to keep a certain level of momentum, the music will adapt accordingly to enhance the feeling of flow. But before implementing an adaptive soundtrack in our game we need to have a system which can handle the music and we also need to know what requirements this system should have in order to play the music correctly.

In this chapter I will go through the background of the chosen research questions, explain what adaptive music is and how it works in-game and I will also present the chosen research questions.

1.1. Background

Music in early games was often very simple and repetitive due to the fact that memory was not a luxury and also that the music was programmed instead of recorded or composed outside of the game (Collins et al., 2008). The music was used to express a feeling of the current level and not much more, but later games, such as *Super Mario Land* (Nintendo, 1989), started to use the music as a way of reflecting what was happening in-game. In Super Mario Land the music's tempo would increase when the in-game timer started to run out. Suddenly the music was not just a soundtrack, it helped the player knowing when something was happening, in this case that the timer is running out and the player should hurry up.

Once the concept of adaptive music reached the gaming industry, games have been using this concept in various different ways. Some used it as an indicator or a reflection of what was happening in-game as mentioned before, for example in the Frontier of *Assassin's Creed 3* (Ubisoft Montreal, 2012). When the player runs through forests the only sounds playing are diegetic ambient sounds, sounds that are

coming from the game world. When the player engages in a combat, the combat music is triggered and is played until the played kills all of the enemies or flees. Another way of using adaptive music can be found in the game *Portal 2* (Valve Corporation, 2011). In Portal 2 most of the music is diegetic, they have a source within the game world. Because of this, the different music tracks are not only part of the soundtrack but also sound effects. This also enables the player to interact almost directly with the music in the sense that if a player stand in one position he or she might hear only a certain amount of tracks, but when the player moves to another position more tracks are either added or subtracted depending on the position of the music source. Some tracks are only triggered when a player does a specific move, for example when the player is flying through the air, or interacts with a beam changing the sound of that track. By having the music in this way gives opens a whole new perspective of what adaptive music is and how it can be used.

1.1.1. Adaptive Music

Game music is very similar to film music, they both reflect on what is happening and portrays a certain feeling. The difference between the two is that movies are linear. They will always be played the same way, end in the same way and therefore the music will also be played the same way. The composer knows not only exactly what is going to happen but also when it is going to happen, which is not a given thing when it comes to games. Most games are non-linear in the sense that the player has the control of when the game should move forward. Using a linear soundtrack on a non-linear game would not really work, some vital sync points would be lost and the emotions of certain scenarios would not be synced. This is where adaptive music is very helpful. Adaptive music is music which adapts to what is currently happening in the game or is setting the mood for what is about to happen next. The music should also be aware of the player and of the game-state and should adapt accordingly, for example if the player runs in to a group of enemies the music might change from ambient and calm to intensive and aggressive.

Composing game music has its similarities with film scoring but they are also very different. There are some parts of games that can be scored the same way as a movie, for example cinematic sequences where the player is just a spectator and the scene is already scripted, but there are also parts where the music should be able to loop endlessly and seamlessly and at the same time not be to irritating listening to, for example during certain game play moments. When the players triggers a certain vital event, the transition between the current track and the track corresponding the triggered event should be seamless without any sudden jumps or noticeable clicks and the transition should happen on a beat, especially if the tracks have distinct beats, so

that the music does not get out of sync. A transition cue, also known as a stinger, could be added to make the transition smoother. When composing adaptive music the composer must have in mind that all the tracks should be able to blend with any other music track at any time, by using the same instrumentation and key would therefore help tremendously (Marks, 2001).

For this essay the adaptive music will reflect the players flow level. For this we will need four tracks which represent the four flow levels, level one being the lowest level and is the first flow track the player will hear and level four being the highest level. We will also need a few stingers to indicate the change of the flow level and a few background tracks to give a richer feeling.

1.2. Related Research

There has been a number of studies related to adaptive music but very few on the actual adaptive music system, how it handles the music and how to design a functioning system. Most studies related to this subject are about how and what to think about when composing adaptive music, for example Karen Collins book *From Pac-Man to pop music: interactive audio in games and new media* (Collins et al., 2008) in which Collins and others explains the concept of adaptive music, how to compose adaptive music and how the music is portrayed in-game.

Another Collins book, *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design* (Collins, 2008), describes not only how to compose game music but also how game audio is implemented. Unfortunately not in a technical way, she explains more of how the audio team works in a game company. But she also explains how the communication works between the audio team and the designer or the programmers, for example if the composer writes a music cue sheet explaining the mood of a track, the name of the track and also some specifics for example if is should loop or not. This sheet helps the composer to organize and design the soundtrack, but also helps the programmers to know how to implement the track in the game (Collins, 2008).

One article which talks a little about the music system is from the Game Developers Conference in 2002 where the composer for the game *Halo: Combat Evolved* (Bungie, 2001) explained shortly how he went about composing the music but also how the music system handles the music.

1.3. Research Questions

There are two parts in implementing an adaptive soundtrack in a game, the first part is the music and how it is composed and the second part is the system in the game which handles the music. But what features should the system have in order to play adaptive

music based on the game state? And how should this system be designed so that it fulfills the specific requirements? I will be using an object-oriented approach when analyzing and designing the system.

1.4. Essay structure

In the next chapter, the methods chapter, I will explain what object oriented analysis and design is and what it is used for. In this chapter I will also explain two different modeling type which are often used when designing a system. In chapter 3 (Theory and Application) I will explain what to think about when composing adaptive music and go through two composing methods which are commonly used when implemented adaptive music in games. I will also explain how the adaptive music will be applied to *A Story About My Uncle* and explain what the concept flow means in this case. In the last part of this chapter I will go through the steps I went trough composing the tracks for the adaptive music test.

In chapter 4 I will present my result by showing the two diagram (Use Case diagram and Class diagram) and also presenting the system requirements. I will then evaluate my results by analyzing my design process and also how the system worked after it had been implemented in the game in chapter 5. And finally, in chapter 6, I will look back at the research questions and see if the essay has answered them.

2. Methods

Is this chapter I will introduce and explain the chosen method used to design the adaptive music system.

2.1. Object Oriented Analysis & Design

Object oriented analysis and design (OOA&D) is an approached used to define how a specific system works and describes the interactions between the user and the system (Gorman, 2005). Using such an approach requires two steps, the object oriented analysis (OOA) step, which leads to the object oriented oriented design (OOD) step. The first step, OOA, is where the systems requirements are modeled out and understood. During this step is also where the objects are defined and the interactions between them are described. In the second step, OOD, the implementations specifics and the necessary mechanisms are created to meet the requirements explained during OOA (C. Schmidt, N/A).

In other words, OOA focuses on what the system does and OOD focuses and how the system does it.

During the OOA&D process, there are a couple of different models that can be used to define the system requirements where each model describes the system in different detailed levels and also in different ways. The two models I will be using are the use case model and class model. With these two models I will be able to describe how the adaptive music system works in a very basic level (use case model). I will also be describing the system requirements needed to be able to handle the music. With these requirements I will be able to create a simple diagram showing what classes, methods and relationships the system will need when implementing the system in the game (class model).

OOA&D is a very iterative method which includes a lot of trial and errors, which is often why such a method helps a system of reaching a high level of functionality.

2.1.1. Use Case Modeling

A use case diagram is often the first step in designing a system. During this step the high level interactions will be identified before moving on to the class diagram. The use case diagram should describe everything the system need to do to be able to work correctly (McLaughlin, 2007).

According to Gorman (2006), a use case is a reason for the actor to use the system. Before creating a use case diagram a few things must be addressed, for example the actors which interacts with the system. We must also identify the goal the actor achieves using the system. After identifying the actor or actors and the specific goal,

key use case scenarios must now also be identified. Use case scenarios are scenarios which have different outcomes but are related to the same goal and have the same starting point. The different outcomes are caused by different circumstances within the system, for example one action might not be available because another action has not been completed yet. (Gorman, 2006)

It is during the use case modeling stage that the system requirements are identified, clarified and organized (Rouse, 2007). With this being said, this step will not go through any technical or implementation details, this step will focus on understanding the interactions between the actors and the system and the different scenarios caused by the interactions.

2.1.2. System Requirements

According to McLaughlin (2007), a requirement is a specific thing a system needs to be able to do to work correctly. If one or more of these requirements are not implemented or achieved correctly, the system will not work as planned. With the help of these requirements, when the system is ready to be created and implemented, the developers know exactly what the system should do and can even test each of the requirements to see if the system meets the criteria for these requirements. (McLaughlin, 2007)

As mentioned in the previous sub-chapter (2.1.1. Use Case Modeling), a use case diagram should represent everything the system needs to do, this means that the requirements should handle all of the steps described in the use case diagram. If they do not, the requirements should be revisited and changed or new requirements should be added so that they cover all of the steps described.

2.1.3. Class Modeling

A class diagram represents the static view of a system. This diagram is used for visualizing, describing and documenting the different classes within the system. It is also used as a building block for the programmers when writing the code for the system. The diagram depicts these classes by showing their variables and functions and also their relationships which other classes. (Martin, 1997)

The class diagram resembles a flow chart in which the classes are represented as rectangles. These rectangles are divided into three part where the top part is the name of the class, the middle part is where the class variables are declared and the lower part is where the class functions are declared. To represent the relationships between the different classes, lines are drawn between them. The different types of line, for example an arrow head or a black diamond head, represents different types of relationships (Ezra, 2009).

3. Theory and Application

In this chapter I will explain how to compose an adaptive music and also explain two composing methods. I will then explain what flow is and how it is related to *A Story About My Uncle* and to the adaptive music. Finally, I will through the steps I took composing the tracks for the adaptive music.

3.1. Composing Adaptive Music

As mentioned before composing music for games is very similar to composing film music. With this being said the composer could use the same techniques as film composers for example to express a certain feeling or to build suspense for an upcoming event. But this is where game music is very different. This upcoming event could be triggered whenever, depending on what the player choses to do. For example if the moment before entering the cave of the boss the music should express a suspenseful feeling suggesting to the player that something is about to happen and once the player enters the cave and revealing the boss the music should transition to a more uptempo and fighting-style music. But if the player decides to run around looking for power-ups and such the music should be able to adapt, the suspenseful music should transition to a more ambient style music if the player moves away from the cave and transition back to the suspenseful music when the player moves closer to the cave. If the player then enters the cave of the boss triggering the adaptive music event the music should then transition as planned from suspenseful to adrenaline filled boss music.

There are many different ways of creating adaptive music and there is no right or wrong way due to the fact that adaptive music is a fairly new concept. Thanks to the advance of new technology new creative ways of implementing adaptive music are being created. The two most common techniques and also the techniques I have used are Horizontal re-sequencing and Vertical re-orchestration. These techniques are said to be first utilized in the game *Monkey Island 2: LeChuck's Revenge* (LucasArts, 1991) with the help from LucasArts interactive music system *iMUSE* (Land & McConnell, 1991). The idea for creating this system came from LucasArts composer Michael Land after composing the soundtrack for *Secret of Monkey Island* (LucasArts, 1990) which had a fairly limited music system which could not handle transitions from one track to another (Moormann, 2013).

Jesper Kaae (Collins et al., 2008) explains the difference between this two techniques very well,

> "To see music as horizontal and vertical is related to our understanding of the manuscript paper as representation of the music. Vertical changes in music are therefore changes that happen at the vertical level of the manuscript paper, that

is to say, changes in the instrumentation or the amount of simultaneously sounding notes. Horizontal changes are changes that happen at the horizontal level of the manuscript paper, that is to say, changes that happen over time like, for example, a melody line. All music can be seen as both horizontal and vertical, but some music seems to be either vertically or horizontally oriented." (Collins et al., 2008)

3.1.1. Horizontal re-sequencing

Horizontal re-sequencing is a technique used to re-arrange pre-composed musical segments adaptively. These segments of music can all be played randomly after each other without any weird tempo changes or noticeable jumps in the music. During game play these segments are put together according to the game-state and transitions between them in the composers desired fashion, for example with a cross-fade where the current track fades out and the next track fades in simultaneously. The transition can also occur after the current track has finished played or at a desired transition mark for example the first beat of a bar.

Some of the benefits of using the horizontal re-sequencing technique are the ability of limiting repetitive music, its reasonably fast responsiveness when it comes to transitioning between tracks according to a game-state change and its limited data storage requirements (van Nispen tot Pannerden et. al., 2011)

A good example of horizontal re-sequencing in practice can be found in the game *Halo: Combat Evolved*. The composer , Marty O'Donnell, composed many different tracks witch corresponding beginnings and endings which he then edited and re-arranged so that the music would be able to be played back in the sound engine. Their sound engine would then start a track, start looping the desired middle part until the end part is triggered with alternative looping segments to prevent the music from being to repetitive and alternative ending parts if the ending is triggered during the alternative music is playing. (O'Donnell, 2002)

The disadvantage of using this technique is that it is very time consuming due to the fact that the composer must compose a good amount of music segments for each new musical piece. These segments must also be able to transition smoothly between each other in any possible order.

3.1.2. Vertical re-orchestration

Vertical re-orchestration is a technique which involves the use of dynamic mixing of the instrumentation of pre-composed audio loops which are faded in and out according to the game-state or pre-defined variables (Winter, 2005). With the use of

this technique the variation of tempo, sound, dynamic, timber etc. are much more easily implemented and modified (Moormann, 2013).

Instead of editing and dividing the musical tracks into segments like in horizontal re-sequencing, in vertical re-orchestration the tracks are divided into layers for example instrument wise (bass, drums, brass, woodwinds etc) or section wise (melody, chords etc). There is not any right or wrong way of dividing the tracks, it is up to the composers and the game designer to decide what kind of music suits the game and how they want to use the dynamic music.

Due to the fact that vertical re-orchestration utilizes multiple tracks for each piece of music, it tends to consume much more disk space than horizontal re-sequencing (van Nispen tot Pannerden et. al., 2011)

3.2. A Story About My Uncle

The adaptive music system being design in this essay in being design to work in the game *A Story About My Uncle*. The game is a first-person non-shooter adventure game developed in the *Unreal Development Kit* (Epic Game Inc., 2013) engine. The game is about a boy adventuring through a mysterious world with an adventure suit and a grappling gun, which gives the boy the ability to jump great distances and grapple to pull himself towards the grappled objet, in search of his uncle.

The game has a very distinct feeling of freedom in the sense of exploration and also the mechanic, the grappling gun, which gives the player the desire of having a steady momentum. This leads to the key aspect of the adaptive music system, the players flow level, which will be explained in the next chapter.

3.3. Flow

According to Chen (2006), flow is reach when the player is totally immersed in a video game. The player experiences the feeling of satisfaction when completing a certain level of a certain part of a game. To experience the feeling of flow, the game needs to balance the difficulty level of the game with the players skill level so that the player does not get bored due to the difficulty being to low or being overwhelmed over the difficulty being to high and the player is not able to advance. If the balance between these two are in sync the player will find himself in the so called Flow Zone, the zone between boredom and anxiety. (Chen, 2006)

Flow is therefore nothing that can be implemented, it is something that is experienced. The developers try to design the game in a way so that the player can achieve and maintain the flow experience. In our game *A Story About My Uncle*, the flow experience can naturally be achieved thanks to the grapple gun. What we, the developers, are trying to do is to design the game so that this feeling is easier to reach

and easier to maintain. The level design of the worlds are of course a big part of sustaining flow, but to add even more depth and feedback would help the player tremendously.

What we came up with was adding music which would reflect the flow experience and also the flow level. The flow level reflects the amount of accumulated velocity during game play. Every movement the player does adds the co-responding amount of velocity of that action to the velocity counter and if the velocity counter amount reaches the flow level 1 threshold, flow level 1 music is then triggered. To prevent the velocity counter to grow continuously, x amount of velocity is subtracted each frame. So if the player looses flow in game play or stands still for a while, the velocity counter will diminish over time resulting the player reaching flow level 0 which causes the music to fade out.

Flow, as described above, is a very abstract concept, it is something that is very personal from player to player. By using flow for the adaptive music is almost an attempt of making flow a concrete concept. With this being said, the adaptive music isn't only implemented to reflected the players flow experience but also to help players achieve this feeling.

3.4. Composition & Orchestration

Before even starting to compose the music for the adaptive system, I wanted to find a suitable sound palette. This sound palette is a list of instruments and sounds which fits the theme of the game. By doing this, it will cut down the effort of always having to go through the sound library looking for an instrument or sound, it will also set a building foundation where I can start from, using the interments and sounds from the palette. Adding instruments or sounds is not prevented but the main set of sounds and instruments will be set and therefore all the music created from this palette will naturally fit together. (Marks, 2001)

After completing a desirable sound palette, I tried to find out how to express the feeling of flow with music and still staying inside the theme of the game and level. I wanted the music to give the player a steady rhythm which would hopefully get them to want to stay in the flow zone. I drew a lot of inspiration from the soundtrack of the game *The Last of Us* (Naughty Dog, 2013), especially when the player encounters enemies. The music goes from very ambient to non-existent when adventuring around the environment to intense drums when encountering enemies. These drums give a sense of pace which fits very well in an intensive game play situation. Once the player defeats the enemies or makes it to the next part of the game the drums either fade out or an ending phrase is inserted notifying the player he or she is out of troubles way. This pace that the drums expressed felt like something I could use for the purpose of

flow, not only because of the steady rhythm and pace, but the drums would fit with the theme of the game and environment.

Now when I had my sound palette completed and I had decided on using drums and percussions as the main instruments I could finally start composing the different tracks in the digital audio workstation *Logic Pro X* (Apple Inc., 2013) . We had decided on four levels of flow, level 1 being the lowest level and level 4 being the highest, which meant that four levels of music intensity would have to be created. I also knew that these levels would transition from one to another quite freely but not randomly, this meant that the tracks would all have to be recorded with the same tempo and meter so that the transitions would be seamless. The tracks also needed to be loop-able because because of the uncertainty of when a certain level of flow was going to be reach or lost. I started by experimenting with different rhythms until I found a suitable one which would then become the base of the adaptive soundtrack. I then fleshed this pattern out by implementing or deleting certain drum hits so that it would not be too repetitive and to give it a more natural feel. I also added some percussion to give it some high range frequency. Now I had a loop-able reference track which gave the right feeling I was looking for, now it was time to create the different levels of intensity using this track.

Before creating the additional tracks, I had to decide how they would be played in the game, for example playing one track and then fading in an extra layer when a player has reached a higher level of flow (vertical re-orchestration, see 3.1.2), or transitioning from one track to another (horizontal re-sequencing, see 3.1.1). I decided to use both techniques for the different levels of flow because of the different type of transitions the techniques offers.

After mapping out how the different tracks are going to be played in-game, I went on creating the tracks needed. The final tracks consists of four main tracks corresponding to the four flow levels and two background tracks which could be played over any flow level. Once the first level is reached, track number one and one of the background tracks fades in. Track number one will loop until the player has reached flow level two or looses flow level one. If the player looses flow level one, the track will simply fade out, but if he or she reaches flow level two, track one will keep on looping while track number two fades in. I decided to do so because I did not want a distinct transition between the two first tracks. I wanted it to feel as a smooth progression instead of a flip of a switch which the player would notice much more easily. For the transitions between level two and three, and three and four I decide to use horizontal re-sequencing by transitioning from one track to another on a beat without any cross-fades or such. The reason for not using vertical re-orchestration for these two transitions was that the transition between these levels work very well

musically. They were sufficiently similar musically that a clean transition would work without sounding weird or having any musical clashes.

Now after all the tracks are completed, all we need is a system which can play these tracks as planned.

4. Results

In this chapter I will present the system requirements, explain the use case diagram created from the system requirements and also show a basic class diagram which resembles how it will be implemented.

4.1. System Requirements

1. The adaptive music system must be able to handle the player-characters velocity count.
2. The system should be able to play the correct track corresponding the current flow level.
3. The transition must happen on a beat so that the transition happens smoothly.
4. The system must be able to do transitions from the lowest flow level to the highest and back.

These are the four requirements for the adaptive music system in *A Story About My Uncle*. These requirements cover every step of the system. If one of these requirements is not met, the system will not work properly.

4.2. Use Case Diagram

As mentioned in chapter 2.1.1. Use Case Modeling, a use case is used to described how a system works and what components it needs to be able to work correctly. It also shows the different scenarios that can or may happen when the system is running. This use case diagram is based on the system requirements shown in the previous subchapter (4.1. System Requirements).

The only actor which can affect the adaptive music system in *A Story About My Uncle* is the player (see *Figure 1*). The only variable the music system needs is the player-characters accumulated velocity, which is affected directly by the player. If the player moves, the current velocity is added to the velocity counter. To prevent the velocity counter from adding up too quickly and also to enable the counter to diminish when the player looses momentum or stands still, x amount of the counter will be taken away every frame.

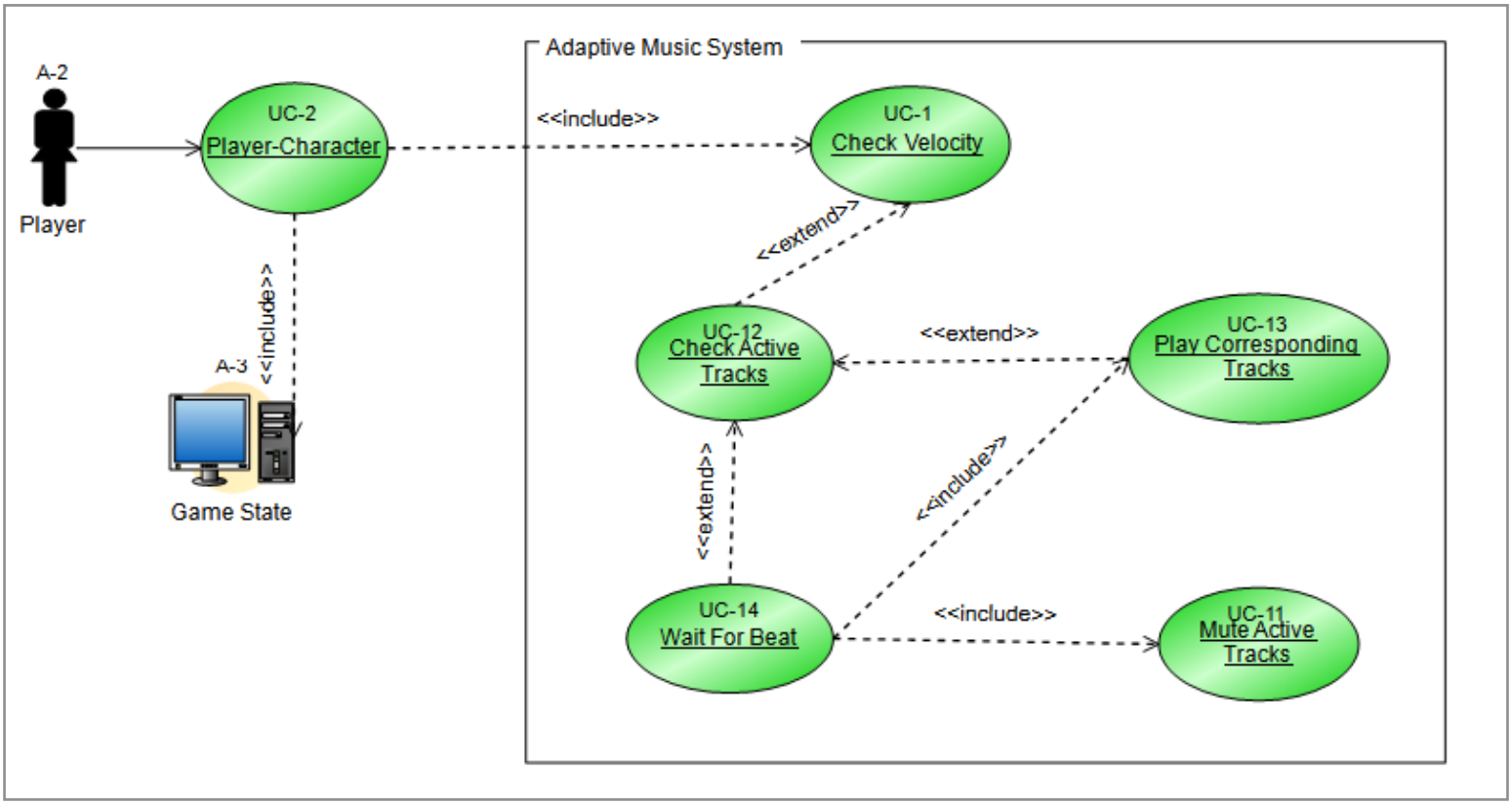

Figure 1. Use Case Diagram

Looking at *Figure 1* again we can se that the actor (the player) is affecting the P*layer-Character* directly and therefore affecting the player-characters velocity. The *Check Velocity* is included in the player-character, this means that when the player-character is being affected it will always affect the check velocity as well. The check velocity node is where the system checks the player-characters current velocity counter count and checks if it has reached a level of flow. Each flow level has a threshold which is an amount of velocity the player-character needs to be able to trigger that level of flow. The node which extend from check velocity (*Check Active Tracks*) is triggered under certain circumstances. The difference between include and extend is that a node which is included in another node will always be triggered, however a node which extends from another node will only be triggered under the right circumstances. With this being said, the check active tracks node will only be triggered when the player-characters velocity has reached a flow level. When the check active tracks node is triggered, it will check if a track is being played and also get the necessary information, for example track ID and number of bars. If there is no track playing or if track number one is playing, *Play Corresponding Track* node will then be triggered. This node is pretty straight forward, it simply plays the track which corresponds to the flow level, for example if flow level 1 is reached, track number 1 will be played. However, if track number 2 or 3 is being played, the node *Wait For Beat* will be triggered instead. Wait for beat will use the track information retrieved from the check active tracks node and it will sync the transition between the current track and the new track. Once the beat is reached, the Mute Active Tracks node will be triggered which

will mute the current tracks and the new track will be played by the play corresponding track node.

When comparing the use case diagram with the system requirements, all of the use cases and use case scenarios show in the diagram should be covered by one of the requirements. If one or more are not covered, the requirements should then be revisited.

4.3. Class Diagram

As mentioned before, this class diagram will be a basic view of the system, it will not depict all of the classes nor all of the functions. However, it will contain all of the necessary classes, functions and variables needed to explain the system on a high level.

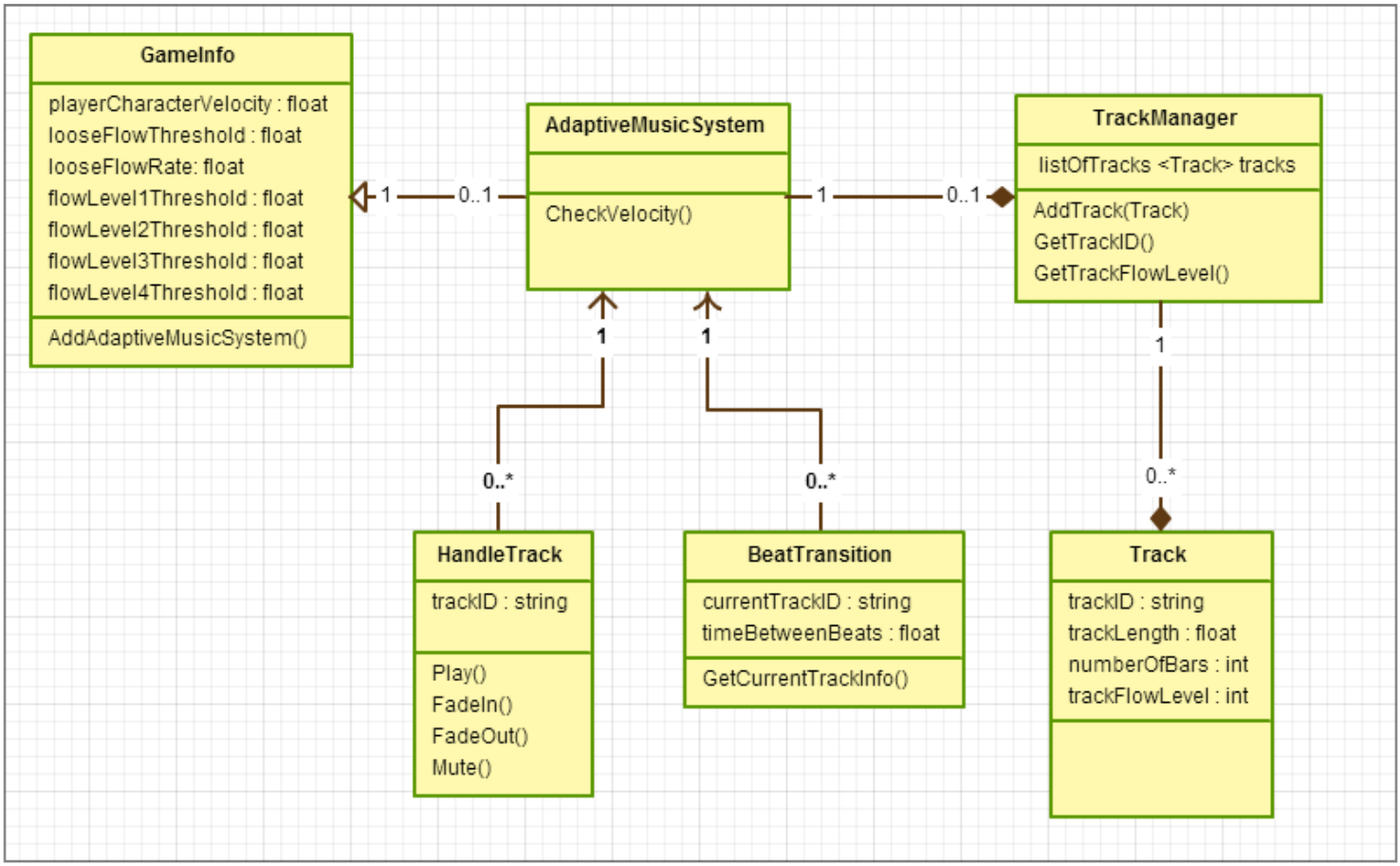

Figure 2. Class Diagram

The diagram in Figure 2 has been derived from the use case diagram shown in chapter 4.2. Use Case Diagram. As we can see the AdaptiveMusicSystem class consists of a TrackManager class, a BeatTransition class and a HandleTrack class. The TrackManager holds an array of all the tracks. It is also in this class that the tracks are added with the AddTrack function. The Track class is where the tracks information are stored, for example trackID (the tracks name) and the trackFlowLevel (the level of flow the track corresponds to). The HandleTrack class is where the tracks are played, muted, faded in or faded out. To handle the transition smoothly, the BeatTransition uses the tracks length and the tracks number of bars to calculate the time between each beat for the track being played. All of these classes can be handled from the AdaptiveMusicSystem class, which in turn derives from the GameInfo class. The GameInfo class is where the flow levels thresholds are declared as well as the rate of which the flow level decreases and the threshold which the player looses flow. This class also handles the player-characters velocity count.

5. Evaluation

During the whole design process a few decisions where made, for example how the music was going to be played by the system and how the tracks where going to be composed. These decisions were not only made from experience and the fact of knowing what I wanted to achieve but also from using the knowledge gained from some of the literature and trial and error. As mentioned in subchapter 2.1. Object Oriented Analysis & Design, OOA&D is a very iterative method. With this being said, during the design process I tried to always test what I was doing to see if something was missing or not working well, for example when composing the music I tried a few different types of transition before finding the ones which satisfied me. When mapping out how the music was going to be played in game, a lot of decisions came from the literatures talking about adaptive music. This was the case because I lacked knowledge and experience in composing adaptive music and decided to compose the music as similar to how others have done it as possible to understand the concept and tricks in composing adaptive music.

To be able to test the system to see if it works correctly, it has been implemented into the game *A Story About My Uncle* by Oliver Eriksson and Philip Lindau. The music used to test the system was the drum tracks that I created, explained in chapter 3.4. Composition & Orchestration. The system was tested by letting people play the first level of the game twice, once with a static soundtrack and once with an adaptive soundtrack. After each playthrough, the respondents had to score a few questions from 0 to 10 about the game, for example how the overall experience was and how the respondents perceived the music in the game. After gathering all the necessary results, Eriksson and Lindau concluded that there was no significant difference in the scores when it came to the music.

When analyzing the system in game it seemed to work properly and also met the requirements mentioned in chapter 4.1. System Requirements. The right track played when the player reached a flow level, when transitioning from one track to another the transition would wait for the current track would come to a beat etc. However, even if the system met the requirements and performed as planned, the results Eriksson and Lindau found showed that the adaptive music did not help the game play or the music overall to be perceived as better. In fact, in some cases, the respondents said that the adaptive music was more irritating then the static music because the music would start and stop to often, detaching the player from the flow experience.

With this being said, the system which was implemented did in fact meet the requirements and also performed as it was designed to do. However, after analyzing Erikssons and Lindaus results, new requirements should be added to make the

adaptive music less irritating and much more adaptive. For example, instead of fading out directly when the velocity reaches the threshold for loosing flow, it should have a delay to see if the player will reach a flow level within that time, if not then start fading out. The system should maybe also be aware of where the player is in the level. For example, if a player is stuck in an area and dies over and over but the player still maintains velocity, the system might then fade the music out and when the player finally makes it past the difficult area the music starts to play again when the player lands.

However, the focus should not only be on the system but on the music itself. The tracks composed for this essay were rapidly composed and I had also no prior knowledge of composing adaptive music. This caused the music to not give justice to adaptive music as a concept. Had the music been a bit more worked on but still using the same adaptive music system, the results Eriksson and Lindau got might have been very different.

6. Conclusions

The purpose of this essay was to conduct a research study about adaptive music by analyzing and designing a music system which could handle adaptive music. First, I explained how adaptive music is composed, which helped me pinpointing the requirements for the system. I then used two object oriented models to help me design the system. The first model, use case diagram, was used to define the different actor, objects and scenarios within the system. With the help from this diagram a class diagram was then created. The class diagram showed a basic overview of the vital classes needed when implementing the system in the game.

The research questions presented in the beginning of the essay were: “what features should the system have in order to play adaptive music based on the game state?” and “how should this system be designed in order to fulfill the specific requirements?”. If we look at these questions and compare them to the results presented in chapter 4 we can see that this essay has answered the questions asked. I have presented the system requirements which are the features the system should have, and I have also presented how the system should be designed by presenting two diagrams which explains how the system should work and also what classes it should have. However, this does not mean that a perfect system has been designed. This only means that the system has been designed correctly based on the requirements. With this being said, and using the results from Erikssons and Lindaus essay, a new and improved adaptive music system could be designed with ease by using an object oriented approach, an expanded requirement list and also by using the new knowledge gained by designing a system.

7. Discussion

Designing an adaptive music system and also composing the music for the system during the short amount of time I had really affected the results. I knew that none of the aspects included in this study would be able to be perfected and I started this study with that mindset. The purpose of this study was not to create the perfect system, it was to research about adaptive music and how a system handles adaptive music in-game. This meant that the music and the system did not need to be anywhere near perfect, they just needed to work properly enough so that vital flaws or any missing aspect could be spotted and also to get an insight on how and what to think about when composing and designing a system. Unfortunately, the composed music tracks did not reached a desired quality which affected the end results. The system designed was fairly basic but it worked properly and handle the music just fine. However, after the implementation and test runs done by Eriksson and Lindau, it felt as if a few elements were missing. One aspect of designing the system I felt I should have spent more time doing was mapping out step by step how the music would perform in-game, for example how the tracks would transition and when they can or should transition. In this study, this step was done at the same time as I was composing and creating the diagrams, which created a bit of unnecessary confusion going back and forth between composing, adding an element to the system and then going back composing trying to make the tracks work with the added element.

This was my first time using an OOA&D approach which also caused a bit a time being “wasted away”. Notice I’m using citation marks because it was not really time wasted in the sense that I learned a lot about OOA&D and it help me create a functional system and will be helping me in the future, but in the sense that if I was comfortable with OOA&D and had used it before I could have had more time focusing on composing and actually designing the system. With this being said, if I had to do this study all over again I would chose to use OOA&D and the two chosen models without a doubt.

References

Apple Inc., 2013, [digital audio workstation]. *Logic Pro X*, Apple Inc.

Bungie, 2001, [video game]. *Halo: Combat Evolved*, Microsoft Game Studios.

Schmidt, D. C., N/A. *Object-Oriented Design and Programming: Overview of Object-Oriented Design Principles and Techniques*. Available at: http://www.cs.wustl.edu/~schmidt/PDF/ood-overview4.pdf.

Chen, J., 2006. *Flow in Games*.

Collins, K., 2008. *Game sound: an introduction to the history, theory, and practice of video game music and sound design*, Cambridge, Mass: MIT Press.

Collins, K. et al., 2008. *From Pac-Man to pop music: interactive audio in games and new media*, Aldershot, Hampshidre, England ; Burlington, VT: Ashgate.

Epic Games Inc., 2013, [game engine]. *Unreal Development Kit*, Epic Games Inc.

Eriksson, O. & Lindau, P., 2013. *Evaluating an Adaptive Music System in an adventure game environment*, Unpublished BA. Södertörns University.

Ezra, A., 2009. UML Class Diagram: Association, Aggregation and Composition, Available at: http://aviadezra.blogspot.se/2009/05/uml-association-aggregation-composition.html.

Gorman, J., 2005. *UML for Managers*. Available at: www.parlezuml.com.

Gorman, J., 2006. *Use Cases - An Introduction*. Available at: www.parlezuml.com.

Land, M. & McConnell, P., 1991, [game engine]. *iMUSE,* Land, M. & McConnell, P.

Lucasfilm Games, 1990, [video game]. *The Secret of Monkey Island*, LucasArts.

LucasArts, 1991, [video game]. *Monkey Island 2: LeChuck's Revenge*, LucasArts.

Marks, A., 2001. *The complete guide to game audio: for composers, musicians, sound designers, and game developers,* Lawrence, Kansas: CMP Books.

Martin, R.C., 1997. UML Tutorial: Part 1 -- Class Diagrams. Available at: http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf

McLaughlin, B. et al., 2007. *Head first object-oriented analysis and design* 1st ed., Beijing ; Sebastopol: O'Reilly.

Moormann, P., 2012. *Music and game: perspectives on a popular alliance*, Springer.

Naughty Dog, 2013, [video game]. T*he Last of Us*, Sony Computer Entertainment.

Nintendo Research & Development 1, 1989, [video game]. *Super Mario Land*, Nintendo.

O'Donnell, M., 2002. *Producing Audio for Halo*. In GDC. Available at: http://halo.bungie.org/misc/gdc.2002.music/.

Rouse, M., 2007. *Definition - Use Case*. Available at: http://searchsoftwarequality.techtarget.com/definition/use-case.

Sick Sheep, 2012, [video game]. *A Story About My Uncle*, Sick Sheep.

Ubisoft Montreal, 2012, [video game]. *Assassin's Creed 2*, Ubisoft.

Valve Corporation, 2011, [video game]. *Portal 2*, Valve Corporation.

Van Nispen tot Pannerden, T. et al., 2011. *The NLN-Player: A System For Nonlinear Music In Games*. In International Computer Music Conference. University of Huddersfield.

Winter, R., 2005. *Interactive Music: Compositional Techniques for Communicating Different Emotional Qualities*. Master thesis. University of York.