Evaluating an adaptive music system in an adventure game environment

Av: Oliver Eriksson, Philip Lindau
Handledare: Petri Lankoski
Abstract

Adaptive music changes depending on parameters which change with events in the game. This essay explores whether an adaptive music system would be an improvement for the game A Story About My Uncle over static music, and if it would be worth the time investment of implementing it.

Data for this is gathered by having people play an introductory level from the game twice, once with static music and once with adaptive music. The players answer a survey after each playthrough of the levels.

The results show no significant difference between the static and adaptive systems in terms of perceiving the music, or the game experience.

Keywords: Adaptive Music, Dynamic Music, Sound Design, Video Games
Adaptiv musik ändras beroende av parametrar som ändras av händelser i spelet. Den här studien undersöker om ett adaptivt musik system är en förbättring i spelet A Story About My Uncle jämfört med statisk musik, om det är värt tidsinvesteringen som implementationen kräver.

Data samlas in genom att ta in testare som spelar en introduktionsnivå i spelet två gånger, en gång med statisk musik och en gång med adaptiv musik. Spelarna svarar på en enkät efter varje genomspelning av nivåerna.

Resultaten visar ingen signifikant skillnad mellan den statiska och adaptiva versionen när man undersöker spelarnas generella spelupplevelse, och upplevelse av musiken i nivåerna.

**Nyckelord: Adaptiv Musik, Dynamisk Musik, Ljud Design, Datorspel**
# Table of Contents

1 Introduction ................................................................................................................................. 5  
  1.1 Background ................................................................................................................................. 5  
    1.1.1 Horizontal resequencing ........................................................................................................ 6  
    1.1.2 Vertical reorchestration .......................................................................................................... 6  
    1.1.3 Adaptive Music Software ..................................................................................................... 6  
    1.1.4 A Story About My Uncle ...................................................................................................... 7  
  1.2 Purpose .......................................................................................................................................... 7  
  1.3 Research Question ....................................................................................................................... 8  
  1.4 Related Research .......................................................................................................................... 8  
  1.5 Adaptive music system implementation ....................................................................................... 9  
    1.5.1 Flow mode ............................................................................................................................ 10  
    1.5.2 Kismet nodes ......................................................................................................................... 11  

2 Method ............................................................................................................................................... 12  
  2.1 Adaptive system .......................................................................................................................... 12  
  2.2 Adaptive sound in practice ......................................................................................................... 12  
    2.2.1 Playtesting Level .................................................................................................................. 12  
    2.2.2 Adaptive sound queues in the level ...................................................................................... 13  
  2.3 Evaluation method ....................................................................................................................... 13  
    2.3.1 Study .................................................................................................................................... 13  
    2.3.2 Playtesting .......................................................................................................................... 14  
    2.3.3 Analysis ................................................................................................................................ 16  

3 Results ............................................................................................................................................ 17  
  3.1 Analysing the results ................................................................................................................... 17  
    3.1.1 T-Test - Music Score .......................................................................................................... 17  
    3.1.2 T-Test - Game Experience Score ....................................................................................... 18  

4 Discussion ....................................................................................................................................... 19  
  4.1 Results ......................................................................................................................................... 19  
  4.2 System Implementation .............................................................................................................. 20  
  4.3 Future work ................................................................................................................................ 20  

5 Conclusions ..................................................................................................................................... 22  

6 References ....................................................................................................................................... 23  

Appendix – Survey ........................................................................................................................... 26
1 Introduction

1.1 Background

We refer to adaptive music as music in games that changes depending on different variables. The concept is about making the music change depending on events in the game and actions the player performs during gameplay, this is used in order to make the music appear more varied and responsive. Music in mediums other than games are usually static, with a track being played from beginning to end. In the case of movies, the sound or music is often tweaked to fit better into the scene at hand. In some cases, this is taken taken to its extreme, where the actions of the characters on the screen gets direct feedback in the music. This is called “Mickey Mousing” (Pichlmair & Kayali 2007). In games on the other hand, each playthrough of a game will be different, and music tracks can not be timed beforehand.

Adaptive sounds and music has been used in games for quite some time, at varying degrees. In games, this method is often used for changing the way a player’s footsteps sound, depending on the surface the player is walking on, or depending on the movement speed. Another example in games is that the music may change tempo or layer more instrumentals on top when a power-up is active.

One of the first games to implement this adaptive style of music was R.B.I. Baseball (Namco LTD 1987), a baseball video game for the Nintendo Entertainment System (Nintendo Co., Ltd. 1983) where the music changes depending on whether a player has a runner on base or not.. Another, more recent example of notable quality is the Halo series (first developed by Bungie, Inc. 2001), where different music tracks would layer each other depending on several factors such as the player retreating, pushing forward or when a specific group of enemies has been cleared. (Sofge 2010).

Another game which has very profound dynamic music is Portal 2 (Valve Corporation 2011). The dynamic music was composed by Mike Morasky specifically to be used in a dynamic fashion. An example of this is layers of music which is added as the player completes parts of puzzles. The music plays as if it is playing through the object connected to the part of the puzzle the player just completed (Wilde 2011). An example of this is a puzzle where you redirect lasers into machines that activate when doing so. In this case, you are supposed to redirect three lasers, and after each machine has been activated, a new layer of music is added. Another example are the pads which launches the player forward. These are called “Faith Plates”. When a player is launched from one of these pads a different track is played as the player flies through the air. (Hamilton 2011).

An example of a game which has a very apparent implementation of adaptive music is Rock Band (Harmonix Music Systems 2007). In this game, the player uses a microphone or a game controller designed to mimic either a guitar or a
drum kit. While playing guitar in this game, colored dots appear along a board with multiple lanes (five lanes for the guitar) where each lane represents a guitar string. As the dot reaches a line on the screen, the player is supposed to hold a button representing the lane on which the dot is on while at the same time pressing another button which represents the actual strumming of the string. If the player does not do this properly, but clicks the strum-button too late, a sound plays indicating that the player wasn’t fast enough. If the player misses clicking altogether, the track of the instrument that the player is controlling stops playing.

The reason for conducting this study, is because we are currently considering an adaptive music system for our game, which we will go over in 1.1.4. We chose to conduct a pilot study prior to this essay in order to achieve a better understanding on how to implement an adaptive music system. The study was very similar regarding the method, but at a much smaller scale. Apart from the smaller test group, the implementation in the pilot study was much simpler. In that study, we used a very simple approach, using only predefined nodes within a visual scripting tool without doing any modifications to the system. The players also played another part of the game, with distinct differences in how adaptive music was used.

There are two types of adaptive music available, horizontal resequencing and vertical reorchestration (Winter 2005). These will be discussed in more detail below.

1.1.1 Horizontal resequencing
Horizontal resequencing in a practical sense means that soundtracks are being played one after another, and being rearranged as the game is being played (van Nispen tot Pannerden et al. 2011). An example of this would be that a player has entered a new area, so a track with a different feel to it should be played, but the game waits until the last track is done before switching over to the new one. This might result in the switch being delayed to the point of where it does not match very well to the change, and thus it might seem off to the player. A solution to this is to have very short tracks and repeat them, but that might result in very repetitive music instead, which can be annoying to the player.

1.1.2 Vertical reorchestration
With vertical reorchestration several tracks are mixed together in order to form a finished soundtrack (Collins 2008a). One way to implement this is to have each track follow the same beat and be the same duration, and then start every track at the beginning of the game. By starting every track and repeating it you can ensure that it is always synced. In order to regulate which tracks the player hears at a time, the tracks’ volume is controlled individually according to events in the game.

1.1.3 Adaptive Music Software
Commercial software exists that provide this kind of functionality to AAA Games, the biggest being FMod (Firelight Technologies 2013). It supports the biggest game development platforms and is used in a multitude of commercial games.
made today. In some of these games FMod is used to achieve an adaptive sound environment in the game, but other aspect of sound management is present within FMod. The downside to these kinds of systems is that they are often bigger than one would need for a smaller size game project.

An early example of a software that was written entirely for the purpose of adaptive music was iMUSE (Land & McConnell 1991), which was created in order to easily create seamless transitions in the current games made by LucasArts at the time, and was first implemented in Monkey Island 2: LeChuck’s Revenge (LucasArts 1991).

A much more recent example is the software G.A.M.E. (Claynote 2013), a system developed for use in indie games which makes use of the horizontal re-orchestration style of system. However, this system is proprietary, and not available for public use.

1.1.4 A Story About My Uncle

As this essay is being written we are working on improving a game we released previously (Zethraeus 2012) for a new release, called A Story About My Uncle (Gone North Games 2013). The game is a first-person adventure game developed in Unreal Development Kit (Epic Games Inc. 2013), where the player takes the role as a young boy travelling through a mysterious world in search of his lost uncle. In this study, we evaluate an implementation of a reactive sound system. The evaluation is conducted using the development version of A Story About My Uncle game.

The main mechanic of the game involves a grapple gun that the players can use to pull themselves towards areas within the game world. This combined with jumping and running around, lets the player freely explore the world with a big sense of space. The game pits the player against obstacles such as gaps, moving platforms and mountainous terrain.

1.2 Purpose

Our purpose of studying an implementation of adaptive sound in the game environment is twofold: First we want to see if adaptive sound is something that will enhance the experience while playing a video game. Secondly, since implementing such a system can be quite the endeavour, resource- and time wise, we want to see if the change to the players experience is enough of an improvement to justify that investment. The study also allows us to try the system for the game, to see if it is something we wish to develop for the game as a finished product.

Additionally, it is also possible that the player might learn faster from adaptive music. This is because we can confirm the actions the player makes correctly by audio cues and thus not breaking the immersion. This could possibly lead to the player picking up mechanics in the game faster because of this extra affirmation. Previous works from Jørgensen (2006) for example, show that this could well be the case in some games.
1.3 Research Question

The research question we chose is as follows:

Does an adaptive music system improve the game experience in the game *A Story About My Uncle*, compared to a linear music system, and if it does, was it improved enough to be worth the time investment?

The study is as the question implies, very narrow, focusing mostly on how well the system works in the game *A Story About My Uncle*. We would not be capable of answering the bigger question whether it is worth developing such a system for any game, since that would be far too much of an undertaking. We hope that this study will give definite proof whether adaptive music is something we should pursue in the game.

1.4 Related Research

Research has been done on the subject of adaptive and interactive audio but not to the extent of similar topics within the video game spectrum.

Karen Collins *Game sound* (2008b) is probably one of the more extensive works regarding the history of adaptive sound in video games and sound production in games in general. Covering topics such as methods for scoring adaptive music, and the different technologies available. Another work of Collins that cover sound interaction is her book *Playing with Sound* (Collins 2013), which does a good job of mapping out the history and phenomena of the players' relationships to sound in games and how interacting with sound has become a staple for some games.

An implementation similar to ours was made by Chris Latham, who created two systems, one for horizontal resequencing (2013a) and one for vertical reorchestration (2013b). These implementations were also made in *Unreal Development Kit* and also by using custom nodes in a built-in system called *Kismet*.

Although not focused entirely on adaptive sound, *Informative sound design in video games* (Ng & Nesbitt 2013) does a great job of highlighting some of the advantages and techniques for delivering additional information to the player while playing through the use of sound: such as dynamic changes to sounds in the environment depending on the state of some object, for example, a container being full or empty as a way to direct the player.
1.5 Adaptive music system implementation

Since our expertise is well outside that of musical orchestration, we had to use something externally created. Since we still needed something that was tailored to our environment, style of music, and overall atmosphere we choose to use music specifically composed for us by Santiago Ferrero (2013). We also worked with Ferrero when developing the system, getting the prerequisites for the music he would compose.

As mentioned in the previous chapter, we chose to implement a system using vertical re-orchestration. The system works by playing every track in the beginning of the map, and then applying multipliers to the tracks' volumes. This allows us to create temporary volume changes on top of already existing volume changes. An example of this would be that we could have the volume of one track increase if the player is running instead of walking, whereupon the player enters an area where we want to lower the volume of all tracks.

The track which had its volume raised will remain raised in relation to the other tracks equally. When the player then exits the area, the multiplier which lowered the volume in the area will be removed and the volume will be set back to the way it was before, unless new multipliers were introduced or old ones were changed.

![Figure 1. Example of kismet nodes created for the system](image)
A quick overview of the system shows that the system is fairly simplistic because of the already developed implementation of sound management in *UDK*. This means that we only need to extend the system to be able to handle the adaptive part of playing tracks, and implement a *Kismet* system for setting up tracks and firing them based on game events and certain player states. The *Kismet* system is what will, in the end, be the determining factor for how the adaptive system sounds and feels, and is detailed below.

### 1.5.1 Flow mode

The first thing we wanted the adaptive music to convey in the game was the feeling of flow, something that is very closely related to the core mechanic of the game. As the player moves through the environment, we will look for consistent movement in the player, and then determine if the player’s movement is *flowing*. If the player is constantly moving without losing the pace, tracks will start playing with varying degrees of intensity, depending on how long the player has kept up the pace.

It works by adding the size of the player’s velocity to a flow value each frame. A range of thresholds are specified, and tracks are set for them. As the player moves around the world, the value will increase, and every time a threshold is reached the tracks bound to that threshold will be played — while the other tracks will stop. At all times, this value is also being reduced, which means the player will have to continue moving in order to not sink to a lower threshold.

As mentioned above, this needs to be dynamic in order to have the players enter the flow mode depending on their own skill level, or pacing in the game. This is done by changing how much the value is being reduced constantly, by applying a multiplier to it, ranging from a number between zero and one, and a number over one with a specified maximum. This multiplier is lowered if the flow level is above the first threshold, and raised if the flow is below the first threshold. The rate at which it is lowered is also dynamic depending on the velocity of the player. It works using a multiplier ranging from 0 to 1. This is calculated by storing the maximum velocity the player reaches, and then comparing it to the player’s current velocity.

In order to determine good thresholds for the different tracks, we created a debug window which we could show while playing, where we can print the current values of different variables. This allowed us to see exactly how the algorithm was behaving, and was especially useful for pinpointing what multiplier and thresholds gave the best result.
1.5.2 Kismet nodes

In this section we will briefly explain what the different custom kismet nodes developed for the system does.

The *Add tracks* node is used to add tracks to the active tracks list. It does so by accessing the adaptive music system through the static game class called *GameInfo*. Each track is given an ID which is then used when retrieving that track using other nodes. This node was changed from adding a single track per node, which quickly became very cluttered, to have multiple tracks added with one node. It also made it easier to follow the visual code. This node is also used for playing the tracks used by the flow mode system described in 3.2.

The *Set track multiplier* node retrieves a track using the ID mentioned in 1.5.2.1, and then adds or sets an already existing multiplier to a specified value. This value then sets the volume, where 0 is muted and 1 is full volume. The time it takes for the adjustment to happen is also configured using this node. After some testing the option for playing a *Stinger* (short, non-looping piece of music) at the transition was added, which allows us to add more intense transitions in some cases (Aav 2005). There is also a version of this node which sets a multiplier on all active tracks, in order to do global changes to the volume. An example of where this could be used is at points that are narrated. The node itself is used in the same way as the node which manipulates a singular multiplier, although you do not need to specify a track id.

The *Track beat node* retrieves a track using its ID and activates its output repeatedly, with a delay which is set by using the duration of said track, and specifying the amount of beats in the track. This is done to achieve an even better transition.

After some testing when level building we chose to add the *Force flow Level Node*. The reason for this is to be able to make sure that the feeling of action is elevated in areas where we want it to. An example of this is in the “leap of faith” discussed in 2.2.2.
2 Method

In this chapter we will introduce the methods used for implementing the adaptive sound environment as well as the playtesting sessions.

2.1 Adaptive system

To achieve our goals for this essay we need an implementation that is good enough to get reliable data from play sessions. Fortunately, we have *A Story About My Uncle* which is described in 1.1.4. Our implementation uses vertical re-orchestration, as explained in 1.1.2. We chose this method this after trying horizontal resequencing in the aforementioned pilot study, since we noticed that the timing of changing tracks would become delayed at times, unless we used very short tracks which instead meant that they became very repetitive.

2.2 Adaptive sound in practice

The playtests are designed in such a way that the player plays a level of the game with either our adaptive music system turned on or simply with a static background music track. The static music track is very ambient and mystical, without any distinct melody. The instruments consists of drums playing in short bursts, and some quiet bow instruments.

After playing through the level they are asked to answer questions about their experience of the session. We cover the exact process of the playtests later in the essay.

2.2.1 Playtesting Level

As mentioned above, since we already have a functional environment to playtest in, implementing the adaptive system in a level was as easy as slightly modifying an already existing map from the game. We used an early level from the game, as this level was easier and featured a tutorial of how to play the game. The level itself is quite linear, more so than similar levels later in the game, which is also a desirable quality for a playtesting level.

One of the things that we had to change for the playtest was the length of the level, as we felt that it would take to long for our playtesters to play the full length twice, and still give us enough time to reach the number of playtesters we were striving for. In the end, the tests became a bit longer than they should have, scaring away potential respondents, and due to the fact that they had to play the long section twice, some players might have been annoyed which may impact the result of their tests.
2.2.2 Adaptive sound queues in the level

Finding the grapple gun is an important moment in the game, because the gameplay changes significantly after doing so. To indicate the importance of the event, a one-shot sound is played. Another possibility here would be to introduce a new track, to signal this even further.

There is a section where the player approaches a huge gap which the player must jump across. As the player jumps the music quiets down and a stinger plays. When the player lands, we force the players’ flow level to be increased, and increase the volume of the music back to normal. We do this to create a feeling of suspense as the player jumps, and increase the feeling of success when the player lands.

We had planned to have a music cue play and change the rest of the music if the player falls into the water at the bottom of the level, however, we chose not to include this because of the time investment that it would require. Just playing the music cue would be quick, however, we would have to change the visuals of how the player is moved back to the previous checkpoint.

2.3 Evaluation method

2.3.1 Study

To extract some tangible results to strengthen our disposition we need to query players who are familiar with games and their structure, but unfamiliar with this specific implementation and the environment, in this case *A Story About My Uncle*. We have designed the questions of the questionnaire used in such a way to not lead the player in a direction where they pick up on the fact that the adaptive music system is the prime focus of the questions. We also found that these other questions could be of use to us in an overall look of the results, even though we are ultimately only interested in the adaptive music aspect.

With this information we can then go about analysing the data, searching for patterns and ultimately try to come to a conclusion. We started off with having the questionnaire, which can be found in the appendix, on paper which players’ would fill in after each playthrough however, this was switched out for a digital form after some time. The digital form is described later in this chapter.

The questions asked in the questionnaire are of a one sided nature, they focus on different aspects of the game such as overall experience, controls, graphics, and of course music. We are naturally mostly interested in how players experience the music, and the other ratings are merely red herrings to lead the players astray of noticing the differences between the sessions too clearly. This because we do not want to steer the players toward declaring one system to be better than the other just by knowing there is a clearly stated difference. However, there are also other accompanied benefits of these additional questions: We can look for patterns in how the players perceive controls and overall playing experience, something that is part of our research question. We also look for patterns in
video game playing habits, as well as their age and gender — to see if this has an effect on whether they think that the experience has improved with the system.

Something equally important was to try to seek out players that were less familiar with this specific game, its genre and gameplay. This because we believe it creates a playstyle that is less automated and the player is more prone to react to subtle changes in the game. Whereas players who feel comfortable with the game and the gameplay in general, tends to rush through the session more.

A problem arises when letting the same player test both the static and adaptive versions after one another. We found in the pilot study that the player naturally becomes accustomed to the ways of the game, and will most likely rate the experience higher this second time around. We combat this by letting some of the players start with the adaptive version first, and some will play the static version first, as to counterbalance.

There is a particular reason why we ended up using continuous scales for the ratings instead of ratings between 1–10. The reason is that this eliminates much of the possibility that, when it comes to rating the second session, the player will remember what they answered previously in the first session (c.f., Passmore et al., 2002).

We chose to conduct a small test with two respondents to make sure the implementation, and survey would work. This test was conducted in the same manner as the full test would be made. Since we already prepared for the tests in our previous pilot study, there was not as many initial mistakes this time around. We found that we had a good length for the test, no game breaking bugs or other inconsistencies.

During the test we realized that some changes were required. Among which, we want to have the thresholds for the flow tracks (explained in 2.2.2) to be dynamic depending on the average speed of the player. This would allow slow players to trigger the flow mode easier, which could mean that if they happen to play faster in some area they too can reach it. This would also solve the problem of setting a threshold which a very fast player could stay in forever, which could be very tiresome.

There was also somewhat of a hiccup when it came to the in-game tutorial showing the player the mechanics in order to learn the game more easily. This resulted in the tutorial not being present in the small test we conducted, severely hurting the flow of the play session. This did however give us a valuable lesson that we could not afford to leave out the tutorial when it came to the proper playtest sessions later on. This was fixed for the rest of the tests, which meant we could test it properly with the rest of the respondents.

2.3.2 Playtesting

Coming from our pilot study, something we wanted to improve was the fact that we really needed to expand the amount of playtest sessions and participants. Last time only six playtest sessions were carried out, a number we felt was sub
par and also affecting our data in a very negative way, especially when it came to our analysis.

Unfortunately, we had to cancel a session with playtesters due to unforeseen circumstances. Having lost this opportunity, we chose to make some changes to the way that the study was conducted, by using a more automatic approach. It is a shame that we could not use these testers, since these had not played the game before and would be ideal respondents. Instead, we chose to ask classmates and other acquaintances to respond to the survey. In the end, we had 16 respondents. Most of these are game developer students, with some exceptions. Further information of the playtested demographic is detailed below.

| Age: (Min=16, Max=46) |
| Gender (Male=81.2, Female=18.7) (Per cent) |

We added a start screen to the game, which quickly described to the players what they were about to play. When the player pressed the button on the interface which started the game, one of the versions was chosen randomly and then loaded for the player to play. At the end of the level, a survey was opened inside of the game, with sliders representing the value of how much the player liked the aspects of the game. After the survey has been filled out, the other level is loaded. When that level is over, the player will receive the same survey which they will also have to fill out.

We did make a critical mistake with the instructions, however. Nowhere in the instructions were the player told that they were to have sounds enabled so the player can hear the music we were testing. This might have affected our results in one way or another.

A text file is saved containing the data from the surveys, storing the answers that the players gave in a JSON-format (Crockford 2006). The files are sent to us using a survey created with the web-based software Google Spreadsheets (Google Inc. 2013) where it is compiled into a spreadsheet for us to analyse. In this survey we ask for the respondents gender and age, as well as an estimate of how many hours they spend each week playing video games. Instructions on how to send the text file and installing the game is provided in this survey. The survey can be found in it’s full form in the Appendix.

There are still areas which could be optimized with this approach, with the most apparent area being that the data could be sent over the internet directly from the application, which would remove parts of the test which can be confusing for some users. However, due to the already delayed testing and low amount of time we had to finish this study, we had to begin testing as soon as possible. We do think that this is a better approach than the first approach we chose though, since we no longer need to observe the players while they are playing, and changing between the levels. The use of a JSON-format means that it would be easy to create a script for whatever software chosen to analyse the data received from the respondents. This could be worth the time investment if the amount of
respondents are a large amount, where the time spent manually entering the values would exceed the time it would require creating the script.

This approach does, however, introduces an issue with randomness, where the order of which the players play the levels are left to pure chance. We found in the pilot study that players’ generally rate the game higher the second time they play it, as discussed in 2.3.1.

2.3.3 Analysis

Analysis of the data we gathered during the playtests were generally straightforward, as we only had to consider two aspects of the variables in the data: music, and everything else. We do not focus on other things rated by the playtesters such as graphics, difficulty, etc., as much as music. One way we approach these other values is by rolling them into one value, as to get an all-encompassing rating for everything not associated with the music.

We do, however, look for patterns in the other categories, as there is always a chance that the adaptive music system could make the player perceive the difficulty of the game to be better suited to them, or maybe bring forth a better gaming experience. In the aforementioned pilot study we had the music quiet down in a section where the player were not to move, and raise it to normal when the player was allowed to move. This was something we chose to leave out, due to the fact that we changed what part of the game the respondents would play, to one more suitable for new players. It is an example of where it could help the player progress through the game easier.

We chose to inspect the selected data in form of a box plot, with a heavy focus on the music, but also plotted the rest of the data categories as to provide an overall picture. This plot is presented in 3.1.

We tested our findings further using Student’s t-test, a common statistical method to test the significance in difference between sets of values by comparing the means and standard deviation. (Arumugam, 1981). The type of t-test used was a Two-Tailed Paired and it was made using the software R (R. Core Team 2012). We looked into the possibility of using ANOVA instead of t-test, but we ultimately chose the latter because the datasets did not meet the requirements of ANOVA, such as variance and distribution.
3 Results

3.1 Analysing the results

By leaving the distribution of level order to pure chance, the results are skewed and favour one of the levels over the other due to it being chosen second, as detailed below:

**Distribution (Static=56.2, Adaptive=43.8) (Per cent, level played first)**

### 3.1.1 T-test – Music Score

Our findings showed that there was no significant difference in the scores for the Music variable in the survey. Below is the results for scores in the music category by the factor adaptive, static:

**Adaptive Music (M=0.670, SD=0.262), Static Music (M=0.651, SD=0.261)**

Conditions; t(30)= -0.21, p=0.83.
3.1.2 T-test – Game Experience Score

We opted to test the correlation between Game Experience scores and the type of music system presented. Below is the t-test results for scores in the game experience category by the factor adaptive, static:

| Adaptive Game Experience (M=0.730, SD=0.150), Static Game Experience (M=0.771, SD=0.117) Conditions; t(28)=0.86, p=0.39. |
4 Discussion

4.1 Results

The data we received tells us that we found no significant difference between the two systems. The study was flawed in several ways though, not only with the way we gathered data, but also the system implementation and the actual music used. In this chapter, we will discuss some of the areas which could be improved.

After the session with playtesters was cancelled as discussed in 2.3.2, we chose to create an automatic system for respondents to partake in the study remotely. At the time, we did not take into consideration that we could in no way ensure that the test environment of the respondents was adequate for our study. The biggest issue of this was discussed earlier, where we had no way to make sure that the player even had sound enabled while playing. In retrospect, we should have made sure to be present at the time when the respondents were playing the game. We could have achieved this if we had planned for more playtesting sessions.

A common comment was that the system was very erratic. It would start and stop the music far too often, which made the players feel that the music was not as good, since it sounded as if the music was not working correctly. This could probably be fixed by introducing some delay with the music, forcing it to play for a certain duration, even if the player would drop below the lowest flow mode threshold. The same goes for when the music has dropped, where it would have to remain quiet for a duration. It may also have been that the parameters for the system was not tweak to perfection, which is highly likely due to the development of the system being very rushed.

We had another comment from a respondent who had not heard any music whatsoever in the non-adaptive version, which makes it reasonable to believe that the music in the static version was too subtle, at least in comparison to the adaptive version. The music that was playing in the static version is a very ambient track, and was perceived as just ambient sounds in the level, rather than a music track.

Another thing to consider when figuring out why the adaptive music was not preferred is the skill of the composer of the music in composing linear compared to adaptive music. Before this study, the composer had never composed an adaptive soundtrack, and received very little time to do so before we started testing. We can assume that the scoring of the music was equal enough though, by looking at the Music variable in the survey we had no significant difference in the score between the static and the adaptive version.

From the $t$-tests we can gather that our research question has been answered. Since there is no significant difference in either Game Experience or Music we can
conclude that the system did not make our game any better, and definitely not worth the time investment.
4.2 System Implementation

Even though software like FMOD exists for bigger projects, lighter and leaner code bases for smaller endeavours like our game will still have purpose and reason. We feel that with a small project on one of the more accessible game development platforms out there, a smaller system for adaptive sound support could be easier to implement on top of an already existing sound system instead of having to integrate a big enterprise-level software suite into the project.

We feel that the system developed for this study does show some potential in some cases, and failing mostly on the flow mode section. Being able to smoothly transition between different tracks may prove useful, however, by just transitioning between different tracks as the player progresses through the levels. It is arguable whether this could be described as adaptive music, seeing how it is more of a linear change of music. This system does, however, attempt to have the transitions be smoother, only switching tracks at appropriate points in the already active tracks. A possible use for this would be where we want to signal to the player that a significant change has happened, an example of this can be found in 2.2.2. This might also include events in the story of the game, for example when the player meets an important character for the first time.

4.3 Future work

As mentioned in 2.2.2, we wanted to have the music change if the player fell into the water, as well as play a music cue. The problem with this is that at the moment we do not have a good time frame for doing this, since the player is moved back to the last checkpoint instantly as he or she hits the water. If we were to do such a change, we would have to delay the return to the checkpoint somehow. The solution we will most likely choose is to fade the screen to black and while fading the music, then fade up when the player is back at the latest checkpoint.

The flow mode discussed in 1.5.1 could use a lot of improvements. As it is right now, it is more or less a proof-of-concept, mainly with the dynamic skill level part of the system. Most importantly in this part, the method we used to determine how fast the player is is very short-sighted. It resets each time the level changes, which mitigates the purpose of the system until after a while when it has adjusted to the player’s speed again. Up until then, the same problems arise that we had before this dynamic system was introduced, where fast players can stay in the flow mode very easily, not being rewarded for being extra fast in some section, and slower players will never be able enter the flow mode.

The versatility and abstract nature of this system makes it quite easy to port to another system or game engine. Even though it does use Kismet for most of the implementation of sounds, in a different system we could do away with it for a code-only approach or another visual solution like the Inspector in Unity (Unity Technologies 2013).
We feel that the approach we used for collecting the data was far from perfect. One of the biggest issues is that the test level we used was far too long, and players would become bored as they played. It also meant that we scared away a lot of potential respondents due to the time investment that would be required from their part.

Another thing to consider is having the quality of the different types of tracks to be equal. In this test, it is very probable that the quality of the linear music was much higher, since it had received more time when being composed, and it being specifically developed for the game map used in the study, as well as the composer being used to composing linear music. Before any further tests, the tracks should be tested on enough people to ensure that the quality is as good. It would be important in this case to make sure that the tracks are tested in a way which allows the player to rate them according to their thoughts about how well the track is suited to the map which they are playing.
5 Conclusions

As discussed in the results chapter, the data we received from the tests showed no significant statistical difference between the two cases in the game. This answers the question we posed whether the game experience would be better if the game had adaptive music, showing that it posed no difference in the experience as a whole. It also answered whether it was worth the time investment for implementing the system, which it was not, since the time it took to implement the static music was only a fraction of the time it took to implement the adaptive system.
6 References


Claynote, 2013. G.A.M.E., Claynote. [Computer program]


Epic Games Inc., 2013. Unreal Development Kit, Epic Games Inc. [Computer program]


Firelight Technologies, 2013. FMOD, Firelight Technologies. [Computer program]


Google Inc., 2013. Google Spreadsheets, Google Inc. [Computer program]


Unity Technologies, 2013. *Unity*, Unity Technologies. [Computer program]


Appendix – Survey

Instructions:
1. After you press play you will get to play a level from the game A Story About my Uncle.
2. After you have navigated through the level you will be met with a series of questions and sliders to answer them.
3. You will then be met with the same level a second time, play through it a second time then answer the questions at the end.
4. There will now be a file on the UnityGame/Saved where the game was installed called “survey file”. Have this file open in notepad as you continue the test part of the survey in the online form.

Thank you very much for your participation and helping us conduct this research.

Figure 3a – Start Screen

Figure 3b – In-Game Survey
Survey for study at Södertörns University

--- READ THIS ---

Hi!

Thank you for helping us with our study! Your help will be incredibly useful for us. You will play a section of the first level of the game A Story About My Uncle two times, and after playing each level you will fill in a survey. Since this is a very rough build of the game, we have chosen not to go entirely public with the survey, so please do not spread the link.

--- STEPS ---

1. Download the installation file here: http://teabit.com/Asamu-Cuppsats-Installer.zip

2. Instructions for installation will be provided with the setup file. Please take note of where you install the game, because you will need to navigate there later. Please do not close this survey until you have played the game.

3. Start the game, and play through it following the information displayed. When you have completed the test, the game will close itself.

4. After this, fill in the questions below. In the survey data, you should navigate to the folder where you installed the game. In this map, there is a folder called UDKGame, and inside that you will find a folder called Stats. Inside of this folder you will find a text file called Survey.txt, open it. This can be done using many programs, notepad as an example. Copy all the text inside it, and paste it in the field below titled "Survey data".

5. Please uninstall the game from your computer. This build of the game is rough and not something to be the experience that is associated with the final product. If you wish to try the rest of the game, you can find the first build we made in 2012 at our website: http://astoryaboutmyuncle.com/

All the data we collect is anonymous.

Thank you!

*Required

Gender *

- Female
- Male
- Other: ____________________________________________

Age *

__________________

How many hours do you estimate that you play video games per week? *

__________________

Survey data *

Here you will paste the data from the survey. You find instructions for this in step 4 above.

Submit

Figure 3c – Questionnaire with instructions