Bachelor thesis for Creative Technology

University of Twente

Dynamic Audio Solutions for Games

Author:
Robert Blaauboer
s1202014
August 2014

Supervisor:
Dennis Reidsma

Abstract
Music in games is a potent way to provide feedback to the player but, due to their respectively often linear and non-linear nature, there are certain systems that must be used in order to do so. This paper outlines two possible systems to help provide musical feedback in games. One, called Procedural Mixing, uses pre-composed segments to generate a score based on the input of multiple real-time descriptive parameters. A system of three algorithmic rules is used to ensure that the score sounds musical and cohesive. The other, called Music Timing, shifts the timing of in-game events such that they align with strong beats in the score. A user test is conducted with a proof of concept in order to test possible influence of such a system. Both solutions show potential for further development. When a strongly timed orchestral score and game-events are well aligned there is potential for a positive influence on the impact of in-game events. The Procedural Mixing system works well considering the current implementation and needs more content and rules in order to provide better feedback to in-game situations.

Chapter 1

Introduction

Sometimes things fall into place and keep you in a flow. Everything seems to complement each other and you are completely immersed in the world. Or, as the film industry has known for a long time, immersed in a movie. Take for example the process known as Mickey Mousing where every action is accompanied by a musical element. Only musical cues are used and still everything feels accentuated. In the average film, music is written according to cues in a reel. These cues can be all sorts of important events from the reveal of the killers motive to the final uppercut in a fight scene. All these important events are underlined — and are able to be underlined since film is a linear medium.

Both sound and film, however, are not required to react to user input or random events, two features that are very much a part of games. There is a certain challenge in attempting to unlock the full expressive power of the combination of games and music. This would involve combining the accentuating of important events and overal expressive power of music in film with a game that is interactive and has its own random game events. There are several solutions that can be examined in the pursuit of solving this challenge. One solution might be to create a score without a steady rhythm so new parts can be introduced at any moment without sounding abrupt. Such a score can be modified on the fly according to environmental factors or events triggered by the player. This can work quite well in calmer games or games without rhythmic music as an artistic choice (as is the case with *Dishonored* (Arkane Studios, 2012)). Still, battle scenes might be left a little stale and uninspiring if a driving beat is left out. Another option, and one of the personal contributions that will be further explored in this thesis, would be to have the music relay information back to the game. If the game has knowledge of the nearest beat in the music it can shift the timing of certain events such, that musical stingers can be introduced in time with the current music (game-follows-music). This keeps the flow of the music going and allows for accentuating important events.

There is another option which is a reverse of the beforementioned option. Here, the result would be a system where game events determine which music is playing (music-follows-game). An ideal combination would be music with film quality expression, carefully created by a composer, with the interactivity and adaptivity[1] of procedurally generated audio. More about this in the Related Work.

The rest of this thesis is structured as follows. In the next section some general related work will be discussed as well as related work specific to the two options explored in this thesis. We then move on the explanation of the Procedural Mixing system. This is followed by an explanation and user test description of the Music Timing system. Then a short recap and final remarks will be supplied in the Discussion and Future Work.

[1] Collins [9] defines interactivity as directly influenced by the player and adaptivity as influenced by the surroundings and game environment.

Chapter 2

Related Work

In this section several relevant and interesting music systems will be discussed. The requirements for the two options explored in this thesis will follow from the weak and strong points of these systems.

2.1 Related Work - General

While dynamic audio middleware, in which audio reacts to for example player input, exists and is available for licensing, some game studios choose to implement their own system for interactive audio. Though the cost of initial effort is higher, it allows for tailoring the system to the kind of game being developed. For example, the latest iteration of the snowboarding game *SSX*(EA Canada, 2012) uses a system called the Realtime User Remix System[5] or in short, RUMR (later changed to Harmony) where users have the choice to import their own music. Runtime parameters are used to control several effects such as sampling, EQ filtering and other DSP effects. A beat detection plugin is used to establish a map for remixing these songs. Parameters include the time a player spends in the air or on a rail. These effects accentuate the players actions and can make them feel more impactful. In addition, the fact that the player can choose their own music also increases their feeling of agency[16].

2.2 Related Work - Procedural Mixing

2.2.1 Borderlands 2

In the FPS game *Borderlands 2* (Gearbox Software, 2012) you roam the wasteland as a vault/bounty hunter. The music system makes use of a context pool

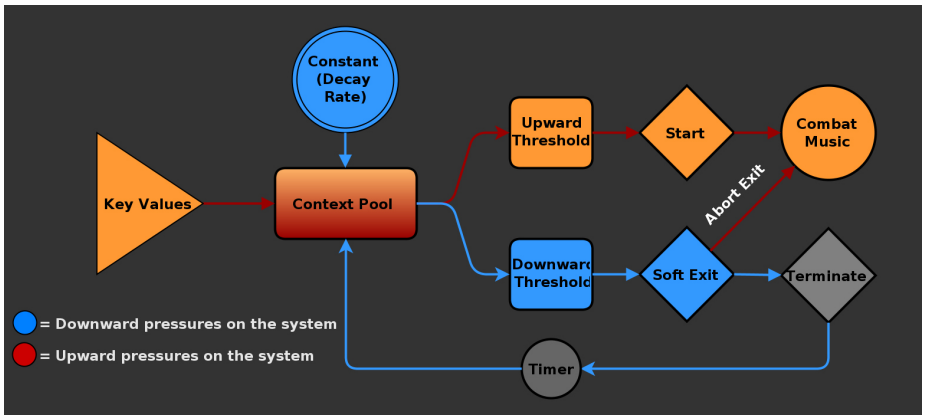

Figure 2.1: Context Pool Diagram of Borderlands as explained in [22]

and a constant decaying parameter during battle scenes (see Figure 2.1, taken from [22].

Several key values such as enemy level, amount of enemies and player health are put into the context pool. Together these values determine the excitement and danger that should be present in the music. In some cases the level of danger may not be high enough to even initiate combat music. After combat the excitement parameter will slowly decay and bring the music back to an ambient mood. Because this happens slowly it is possible to have gaps between combat gameplay sections without the combat music ending altogether. This helps maintain a constant level of excitement.

2.2.2 Psai

The psai system[4] is an audio system build specifically for interactive music. It deals with pre-composed music segments that have been given an intensity parameter and are linked to a specific theme. This theme can be any of six types. First, there is the Basic Mood, this is general background music, and Basic Mood Alteration, which is an enhancement of the Basic Mood for locations that need emphasis. Then there is the Dramatic Event, which interrupts the Basic Mood immediately and can be triggered to emphasize important moments of dialogue. The Action Event interrupts the previous themes immediately and is fit for combat. The Shock Event will interrupt any theme and is used for a sudden attack or death. Lastly, there is the Highlight Layer which is the only theme that can play simultaneously with another theme. The entire system can be implemented into Unity and be set up to interact with the game worl d. It seems to be quite similar to the system used in *Borderlands 2* since the intensity of the music has a constant drop. The fact that some themes can react immediately to a situation is beneficial to the player feedback on in-game

events and the combination of different themes for different areas allows for a wide variety of music for any area. One must be careful however to make the conte nt pool sufficiently large for every level of intensity in order to combat listener fatigue[9] during for example a battle sequence of more than five minutes since the segments themselves are pre-composed and dont change. Another nice addition is the pairing of highlight layers with themes. This ensures that the extra layer fits harmonically but does limit the amount of change one can have within a theme since it all has to fit with the same highlight layer.

2.2.3 Proteus

Whether *Proteus*(Curve Studios, 2013) is a game or an interactive experience depends on your definition of games. In *Proteus* a procedurally generated environment can be explored which is fully voiced through all sorts of musical cues. When you get close to a frog, it will jump away and trigger a short two-note melody. This is the case for everything you encounter in the immersive world. Depending on parameters such as time of the day, location and objects in the environment the music will change. While parts of the music have pre-composed melodies, large parts are also generated and together create the synthesizer based score.

2.2.4 SimCell

"*SimCell*(Strange Loop Games, 2013) is an educational game meant to teach high-school students about the human cell [17]. The music is inspired by the generative score in the game *Spore*(Maxis, 2008) . It is being generated real-time in a program called PureData[20] based on parameters in the game. A couple of PureData instrument patches created by Martin Brinkmann[8] are used to play several melodies. Different scenarios have different pre-composed note sequences. The specific timbres the melodies have are determined by the in-game parameters and their instruments. The entire score sounds very ambient and wide with a lot of reverb and echo. This works well with this specific game since it fits with the organic nature of the simulation. When specific events happen in the game it triggers a certain scene in the music which is specified to accompany certain events. Besides being able to change scenes it is also possible to change other parameters influencing the music such as the bpm.

2.2.5 Experience-Driven Procedural Music Generation for Games

In this paper by D.Plans and D.Morelli[19], a system for experience driven procedural music for games is described and tested. Frustration, challenge and fun have been mapped as functions of the player's gameplay data and have a

correlation to certain gameplay mechanics. In order to relate these functions to music they have been mapped to an excitement parameter. Then five rules are set up which have a certain effect on this excitement, an example being musical novelty, where repetition of material provides low excitement and new material provides excitement. A small set of phrases, called seeds, provide some musical sequences from which a large number of variations are made. These phrases, the melodic element, and the generated chord sequences, which is the harmonic element, are then evaluated by fitness functions in order to determine their excitement. The phrases and sequences with the minimum deviation from the desired excitement are then chosen.

2.2.6 CBS: A Concept-Based Sequencer for Soundtrack Composition

Four categories of algorithmic music composition have been described in this paper by Jewell et al. [14]. The categories are as follows: Rule Based, where the system must follow strict guidelines, Stochastic, using Markov Models to represent movements between states, Grammar-based, where musical elements are encoded as grammar symbols in a string which is manipulated, Genetic, based on Darwinian evolution and utilizing a collection of chromosomes. The rest of the paper goes on to describe the main subject, the concept-based sequencer, which is less relevant here.

2.2.7 General Remarks

Looking at the list above containing different solutions for interactive music it can be seen that there can be many different approaches. *Borderlands 2* and the Psai system both make use of a decaying value which represents the amount of action and allows for a smooth transition back to ambient music. This system works well especially in combat sequences since it is possible to indicate the amount of danger a player is in. If one would want the music to represent more information, then more parameters/values can be implemented. This enables the system to more accurately describe situations by, in a way, increasing its vocabulary. *Proteus* and *SimCell* make use of procedurally generated music combined with pre-composed note sequences. This keeps the music new to the listener and also retains a recognizable melody. This is important when dealing with generated music since the listener can quickly feel lost when there is no recognizable element to ground the player. Due to its generated nature it is able to react well to player actions and game events and thus provide the player with feedback about his actions and surroundings. Still, both games are quite unique in objective and style so this generated approach might be less suited to, for example, a first person shooter style game. These games generally make use of orchestral scores which are much less suited for procedural generation since most scores are pre-recorded.Collins [10] says that having more procedural music

in games is likely, particularly with mood cues tagged to in-game events and goes on to suggest that not the entire clip needs to be procedurally generated but that algorithms could control instruments, melody lines or sections of the piece. In another article [9] she mentions that there are difficulties in composing dynamically, but that, for example, randomization can be used to alleviate this difficulty. Still, it is important to make any dynamic music solution as easy to compose for as possible.

Other systems such as Wwise[6] or Fmod[21] are presently audio delivery frameworks for pre-recorded data rather than real sound engines capable of computing sources from coherent models.[11]. They provide a way to call different music sections and layers from the game engine which allows game studios to integrate such an interactive music system without having to create their own. This can be a good enough solution in many cases as not every game needs a custom system or has the resources to create one.

Based on this related work, in this thesis a prototype of procedural mixing will be built. This prototype uses the Rule-Based system [14] in order to attain certain target variables in the generated music.

2.3 Related Work - Music Timing

Although academic related work fitting to Music Timing was searched for, this could not be found. This leads to the assumption that not much research has been done in this area, the area of musical feedback on visual elements, and that there is an opportunity for research. Given the amount of communication already possible between audio and game engines it is possible to achieve this in modern games. For example, the game *SSX* uses beat tracking in order to remix songs according to the player's actions, but most likely, gameplay events are not synchronized to the music beat. This might be an opportunity for improvement especially when landing after a long jump. Currently a short moment of slow motion is already implemented to increase the sense of impact when hitting the ground. This moment could be tweaked slightly such that it aligns with a strong beat in the music.

Kastbauer [15] says the following in an article in Game Developer Magazine:"There lies a magical place somewhere between content creation and tool-side implementation, where the flow of game audio is fed back into the game in a two-way street of communication. And that place lives just outisde the reach of most toolsets". While the Music Timing system is not a two-way street, it does communicate in a direction that most systems don't since the game follows the music.

Chapter 3

Procedural Mixing

In this chapter a possible solution to some of the issues in interactive music systems will be described. The solution described here is called Procedural Mixing and uses pre-composed segments and parameters to create music segments that fit with in-game situations.

3.1 Objective

From the Related Work it follows that there are several issues in interactive music that can be improved.

Firstly, the system should possess multiple parameters that can be used to describe different gameplay scenarios and enable the system to provide well fitting music . It has been shown that music can induce mood to a listener[18] and thus it is imporant that the mood of the music fits the mood of the gameplay scenario.

Secondly, the system should be able to create a score from pre-recorded musical segments that provides feedback about in-game events. This is important since many games make use of pre-recorded scores but in doing so are less able to integrate a lot of the interactivity that is present in real-time generated music. Current systems are able to shift between music segments based on in-game situations but usually these transitions take too long or are abrupt and also have a very limited amount of freedom. For example when using horizontal composing[1], a new segment has to be created for every situation that has different parameters. Collins [10] draws the link between composing in segments of which the order is determined by the game and open form composition, also

[1]Horizontal composing is a way of creating a dynamic score. Musical segments are composed and can transition into eachother directly or through the use of a transition segment. The result is a score very similiar to film scores.

known as recombinatorial music, where the sections of music are composed but the sequence in which the sections are performed is left to the the conductor, performer or even a pair of dice. Due to each segment being divided into multiple layers there is a large number of possible segments to play and thus makes it quite likely that not every combination will be played.

Lastly, the system should make it easier for a composer to compose a dynamic score that needs to have a large range of variations. Often a composer likes to change the scale he is using or add a note or chord that is not part of the scale used. While this can sound interesting it does become a problem when two parts including notes or chords outside of the scale are played simultaneously and end up sounding very dissonant while this not being the intention of the composer. If the system knows which parts fit together and which don't it can determine which combinations sound as intended by the composer. Various approached can be taken here such as grouping segments or identifying the correct matches through machine learning. So in short, there are three main objectives for the Procedural Mixing system and they are as follows: Provide multiple parameters to accurately describe gameplay, Use these parameters to create a fitting score from pre-composed segments, Utilize the segments in such a way that it is easier for a composer to create the dynamic score.

3.2 Workings of prototype

A working prototype has been created by me in order to achieve the objectives mentioned above. The way in which these objectives are achieved are described below.

As mentioned previously, the music will be composed in segments. The length of these segments determines the speed in which the system can react to in-game situations and has to be decided based on the kind of gameplay that it is accommodating. Every segment will be rated through parameters, which are predetermined by the development team, and should be able to describe every situation that the music needs to accomodate. In the current prototype four parameters are used (Eg. Happy/sad, Epic/Scary etc.) and the gameplay situation is a battle situation. Some possible situations that may have to be accommodated are "Player is winning", "Player is losing","Player is in severe danger","Enemy is defeated". Ideally the composer is the one that rates all his composed segments in order to have the music stay true to his intent.

Additionally, the segment combination chosen to fit the current game parameters have to fit together musically. This is solved through placing all segments in a grouping hierarchy which describes which segments fit together and which don't. This grouping hierachy will also be created by the composer and allows the composer to compose segments that are very different and let the compatibility be handled by the grouping hierarchy. In this way not all segments have

to fit together musically which makes it easier for the composer to compose a dynamic score.

When all the composing, rating and grouping has been done the system has content and information to work with. The game engine will supply data to the procedural mixing system which translates these to the, in this case four, parameters. Of the four categories of algoritmic composition specified by Jewell et al. [14], this system falls into the Rule Based category because the system creates combinations according to, currently three, main rules:"Parameter Check","Compatibility" and "Amount of Change", which was already lightly touched upon in the previous two paragraphs. If a combination of segments passes all three rules it is approved and set as the next upcoming combination to play (See figure 3.1), unless a change in parameters happens before the combination has started playing.

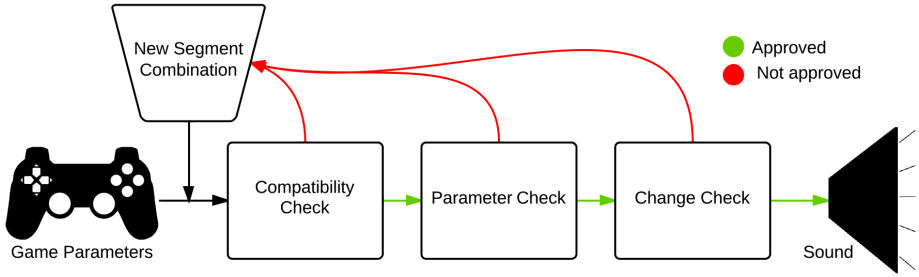

Figure 3.1: Diagram showing the Procedural Mixing Process.

The rest of this section explains these steps in more detail.

3.2.1 Parameter Check

This rule deals with selecting the right segments for the job. It monitors the difference between the ideal parameter values, based on current in-game parameters, and the rated values of the segment. If this difference is lower than the set threshold then the parameter will become satisfied. Parameters can be satisfied multiple times so an ideal combination would be when the amount of satisfied parameters is 16 (which is the maximum in this case, since there are 4 tracks and 4 parameters). But the satisfaction number can be lower while still describing a fitting combination. Some tweaks are added to this check so a chosen combination will feel more fitting to in-game changes. First a priority parameter is chosen. This is the parameter that has the largest difference from the parameter values on which the currently playing combination is based. This priority parameter has an individual minimum satisfaction number. This remedies the possibility of large changes happening in a game while playing a segment combination that satisfies all but the parameter most indicative of these changes. The other tweak monitors the maximum parameter difference.

The parameter value difference between a segment and the ideal parameter may not be larger than this number. This ensures that, while the segment may not satisfy the parameter, it also does not dissatisfy it. Meaning for example that when the ideal value is 8 on a scale where 0 is sad and 10 is happy, that the segment is not too far towards the sad end of the scale. In this way large differences between segments can be prevented and allow for more cohesion between segments in a combination.

3.2.2 Compatibility Check

This part monitors the compatibility between segments playing at the same time.

The issue regarding ease of composition with regard to well sounding combinations was approached as follows. Due to the scope of this prototype, dividing the music into groups was opted for instead of machine learning. While the latter might provide very interesting results, it is an intensive subject and is not part of the scope of this research (see Discussion). It will be assumed that music is composed in different tracks, each with its own timbre. This division can be per instrument, instrument group(brass, woodwinds etc.), texture (rough, smooth) or musical purpose(bass, accompaniment, melody). In this prototype the latter has been used. For each track a range of different musical segments are composed, each of the same length. The way in which these layers are combined through rules fits the profile of a transformational algorithm more than a generative algorithm as described by Wooller et al. [24]. Transformational algorithms can control for example the instrumental parts that are added instead of generating entire musical sequences, which would be generative. Each segment is composed in such a way that it fits with all the segments from other tracks within its group (see figure 3.2). While this does not completely solve the difficulty of composing a dynamic score, it does make it easier by decreasing the amount of segments that have to be compatible with each other.

All groups are arranged in a pre-determined hierarchy. The segments in the lowest level groups are compatible only within their own group and can't be combined with segments from other groups on the same level. The segments from mid-level groups fit with segments from their own group and a combination of lower level groups (see the relations between groups in figure 3.3).

In the example in figure 3.3 the low-level groups are the C-groups. The mid-level groups, in this case B-groups, are compatible with a combination of low-level groups. Eg. B2 is compatible with either C3, C4 or C5. This can result in a combination of three different segments from group C5 and one segment from group B2. But, a combination with a segment from group B2 and one from C3, C4 and C5 each is not possible since groups are not compatible with other groups on the same level. In the same way a combination of B1 and B2 is also not possible. There is one highest group which is compatible with all groups and thus also with all mid-level groups. This might result in a combination of

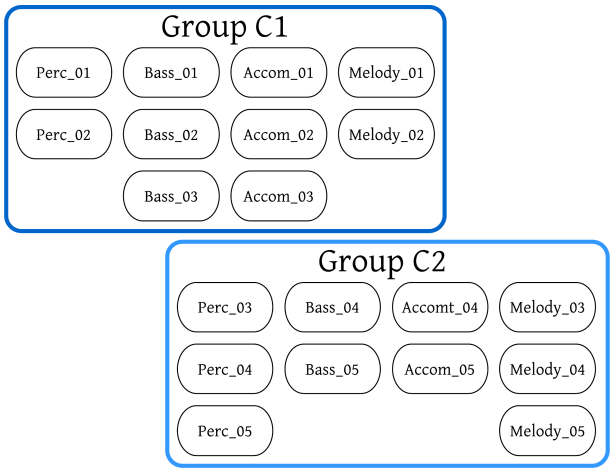

Figure 3.2: Possible segments in a group.

A1, B1 and two segments from C1. There may not be many tracks that fit this highest group ,except for example some percussion segments, and it could also be left completely empty serving only a symbolic role in the hierarchical structure.

Of course, this hierarchy can consist of more than three levels. Another level could be added containing for example a group that fits with group B1 and B2 but not with B3 (not shown in figure 3.3). This group would then be placed between the A- and B-level. Using this type of hierarchy, not all segments have to fit together and a large amount of possibilities and variety is maintained through combining the different groups.

When all segments in a combination fit together, according to the grouping hierarchy, the combination is approved.

3.2.3 Amount of Change

This rule monitors the change between the currently playing combination and the newly chosen combination. When there are multiple tracks that all have a range of different segments it is easy to create combinations that have nothing in common with the previous combination. As mentioned by Fay et al. [12, 373] "Too much variation, however, can unglue the music's cohesion". It can also increase the chance of listener fatigue[9] and make the sequence of combinations feel like just that, a sequence of combinations, instead of feeling like a cohesive soundtrack. Thus, the Amount of Change rule limits the difference in mood between the old and new combination. For example, the maximum difference could be set to 2 new segments. So a hypothetical combination of S1, S2, S3 and S4 can't be followed by a combination of S5, S6, S7 and S8 but can be

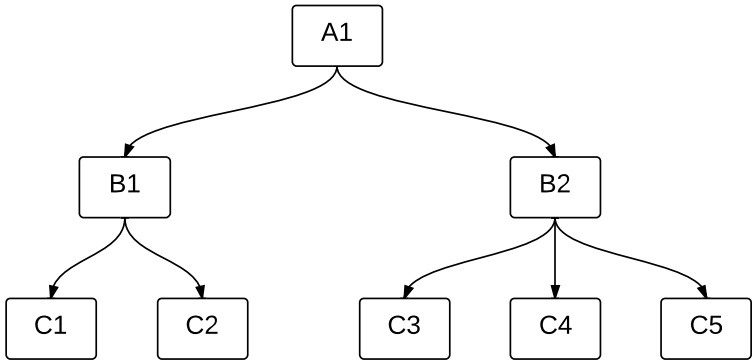

Figure 3.3: Group Hierarchy (example).

followed by a combination of S1,S2,S7,S8 for example. The maximum amount of change however does depend on the amount of change between the old and new parameter values. If all parameters suddenly experience a big shift the maximum amount of change may go up to 4 and all segments are allowed to change. In this way it is still able to accommodate large or sudden in-game changes while staying congruent in other situations.

If a combination of segments passes these three main rules it will be the next playing combination. If not however, a new combination of segments is chosen and run by all rules until a combination is found that satisfies all three (fig 3.1).

3.3 Used Technologies

In order to have smooth playback of music a dedicated audio engine dedicated can be used. When dealing with multiple segments which all fit together it is important to have sample perfect transitions. Max/MSP[2] is a visual programming tool which is used mainly for audio purposes. As discussed in related work, visual audio programming tools have already been used for creating and regulating music and sound in games. The free alternative of Max/MSP, PureData[20], has a library called LibPD[1] which can run alongside Unity[3]. The visual aspect allows a non-programmer to more easily understand what is happening and usually allows for easier troubleshooting. In this case and for this prototype, Max/MSP was used. Even though it is not free and as light as PureData, the ease of use and the availability of more dedicated objects for specific tasks makes it more intuitive. Whether or not it is possible, technically or license-wise, to

implement Max/MSP into a game engine such as Unity is not a problem here. The goal of the prototype is merely a proof of concept. When the entire structure of the program is clear and tested it is still possible to port it to PureData or another programming language if necessary.

3.4 Discussion

Procedural Mixing in the current stage already shows decent results. The grouping hierarchy works well and using parameters to request a new combination allows for a large variation in feedback to gameplay. As a proof of concept Procedural Mixing works good enough to warrant a next step in which cooperation with a game development team is important for proper testing and adjustments to the system.

As always with a prototype, there is room for improvement. While the main idea shows promise, some additions are needed for an improved experience. Firstly, the content pool of music segments needs to be increased. Currently there are between 20 and 30 segments in the system which only allows for a very limited number of combinations. When taking the grouping compatibility and parameter rating into account the amount of combinations for a random set of parameter values might be 1 or perhaps not exist at all. If the content pool can be increased such that any set of parameter values has at least two possible combinations this would improve the flexibility and experience of the system drastically.

While the currently implemented rules can provide interesting results when combined with a larger content pool, more rules can be added for a better experience. These additions can monitor for example the mix of the combinations to make sure that a certain frequency band is not too saturated and making the whole sound muddy. Another rule can monitor the link between segment combinations. It is possible that the composer would want to create melodies that are longer than the standard segment length. When certain melodies have follow-up segments that extend the melody, this rule could suggest a follow-up segment for the new combination and ensure a longer build up for the melody over several combinations.

Lastly, due to this being a prototype the choice was made to use grouping instead of machine learning for compatiblity between segments. Still, replacing the grouping with machine learning might provide interesting results. Ideally, the composer would judge a wide array of random combinations and judge whether they fit together. But as coined by J.A. and Juslin [13] and later confirmed by Berg and Wingstedt [7], non-musicians are just as capable of experiencing musical emotions as musicians are and thus are also able to judge such an array of combinations. After a sufficient number of iterations the system will have a map of which segments fit together and which don't. Segment rating might be implemented in the same process as the grouping. Here the whole

combination would be judged on the parameters and again, after a sufficient number of iterations, the system is able to deduct the rating for each segment individually.

Chapter 4

Music Timing

As games increase in size and budget its production qualities also increase. More and more scores are recorded by large orchestras and orchestral soundtracks for some games rival the quality of those made for blockbuster movies. When dealing with games that benefit from these types of soundtracks we run into a problem. The non-linearity of games does not immediately fit with the linear nature of blockbuster nature of orchestral scores. In a film context, these scores synchronize exactly to cues in the film and thereby enhance the effect this moment has. It might be expected that this positive effect can also be used to amplify the impact of certain events in a game with an orchestral soundtrack. Still, modifying the soundtrack at any moment is tricky since it has a certain rhythm, timing and flow. Instead of trying to modify a fairly linear soundtrack, what if in-game events are to be modified such that they are more congruent with the soundtrack? Many videogames already have non-linear events that can happen at any moment and are usually the result of AI behaviour. Some of these events can be adapted to fit the timing of the music so the game can fully profit from having an orchestral score.

This chapter presents the second part of my work and another solution to some of the problems in interactive music. The system described here is called Music Timing and is based on principles described in the previous paragraphs.

4.1 Prototype

An experiment has been performed to test the validity of the assumption that the impact of in-game events depends on its timing with regard to the orchestral score. If it is true that strongly timed music underscores important game events, then it is possible to use a fairly linear film-like score while still providing feedback to game events, if used in tandem with the Music Timing system. In this system the audio engine tell would tell the game engine about upcoming beats

in the music. The game engine would then shift the timing of some in-game events such that they align with these beats.

In order to test this assumption, two videos were created, one with the score and game events being well-aligned and the other without having the score and game events being well-aligned. The video consisted of a gameplay segment from *The Elder Scrolls V: Skyrim* (Bethesda Game Studios, 2013). Using video editing software, important game events were synchronized to the closest first beat of the score. These important events could for example be: A dragon landing from the sky, a dragon breathing fire or a dragon flying away into the sky. The second video was subtly edited such that none of these important events lined up with the score but without noticeably breaking the visual flow of the movie. For both videos, the audio was identical and the only element that was different between the videos was the synchronization of the imagery to the audio.

4.2 Research

4.2.1 Objective

This study is aimed at finding out whether adjusting timing of in-game events according to the music influences the impact of these events. This leads to the following research question: Does the impact of in-game events depend on whether these game events and a strongly timed orchestral score are well-aligned? And this in turn leads to two hypotheses:

H0: *There is no change in the impact of in-game events when game events and strongly timed orchestral music are well-aligned.*

H1: *There is a change in the impact of in-game events when game events and strongly timed orchestral music are well-aligned.*

4.2.2 Method

Participants

There were 30 participants in total, 15 in each condition. All participants were selected using Accidental Sampling and the majority were Dutch university students. Ages ranges from 19 to 26, with a mean age of 22.3 years. 26 were male (86.7%) and 4 were female (13.3%), both distributed evenly across each condition.

Questionnaire

The questionnaire consisted of four sections. The first section regards general questions about participants opinion about the game, video and music as well as two questions about their impression of the in-game characters. This section will supply some general data about the experience of the participant such as to whether they generally liked the music. This is important because if the music is perceived as bad it can influence the results in a negative way. The second section uses the PANAS scale [23] to measure the difference in influence the two conditions have. The scale is used to, according to Watson et al. [23], measure the two primary dimensions of mood — Positive and Negative Affect. The scale is comprised of 20 items, of which 10 each represent either the positive or negative dimension of mood. This section should represent the effect the two conditions have on the moods of the participant and provide good reliability since the scale has been well validated. The third section deals with questions regarding timing and matching of gameplay of music. Some questions dealt with similar topics but did so in an inverse manner or with a different formulation. This section should show if players noticed the synchronization differences between the two conditions. It can show whether the synchronization had an effect on the participants impression of the in-game characters. The fourth section serves to ask some general demographic questions as well as the participants familiarity with gaming and the game in question.

Conditions

There are two different conditions that will be tested: synchronized and not-synchronized. The difference between the two conditions lies in the videos that will be shown to the participant. The imagery in the synchronized video was synchronized to the music while the not-synchronized video was not. Audio was identical in both conditions.

Data Analysis

The data from the digital questionnaire has been analyzed after all participants had finished the user test. During the analysis the results were tested on significance and reliability and have been described in the results section. Also, all statistical analysis was performed using SPSS 22.

4.2.3 Control and Manipulation Check

On average all participants found both the music and watching the video more interesting than not, with respective means of 3.4 and 3.3 on a 5-points scale. This gives reason to believe that the quality of the video and music did not distract from the user test.

	η	N	σ
The music was...	3.400	30	1.0034
I found watching the video...	3.333	30	.8841

In order to verify that the intended difference in conditions was indeed perceived as such, a so called manipulation check is performed. Between both conditions there was an intended difference with regard to the synchronization of music and events. One video had no events that were synchronized to the music and the other video did have these events synchronized to the music. When looking at the following questions: "The music felt in line with the gameplay","In-game events felt not in line with the music" and "The music underlined big in-game events", it can be seen that creating two different conditions that differ in their alignment of video and audio has been successful. Grouping these questions to a Timing variable ($\alpha = .774$) outputs the following results.

Manipulation Check for Timing Conditions				
	df	F	η	p
Timing	28	1.986	.2.533	.005**
**<0.01				

Now that is has been determined that the two conditions were appropriately differentiated, one can determine whether there is an actual change between the two conditions.

4.2.4 Results

When looking at the PANAS Scale [23], it can be seen that the results seem internally consistent ($\alpha = .825$). Still, independent of what they thought of the videos, the participant's mood was not significantly affected by the different conditions. Other questions regarding for example the perceived power of in-game characters supplied no significant results.

Still, some questions did provide results with statistical significance. The first 5-point scale question is "I found watching this video (Dull — Exciting)". Looking at the results it can be deducted that participants enjoyed watching the synchronized video more than the non-synchronized video.

Influence of Enjoyment				
	df	F	η	p
I found watching this video...	28	.217	.6667	.036*
*<0.05				

The second 5-point scale question is "The music distracted from the video (Disagree Completely — Agree Completely) ". Looking at the table it can be seen that participants found the music in the non-synchronized video more distracting even though the audio of both conditions was identical.

Influence of Focus				
	df	F	η	p
The music distracted from the video	28	.138	-.6667	.039*
*<0.05				

4.2.5 Conclusion

According to the research findings the following can be concluded. Both conditions were appropriately differentiated such that it was apparent to the participants ($\alpha = .774, p = 0.005$). This allows for increased confidence in further findings during the research. Regarding enjoyment, participants enjoyed watching the synchronized video more than the non-synchronized video ($p = 0.036$). Furthermore, participants were more distracted by the music during the video that was not synchronized than during the video that was synchronized.

According to these results, strongly timed orchestral scores have potential for positive influence on impact of in-game events. This positive influence is only apparent when score and game-events are well-aligned because if score and game events are not well-aligned it will cause the music to distract the player (See table Influence of Focus) and it will cause less enjoyment for the player (See table Influence of Enjoyment). If games want to make use of this effect they should be designed with this aspect in mind. It should be possible for in-game events to be aligned to the score by adjusting their timing and thus allowing these game-events to have more impact.

4.2.6 Discussion

The PANAS Scale [23] did not show any significant change of the participants mood between the two conditions. A possible explanation might be that the (mainly Dutch) participants misinterpreted terms and thus provided differing

answers. Another reason might be that some terms are not directly relevant to watching a video and thus cause participants to vary more in their interpretation and answers. Still, these possible explanations are merely food for thought and have no basis from data. Some users mentioned they noticed a lack of sound effects in the video. While it is true that all sound was removed and only music was replaced, the audio was identical in both videos. Still, it is possible that the synchronized music has partially taken over the role of sound effects in providing feedback for certain in-game events.

Chapter 5

Discussion and Future Work

First, a short recap of the conclusions of both sections.

According to the results of the Music Timing user tests, strongly timed orchestral scores have potential for positive influence on the impact of in-game events. But, it is only apparent when score and game-events are well aligned. Otherwise, it might distract the player or cause less enjoyment.

In this proof of concept stage, Procedural Mixing shows decent results. The currently used rules work well to bring variation between segment combination and provide feedback to gameplay situations. The grouping hierarchy works well and gives the composer more freedom in composing.

5.1 Music Timing

The PANAS scale used in the Music Timing research did not show any significant results. This may have been the result of the (mainly) Dutch participants misinterpreting the terms used in the scale. Another reason may be that some terms do not directly apply to watching a video and thus allow for a broader interpretation with regard to the experiment. Some participants noted the lack of sound effects in the video. While audio was identical for both videos, this may have influenced the results of the experiment since the music may have partially taken over the role of sound effects.

The next step in Music Timing would be to implement the system into a game. This requires the game to have events that can be adjusted such that they synchronize to the beat in the music. This usually requires consideration from the beginning of development and is hard to implement in a later stadium. Still, taken into regard during development and implemented correctly it could provide interesting results.

5.2 Procedural Mixing

The main limitation for Procedural Mixing currently lies in the content. More content should be produced to test with the system so that there are enough segment combinations for every possible situation. This should already increase the quality of the output drastically and ensure better suited music for every scenario.

The next step would be to have more rules added to the system. A possible addition can be a section that monitors the frequency spectrum and blocks combinations that sound muddy when multiple segments operate in the same frequency band. Another addition can be a follow-up section. This process monitors the current combination and suggests segments that are follow-up segments of the ones currently playing. These follow-up segments will probably have been determined beforehand and are a continuation of the melody that was started in the previous segment. Last, machine learning can be implemented instead of grouping the segments manually. Ideally, the composer would listen to a wide array of combinations and judge whether or not they fit. After enough iterations the system will have a map of which segments fit together and which don't. Parameter rating could also be implemented into this process.

The next step for Procedural Mixing would be to continue development of the system together with a game development team. This allows for proper testing of the system with gameplay and adjusting for example the parameters in order to be specifically tuned to the game being developed. This will benefit the development of the system a lot and provide the development team with a music system that can provide detailed musical feedback and enhance the overal experience.

Chapter 6

Acknowledgements

First and foremost, I would like to thank my research supervisor, Dennis Reidsma, for aiding me during every step of the realization of this dissertation and for always providing thoughtful feedback where it was needed. I would also like to thank the STEIM institute for providing feedback to my ideas and helping me further define the final subject of this dissertation. For providing me with feedback and pointing me to material I would never have found otherwise I would like to thank Damian Kastbauer. Lastly, I would also like to thank Andy Farnell for providing me with feedback in the earliest stages of my dissertation. While the eventually chosen subject differs greatly, the early feedback still provided invaluable information and constructive criticism.

Bibliography

[1] LibPD. URL https://github.com/libpd/libpd/wiki/unity.

[2] Max Documentation and Resources - Cycling '74 Wiki. URL http://cycling74.com/wiki/index.php?title=Max_Documentation_and_Resources.

[3] Unity - Game Engine. URL http://unity3d.com/.

[4] home of psai — periscope studio audio intelligence. URL http://www.homeofpsai.com/.

[5] Demonstration of SSXs RUMR system — Designing Sound, 2012. URL http://designingsound.org/2012/03/demonstration-of-ssxs-rumr-system/.

[6] Audiokinetic. Audiokinetic - WWise. URL https://www.audiokinetic.com/.

[7] Jan Berg and Johhny Wingstedt. No Title. *ACE '05 Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, (ACE 2005):164–171, 2005.

[8] M. Brinkmann. PureData patches. URL http://www.martin-brinkmann.de/.

[9] Karen Collins. *An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio*. 2007.

[10] Karen Collins. An Introduction to Procedural Music in Video Games. *Contemporary Music Review*, 28(1):5–15, February 2009. ISSN 0749-4467. doi: 10.1080/07494460802663983. URL http://www.tandfonline.com/doi/abs/10.1080/07494460802663983.

[11] Andy Farnell. An introduction to procedural audio and its application in computer games. *Audio Mostly Conference*, (September):1–31, 2007. URL http://www.cs.au.dk/ dsound/DigitalAudio.dir/Papers/proceduralAudio.pdf.

[12] T.M. Fay, S. Selfon, and T.J. Fay. Directx 9 Audio Exposed: Interactive Audio Development. page 373. 2004.

[13] Sloboda J.A. and P.N. Juslin. Psychological Perspectives on Music and Emotion. *Music and Emotion: Theory and Research*, 2001.

[14] Michael O Jewell, Mark S Nixon, and Adam Pr. CBS : A Concept-Based Sequencer for Soundtrack Composition. pages 0–3, 2003.

[15] D. Kastbauer. Crossing the Two-Way Street. *Game Developer Magazine*, page 65, 2012.

[16] J. Murray. Hamlet on the Holodeck. (MA: MIT Press), 1998.

[17] L.J. Paul. The Generative Music and Procedural Sound Design of Sim Cell, 2013. URL `https://www.youtube.com/watch?v=0xr4aL1C24E`.

[18] M.F. Pignatiello, C.J. Camp, and L.A. Rasar. Technique, Musical mood induction: An alternative to the Velten. *Journal of Abnormal Psychology*, 95(3):295–297, 1986.

[19] David Plans and Davide Morelli. Experience-driven procedural music generation for games. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):192–198, 2012.

[20] Miller Puckette. Pure Data PD Community Site. URL `http://puredata.info/`.

[21] Firelight Technologies. FMOD. URL `http://www.fmod.org/`.

[22] Raison Varner. Building Contextualized Systems for the Next Generation, 2013. URL `http://creatingsound.com/2013/12/building-contextualized-systems-for-the-next-gene`

[23] D Watson, L a Clark, and a Tellegen. Development and validation of brief measures of positive and negative affect: the PANAS scales. *Journal of personality and social psychology*, 54(6):1063–1070, 1988.

[24] Rene Wooller, Andrew R Brown, Eduardo Miranda, Rodney Berry, and Joachim Diederich. A framework for comparison of process in algorithmic music systems. (Eno 1996):109–124, 2005.

Audio-Visual References

Borderlands 2 (Gearbox Software, 2012)
Dishonored (Arkane Studios, 2012)
Proteus (Curve Studios, 2013)
SimCell (Strange Loop Games, 2013)
Spore (Maxis, 2008)
SSX (EA Canada, 2012)
The Elder Scrolls V: Skyrim (Bethesda Game Studios, 2013)