University of the Arts Utrecht

Tools for Designing Nonlinear Game Audio Systems

Music and Technology
June 1, 2020

Stijn de Koning

Abstract

As a game audio designer I am hindered during my creative process, due to the discrepancy between the nonlinearity of my work and the linear character of standard audio production software. First, my creative process gets interrupted, due to the necessary switch between creating audio and testing it within its nonlinear system. Secondly, the visual representation of linear composition tools offer an overview of a linear composition, however, such tools are not optimised for providing a visual representation of nonlinear systems.

By researching nonlinear game audio and further investigating the issues that arise, prototypes and mockups for 3 potential solutions have been designed and applied to my work as a game composer and sound designer. The first experiment tackles the issue of the amount of time and effort it takes to switch between software. The second experiment visualises nonlinear sequencing and provides an interface for parameter adaption. The last experiment provides a framework for procedural music and direct audio implementation. A broad approach to designing these tools has been used, to gain perspective on which solutions work well in which situations and how they relate to each other.

Contents

Abstract **2**

1 Introduction **4**

Hindrances in nonlinear audio design 4

About me 4

Tackling the issues 5

2 Nonlinear music **6**

Nonlinear composition techniques 6

Motivations for using nonlinear techniques 7

3 Current hindrances **8**

Testing and prototyping 8

Overview 8

Context 9

4 Approach **10**

Applied projects 11

5 Designing and testing solutions **12**

5.1 Automatically loading files and parameters 12

Quantised nonlinear system 12

File loading structure 13

Application 13

Reflection & future improvements 13

5.2 Systems visualisation and parameter adaption 14

Nonlinear sequencers 14

Progression 14

Systems design 15

Application 15

Reflection & future improvements 16

5.3 Procedural note-based sequencing framework 17

Procedural music 17

Procedural Imagination 17

Systems design 18

Reflection & future improvements 19

6 Conclusion & discussion **20**

References **21**

1 Introduction

With an estimated over 2.7 billion gamers all over the world, video games are a widely used medium (Cough, 2020). However, video games are still relatively new. The first video game 'Pong' was only developed in 1958. The first console 'the Magnavox Odyssey' was released in 1972 and did not have any audio playback capabilities. A substantial part of the development of the foundation of which currently available standard audio composition and production tools are build on, began before video games became a popular medium. These tools have been designed to produce linear audio, using timelines to predetermine the course of the audio before playback. Using linear DAWs[1] to develop audio is a tried and trusted method.

Due to the absence of a predetermined timeline, video games are a nonlinear medium. The input of players cause variations in the way the experience unfolds. The adaptive nature of games has increased the demand and practical need for nonlinear audio. The nonlinearity of video games in relation to the linearity of audio production software, causes hindrances in my workflow, creative process and (interdisciplinary) collaborations and those of other audio developers.

Standard linear audio production tools offer little to none nonlinear sequencing, transitioning, parameter adaption and probability functionalities. Additionally, linear sequencers are not optimised for providing a visual representation of nonlinear systems. Lastly, standard game audio middleware solutions are catered towards audio implementation and do not tackle certain issues that arise in the design stage of a project. This causes testing to take an unnecessary amount of time and obscures communication with collaborators. This hinders the workflow and discourages innovation. By investigating the issues that arise when designing nonlinear audio systems and formulating and testing possible solutions, I hope to tackle these issues.

Hindrances in nonlinear audio design

The issues I come across when designing nonlinear audio, are divisible into 3 main subjects. First, my creative process gets interrupted due to the required back and forth between different tools. Secondly, linear composition tools do not offer a solution for a visual representation of nonlinear systems. Lastly, the nonlinearity and interactivity of video games causes the audios visual and interactive context to be absent during development. Games can not be loaded in a DAW session, as is possible with linear visual media. Even though this last issue should not be overlooked, it will not be investigated as in depth as the other issues, because it relates more to user experience and less to audio development. Nevertheless, because it is intertwined with the other issues, it will still be addressed.

About me

As an audio programmer, composer and sound designer, I work on interdisciplinary visual media projects. I try to combine my skills in these different fields to design innovative solutions and bridge different disciplines. Communication plays a major role in most of my projects. I find it very important to surround myself with motivated people to keep me inspired, receive feedback and learn. As different solutions will be mocked or prototyped and

[1] DAWs or Digital Audio Workstations are software applications used for digital audio production and composition, such as 'Logic Pro', 'Ableton Live' and 'Pro Tools'.

applied to games that I am working on, working on possible solutions applies to my work as an audio programmer as well as to my work as a composer and sound designer.

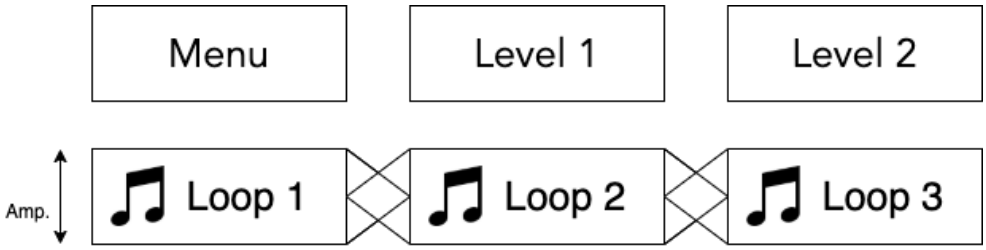
Tackling the issues

I experience hindrances in my creative process as an audio developer for games. By researching these issues and testing different possible solutions I hope to gain insight in how best to address them. The following chapter two will discuss how to define nonlinear audio and what its advantages and disadvantages are. In chapter three I will be analysing the currently arising issues with nonlinear audio design.

Opposite to focussing on a specific issue for a certain type of nonlinear audio, my goal is to test different approaches to various general issues. This serves to analyse which solutions work well in which cases and how they relate to each other. Because of this fairly broad approach, the subject will be contained by mostly focussing on my own process as game audio developer. Furthermore, as most issues arise during the music designing and development process, the focus will be on music. A lot more work has already been done on nonlinear sound design, leading to less issues there. However, there is only a fine line between music and sound design. Sound design and music tools often are similar or even the same, so sound design will still be touched upon in various places. Based on the chapters two and three, a method will be established in chapter four. This method will be applied to design, prototype, mock and test solutions in chapter five. Hereafter, the final chapter will summarise, discuss and reflect on the process and its results.

2 Nonlinear music

Nonlinear audio is a term used to describe audio with a dynamic course that is not predetermined. Based on a set of rules, possibly influenced by chance or other external parameters, the course of the audio will develop. Nonlinear audio is adaptive when it reacts to external parameters or interactive when the audio responds to interaction. A nonlinear audio system describes how and when the music changes and in the case of adaptive or interactive audio, also to what external parameters it responds.



f1. A 'basic' loop-based crossfading nonlinear audio system

Figure 1 illustrates a basic example of a nonlinear game audio system. It depicts a game consisting of a menu and two levels. An assigned audio loop per level crossfades with another loop when transitioning to and from the menu or to another level. So when the player navigates from the 'Menu' to 'Level 1', 'Loop 1' will automatically fade out during which 'Loop 2' will fade in.

Nonlinear composition techniques

Over the years, a substantial amount of nonlinear composition techniques have been developed. Different nonlinear composition techniques serve different purposes and are optimal in different situations. Some techniques can respond to game events very quickly, but restrict the composer while others allow the composer a lot of freedom but restrict game-linking and transitioning capabilities. The optimal balance has often been tried to find by combining different techniques. Investigating the most prevalent nonlinear composition techniques and the different ways they are subdivided, serves to provide a general sense of what techniques are most important to process into the to be designed solutions.

Game audio composer Michael Sweet divides adaptive audio composition techniques into 2 different categories. 'Vertical remixing or layering' describes techniques using different audio layers on top of each other and 'horizontal remixing or looping' describes cues to be played one after the other (Sweet, 2016). Charlie Huguenard, Sound Technology Lead at Meow Wolf, seconds this approach by stating that:

> *'Currently, the most popular methods of composing music for games are the two provided by many audio middleware engines right out of the box: "horizontal" and "vertical" composition.'* (Somberg, 2019)

Another approach is to classify techniques based on their synchronicity with the game. This approach can be used to divide nonlinear composition techniques into 3 different categories (Tol and Huiberts, 2013):

- Trail: Wait for an event in the audio before transitioning or slowly fade the audio. This can be useful for the music, because longer lines can be produced that do not have to be able to instantly transition.
- Sync: Immediately react to game events, this allows for using music as a gameplay element.
- Lead: Change the game state based on the audio, such as with rhythm games.

Motivations for using nonlinear techniques

Nowadays, nonlinear music composition has become a standard approach in game audio. There are a lot of reasons for using nonlinear audio composition techniques for games. Colin Walder, lead programmer for audio and localisation at CD Project Red, states that:

> *'By making a connection between the music and the game action we can effectively tell our audience what to feel'* (Somberg, 2019)

There are 3 main reasons why I use nonlinear techniques to design game audio. First of all, nonlinear systems can be used to generate a lot of material from very little source material. By applying a set of rules on a collection of audio material, a lot of combinations can be generated from a relatively small amount of material. This makes the music less repetitive and reduces the amount of source material that has to be delivered. Furthermore, because less source material is needed, the audio will also take up less disk space. Secondly, having the audio react to other media can help significantly in making the audio part of the world and its interaction. Lastly, it allows for using music as a gameplay element. In Risk of Rain 2, for example, composer Chris Christodoulou composed a soundtrack in which a vertical musical layer is emitted from a certain object in the level, which the player needs to find. When the amplitude of this layer increases, the player knows the object is closer (Koning, 2020).

However, a nonlinear system is not always the right option to go for. Mick Gordon, composer for games such as DOOM and Wolfenstein, explains that dynamic music can get in the way of the flow of a game in a talk about the soundtrack for DOOM (2016):

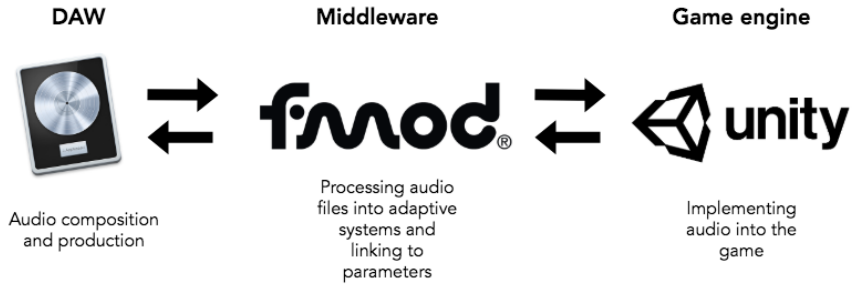
> *'Pretty much every battle in DOOM is a linear path. You enter the arena, the demons come out, your role is to kill all the demons and then you move forward. … We chose not to do too much crazy dynamic music stuff within that structure, because it just seemed to get in the way of the groove.'* (Gordon, 2017)

Furthermore Kubatko, an experienced composer for mobile games, does not use adaptive systems. He states that people do not play mobile games for as long as other games and that music is often turned off or other music is played over the game (Koning, 2020).

3 Current hindrances

Testing and prototyping

Constructing a general concept is a common way to approach the start of a new project. Because of the influence of the nonlinear audio system on the available musical options, it is common to start this concept with an initial design for the nonlinear audio system. If layers need to be played over one another for example, it can be useful for them to be in the same key or have the same tempo. This systems concept is usually based on reference, previous experiences and general knowledge. While these are important points to base an audio system on, a key influence that is missing is the ability to try out various options. Testing different possibilities currently takes an unnecessary amount of time and effort. To test an audio system properly, I usually have to go through 3 different software programmes (f2). Because there is no quick way to try out different systems, I experience a hinderance in my creativity and am obstructed in trying out innovative approaches.

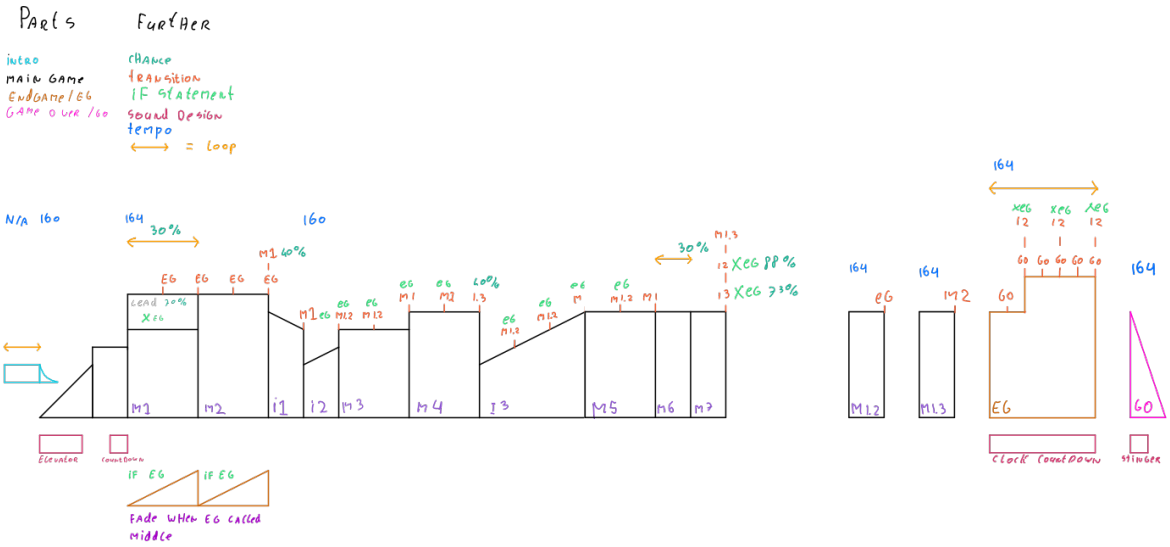


f2. Switching between programs in my standard nonlinear audio

Overview

Another issue that hinders creativity and originality, is the absence of an easily accessible and clear overview of the audio system. Even though middleware can save a lot of time in implementing audio, they often do not provide a clear schematic overview of the entire system. Because of this I often choose to draw out an overview by hand. Figure 3 depicts a scheme I drew to provide an overview of the nonlinear system for a recent game that I worked on, called Evasion[2]. Doing this takes a lot of time, the scheme can not quickly be adapted and can still be quite unclear, especially for developers in other disciplines. Inherently humans are taught music as a linear occurrence that happens over time. When showcasing nonlinear audio, it can often subconsciously be experienced as a linear track, as if the current course is decided beforehand, such as with linear music. A visualisation could help showcasing game audio in its nonlinear form and in how it differs from linear music.

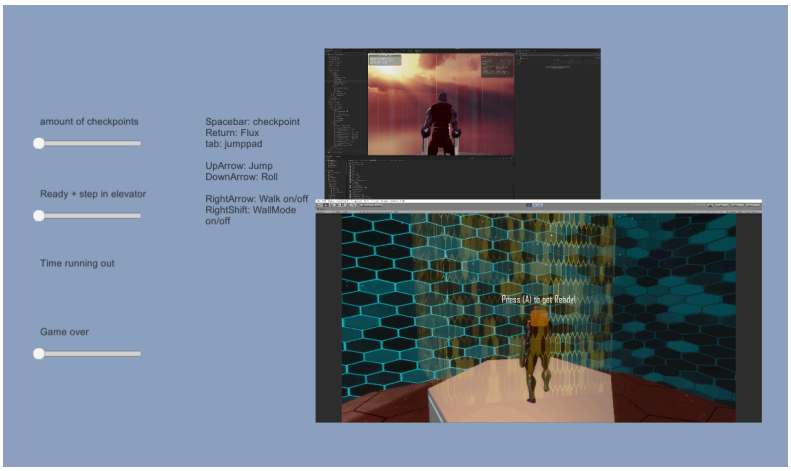
[2] sdkoning.com/PF/Evasion.html



f3. Drawing of a nonlinear music track for Evasion

Context

When composing for linear visual media, the image can often be loaded into the DAW session. This serves to directly link the different media, but also to have the context for the audio constantly and directly available. For games, it often is not until the audio is implemented in the game, that the interactive and visual context is available. Because this takes a lot of time, it hinders the ability to design the right system and showcase the audio in its eventual context.



f4. Screenshot of Unity project to test audio for Evasion

Figure 4 shows a screenshot of a sample project I created in Unity in the early stages of the development of Evasion. It shows sliders and key input to change parameters for interactive context and loaded videos for visual context. Creating this project took a lot of time, majorly interrupted my workflow and still provided limited context as gameplay interaction would have taken too to prototype.

4 Approach

Defining base lines for the process of designing and testing possible solutions, serves to help to allow for the ability to test multiple possible solutions. An approach that favours quick testing and prototyping will be used. This should provide a general sense of what possible solutions work well and what do not, before possibly going more in depth into specific elements in the future. Furthermore, this broad approach will serve to provide insight in how different solutions perform in relation to each other.

Solutions will either be prototyped or mocked. Hereby, a base for every tool will be established that can possibly be built upon in the future. Before designing every specific solution, the goal for the tool, the problem to address and how this will be addressed will be stated. Once a solution has been sufficiently tested in regard to the originally stated goals, the most prominent future improvements will be listed, before moving on to the next experiment.

The main disadvantage of this broad approach is that only the surface of possibilities will be addressed. Furthermore, mocks and prototypes can indicate if certain solutions could possibly work, but are unable to fully test the solution. Another risk to keep in mind, is that in the tools fixing a certain problem, they could cause new issues of their own. This is why functionalities should be designed and tested in a pragmatic fashion. Every feature should be sufficiently tested to make sure its advantages outweigh the disadvantages it possibly creates.

To function as a proof of concept on how to tackle the issues, the tools need to be flexible and should not be specific to any one game or phase in game audio. They need to improve the general nonlinear systems design process.

> *'.. the game industry is undergoing constant change .. . The only solution to this unpredictable situation is granularity—that is, the goal to build a flexible set of tools that are constantly able to update, evolve and adapt'* - Florian Füsslin, Audio Director at Crytek (Somberg, 2019)

To be able to try different approaches and focus on the solutions, it is important not to do work that has already been done and use as many existing tools as possible.

> *'Although it's fun and rewarding to create something from scratch, it's not always the best thing to do for your tools. It's important to evaluate existing tools and applications aside from the custom approach to find the best value.'* - Guy Somberg, Lead Programmer at Echtra (Somberg, 2017)

The created solutions should be easy to expand. Because this is the first step to be built upon in the future and because the game industry is undergoing constant change, components should be easy to add and adapt.

Code, technical documentation, videos of the tools applied to games and more specifics about the solutions are all published online on my Github page[3] and my website[4]. These sources provide more in depth information on the solutions themselves.

[3] https://github.com/StijndeK/VASD

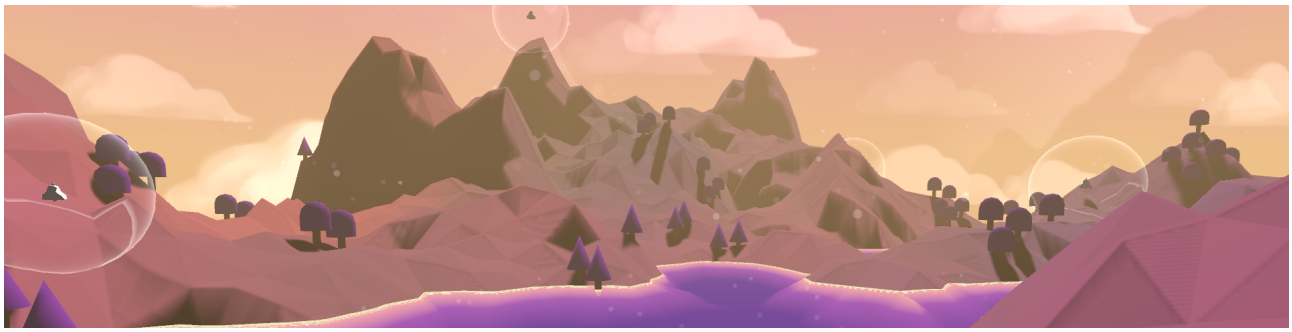
[4] http://sdkoning.com/PF/N.A.D.T..html

Applied projects

Experiments will be applied to games I am working on as composer, sound designer and audio programmer in conjunction with the development of the tools. The priority of which solutions to try out, is partly based on what these projects require. Because of this and to provide insight on how the solutions are tested, a brief description of the projects is given.

f5. Screenshot of Project Rookery

Project Rookery[5] is a walking simulator game, set in the Victorian era in London. As an extremely poor woman living in the Rooks, you journey to the richer parts of the city, during which you experience worsening hallucinations caused by your delirium. The focus for this game is on its world and story. Because of this, the the audio focuses mostly lies on the storytelling and world-building.



f6. Concept art for Procedural Imagination

Procedural Imagination is a procedurally generated game world that dynamically responds to the player's actions. This game's procedural nature allows for researching procedural music.

[5] sdkoning.com/PF/Rookery.html

5 Designing and testing solutions

By mocking and prototyping various tools to address issues caused by the nonlinearity of video games in relation to the linearity of audio production software, I hope to improve the process of designing nonlinear audio systems. This chapter discusses three experiments in the form of prototypes and mockups that have been done. The first experiment addresses wasted time when switching between software, the second tests a dynamic visual approach and the last tests a procedural approach to nonlinear audio design and implementation.

5.1 Automatically loading files and parameters

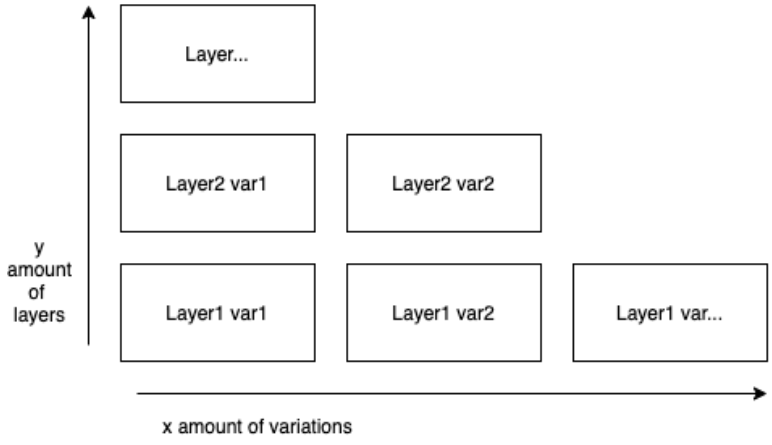
The first experiment is a Python console application that allows for quickly testing quantised nonlinear audio. The project tackles the disturbance caused by the unnecessary amounts of time spent from creating audio to being able to hear and test it within its nonlinear system (and eventually within the context of the game). The solution investigated is to export audio directly into specific folders using specific file and folder naming. This way the program will automatically sequence and layer the audio based on the obtained data from the naming.

Quantised nonlinear system

Because of the broad approach described in chapter four, I decided to test this potential solution using a more basic nonlinear technique named layered quantised synchronisation. I opted to start with this technique, for the same reasons stated by Colin Walder here:

> *'For basic synchronization, I start with bars, beats and grid since it is trivial to prepare the interactive music to support these, and because they are the most straightforward to understand in terms of mapping to gameplay.'* (Somberg, 2019)

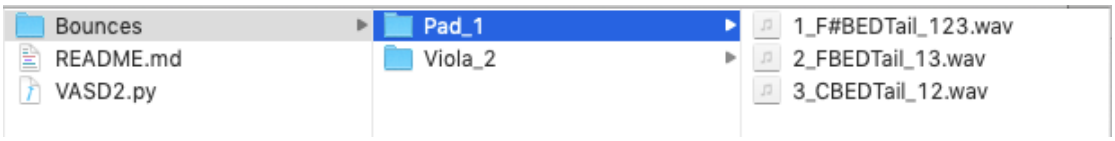
The quantised system to be tested with this tool, is based on layers containing different variations (f7). Audio is sequenced on a given bpm and loop length. For every layer a variation is chosen and played. The data obtained from a filename indicates to what variations a variation within a layer can transition. This means that variations withins layers transition independent of the other layers. Because of this, every variation of every layer needs to be able to loop over the other variations.



f7. Layer and variation structure

File loading structure

To add a layer, a folder needs to be created in the 'Bounces' folder. The title of every folder should end with an underscore followed by the number of the layer. Within every folder, audio files containing variation-loops can be put. A variation-loop's filename starts with the number of the variation, followed by its title, followed by the numbers of variations it can transition to, all separated by underscores (f8).



f8. File loading structure example

For example, the first layer in figure 8 is called Pad and contains 3 variations. The first variation F#BEDTail_123 can transition to itself, variation 2 and variation 3. An infinite amount of layers can be added with an infinite amount of variations per layer. If a layer should not always be audible, an empty audiofile can be put in as one of the variation loops.

Application

After the audio is loaded in and data is obtained from the folder and file names, the application asks for the bpm and length per loop in beats, to then start playing by choosing a random variation per layer (f9).

```
set bpm: 97
set loop length in beats: 16
Type 'p' to play or 'q' to quit: p
Now playing
Press return to stop audio
playing: ./Bounces/Viola_2/new_1_VIOLAT_23.wav
playing: ./Bounces/Pad_1/new_1_F#BEDTail_123.wav
playing: ./Bounces/Viola_2/new_2_VIOLAT_13.wav
playing: ./Bounces/Pad_1/new_1_F#BEDTail_123.wav
playing: ./Bounces/Viola_2/new_1_VIOLAT_23.wav
playing: ./Bounces/Pad_1/new_3_CBEDTail_12.wav
```

f9. Screenshot of the application

Reflection & future improvements

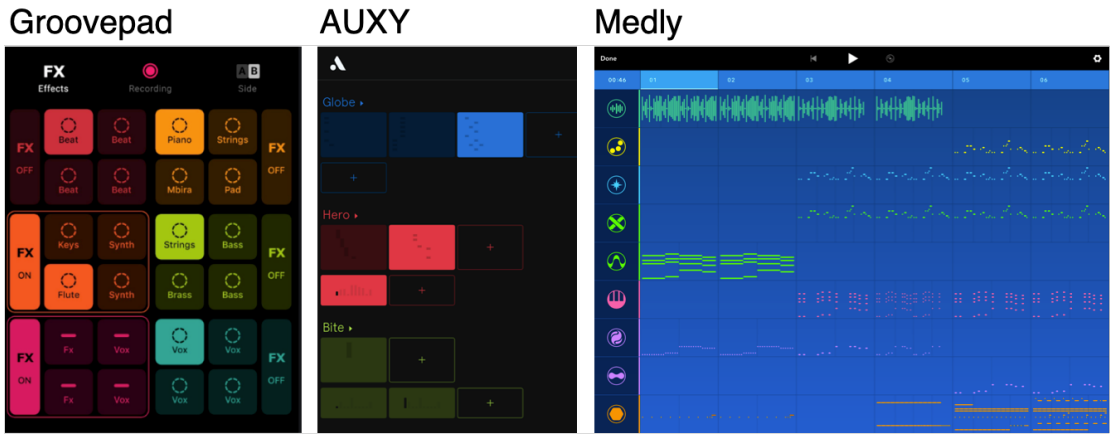
Quick testing, by directly bouncing the audio with a specific naming convention, saved time and allowed me to stay in the flow while designing the basis of the music system of Project Rookery. However, the options are still very limited and files do need to be named very specifically. User input could be completely avoided by obtaining bpm and the amount of beats from the folder or file titles as well. This system is very simple and can only be used in testing a specific type of nonlinear sequencing, however it has achieved its goal in testing this type of approach.

5.2 Systems visualisation and parameter adaption

As I started experimenting with more intricate nonlinear systems for Project Rookery, the next issue that became prevalent was the lack of provided overview. Both to quickly try out ideas and to communicate about them, I required a better visual overview of the nonlinear system. Furthermore, the game required audio to be adaptive and not loop in the same way forever, as in the previous experiment. A form of parameter adaption was required. These 2 points became the goals for the second experiment. This experiment tests a potential solution that uses a visual overview and interface along with the automatic file loading functionality carried over from the previous experiment, to look into upgrading the process of prototyping nonlinear systems and improving (interdisciplinary) collaboration.

Nonlinear sequencers

For inspiration on how to approach visualising adaptive nonlinear sequences, I analysed a collection of mobile music creation applications (f10). Because of the lack of user input capabilities and relatively small screens of mobile devices, sequencing as with standard DAWs can be cumbersome. As a solution to this problem, some mobile music creation applications implemented nonlinear sequencers. Figure 10 shows screenshots of a few mobile applications that use different approaches to nonlinear sequencing. Groovepad divides various vertical layers in coloured squares. Within every layer there are 4 different horizontal loops to choose between. Tapping a loop turns it on or off and tapping another loop within the layer transitions to that one on a new measure. The second mobile application, AUXY, allows the user to choose multiple instruments and create different midi loops per layer, to be activated or cycled between by tapping on them. On top of this, it also provides automation loops per instrument, which can be activated and cycled between in the same way as the midi loops. Lastly, Medly looks like a more standard DAW, with a linear looking timeline. However, the timeline actually consists of a list of loops, that can be played in any sequence.

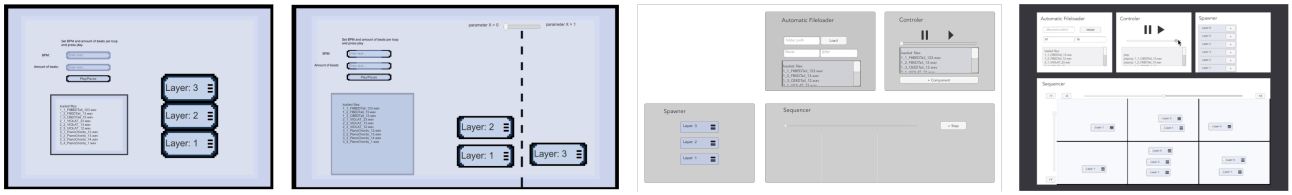


f10. Mobile music creation applications that apply forms of nonlinear sequencing.

Progression

Because this tool required more functionalities than the previous experiment, it went through more iterations (f11). By adding and reflecting on features on a step by step basis, I tried to keep track of potential drawbacks. The first (left) prototype simply loads in files as in the

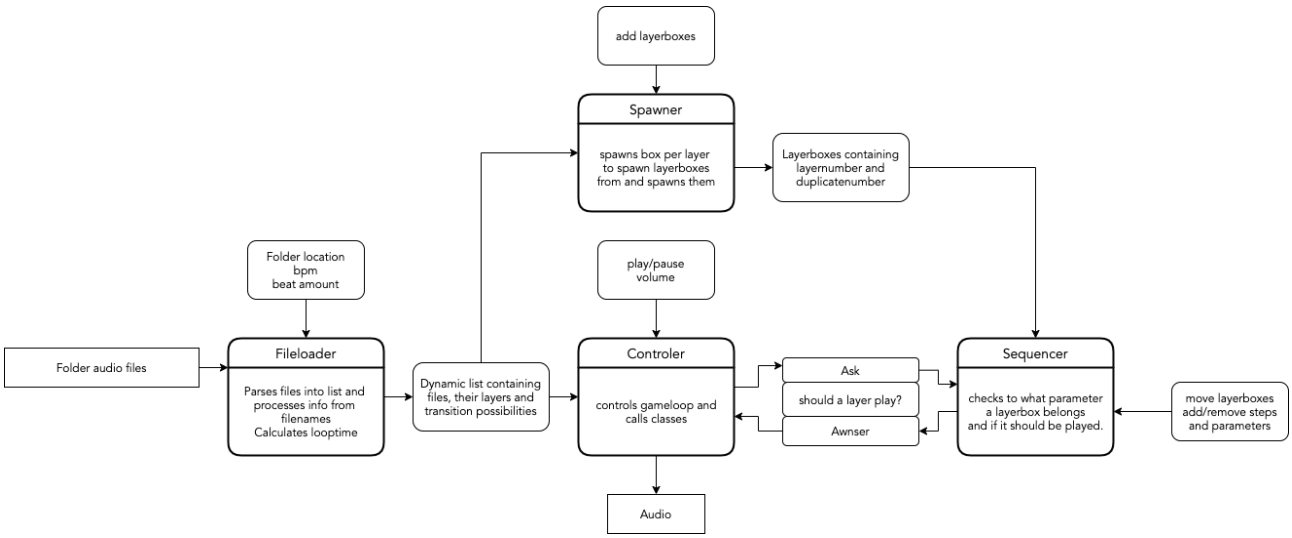
previous experiment to visualise them in boxes. From here on I added features such as parameter linking and spawning of copies of layers.



f11. Timeline of various versions of the prototype

Systems design

The latest prototype consists of 4 main components (f12). The Fileloader loads audiofiles and parses information obtained from their names. The Spawner allows the user to spawn layerboxes containing horizontal loops with variations, that can then be dragged into the Sequencer. The Sequencer checks if and what sound needs to be played. The Controller eventually plays the audio back.

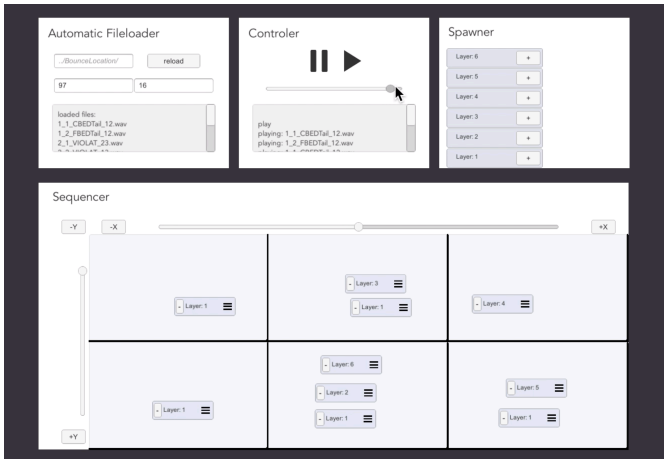


f12. Dataflow of the prototype

Application

Figure 13 is a screenshot of a video[6] that showcases the latest prototype used to test an audio system on Project Rookery. Layerboxes can be spawned from the spawner. A Layerbox holds a layer with all its variations. When dragged into the sequencer the Layerbox is given a X and Y parameter. These parameters can be set to certain values by moving the X and Y sliders. The transitions between variations within layers are not visualised (yet), but behave in the same way as the previous experiment, where the numbers at the end of a filename indicate to what variations a variation can transition.

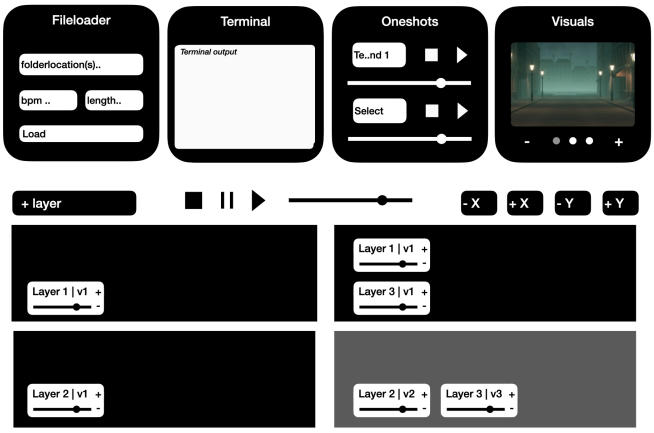
[6] http://sdkoning.com/Media/VPA_Prototype.mp4



f13. Screenshot of the latest prototype

Reflection & future improvements

As the initial goals for the prototype had been reached and I was able to use it to quickly test and showcase the audio for Project Rookery, I created a mockup video[7] to showcase some possible future improvements (f14). The mockup shows how components can be loaded in, based on what is useful or required for a specific project. As visual context is an important part of game audio, a component to load in visual context has been added. The Layerboxes now show what variation of the layer is being played. They now also contain a volume slider. As oneshot sound effects and stingers are another important context to test in looping audio, a component to play single sound effects has also been added.



f14. Screenshot of the mockup

Applying a visual approach allowed me to test different systems and their audio. Furthermore, it allowed me to express my ideas better to the Project Rookery team. However, the project only scratches the surface of nonlinear game audio possibilities and its current version conditions the user quite heavily into using certain techniques. Aside from features shown in the mockup, there are a lot of future improvements to be made such as side chaining, DSP effects, interactive context, other transitioning techniques and midi sequencing.

[7] http://sdkoning.com/Media/VPA_Mockup.mp4

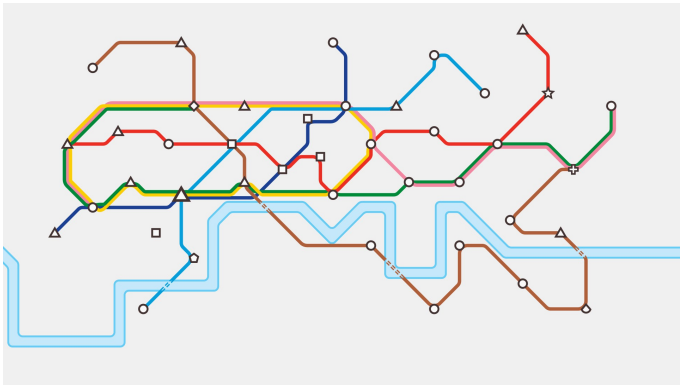
5.3 Procedural note-based sequencing framework

The final experiment is a prototype for a framework designed to improve procedural music systems prototyping and implementing in games. Even though procedural sound design has become somewhat standard in game audio, I find procedural music is still relatively underrepresented. It can be difficult, time consuming and expensive to prototype, test and implement. Because of this, it often is a risk to even attempt procedural music for a game.

The prototype allows for quick testing and implementing procedural music, using a sample-based sequencer. The automatic file loading functionality has again been carried over from the first experiment. The prototype serves to directly implement the audio within its system in the game as well. By having the audio implemented into the game directly, all the context is immediately available as well. Furthermore, automatically loading audio and implementing it, removes the need for middleware. To be able to focus on implementation, the project shies away from a visual approach. This does however lead to the programmer also having to be the composer and vice versa. The project has a larger emphasis on not conditioning the user too much, which was one of the biggest drawbacks of the previous tool.

Procedural music

Procedural game music is music that is generated in real time to data from the game. This can be done in various ways such as by generating notes to play or by synthesising audio live. The main advantages of procedural music are that the music can quickly adapt to the game and that by generating the music based on game data, it is directly linked to the game. A disadvantage is that linking the music and game too heavily, can get in the way of the flow as previously stated in chapter two. A primary example of procedural music being successfully used is the game: Mini Metro (f15). Composed for by Richard 'Disasterpeace' Vreeland. In Mini Metro the player needs to create and manage metro lines. The music is directly generated based on these lines and the trains riding them (Vreeland, 2018).

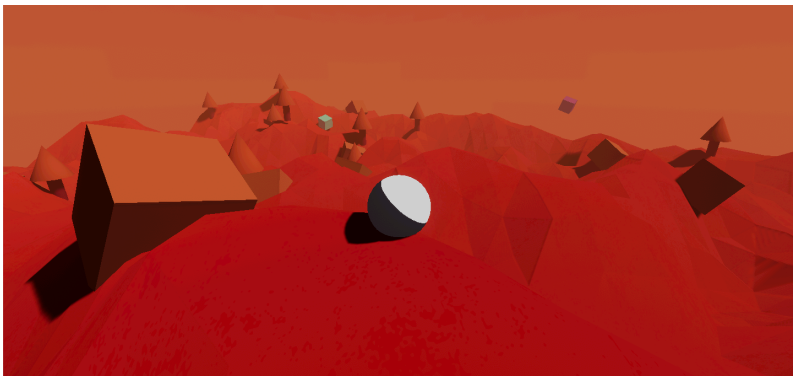


f15. Screenshot of Mini Metro

Procedural Imagination

The experiment has been used as an audio engine for Procedural Imagination (f16). In Procedural Imagination, the player can move around freely through a generated environment. Throughout the environment floating squares are generated. By moving through such a square, the environment regenerates based on the position and color of the square. Using

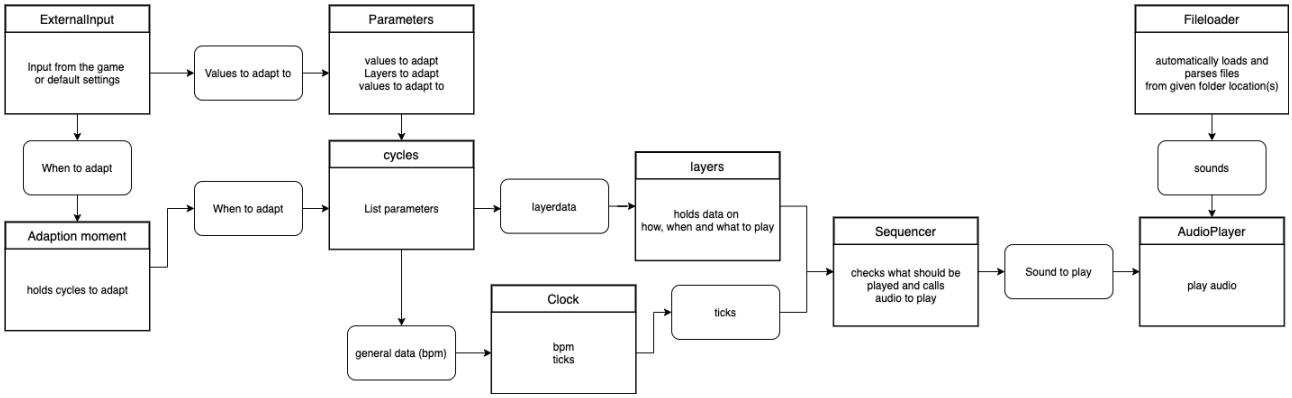
this prototype, the audio regenerates on these points as well, based on the same parameters as the visual environment[8].



f16. Screenshot of Procedural Imagination

Systems design

Game data is used to trigger and generate Cycles. These Cycles set audio parameters for Layers. Layers hold audiofiles and data on when and how to play them. The Sequencer receives ticks from the Clock and checks variables within Layers to trigger audio.



f17. Dataflow of the project

Layers

The loaded audio is divided into vertical layers, to be played independently or over each other. A layer can consist of vertical loops or oneshot sound effects. A layer holds data on how, when and what variation to play.

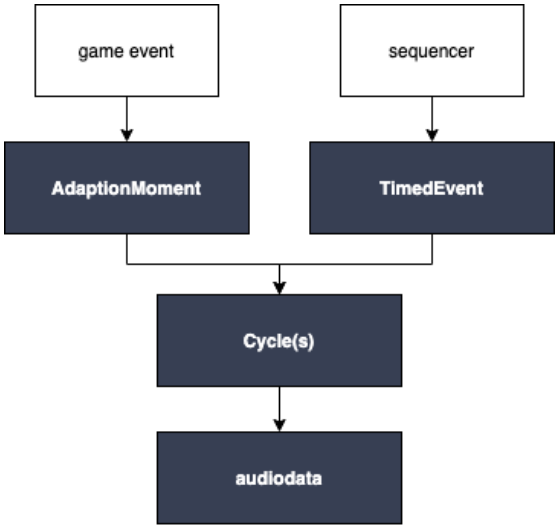
Musical values

Layers contain 'musical values', among which a rhythm and a tonal layer. These rhythms and melodies are used in the Sequencer to check when and what file to play. The rhythm calls on what ticks audio needs to be played. The tonal layer decides what notes need to be played, based on the layer type (melody, countermelody, chord, percussion, etc) and the available samples within the layer.

[8] sdkoning.com/Media/PI_Screencap.mp4

Cycles

A Cycle holds parameters, that hold values to adjust, values to adjust to and what layers to adjust. Using a Cycle is the only way to adapt audio data. Cycles can be triggered from game events using an AdaptionMoment or from the sequencer, using a CycleTimer. A DynamicCycle only sets the on/off value of layers. By using a CycleTimer the DynamicCycle can create a natural sounding buildup. For now this buildup is mostly random. In the future specific dynamic structures could be given and/or influenced by external parameters. This system is build so that any one parameter or game event can easily be linked to multiple aspects of audiodata. This prevents conditioning the user into using certain systems.



f18. Trigger dataflow

Reflection & future improvements

This tool saved a lot of time by directly loading audio and implementing it in the game. Furthermore, it allowed for a procedural approach that would fit the game. When certain aspects of the game changed during development, it was quite easy to adapt the music system to it as well.

Because of the focus on quick prototyping, a lot of interesting features have been overlooked. Even though a lot of audio parameters can be linked to game data, not every parameter is adaptable. Furthermore, the composition algorithms currently are very simple. A lot of interesting algorithms could be added, or even a feature to design them easily in an interface. Furthermore, some features that could be added in the future are: a visual interface, DSP effects and live synthesis.

6 Conclusion & discussion

Designing and applying the 3 experiments has served to upgrade my workflow. Also it has formed a basis for further development of tools to improve the workflow. Automatic file loading saved a lot of time relatively easily, which is also why I used it on both other experiments. However, when expanding this feature with more nonlinear composition techniques, the naming conventions might become to complicated. A balance should be found between what data can be obtained from folder and file names and what data should be obtained manually. Visual sequencing improved the workflow by allowing me to immediately test systems and different combinations of audio layers. When expanding on this feature it will be important to keep prioritising the quick workflow. Furthermore, it could work well to have this tool automatically implement the audio as well, as with the third tool. With the procedural tool a basic framework to build procedural systems has been developed. By first automatically loading the audio and then directly implementing it and thereby removing the need for middleware, this tool improved my workflow substantially. Because of all the seemingly endless possibilities of procedural audio, a lot can be done to further expand this tool. However, it might be counterintuitive to make every possibility available. Providing too many options might get in the way of the workflow.

The broad approach that favours prototyping and trying out different approaches has served to gain insight in what possible solutions work well and in what situations. The file loading turned out to be an easily applicable and useful feature. The data visualisation substantially improved collaboration and allowed for quicker testing and more experimentation. Direct implementation also saved a lot of time and gave me a lot of room for experimentation. However, this broad approach did result in all solutions being very general prototypes, testing a specific and small amount of all nonlinear composition techniques. New issues could arise when further developing these tools.

The prototypes and mockup form a basis for further developments. By building upon the obtained results, more efficiency and room for experimentation can be gained. Developing the prototypes has provided insight in the issues and how to tackle them. Furthermore, they have already improved my creative process and interdisciplinary collaboration substantially.

References

Gough, C. (2020) Number of video gamers worldwide in 2020, by region [Online]. Available at https://www.statista.com/statistics/293304/number-video-gamers/ (Accessed: May 2020)

Gordon, M. (2017) DOOM: Behind the Music. Available at https://www.youtube.com/watch?v=U4FNBMZsqrY&feature=youtu.be&t=2982 (Accessed: February 2020)

Koning, S. de (2019) Composing for Games as a Film Composer. Available at https://sdkoning.com/PF/ComposingforGamesasaFilmComposer.html

(Accessed: May 2020)

Sweet, M. (2016) Top 6 Adaptive Music Techniques in Games – Pros and Cons [Online]. Available at https://www.designingmusicnow.com/2016/06/13/advantages-disadvantages-common-interactive-music-techniques-used-video-games/ (Accessed: April 2020)

Somberg, G. (eds.) (2017) Game Audio Programming: Principles and Particles. Boca Raton: Taylor & Francis Group

Somberg, G. (eds.) (2019) Game Audio Programming 2: Principles and Particles. Boca Raton: Taylor & Francis Group

Tol, R. van and Huiberts, S. (2013) The Game Pulse - Timing Game Events and Music Events. Hilversum: HKU University of the Arts Utrecht

Vreeland, R. (2018) Serialism & Sonification in Mini Metro. Available at https://www.youtube.com/watch?v=FgV4hSfsl00&t=1s (Accessed: June 2020)

sdkoning.com/PF/ComposingforGamesasaFilmComposer.html