THE UNIVERSITY *of* EDINBURGH
Edinburgh College of Art

Music Technology Project Dissertation

An Examination of the Current Standards of Adaptive and Generative Music in Video Games

Nani Porenta

BMus in Music Technology

Supervisor: Dr Tom Mudd
May 2021
Word count: 3,885

Abstract

This dissertation examines various approaches to adaptive and generative music within video games. It will aim to explain how different methods can be implemented and how their performances compare. These methods were mainly implemented with either Audiokinetic's audio middleware Wwise or open-source visual programming language Pure Data (Pd) in a video game created in Unity specifically for this project, titled "Inside The Walls". Each scene and its musical source were categorized in a typology for adaptive and generative music. The dissertation concludes that established industry game companies should participate in new developments and research generative tools in video game music more actively to allow their integration in audio middleware such as Wwise as standardized tools. Education on dynamic music should also be encouraged to establish a stronger bond between industry and academia.

Declaration

I do hereby declare that this dissertation was composed by myself and that the work described within is my own, except where explicitly stated otherwise.

Nani Porenta

May 2021

Acknowledgments

I wish to gratefully acknowledge the support that I've been given by so many people over these past four years.

I would like to thank my supervisor, Dr Tom Mudd, for his advice and help throughout this entire project as well as during numerous other courses that I've had the pleasure of participating in.

Additionally, I would like to express my gratitude towards the members of staff at the Reid School of Music that have always been there to support me during my studies, and always replied to the occasional 11:30 pm email in a split-second.

I want to thank the many people in my life that have shown me so much love and have been shoulders to lean on during my undergraduate years. But some deserve a special mention.

The friends I have made through the Music Tech course, who I have spent so many long nights in the micro lab of Alison with, desperately trying to finish assignments the night before they were due. My FilmSoc pals, who next to being great friends have been equally great for grammar checks (thanks Hannah). My flatmates, for providing the late-night tea and food deliveries.

And last but not least, my family. Who've had to endure listening to my early endeavours into music. And who, although we've now nearly been apart for a whole year, have always shown me love and support from afar and encouraged me to follow my passion - all the way to Scotland.

Table of Contents

1. Introduction

Video game audio has changed greatly since the first at-home games and consoles were introduced. In general, game audio before the 1990s was generated by sound chips with a highly synthetic 8-bit sound [1, p. 20]. While the adoption of MIDI into video games was one of the biggest advancements for video game sound in the 1980s, it was the shift to pre-recorded human performances in the 1990s that changed the music of games drastically [1, p. 22]. Largescale orchestral scores of the same scope as big Hollywood blockbusters have now become the standard of music in industry games.

But as opposed to popular linear audio-visual media like film or TV, video games fall within the category of non-linear media. The progression of a game, and therefore its music, is unpredictable and changes at every playthrough [2, p. 277]. Music for such a medium needs to be able to adapt accordingly.

More recently, there has been an increased interest in academia in exploring the advantages of generative music tools in video games. But as video game music was originally rooted in real time synthesis, generative music in video games isn't actually new [3, p. 298]. Nevertheless, with the change to orchestral scores, generative music now needs to be able to compete with the quality of recorded audio that audiences have grown accustomed to [4, p. 101].

For the purpose of exploring such techniques in a video game setting, a game - entitled "Inside The Walls" - has been developed to implement various game music methods. It is important to mention that this game, in line with the extent of this project

and dissertation, is naturally limited in scope. This dissertation will then discuss how the methods compare and categorize them in a typology of generative, adaptive music.

2. Categorisations of dynamic game music

Game sound and music needs to be dynamic in line with a game's general non-linear character. Dynamic music can be defined as music that "is able to flexibly and smoothly adapt to the game state and interact with the player's in-game actions" [5, p. 2]. But just as video games are still a relatively new medium, so too is the field of game music scholarship [6, p. 2]. Certain sub-terms of dynamic music can be defined.

Karen Collins identifies two categories of dynamic audio: interactive audio and adaptive audio. While interactive audio refers to sound occurring in reaction to the gameplay directly triggered by the player, adaptive audio does not directly respond to the user but reacts to changes in the game environment [2, p. 265].

Both adaptive and interactive music can use two different types of algorithms to shape their output. Transformational algorithms "act upon an already prepared structure (audio clips, MIDI files, etc.)", while generative algorithms "create the musical structure themselves" [7, p. 2]. Collins' definition of procedural (generative) music as "composition that evolves in real time according to a specific set of rules or control logics" [8, p. 13] compliments this characterization of generative algorithmic music.

Andy Farnell, conversely, classes game audio based on its sound source into procedural audio and sample-based (or "data model") audio. He emphasises the potential of procedural audio for SFX, but states that procedural audio for music is merely "second

tier" to fully composed pieces. This establishment of a dichotomy between procedural and sample-based music, however, is outdated. Real-time generation of music (procedural music) does not equal real-time synthesis of music and doesn't imply the exclusion of sample-based audio. Such generative music at today's standards is certainly not "second tier".

2.1. Definitions by Plut and Pasquier

In their 2020 paper "Generative music in video games: State of the art, challenges, and prospects", Cale Plut and Philippe Pasquier provide a comprehensive definition and categorisation for adaptive and generative music in video games. They define adaptive and generative music as the "two primary techniques to extend linear, composed music in games" [9, p. 2].

They describe adaptive music as music that "reacts to a game's state" and can "provide large amounts of unique music from limited musical content" [9, p. 2]. Generative music, on the other hand, "addresses the creation of musical content itself", is "created via systemic automation", and is "sometimes called procedural music, musical metacreation, or algorithmic music" [9, p. 2]. A game's music is truly generative "if the music is produced by a systemic automation that is partially or completely independent of the gameplay" [9, p. 2]. But such independence "can have a large range of possibilities" and a "highly adaptive generative music system may use a large array of game variables to inform the generation of musical content" [9, p. 2]. Generative music, therefore, provides exceptional opportunities regarding variability: capable of producing endless newly generated game music.

The paper then proceeds to provide a detailed typology of generative music which includes three main groups of dimensions: musical, gameplay and architectural. This typology will be used to categorise the musical approaches of "Inside The Walls".

3. The Game - "Inside The Walls"

The game developed for this project is based on the "Choose-Your-Own-Adventure" principle. The player moves from scene to scene and thereby from one approach to game music to another.

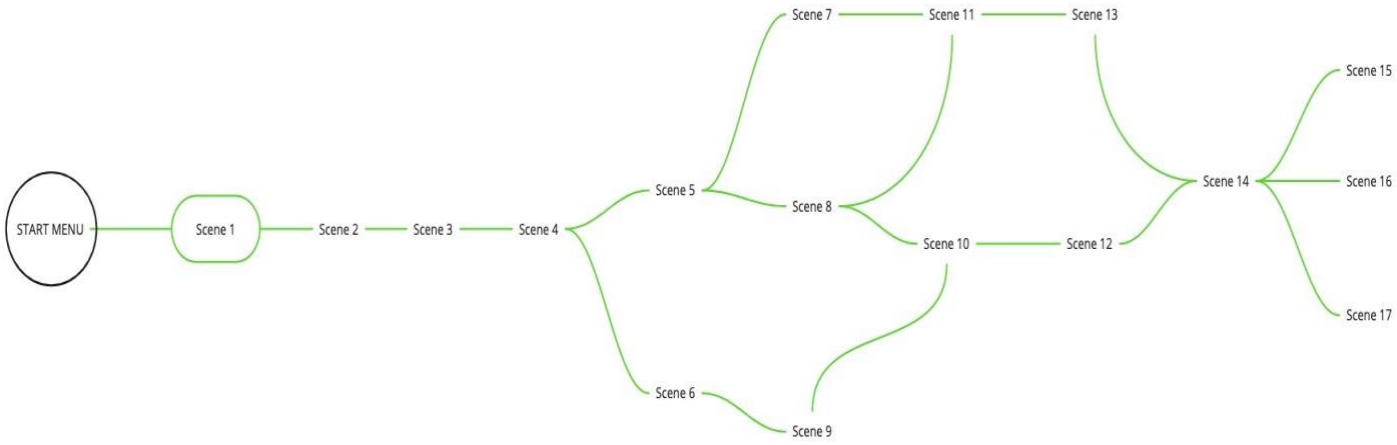

Figure 1 - Possible game paths in "Inside The Walls"

"Inside The Walls" seeks to compare the application of these adaptive and generative processes between the audio middleware Wwise, which is widely used in the video game industry, and the open-source visual programming language Pd, which is more commonly used in academia. The game was developed in game engine Unity. For Pd's integration inside Unity, libpd was used - an embeddable library wrapper for Pd, written in C [10, p. 44].

For details regarding the musical structures and programming of each scene, please refer to the diagrams provided as well as the additional material (see Appendix).

3.1. Start menu and final scenes (15, 16 & 17)

The start menu's musical background is a singular loop being repeated in Wwise. There is no direct player interaction in this instance except when the player presses a button, triggering the "click" sound used for all the choice buttons throughout the game. The final scenes (no. 15, 16 & 17) are structured in the same way.

3.2. Interactive music hierarchy in Wwise - Scene 1

The first scene displays some of the basic functions of Wwise's interactive music hierarchy. The different music segments are being selected based on which state is currently active. As the player progresses through the scene, they encounter triggers that change the current state. Sections 3 and 4 both contain a separate musical segment for a synth sound, the volume of which is being adjusted according to where the player is positioned on the game's x-axis (see "Scene_1_Video.mp4" at 0:27).

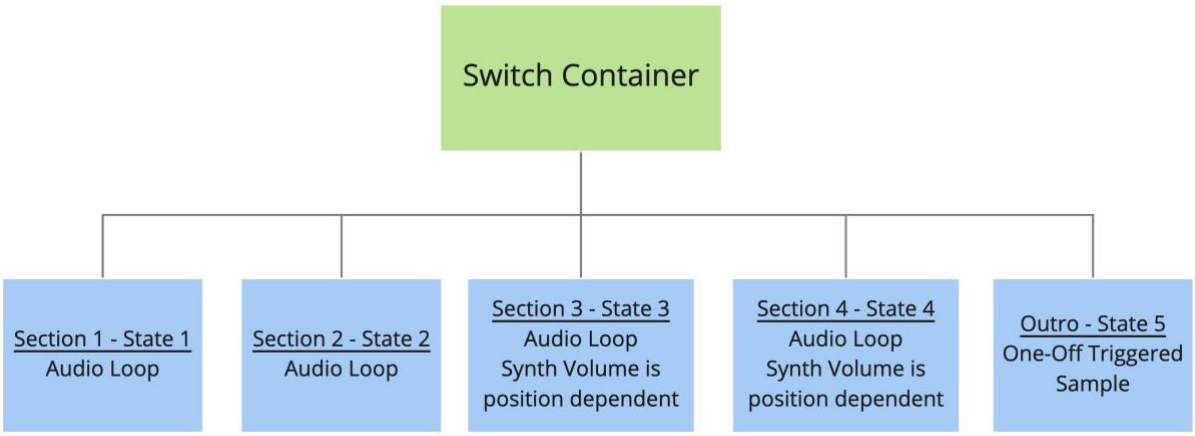

Figure 2 - Musical structures of Scene 1 (Wwise - Interactive music hierarchy)

These two approaches, the progression from one section to another based on some cues and the changing of the output mix, are referred to as horizontal re-sequencing and vertical re-orchestration - two common practices in adaptive music [11, p. 145].

3.3. Interactive music in Pd - Scenes 2 & 3

In Pd, changing the music's scales and progressions can be done within the flow of the music without needing to wait for a transition point such as the end of a phrase. In scene 2, the player enters trigger containers within the room that change the notes inside an array in Pd that the synthesiser reads the notes from. These notes are then transposed depending on how far away from the middle the player is positioned.

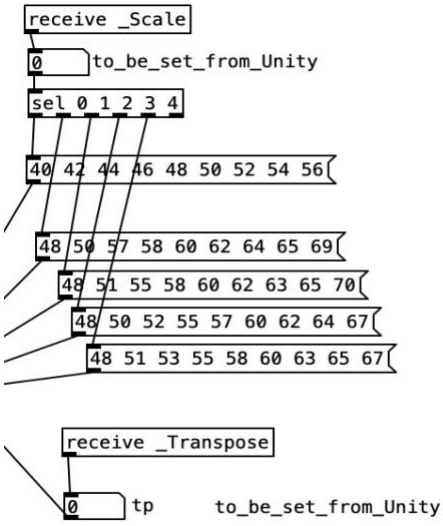

Figure 3 - Possible scales inside the Pd patch for Scene 2

In scene 3, the "Groovebox" patch by Martin Brinkman creates a large number of various melodies and rhythms from scratch. This scene can also be taken as an extreme example of interactive music, where the user is directly influencing the music by changing the groove and instrumentation at the click of a button.

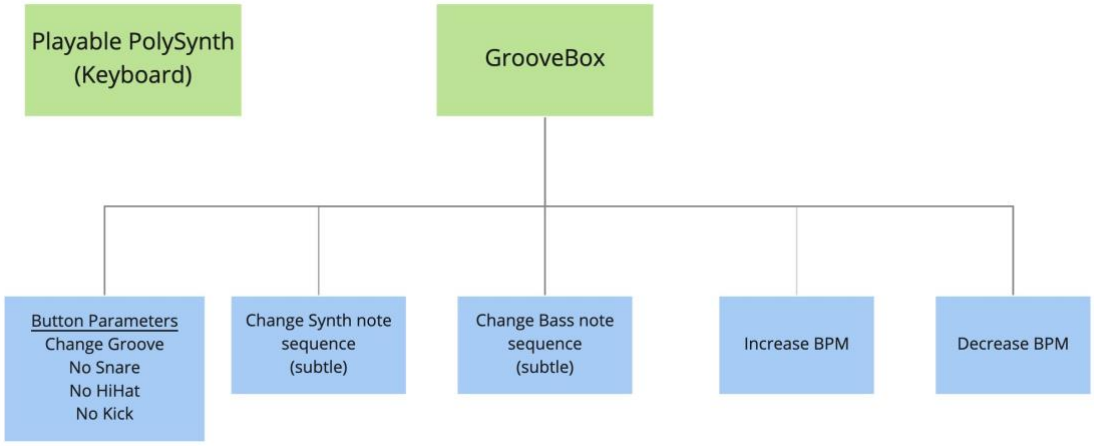

Figure 4 - Structure of Pd patches in Scene 3

3.4. Beyond Wwise's interactive music hierarchy - Scene 4

For this scene, Wwise's musical possibilities beyond its interactive music hierarchy were explored. Multiple cubes were positioned around the room that play samples from a random SFX container consisting of drum noises. The closer the player gets to each cube, the more it will switch from being 3D positioned to being played back in its stereo format (through Wwise's "spread" parameter). The two "scary string" sample container objects in the scene are also only audible when the player moves closer to them, but they are always played back in stereo (see "Scene_4_Video.mp4" at 0:26).

Additionally, a SFX blend container determines which of five different drone samples is being played back in accordance with where the player is positioned on the x-axis. A high pass filter is applied to the container's playback: its intensity is defined by the player's position on the z-axis.

Another useful tool within Wwise is the possibility to create a sampler instrument by routing a MIDI segment's output to an SFX object. This way the SFX object is being played by the current MIDI note in the MIDI segment.

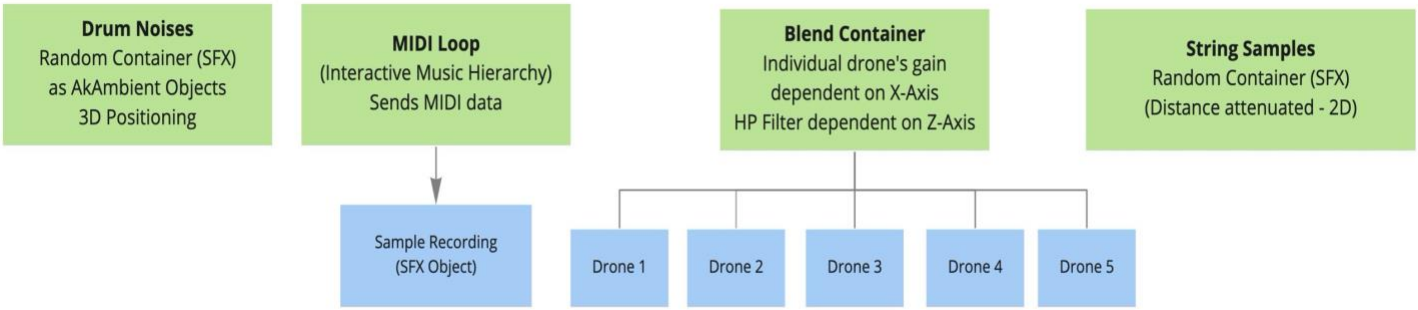

Figure 5 - Musical structures of Scene 4 (Wwise - Beyond the interactive music hierarchy)

Scene 4 demonstrates how Wwise's different elements and hierarchies can be used to compose an adaptive, generative piece of music without needing to depend upon the interactive music hierarchy.

3.5. RTPCs in Wwise - Scene 5

Real-time-parameter-controls (RTPCs) are "continuously evolving parameters" [6, p. 161] that are changed by real-time gameplay data from Unity.

In scene 5, all of the outputs by the different instruments are being adjusted in real-time in different ways through the use of RTPCs on certain effect. The Juno synth's make-up gain is determined by the player's position on the x-axis, the Absynth synth's output is being time-stretched according to the player's proximity to the cylinder and the Arcade sample is run through a tremolo effect, the intensity of which is reliant upon how many spheres the player has collected.

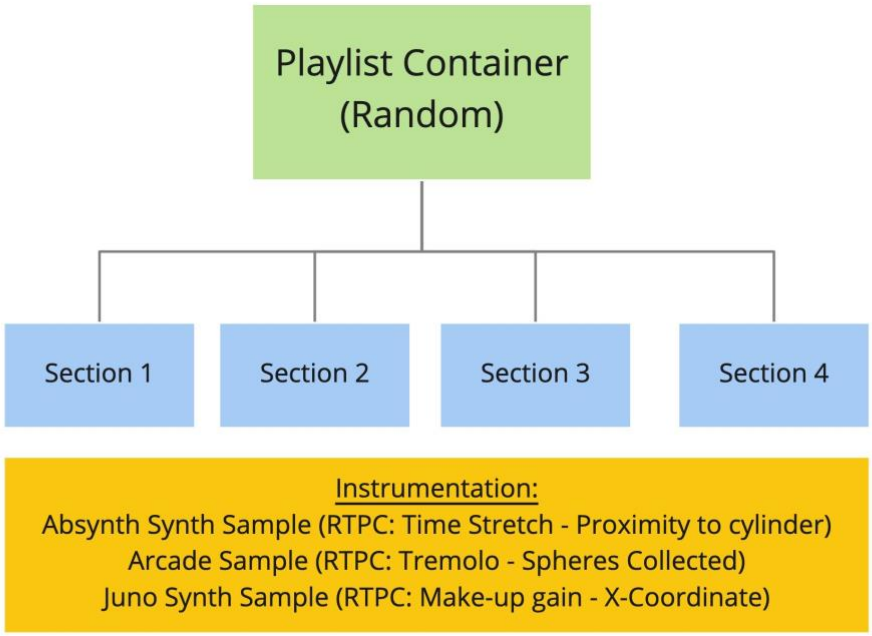

Figure 6 - Musical structures of Scene 5 (Wwise - RTPCs)

3.6. Unity's own audio engine - Scene 6

Although an audio middleware such as Wwise is easily integrated into a Unity project, Unity also possesses its own audio engine. However, the options a user is provided with by the audio source elements inside Unity are limited, and more fitted towards the use of SFX objects. It also seems as though these adjustable parameters were chosen with a certain type of game in mind - action games. This becomes especially obvious when considering why something as specific as a doppler effect is one of the parameters available in a generic audio source object: because such an effect would be used for simulating an object, presumably a bullet, quickly moving past the player [12] (see "Scene_6_Video.mp4" at 0:11).

But there are certain cases where using Unity's audio engine for music could actually be preferred. In scene 6, the capsules which keep spawning choose a single sample from a folder that they repeat until they are re-spawned. This way the room itself provides a space for a generative musical composition to unfold. The script at the core of

the object (written by Dr Jules Rawlinson) demonstrates a generative method that Wwise isn't capable of in this particular manner. Hence, there are certain niches in video game music where Unity's own audio engine is more suitable.

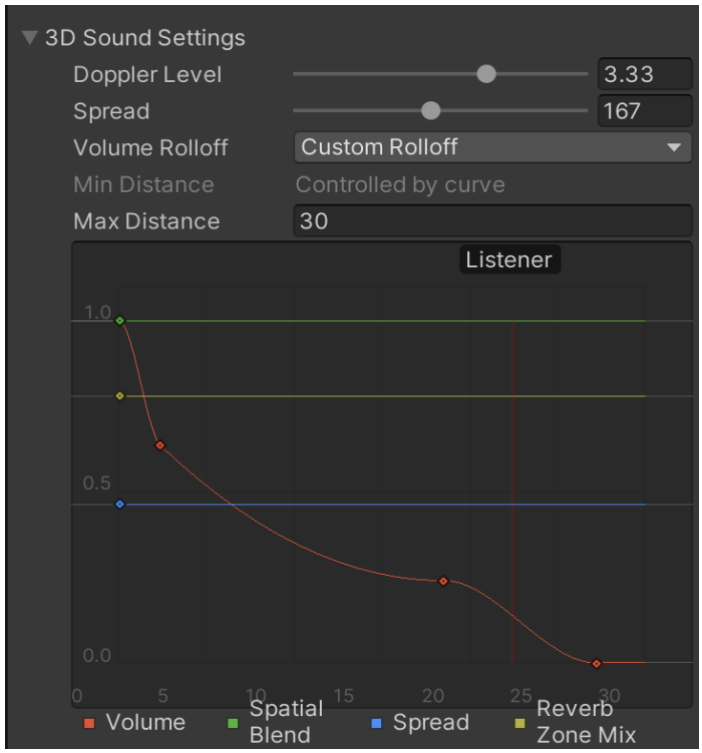

Figure 7 - 3D Sound Settings of a generic audio source in Unity (incl. Doppler effect)

3.7. Real-time synthesis - Scenes 7 & 8

Scene 7 takes a more simplified look at how easily and quickly real-time synthesis can adapt to a game. Even with such simple structures as used here, a few elements can already form an adaptive piece of music, the sound parameters of which can be tweaked and readjusted as needed. The music can seamlessly adapt to the game's environment and narrative. And due to Pd's open-source format, the possibilities of synthesis are endless.

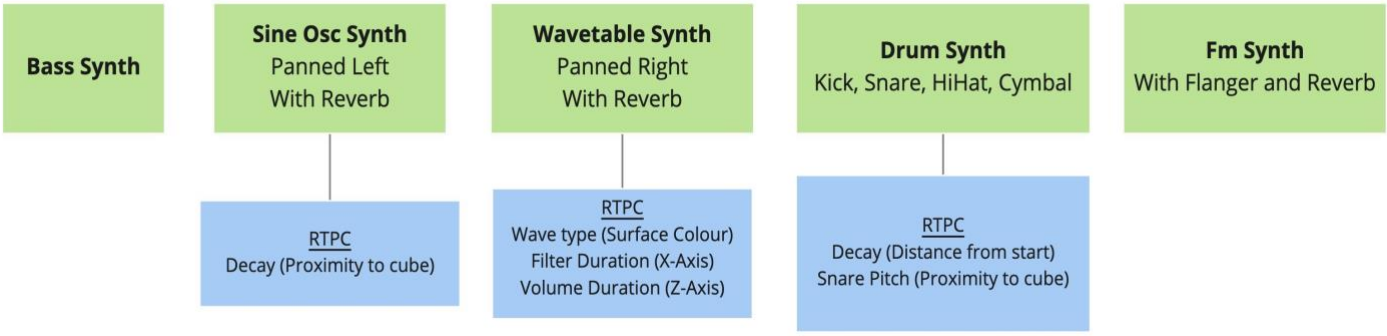

Figure 8 - Sonic elements and Pd patches of Scene 7

As a comparison, scene 8's source of sound is Wwise's own real-time synthesis tool, Synth One (plus a simple pre-recorded 808 drum loop). This frequency modulation synth offers a limited range of adjustable parameters such as both oscillators' waveform and pulse-width modulation. It is very challenging to create a musically interesting sound from this plug-in. Synth One is no competition to the possibilities of building any kind of synth in Pd. With the option to import MIDI files into Wwise however, Wwise could allow the integration of more musically complex plug-ins to fully utilize this functionality.

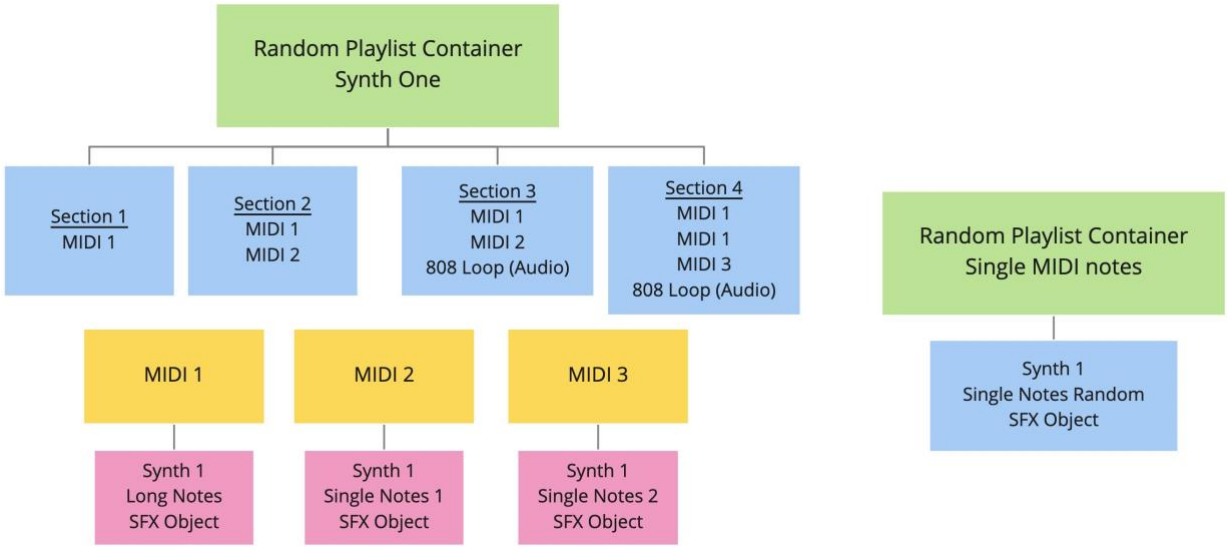

Figure 9 - Musical elements of Scene 8 (Wwise - Synth One/1)

3.8. Granular synthesis - Scenes 9 & 10

Both scenes 9 and 10 use granular synthesis to create their musical soundscapes. Granular synthesis "can be defined as using short sound events to produce a full sound texture from these grains" [13, p. 135].

Scene 9 applies this method with two different granular synthesisers in Pd. There is no player interaction. And still, it manages to generate a great number of constantly evolving musical structures. This truly generative method (as per Plut and Pasquier's definition) proves a great asset for avoiding listener fatigue (when listening to repeated music becomes tiring [2, p. 278]).

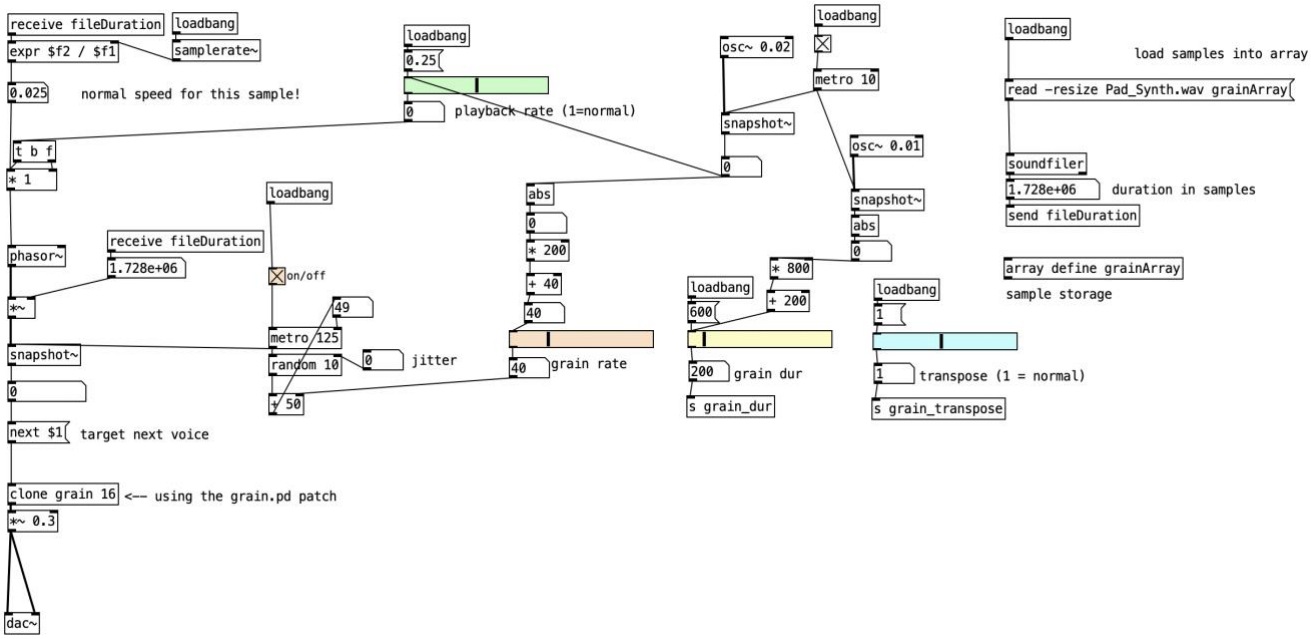

Figure 10 - Granulator Pd patch in Scene 9

Scene 10 uses Wwise's built-in tool SoundSeed Grain. Certain parameters can be randomised, and the sound can be controlled by MIDI input (such as the piano sample, see "Scene_10_Video.mp4" at 0:29). However, the randomisation for this plug-in has its limitations and the sound becomes repetitive after a certain amount of time.

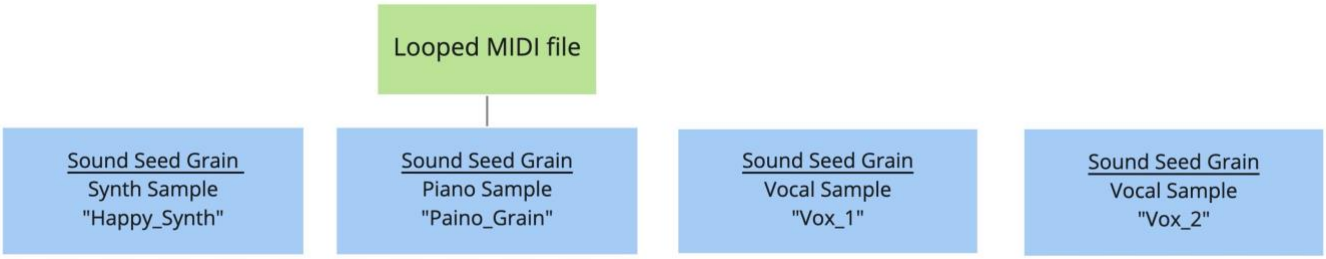

Figure 11 - Sound Seed Grain elements Scene 10 (Wwise)

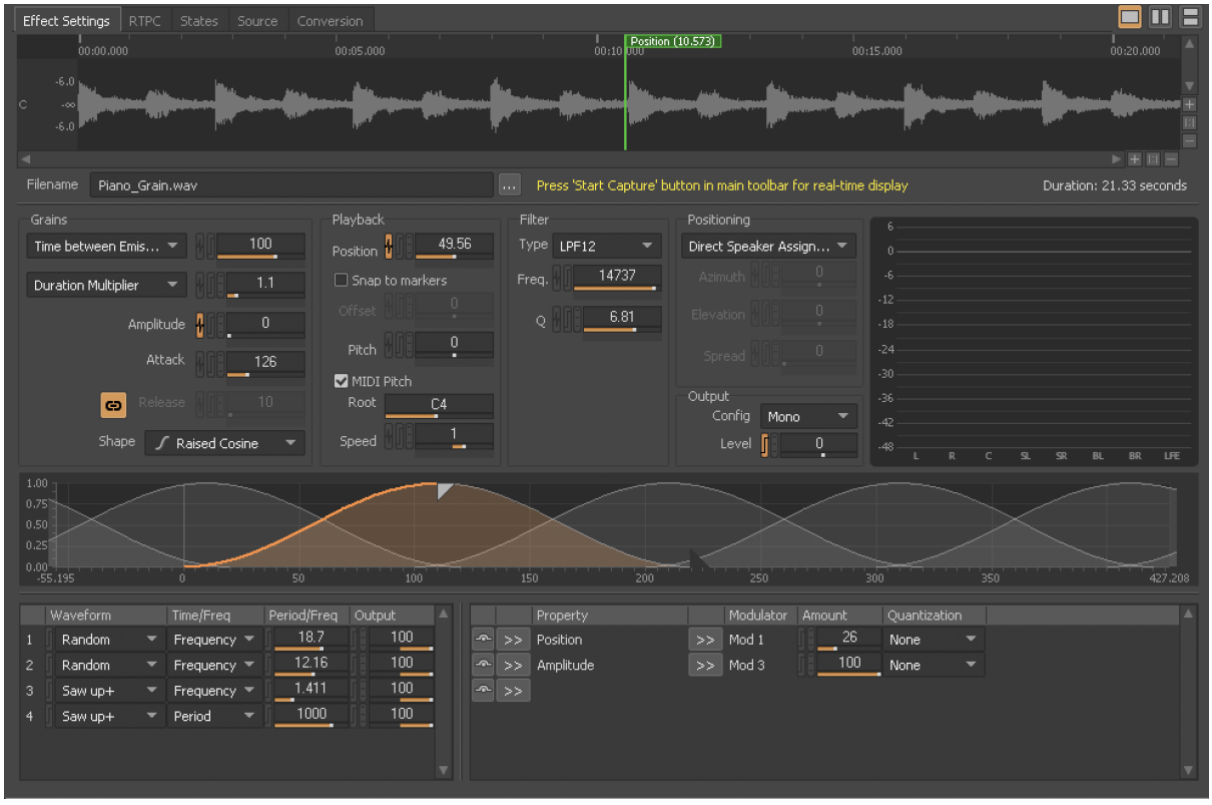

Figure 12 - Sound Seed Grain granular synth interface (Wwise)

3.9. Physical modelling of instruments - Scenes 11 & 12

An issue that often arises when talking about generative music in games is the ability to be on par with recorded music. A generative method that is able to compete with orchestral scores is physical modelling of instruments. Through physical modelling, one can influence the many small details that are so crucial to the sound of an instrument and change parameters in real-time. Both Collins [6, p. 149] and Farnell [3, p. 301] have highlighted the benefits of DSP and physical modelling in video games.

Scene 11 displays three examples of physical models of instruments - a piano, a marimba and a guitar - all completely synthesised. While the marimba and the piano's timbre do not change in this instance, the guitar's sound is being modified in real-time as the player moves between the two spotlights.

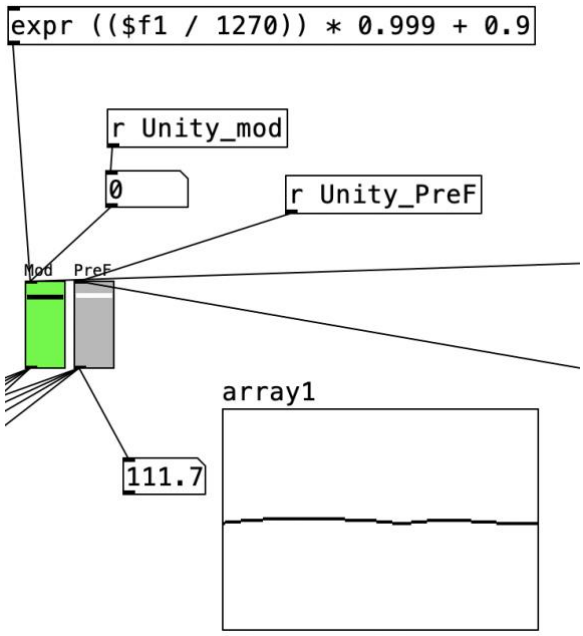

Figure 13 - Physical modelling of a guitar in Pd (Karplus-Strong), Input parameters from Unity and array displaying string movement

Scene 12 then provides a direct comparison to the physical modelling explored in Pd, using a pre-recorded looped guitar chord sequence and single guitar notes in Wwise. When considering the single guitar notes, which vary in length and damping, they are always fixed in time and need to be recorded individually. Physical modelling in Pd avoids this laborious process and provides greater variability.

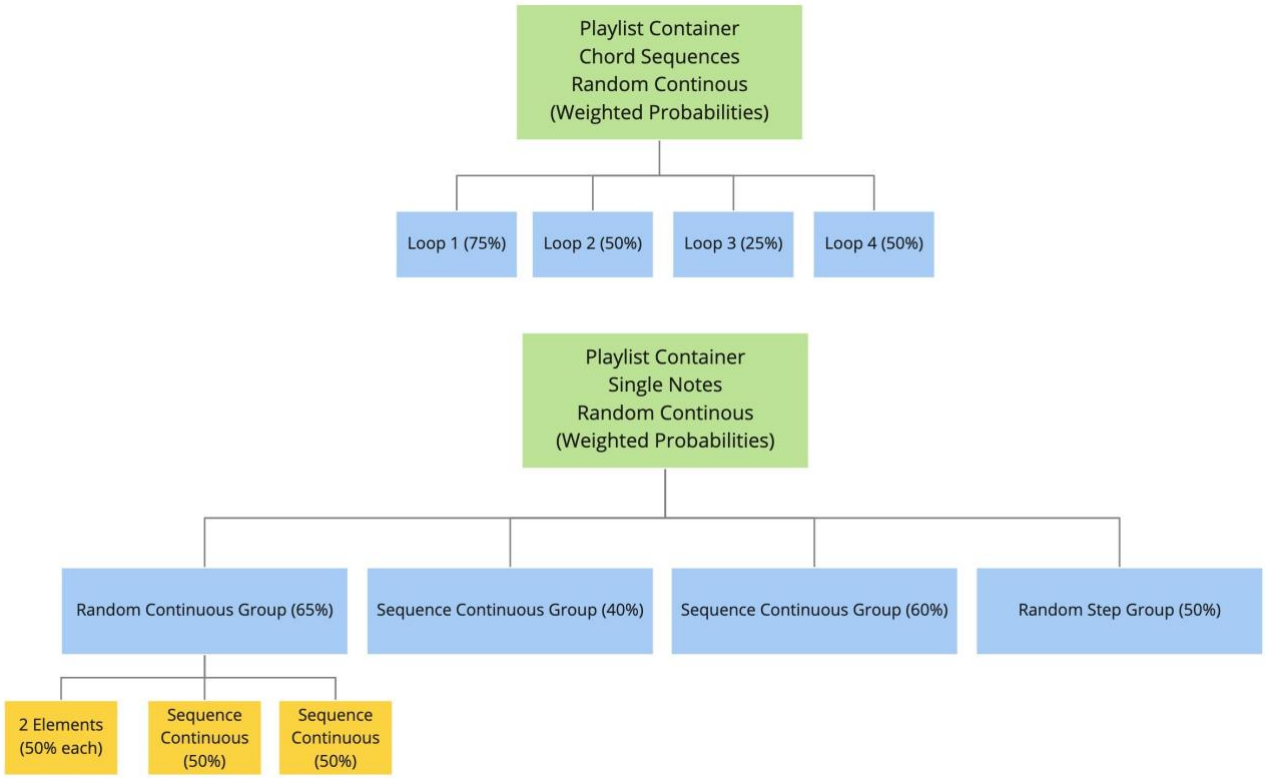

Figure 14 - Musical structure of Scene 12 (Wwise)

3.10. Markov models - Scene 13

Scene 13 implements two different types of Markov models. A Markov model "describes sequences of events in time" and "calculates the stochastic probability for a future state based on previous states" [14, p. 47]. It has been proven that Markov models "can be used to encode many rules of Western music theory" [15, p. 23], which makes them an ideal generative tool in (Western) video games.

The first instance of a Markov model in this scene uses predetermined probabilities to determine the sequence of chords. The second Markov Model however is being taught all possible sequences in real-time. As the player collides with the cubes in the room (see "Scene_13_Video.mp4" at 0:13), the Pd patch associated with these cubes receives a specific MIDI note associated with each cube, thus learning all possible note

progressions. This isolates this scene as the only one using a learned algorithm in the entire game.

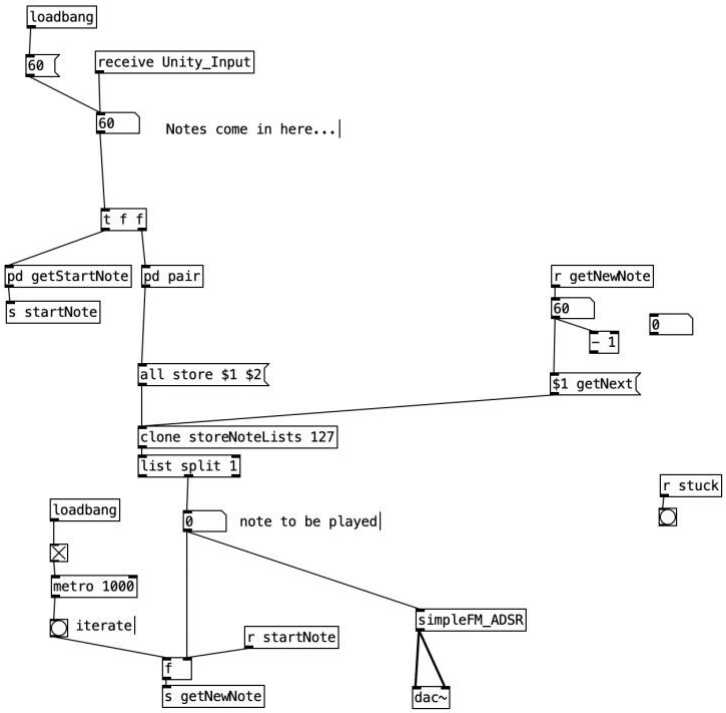

Figure 15 - Pd patch of learned Markov model for single notes in Scene 13

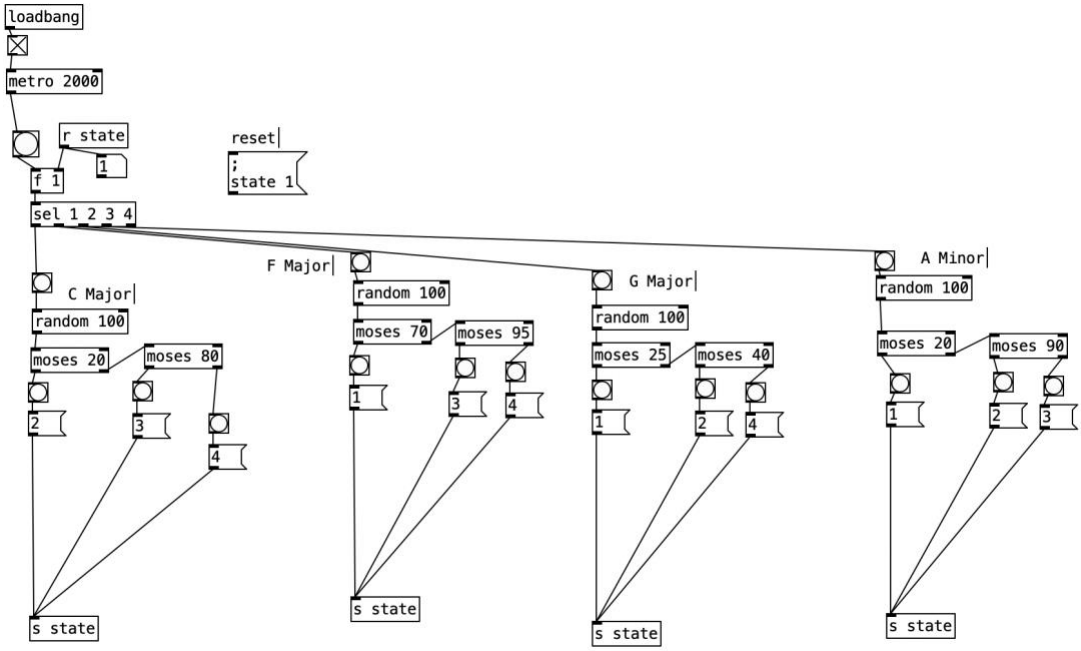

Figure 16 - Pre-set Markov model for chord progression in Scene 13

3.11. Combination of all methods - Scene 14

Scene 14 displays a combination of elements from all three different audio engines. Moving capsules that loop a single audio file inside Unity's audio engine, a switch container in Wwise that triggers musical segments according to the player's progression through the corridor and a granular synthesiser in Pd that provides endless new musical material even after the player has triggered the last state in Wwise.

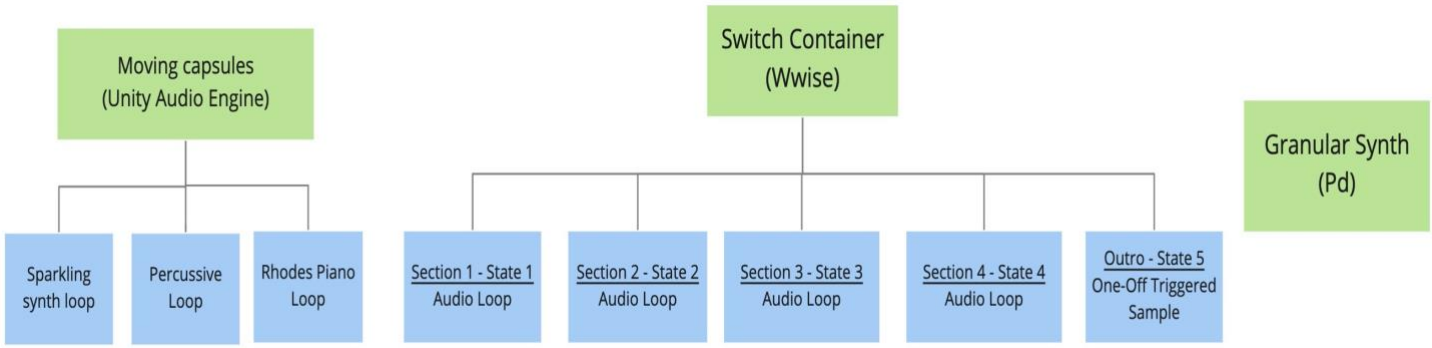

Figure 17 - Musical elements of all three audio engines in Scene 14 (Wwise, Pd, Unity)

4. Comparison of the methods

It is evident that both Wwise and Pd are capable of various generative and adaptive techniques for music creation, but that Wwise due to its pre-set parameters and thus limited interface obviously lacks the endless opportunities and possible add-ons that Pd and its open-source format provide.

Pd is certainly capable of providing the user with all features that are available in Wwise. But the question isn't whether it is possible, the question is if such tools have already been built in Pd, as having to build every single one of these generative tools and connecting them to Unity is a laborious task and therefore often not a feasible alternative.

4.1. “Inside The Walls” categorised in Plut and Pasquier’s typology

When looking at every scene categorised within Plut and Pasquier’s typology (see Figure 18), Wwise’s strength clearly lies with the generative task of arrangement, as it was essentially made to “adapt linear, pre-recorded music to a nonlinear environment” [15, p. 4] (transformational algorithmic music). Pd on the other hand is more suited for the compositional generative task (generative algorithmic music). Both systems are able to use either rule-based or stochastic generative algorithms, yet Pd is clearly the stronger choice when it comes to real time synthesis (musical representation category).

Moreover, Plut and Pasquier’s distinction between symbolic or audio based musical representation (so synthesis vs sample-based) demonstrates again how generative music is not equal to real-time synthesised music, thus discrediting Farnell’s previously mentioned categorisation.

Overall, “Inside The Walls” as a whole game would fall into the “fringe system” category: a category for games that exist on the fringes of Plut and Pasquier’s typology, due to its combination of various techniques [9, p. 15]. A combination of the best features of both applications could ensure great adaptability and variability for a game. Their features could be compounded to create an optimised “fringe system”.

Basic Information			Musical Dimensions				Gameplay Dimensions		Architecture Dimensions		
Scene No.	Audio Engine	Short Description	Generative Task	Directionality	Granularity	Grid	Ambience	Adaptivity	Gen. Algorithm	Musical Rep.	Musical KS.
1	Wwise	Interactive music hierarchy	Arrangement	Mixed	Measure, Instrument	On	Ambient	Adaptive	Rule-based	Audio	External
2	Pd	Changing scales, transposition	Composition	Horizontal	Note	On	Ambient	Adaptive	Rule-based	Symbolic	External
3	Pd	GrooveBox, Poly Synth	Arrangment, Composition, Performance	Mixed	Phrase, Instrument	On, Off	Ambient	Adaptive	Rule-based	Symbolic	External
4	Wwise	Random containers, Blend containers	Arrangement, Composition	Mixed	Note, Instrument, Inst. Parameter	Off	Ambient, Sourced	Adaptive, Linear	Rule-based, Stochastic	Audio	External
5	Wwise	RTPCs	Arrangement, Performance	Mixed	Phrase, Inst. Parameter	On	Ambient	Adaptive, Linear	Rule-based, Stochastic	Audio	External
6	Unity	Moving sonic objects	Arrangement, Performance	Mixed	Phrase, Inst. Parameter	Off	Sourced	Adaptive, Linear	Rule-based, Stochastic	Audio	External
7	Pd	Real time synthesis	Composition, Performance	Horizontal	Note, Inst. Parameter	On	Ambient	Adaptive	Rule-based, Stochastic	Symbolic	External
8	Wwise	Real time synthesis with Synth One	Arrangment, Composition	Horizontal	Note, Phrase	On	Ambient	Linear	Stochastic	Symbolic, Audio	External
9	Pd	Granular synth	Composition	Horizontal	(Micro-) Note	Off	Ambient	Linear	Rule-based, Stochastic	Audio	External
10	Wwise	Granular synth	Arrangement, Composition	Horizontal	(Micro-) Note, Note	Off	Ambient	Adaptive	Rule-based, Stochastic	Audio	External
11	Pd	Physical modelling	Composition, Performance	Horizontal	Chord, Note	On, Off	Ambient	Adaptive	Rule-based, Stochastic	Symbolic	External
12	Wwise	Guitar chords and single notes	Arrangment, Composition	Horizontal	Phrase, Note	On	Ambient	Linear	Stochastic	Audio	External
13	Pd	Markov fixed chain and learned melodies	Composition	Mixed	Chord, Note	On	Ambient	Adaptive	Stochastic	Symobolic	External, Learned
14	Mixed	Combination of audio engines	Composition, Arrangement	Mixed	Measure, Note, (Micro-) Note	On, Off	Ambient, Sourced	Adaptive, Linear	Rule-based	Audio	External
15 + 16 + 17	Wwise	Loops in Wwise	N/A	Horizontal	Phrase	On	Ambient	Linear	Rule-based	Audio	External

Figure 18 - "Inside The Walls" scenes categorised in Plut and Pasquier's typology of generative music in video games

5. Contextualisation within contemporary research

More recently, in an attempt to explore generative music in games that is capable of competing with the sound quality and characteristics of composed pre-recorded music, there have been a number of studies exploring generative music in video games.

The music generator MetaCompose was built "to create music in real-time that expresses different mood-states in a game environment" [7, p. 1]. Its purpose is to "modify the music played in real-time depending on the affective state the interactive media wishes to convey", generating "affective expression music" that conveys the mood best fitting for the current gameplay state [7, p. 3]. When tested in a gameplay setting, the music generated by MetaCompose was perceived to have a better overall quality by the players and was generally preferred over linear music [7, p. 8].

As another example, Anthony Prechtl made similar observations in his research on the effect his own adaptive music engine built in Max MSP would have on player experience. His paper concludes that players prefer dynamic music to static music and that it greatly supports a game's narrative with its emotional expression [15, p. 101].

While this dissertation does not include a comprehensive study on how its different approaches to music generation affect a group of players, the advantages the generative techniques in "Inside The Walls" have for adaptability and variability are clearly demonstrated.

Generative music engines like these could potentially lead to personalized game music [15, p. 104]. How a game's music might best suit the gameplay is often dependent

on the individual player. As music is “a potent emotional conveyer” and game music purpose can be understood to “evoke emotion” and “convey a narrative arc musically” [5, p. 9], personalising music to each player would enhance these qualities.

Adaptive music has become the standard for video game composition, and from the results of all these independent works of research, it seems that composition for generative tools is the next step forward. Curiously, while these experiments have been present in academic research on game music for some time, there seem to be remarkably few standardized and commonly used applications available for such forms of composition.

5.1. The disconnect between industry and academia

One AAA game whose music was created with a generative composition tool using a Pd interpreter was EA’s “Spore” (2008) [3, p. 304]. The system devised for Spore produces completely improvised music based on smaller elements that were assembled by EA together with composer Brian Eno. Spore exhibits “the power that could be harnessed in generative music” [1, p. 29] and how it can easily avoid listener fatigue. Its use of generative music has been understood as a great advancement in game music, yet it has been 13 years since its release and it remains unrivalled in its use of generative music amongst AAA games.

There is evidently a clear disconnect between academic research into generative music and its actual implementation in industry games. Various factors might explain such a detachment.

When industry standard audio middleware such as Wwise first became available, it wasn't possible for individuals to obtain a license. Pd's open-source format was therefore far more appealing to academics [12].

Furthermore, generative music systems in video games have often been incredibly CPU-intensive. As video games and their graphics use a lot of CPU-power already, developers are not likely to assign too much CPU-power to a game's audio. However, this argument is becoming increasingly insignificant as newer generations of game consoles have "sufficient computing power to allow advanced levels of audio synthesis and processing" [16, p. 187].

Additionally, for many years, academic discourse on video game audio was absent and education on video game development at industry standard was not available [4, p. 101]. While the experts applied their knowledge in the industry, research in the field of dynamic music has been lacking [1, p. 49].

6. Conclusion

There are many clear advantages when using generative music tools in video games: increased adaptability and variability avoids listener fatigue while personalised music enhances the individual's game experience. To achieve a more prevalent use of generative music in video games, certain steps could be taken.

It would be logical for audio middleware such as Wwise to allow the direct inclusion of plug-ins in the same way DAWs do. Obviously, due to licensing issues, including plug-ins by bigger companies would not be possible in most cases. Therefore, collaboration between not just industry and academia would be necessary, but also between audio middleware developers and plug-in developers, in order to create easy to use generative music plug-ins. Wwise already work with a handful of partner companies (such as iZotope) to integrate their plug-ins. But such collaborations should become a priority. Wwise also allows for its users to create their own plug-ins to include in their Wwise projects but their construction is complex and community developers simply lack the financial means of large plug-in companies.

Furthermore, education on dynamic music and its composition should receive more support, especially on the undergraduate level. Linear music is still overly favoured in music education. And as Collins puts it, "marketing the advantages of procedural audio is difficult to an audience that may not understand what it means" [8, p. 12]. Equally, this would discredit the assumption that more widespread use of generative music tools would make composers redundant. On the contrary, it would merely introduce them to a new way of bringing their compositions to life.

“Inside The Walls” shows how easily adaptive and generative music can be implemented in a game and therefore how simple it could be to adapt these techniques for industry games. As more students will continue to take an interest in this subject, such practices will quickly become standard practices in the game industry.

Appendix

Figures

LibPd Wrapper

"A libpd wrapper for Unity" by Niall Moddy, https://github.com/LibPdIntegration/LibPdIntegration.

External Pd patches used

- "Stepcycles.pd" by Martin Brinkmann (Scene 2), 2015, http://www.martin-brinkmann.de/pd-patches.html.
- "groovebox1r4.pd" by Martin Brinkmann (Scene 3), 2010, http://www.martin-brinkmann.de/pd-patches.html.
- "polyphony.pd" by Really Useful Plugin (Scene 3), https://reallyusefulplugins.tumblr.com/richsynthesis.
- "Tutorial9_wavetable_with_clone.pd" by Really Useful Plugins (Scene 7), https://reallyusefulplugins.tumblr.com/richsynthesis.
- "Tutorial5_notes.pd" by Really Useful Plugins (Scene 7), https://reallyusefulplugins.tumblr.com/richsynthesis.
- "2synths_samplers1.pd" by Martin Brinkmann (Scene 7), 2019, http://www.martin-brinkmann.de/pd-patches.html.
- "4effects1.pd" by Martin Brinkmann (Scene 7), 2019, http://www.martin-brinkmann.de/pd-patches.html.
- "grainstates1r.pd" by Martin Brinkmann (Scene 9, Scene 14), 2011, http://www.martin-brinkmann.de/pd-patches.html.
- "granulator_patch.pd" by Really Useful Plugins (Scene 9), https://reallyusefulplugins.tumblr.com/richsynthesis.
- "piano~-help.pd" by Miguel Moreno (Scene 11), 2019, https://github.com/MikeMorenoDSP/pd-mkmr/tree/master/06-Instruments.
- "marimba~-help.pd" by Miguel Moreno (Scene 11), 2019, https://github.com/MikeMorenoDSP/pd-mkmr/tree/master/06-Instruments.
- "ks-synth.pd" by Roady Kin (Scene 11), 2017, https://github.com/roadyyy/KarplusStrong.

Additional files

- Unity Folder - “np_InsideTheWalls_UnityProject”
- Wwise Folder - “np_InsideTheWalls_WwiseProject” (Inside Unity folder)
- MacOS Game Build - “np_GameBuild_MacOS”
- Videos of each individual scene - “np_Scene_Videos”

Videos of the individual scenes

- Start_Menu_Video.mp4
- Scene_1_Video.mp4
- Scene_2_Video.mp4
- Scene_3_Video.mp4
- Scene_4_Video.mp4
- Scene_5_Video.mp4
- Scene_6_Video.mp4
- Scene_7_Video.mp4
- Scene_8_Video.mp4
- Scene_9_Video.mp4
- Scene_10_Video.mp4
- Scene_11_Video.mp4
- Scene_12_Video.mp4
- Scene_13_Video.mp4
- Scene_14_Video.mp4
- Scene_15_Video.mp4
- Scene_16_Video.mp4
- Scene_17_Video.mp4

Google drive link to Unity and Wwise folder

https://drive.google.com/drive/folders/1tedkY-sXnJF2gFnb2VuK96TOUF215ZGw?usp=sharing

Bibliography

[1] D. Young, “Adaptive game music: The evolution and future of dynamic music systems in video games,” BSc dissertation, Media Arts and Studies, Ohio Univ., Athens, OH, 2012.

[2] K. Collins, “An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio,” in *Essays on Sound and Vision,* S. Hawkins and J. Richardson, Eds. Helsinki: Helsinki University Press, 2007, pp. 263-298.

[3] A. Farnell, *Designing Sound.* London: Applied Scientific Press, 2008.

[4] T. van Geelen, “Realizing groundbreaking adaptive music,” in *From Pac-Man to Pop Music: Interactive Audio in Games and New Media,* K. Collins, Ed. Aldershot: Ashgate Publishing Limited, 2008, pp. 93-102.

[5] L. F. Smith, “The Effect of Dynamic Music in Video Games,” BA dissertation, Dept. of Game Design, Uppsala Univ., Uppsala, Sweden, 2020.

[6] K. Collins, *Game Sound.* London: The MIT Press, 2008.

[7] M. Scirea, J. Togelius, P. Eklund, S. Risi, ”Evolving in-game mood-expressive music with MetaCompose,” in *AM’18: Sound in Immersion and Emotion,* Wrexham, UK, 2018, pp. 1-8.

[8] K. Collins, “An Introduction to Procedural Music in Video Games,” *Contemporary Music Review,* vol. 28:1, pp. 5-15, 2009.

[9] Plut, and P. Pasquier, “Generative music in video games: State of the art, challenges, and prospects,” *Entertainment Computing*, vol. 33, 2020.

[10] P. Brinkmann, *Making Musical Apps.* Sebastopol: O’Reilly Media, 2012.

[11] A. Marks and J. Novak, *Game Development Essentials: Game Audio Essentials.* Clifton Park: Delmar, 2009.

[12] Y. Seznec, private communication, March 25, 2021.

[13] L. J. Paul, “An introduction to granular synthesis in video games,” in *From Pac-Man to Pop Music: Interactive Audio in Games and New Media,* K. Collins, Ed. Aldershot: Ashgate Publishing Limited, 2008, pp. 135-149.

[14] A. E. Lopez Duarte, “Algorithmic interactive music generation in videogames,” *SoundEffects,* vol. 9:1, pp. 38-59, 2020.

[15] A. Prechtl, “Adaptive Music Generation for Computer Games,” PhD thesis, Centre for Research in Computing, The Open Univ., Milton Keynes, UK, 2016.

[16] L. J. Paul, “Video Game Audio Prototyping with Half-Life 2,” in *Transdisciplinary Digital Art: Sound, Vision and the New Screen,* R. Adams, S. Gibson and S. Müller Arisona, Eds. Heidelberg: Springer, 2008, pp. 187-198.