Tampere University of Applied Sciences

Procedural audio in mobile video games

Using Unreal Engine's MetaSounds to bring convincing soundscapes on mobile phones

Andrei-Alexandru Oprișan

BACHELOR'S THESIS
June 2024

Bachelor's Degree Programme in Media and Arts
Music Production

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Media and Arts
Music Production

Andrei-Alexandru Oprișan
Procedural Audio in Mobile Video Games
Using Unreal Engine’s MetaSounds to Bring Convincing Soundscapes on Mobile Phones

Bachelor's thesis 47 pages, appendices 6 pages
June 2024

The objective of this thesis was to explore the integration of procedural audio techniques in mobile games, focusing on Dream Primer's upcoming action role-playing game, "Astera." Dream Primer, the employer, sought to find a solution to optimise storage capacity for audio without sacrificing sound quality and immersion.

To address this challenge, MetaSounds were used. This technology allowed the creation of interactive and dynamic soundscapes, including procedural audio actors that serve as distinct fauna sounds, wind machines, and ambient magical textures, while maintaining low CPU consumption and storage efficiency. The results demonstrated that procedural audio could significantly enhance the game's atmosphere without compromising performance on mobile platforms.

It is planned that "Astera" will launch in early access on mobile and PC from late 2024 to early 2025, with console ports expected to follow. To further develop this work, examining a broader range of devices and focusing on RAM utilisation would offer a more comprehensive understanding of the limitations and challenges of implementing procedural audio in mobile gaming.

Key words: procedural audio, sound design, mobile games, unreal engine 5, metasounds

CONTENTS

GLOSSARY of ABBREVIATIONS AND TERMS (choose one or other)

TAMK	Tampere University of Applied Sciences
cr	credit

SFX - Sound Effects
DAW – Digital Audio Workstation
EQ - Process of adjusting the volume of different frequency bands within an audio signal
Voices - Every sound played in game requires something called a voice through which it is played
Attenuation - A curve that dictates various aspects of how sound behaves over distance relative to the listener
Monophony – Limitation of playing only a note or sound simultaneously
Polyphony - Ability to play a number of notes or sounds simultaneously
ADSR - Acronym for attack, decay, sustain, release
QA – Quality assurance
DSP - Digital signal processing
CPU - Central processing unit
NPC - Non-playable character

1 INTRODUCTION

The video game industry has evolved rapidly in the past few decades, with more and more players, and an increased number of games to play each year. We can see a stark difference in graphics and design, but something that is usually overlooked is the audio of video games. Game audio is typically composed of three categories: speech, sound effects, and music. Evidently, this important part of video games did not lag behind, and the technical aspects of what is possible to do with audio today have also undertaken a tremendous improvement.

One noteworthy advancement is the emergence of procedural audio, which has become a point of interest among industry professionals. Procedural audio is a versatile approach to generating and rendering sound in video games, enabling dynamic and adaptive soundscapes that respond to in-game events, environmental changes, and player actions. By generating sounds algorithmically, procedural audio can produce a virtually infinite variety of audio content, leading to more diverse and lifelike environments.

This thesis explores the concept of procedural audio in video games through MetaSounds, the DSP technology offered by Unreal Engine 5. We will explore its applications in sound design in the context of mobile gaming, using "Astera"—a game where I served as a sound designer—as a case study. We also aim to show the potential of dynamic and adaptive sound generation techniques with MetaSounds in order to create more immersive video game worlds and offer practical benefits such as reduced data storage requirements and scalability for big projects that require a massive amount of audio content. This should be compatible with CPU utilisation to make sure the procedural audio techniques achieved through MetaSounds can be utilised on mobile.

2 LITERATURE REVIEW

2.1 Evolution of audio in mobile games

The landscape of mobile gaming has undergone a profound transformation over the past two decades, with significant advancements in the realm of audio.

In the beginning stages of mobile gaming, audio was a rudimentary affair, characterised by simplistic beeps and tones. The earliest known game on a mobile phone was a Tetris variant on the Hagenuk MT-2000 device from 1994. (Phone Arena 2015.) This game however didn't support any type of sound. According to the online publication "It's Nice That" it wasn't until 1997, when Nokia launched Snake, which was pre-installed in most mobile devices manufactured by Nokia that players could hear sounds for the first time coming from a mobile game (2021). The limitations of early mobile devices necessitated minimalistic approaches to sound design. These games relied on basic sound effects for user feedback. These early audio elements, though modest, played a crucial role in enhancing the gaming experience and establishing a foundation for future developments.

As mobile hardware capabilities progressed, particularly with the advent of smartphones, the audio landscape of mobile games underwent a significant transformation. A major achievement was the possibility to play multiple layers of sound or voices simultaneously. This marked a departure from the monophonic constraints of early devices, allowing for music and sfx to be played simultaneously. (Collins 2014.) Games like "Soul Trapper" leveraged the storage and CPU of the first generation iPhone to deliver a game centred on audio that was 264 MB in size. The game featured a wide variety of audio puzzles that interweave seamlessly with the story, requiring stereo perception, memorization, fast-action reflexes, and audio pattern matching to complete (IGN 2009).

There are several distinctions between console and mobile games that have had a significant impact on the evolution of handheld games besides the hardware limitations. The inherent portability of handheld devices implies that

these games are frequently experienced in public settings, prompting designers to be aware of the potential intrusion of sound, which may be perceived as disruptive rather than enhancing gameplay for some game developers. Nevertheless, the development of headphones, along with the creation of rhythm-action and sound or music-oriented games for handheld consoles and smartphones, suggests that these devices are far from being intended for silent gameplay. (Collins 2014.)

Another important aspect is the market dominance of casual games and their lower price points. A casual game is a video game targeted at a mass market audience. Casual games may exhibit any type of gameplay and genre. They generally involve simpler rules, shorter play-time sessions, and require less skill. (Kultima 2009.) This led to the involvement of non-professional sound designers and composers. This is often the case because these games are frequently developed by individuals or very small teams. Even among major mobile phone developers, it is common to find a lack of in-house sound designers or composers. Instead, sound-related tasks are either outsourced or assigned to a team member whose primary focus is not sound. Typically, outsourcing occurs after the game's development is already underway, limiting the potential impact of audio. In contrast, the trend with console games is increasingly leaning towards early involvement of the sound team in the design process, allowing them to play a more substantial role. (Collins 2014.)

In contemporary mobile gaming, with console-quality games being ported on the iPhone 15 Pro max (Leadbetter 2023) we can see serious improvements in the resources available to create a high-quality auditory experience. More games are starting to take advantage of higher voice counts, spatialization, and real time effects. Moreover, mobile games such as usTwo’s “Monument Valley 2” or miHoYo’s “Genshin Impact” employ dynamic soundtracks that respond to in-game events and player actions, creating a seamless and interactive auditory experience. Nowadays, the implementation of procedural audio systems for both music and sfx, facilitated by tools such as Audiokinetic’s Wwise, Firelight Technologies’ FMOD, or Unreal’s Engine MetaSounds, is achievable on Android and iOS, featuring a workflow similar to that of console games. Titles like PopCap Games’ “Peggle Blast” highlight the potential of using data-driven

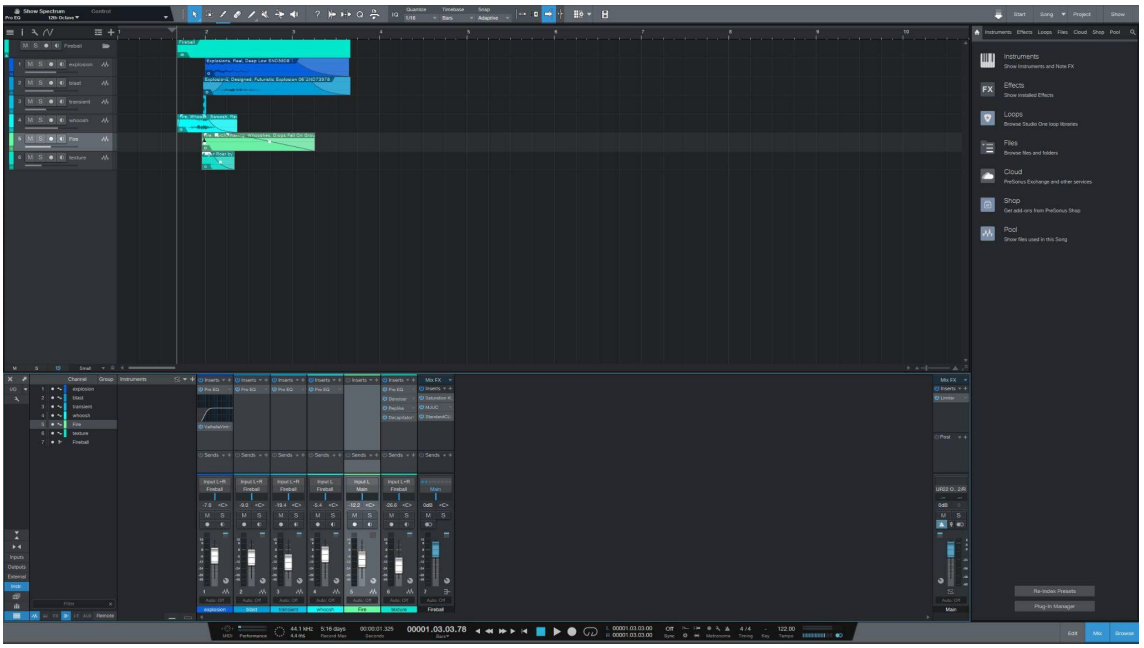
sound and music systems to create a dynamic soundtrack that responds very accurately to player action and gameplay states. This approach ensures a seamless integration of tonal sound effects within the musical key. All this audio feedback can enhance the gameplay experience by making it more responsive and interactive. (Mattingly, Shumate & Whitmore 2015.)

The evolution of audio in mobile games reflects the broader advancements in technology and a bigger number of games are leveraging not only superior quality audio but also better integration techniques to have a competitive edge and cater to player expectations.

2.2 Traditional audio integration in video games

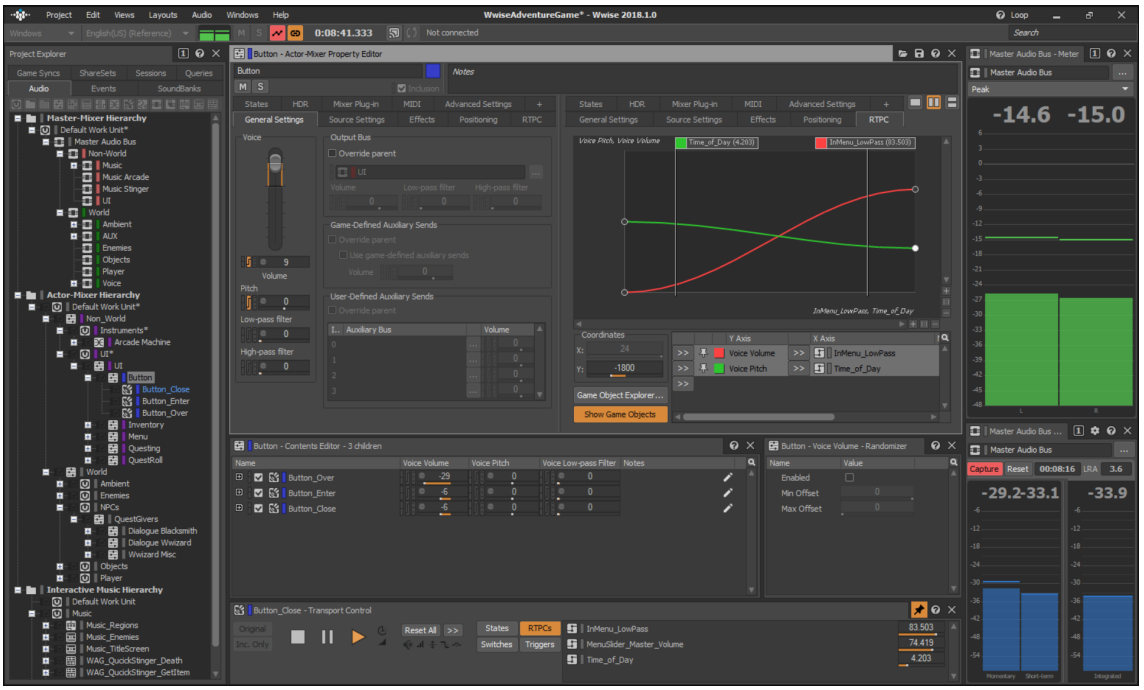
Compared to linear media like movies or animation shows, games are highly dependent on the integration process to offer a good audio experience. The process from start to finish is both technical and artistic. The sound integration pipeline commences with the audio recordings usually sourced from sound design libraries or recorded from scratch by the sound designers. Sound designers often embark on field recordings or studio sessions to capture a diverse range of sounds, from footsteps to natural ambiances.

Once recorded, the raw audio files undergo meticulous editing and processing. Sound designers use digital audio workstations (DAWs) to trim, clean, and enhance the captured sounds. Editing may involve removing background noise, adjusting volumes, and applying various effects to achieve the desired tonal characteristics. (Andersen 2020.)



Picture 1. Example of SFX asset creation in Studio One (Oprișan 2024)

After the assets are exported and organised, sound designers leverage game engine functionalities to implement the audio in the video game. Integration involves defining how sounds respond to in-game events, player actions, and the virtual environment. Parameters such as distance attenuation, spatial positioning, and environmental effects are configured to enhance realism and immersion Modern video games often utilise dynamic audio systems and middleware to enhance the adaptability and interactivity of sound elements. Middleware solutions like Wwise and FMOD provide tools for implementing dynamic music systems, adaptive audio, and interactive dialogues. These systems enable sound designers to take care of the sound logic without the need of an additional programmer. (Andersen 2020.)



Picture 2. Wwise project overview (Audiokinetic 2018)

To optimise storage space and ensure efficient resource utilisation, sound files are often converted to mono and compressed without significant loss of quality. The choice of audio formats and compression algorithms depends on the platform, storage constraints, and the desired balance between audio quality and file size. Various audio file formats offer distinct advantages and drawbacks in terms of file size, compression, quality, and compatibility. Uncompressed WAV files are of high quality but consume significant storage space and memory. On the other hand, MP3 files, being compressed and smaller, sacrifice some quality and may introduce latency. OGG files, also compressed and smaller, provide a balance by being more compatible and flexible compared to MP3 files.

In optimising game performance related to audio, various approaches can be employed to enhance efficiency. Some of these strategies involve minimising the memory usage of audio assets, while others concentrate on alleviating the load on the audio thread. Many of these practices contribute to improvements in both memory and CPU utilisation. (Antić 2023.)

Audio format	File size (Mb)	Compression	Audio quality
Wav	50	None	Best
Mp3	5	High	Good
Ogg	8	Moderate	High

Figure 1. The comparison of audio formats quality and file size (Simon Zolin)

After all the processes are done testing is required from both sound designers and game testers. This is a continuous and iterative phase in the sound integration pipeline. Sound designers or Audio QAs playtest extensively to evaluate how the audio elements work in the game (Laven 2023). Iterative adjustments may involve fine-tuning volumes, refining spatialization, or re-evaluating the effectiveness of dynamic audio systems. Sometimes bugs related to audio implementation may also occur that require the implication of both a sound designer and a programmer to fix the issue. Feedback from playtesting sessions guide the refinement process, ensuring that the audio aligns seamlessly with the overall gaming experience. (Laven 2023.)

2.3 Procedural audio in video games

Due to the small body of research, there is no globally accepted definition for procedural audio. Therefore in this thesis we will refer to procedural audio as the dynamic generation of sound during runtime, primarily informed by gameplay parameters or a set of logic rules. In other words, procedurally generated audio constructs in-game sound effects in real-time, relying on a predetermined set of behaviours.

This methodology bears a resemblance to other forms of procedural generation encountered in diverse facets of game development, such as level design or environmental art. Despite the additional computational load, it may impose, procedural audio boasts the distinct advantage of maintaining an economical RAM footprint, thereby affording sound designers the capacity to craft a soundscape intricately intertwined with the game's mechanics. (Mraz 2021.)

The origins of procedural audio can be traced back to the early days of video games, in which simple sound effects were generated using basic algorithms. However, the limitations of hardware at the time and the infancy of video games sound design and music prevented the development of procedural audio systems. With the advent of more powerful computing systems, the development of new software tools, and higher demands for video game audio from the general public, procedural audio has regained attention as a creative component of game audio design. (Sinclair 2020.)

Unlike static audio, where developers rely on pre-recorded sound clips, procedural audio enables the creation of dynamic and adaptive soundscapes that respond to the ever-changing game environment and player actions. This dynamic soundscape can range from wind generated through noise modulation to the revving engine of a car, making the in-game audio experience more believable.

2.3.1 Emergence of procedural audio in video games

The connection between procedural audio and early video games leans more toward music rather than sound design. This could be attributed to several factors such as technical limitations due to hardware constraints of earlier consoles and computers or emphasis on gameplay and visuals with rudimentary sound design. Additionally, we could also take into account the scarcity of audio specialists that possessed both programming skills and sound design capabilities that would lead to desirable and compelling results using procedural audio, rather than traditional techniques. (Collins 2008.)

In 1987 Toshio Iwai, a composer and artist, played a pivotal role in introducing innovative procedural techniques to game audio with the game “Otocky” by SEDIC. In “Otocky”, a side-scrolling shooter, players manoeuvre a ship that shoots round balls at enemy shapes and airborne musical notes. The firing actions of the player, accompanied by a simple two-note bass line, seamlessly merge with in-game mechanics to form the melody, dynamically quantized in real-time to the beat (Collins 2009).

The critically acclaimed game "Spore" (2008), crafted by Maxis, stands out as one of the earliest games which had commercial success and used procedural audio. In "Spore," procedural audio was employed to compose its soundtrack. Despite maintaining a traditional approach to sound design, the game showcased the innovative utilisation of procedural systems to intricately shape a unique sonic identity. "Spore" allows a player to control the development of a species from its beginnings as a microscopic organism, through development as an intelligent and social creature, to interstellar exploration as a spacefaring culture and relies heavily on procedural generation. The music was created in Pure Data, which is a visual programming language developed by Miller Puckette in the 1990s for creating interactive computer music and multimedia works, and consists of many tiny samples which generate the soundtrack in real time (Pure Data Community Site 2024). Melodies and rhythms are all generated within limited rules; for example, a sequence might use notes from only within one specific scale. The player can also construct and edit their own music, including various anthems for cities, by selecting from a number of rhythmic sequences and note samples. Mechanisms were put into place to limit the player's input, however, so that changes would not happen too radically and the pieces could 'make sense' musically (McLeran 2008).

In "No Man's Sky" (2016) by Hello Games, you explore, build and survive in a seemingly never ending outer space, which also relies heavily on procedural generation. In this game the fauna soundscapes and creature vocals were procedural generated. This is one of the first examples of successful video games that used procedural sound design 8 years after the release of "Spore". We will talk more about "No Man's Sky" procedural audio systems in the following chapter.

A significant step in the evolution of procedural audio through a "lower barrier to entry" was its incorporation into widely used game engines, thus making procedural audio more readily available for indie development teams. Notably, the possibility to include Pure Data in Unity in 2017 (Pure Data Community Site 2024) and its own DSP Graph in 2019 (Johnson 2019), though it retained its status as an experimental package until 2024. Similarly, Unreal Engine 5 unveiled MetaSounds in 2021 (Unreal Engine 2023), marking a noteworthy

development in the integration of procedural audio into mainstream game development frameworks.

The emergence of procedural audio in video games signifies a shift in the way audio is conceptualised and implemented. We will explore some games that realise the potential of procedural audio.

2.3.2 Notable games using procedural audio

No man sky

No Man's Sky is an action-adventure survival game developed and published by Hello Games set in a procedural generated outer space. Audio director Paul Weir introduced 2 major procedural systems into the game such as a procedural music generator called Pulse and the VocAlien system for all the randomised creatures in the game. Besides this, more ambiance elements like the fauna used the VocAlien plugin as well. We will focus on the sound design part.

Picture 3. No Man Sky gameplay (Hello Games 2016).

VocSlien is a real-time synthesis plug-in integrated into the game, offering a unique vocal identity for each creature while keeping memory usage in check

(Weir 2017). The plug-in sits within Wwise and can also be performed via MIDI like an instrument.

VocAlien employs resonant filters to establish peaks, modelling the essential components necessary for generating sound through a pipe. The equivalent of a 'Reed' propels the pipe, constituting a reciprocal interaction where the pipe also influences the reed. The final stage involves a filter that emulates mouth movements, specifically focusing on phonemes. This is a bidirectional process, wherein alterations to the mouth section dynamically impact the overall behaviour. The nature of the system is inherently chaotic to avoid a static predictable sound.

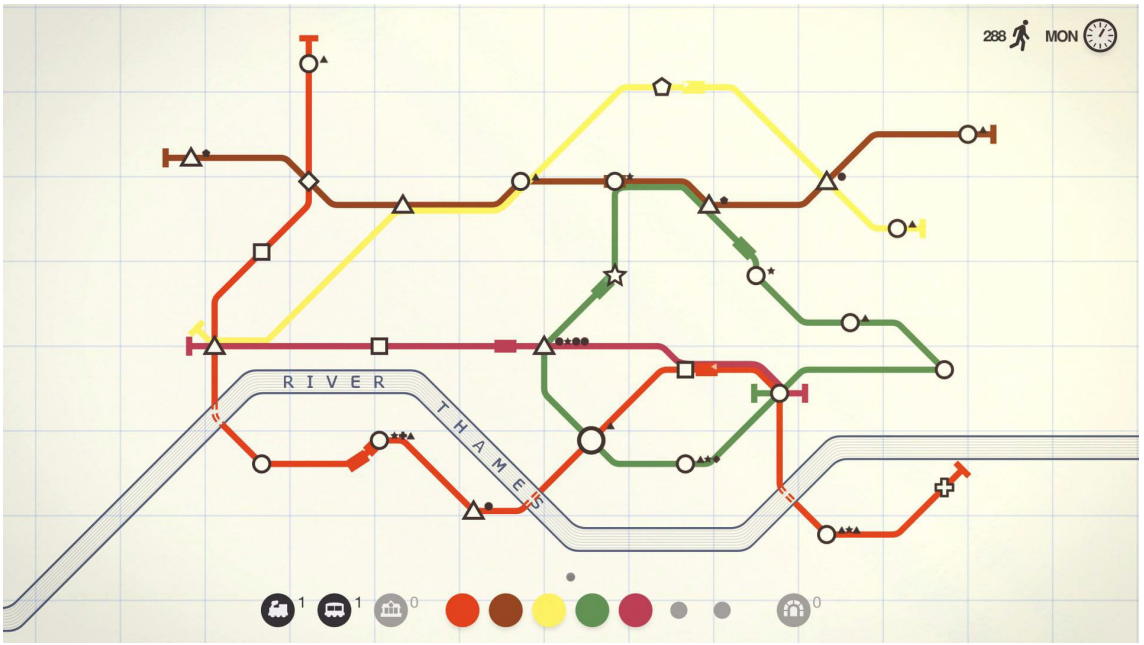
To get performances MIDI data is being authored through the plugin with a control surface app for the iPad and then saved in a DAW as MIDI. Information such as AD envelope, length of the pipe, intensity, harshness, filters, phonims, formants are all captured in the performance as emotions. This is a static MIDI performance that can be altered further in the game where multiple parameters can be jittered to give more randomness in that set performance.

At runtime the game sends information about the creatures to the plugin in order to determine what type of creature sound it should make such as size, head to body ratio, emotion and the seed in order to not randomise a different set of sounds for the same creature.

This system allowed a lot of flexibility for the ambiance fauna as well. The sound designers had the option to modify the sound of the birds based on in-game parameters such as time and weather making the transition seamless for each type of bird created by the plugin. Moreover, in a game with multiple biomes and planets that meant that repetition was not desirable as it would give away the feeling of exploring new worlds.

Mini Metro

In “Mini Metro” (2015) by Dinosaur Polo Club players draw lines between stations for trains to run. It is a strategy simulation game about designing a subway map for a growing city.



Picture 4. Mini Metro gameplay (Dinosaur Polo Club 2015).

Even though the procedural generation handles the music of the game, it is fair to argue that the line between soundscape and music tend to blur in “Mini Metro” especially at the start of a level.

"Mini Metro" has the audio generated based on a master pulse that incorporates authorised data including rhythm, pitches, and basslines. Additionally, game-driven data, such as the station type (circle, triangle, etc.), its placement on the map, and the number of passengers, serves as rules for shaping the soundscape. (Vreeland 2018.)

Each city or level possesses inherent musical characteristics, including access to specific rhythms, harmonic choices, and train engine sounds, among other elements. The player exerts some control over which of these qualities are played at any given time through their decisions regarding the size and configuration of their subway system. In each level, sequences of notes

symbolise the harmonic structure and voice leading of the music. Each subway line is associated with a rhythm and a note at any given moment. Modifying the subway line that has been altered least recently substitutes the oldest note in the harmonic structure with the next one in line. The rhythms operate in a similar manner, with altering a line resulting in a shift to the next available rhythm in a predefined list. Occasionally, the harmonic structure of the music may change based on the current week of gameplay. (Vreeland 2016.)

Most of the tonal sounds were designed with Serum, a software synth. For a lot of the UI sounds, Vreeland (2016) took sine tones and utilised effects on them. The sounds of the passengers are mostly sample based, mouth sounds and old drum machine samples, with lots of pitch and sometimes granular manipulation.

Reigns

“Reigns” (2016) by Nerial is a strategy video game where the player takes control of a kingdom and must make decisions to make their tenure as monarch last as long as possible. The decisions are based on cards offered to the player with events and binary choices.

Picture 5. Reigns gameplay (Nerial 2016).

The in-game language in “Reigns” for the characters that appear on the cards is done using procedural audio. The system was based on pre-recorded lines that were caught up in syllables and then reassembled in game at runtime based on a set of rules for each unique character in the game. The set of rules included the length of the written text on the card, the pitch, frequency and the syllables overlap. The pre recorded dialogue was improvised around a seed phrase for each character. e.g. “Money banana stand” for the joker character. That resulted in a unique voice for each character but the game doesn’t give the impression of a unified language. Nevertheless, the game works as a proof of concept of what can be achieved with procedural audio for speech in video games. (Vreeland 2020.)

2.4 Procedural audio techniques

Procedural audio encompasses a wide array of techniques aimed at generating dynamic and adaptive soundscapes in real-time.

First, we can mention a hybrid approach which entails using samples to create realistic soundscapes with procedural audio. With this technique, we rely on pre-rendered audio assets that are being played back under a set of rules alongside DSP processes like frequency modulation, compression, distortion etc. that can also be defined under a set of rules to provide multiple variations of sound. By incorporating pre-rendered audio assets, which are often recorded from real-world sources or high-fidelity recordings, we can achieve a level of realism and authenticity that may be challenging to replicate solely through procedural generation.

Another prominent procedural audio technique is physical modelling. Physical modelling synthesis is a collection of methods that employ mathematical models to simulate the physical properties of acoustic instruments or other sound-producing objects (Smith 2010). These instruments can be integrated into plugins. For instance, you can utilise Audiokinetic's Wwise open architecture to develop source plugins for generating sound and motion objects (Audiokinetic 2023). Although physical modelling can be computationally intensive and potentially time-consuming, when done right it can provide

endless variations with realistic results when combined with procedural audio. That can prove useful in big simulated video game worlds.

Granular synthesis is another powerful procedural audio technique that involves breaking down sound into tiny "grains" and recombining them to create complex textures and timbres. This technique offers flexibility and control over sound generation, allowing for the creation of evolving and morphing audio textures (Audiokinetic 2023). While the resulting audio may not mirror natural sounds, depending on the art style of the game, it can provide a rich aesthetic palette of sounds. In a game scenario, granular synthesis can be used to generate ambient soundscapes that evolve in response to player movement or environmental changes, adding depth and richness to the game's audio environment.

Subtractive synthesis involves filtering out specific frequencies from a sound source to shape its timbre and character. This technique is commonly used to create a wide range of sound effects, from rumbling explosions to whispering winds. By manipulating filters in real-time, subtractive synthesis can enable procedural sound generation that can adapt to evolving gameplay conditions. (Collins 2008.)

Additive synthesis, on the other hand, involves combining multiple sine waves with different frequencies, amplitudes, and phases to create harmonic and inharmonic sounds. This technique offers control over sound generation, allowing for the creation of various timbres that can result in musical instruments (Collins 2008). In a game context, additive synthesis can be used to generate expressive instruments that can be linked to evolving, procedural musical compositions. With enough work even realistic sound effects can be achieved such as bells, birds, or fantastical elements, that can work with a set of rules to create procedural soundscapes.

Wavetable synthesis is yet another synthesis technique that relies on a table of predefined waveforms to create simple or complex sounds (Collins 2008). By modulating the playback of these waveforms, wavetable synthesis can generate a wide range of sounds, from classic synthesiser tones that can be used in

virtual instruments to futuristic sci-fi sound effects and ambiances. It is also possible to obtain wavetables from scanning samples that will retain the timbre of the sound source as the algorithms will try to replicate the sound information in different wavetable positions. While additive and subtractive synthesis depend on oscillators that each generates one type of wave (ex. sine, triangle, saw, square), wavetable synthesis can cycle through and fade between waves on demand. These shifts are usually accomplished by manually changing the wavetable position on the synth itself, or using an LFO to automate things a bit more. Because the raw sound source is being stored and played back rather than generated from scratch, wavetable synthesis can be a lot less memory and CPU-intensive (Malinervo 2021). Like with the additive synthesis In a game scenario, wavetable synthesis can be used to create procedural music tracks or fantastical sci-fi sounds to great effect.

These techniques help in making audio experiences that can adapt to different situations and fit the context. The algorithms behind procedural audio can work in almost any aspect of a video game from vocal synthesis to music to soundscapes.

2.5 Benefits and limitations of procedural audio

2.5.1 Benefits

Procedural audio offers several advantages over traditional audio methods. While building soundscapes for a virtual world we can consume a lot of storage depending on the expanse of that world. Using traditional rendered assets, we are also bound to a recorded loop that does not change based on player input.

Procedural generated soundscapes can produce a vast variety of audio content, ensuring that each experience is unique. This is especially valuable in open-world games. Moreover, the audio content can dynamically adapt to the player's input. To illustrate, consider a scenario where a player's objective is to ascend a mountain in search of treasure. By employing generated wind sounds with noise modulation, it becomes possible to tie specific parameters to the

player's altitude, seamlessly altering the sound as the player ascends towards their goal.

In the development of "No Man's Sky" (2016) without a procedural audio approach to the creature vocals, it would have taken Weir years to handcraft a library of sounds for each creature you can encounter randomly in it.

2.5.2 Challenges

While procedural audio has numerous advantages, its implementation is not without challenges. The most important objective of procedural audio is to sound natural and immersive rather than a stream of endless audio.

Creating procedural audio systems can be complex, requiring specialised knowledge in sound design, mathematics, and programming. This usually means a collaborative effort between programmers and sound designers.

Generating audio procedurally in real-time can be resource-intensive, requiring powerful hardware and careful optimization to prevent performance issues (Mraz 2021).

When building soundscapes as more sounds are being created, verifying the quality and consistency of procedural audio assets in different game scenarios while taking into consideration player actions can prove error-prone and more time-consuming to manage.

Generative and adaptive music is gaining popularity in the gaming industry, but most games still rely on linearly composed music. This is partly due to the resource-intensive nature of generative systems and the industry's focus on pushing the limits of computing power for graphics and simulations, rather than audio. Additionally, generative music can be unpredictable and challenging to control, potentially resulting in less satisfying or monotonous compositions. Paradoxically, an endless stream of music can become repetitive. These factors have limited the widespread adoption of generative music in games, despite its potential benefits (Pult & Pasquier 2020).

3 METASOUNDS

3.1 Introduction to MetaSounds

MetaSounds can be defined as an audio system that provides audio designers with complete control over a DSP graph for the generation of sound sources. They empower audio designers to construct robust procedural audio systems, delivering precise timing and control down to the audio-buffer level (Unreal Engine 2023). This technology comes as a plugin in the Unreal Engine game engine.

With MetaSounds, audio designers can dynamically generate audio content during runtime and seamlessly blend procedurally generated sounds with other audio sources.

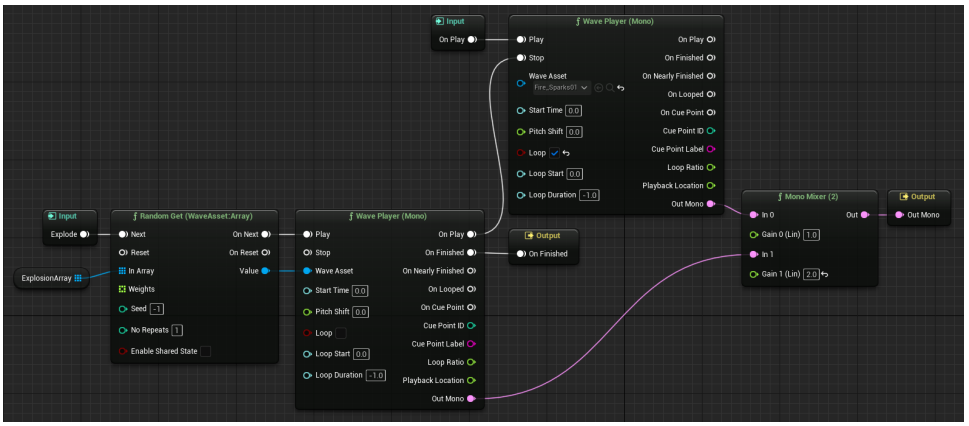
Moreover, MetaSounds are engineered for seamless integration with game data and player interactions, enabling the creation of immersive experiences triggered by in-game events (Unreal Engine 2023).

Each individual MetaSound operates as its own audio rendering engine, running in parallel with others and potentially offering independent rendering specifications, such as sample rate, buffer size, and channel count (Unreal Engine 2023).

A MetaSound can be played by dragging it from the engine’s content browser into the world, attaching it to in-game objects, placing it on animations, or by scripting its behaviour using code, etc.

3.2 Building MetaSounds

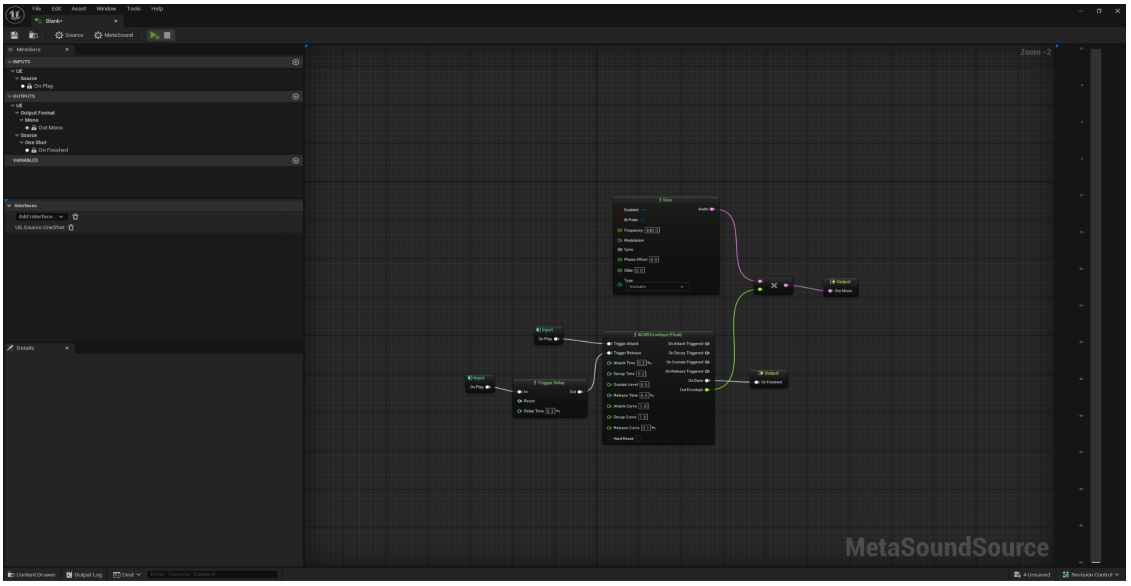
MetaSounds are created in a new MetaSound Editor where audio designers with no programming experience can create procedural sounds using a node-based interface.



Picture 6. Metaound with 2 wave players (Unreal 2023).

3.2.1 MetaSound Source

The MetaSound Source is a type of audio asset introduced in Unreal Engine 5. This asset is designed for playing or generating sounds. The source can come from wave assets or “generator” nodes that produce audio signals with simple waveforms, such as the Sine Wave, Square Wave, Sawtooth Wave, and Triangle Wave. There are also noise generators for White noise and Pink Noise. By default, the MetaSound includes an "On play" trigger input, a mono audio output, and an "On Finished" execution, which, when activated, stops the MetaSound but users can adjust the output format and remove the "On Finished" node for looping sounds.

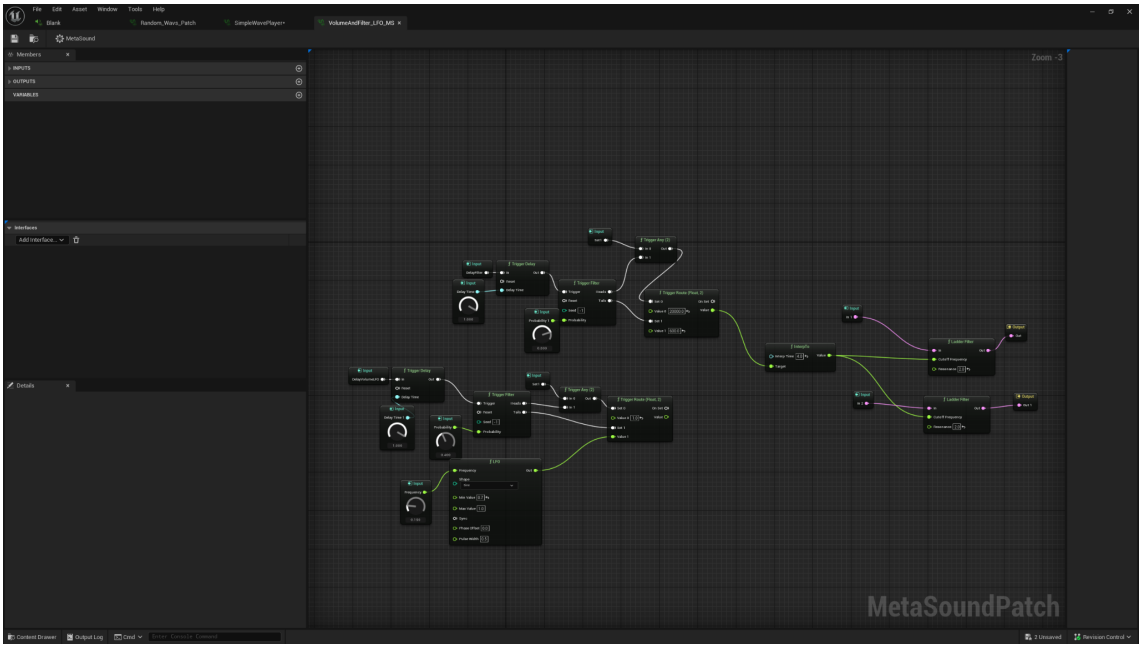


Picture 7. MetaSound source with Sine Generator and ADSR Envelope (Oprișan 2024).

3.2.2 MetaSound Patch

The MetaSound Patch can be considered an asset identical in terms of graph manipulation but intended to be referenced by other MetaSounds to encapsulate and reuse functionality (Unreal Engine 2023). They are essential in making the workspace more organised and they also provide productivity benefits.

Usually, the patches are made out of multiple nodes that can serve as logic structures that are being repeatedly used in the game for sfx or music. Converging multiple nodes into a single patch can work with building stereo effects as well.



Picture 8. Building a filter with MetaSound patch (Oprișan 2024).

Another great use of the Metasound patch feature is to simplify a node that has multiple parameters to have a cleaner workspace. For example, we can take the wave player (mono) and use only the play, wave asset input, signal output, and on finished output.

4 CASE STUDY: ASTERA

The practical part of this thesis will demonstrate the audio design techniques employed in creating an immersive sound environment for “Astera” using MetaSounds. "Astera" is a multiplayer isometric action video game developed in Unreal Engine 5 by Dream Primer. Dream Primer is a small independent video game studio. The game is set up in a fantasy world with steam-punk nuances and a sci-fi twist. At the time of writing this thesis, I’ve been working as a full-time employee on this game as the sole audio professional for 12 months. The scope of the game is big with over 40 hours of planned content including multiple locations with different biomes, a day and night cycle, characters, creatures, abilities, boss battle encounters, and cinematic scripted events. The release date was set for November 2024 in an early access state.

Picture 9. Astera in an alpha build (Dream Primer 2024).

When I was first approached to work on this game at the start of 2023 the requirements were to prioritise mobile as the main platform, to make sure the audio size is conservative with a limit of 400 to 500 MB for all the sounds and music, and to keep the audio processing and voice count efficient for CPU optimization and limitations on older hardware.

While it's technically possible for a single sound designer to provide all the

necessary audio within a two-year timeframe, it still poses a challenge. The scope and complexity of a game of this magnitude require a significant amount of work across various aspects of sound design, including creating, implementing, and fine-tuning numerous audio elements. Building tools and systems to manage all this becomes a must as we want to maximise the use of each asset created while maintaining a believable and immersive open world for the players.

Utilising procedural audio provided a solution to some of the challenges faced in the development of "Astera". By leveraging procedural audio techniques, the creation of dynamic and adaptive soundscapes became more manageable within the constraints of mobile platforms and limited storage space. Procedural audio allowed for the generation of diverse audio content on the fly, reducing the reliance on pre-recorded assets and optimising file sizes. This approach not only enhanced efficiency in audio processing and resource management but also facilitated the seamless integration of sound elements with gameplay mechanics, ultimately benefiting the game. See Appendix 1 for a gameplay video of "Astera".

4.1 MetaSounds driven soundscapes and ambiance

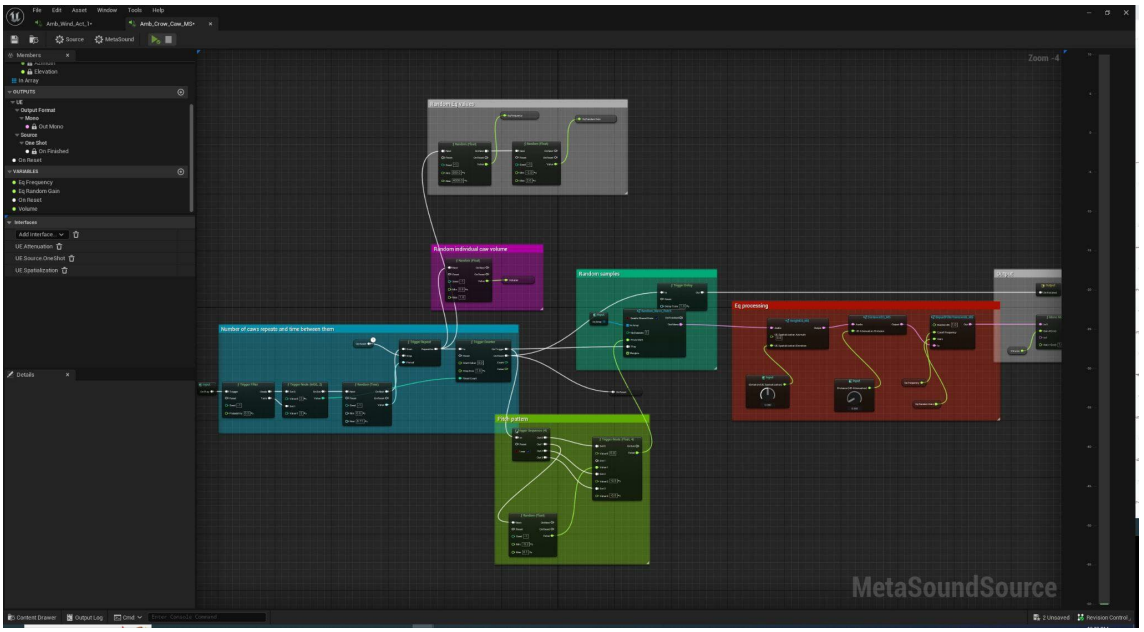
One of the primary applications of MetaSounds and procedural audio in “Astera” was the generation of dynamic environmental soundscapes. By leveraging Meta Sounds, we were able to generate ambient sounds, in general starting from pre-rendered audio assets that followed a specific set of rules to obtain a lot of variations for the fauna, various nature elements, and ambient textures for places like dungeons in the game.

Soundscapes and ambient noise serve as the backdrop to the primary audio elements, including combat sounds and music. While they may not always be prominent, they play a crucial role in establishing the realism and atmosphere of the game's environment and actions. For instance, in Astera a nighttime forest soundscape might feature fantastical creature noises, birds, rustling leaves, etc while ambient noise in a town setting includes NPC chatter, sounds coming from busy buildings such as the local inn or blacksmith, domestic animals, birds

etc. These audio assets were sourced from online libraries and generated using synths.

4.1.1 Procedural audio fauna

In “Astera” a vast amount of fauna sounds are made using the following technique. By deconstructing vocalisations in isolated animal recordings, we gathered building blocks or samples that went through editing and post-processing in the DAW. These building blocks can have multiple variations and can be assembled back together in A MetaSound graph by following a set of rules. By adding possible variations in timing, pitch, rhythm, number of repetitions, sample combinations, sample length, and so on that mimic the vocalisation of the desired animal we get an exponential value in variations with a minimum amount of storage when combining multiple MetaSounds. Take this crow MetaSound as an example (Appendix 2).



Picture 10. Crow one-shot sfx made with MetaSounds (Oprișan 2024).

These fauna MetaSounds can function as either one-shot sound effects or loops. Typically, we want certain one-shots to repeat, like the crow example. We have two options for achieving this. First, we can set a trigger repeat node in the MetaSound, causing our set of rules or logic to repeat, effectively creating a

looping one-shot. Alternatively, we can attach the MetaSound to an in-game component with a script that accomplishes the same task.

Using a script has a significant advantage. When the MetaSound is silent, it still runs and occupies a voice count. Even if there's minimal processing involved, the project has a limit on voices, so looping the MetaSound without optimization becomes inefficient.

These components or looping MetaSounds can then be dragged into the world map and benefit from 3D spatialization. We can either suggest the presence of the fauna, for example, an unseen owl in a tree or attach the components to a game model that is visible to the player, for example, a dog in the town square. Because our game is isometric the camera is further back and the mobile screen is smaller, so some visual elements have a more suggestive role. We have a lot of flexibility with the sounds in this case.

We can also combine multiple MetaSounds to craft a complete procedural ambiance. However, a downside to this approach is the loss of the ability to spatialize each MetaSound. When we group multiple MetaSounds, all the processing, including effects, attenuation, and spatialization, is applied to a single Source object. It is similar to having a rendered stereo or mono master track. Despite this, a benefit is that we only consume one voice in this scenario.

To further integrate these sounds into the game world, I applied attenuation and employed dynamic reverb, delay, and EQ techniques during the mixing process. This approach proved particularly effective during nighttime sequences, where fewer animals are active, and solitary species emerge for hunting activities. As an example, let us listen to the representation of a crow within the game environment (Appendix 2).

4.1.2 Wind Machine

The wind machine utilises multiple filtered and modulated pink noise generators (Appendix 3). The choice of pink noise over white noise in the wind machine design is deliberate and stems from considerations related to human auditory

perception. Pink noise is characterised by a more balanced frequency distribution and is often perceived as gentler and more soothing to the human ear compared to white noise. To create a more realistic scenario random wind gusts are introduced, and the wind intensity adjusts in real-time based on the player's in-world altitude.

Additionally, a real-life recording of wind passing through leaves is integrated into the same MetaSound. The wave player containing the leaves rustle is always playing with the volume level determined by the number of trees in proximity to the player. This logic is handled in a different script that uses collisions between the player and the trees to count the number of nearby trees. The count of trees at any given time is then clamped to a value from 0 to 1 that is being sent to the Metasound. This value is hooked to the volume of the loop. With this technique, there is no need to manually set different pre-recorded wind loops all over the open world map. The transition and variation happen seamlessly. We also limit the need for pre-recorded material. Moreover, with this integration, we leave room for easy integration with further systemic developments such as weather and seasons. See Appendix 4 for a demonstration of the wind sound in the MetaSound editor.

4.1.3 Ambiental magical textures

Using procedural audio works well with abstract sounds. For example, in “Astera” we created a choral pad assembled from different vocal samples processed to create a foreboding atmosphere (Appendix 5).

We are creating a fantasy game with a sci-fi twist so there are a lot of opportunities to imbue the world with ethereal textures that feel endless while adding a layer of wonder and mystique to the environment.

Designing these abstract ambient sounds doesn’t come with a standard blueprint and there is a lot of creative potential in the design and integration part as well. Not being tied to an actual physically recognizable sound means that we are limited just by our imagination. It is possible to create ambiances from noise and wave generators mixed with traditional samples and create different

rules for every sound element. Like so, navigating the game world and its locations, the player can hold onto multiple recognisable ambiances with a strong sonic identity that feels boundless in its variety (Appendix 6).

These magical textures are integral to the immersive experience we're striving to deliver. Whether it's the hum of an ancient relic, the resonance of a celestial portal, or the foreboding atmosphere of a dungeon, we want to be sure players can put on their headphones and experience a mobile game with the same sound quality you would typically experience on PC or console games.

5 MOBILE RESOURCE UTILISATION

When evaluating resource utilisation for audio in mobile games, the most important aspects to consider are performance and storage efficiency.

Performance involves assessing CPU usage to ensure smooth gameplay with no stuttering, especially in demanding scenarios. Memory is also an important part as all the game objects and their properties necessary for good sound design take up resources. It includes details like position, orientation, and other custom parameters. Higher object memory usage can lead to out-of-memory (OOM) issues if not managed properly.

Storage efficiency relates to managing the game's footprint on the device, including the use of compression and efficient asset management to reduce the space taken up by game files.

While memory management is important, “Astera” has a limited voice count of 16 voices which restrictss the number of simultaneous audio streams, reducing the potential for out-of-memory (OOM) issues from excessive audio processing. A useful experiment for future work would be to run the game on a test map with and without audio code, focusing specifically on scenarios with multiple MetaSounds running concurrently while disabling other engine features like effects and spatialization. This isolated test could provide a clearer understanding of the memory consumption directly attributable to MetaSounds, allowing developers to assess their impact on resource utilisation without interference from other engine elements. With that in mind the final discussion of mobile resource utilisation will focus on CPU usage and storage.

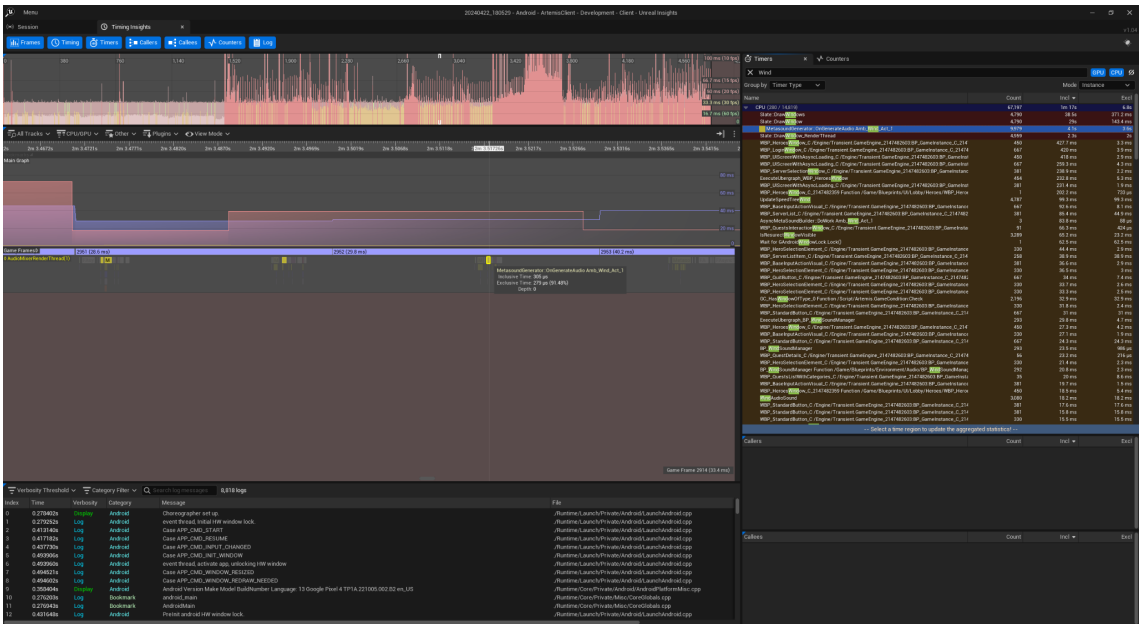
5.1 MetaSounds mobile CPU usage in “Astera”

To get the necessary data regarding the CPU usage of the example MetaSounds discussed in this paper, Unreal Engine provides a useful tool called Unreal Insights. At a high level, Unreal Insights is a stand-alone profiling system that integrates with Unreal Engine to collect, analyse, and visualise data emitted by the engine. In addition to providing robust coverage of the Engine's

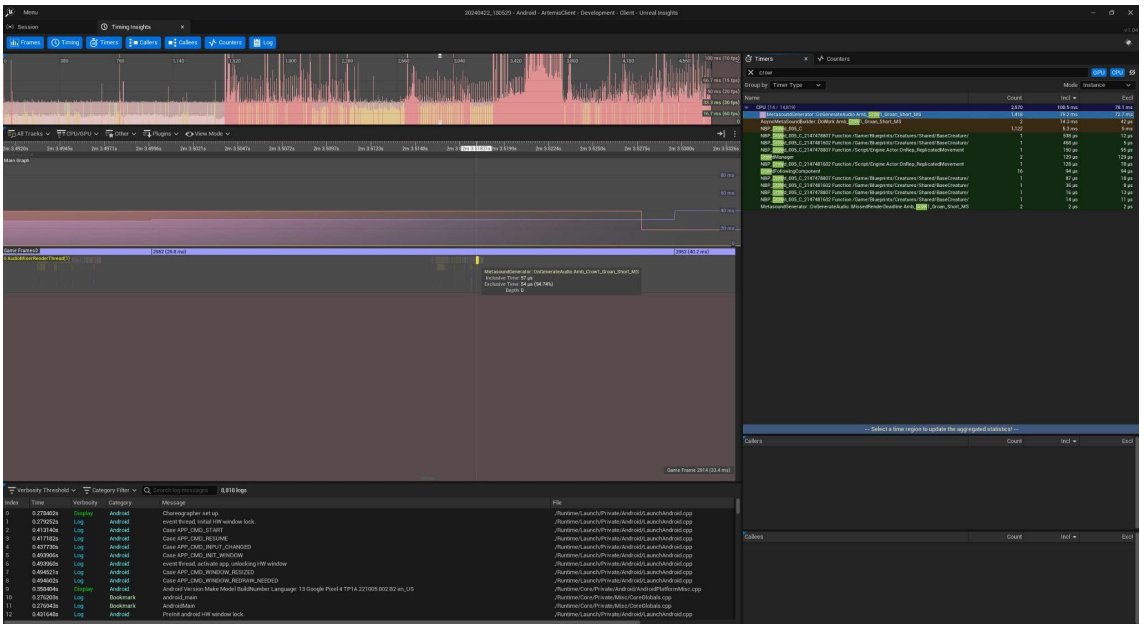
existing systems, Unreal Insights makes it easy to add our own profiling data (Unreal 2023).

We can look at how much time, in a given frame, a specific MetaSound needs for rendering. Audio rendering is the process by which sound sources are decoded and mixed together, fed to an audio hardware endpoint (called a digital-to-analog converter, or DAC), and ultimately played on one or more speakers (Unreal 2023). In Unreal Insights we have a dedicated AudioMixerRenderThread that we can look at for this information. To run this game at 30 frames per second we have a budget of 33 ms per frame. In this frame we have to take into account that audio should be just a small part, as a lot of the CPU power should be allocated to AI, physics, collisions, game logic, rendering and other tasks.

We can take a look at a test running Unreal Insights on a short gameplay session on a Google Pixel 5. This is a phone released in 2020 and serves as Dream Primer's benchmark phone for hitting 30 frames per second on the Android operating system.

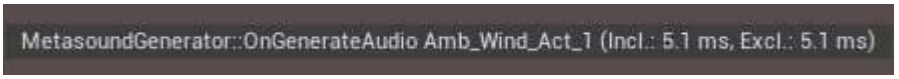


Picture 11. Unreal Insights WindManager MetaSound at runtime on a Google Pixel 5 (Oprișan 2024).



Picture 12. Unreal Insights Crow MetaSound at runtime on a Google Pixel 5 (Oprișan 2024).

Overall from our gameplay tests we can see that even the complex MS will sit below 1ms. Being a game in alpha state the big spikes that are present in the pictures are reflective of other problems that will be addressed as the game gets closer to its release date.

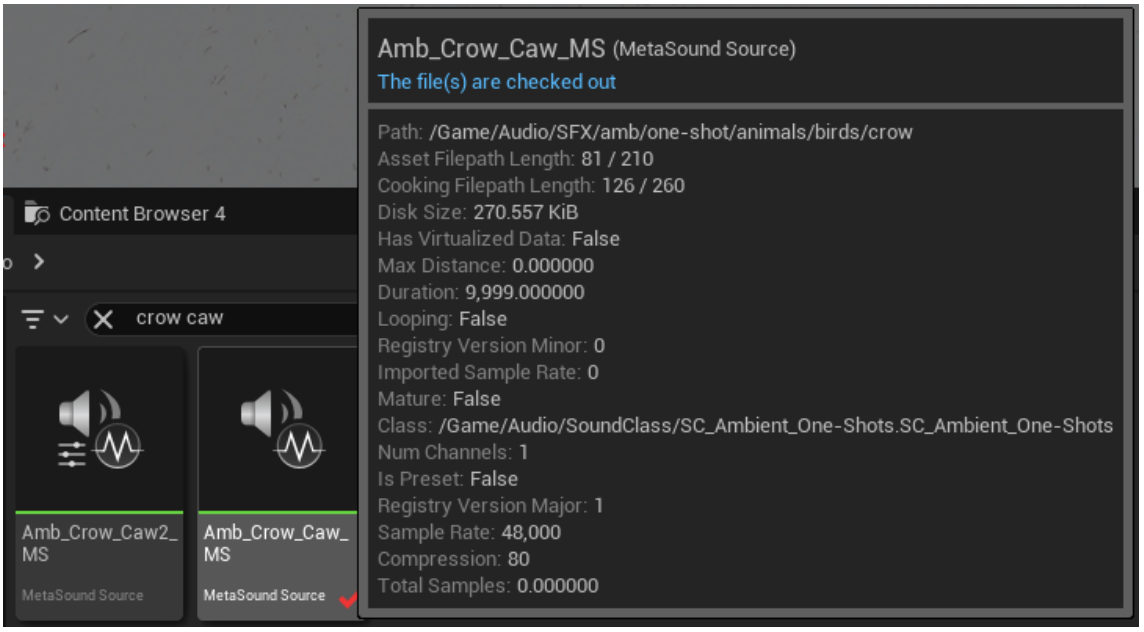


Picture 13. Unreal Insights WindManager MetaSound spike at runtime on a Google Pixel 5 (Oprișan 2024).

We encountered a big 5.1 ms spike during a monitoring session on the wind manager. As of the writing of this thesis we are not sure what was the cause and further tests and optimisations can be made with the MetSounds graphs at the cost of latency. This solution of course is not suitable for all genres like rhythm games or fast paced action or sports games.

5.2 Storage optimisation

On the storage optimisation side using MetSounds proved a great way of reducing the footprint of lengthy files. It's important to note that all the mentioned files are uncompressed WAV files at 16 bit 44.1kHz. During the packaging and exporting of the game all audio is compressed heavily, reducing the files footprint even further with the bink audio codec which has lossless 10:1 compression rate.



Picture 14. Crow MetaSound disk size (Oprișan 2024).

Constructing the crow ambient actor with edited, smaller audio assets allowed us to minimise disk space usage, bringing it down to just 270 kilobytes. This approach provides a high degree of reusability, as the asset's longer, seamless loop lacks distinct or recognizable segments, unlike shorter loops that can quickly become repetitive.

Using Unreal's size map we can take a look at all the one-shot audio assets that we will probably use in our final build of the game. These ambient one-shots will cover multiple regions and biomes in the game world.

Picture 15. All ambient assets size map in the alpha build of “Astera” (Oprișan 2024).

It is important to take into consideration that a loop can take from 6Mb to 30Mb for a 35 seconds to 175 seconds WAV file at 16 bit 44.1kHz. We can see how quickly we can consume our disk size budget on a restrictive platform by adding up multiple stereo loops.

Instead of relying on static loops, MetaSounds enabled us to have more dynamic, modular soundscapes, where audio components can change based on gameplay events with the benefit of reducing the files size.

6 DISCUSSION

The integration of MetaSounds and procedural audio techniques in "Astera" elevated the depth and dynamism of its environmental soundscapes, while keeping the resource utilisations manageable. By generating variations of ambient sounds based on predefined rules, the game's audio design achieved a level of complexity that contributed to the immersive nature of the game world. From the slight variation in the wind sound to the ambiance fauna, each sound element served to create a closer-to-life auditory experience for the environment.

There is a justified motivation to use procedural audio for these varied dynamic ambiances and the concepts discussed in this thesis can be further applied to multiple aspects of sound design in various game genres. Additionally, there's potential to incorporate fully synthesised sounds at runtime, particularly in games with retro, sci-fi, minimalist, and other artistic styles.

Our hybrid technique of using small pre-recorded samples reduced our need to store large amounts of audio data in the game build. Going with synthesis or other generative techniques can result in a minuscule storage footprint.

Another advantage lies in the scalability of the procedural audio systems we construct using MetaSounds and scripts. This means that the audio components integrate smoothly, allowing us to effortlessly introduce additional complexity or repurpose elements in the future. As a result, the audio could become more immersive and responsive to the player’s actions or in-game events.

As game development processes evolve alongside advancements in hardware, there's a noticeable surge in the attention given to sound design, even within the realm of mobile platforms. This shift implies that, for developers not exclusively focused on hyper casual games, audio quality has become another crucial aspect for maintaining competitiveness in the market.

Thanks to the accessibility of powerful tools like Unreal Engine, Wwise, Pure Data, and abundant learning resources, game developers can now tap into the potential of procedural audio with relative ease. This accessibility empowers developers and sound designers to leverage procedural audio tools for creating procedural soundscapes.

While the tools and resources are easily available, mastering these tools requires time and expertise. Quality control becomes more intricate, as procedural audio inherently produces a wide range of variations, necessitating thorough testing to ensure consistency both in quality and performance. Sound designers may need to invest in learning new techniques and workflows to effectively use procedural audio in their mobile game projects. Additionally, collaboration between sound designers and game developers is essential to ensure a better integration of procedural audio into the game's design and implementation process.

Developing for mobile platforms presents its own challenges, as storage and hardware capabilities can vary significantly among consumers. In this context, optimising audio storage becomes a huge necessity, and procedural audio emerges as a viable solution for sound design and music generation. However, it's important to note that, due to CPU processing constraints, a conservative approach to procedural sound design in mobile games is advisable in the near future.

One of the key advantages of procedural audio lies in its systematic approach, which can allow expansive worlds to benefit from credible and immersive sound. By generating audio content algorithmically, procedural audio offers the potential for a more reactive game world while accommodating for mobile hardware and storage resources.

REFERENCES

Andersen, A. 2020. The Ultimate Guide to Game Audio Pipeline. [Online article]. Released on 05.08.2020. Updated on N.d. Read on 25.11.2023, from https://www.asoundeffect.com/game-audio-pipeline/

Antić, M. Audio Optimization Practices in Scars Above. 2023. [Online article]. Released on 24.08.2023. Updated on N.d. Read on 25.11.2023. https://blog.audiokinetic.com/en/audio-optimization-practices-in-scars-above/

Collins, K. 2008. An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design. Machetutes: the MIT press

Collins, K. 2009. An Introduction to Procedural Music in Video Games. Contemporary Music Review 28.

Collins, K. 2014. The Oxford Handbook of Mobile Music Studies, Volume 2. Oxford: Oxford University Press.

IGN. Realtime Audio Adventures Announces Free-to-Play Soul Trapper: Preview. 2009. [Online article]. Released on 12.05.2009. Updated on 11.05.2012. Read on 17.12.2023. https://www.ign.com/articles/2009/05/12/realtime-audio-adventures-announces-free-to-play-soul-trapper-preview

It's Nice That. 2021. The history of Snake: How the Nokia game defined a new era for the mobile industry. [Online article]. Released on 23.02.2021. Updated on N.d. Read on 24.11.2023. https://www.itsnicethat.com/features/taneli-armanto-the-history-of-snake-design-legacies-230221

Johnson, W. Introducing DSPGraph, the new audio rendering/mixing engine. Unite Copenhagen on 23-26.09.2019. Copenhagen.

Kultima, A. 2009. MindTrek '09: Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era, 58–65.

Leadbetter, R. 2023. Triple-A games are coming to iPhone 15 Pro and the implications are mouthwatering. [Online article]. Released on 18.10.2023. Read on 24.11.2023. https://www.eurogamer.net/digitalfoundry-2023-triple-a-games-are-coming-to-iphone-15-pro-and-the-implications-are-mouthwatering

Laven, A. Sounds as Intended: Quality Assurance in Game Audio. The Game Developers Conference (GDC) 37th edition on 20-24.03.2015. San Francisco.

Matteo Malinverno, M. What is wavetable synthesis?. 2021. [Online article] Released on 06.07.2021. Updated on N.d. Read on 17.03.2024. https://splice.com/blog/what-is-wavetable-synthesis/

Mattingly, R. J., Shumate, J. & Whitmore, G. Peggle Blast: Big Concepts, Small Project. The Game Developers Conference (GDC) 29th edition on 02-06.03.2015. San Francisco.

McLeran, A, Kent, J. Procedural Music in Spore. The Game Developers Conference (GDC) 30th edition on 18-22.02.2008. San Francisco.

Mraz, R. 2021. How procedural audio brings sounds to life in video games. [Online article]. Released on 13.05.2021. Updated on N.d. Read on 22.10.2023. https://splice.com/blog/procedural-audio-video-games/

Phone Arena. 2015. This was the world's first cell phone with a game loaded on it. [Online article]. Released on 12.01.2015. Updated on N.d. Read on 24.11.2023. https://www.phonearena.com/news/This-was-the-worlds-first-cell-phone-with-a-game-loaded-on-it_id62920

Plut, C., Pasquier, P. 2020. Generative music in video games: State of the art, challenges, and prospects. Entertainment Computing 33.

Pure Data Community Site. https://puredata.info/. [Online documentation] Released on N.d. Updated on 25.02.2024. Read on 25.02.2024.

Sinclair, J.-L. 2020. Procedural Audio: Beyond Samples, in Principles of Game Audio and Sound Design. Focal Press, New York, NY.

Smith, J.O. 2010. [Online book]. Physical Audio Signal Processing. Read on 17.03.2024. http://ccrma.stanford.edu/~jos/pasp/

Spatial. 2023. Reimagining Sound Design Pt.2. [Online article]. Released on 30.08.2023. Updated on N.d. Read on 22.10.2023.

Unreal Engine. 2023. MetaSounds: The Next Generation Sound Sources in Unreal Engine. [Online documentation]. Read on 25.11.2023. https://docs.unrealengine.com/5.3/en-US/metasounds-the-next-generation-sound-sources-in-unreal-engine/

Vreeland, R. Music composer/sound designer. Interview on 18.02.2016. Richard Gould.

Vreeland, R. Reigns: A Thousand Years of Deterministic Babble. Enjmin game conference in 2020. Angouleme.

Vreeland, R. Serialism & Sonification in Mini Metro. The Game Developers Conference (GDC) 32nd edition on 19-23.03.2018. San Francisco.

Weir, P. The Sound of No Man's Sky. The Game Developers Conference (GDC) 31st edition on 27.02-03.03.2017. San Francisco.

Zolin, S. Audio Formats Comparison. [Online article]. Released on 29.08.2016. Updated on 27.03.2017. Read on 22.04.2024. https://stsaz.github.io/fmedia/audio-formats/

Games

Genshin Impact, 2020, miHoYo, Shanghai Miha Touring Film Technology Co., Ltd.

Mini Metro, 2015, Dinosaur Polo Club.

Monument Valley, 2017, Ustwo.

Nerai, 2016, Reigns.

No Man's Sky, 2016, Hello Games.

Otocky, 1987, SEDIC.

Peggle Blast, 2014, PopCap Games.

Reigns, 2016, Nerai.

Snake, 1997, Nokia.

Soul Trapper, 2008, Realtime Associates.

Spore, 2008, Maxis.

APPENDICES

Appendix 1. Gameplay footage of Astera in alpha version.

https://www.youtube.com/watch?v=gEPNfzD1INY

Appendix 2. Hearing a crow MetaSound one-shot in a test level.

https://youtu.be/00JhuBNHXZs

Appendix 3. A wind machine built from scratch with MetaSounds.

https://youtu.be/BTMqvTAQkQE

Appendix 4. The wind manager MetaSound in Astera.

https://youtu.be/x-LGESCnpYs

Appendix 5. Demonstration of an ambiance pad inside MetaSounds.

https://youtu.be/00_PnwvnPJU

Appendix 6. Sand tomb ambiance inside MetaSounds.

https://youtu.be/roNe-nfk0Vw